CAP5415: Computer Vision

*Report by: Ehtesamul Azim (ehtesamul.azim@ucf.edu) **UCFID**: 5460629*

The task is to write a python code which can:

- 1. Implement the Canny Edge Detector on three grayscale images from the Berkley Segmentation Dataset
- 2. Generating final edge plot as well as five other intermediate state plots
- 3. Experimenting with the effects of σ on edge detection.

P.S. I made changes according to Binay's last announcement which said we need to show 63 images from my rough implementation added to the zip file. Refer to the 'main_code.ipynb' file for the main implementation. Sorry for not using '.py' files, I usually do my coding stuff first of notebook

Procedures:

The implementation includes a custom CannyEdgeDetector class written in python.

We start by importing a grayscale image from the dataset using the `imread()` method from the Python `opencv` package. The image was then stored as a NumPy array. This image array was passed as the first argument to the class.
Additionally, the following parameters were set as arguments for the class:

- `sigma = sigma`, which is a list with values 0.1,2,10
- `kernel_size = 7`
- `high_threshold_ratio = 0.2`
- `low_threshold_ratio = 0.08`

Next, a 1D Gaussian mask, denoted as `G`, and its derivative, `G_x`, were generated using the `get_gaussian_mask_1d()` method. The derivative in the y-direction, `G_y`, was obtained by taking the transpose of `G_x`.

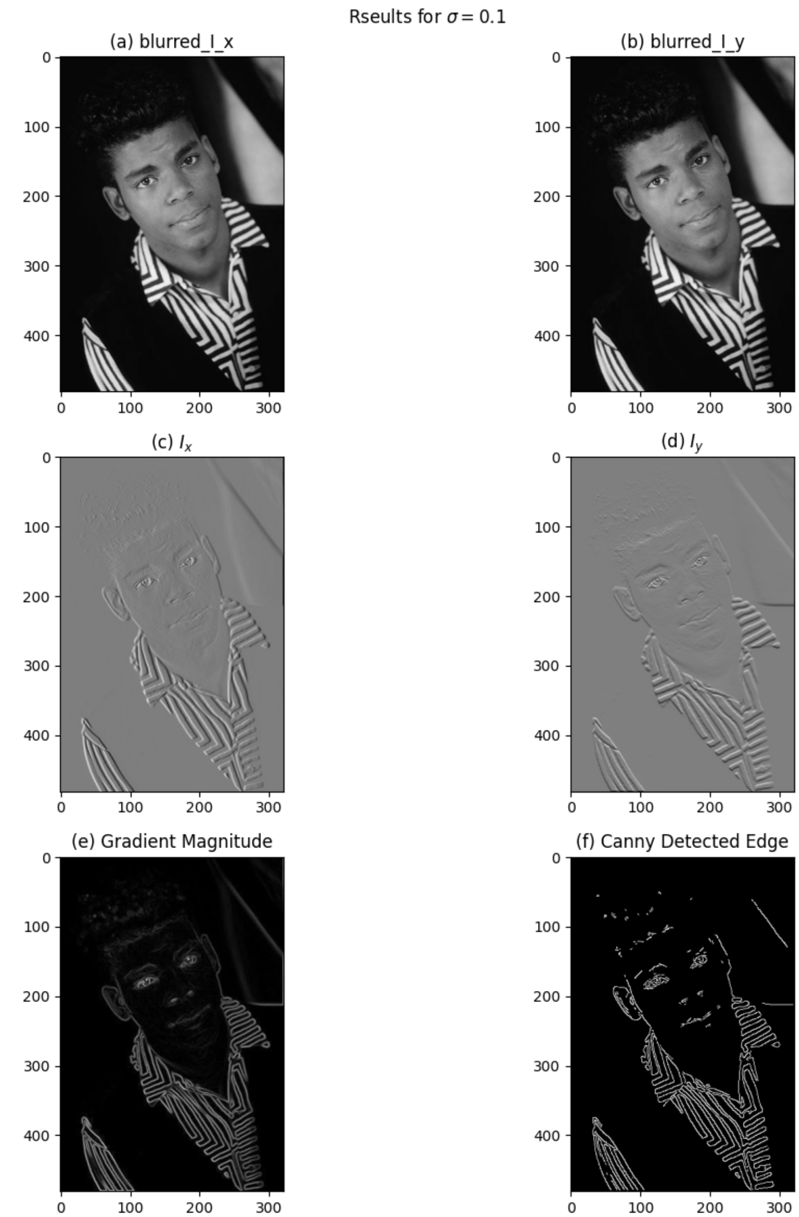
Next, the grayscale input image underwent a blurring process achieved by convolving it with the Gaussian kernel `G` in both the horizontal and vertical directions. It's important to note that the convolution operation was custom-defined within the class.

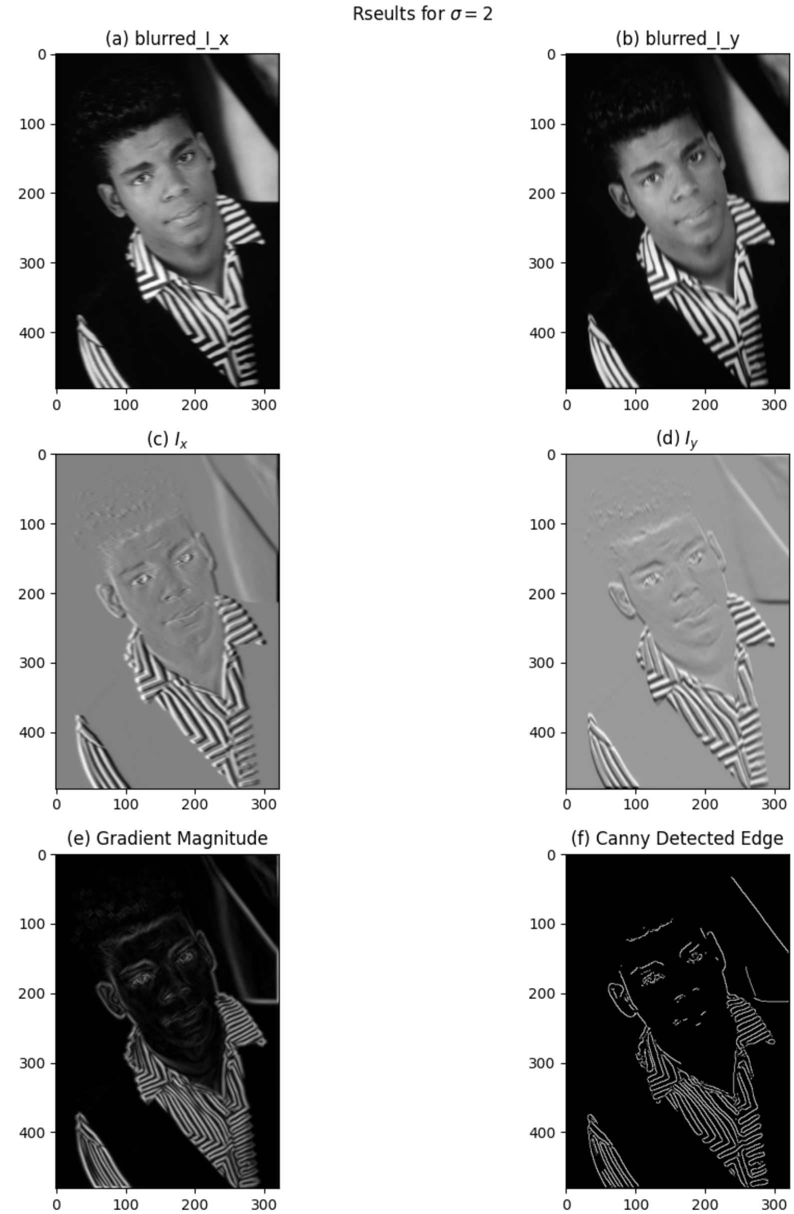
Following the blurring step, the blurred images in both the x and y axes, denoted as `blurred_I_x` and `blurred_I_y`, were further processed by convolving them with the derivatives `G_x` and `G_y`, respectively. This resulted in the generation of two gradient images: `I_x` and `I_y`.

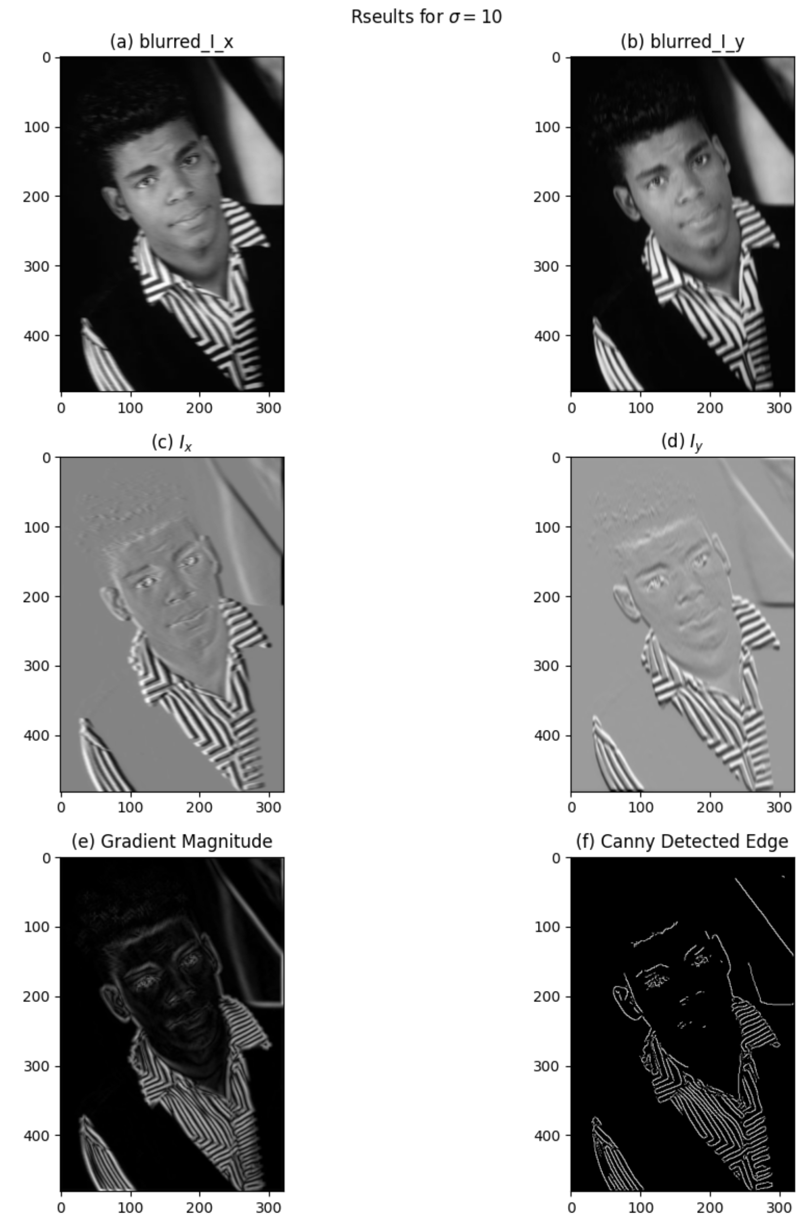
In the subsequent stage, the gradient magnitude, represented as `M_xy`, and the gradient orientation, referred to as `theta`, were computed. During this process, the magnitude values were normalized to ensure they fall within the same intensity range as the original input image. Following these calculations, the two arrays were employed to perform non-maximum suppression, a technique that retains only the most salient edge pixels. As a result, the edges in the image became more distinct and prominent.

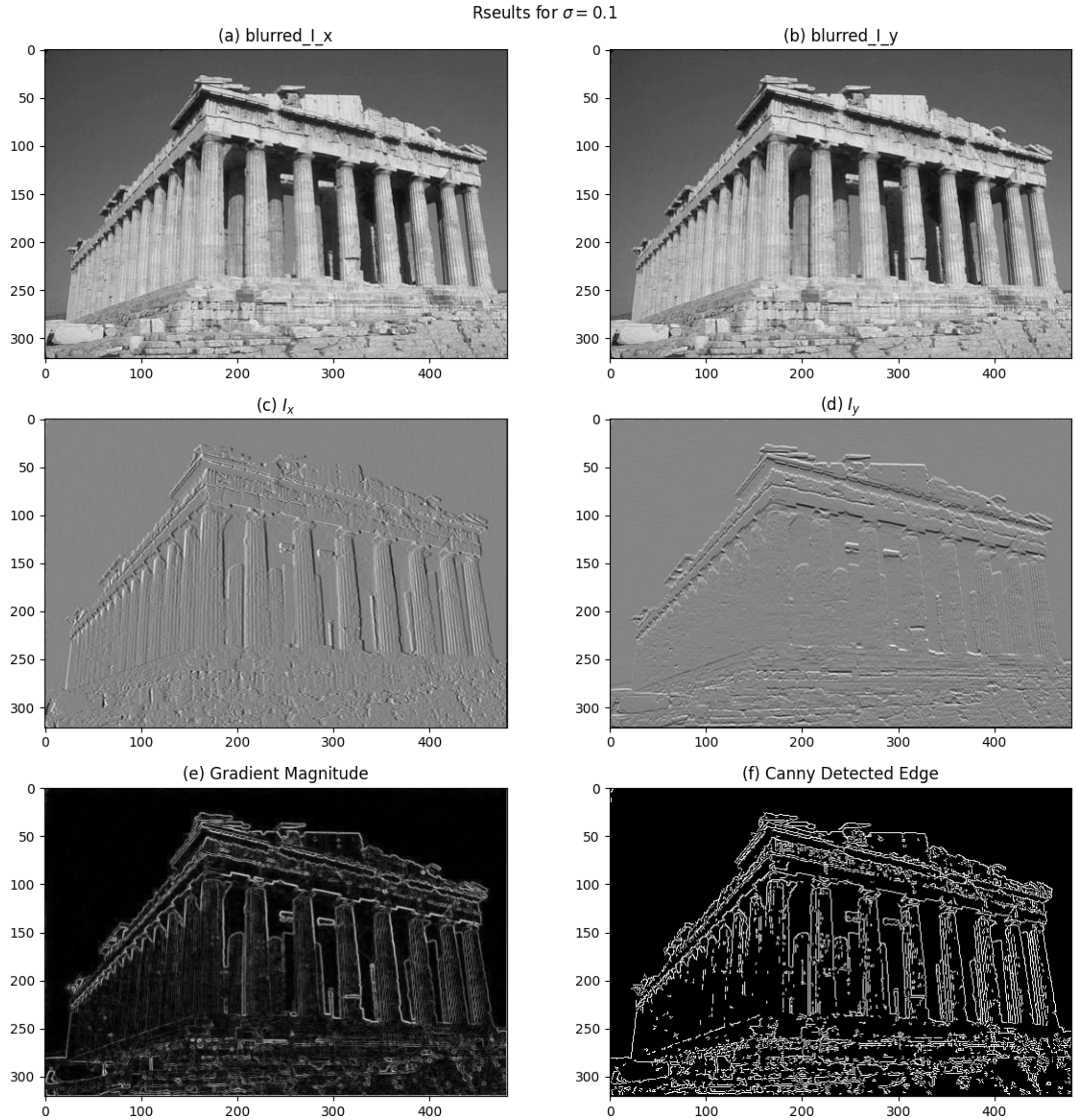
Finally, a hysteresis thresholding technique was applied to enhance the edges in the image. This method helps suppress false edges while reinforcing weaker edges that are in proximity to stronger ones. The resulting image effectively represents the edges present in the original image.

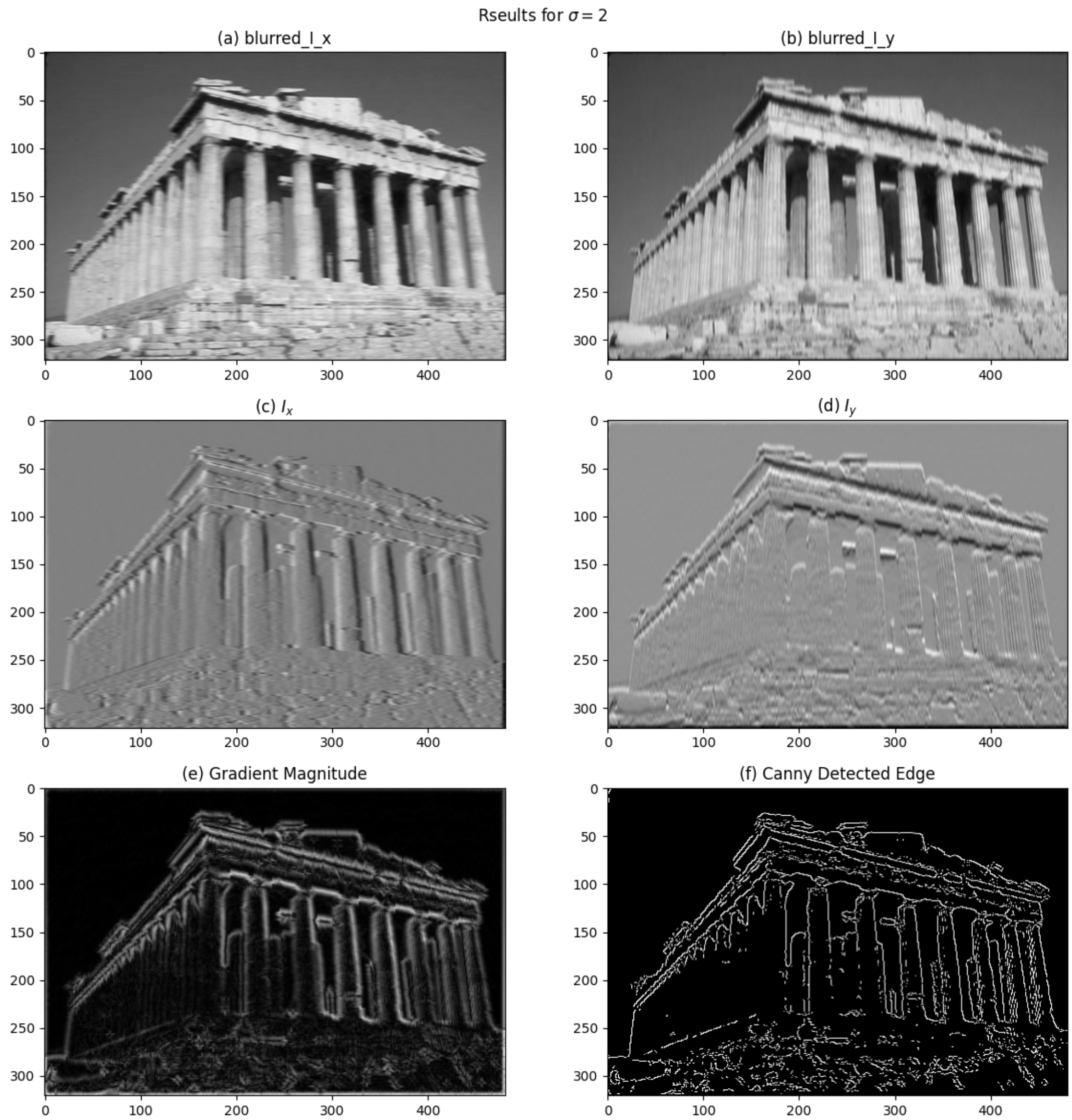
This entire process was repeated for each of the three images, and the outcomes are visualized in the subsequent figures.


Rseults for $\sigma = 0.1$
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge
0
100
200
300
400


Rseults for $\sigma = 2$
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge

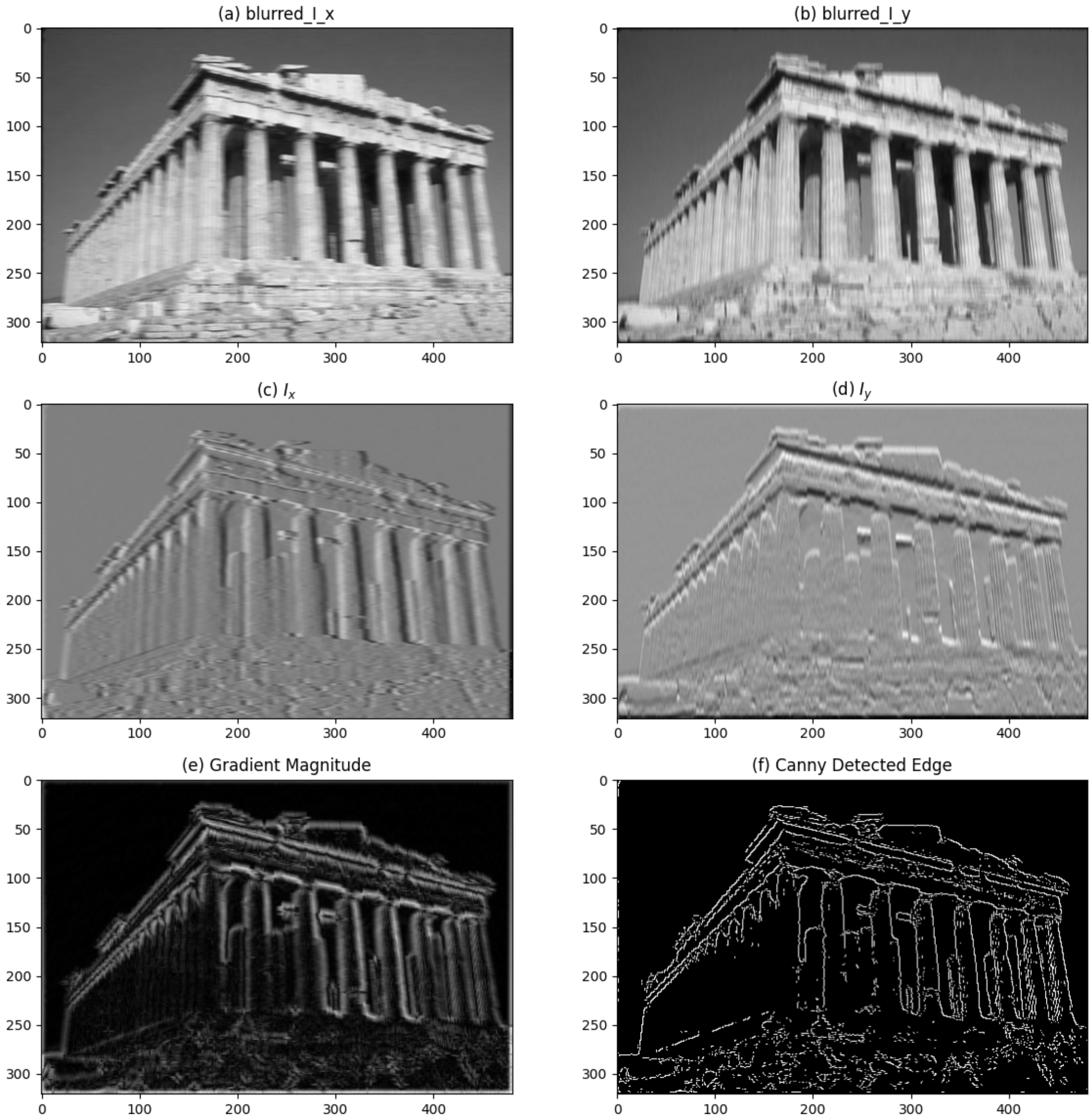

Rseults for $\sigma = 10$
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge
0
100
200
300
400

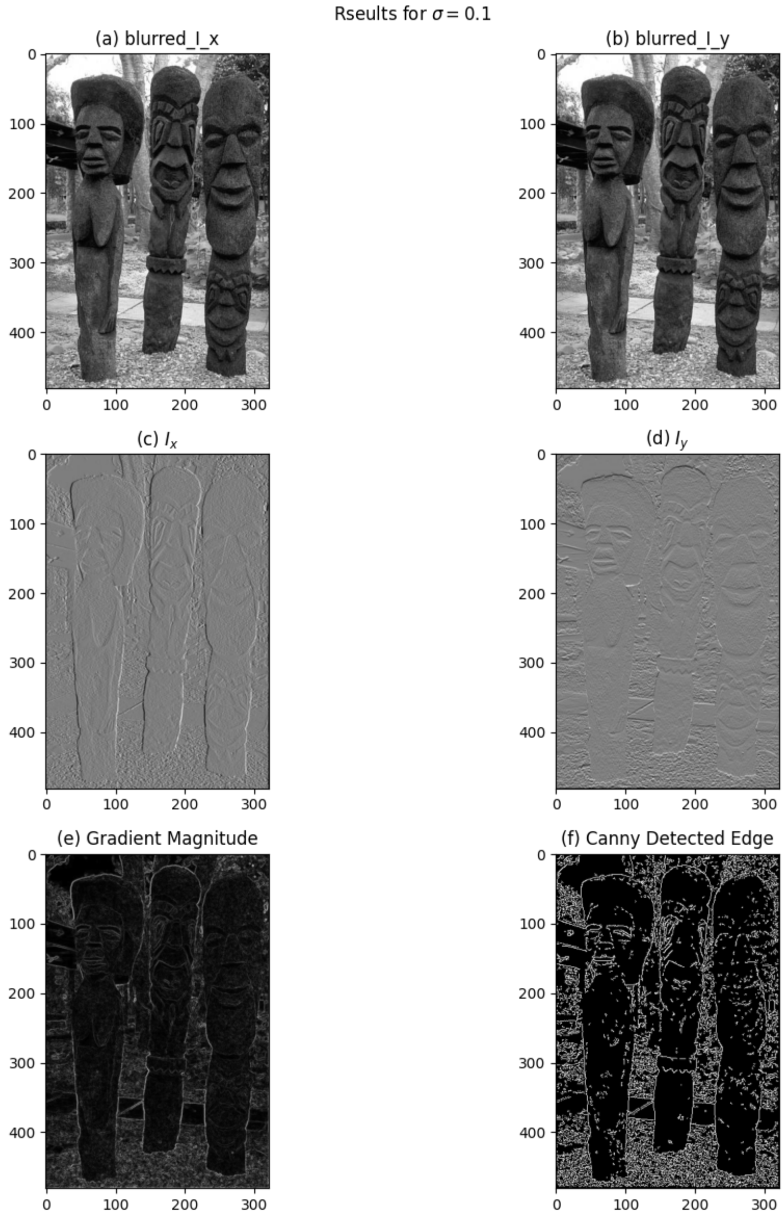

Rseults for $\sigma = 0.1$
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge

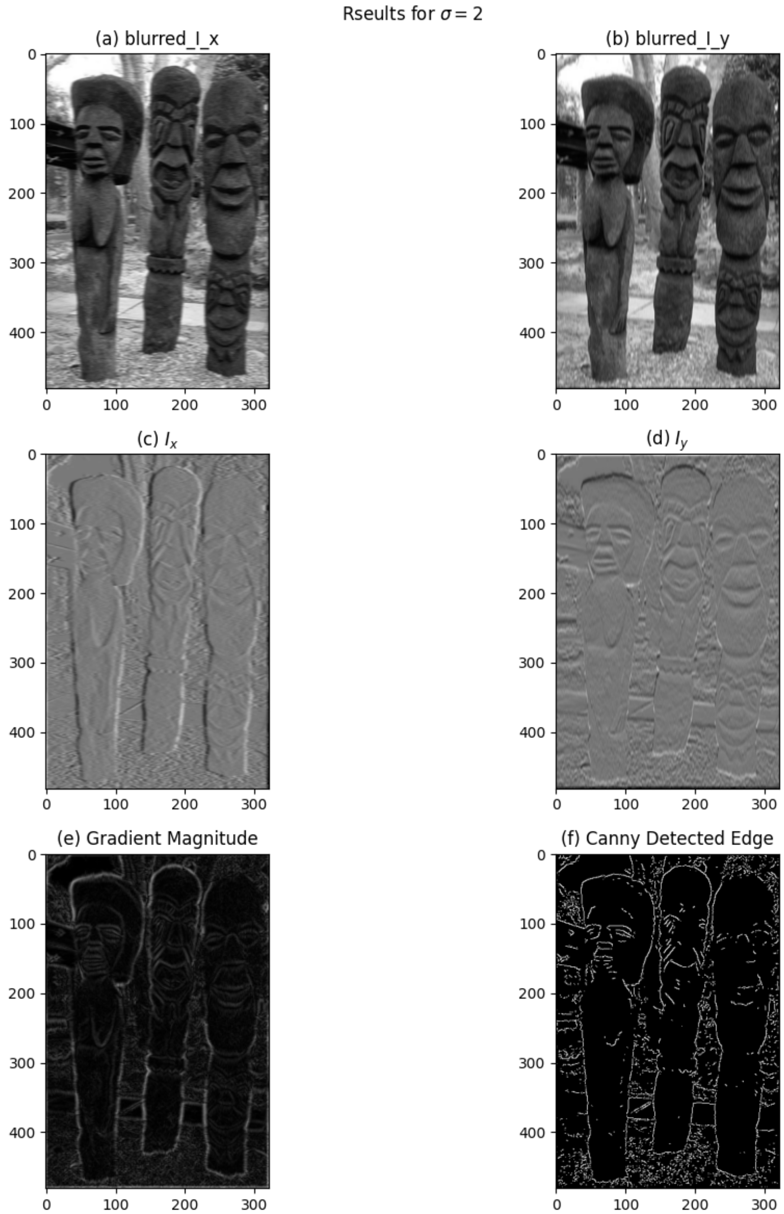


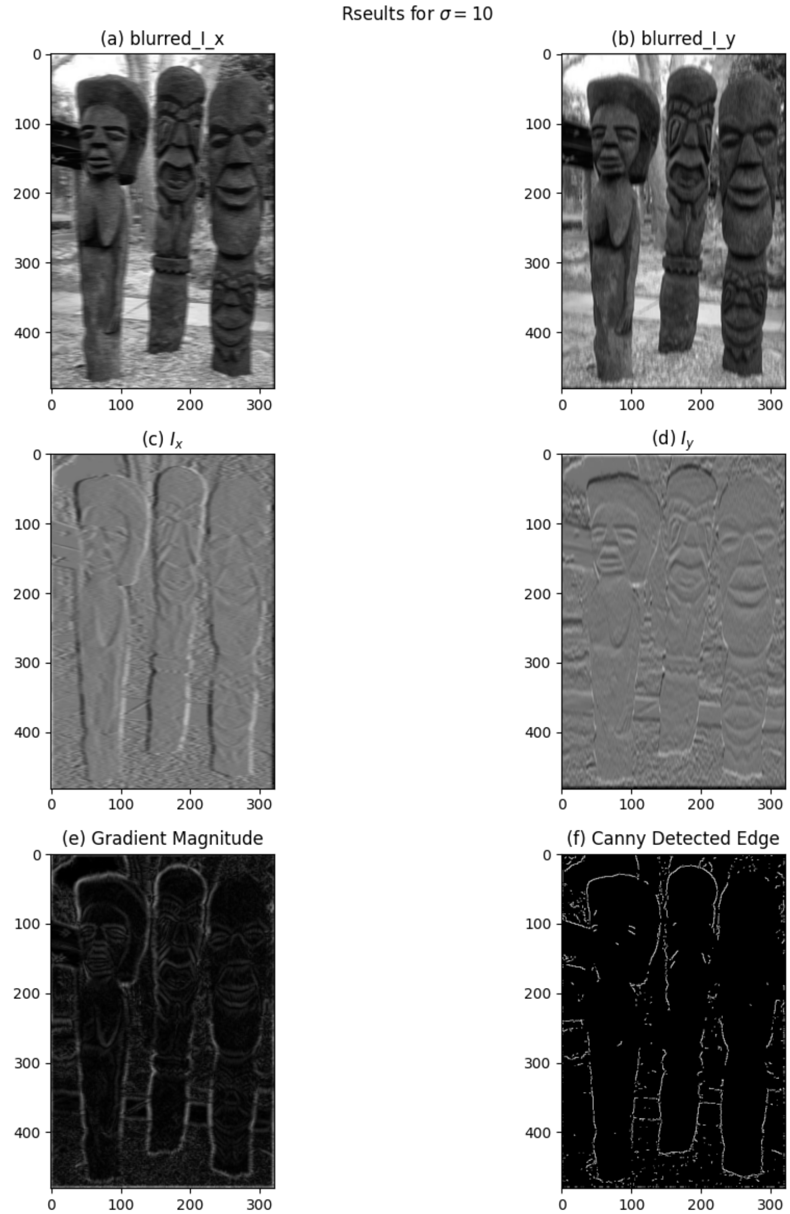
Rseults for $\sigma = 2$
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge

Rseults for $\sigma = 10$



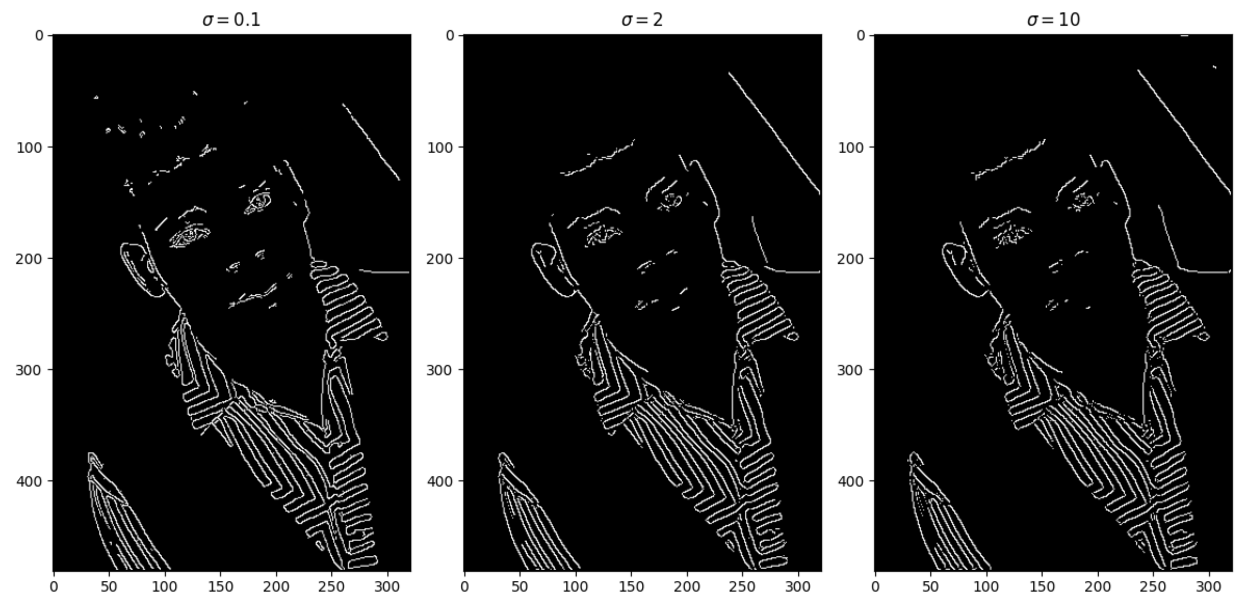

Rseults for $\sigma = 0.1$
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge
0
100
200
300
400


Rseults for σ = 2
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge

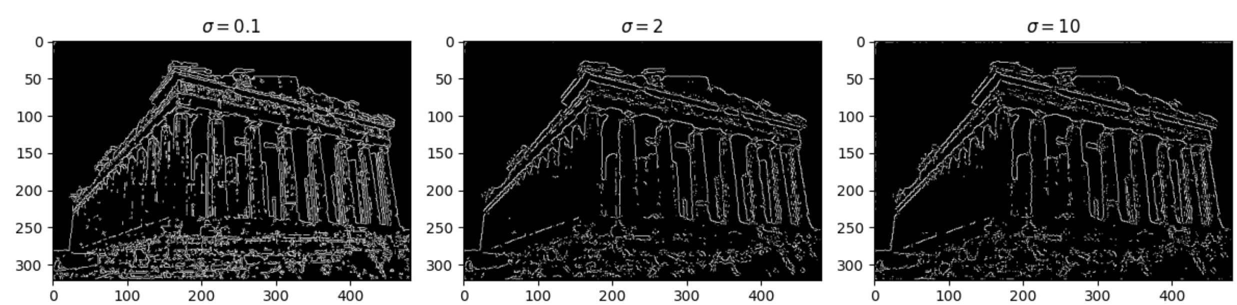

Rseults for $\sigma = 10$
(a) blurred_I_x
(b) blurred_I_y
(c) I_x
(d) I_y
(e) Gradient Magnitude
(f) Canny Detected Edge
0
100
200
300
400

Effects of σ on Canny Edges

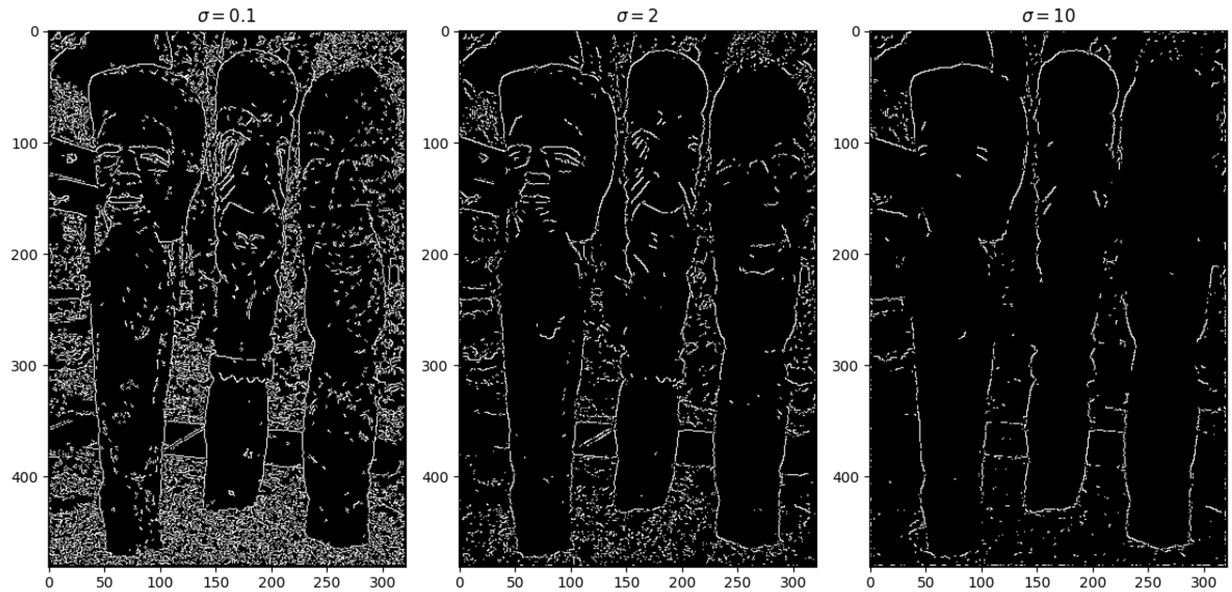
We selected three specific values for σ (0.1, 2, and 10) for this experiment. These values were deliberately chosen because they lead to noticeable differences in the output results for each image. Discussions regarding the variations observed in the outputs for each image are provided individually.



From this illustration, we observe that as σ increases, the subtle features of the dude's portrait start to disappear, specially in the hair, eyes, nose and lips region. The stripes on the shirt aren't affected that much because they are rather sharper.



For parthenon, as we increase σ value, the columns lose details and the columns in the background begin to disappear as well.



As σ increases, with the Easter Island Moai statues, there is a loss of fine details, including those depicting facial features and body parts, on the stone sculptures.

By examining the three examples with varying σ values, it's evident that a low σ value makes most of the textures highly visible, potentially overwhelming the image's clarity. Conversely, a very high σ value could result in the suppression of valuable image features.