Project 2: Histogram Modification and Color Processing

Daniel Sawyer U3363-7705
CAP4401 Image Processing Spring 2017
University of South Florida, Tampa

***Abstract*— Histogram modification using intensity stretching with color processing and conversion between HSI and RGB**

I. Introduction and overall description

This assignment focuses on histrogram modification by intensity stretching and basic color processing. The histogram intensity stretching is used to enhance the contrast on the pixel intensity range set by the user. It is then stretched to another range which is also set by the user. The new image will be easier for humans to view/read, assuming the user chooses the right ranges to stretch. The color processing is done in two ways. The first way is to stretch the original RBG intensity values in each pixel, which isnt recommended as the best solution. The second way is to first convert the RGB to a different color representation called HSI. HSI allows the intensity to be stretched by itself, the images brightness, instead of having to stretch the colors. This allows the brightness to be separate from the colors, which we cannot do in RGB manipulations. The hue and saturation parts of the HSI representation are what contain the information on the colors and can be used to tweak the colors as well if needed. In Section 2, the basic algorithms used in the assignment are described. Section 3 describes the implementation details of the assignment. In Section 4 the results for the assignment is presented and finally we conclude in Section 5.

II. Description of algorithms

In this section the basic algorithms used in this assignment are described. The algorithm for histogram stretching, RGB to HSI, and HSI to RGB are described as follows.

A. Histogram Stretching

Histogram stretching does not require you to actually construct a histogram of the image. The user just inputs a range of intensities A and B, along with values C and D for it to be stretched too. A greater equal to C and B less than or equal to D. Any pixel intensity less than A goes to 0/black and any intensity greater than B goes to 255/white. For pixel intensities from [A,B], the new pixel value = $\frac{D-C}{B-A}*(Iij-A)+C$. This is the equation to use for all histogram stretching, whether it be Gray Scale, Red Channel, Green Channel, Blue Channel, RGB all channels, Hue, Saturation, or Intensity the equation is the same. Saturation has to be normalized to 255 and Hue is from [0,360] instead of [0,255] but the stretching is basically the same algorithm for all of them.

B. Gray Scale to HSI

This algorithm just puts the one and only intensity value into I and sets H and S to zero. Very simple conversion for gray scale to HSI.

C. RGB to HSI

This algorithm goes through every pixel and converts them to Hue, Saturation, and Intensity representation depending only on intensity values of each pixels' Red, Green, and Blue channels. I = $\frac{(R+G+B)}{3}$, S = 1 - $\frac{min(R,G,B)}{I}$, θ = $cos^{-1}[\frac{\frac{1}{2}[(R-G)-(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}]$. If B<=G, H = θ. Else, H = 360 - θ. It is a pretty straight forward algorithm and is not quite as complicated as its conversion back to RGB in the next sub-section.

D. HSI to RGB

This algorithm largely depends on the value of H and simply converts each pixel like so:

1. If (0 <= H <120):
   B = I*(1-S)
   R = I*[1 + $\frac{S*cos(H)}{cos(60-H)}$]
   G = 3*I - (R+B)
2. If (120 <= H <240):
   H = H - 120
   R = I*(1-S)
   G = I*[1 + $\frac{S*cos(H)}{cos(60-H)}$]
   B = 3*I - (R+G)
3. If (240 <= H <= 360)
   H = H - 240
   G = I*(1-S)
   B = I*[1 + $\frac{S*cos(H)}{cos(60-H)}$]
   R = 3*I - (G+B)

III. Description of implementation

The entire code is developed in C++ language on Ubuntu 16.04.1 and USF's FSPrime linux server. The code for reading, writing the ppm and pgm image files, and conversion from HSI is provided as part of the Image Class file. The HSI Class contains conversion to HSI from the Image Class and all HSI stretching. The main is in project2.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). For all histogram stretching besides Hue, the user enters intensity values [0,255] for A,B,C,and D. For Hue stretching, the user enters values [0,360] for A,B,C,and D. The conversions

between HSI<->RGB just converts each pixel and does not require any user input. All processes also have an ROI that uses the following format that is put in after the process parameter(s): startX startY SizeX SizeY. The README includes a more detailed explanation.

IV. Description and Analysis of Results

A. Description of Results

This section illustrates the results of the algorithms used. The first group of images are the before and after of the gray scale histogram stretching(Figures 1 and 2). The second group shows the color stretching of individual channels R, G, B, all channels RGB, and stretching I with the baboon(Figures 3-8). Last is I with the Pen image(Figures9-10).

Fig. 1. Original Baboon Gray Scale

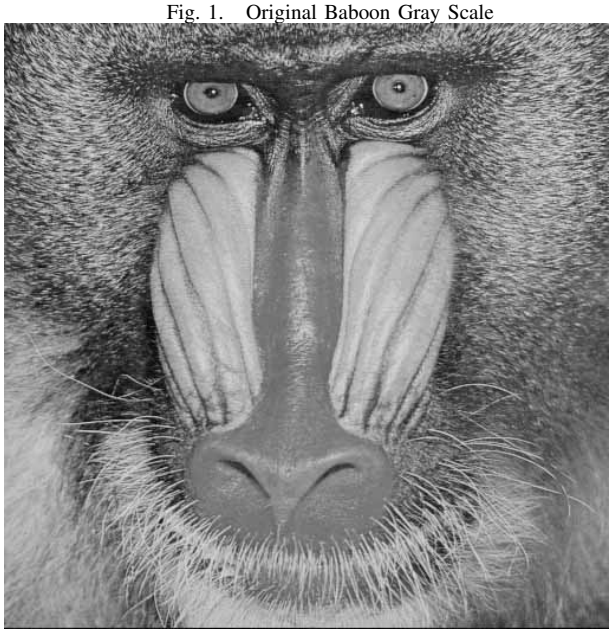


Fig. 2. A,B,C,D = 100,200,50,255

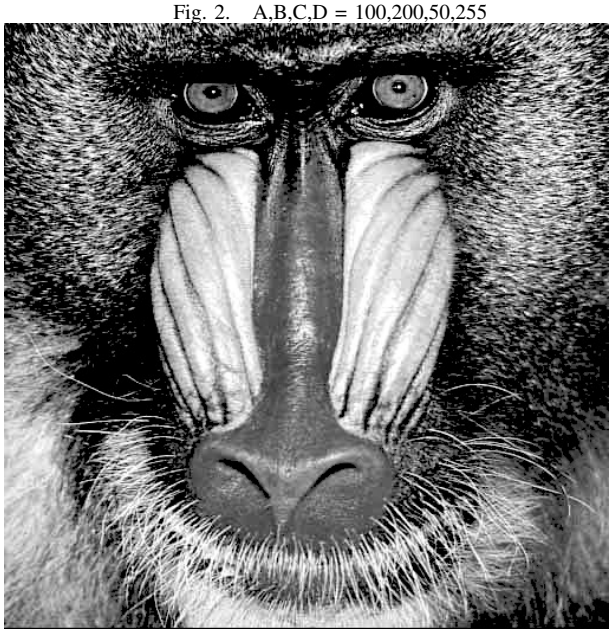


Fig. 3. Original Baboon Color

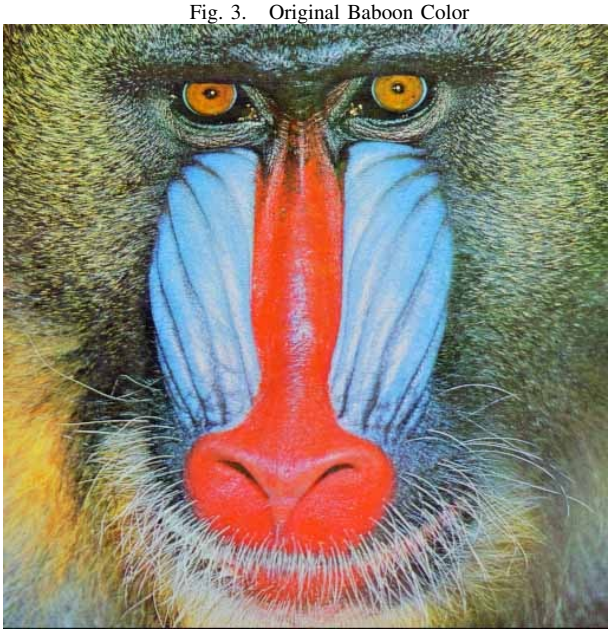


Fig. 4. Red Stretch: A,B,C,D = 100,200,50,255

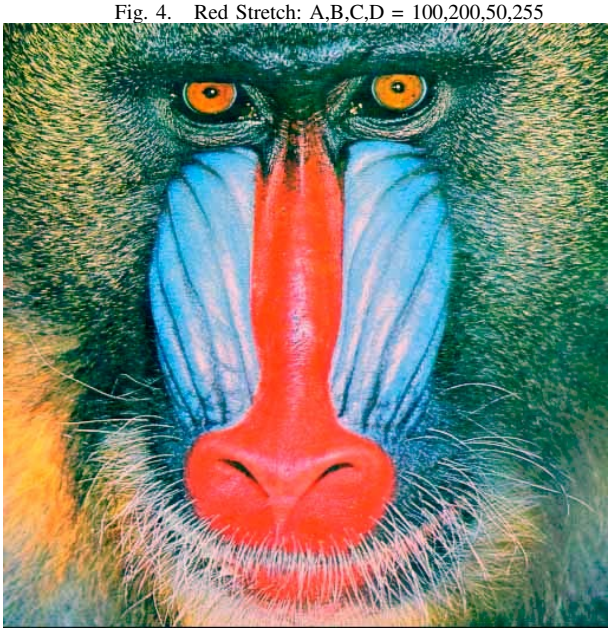


Fig. 5. Green Stretch: A,B,C,D = 100,200,50,255

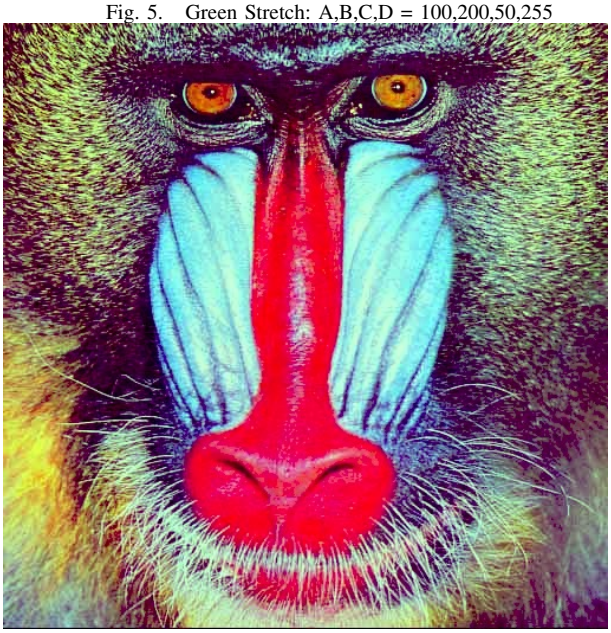


Fig. 6. Blue Stretch: A,B,C,D = 100,200,50,255

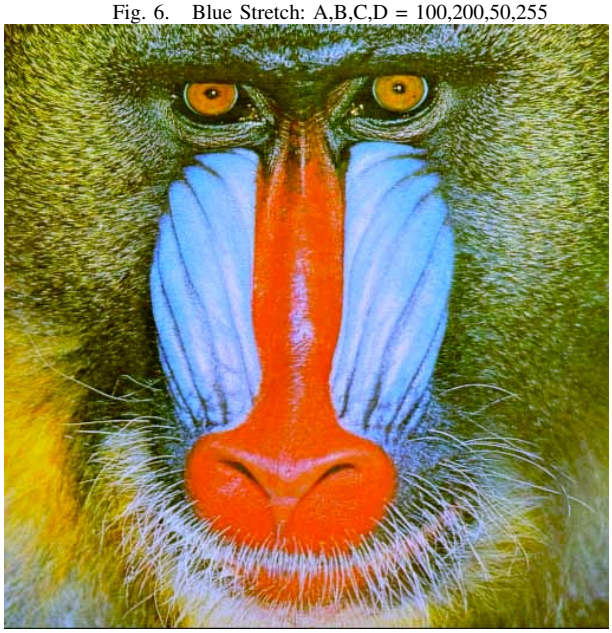


Fig. 7. RGB Stretch: A,B,C,D = 100,200,50,255

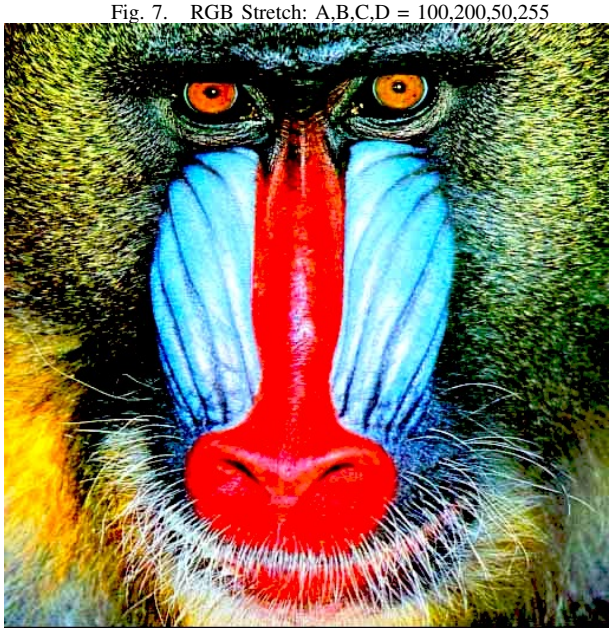


Fig. 8. I Stretch: A,B,C,D = 100,200,50,255

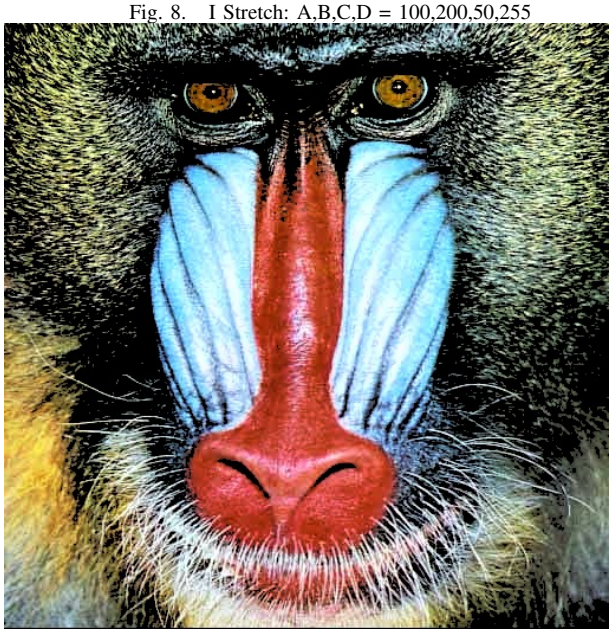


Fig. 9. Original Pen

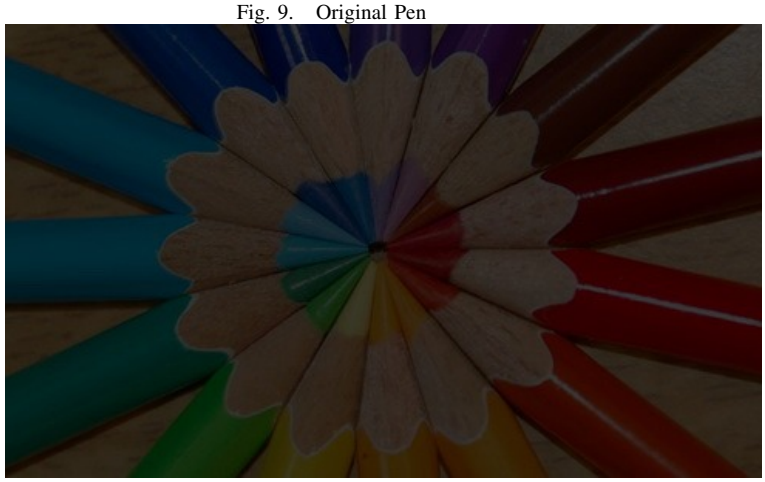
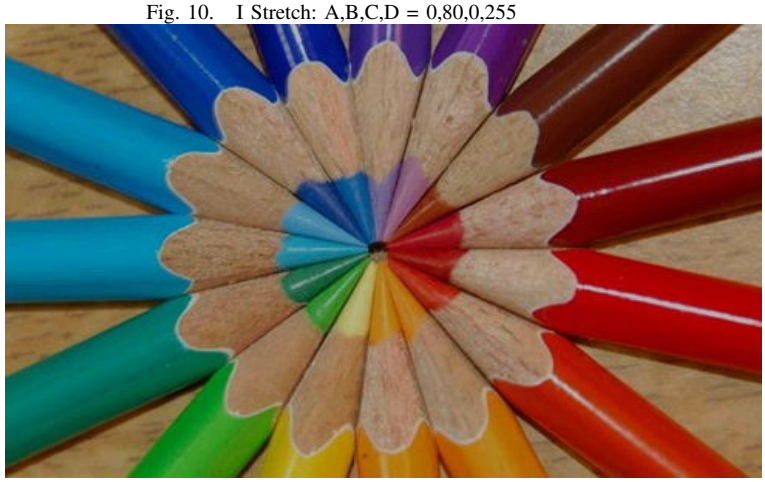


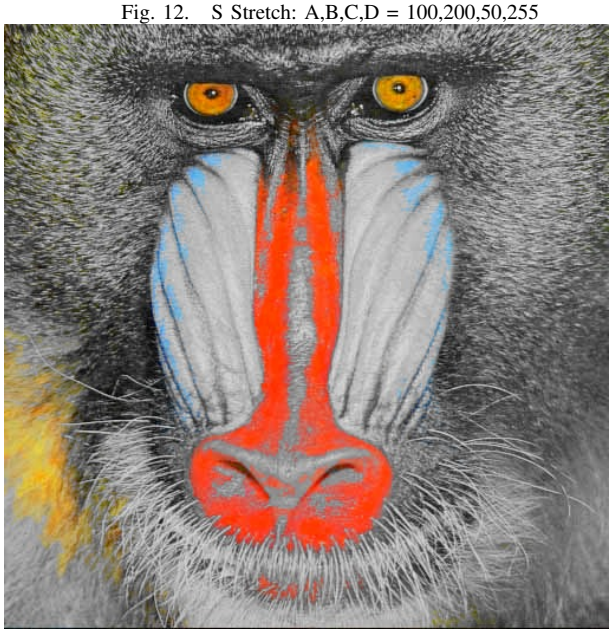
Fig. 10. I Stretch: A,B,C,D = 0,80,0,255



Stretching H and S from original Baboon Image Below.

Fig. 11. H Stretch: A,B,C,D = 100,200,50,255

Fig. 12. S Stretch: A,B,C,D = 100,200,50,255

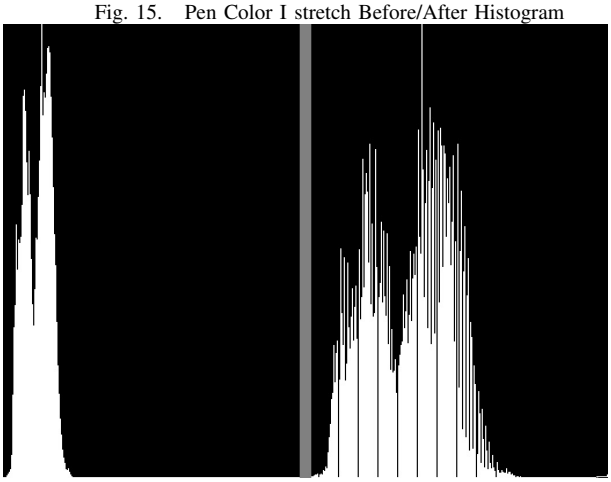


The following are the before and after histograms of Figures 2, 8, and 10.

Fig. 13. Baboon Gray Scale Stretch Before/After Histogram

Fig. 14. Baboon Color I stretch Before/After Histogram

Fig. 15. Pen Color I stretch Before/After Histogram



B. Performance Evaluation and Analysis of Results

All three algorithms run at O(n^2), where n^2 is the size of the 2D pixel array or i x j. This is because all the algorithms just go through every pixel and converts them as needed. The conversion from RGB to HSI and back does take a little more time to execute if you are stretching just one channel of RGB, 129ms for stretching just Red vs 133ms for conversion and stretching I, but as you can see this hardly slows down the execution speed and is well worth the time for the advantages that the HSI format has. Also, stretching I vs stretching R,G,and B at the same time, stretching the I value is much faster, 318ms for stretching all RGB vs 138ms for conversion and stretching I. This shows the clear advantage HSI has for color processing over using RGB. (All execution times from Intel i7-3770K @4.2GHz)

V. Conclusion

This assignment introduces us to histogram modification by stretching, HSI color representation and manipulation, as well as basic color processing. Histogram stretching is a way to improve contrast for the human eye as well as a way to manipulate a certain color channel, Hue, or Saturation. The HSI color representation is great for computation time and allows for easier manipulation through a computer program as well as allowing I to control brightness in a sense. Where as if you were using RGB, you would have to manipulate all the color channels in order to increase the ”brightness”. I can see the clear advantages of HSI and that we will probably be using it exclusively for future image manipulations.

References

[1] Example Project Code and Example Report from TA website.