# Raw Image Decoder

Beatriz Aline Arend

April 2, 2025

## 1 Introduction

The goal of this report is to explore how digital cameras acquire and process images. The implementation is done in Python3, utilizing the `rawpy` library to handle `.dng` files and `numpy` for pixel-level operations.

## 2 Raw Image Decoder

### 2.1 Converting .CR2 to .DNG

Using Adobe's Digital Negative (DNG) Converter, the file scene_raw.CR2 was converted into the DNG format, as shown in Figure 1.

Figure 1: Converted raw image.

### 2.2 Demosaicking

Demosaicking reconstructs a full-color image from a color filter array (CFA). The GRBG pattern (Figure 2) is used, where:

Red and blue pixels interpolate green from adjacent horizontal and vertical neighbors, and their opposite color from diagonals.

Green pixels interpolate red from left/right neighbors in even rows and top/bottom in odd rows, while blue is interpolated from top/bottom in even rows and left/right in odd rows.

Bilinear interpolation was applied to fill in missing color values. The result is displayed in Figure 3.

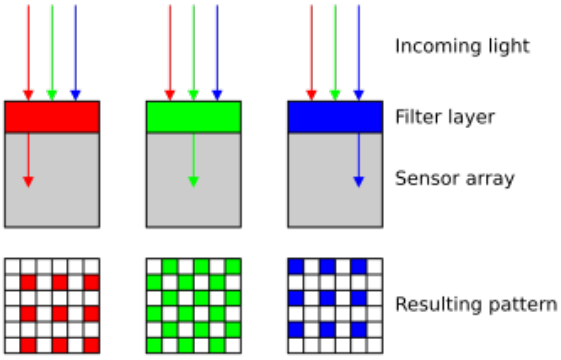


Figure 2: GRBG CFA.

Figure 3: Demosaicked image.

The demosaicking process left the image with a greenish tint, and it does not appear visually accurate. To correct this, we will apply white balance.

## 2.3 White Balance

White balance adjusts colors to neutralize color casts under different lighting conditions. Two methods were implemented:

**Gray World**: Assumes the average of RGB values in an image should be gray. The color gains are adjusted to balance the image (Figure 4).

**White Patch**: Assumes the brightest area of the image represents white. The image is adjusted based on this reference (Figure 5).

Figure 4: White balance using Gray World.

Figure 5: White balance using White Patch.

After comparing the results, we found that the Gray World method provided a more visually accurate correction. The White Patch method was less effective due to the strong green hue in the original image. Therefore, we will proceed with the Gray World method in the following steps.

## 2.4 Gamma correction

Gamma correction adjusts image brightness and contrast through a nonlinear transformation of pixel values. Different $\gamma$ values can significantly alter the perception of the image:

$\gamma = 0.5$ results in an image that is overly bright and loses saturation;

$\gamma = 0.75$ produces an under-saturated appearance, making the image look dull;

$\gamma = 1.25$ makes the image too dark, causing a loss of details in highlights.

After testing various values, $\gamma = 0.85$ was found to offer the best balance between brightness and contrast, preserving details while maintaining some natural looks.

Figure 6: Gamma correction comparison: $\gamma = 0.5$ (Brighter), $\gamma = 0.75$ (Moderate), $\gamma = 1.25$ (Darker).

Figure 7: Gamma correction with $\gamma = 0.85$.

## 2.5 Conclusion

Achieving an accurate image from raw data is challenging, with white balance seeming to be the most critical and subjective step.

Through demosaicking, white balance, and gamma correction, we refined the image. In my observation, the Gray World method and a gamma of 0.85 provided the best balance of color accuracy, brightness, and detail preservation.