

1 Introduction

Most modern cameras have color filters that allow to capture, depending on the image pixels, different information about the image colors. These filters are of red, blue or green color and arranged in patterns (or mosaics) of two main types : the **Bayer** pattern and the **quad Bayer** pattern. Because the filters will only acquire one color during the capture process, the photography will be a grayscale image. In order to restore the true image colors, an operation called **demosaicking** is performed. The objective of this project is to test different demosaicking methods on a set of four images from the National Gallery of Art database, in order to evaluate their performances through the metrics **PSNR** (Peak Signal to Noise ratio) and **SSIM** (Structural Similarity Index Measure).

2 Presentation of the previously implemented method

In this project, a first method was already implemented, named the *naive interpolation*. In case of the Bayer filter, it consisted of the classical *bilinear interpolation*, which relied on the convolution with zero-padding between the original image and the following

kernels : $\frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ for red and blue pixels and : $\frac{1}{4} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ for green pixels. For the

quad Bayer filter, a convolution with varying kernels was performed. With this algorithm, the following results were obtained for the first image :

Bayer filter :

— PSNR : 34.63

— SSIM : 0.9502

Bayer filter :

— PSNR : 30.98

— SSIM : 0.9108



(a) *Original image*



(b) *Result (Bayer)*

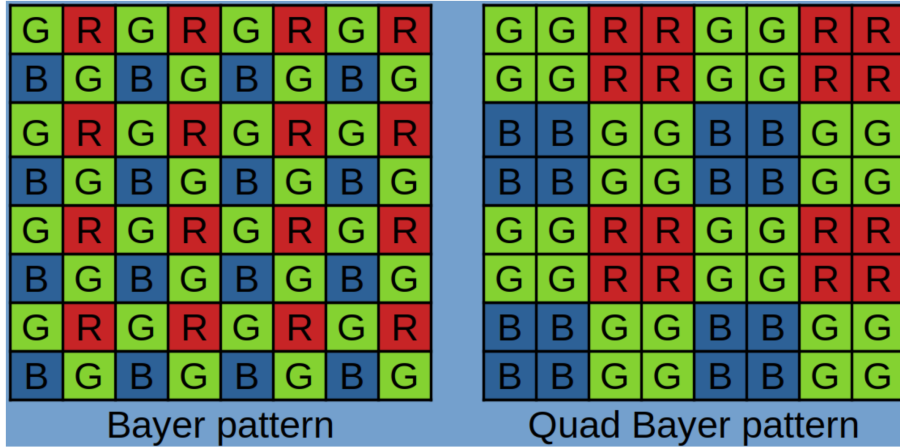


(c) *Result (Quad Bayer)*

FIGURE 1 – *Initial implementation results*

The PSNR represents the ratio of useful information with respect to noise within an image in dB. In general, a value between 20 and 30 dB indicates that the image is of good quality with some artefacts and distortions, and values between 30 and 50 are considered very good. The SSIM indicates the similarity between two images, with 1 representing identical images and -1 perfectly anti-correlated images. Based on these results, we can conclude that the initially implemented result is already very good, as the PSNRs are above 30 dB and the SSIMs above 0.9. In addition to that, we observe visually very little to no difference between the original and the restored images. The goal of the next sections is to test different methods to obtain at least as good results as this method.

3 A second "naive interpolation"



By looking at the two filter patterns above, we observe that they are quite similar, except that the quad Bayer pattern components (squares) for each color are 4 times bigger (on 4 pixels) than the bayer pattern. A "naive" idea that was thought of is to create inspired kernels from the classical Bayer bilinear interpolation in order to restore

the initial image. The following kernels were used : $\frac{1}{16}$ $\begin{bmatrix} 1 & 1 & 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 \\ 2 & 2 & 4 & 4 & 2 & 2 \\ 2 & 2 & 4 & 4 & 2 & 2 \\ 1 & 1 & 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 \end{bmatrix}$ for red and

blue pixels and : $\frac{1}{16}$ $\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 4 & 4 & 1 & 1 \\ 1 & 1 & 4 & 4 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$ for green pixels.

Therefore, the method consisted simply on keeping the bilinear interpolation for the Bayer filter and performing the convolution of the image with the kernels above for the quad Bayer filter. The table below presents the obtained results (quad Bayer filter) :

| Image N° | PSNR | SSIM |
|----------------|-------|--------|
| Image 1 | 28.88 | 0.8474 |
| Image 2 | 26.22 | 0.6236 |
| Image 3 | 27.41 | 0.7382 |
| Image 4 | 26.33 | 0.6017 |

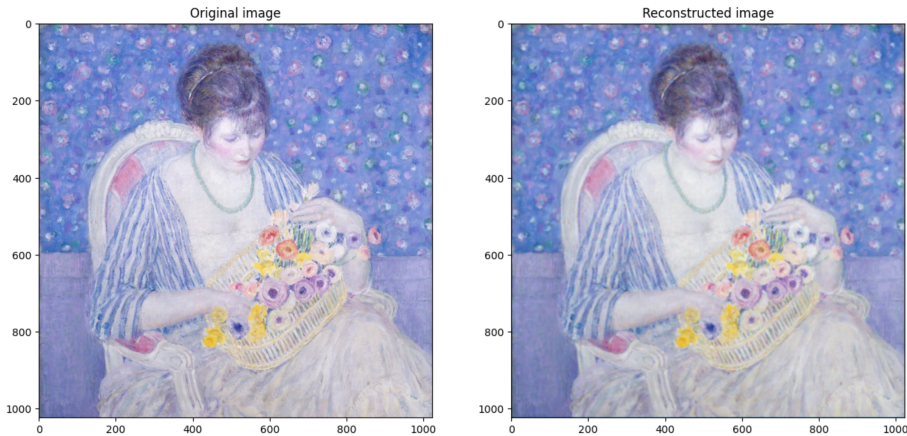


FIGURE 2 – Results obtained for the lowest PSNR and the second lowest SSIM

We observe that visually, the true image colors are well represented in the reconstructed image and that in general, the PSNRs are quite high, which indicates that the reconstruction is done fairly well. However, the SSIM is not very high and the results are overall lower than those of the initial method, which could be due to the fact that in this method, we perform the convolution with a stride of 1 (instead of a stride of 2), which could have created crosstalks with the colors. Unfortunately, the module `scipy.signal` does

not provide a convolution with custom stride. A function `convolution_pad_stride` was implemented in the file `functions.py`, but did not give satisfactory results.

4 Algorithm for the bayer filter

Even though the bilinear interpolation gives quite satisfactory results, one of its drawbacks is that it can be not efficient enough when it comes to region of high gradient and contrast. This led to researchers in image processing trying to find out different techniques to better reconstruct the camera-captured images. One of the techniques is called the **Directional Filtering with a *posteriori* Decision**, proposed by Daniele MENON et al. in the paper [1]. Its principle consists of firstly interpolating the green pixels with respect to edges (the interpolation is mainly done on the horizontal and vertical directions G^H and G^V). The red and blue components are interpolated based on the information about the green component : the chrominance components are calculated for the horizontal ($R - G^H$, $B - G^H$) and the vertical ($R - G^V$, $B - G^V$) interpolated green image, then a decision between the two chrominances is made based on the smoothness and color information : in fact, a "natural property of images" is that the color differences are smooth except for the edges, and that the big color changes are more observed "across the edges than along them". Based on this information, we can select the chrominance value that gives the best performance [1]. An implementation of the algorithm is proposed on the GitHub repository [2], which was used to evaluate the efficiency of this method. A comparison between the initial method and the Daniele Menon method (Bayer filter) is established in the table below :

| Image N° | Initial method | Daniele Menon method |
|----------|-------------------------------|-------------------------------|
| Image 1 | PSNR : 34.63 SSIM : 0.9502 | PSNR : 35.87 SSIM : 0.9622 |
| Image 2 | PSNR : 30.31 SSIM : 0.8430 | PSNR : 31.31 SSIM : 0.8830 |
| Image 3 | PSNR : 31.98 SSIM : 0.8941 | PSNR : 32.88 SSIM : 0.9161 |
| Image 4 | PSNR : 29.88 SSIM : 0.8145 | PSNR : 30.71 SSIM : 0.8552 |

We observe that this method gives satisfactory results and that the values are better than the initial method, however, from a computational point of view, we can question the use of this method, especially takes it takes time and memory resources and that it gives only slightly better results than the bilinear interpolation, which is much faster and less memory-consuming.

5 Conclusion

This project allowed to study different interpolation methods in the case of image demosaicking in order to evaluate the efficiency and the limits of each method. For example, the "naive quad Bayer implementation" has a very simple principle but the constructed kernels have their limits when it comes to the crosstalk between the different colors. The bilinear method is simple and efficient, but is quite limited when it comes to abrupt variations across edges. The Daniele MENON method gives better results, but is resource-consuming. In addition to that, the proposed methods in this project are only applicable to a specific filter category. Some of the improvements we could think of is to perform a generalization of both filter types, reduce the computational time for the Directional Filtering, as well as test the "naive quad Bayer implementation" with a convolution of a stride of 2.

6 References

- [1] Daniele MENON, Stefano ANDRIANI, *Student Member, IEEE* and Giancarlo CALVAGNO, *Member, IEEE*, "Demosaicing With Directional Filtering and a posteriori Decision". In : *IEEE Transactions on Image Processing*, Vol. 16, No. 1 (January 2007), p :132 - 141, [Online, last visited on 31/01/2024] : http://elynxsdk.free.fr/ext-docs/Demosaicing/todo/Menon_Andriani_IEEE_T_IP_2007.pdf
- [2] Alexander SRIBNYACK, *GitHub Repository : Daniele Menon's Demosaicing Algorithm* (2023), [Online, last visited on 31/01/2024] : https://github.com/sribnyak/demosaicing/blob/master/demosaicing_naive.py