Leo Chen and Jared Waters
CE264
Project 1 Lab report

Introduction:

HDR is a technology which creates a composite image out of several other images. By altering the exposure from exposed to the right to varying amounts of saturation you can acquire information about various parts of the image. Synthesizing information from all these parts into the composite creates an image with high contrast and higher SNR in both bright and dark regions than would be possible with a single image.

Part 1: Radiometric Calibration

Intro: We need to understand how the camera we used, a Canon SL1, alters the raw brightness values in its processing. To do this we took a piece of printer paper and taped it to a wall outside. I placed a single dot in the center so the camera would autofocus faster. We then took a series of pictures at varying exposure. The camera was on a mini tripod a few feet from the paper.

Methods and Results: The first picture was the longest exposure that didn't produce any saturated pixels. I then took a picture at every exposure setting between the first, 1/60, and when the image became too dark to generate any more useful data, 1/2500. We acquired 17 images for radiometric calibration.

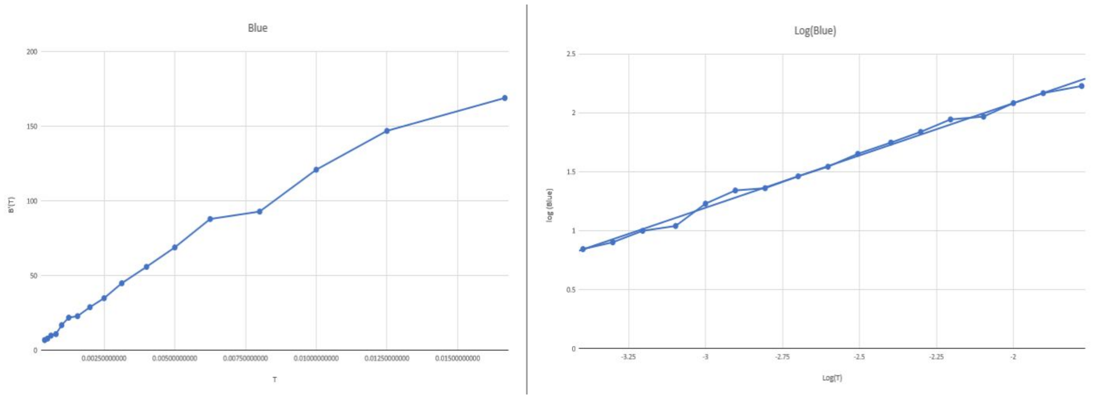
Next we developed code to get the brightnesses in each of these images. We first developed a method, averageVal, which takes a file name string and a pointer to an array of int. This method opens the file and scans a 10x10 pixel patch. It adds up the values for each pixel/channel in a loop. Once all 100 pixel values are added up for the three color channels, it divides each of them by 100, thus calculating an average for each channel in that region. This reduces the noise significantly. We could have performed averaging on a larger patch and perhaps gotten slightly better data, but 100 pixels seemed sufficient. Care was taken to only sample from the region which corresponds to white paper, not the focusing dot in the center or the borders which showed the wall behind the paper.

The function getBrightness calls averageVal on each of the 17 images and prints the values to the console. Once the average brightness values for each picture/channel were obtained, we plotted them using Microsoft Excel since since plotting in opencv requires additional libraries. This resulted in curves which looked as expected.

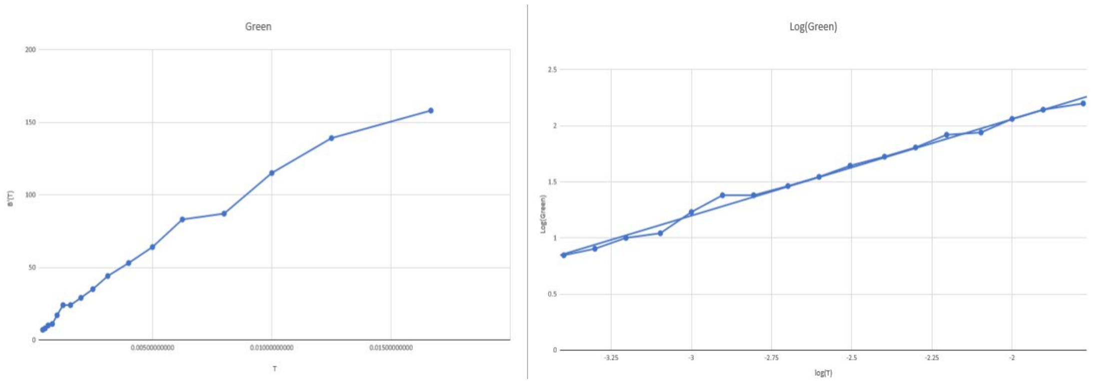
Images used for radiometric calibration. The gain of all images was ISO400.

Next we plotted the log(brightness) vs log(Time) and fitted the 17 data points to a linear regression. This resulted in three G values we acquired from the inverse of the slope of the graph. G seemed lower than expected, but was consistent.

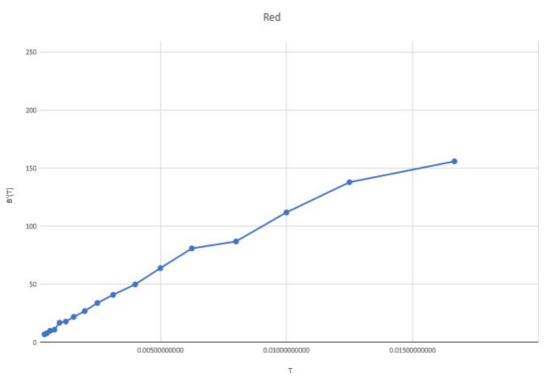
The three G values are 1.12, 1.15, 1.16. We were expecting values closer to 1 as these values will alter the brightness quite a lot.

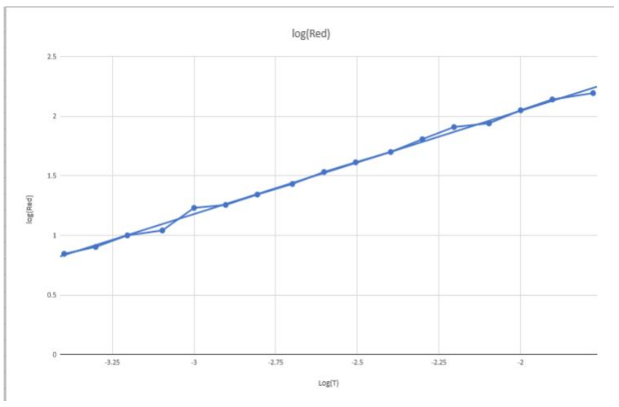


Blue and Log(Blue)

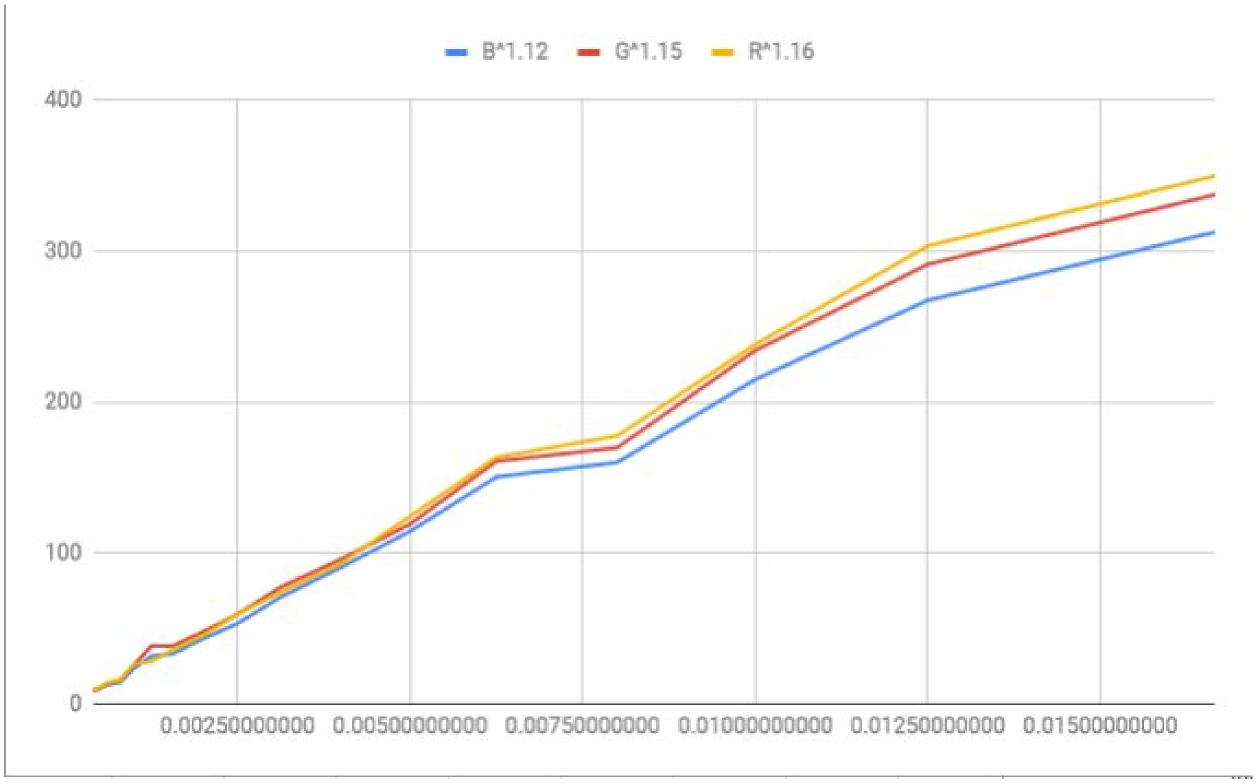


Green and Log(Green)





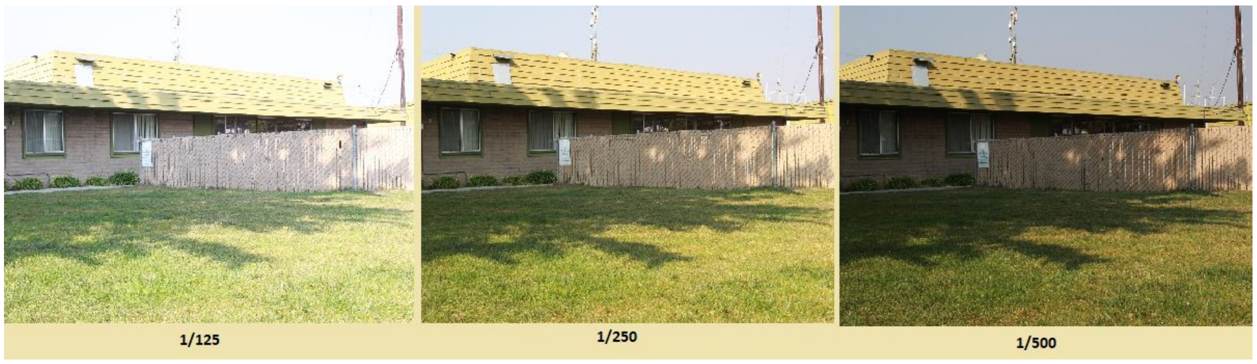
Red and Log(Red)



Linearized brightness values

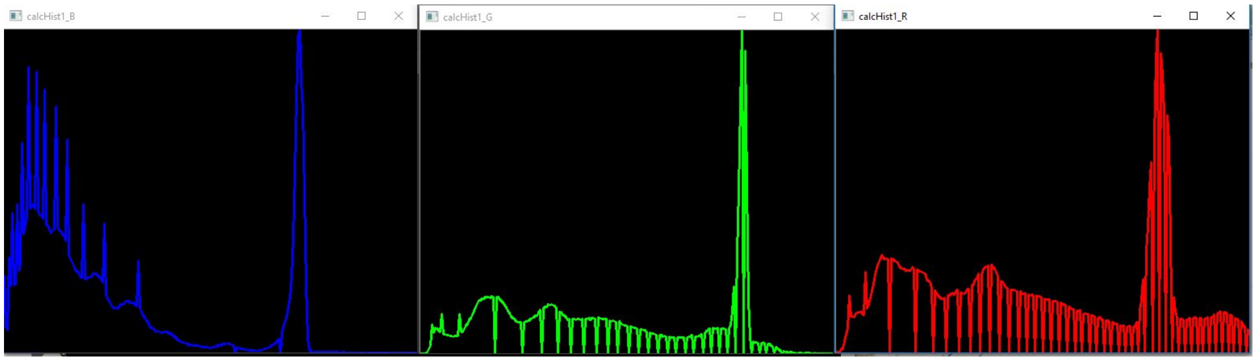
Part 2: Acquiring an image stack

Next we took several images of the Moss Landing Harbor Office. The tripod did have a tendency to move a little, so we had to be sure to get pictures carefully so things were spatially registered. We took images exposed from 1/500, which was properly exposed to 1/30, which was mostly saturated. We decided to use 500, 250, and 125 since the images were spatially registered better than some others and since the exposure times are multiples of 2 of each other. Using these three images produced good results.

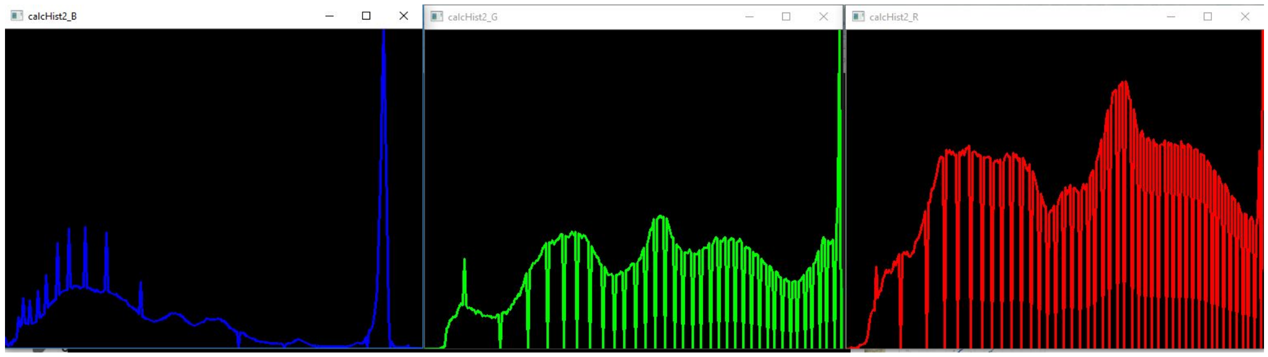


The gain of all images was ISO400. Exposure times 1/125, 1/250, 1/500.

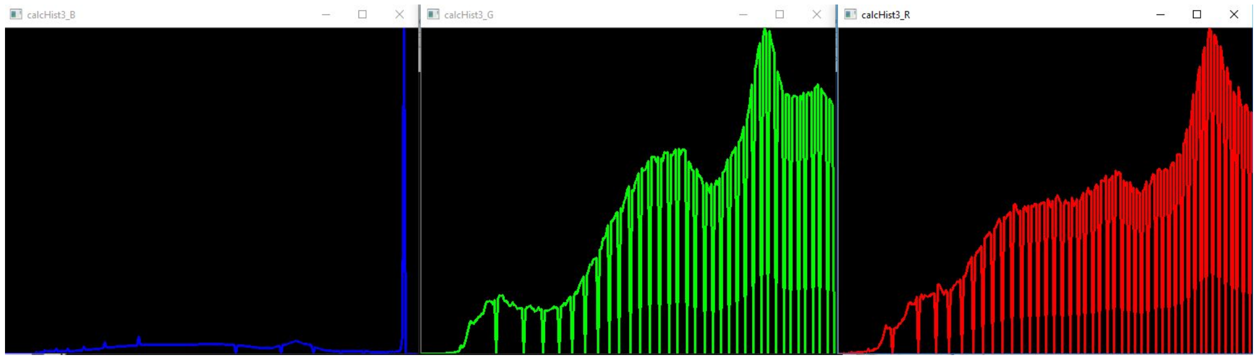
After choosing the images, we needed to apply the inverse of the function which the camera applied in its processing. This results in the pixel/channel values being modified to be closer to what the raw values likely were. This is obtained by raising each value to the G for that channel. This results in linearization of the image.



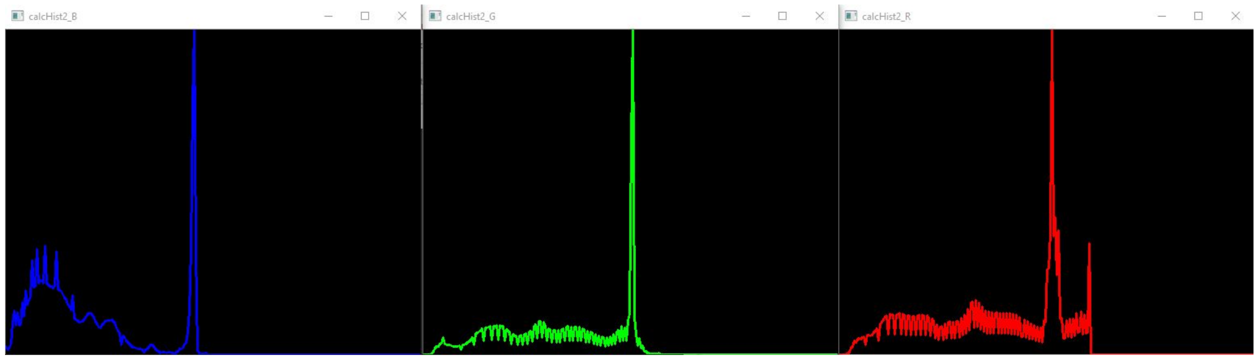
First Image B'g(T)



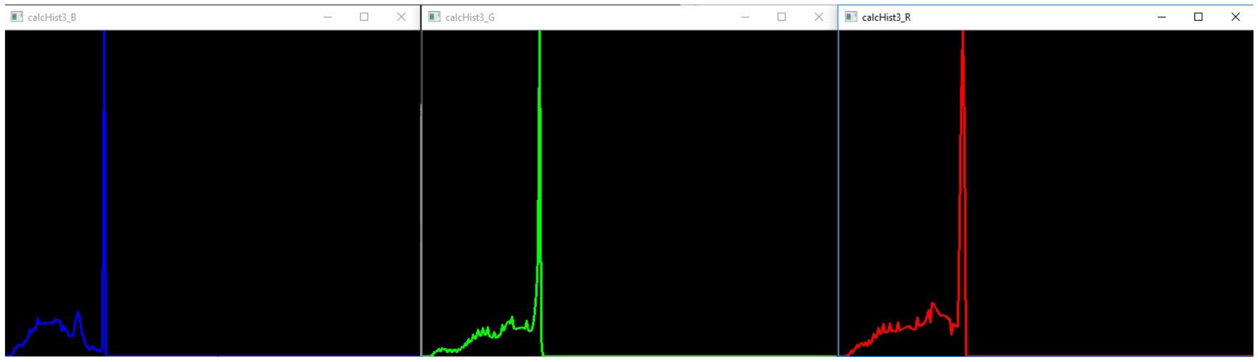
Second Image B’g(2T)



Third Image B’g(4T)



Second Image B’g(2T)/2

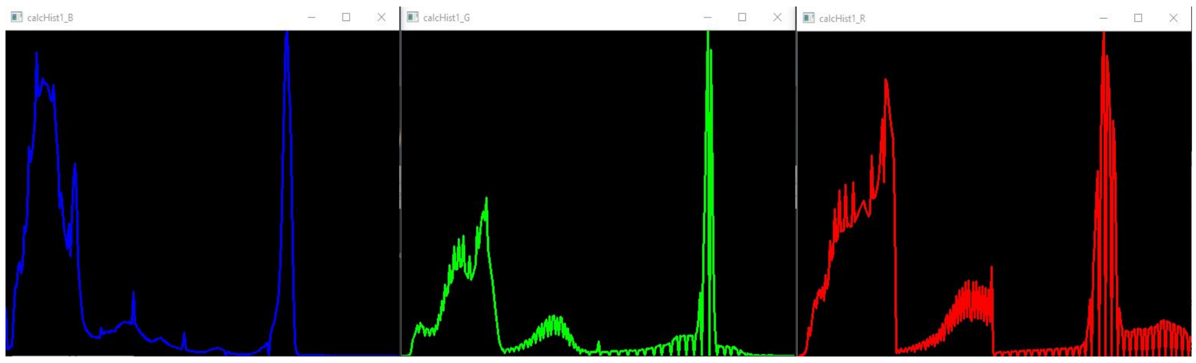


Third Image B’g(4T)/4

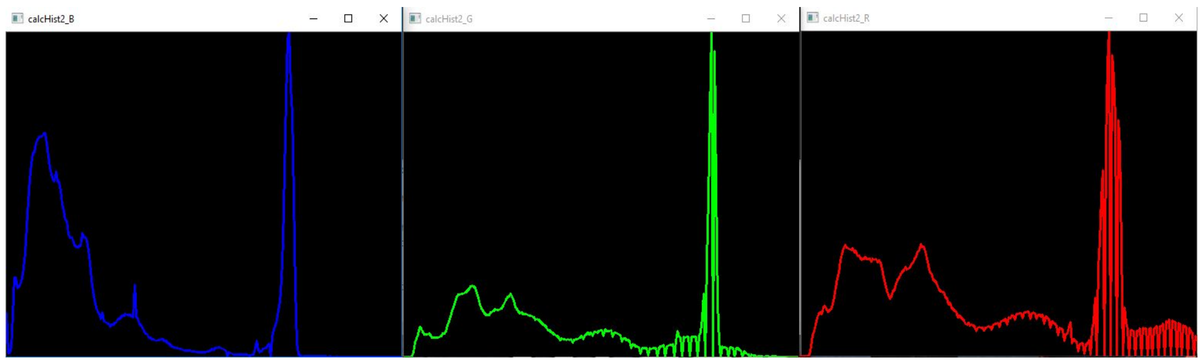
Part 3: Create a composite image

We took the linearized images and applied two different HDR methods. In the first, to determine the value of a pixel/channel in the composite image, we simply used the brightest image in which that same location was not saturated. The value was then scaled by a factor representing how much longer that image was exposed compared to the image which was exposed to the right.

In the second method, to determine the value for a pixel/channel, we look at all images which aren't saturated at that same location. The values are then adjusted for exposure time difference and averaged to produce the composite value.

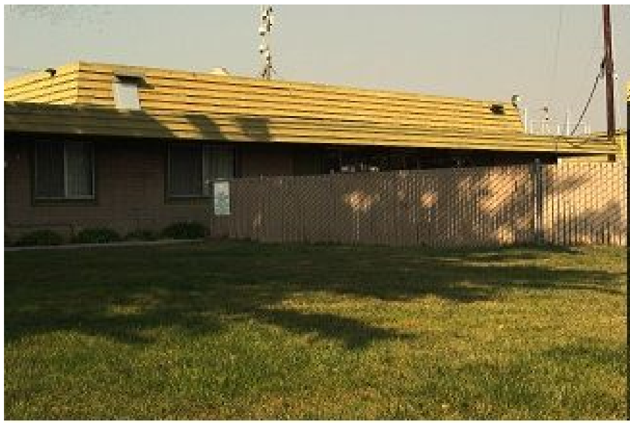


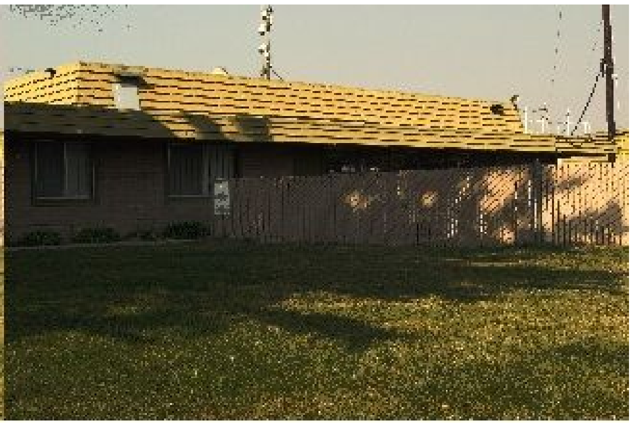
Method 1 Composite Histogram



Method 2 Composite Histogram

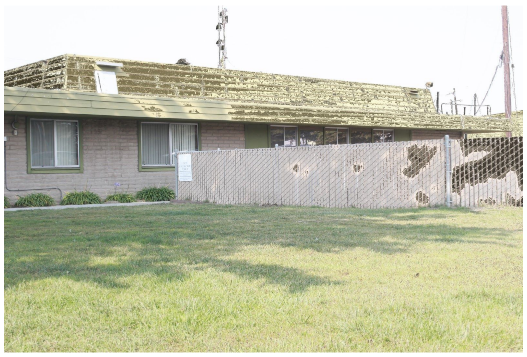
The second, averaging, method produced a much better result than the first. This is also apparent when looking at the histogram. You can see that the histogram for the first has more spikes or peaks of near values, while the second has a much smoother histogram, while the overall shape between the two is very similar. This makes sense as the averaging method would tend to represent the same range of colors but remove some of the noise via averaging. When you look at the two images side by side, the second method looks a lot better.

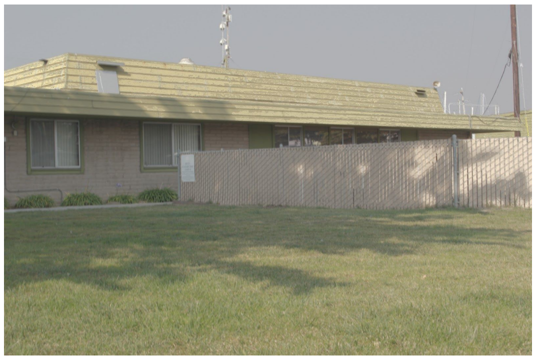

Results from the averaging method.


Results from the pick and place method

Part 4: Reproduce composite image

For this part of the lab, we wanted to get a better composite image quality. This can be done by enhancing the small values and reducing the large values. We simply used the tonemapping function in OpenCV. There was an example on the documentation which we adapted our code implementation from. As you can see the results have an improvement from the previous results of part 3.


Tonemapping of the first Algorithm.


Tonemapping of the second Algorithm

Conclusion

Through this project we had our first High Dynamic Range Implementation. Using OpenCV was sort of challenging in certain tasks but enhanced our skills for future work. It was interesting to see how we had to find the camera radiometric calibration. This type of reverse engineering will definitely be useful for future work.

For the second half of the project we implemented two composition algorithms in which we got different results. This underscored the way in which seemingly small differences in an algorithm can produce superior results. The first image used a pick and place method to write values from one of the source images into the composite image. The second method used averaging to use data from any of the source images with valid data at that pixel location. The data was then adjusted for brightness levels and averaged.

The first image had some artifacts introduced through the pick and place method. The images we used were close, but not perfectly spatially related. This resulted in values being pulled from improper regions for areas which were very bright. We would try to take more care with the image quality if we were to do this again.

The last thing we did was tonemap the image using the opencv library. This produced an image with better visual presentation than the originals.