EE5176: Computational Photography
Programming Assignment-1: Basics of Imaging Report

Ritabrata Mandal
EE25S009

August 28, 2025

1. **Introduction**
   This assignment focuses on fundamental concepts in digital imaging, beginning with demosaicing and extending to noise removal. All processed images generated during the assignment are available in the GitHub repository under the `output` folder.

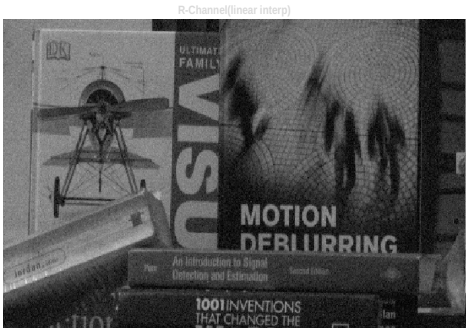
   (a) **Implementation**:
   The implementation has been carried out and tested in MATLAB R2025a. A `main.m` script invokes four functions: `demos.m`, `white_balance.m`, `tone_map.m`, and `denoising.m`. The `demos.m` function performs image demosaicing for two given raw images, `RawImage1` and `RawImage2`. The functions `white_balance.m` and `tone_map.m` apply white balancing and tone mapping techniques, respectively, on images such as `RawImage1` and `kodim19`.

   (b) **Running the Code**:
   To execute the assignment, navigate to the `Assignment1` directory, run the `main.m` file, and select options `1`, `2`, or `3` corresponding to the three tasks.

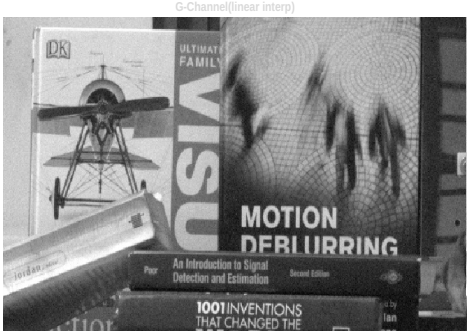
2. **Demosaicing**
   Demosaicing is a process of interpolating the red, green and blue channels from the raw pixel image. Here we will try to ll in the missing pixel values from the sampled raw images through interpolation.
   The CFA for `RawImage1.mat` is rggb whose mask is provided in the le `bayer1.mat`. In `bayer1.mat` a value of 1 at a particular pixel means that only red value is sampled at that pixel. Similarly values of 2 and 3 mean that only green and blue values are sampled at those pixels. We use bilinear and bicubic interpolation for filling in the missing data. We interpolate only for those pixels whose values are missing in the grid. We use the MATLAB built-in function `griddata` for an efficient implementation.

   (a) Performing bilinear interpolation of the missing pixels in each color channel R, G and B to reconstruct a full color image from RawImage1 with the CFA provided in `bayer1.mat` and figure-1 shows the different color channel after interpolation. Figure 1d shows the RGB reconstruction with linear interpolation

   (b) Performing bicubic interpolation of the missing pixels in each color channel R, G and B to reconstruct a full color image from RawImage1 with the CFA provided in `bayer1.mat` and figure-2 shows the different color channel after interpolation. Figure 2d shows the RGB reconstruction with cubic interpolation.
   Comparision with Bilinear Interpolation: When comparing the results of bicubic interpolation and bilinear interpolation, the images appear almost identical at first glance. However, on closer inspection by zooming in, bicubic interpolation produces sharper edges with less blurriness than bilinear interpolation. In general, unless we specifically look for these differences at higher magnification, it is difficult to distinguish between the two images.

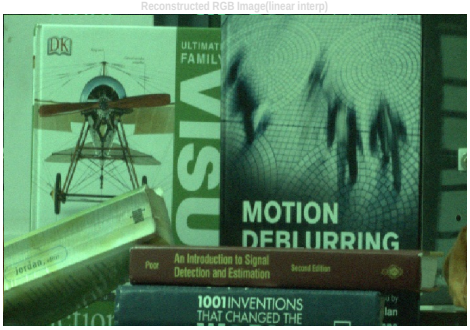


(a) R-Channel



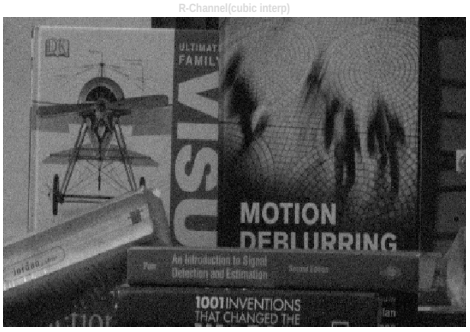
(b) G-Channel

(c) B-Channel

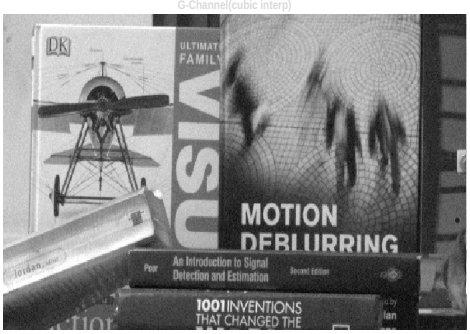


(d) Reconstructed RGB

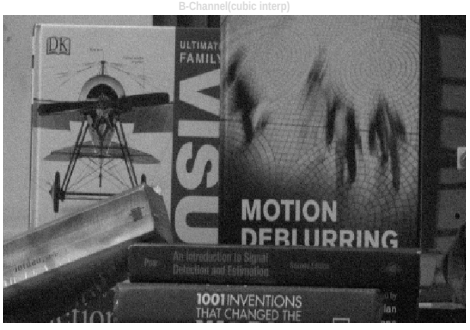
Figure 1: Reconstructed `RawImage1` with linear interpolation.



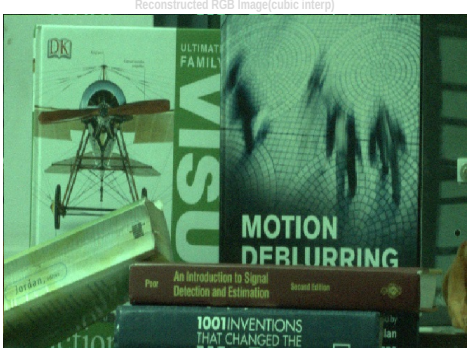
(a) R-Channel



(b) G-Channel



(c) B-Channel



(d) Reconstructed RGB

Figure 2: Reconstructed `RawImage1` with cubic interpolation.

(c) As shown in Figure 3, the `RawImage1` was demosaiced using the rggb pattern with MATLAB's built-in function `demosaic`.

Comparison with Interpolation Methods:

The three interpolation methods produce images that look quite similar, with differences only noticeable upon zooming in. In such cases, the `demosaic` function retains edge information more effectively, giving sharper results. Since `RawImage1` contains little high-frequency content, even bilinear interpolation performs well, though tone-mapped outputs from `demosaic` show a purple hue compared to the more neutral colors of bicubic interpolation.

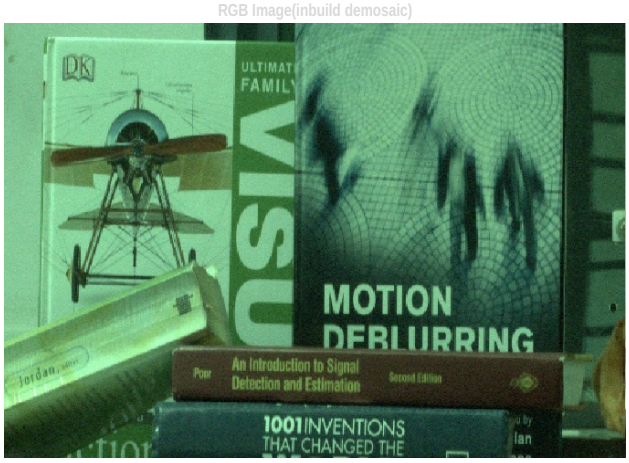


Figure 3: Reconstructed `RawImage1` with `demosaic` function

(d) **Assumptions in Interpolation of Missing Pixel Values:**

i. **Local Smoothness:** The image intensity is assumed to vary smoothly in a local neighborhood.

ii. **Correlation of Neighboring Pixels:** Adjacent pixels are assumed to have similar values, especially within homogeneous regions.

iii. **Limited High-Frequency Content:** The image is assumed to have relatively low noise and not contain abrupt intensity changes at small scales.

iv. **Edge Continuity:** Edges are assumed to be continuous, allowing interpolation methods to preserve structure without introducing artifacts.

Consequences When Assumptions Do Not Hold:

i. **Loss of Detail:** If the image has high-frequency textures, interpolation may blur fine details.

ii. **Artifacts:** Discontinuities in intensity (e.g., at edges) may cause artifacts such as color fringing or checkerboard patterns.

iii. **Distortion of Structures:** Interpolated pixels may not follow the true underlying structure, leading to edge distortion.

iv. **Noise Amplification:** In the presence of noise, interpolation may reinforce or spread the noise instead of reconstructing the true signal.

(e) Performing bicubic interpolation on `kodim19.mat` which has been sampled with the CFA pattern 'RGGB' to reconstruct a full color image shown in Figure 4.

(f) Now taking the Figure 4d converting to YCrCb and applying the median filter to the chrominance channels and then converting back it to RGB channel I get Figure 5

(g) in the 4d image (i.e. cubic interpolated rgb) we can see the moire pattern where as in 5 image (i.e. after applying median filter in chrominance channel) there is no moire pattern in the image.

3. **White Balancing and Tone Mapping**

In this section we do white balance and tone mapping for images `RawImage1`, `RawImage2` and `RawImage3`. In build `demosaic` function is used to demosaic the images. There are basically three ways to do white balancing.

- Assume the average color of the scene to be gray and then do white balancing.
- Assume that the brightest pixel is a specular highlight and hence should therefore be white
- Assume that some part of the scene to be neutral.

After white balancing the images, tone mapping is performed on them. We try two methods of tone mapping.

- Histogram equalization
- Gamma correction for the $\gamma \in \{0.5, 0.7, 0.9\}$

(a) In this method, the underlying assumption is that the average color of a natural scene is gray. Accordingly, the image is corrected so that its average color also shifts to gray. This is done by computing the mean value of each channel, estimating the necessary transforms to equalize these means, and then applying the transforms to every pixel in the corresponding channels. Figure 6b shows the white-balanced result of `RawImage1` under the gray-world assumption, while Figures 7b and 8b present the corresponding results for `RawImage2` and `RawImage3`, respectively.

(b) In this method, the brightest pixel is assumed to be a specular highlight and is therefore considered white. Using the specified pixel coordinates $(830, 814)$ for `RawImage1`, $(1165, 280)$ for `RawImage2`, and $(175, 675)$ for `RawImage3` the corresponding white balance results are obtained. These are shown in Figure 6c, Figure 7c, and Figure 8c, respectively.

(c) In this method, it is assumed that a certain part of the scene is neutral. The chosen pixel coordinates are: $(2000, 435)$ for `RawImage1`, $(445, 715)$ for `RawImage2`, and $(1550, 565)$ for `RawImage3`. Using these specified coordinates, the corresponding white balance results are obtained, as shown in Figure 6d, Figure 7d, and Figure 8d, respectively.

(d) Done tone mapping after white balancing of `RawImage1`, `RawImage2` and `RawImage3` and can be see in Figure 9, Figure 10 and Figure 11 respectively.

(a) R-Channel

(b) G-Channel

(c) B-Channel

(d) Reconstructed RGB

Figure 4: Reconstructed `kodim19.mat` with cubic interpolation.

Figure 5: After applying the Median filter to the chrominance channel.

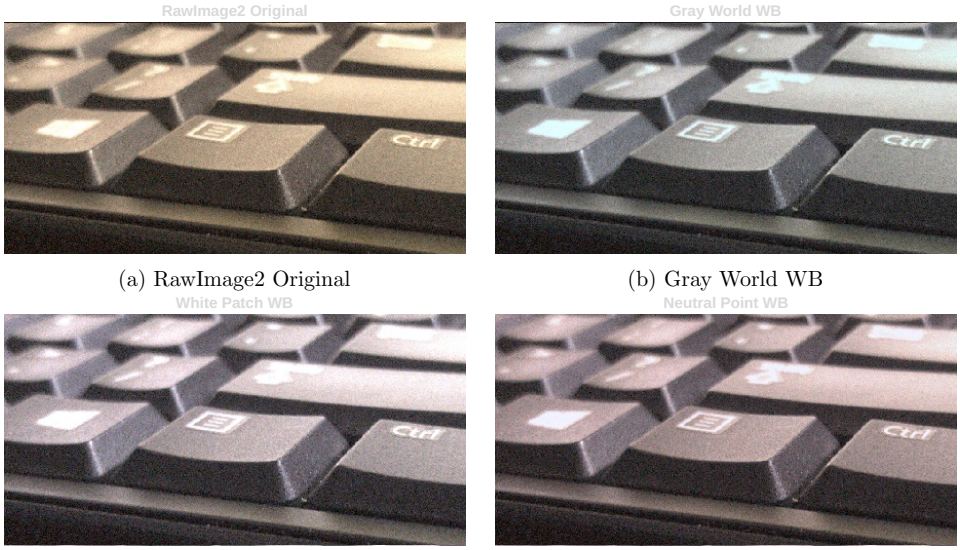
(a) RawImage1 Original

(b) Gray World WB

(c) White Patch WB

(d) Neutral Point WB

Figure 6: `RawImage1`-White Balance



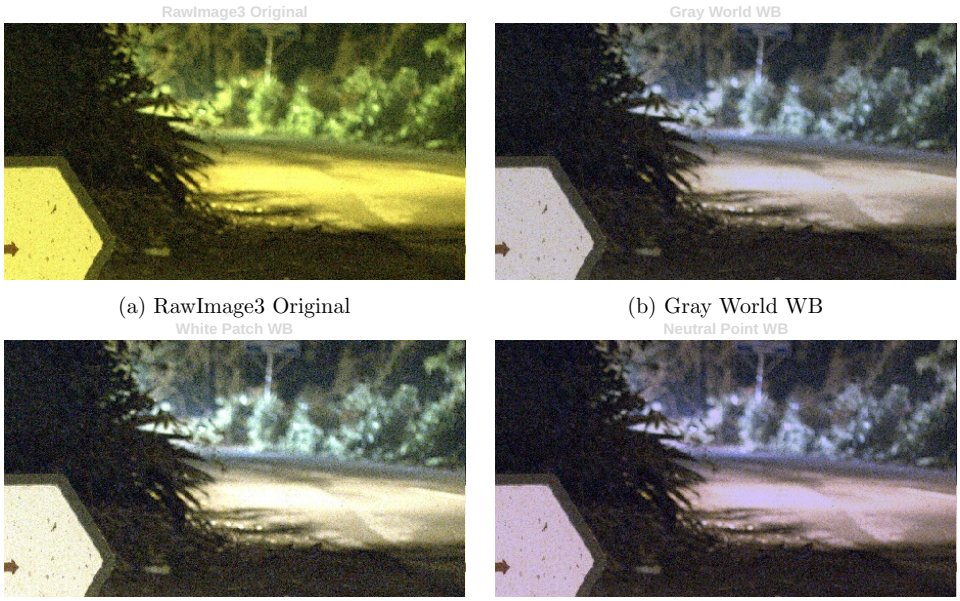
(a) RawImage2 Original

(b) Gray World WB

(c) White Patch WB

(d) Neutral Point WB

Figure 7: `RawImage2`-White Balance



(a) RawImage3 Original (b) Gray World WB

(c) White Patch WB (d) Neutral Point WB

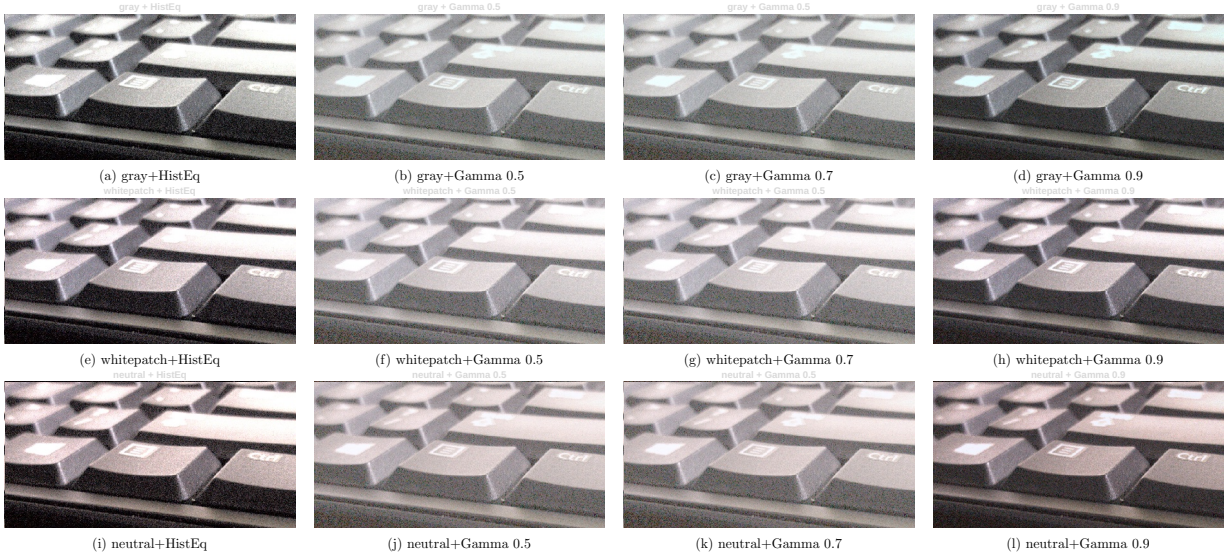
Figure 8: `RawImage3`-White Balance

(a) gray+HistEq (b) gray+Gamma 0.5 (c) gray+Gamma 0.7 (d) gray+Gamma 0.9

(e) whitepatch+HistEq (f) whitepatch+Gamma 0.5 (g) whitepatch+Gamma 0.7 (h) whitepatch+Gamma 0.9

(i) neutral+HistEq (j) neutral+Gamma 0.5 (k) neutral+Gamma 0.7 (l) neutral+Gamma 0.9

Figure 9: `RawImage1`-tone mapping

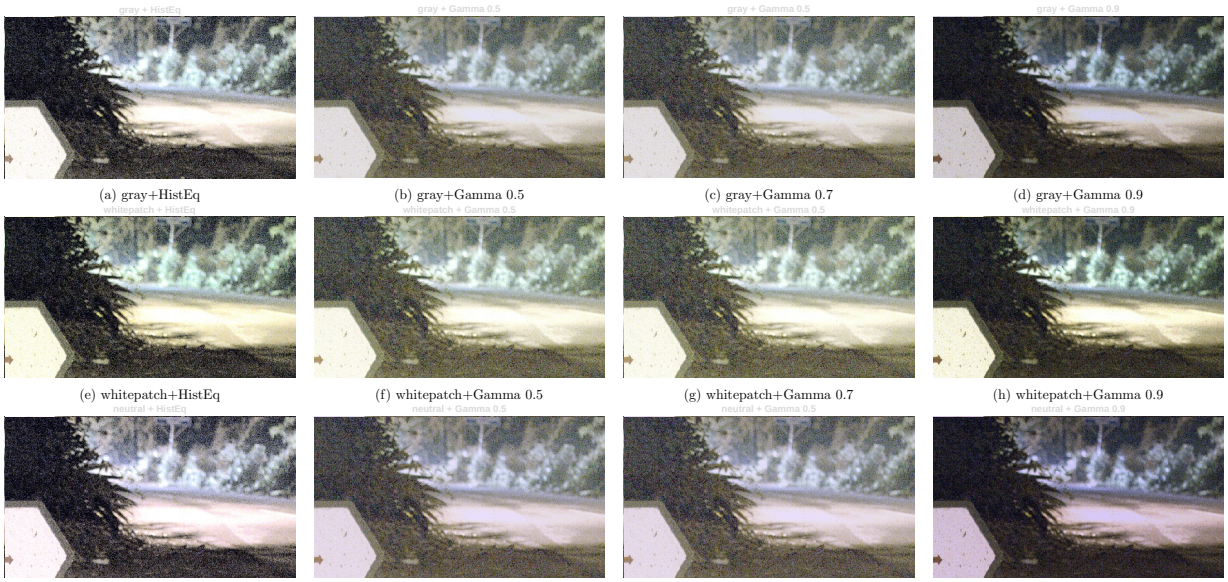


(a) gray+HistEq (b) gray+Gamma 0.5 (c) gray+Gamma 0.7 (d) gray+Gamma 0.9

(e) whitepatch+HistEq (f) whitepatch+Gamma 0.5 (g) whitepatch+Gamma 0.7 (h) whitepatch+Gamma 0.9

(i) neutral+HistEq (j) neutral+Gamma 0.5 (k) neutral+Gamma 0.7 (l) neutral+Gamma 0.9

Figure 10: `RawImage2`-tone mapping



(a) gray+HistEq (b) gray+Gamma 0.5 (c) gray+Gamma 0.7 (d) gray+Gamma 0.9

(e) whitepatch+HistEq (f) whitepatch+Gamma 0.5 (g) whitepatch+Gamma 0.7 (h) whitepatch+Gamma 0.9

(i) neutral+HistEq (j) neutral+Gamma 0.5 (k) neutral+Gamma 0.7 (l) neutral+Gamma 0.9

Figure 11: `RawImage3`-tone mapping