Wooyoung Kim
323-834-4126
wkim9450@usc.edu

EE 569 Homework #1
January 23, 2026

Problem 1: Image Demosaicing and Histogram Manipulation

I. Abstract and Motivation

Digital image processing often begins with the raw data captured by camera sensors. This report explores three fundamental techniques for processing and enhancing such images: (a) Bilinear Demosaicing, (b) Global Histogram Equalization, and (c) Contrast Limited Adaptive Histogram Equalization (CLAHE).

The primary motivation for demosaicing is to reconstruct a full-color image from the incomplete color samples captured by a Bayer pattern sensor. Following this, histogram manipulation techniques are essential for correcting exposure and contrast issues. I examine broadly applied global methods versus exact specification methods to understand the trade-offs between mathematical rigorousness and visual quality. Finally, I apply these concepts to a real-world problem: removing haze from landscape photography using both global and adaptive methods (CLAHE) to restore visibility and detail in degraded images.

II. Approach and Procedures

II..1 Bilinear Demosaicing

Most digital cameras use a Color Filter Array (CFA) known as the Bayer pattern. In this assignment, I processed the `sailboats_cfa.raw` image, which follows the GRBG pattern (Green-Red for even rows, Blue-Green for odd rows).

I implemented a bilinear interpolation algorithm to estimate the missing color values at each pixel $P(i, j)$. The missing values are calculated as the arithmetic mean of the nearest valid neighbors of that color. For example, at a Red pixel location (i, j), the missing Green value $\hat{G}_{i,j}$ and Blue value $\hat{B}_{i,j}$ are estimated as:

$$\hat{G}_{i,j} = \frac{G_{i-1,j} + G_{i+1,j} + G_{i,j-1} + G_{i,j+1}}{4} \tag{1}$$

$$\hat{B}_{i,j} = \frac{B_{i-1,j-1} + B_{i-1,j+1} + B_{i+1,j-1} + B_{i+1,j+1}}{4} \tag{2}$$

Boundary mirroring was used to handle edge pixels to prevent artifacts at the image borders.

II..2 Histogram Manipulation

I implemented two distinct methods to enhance the contrast of the grayscale image `airplane.raw`.

II..2.1 Method A: Transfer-Function-Based Method

This is the standard Histogram Equalization (HE) approach. I first compute the Probability Density Function (PDF) and the Cumulative Distribution Function (CDF) of the input image. The transformation function $T(r_k)$ maps an input intensity level r_k to an output level s_k:

$$s_k = T(r_k) = \text{round}\left((L-1) \cdot CDF(r_k)\right) \tag{3}$$

where $L = 256$ is the number of gray levels. This method stretches the histogram to span the full dynamic range but cannot split pixels belonging to the same input bin.

II..2.2 Method B: Bucket-Filling Method

This method (exact histogram specification) guarantees a perfectly uniform output histogram.

1. All pixels in the image are stored in a list and sorted by intensity.
2. Pixels with the same intensity are effectively ranked by their spatial position (or stable sort order).
3. I assign new pixel values based on the pixel's rank, dividing the total number of pixels N into L equal buckets.

The new value for a pixel with rank k (where $0 \leq k < N$) is:

$$val = \text{floor}\left(\frac{k \cdot L}{N}\right) \tag{4}$$

II..3 Haze Removal via CLAHE

To remove haze from `towers.raw`, I utilized the YUV color space to separate luminance (Y) from chrominance (U, V).

1. **Color Space Conversion:** The RGB image is converted to YUV.
2. **Processing:** I apply histogram equalization techniques only to the Y channel. I compared Global HE (Method A&B) against Contrast Limited Adaptive Histogram Equalization (CLAHE).
3. **CLAHE Implementation:** Unlike global methods, CLAHE divides the image into tiles. It computes the histogram for each tile, clips the histogram at a predefined limit to suppress noise amplification, and redistributes the clipped pixels. Bilinear interpolation is used to blend the tiles and remove boundary artifacts.
4. **Reconstruction:** The enhanced Y channel is combined with the original U and V channels and converted back to RGB.

III. Experimental Results

III..1 Bilinear Demosaicing

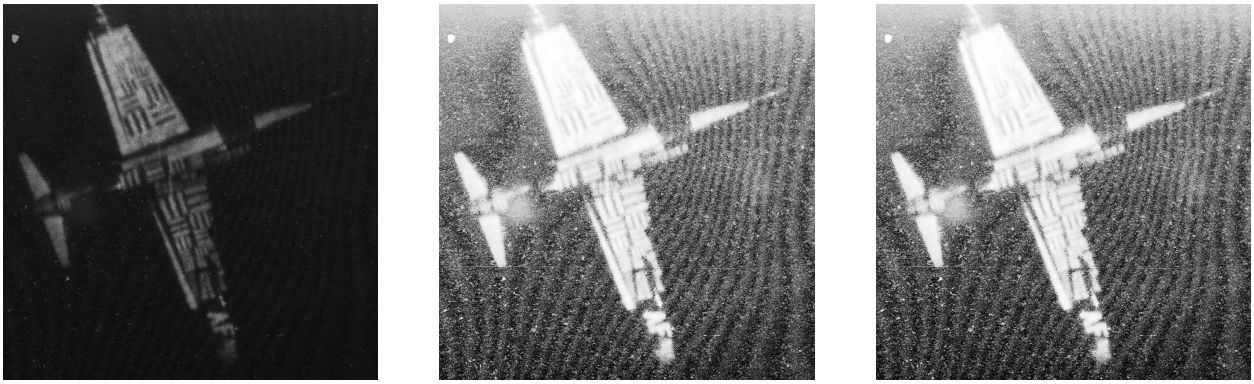
The `sailboats_cfa.raw` file was successfully demosaiced. Figure 8 shows the result.

Figure 1: Original Image of `sailboats_cfa.png`.

Figure 2: Result of Bilinear Demosaicing on `sailboats_cfa.raw`.

III..2 Histogram Manipulation

Figure 3 displays the original and enhanced images for the airplane dataset. Figure 4 and Figure 5 present the corresponding plots.



(a) Original Image (b) Method A (Standard HE) (c) Method B (Bucket Filling)

Figure 3: Visual comparison of histogram enhancement methods.

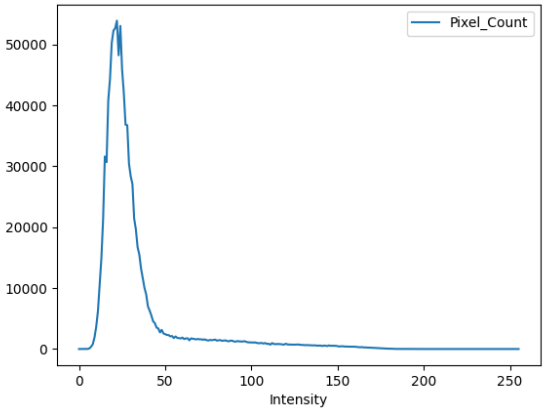
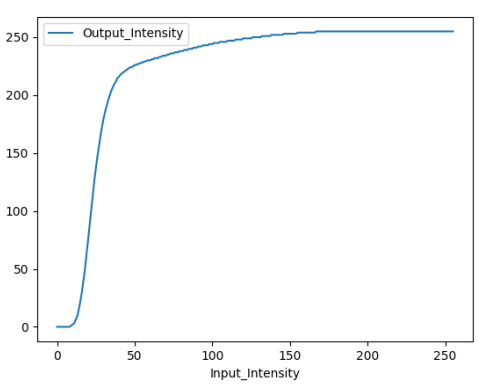
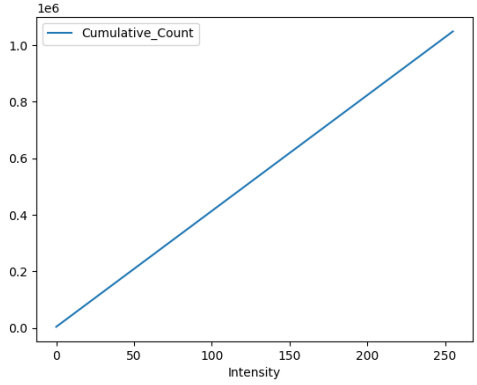


Figure 4: Original Image Pixel Count Histogram





(a) Method A (Transfer Function) (b) Method B (Cumulative Histogram)

Figure 5: Quantitative analysis of histogram manipulation.

III..3 Haze Removal (CLAHE)

The results of haze removal on `towers.raw` are shown below.

(a) Method A (b) Method B (c) CLAHE (Adaptive)

Figure 6: Comparison of De-hazing techniques.

IV. Discussion

IV..1 Demosaicing Artifacts

As I can see Figure 7, the most obvious difference lies in the sharpness of the edges. In my implementation, I observed distinct zipper artifacts along the high-contrast boundaries, such as the edges of the sails and the rigging lines against the sky.

The primary cause of these artifacts is that bilinear interpolation averages neighboring pixel values without considering the presence of edges. For example, when interpolating a green value on a red pixel located on a sharp boundary, the algorithm averages the green neighbors from both the bright sail and the dark sky. This results in an intermediate color that does not physically exist, creating a zipper-like on-off pattern along the edge.

To improve performance, the algorithm should be edge-aware. Instead of simply averaging all four neighbors, I could first calculate the horizontal and vertical gradients. If the vertical gradient is large, I should only interpolate using the horizontal neighbors to avoid crossing the edge boundary. Furthermore, interpolating the *color difference* or *color ratios* rather than the raw intensity values would help maintain consistent hue, as hue typically changes much more smoothly than luminance in natural images.

(a) Bilinear Demosaicing Method (b) Advanced Demosicing Method

Figure 7: Comparison of Demosaicing techniques.

IV..2 Histogram Manipulation: Method A vs. Method B

Visual Comparison: Subjectively, the results from Method A and Method B appear quite similar at first glance. Both successfully enhance the global contrast of the airplane, making details more visible compared to the washed-out original. However, upon closer inspection of smooth regions (such as the sky), Method B introduces subtle graininess. This is because Method B forces pixels with identical original values into different bins to satisfy the uniform distribution requirement, effectively adding quantization noise to flat gradients.

Computational Efficiency: From an computational perspective, **Method A** is significantly more efficient. Method A relies on histogram binning, which is an $O(N)$ linear operation. In contrast, Method B requires sorting all pixels in the image to determine their rank, which is an $O(N \log N)$ operation. For high-resolution images, Method B would be computationally prohibitive without offering a commensurate improvement in visual quality.

Conclusion: I consider **Method A to be better**. It achieves the goal of contrast enhancement with much lower computational cost and respects the natural grouping of pixel values, avoiding the artificial noise introduced by the exact bucket filling of Method B.

IV..3 CLAHE vs. HE (Haze Removal)

As you can see in Figure 6, comparing the global histogram equalization (Method A/B) with CLAHE on the *towers.raw* image highlighted the limitations of global processing.

When I applied Method A and B, the haze was indeed removed, but the rest of the image suffered. The large amount of bright haze in the original image dominated the global histogram. To stretch this data, the algorithm heavily compressed the darker regions. As a result, the trees and the shadows of the buildings became crushed into black blobs, losing almost all texture and detail.

CLAHE produced a significantly more balanced result. Because it operates on small local tiles, the bright sky did not dictate the contrast mapping for the dark trees. The algorithm was able to enhance the local contrast of the vegetation independently of the sky. Additionally, the CLAHE prevented the noise in the flat sky regions from being amplified, which is a common issue with standard adaptive histogram equalization.

CLAHE is the superior method for this scene. It successfully removed the haze while simultaneously preserving the structural details in the dark foreground, resulting in a much more pleasing and realistic image compared to the global methods.

Problem 2: Image Denoising

I. Abstract and Motivation

Image denoising is a fundamental challenge in digital image processing, aimed at recovering a clean signal from a noisy observation while preserving essential structural details such as edges and textures. In this work, I explore and compare the performance of various denoising algorithms on the `flower_gray` and `flower` color images. I implement linear filters (Uniform and Gaussian), Bilateral filtering, and Non-Local Means filtering methods. Additionally, I address the challenge of mixed noise in color images by designing a cascaded filtering approach. The performance of each method is quantitatively evaluated using the Peak Signal-to-Noise Ratio (PSNR) and qualitatively assessed through visual inspection.

II. Approach and Procedures

II..1 Evaluation Metric: PSNR

To objectively measure the quality of the denoised images, I utilize the Peak Signal-to-Noise Ratio (PSNR). Given an original noise-free image X and a filtered image Y of size $N \times M$, the Mean Squared Error (MSE) is defined as:

$$\text{MSE} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (Y(i,j) - X(i,j))^2 \tag{5}$$

The PSNR is subsequently calculated as:

$$\text{PSNR (dB)} = 10 \log_{10} \left(\frac{\text{Max}^2}{\text{MSE}} \right) \tag{6}$$

where Max = 255 for 8-bit images.

II..2 Linear Filters

I implemented two basic linear filters:

1. **Uniform Filter:** Replaces each pixel with the average of its $K \times K$ neighborhood. While effective at reducing variance, it indiscriminately blurs edges.

2. **Gaussian Filter:** Uses a Gaussian-weighted kernel where weights decay with distance from the center. This generally preserves structure better than a uniform box filter.

II..3 Bilateral Filtering

To address the edge-blurring artifacts of linear filters, I implemented the Bilateral Filter. This non-linear filter weights neighbors based on both spatial distance and intensity difference:

$$w(i, j, k, l) = \exp \left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_c^2} - \frac{\|I(i,j) - I(k,l)\|^2}{2\sigma_s^2} \right) \tag{7}$$

where σ_c controls the spatial extent and σ_s controls the intensity sensitivity (range).

II..4 Non-Local Means Filtering

Unlike local filters that rely on spatial proximity, the Non-Local Means (NLM) algorithm exploits the self-similarity of images. It estimates a denoised pixel $Y(i, j)$ by computing a weighted average of pixels $I(k, l)$ within a large search window $\aleph$, where the weights depend on the structural similarity between the local neighborhood patches $N_{i,j}$ and $N_{k,l}$.

The discrete NLM filter is defined as:

$$Y(i, j) = \frac{\sum_{(k,l) \in \aleph} I(k, l) w(i, j, k, l)}{\sum_{(k,l) \in \aleph} w(i, j, k, l)} \tag{8}$$

The similarity weight $w(i, j, k, l)$ is computed using a Gaussian decay function:

$$w(i, j, k, l) = \exp \left(-\frac{\|I(N_{i,j}) - I(N_{k,l})\|_{2,a}^2}{h^2} \right) \tag{9}$$

where the Gaussian-weighted Euclidean distance between two patches is given by:

$$\|I(N_{i,j}) - I(N_{k,l})\|_{2,a}^2 = \sum_{n_1, n_2 \in N'} G_a(n_1, n_2) \left(I(i - n_1, j - n_2) - I(k - n_1, l - n_2) \right)^2 \tag{10}$$

Key Parameters:

- **א (Search Window Size):** The large neighborhood around the pixel searched for similar patches.
- **N' (Patch Window Size):** The small local neighborhood used to compare the structural similarity of two pixels.
- ***h* (Filtering Parameter):** Controls the decay of the weights. A larger h results in a smoother image but may blur fine textures, while a smaller h preserves details but may retain noise.
- ***a* (Gaussian Kernel Parameter):** The standard deviation of the Gaussian kernel G_a used within the patch distance calculation. It assigns higher importance to the center of the patch, making the comparison robust to small geometric distortions.

For implementation, I utilized the OpenCV function `fastNlMeansDenoising`. Note that this standard implementation typically utilizes a uniform patch weighting (equivalent to $a \to \infty$).

II..5 Mixed Noise Removal

Noise Identification: The input image was analyzed and found to contain:

- **Impulse Noise:** Identified by random sharp black and white outliers.
- **Gaussian Noise:** Identified by the background grainy texture.

Filter Selection and Order: To achieve the best results, I adopted a cascaded strategy.

1. **Step 1: Median Filter**(3×3): A filter used to eliminate the impulse outliers.
2. **Step 2: Bilateral Filter**$(5 \times 5,\ \sigma_d = 2.0,\ \sigma_r = 30.0)$: An advanced edge-preserving filter used to smooth the Gaussian grain without blurring the sharp edges of the object.

Explanation of Filter Order: The Bilateral filter relies on analyzing pixel intensity differences to preserve edges. If the salt-and-pepper outliers are not removed first, the Bilateral filter will misinterpret them as strong edges or features, preserving them or smearing them locally. Removing outliers first ensures the Bilateral filter operates only on the underlying Gaussian noise and true image geometry.

III. Experimental Results

III..1 Original Image

The baseline PSNR for the noisy image was established before filtering.

Table 1: PSNR of Original Image (dB)

Image	PSNR
flower_gray_noisy	19.04260

III..2 Basic Linear Filtering

I tested Uniform and Gaussian filters with varying kernel sizes. For the Gaussian filter, I used the optimal σ derived from the kernel size, as well as a large σ to simulate a uniform filter.

Table 2: PSNR Comparison of Linear Filters (dB)

Kernel Size	**Uniform Filter**	**Gaussian (Optimal σ)**	**Gaussian (Large σ)**
3×3	26.90	26.43 ($\sigma = 0.8$)	26.90
5×5	26.74	**27.67** ($\sigma = 1.1$)	26.74
7×7	25.42	27.50 ($\sigma = 1.4$)	25.42
9×9	24.23	26.92 ($\sigma = 1.7$)	24.23
15×15	22.46	25.13 ($\sigma = 2.6$)	22.46

III..3 Bilateral Filtering

I analyzed the performance by varying the spatial parameter σ_c and the range parameter σ_s with a fixed 5×5 window.

Table 3: PSNR (dB) of Bilateral Filter (5×5 Window) under varying σ_c and σ_s

Spatial	**Range Parameter (σ_s)**				
Parameter (σ_c)	**10**	**30**	**60**	**100**	**150**
0.5	19.11	19.74	20.58	21.43	22.06
1.0	19.34	21.07	23.39	26.28	27.43
2.0	19.49	21.54	24.73	27.64	**27.87**
3.0	19.53	21.62	25.00	27.66	27.68
4.0	19.54	21.64	25.08	27.64	27.59

The best configuration observed was $\sigma_c = 2$ and $\sigma_s = 150$, with a PSNR of **27.8738 dB**.

III..4 Non-Local Means Filtering

I conducted three separate experiments to isolate the effects of the filtering strength h, the patch size N', and the search window size $\aleph$.

Table 4: NLM Parameter Analysis (PSNR in dB)

Variable Parameter	**Fixed Parameters**	**PSNR**
Experiment 1: Effect of Filtering Strength (h)		
$h = 5.00$	$N' = 7, \aleph = 21$	19.04
$h = 10.00$	$N' = 7, \aleph = 21$	19.06
$h = 15.00$	$N' = 7, \aleph = 21$	20.64
$h = 20.00$	$N' = 7, \aleph = 21$	24.83
$\mathbf{h = 25.00}$	$\mathbf{N' = 7}, \aleph = \mathbf{21}$	**27.43**
$h = 30.00$	$N' = 7, \aleph = 21$	27.04
$h = 35.00$	$N' = 7, \aleph = 21$	26.13
Experiment 2: Effect of Patch Size (N′)		
$N' = 3$	$h = 25, \aleph = 21$	24.67
$N' = 5$	$h = 25, \aleph = 21$	27.20
$\mathbf{N' = 7}$	$\mathbf{h = 25}, \aleph = \mathbf{21}$	**27.43**
$N' = 9$	$h = 25, \aleph = 21$	27.29
$N' = 11$	$h = 25, \aleph = 21$	27.07
Experiment 3: Effect of Search Window Size (ℵ)		
$\aleph = 11$	$h = 25, N' = 7$	26.17
$\aleph = \mathbf{21}$	$\mathbf{h = 25}, \mathbf{N' = 7}$	**27.43**
$\aleph = 31$	$h = 25, N' = 7$	27.27
$\aleph = 41$	$h = 25, N' = 7$	27.05

Final Parameter Choice: $h = 25$, Patch Size $N' = 7$, Search Window $\aleph = 21$.

III..5 Mixed Noise Removal

Visually, the salt-and-pepper noise was completely removed. Unlike standard Gaussian blurring, the Bilateral filter successfully smoothed the wall and petals while maintaining the sharp boundaries of the window shutters.

Table 5: PSNR Performance Comparison

Image State	**PSNR (dB)**
Noisy Input	19.0551
Denoised Output	**28.8088**

Figure 8: Mixed Noise Removal Result of `flower.raw`.

IV. Discussion

IV..1 Basic Linear Filtering

Comparison of Parameters: Unlike the Uniform filter, the Gaussian filter assigns weights based on spatial proximity.

- As shown in Table 2, the **Gaussian filter** with optimal sigma consistently outperformed the Uniform filter for kernels larger than 3×3. The optimal performance was achieved with a 5×5 Gaussian kernel ($\sigma = 1.1$) at **27.67 dB**.
- The **Uniform filter** performance degraded as kernel size increased. This suggests that the image contains Gaussian noise rather than impulse noise. The Uniform filter over-smooths detailed textures, whereas the Gaussian filter preserves local structure better by prioritizing central pixels.

IV..2 Bilateral Filtering

Effect of σ_c and σ_s:

- $\boldsymbol{\sigma_s}$ **(Range)**: This was the most critical parameter. At low values ($\sigma_s = 10$), the filter barely improved the image because it interpreted the noise variations as edges and preserved them. As σ_s increased to 150, the PSNR improved significantly to 27.87 dB, indicating that a wider intensity tolerance was needed to smooth out the noise levels in this specific image.
- $\boldsymbol{\sigma_c}$ **(Spatial)**: Increasing σ_c improved performance up to approximately $\sigma_c = 2$, after which the gains saturated.

Comparison with Linear Filters:

- **Linear Max:** 27.67 dB (5×5 Gaussian)
- **Bilateral Max:** 27.87 dB ($5 \times 5, \sigma_c = 2, \sigma_s = 150$)

The Bilateral filter outperforms the linear Gaussian filter because it selectively smooths. While the Gaussian filter applies the same smoothing kernel across the entire image (blurring actual edges), the Bilateral filter adapts to local intensity differences. Even though a high σ_s was required, the combination of spatial decay and intensity range allowed it to reduce noise in flat regions more effectively than the linear filter.

IV..3 Non-Local Means Filtering

Analysis of Filter Parameters:

1. **Filter Strength (h):** This is the most sensitive parameter. Low values resulted in negligible denoising (19.04 dB) because the weights decayed too sharply, assigning zero weight to almost all neighbors. Increasing h to 25.00 provided the necessary bandwidth to average similar patches, achieving the peak PSNR of 27.43 dB.
2. **Patch Window Size (N'):** Our experiments identified $N' = 7$ as the optimal size.
    - **Too Small ($N' = 3$):** PSNR dropped to 24.67 dB. Small patches are not robust to noise. A noisy pixel can easily make two disparate patches look similar.
    - **Too Large ($N' = 11$):** PSNR dropped to 27.07 dB. Large patches are too strict. It becomes difficult to find other patches that match the complex structure of a large neighborhood, leading to fewer candidates for averaging.
3. **Search Window Size ($\aleph$):** I found $\aleph = 21$ to be optimal (27.43 dB). Increasing the search window further actually decreased the PSNR. While a larger window theoretically offers more candidates, it also increases the probability of finding false patches that look similar due to noise but belong to semantically different structures. This dilutes the quality of the estimate.
4. **Gaussian Patch Parameter (a):** The OpenCV implementation 'fastNlMeansDenoising' does not expose the parameter a, which represents the standard deviation of the Gaussian kernel used to weight pixels inside the comparison patches. In the standard implementation, $a \to \infty$. This implies a uniform weighting where every pixel in the patch N' contributes equally to the distance calculation $\|I(N_{i,j}) - I(N_{k,l})\|^2$. If I could use a finite a, pixels at the center of the patch would contribute more to the similarity score than pixels at the boundary. This is theoretically superior because it makes the comparison more robust to slight geometric distortions. The lack of this parameter in the implementation likely results in slightly lower performance near edges.

Comparison with Previous Methods:

- **NLM (27.43 dB) vs Bilateral (27.87 dB) vs Gaussian (27.67 dB).**

- In this specific experiment, the NLM filter performed worse than the Bilateral filter and slightly worse than the simple Gaussian filter.
- While NLM is theoretically more powerful for repetitive textures, it is highly sensitive to parameter tuning. The optimal $h = 25$ suggests a very high smoothing requirement, which may have blurred fine details. Additionally, the Search Window effect obeservation (where larger windows hurt performance) suggests that the image may not possess enough self-similarity within the local vicinity to justify the NLM approach over the locally adaptive Bilateral filter.

IV..4 Mixed Noise Removal

1. **Shortcoming:** While the Bilateral filter preserves edges better than a Gaussian filter, it is computationally more expensive ($O(N^2)$ per pixel) due to the calculation of range weights for every neighbor.
2. **Shortcoming:** The fixed parameters (σ_d, σ_r) may not be optimal for all regions if the noise variance changes spatially.
3. **Improvement Suggestion:** To speed up the process, Fast Bilateral Filtering or Guided Filtering could be implemented, which offers similar edge-preserving properties with $O(1)$ complexity.
4. **Improvement Suggestion:** Using BM3D (Block-Matching and 3D Filtering) would likely yield even higher PSNR as it exploits non-local self-similarity in the image.

Problem 3: Color Correction - Auto White Balancing

I. Abstract and Motivation

The visual quality of a digital image is heavily dependent on the spectral distribution of the illuminant. When a scene is captured under non-neutral lighting, such as sunset or indoor incandescent light, a color cast occurs, shifting the colors away from their natural appearance. The motivation of this experiment is to implement the Gray World Theory for Auto White Balancing (AWB). This theory assumes that the average reflectance of a natural scene is achromatic. By forcing the average of the Red, Green, and Blue channels to be equal, we can neutralize the influence of the light source and restore natural color balance.

II. Approach and Procedures

To implement the AWB, the following mathematical procedures were followed:

1. **Mean Calculation:** I calculated the average intensity for each channel across the image:
$$\mu_R = \frac{1}{N}\sum R, \quad \mu_G = \frac{1}{N}\sum G, \quad \mu_B = \frac{1}{N}\sum B$$
2. **Target Estimation:** I averaged the three channel means to get target grayscale value:
$$\mu = \frac{\mu_R + \mu_G + \mu_B}{3}$$
3. **Channel Scaling:** Gain coefficients were calculated to match each channel to the target:
$$\alpha_R = \frac{\mu}{\mu_R}, \quad \alpha_G = \frac{\mu}{\mu_G}, \quad \alpha_B = \frac{\mu}{\mu_B}$$
4. **Transformation and Clipping:** Each pixel was scaled by its respective gain. Values exceeding 255 were clipped to ensure the output remained within the 8-bit range.

III. Experimental Results

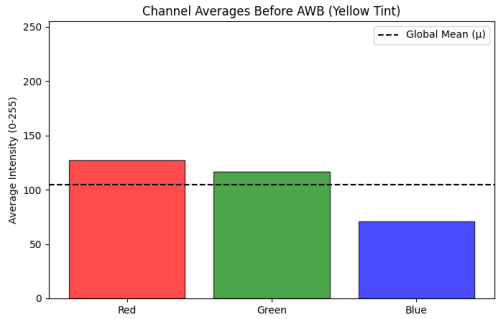
The implementation yielded the following statistical data:

Metric	**Red (R)**	**Green (G)**	**Blue (B)**
Means Before	126.9695	116.7010	70.9956
Target Mean (μ)		104.8887	
Means After	104.4001	104.4065	104.3949

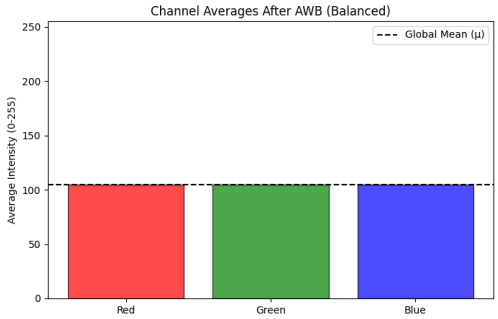
Table 6: Comparison of channel means before and after AWB.

(a) Original Image (b) Color-corrected Image

Figure 9: Comparison of Color Correction Techniques.



(a) Original Color Distribution Histogram



(b) Processed Color Distribution Histogram

Figure 10: Comparison of Color Distribution Histogram.

Before applying AWB, the channel averages show a significant discrepancy. The Red and Green channels are much higher than the Blue channel. This indicates a strong yellow-orange color cast. The Blue channel is under-represented, which explains why the original image appears muddy and warm.

After applying the Gray World algorithm, the channel averages were re-calculated. The resulting means are nearly identical and align closely with the target μ. This indicates that the histograms have been shifted to overlap, effectively balancing the color distribution.

IV. Discussion

- **Visual Observation:** The original image had a distinct yellow tint caused by the high intensity of Red and Green relative to Blue. After correction, the image appears more natural. The clouds should appear white or gray, and the water should regain its characteristic blue-cyan hue.

- **Mathematical Observation:** The success of the Gray World Theory is evident as the standard deviation between the three channel means dropped from approximately 24.8 to nearly 0.

- **Clipping Effects:** We notice the "Means After" are slightly lower (104.4) than the "Target Mean" (104.88). This is due to the clipping process. When the Blue channel gain $\alpha_B \approx 1.47$ was applied, several pixels likely exceeded 255 and were capped, resulting in a slightly lower final average than the theoretical target.