Homework Assignment 1
15-463 Computational Photography F22
Kavya Tummalapalli
Friday, September 16th, 2022

1.1 Implement a basic image processing pipeline

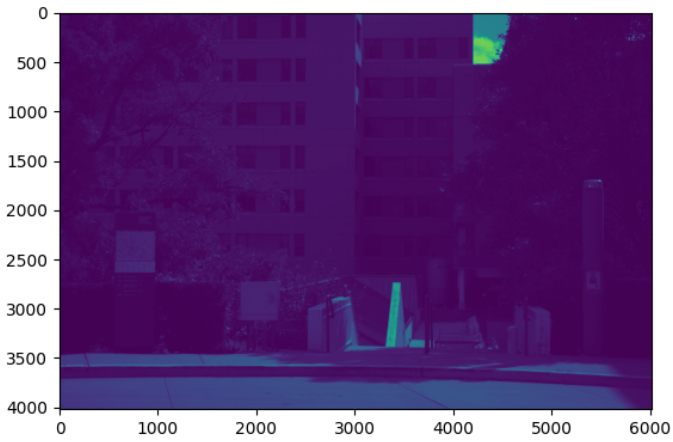
RAW image conversion

The darkness (<black>) was 150 and the saturation (<white>) was 4095. The three multipliers <r_scale>, <g_scale>, and <b_scale> were 2.394531, 1.000000, and 1.597656 respectively.

Python initials

The image has 16 bits per pixel and width and height dimensions of 6016 by 4016 pixels.

Linearization



Linearized image from raw

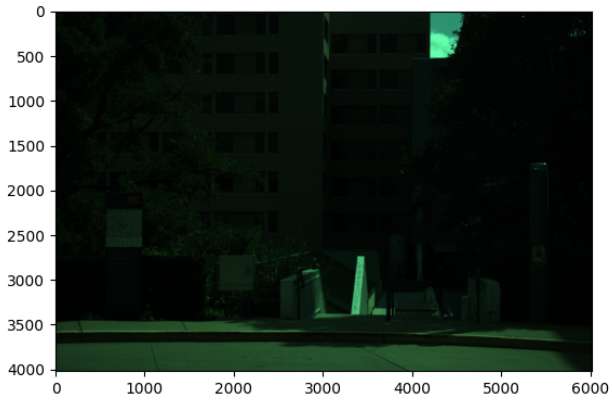
Identifying the correct Bayer pattern

A way for identifying which Bayer pattern applies to my image file is that I first implemented white balancing. Then, to actually figure out the Bayer pattern now that the image would have the expected coloring to the human eye, I implemented demosaicing modularly so that I could easily switch between the four patterns. Next, knowing what colors certain parts of the image should be (the 'Carnegie Mellon University' sign should be red, the sky should be blue, etc.) I went through the four options for the Bayer patterns to compare the coloring. The one that was correct stood out immediately, as there was no tint on the overall image and the sign was bright red and the sky was visibly blue. The other three patterns caused the image to be discolored or not look realistic.

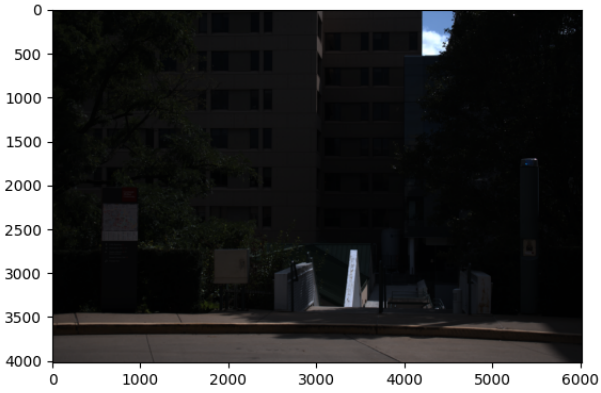
Through this method, the Bayer pattern that I identified for the image was ‘rggb’.

White balancing

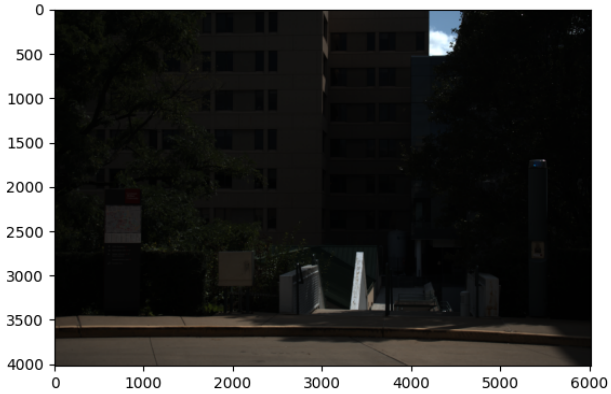
Out of the three white balancing algorithms, the white world one looked very green tinted and did not look great with the given image, while the grey world and the multiply scales white balancing algorithms both looked the best and like a real life tint of the image. Though there was not much discrepancy between the grey world and multiply scales resulting images, I think the multiply scales method would work best overall given a specific image because these scalar values were determined by the specific camera and image settings while the grey world was a general formula that would work generally well on any image.



White world white balancing

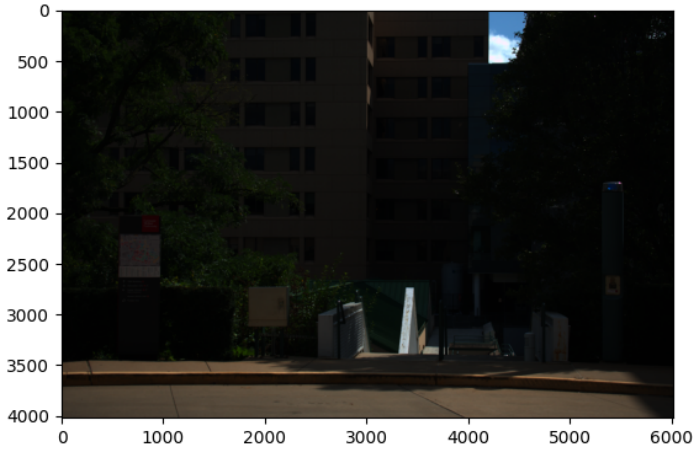


Grey world white balancing



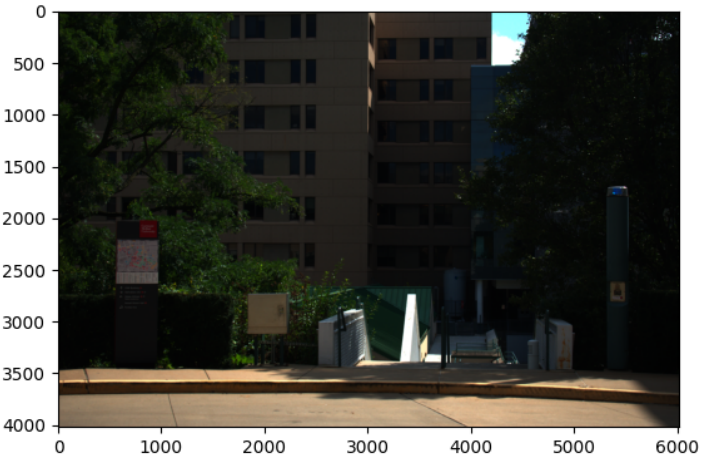
Multiply white balancing

Color space correction

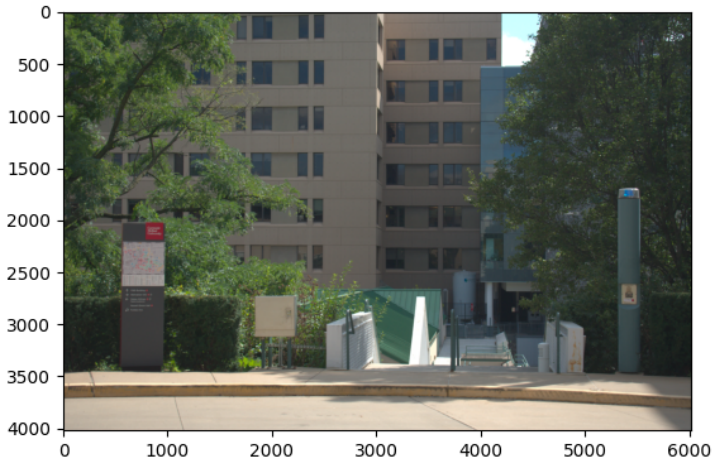


Color space corrected image

Brightness adjustment and gamma encoding



I found the suggested value of 0.25 for the post-brightening mean grayscale intensity to look too bright and you could not see the blue in the sky or and shadows on white areas. On the other end, lower values under 0.10 continued to look too dark and so the intensity value that looked best to me was 0.15.



Gamma encoded image

Compression

With a quality parameter set to 95, I could not tell the difference between the two files. The compression ratio is 34,405,798 bytes (png) / 6,385,437 bytes (jpeg) = 5.388. The lowest setting for which the compressed image is indistinguishable from the original is 75, which resulted in a compression ratio of 7.98.

1.2 Perform manual white balancing

1.3 Learn to use dcraw

The correct flags in order for dcraw to perform all the image processing pipeline steps I implemented in Python are dcraw -4 -D -T -k <black> -S <white> -f -r <r_scale> <g_scale> <b_scale> <g_scale> -o 5 -b <1.0> -g <2.4 4.5> campus.nef.

The three developed images I have are different because the given processed image is based on the camera's settings while the other two images we had control over the exact image processing steps and values. The image I like the best is the one I created through my image processing pipeline as at each step, I could adjust values to create an image to my liking.

2.1 Build the pinhole camera

My design decisions were very limited by my two box options, one being around the size of a shoebox, and the other about 2 times the width and length but with the same height. I chose against the second option because I believed due to the minimal height in the screen size for a larger field of view, I would be capturing a lot of the box itself rather than just the image appearing on the screen. For the box I did pick, the screen size was

12 inches by 6 inches with a focal length of about 8 inches and the field of view varied with each pinhole size.

2.2 Use your pinhole camera

The three pinhole diameters I used were 0.5mm, 1mm, and 5mm. I originally started out with the 0.1mm pinprick for the smallest pinhole, but I believe in the lighting I took the pictures, this size was not enough light to display a visible image and so I made it

slightly larger. The differences I observed for the different pinholes were that for my smaller pinholes, the images were much darker. This may have been due to the overcast weather conditions, so the largest pinhole had a brighter view than the others. However, I found that the smallest pinhole had the most detailed images as expected, especially for the first scene, compared to the larger pinholes.