CS484-001
Introduction to Computer Vision
Spring 2021–2022
Assignment - 1

Mohammed S. Yaseen - 21801331

For this assignment I've used NumPy for the calculations and matrix operations, openCV to read the images, and matplotlib to view the images and plot the charts.

1. Dilation and Erosion

(i) Dilation was implemented by iterating over each pixel of the input and updating it using a 5X5 kernel.

(ii) Erosion was done in a similar manner, the only difference is that in dilation we activate all the pixels that are in the moving kernel window if any of them is active, but in erosion, we don't activate the central pixel unless all pixels in the moving window are active.

Following are the results of applying dilation and erosion to a picture of a license plate.

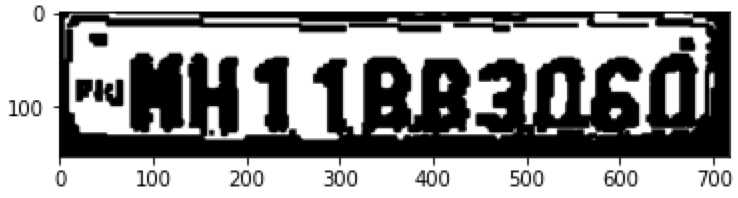

Figure 1: The image after dilation.

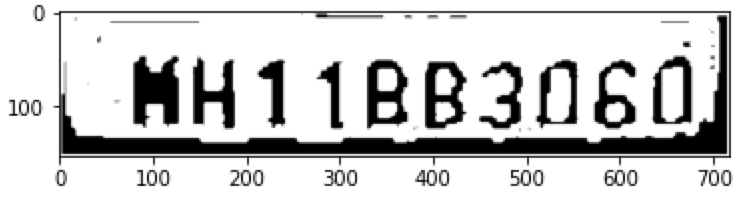

Figure 2: The image after erosion.

2. Creating a Histogram from Image

For this, I’ve flattened the image, which comes as a 2-dimensional array, into a one-dimensional array using NumPy's function *flatten()* then used its other function *histogram* that generates a histogram from an array of values. I’ve configured it to have 256 bins, equal to the number of possible integer values for the pixel intensity. I finally used the result to plot the histogram using matplotlib.pyplt. Following are two images with their histograms next to them.

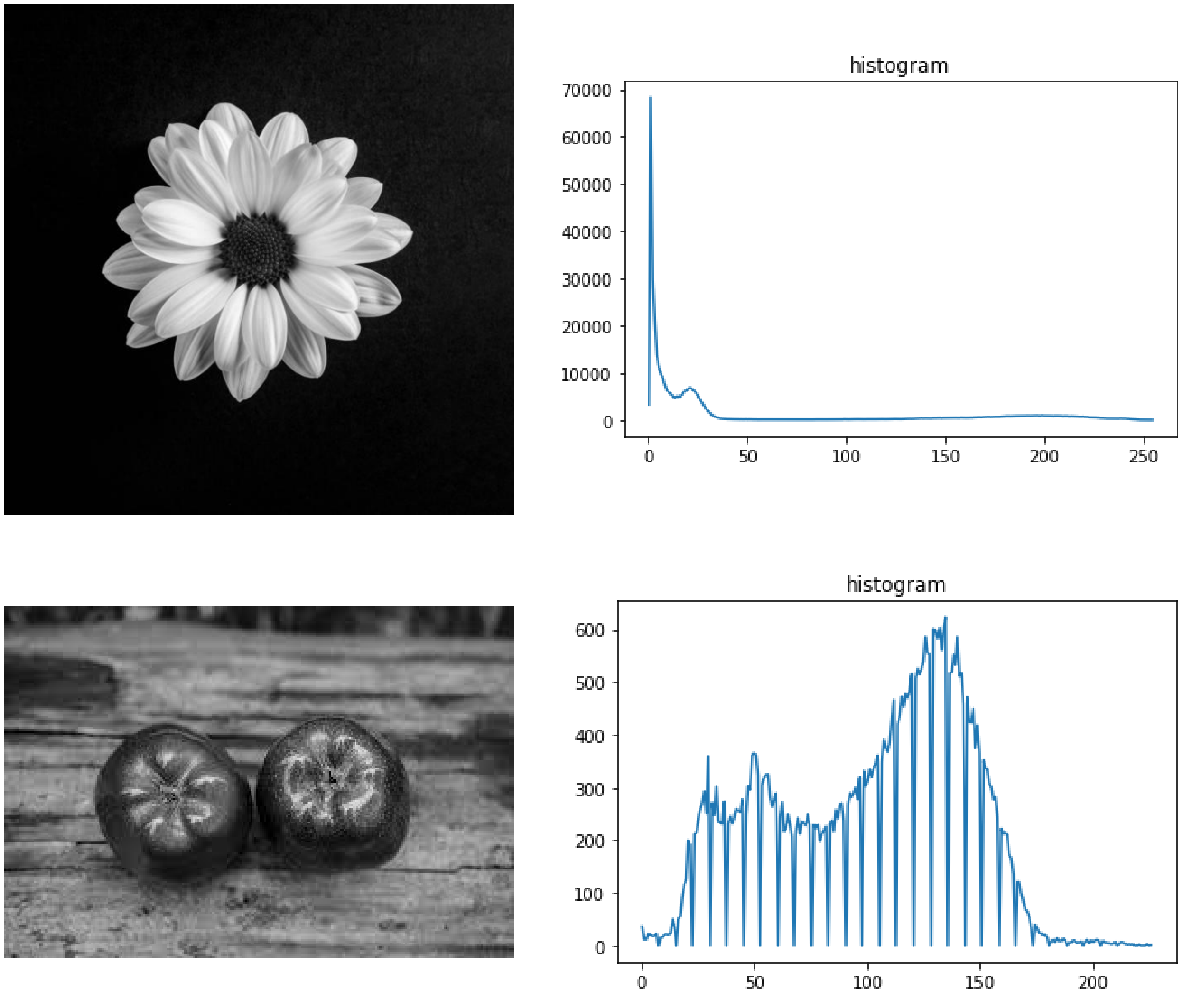

Figure 3: Images with their histograms.

3. Thresholding with Otsu's method

For this part, Otsu's method is used to separate the foreground from the background of the given images. Otsu's algorithm is used to find some intensity value called a threshold, then using that threshold, the image pixel intensities can be filtered, where the ones below the threshold will be zeroed and the ones above will have a max intensity. This threshold is found by minimizing the intra-class intensity variance. Following are the results of applying this method to two images.

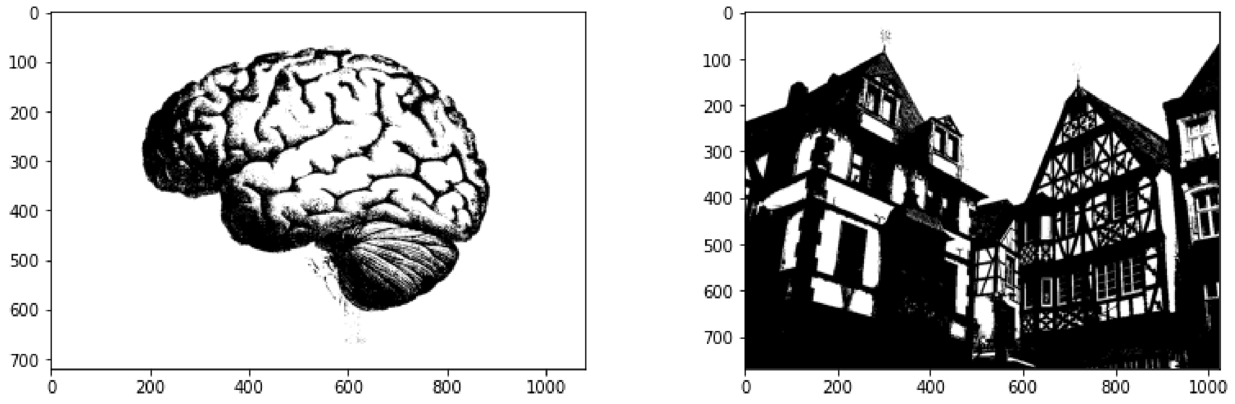

Figure 4: Thresholding using Otsu's method.

4. Edge detection using Sobel kernel

Here, an edge would be detected by convolving the Sobel kernel, which is a 3x3 kernel, with the image. For that reason, I've implemented a convolution function, which is basically a function that updates the value of each pixel of the image using a weighted sum of the pixels in the kernel window. The kernel window would be centered on the pixel being updated. Following is the result of applying the Sobel filter on an image.

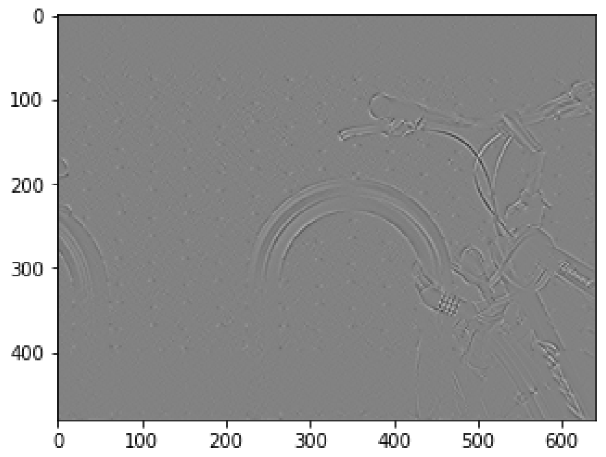

Figure 4: Edge detection using the Sobel filter.