15663 Computational Photography
Homework 1

1. Developing RAW Images

1.1. Implement a basic image processing pipeline

RAW Image Conversion:

Make sure to record the integer numbers for <black>, <white>, <r scale>, <g scale>, and <b scale>, and include them in your report

<black> = 150 <white> = 4095
<r_scale> = 2.394531 <g_scale> = 1.000000 <b_scale> = 1.597656 <g_scale> = 1.000000

Python Initials:

Check and report how many bits per pixel the image has, its width, and its height.

The image has 16 bits per pixel, with shape 4016 x 6016

Identifying the correct bayer pattern:

Think of a way for identifying which Bayer pattern applies to your image file, and report the one you identified.

I generated images after applying demosaicing and white balance with different Bayer patterns. The image produced using the ‘RGGB’ pattern appears visually accurate. I further verified this using dcraw.

RGGB

GRBG

GBRG

BGGR

Bayer pattern verification using dcraw :

```
(base) revanthvarma@revanths-air data % dcraw -v -i campus.nef

Filename: campus.nef
Timestamp: Fri Sep  3 00:55:46 2021
Camera: Nikon D3400
ISO speed: 400
Shutter: 1/1600.0 sec
Aperture: f/5.0
Focal length: 40.0 mm
Embedded ICC profile: no
Number of raw images: 1
Thumb size:  6000 x 4000
Full size:   6016 x 4016
Image size:  6016 x 4016
Output size: 6016 x 4016
Raw colors: 3
Filter pattern: RG/GB
Daylight multipliers: 2.231936 0.932648 1.141543
Camera multipliers: 2.394531 1.000000 1.597656 0.000000
```

White Balancing and Demosaicing:

Check what the image looks like when using each of the three white balancing algorithms, decide which one you like best, and report your choice.

I prefer the image generated using the RGB scaling values provided by dcraw, which are the camera's preset values. The image produced by the gray world algorithm is also fairly good but has a bluish tint. However, the image created by the white world algorithm appears very greenish.

Image generated using gray world white balancing algorithm (brightness scaled linearly for visualization)

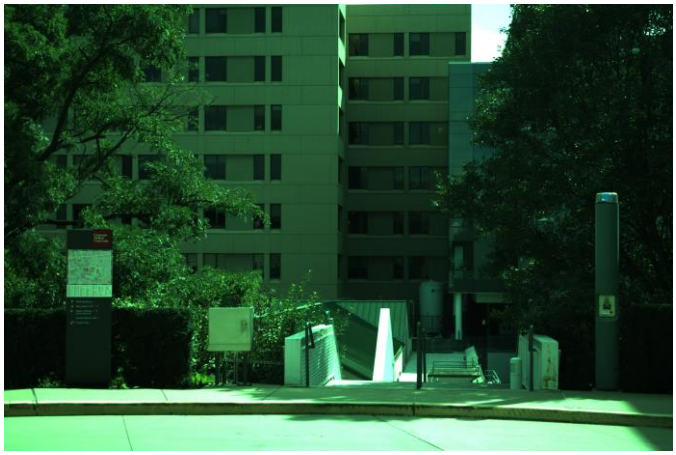


Image generated using white world white balancing algorithm (brightness scaled linearly for visualization)

Image generated using white balance presets of camera

From now on, images will be generated using AWB parameters from dcraw

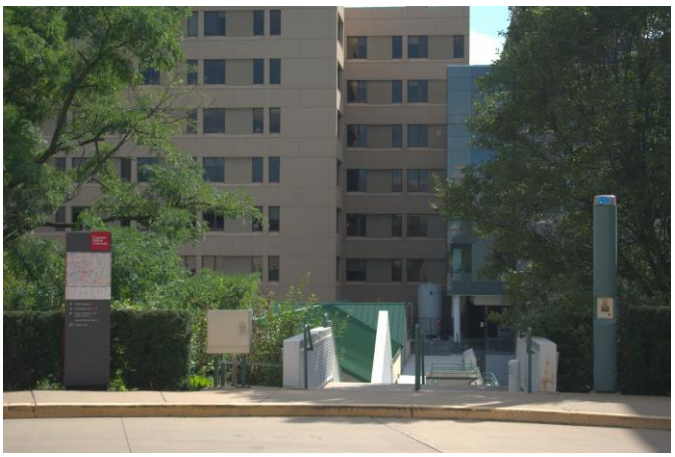
Color Space Correction:

Color corrected image (brightness scaled linearly for visualization)

Brightness Adjustment and Gamma Encoding:

You should experiment with many percentages and report and use the one that looks best to you.

I tried with several brightness target values, and among them target value of 0.15 looks good to me



Gamma Encoded Image

Compression:

Can you tell the difference between the two files?

Both files look exactly same visually, but the jpg file takes significantly less disk space compared with png file

What is the compression ratio?

Uncompressed image size is 33606513 bytes

Compressed image size is 2034720 bytes

Compression ratio = 16.516529547062987

By changing the JPEG quality settings, determine the lowest setting for which the compressed image is indistinguishable from the original. What is the compression ratio?

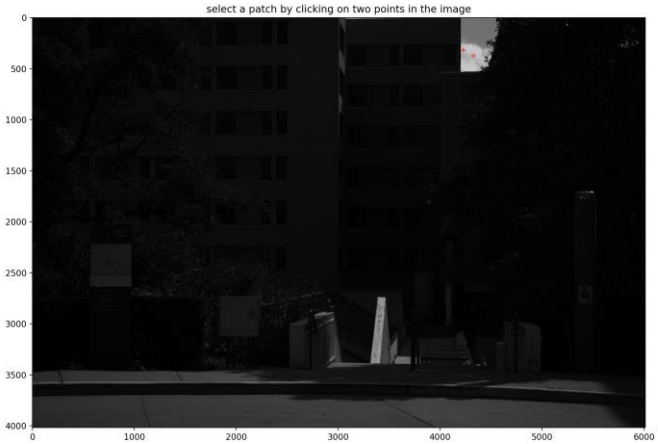
The lowest jpeg quality setting I can use is 45%. Beyond this, when I zoom in on the image, small artifacts become visible.

Uncompressed image size is 33606513 bytes

Compressed image size is 1141927 bytes

Compression ratio = 29.429650932152406

1.2. Perform Manual White Balancing



Zoom in to see the patch used (marked with red points)

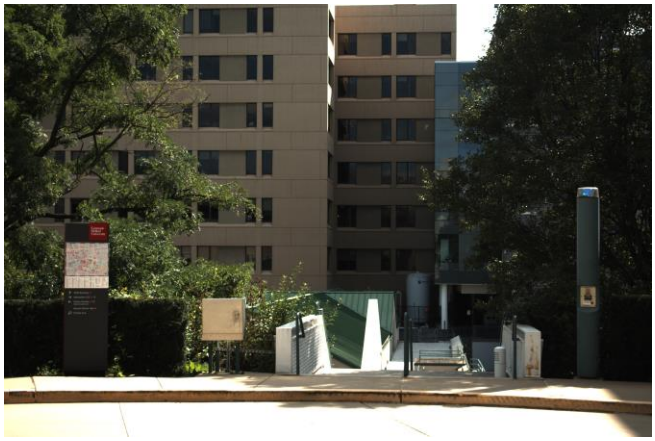
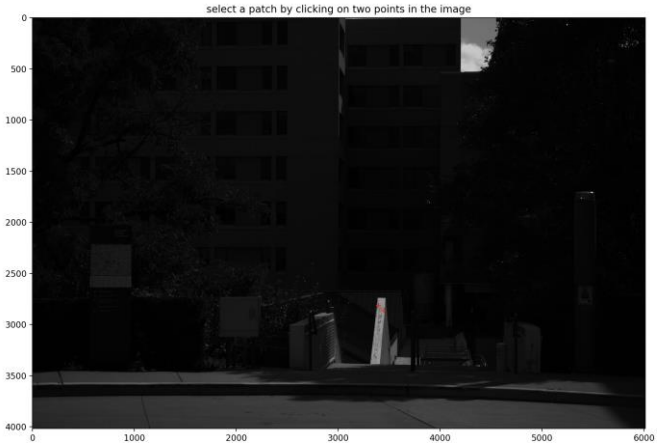


Image generated with white balance parameters calculated using a patch from clouds (brightness scaled linearly for visualization)



Zoom in to see the patch used (marked with red points)

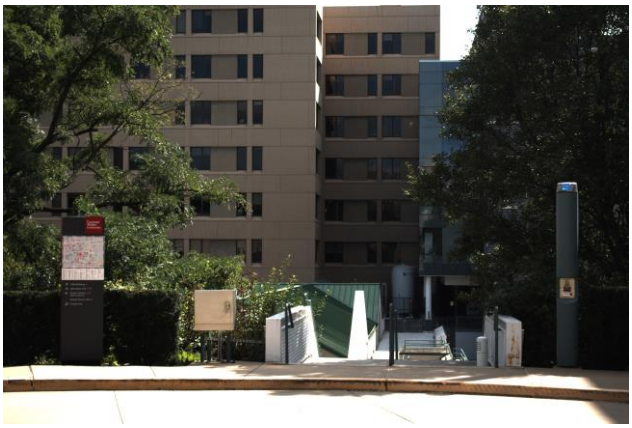
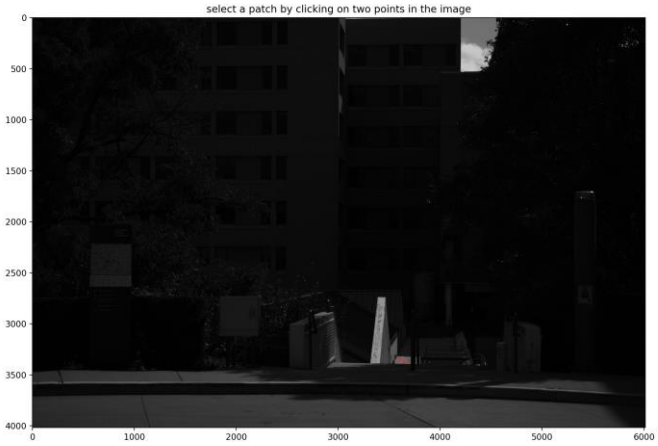


Image generated with white balance parameters calculated using a patch from wall (brightness scaled linearly for visualization)



.

Zoom in to see the patch used (marked with red points)

Image generated with white balance parameters calculated using a patch from wall (brightness scaled linearly for visualization)

The patch selected from the ground area provides more accurate natural, while the other two appear slightly reddish.

1.2. Learn to use dcraw

Linearization : It is done automatically by dcraw, no specific flag needed

Bayer Pattern : dcraw –v –i campus.nef

White Balance : dcraw –w campus.nef (uses cameras auto white balance params)

Demosaicing: dcraw –q 3 campus.nef (-q ranges from 0 to 3, with 3 being the highest interpolation quality)

Color Correction: dcraw –o 1 campus.nef (Output colorspace – raw, sRGB, Adobe, Wide, ProPhoto, XYZ, ACES)

Gamma Encoding: dcraw –g 2.4 12.92 campus.nef (2.4 represents the gamma power and 12.92 represents the toe slope)

Writing to .ppm file: dcraw by default saves the file in ppm format, unless –T is specified in which it saves the file in TIFF format

```
(base) revanthvarma@revanths-air data % dcraw -v -w -q 3 -o 1 -g 2.4 12.92 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
AHD interpolation...
Converting to sRGB colorspace...
Writing data to campus.ppm ...
```

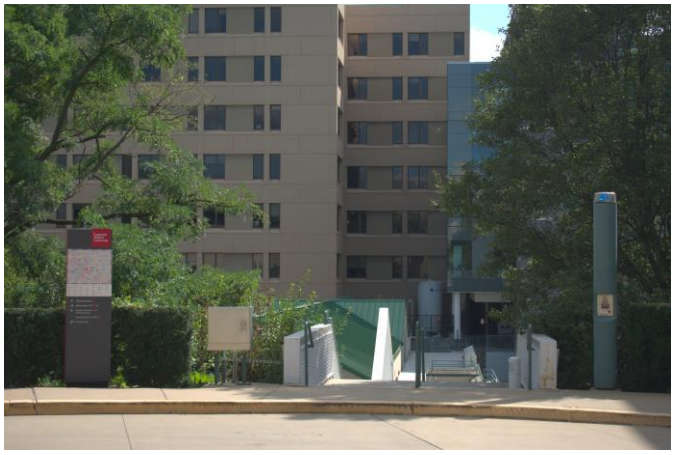


Image generated using functions from dcraw

Compare the three developed images, and explain any differences between them. Which of the three images do you like best?

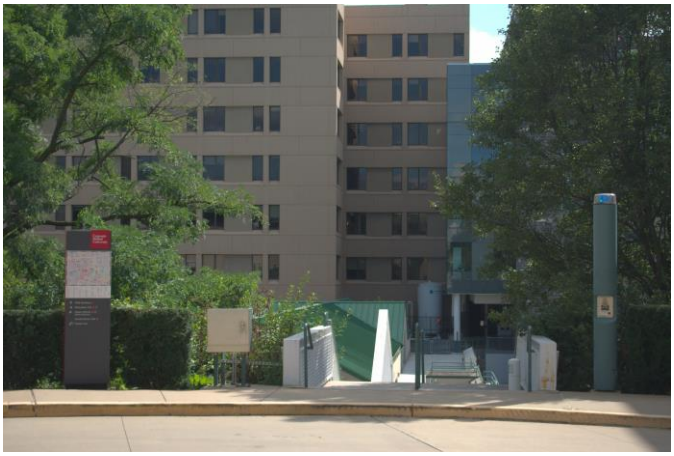


Image generated using functions with my pipeline

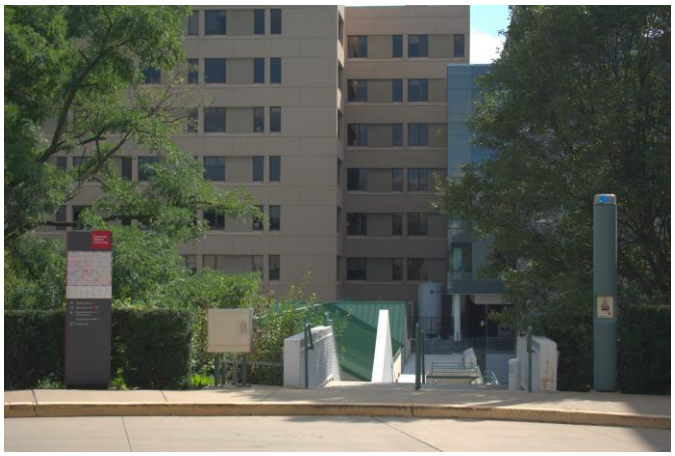


Image generated using functions from dcraw

Image generated using functions with camera's pipeline

Among the three developed images, the one generated using the camera's image processing pipeline looks the best visually, especially the contrast looks well balanced. The contrast in the other two images seems slightly off, likely due to the assumption that the colored space is sRGB, which we are not sure of.

As for the images generated by dcraw and my pipeline, they are almost identical since I used the same parameter values in both cases. However, the dcraw image appears slightly more sharper, probably because I chose a high quality interpolation (param q = 3)

2. Camera Obscura

2.1. Build the Pinhole Camera

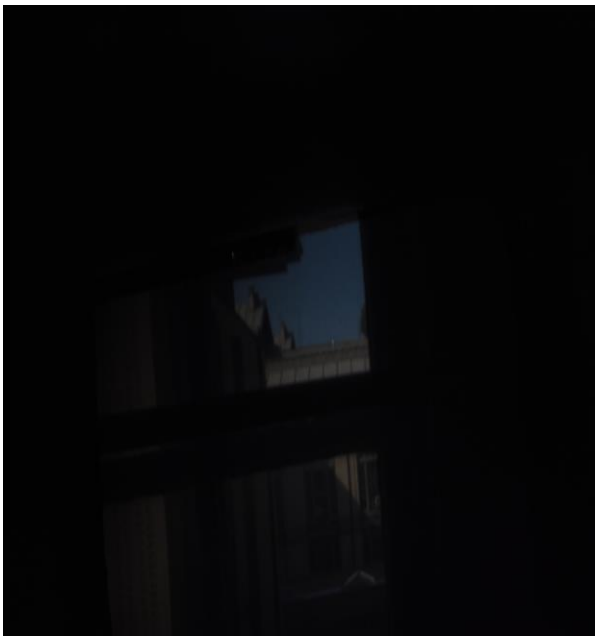
Once you have completed everything, submit a few photos of your constructed pinhole camera. Make sure to also report on all your design decisions, such as screen size, focal length, and field of view.

Earlier, I attempted the setup without painting the inner walls with black spray paint, but the image didn't form properly. So, I covered the interior of the box with black spray paint to prevent further light reflections inside. I had previously made a pinhole camera with a smaller box, but the image formed was too small to be captured by the camera, so I took a bigger box. The screen size is approximately 10 cm, although there was no specific reason for choosing this size. Given the screen size and the distance between the camera and the screen, I experimented with different focal lengths, and a focal length of 35 mm worked well. It perfectly captured my entire screen with minimal areas outside the frame, which is also why I chose this particular field of view.

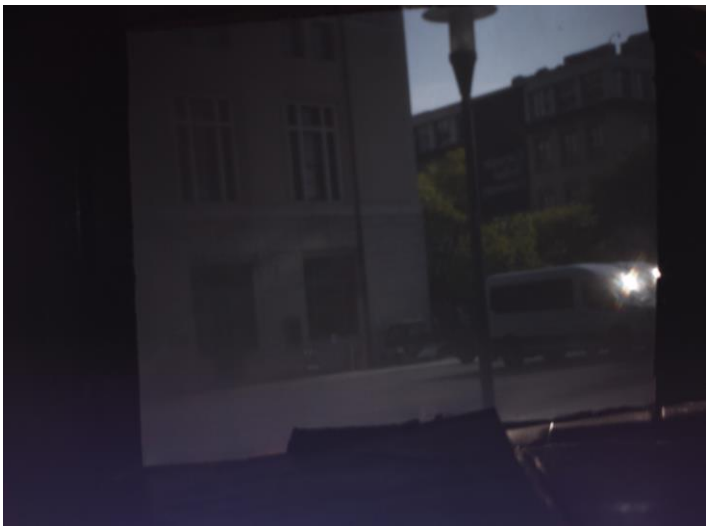
2.2. Use your Pinhole Camera

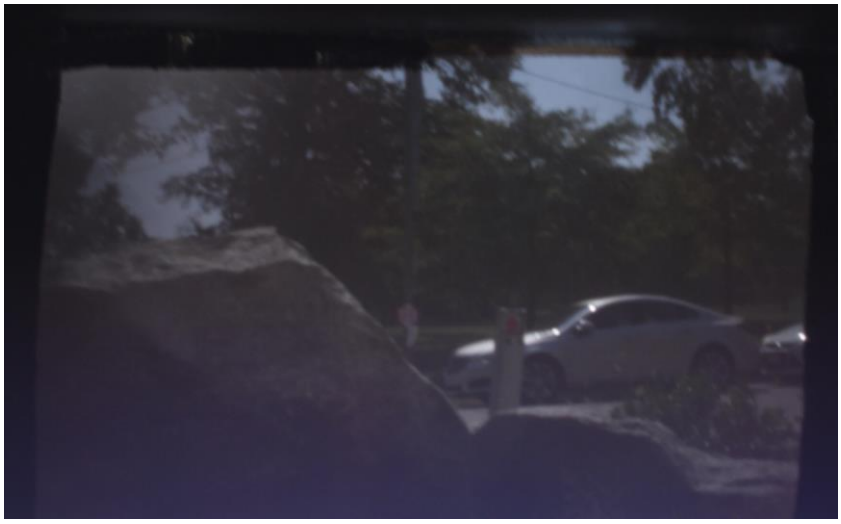
I tried with pinhole sizes of 0.1mm, 1mm and 5mm.

Images captured with 0.1mm:

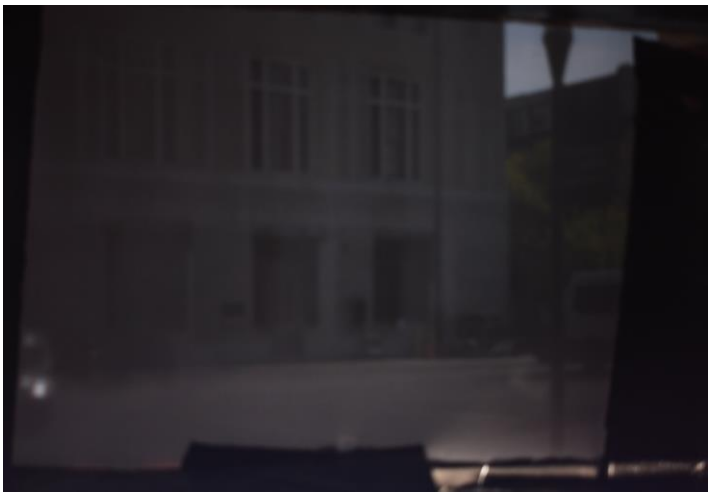


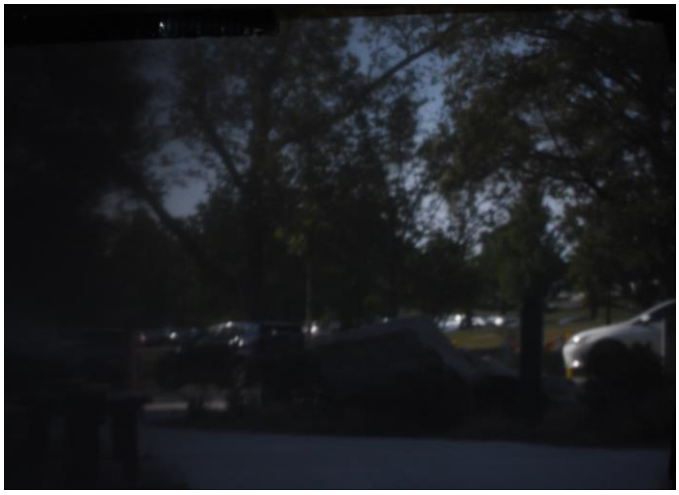
Images captured with 1mm:





Images captured with 5mm:





The images captured with 0.1mm pinhole were the sharpest among the three. As the pinhole size increased, the sharpness reduced, and the images became progressively blurrier. Also, as the pinhole size grew, more light entered the camera, and hence I had to reduce the shutter time to avoid overexposing the images at larger diameters.