CS 474: Image Processing and Interpretation

Programming Assignment 1

Joshua Insorio, Mark McCord

Joshua: Parts 2 and 3 of code, theory, implementation, and results. Compiled result images.

Mark: Parts 1 and 4 of code, theory, implementation, and results. Compiled result histograms.

Theory

In part 1, image sampling was conducted on the two images. Image sampling is used to decrease the overall amount of pixels in an image, effectively scaling the image down to a smaller file size. Not only is the size of the image reduced however; the amount of information in the image is also lower after sampling. In order to sample an image, pixel values are read at certain intervals, skipping over some of the pixels and keeping others for the sampled image. These intervals are equal in the x and y direction. Then, the sampled pixels are put into their own image corresponding to their positions in the original image, and the result is a smaller image with fewer pixels than the original. There is also less information in the smaller image, since the pixels that weren't sampled from the original image are lost in the sampling process. While sampling is often not as useful as other reduction methods, due to the sheer amount of information loss, It's still practical in some cases due to how easy it is to perform on an image.

In part 2, image quantization was conducted on two images. Quantization determines how many different colors an image can have. In this case(gray scale), it determines how many levels of "gray" there are. This is broken down into 2, 8, 32, 128, and 256 as the gray levels. This is a technique also known as "lossy compression", in which compression of a range of values to a single value with minimal subjective loss. One of the applications is for a reduction in file size. The relationship between image quantization and image sampling is that they are opposite, in which image sampling is done on the x-axis and image quantization is the digitization of amplitudes.

In part 3, histogram equalization is an image processing technique that adjusts the contrast of an image by using its histogram. This technique seeks to enhance the image's contrast, as it spreads out the most frequent pixel intensity values or stretches out the intensity range of the image. These are particularly useful in images that's histogram values are "too close", and thus resulting in an image's background and foregrounds that are both bright or both dark. As a result it can lead to more detailed images as it brings out details that are either over or under exposed.

In part 4, histogram specification was conducted on two images, with the specified histograms coming from two other images. While histogram equalization deals with making an image's histogram look more like the uniform distribution through a transformation, histogram equalization aims to make an image's histogram look more like a specified histogram of any shape. In our case, the specified histogram came from the histogram of another image. This transformation is done through the use of an inverse transformation function. After equalizing the two histograms in question (the specified histogram and the histogram from the image), the histograms can be used to find an inverse transformation, which can then be used as the histogram of the new image. An example of this process is shown in the figure below.

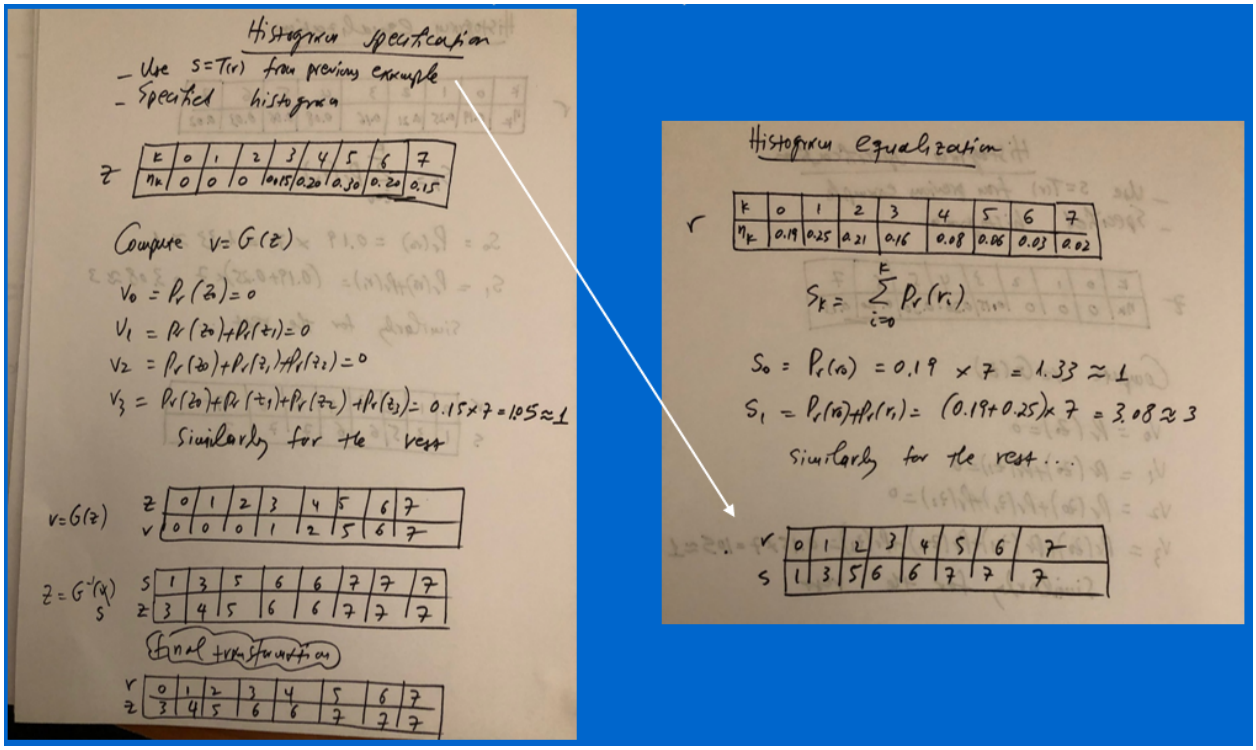

Figure 1: Example of histogram specification. The r&s table on the right represents the original image's histogram after equalization, z&v represents the specification images' equalized histogram, s&z represents the use of the inverse transform, and r&z represents the final histogram used for the transformed final image.

While histogram equalization is more useful in the general sense, histogram specification can give a much more controlled level of detail, so long as a proper histogram is specified.

Implementation

Part 1 uses the sample function in the main file. This function takes a filename as input, which it attempts to read into a new instance of the image class. Then, since the image needs to be scaled down four times, a for loop is created to scale at a rate of 2, 4, 8, and 16, using the scaling factor as an iterator. An image is created at a size based on the scaling factor, and then a filename is determined for the output image. Then, a nested for loop is used to sample every pixel that needs to be sampled (again based on the scaling factor) and add that pixel to the smaller image class. Finally, another nested for loop is used to fill another image of the original size with pixel values from the shrunken image, and that correctly sized image is written to file using the name created earlier. After the main loop finishes running 4 times, the function is complete, and the results at each sampling level are complete.

Part 2 uses the quantization function in the main file. This function takes a filename as input, which it attempts to read into a new instance of the image class. Since the quantization levels need to be scaled by four, a for loop is created to scale at a rate of 2, 8, 32, 128, and 256, using the quantization levels as an iterator. The image is created as the same size as the original, and

a filename is determined for the output image. Then, a nested for loop is used to extract each pixel and divided by the quantization level iterator and its value rounded with the round function.. It is then scaled back to the correct pixel value and written into a file using the name created earlier. After the main loop finishes, the function is complete, and the results at each quantization level are complete.

Part 3 uses the eqHistogram function in the main file. This function takes a filename as input, which it attempts to read into a new instance of the image class. First, a nested for loop is used to extract each pixel value and to be placed in a vector for storage(these are the histogram values). After, another for loop is used to calculate the cumulative frequency and the new gray levels. Each value inside the histogram vector is replaced by the rounded value of the current cumulative frequency multiplied by 255 and then divided by the total amount of pixels. Another image is created with the same size as the original. Finally, another nested for loop is used to iterate through the original image and map the pixel values with the equalized histogram vectors values into the final image to be written to file.

Part 4 is implemented in the specHistogram function. First, the same histogram equalization used in part 3 is reused on the two images passed to the function. Afterwards, the inverse mapping is completed by the use of a nested for loop. It searches for the correct mappings in the equalized histograms, and stores the new histogram values in a vector. However, since the mapping is not one to one with the entire range of possible gray levels, an extra loop is required to replace the empty values with the closest approximation from the mapping. Finally, the inverse transform (with no holes) is used for the histogram of the final image, which is then used to create the image itself via a nested for loop, and then the image is written to file.

Results and Discussion

The part 1 results are shown below in figures 2 through 9, in order of descending spatial resolution (from 256 by 256 to 32 by 32). The smaller images have been resized to the same size as the original for comparison.


Fig. 2.(left) and Fig. 3.(right), the original images to be sampled.


Fig. 4.(left) and Fig. 5.(right), the images after being scaled down to 128 by 128.


Fig. 6.(left) and Fig. 7.(right), the images after being scaled down to 64 by 64.

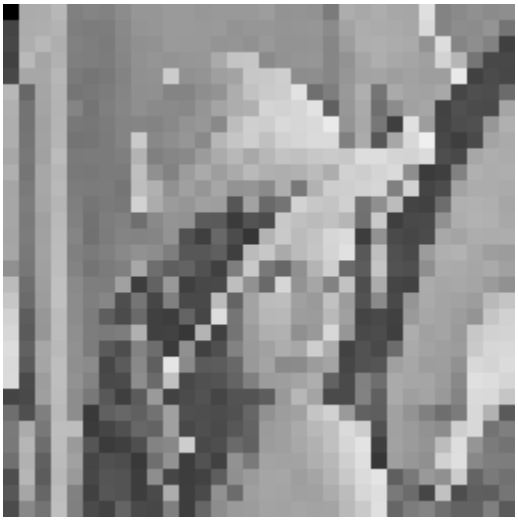


Fig. 8.(left) and Fig. 9.(right), the images after being scaled down to 32 by 32.

Overall, it seems like the process used to scale the images down was successful. As the size decreases, the image becomes more grainy, and it becomes harder to tell what the image was in the first place. While for the 128 by 128 and 64 by 64 images, the pictures are still clear enough to understand, the 32 by 32 image of peppers would be nearly unrecognisable without context, as so much information is lost.

One interesting side effect in this process is the accidental amplification of noise. This is most notable in the Lenna images, specifically figures 4 and 6. There are a few black pixels in these images that were barely noticeable in the original image. But because of how the sampling was conducted, and how the noise pixels were positioned, some of them became much more pronounced in the shrunken images. This would show that this method of image sampling is somewhat flawed, being sensitive to noise in an image. However, the methods were overall mostly successful in resizing the image in a representative fashion.


Fig. 10.(left) and Fig. 11.(right) are the resulting images with 128 gray levels(7 bits/pixel).


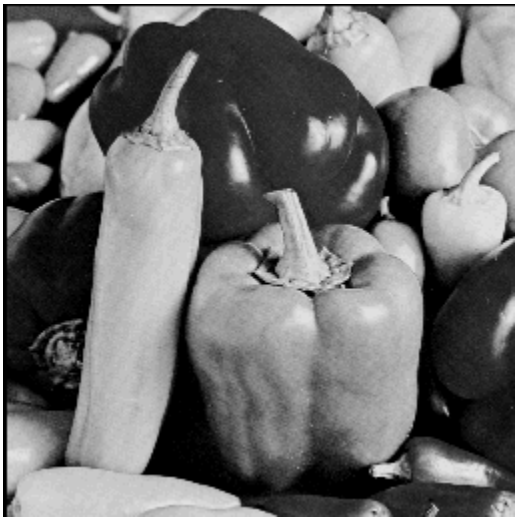

Fig. 12.(left) and Fig. 13.(right) are the resulting images with 32 gray levels(5 bits/pixel).


Fig. 14.(left) and Fig. 15.(right) are the resulting images with 8 gray levels(3 bits/pixel).


Fig. 16.(left) and Fig. 17.(right) are the resulting images with 2 gray levels(1 bit/pixel).

As provided in images above(Figures 10 through 17), these are the resulting images from the calculations of the image quantization function. With the exception of 256 gray levels, as that is the original image. Following downward, the image progressively becomes more compressed as details are slowly being reduced with each iteration. Following in accordance with the sample images in the *Intro to Image Processing* powerpoint, the function properly performs the image quantization technique. Interestingly, it becomes visually more apparent below 32 gray levels. This effect is almost “exponential” from the transition between 8 gray levels to 2 gray levels. This leads to a possible hypothesis to be formed such that it may become easier to select certain parts or areas of an image as most “distracting” details have been removed.


Fig. 18.(left) and Fig. 19. (right) are the original images to be tested.


Fig. 20.(left) and Fig. 21. (right) are the resulting images after histogram equalization.

As provided in images above(Figures 18 through 21), these are the resulting images from the calculations of the histogram equalization function. Most noticeably, Fig. 19 to Fig. 21. had the most change. This is partially due to the fact that the original image's histogram values for the background and foreground fell on the "brighter" side. After the calculations, the contrast was improved. In the case of Fig. 18, it seems that the image already came with "good" contrast and such resulted in minor improvements such as the trees being reshaded and better reflection in the water. Overall, these details that were brought out already existed in the image, and by increasing the contrast allowed for these details to be easily seen with the human eye.

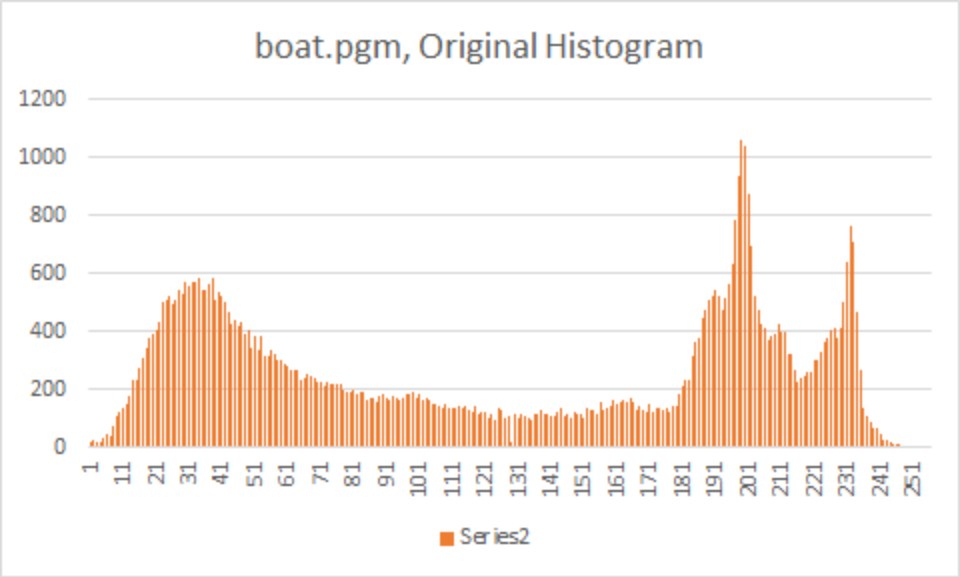

Fig. 22. Is the original histogram of the unedited Fig.18 image.

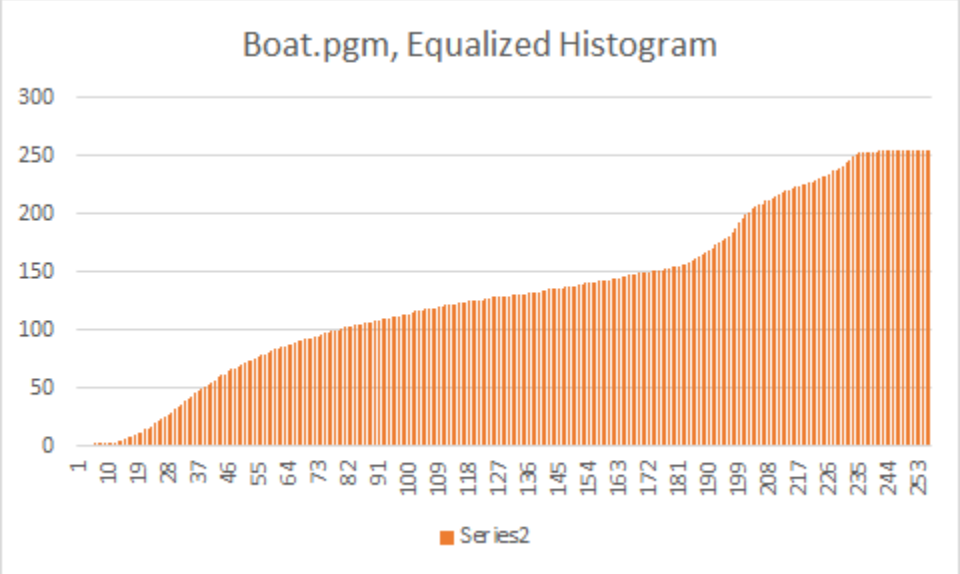

Fig. 23. Is the equalized histogram of the edited Fig.18 image.

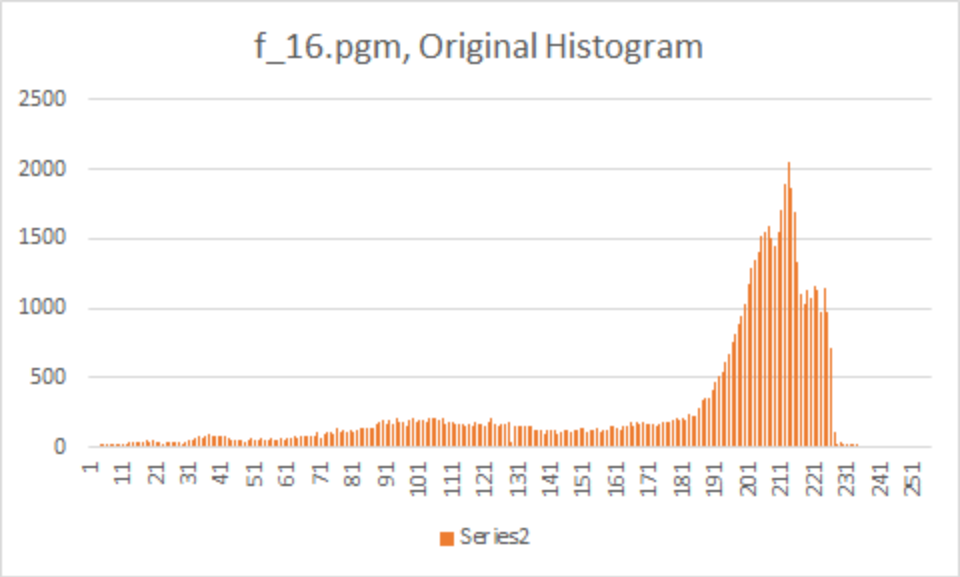

Fig. 24. Is the original histogram of the unedited Fig.19 image.

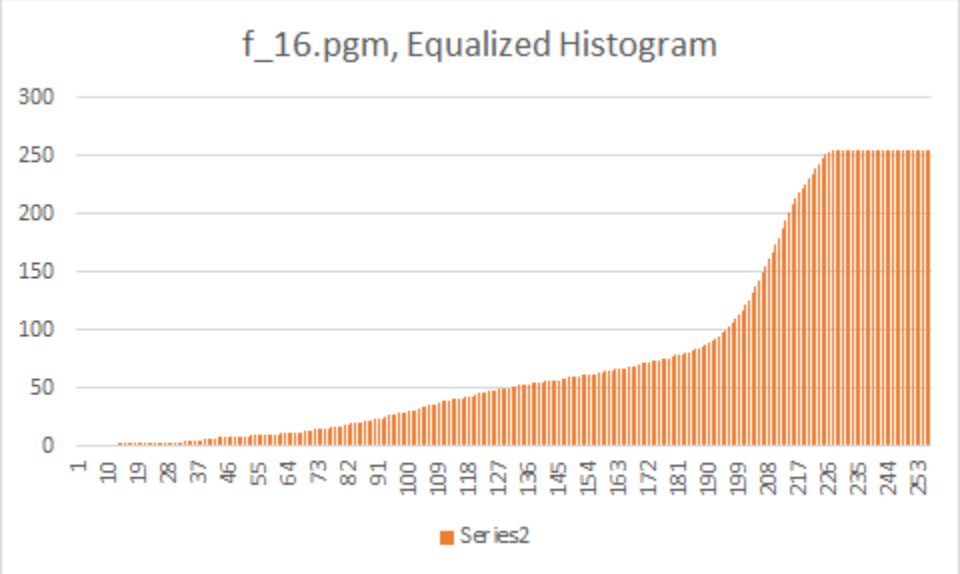

Fig. 25. Is the equalized histogram of the edited Fig.19 image.

In part 4, not only were the histograms (original and equalized) reused for the images to be transformed (Figures 22 through 25), but new histograms needed to be created for the images that the specified histograms were coming from. The below figures (26 through 29) show the histograms (original and equalized) for the specification images. As per the instructions, the data from sf.pgm was used as specification for the boat image, and the data from peppers.pgm was used as specification for the Fig. 16 image.

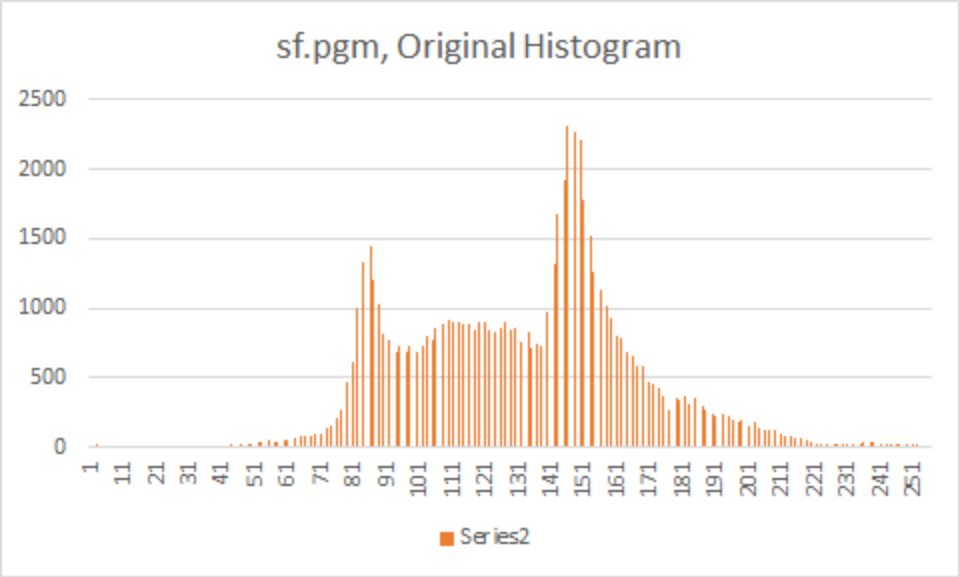

Fig. 26. The histogram for the sf.pgm image.

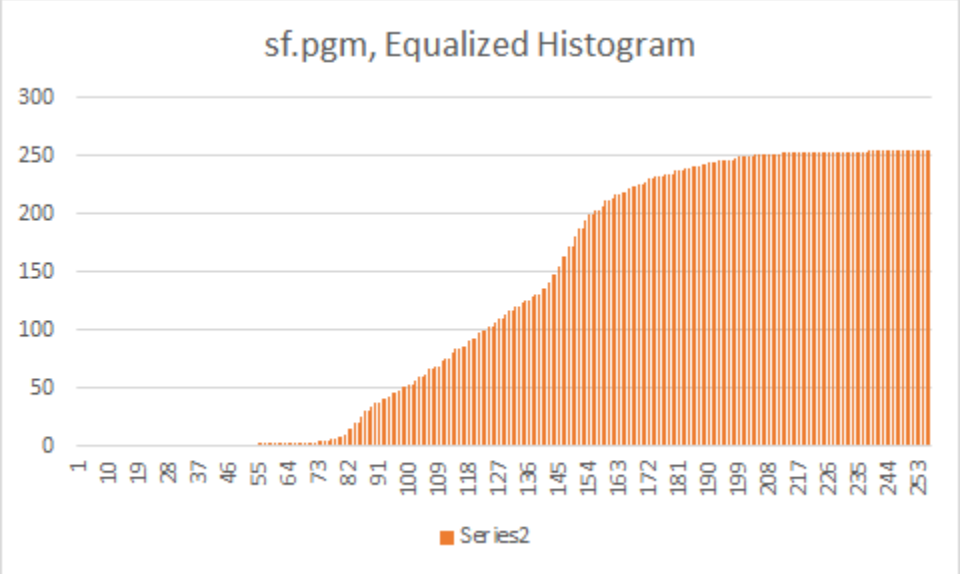

Fig. 27. The histogram for the sf.pgm image after equalization.

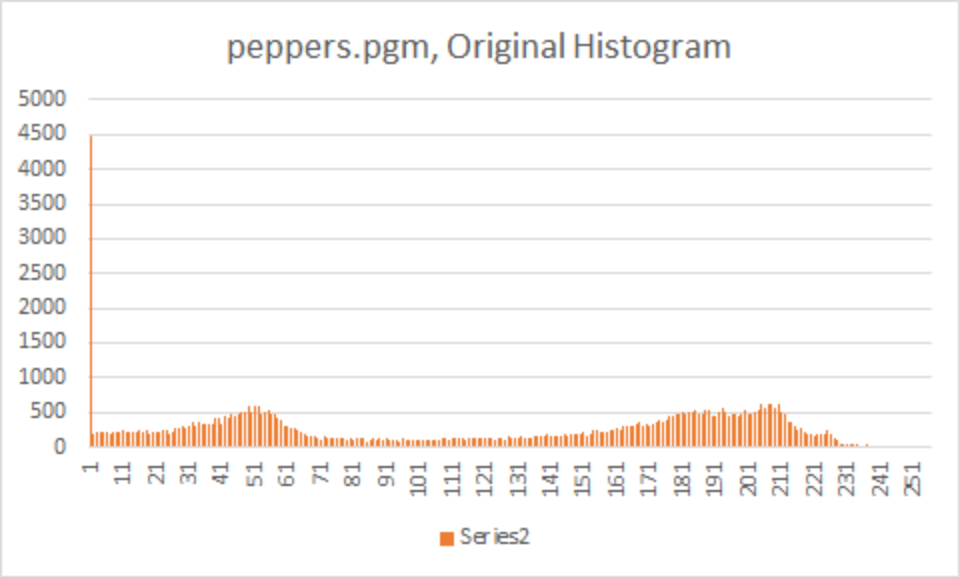

Fig. 28. The histogram for the peppers.pgm image.

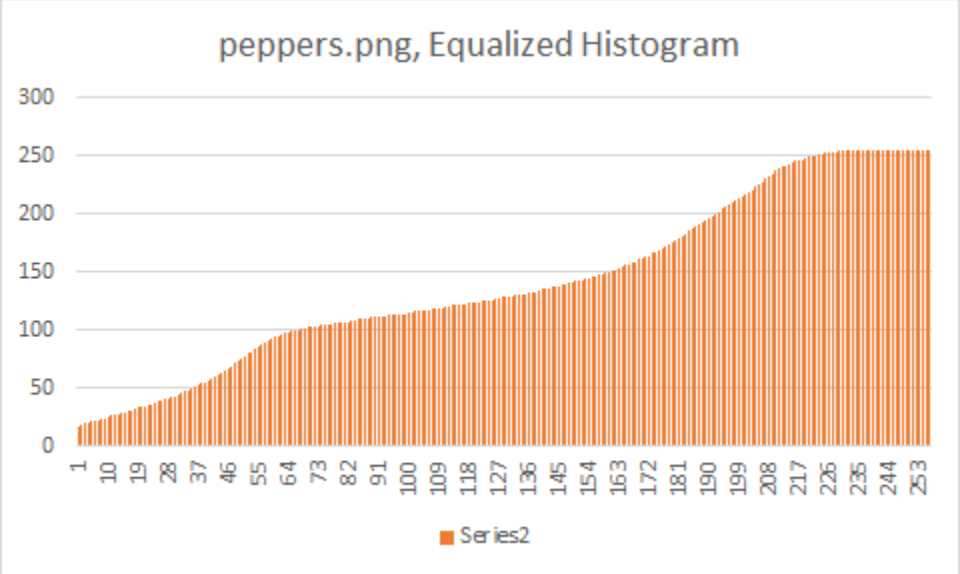

Fig. 29. The histogram for the peppers.pgm image after equalization.

While both of the resultant equalized histograms differed greatly from the originals, there are still some similarities. Namely, in both examples, the slope of the equalized histogram seems to increase when there is a spike in the original, and the slope is decreased when the original histogram has a lower section. However, it seems some of the original features are harder to track when equalized, such as the massive spike at the lower gray levels of the original peppers histogram. But overall, the equalization makes sense for these histograms.

The histograms for boat.pgm and f_16.pgm, before and after transformation, are shown below in Fig. 30 through 33.

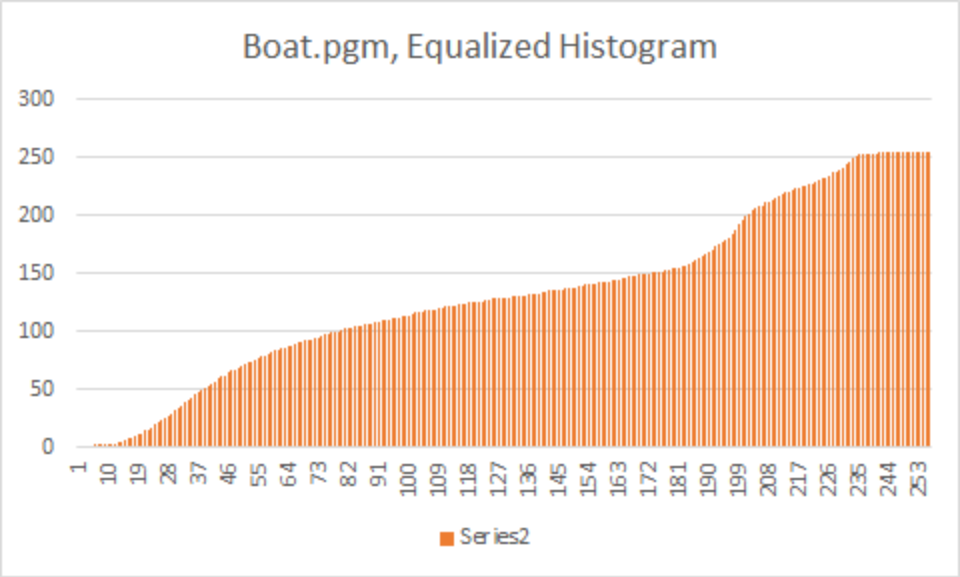

Fig. 30. Equalized histogram of boat.pgm.

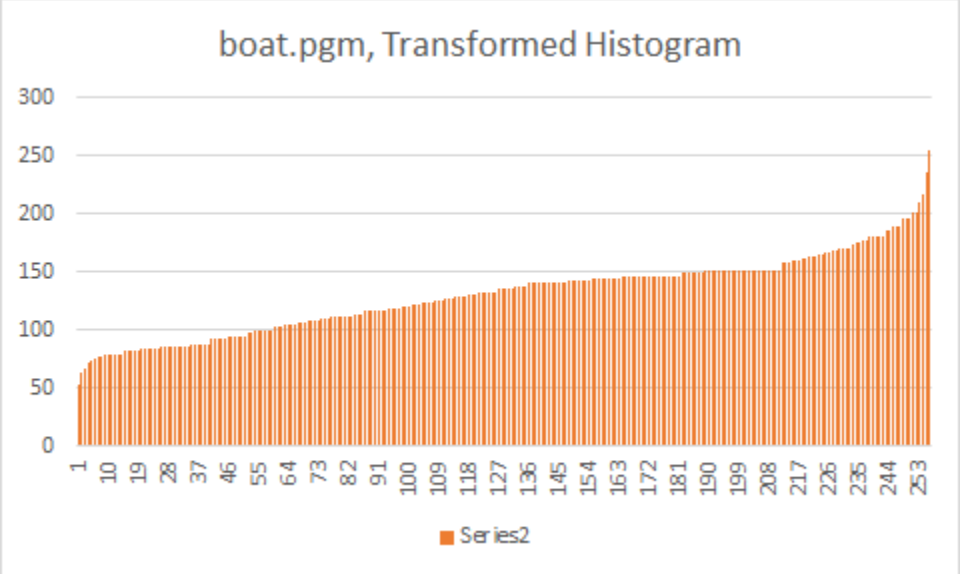

Fig. 31. Histogram of boat.pgm after the transformation.

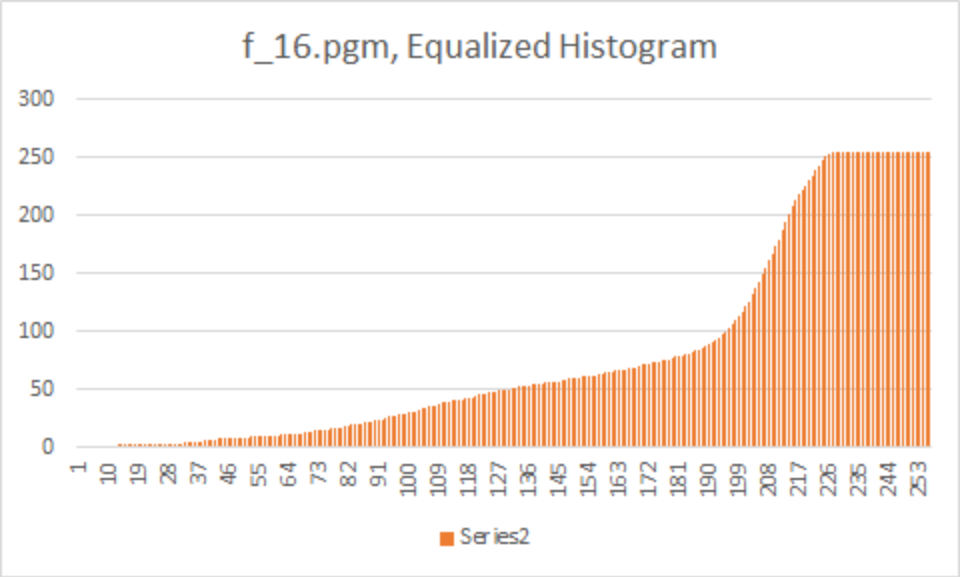

Fig. 32. Equalized histogram of f_16.pgm.

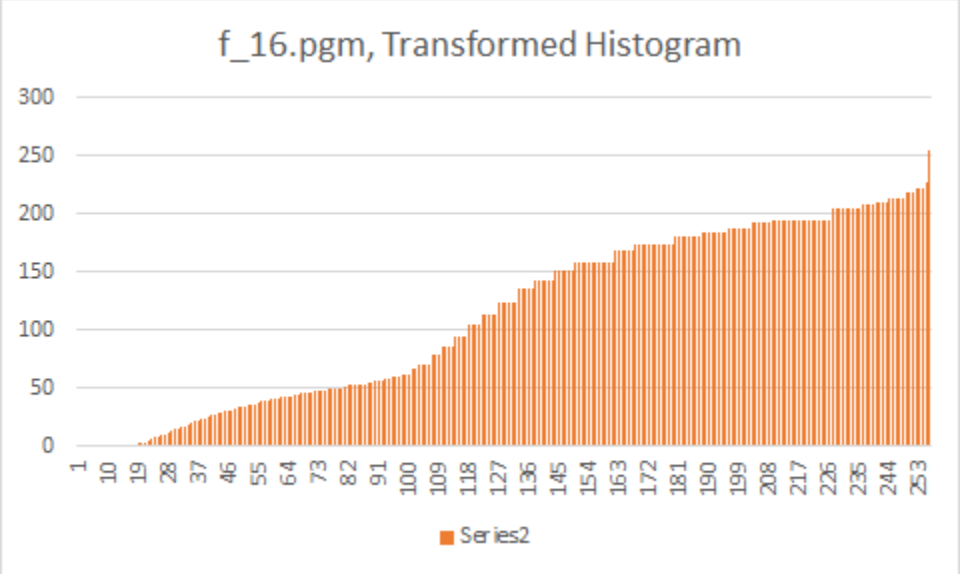

Fig. 33. Histogram of boat.pgm after the transformation.

The results here are a bit more difficult to parse. While the transformation did make quite the change to the histograms, it's difficult to say whether or not they resemble the specified histograms any better. Some similarities can be drawn in the shapes of the transformed f_16 histogram and the equalized peppers histogram, but the same can not be said for boat and sf. Also, some of the edges of the transformed histograms look more rigid than the other histograms, namely due to the method of filling in gaps in the inverse transformation, which filled large groups of empty spaces with the space's nearest neighbor.

Finally, the below figures display the comparison between the images pre and post transformation (Figures 34 through 37).


Fig. 34.(left) and Fig. 35.(right) The original Fig.18 image, and the image after histogram specification.

Fig. 36.(left) and Fig. 37.(right) The original Fig.19. image, and the image after histogram specification.

Overall, the histogram specification feels less successful than the equalization by itself. The boat image mostly became more muted in color and less well defined, and the F16 image just became slightly darker in places where it was already dark. The equalization seemed to add much more information to the images, whereas this doesn't really feel like it added much of anything. However, this only goes to show the importance of picking a proper histogram to specify. A better designed histogram (not just taken from another random image) would most likely have provided more information gain than even the equalization by itself, but given the current parameters, not much was gained from the use of histogram specification on these images.