COGS 260: Assignment 1

Aditya Verma
A53219148
Department of Electrical and Computer Engineering
University of California, San Diego
arv018@ucsd.edu

Abstract

In this assignment, we experiment with different operations on image like smoothing, enhancement and edge detection. We further extend the study to analyze/compare the efficiency of different techniques for each of the operations. The data-set used for this assignment is a subset of BSD300 dataset. The tasks were implemented using OpenCV and Scikit image library functions in Python.

1 Basic Image Operations

1.1 Image Read/Write

In this part of the question, we input any particular image in RGB format. The image in converted into Grayscale using the OpenCV library as shown in Figure 1. The grayscale image is then written to a file.

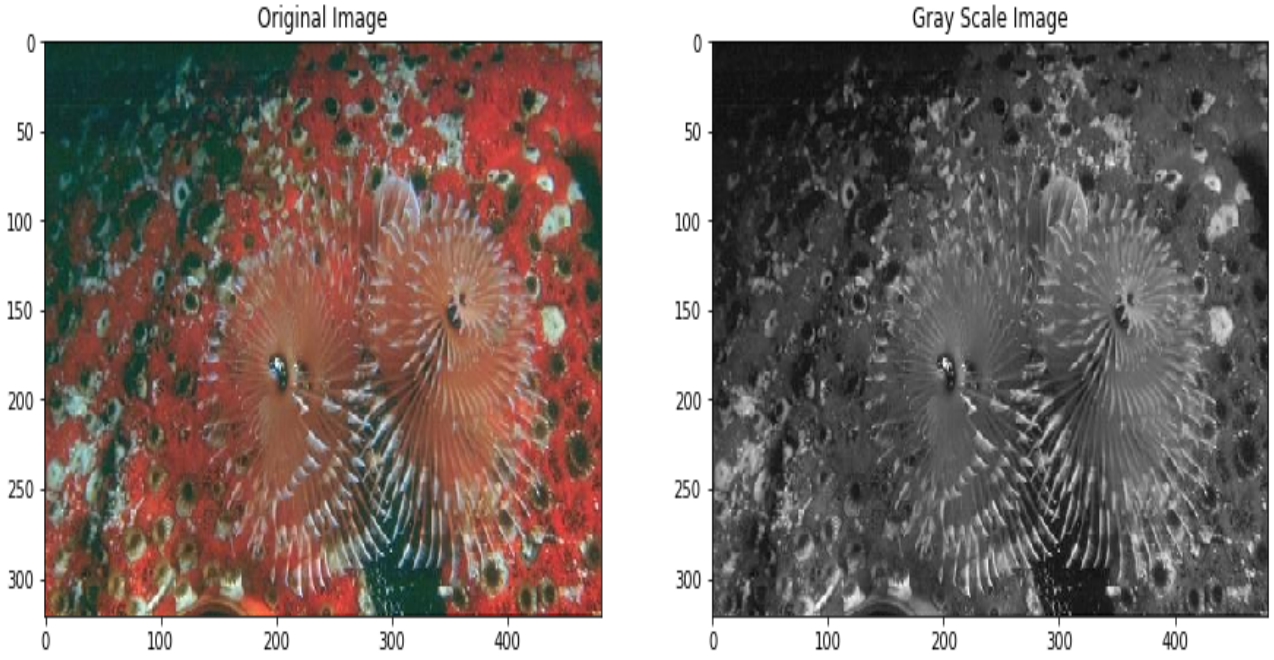

Figure 1: Original RGB image and its equivalent Grayscale image

1.2 Image Smoothing

1.2.1 Average Smoothing

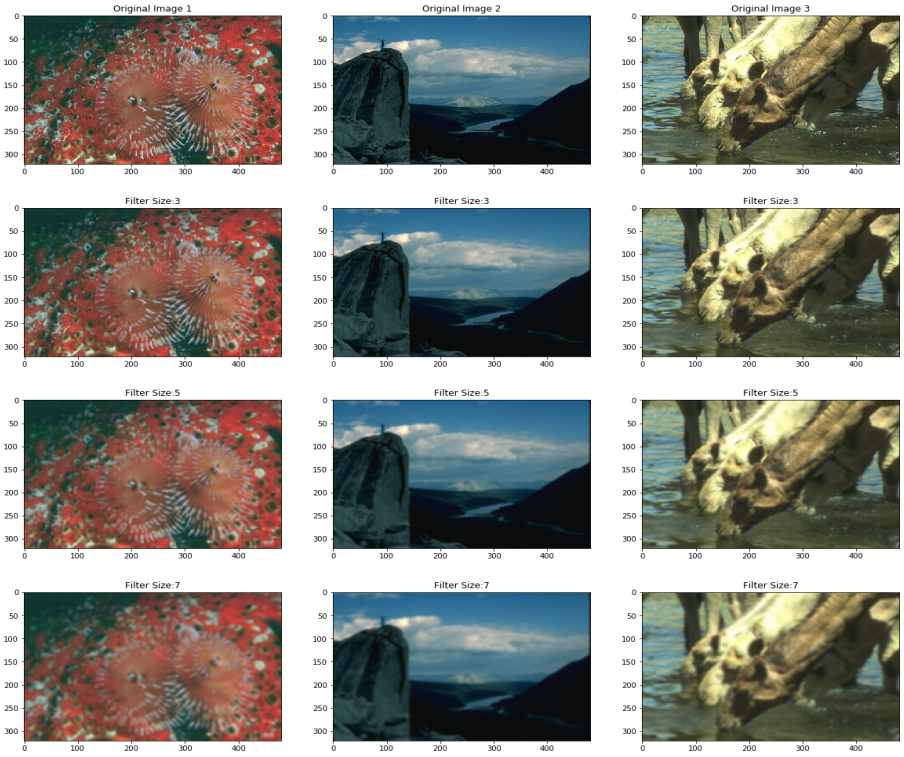

Figure 2: Smoothen image with Averaging filter of different kernel sizes.

1.2.2 Gaussian Smoothing

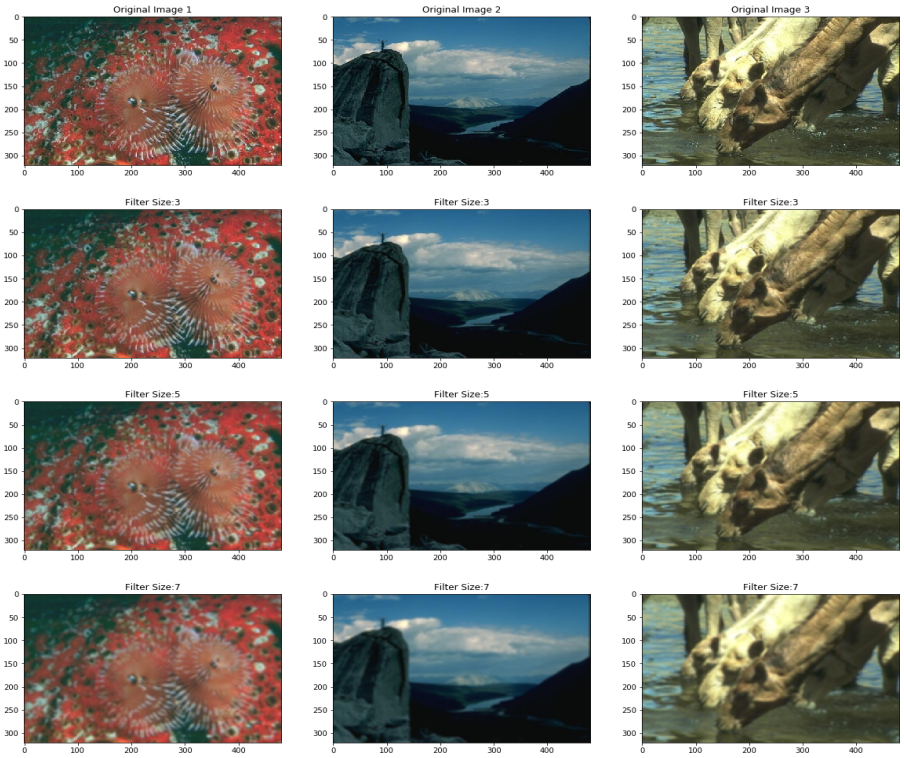

Figure 3: Smoothen image with Gaussian filter of different kernel size.

1.2.3 Compare Gaussian Smoothing vs Average Smoothing

Figure 2 & 3 shows the smoothing operation of input image using average smoothing kernel and gaussian kernel respectively. The first row of figure shows the original input image, while row 2,3,4 correspond to smoothing with kernel size 3,5,7 respectively. From the figures, it appears that Gaussian blurring still maintains more clarity in the smoothed imaged, while Average smoothing blurs the image to more extent.

As we increase the kernel size from 3 to 7, the smoothed image becomes more blurred as can be seen in any of the columns.

1.3 Denoising

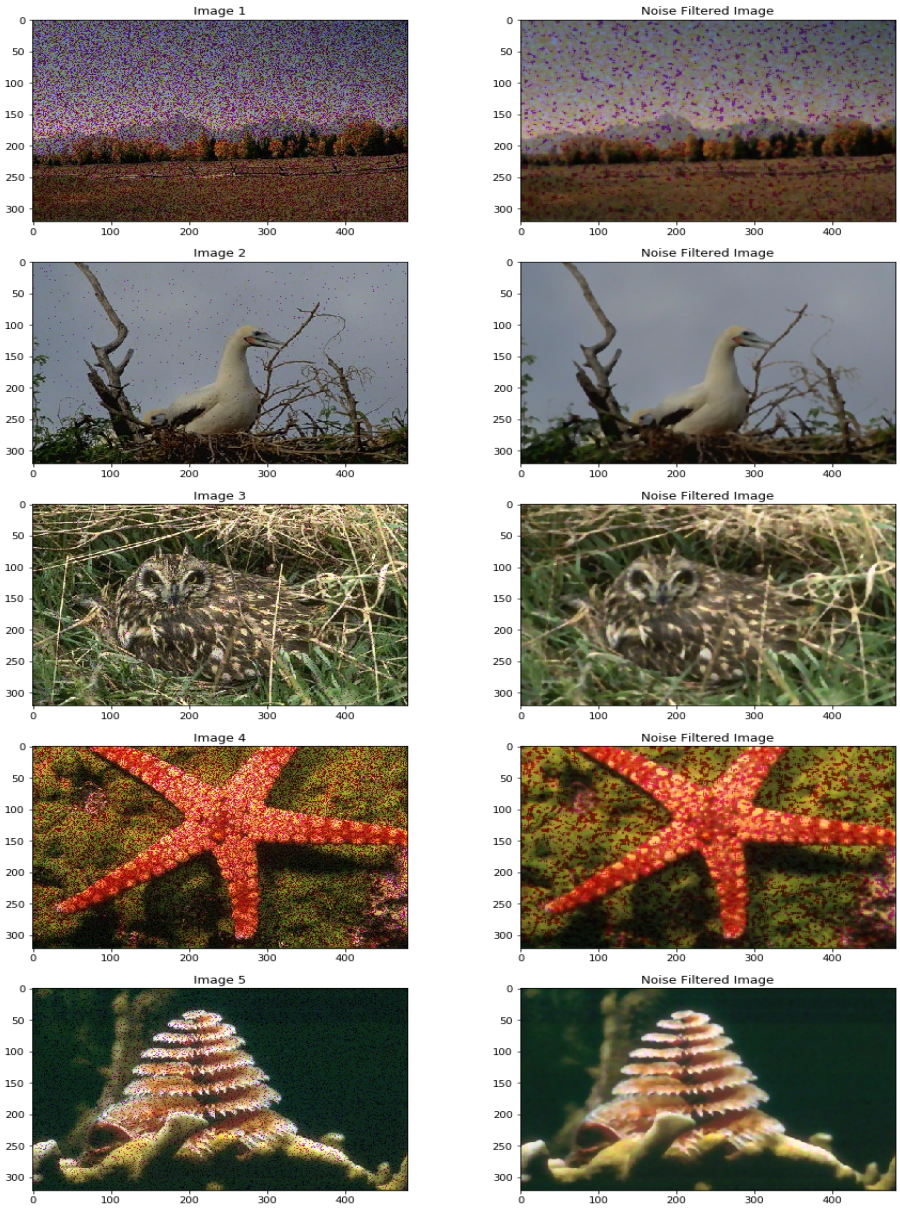

Figure 4: Denoising image by median filtering. Left: Original image with noise. Right: Denoised image

1.3.1 Observation

In Figure 4, we denoise the image which has salt and pepper noise, with median filtering. For this assignment, I used kernel size of 5 for filtering. In the figure, the left image corresponds to the original image with nosie, while right image is the denoised image. From the observation, we can say that median filtering has done more than descent job in denoising the images. In 1st image, we can see some noise still remain in the image, however a lot of noise has been removed. In the 2nd image of bird almost all of noise is removed. Similar observations can be made for all the other images.

2 Image Enhancement

2.1 Histogram

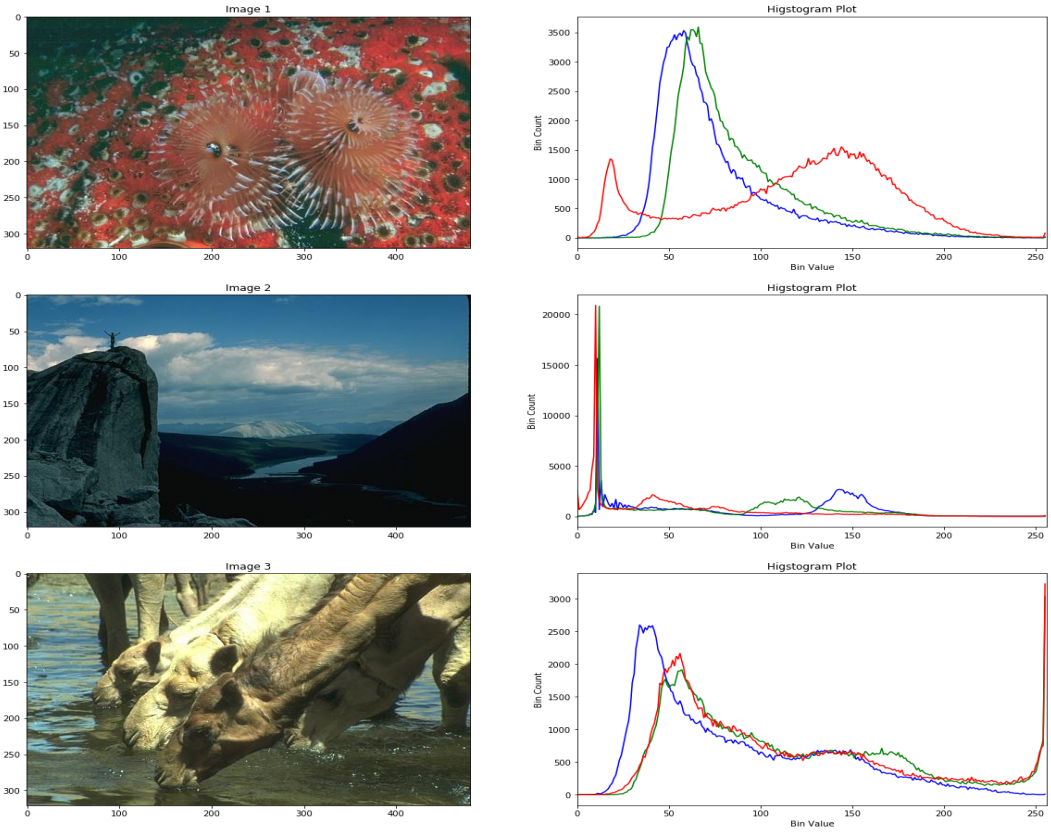

Figure 5: Higtogram plot of diffrerent images. Left:Image. Right: Histogram plot along all the 3 RGB channel

2.2 Global Histogram Equalization

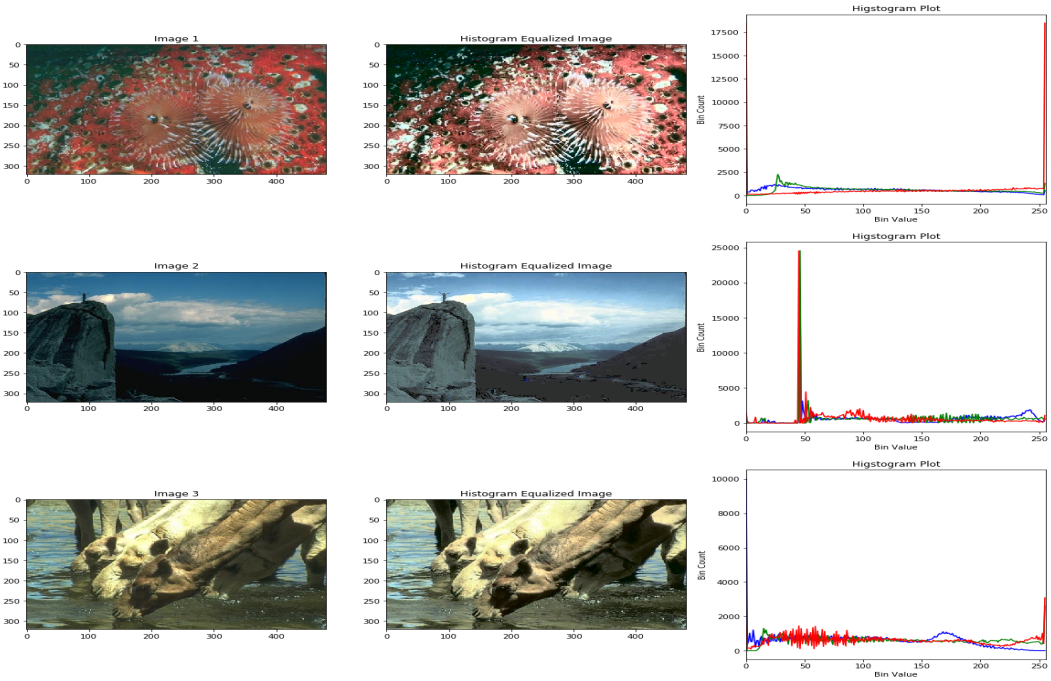

Figure 6: Global histogram equalization of images. Left:Original image. Middle: Histogram equalized image. Right: Histogram plot of equalized image

2.3 Contrast limited Adaptive Histogram Equalization(CLAHE)

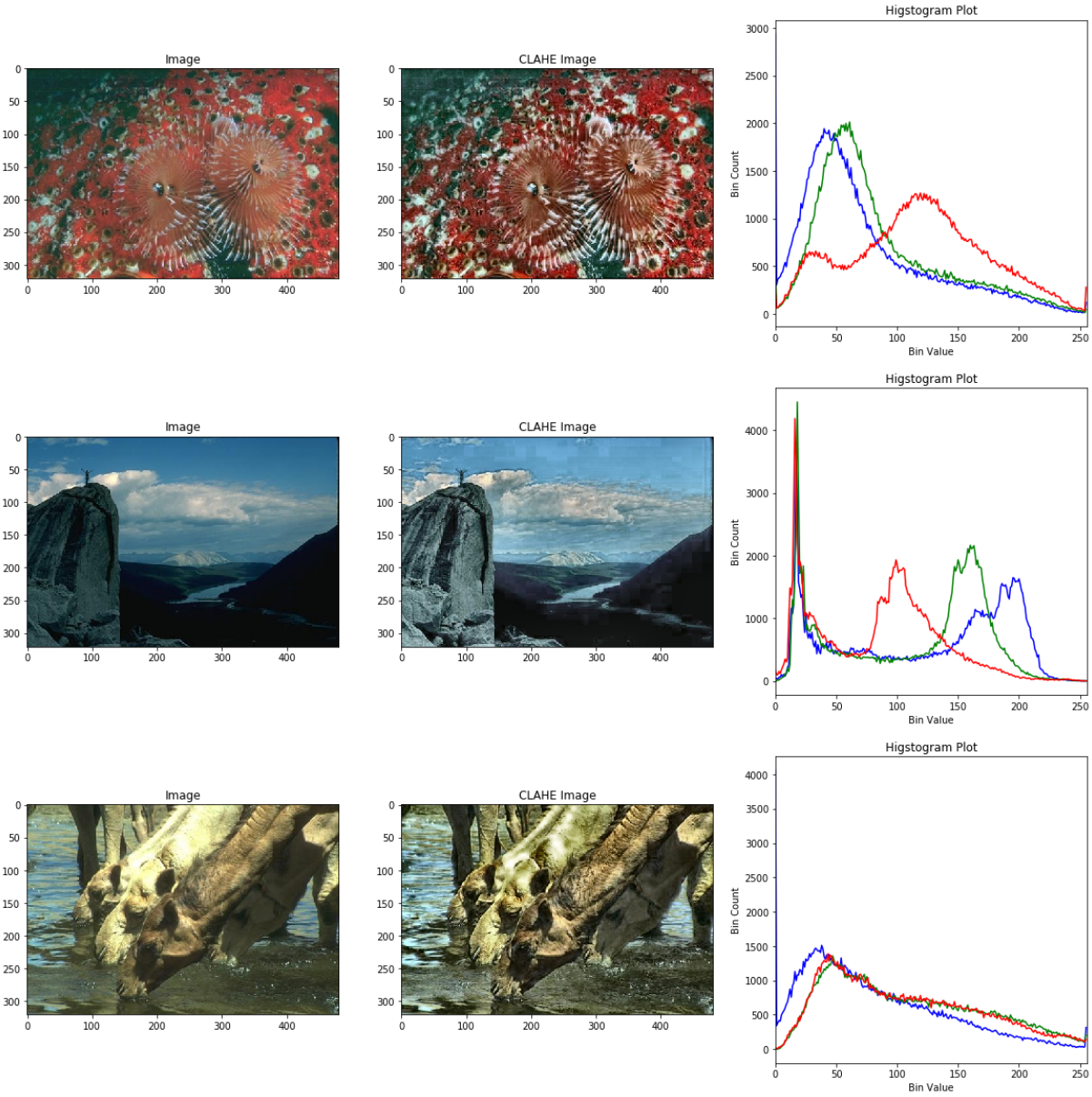

Figure 7: Adaptive histogram equalization(CLAHE). Left:Original image. Middle: Adaptive Histogram equalized image. Right: Histogram plot of adaptive histogram equalized image

2.4 Observation

In Figure 5, we have the input image on the left, and the right image shows its equivalent histogram. It can be observed that the pixels are concentrated along particular pixel intensities in RGB, lacking contrast.

In Figure 6 we perform global histogram equalization, where the pixels are distributed along all the possible intensities evenly. Third column shows the histogram plot for each of the global histogram equalized images. For the equalization, we first convert the image from RGB to YUV format, and equalization is performed along intensity(Y) channel.

In Figure 7 we perform contrast limited adaptive histogram equalization, where the pixels are contrasted with its local neighbors and accordingly normalized in the range (0,255). Local redistribution of pixel intensity increase image contrast and better defines edges in the image. For this part of the question also, we convert the image from RGB to YUV format, and perform clahe on Y-intensity channel. The hyper-parameters are selected as, clipLimit=2.0, tileGridSize=(8,8). We use the opencv inbuilt function for clahe operation.

3 Edge Detection

3.1 Sobel operator

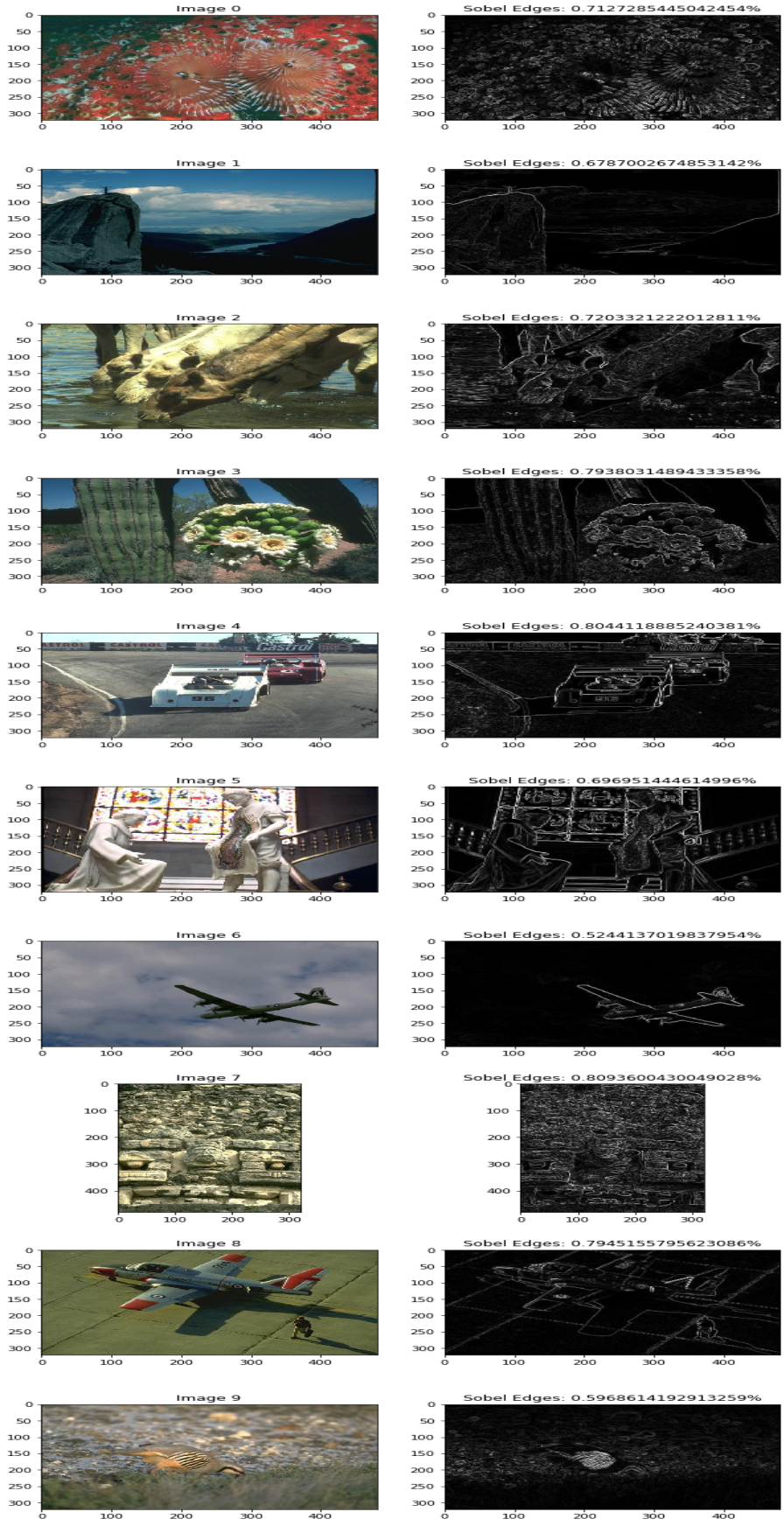

Figure 8: Edge Detection using Sobel operation. Left: Original image. Right:Detected edges

3.2 Canny edge detection

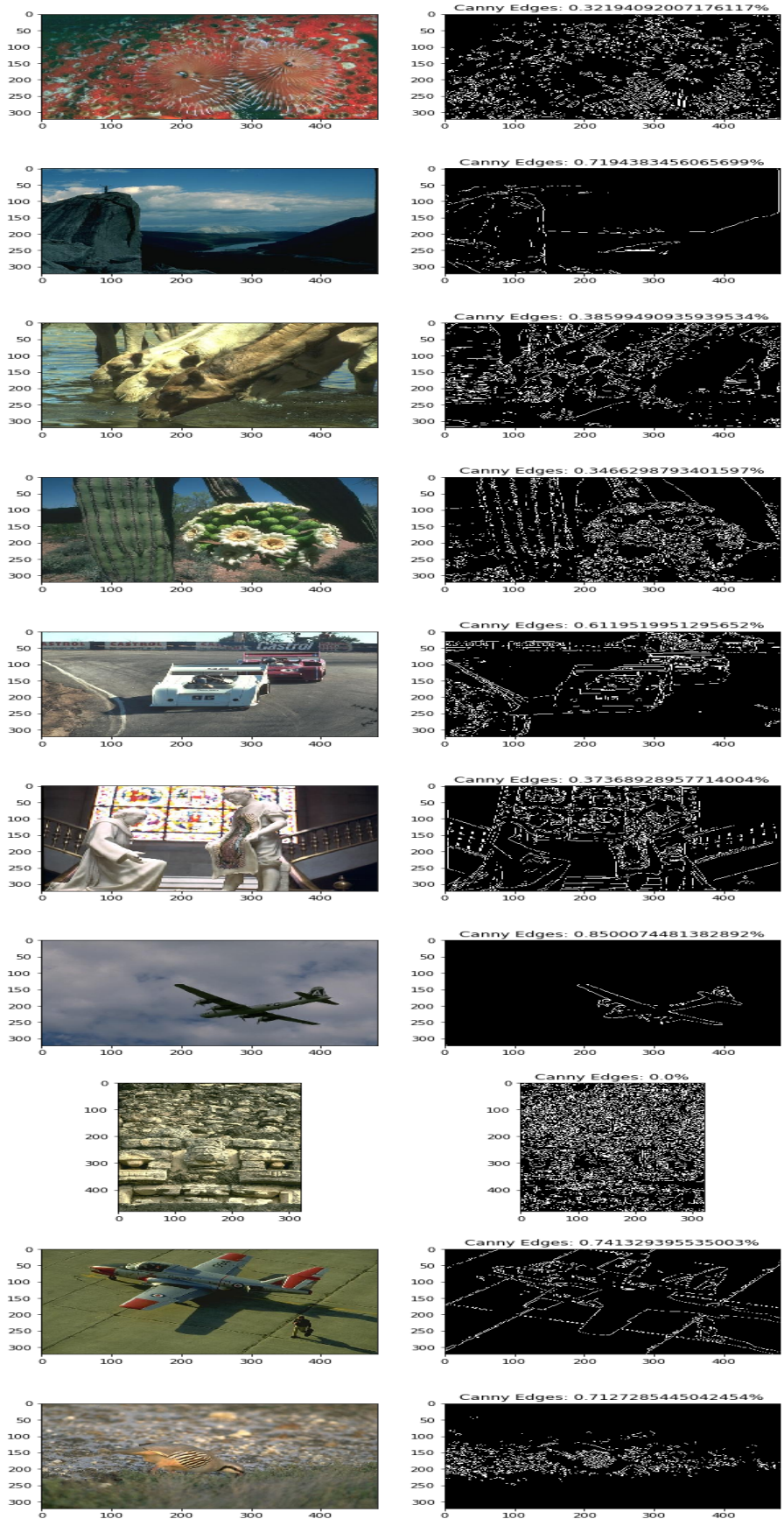

Figure 9: Edge Detection using Canny edge detector. Left: Original image. Right:Detected edges

3.3 Laplacian edge detection

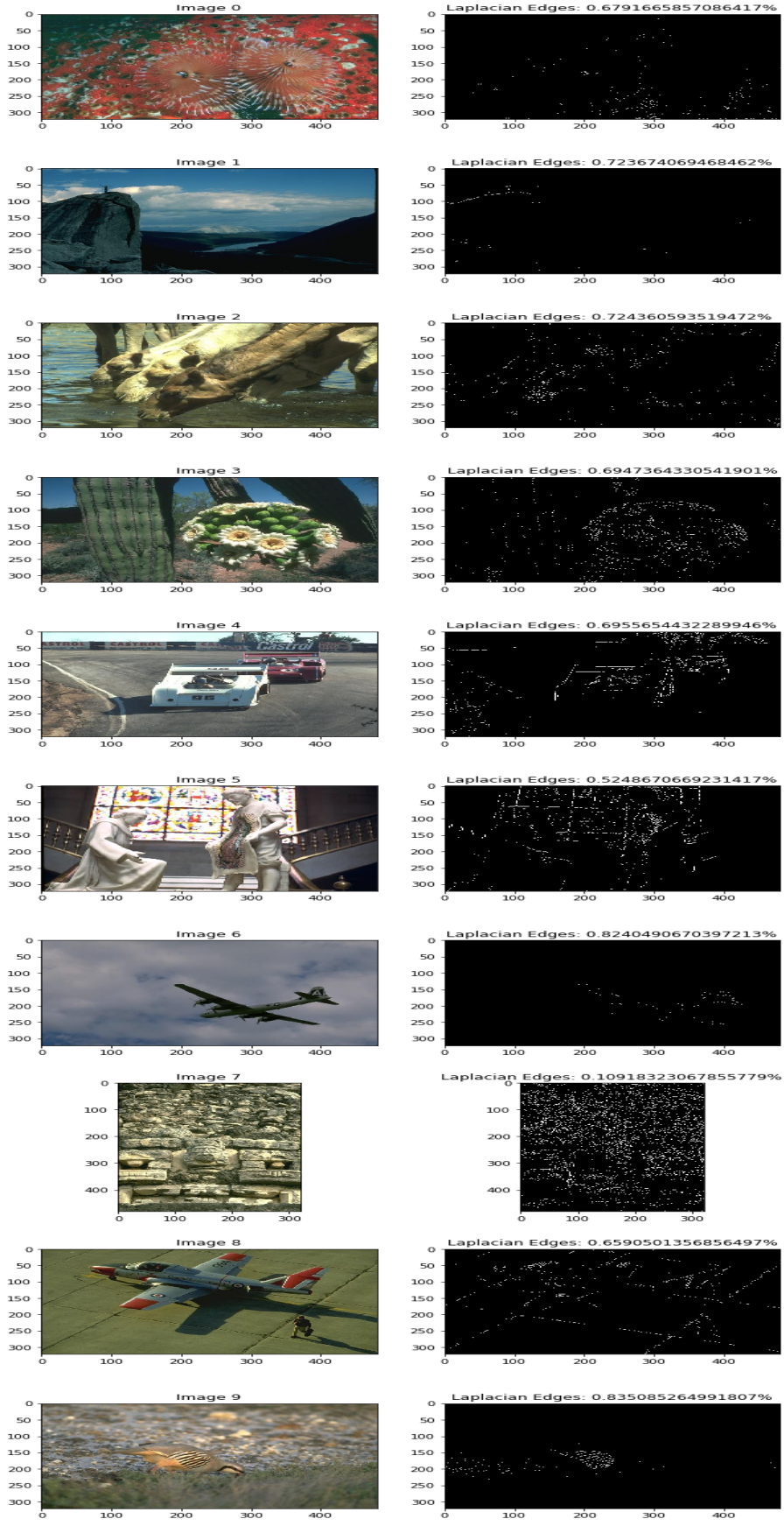

Figure 10: Edge Detection using Laplacian edge detector. Left: Original image. Right:Detected edges

3.4 Comparison of different edge detection techniques

In this part of assignment, we have performed edge detection of several images using different edge detection techniques. Figure 8,9 & 10 shows edge detection with Sobel operator, canny edge detector and THIRD TECHNIQUE respectively. Below we give a quantitative comparison of these techniques when implemented on images.

3.4.1 Accuracy of different edge detection techniques

SrNo	FileName	Sobel	Canny	Laplacian
1.	Image: 12084.jpg	0.68	0.32	0.68
2.	Image: 14037.jpg	0.72	0.72	0.72
3.	Image: 16077.jpg	0.79	0.39	0.72
4.	Image: 19021.jpg	0.80	0.35	0.69
5.	Image: 21077.jpg	0.70	0.61	0.70
6.	Image: 24077.jpg	0.52	0.37	0.52
7.	Image: 3096.jpg	0.81	0.85	0.82
8.	Image: 33039.jpg	0.79	0.00	0.11
9.	Image: 37073.jpg	0.60	0.74	0.66
10.	Image: 8023.jpg	0.86	0.71	0.84

From the quantitative metrics, it appears that Sobel operator and Laplacian operator gives better edge detection accuracy over Canny edge detector. However this argument is very subjective, since Canny needs a number of hyperparameters like lower and upper threshold bounds, sigma for the gaussian smoothening, etc. Fine tuning these parameters can boost Canny edge detection accuracies. However, for the images we use in this assignment, Laplacian and Sobel outperform Canny edge detector