Answer 1:

Steps that are involved in Canny Edge Detection is :

- Smoothing the image with Gaussian filter
- Compute Derivative of filtered image
- Find the magnitude and Orientation of Gradient
- Apply non-maximum suppression
- At last, apply Hysteresis Threshold

In our implementation:

1. Firstly, we read a 4 gray scale images from Berkeley Segmentation Dataset and stored it in matrix named I.
2. Then, 1D Gaussian filter is created with function name as gaussian1D which takes input as kernel size and sigma. Below is the equation of Gaussian that is being calculated and generates the 1D array G(x) as G1 and reshape it to get G(y) as G2 also.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

3. In the next step, computed the first-derivation of Gaussian filter with function name as gaussian1DDerivative which takes input as kernel size and sigma. Below is the equation of first-derivation of Gaussian that is being calculated G'(x) as Gx and reshape it to get G'(y) as Gy

$$(-x/\sigma^2)*G(x)$$

4. Convolve the image I with G along the rows to give the x component image (Ix), and down the columns to give the y component image (Iy).

5. Convolve Ix with Gx to give the x component of I convolved with the derivative of the Gaussian as Ix_new, and convolve Iy with Gy to give the y component of I convolved with the derivative of the Gaussian as Iy_new.

6. Compute the gradient magnitude and gradient direction with the below formula:

gradient_magnitude = np.sqrt(np.square(new_Ix) + np.square(new_Iy))
gradient_direction =(np.arctan2(new_Iy,new_Ix))

7. Converting Gradient direction from radian to degree
   gradient_direction = np.rad2deg(gradient_direction)

8. Applying non_max_suppression on gradient_magnitude and gradient_direction

   We consider the point in the center say q, along the direction of the gradient axis we find two other points, p & r such that p lies away from q & r lies before q on the axis, if q is greater than both p & r we keep q or else we mark it as zero.

   Depending on the angle we have considered 4 cases and according set the value of before_pixel and after_pixel.

```
 if (0 <= direction < PI / 4) or (PI <= direction <= 5 * PI / 4):
    before_pixel = gradient_magnitude[row, col - 1]
    after_pixel = gradient_magnitude[row, col + 1]

elif (PI / 4 <= direction <  PI / 2) or (5 * PI / 4 <= direction < 3 * PI / 2):
    before_pixel = gradient_magnitude[row + 1, col - 1]
    after_pixel = gradient_magnitude[row - 1, col + 1]

elif ( PI / 2 <= direction < 6 * PI / 8) or (3 * PI / 2 <= direction < 7 * PI / 4):
    before_pixel = gradient_magnitude[row - 1, col]
    after_pixel = gradient_magnitude[row + 1, col]

else:
    before_pixel = gradient_magnitude[row - 1, col - 1]
    after_pixel = gradient_magnitude[row + 1, col + 1]
```

9. The last step, Is to apply the Hysteresis Threshold[L,H]
   Here, we will use two threshold low and high.
   If gradient at a pixel is,
   - Above high, declare it a strong edge
   - Below low, declare it a non-edge pixel
   - Between high and low then declare it as a weak edge.

Observations:

For our observations, we have considered Sigma = 1.8, Sigma = 2.0 and Sigma = 2.5.

Below are the points that are noted:

1. When we increase the sigma, it blurs out the image

2. When we increase the sigma, the density of the edges is reducing and instead of local neighboring we will get only high level edges.

3. The increase in sigma, the edges will be moving from fine-grained to coarse-level.

4. In short, large sigma detects large scale edges and small sigma detects fine edges.

5. With all these points keeping in mind, and as we need to find as close as possible to true edges by ignoring noise.

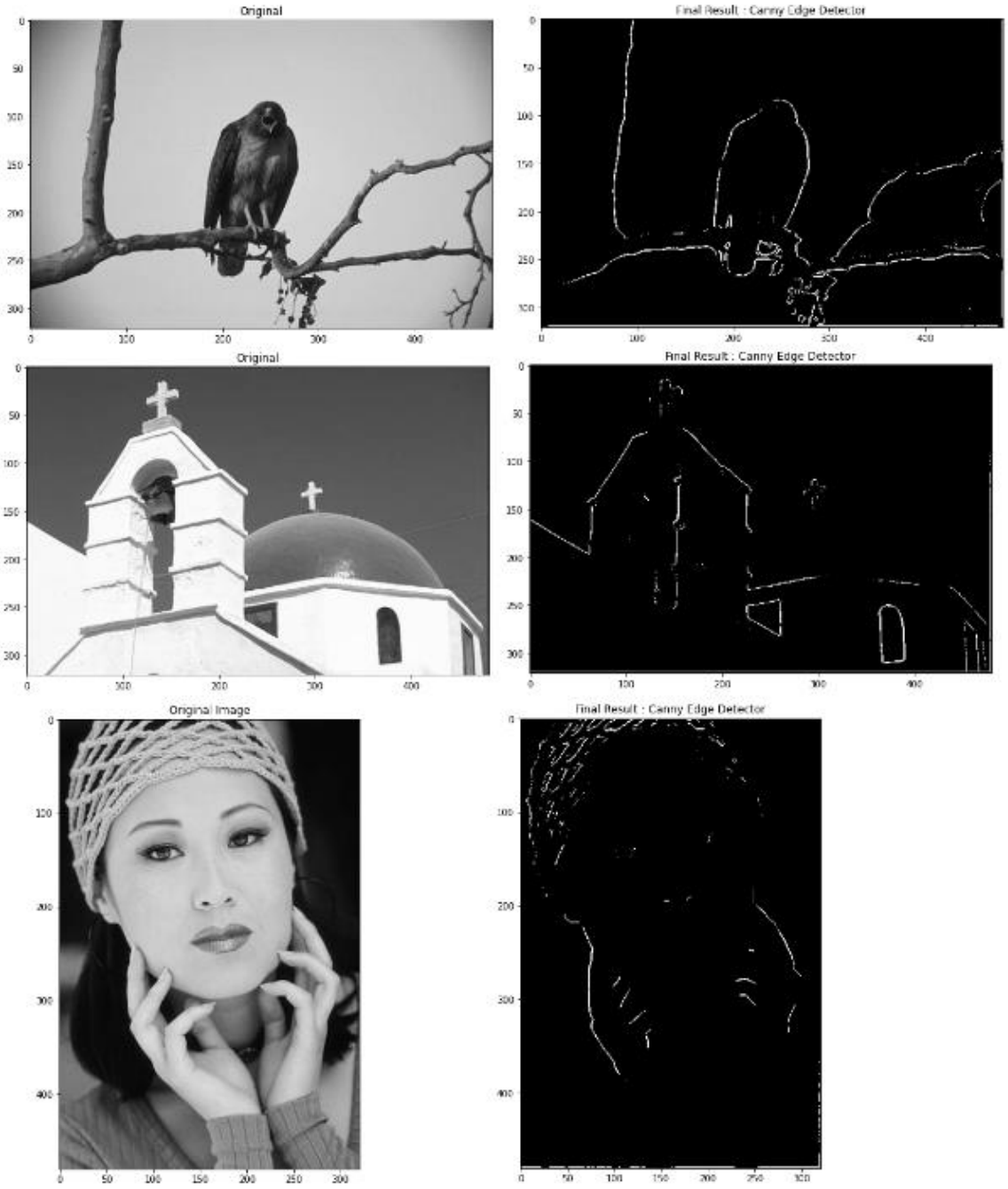
   So, we have considered Sigma = 1.8 is the best one.

   Please find below the screeshot:

Effect of Sigma in edge detection when Sigma value is 1.8

Effect of Sigma in edge detection when Sigma value is 2.0

Effect of Sigma in edge detection when Sigma value is 2.5



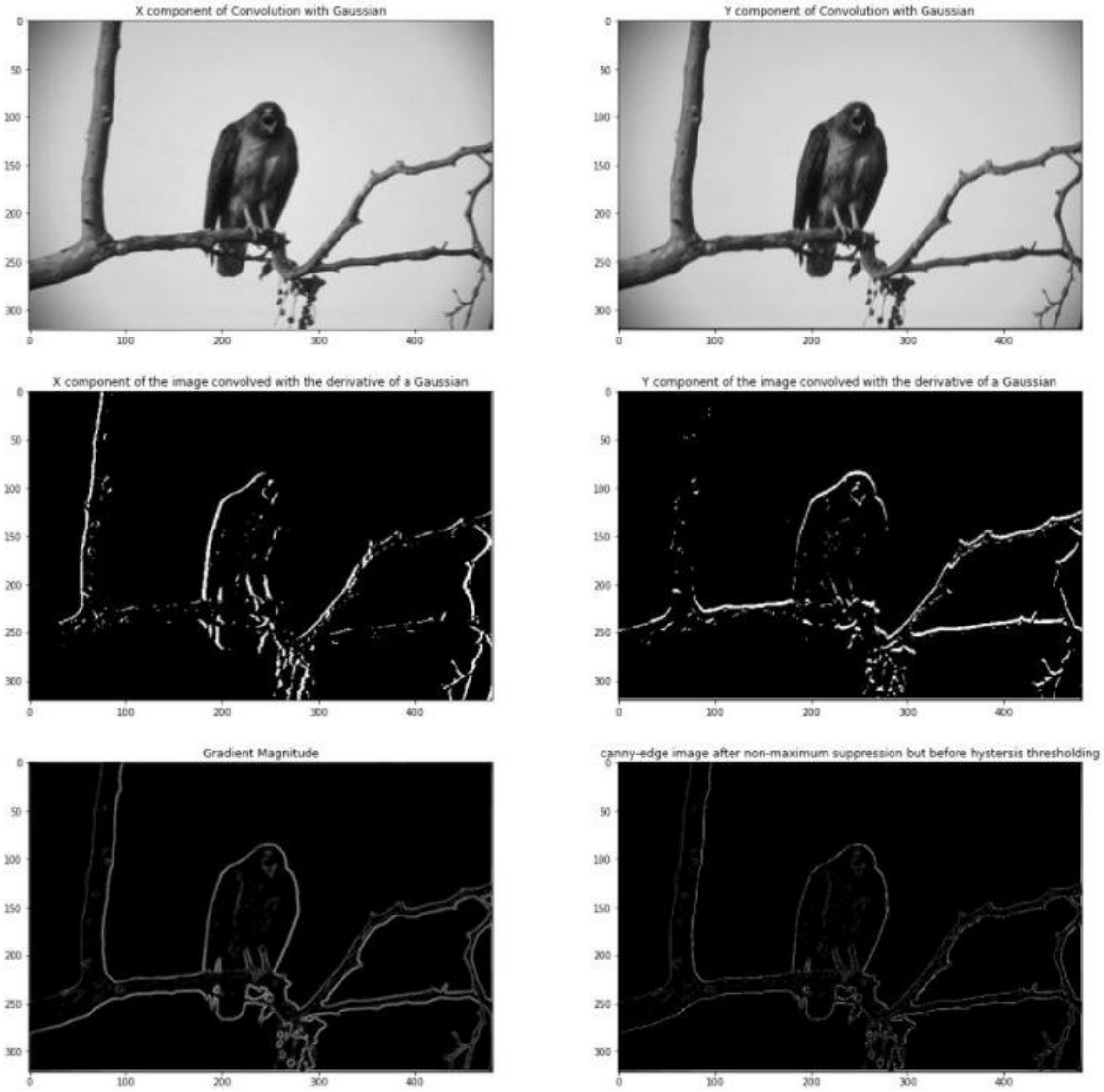
Intermediate Results of 4 images with Sigma 1.8:- Sigma = 1.8 is the best one that I have considered

IMAGE 1:

Chessboard images

(a) X component of the convolution with a Gaussian
(b) Y component of the convolution with a Gaussian
(c) X component of the image convolved with the derivative of a Gaussian
(d) Y component of the image convolved with the derivative of a Gaussian
(e) magnitude image
(f) canny-edge image after non-maximum suppression

Please refer to figure below:



Final Output :

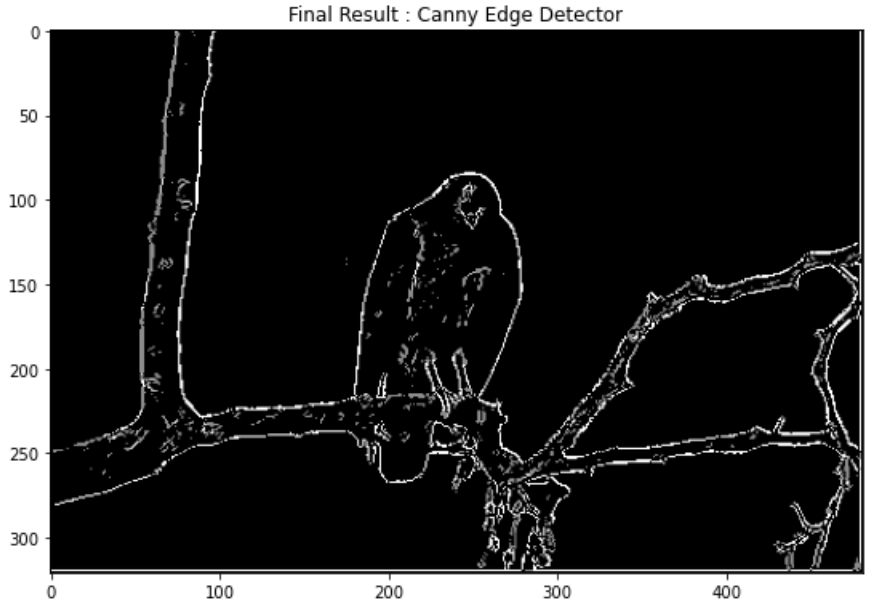
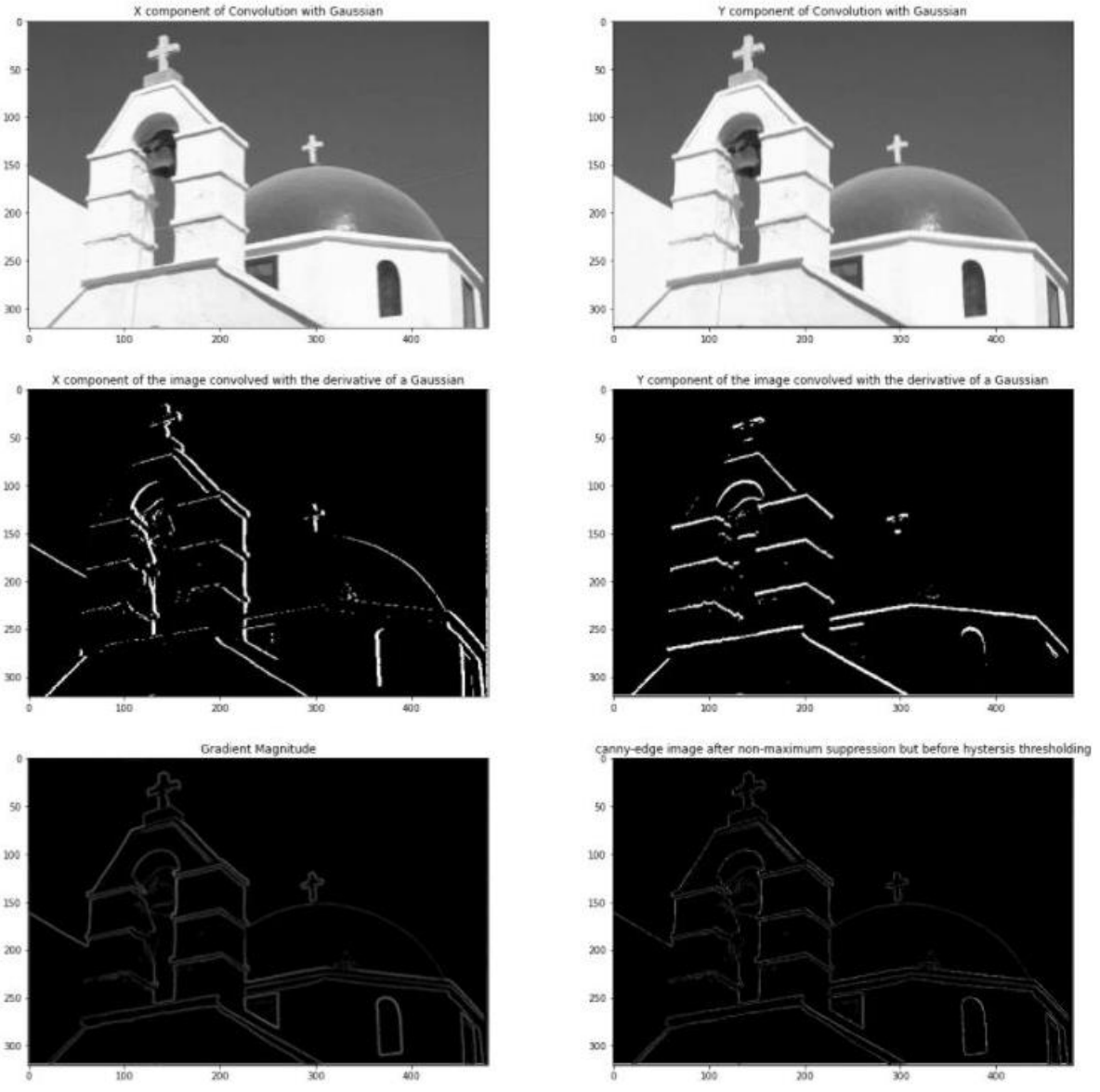


IMAGE 2:

Church images

(a) X component of the convolution with a Gaussian
(b) Y component of the convolution with a Gaussian
(c) X component of the image convolved with the derivative of a Gaussian
(d) Y component of the image convolved with the derivative of a Gaussian
(e) magnitude image
(f) canny-edge image after non-maximum suppression

Please refer to figure below:



Final Output :

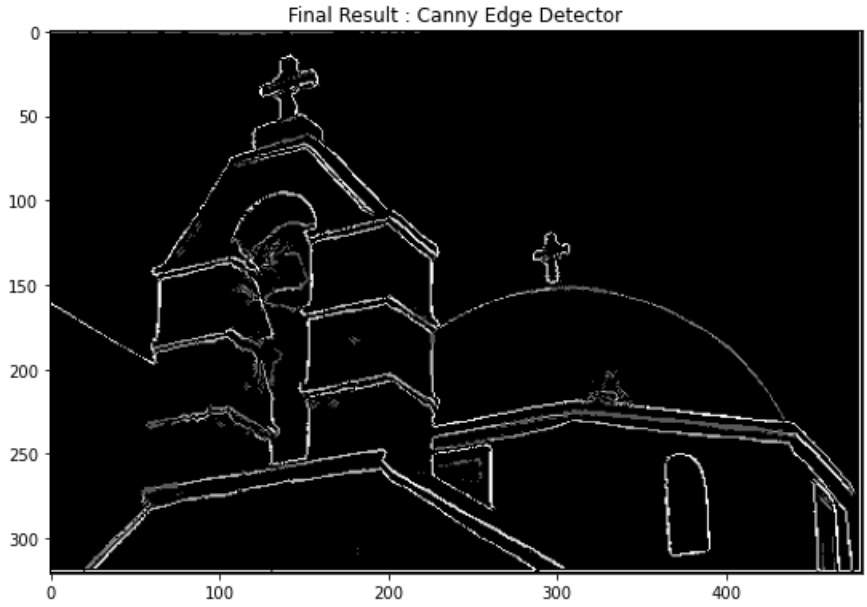
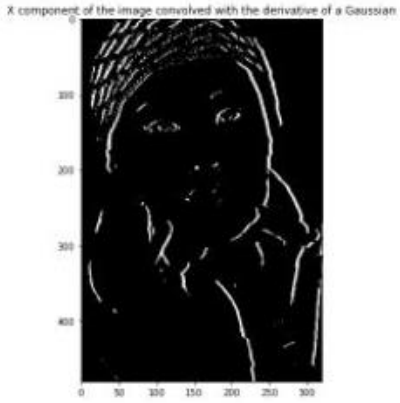


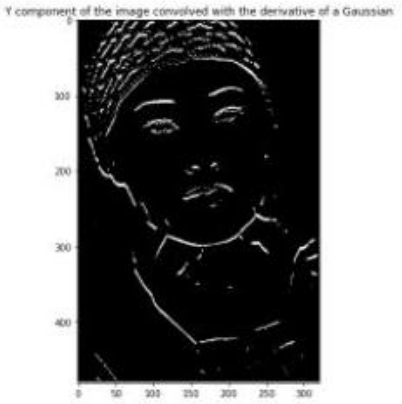
IMAGE 3:

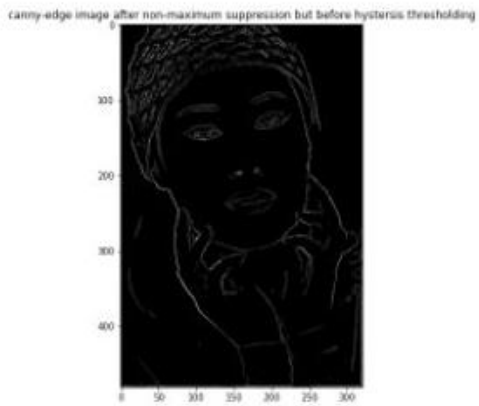
Human images

(a) X component of the convolution with a Gaussian
(b) Y component of the convolution with a Gaussian
(c) X component of the image convolved with the derivative of a Gaussian
(d) Y component of the image convolved with the derivative of a Gaussian
(e) magnitude image
(f) canny-edge image after non-maximum suppression

Please refer to figure below:







Output :

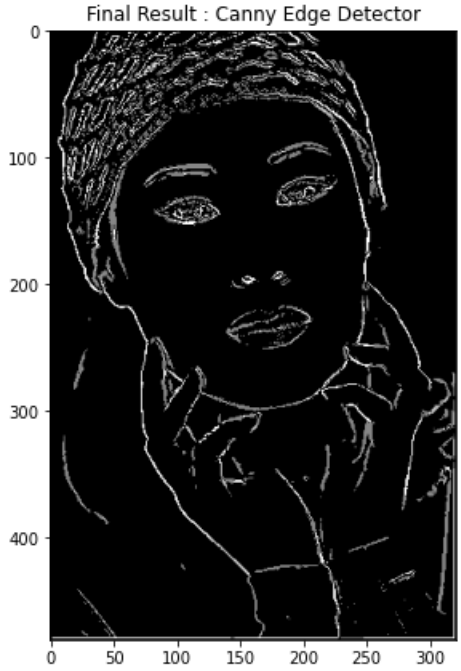
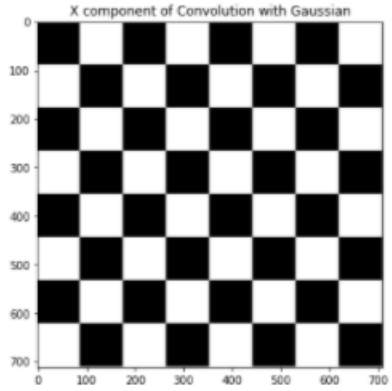

Final Result : Canny Edge Detector
0
100
200
300
400
0
50
100
150
200
250
300

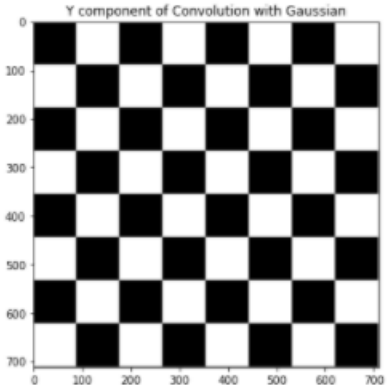
IMAGE 4:

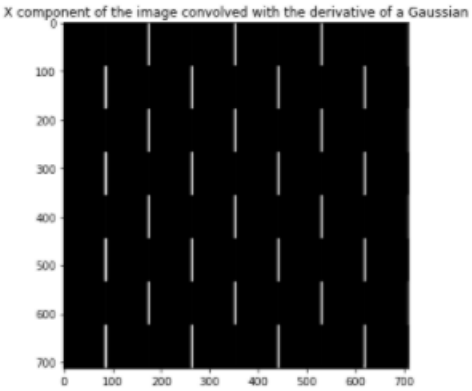
Chessboard images

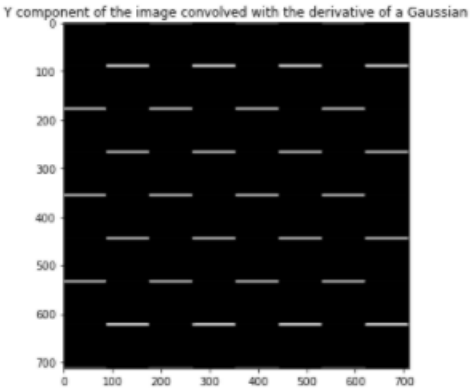
(a) X component of the convolution with a Gaussian
(b) Y component of the convolution with a Gaussian
(c) X component of the image convolved with the derivative of a Gaussian
(d) Y component of the image convolved with the derivative of a Gaussian
(e) magnitude image
(f) canny-edge image after non-maximum suppression

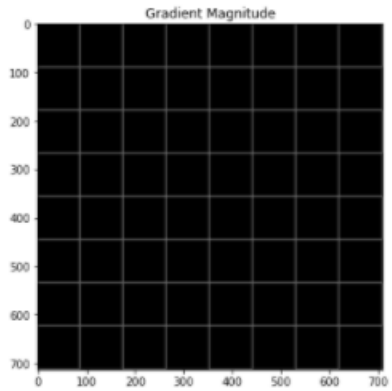
Please refer to figure below:

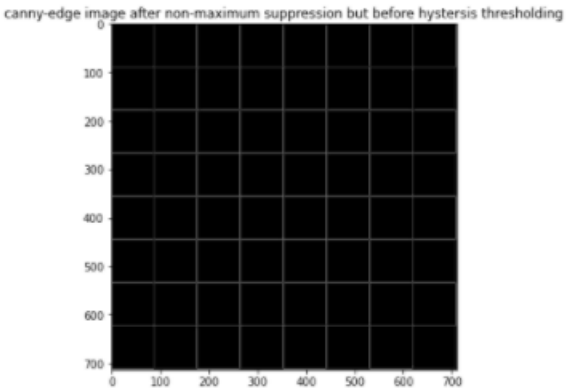




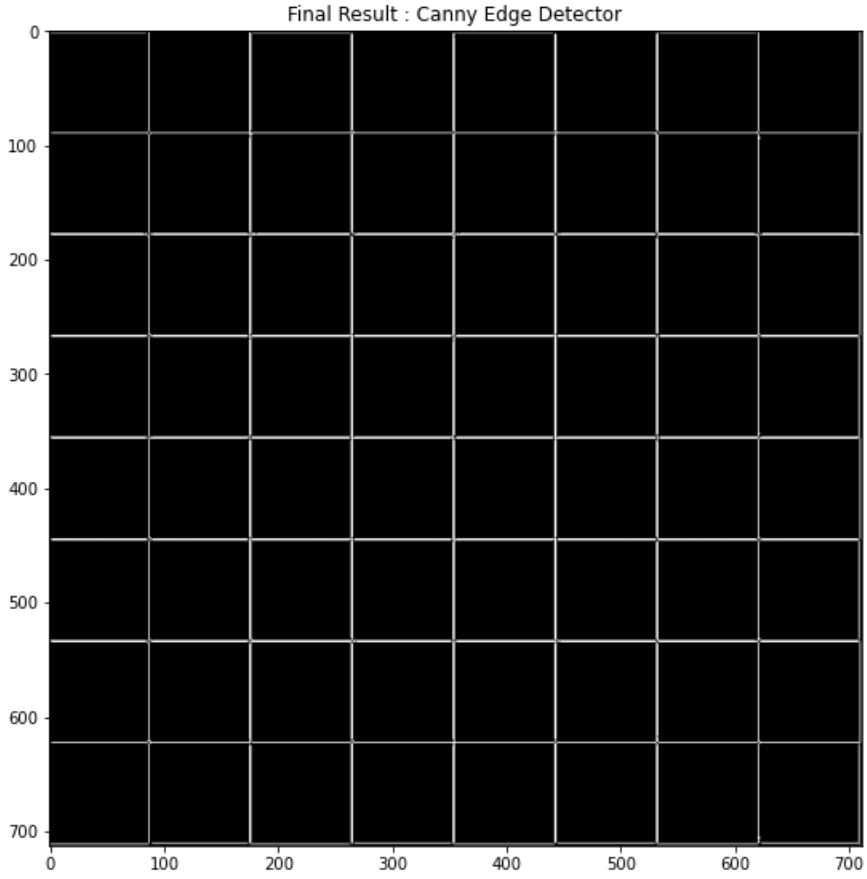








Output :



GitHub Link:

https://github.com/SunilDevlops/Programs/blob/master/ComputerVision/ProgramAssignment1/Question1/ProgramAssignment1_Question1_CannyEdgeDetector.ipynb