ASSIGNMENT 1

15663 Computational Photograpy Fall 2023
DUE: September 15, 2023

1 Developing RAW images

1.1 Implement a basic image processing pipeline (80 points)

RAW image conversion (5 points

- black: 150
- white: 4095
- r_scale: 2.394521
- g_scale: 1.000000
- b_scale: 1.597656
- g_scale: 1.000000

```
(base) cindy@cindy200517:~/15663/assignment1/data$ dcraw -4 -d -v -w -T campus.nef
Loading Nikon D3400 image from campus.nef ...
Scaling with darkness 150, saturation 4095, and
multipliers 2.394531 1.000000 1.597656 1.000000
Building histograms...
Writing data to campus.tiff ...
```

Figure 1: RAW image conversion

Python initials (5 points)

- height: 4016
- width: 6016
- number of bits per pixel: 16

```
(base) cindy@cindy200517:~/15663/assignment1$ python main.py
img height: 4016, img width: 6016, every pixel is a uint16
```

Figure 2: Image Info

Identifying the correct Bayer pattern (20 points) The green pixels that are close to each other in space should have similar values. Therefore, we calculate the averaged absolute differences of the up-left and down-right pixels ('grbg' and 'gbrg' patterns) and the up-right and down-left pixels ('rggb' and 'bggr' patterns). The result is as follows:

- 'grbg' and 'gbrg': 0.011146

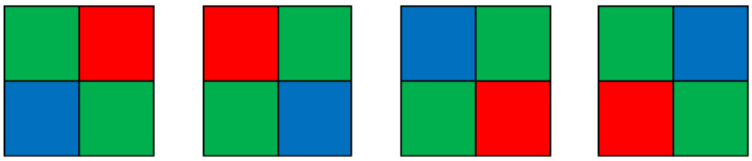


Figure 3: Possible Bayer Patterns

- 'rggb' and 'bggr': 0.005316

The values of the up-right and down-left pixels are closer to each other. Therefore, there is a higher chance that the pattern is either 'rggb' or 'bggr'.

To identify the correct Bayer pattern, we first perform white balancing assuming the pattern is 'rggb' or 'bggr', and compare the results. The images shown here were acquired using the gray world white balancing algorithm:

Figure 4: 'bggr' image after white balancing

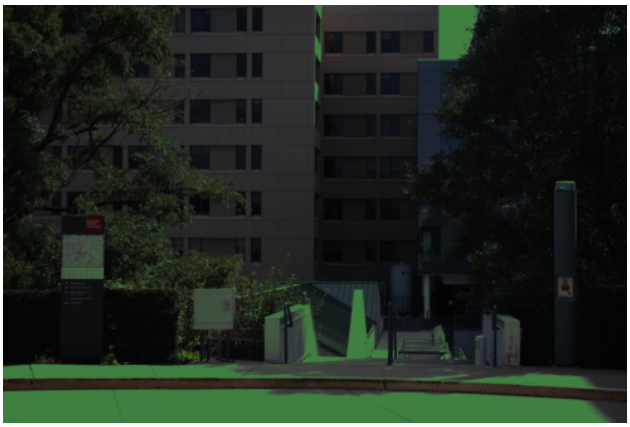


Figure 5: 'rggb' image after white balancing

If we take a look at the original image, the glass of the rightmost building appears blue ('rggb' pattern)

instead of red ('bggr' pattern). Thus we choose 'rggb' as the Bayer pattern.

White balancing (10 points) We choose gray world white balancing for the following sections.

Figure 6: White Balancing Results

Demosaicing (10 points) The demosaiced image:

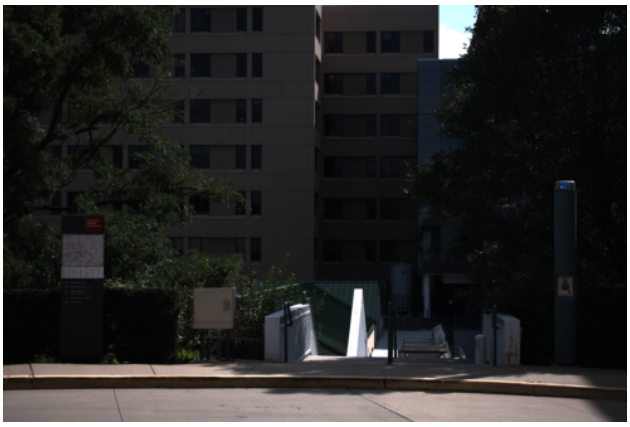


Figure 7: Demosaicing

Color space correction (10 points) The image after performing color space correction:

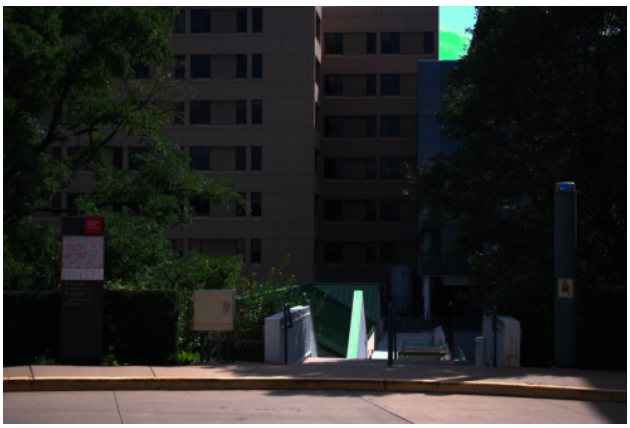
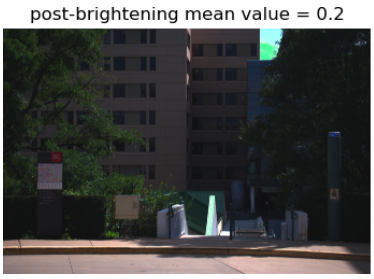
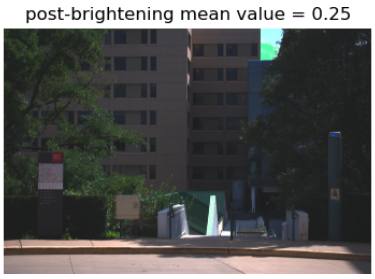


Figure 8: Color Correction

Brightness adjustment and gamma encoding (10 points) We experiment with 6 different percentages, and the result is shown as follows:





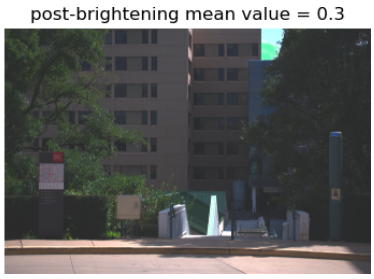
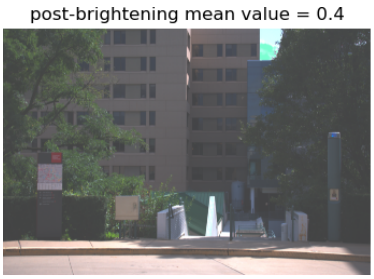


Figure 9: Brightened Images



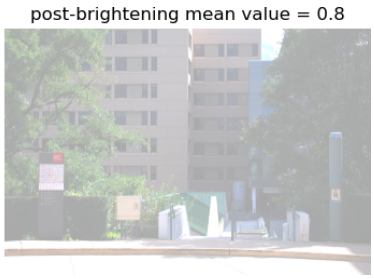


Figure 10: Brightened Images

The result when the post-brightening mean grayscale intensity is 0.3 looks best to me.

The image after gamma encoding:

Figure 11: Gamma Encoding

Compression (5 points) It is very hard to tell the difference between the two files. The compression ratio is $21789193/5460414 \approx 3.99$.

The artifacts become noticeable when quality is around 30-40, where some details (e.g. the CMU map on the left) start to disappear.

1.2 Perform manual white balancing (10 points)

We perform manual white balancing using the 4 areas shown in the image. The results are displayed as follows:

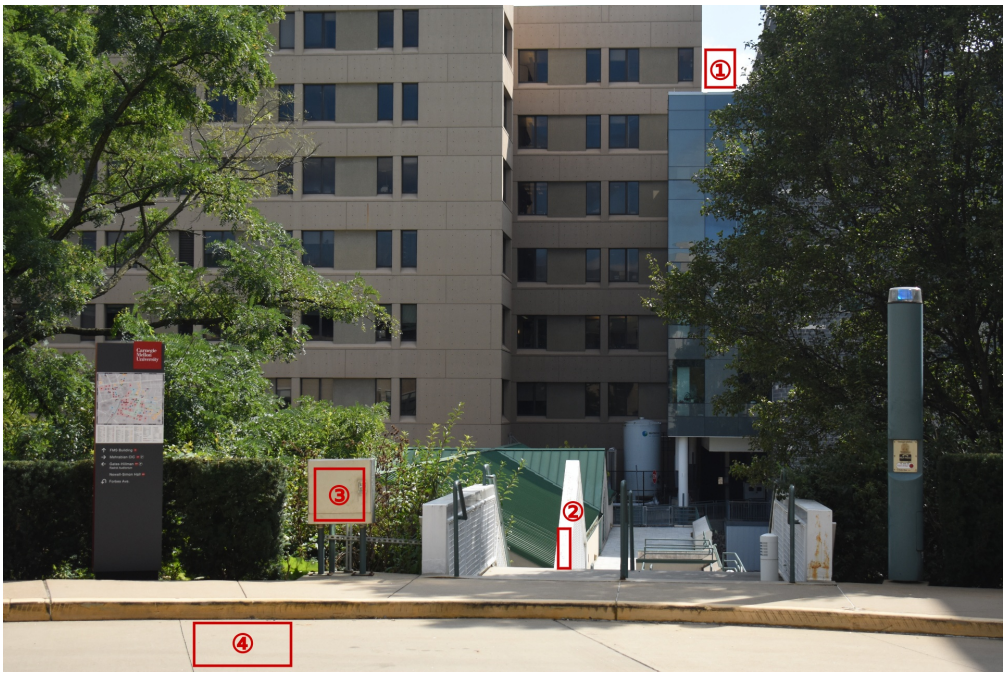


Figure 12: Campus

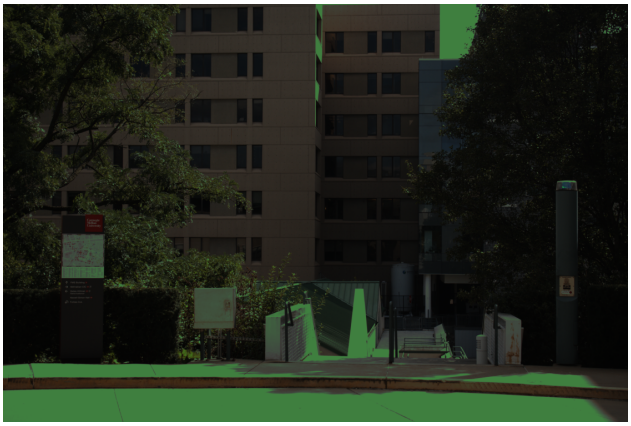


Figure 13: White balancing using area 1

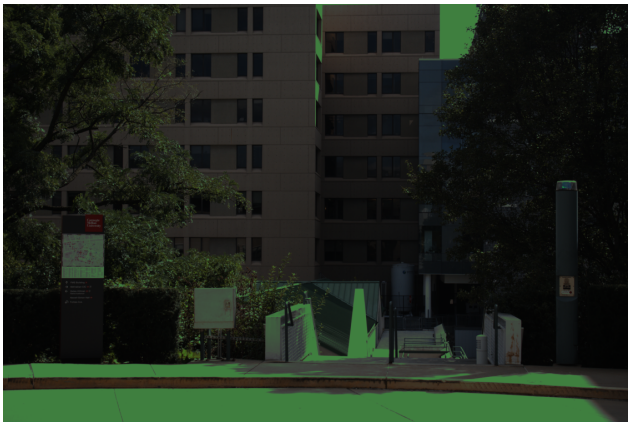


Figure 14: White balancing using area 2

Figure 15: White balancing using area 3

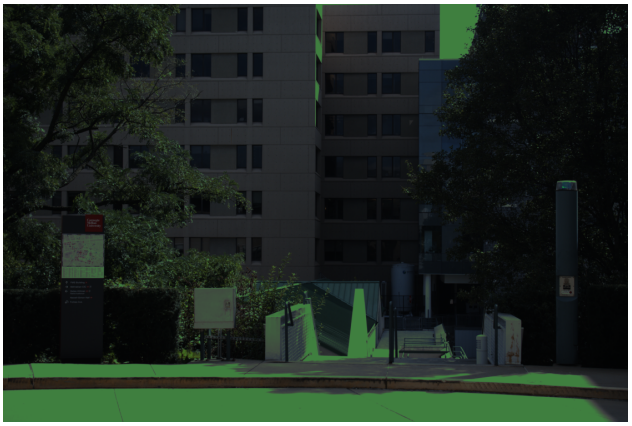


Figure 16: White balancing using area 4

1.3 Learn to use dcraw (10 points)

```
cindy@cindy200517:~/15663/assignment1/data$ dcraw -k 150 -S 4095 -a -q 2 campus.nef
```

Figure 17: dcraw flags

- -k 150: set the darkness level as 150
- -S 4095: set the saturation level as 4095
- -a: Average the whole image for white balance
- -q 2: Set the interpolation quality as 2

Figure 18: Image acquired via dcraw

2 Camera Obscura

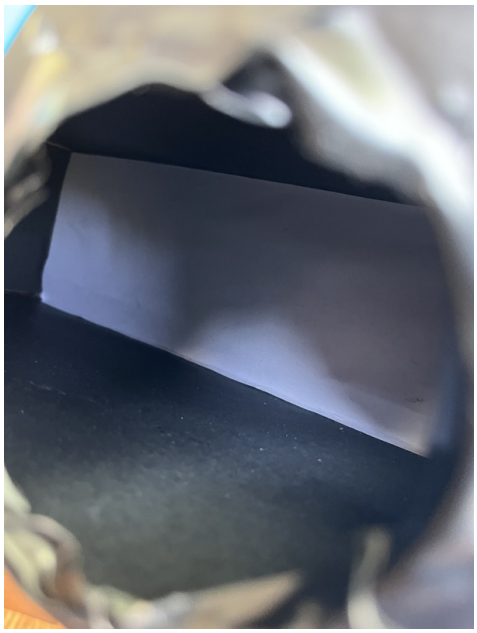
2.1 Build the pinhole camera (70 points)

Design decisions:

- screen size: 29.5cm * 8cm
- focal length: 13cm
- field of view: around 97 degrees across the horizontal axis, and around 34 degrees across the vertical axis


Easy
Tabs


Easy
Tabs



2.2 Use your pinhole camera (30 points)

Figure 19: Scene 1

Figure 20: Scene 2

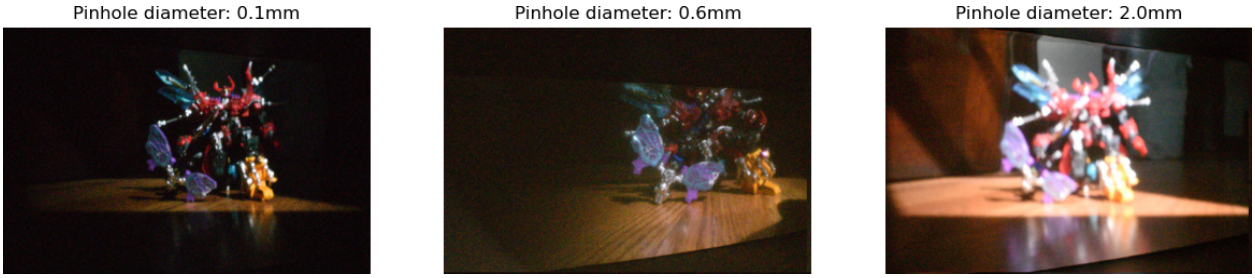


Figure 21: Scene 3