15-663 Homework Assignment 1 Report

Jingguo Liang

1 Developing RAW Images

1.1 Implement a basic image processing pipeline

RAW image conversion

The "reconnaissance run" produces the following result:

```
Scaling with darkness 150, saturation 4095, and
multipliers 2.394531 1.000000 1.597656 1.000000
```

Python initials

The image has width 6016 and height 4016, with 16 bits per pixel.

Identifying the correct Bayer pattern

I try to identify the Bayer pattern by separating the RAW image into four sub-images, each corresponding to one pixel in the 2×2 square. Then I stack 3 of them together according to the possible Bayer patterns, forming several 3-channel RGB images (with the camera preset white balancing), which is shown in figure 1. It is obvious that the image corresponding to '`rggb`' pattern is most likely to be the correct image.

White balancing

Figure 2 shows the final developed image using three different white balancing algorithms. The color of the white world assumption image is clearly off. And between the remaining two images, I prefer the one using the camera white balancing preset. Its color looks warmer, and it looks more likely as an image that is taken outside on a sunny day.

Brightness adjustment and gamma encoding

After some experiments, I decide that the image with mean grayscale indensity equal to 0.4 looks the best to me. This brightness adjustment requires scaling up the brightness by factor of approximately 5.39.

Compression

With `quality` parameter set to 95, I can hardly notice any difference between two images unless I zoom in and examine pixel by pixel. Even if I notice some difference in pixels around edges of objects (for example, leaves), the difference is not significant enough for me to say that the `.JPEG` image is more blurred than the `.PNG` image.

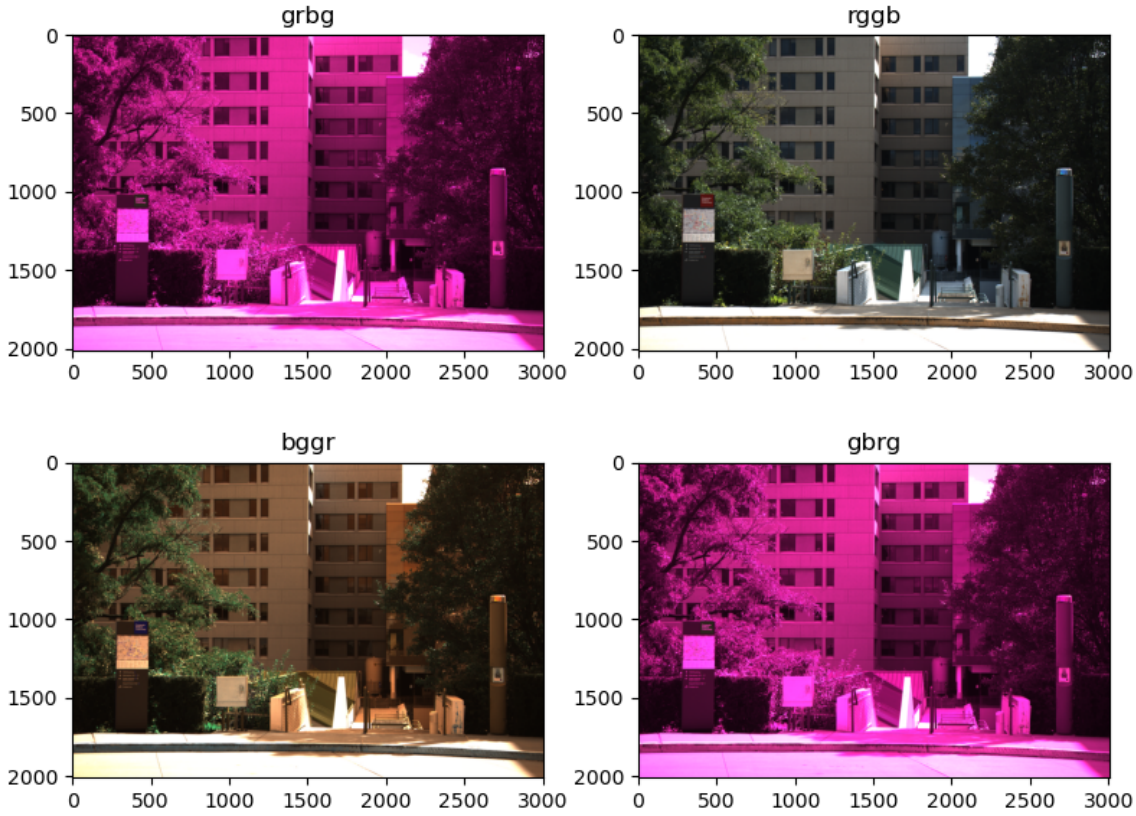


Figure 1: Bayer pattern

The size of the uncompressed file is 35761230 bytes, and that of the compressed file is 8024064 bytes. The compression ratio is about 4.457.

Starting from `quality` 80, I start to notice some artifacts in the image. When zoomed in, the image seems to be divided into small grids, and the transition between grids is not smooth. In this case, the compressed image has size 3200729 bytes, with compression ratio 11.173. This artifact become much more obvious when `quality` is lowered to 60, and is visible even without zooming in when `quality` is lowered to 30.

1.2 Perform manual white balancing

Figure 3 shows the final developed image with manual white balancing using different patches. They uses patches of points in the sky, on the sidewall of the stair, and on the ground to perform white balancing, respectively. The image using points in the sky has an orange

(a) White world (b) Gray world (c) Camera preset

Figure 2: Three white balancing algorithms

(a) Sky (b) Stair (c) Ground

Figure 3: Manual white balancing

appearance, because the color of the sky tends to be blue, meaning the red channel and green channel has smaller means. The image using points on the ground has a blue appearance, because the color of the ground tends to be yellow, meaning the blue channel has a smaller mean. The image using points on the stair looks generally the best.

1.3 Learn to use dcraw

I run `dcraw` with the following command:

```
.\dcraw.exe -w -k 150 -S 4095 -o 1 -W -b 5.39 .\campus.nef
```

And figure 4 shows the output image. The image 2c produced through my implemented pipeline looks the brightest, with the `dcaw` one being slightly darker. The image from the camera's pipeline is much darker. This is likely to be due to the different brightness adjustment and gamma correction settings used in those different pipelines.

2 Camera Obscura

2.1 Build the pinhole camera

Figure 5 shows the pinhole camera built by me. The screen size of this pinhole camera is 235 mm $\times$ 200 mm, with depth 120 mm. The lens is close to the screen, so I use the smalles

Figure 4: Image produced by dcraw

focal length possible (18 mm) to maximize the field of view.

2.2 Use your pinhole camera

Figure 6 shows the three pinholes that I use. They have diameters of 1 mm, 4 mm, 7 mm respectively. Due to the limitation by the tools that I have, the 1 mm pinhole is the smallest that I can have.

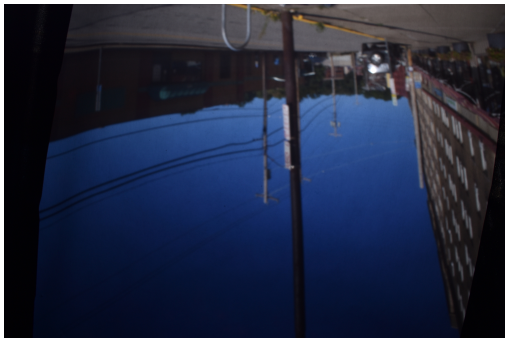
Figure 7, 8, 9 shows three scenes taken by the pinhole camera. Within each group of images, the first one is taken directly by the camera, and the remaining three are taken using the pinhole camera, with three different pinhole sizes. All pinhole camera images are taken with exposure time of 30 s. We observe that among the three pinholes, the one with 1 mm

Figure 5: The pinhole camera

Figure 6: Three pinholes

diameter produces the clearest image, while the image gets more blurred as the diameter increases. This is possibly because of the fact that a single point will be visible to a larger area of the screen with a larger pinhole, making the image looks blurred. We also observe that image is brighter with larger pinholes, as more light will go through larger pinholes.

(a) Scene 1, direct shot



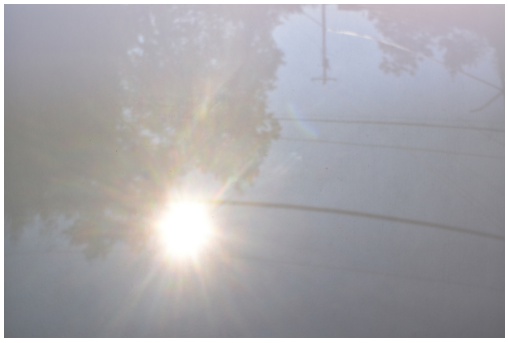
(b) Scene 1, 1mm pinhole

(c) Scene 1, 4mm pinhole

(d) Scene 1, 7mm pinhole

Figure 7: Scene 1

(a) Scene 2, direct shot



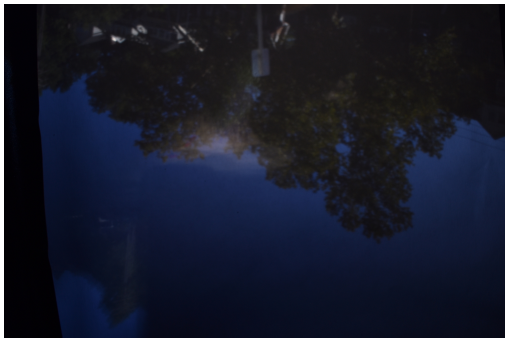
(b) Scene 2, 1mm pinhole

(c) Scene 2, 4mm pinhole

(d) Scene 2, 7mm pinhole

Figure 8: Scene 2

(a) Scene 3, direct shot



(b) Scene 3, 1mm pinhole

(c) Scene 3, 4mm pinhole

(d) Scene 3, 7mm pinhole

Figure 9: Scene 3