17th of April

COMP 590

Elias Capriles

Camera ISP Pipeline

1.1

RAW Image Conversion:

I started the processing process by converting the baby.nef file to a tiff using the dcarw command line function. I started by doing a *reconnaissance run* of dcraw to obtain the following values:

- <black> = 0
- <white> = 16383
- <r_scale> = 1.628906
- <g_scale> = 1.00000
- <b_scale> = 1.386719

After writing down these values for later processing I then reran dcraw to obtain a the baby.tiff file to fully process the image.

Python Initials:

I first processed the image by writing down the heigh and width of the array obtained by using skimage.io.read. The image has a Height 4282 and a width 2844. Moreover, each pixel holds 16 bits.

Linearization:

To set the image into a scale from [0,1] I linearized it in the following way. I divided every value in the image by the <white> value to scale the image down. I would have subtracted the <black> value from the image as well, but in this case, it turned out to be 0, therefore it did not change anything.

In order to finish off the transformation I used the np.clip function to “clip” all of the values that were above 1, or below 0. This resulted in the following image:

Bayer Pattern

The image uses a Bayer pattern that is RGGB. I first found this by using the following strategy. When looking at the first 2x2 square of the image I realized that the top right, and bottom left pixel were almost identical letting me know that these were the green values. The way I realize where the red and blue pixel went happened during the white balancing. When doing the white balancing with the camera values I blew up one of the values and then blew up the other one. When I blew up the top right corner the picture came out a lot more red. Meanwhile blowing up the latter caused the picture to be bluer.

White Balance

The white world algorithm was implemented by dividing each of the channels by the result of the np.percentile function, with 94 as the percentile parameter. This chooses the 94^{th} percentile of the brightness on each channel and divides that channel by the value. This give the following image.

The gray world algorithm is similar to the white world algorithm, but instead of using the np.percentile function, I used the np.mean function. That way I divide all the values of the channel by their mean values. Resulting in the following image.

The final white balancing algorithm was using the preset values given by the camera which were found in the “reconnaissance run” of dcraw. In this case what we do is multiply all values in red channel by <r_scale>, we do the same for the blue and green channels but with their preset values. This results in the following picture which I used for my the next steps picture.

Demosaicing and Color Space Correction

In this step I used the interp2d function from the scypi function on each channel in order to interpolate the values of each channel. To use the interp2d function I first created an x and y linespace where each one is the width and height respectively. Since I got x and y from the channel arrays the values were halved. This is later fixed by creating a second line space that is used in the function created by interp2d that has the correct dimensions for the image. The green values are averaged, and at the end all three values are stacked on top of each other in order to create the RGB channels.

The Color space correction was done by taking each pixel and multiplying the transpose of the rgb channels by the inverse of $M_{sRGB \rightarrow cam}$. This resulted in the following image.

Brightness and Gamma Correction

I ended up choosing to brighten the image by 15% as a way to get rid of the warmness of the image. In my opinion this looked the best when compared to all the other presets. I then gamma corrected the image according to the sRGB color space guidelines. This eventually resulted in the final image.

Compression

The final step was to finally compress the image. I compressed the image as a png and as a jpeg with the quality parameter set to 95. I could see that the PNG was a little bit blurrier on the ends. However, I had to really look for differences in order to notice anything. The compression ratio between the .nef file and the jpeg is 9.59, an extremely large difference.

When decompressing the jpeg I started to see differences at a quality parameter of 35. At this point I started noticing waves around the baby. The compression ratio between the nef file and this distorted image was 65.4.

The distortions can be seen in the Image below around the baby.

1.1 Final Images

Using the white world approach I got the following final image:

Using the camera presets I got the following pictures:

1.2 Manual White Balance

I attempted to implement this algorithm, but every normalization effort made the picture into a green Hue. The patches that I used where from the top right corner as it is a clear light source. I tried to use the dog picture on the baby's shirt but it just made the image imperceptibly dark. One can see my two best attempts below:

M1:

M2:

1.3 Using dcraw

I created the following image using dcraw. The flags used where: -W, -o 1, -q 0, -g 2.4 12.92. These flags resulted in the image:

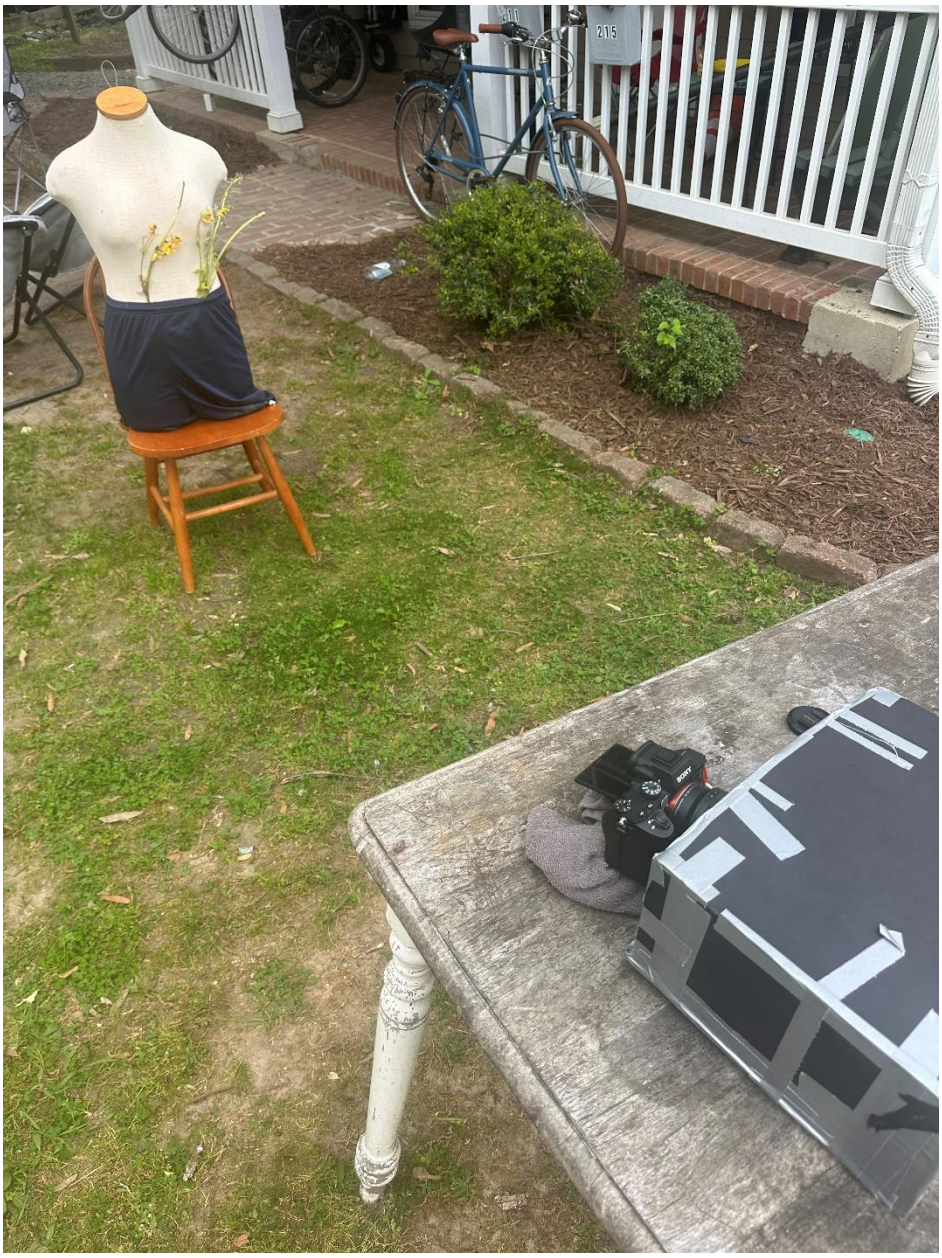
In conclusion, we can see three distinct images created through three different methods. I believe that the lack of green hue in the one made by my code and the one made by dcraw are the best ones shown. The main difference between these two pictures is that the one that was outputted through my code has brighter reds in general. This is evident when looking at the shirt of the baby.

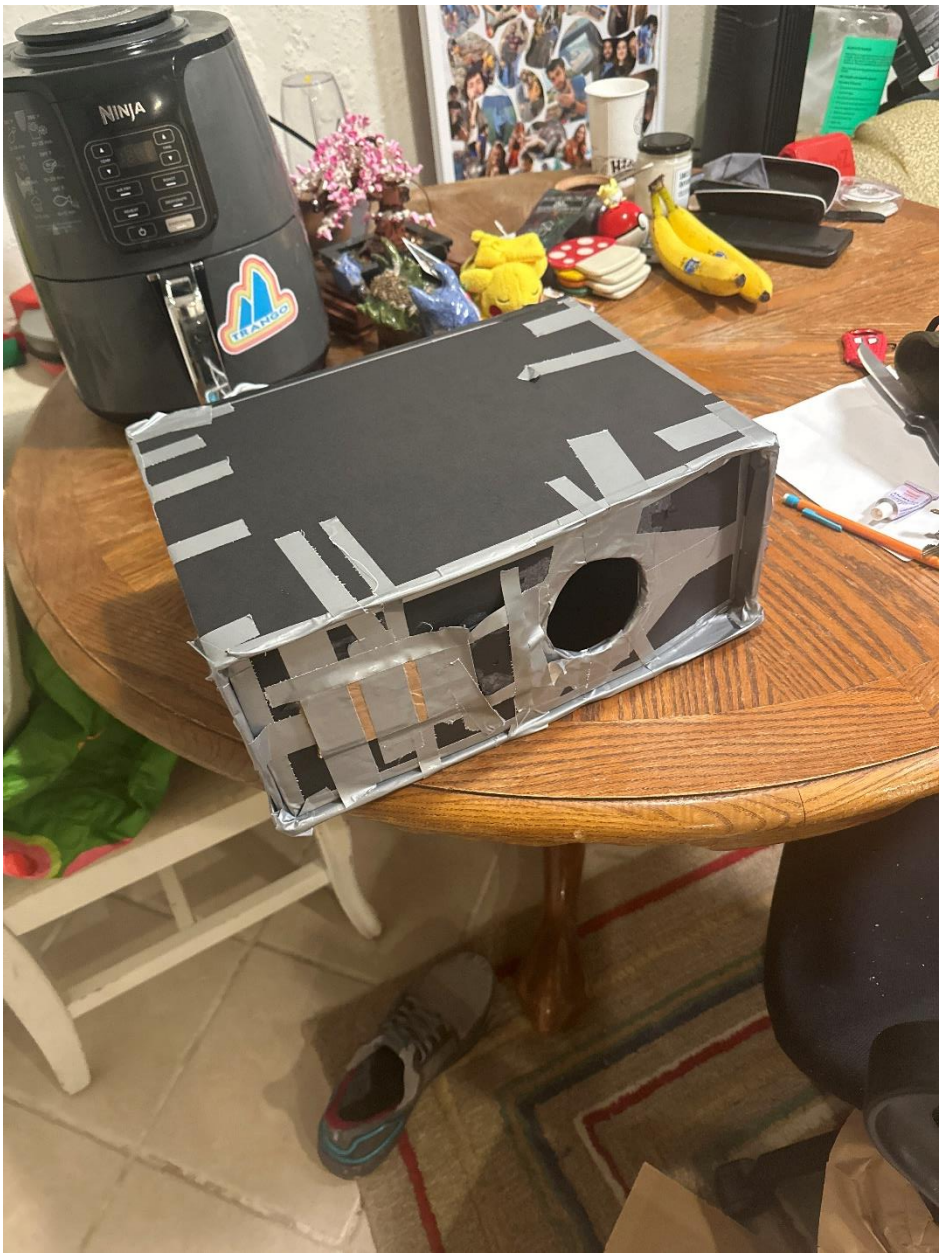
Overall, I like the picture that my code produced better. The one made by dcraw is closer to the baby.jpeg provided in the assignment, but I like that the red pops more in mine. I am extremely proud of the image that I have produced.

2.1 Pinhole Camera

I created a pinhole camera out of black cardboard. I was able to acquire this black cardboard from a supply store. This allowed me to create a box that knew would not be needed to be duct taped all over. Instead, the only part that would not to be duct taped would be the edges. There was a lot of tape used to ensure that the box was fully light proof. I attached printer paper to the box as a screen. Moreover, I created a the whole for the camera a little bit bigger than originally needed, and then duct taped around the whole to allow for a snug fit.

I chose a focal length of around 27cm depending on how far the camera went in. The field of view was wide in the camera in order to allow capturing my whole scene. This is reflected by the white scene because I was also curious how much light would be captured by the camera. I chose three different pin hole sizes. I chose 0.1mm, 0.7mm, and 3mm. The first and last were chosen as recommended in the assignment instructions. The reason I chose 0.7mm is because this was the value I obtained following the formula given in the assignment instructions. The final camera can be seen below.

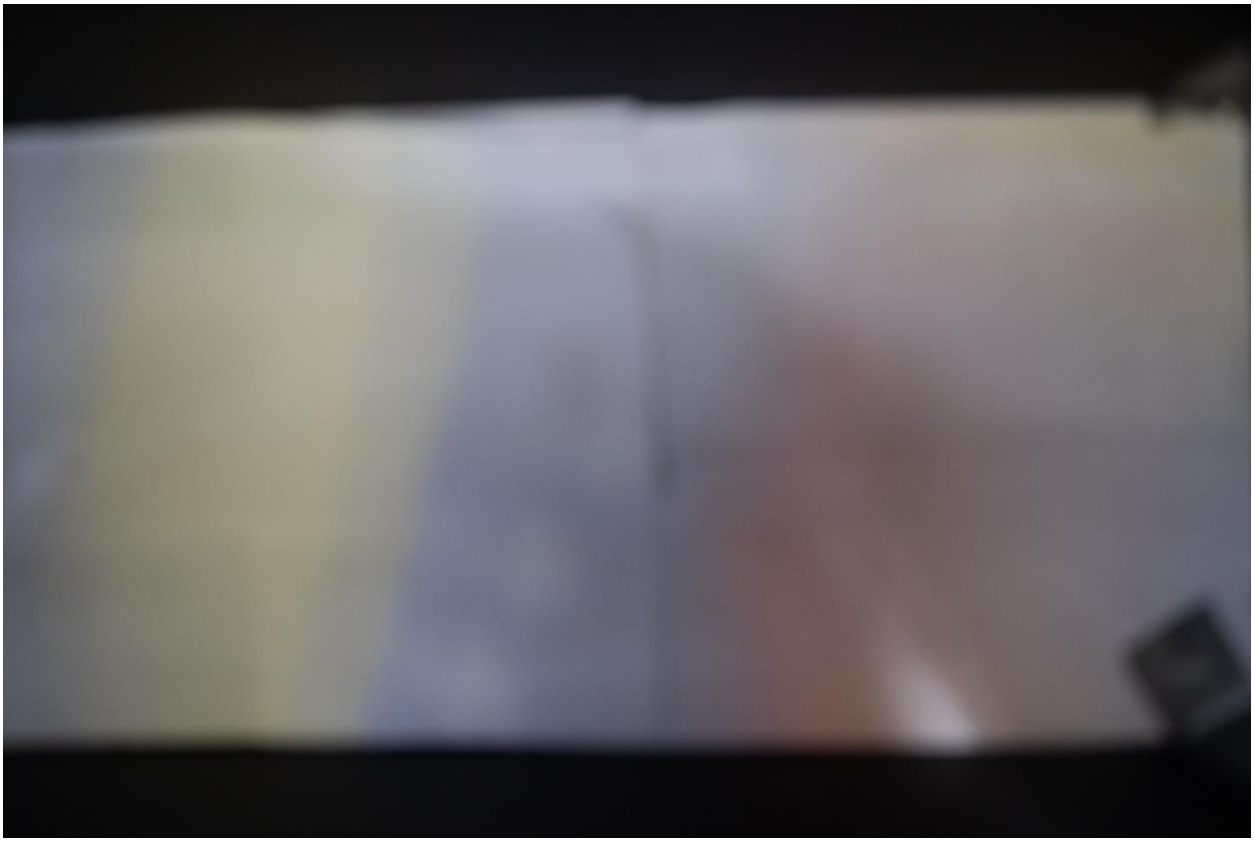

215
SONY



2.2 Using the pinhole camera.

The first picture I took was of the sun with the clouds. I finished the pinhole camera later on the day and wanted to capture that last little bit of sunlight. The following pictures are in order of increasing pinhole size. Since, I was directly looking at the sun one can see that the camera gets overexposed during the larger pinhole sizes.

The second picture that I chose to take was of my Venezuelan flag. I have it nailed outside of my house and thought that the light going through the image pinhole would allow for some cool effect. The images came out really well. The effects on the images were opposite to the ones seen in the first iteration. In this case the higher pin hole sized allowed for more light to come through and more detail to be recorded. The images below are in order of increasing pin hole size.



The third set of images was of a mannequin one of my roommates was gifted. Her name is Amy and the reason that I chose her as my model was because of the ability to stay completely still while the camera was being exposed to the light. Coincidentally the image that came out the best was the mid-sized pinhole that I used. I think it just allowed the perfect amount of light in, since the other two were either too exposed or not exposed enough. The images below are in order of increasing pin hole size.