École nationale supérieure de physique, électronique, matériaux - Grenoble

Project
Image Analysis

Realized by :

Mouna El Amrani

Academic year 2023/2024

Chapter 1

Image Reconstruction

1.1 Introduction

This report aims not to present findings and conclusions about the demosaicking techniques used in this project, as well as perspectives for future improvement in this dynamic and technically demanding field.

1.2 Problem Statement

Demosaicing, also known as color reconstruction is a process used in digital image processing to reconstruct a full-color image from incomplete color samples obtained by an image sensor overlaid with a Color Filter Array (CFA). Common in most digital cameras, CFAs allow each sensor pixel to capture only one of the three primary colors: red, green, or blue. However, this leads to a mosaic of color information where each pixel contains data for only one color channel. Demosaicing algorithms then interpolate this data to estimate the two missing color channels at each pixel. This is achieved by using various techniques that analyze the color information of neighboring pixels and the goal is to create a full-color image that is as close as possible to the original scene. The Bayer pattern is a widely used CFA in digital imaging particularly in digital camera sensors. It arranges color filters on a square grid of photosensors in a repeating pattern of 2x2 cells.

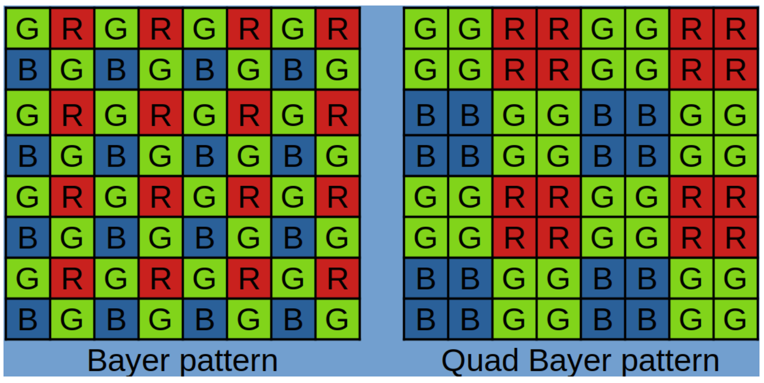

Figure 1.1: Bayer Pattern Layout

In cameras using Color Filter Arrays (CFA), each pixel on the sensor is covered by a filter that only captures light of a specific color red, green or blue. Due to this filtering each pixel gathers information for only one of the primary colors leading to the creation of a raw image that resembles a grayscale picture. However, this grayscale image is unique instead of representing overall brightness each pixel's intensity corresponds to the amount of RGB light it receives. The raw data requires a process called demosaicing to reconstruct a full-color image where each pixel has combined information of all three primary colors.

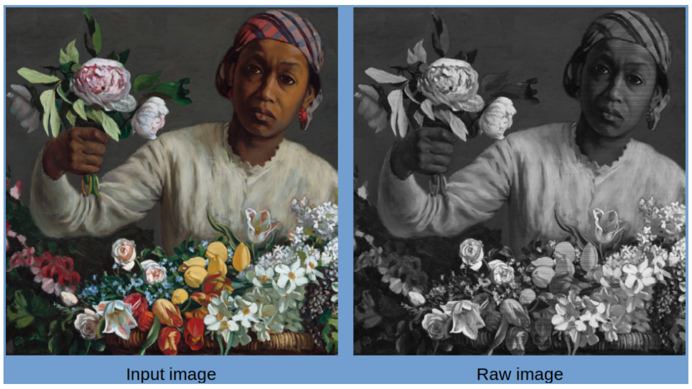

Figure 1.2: Input VS Raw image

1.3 Proposed Solutions

In this part we are going to see different methods used as demosaicking techniques to the image reconstruction: The methods represents a sophisticated approach to demosaicking for Bayer and Quad Bayer CFA patterns. These methods employes adaptive gradient-based techniques focusing on accurately interpolating each color channel (red, green, blue) by analyzing local image characteristics. The process starts with establishing an initial framework for the mosaicked image followed by a series of steps to interpolate the missing colors. Key techniques include finding nearest neighbors, calculating directional gradients and computing adaptive weights. So the functions i used in this process are :

`find_Knearest_neighbors`

This function finds the neighbors of a given pixel in a specific channel (red, green, or blue) within an image. For a pixel at position (i, j) in an image of size $N \times M$, it retrieves the values of the 3x3 neighborhood around that pixel. The operation wraps around the image edges using modulo arithmetic $(i + \text{di})\%N$ and $(j + \text{dj})\%M$.

`calculate_directional_gradients`

Calculates the directional derivatives (gradients) of a pixel, which are useful in determining the direction of highest color variation.
Given the neighbors $P1, P2, \ldots, P9$ (arranged in a 3x3 grid), it calculates the gradients as follows:
Horizontal derivative, $Dx = \frac{P4-P6}{2}$
Vertical derivative, $Dy = \frac{P2-P8}{2}$
Diagonal derivatives, $Dxd = \frac{P3-P7}{2\sqrt{2}}$ and $Dyd = \frac{P1-P9}{2\sqrt{2}}$

`calculate_adaptive_weights`

Calculates adaptive weights based on the gradients. These weights are used to interpolate the color values at a pixel, taking into account the variations in different directions.
The weights E are calculated using the inverse square root of the sum of squares of the corresponding directional derivatives. For example, for neighbors n and their derivatives dd:
$E = \frac{1}{\sqrt{1+Dxd^2+dd[2]^2}}$ for diagonals, and similarly for other directions.

`interpolate_pixel`

Interpolates the value of a pixel based on its neighbors and the computed weights. This is used for interpolating the green channel in the Bayer pattern.
The interpolated value $I5$ of the pixel is calculated as a weighted average of its neighbors:
$I5 = \frac{E2\times P2+E4\times P4+E6\times P6+E8\times P8}{E2+E4+E6+E8}$

`interpolate_RedBlue`

Specifically interpolates the red or blue channel pixel, taking into account the green channel as a reference.
This function uses the ratios of the neighboring red/blue values to their corresponding green values to interpolate

the missing red/blue value.
$I5 = G5 \times \frac{(E1\times\frac{P1}{G1})+(E3\times\frac{P3}{G3})+(E7\times\frac{P7}{G7})+(E9\times\frac{P9}{G9})}{E1+E3+E7+E9}$
The `run_reconstruction` function performs demosaicking on a mosaicked image using various sub-functions. It takes a mosaicked image `y` and a CFA pattern name `cfa` as inputs and returns the demosaicked image.

Reconstruction Process

1. Definition constants and create an operator `op` for the CFA pattern.
2. Application of the adjoint operation of `op` to `y` to initialize the reconstruction process.
3. Interpolation of each channel (red, green, blue) of the image separately.
4. For each pixel in the channel, checking if it needs interpolation based on the CFA pattern and its position.
5. Finding the nearest neighbors of the pixel and calculate the directional gradients and adaptive weights.
6. Interpolating the pixel's value using `interpolate_RedBlue` for red and blue channels in the first pass, and `interpolate_pixel` for all channels in the subsequent passes.
7. Cliping the values in the image to be within the valid range [0, 1].

1.4 Results

In this section we are going to visualize the results of the methods explained before. We first are going to see the original image and then compare it to the reconstructed one:

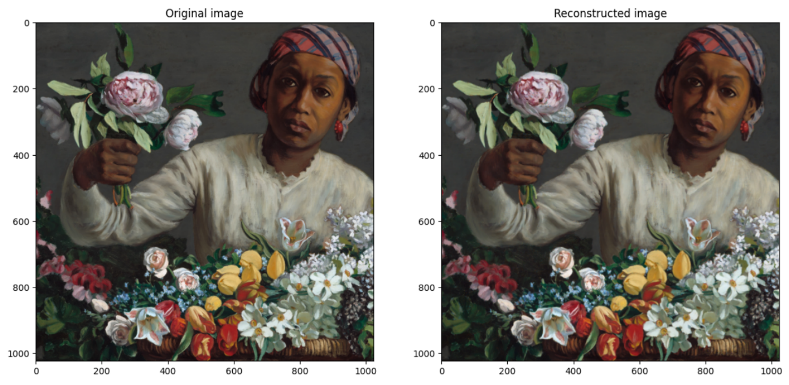

Figure 1.3: First reconstructed image

PSNR: 35.55 SSIM: 0.9574

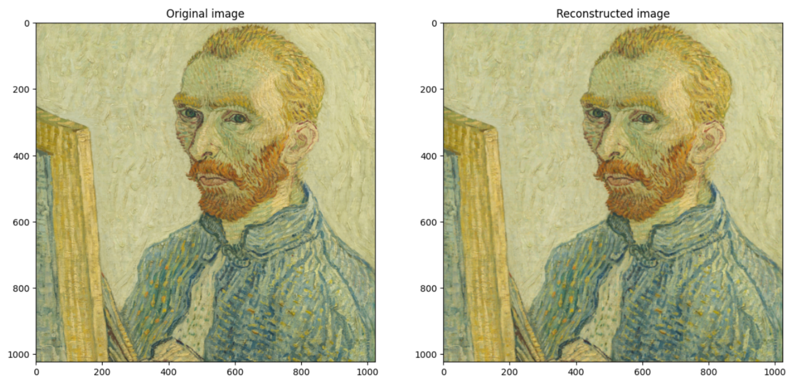

Figure 1.4: Second reconstructed image

PSNR: 31.44 SSIM: 0.8602

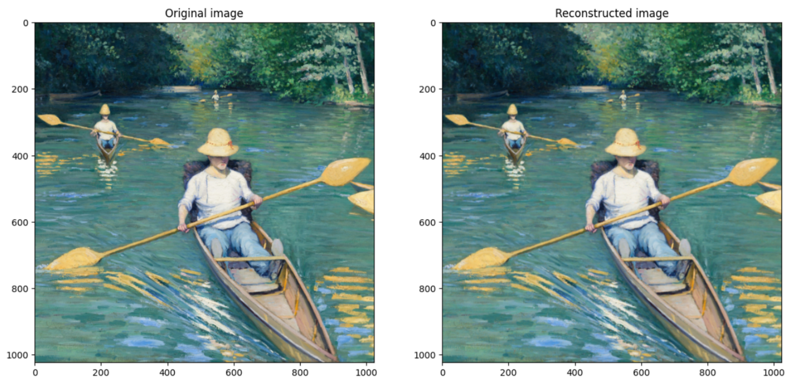

Figure 1.5: Third reconstructed image

PSNR: 32.79 SSIM: 0.9046

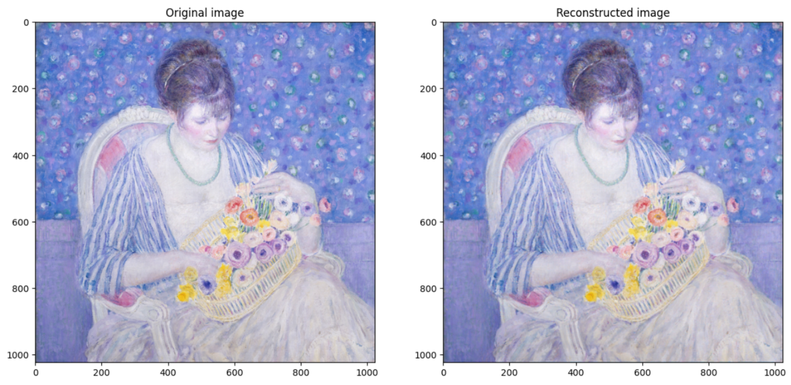

Figure 1.6: Fourth reconstructed image

PSNR: 30.95 SSIM: 0.8352

1.5 Conclusion

In conclusion, the methods i used in this project are mainly :

- **Adaptive Gradient-Based Interpolation:** I think this can be considered as a good choice as it adapts to local image features helping to preserve details and reduce color artifacts that are common issues in demosaicking.
- **Directional Gradients and Adaptive Weights:** Calculating these metrics aids in making informed decisions about how to interpolate each pixel which is crucial for maintaining image sharpness and reducing blurring.
- **Two-Pass Interpolation for Red and Blue Channels:** This strategy is particularly effective given the sparser distribution of red and blue pixels in Bayer patterns ensuring more accurate color reconstruction.

These methods has given good results in reconstructing images that looks like the original ones, we can try to improve the results by improving Noise Sensitivity or exploring ways to optimize the calculations such as using more efficient data structures.