Assignment 2: Histogram Modification and Color Processing.

KEJVI CUPA
Department of Computer Science and Engineering
University of South Florida, Tampa, Florida, USA

1. Introduction and overall description

This assignment focuses on practicing histogram modification and color processing. Specifically, we are focusing on contrast enhancement methods. Histogram stretching is a very useful method in enhancing contrast in a gray level or color image by stretching the dynamic range of brightness levels in an image to the dynamic range of display (human vision). Regions of Interest (ROI) are presented in this assignment and allow these methods to be implemented in certain regions of the image rather than the whole image. Up to three regions of interest can be used on an image. Image processing will be applied to Grey level and color images. Overall, there will be nine new functions that will be implemented: Histostretch, althistostretch, and histothres which will be used on Grey level images and percchastrech, rgbstretch, isstretch, hstretch, sstretch, and fullhsistretch that will be used on Color images. A more detailed description of each function will be provided in this report.

2. Description of Algorithm

The first algorithm to be implemented is “Histostretch”. This is actually more like a helper function used to output the histogram of an ROI before and after it has been stretched. You can simply call this function from within the other algorithms implemented below. All you need to pass is the histogram data which is inside a 2D array, and the name of the histogram which you can adjust for each ROI so that it is easier to spot the difference between multiple histograms. Time Complexity is $O(n^2)$.

The second algorithm to be implemented is “althistostretch”. In this algorithm, a histogram of the ROI in the original image is created. The C, D values which represent the Min and Max grey levels in the histogram are identified. In addition to the ROI parameters, the user provides the intensity range values [A,B] for each specified ROI. Histogram stretching is performed on the pixels of the image using the formula: ((current_pixel_value - C) * ((B - A)/(D-C))) + A. In order to prevent outliers from affecting the stretching, all pixel values less than 5% of 255 and larger than 95% of 255,

are mapped to A, B respectively. The new histogram is generated and visualizes the stretching that took place. Multiple helper functions are used to help with code organization and calculations needed to be performed for the stretching. Time Complexity is $O(n^2)$.

The third algorithm to be implemented is “histothres”. This algorithm builds upon the “alhistostretch” algorithm. In this one, we have added thresholding to improve contrast enhancement. Therefore, an additional parameter which is Threshold has been added for each ROI and it is a user input. In this algorithm, we separate the image into bright pixels (>= threshold) and dark pixels (< threshold). Histogram stretching is performed only on the dark pixels only. As mentioned already, the histogram stretching performed is from the “alhistostretch” algorithm. This algorithm allows for more enhancement and parameter selections. Time Complexity is $O(n^2)$.

The fourth algorithm to be implemented is “percchastrech” which performs histogram stretching but on Color images. In addition to the ROI parameters and the intensity range [A,B], the user needs to specify the R,G,B channel that histogram stretching will be performed on. This algorithm only stretches one channel, and keeps the other two with the original values. The implementation is very similar to the previous algorithms. Additions were made to represent RGB images which have 3 channels instead of 1 channel compared to Grey level images. Therefore, more initializations and transformations are performed, but the logic and the stretching formula is the same. RGB channels have data in the range 0-255. Time Complexity is $O(n^2)$.

The fifth algorithm to be implemented is “rgbstretch”. This algorithm is a more advanced version of the fourth algorithm. Instead of applying histogram stretching to only one channel, it applies it to all channels (R,G,B) for each ROI specified by the user. The ROI parameters and intensity range parameters are provided by the user. The logic of this algorithm is the same as that from the fourth algorithm and the previous algorithms. The main difference is that it is applied on Color images and on all three channels. This algorithm provides good contrast enhancement. Time Complexity is $O(n^2)$.

The sixth algorithm to be implemented is “istretch”. We will only stretch the I component. User provides the channel, and ROI parameters and intensity range values. This algorithm is a bit more complicated than the previous ones because firstly we need to convert from RGB color space to HSI color space. Two helper functions “rgbtohsi” and “hsitorgb” were created to facilitate this conversion. The input image will still be an RGB image, however, the RGB image is converted to HSI and histogram stretching is performed on the I - intensity component. The histogram stretching logic is the same as

before. After all, I is also in range 0-255 so it is technically a grey level image. However, for the display we still need to display the color image in RGB space, so after performing stretching on I only, the image is converted back to RGB. Of course, because of the stretching of I, RGB values will be adjusted after the conversion resulting in a contrast enhanced output image. Time Complexity is $O(n^2)$.

The seventh algorithm to be implemented is “hstretch”. The user will provide the ROI parameters, intensity range values and the channel (the channel will be H). This algorithm is similar to the previous one, but instead of stretching the I component of the HSI space, we will be performing histogram stretching on the H component. The conversion of RGB to HSI is performed. The range of H (hue) values is 0 - 360. After the stretching has been performed the HSI image is converted back to RGB for display. The histogram stretching logic and calculations are similar to the previous algorithms. Adjustments are made where necessary, but very few are made compared to the previous algorithm. The [A,B] range needs to be optimized for best contrast enhancement results. Time Complexity is $O(n^2)$.

The eighth algorithm to be implemented is “sstretch”. The user will provide the ROI parameters, intensity range values and the channel (the channel will be S). This algorithm is similar to the previous one, but instead of stretching the I or H component of the HSI space, we will be performing histogram stretching on the S (saturation) component. The conversion of RGB to HSI is performed. The range of S values is 0 - 100. After the stretching has been performed the HSI image is converted back to RGB for display. The histogram stretching logic and calculations are similar to the previous algorithms. The [A,B] range needs to be optimized for best contrast enhancement results. Time Complexity is $O(n^2)$.

The ninth algorithm to be implemented is “fullhsistretch”. The user provides the ROI parameters, and the intensity range values [A,B] for each component H, S, I. The logic of this algorithm is the same as the previous ones. The main difference is that histogram stretching is performed on all H,S,I channels instead of only one of them. So after the image has been converted from RGB color space to HSI color space, histogram stretching is performed on the H, S, I component and then everything is combined together. Then we convert back to RGB color space to display the output image. The [A,B] range needs to be optimized for best contrast enhancement results. Time Complexity is $O(n^2)$.

3. Results

Below are shown the results (images) with different combinations of the respective function parameters.

Figure 1: Original Image

Figure 2: “althistrostretch” function. Performing histogram stretching on the 3 used ROIs. Different intensity range values were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 30, Sx1= 200, Sy1= 150, A1= 0, B1= 255, x2 = 50, y2 = 200, Sx2 = 110, Sy2 = 210, A2 = 0, B2 = 255, x3 = 200, y3 = 200, Sx3 = 150, Sy3 = 150, A3 = 40, B3 = 180

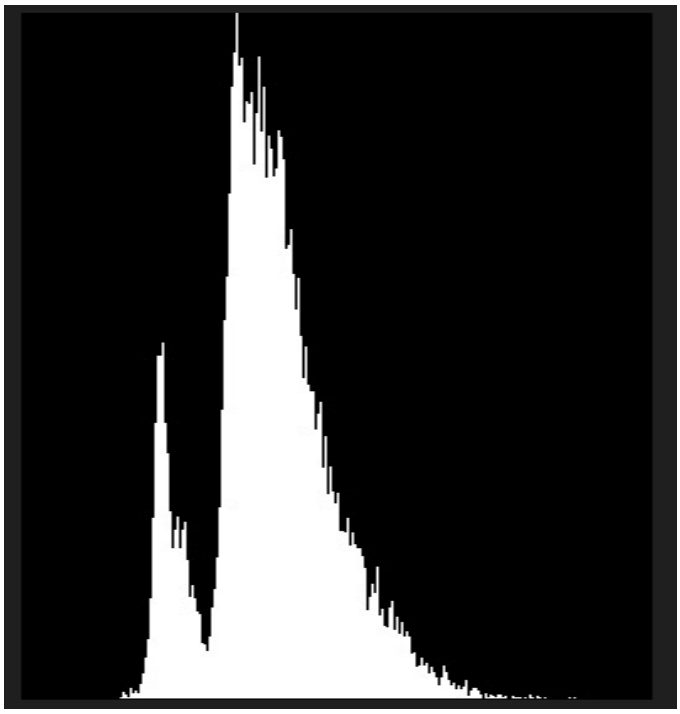

Figure 3.1 “Histostretch” function called on “althistretch” function. The parameters passed were specified in the Figure 2 label. This is the Histogram of ROI - 2, in figure 2, ***before*** Histogram Stretching was applied. As we can see from this figure, not all intensity levels have pixels.

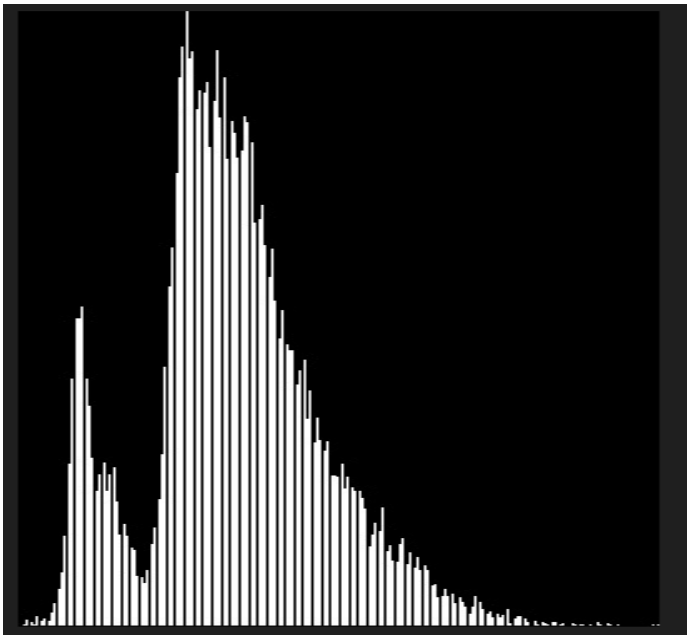

Figure 3.2 “Histostretch” function called on “althistretch” function. The parameters passed were specified in the Figure 2 label. This is the Histogram of ROI - 2, in figure 2, ***after*** Histogram Stretching was applied. As we can see from this figure, the histogram has been stretched.

Figure 4: Original Image

Figure 5: “althistrostretch” function. Performing histogram stretching on the 3 used ROIs. Different intensity range values were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 150, A1= 0, B1= 255, x2 = 250, y2 = 50, Sx2 = 200, Sy2 = 150, A2 = 0, B2 = 255, x3 = 50, y3 = 350, Sx3 = 400, Sy3 = 400, A3 = 0, B3 = 255

Figure 6: “histothres” function. Performing histogram stretching with thresholding on the 3 used ROIs . Different intensity range values and threshold were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 150, Threshold1 = 170, A1= 0, B1= 255, x2 = 250, y2 = 50, Sx2 = 200, Sy2 = 150, Threshold2 = 210, A2 = 30, B2 = 235, x3 = 50, y3 = 350, Sx3 = 400, Sy3 = 400, Threshold3 = 150, A3 = 0, B3 = 255

Figure 7: “histothres” function. Performing histogram stretching with thresholding on the 3 used ROIs . Different intensity range values and threshold were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 30, Sx1= 200, Sy1= 150, Threshold1 = 150, A1= 0, B1= 255, x2 = 50, y2 = 50, Sx2 = 110, Sy2 = 210, Threshold2 = 180, A2 = 0, B2 = 255, x3 = 200, y3 = 200, Sx3 = 150, Sy3 = 150, Threshold3 = 180, A3 = 40, B3 = 180

Figure 8: Original Color Image

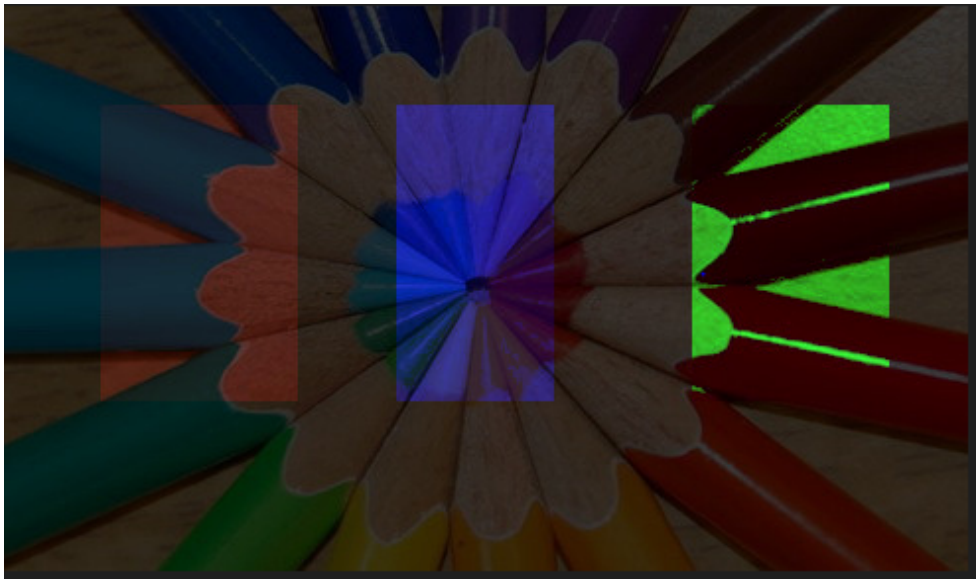

Figure 9: “percchastrech” function. Performing histogram stretching on Color Images on RGB color space, on the 3 used ROIs . Different intensity range values and channels(R, G, B) were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 100, Channel1 = R, A1= 20, B1= 100, x2 = 50, y2 = 200, Sx2 = 150, Sy2 = 80, Channel2 = G, A2 = 50, B2 = 200, x3 = 50, y3 = 350, Sx3 = 150, Sy3 = 100, Channel3 = B, A3 = 0, B3 = 255

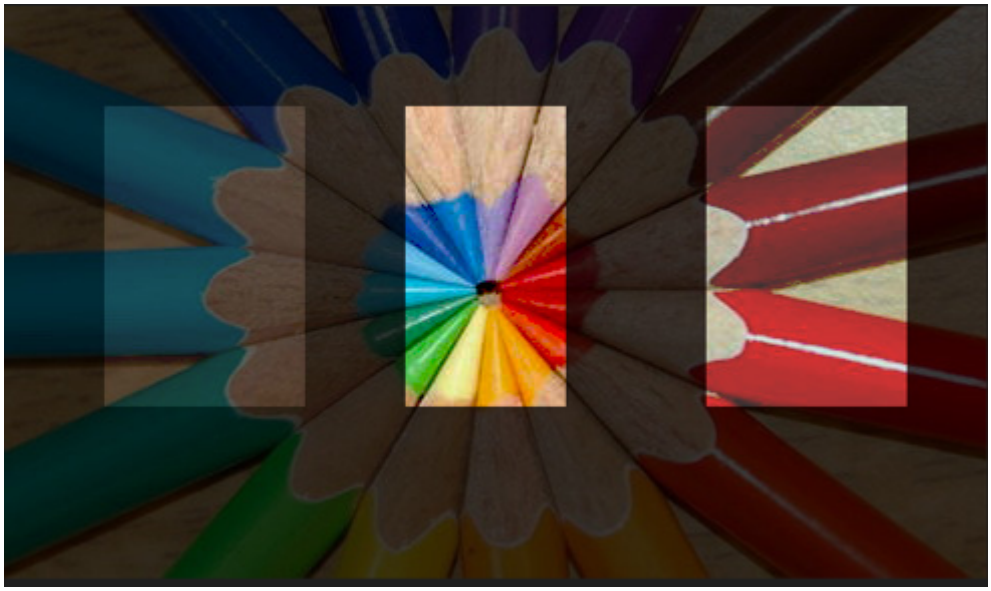

Figure 10: “rgbstretch” function. Performing histogram stretching on Color Images on RGB color space, on the 3 used ROIs . Histogram stretching was performed on all channels (R,G,B) at once. Different intensity range values were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 100, Channel(not used) = R, A1= 20, B1= 100, x2 = 50, y2 = 200, Sx2 = 150, Sy2 = 80, Channel2(not used) = R, A2 = 0, B2 = 255, x3 = 50, y3 = 350, Sx3 = 150, Sy3 = 100, Channel3(not used) = R, A3 = 50, B3 = 240.

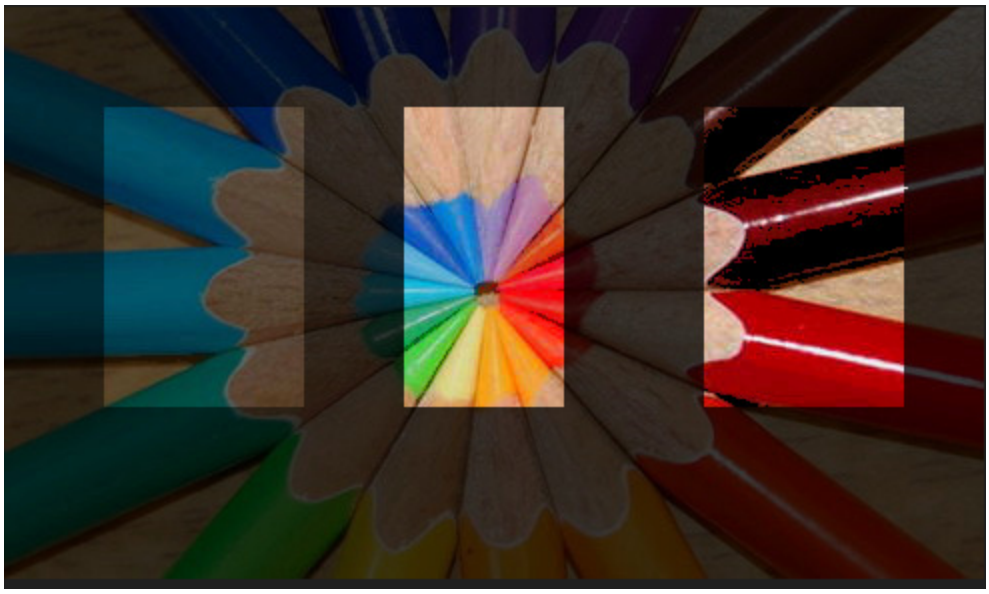

Figure 11: “istretch” function. Performing histogram stretching on Color Images on HSI color space, on the 3 used ROIs . Histogram stretching was performed on the I component of the HSI representation of the original RGB color image. The output image was converted back to RGB for display. Different intensity range values were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 100, Channel(not used) = I, A1= 20, B1= 100, x2 = 50, y2 = 200, Sx2 = 150, Sy2 = 80, Channel2(not used) = I, A2 = 0, B2 = 255, x3 = 50, y3 = 350, Sx3 = 150, Sy3 = 100, Channel3(not used) = I, A3 = 50, B3 = 240.

Figure 12: “hstretch” function. Performing histogram stretching on Color Images on HSI color space, on the 3 used ROIs . Histogram stretching was performed on the H component of the HSI representation of the original RGB color image. The output image was converted back to RGB for display. Different intensity range values were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 100, Channel(not used) = H, A1= 0, B1= 360, x2 = 50, y2 = 200, Sx2 = 150, Sy2 = 80, Channel2(not used) = H, A2 = 50, B2 = 350, x3 = 50, y3 = 350, Sx3 = 150, Sy3 = 100, Channel3(not used) = H, A3 = 50, B3 = 360.

Figure 13: “sstretch” function. Performing histogram stretching on Color Images on HSI color space, on the 3 used ROIs . Histogram stretching was performed on the S component of the HSI representation of the original RGB color image. The output image was converted back to RGB for display. Different intensity range values were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 100, Channel(not used) = S, A1= 40, B1= 80, x2 = 50, y2 = 200, Sx2 = 150, Sy2 = 80, Channel2(not used) = S, A2 = 50, B2 = 90, x3 = 50, y3 = 350, Sx3 = 150, Sy3 = 100, Channel3(not used) = S, A3 = 50, B3 = 60.

Figure 14: “fullhsistretch ” function. Performing histogram stretching on Color Images on HSI color space, on the 3 used ROIs . Histogram stretching was performed at once on the H, S, I components of the HSI representation of the original RGB color image. The output image was converted back to RGB for display. Different intensity range values were used for each ROI, and you can notice those changes based on the output. The used parameter configuration is: #roi = 3, x1= 50, y1= 50, Sx1= 150, Sy1= 100, A1_H= 0, B1_H= 360, A1_S = 0, B1_S = 100, A1_I = 0, B1_I = 255, x2 = 150, y2 = 200, Sx2 = 150, Sy2 = 80, A2_H= 0, B2_H= 360, A2_S = 0, B2_S = 100, A2_I = 0, B2_I = 255, x3 = 50, y3 = 350, Sx3 = 150, Sy3 = 100, A3_H= 0, B3_H= 360, A3_S = 0, B3_S = 100, A3_I = 0, B3_I = 255.

4. Discussion of Results

We were able to test histogram modification (histogram stretching) in both grey level and color images. Different algorithms were implemented allowing the use of thresholding, and selection of channels for histogram stretching, as well as histogram stretching in all channels. Even though the algorithms were similar in logic there were changes that are clearly reflected in their ability to enhance contrast in low contrast images. As far as histogram stretching on Grey level images is concerned, from our algorithms the "histothres" performed better in enhancing the contrast. The reason is because by allowing the use of thresholding the user is able to better optimize the parameters to boost contrast enhancement. The more adaptive an algorithm is, the better the results can be via optimization. On color images we implemented more histogram stretching algorithms and experimented with RGB and HSI color space. From the algorithms in the RGB color space the "rgbstretch" performed better in contrast enhancement. Being able to perform histogram stretching on all R,G,B channels enhanced the contrast of the image. However, we noticed that it isn't exactly the best color space to use considering that R,G,B channels have a ratio between them, and that ratio might not be preserved necessarily after stretching. That was the reason why another color space - HSI was used. Here histogram stretching was performed on all H,S,I components individually and all at once. From the results that we got we noticed that histogram stretching on only the I component in the HSI color space provided very good contrast enhancement. The contrast enhancement for the same [A,B] intensity range was better than the contrast enhancement on the RGB color space. We performed histogram stretching on the H, S components as well as on all H,S,I components but color was not preserved exactly. While there was some contrast enhancement, the original colors changed, affecting the color image's features. Based on the results, utilization of HSI color space for histogram stretching was better than RGB color space because the contrast enhancement was better.

5. Conclusion

Overall, in this assignment we were able to experiment with histogram modification and color processing. We implemented multiple algorithms that implemented histogram stretching on both Grey level and color images. Thresholding was implemented to make the algorithm more adaptive which improved results. We were able to test and compare histogram stretching on single channels/components and on all of them at once. Furthermore, we explored both RGB and HSI color spaces and were able to deduce based on results that HSI color space was better for histogram stretching. The contrast enhancement was better and more significant. Of course, the optimization of intensity range parameters is important to maximize the contrast enhancement from these algorithms. Adding thresholding and making these algorithms more adaptive can provide

better contrast enhancement results. There was a lot of learning and implementation in this assignment and provided a good opportunity to test more advanced methods of image processing and manipulation. Ultimately, this assignment provided a productive way to further improve our knowledge in Image Processing.