Boolean Algebra-Based Channel Isolation and Bilinear Demosaicing of Bayer CFA: Implementation and Quality Analysis Using Sony A6400 RAW Imagery

Jonathan Lewie - 13525136
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: levvrae@gmail.com , 13525136@std.stei.itb.ac.id

***Abstract*—A Color Filter Array (CFA) restricts each photosite on a camera sensor to a single color channel, so a RAW capture is never a finished color image, and must instead be reconstructed by an algorithm called demosaicing. The filter arrangement that forces this reconstruction is almost always treated as an engineering detail, even though it carries an exact Boolean structure that is seldom written down. This paper presents a formal analysis of the Bayer CFA pattern using Boolean algebra, and implements a bilinear demosaicing pipeline applied to RAW imagery from a Sony Alpha 6400. We prove that the RGGB Bayer pattern induces a Boolean partition of the pixel coordinate space into three mutually disjoint channel subsets, and show that channel isolation reduces to Boolean-mask gating, equivalent to a Boolean AND on the binary support of the sensor array. Evaluated across five photographic conditions, PSNR and SSIM metrics are computed against rawpy's AHD demosaicing output as a reference, not against a ground-truth full-color sensor, so reported values reflect the pipeline gap between bilinear and AHD interpolation. Results show that reconstruction quality degrades with increasing spatial frequency, PSNR and SSIM capture complementary aspects of quality, and channel sampling density governs per-channel interpolation error.**

Keywords—Bayer filter; Boolean algebra; color filter array; demosaicing; bilinear interpolation; RAW image processing; image reconstruction

I. INTRODUCTION

Most people who shoot with a camera have never thought about what actually happens between pressing the shutter and seeing a color photo. The sensor itself cannot tell colors apart. It just counts light. Every photosite on the sensor produces a single number representing how bright the incoming light was, with no information about whether that light was red, green, or blue. To get color, a mosaic of tiny colored filters is placed over the sensor so each pixel only receives one color band. The result is that the raw readout from a camera sensor is not a color photo at all. It is a mosaic of single-color dots, where each pixel carries only one out of three color channels. Turning that mosaic into an actual RGB image requires an algorithm that guesses the two missing colors at every single pixel. This reconstruction process is called demosaicing, and without it every digital photo would just look like a grid of colored speckles.

The motivation for this paper came from working with RAW files from the Sony Alpha 6400 in Lightroom and noticing how much more data is available compared to JPEG. When shooting JPEG, the camera applies its own demosaicing and color processing internally, then discards the original 14-bit sensor data and writes an 8-bit output, all before the file reaches storage. Shooting RAW instead saves the raw sensor mosaic at full 14-bit precision, which is why ARW files are much larger than JPEGs and why they can recover significantly more detail in post-processing. There is genuinely more information in the file. What is less obvious, and what this paper investigates, is that the arrangement of color filters in the Bayer CFA follows a strict repeating pattern where every pixel's color is completely determined by just two things: whether its row index is odd or even, and whether its column index is odd or even. Those are two Boolean variables. This means that isolating a single color channel from the raw data is, at a formal mathematical level, a Boolean AND masking operation.

This paper looks at Bayer CFA demosaicing through a Boolean algebra lens. The core idea is to formally prove that the RGGB pattern is a Boolean partition, show that channel isolation is Boolean AND masking, implement bilinear demosaicing from scratch in Python, and measure how well it works across five different shooting conditions using actual ARW files from the A6400. Three questions structure the investigation: (1) Can the Bayer CFA be formally expressed as a Boolean partition function? (2) How do Boolean masking operations map to channel isolation? (3) How does bilinear demosaicing performance vary across different photographic conditions?

The paper is organized as follows. Section II covers theoretical background. Section III develops the Boolean formalization and proof. Section IV describes the implementation. Section V presents results. Section VI concludes.

II. Theoretical Background

A. Boolean Algebra

A set restricted to {0,1} is not automatically a Boolean algebra. What matters is that AND, OR, and NOT stay closed under a specific set of laws. Every element must have a complement that returns an identity when the two are combined, each of the two binary operations must distribute over the other, and both must be commutative and associative [3]. The structure dates to Boole's 1854 work [2], which applied algebraic manipulation to logical propositions instead of quantities, and circuit design and binary computation later adopted it as their formal foundation. For this paper, the operation that matters is AND, since given two binary-valued functions over the same domain, AND returns 1 only where both inputs agree on 1, and this single property is what makes channel isolation in Section III well-defined. Equally important for this paper is the concept of a partition, namely a collection of non-empty subsets of a set S that are mutually disjoint (no overlap) and whose union covers all of S. Boolean algebra provides a natural language for defining and verifying partitions because the disjointness condition can be stated directly in terms of AND operations between indicator functions, and exhaustive coverage can be verified via OR.

B. Bayer Color Filter Array

Each photosite on a digital sensor functions as a photon counter, and nothing more, as it accumulates charge proportional to how much light hits it during exposure, regardless of that light's wavelength. Color is not something a single photosite can report on its own, since it captures intensity rather than wavelength. This is precisely the gap that the Color Filter Array exists to fill, restricting which wavelengths reach each photosite so that the array converts an otherwise color-blind sensor into one capable of color imaging. Among CFA designs the Bayer pattern dominates in practice; Bayer filed the original patent in 1976 [1]. It arranges red, green, and blue filters in a repeating 2×2 tile with the RGGB layout, namely R at position (0,0), G at (0,1) and (1,0), and B at (1,1). Two of those four tile positions go to green, a ratio that traces back to how human vision processes brightness rather than to any constraint of the filter manufacturing process. Perceived sharpness in an image is driven mostly by luminance contrast, and the eye's luminance response peaks in the green region of the visible spectrum [4]. Doubling the green sample density means the channel responsible for most of what we perceive as detail gets twice the spatial resolution of red or blue, at no added cost in filter complexity, since the RGGB tile remains a simple repeating 2×2 unit. For the Sony Alpha 6400 used in this study, rawpy reads the sensor pattern as raw_pattern = [[0,1],[3,2]], where 0 = R, 1 = G1, 3 = G2, and 2 = B. The two green positions (G1 and G2) are treated as a single logical G channel in demosaicing. The ARW format stores raw values as 14-bit integers in the range [0, 16383], giving 16,384 possible intensity levels per pixel before any processing.

C. Demosaicing

Because each pixel only recorded one color, the camera now has to guess the other two colors at every single pixel location. This is the demosaicing problem, and it is ill-posed, meaning two out of three color values at every pixel were never recorded, so a single mosaic is consistent with many different full-color scenes. The partition from Section III makes the severity concrete, since wherever a channel is sampled at only 25% density there are fewer known neighbors to pin the guess down, so reconstruction there is less constrained than in the 50%-sampled green channel. Different demosaicing algorithms make different assumptions about how a missing color value relates to its known neighbors, ranging from assuming local smoothness, where nearby same-channel pixels are simply averaged, to more sophisticated schemes that adapt the interpolation to local image structure before making a guess [5]. The simplest approach is bilinear interpolation, which fills in a missing color at some pixel by looking at the nearest pixels that do have that color and taking a weighted average. For instance, to estimate the red value at a pixel position that recorded green, take the red values from the nearest red-sampled neighbors and average them. This works reasonably well in smooth areas where colors do not change rapidly, but it fails at sharp edges. Averaging across a color boundary produces a value that belongs to neither region, which shows up as color fringing or blur along edges [6]. The relationship to Boolean algebra is that which neighbors exist and how far away they are is directly determined by the partition structure from Section III. Adaptive Homogeneity-Directed (AHD) demosaicing [7] handles edges better by detecting gradient direction and interpolating along edges rather than across them. In this study, AHD is used as the reference output to compare against, not as a true ground truth but as a reasonable upper bound, for what a more sophisticated algorithm can do from the same raw input.

III. Boolean Formalization of Bayer CFA

A. Pixel Coordinate Space as an Ordered Set

Let the sensor pixel coordinate space be defined as a finite set P of ordered pairs:

$$P = \{(i, j) \mid i \in \{0, \dots, H-1\}, j \in \{0, \dots, W-1\}\} \quad (1)$$

where H and W are sensor height and width. For the Sony Alpha 6400, H = 4024, W = 6024, giving |P| = 24,240,576. Each element (i,j) in P corresponds to one photosite recording one scalar intensity value.

B. Bayer Pattern as a Boolean Function

Define the channel assignment function $\phi: P \rightarrow \{R, G, B\}$ mapping each pixel coordinate to its color channel. Let r = (i mod 2) and c = (j mod 2) denote row and column parities. For the RGGB arrangement, φ reduces to Boolean expressions:

$$\phi = R \Leftrightarrow \neg r \wedge \neg c \quad (2)$$

$$\phi = G \Leftrightarrow [\neg r \wedge c] \vee [r \wedge \neg c] \quad (3)$$

$$\phi = B \Leftrightarrow r \wedge c \quad (4)$$

The function φ is periodic with period 2 in both dimensions, total (defined for all (i,j) in P), and surjective onto {R, G, B}. The Bayer pattern is entirely determined by two Boolean variables, namely row parity and column parity.

C. Proof: RGGB Pattern Forms a Partition of P

Define P_R = {(i,j) in P | φ(i,j) = R}, P_G = {(i,j) in P | φ(i,j) = G}, P_B = {(i,j) in P | φ(i,j) = B}.

Theorem 1: {P_R, P_G, P_B} is a partition of P.

Proof. *Condition 1 (Non-emptiness):* Since H >= 2 and W >= 2, pixel (0,0) in P_R, (0,1) in P_G, (1,1) in P_B. All three subsets are non-empty.

Condition 2 (Mutual disjointness): Suppose P_R ∩ P_G != empty. Some (i,j) satisfies φ = R and φ = G. From (2), φ = R requires r = 0 and c = 0. Substituting into (3): NOT(0) AND 0 OR 0 AND NOT(0) = 0, contradicting φ = G. For P_G ∩ P_B: suppose some (i,j) satisfies φ = G and φ = B. From (4), φ = B requires r = 1 and c = 1. Substituting into (3): NOT(1) AND 1 OR 1 AND NOT(1) = 0 OR 0 = 0, contradicting φ = G. The case P_R ∩ P_B follows symmetrically: φ = R requires (r,c) = (0,0) while φ = B requires (r,c) = (1,1); these are distinct Boolean input combinations. All three pairs are therefore disjoint.

Condition 3 (Exhaustive coverage): Since φ is total, every (i,j) in P maps to exactly one of {R, G, B}, so P_R ∪ P_G ∪ P_B = P. QED

For the A6400 sensor, |P_R| = |P_B| = 6,060,144 (25.0% each), |P_G| = 12,120,288 (50.0%). These were verified computationally and match exactly. The partition structure established in Theorem 1 has a direct consequence for implementation, since it guarantees that the Boolean mask construction in Algorithm 1 will produce exactly three non-overlapping coverage arrays that together tile the full sensor array. This means there is no need to check for overlap or handle boundary cases between channels, as the partition proof already guarantees correctness. The implementation in Section IV follows directly from this structure.

D. Channel Isolation as Boolean-Mask Gating

Define the Boolean mask M_C : P → {0,1} as M_C(i,j) = 1 if (i,j) ∈ P_C, and 0 otherwise. Each mask is the indicator function of one partition subset, so by Theorem 1 the three masks satisfy M_R ∨ M_G ∨ M_B = 1 and M_C ∧ M_D = 0 for C ≠ D at every coordinate. Let f : P → [0,1] be the normalized Bayer array. The isolated channel array is obtained by gating f with the mask:

$$f_C(i,j) = f(i,j) \cdot M_C(i,j) \quad (5)$$

Here the multiplication acts as a gate: where M_C = 1 the intensity passes through unchanged, and where M_C = 0 it is forced to zero. On the binary support of f (treating any nonzero intensity as logical 1), this gating coincides exactly with the Boolean AND of f against M_C, which is why channel isolation inherits its structure from the partition rather than from the arithmetic. Three properties follow directly from Theorem 1: (1) Completeness: f_R + f_G + f_B = f pointwise, since the masks tile P; (2) Non-interference: f_C · f_D = 0 for C ≠ D, since disjoint masks never co-activate; (3) Mask-invariance: f_C · M_C = f_C, since re-gating an already-gated channel changes nothing.

IV. Implementation

A. Dataset

Five RAW images were captured with a Sony Alpha 6400 in ARW format at 14-bit per-channel depth across five distinct photographic conditions. All images were captured with in-camera processing disabled in ARW-only mode. Table I summarizes capture parameters. The five conditions were chosen to cover the range of situations where bilinear demosaicing is known to behave differently. Condition 1 (high spatial frequency) tests the worst case, since dense edges and fine textures are exactly where the local-averaging assumption breaks down. Condition 2 (low light, ISO 8000) tests behavior under sensor noise rather than scene content variation. Conditions 3 and 4 (single-channel saturation) were included specifically to test the partition symmetry prediction from Theorem 1: if the Boolean partition assigns exactly 25% density to both R and B, the interpolation difficulty should be symmetric regardless of which channel dominates. Condition 5 (flat, low contrast) provides the best-case baseline, where smooth regions make bilinear averaging most accurate. Together, they form a set that probes worst case, noise case, symmetry case, and best case in a single experiment.

TABLE I. Dataset Capture Parameters

Condition	File	ISO	Shutter	f-stop
High Detail	LEW02292.ARW	125	1/1500s	f/1.8
Low Light	2026-02-15_037.ARW	8000	1/1000s	f/1.8
High Sat. Red	LEW00005.ARW	400	1/45s	f/5.6
High Sat. Blue	LEW00007.ARW	100	1/45s	f/5.6
Flat/Contrast	LEW00009.ARW	200	1/60s	f/1.8

Figures 1-5 show the raw photographic content for each condition as captured by the Sony Alpha 6400. Each image is shown as rendered by rawpy postprocessing for visualization purposes, while the actual pipeline uses unprocessed ARW sensor data.

Fig. 1 Condition 1 (High Detail), LEW02292.ARW. ISO 125, 1/1500s, f/1.8, 14:21 (daylight). High spatial frequency scene with rich edge content and fine texture detail. Best for testing demosaicing performance at high-frequency regions.

Fig. 2 Condition 2 (Low Light), 2026-02-15_037.ARW. ISO 8000, 1/1000s, f/1.8, 21:17 (night). High ISO condition with significant sensor read noise. Noise ratio (std/mean) = 1.312, highest in dataset. Tests demosaicing behavior under noise-dominated signal.

Fig. 3 Condition 3 (High Saturation Red), LEW00005.ARW. ISO 400, monitor displaying #FF0000, 1/45s, f/5.6. Shutter was 1/45s as recorded by EXIF; at this speed minor 60Hz banding may theoretically occur, but was not observed in channel statistics. R channel mean = 0.0700; B channel near zero (0.0037). Tests single-channel dominance.

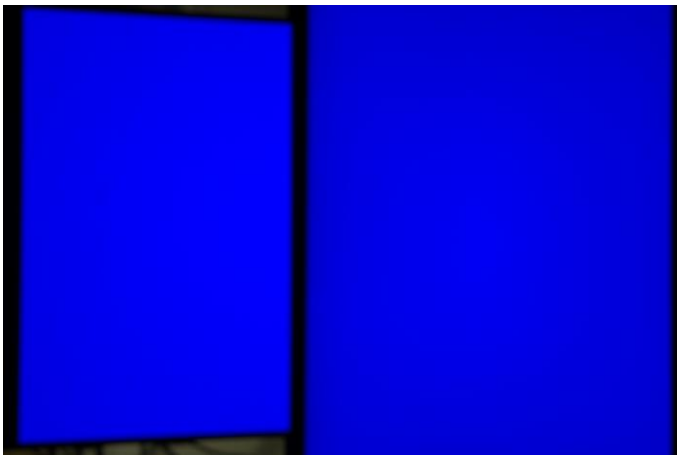

Fig. 4 Condition 4 (High Saturation Blue), LEW00007.ARW. ISO 100, monitor displaying #0000FF, 1/45s, f/5.6. B channel mean = 0.0600; R channel near zero (0.0032). Symmetric counterpart to Condition 3 for partition geometry analysis.

Fig. 5 Condition 5 (Flat/Low Contrast), LEW00009.ARW. ISO 200, 1/60s, f/1.8, 22:17. Local variance = 693.5 (lowest in dataset), gradient magnitude = 39.9. Smooth, minimal-edge content serving as baseline for bilinear interpolation.

B. Preprocessing

Raw ARW files were loaded using rawpy [8]. The visible Bayer array was extracted via raw.raw_image_visible, yielding a 2D integer array of shape (4024, 6024). Black-level subtraction and normalization were applied:

$$f_{\text{norm}} = \text{clip}\left(\frac{f_{\text{raw}} - L_{\text{black}}}{L_{\text{white}} - L_{\text{black}}}, 0{,}1\right) \quad (6)$$

with L_black = 512 and L_white = 15360 for the A6400. Clipping prevents noise floor artifacts from propagating downstream.

C. Boolean Channel Isolation

Boolean masks M_R, M_G, M_B were constructed as NumPy boolean arrays of shape (4024, 6024) by iterating over

the 2x2 raw_pattern tile and assigning positions per equations (2)-(4). rawpy encodes channel assignments as integers, where 0 maps to R, 1 and 3 map to G1 and G2 respectively, and 2 maps to B, consistent with the channel assignment function phi defined in (2)-(4). Pixel counts confirmed Theorem 1 exactly, as M_R and M_B each cover 25.0% of positions (6,060,144 pixels each), while M_G covers 50.0% (12,120,288 pixels).

Algorithm 1 presents the pseudocode for Boolean mask construction and channel isolation:

Algorithm 1: Boolean Channel Isolation from Bayer CFA

```
   Input : bayer_array (H x W), raw_pattern (2 x 2)
   Output : f_R, f_G, f_B (isolated channel arrays)

   1.   Initialize M_R, M_G, M_B ← zeros(H, W,
dtype=bool)
   2.  FOR row FROM 0 TO 1 DO
   3.      FOR col FROM 0 TO 1 DO
   4.          ch ← raw_pattern[row, col]
   5.          IF ch == 0 THEN
   6.              M_R[row::2, col::2] ← True    //
Assign R positions
   7.          ELSE IF ch IN {1, 3} THEN
   8.              M_G[row::2, col::2] ← True    //
Assign G positions
   9.          ELSE IF ch == 2 THEN
   10.             M_B[row::2, col::2] ← True    //
Assign B positions
   11.         END IF
   12.     END FOR
   13. END FOR
   14. f_R ← bayer_array ⊙ M_R  // Mask-gating, Eq.
(5)
   15. f_G ← bayer_array ⊙ M_G
   16. f_B ← bayer_array ⊙ M_B
17. RETURN f_R, f_G, f_B
```

D. Bilinear Demosaicing

Bilinear demosaicing was implemented from scratch using SciPy ndimage.convolve on isolated channel arrays. The kernel weights follow a simple principle, namely that closer neighbors get more influence than farther ones. To understand why the specific values are 0.5 and 0.25, consider what the neighborhood looks like for an R or B pixel that needs to be filled in. Because R and B each occupy only 25% of positions (every other row and every other column), the nearest known same-channel pixels sit at two different distances. The four pixels directly above, below, left, and right are one step away. The four diagonal pixels are farther, at a distance of roughly 1.41 steps (diagonal of a unit square). Bilinear interpolation weights contributions proportionally to distance: the four close neighbors each get weight 0.5, and the four diagonal neighbors each get weight 0.25. When this kernel is applied and the result is divided by the count of contributing known pixels, the weighted average gives a reconstructed value that reflects the actual spatial arrangement of same-channel pixels in the Bayer partition. For the G channel, which occupies 50% of positions in a cross pattern, same-channel neighbors only exist at the four cardinal positions (distance 1), so only those four weights are non-zero. Two 3x3 kernels encode this geometry, following the standard bilinear convention for Bayer arrays [6]. For R and B channels (25% density):

$$K_{RB} = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} \quad (7)$$

For the G channel (50% density, cross-pattern positions):

$$K_G = \begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 1 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix} \quad (8)$$

Algorithm 2 presents the interpolation pseudocode:

Algorithm 2: Bilinear Interpolation Demosaicing

```
   // Sub-procedure: InterpolateChannel(f_C, M_C,
K)
   Input : f_C (isolated channel array), M_C
(Boolean mask), K (kernel)
   Output : Reconstructed full single-channel array

   1.  val_conv ← convolve(f_C, K, mode='reflect')
// Sum weighted pixel values
   2.  cnt_conv ← convolve(M_C, K, mode='reflect')
// Count known pixels
   3. interp  ← val_conv / cnt_conv              //
Normalized weighted average
   4.     ch_full     ←  WHERE(M_C,   f_C,   interp)
// Keep original, fill rest
   5.  RETURN clip(ch_full, 0, 1)                //
Clamp to valid range [0, 1]

   // Main Procedure: Demosaicing Reconstruct
   Input : f_R, f_G, f_B, M_R, M_G, M_B, K_RB, K_G
   Output : output (H x W x 3 RGB array)

   6.  R_full ← InterpolateChannel(f_R, M_R, K_RB)
// Reconstruct red channel
   7.  G_full ← InterpolateChannel(f_G, M_G, K_G)
// Reconstruct green channel
   8.  B_full ← InterpolateChannel(f_B, M_B, K_RB)
// Reconstruct blue channel
   9.    output  ←  stack(R_full,  G_full,  B_full)
// Combine into RGB array
10. RETURN output
```

E. Evaluation

Demosaiced output was evaluated against rawpy's AHD pipeline [7] with camera white balance as reference. Two metrics were computed using scikit-image [9], namely PSNR and SSIM [10]. As noted, the AHD reference does more than interpolate: on top of edge-directed interpolation, rawpy applies camera white balance, gamma correction, and sRGB color-space conversion, none of which the bilinear pipeline here performs. Absolute metric values therefore reflect this total pipeline gap rather than interpolation quality alone, while relative patterns across conditions remain valid for analysis.

V. Experiments and Analysis

A. Experimental Setup

All experiments were conducted in Python 3.11 on Windows 11 using rawpy [8], NumPy 1.26, SciPy 1.12, scikit-image [9], and Matplotlib 3.8 on an Intel Core Ultra 9 285H (16 CPUs, ~2.9GHz) processor with 32 GB RAM. The complete pipeline was applied identically to all five images with no condition-specific tuning. Each run processed the full 4024x6024 Bayer array (approximately 24.2 million pixels) without subsampling. Average bilinear demosaicing time per image was approximately 1.47 seconds (range: 1.41 to 1.63 seconds across five conditions), with the two convolution operations in Algorithm 2 accounting for the majority of processing time. Peak memory usage was under 2 GB per run. The complete implementation and dataset are publicly available at https://github.com/Levvroy/BayerDemosaic for reproducibility.

B. Quantitative Results

Table II reports PSNR and SSIM for each condition alongside per-channel mean intensity values measured from the Boolean-masked Bayer arrays.

TABLE II. Demosaicing Performance Per Condition

Condition	PSNR (dB)	SSIM	R Mean	G Mean	B Mean
High Detail	10.7849	0.4680	0.1085	0.3009	0.1768
Low Light	15.1818	0.2407	0.0437	0.0870	0.0335
High Sat. Red	12.7243	0.3428	0.0700	0.0278	0.0037
High Sat. Blue	12.6733	0.4538	0.0032	0.0270	0.0600
Flat/Contrast	14.3797	0.4461	0.0341	0.0783	0.0337
Average	13.1488	0.3903	-	-	-

C. Analysis

1) Effect of Scene Spatial Frequency

The High Detail condition produced the lowest PSNR at 10.7849 dB and a moderate SSIM of 0.4680. This result is expected, since the scene captured in Condition 1 has dense edge content and fine texture, which is exactly the type of content where bilinear interpolation performs worst. The underlying reason is straightforward, as bilinear interpolation assumes color values vary smoothly between neighboring pixels of the same channel. At sharp edges, this assumption fails. The pixel immediately to the left of an edge might belong to a completely different region than the pixel to the right, so averaging them produces a value that is wrong for both regions. The visible result is color fringing and slight edge blur. At the opposite extreme, the Flat/Low Contrast condition (Condition 5) produced PSNR = 14.3797 dB and SSIM = 0.4461, which are the best scores outside of the noise-affected Low Light case. In a smooth, low-contrast scene, neighboring same-channel pixels genuinely do have similar values, so the bilinear average is a reasonable estimate. The gap of approximately 3.6 dB between these two conditions illustrates directly how much scene spatial frequency affects bilinear demosaicing quality in practice.

2) PSNR-SSIM Divergence in Low-Light Conditions

The Low Light condition gives a result that seems contradictory, namely the highest PSNR (15.1818 dB) but the lowest SSIM (0.2407) in the whole dataset. One metric says the quality is best, the other says it is worst. To understand why, it helps to think about what each metric actually measures. PSNR is essentially an accountant, in that it compares each pixel's numerical value in the bilinear output against the same pixel in the AHD reference and calculates how different those numbers are on average. Under ISO 8000, both the bilinear pipeline and the AHD reference receive the same noisy raw input. Both outputs end up full of similar noise patterns from that shared source. When PSNR compares them pixel by pixel, the numbers are actually quite close, since they are both noisy in the same way, so PSNR reports a high score. SSIM works differently. Instead of comparing individual pixel values, it looks at whether local patterns (edges, textures, contrast variations) are preserved from one image to the other. Noise does not form coherent patterns, as it is random speckle that destroys the local structure of whatever is underneath. So while PSNR sees two numerically similar images, SSIM sees two images where every textured region has been replaced by random dots and reports a very low score. The practical implication is that using only PSNR here would lead to the wrong conclusion that Low Light is actually the easiest condition to reconstruct, when visually it is clearly the worst. This result confirms that both metrics are needed, as PSNR alone is not sufficient to evaluate perceptual quality under noisy conditions.

3) Channel Dominance and Partition Geometry

Conditions 3 and 4 were designed to test what happens when one color channel completely dominates the scene. In Condition 3, the scene is a monitor displaying pure red (#FF0000), where the R channel mean is 0.0700, while the B channel mean is only 0.0037, essentially nothing but sensor noise in the blue channel. The PSNR result is 12.7243 dB. In Condition 4, the symmetric case is pure blue (#0000FF), with B channel mean 0.0600, R channel mean 0.0032, and PSNR = 12.6733 dB. The similarity between these two PSNR values is not a coincidence. It reflects something directly predicted by Theorem 1, namely that R and B channels each occupy exactly 25% of the pixel coordinate space under the RGGB partition, symmetrically. When one of these channels carries a flat near-zero signal, it is trivially easy to interpolate. A flat field is perfectly reconstructed by any

interpolation algorithm. But the dominant channel, which occupies only 25% of pixels, is where all the reconstruction error concentrates. The near-identical PSNR between the two saturation conditions is a computational confirmation that the partition geometry is truly symmetric, so it does not matter whether the dominant channel is R or B, the interpolation difficulty is the same because both channels have the same density in the partition. The G channel, at 50% density, does not have a symmetric saturation counterpart tested here, but this asymmetry in sampling density is what makes the green channel generally more reliably reconstructed in bilinear demosaicing. A secondary observation from the per-image runtime data is that Conditions 3 and 4 (High Saturation Red and Blue) completed in approximately 6.7 seconds each, compared to roughly 9.5 to 10.0 seconds for the other three conditions. Since bilinear demosaicing time itself was consistent at around 1.4 to 1.6 seconds across all five images, the difference originates from the SSIM computation, as saturated single-channel images have simpler local structure, which the SSIM sliding-window computation traverses faster than scenes with complex textures and edges.

D. Visual Results

Figures 6-10 show four-panel visualizations for each condition: (a) raw Bayer mosaic after normalization, (b) Boolean channel mask overlay confirming the partition structure of Theorem 1 with no overlap between R, G, B masks, (c) bilinear demosaiced output, (d) AHD reference. Color fringing at edges is most visible in Fig. 6. Structural degradation under noise is evident in Fig. 7. Symmetric channel dominance of Figs. 8 and 9 is confirmed in the mask overlay. Fig. 10 shows the closest visual correspondence between output and reference.

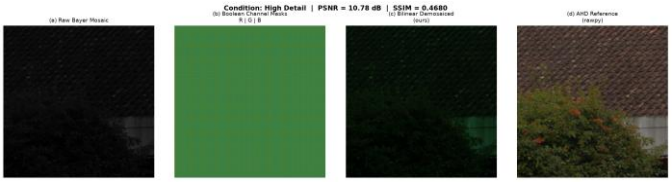

Fig. 6 High Detail condition (LEW02292): (a) normalized Bayer mosaic, (b) Boolean channel masks, (c) bilinear demosaiced output, (d) AHD reference. PSNR = 10.78 dB, SSIM = 0.4680. Color fringing and edge blurring most visible in (c) compared to (d).

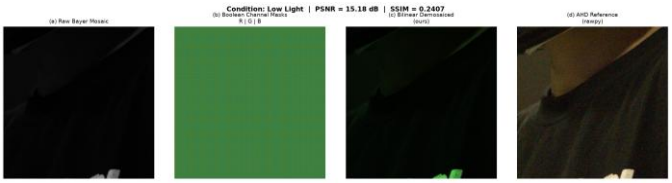

Fig. 7 Low Light condition (ISO 8000): PSNR = 15.18 dB, SSIM = 0.2407. High numerical proximity to reference (high PSNR) coexists with severe structural degradation (low SSIM), demonstrating metric complementarity.

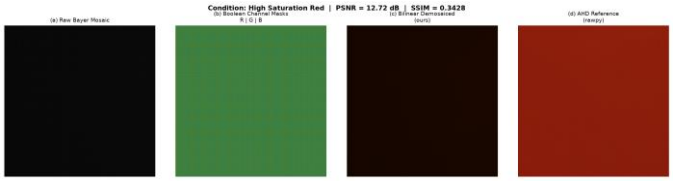

Fig. 8 High Saturation Red condition (#FF0000): PSNR = 12.72 dB, SSIM = 0.3428. B channel mask (blue) shows near-zero signal. Dominant R channel at 25% Bayer density bears the dominant reconstruction error.

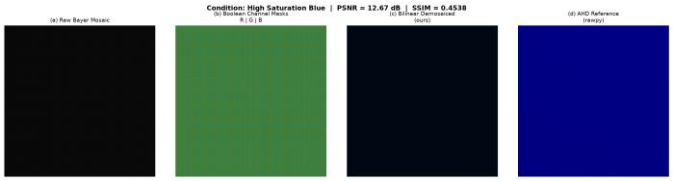

Fig. 9 High Saturation Blue condition (#0000FF): PSNR = 12.67 dB, SSIM = 0.4538. Symmetric error pattern with Fig. 8, confirming that partition geometry, not color identity, determines per-channel interpolation error.

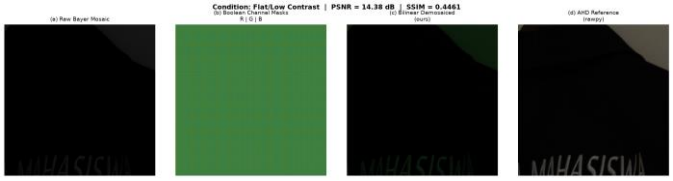

Fig. 10 Flat/Low Contrast condition (LEW00009): PSNR = 14.38 dB, SSIM = 0.4461. Closest visual match between bilinear output and AHD reference across all five conditions, consistent with highest non-noise SSIM.

VI. Conclusion

The main question this paper tried to answer was whether the Bayer CFA pattern has a genuine Boolean algebra structure, or just a loose analogy. Based on the proof and the computational verification, the answer is that it is genuinely formal. Theorem 1 shows that the RGGB pattern divides the pixel coordinate space P into three subsets P_R, P_G, and P_B that form a valid Boolean partition, meaning non-empty, mutually disjoint, and exhaustively covering P. The proof works by showing that the channel assignment function φ reduces to simple Boolean expressions in row and column parities, making disjointness and exhaustive coverage directly verifiable. The pixel counts from the actual A6400 sensor then matched the prediction exactly, namely 25% R, 50% G, 25% B. What this means practically is that channel isolation, which in software just looks like array indexing, has a provable algebraic basis. It is Boolean-mask gating that coincides with a Boolean AND on the binary support, and properties like non-interference between channels follow directly from the partition, not from implementation details.

The bilinear pipeline was implemented from scratch and tested across five RAW conditions. A few results stood out. The most significant was probably the Low Light result, namely the highest PSNR but lowest SSIM in the dataset, which showed that the two metrics can tell completely opposite stories when the primary image degradation is noise rather than interpolation error. That difference would have been invisible with only one

metric. The spatial frequency result was more expected, as the high-detail scene scored about 3.6 dB lower than the flat scene, confirming that bilinear averaging across sharp edges is where the method visibly breaks down. The symmetric result between the Red and Blue saturation conditions (12.72 vs 12.67 dB) was a direct confirmation that the partition structure from Theorem 1 is symmetric. R and B have identical 25% density, so interpolation difficulty between them is the same regardless of which channel dominates the scene.

There are two main limitations to acknowledge. First, the AHD output used as reference is not a true ground truth. It is just a more sophisticated algorithm's output from the same noisy raw input, and on top of edge-directed interpolation it carries rawpy's white balance, gamma correction, and color-space conversion that the bilinear pipeline here does not. So the reported PSNR and SSIM values measure the gap between bilinear and AHD, not the gap between bilinear and the original scene. For the purposes of this paper, this is sufficient to show relative trends across conditions, but it means the absolute numbers should not be interpreted as absolute accuracy. Second, the five capture conditions, while chosen to be representative, cover specific points in the space of possible scenes and are limited by the images available. Potential directions for future work include constructing a synthetic test dataset using a sensor that captures full color per pixel (a three-chip or Foveon sensor) as ground truth, which would allow the absolute accuracy of bilinear demosaicing to be measured directly. Another direction would be to extend the Boolean partition framework to non-uniform or higher-order CFAs beyond the standard 2x2 Bayer tile, or to direction-aware kernels where interpolation weights adapt to local gradient direction within each partition subset, moving closer to how edge-directed algorithms like AHD operate but with an explicit Boolean algebraic foundation.

VIDEO LINK AT YOUTUBE

https://youtu.be/91pWSOkAe8g

ACKNOWLEDGMENT

Above all, the author is grateful to God for the strength and clarity needed to see this work through to completion. The author also owes a debt of gratitude to Dr. Ir. Rinaldi Munir, M.T., who taught IF1220 Discrete Mathematics at Institut Teknologi Bandung and whose lecture materials on Boolean algebra formed the conceptual groundwork this paper builds on. Without that foundation, the formal treatment of the Bayer CFA partition in Section III would not have been possible. Thanks are also due to the open-source developers behind rawpy, NumPy, SciPy, and scikit-image, the tools this implementation relied on from start to finish.

REFERENCES

[1] B. E. Bayer, "Color imaging array," U.S. Patent 3 971 065, Jul. 20, 1976.

[2] G. Boole, An Investigation of the Laws of Thought. London: Walton and Maberly, 1854.

[3] K. H. Rosen, Discrete Mathematics and Its Applications, 8th ed. New York: McGraw-Hill Education, 2019.

[4] D. Alleysson, S. Susstrunk, and J. Herault, "Linear demosaicing inspired by the human visual system," IEEE Trans. Image Process., vol. 14, no. 4, pp. 439-449, Apr. 2005.

[5] D. Menon and G. Calvagno, "Color image demosaicking: An overview," Signal Process.: Image Commun., vol. 26, no. 8-9, pp. 518-533, 2011.

[6] R. Ramanath, W. E. Snyder, G. L. Bilbro, and W. A. Sander, "Demosaicking methods for Bayer color arrays," J. Electron. Imaging, vol. 11, no. 3, pp. 306-315, Jul. 2002.

[7] K. Hirakawa and T. W. Parks, "Adaptive homogeneity-directed demosaicing algorithm," IEEE Trans. Image Process., vol. 14, no. 3, pp. 360-369, Mar. 2005.

[8] M. Riechert, "rawpy: RAW image processing for Python," GitHub, 2024. [Online]. Available: https://github.com/letmaik/rawpy. [Accessed: Jun. 2026]

[9] S. van der Walt et al., "scikit-image: image processing in Python," PeerJ, vol. 2, e453, 2014. [Online]. Available: https://scikit-image.org. [Accessed: Jun. 2026]

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Trans. Image Process., vol. 13, no. 4, pp. 600-612, Apr. 2004.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Jonathan Lewie , 13525136