Jason Kayser
Digital Image Processing 4650
Jpk2f2

Assignment 1

For assignment 1 I utilized Python 3.7 alongside opencv2 and numpy in order to read images and perform histogram equalization on them. Additionally, I used pyplot for more attractive and readable histogram plots for verifying my algorithm was performing as expected. After performing and applying my histogram equalization each image was then stacked beside the original as well as a perfect example of histogram equalization performed by cv2.equalizeHist().

For each image I began by reading in two copies, one standard and one in grayscale for use with cv2.equalizeHist() for comparison. For example, with image 309:

```
image_b = cv2.imread('resources/Fig0309(a)(washed_out_aerial_image).tif')
image_b2 = cv2.imread('resources/Fig0309(a)(washed_out_aerial_image).tif', 0)
```

All remaining image processing code was placed in a function in order to remove the necessity of repeat code, and so this function was then called on each image. This function reads in two image objects: the standard for my algorithm, and the grayscale for the comparison image. It then resizes both images in order to shrink them for later concatenation and display using the cv2.resize() function:

```
image = cv2.resize(im1, (0, 0), None, .70, .70)
image2 = cv2.resize(im2, (0, 0), None, .70, .70)
```

Next, the type of the grayscale image is corrected to uint8 and is used to create the comparison image:

```
image2 = image2.astype(numpy.uint8)
equ = cv2.equalizeHist(image2)
```

The example is now stored in equ and is left along until displaying at the end. Following this, I flattened the image down into a 1D array using numpy.flatten(). I then displayed the histogram using pyplot.hist() in order to compare the image's prior values to the equalized ones:

```
flatimage = image.flatten()  # flatten to 1D
plot.hist(flatimage, bins=256)  # create histogram
plot.xlim(0, 256)  # prevent cutoff
plot.title('first')
plot.show()
```

This plot can be seen here:

Jason Kayser
Digital Image Processing 4650
Jpk2f2

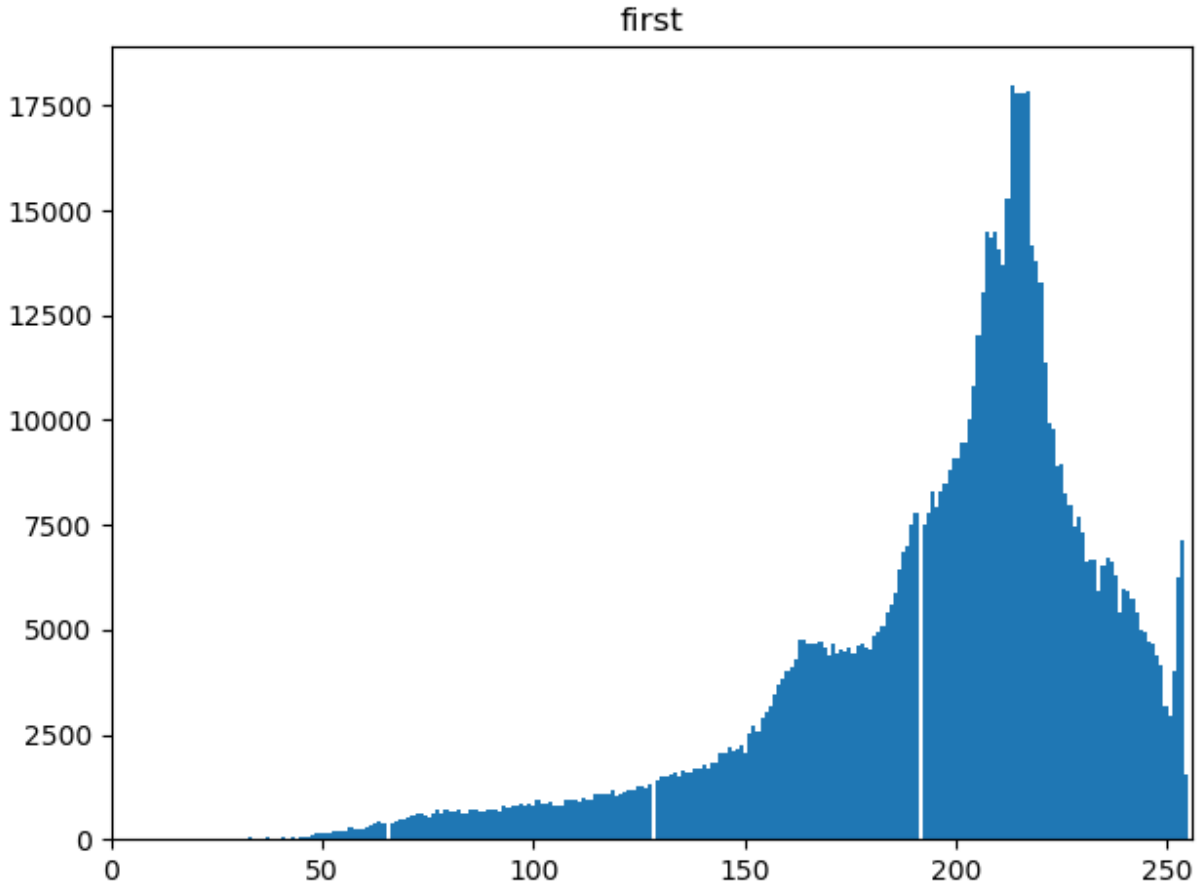

Figure 1 image 309 initial histogram

I then created and used my own histogram function to obtain an array containing the histogram values for the image. This array is then used to obtain the cumulative sum:

```
csum = 0  # holds current cumulative sum
cumusum = numpy.zeros(len(histogram))  # array for holding cumulative sum steps

# get cumulative sum
for i in range(0, len(histogram)):
    csum = csum + histogram[i]
    cumusum[i] = csum
```

I the plotted the cumulative sum:

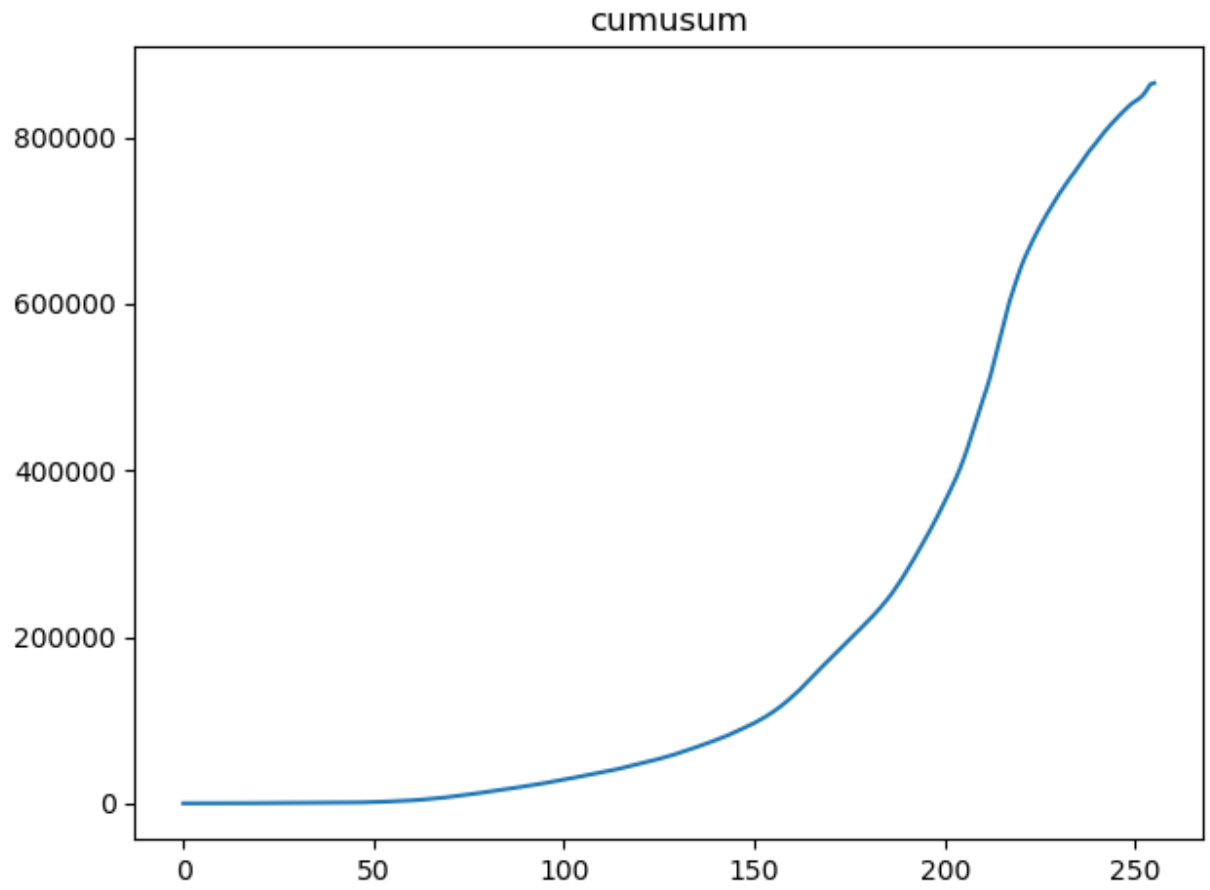

Figure 2 Image 309 Cumulative Sum

I then normalized the cumulative sum for the image. Here I ran into a number of issues, however I found the Wikipedia page on histogram equalization quite helpful, especially the part on implementation. The url is https://en.wikipedia.org/wiki/Histogram_equalization#Implementation . The logic can be seen here, followed by the plot:

```
num = (cumusum - cumusum.min()) * 255
denom = cumusum.max() - cumusum.min()
cumusum = num / denom
```

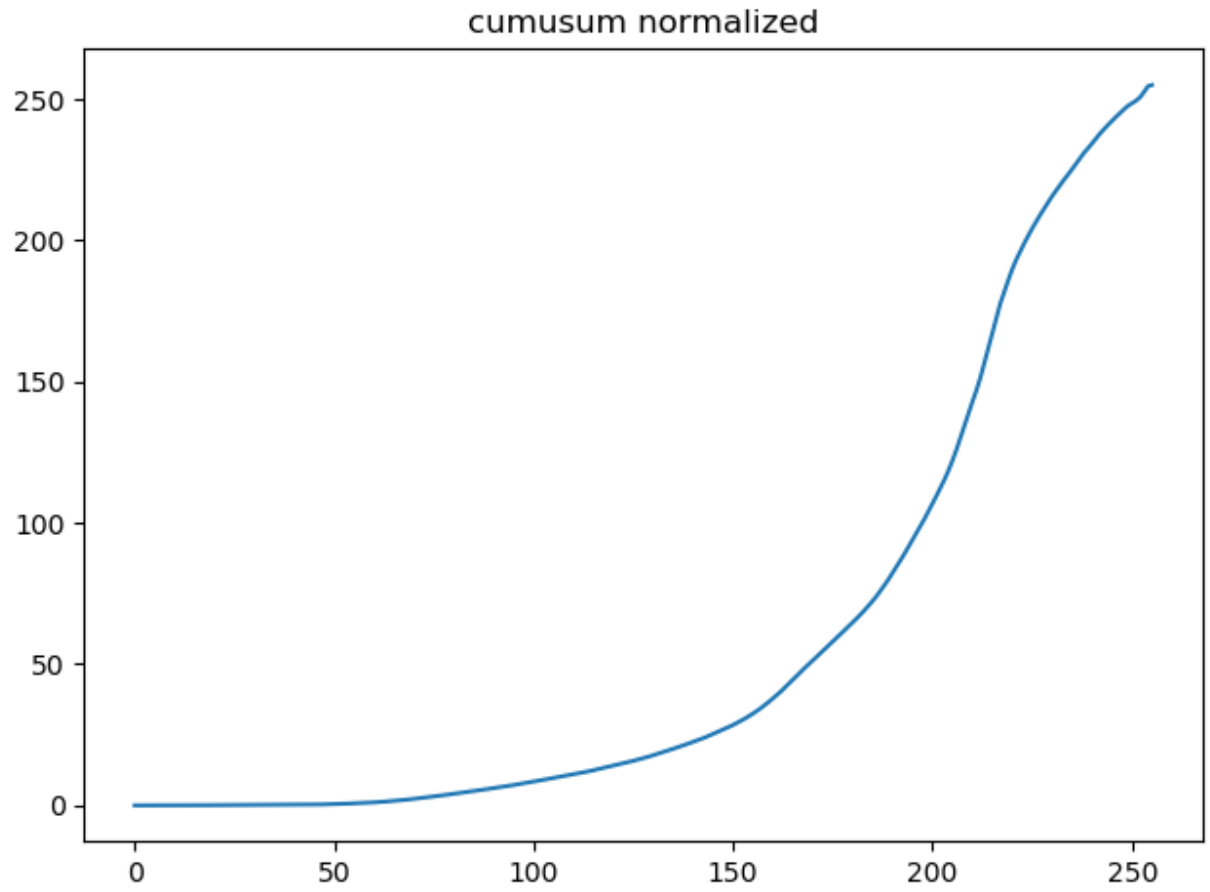

Figure 3 Image 309 Normalized Cumulative Sum

As can be seen in the image, the cumulative sum now appears to be normalized between 0 and 255. Next, I corrected the type of the cumulative sum before using it to create a new image and plotting its histogram:

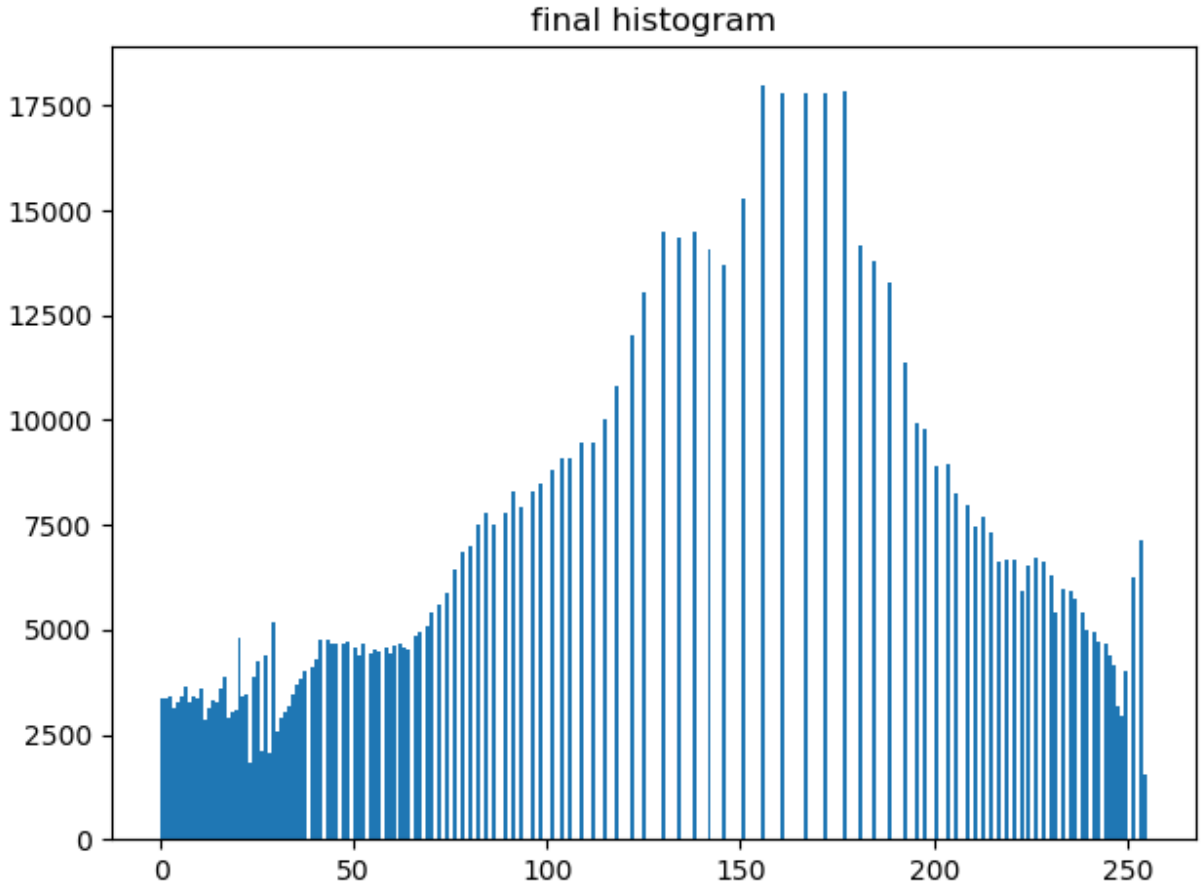

Figure 4 Image 309 Equalized Histogram

As can be seen by comparing this image to the earlier histogram, the values are far more spread out and the differences less drastic. Finally, I reshaped the new image back into proper dimensions using numpy.reshape():

img_new = numpy.reshape(img_new, image.shape)

Additionally, I altered the channels of the 'ideal' comparison image to match the original and my equalized image:

equ_3_channel = cv2.cvtColor(equ, cv2.COLOR_GRAY2BGR)

This allowed me to concatenate them for display:

```
res = numpy.hstack((image, equ_3_channel, img_new))  # concatenate images for display
cv2.imshow('test', res)  # show stacked images
cv2.waitKey(0)  # wait for button press to close window
```

Left to right, the images are the original, the ideal equalized image from cv2.histEqualize(), and my equalized image.

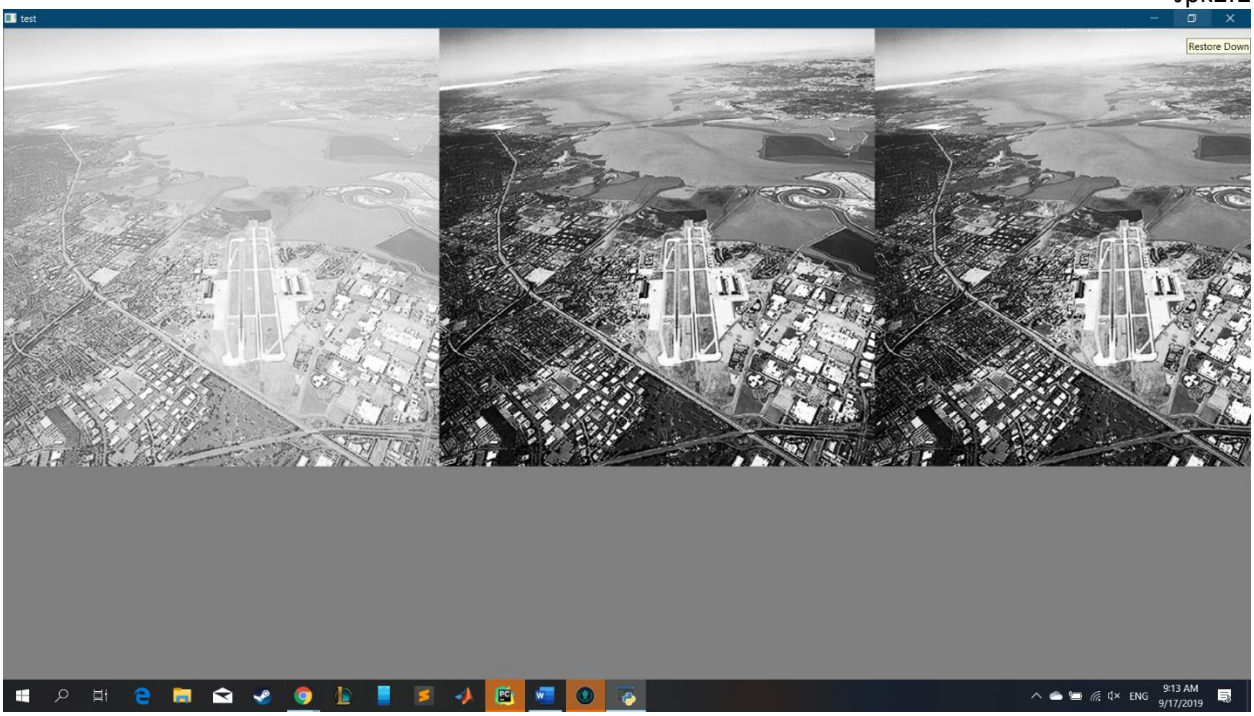

Figure 5 Image 309 left to right: original, equalizeHist() example, personal algorithm

Using the same code, the following results were achieved for image 308

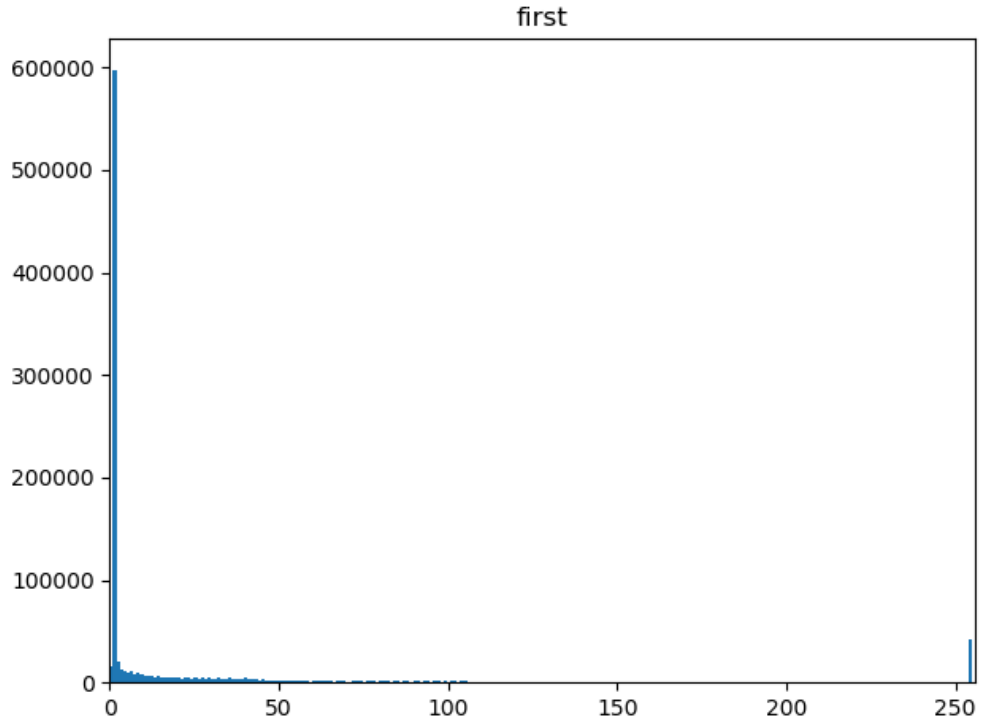

Figure 6 Image 308 Initial Histogram

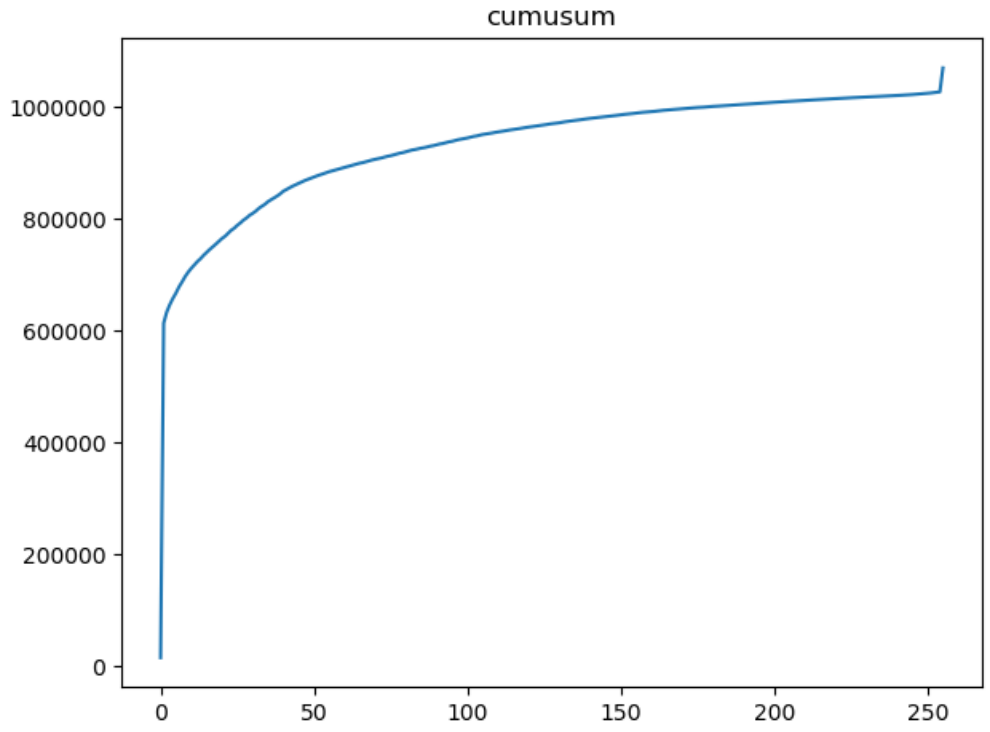

Figure 7 Image 308 Cumulative Sum

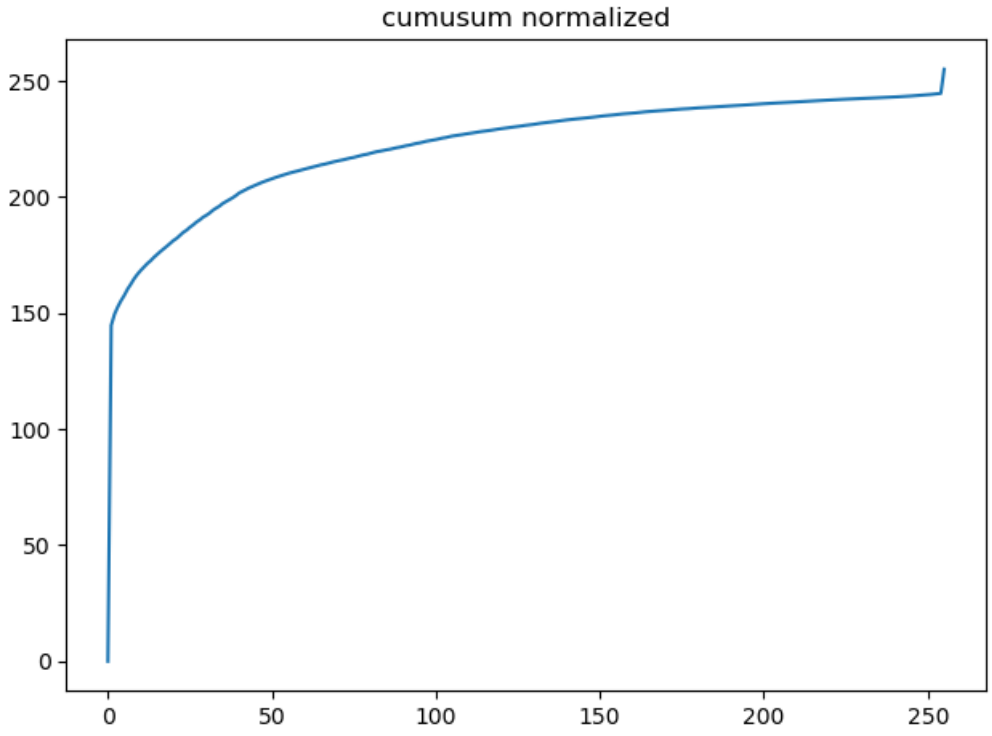

Figure 8 Image 308 Normalized Cumulative Sum

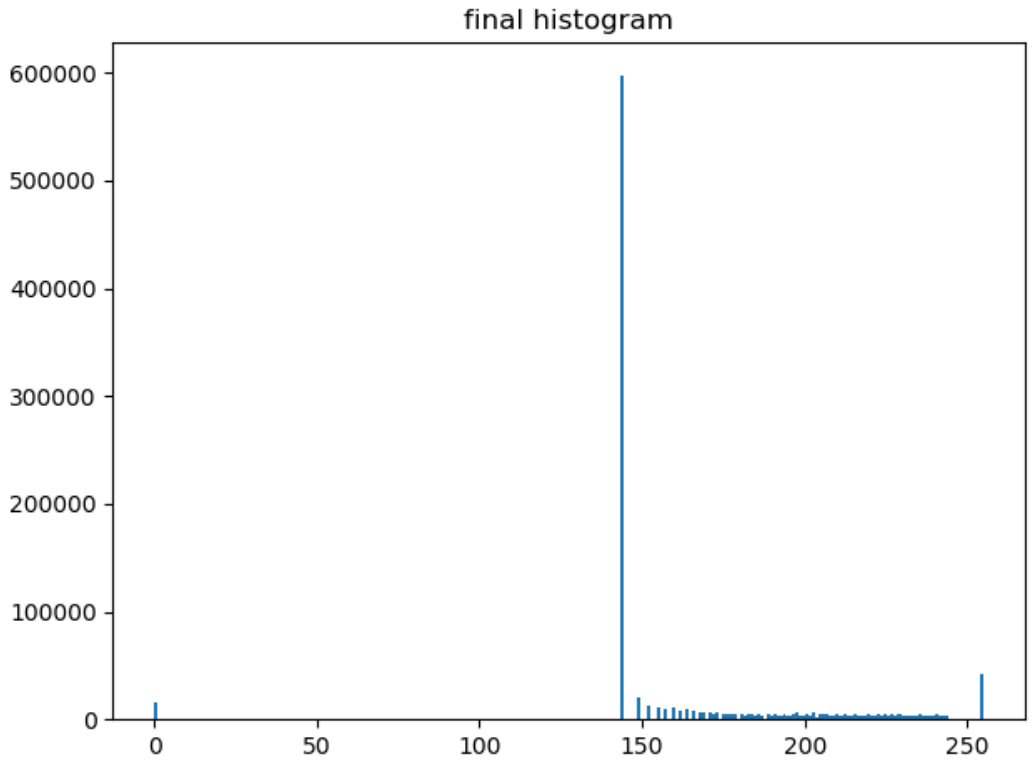

Figure 9 Image 308 Equalized Histogram

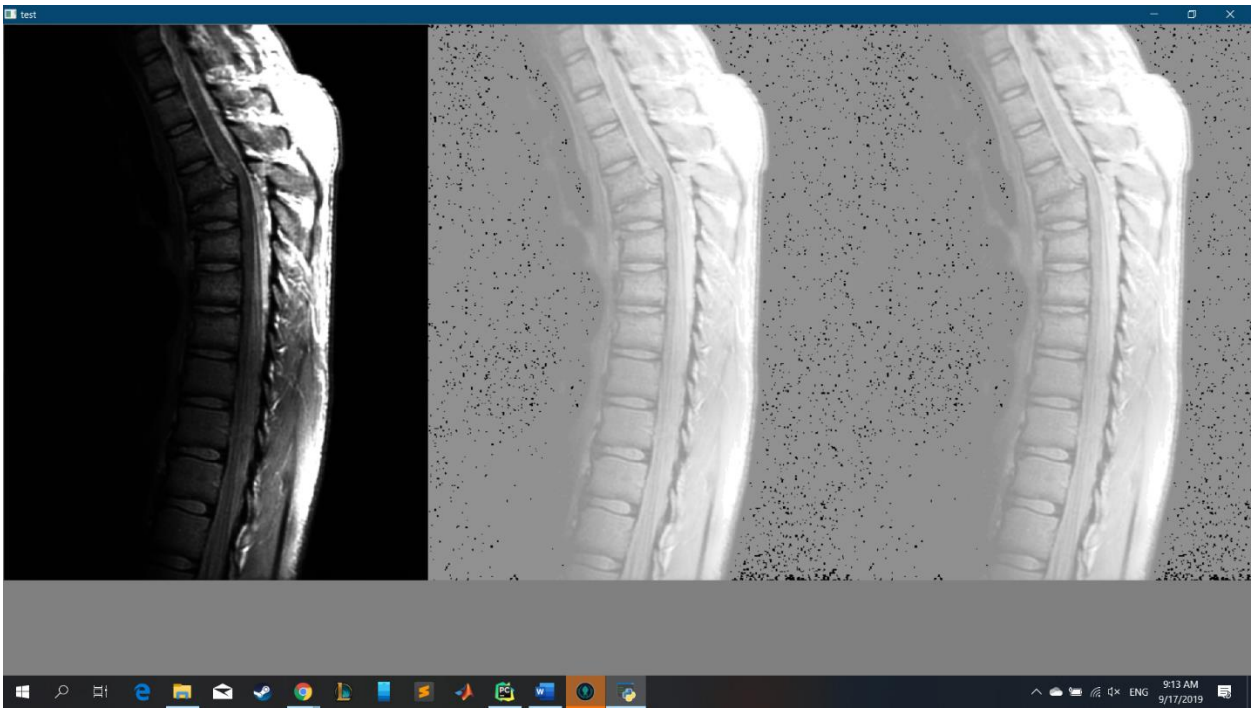

Figure 10 Image 308 left to right: original, cv2.equalizeHist(), personal algorithm

And for image 316(2):

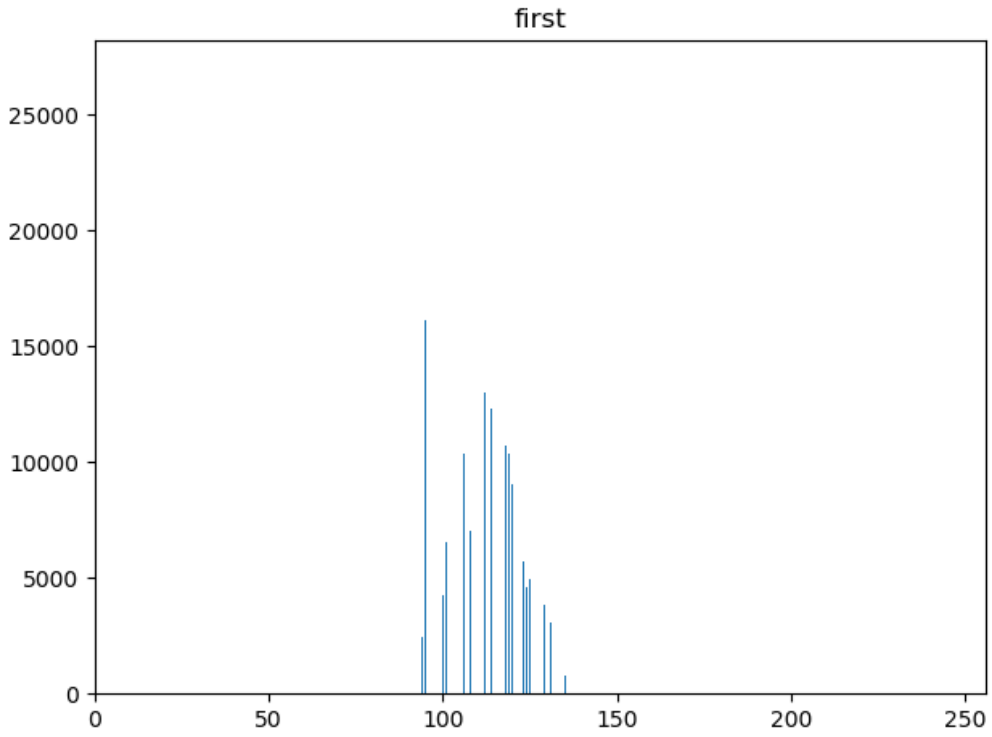

Figure 11 image 316 Initial Histogram

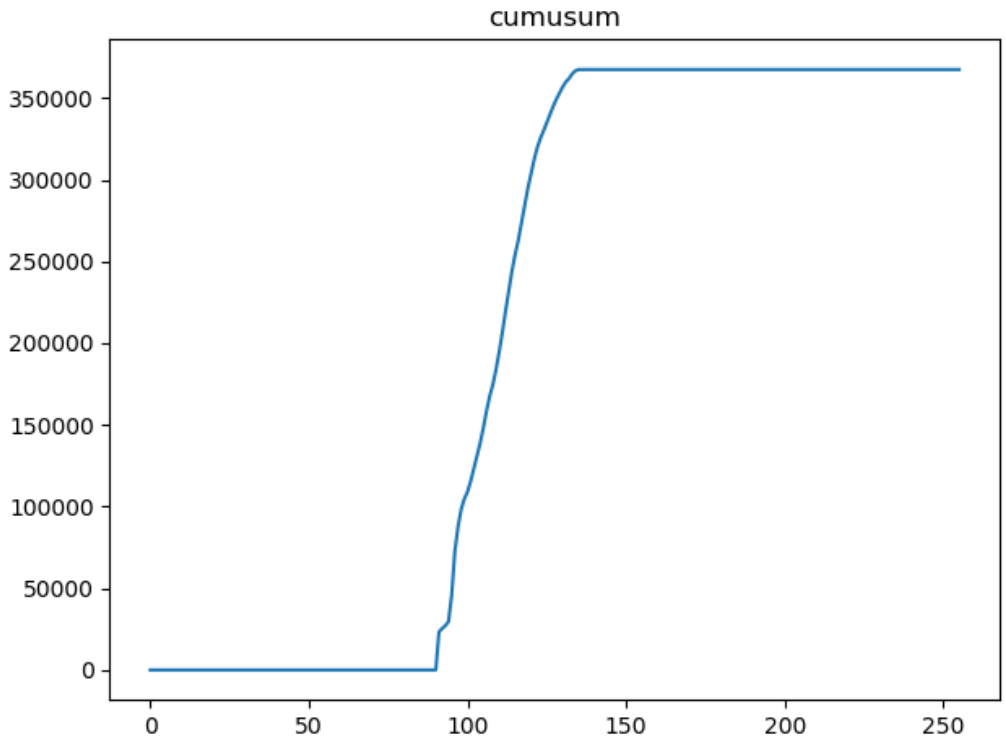

Figure 12 Image 316 Cumulative Sum

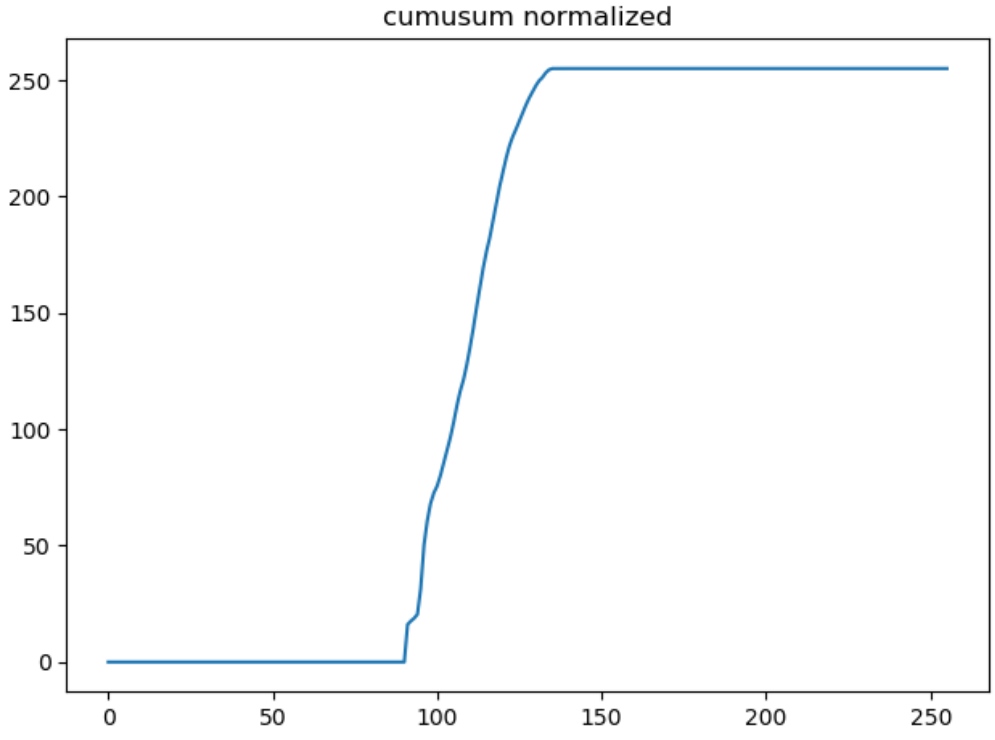

Figure 13 Image 316 Normalized Cumulative Sum

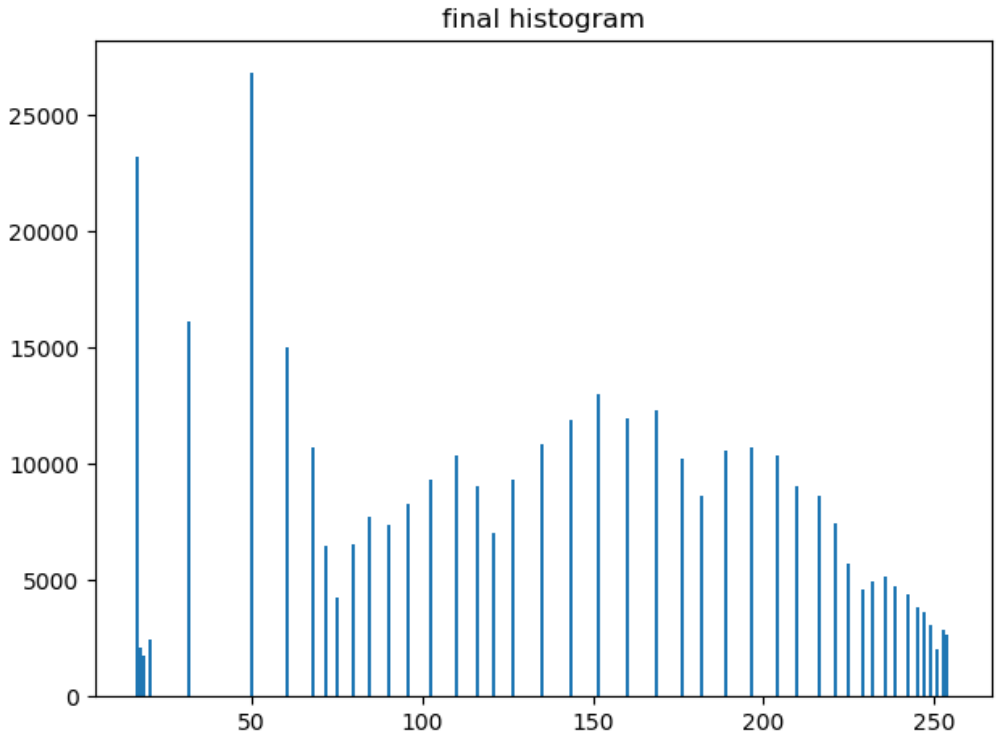

Figure 14 Image 316 Equalized Histogram

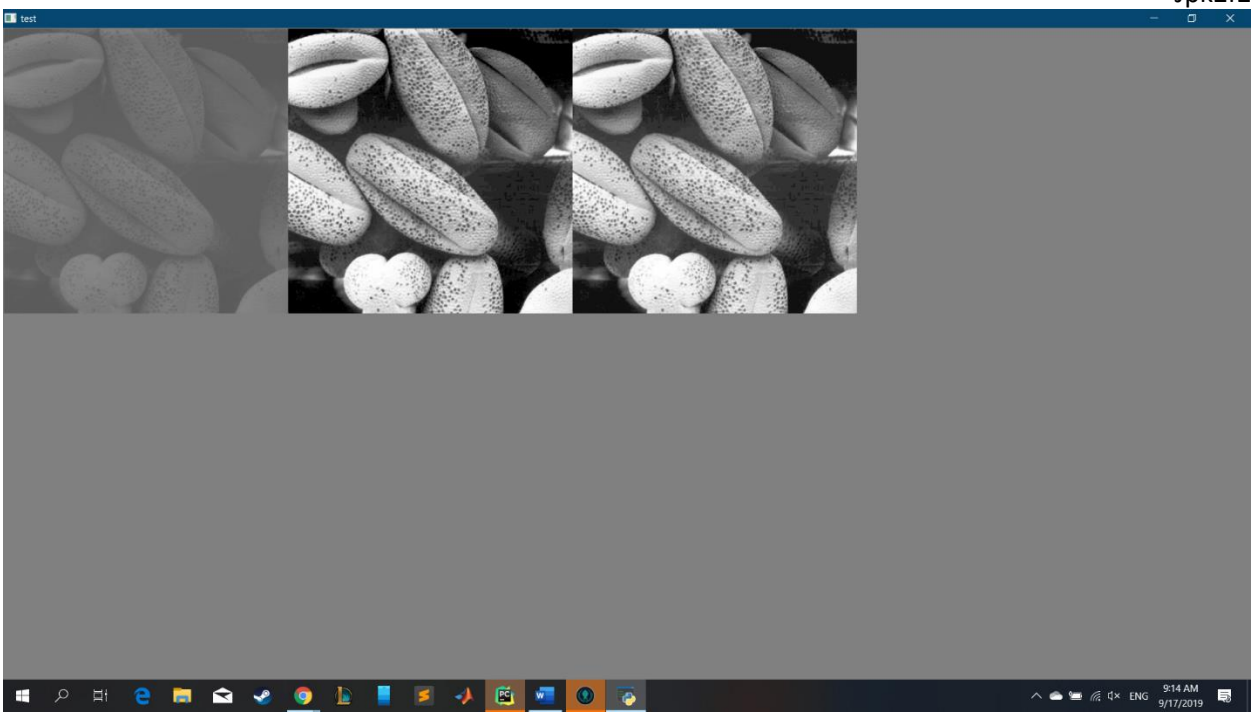

Figure 15 Image 316 Results left to right: original, cv2.equalizationHist(), personal algorithm

Overall, my results were quite satisfactory and compared well to the ideal end result. As can be seen by the results from image 308, the results of histogram equalization utilizing all pixels is not necessarily an improvement in all cases.