Project 2: Linear and Bilinear Historgram Stretching with Optimal Thresholding

Daniel Sawyer U3363-7705
CAP5400 Image Processing Fall 2019
University of South Florida, Tampa

I. 1. Introduction and overall description

In this assignment we focus on histogram stretching, linear and bilinear, as well as optimal thresholding to separate the background from the foreground. We also convert color spaces from RGB to HSI, where we do the above image manipulations, and convert back to RGB before saving. These are basic image processing techniques which are extremely useful for increasing contrast in an image as well as separating background from foreground.

II. Description of algorithms

In this section the basic algorithms used in this assignment are described. The algorithm for gray scale linear histogram stretching, bilinear histogram stretching, optimal thresholding, and color space conversions are described as follows.

A. 2.1 Gray Scale Linear Thresholding

This algorithm is very basic, you take two intensity values A and B, which will be the range of intensities [A, B] that will be stretched to [0, 255] in order to improve contrast. In order to remap all the pixels, you create a ratio which is 255 / (b-a), you loop through the image and check if pixel value is within the [A, B] range. If it is within range, new pixel value = ratio * (pixel - A). If the value is less than A, set it to 0 and if greater than B, set to 255.

B. 2.2 Gray Scale Bilinear Thresholding

This algorithm is similar to the linear version except you stretch/compress to [C, D] rather than [0, 255]. You take two intensity values A and B, which will be the range of intensities [A, B] that will be stretched or compressed to C and D in order to improve or reduce contrast. In order to remap all the pixels, you create a ratio which is (D-C) / (B-A), you loop through the image and check if pixel value is within the [A, B] range. If it is within range, new pixel value = ratio * (pixel - A) + C. If the value is less than A, set it to C and if greater than B, set to D.

C. 2.3 Optimal Thresholding

Optimal thresholding is an iterative algorithm that is used to find a threshold value in order to separate the background and foreground of an image. The algorithm starts by finding the mean or median value in the image and uses that to inialize the thresholding value. You then go through the entire image and separate it into two separate images, background and foreground, using the mean/median threshold. Once separated, you take the mean of both the background and foreground independently, then take the average of those two values and that is your new threshold. You keep iterating until the difference between the old and new threshold are under some preset limit. Once it reaches below the limit, you have found your optimal threshold for separating background and foreground.

D. 2.4 Color Space Conversions, RGB ¡−¿ HSI

For color space conversion from RGB to HSI and back are fairly simple calculations. For H, S, and I you can easily program the formulas found on the conversion sheet from the TA website that I cited below. After converting, you can run image manipulations on them and then convert back to RGB in order to be viewed. Stretching I in a color image is extremely effective and far less computationally complex compared to stretching R, G, and B values separately.

III. 3.Description of implementation

The entire code is developed in C++ language on Ubuntu 18.04.3 and USF's FSPrime linux server. The code for reading and writing the ppm and pgm image files is provided as part of the image class file. The utilities class file contains all 4 process that were described in this report above. The main is in main.cpp and reads the parameters file, which contains all the info on the input image, output image, and process to run on the image with their ROIs. It then saves the resulting image(s).

IV. 4.Description and Analysis of Results

A. 4.1. Description of Results

Fig. 1. Linear Histogram Stretching 100-200

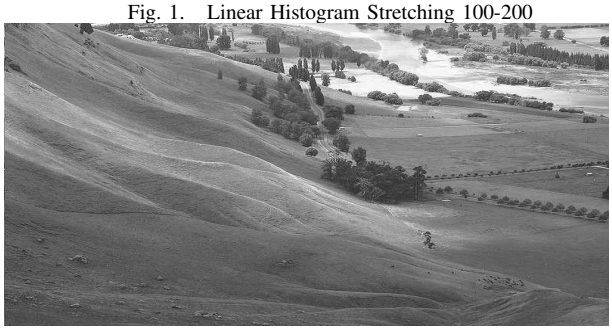


Fig. 2. Before and After Histograms

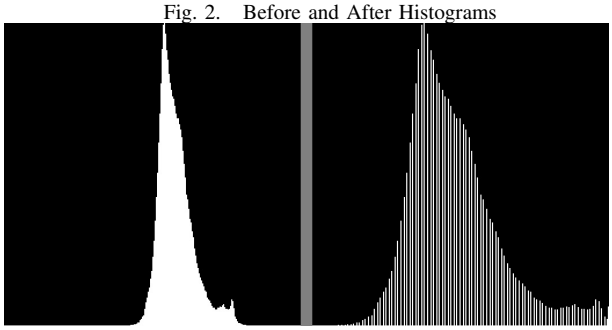


Fig. 3. Linear Histogram Stretching 100-150

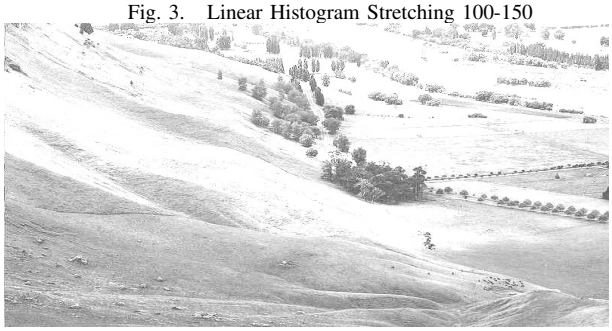


Fig. 4. Before and After Histograms

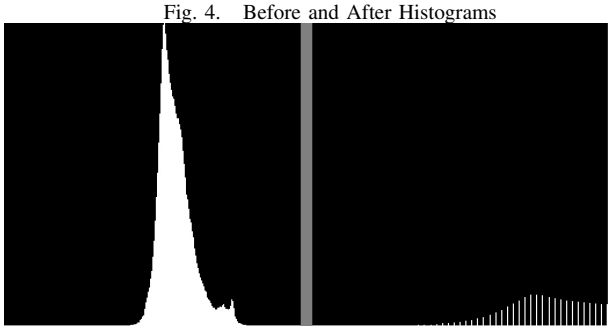


Figure 1 and 3 show the slope image being stretched and figures 2 and 4 show the before and after histograms. The first figure had [A, B] = [100, 200], while figure 3 used [A, B] = [100, 150]. As you can see, both times the histogram was stretched pretty well but it is hard to pinpoint the exact [A, B] you should be using.

B. 4.1. Description of Results

Fig. 5. Bilinear Histogram Stretching 100-150, 50-200

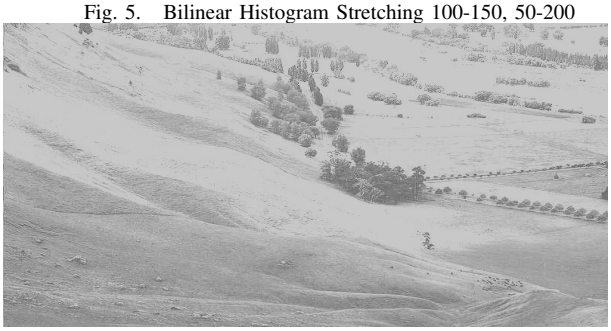


Fig. 6. Before and After Histograms

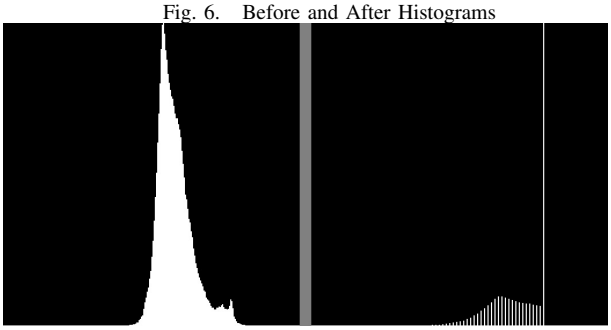


Fig. 7. Linear Histogram Stretching 100-200, 50-240

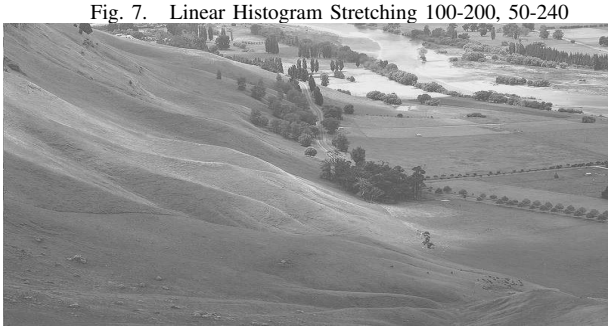
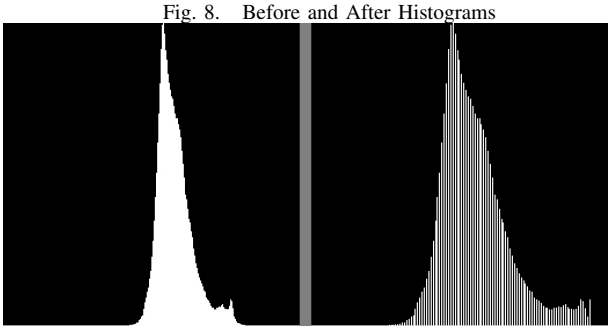


Fig. 8. Before and After Histograms



This shows bilinear histogram stretching. Similar results as the linear version except you have more control over how far you can stretch and can also compress the histogram, which you cant do with linear.

Fig. 9. Optimal Thresholding Binarization

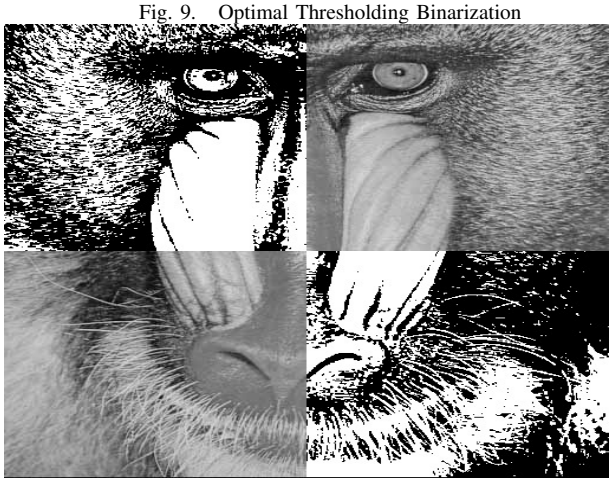


Fig. 10. Slope Background

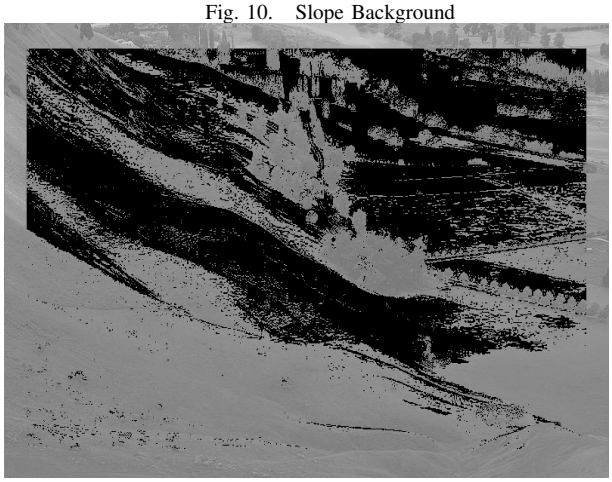
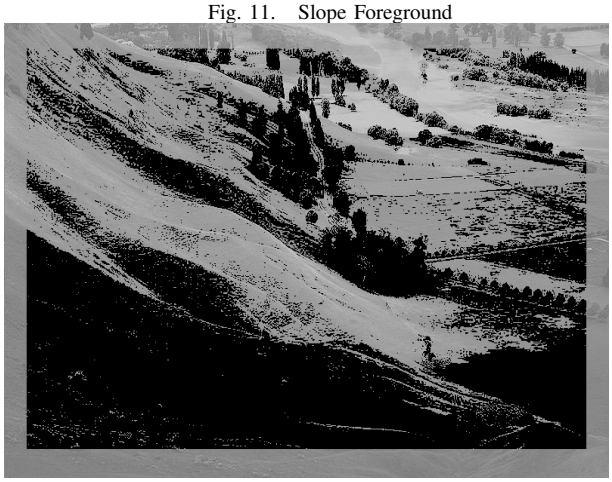
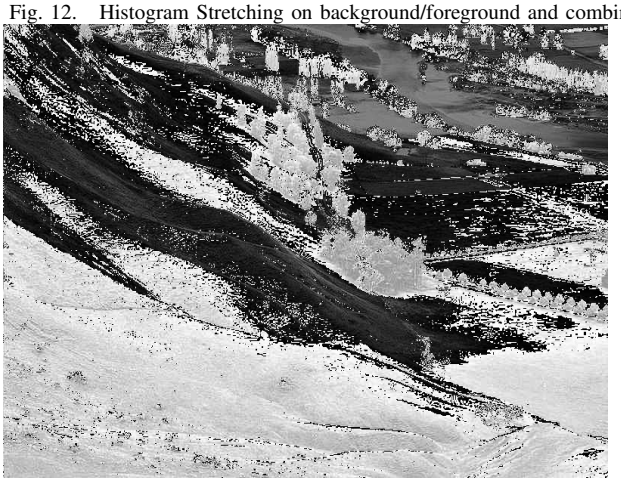


Fig. 11. Slope Foreground



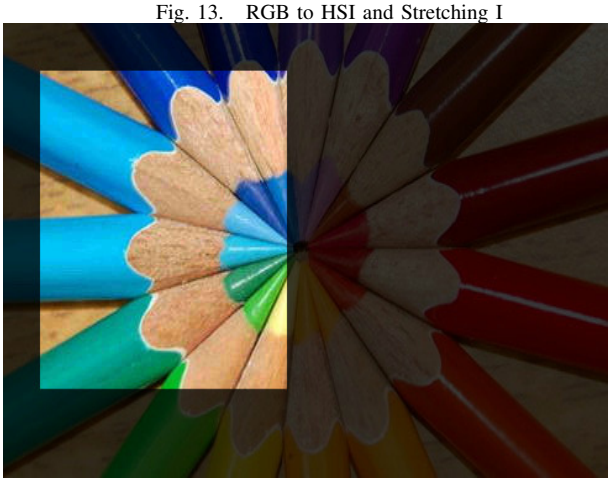
In figures 9 through 11 you can see the background and foreground being separated using the optimal thresholding algorithm. Figure 9 has been binarized so black is background and white is foreground. Figure 10 and 11 are the background and foreground parts of the slope image.

Fig. 12. Histogram Stretching on background/foreground and combined



In this image, you can see the background and foreground have been stretched independently and then recombined to result in quite a weird looking image.

Fig. 13. RGB to HSI and Stretching I



This is color histogram stretching. This has only been stretched on one channel of HSI, the I or intensity. This allows you to stretch color images very well and makes the constrast across colors look really nice.

C. 4.2. Performance Evaluation and Analysis of Results

All the algorithms ran pretty well and pretty quick. N = image rows, M = image cols. With the histogram stretching, the complexity will always be O(NM), which is the number of pixels in the image. With optimal thresholding you can only iterate a certain number of times before it converges, no matter what. I believe worst case iterations would be 8, so it would also just be O(NM) as well since N and M are significantly larger than 8. Color conversion also is O(NM) as well.

V. 5.Conclusion

This assignment was used as a way to learn how to perform contrast stretching and compression along with being able to separate the background from the foreground using

the optimal thresholding algorithm. Color space conversion is also extremely useful and will have a lot of use in computer vision since most images are in color.

REFERENCES

[1] Example Project Code and Example Report from TA website.
[2] RGB to HSI Conversion Calculations Document on TA website