Image analysis course SICOM 3A – Project

Table des matières

Table des figures

I – Introduction

The project involves color reconstruction of images captured by cameras using Color Filter Array (CFA) technology. Digital cameras often use this technology to capture color images. CFA consists of red, green and blue filters arranged in a periodic pattern on the image sensor, filtering out incident light before it is captured by the sensor. However, this technique means that each pixel of the sensor captures only one color (red, green or blue), resulting in a raw acquisition in the form of a grayscale image. The aim of this project is to carry out demosaicking, i.e. to recover all the missing colors for each pixel, in order to restore the complete RGB image from the grayscale image obtained by the sensor. All this work was carried out with the help of this research paper from the magazine « Signal Processing Magazine January 2005 » .

Here are the two-color filters we will use:

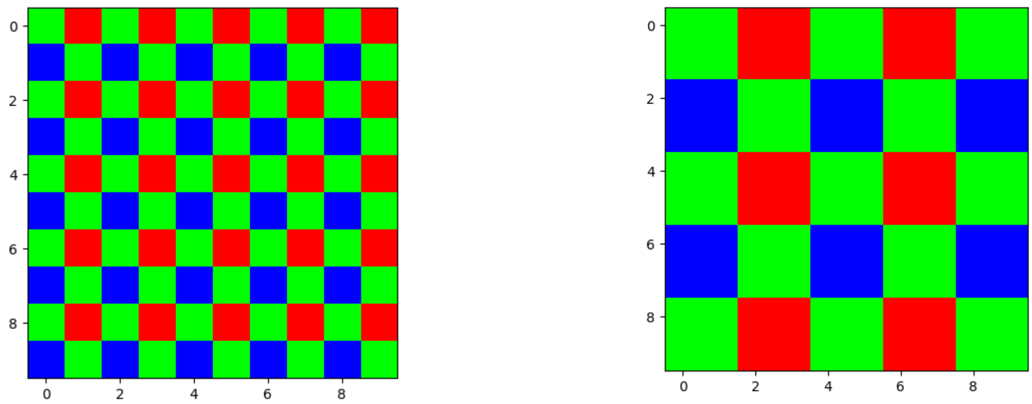

Figure 1 – Bayer and Quad Bayer filters

II- Image reconstruction solution: Constant-difference-based interpolation

To understand the benefits of this method, we need to understand the actual structure of the initial image for each channel after the adjoining operation via the CFA filter.

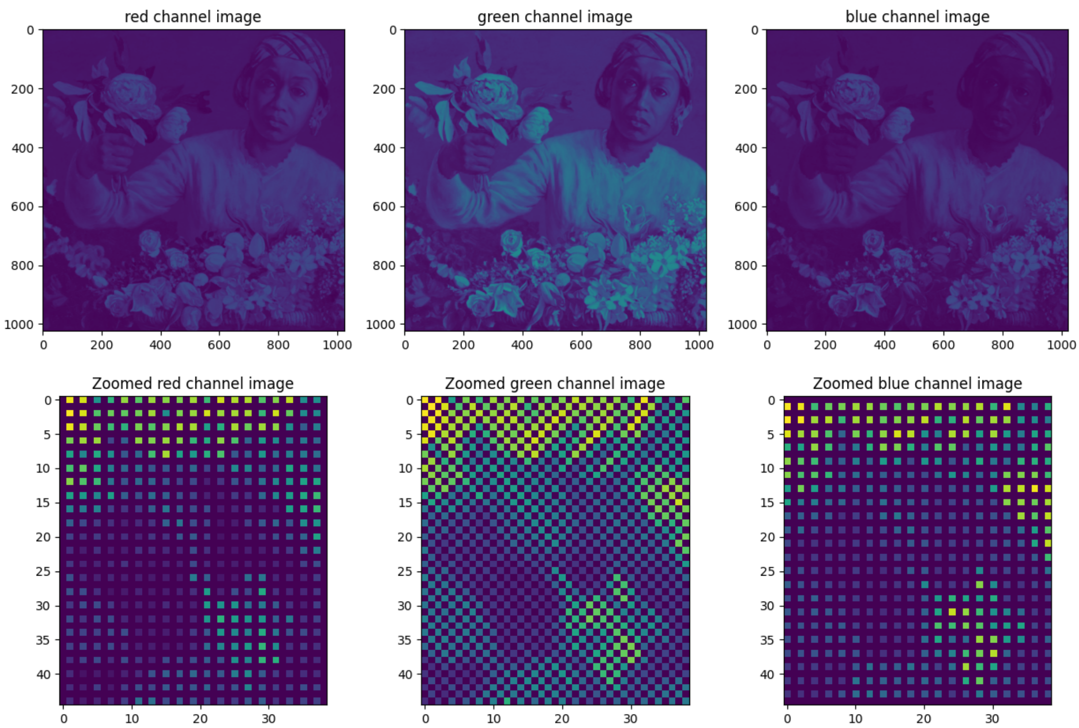

Figure 2 - Visualization of each image channel z

Looking at this figure, we immediately see that the channel with the most information is the green channel. This is consistent with the fact that the CFA mask is made up of 50% green pixels, 25% red and blue, as the eye is more sensitive to the color green. The aim is first to reconstruct the green channel, then to reconstruct the other two channels.

II.1 – Reconstruction du canal vert : La méthode Hamilton et Adams

For this first step I followed the interpolation method developed in this research paper [here], here's a diagram that explains this:

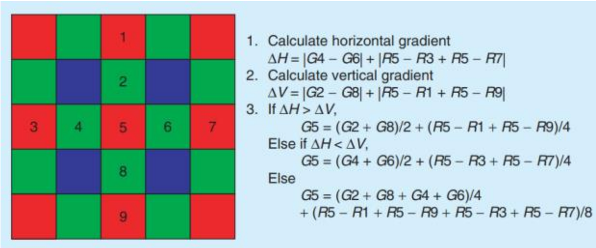

Figure 3 - The Hamilton and Adams method developed in "Demosaicking: Color Filter Array Interpolation"

I implemented this function, taking care to apply it only to pixels that weren't green, to avoid overwriting the information.

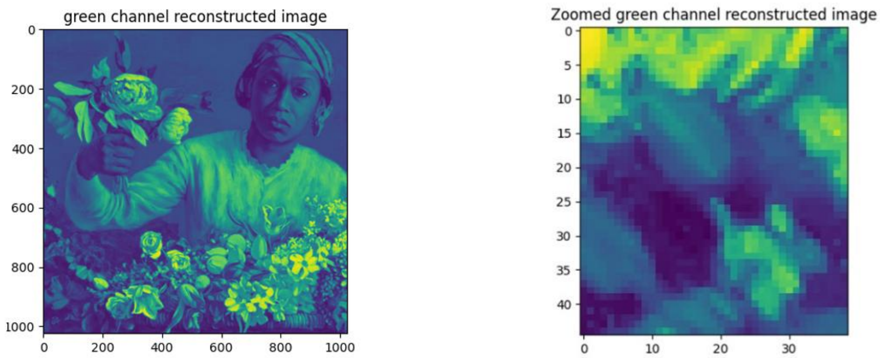

Figure 4 - Visualization of the reconstructed green channel

We can now compare the result with the green channel in figure 2, and see that the reconstruction is satisfactory. It's important to have a good result, as we're now going to use this green channel to reconstruct the other two channels, blue and red.

II.2 – Blue and red channel reconstruction: Constant-difference-based interpolation

I've implemented the difference channel interpolation process presented in the source [here]. I first calculate the difference between the mosaic channel and the interpolated green channel, which represents the color information missing from the mosaic channel. This difference is then interpolated with a simple kernel (3x3 matrix with equal values that sum to 1), which smoothed the difference over neighboring pixels. The interpolated difference is added to the interpolated green channel. The final phase consists of filtering the result again with a kernel that simulates the response of the Bayer filter (ker_bayer_red_blue), giving the final image for the red or blue channel. The following diagram summarizes these explanations:

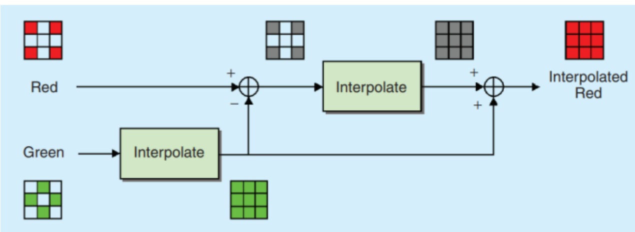

Figure 5 - Constant difference-based interpolation presented in " Demosaicking: Color Filter Array Interpolation"

Here is the result of these steps to reconstruct the blue and red channel:

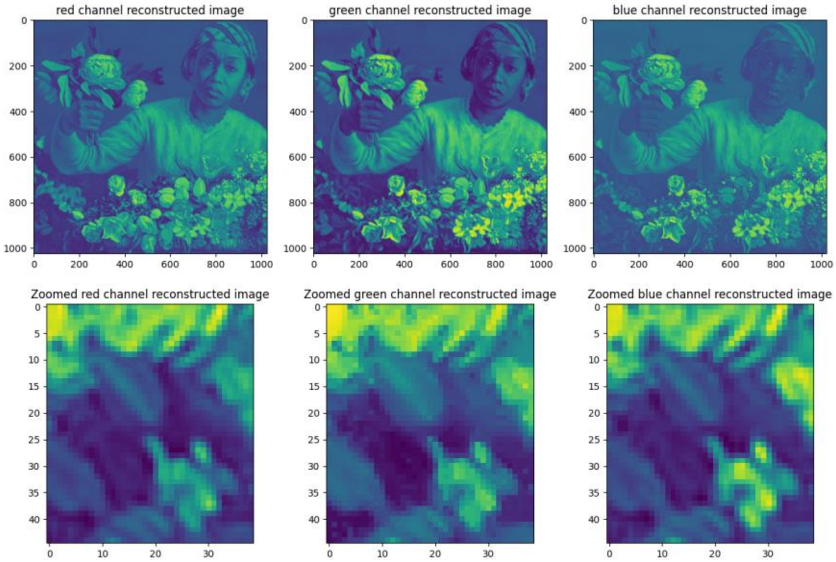

Figure 6 - Visualization of red, green and blue channel reconstruction

We can see that the reconstruction is satisfactory compared to figure 2, and we can analyze the maximums of each channel to see if one color predominates.

	Red channel	Green channel	Blue channel
Minimum	-0.10	0	-0.20
Maximum	1.16	0.94	1.12
Mean	0.35	0.33	0.30

Figure 7 - Max, Min and average for each channel

The negative values and values greater than 1 for the red and blue channels are caused by the difference with the green channel. To solve this problem, you'd have to make a .clip(0,1) to limit all values so that they're all displayable. Concerning the average values, red predominates, but this doesn't necessarily mean that it's a problem, it just means that there's redder on average in the image.

III- Results analysis

We're going to apply these reconstruction methods to the four images below. I haven't shown the results of the processing because you can't really see the difference, so it's more interesting to analyze the PSNR and SSIM for each filter.

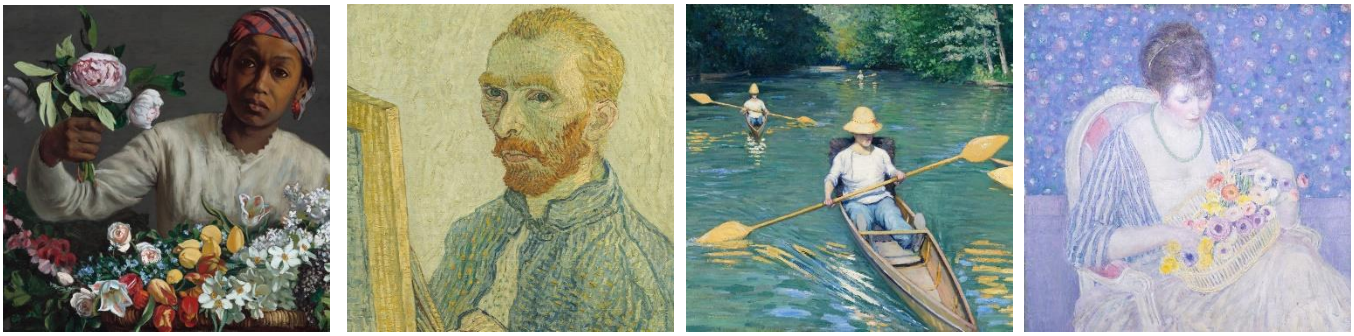

Figure 8 - Image 1, 2, 3 et 4

	Image 1	Image 2	Image 3	Image 4
PSNR (dB)	33.54	28.73	30.79	28.81
SSIM	0.9509	0.8720	0.9031	0.8378

Figure 9- Results: PSNR and SSIM for a Bayer filter

	Image 1	Image 2	Image 3	Image 4
PSNR (dB)	29.82	26.29	27.70	26.74
SSIM	0.8820	0.6775	0.7713	0.6717

Figure 10 - Results: PSNR and SSIM for a Quad Bayer filter

The results depend on each image. Image 1 gives the best results, of the order of 30 dB, which is satisfactory. This is consistent with the fact that, to process the quad bayer filter, I under-sample to return to a bayer filter and then over-sample to return to the base image. There is therefore 4 times less information, so it's consistent.

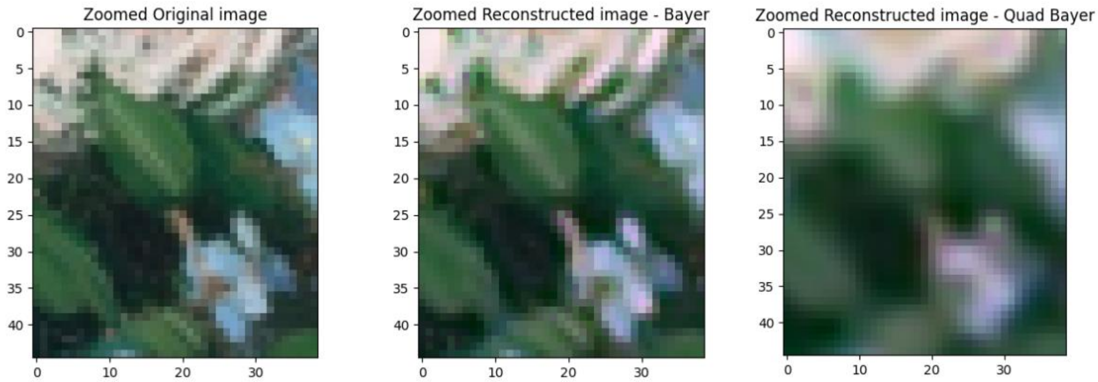

Figure 11 - Comparison of Bayer and Quad Bayer results with the original image

An analysis of this last figure shows that using the bayer filter produces a result similar to the original image, but with less detail and contrast. Last but not least, the use of the quad bayer filter results in blurring due to under sampling.

IV- Conclusion: Milestone and improvements

To conclude this project involving the demosaicking of images captured via a color matrix filter (CFA), the method chosen for reconstruction was based on difference-based constant interpolation using the Hamilton-Adams method for the green channel and interpolation for the red and blue channels. The results obtained are of the order of 30 dB, which is satisfactory, and the SSIM is close to 1, so the reconstructed images are very close to the original image. To improve the results, more sophisticated interpolation methods should be tried, such as bicubic interpolation, or approaches based on machine learning algorithms.

This project was interesting because it enabled me to read and understand a research paper, and to implement these reconstruction methods. Although the results are not as good as the solution initially proposed, I feel that I have understood a lot by implementing this reconstruction myself, and that it has helped to clarify the theoretical aspects of the course.

V – Annexes

V.1 – Details of each step in the demosaicking process

To make sure I've understood the subject properly, I'm going to detail the steps in the demosaicking process below, using the example of a pixel at each stage. Let's imagine a pixel in the original image and follow it through the process:

Step 1 : Input Image

Let's imagine that our pixel of interest in the input image has the following color values (in a full RGB image):

Red (R): 120; Green (G): 80; Blue (B): 200

Step 2: Applying CFA mask (Direct function)

The CFA mask decides which color component is captured for each pixel. The most common pattern, Bayer, has a distribution where half the pixels are green (since the human eye is more sensitive to this color), a quarter red and a quarter blue. Let's assume that our pixel is under a blue CFA mask filter. Applying the CFA mask will give:

Red (R): 0; Green (G): 0; Blue (B): 200

Step 3: Mosaic or Raw Image (After Direct Application)

Our pixel in the mosaic image will have a value of 200 for blue, and 0 for the other colors, represented in a grayscale image as a pixel with intensity 200 (if blue is considered). From this, the aim is to recover all the missing colors for each pixel. In this way, we'll recover the complete RGB image.

Step 4: Application of Assistant Operation (Assistant Function)

The adjoint operation is a kind of inverse process that attempts to remap the color values captured by the CFA sensor back to their original positions, giving:

Red (R): 0; Green (G): 0; Blue (B): 200

Step 5: Naïve interpolation (naive interpolation function - proposed solution)

Interpolation estimates the missing values of the red and green channels using information from neighboring pixels. Let's assume that interpolation gives the following values for our pixel:

Green (R): estimated at 110 (based on neighboring red pixels)

Green (G): estimated at 90 (based on neighboring green pixels)

Blue (B): 200 (unchanged as this is the captured value)

In this example, the pixel will appear slightly less red and green than the initial image. Using this reconstruction method, we obtain a PSNR (Peak Signal to Noise Ratio) of 34.63dB, which is already satisfactory. A PSNR greater than 30dB indicates that the reconstruction is of high quality. The SSIM (Strucutral Similarity Index) is 0.95, which is very close to 1, suggesting that the structure, texture and luminance of the reconstructed image are similar to those of the original.