CS 484: Introduction to Computer Vision

Spring 2024

Homework 1

Modules/Libraries Used

Numpy is used for mathematical operations and working with matrices. **Matplotlib** is used for reading, writing, displaying images and plotting. **OpenCV** is used for connected component analysis and labeling these components in the image.

Question 1

Implementation of dilation and erosion require convolution operations to be performed between the structuring element and the image. In my code, this is accomplished by looping over each pixel, then looping over each overlapping pixel when the center of the structuring element is placed on the pixel represented in the outer loop. Much more computationally efficient methods are possible.

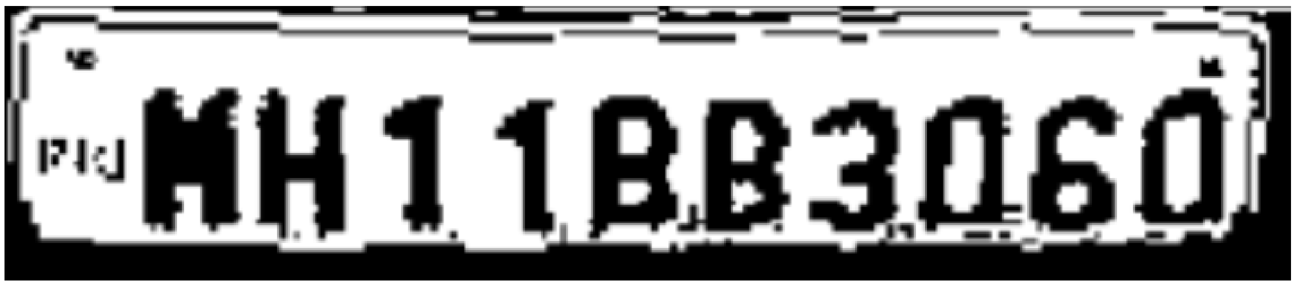


Image 1: Image Q1

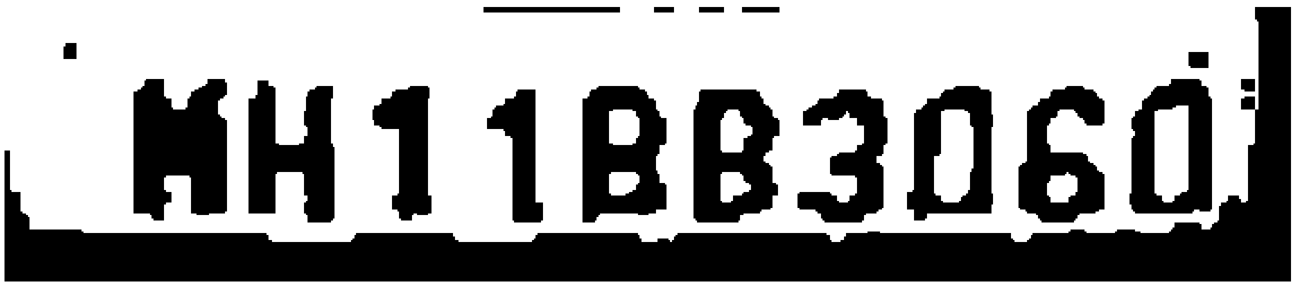


Image 2: After closing operation by 6x6 structuring element

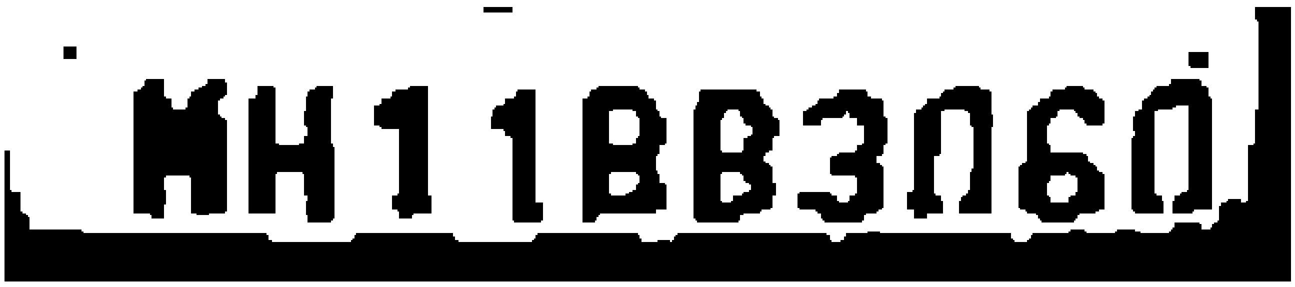


Image 3: After closing operation by 7x7 structuring element

The 6x6 structuring element protects the integrity of the characters while the 7x7 structuring element gets rid of more noise and imperfections.

Question 2

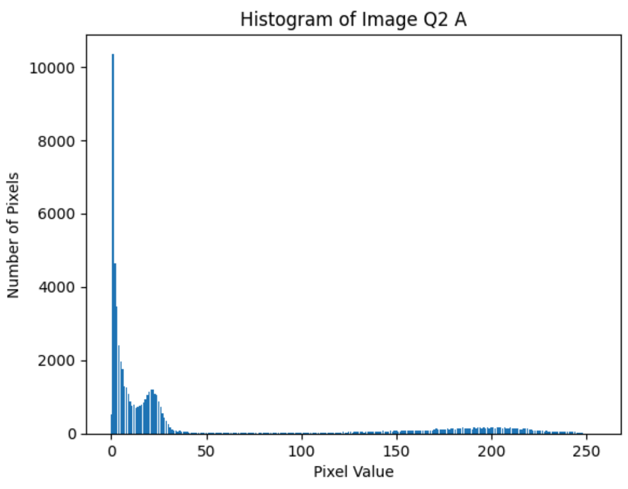
When working with images in this homework, some important points to keep in mind are:

- *Grayscale or RGB*: When read by matplotlib, some images have 3 channels while others have 1. The images that have 3 channels must be converted to 1 channel grayscale.
- *Float or 8-bit*: When read by matplotlib, ".png" images have float pixel values between 0 and 1 while all other formats have 8 bit unsigned integer values between 0 and 255. I converted all ".png" images to 8 bit unsigned integer values before any operations.

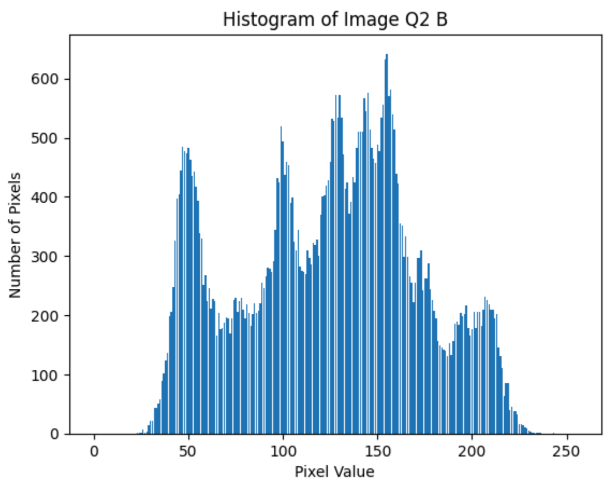
Histograms are calculated by looping over each pixel and tallying up pixel values.

Image 4: Image Q2 A

Image 5: Image Q2 B



Plot 1: Histogram of image Q2 A

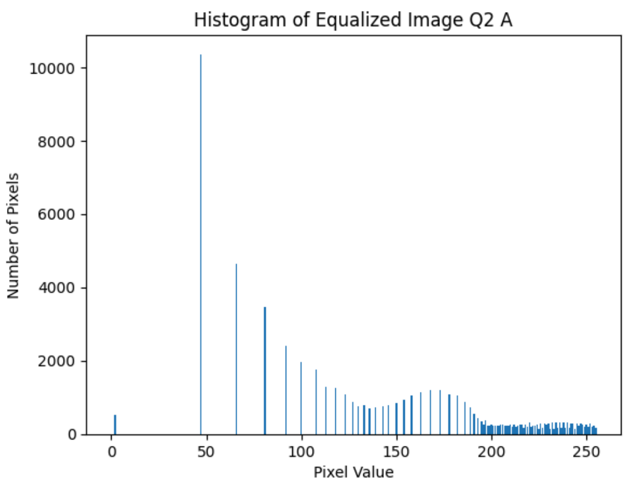


Plot 2: Histogram of image Q2 B

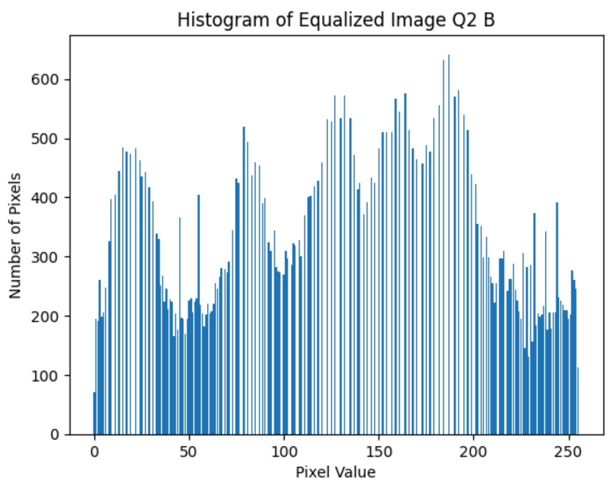
Image Q2 A contains many dark pixels while the pixel values of image Q2 B are more uniformly distributed.

Question 3

Histogram equalization is achieved by first calculating the normalized cumulative density function (CDF) of the image, then interpolating the values of the original image using the normalized CDF.



Plot 3: Histogram of image equalized Q2 A



Plot 4: Histogram of image equalized Q2 B

In the equalized images histogram, the pixel values containing more pixels are pushed apart and vice versa. The CDF of the resulting image looks like a ramp with slope 1.

My implementation of histogram equalization **does not** yield an exactly uniform histogram. This is because images are composed of **discrete** pixels. If the image were composed of infinitely many points and my computer was able to process continuous data, the histogram would be exactly uniform.

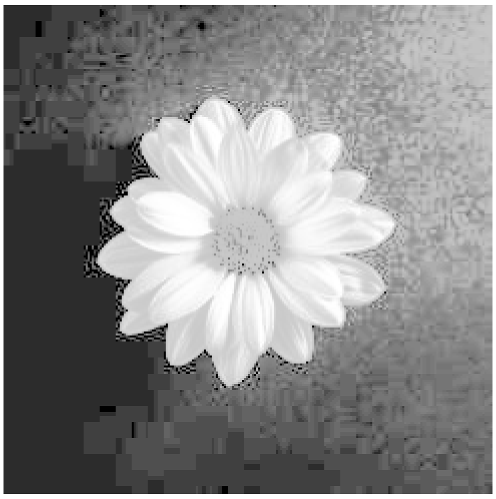


Image 6: Equalized Image Q2 A

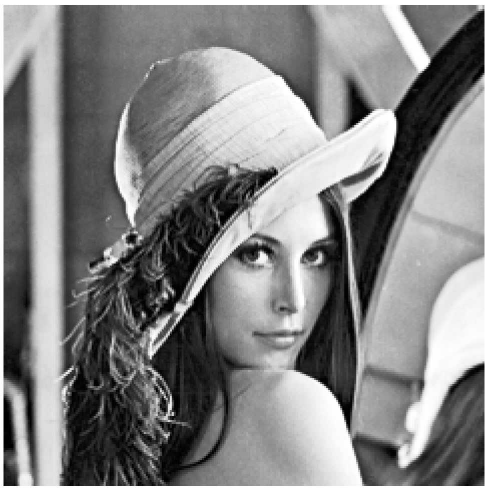


Image 7: Equalized Image Q2 B

Question 4

Otsu's method assumes that the histogram of an image has a bimodal distribution and tries to find the best pixel value to put a threshold separating the white pixels from the black pixels in the binary image to be obtained after the operation. This is accomplished by minimizing the weighted sum of within group variances after separating pixels according to the threshold.

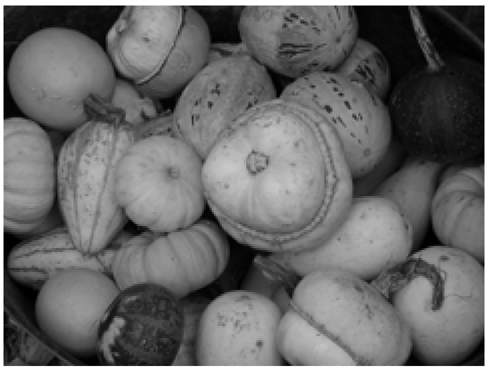


Image 8: Image Q4 A

Image 9: Otsu thresholded image Q4 A

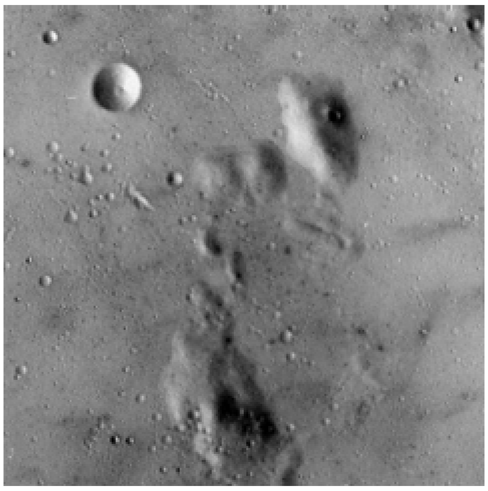


Image 10: Image Q4 B

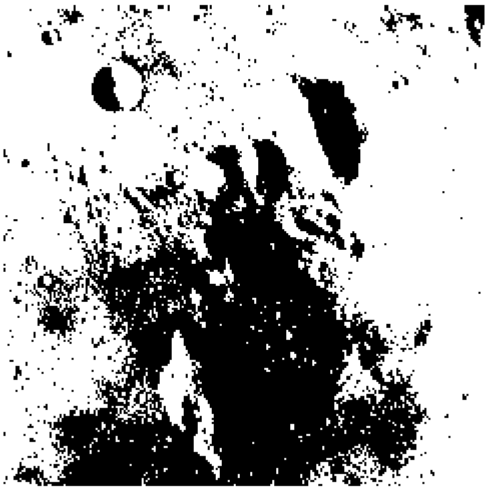
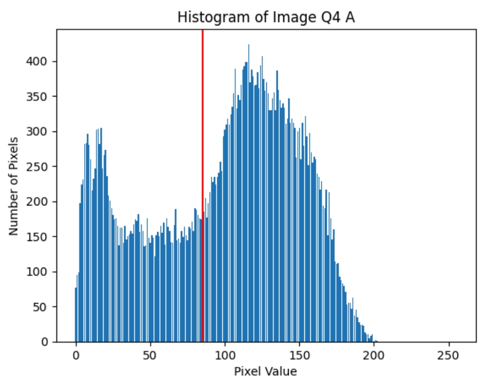
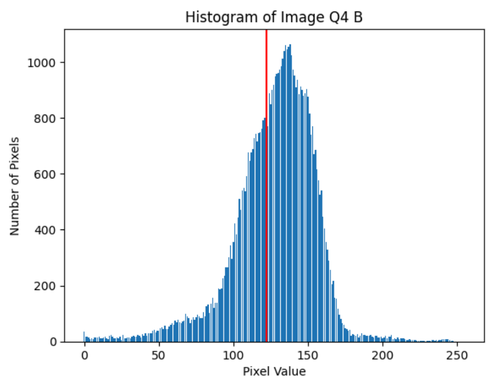


Image 11: Otsu thresholded image Q4 B

As the Otsu's method requires an image with a histogram that has a **bimodal distribution**, the results are not always perfect. As can be seen from the following plots, in this case, image Q4 A has a somewhat bimodal distribution while image Q4 B does not.



Plot 5: Histogram of image Q4 A and the threshold (red line)



Plot 6: Histogram of image Q4 B and the threshold (red line)

Question 5

In the **thresholding** step, I chose to use the **Otsu's method** because of two reasons. Firstly, the histogram of the image is mostly bimodal. Secondly, the method is adaptable to different images, and it does not require fine tuning. These properties make it easier to use for actual scenarios. This step **separates** the background from the foreground.

In the **morphological operations** step, the list of possible operations are (assuming objects are white and background is black):

- *Dilation*: Makes objects bolder.
- *Erosion*: Makes objects thinner.
- *Opening*: Erosion + Dilation. Gets rid of noise and thin features.
- *Closing*: Dilation + Erosion. Fills gaps in objects.

I first performed closing to fill the gaps in the objects, then performed erosion with a 3x3 structuring element 7 times to separate the objects. Performing erosion multiple times with a smaller structuring element prevents object deletion and gets rid of connections better than a single time with a larger structuring element. I used a square structuring element instead of a circle because it is easier to implement and it gets acceptable results.

In the **labeling** step, I used OpenCV to conduct connected component analysis and label each object.

As a result, I counted **38** objects in the image.

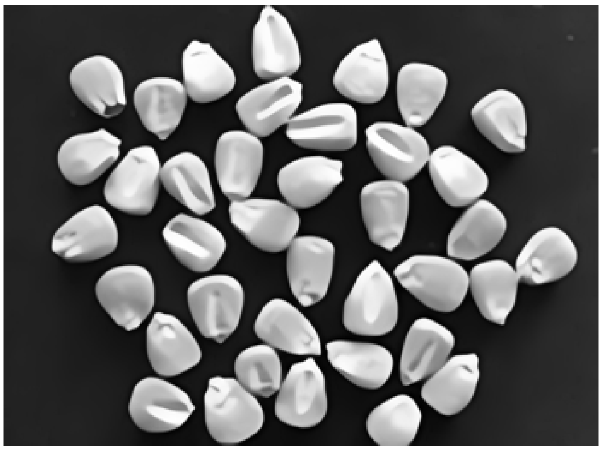


Image 12: Image Q5

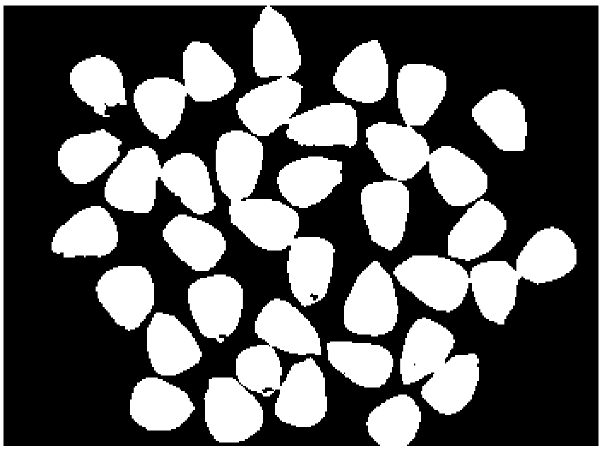


Image 13: Otsu thresholded image Q5

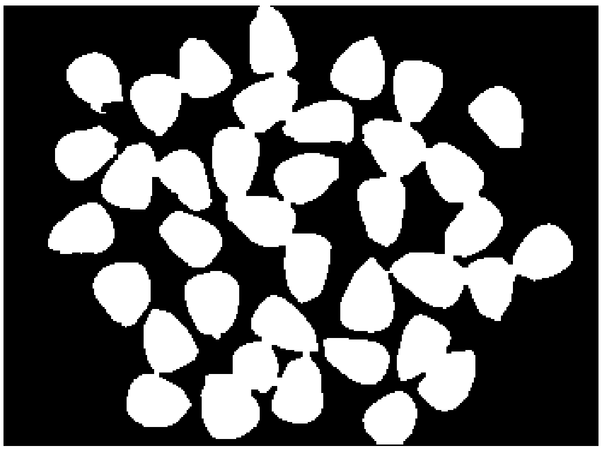


Image 14: Image Q5 after closing

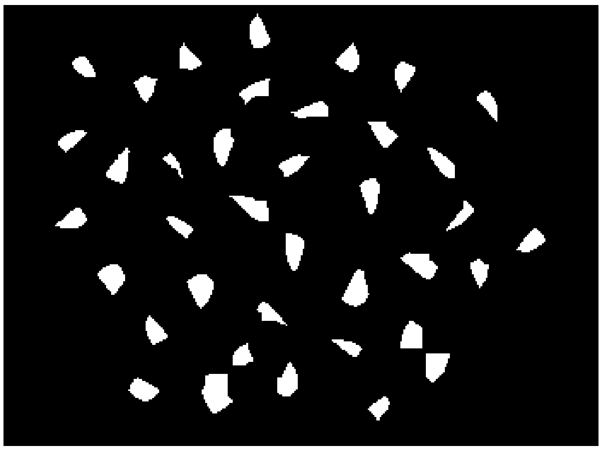


Image 15: Image Q5 after erosions

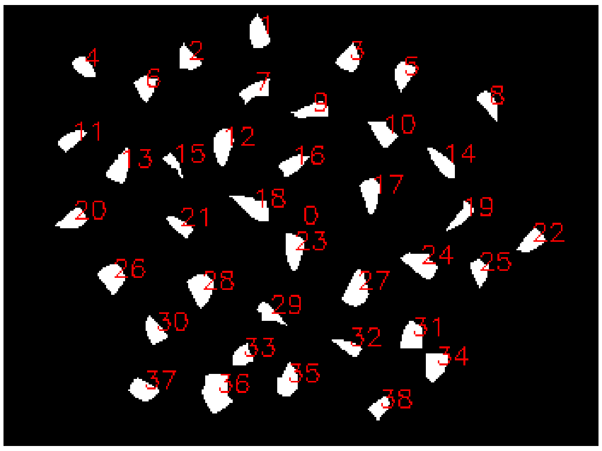


Image 16: Labeled components of image Q5

Question 6

Sobel and prewitt operators are both used to compute the gradient in the x and y directions in an image. While they are very similar, the Sobel filter gives more importance to the pixels in line with the center pixel compared to the diagonal pixels.

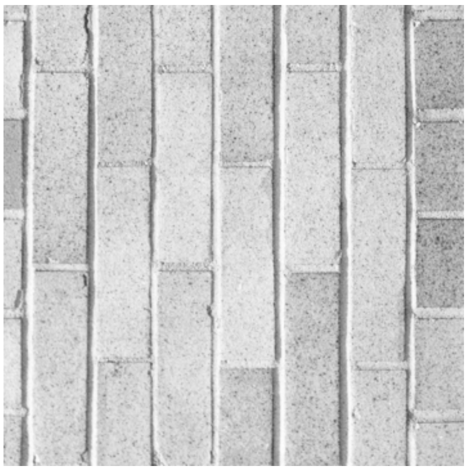


Image 17: Image Q6

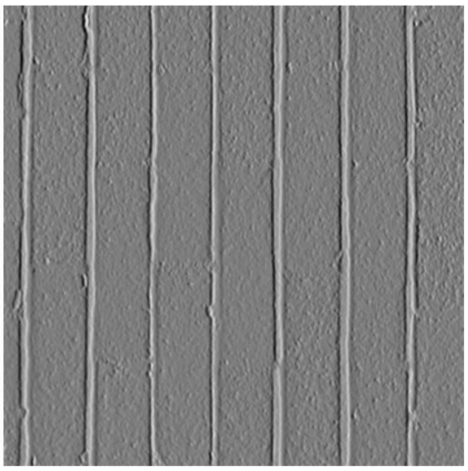


Image 18: Image Q6 after convolution with the Sobel operator in the x direction

Image 19: Image Q6 after convolution with the Sobel operator in the y direction

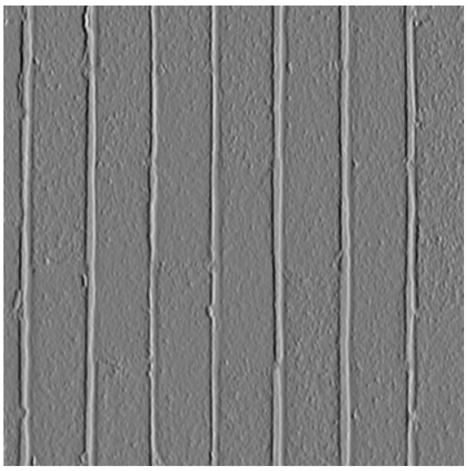


Image 20: Image Q6 after convolution with the Prewitt operator in the x direction

Image 21: Image Q6 after convolution with the Prewitt operator in the y direction

The Sobel operator seems to show the little cracks more than the Prewitt operator.