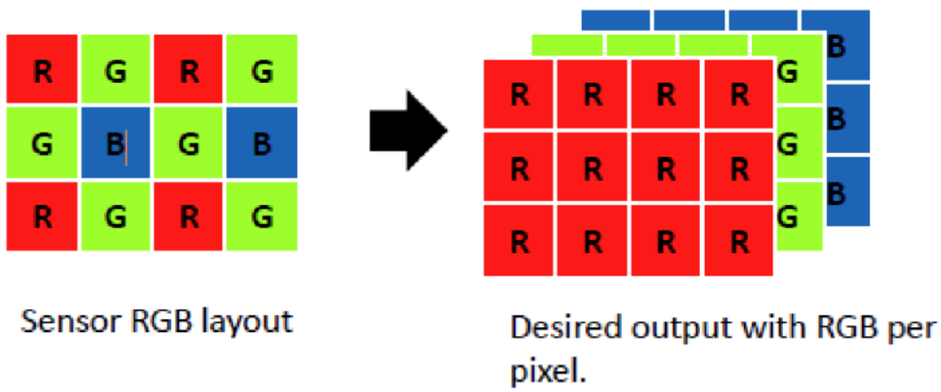
Homework Assignment 1
CFA Demosaicing

Digital Image Processing R14942096 林祐群

I. Implementation Details Description

All methods assume the RGGB Bayer pattern and process grayscale input images to produce full-color RGB outputs



(Source: Class Material)

i.) p1.py (Simple Interpolation):

In p1 implementation, I use basic neighborhood averaging for missing pixel interpolation. Firstly, apply **Bayer Pattern Assignment**, which directly assigns known R, G, B values to their respective positions. Then, **Green Channel Interpolation** is performed:

At R positions: Average of available horizontal/vertical G neighbors

At B positions: Average of available horizontal/vertical G neighbors

Lastly, **R/B Channel Interpolation** is followed:

At G positions: Horizontal averaging of R/B neighbors

Cross-channel interpolation: Diagonal averaging for R at B positions and B at R positions

ii.) p2.py (Simple Edge-Aware Interpolation):

P2 extends P1 with adaptive interpolation based on local edge detection, where edge detection is conducted by calculateing vertical and horizontal differences between neighbor pairs. We have to assess the absolute difference properties with predefined priority. That is:

Step 1. If both differences < threshold: Use all available neighbors

Step 2. If vertical difference > threshold: Use horizontal neighbors only

Step 3. If horizontal difference > threshold: Use vertical neighbors only

We apply this technique to both Green channel and R/B channel. Note that I use **cross-interpolation** for R/B channel, since their neighbors appear not in horizontal or vertical places.

iii.) p3.py (Advance Interpolation):

Core Algorithm of P3 is based on stochastic policy-based interpolation system [1]. In short, the approach combines directional edge detection, probabilistic weighting, and neighborhood-based interpolation to minimize artifacts in the final color image. However, instead of 12 direction edge indicators as considered in the paper, I use four-direction edge indicators:

```
# Four-directional edge indicators
E_N = abs(CFA[pi-1, pj] - CFA[pi+1, pj]) + abs(CFA[pi, pj] - CFA[pi-2, pj])
E_S = abs(CFA[pi+1, pj] - CFA[pi-1, pj]) + abs(CFA[pi, pj] - CFA[pi+2, pj])
E_E = abs(CFA[pi, pj+1] - CFA[pi, pj-1]) + abs(CFA[pi, pj] - CFA[pi, pj+2])
E_W = abs(CFA[pi, pj-1] - CFA[pi, pj+1]) + abs(CFA[pi, pj] - CFA[pi, pj-2])
```

And follow the paper with stochastic weight calculation:

```
# Adaptive sigma calculation
mu = np.mean(indicators)
sigma_d = mu * (math.pi / 2)**0.5

# Complementary error function weighting
weights = [math.erfc(ind / (sigma_d * math.sqrt(2))) for ind in indicators]
```

The usage of edge indicators aims at detecting directions with **minimal intensity variation** (hence, likely less color discontinuity) and applies stochastic weights based on those indicators, encouraging interpolation along edges rather than across them, which helps reduce color artifacts. The high-level idea is that if one direction has strong edges, its weight is reduced; interpolation prefers smoother directions. Finally, we leverage local patterns and masks to recognize known R, G, and B pixel positions and then fills in unknown values using weighted, directional neighbor values.

In the code, I treat interpolation of R/G/B differently in accord to the paper. This can be summarized as the following:

1. Interpolate G at R/B locations:
    - Compute four direction edge indicators and estimates.
    - Calculate stochastic weights.
    - Merge directional estimates using weights for edge-preserving interpolation.

2. Interpolate R/B at G locations:
    - Use local averages (horizontal/vertical) of color differences.
    - Restore original CFA values where available.
3. Interpolate R/B at B/R locations:
    - Use diagonal convolution of color differences.
    - Restore measured CFA values.

Then we can construct RGB images by stacking channels and save the output results.

II. Results Discussion

The results of the three techniques for CFA Demosaicing can be roughly summarized as below:

Method	**Complexity**	**PSNR (dB)**	**Edge Handling**
P1	Low	16.75	Basic
P2	Medium	16.68-16.73	Adaptive
P3	High	31.14	Advanced

We can discover significant improvement for p3 over p1 and p2, where the consideration of directional edge indicators and stochastic weighting are of importance. An interesting discovery is that p2 is not greater than p1 with performance gap in the aspect of PSNR, despite of the adaptive edge handling technique. This may indicate that the given 5 images might not be effective for comparing the two techniques (p1 and p2).

The reason the PSNR scores for p2 is a bounded range is because I've tried several thresholds for edge-aware interpolation. I utilize 3, 5, 10, 20, 30, 50, 70, 100 to fine-tune a near-optimal threshold. However, it turns out that the given dataset is insensitive to the changes of threshold parameter. Though in visually speaking, the raw images contain several sharp intensity differences, such as fence, the overall impact of "simple" edge-aware is minor. Even with parameter tuning, the performance remains on par with p1.

We can discover the phenomena in Fig.1 clearly, where p3 significantly outperform any other simple implementations with parameter tuning. We can also notice the comparison of p2 with different threshold parameter, which turns out to be insensitive in this case.

III. Analysis and Ablation Study

Beside the discussion above, I also analyze standard deviation of average PSNR score of each technique and average PSNR by image. From Fig.2, you can first clearly observe the significant performance improvement of p3 over others, where it has the highest average PSNR score over all the implemented methods.

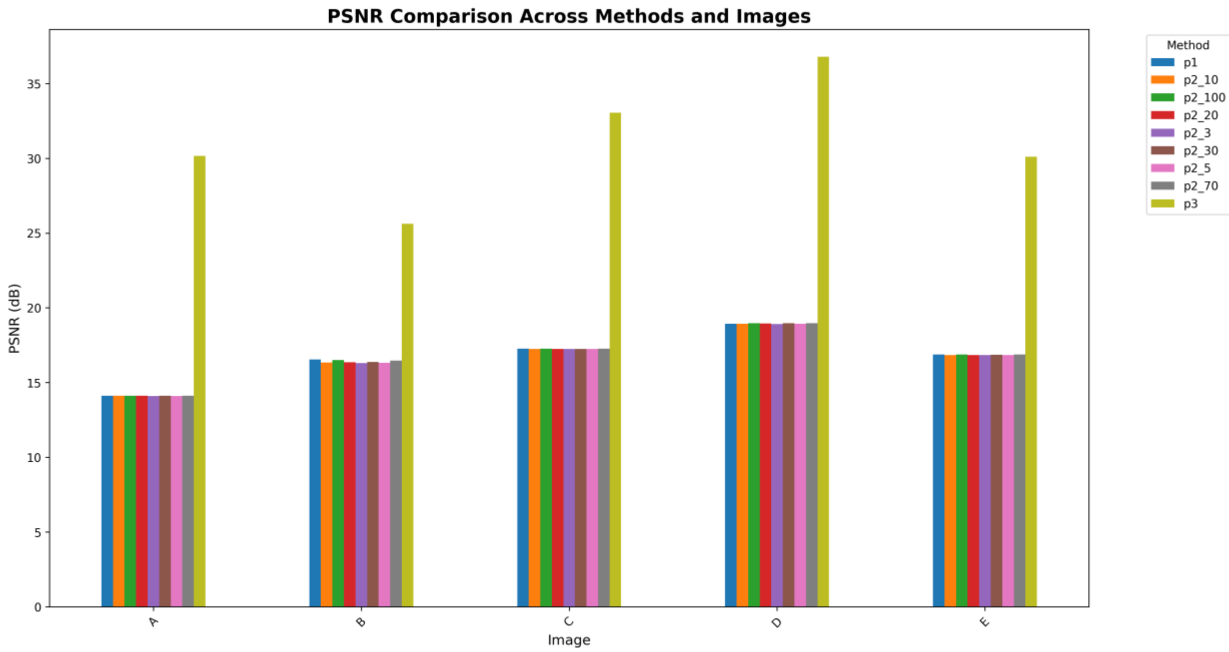


Fig. 1 PSNR Comparison Across Methods and Images

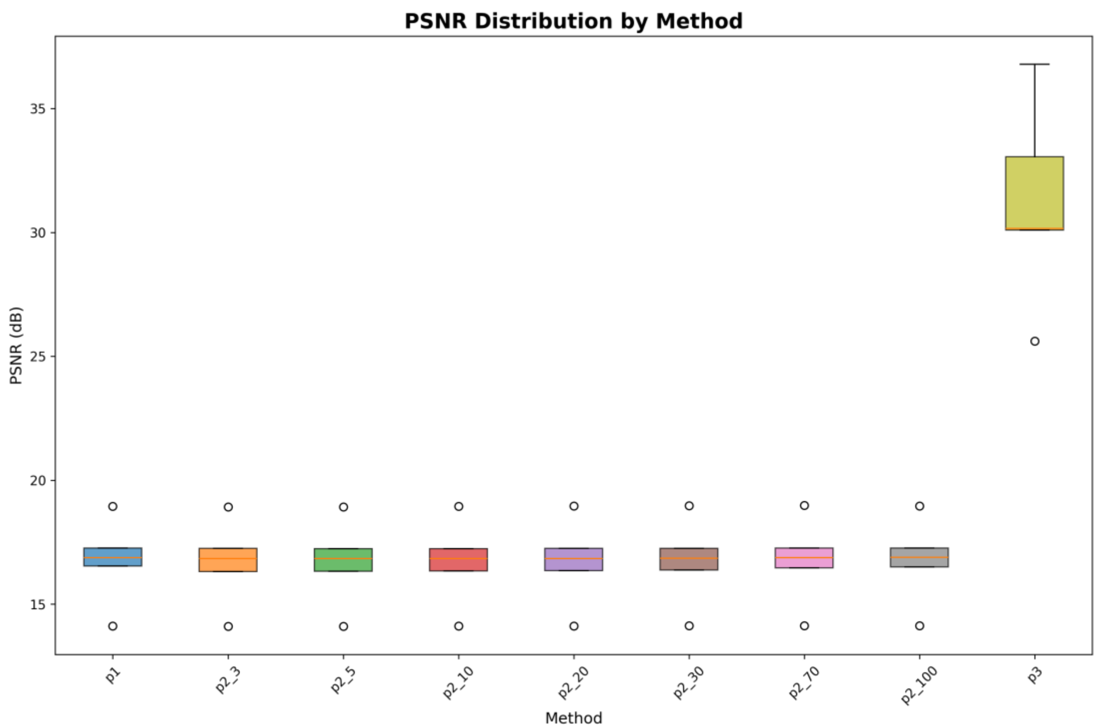


Fig. 2 PSNR Distribution by Method

It's also shown in Fig.2 that the PSNR score of p3 has higher standard deviation compared with other techniques. This can be noticed from the performance drop of p3 in image B, which is 25.62dB; this is the only image whose PSNR score is below

30dB using p3. (Fig. 1)

In addition to method-wise comparison, we can as well assess image-wise difference. By ranking the average PSNR score of all methods from high to low, left to right, you can statistically understand the adaptiveness of our three methods to the given data. (Fig. 3)

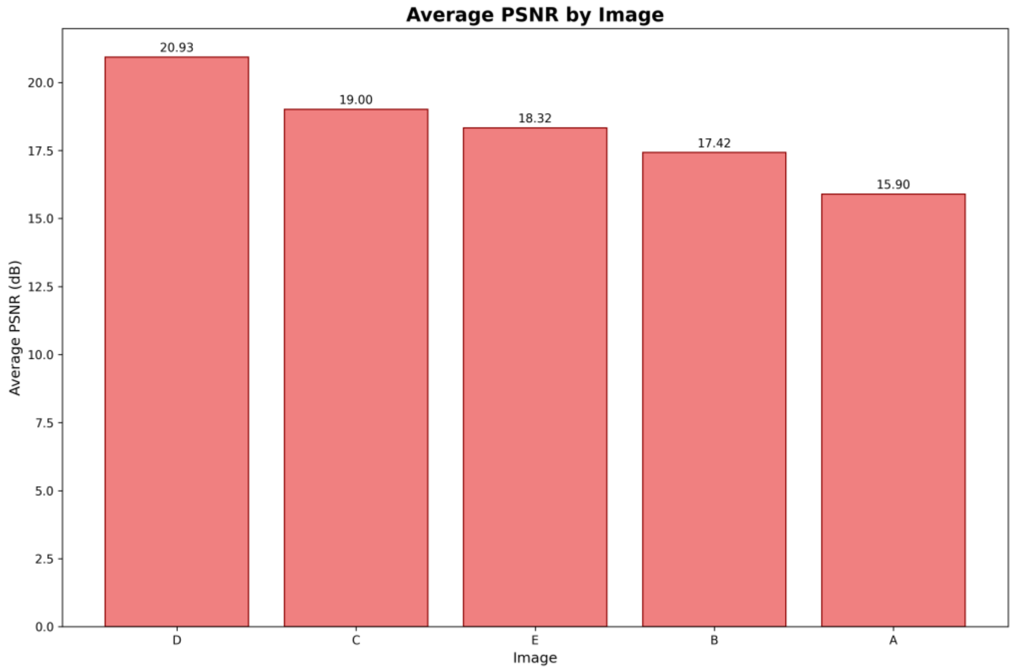


Fig. 3 Average PSNR by Image

Fig. 4 Ground Truth Image A (Left), and Ground Truth Image D (Right)

From Fig. 4, we can further perceive the differences between the extreme two images according to analysis provided in Fig. 3. You can discover intense high-frequency changes in the region of fence, which is somehow more difficult than the lower frequency changes in image D. This may explain the ranking in Fig. 3.

Compared with CFA demosaiced images using p3 in Fig. 5 with respect to ground truth images in Fig. 4, you can see some artifacts, especially the region marked in orange in the image A in Fig. 5.

Fig. 5 Constructed Image A (Left), and Image D (Right) Using p3

As for the analysis on strengths and weakness, we can elaborate as below:

Methods	Strengths	Weakness
P1	- Easy Implementation	- Low PSNR
P2	- Edge Awareness (not significant in this homework)	- Low PSNR
P3	- High PSNR - Edge Awareness - Color maintenance (Fig. 6)	- Complexity (while affordable)

Fig. 6 Color Difference between p1 (Left), p2 threshold=50 (Middle), and p3 (Right)

IV. Code Usage

Requirements:

- Python 3.6+
- OpenCV (cv2)
- NumPy
- Matplotlib
- Pandas

Install Dependencies:

```
pip3 install opencv-python numpy matplotlib pandas
```

Run Individual Methods:

```
# Run P1 method:
    python3 p1.py --input_image images/raw_image/A.tiff
# Run P2 method (specify threshold)
    python3 p2.py --input_image images/raw_image/A.tiff --threshold 10
# Run P3 method
    python3 p3.py --input_image images/raw_image/A.tiff
```

V. References

[1] H. -A. Chang and H. H. Chen, "Stochastic Color Interpolation for Digital Cameras," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 8, pp. 964-973, Aug. 2007, doi: 10.1109/TCSVT.2007.897471.