In this activity, we were tasked to enhance color images via three (3) different automatic white balancing techniques, namely: **contrast stretching**, **white patch algorithm**, and **gray world algorithm**. The image that was used for this activity is a faded photograph of my sister years back (because I could not find any old photo of mine).

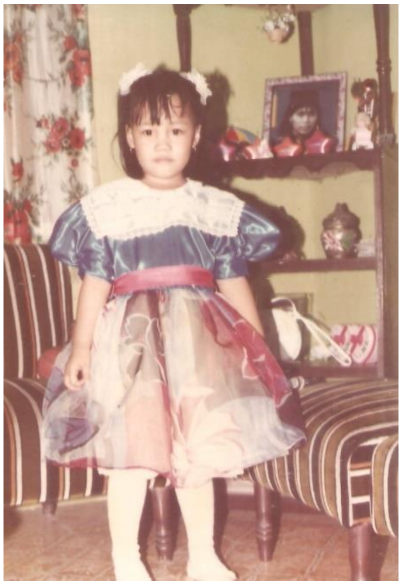

Figure 1. Faded photograph of my sister.

1 Contrast Stretching

Instead of performing a contrast stretch on the histogram of a grayscale image, we do it on the RGB channels of our color image. In the same sense, we "stretch" the histogram such that the minimum grayscale value would be 0 and the maximum would be 1. The enhanced image is shown in Fig. 2.

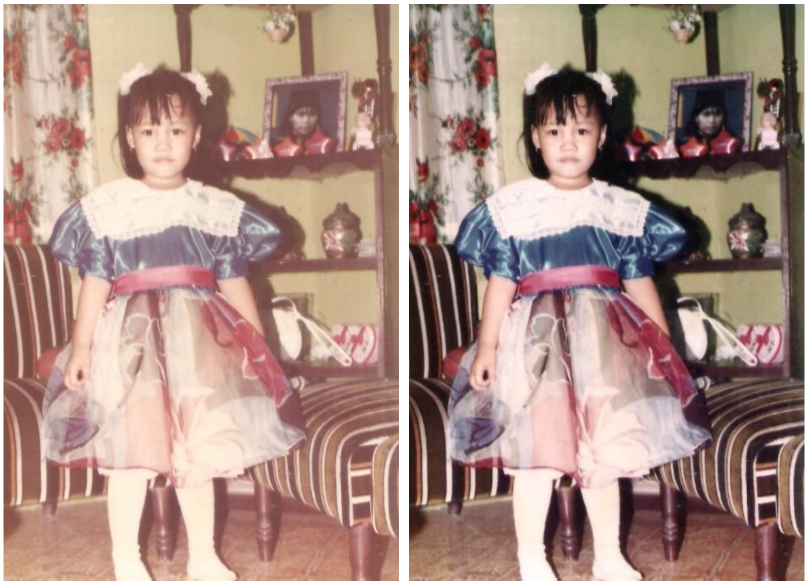

Figure 2. L: Old Photo. R: Enhanced image via contrast stretching.

2 White Patch Algorithm

For this technique, we divide the raw camera outputs by the camera output for a white object[1]. In this case, we take a region from the image we know to be white, and then we get the averages of the RGB channels separately. Then, we divide each channel by their respective averages. From my image, I take my white patch from the white socks.

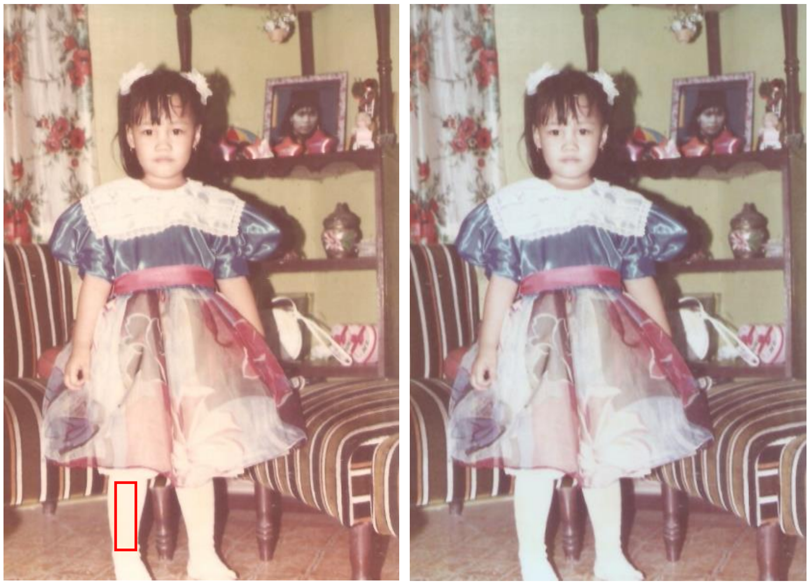

Figure 3. L: White patch (enclosed by red box) used for the white patch algorithm. R: Enhanced image via white patch algorithm.

From the enhanced image above, we can see that not much has changed as compared to the original image. I shall try to discuss later as to why this is so.

3 Gray World Algorithm

For this technique, we assume that the average color of the world (that is, the image) is gray[1]. In essence, if we take the average of an image enhanced via gray world algorithm, the value must be 127.5 (or 128, rounded to the nearest integer). To do this, we take the average of the RGB channels of our image, and then divide the channels by their respective averages. We then multiply each with a constant factor such that the average of the image would amount to 127.5.

[1] Soriano, M. (2019). Activity 6 – Enhancement of Color Images [Lecture Notes].

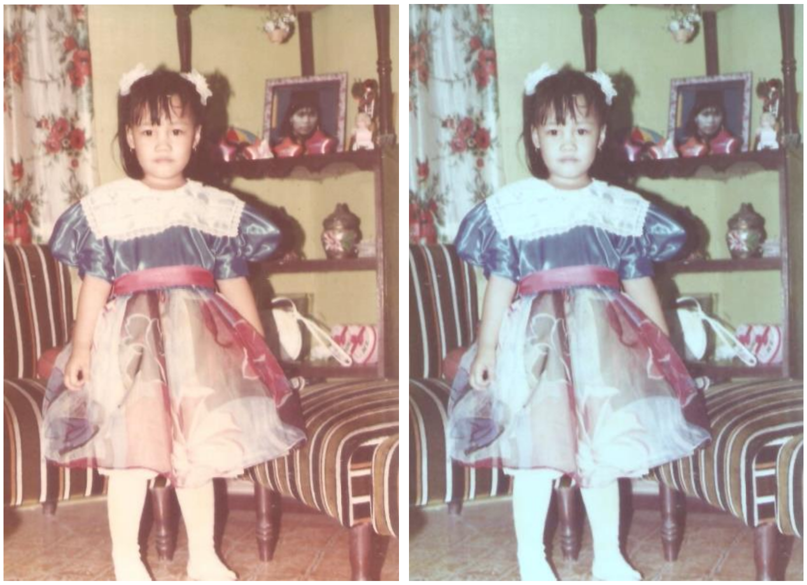

Figure 4. L: Old photo. R: Enhanced image via gray world algorithm.

4 Limitations

I think that each of these automatic white balancing techniques have their own limitations, and I shall try to discuss each one of them.

For contrast stretching, little to no change will be visible if we start with an original image whose histogram of RGB channels already have minimum and maximum grayscale values that are near 0 and 255, respectively.

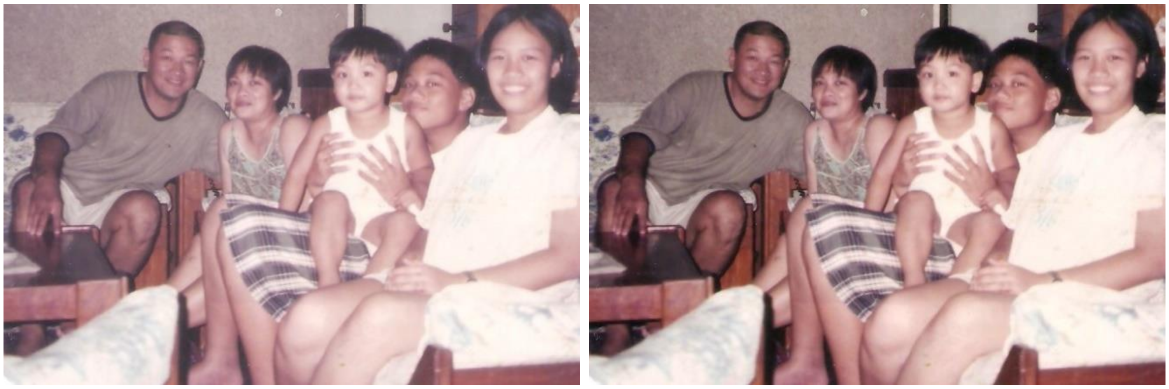

Figure 5. L: Old family photo. R: Enhanced image via contrast stretching.

While we see a little change from the original photo in Fig. 5, the improvement isn't as drastic as compared to the faded photograph of my sister. Looking at the histogram of the faded photo of my sister and that of my family, we see why.

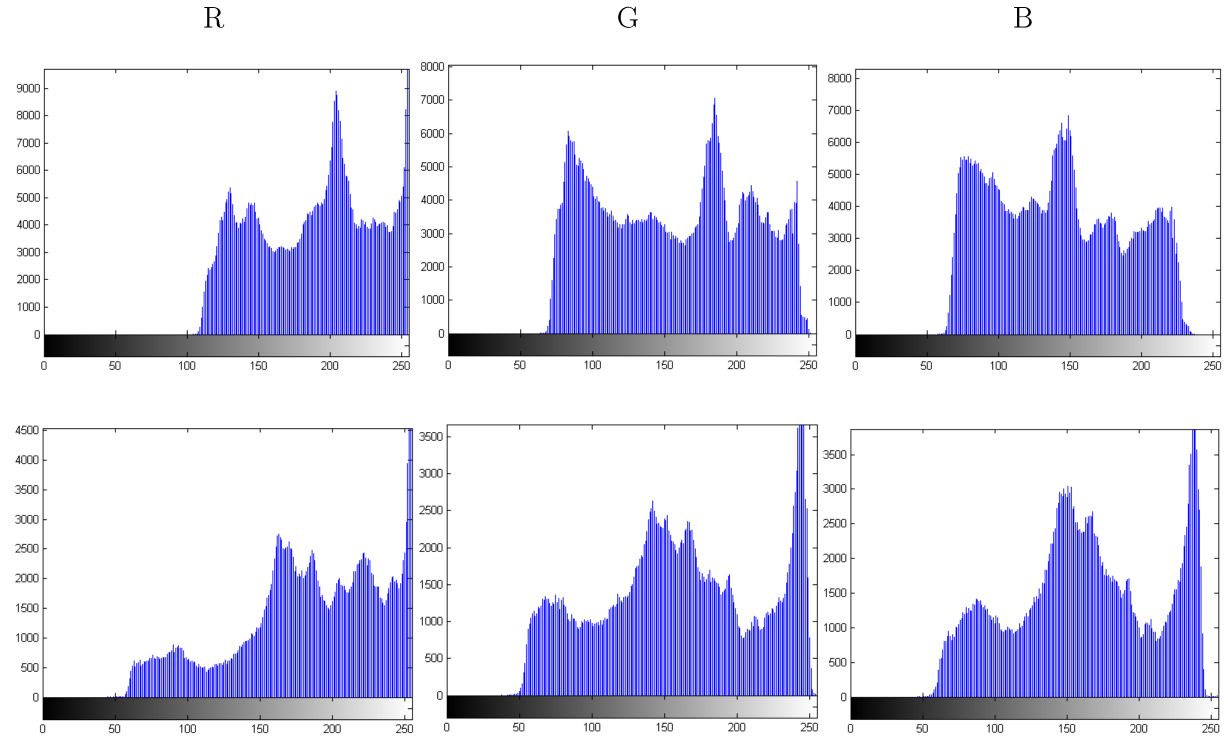

Figure 6. Top: Histogram of RGB channels of faded photo of sister. Bottom: Histogram of RGB channels of old family photo.

Looking at the top and bottom rows in Fig. 6, we see that the histogram of grayscale values (of the RGB channels) for the old family photo are relatively nearer to 0 and 255, as compared to that of the faded photo of my sister. And with this, we'd expect to see better improvement for the latter, and we did.

For white patch algorithm, if the white patch taken from the original image is near the actual color of white, the enhanced image will present little to no improvement. To test this, I tried enhancing our unbalanced image in Applied Physics 187 (whose known white patch is greenish in color), and I got satisfying results. Therefore, maybe, white patch algorithm would be suitable to use on unbalanced images (where the color known objects to be white isn't white at all), rather than on faded photographs whose supposedly white parts are still near white.

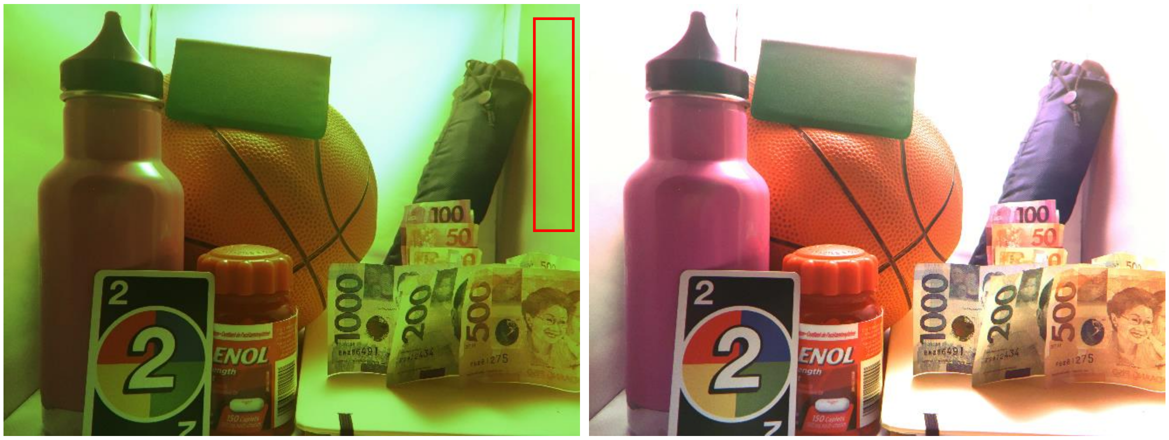

Figure 7. L: Unbalanced image (white patch enclosed by red box). R: Enhanced image via white patch algorithm.

For gray world algorithm, this technique works under the assumption that an image has a sufficient amount of color variations[2], i.e. objects in the image have wide range of colors. With this assumption, the RGB components of the image should average to a common gray value[2]. Otherwise, if there is only presence of similar colors, the algorithm might only illuminate this dominant color.

For this activity, I grade myself 11/10 because I was able to produce all required output, and I also tried to explain the limitations of each of the automatic white balancing techniques.

[2] Liu, D. (2013). Comparison Analysis of Color Constancy Algorithms. Retrieved from: http://www.diva-portal.se/smash/get/diva2:631476/FULLTEXT01.pdf

APPENDIX: Matlab Code[3]

```
%% Load Image
old_im = imread('old_im1.jpg');
old_imR = old_im(:,:,1); old_imR =
double(old_imR);
old_imG = old_im(:,:,2); old_imG =
double(old_imG);
old_imB = old_im(:,:,3); old_imB =
double(old_imB);

%% Contrast Stretching
min_R = min(old_imR,[],'all');
min_G = min(old_imG,[],'all');
min_B = min(old_imB,[],'all');

max_R = max(old_imR,[],'all');
max_G = max(old_imG,[],'all');
max_B = max(old_imB,[],'all');

st_R = (old_imR - min_R)/(max_R -
min_R);
st_G = (old_imG - min_G)/(max_G -
min_G);
st_B = (old_imB - min_B)/(max_B -
min_B);

st_im = cat(3,st_R,st_G,st_B);
imshow(st_im);

%% Gray World Algorithm
gwR_ave = mean(mean(old_imR));
gwG_ave = mean(mean(old_imG));
gwB_ave = mean(mean(old_imB));
gw_ave = mean([gwR_ave gwG_ave
gwB_ave]);

gwR = old_imR/gwR_ave*gw_ave/255;
%change to 127.5 if necessary
gwG = old_imG/gwG_ave*gw_ave/255;
%change to 127.5 if necessary
gwB = old_imB/gwB_ave*gw_ave/255;
%change to 127.5 if necessary

gw_im = cat(3,gwR,gwG,gwB);
imshow(gw_im);

%% White Patch Algorithm
wp = imread('white_patch1.png'); wp =
double(wp);

wpR = wp(:,:,1);
wpG = wp(:,:,2);
wpB = wp(:,:,3);

wpR_ave = mean(mean(wpR));
wpG_ave = mean(mean(wpG));
wpB_ave = mean(mean(wpB));

wp_R = old_imR/wpR_ave;
wp_G = old_imG/wpG_ave;
wp_B = old_imB/wpB_ave;

wp_im = cat(3,wp_R,wp_G,wp_B);
wp_im_adjusted = wp_im*0.8; %multiplied
constant may vary, depends upon choice
imshow(wp_im_adjusted);
```

[3] Code may be found at: https://github.com/loujoseftan/AP186/tree/master/Act6