Università della Svizzera italiana

Image and Video Processing SP 2024

Assignment 1: Point Operation

Liu Peiyu
Xiong Haibo

Faculty Of Informatics
March 20, 2024

1 Tone mapping & Linearization

In this part, we perform image processing on an image of a Ferrari to adjust its brightness and contrast. Initially, we converted the image to a double precision floating-point format with values normalized between 0 and 1. Then we invert the gamma correction to linearize it and then make it brighter by multiplying it by a factor of 1.6. To enhance the contrast, the brightened image is raised to the power of 1.1. Finally, to correct the gamma and return the image to a displayable format. The result is shown in Figure 1.

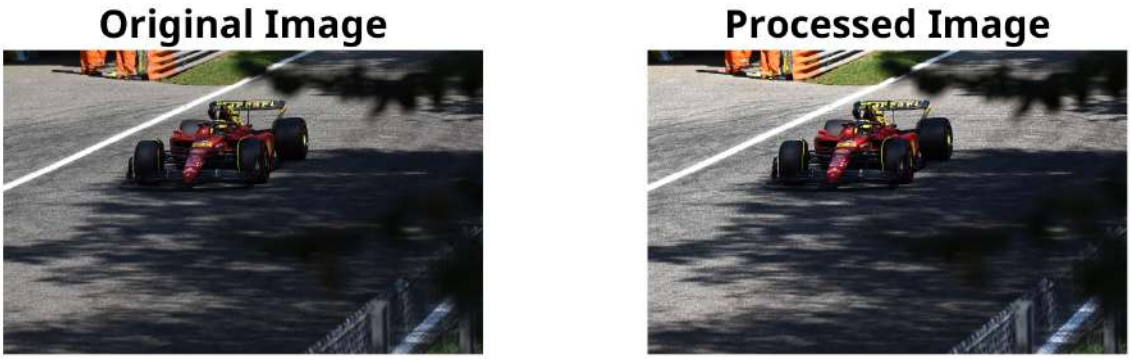


Figure 1: Tone mapping & Linearization result

2 Color correction

2.1 Pixel-based correction

As for pixel-based correction, we firstly ask the user to pick a gray point of picture, then we retrieve the RGB value of this pixel. Then we calculate the gains for each color channel (R, G, B) through dividing an overall mean value (a scalar average of the RGB means) by the RGB values of the selected gray pixel. After the gains are calculated, we apply these gains by linear transformation to each pixel in the image adjusts the color balance. The result is shown in Figure 2.

2.2 Gray-world assumption

Instead of picking a gray point to calculate the gains, with gray-world assumption we directly use the mean RGB value of the whole picture to get the gains and then apply them to each pixels of the picture. The result are shown in Figure 2.

In Figure 3 we experiment the result with gamma correction and also multiply a factor to increase the intensities in Figure 4.

Figure 2: white balance correction result

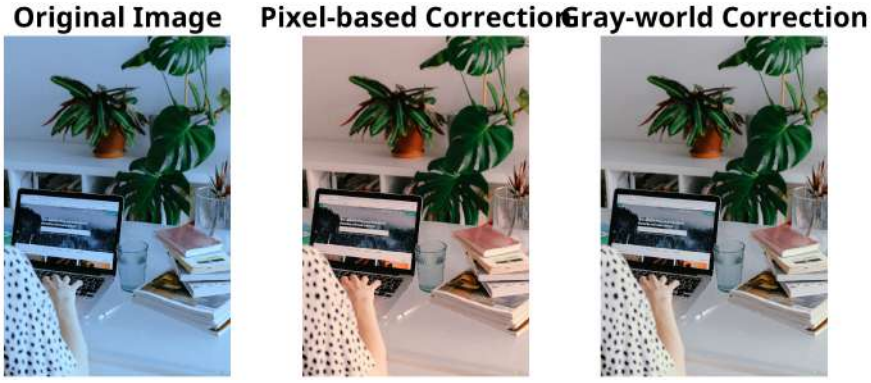


Figure 3: white balance correction result with gamma correction

Figure 4: white balance correction result with linear multiplications

3 Histograms

In this section we simply count the pixels corresponding to each intensity for different channels, then we draw them. (We also calculate the cumulative distribution and plot them in graph) The result is shown in Figure 5.

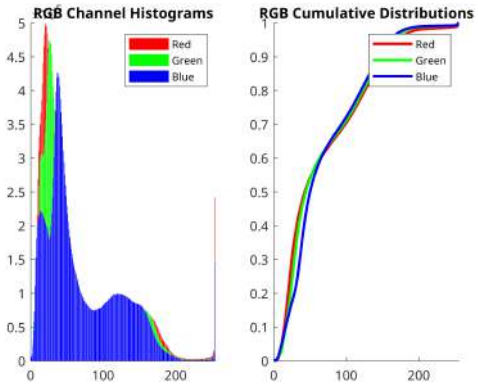


Figure 5: Color histogram and cumulative distribution of Ferrari picture

4 Histogram Equalization

4.1 Global Histogram Equalization

In this part, we have implemented the algorithm for global histogram equalization. By converting the img from RGB to HSV, we can just focus on adjust the brightness.

$$T(x) = \left(\frac{L-1}{N}\right) \sum_{m=0}^{x} H(m)$$

By calculating the cumulative distribution of the V-channel histogram, we get the $\sum_{m=0}^{x} H(m)$ for each intensity. Then according to the above discrete form of Normalize cumulative histogram function we could calculate the equalized value of V-channel of each pixel. Then we could assign back the V-channel and get the equalized picture. The result is shown in Figure 6.

4.2 Local Histogram Equalization

The algorithm for local histogram equalization divides the image into square windows of size `windowSize`. It applies the `global_hist_eq` function to each window

(a) Original Figure

(b) Global Histogram Equalization

Figure 6: Global Histogram Equilibrium

for global histogram equalization. Then, it stitches the equalized regions together to generate the final equalized image. Larger window sizes provide smoother effects in local histogram equalization but may lead to loss of details; conversely, smaller window sizes can better preserve image details but may increase computational complexity and produce boundary effects.

In this experiment, We chose 50, 200, 2000 as the window sizes. From the results, small window does a better job of enhancing image detail, but it may also introduce a lot of noise, which can lead to over-enhancement of the image. A large window, on the other hand, although it may not enhance details as well as a small window, it introduces relatively little noise.

However, noticeable boundaries exist between the equalized parts. We attempted to mitigate these boundaries using mean filtering techniques, but the results were unsatisfactory. The result is shown in Figure 7.

4.3 Locally Adaptive Histogram Equalization

We implement adaptive local histogram equalization in this part. Similar to local histogram equalization, it extracts a window of size windowSize centered at each pixel instead of dividing the image into fixed-size windows, and applies global histogram equalization to each window. This allows it to adapt more flexibly to the contrast of different regions in the image.And it also mitigate the noticeable boundaries in local equalization. Because the calculation complexity is increased fast with the size of window, we chose 20 to do the equalization and the result is shown in Figure 8.

(a) Original Figure

(b) Local Histogram Equalization(windowSize:50)

(c) Local Histogram Equalization(windowSize:200)

(d) Local Histogram Equalization(windowSize:2000)

(e) Mean Value Filter

Figure 7: Local Histogram Equalization Results

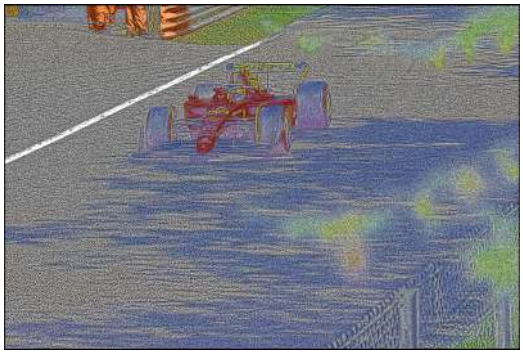


Figure 8: Adaptive Local Histogram Equalization

5 Matting

As for this section, we obtain the alpha mask by manually setting the threshold (Multiple point selection experiments) and filtering the cat picture, and then based on this mask we composite the pictures.To ensure our filters can better handle the boundary transition colors, we choose to convert the image from RGB to HSV.

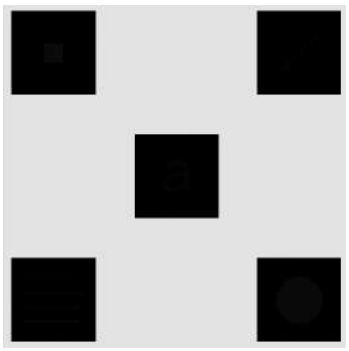
According to the given cat picture we could see that the background color is a type of green and the cat is black and white, so that we decide to set two filters for creating the mask, one filter is used to filter the parts that transition from green to white, while another filter is used to filter the parts that transition from green to black (of course, both filters will also filter out the primary color of the background). By using these two filters, we can improve the effect of handling at the boundaries of the cat. In fact we could add more filters to improve the performance, but the increase number of filter also increase the complexity of the program. Also we use gauss filter to remove the hardness from the boundary to improve the performance around the edge. The result is in Figure 9.

Figure 9: one filter with RGB (up) and two filters with HSV (down)

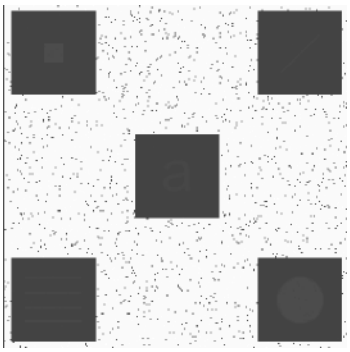
6 Bonus

6.1 Equalization

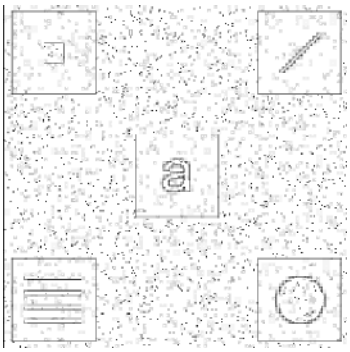
To make the effect of our equalization function more obvious, we found a very interesting picture from the internet. The original image and the processed image are as follows



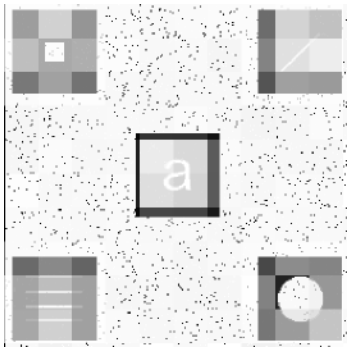
(a) Original Figure



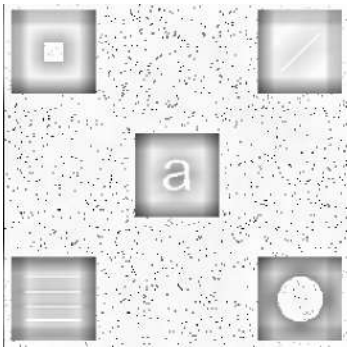
(b) Global Histogram Equalization



(c) Local Histogram Equalization(windowSize:5)



(d) Local Histogram Equalization(windowSize:50)



(e) Adaptive Local Histogram Equalization(windowSize:50)

Figure 10: Local Histogram Equalization Results for test.png

As you could see in Figure 10, in original picture and global histogram equalized picture, the local details (in black square) are not significant. But in local histogram equalized picture and adaptive local histogram picture, these details are amplified and adaptive histogram picture has better performance.

6.2 Matting

Instead of picking a cat, this time we take a dog and do all the matting work again. The result is in Figure 11.

(a) Background Picture

(b) Dog

(c) A New Huge Dog on Snowy Mountain

Figure 11: Another matting