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

Ritabrata Mandal
EE25S009

September 5, 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.

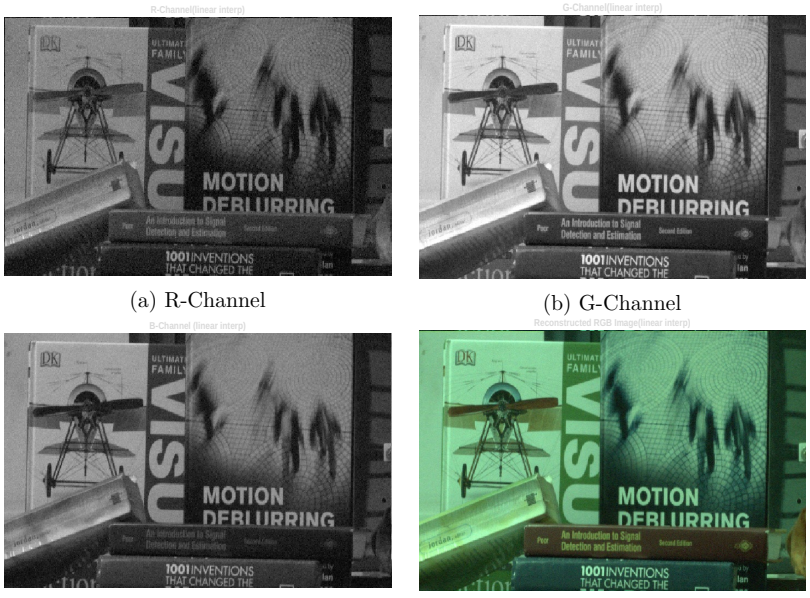
   (a) **Implementation**:
   The implementation has been carried out and tested in MATLAB R2025a. A `main.m` script invokes four functions: `demos.m`, `white_balance.m`, `tone_map.m`, and `denoising.m`. The `demos.m` function performs image demosaicing for two given raw images, `RawImage1` and `kodim19`. The functions `white_balance.m` and `tone_map.m` apply white balancing and tone mapping techniques, respectively, on images such as `RawImage1`, `RawImage2` and `RawImage3`. Finally `denoising.m` apply bilateral filtering for denosing the same three images `RawImage1`, `RawImage2` and `RawImage3`

   (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 fill 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 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.

(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



(a) R-Channel (b) G-Channel

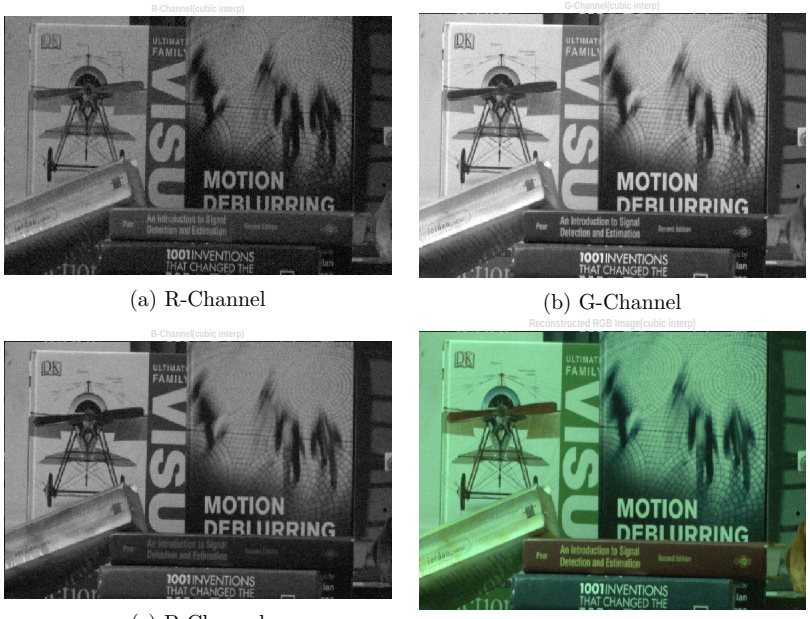
(c) B-Channel (d) Reconstructed RGB

Figure 1: Reconstructed `RawImage1` 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 ap-

pear 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 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.

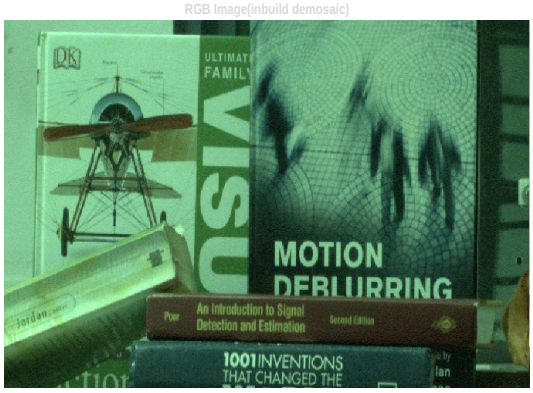


Figure 3: Reconstructed `RawImage1` with `demosaic` function

(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:

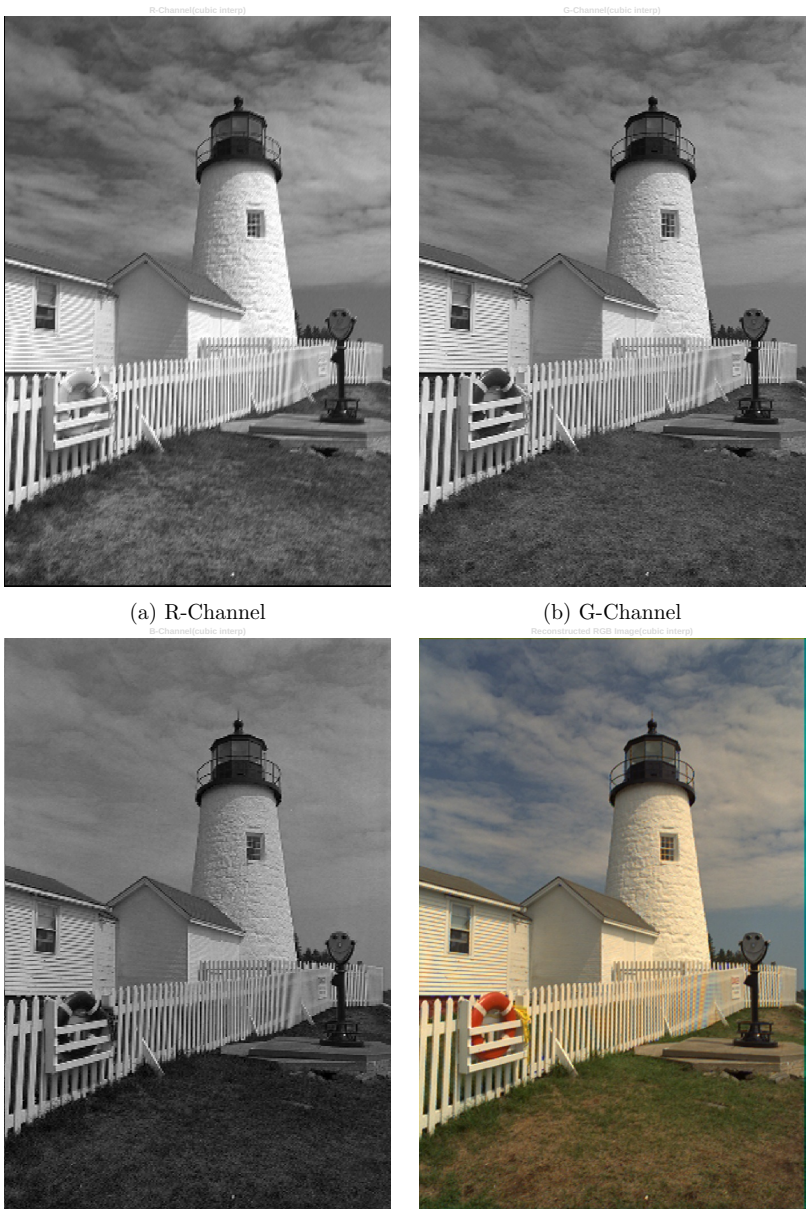
i. **Loss of Detail:** If the image has high-frequency textures, interpolation may blur fine details.

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.



(a) R-Channel (b) G-Channel

(c) B-Channel (d) Reconstructed RGB

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

(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. We get Figure 5b

(a) `kodim19` original (b) Median after reconstruction

Figure 5: `kodim19.mat` original and median filter with cubic interpolation.

(g) In the Figure 4d (i.e. cubic interpolated rgb) we can see moire pattern where as in Figure 5b(i.e. after applying median filter in chrominance channel) there is less moire pattern present in the image. Figure 5a shows the original `kodim19.png`

3. **White Balancing and Tone Mapping**
In this section, we perform white balancing and tone mapping on `RawImage1`, `RawImage2`, and `RawImage3`. The built-in `demosaic` function is used to reconstruct full-color images from the raw sensor data. White balancing is carried out using three different approaches:

   - Assuming the average color of the scene is gray (gray-world assumption).
   - Assuming the brightest pixel represents a specular highlight and should therefore be white.
   - Assuming that a specific region of the scene is neutral.

   Once the images are white-balanced, tone mapping is applied using two methods:

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

The white balancing results are summarized as follows:

(a) **Gray-world assumption:** The mean value of each channel is computed, and scaling factors are applied to shift the average color toward gray. Figure 6b shows the result for `RawImage1`, while Figure 7b and Figure 8b present the corresponding results for `RawImage2` and `RawImage3`.

(b) **Specular highlight assumption:** The brightest pixel is assumed to be white. Using the specified pixel coordinates $(830, 814)$ for `RawImage1`, $(1165, 280)$ for `RawImage2`, and $(175, 675)$ for `RawImage3`, the results are shown in Figure 6c, Figure 7c, and Figure 8c.

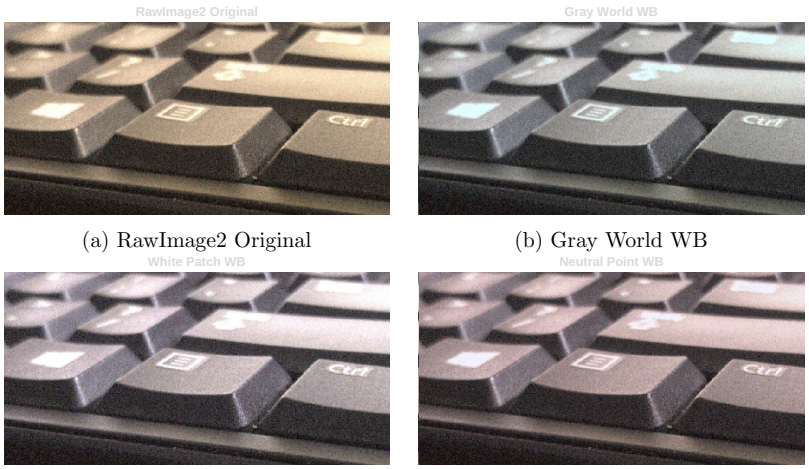
(c) **Neutral region assumption:** A selected region is assumed neutral. The chosen pixel coordinates are $(2000, 435)$ for `RawImage1`, $(445, 715)$ for `RawImage2`, and $(1550, 565)$ for `RawImage3`. The results are presented in Figure 6d, Figure 7d, and Figure 8d.

(d) **Tone mapping:** After white balancing, tone mapping is performed on all three images. The results are shown in Figure 9, Figure 10, and Figure 11.

(a) RawImage1 Original (b) Gray World WB

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

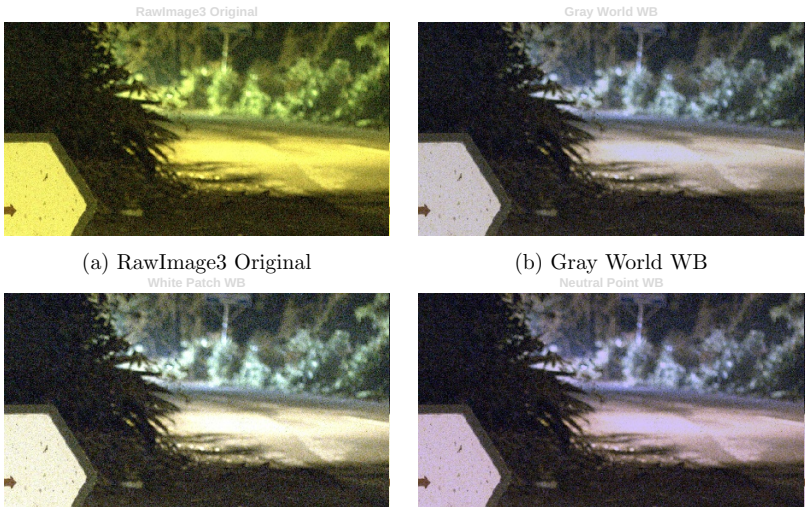
Figure 6: `RawImage1`-White Balance



(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

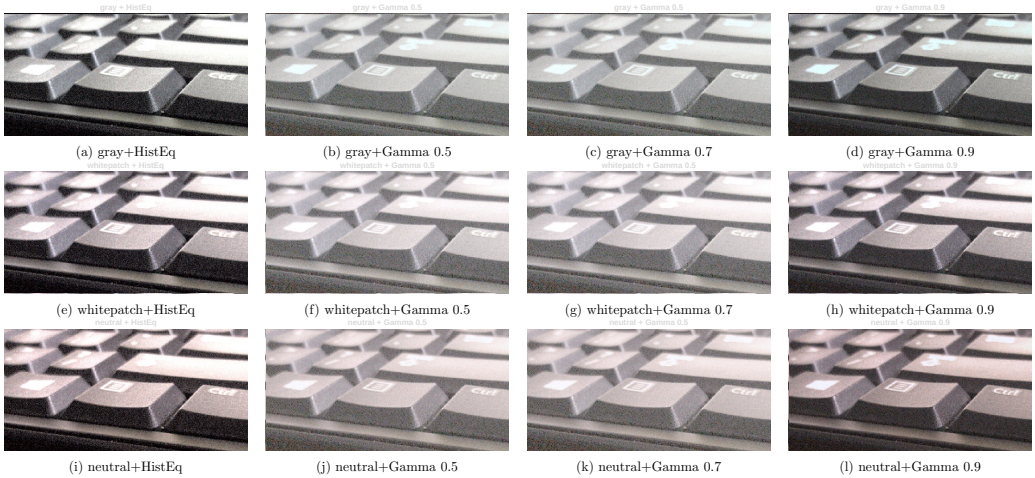
Figure 8: `RawImage3`-White Balance

(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

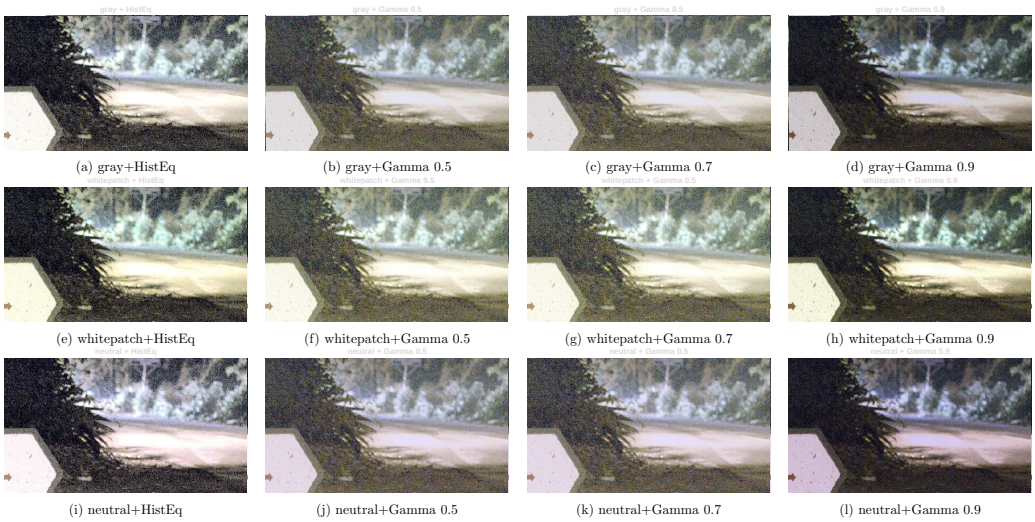


(a) gray+HistEq (b) gray+Gamma 0.5 (c) gray+Gamma 0.7 (d) gray+Gamma 0.9

(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

4. **Image Denoising**

   In this section, we employ bilateral filtering for image denoising. The bilateral filter is governed by two parameters, σ_s and σ_r, which control the spatial weighting and the intensity alignment, respectively. Ideally, one would locally estimate the noise level and adapt these parameters accordingly; however, due to computational cost, we use a fixed set of parameters.

   To estimate the noise, a constant region is selected from each image, and the standard deviation is computed across the three channels (R, G, and B) within that region. The chosen filter parameters are:

$$W = 11, \quad \sigma_s = 2.5, \quad \sigma_r = 1.95 \times \sigma_n,$$

   where σ_n denotes the noise standard deviation for the corresponding color channel.

   The selected regions are:

   (a) `RawImage1`: $(100, 100)$ to $(160, 160)$

   (b) `RawImage2`: $(705, 924)$ to $(765, 984)$

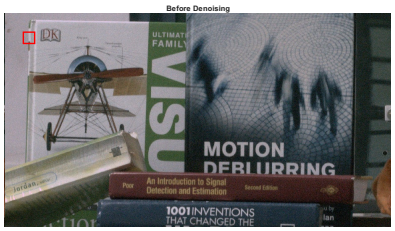
   (c) `RawImage3`: $(50, 400)$ to $(110, 460)$

   These regions are shown with red box in Figures 12a, 12c, and 12e, corresponding to `RawImage1`, `RawImage2`, and `RawImage3`, respectively.

   The estimated values of σ_r for the R, G, and B channels are as follows:

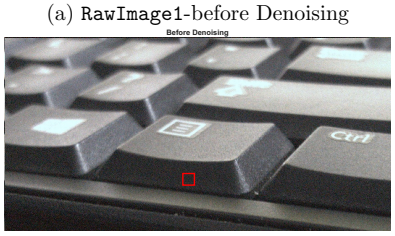
- `RawImage1`: 0.068, 0.0614, 0.0774
- `RawImage2`: 0.0682, 0.0510, 0.0729
- `RawImage3`: 0.0694, 0.0694, 0.1202

The denoised results for `RawImage1`, `RawImage2`, and `RawImage3` are shown in Figures 12b, 12d, and 12f, respectively.

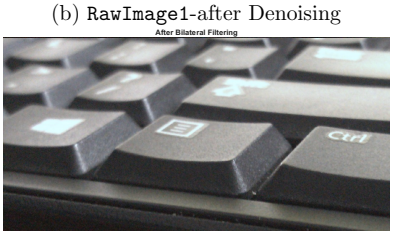


(a) `RawImage1`-before Denoising

(b) `RawImage1`-after Denoising



(c) `RawImage2`-before Denoising



(d) `RawImage2`-after Denoising

(e) `RawImage3`-before Denoising

(f) `RawImage3`-after Denoising

Figure 12: Denoising

5. **Questions:**

1. Since a Gaussian distribution has infinite support, we must truncate it when constructing a discrete approximation. This truncation must be done carefully. For a given standard deviation σ, the kernel size is selected as the smallest odd integer greater than or equal to 6σ. This choice is justified because approximately 99% of the Gaussian's total mass lies within a radius of 3σ around the mean. Thus, by taking 6σ as the kernel

size, we effectively capture the essential structure of the Gaussian. Moreover, choosing an odd-sized kernel ensures the presence of a well-defined central pixel, allowing the kernel to be symmetrically aligned with pixels in the input image.

2. σ_r acts as a soft threshold to distinguish genuine edge differences from variations caused by noise. In practice, σ_r should be chosen larger than σ_n, but not excessively so, since both extremes lead to undesirable effects.

   - **If $\boldsymbol{\sigma_r \leq \sigma_n}$:** Even pixels with slightly different intensities from the reference pixel receive very low weights. Such small variations often arise purely from noise, particularly in flat regions. As a result, neighboring pixels do not contribute effectively to denoising, and little to no noise reduction is achieved.
   - **If $\boldsymbol{\sigma_r \gg \sigma_n}$:** Pixels with large intensity differences still receive considerable weights, causing excessive averaging. This oversmoothing effect blurs edges and reduces important structural details in the image.

   These effects were verified experimentally using $\sigma_r = 0.95\sigma_n$ and $\sigma_r = 95\sigma_n$. Based on these observations, a good practical choice is $\sigma_r \approx 2\sigma_n$.