Antonio Rios
alr150630

Creating each program was anything but trivial. For much of the time, I was unsure about if I was implementing them correctly. As I am writing this, I can safely say I managed to get histogram equalization using the OpenCV function and linear stretching in the LUV color space working. For everything else, I am less certain about if they were implemented properly. Histogram equalization using the OpenCV function and linear stretching in the XYZ color space I am almost certain are working properly, but I still have some doubt. Histogram equalization using the equation we learned in class I am almost certain is not implemented properly, however.

As I coded each program, I did indeed run into some strange behavior. If you were to run the provided code, you would not see the anomalies, however. I have hidden the anomalies by forcing all linearly stretched or histogram equalized values to be 255 if they were higher than that or 0 if they were lower than that. But if you desire to generate an image with an anomaly, you can simply comment out the if statements in the _HistogramStretch function. The gist of the strangeness that occurred is that, in the user defined area, when linear stretching or histogram equalization is applied, certain areas of the image will become almost neon green, completely black, or a washed-out sort of white among other colors. Looking into it, when the linear stretching and histogram equalization formulas are applied, it is possible for a value higher than 255 to be calculated. As mentioned in the project instructions, said value turns out to be the cause of the glitches. An image of this occurring that was produced by "luv_lscl.py" can be seen directly below.

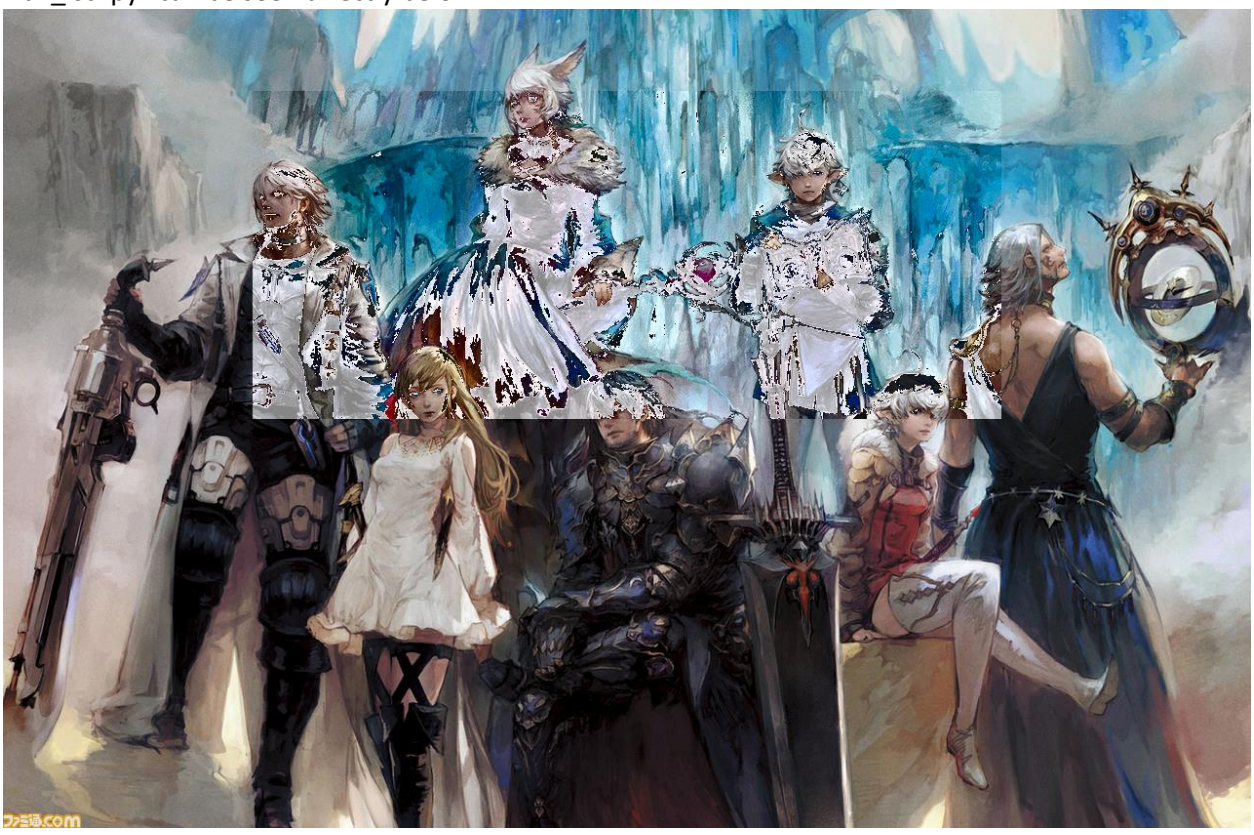

*Figure 1 – Image with the anomaly present due to commented out code that clamps out of bound values.*

Antonio Rios
alr150630

The benefits of applying linear scaling in the LUV color space seems to be that for images like figure 2 and 3 below, it will make the image visibly darker. They also seem to keep the look of the image relatively intact. By this I mean that there is not anything like a filter of red over the user-defined area.

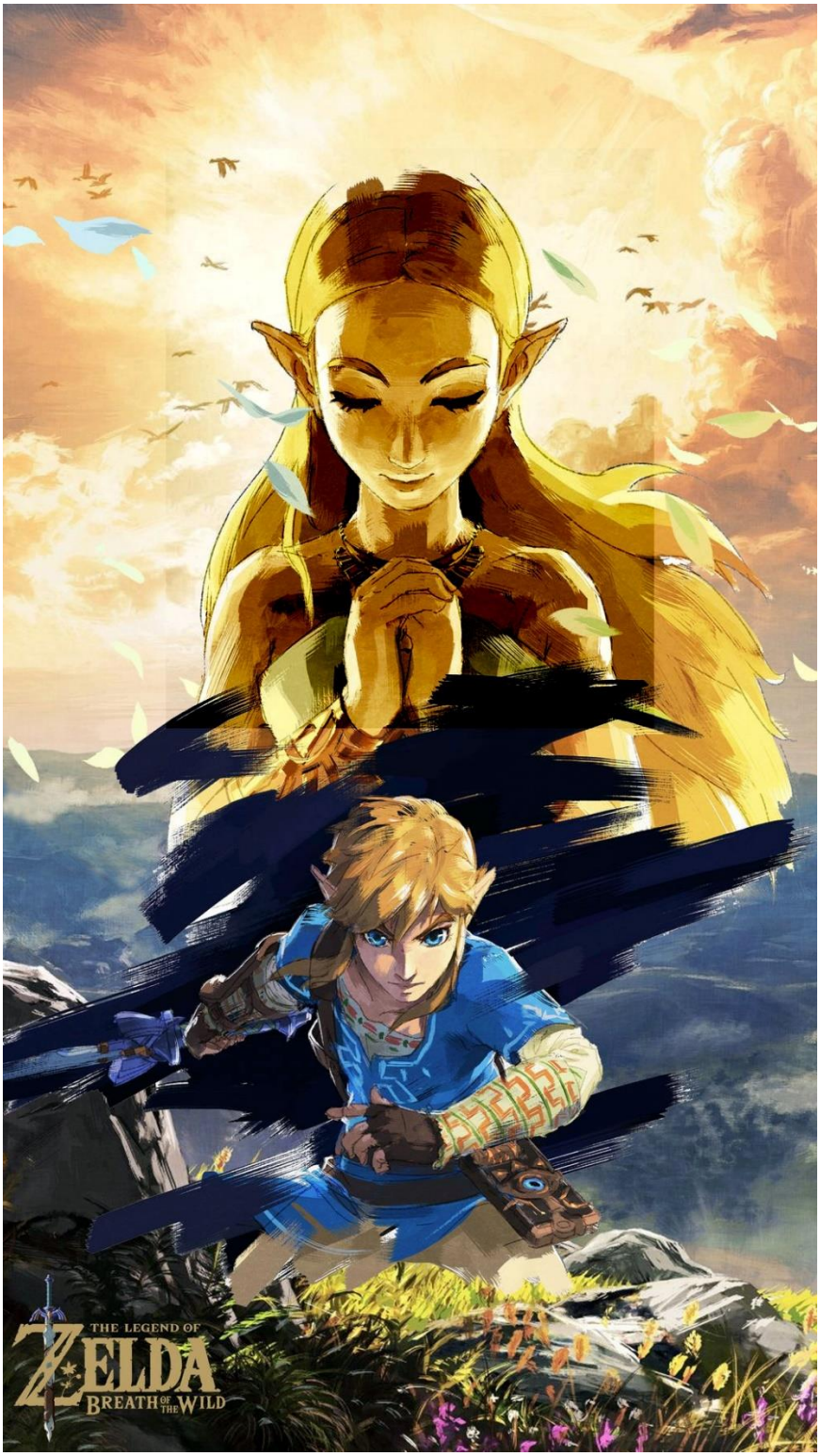


*Figure 2 - Linear scaling applied to an image in the LUV color space.*

*Figure 3 - Another image with linear scaling applied in the LUV color space.*

When linear scaling is applied to an image in the XYZ color space, the benefits are not readily apparent. While it does make the hard to make out parts of the image easier to see, it ruins the image by adjusting brightness using a non-black or non-white color. This result is most evidently seen in figures 4 and 5 below.

*Figure 4 - Image with linear scaling applied in the XYZ color space.*

*Figure 5 - Another image with linear scaling applied in the XYZ color space.*

Of the two programs, there is no question as to which one is the better one. The program called "luv_lscl.py" is the better of the two. It makes an image brighter or darker using either white or black rather than with pinks or greens. Thus, the worst of the two programs is "xyz_lscl.py" because, as I mentioned before, it dramatically alters the image when it adjusts brightness using non-white or non-black filters it applies to the user-defined area.

As for which of the histogram equalization programs is the best, there is no question that it is the program called "luv_histeq.py". It increases or decreases the brightness of the image without applying a non-black or white filter over the user-defined area. Furthermore, it does not seem to go overboard with its histogram equalization. This is likely due to the implementation of the OpenCV

function for histogram equalization. Below are two example images labeled figure 6 and 7 respectively that “luv_histeq.py” produces.

*Figure 6 - Histogram equalization applied in the LUV color space.*

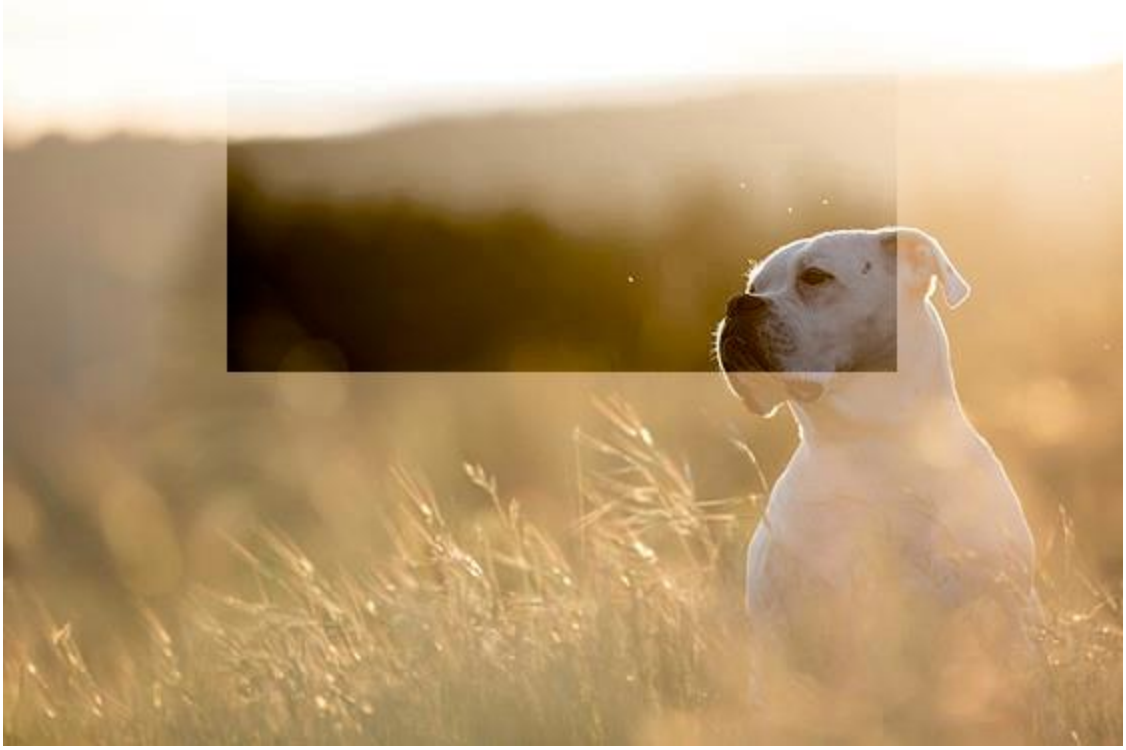

*Figure 7 - Another histogram equalization applied in the LUV color space.*

As for which is the worst of the programs, the rest are all very bad for a variety of reasons. However, in my case, only one program is significantly worse than the rest. That would be “xyz_histeq.py”. Below I have two images labeled figures 8 and 9 respectively that show just why I say that program is the worst.

*Figure 8 - Histogram equalization applied in the XYZ color space.*

*Figure 9 - Another histogram equalization applied in the XYZ color space*

In the case of this program, it seems to go overboard with the histogram equalization. The original images of figures 8 and 9 had no visible pixilation. However, by applying the algorithm, it somehow caused pixilation to occur. That coupled with the non-white or black filter that gets placed over the user-defined area makes “xyz_histeq.py” truly the worst of the histogram equalization programs.