Computational Photography Assignment 1

Sagnik Ghosh

September 2024

1 Developing RAW Images

1.1 Implement a basic image processing pipeline

RAW image conversion. $< black > = 150$, $< white > = 4095$, $< r\_scale >$ $= 2.394531$, $< g\_scale > = 1.000000$, $< b\_scale > = 1.597656$
Python initials. Number of bits per pixel = 16, Width = 6016, Height = 4016.
Linearization. Code use to implement this is in utils.py in src.
Identifying the correct Bayer pattern. After looking at multiple 2x2 grids, the top-right and bottom-left values seem to be close. This indicates that the pattern is either 'rggb' or 'bggr'. On comparing the images formed after white-balancing and demosaicing, it is clear that the pattern is 'rggb' as it reproduces natural scene colors. Here are the images:

Figure 1: RGGB

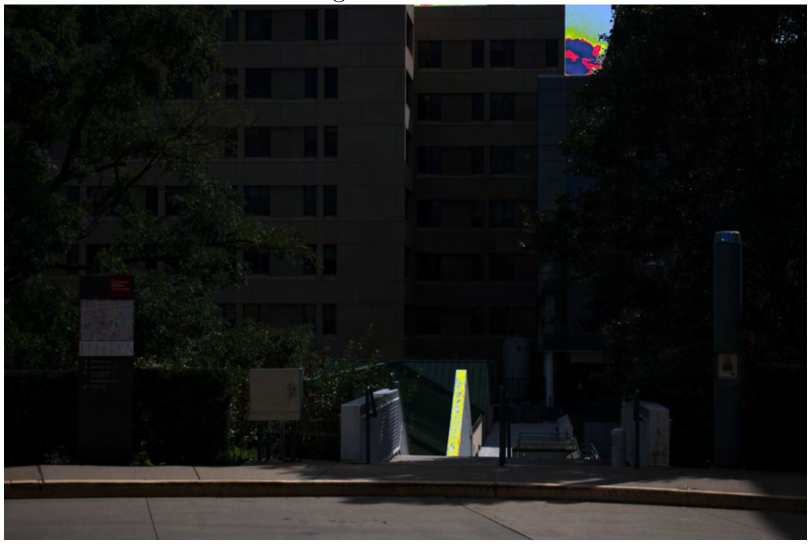
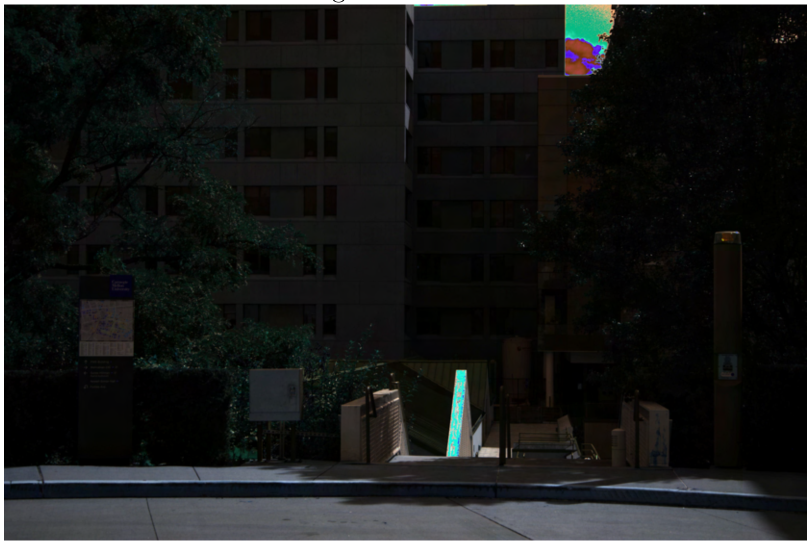


Figure 2: BGGR



White balancing. Here are the fully developed images using the three different white-balancing algorithms:

Figure 3: Gray-world

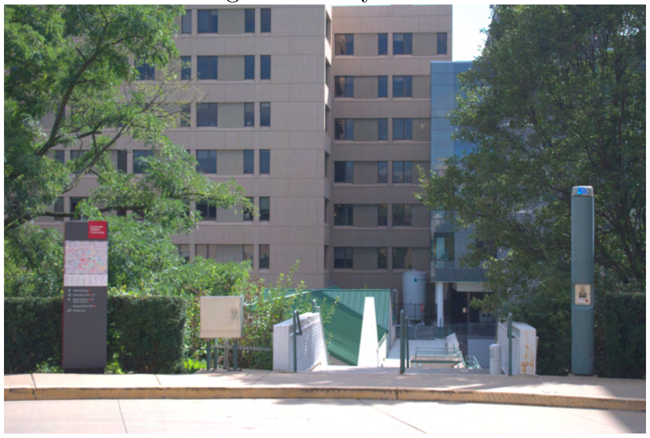


Figure 4: White-world

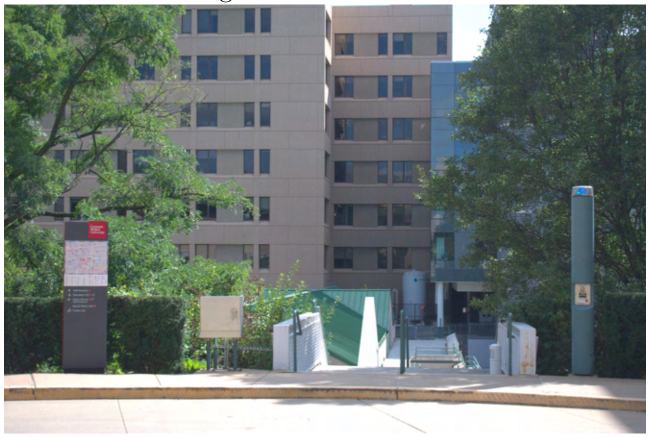
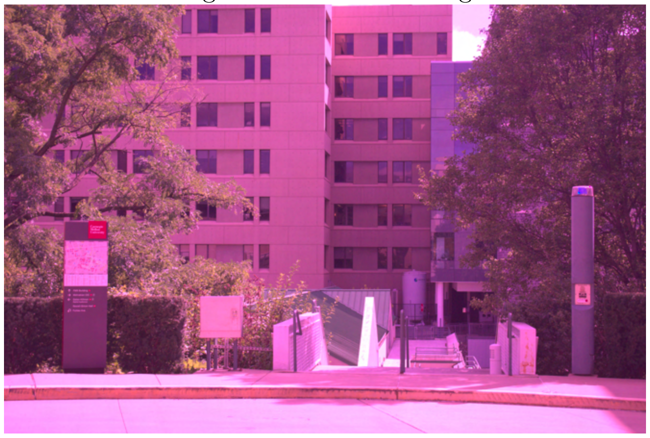


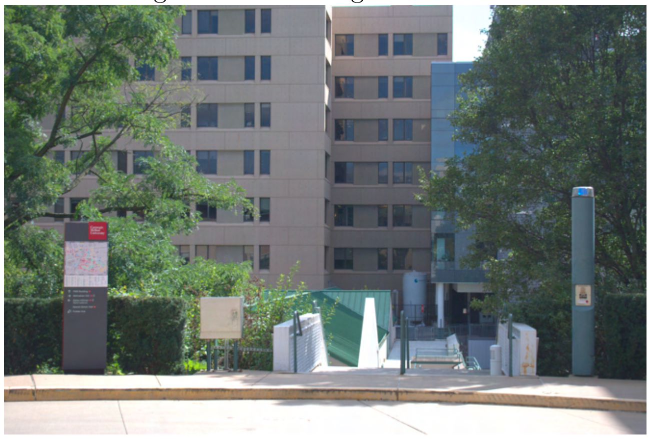
Figure 5: dcraw rescaling



I prefer the image developed using the white-world assumption because it produces natural scene colors and has high contrast. The grey-world image looks very dull and the image developed using the scale factors has a strong red bias.

Demosaicing. The code is in utils.py under a function called demosaic. I used RegularGridInterpolator because the one suggested in the assignment is depricated. The images shown in the previous section were post-demosaicing (and post-brightening and gamma-encoding).
Color space correction. The code is in main.py.
Brightness adjustment and gamma encoding. After trying post-brightening mean values of 0.1, 0.25 and 0.5, 0.25 worked the best. A value of 0.5 oversaturated the scene (the ground was completely saturated with 0.5), whereas with 0.1 the image looked too dark. Here's the image using a value of 0.25:

Figure 6: Mean brightness = 0.25



Compression. The uncompressed and the compressed image with quality 95 are basically indistinguishable to me. The uncompressed file is around 34 MB whereas the compressed file is around 7 MB. Hence, the compression ratio is around 0.2.

The lowest quality setting was 70 for which the image looked indistinguishable from the uncompressed image. On trying quality of 60, I could start to notice some compression artifacts (especially on the CMU logo). The compression factor for a quality of 70 is around 0.06.

1.2 Perform manual white balancing

The first patch I chose is the white region below the map (below the CMU logo). The second patch I chose is the white pillar below the glass building at the back. The third patch is the cloud in the top-right corner. I prefer the first one because the other two are too warm for my taste.

Figure 7: Manual White Balancing - Patch 1

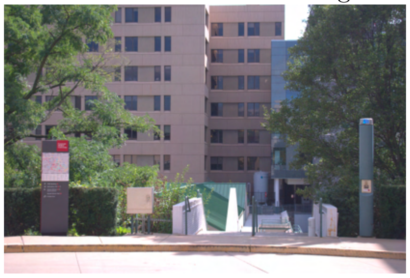


Figure 8: Manual White Balancing - Patch 2

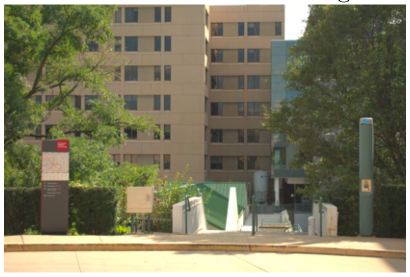
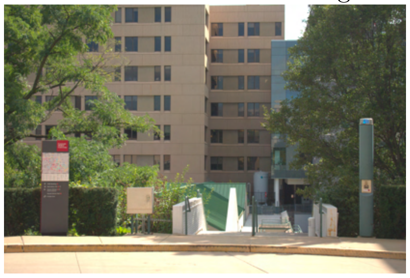


Figure 9: Manual White Balancing - Patch 3



1.3 Learn to use dcraw

The dcraw command I used: dcraw -v -w -o 1 campus.nef Among the 3 images, I prefer the one generated by camera's own image processing pipeline. The colors really pop in this image compared to the one generated by dcraw although they're pretty similar. The one generated using my own pipeline in Python looks washed out (it looks okay standalone but the comparison makes it look washed out). This is probably because of the white-world assumption (it biases the image towards the color white). Using more sophisticated histogram-based whitebalancing would probably create images with better contrast.

Figure 10: Camera's own pipeline

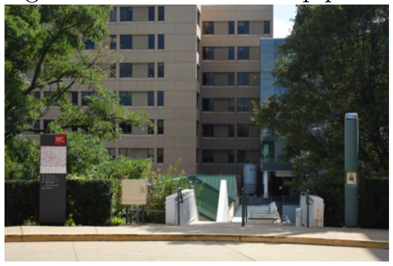


Figure 11: dcraw

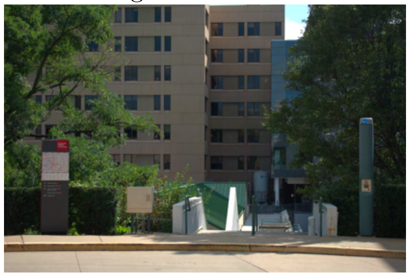
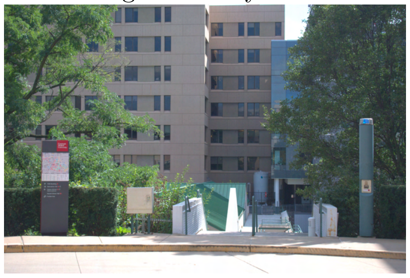


Figure 12: Python

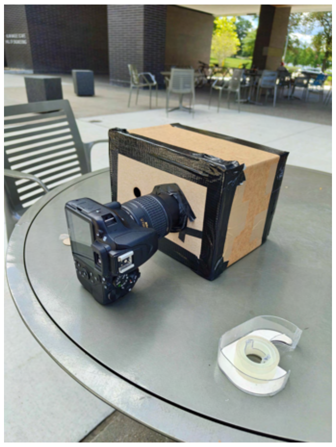


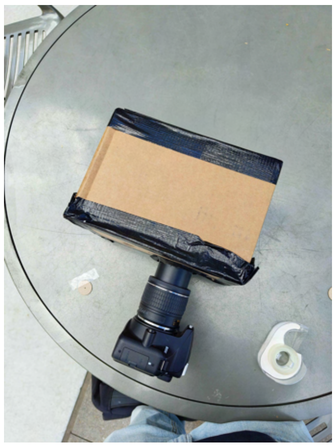
2 Camera Obscura

2.1 Build the pinhole camera

- I used a regular cardboard box to build it. I tore off the top and used another piece of cardboard instead.
- I used the laser cutting machines at TechSpark to cut a hole of diameter 6mm for the camera lens and another 0.7 inch hole for the replaceable pinhole apertures.
- Using the laser cutting machine, I could create apertures of precise diameters of 0.5mm, 1mm, and 5mm.
- Ducttaped the whole box and taped a black cardboard behind the box as that side was leaking light from behind the screen.
- Glued black paper inside the box on all sides except the screen, which was glued with white paper.
- Taped the camera lens to the 6mm hole. (with tape that comes off easily so as to not damage the lens)
- The focal length of the pinhole camera (pinhole-to-screen distance) was 16 cm. The screen width and height were 23 cm and 16 cm respectively. From the screen height and the focal length, the FoV can be calculated to be around 53 degrees.

Here are some images:





2.2 Use your pinhole camera

The 3 pinhole sizes I used are 0.5mm, 1mm and 5mm. I tried 0.1mm but the images were too dark even with an exposure time of 30 seconds. I use f5.3 and ISO 1600 for all captures. I use exposure times of 1 second, 15 seconds and 30 seconds for pinhole diameters of 5mm, 1mm and 0.5mm respectively. Ignore the text on the screen that was used to focus the camera onto the screen. From the captured images, it is clear that as the pinhole diameter increases, the images get blurrier. On the other hand, images captured with a smaller pinhole are more noisy and require a large exposure because less light reaches the screen. Here are the captured images:

Figure 13: Scene1 - 0.5mm

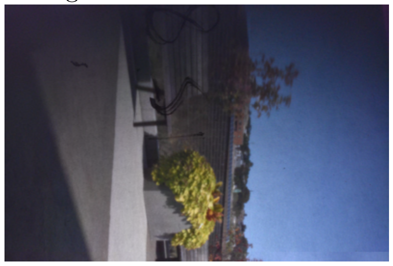


Figure 14: Scene1 - 1mm

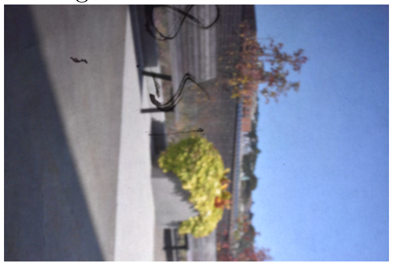


Figure 15: Scene1 - 5mm

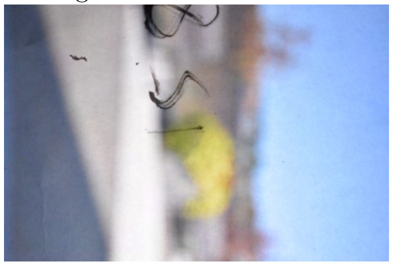


Figure 16: Scene2 - 0.5mm

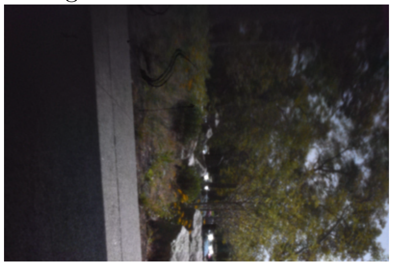


Figure 17: Scene2 - 1mm

Figure 18: Scene2 - 5mm

Figure 19: Scene3 - 0.5mm

Figure 20: Scene3 - 1mm

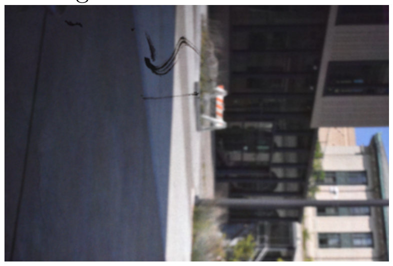
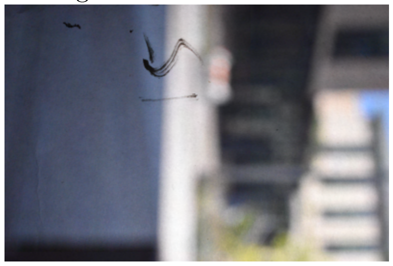


Figure 21: Scene3 - 5mm



3 Descriptions of all included images

- grayworld.png, whiteworld.png, rescalewhitebalance.png - Images developed using three different whitebalancing methods.
- manualwb1.png, manualwb2.png, manualwb3.png - Images developed by using manual whitebalancing with 3 different patches.
- dcraw.png - png file produced by dcraw.
- ph1-4.jpg - Images of constructed pinhole.
- 1.jpg to 9.jpg - Images captured using pinhole.