Homework 2: Contrast Enhancement with Histogram Modifications

Christopher Kantor

Department of Computer Science and Engineering

University of South Florida

1. Introduction and description

This assignment focuses on the implementation of image contrast enhancement techniques using histogram modifications, in both gray and color space. Contrast enhancement makes details in an image easier to see, which is useful in a variety of circumstances. The two methods of contrast enhancement used in this project involve histograms, which represent how often an intensity value appears in the image. By modifying the histogram of an image, we can change how often different intensity values appear, which we can use to increase the total range of intensity values of an image, thus increasing the contrast. We also implemented an extra version of histogram modification, which is called histogram equalization with a threshold.

The report is organized as follows. In section 2, the algorithms for contrast enhancement using histograms are described. Section 3 describes the implementation details of these algorithms. In section 4, the results of the project are displayed, and we conclude in section 5.

2. Description of algorithms

We implemented 2 image contrast enhancement techniques in this project, one uses histogram stretching, and the other uses histogram equalization. We also implemented a threshold version of histogram equalization.

2.1. Histogram Stretching

The first approach for contrast enhancement, and the most naïve, is called histogram stretching. This works by creating a histogram of an image, with min and max values being called C and D respectively. We then stretch this range of values to a new range of values called A and B (with A usually being 0 and B usually being 255). The formula used to do this for any given pixel P is as follows: P' = ((P - C) * ((B - A) / (D - C))) + A. If C and D are equal to 0 and 255, histogram stretching will have no effect, so we use the 5th and 95th percentile of pixel intensities for C and D. We can focus on low contrast ROIs of an image, but this will create a false edge between where we performing histogram stretching and the rest of the image Histogram stretching is relatively easy to implement, needing only a histogram of the original image to calculate C and D, and the formula above, but it is not the best way to enhance image contrast, and we need to supply A and B to the formula.

Histogram stretching takes O(n^2) time. Creating the histogram, finding C and D, and then performing the equation takes O(n^2) time as we need to perform these operations on every pixel. Creating the image of the histogram also takes O(n^2) time, as we need to make the histogram out of the image, which means looking at every pixel.

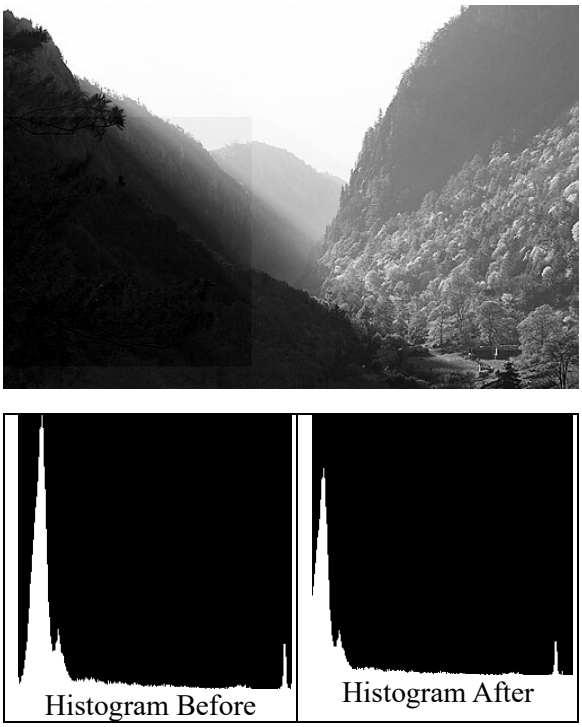


Figure 1: Histogram stretching done on an area with a wide range of intensities has little to no effect.

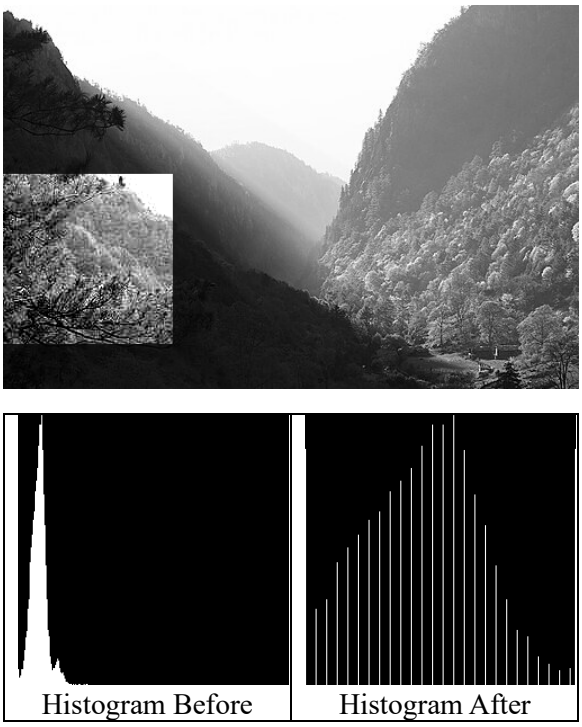
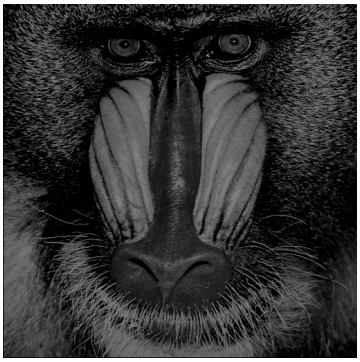


Figure 2: Histogram stretching done on on area with a narrow range of intensities, in this case dark pixels, has a much more noticeable effect.

2.2. Histogram Equalization

Histogram equalization doesn't need input for the interval and aims to create an image with equally distributed brightness throughout. It does this by increasing the contrast for the range of intensities near maxima in the histogram and decreasing contrast for pixels near a minima in the histogram. This means that intensity ranges that have a ton of pixels will receive more contrast than ranges that have very little or no pixels. This is a far more intelligent way to increase the contrast of an image, but it does not work well on images that have intensity values concentrated on one end of the greyscale. It also provides a uniformly distributed intensity range, which we may or may not want in some cases.



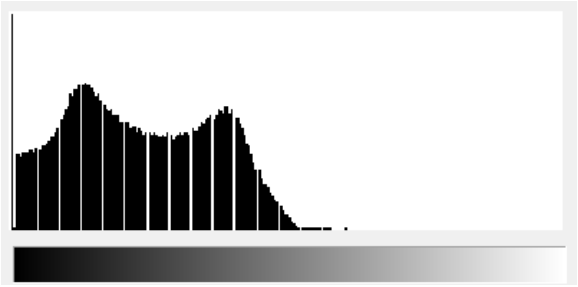
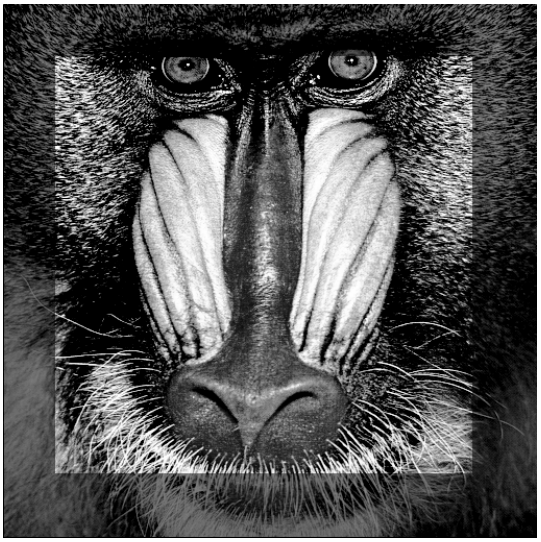


Figure 3: Histogram before histogram equalization



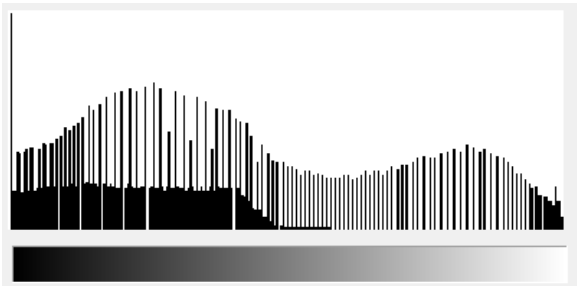


Figure 4: Histogram after histogram equalization

2.3. Histogram equalization with threshold

By providing a threshold to histogram equalization, we can set a limit to the intensity values we want to look at in the image. This is especially useful if we want to only equalize a specific range of the image.

Figure 5: Histogram equalization with threshold = 100

Figure 6: Histogram equalization with threshold = 200

2.4. Color space manipulations

Histogram stretching and equalization are easy to implement in gray images, but color images have additional things to consider. In an RGB image, we have 3 different channels, which hold the intensities of each of the 3 colors, red, green, and blue respectively. If we perform a histogram modification on any of the channels, we will completely change the color of the output image, as channels that have less representation in the image than others will see the most improvement using histogram equalization.

Since we cannot do histogram modifications on the RGB color space, we must convert to a different one. The color space we convert to is called HSI, which stands for Hue, Saturation, and Intensity. Hue is defined for 0 to 360, and represents the color of the pixel, with 0 being red, 120 being green, and 240 being blue. Saturation is how white the pixel is, with 0 being pure white and 1 meaning no white in the pixel. Intensity is how black or white the pixel is, with 0 being black and 1 being white.

We perform histogram equalization on the intensity channel, as modifying the hue channel would change the color of the image, and saturation could change the brightness of the image.

After performing histogram equalization on the intensity channel, we convert the equalized image back to RGB so that we can display it.

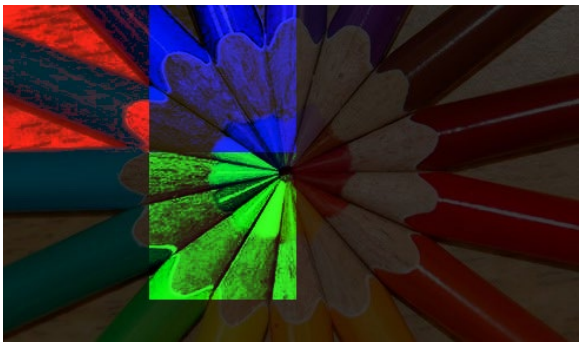


Figure 7: Performing histogram modification on each of the 3 RGB channels individually makes the image look more of that color.

Figure 8: Performing histogram equalization on all of the RGB channels together makes the image look brighter, but we lose color contrast as each of the channels is stretched.

Figure 9: Histogram equalization on the HSI channels separately leads to a change in color in the H case (top left) and no change at all in the saturation channel (below bright part). Intensity alters the brightness pixels in the ROI, without changing the color, which is what we want.

Figure 10: Performing histogram equalization on all HIS channels leads to a brightening of the image, but also a drastic change in the color.

Figure 11: Performing histogram equalization only on the Intensity channel has the most accurate result.

3. Implementation

We keep the same function call structure as the previous project, where we need to supply input/output file names, number of ROIs, rotation angle, and then each ROI defined by size, function, and parameters.

These histogram modification functions are built upon the previous projects and work on the ROI scheme.

Histogram stretching is only available on grey images and has us manually create a histogram for the image, find C and D using the 5th and 95h percentile of values, and the conversion formula mentioned above. For histogram stretching, we also implemented a function to create an image of the histogram before and after the stretch. This function gets called automatically and saves the image with a filename equal to the output filename of the original ROI, with the ROI dimensions appended after it.

Histogram equalization can be used in both grey and color space, and is implemented using the OpenCV library, which also has functions for converting between color spaces, and separating/merging the channels in an image. These functions were easy to implement conceptually as we had the functions to equalize and convert between color spaces available to us, but learning how to use the OpenCV library took a bit of time. We can perform histogram equalization on either the RGB space or HSV space and can call the function with any number of channels in each space (minimum 1).

Histogram equalization with a threshold is also only available on grey images. This algorithm works by looking at every pixel in the input image, and copying them to a temporary image temp. If the pixel intensity is greater than the threshold, we set it equal to the threshold, otherwise we keep it the same. We perform histogram equalization on temp, then we begin to write to the output file, by reading the pixel values in the input image again. If the pixel in the input image at (i,j) is greater than the threshold, we copy it over to the output image, otherwise we get the pixel at (i,j) from the equalized image.

All these functions are defined in README.txt and in utility.h/utility.cpp.

4. Description and analysis of results

The results of our algorithms used are displayed here. We have 22 images for grey, and 12 for color. We also have some miscellaneous images that show off equalizations on different channels, both in RGB and in HSV space.

For grey images:

- 1 original file
- 3 rotated versions of the original file with 3 ROIs each. Rotations are 90, 180, and 270 degrees.
- 4 images with 3 ROIs each that have been enhanced using histogram stretching, with A = 0 and B = 255. These are then rotated by 0, 90, 180, and 270 degrees.
    - For each ROI in each image, we also have 2 histograms for them, one before, and then one after the stretch.
    - I will only display the histograms for 1 images ROIs, as they are all the same between images as rotation doesn't affect the histogram.
- 4 images with 3 ROIs each that have been enhanced using histogram equalization. These are then rotated by 0, 90, 180, and 270 degrees.
- 4 images with 3 ROIs each that have been enhanced using histogram equalization with a threshold of 100. These are then rotated by 0, 90, 180, and 270 degrees.

For color images:

- 1 original file
- 3 rotated versions of the original file with 3 ROIs each. Rotations are 90, 180, and 270 degrees.
- 4 images with 3 ROIs each that have been enhanced using histogram equalization on the V component. These are then rotated by 0, 90, 180, and 270 degrees.

- 4 images with 3 ROIs each that have been enhanced using histogram equalization on the S component. These are then rotated by 0, 90, 180, and 270 degrees.

4.1. Grey Images

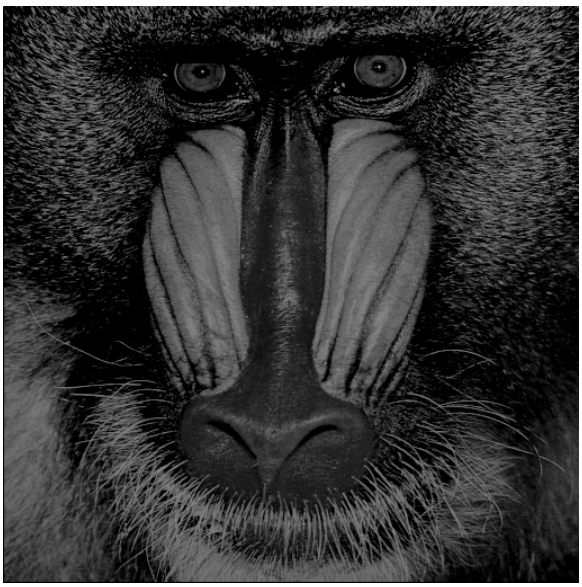
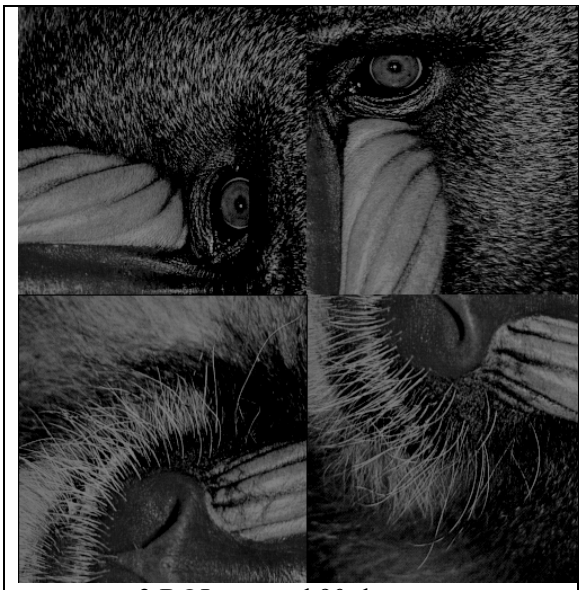
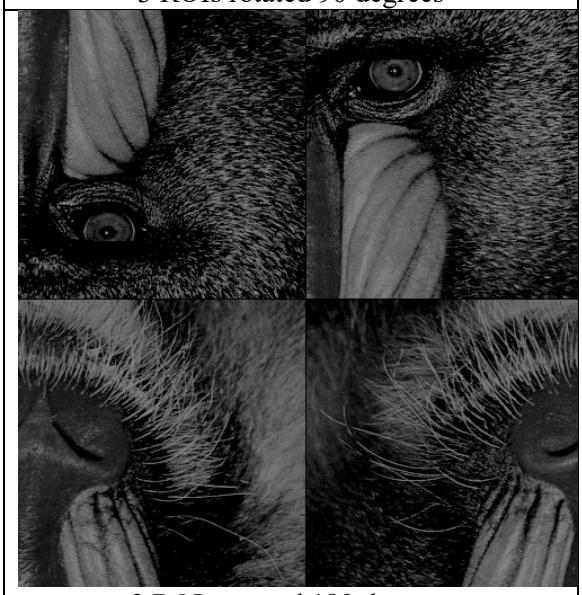


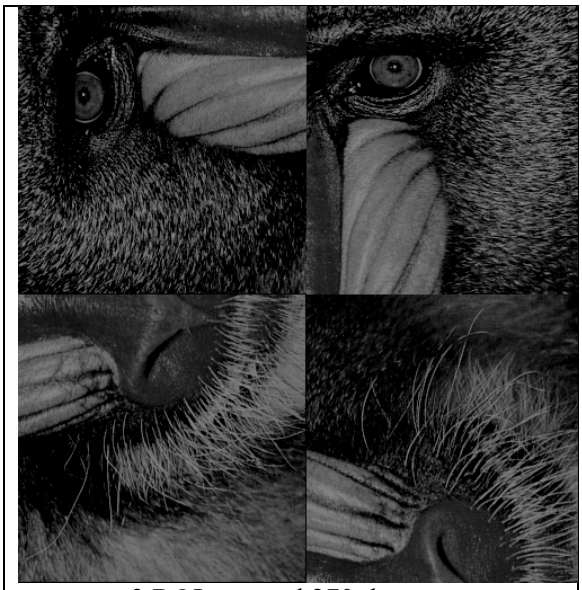
Figure 12: Original image used for greyscale manipulations.



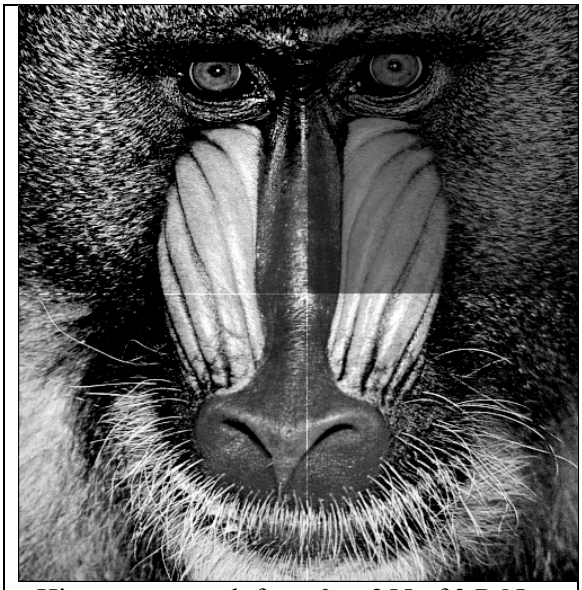
3 ROIs rotated 90 degrees



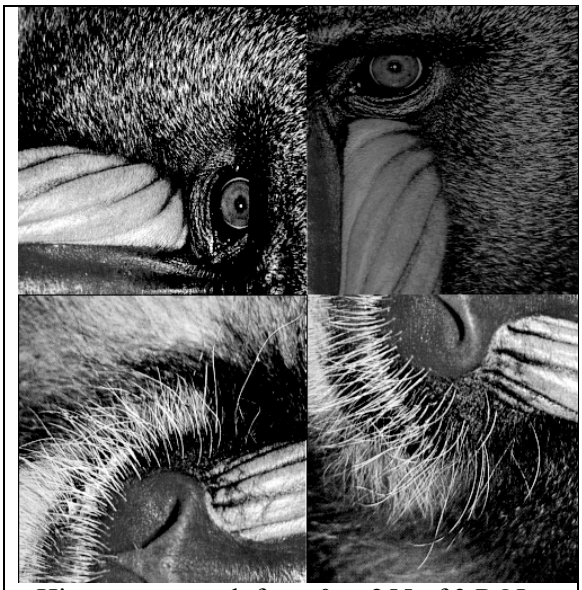
3 ROIs rotated 180 degrees



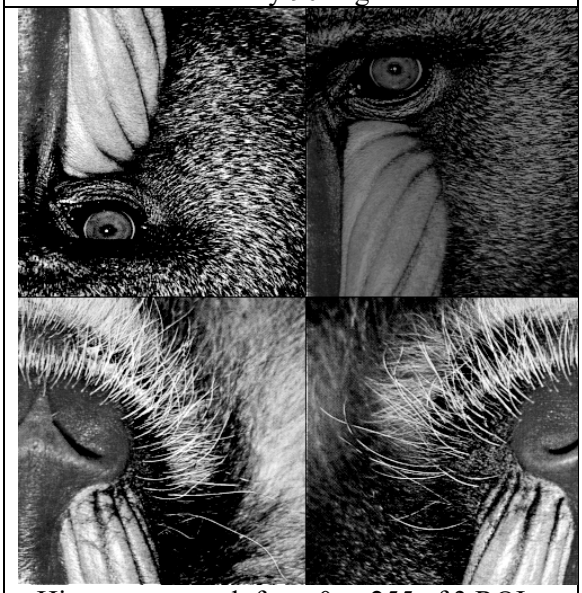
3 ROIs rotated 270 degrees



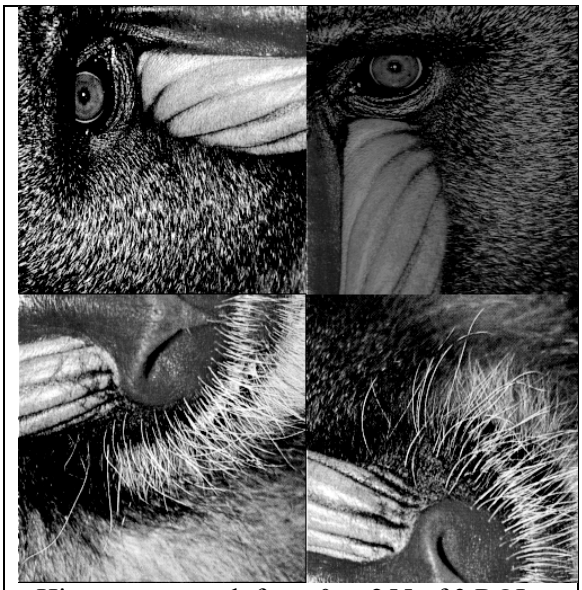
Histogram stretch from 0 to 255 of 3 ROIs, rotated by 0 degrees



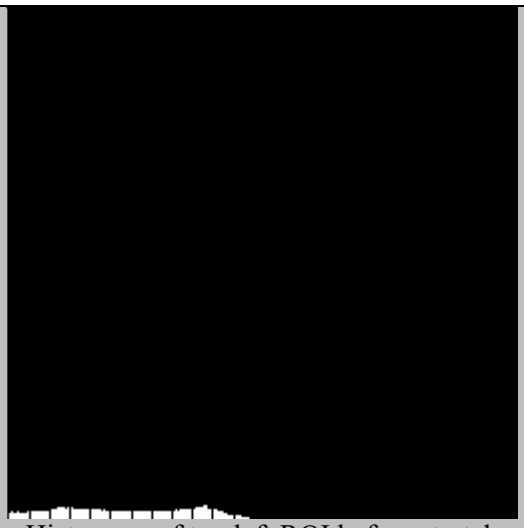
Histogram stretch from 0 to 255 of 3 ROIs, rotated by 90 degrees



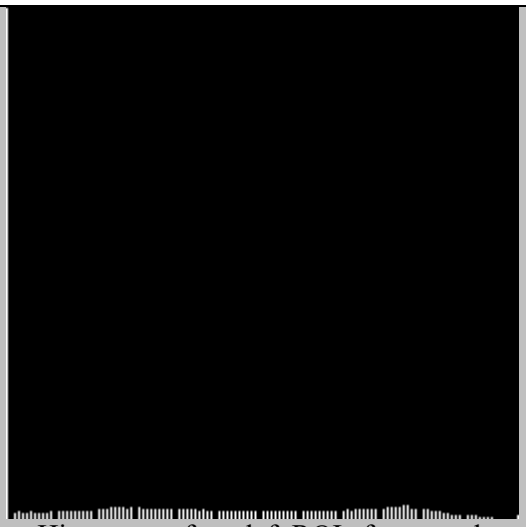
Histogram stretch from 0 to 255 of 3 ROIs, rotated by 180 degrees



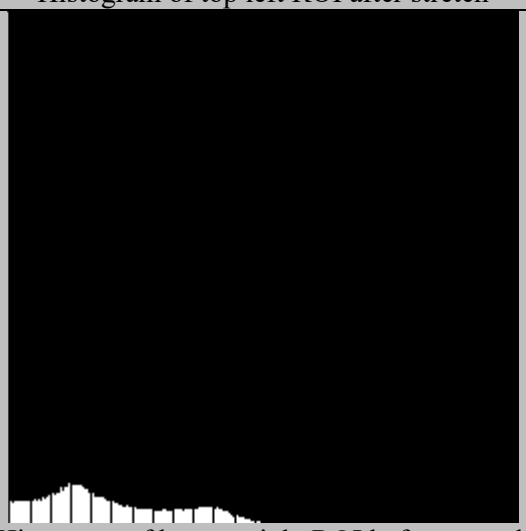
Histogram stretch from 0 to 255 of 3 ROIs, rotated by 270 degrees



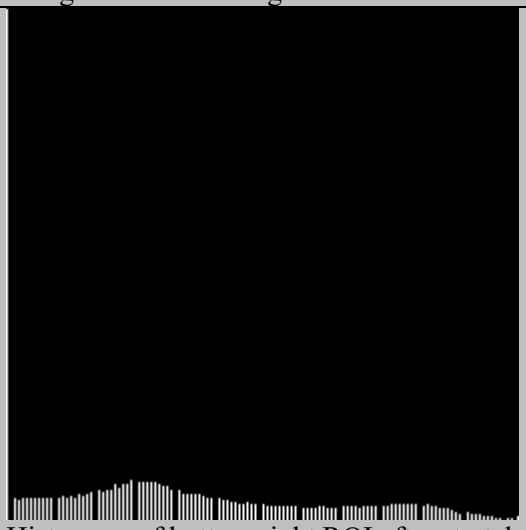
Histogram of top left ROI before stretch



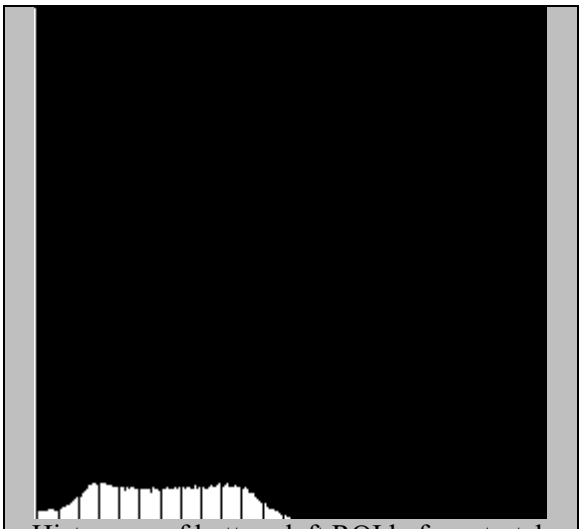
Histogram of top left ROI after stretch



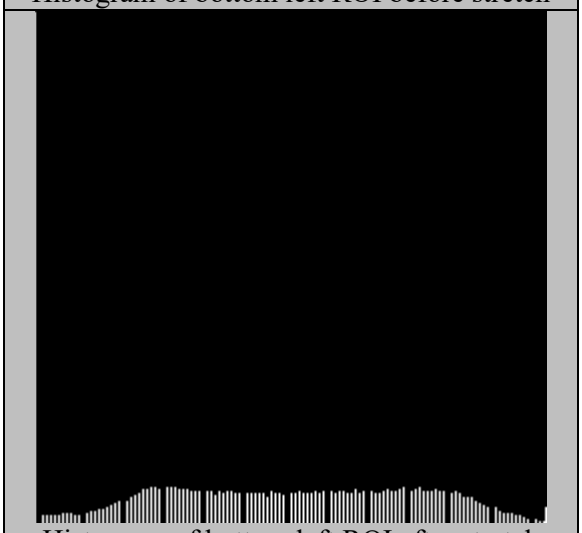
Histogram of bottom right ROI before stretch



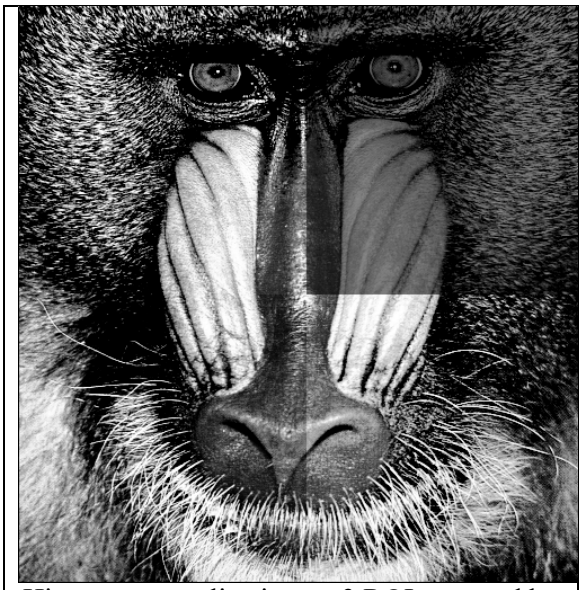
Histogram of bottom right ROI after stretch



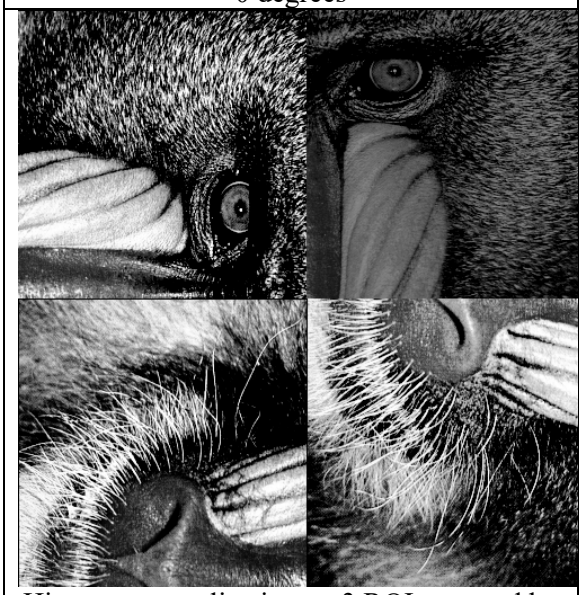
Histogram of bottom left ROI before stretch



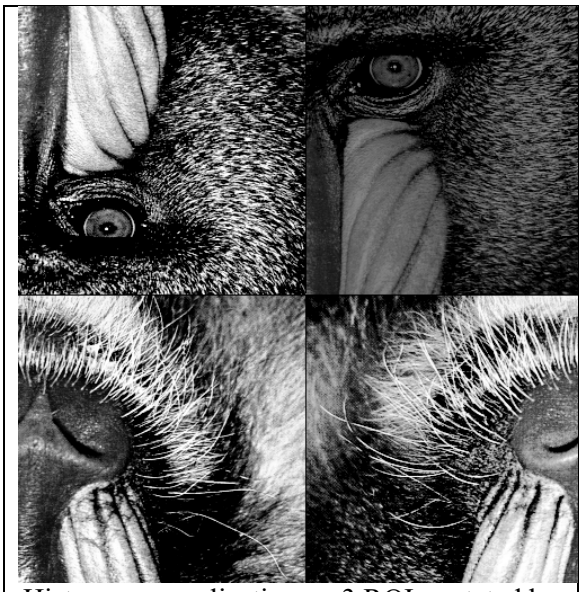
Histogram of bottom left ROI after stretch



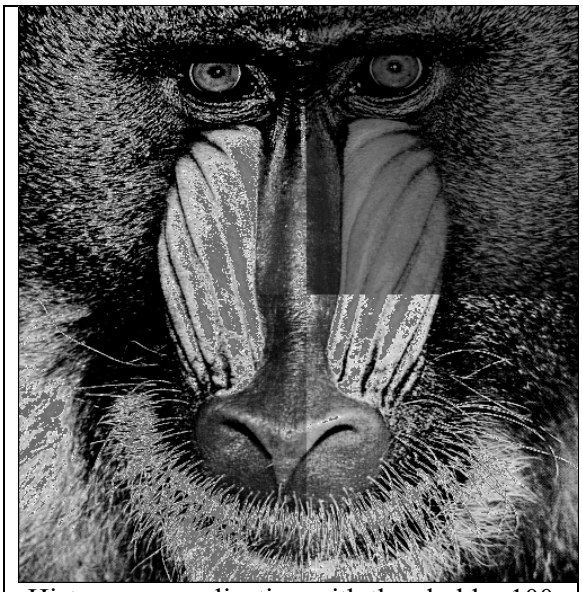
Histogram equalization on 3 ROIs, rotated by 0 degrees



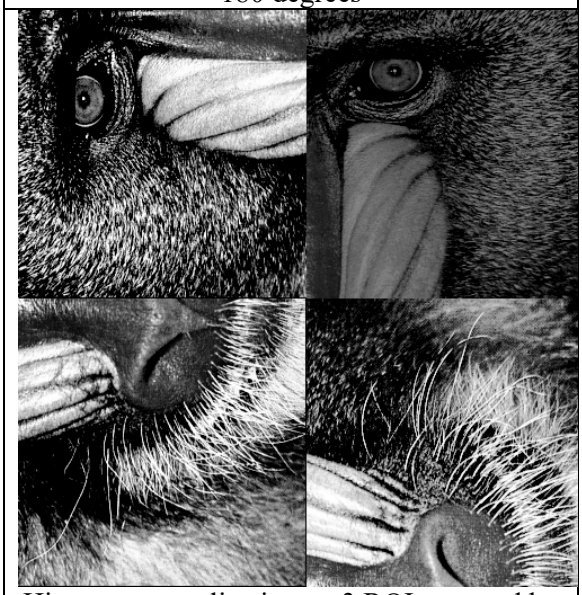
Histogram equalization on 3 ROIs, rotated by 90 degrees



Histogram equalization on 3 ROIs, rotated by 180 degrees

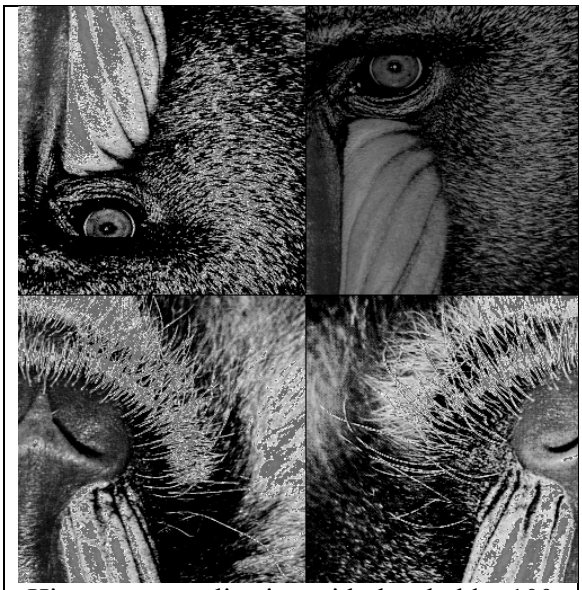


Histogram equalization with threshold = 100 on 3 ROIs, rotated by 0 degrees

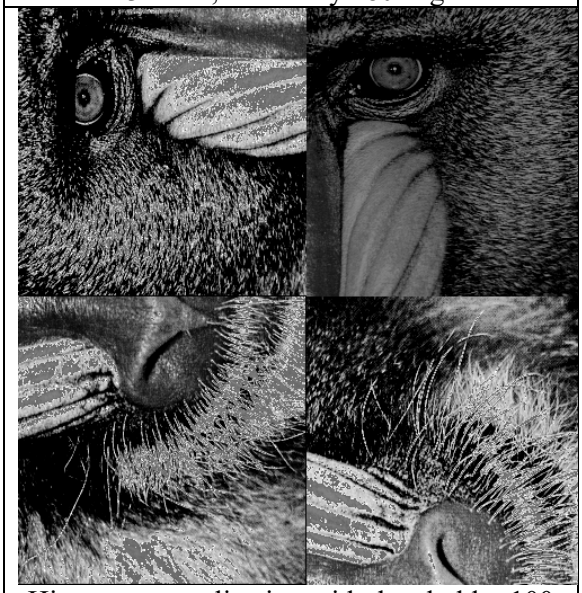


Histogram equalization on 3 ROIs, rotated by 270 degrees

Histogram equalization with threshold = 100 on 3 ROIs, rotated by 90 degrees



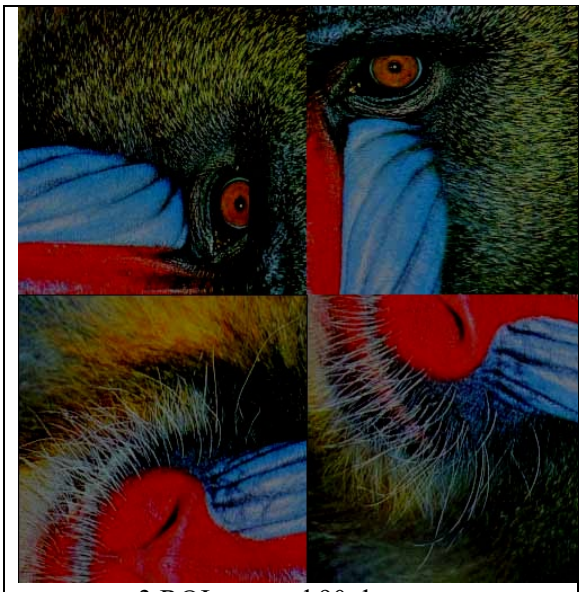
Histogram equalization with threshold = 100 on 3 ROIs, rotated by 180 degrees



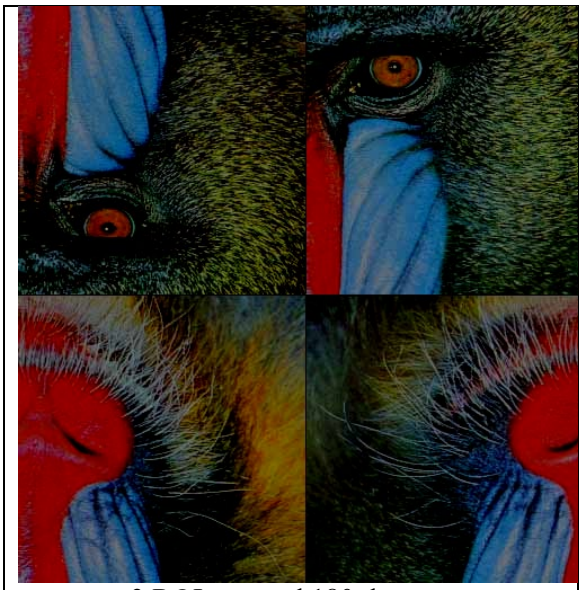
Histogram equalization with threshold = 100 on 3 ROIs, rotated by 270 degrees

4.2. Color Images

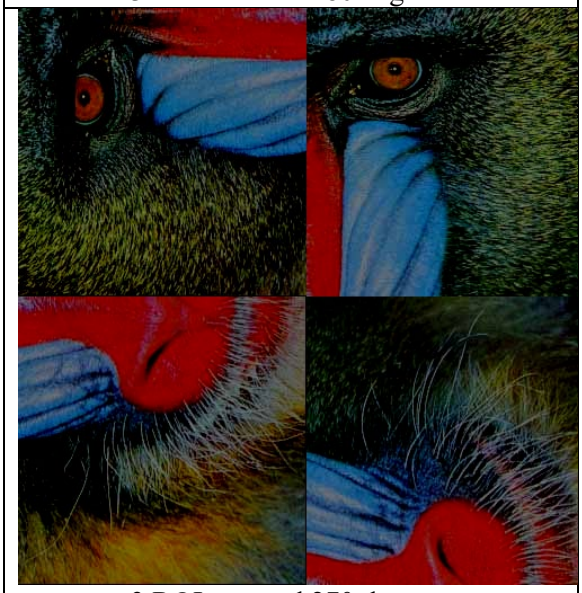
Figure 14: Original image used for all color manipulations.



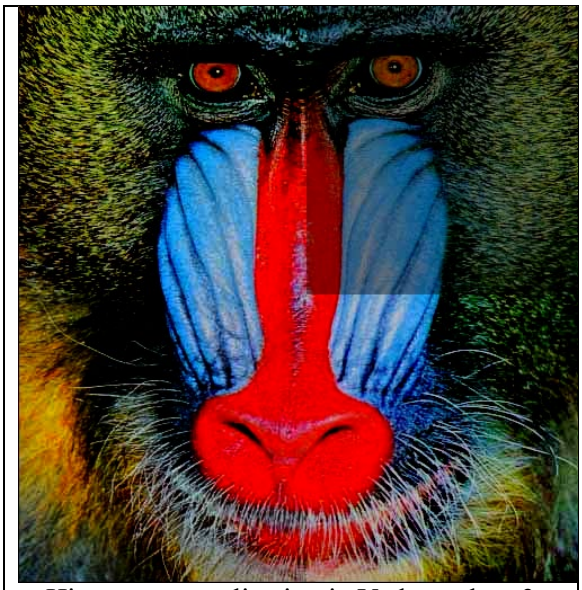
3 ROIs rotated 90 degrees



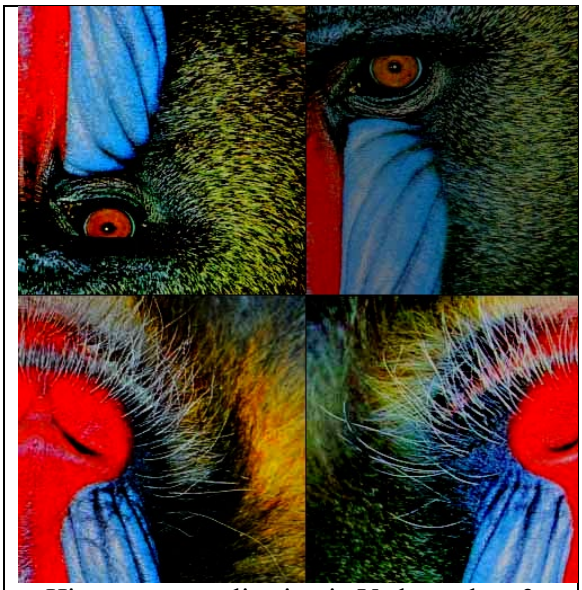
3 ROIs rotated 180 degrees



3 ROIs rotated 270 degrees

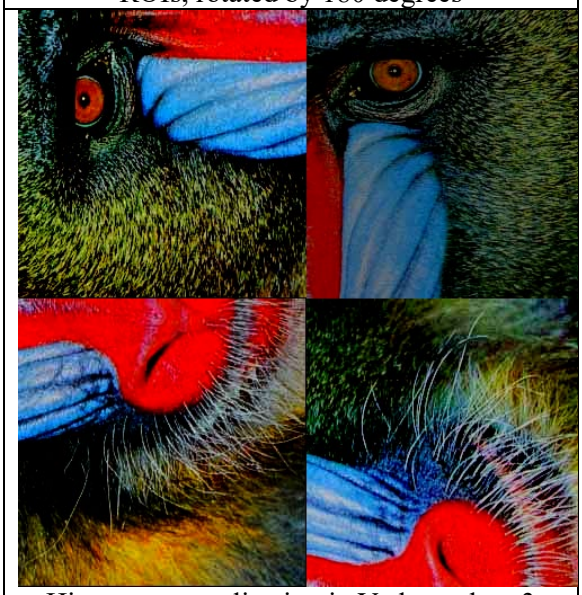


Histogram equalization in V channel on 3 ROIs, rotated by 0 degrees

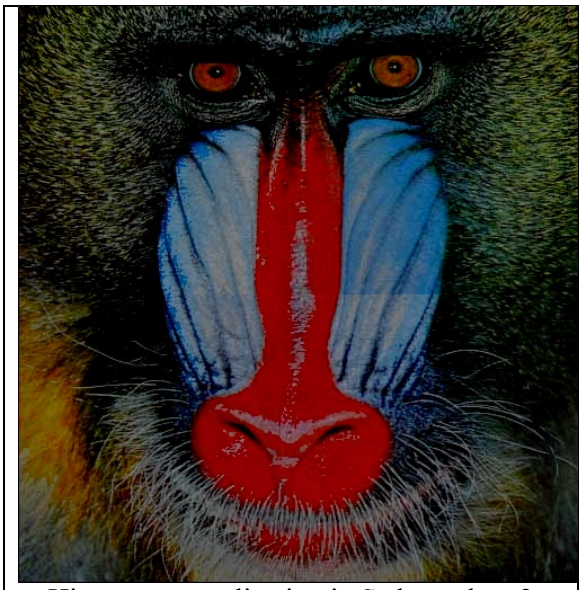


Histogram equalization in V channel on 3 ROIs, rotated by 180 degrees

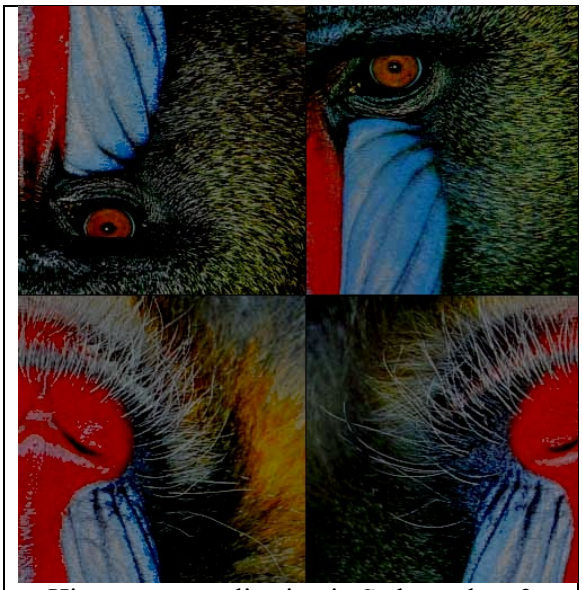
Histogram equalization in V channel on 3 ROIs, rotated by 90 degrees



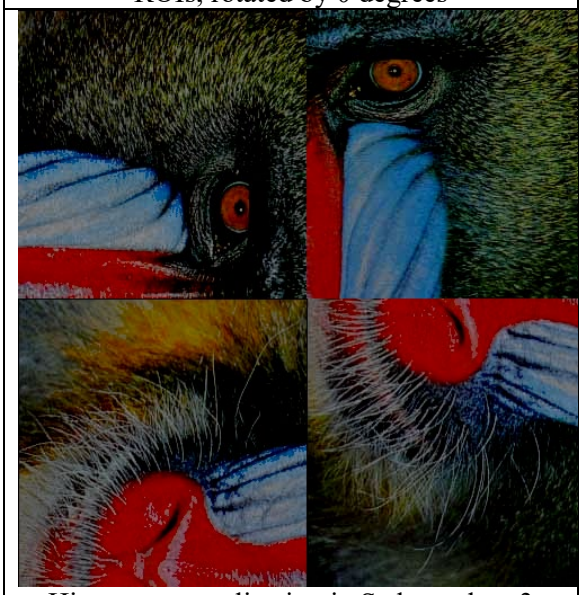
Histogram equalization in V channel on 3 ROIs, rotated by 270 degrees



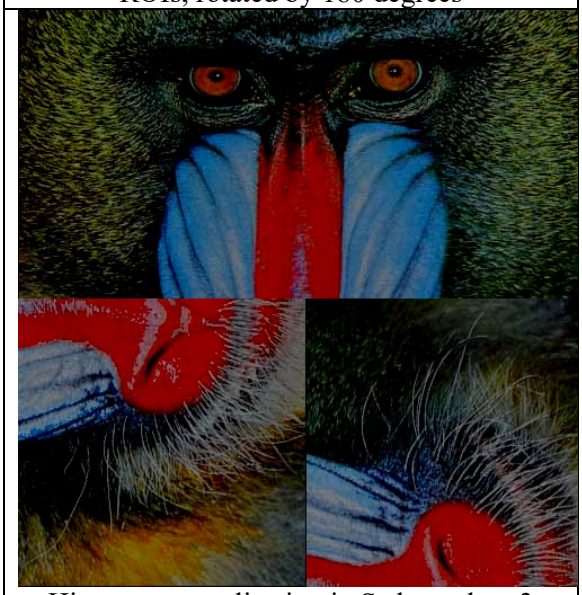
Histogram equalization in S channel on 3 ROIs, rotated by 0 degrees



Histogram equalization in S channel on 3 ROIs, rotated by 180 degrees

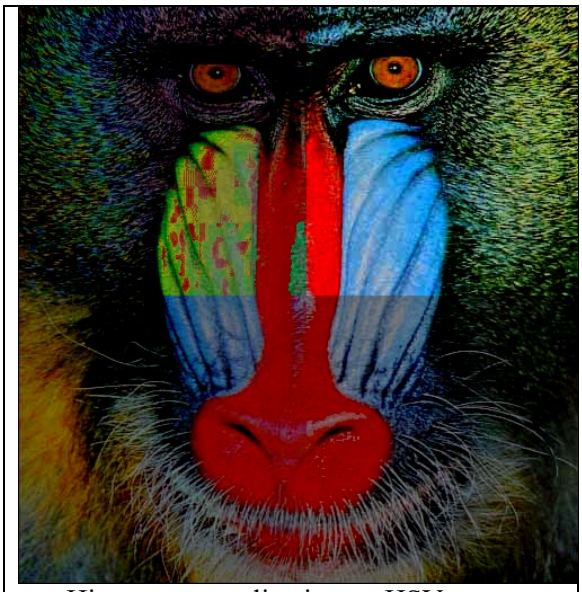


Histogram equalization in S channel on 3 ROIs, rotated by 90 degrees



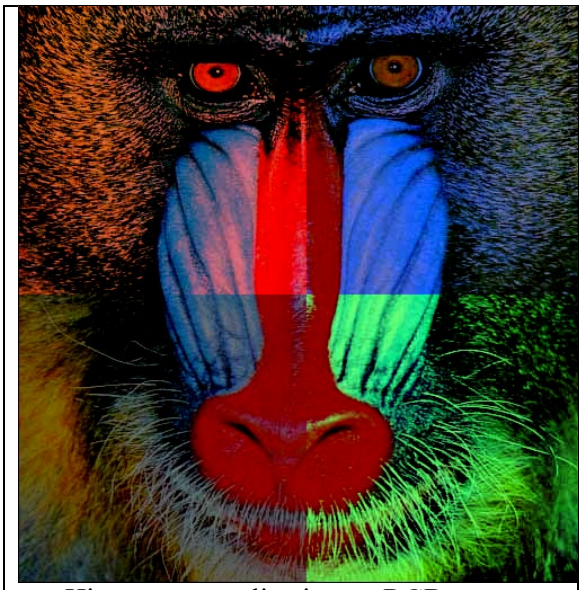
Histogram equalization in S channel on 3 ROIs, rotated by 270 degrees

4.3. Miscellaneous Images

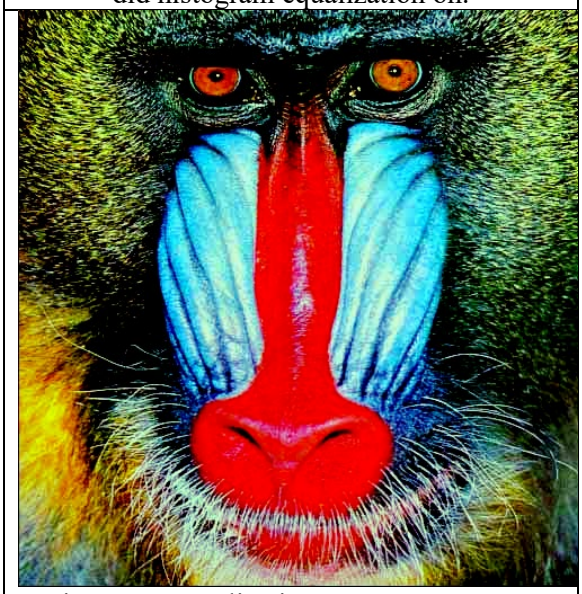


Histogram equalization on HSV space separately. H is top left, V is top right, and S is bottom right.

Histogram equalization on HSV space on ALL channels. Leads to a completely different color change in some spaces, and others turn out mostly black.



Histogram equalization on RGB space separately. Each ROI becomes far more saturated in the particular color channel we did histogram equalization on.



Histogram equalization on RGB spaces on ALL channels. Image becomes a lot brighter, and there are slight color changes.

5. Conclusion

In this project, we were tasked with implementing contrast enhancement techniques using histogram modifications. We learned about 2 common histogram modification techniques, stretching and equalization, and were able to implement them in our image processing library. We also learned how to properly enhance the contrast in colored images, by first converting it to HSI, performing the equalization, then converting back to RGB.

Some examples of how to call our functions are listed below as well as in readme.txt. All of these must be typed into a single line in parameters.txt, having newlines in the middle of a parameter call with mess up the reading of it. Examples 6 and 7 are specific to this project. These are also in README.txt.

*** EXAMPLE PARAMETERS ****

Example 1:

infile.pgm outfile.pgm 1 0 ROI 0 0 512 512 addGrey 50

// We only have 1 ROI and will not rotate it

// ROI is defined from 0,0 to 512,512, and we will add 50 intensity to every pixel in int

Example 2:

infile.pgm outfile.pgm 2 90 ROI 0 0 256 256 addGrey 50 ROI 256 256 256 256 scale 1.67

// We have 2 ROIs not, the first one is from (0,0) to (256,256) This is the top left quadrant if the image is 512 x 512

// The second is from (256, 256) to (512, 512) the bottom right quadrant

// on the first ROI, we will add 50 brightness to every pixel in it

// on the second ROI, we will scale it by a factor of 1.67

// we will also rotate each of these ROIs by 90 degrees

Example 3:

infile.ppm outfile.ppm 3 270 ROI 0 0 256 256 multColor 1.25 ROI 256 0 256 256 multColor 1.5 ROI 0 256 256 256 multColor 2

// We have 3 ROIs, and will rotate each of them by 270 degrees

// The first ROI calls multColor with a scale factor of 1.25

// The second ROI calls multColor with a scale factor of 1.5

// The third ROI calls multColor with a scale factor of 2

Example 4:

infile.pgm outfile.pgm 2 90 ROI 0 0 256 256

addGrey 50 ROI 256 256 256 256 scale 1.67

// This is incorrect as there is a newline before all of the ROIs get read.

// This will cause an error

Example 5:

infile.pgm outfile_1.pgm 1 90 ROI 0 0 256 256 addGrey 25

infile.pgm outfile_2.pgm 1 270 ROI 256 256 256 256 scale 1.56

// This has two full parameters, and will create 2 separate images, outfile_1 and outfile_2

// each one is processed separately

Example 6:

```
infile.pgm outfile.pgm 1 0 ROI 0 0 256 256 histoEqual          //incorrect

//even though histoEqual doesn't require a parameter,
//this will result in an error as there is nothing after the operation call.
//we can fix this by putting in ANY character after histoEqual like so:

infile.pgm outfile.pgm 1 0 ROI 0 0 256 256 histoEqual 0          //correct
```

Example 7:

```
infile.ppm outfile.ppm 1 0 ROI 0 0 256 256 histoEqualRGB r g b      //incorrect
infile.ppm outfile.ppm 1 0 ROI 0 0 256 256 histoEqualHSV h s v      //incorrect

//no spaces are allowed in the channel arguments, do this instead

infile.ppm outfile.ppm 1 0 ROI 0 0 256 256 histoEqualRGB rgb      //correct
infile.ppm outfile.ppm 1 0 ROI 0 0 256 256 histoEqualHSV hsv      //correct

//case and position do not matter, so the following are correct as well
infile.ppm outfile.ppm 1 0 ROI 0 0 256 256 histoEqualRGB GBR      //correct
infile.ppm outfile.ppm 1 0 ROI 0 0 256 256 histoEqualHSV vHs      //correct
```