Proramming Assignment 1 : Camera Pipeline

EE17B031

February 2021

1 Introduction

In this assignment, we look at the basics of the camera pipeline. Starting from demosaicing to noise removal. The image results have been presented here as succinctly as possible. However, this comes at the cost of not being able to see those images at full resolution. The full resolution images for verification can be found **in this Google Drive folder.** The naming of the images is not the best and it is recommended to open the images and look at the titles to see what result do they correspond to.

1.1 Implementation

The code has been implemented and tested to work on MATLAB R2020b. The different sections of the assignment have a main code each (`main_demosaic`, `main_white_balance_tone_mapping`, and `main_denoising`) and the support functions are present in the same directory as the main codes. The data and the supplied functions are assumed to be present within the `data` and `data/bil_filter` folders. The `data` folder is assumed to be present in the same directory as the main codes.

1.2 Running the code

Results corresponding to each section of the assignment can be obtained by executing the corresponding main code. The figures are generated and displayed only and not saved. The code has also been broken down into sections using MATLAB's %% operator. Individual sections can be executed clicking Ctrl+Enter while the cursor is on that section.

2 Demosaicing

Demosaicing is a process of interpolating the red, green and blue channels from the raw pixel image. Here we will try to fill in the missing pixel values from the sampled raw images through interpolation. The raw image can be seen in Figure 1.

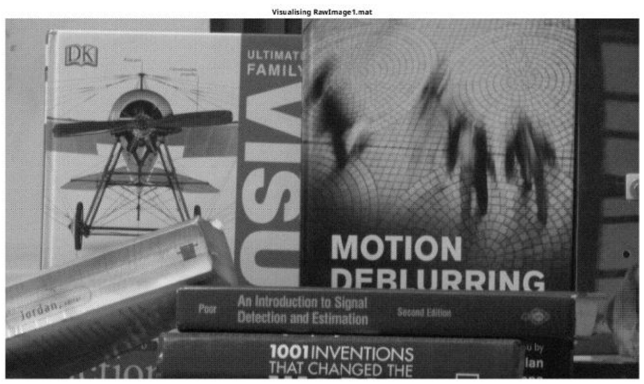


Figure 1: RawImage1.mat visualised

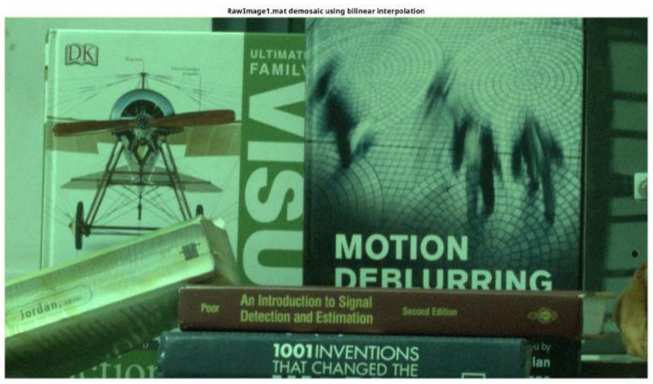


Figure 2: RawImage1 bilinear interpolation

The CFA for 'RawImage1.mat' is 'rggb' whose mask is provided in the file '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.

2.1 Bilinear Interpolation of RawImage1

In this section we perform bilinear interpolation to construct a full color image from RawImage1 using the provided CFA array 'b1.mat'. The resulting RGB image can be seen in Figure 2.

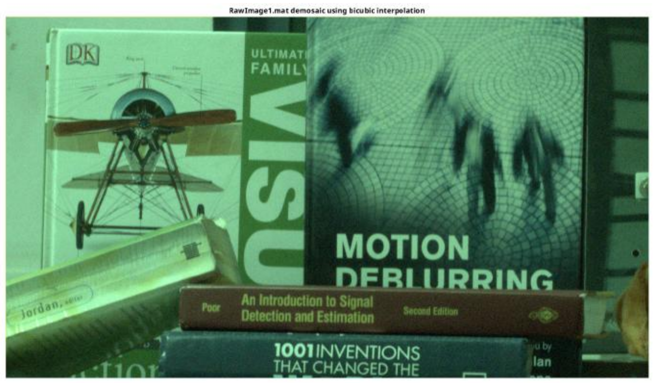


Figure 3: RawImage1 bicubic interpolation

2.2 Bicubic Interpolation of RawImage1

In this section we perform bicubic interpolation to construct a full color image from RawImage1 using the provided CFA array 'b1.mat'. The resulting RGB image can be seen in Figure 3.

2.2.1 Comparision with Bilinear Interpolation

The following observations can be made on seeing the results of Bicubic Interpolation and Bilinear Interpolation.

- The images are very similar in terms of how they look.
- On zooming in, there is less blurriness near the edges when using bicubic interpolation.
- Unless we are specifically looking for differences by zooming in, it is hard to tell the images apart.

2.3 Using MATLAB in-built function `demosaic`

In this section we perform demosaicing to construct a full color image from RawImage1 using the MATLAB in-built function `demosaic`. We specify the patters to be 'rggb' to the function. The resulting RGB image can be seen in Figure 4.

2.3.1 Comparision with the interpolation methods

The following results can be made on comparing the three methods mentioned.

- Again, the images are very similar in terms on how they look and hard to tell apart unless we are zooming in and looking for differences.

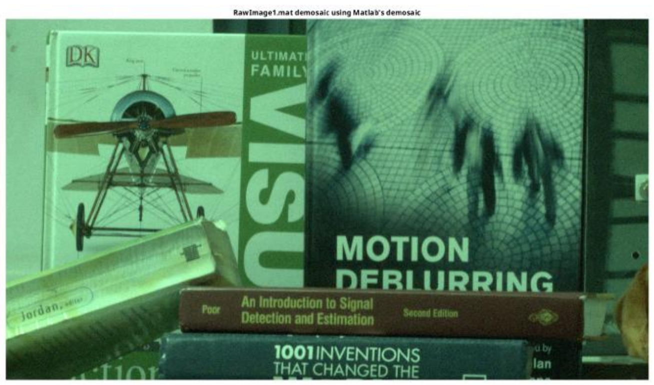


Figure 4: RawImage1 using MATLAB's in-built demosaic

- On zooming in, however, we can see that the `demosaic` function manages to retain edge information better and is hence, sharper than the other two images.
- The lack of perceptible difference be attributed due to the fact that RawImage1 has very less high frequency content and hence, even the basic bilinear interpolation does a really good job.
- Another observation, the result of which has not been presented is that images demosaiced using the `demosaic` function, after tone mapping using the neutral object given, have a very noticeable purple hue to them. Those demosaiced using bicubic interpolation have a much more neutral/whitish color.

2.4 Caveats of demosaicing

We need to be careful of the following facts during demosaicing using interpolation.

- It assumes there are no very high frequency content in the images.
- If there is the presence of very high frequency content, interpolation can be erroneous.
- Color fringing and Moire patterns are common errors when the assumption about the lack of high frequency content does not hold.
- This issue can be tackled by converting the generated RGB image to YCbCr and lowpass filtering the chrominance channels.

2.5 Demosaicing the "kodim19.mat" image

In this section we demosaic the "kodim19.mat" image using the three methods mentioned above and compare them to the provided "kodim19.png" image. For Bilinear and Bicubic Interpolation, we use the provided CFA pattern in "kodim_cfa.mat". For using `demosaic`, we use the pattern as 'rggb'. The results can be seen in Figure 5

2.5.1 Observations made on demosaicing

The following observations can be made from the results in Figure 5.

- The true color image shows no fringing or Moire patterns. All of the other three images show these near the high frequency regions like the fence and the wall of the room closer to the camera.
- There is a considerable loss in structure and edge sharpness, especially in the bilinear and bicubic interpolated images. This becomes apparent on zooming into the lighthouse and looking at the texture on it. `demosaic` does a considerably better job than the other two methods but the true color image is still better.

3 White Balance and Tone Mapping

In this section we do white balance and tone mapping for RawImage1, RawImage2, and RawImage3. A brief description of the methods is provided. The results are present in Figures 6-17. **Bicubic interpolation is used to demosaic the images.** There are essentially three ways to do white balancing.

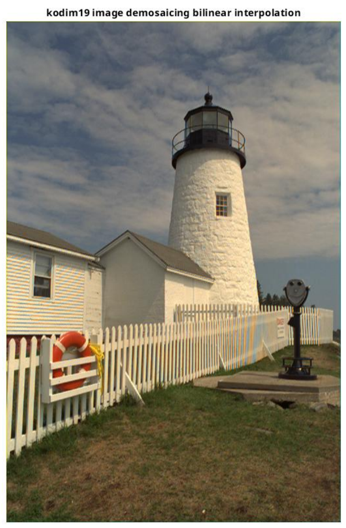
- Assume the average color of the scene to be gray and then do white balancing
- Assume that the brightest pixel is a specular highlight and hence should therefore be white
- Assume that some part of the scene to be neutral

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

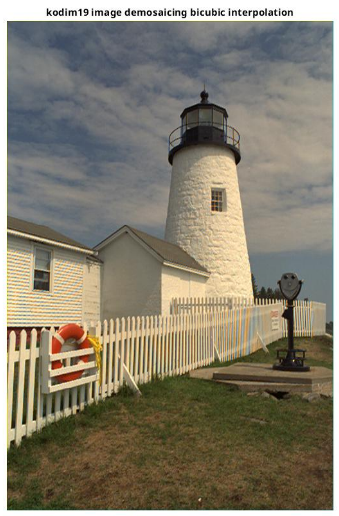
- Histogram equalization
- Gamma correction for $\gamma \in \{0.5, 0.7, 0.9\}$

3.1 Assuming average color to be gray

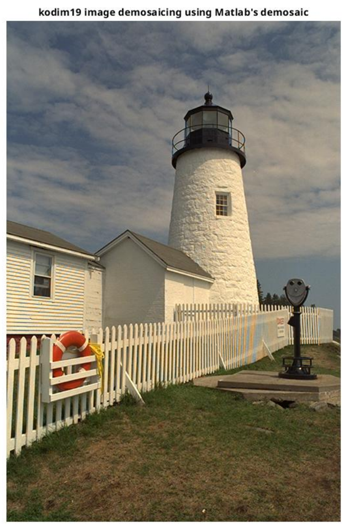
In this the underlying assumption is that the average color of a scene is gray, so force the average color of our image to be gray. We essentially calculate the average value of each channel, and then estimate the transforms which makes the average values equal. The respective transforms are applied to each of the pixels in the color channels.



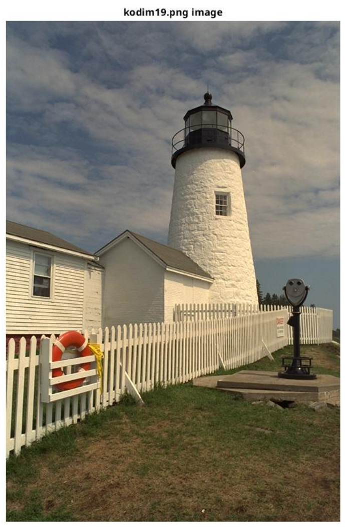
(a) Bilinear Interpolation



(b) Bicubic Interpolation



(c) MATLAB's `demosaic`



(d) True Color Image

Figure 5: "kodim19.mat" demosaicing results along with the true color image.

3.2 Specular Highlight

The underlying assumption is that the brightest pixel after demosaicing is a specular highlight, and hence, must be white. We are given the coordinates for the specular highlight in this problem. In real life situations, however, it has to be estimated. This approach fails in two major scenarios.

- If the brightest pixel has saturated, we will not see any change after tone mapping using this method.
- Metals have highlights which are not necessarily white. Say for example gold, which has golden looking highlights. Assuming that highlight to be white, calculating the transforms and applying those to the rest of the image might make it worse off than before.

3.3 Neutral Object

In this method, we detect an object which we believe has neutral color and estimate the transforms using it. We were given the coordinates for the neutral object in this problem but in real life problems, it has to be estimated.

3.4 Histogram Equalization

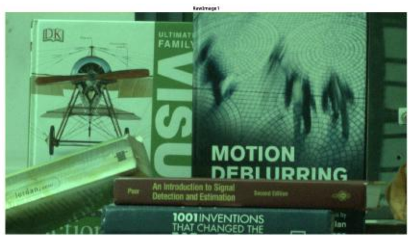
Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. The processing in done on the intensity channel. We hence convert the images from RGB to HSV first, perform equalization on the V channel, and convert the image back to RGB.

3.5 Gamma Correction

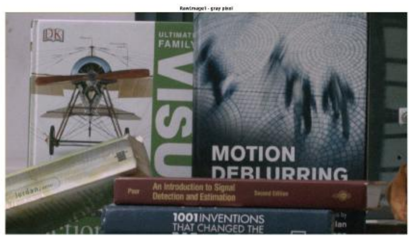
Gamma correction is another method for contrast adjustment. The equation is given by $Y_i = X_i^\gamma$ where Y is the output image, X is the input image, and γ is the Gamma value used. $\gamma > 1$ leads to darkening and $\gamma < 1$ leads to brightening. It must be noted that gamma correction too works the best when applied on the intensity channel and when the intensity values are normalized to lie between 0 and 1. Hence, likewise to what we did for histogram equalization, we convert the images from RGB to HSV first, perform gamma correction on the V channel, and convert the image back to RGB.

4 Image Denoising

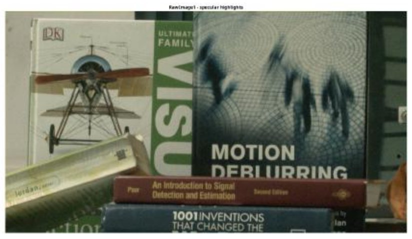
In this part we use bilateral filtering for denoising. The bilateral filter is controlled by two parameters, σ_s and σ_r which control the spatial weighing and alignment of image intensities respectively. Ideally we would like to locally estimate the noise and adjust both the parameters accordingly but given the cost here we would go with single set of parameters. To estimate the noise we will choose a constant region in the image and calculate the standard deviation



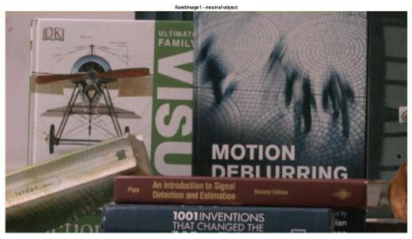
(a) Original



(b) Gray World

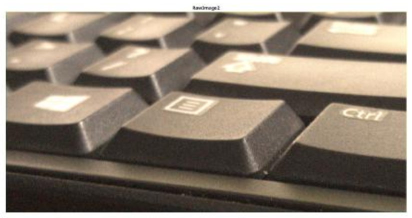


(c) Specular Highlights

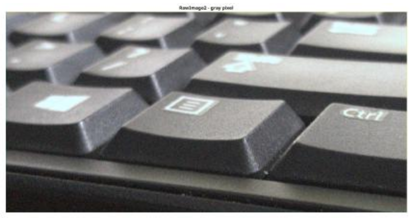


(d) Neutral Object

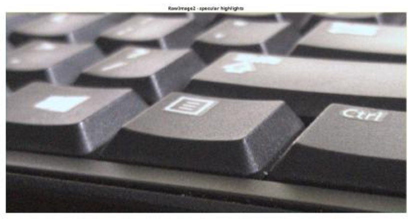
Figure 6: White Balancing algorithms on RawImage1



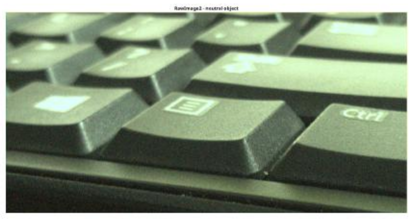
(a) Original



(b) Gray World

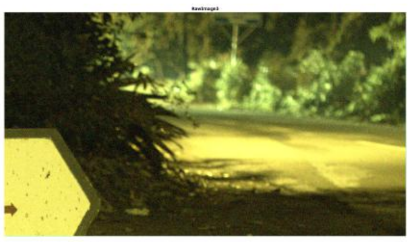


(c) Specular Highlights

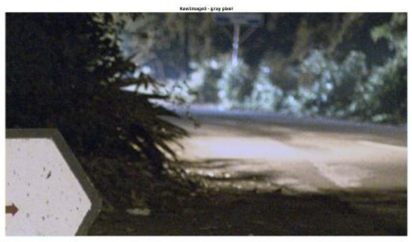


(d) Neutral Object

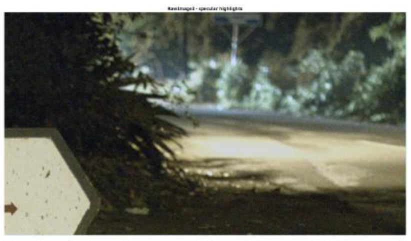
Figure 7: White Balancing algorithms on RawImage2



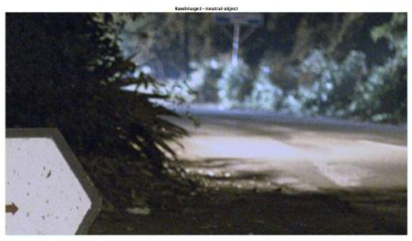
(a) Original



(b) Gray World



(c) Specular Highlights



(d) Neutral Object

Figure 8: White Balancing algorithms on RawImage3

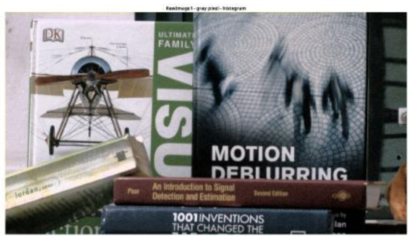
along all the three channels R, G and B in that region. The parameters for the filter are : window size (w) = 11, $\sigma_s = 2.5$ and $\sigma_r = 1.95 \times \sigma_n$ where σ_n is the noise standard deviation in the corresponding colour channel. σ_n is estimated by choosing a flat region in the image. Apart from being flat, we must ensure that we do not choose a region where the pixels have saturated, else we will not get an accurate estimation. The regions chosen are as follows.

- RawImage1 : (75, 690) to (135, 750)
- RawImage2 : (705, 924) to (765, 984)
- RawImage3 : (1060, 750) to (1120, 810)

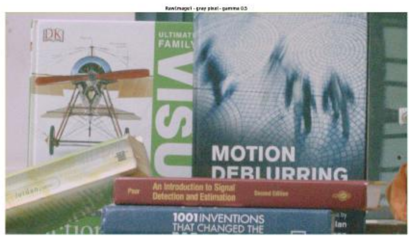
The σ_r values (in the order for R, G, and B channels respectively) estimated for the three images are mentioned below.

- RawImage1 : 0.0669, 0.0413, 0.0545
- RawImage2 : 0.1493, 0.1038, 0.1306
- RawImage3 : 0.1312, 0.1426, 0.2426

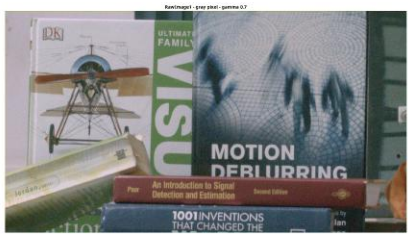
To show the effect of denoising, I chose the image which looked the best to me the best after demosaicing, white balancing, and tone mapping from each of RawImage1, RawImage2, and RawImage3. It has to be kept in mind that



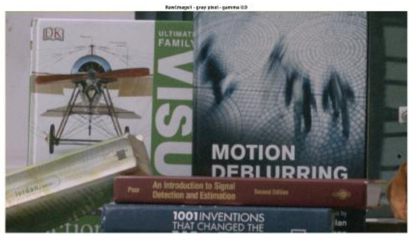
(a) Histogram



(b) $\gamma = 0.5$

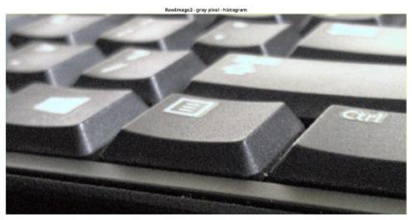


(c) $\gamma = 0.7$

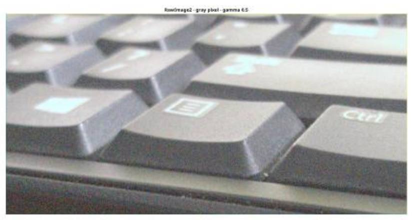


(d) $\gamma = 0.9$

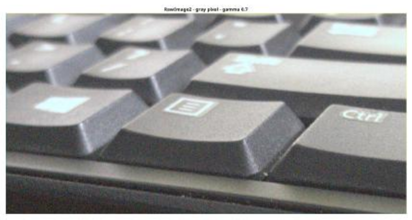
Figure 9: Tone mapping RawImage1 after White Balance using Gray World



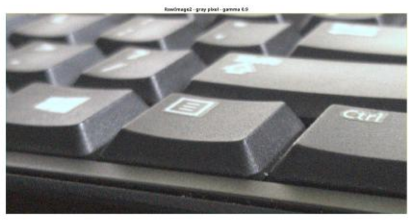
(a) Histogram



(b) $\gamma = 0.5$



(c) $\gamma = 0.7$

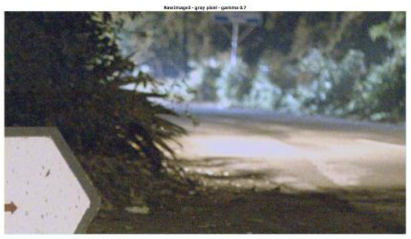


(d) $\gamma = 0.9$

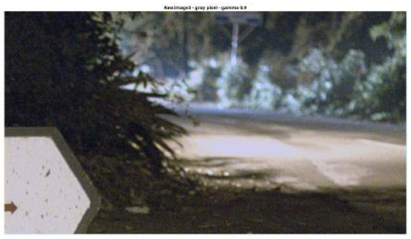
Figure 10: Tone mapping RawImage2 after White Balance using Gray World

(a) Histogram

(b) $\gamma = 0.5$



(c) $\gamma = 0.7$



(d) $\gamma = 0.9$

Figure 11: Tone mapping RawImage3 after White Balance using Gray World

this is a complete subjective choice and what image looks good depends on an individual's perception. The images tone mapped using histogram equalization have a more *contrasty* look whereas those gamma equalized with $\gamma = 0.9$ have slightly lower contrast but more uniform lighting. I prefer the latter look but choosing the former too is an equally valid option. The white balance and tone mapping combinations chosen for each image are mentioned below.

- RawImage1 : Neutral Object, Gamma Correction with $\gamma = 0.9$
- RawImage2 : Specular Highlight, Gamma Correction with $\gamma = 0.9$
- RawImage3 : Neutral Object, Gamma Correction with $\gamma = 0.9$

The resulting images are shown in 18

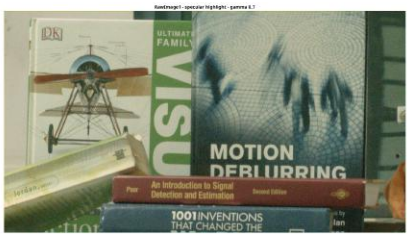
4.1 Kernel size for Gaussian kernel with given σ

A Gaussian has infinite spread. For a discrete approximation, we have to truncate it. This has to be done carefully, however. For a given standard deviation σ, the kernel size chosen is the **smallest odd integer greater than or equal to 6σ.** This can be justified as follows.

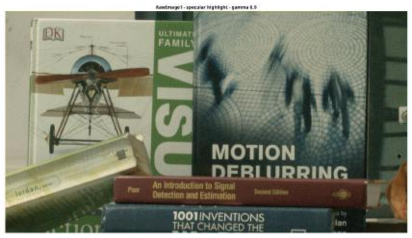
- We choose 6σ as we know more than 99% of the area of a Gaussian lies in a 3σ radius around its mean. Choosing 6σ means we capture the structure of the Gaussian in an ample way.

(a) Histogram

(b) $\gamma = 0.5$



(c) $\gamma = 0.7$



(d) $\gamma = 0.9$

Figure 12: Tone mapping RawImage1 after White Balance using Specular Highlights

- We choose an odd integer as then we have a distinct central pixel for the kernel and we can "center" that kernel on a pixel in our input image.

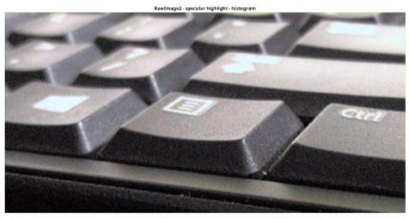
4.2 Choosing σ_r from σ_n

σ_r is a soft thresholding of what we consider a genuine edge value except the difference due to noise. σ_r should be in general greater than σ_n. However, we cannot make it too large either.

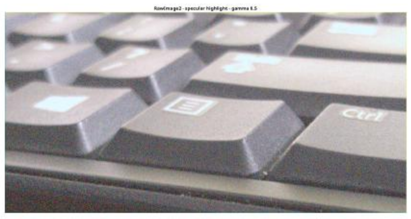
- If $\sigma_r \leq \sigma_n$, weights given to pixels with intensities even slightly different from the intensity of the current pixel will be very low. This slight difference can occur even due to noise. Especially in flat regions, small difference in intensity values will occur only due to noise and it is important that we give the neighborhood pixels a good enough weight for denoising to take place. Choosing $\sigma_r \leq \sigma_n$ we can expect to see little to no denoising effect.

- If $\sigma_r >> \sigma_n$, there will be considerable weights given even to pixels with very different intensity values from the current pixel. This will lead to a lot of averaging resulting in an image with all its edges blurred.

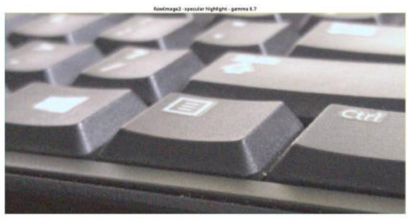
Both of the aforementioned points were verified by choosing $\sigma_r = 0.95\sigma_n$ and $\sigma_r = 95\sigma_n$. A good choice hence is $\sigma_r \approx 2\sigma_n$



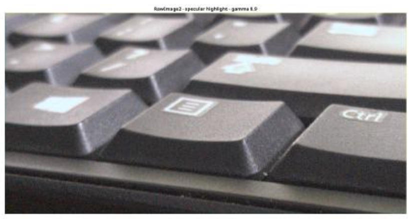
(a) Histogram



(b) $\gamma = 0.5$

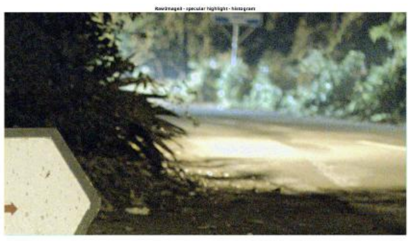


(c) $\gamma = 0.7$

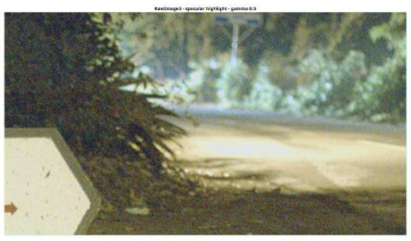


(d) $\gamma = 0.9$

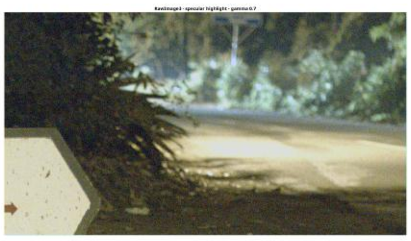
Figure 13: Tone mapping RawImage2 after White Balance using Specular Highlights



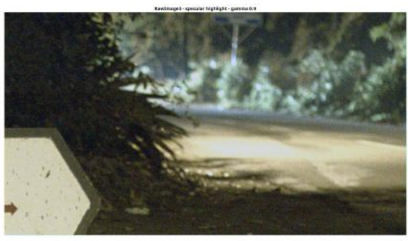
(a) Histogram



(b) $\gamma = 0.5$



(c) $\gamma = 0.7$

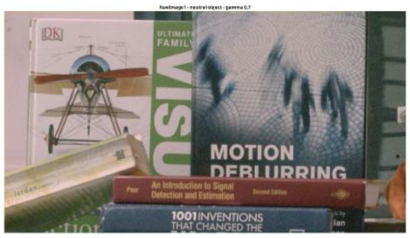


(d) $\gamma = 0.9$

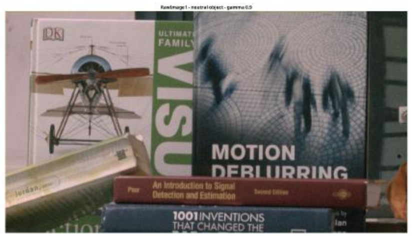
Figure 14: Tone mapping RawImage3 after White Balance using Specular Highlight

(a) Histogram

(b) $\gamma = 0.5$

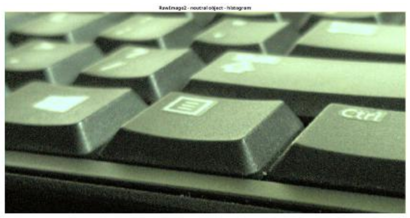


(c) $\gamma = 0.7$



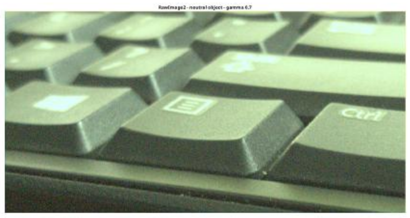
(d) $\gamma = 0.9$

Figure 15: Tone mapping RawImage1 after White Balance using Neutral Object

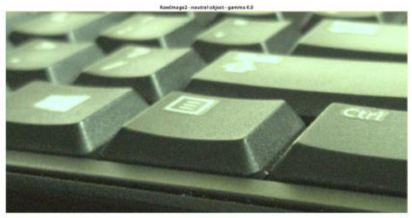


(a) Histogram

(b) $\gamma = 0.5$

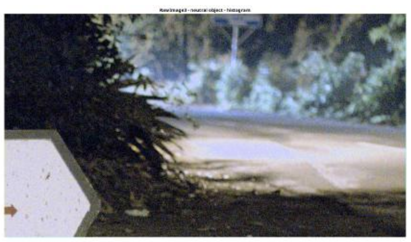


(c) $\gamma = 0.7$

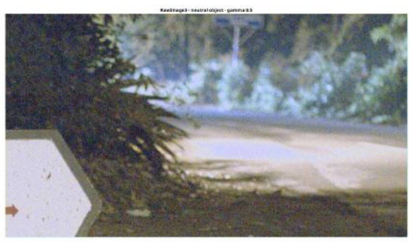


(d) $\gamma = 0.9$

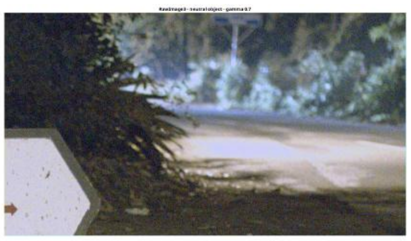
Figure 16: Tone mapping RawImage2 after White Balance using Neutral Object



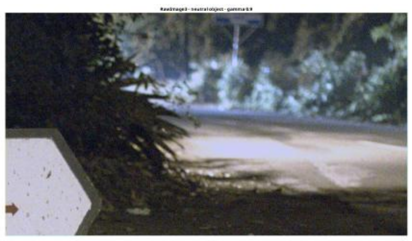
(a) Histogram



(b) $\gamma = 0.5$



(c) $\gamma = 0.7$



(d) $\gamma = 0.9$

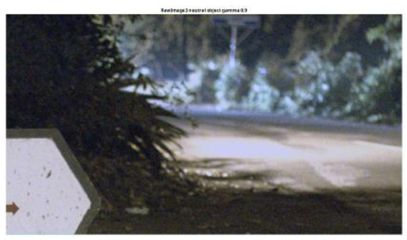
Figure 17: Tone mapping RawImage3 after White Balance using Neutral Object

(a) RawImage1 before denoising

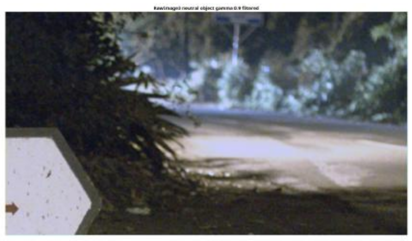
(b) RawImage1 after denoising

(c) RawImage2 before denoising

(d) RawImage2 after denoising



(e) RawImage3 before denoising



(f) RawImage3 after denoising

Figure 18: Images before and after denoising