Name : **Peter Sunny Shanthveer Markappa**

Subject : **Machine Vision**

Student No. : **R00208303**

Submission Date : **18th March 2022**

University : **Munster Technology University, Cork, Ireland**

Index

1. Introduction of Image
2. Task 1
    - Task 1-A
    - Task 1-B
3. Task 2
4. Task 3
5. Task 4

Image is more worth than 10000 words

What is an Image?

Image is 'Projection of 3-Dimension scene into the 2-Dimension scene'.

Image is made up of small segments called 'picture cells' aka pixels. The process of capturing the image is called 'Acquisition' wherein the instrument (camera) is called an image sensor.

The main motivation behind processing the image is:

a) Improvement of pictorial information for human interpretation
b) Processing of image data for storage, transmission, and representation for autonomous machine perception

Task - 1

Task 1 – A:

The task to load the image and convert it into float32 type from uint8, the code that was use is

```
input_image = input_image.astype(np.float32)
```

when we upload the image the data of the image will be in the form of a matrix where each data matrix will represent the intensity, when the elements of the matrix are in uint8 or uint16 the value are will range from [0,255] for uint8 and [0, 65535] for uint16. If it is converted into float then values will be single-precision floating-point numbers.

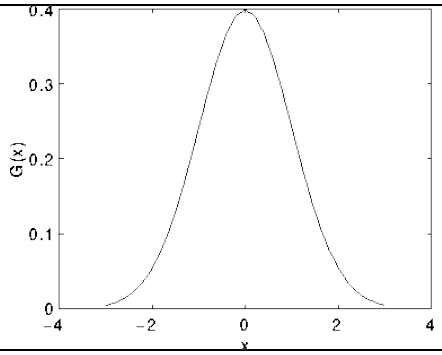
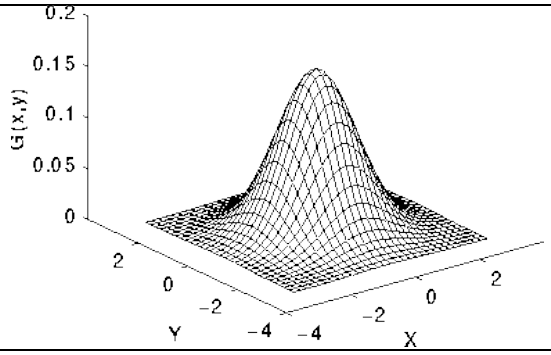
Input image (dtype = float32)	GreyImage (dtype = float32)

Task 1 – B: Create Gaussian smoothing kernel with sigma value from 0 to 11

Gaussian Smoothing: this operator is a 2-Dimension convolution operator which can be used to 'blur' the images and remove detail and noise which is similar to the mean filter but here it uses the kernel. This kernel has properties

Working of the Gaussian kernel: **[6] refer 6th reference**

The formula for both 1-Dimension and 2-dimension

For 1-Dimension	For 2-Dimension
$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$ Here there is only one co-ordinate so the expression numerator has only one variable 'x', where σ is the standard deviation	$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$ It is two dimensions so the expression numerator has x and y coordinates where σ is the standard deviation
	

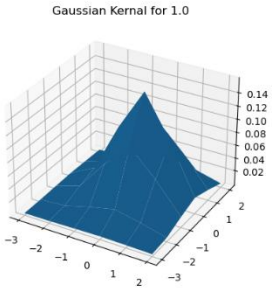
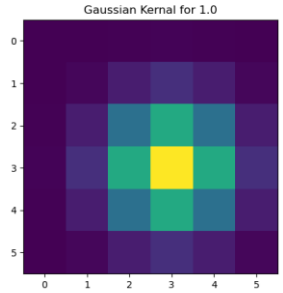
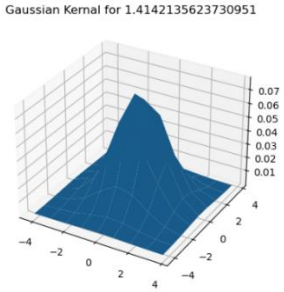
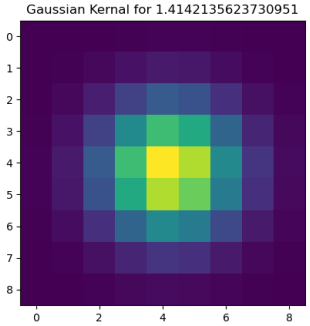
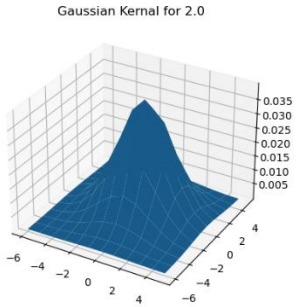
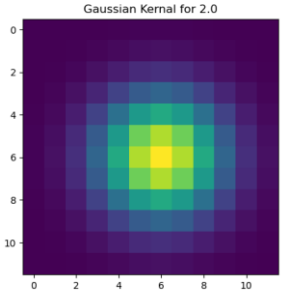
"Smoothing is the process of averaging data points with their neighbors in a series, such as a time series or image. This has the effect of blurring the sharp edges in the smoothed data (usually). Smoothing is sometimes referred to as filtering because it suppresses high-frequency signals while enhancing low-frequency signals. Smoothing can be accomplished in a variety of ways, but this article will focus on smoothing with a Gaussian kernel. We hope that the explanation below adequately explains this phrase.

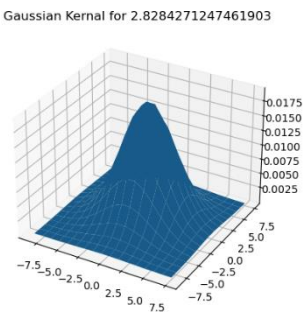
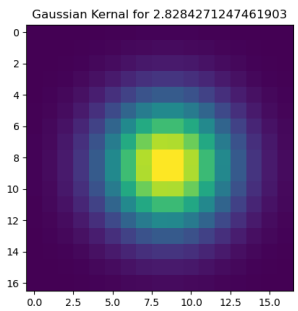
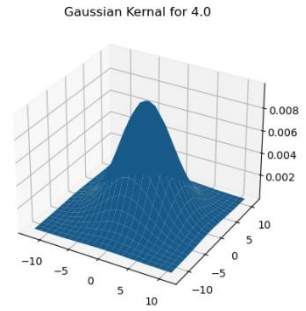
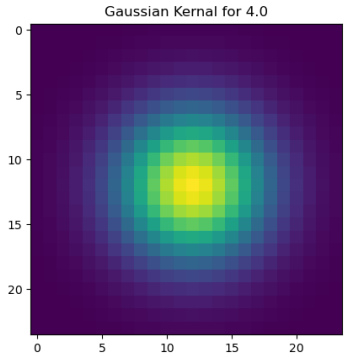
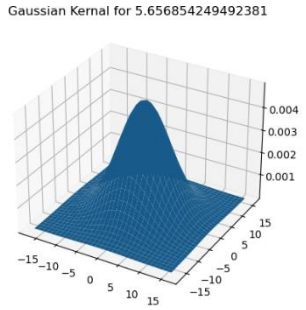
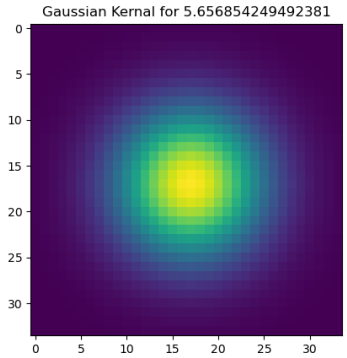
The 'kernel' for smoothing defines the shape of the function that is used to take the average of the neighboring points. A Gaussian kernel is a kernel with the shape of a Gaussian (normal distribution) curve.

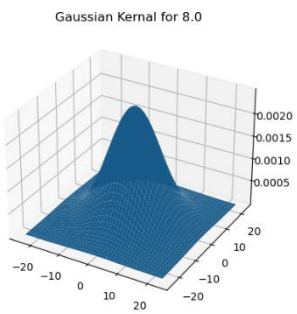
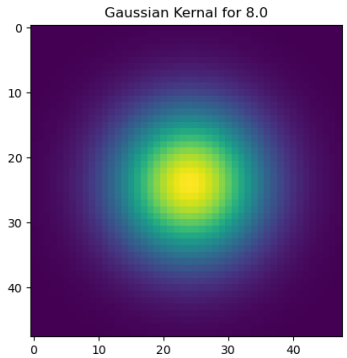
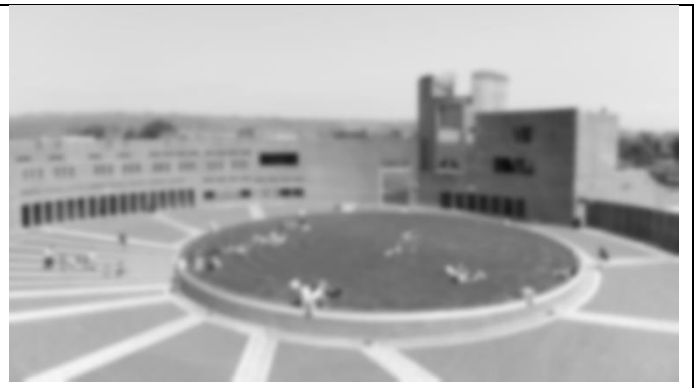
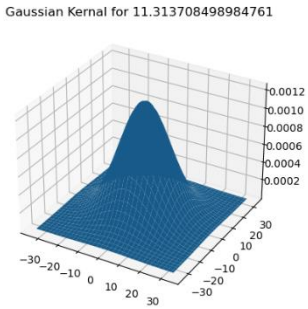
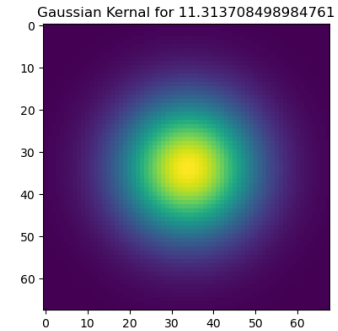
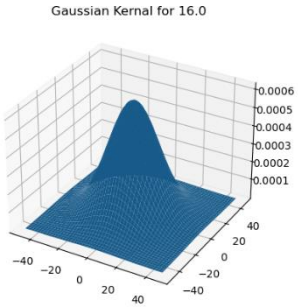
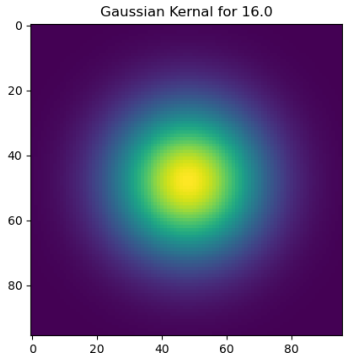
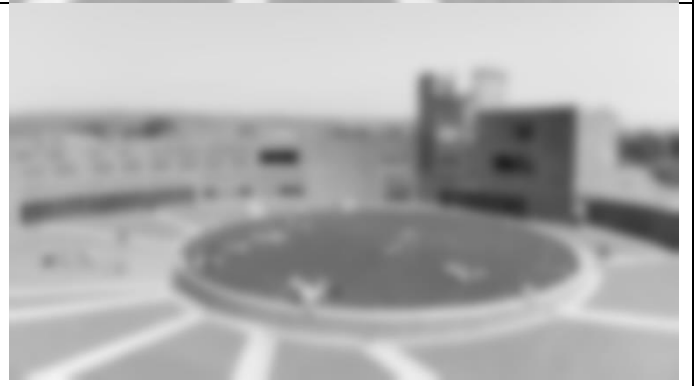
Smoothing with the kernel

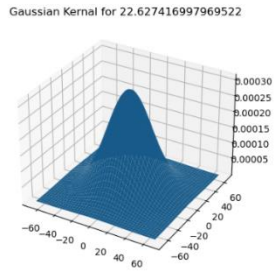
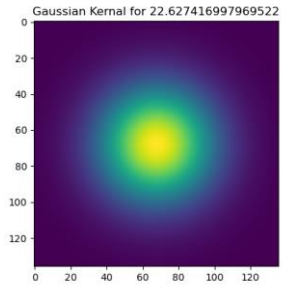
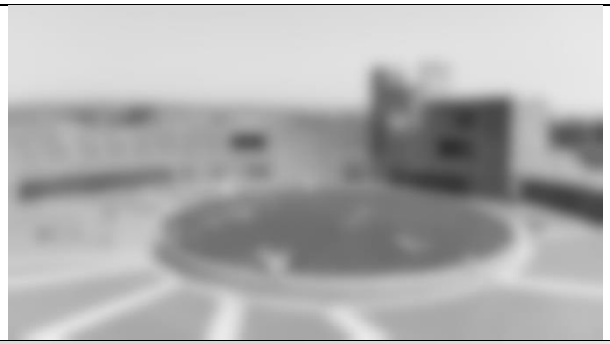
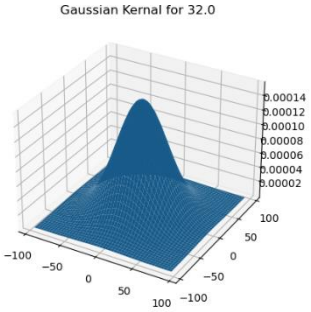
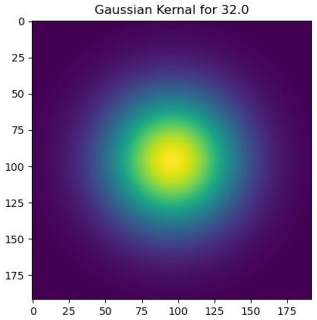
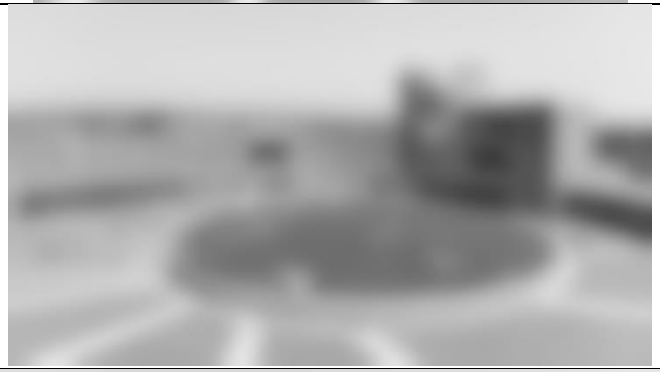
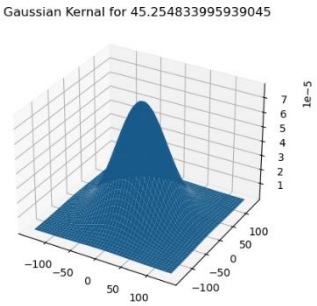
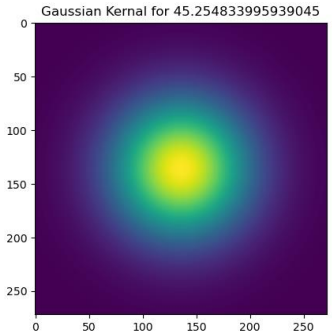
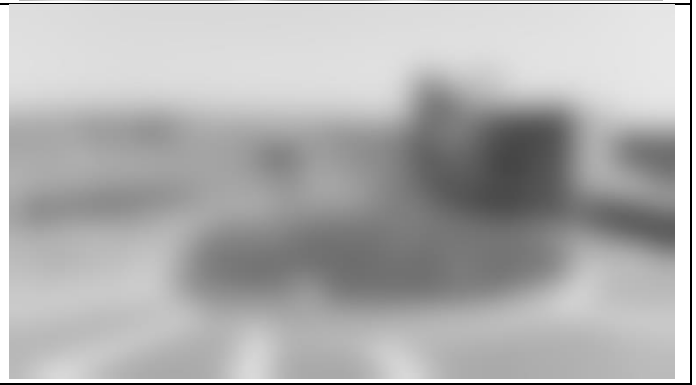
Smoothing is a very simple process in its most basic form. We work our way through the data, one point at a time. We generate a new value for each data point that is some function of the original value at that point and the surrounding data points. The function used in Gaussian smoothing is our Gaussian curve." [7]

Gaussian Kernels for different σ values are listed below

Sigma Value			Images
σ = 1.0			
σ = 1.4142			
σ = 2.0			

σ = 2.82842			
σ = 4.0			
σ = 5.6568			

σ = 8.0			
σ = 11.3137			
σ = 16.0			

σ = 22.62			
σ = 32.0			
σ = 45.2548			

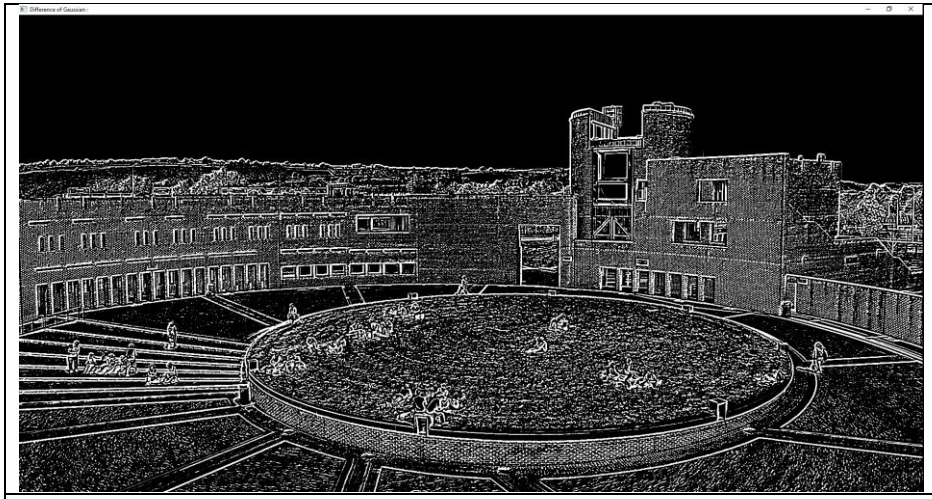
From the above table, we can see that as the sigma value is increasing the kernel of the image is getting blurred, because the kernel is getting blurred the image is also getting blurred

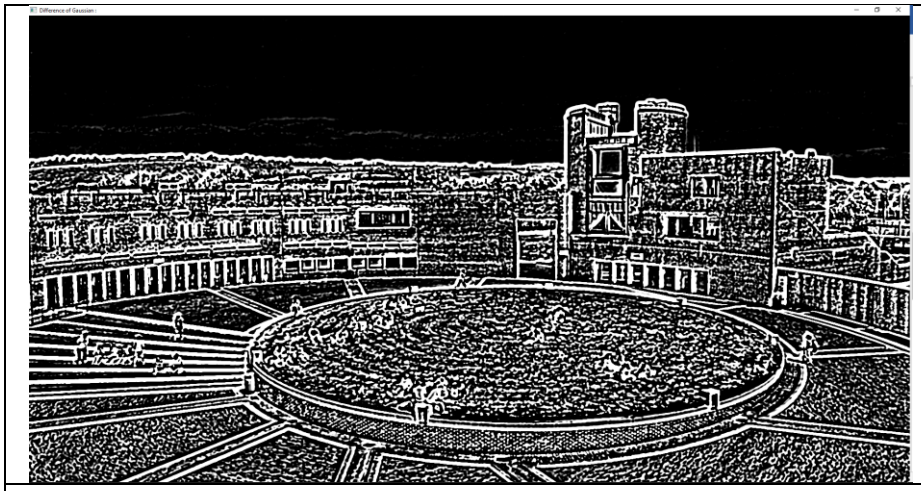
Task - 2

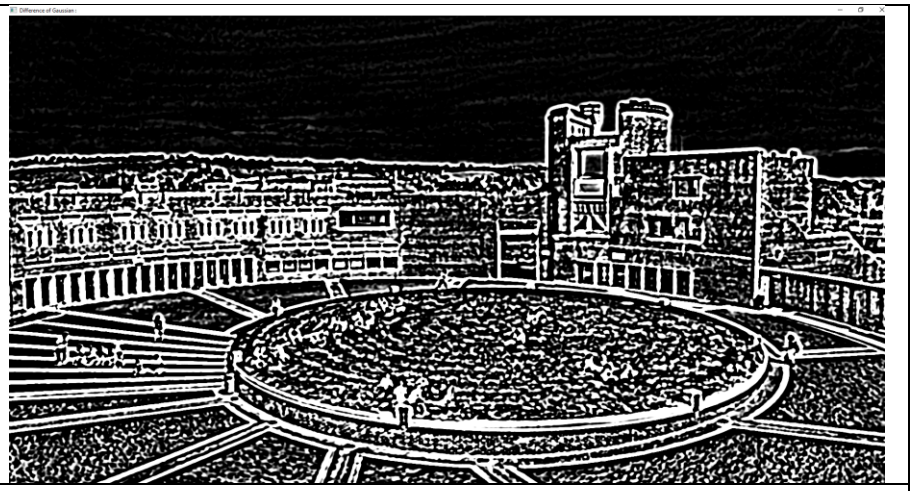
Task 2 – A: Difference of Gaussian Images

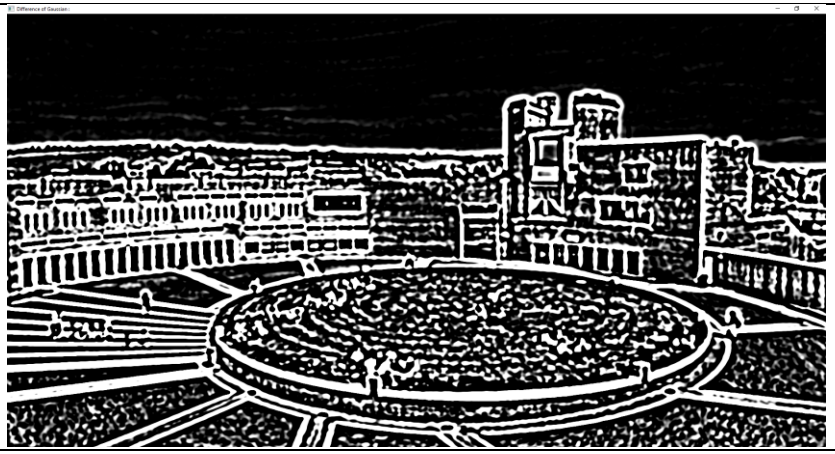
“The subtraction of one blurred version of an original grayscale image from another, less blurred version of the original is the difference of the Gaussians grayscale image enhancement algorithm. The blurred images are created by convolution of the original grayscale image with Gaussian kernels with varying standard deviations. Only high-frequency spatial information is suppressed when employing a Gaussian kernel to blur an image. The spatial information between the ranges of frequencies preserved in the two blurred images is preserved by subtracting one image from the other. As a result, the difference of Gaussians is akin to a band-pass filter that removes all but a few spatial frequencies from the original grayscale image. The difference of the gaussian algorithm is thought to work in the same way that neural processing in the retina pulls features from images ready for transmission to the brain.

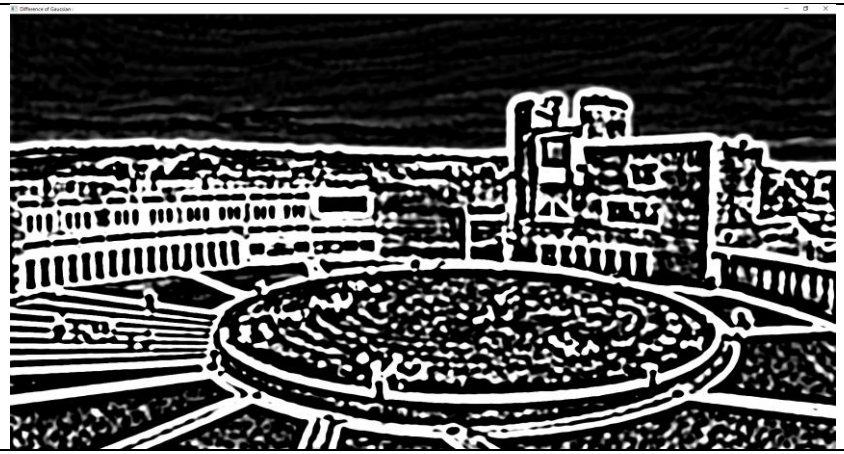
After subtracting 12 images 11 Difference of gaussian images are formed.” [8] [9]

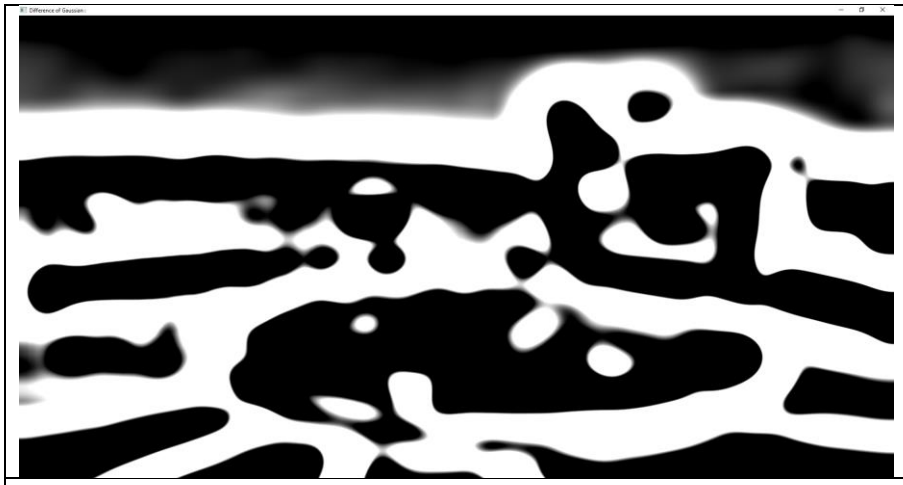
	


Difference of Gaussian :


Difference of Gaussian :


Difference of Gaussian :


Difference of Gaussian :

The difference of gaussians can be used as an image enhancement algorithm to improve the visibility of edges and other information in a digital image. Many alternative edge sharpening filters work by improving high-frequency detail, however because random noise has a high spatial frequency, many of these sharpening filters also tend to amplify noise, which is an unwanted artifact. The difference of gaussians algorithm removes high frequency detail, which typically contains random noise, making it one of the best approaches for dealing with photos containing a lot of noise. The intrinsic reduction in overall picture contrast caused by the implementation of the difference of Gaussians algorithm is a severe disadvantage.

Under the relevant blurred photos, the sizes of the Gaussian kernels are used to smooth the specimen image. The difference of Gaussians algorithm is commonly used for image enhancement when the size ratio of the kernel (2) to the kernel (1) is 4:1 or 5:1. When the ratio of (2) to (1) is about equal to 1.6, the procedure can also be used to approximate the Laplacian of Gaussian. The Gaussian Laplacian can be used to detect edges at different image sizes or degrees of image focus. The exact values of σ(1), σ(2),..., and so on used to approximate the Gaussian Laplacian will determine the scale of the difference image, which may appear blurry as a result.

Task 2 – B (1): Thresholding of Gaussian Images

"Thresholding is an image segmentation technique in which we change the pixels of an image to make it easier to analyze. We use a binary image, which is simply black and white, for thresholding. Most of the time, we use thresholding to select areas of an image that are of interest while ignoring the parts that aren't."
"Assume we want to choose a specific shape from the image. In other words, we want to keep the pixels associated with the shapes "on" while turning the rest of the pixels "off" by setting their colors channel values to zeros. We'd like to use the threshold t so that pixels with grayscale values on one side of t are turned "on," while pixels with grayscale values on the other side are turned "off." [2]

How can we do that?
We can turn all the values of pixels into binary 0 and 1, in that case, the threshold value ranges from 0.0 to 1.0 or if we have kept pixel value from 0 to 255 then we take any range between this value.

How to determine the Threshold Value?
One of the ways we could set the threshold value is to look at the histogram of the image and then identify what Grayscale ranges correspond to our required shape in the image or background

Task 2 – B (1): Key Points

Selecting the key-point involves 2 steps

1. First to select the key-point the pixel value has to be more than the threshold value then we will consider step 2 else we will move for the next point. Let us take a simple grid for visualization.
   Considering the Threshold value is 10

5	0	1
15	17	20
25	20	22

So, from above grid we can see that 3 of the pixels values 5, 0, and 1 cannot be considered as they are less then threshold value

2. So now once we get a pixel that is more than the threshold then we will compare with neighbouring pixels, in our case as per assignment we are not just considering neighbours of same image but also previous image and next image as we are taking use of Difference of gaussian image that is there are 11 images
   How it works

5	0	1
15	17	20
25	20	22

15	20	25
18	17	45
25	20	40

60	78	70
40	85	72
35	80	60

So here if the red marked pixel is selected which is more than the threshold then this pixel has to be more than not only 8 other pixels of its own image but also 18 pixels (9+9) pixels of the other two images then only we will consider this pixel as key-point or point of interest. All this process is done on the grayscale image and then plotted onto the original.

Below is the image that shows key-points

Task 3 – A: Derivatives of all Scale-Space Images from Task 1

In two dimensions especially gaussian is not just about the derivative but also it says about the direction. In gradients, we have deriviatives in x direction and y directions
So basically while taking the derivative of the filter then we must also mention in what direction are we taking that derivative either x or y direction

$$(I * g) * h_x = I * (g * h_x)$$

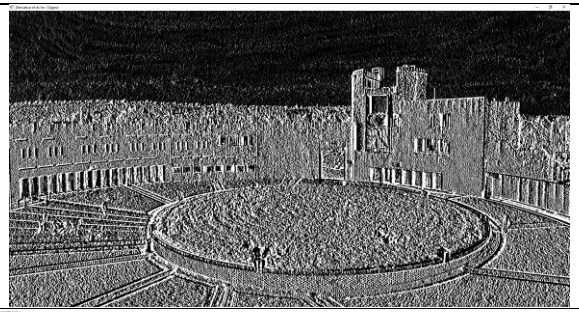
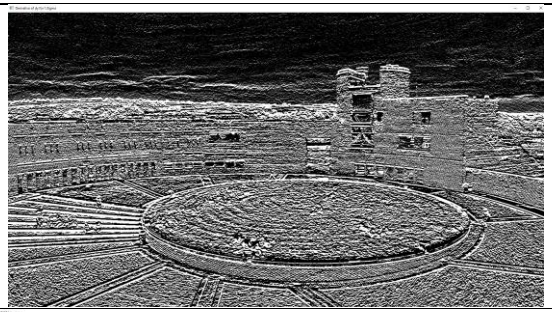
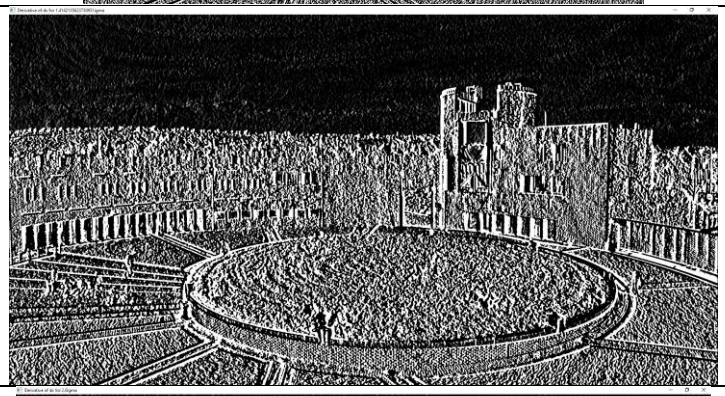
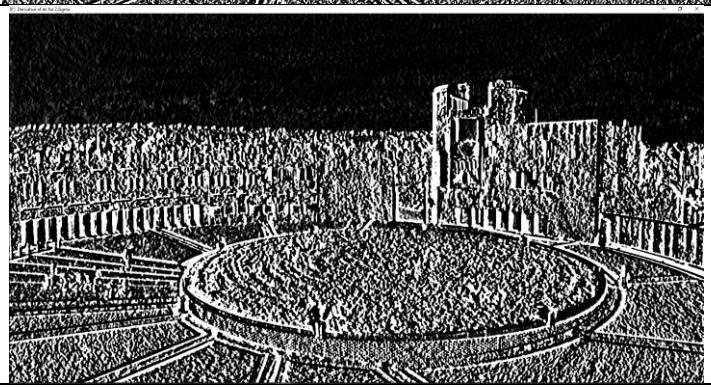
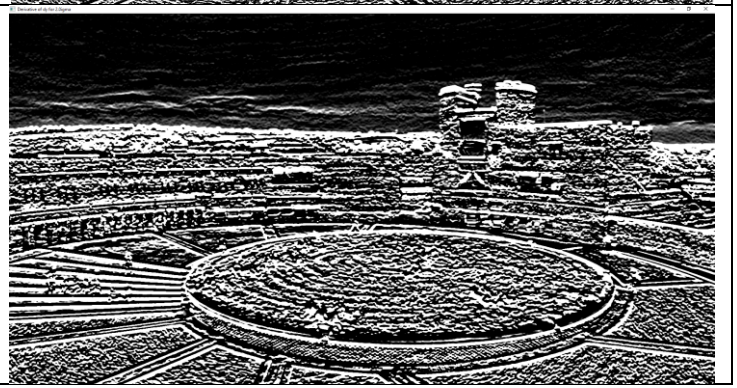
h_x = suppose if we want to take the derivative in the x direction then h_x is ment to imply is that filter take derivative in the x direction may be using any operator.
G = gaussian
I =

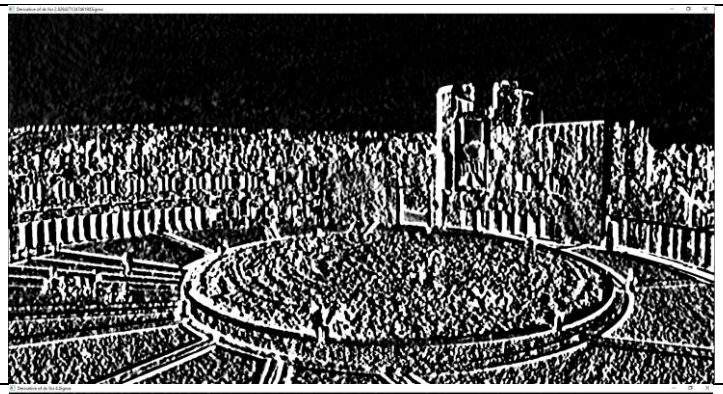
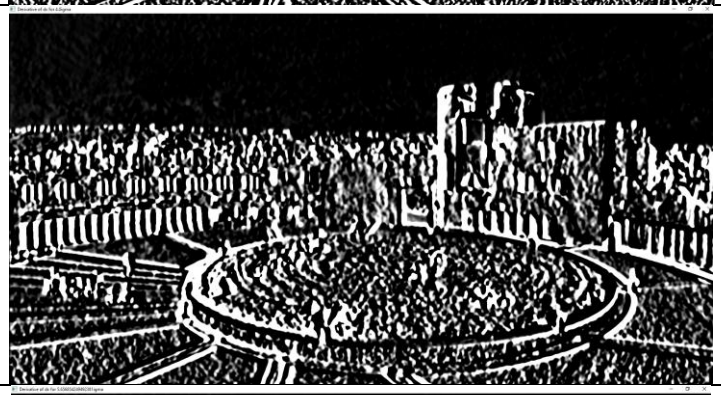
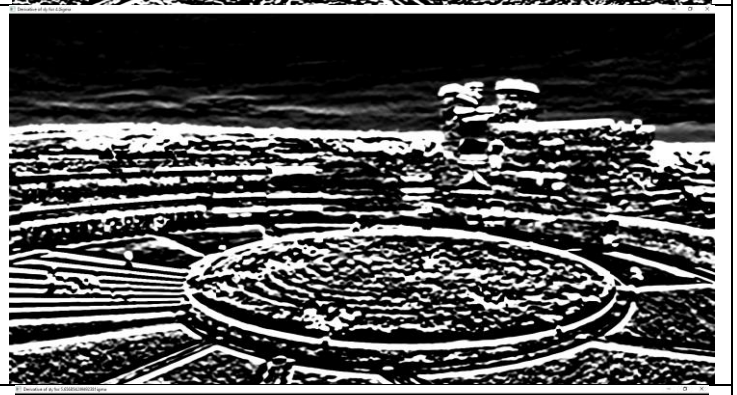
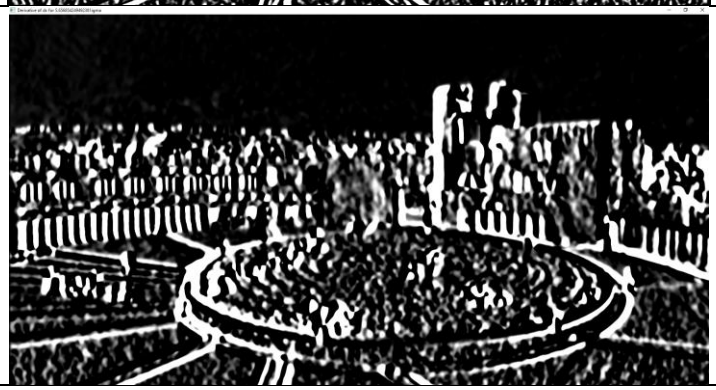
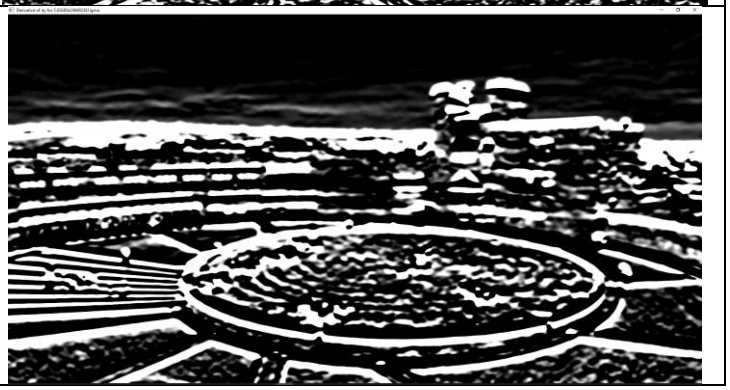
Let me take matrix and explain with derivative of x and y

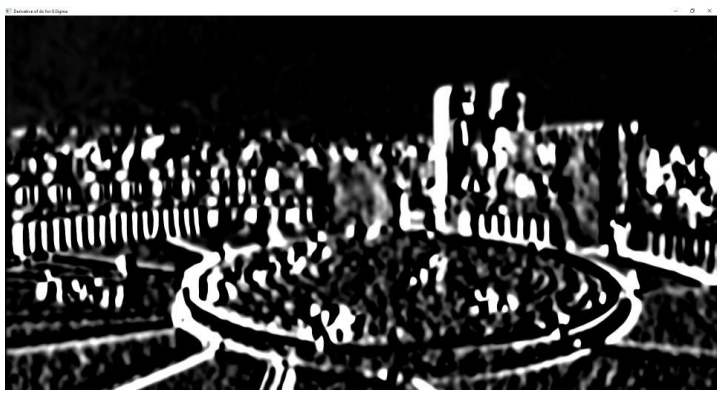
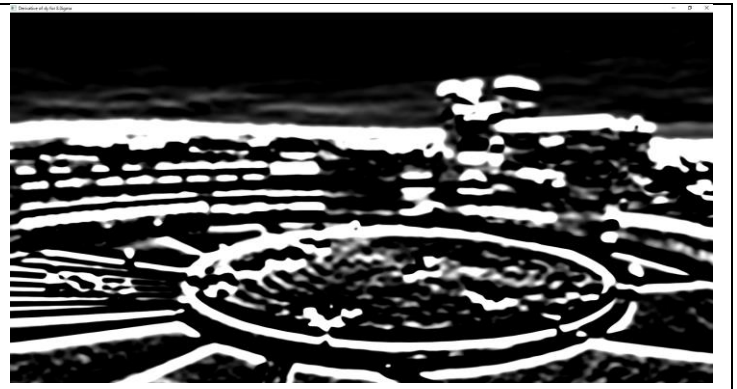
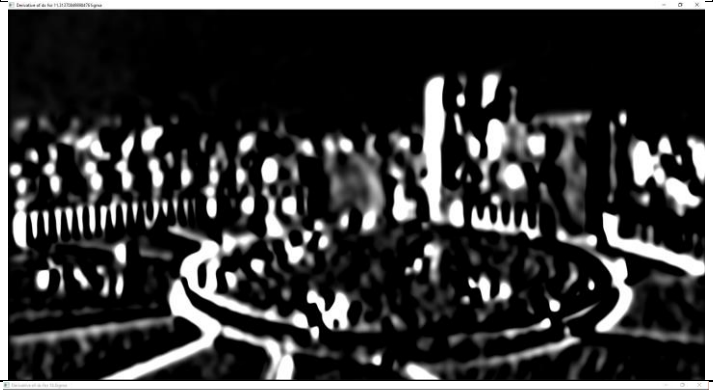
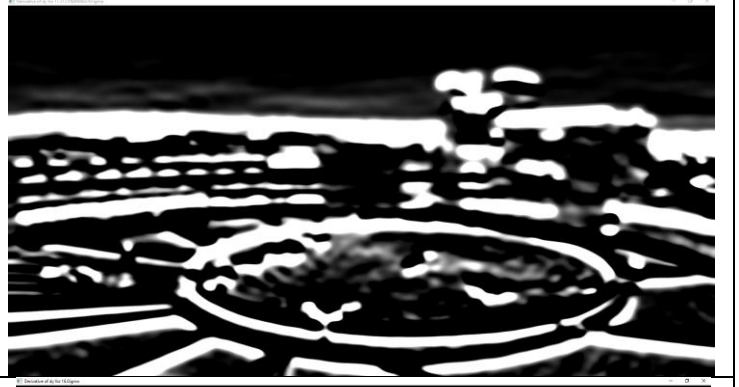
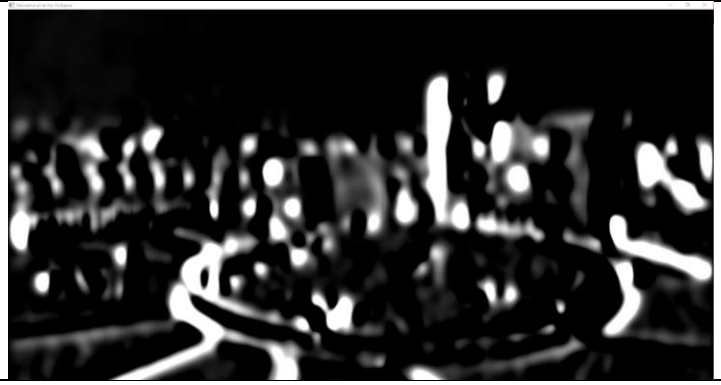
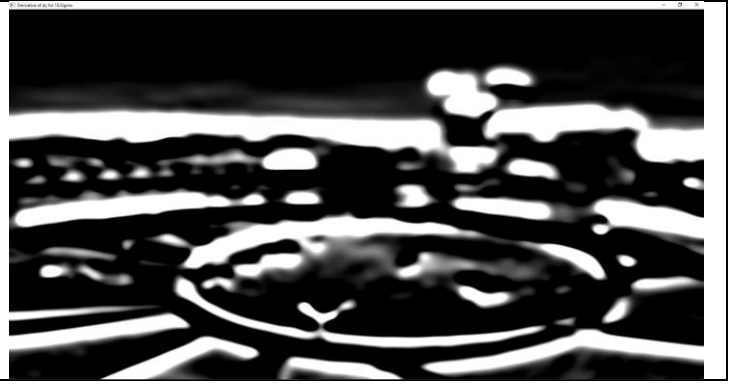
$$(I * g) * h_x = I * (g * h_x)$$

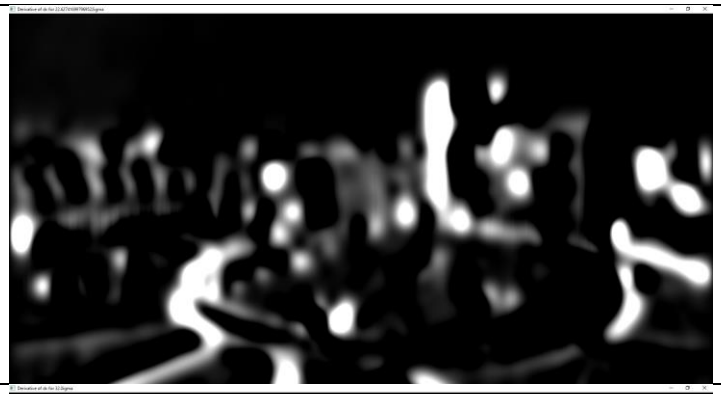
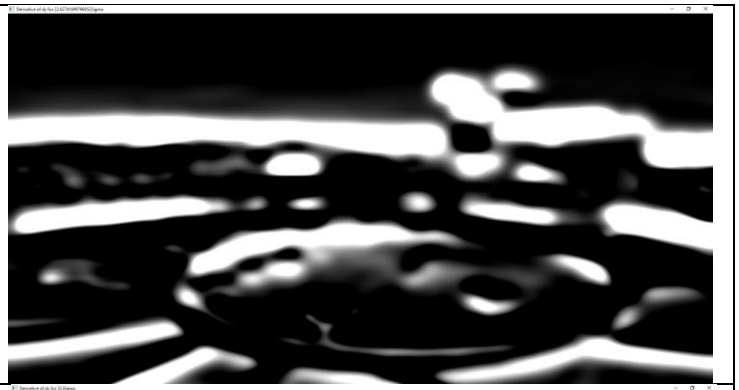
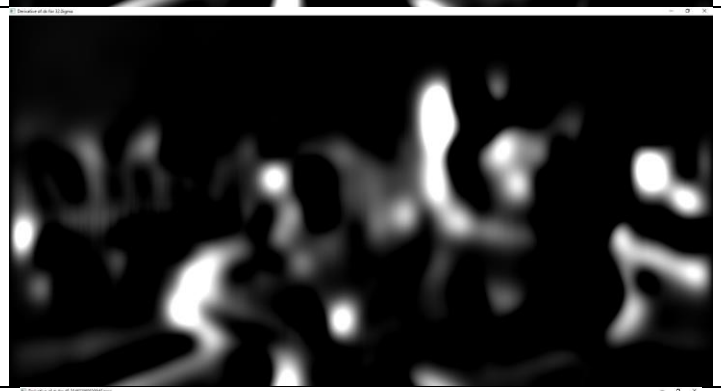
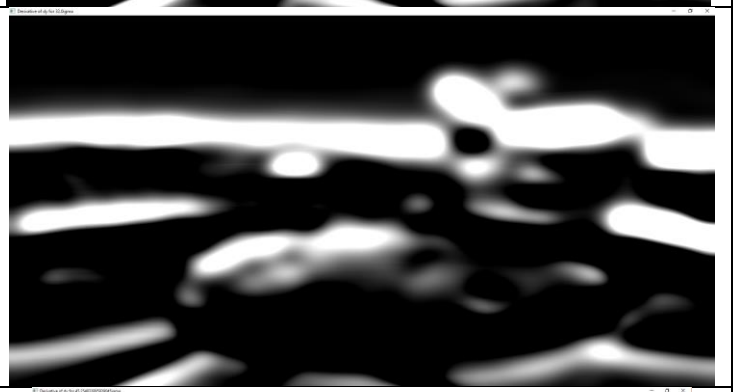
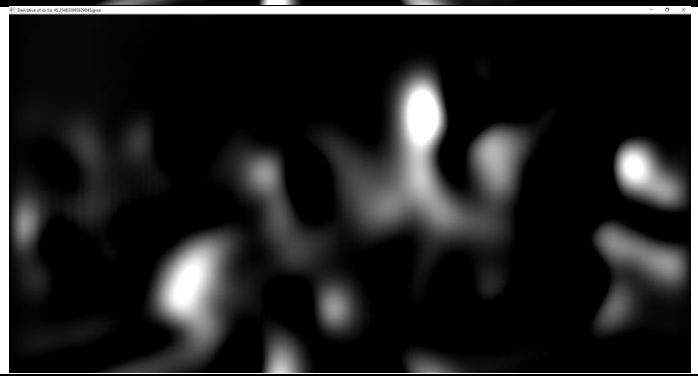
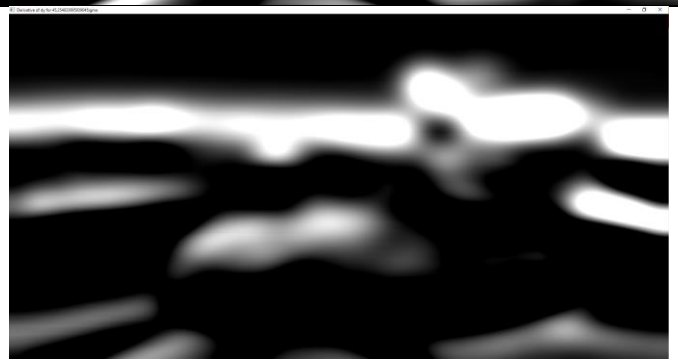
0.0030 0.0133 0.0219 0.0133 0.0030 dx = [1 0 -1]
0.0133 0.0596 0.0983 0.0596 0.0133

After applying to this image, the result will be a Gaussian smoothed image in both x and y directions.
In our case, we get 24 images 12 for x-direction and 12 for y-direction.
Basically, we can see the smoothed image edges clearly in a specific direction

Sigma Value	Derivative of X X direction	Derivative of y Y direction
1.0		
1.41		
2.0		

2.82		
4.0		
5.65		

8.0		
11.31		
16.0		

22.62		
32.0		
45.25		

From above Derivative of images, we can visualise that with increasing in the value of sigma the derivative in each direction is increasing.

Task 3 – B (1): Make grid 7x7

To make the grid meshgrid is used using linespace

```
q, r = np.meshgrid(
    (3/2) * np.linspace(-2, 2, 7) * k['sigma'],
    (3/2) * np.linspace(-2, 2, 7) * k['sigma'],
)
```

A 7 x7 grid is made for the orientation

Task 3 – B (2): Find gradient length and Gradient Directions

"An image gradient is a directional change in an image's intensity or color. The image's gradient is a fundamental building block in image processing. The Canny edge detector, for example, detects edges using an image gradient. The term gradient or color gradient is also used in graphics software for digital image editing to describe a gradual blend of color that can be thought of as an even gradation from low to high values, as used from white to black in the images to the right. Color progression is another term for this.

The gradient of a two-variable function (in this case, the image intensity function) at each image point is a 2D vector with components given by the derivatives in the horizontal and vertical directions. The gradient vector at each image point points in the direction of the greatest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction.

Because a digital image's intensity function is only known at discrete points, derivatives of this function cannot be defined unless we assume that there is an underlying continuous intensity function that has been sampled at the image points. The derivative of the continuous intensity function can be computed as a function on the sampled intensity function, i.e., the digital image, with some additional assumptions. These derivative functions can be approximated to varying degrees of accuracy. Convolving an image with a kernel, such as the Sobel or Prewitt operators, is the most common way to approximate the image gradient".[3]

Task 3 – B (3): Finding angle, orientation histogram

The histogram of oriented gradients (HOG) is a feature descriptor that is used in computer vision and image processing to detect objects. The technique counts the number of times a gradient orientation appears in a localized portion of an image. This method is similar to edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but it is computed on a dense grid of uniformly spaced cells and employs overlapping local contrast normalization for improved accuracy.

Orientation binning

"The next step in the calculation is to generate the cell histograms. Based on the values found in the gradient computation, each pixel within the cell casts a weighted vote for an orientation-based

histogram channel. The cells can be rectangular or radial in shape, and the histogram channels are evenly spaced across 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is "unsigned" or "signed." In their human detection experiments, Dalal and Triggs discovered that unsigned gradients combined with 9 histogram channels performed best. In terms of vote weight, pixel contribution can be either the gradient magnitude itself or some function of it. In tests, the magnitude of the gradient produces the best results. The square root or square of the gradient magnitude, or some clipped version of the magnitude, could also be used as the vote weight." [4]

Task 4: Feature Descriptors

"A feature detector is an algorithm that takes an image and outputs the locations (pixel coordinates) of significant areas in the image. A corner detector is an example of this, as it outputs the locations of corners in your image but does not provide any additional information about the features detected.

A feature descriptor is an algorithm that takes an image and generates feature descriptors/feature vectors from it. Feature descriptors encode interesting information into a series of numbers and serve as a numerical "fingerprint" that can be used to distinguish one feature from another. This information should ideally be invariant under image transformation so that we can find the feature even if the image is transformed in some way".[5]

Here in we are making 4x4 small grid and joining them to make it 16x16 grid later concatenating all the 16 histograms formed with 8 vectors into single 128- vector.

Visualization of feature descriptor

Reference:

1. https://matthew-brett.github.io/teaching/smoothing_intro.html
2. https://datacarpentry.org/image-processing/07-thresholding/
3. https://en.wikipedia.org/wiki/Image_gradient#:~:text=At%20each%20image%20point%2C%20the,of%20change%20in%20that%20direction.
4. https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients#:~:text=The%20histogram%20of%20oriented%20gradients,localized%20portions%20of%20an%20image.
5. https://stackoverflow.com/questions/27595455/what-is-a-feature-descriptor-in-image-processing-algorithm-or-description
6. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
7. https://matthew-brett.github.io/teaching/smoothing_intro.html
8. https://en.wikipedia.org/wiki/Difference_of_Gaussians
9. https://www.olympus-lifescience.com/en/microscope-resource/primer/java/digitalimaging/processing/diffgaussians/