

---

Université Grenoble Alpes  
INP PHELMA  
Physique, Électronique, Matériaux  
Signal Communication Multimédia

Université Grenoble Alpes  
INP ENSE3  
École Nationale Supérieure de l'Eau,  
de l'Énergie et de l'Environnement  
Signal Communication Multimédia

# Image Analysis

## Rapport

31/01/2024

par

SADONES Tomé

*Encadrants universitaires :*  
*Mauro Dalla Mura*  
*Matthieu Muller*

# 1 Problem statement

Nowadays, most cameras use a so-called **CFA (Color Filters Array)** which will filter red, green and blue wavelengths over the sensor. This means that each pixel of a sensor will receive a light intensity, relative to the energy contained in the given wavelength.

Most of the time, the **CFA patterns** are periodic and well-known, allowing for the camera software (or any given image software) to reconstruct an RGB image, given only the raw acquisition.

The more used one is the **Bayern patter**, followed by the **Quad Bayer pattern**. They are depicted in the following figure.

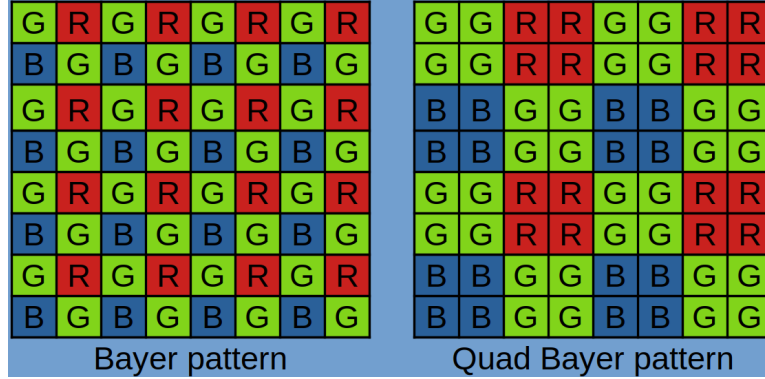


Figure 1: Bayer patterns

It is very noticable that the green channel is represented twice as much as the red or the blue one. This is because the human eye is more sensitive to green.

By applying such a filter over a sensor, it means that acquired images will be gray-scaled, and that each pixel corresponds to the intensity of only one color at a given position. The impact of such a process can be seen in the following figure.

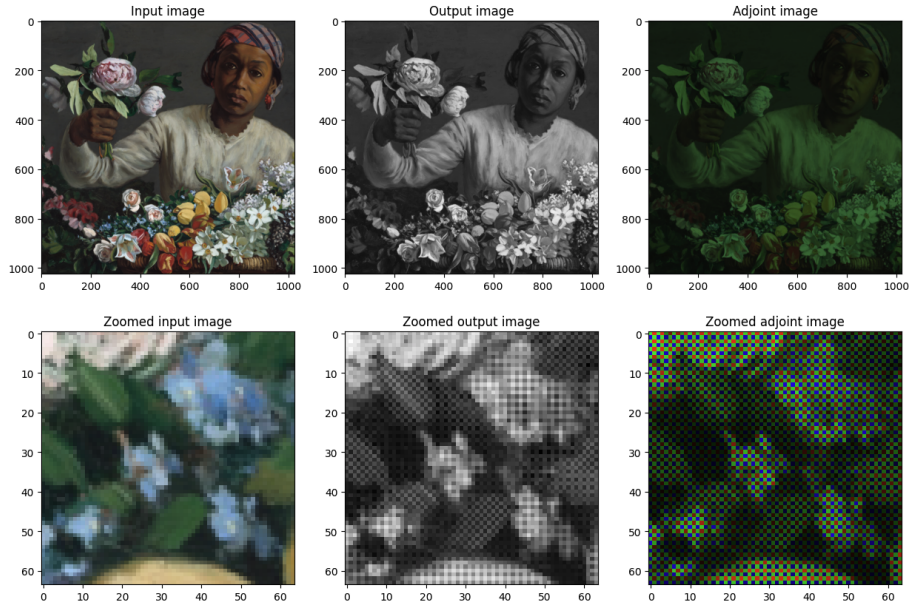


Figure 2: Effect of CFA

One can easily see that the pixels intensity are correlated to only one color, and we could cast them to their relative colors. This is done to the right-hand side images. This is clearly not useable, given the periodicity of the patterns and the amount of green.

We are looking for a solution to *reconstruct* the original image, or at least to be as close as possible to the real image.

It is simple to understand that it is an ill problem, since each channel is downsampled. Information has been lost, and there is no magical way to recover it. However, it is possible to make assumptions and hypothesis to recover as much as possible.

By taking a signal approach to the problem, we can understand that the signal, if we consider each channel separately, has to be upsampled. In order to do such a thing, we have to interpolate for each new value considered. An approach would be to apply a low-pass filter, to suppress the high frequencies of the 0s. Another approach would be to compute the Fourier coefficients of each channels (without the zeros) and use them to reconstruct the image. This method does not allow us to recreate very high frequencies, such as sharp edges.

## 2 Image reconstruction using naive interpolation

### 2.1 Bayer pattern

#### 2.1.1 Basic bi-Linear Demosaicking using 3x3 kernel as convolution filter method

A very simple and naive way of solving this problem is by considering each channel independantly. Then, it is simply a game of finding the best interpolation matrix for each channel, and use it to reconstruct the image. The Basic bi-Linear Demosaicking using 3x3 kernel as convolution filter method is already implemented, and provides one of the easiest way to face down the problem. This method simply provides the 2 following kernels :

$$K_G = \frac{1}{4} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad K_C = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Please note that  $C \in R, B$  corresponds to the red or the blue channel. The kernels described above simply average the closest pixels.

- For the green, there is either a green pixel at the center, then we do not change it, or 4 pixels around, then we average.
- For red and blue, either there is one at the center, either there are 4 at the corners, either they are only two next to the center. Either way, we simply average and the sum always goes to 1.

This method is very effective, and very used. The results on the first image are :

$$\begin{array}{l} \text{PSNR} \parallel 34.63 \\ \text{SSIM} \parallel 0.9502 \end{array}$$

Table 1: Bi-linear interpolation

This method is presented as the starting point, and will be very hard to defeat.

#### 2.1.2 Cubic Demosaicking with 7x7 kernel convolution

Then, we can try to improve it by taking bigger kernels, which may give us better interpolations results. This time, the kernels are the following :

$$K_C = \frac{1}{256} \begin{bmatrix} 1 & 0 & -9 & 16 & -9 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -9 & 0 & 81 & 144 & 81 & 0 & -9 \\ -16 & 0 & 144 & 256 & 144 & 0 & -16 \\ -9 & 0 & 81 & 144 & 81 & 0 & -9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -9 & 16 & -9 & 0 & 1 \end{bmatrix}, \quad K_G = \frac{1}{256} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -9 & 0 & -9 & 0 & 0 \\ 0 & -9 & 0 & -81 & 0 & -9 & 0 \\ 1 & 0 & 81 & 256 & 81 & 0 & 1 \\ 0 & -9 & 0 & -81 & 0 & -9 & 0 \\ 0 & 0 & -9 & 0 & -9 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

This often is presented as the best linear method, and indeed works better. Actually, it works better on softer parts of the image, and is slightly worse on high frequency zones, since it tries to average on more pixels (low-pass filter).

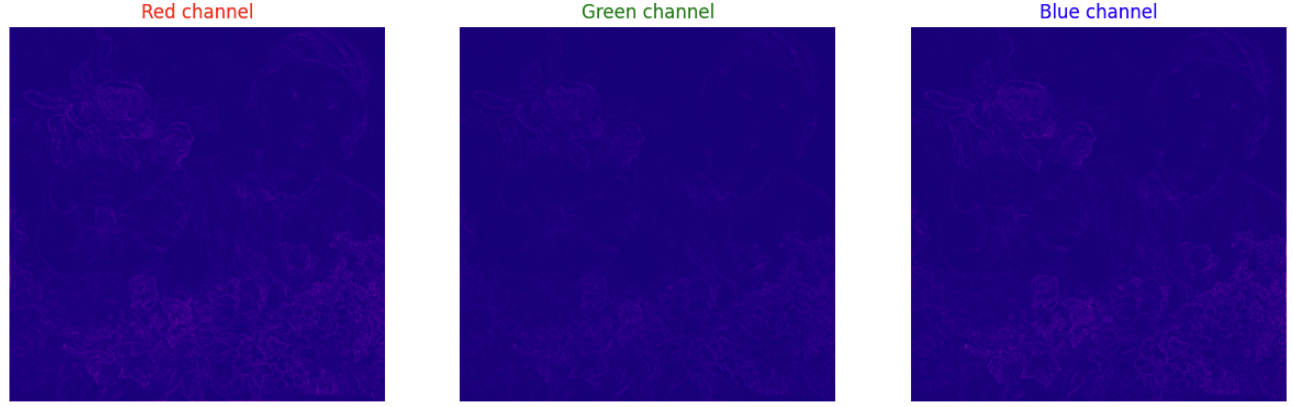


Figure 3: 7x7 kernel interpolation errors (log-scale)

As anticipated, the green channel is more precise since it is sampled at a higher frequency. In the low-frequency zones, there are only very few errors. It mostly is the edges that are hard to reconstruct and get right. This method slightly improves the accuracy of the results, as depicted in the following table.

PSNR	35.07
SSIM	0.9522

Table 2: Cubic Demosaicking

### 3 Image reconstruction using inter-channel correlation

By taking a closer look at the problem, it is very clear that the three  $\{\mathcal{R}, \mathcal{G}, \mathcal{B}\}$  channels are deeply correlated. Many methods make the assumption of the inter-channel correlation, and then uses it in order to obtain better results.

The final presented solution of this project will be constructed on the inter-channel correlation. Plus, the correlation will be weighed considering the presence of an edge or not. The algorithm is based on the work Duran, Joan and Buades, Antoni (2015), **A Demosaicking Algorithm with Adaptive Inter-Channel Correlation**, *Image Processing On Line*.

The first method proposed in this article, to which we will refer as `local_int()`, computes the gradient of each channel in the  $\mathcal{UV}$  space and uses it to weight each interpolation made on a directionnal gradient.

The steps of the algorithm `local_int()` are as follow :

1. Create a  $\{\mathcal{W}, \mathcal{H}, 4\}$  ( $\mathcal{W} = 1024$ ,  $\mathcal{H} = 1024$ ) array for each channel. The vector at each pixel will contain a directional ( $m \in \{n, s, e, w\}$ ) interpolation.
2. Fill the recently created arrays with the known values.
3. For the greens  $\mathcal{G}$ , interpolate with the factor  $\beta$  for the four directions. (e.g. for north :

$$\mathcal{G}_{i,j}^n = \mathcal{G}_{i,j-1}^n + \frac{\beta}{2}(\mathcal{C}_{i,j} - \mathcal{C}_{i,j-2})$$

with  $\mathcal{C} \in \{\mathcal{R}, \mathcal{B}\}$ .

4. Create two  $\mathcal{C}$  arrays of size  $\{\mathcal{W}, \mathcal{H}, 4\}$ , filled with their respective known values. Create two  $\mathcal{CG}$  arrays of size  $\{\mathcal{W}, \mathcal{H}, 4\}$ , such as  $\mathcal{CG}^m = \mathcal{C} - \beta\mathcal{G}^m$ .
5. Apply a bilinear convolution kernel to  $\mathcal{CG}$  to fill the missing values of  $\mathcal{C}$ , and add  $\beta\mathcal{G}^m$ .

6. Compute the gradient on  $\mathcal{U}, \mathcal{V}$  estimated for each directionnal interpolation. e.g. for north :

$$\nabla_{i,j}^n = \frac{1}{L} \sum_{x \in \{U,V\}} \left( \sum (X_{i,j-l}^n - X_{i,j}^n)^2 \right)^{\frac{1}{2}}$$

7. Inverse and normalize them into  $w_{i,j}^m$ .

8. For each pixel, weight the four directionnal interpolation with  $w$ .

This method is much slower than the previous ones. However, it leads to very good results.



Figure 4: Inter-channel correlation

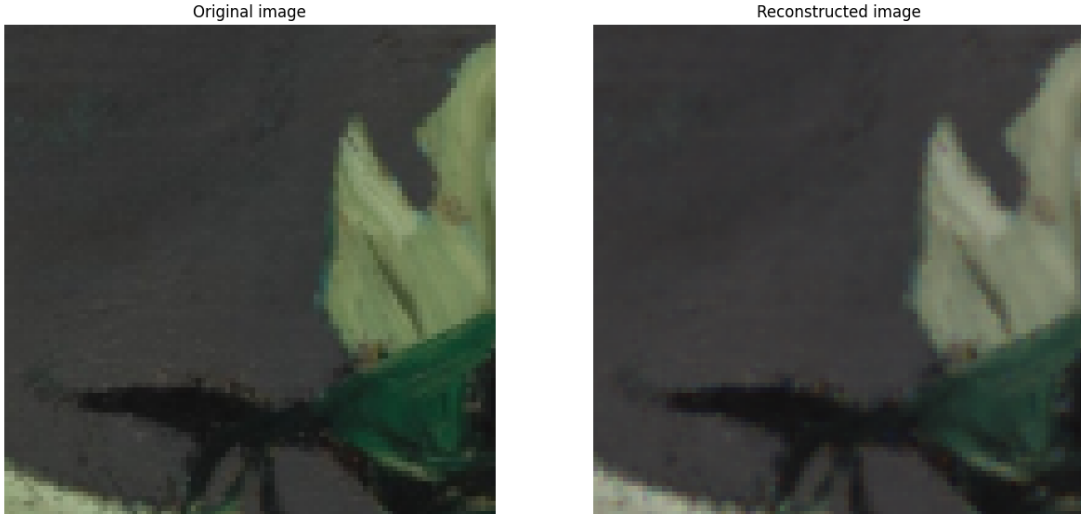


Figure 5: Zoom - Inter-channel correlation

As we can see in the above figure, the results are both extremely accurate and inaccurate at the same time. The spatial representation is very sharp and precise, and edges are well reconstructed, thanks to the gradient computation. However, there are some issues with the colors. Weirdly enough, the reconstructed image, even after normalization, does not have the same colors as the original image. In fact, there is even a very noticable difference. This is shown in the table, in which the PSNR is lower than the one of the previous method. This metric is indeed sensible to color mismatch. However, one can appreciate the fact that the SSIM is higher. This is due to the fact that the SSIM is more about forms and structure, which are better reconstructed.

PSNR	30.87
SSIM	0.9601

Table 3: Local interpolation

Other methods could be considered to reconstruct the colors, while keeping the spatial precision of this method. For instance, Daniele Picone faced this issue in *Model Based Signal Processing Techniques for Nonconventional Optical Imaging Systems*, in which she reconstruct a high resolution colored image using a low resolution colored image and a high resolution grayscale image.

Moreover, the proposed method is only the first step of the reconstruction process proposed in the article.

Actually, a second step would be required to make some non-local interpolation, comparing for each pixel its neighbourhood with neighbourhood of pixels within a certain distance. This method allow to reconstruct with precision when dealing with repetitive patterns, such as fences. However, the implementation of the method is unsuccesfull, since it requires to create and manipulate  $\{\mathcal{H}, \mathcal{W}, \mathcal{H}, \mathcal{W}\}$  arrays. The tested implementation all led to the OS deleting the job, which consumed way to much RAM.

## 4 A word on Quad-layer Bayer

From the different research made, two things stand out :

1. The Quad-layer Bayer layout is not very used, compared to the classical Bayer layout
2. The best algorithm to reconstruct Quad layer Bayer is to convert it to a Bayer layout

The website *Pyxalis* proposes three methods for the Quad bayer. The first one is a simple interpolation, already implemented. Two others method are presented, to swap the pixel into a Bayer layout. Since the proposed method already yields satisfying results, most of the work has been focused on the Bayer layout. Hence, there are no modifications to the Quad Bayer.

## 5 References

- [1] - Joan Duran, and Antoni Buades, *A Demosaicking Algorithm with Adaptive Inter-Channel Correlation, Image Processing On Line*, 5 (2015), pp. 311–327. <https://doi.org/10.5201/ipol.2015.145>
- [2] - Daniele Picone. *Model Based Signal Processing Techniques for Nonconventional Optical Imaging Systems*. Université Grenoble Alpes [2020-..], 2021.

## 6 Conclusion

It seems that a lot of work is conducted on demosaicking, and it is very understandable given the current structure of the acquisition hardware. Hence many methods, utilizing some propoities (the image pixels are not non-correlated) are developped to reconstruct colored images as good as possible. Trough my research, many deterministic methods were found. However, it seems that the current trend for the past few years has been to use CNNs. Indeed, given enough training, the convolutionnal networks might be able to outclass classical methods. However, for high pixel images, the training and data is a real bottleneck, but the deployment of more and more HPCs can explain this transition.

The work used in this project, based on adaptative cross-correlation, exhibits very good spatial results, even though everything has not be implemented. However, an issue remain, consisting of some degradation of the colors. The method is also quite slow, since many gradients have to be computed for each pixels (currently, 12 pixels are taken into account for each pixel). Plus, the choice of the parameters was completely empiristic. The author proposed a method to estimate the best parameters, but since the color reconstruction is unsuccessful, only one iteration of the algorithm is performed.