Image & Video Processing: Assignment 1

Kyla Kaplan
March 2024

Contents

1 Point Operations

1 Tone mapping & Linearization

The assignment instructs to utilize the image `ferrari.jpg`, and to gamma correct in order to revert the already-linearized intensity for viewers. As such, in this task I inverted the operation of gamma correction by using the function `gammacorrection.m`. Following this, I increased the brightness by a factor of 2 in a multiplicative fashion. Similarly, I applied contrast to the image although in an exponential form.

```
gamma = 2.2;
alpha = 2;
beta = 0.6;
```

In the figure below are the displayed results. Driver code is included in `task1.m` in the `task1` folder.

Figure 1: `ferrari.jpg`

2 Color correction

In this task we were instructed to create 2 different white-balancing methods. For the pixel-based correction we manually choose a pixel, and average the rest of the image according to the "grayness" of the chosen pixel. For the "gray-world assumption" implementation, we assume the average is gray, and we balance the channel-mean instead. Below are the results visualized side-by-side (all related code can be found in the `task2` folder.

Figure 2: Side-by-side in order: `white_balance.jpg`, Pixel-based approach, Gray-world assumption

As we can see, the gray-world assumption approach is able to mitigate the "coldness" of the original images blue-ish tint. For the pixel-based implementation, I wanted to be able to highlight the pixel chosen, so I manually save the interim in a separate file, as can be seen slightly below.

Figure 3: Pixel chosen (highlighted red) for the "pixel-based approach"

3 Histograms

Histograms are useful to show the distributions of the different pixels in the channels. Below is a majority of the code for the implementation of the histogram (the code can be found in the `task3` folder, in the `task3.m` file). The histogram can be seen further below. A confusion I ran into is attempting to map the values on the x-axis to the appropriate `0:255` axis.

```
...
% Separate the colors into channels
R = I_double(:, :, 1);
G = I_double(:, :, 2);
B = I_double(:, :, 3);

[counts_r, bins_r] = histcounts(R(:), 255);
[counts_g, bins_g] = histcounts(G(:), 255);
[counts_b, bins_b] = histcounts(B(:), 255);

% Plot
figure;
bar(bins_r(1:end-1), counts_r, 'r', 'FaceAlpha', 0.3);
hold on;
bar(bins_g(1:end-1), counts_g, 'g', 'FaceAlpha', 0.3);
bar(bins_b(1:end-1), counts_b, 'b', 'FaceAlpha', 0.3);
hold off;
title('Histogram');
legend('r', 'g', 'b');
```

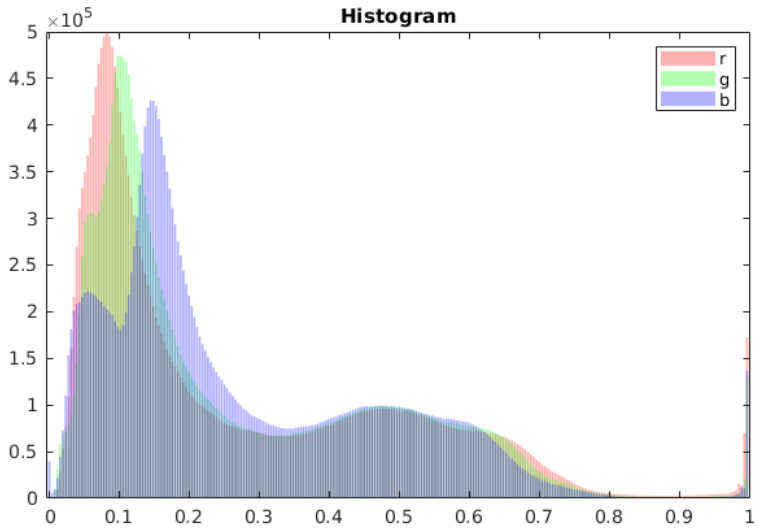

Figure 4: Histogram

4 Histogram Equalization

For this part of the assignment, when it comes to equalizing the histogram, firstly I converted the image space to the HSV space (as the assignment suggests). I extract the "V" (value) channel; determine it's size and width, and then utilize the method `imhist` to build a histogram for the un-touched image there (it is exact to the previous task).

Firstly, I approach the global equalization method, by taking the previous histogram, calculating the cumulative distribution function (CDF) for it (via `cumsum`), and then normalizing it to the previously gathered height and width. I use the `interp1` to linearly interpolate the the normalized CDF at the "points" of the original value channel; resulting in an equalized value channel which contains all the pixel intensities post-global histogram equalization (also scaled).

Proceeding to the local histogram equalization, I utilize the MATLAB function `adapthisteq`, with a "window" setting of `5`, to go over the image in 4x4 chunks. Retrospectively, the resulting image contains a substantial amount of noise and some abrupt intensities. I believe I should have manually approached this part of the problem by building a function that will go over the image in said tiles, and globally equalize each one.

Below you can find the original image, globally and locally equalized along with their respective

histograms. All the code can be found in `task4.m`.

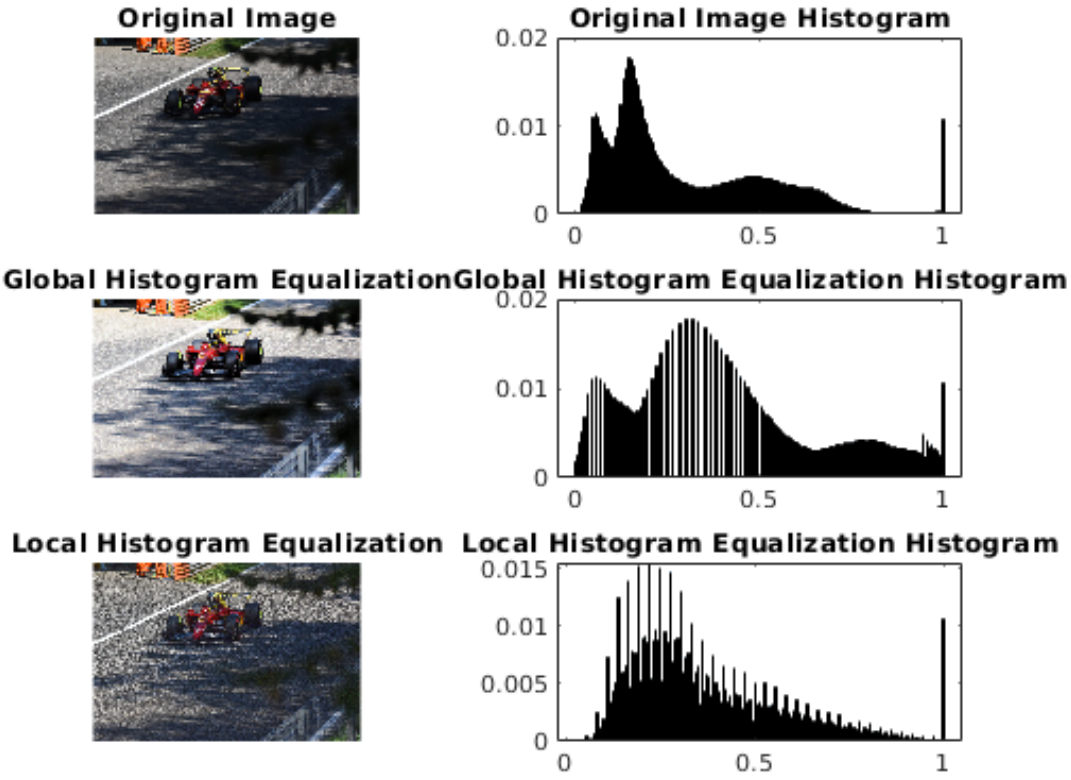

Figure 5: Original image, alongside globally and locally equalized versions, with histograms

5 Matting

For this task, I was unable to successfully utilize the threshold without modifying more significant parts of the image if using a selector for said pixel. Below is the code implementation for the chroma key followed by the results. All other materials can be found in `task5.m`.

```
function img = chroma_key(img, background)
    figure, imshow(img);
    [x, y] = ginput(1);
    key = img(round(y), round(x), :);
    mask = sum(abs(double(img) - double(key)) < 13, 3) == 3;
    mask = cat(3, mask, mask, mask);
    img(mask) = background(mask);

    imshow(img);
end
```

Figure 6: Matting of `cat.png`, based on a selected pixel