



## IMAGE ANALYSIS

---

# Project Demosaicking - Report

---

Date : February 1, 2024  
Simon LAURENT

### Contents

<b>1</b>	<b>Project Statement</b>	<b>1</b>
<b>2</b>	<b>Solution implemented</b>	<b>1</b>
<b>3</b>	<b>Results</b>	<b>2</b>
<b>4</b>	<b>Conclusion</b>	<b>3</b>

# 1 Project Statement

RGB cameras use Color Filters Array (CFA) in order to get images. These CFA are divided in three filters (Red, Green and Blue) and can represent a periodic pattern or can be totally random. Here, in this project we will focus on some periodic pattern known as Bayer pattern or quad Bayer pattern. These patterns have higher filters for green color, and this can be explain because our eyes are more sensitive to green.

The output of these three filters is an image in gray scale because the different filters keep only the value of one channel by pixel. From this image in gray scale we want to get the total image in RGB, so we have to reconstruct the missing values for each channel of every pixels. This process is called demoisaicking. That's the aim of our project here.

A solution was already implemented and the aim was to find another one in order to compare results.

## 2 Solution implemented

By looking on the solution already implemented, I saw that it was naive interpolation, so I looked for other methods in scientific papers. There were many methods using interpolation (bilinear interpolation, constant hue-based interpolation, median-based interpolation, gradient-based interpolation, ...). I also chose a method which is based on interpolation, this method is called Freeman Median Demosaicking.

This method consists in using median filter with a kernel of 3x3 (or 5x5) on the difference between two channels of the image in gray scale that we have. We get 3 matrices which will be used to reconstruct the three channels of our image. We will reconstruct each channel by using the pixels that we had at the begining, and for the two missing regions, we will use a combination between one of the median matriw that we got and the initial pixels of the region that we want to reconstruct. For example, we want to reconstruct the red channel, so we have to reconstruct the red pixels in green positions. So we add the initial green pixels with the matrix obtained with the median filter between red and green.

On the figure 1, we can see the code for the implementation of this method. The same processus is done for all the three channels. Binary masks are made at the beginning in order to keep only some pixels in the median matrices during the operations.

```

def freeman_median_desaicking(op: CFA, y: np.ndarray) -> np.ndarray:
    """Performs the Freeman's method with median for desaicking.

    Args:
        op (CFA): CFA operator.
        y (np.ndarray): Mosaicked image.

    Returns:
        np.ndarray: Desaicked image.
    """
    z = op.adjoint(y)
    res = np.empty(op.input_shape)

    mask_r = np.zeros_like(z[:, :, 0], dtype = float)
    mask_g = np.zeros_like(z[:, :, 1], dtype = float)
    mask_b = np.zeros_like(z[:, :, 2], dtype = float)

    D_rg = z[:, :, 0] - z[:, :, 1]
    M1 = medfilt2d(D_rg, kernel_size=5)

    D_gb = z[:, :, 1] - z[:, :, 2]
    M2 = medfilt2d(D_gb, kernel_size=5)

    D_rb = z[:, :, 0] - z[:, :, 2]
    M3 = medfilt2d(D_rb, kernel_size=5)

    res[:, :, 0] = z[:, :, 0] + (M1 * mask_g + z[:, :, 1]) + (M3 * mask_b + z[:, :, 2])
    res[:, :, 1] = z[:, :, 1] + (z[:, :, 0] - M1 * mask_r) + (M2 * mask_b + z[:, :, 2])
    res[:, :, 2] = z[:, :, 2] + (z[:, :, 1] - M2 * mask_g) + (z[:, :, 0] - M3 * mask_r)

    return res

```

Figure 1: Function implemented for the reconstruction

### 3 Results

This method has been used on the four images using the two different types of CFA. In order to measure the performances of this method, the PSNR (Peak Signal to Noise Ratio) and the SSIM (Structural Similarity Index Measure) are computed. All the results are given in the table 1

Table 1: PSNR and SSIM for the different images with the two types of CFA

		Bayer CFA	Quad Bayer CFA
Image 1	PSNR	22.75	22.75
	SSIM	0.7482	0.7477
Image 2	PSNR	16.50	16.50
	SSIM	0.3698	0.3670
Image 3	PSNR	18.48	18.48
	SSIM	0.4122	0.4111
Image 4	PSNR	17.00	17.00
	SSIM	0.4888	0.4874

results are really different for each images, these are really satisfying only for the first image. We can also check the images reconstructed that we got in output. We can see these images on figure 2.

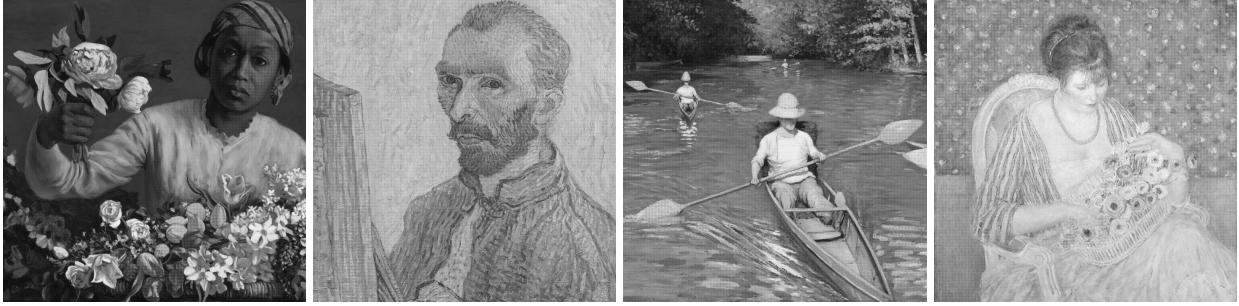


Figure 2: Reconstructed images with Bayer CFA

Unfortunately, there seems to be an issue with the displays of the images, they are still in gray scale but when we checked the matrix of the image, all the three channels are full of values, and the results given by the measures show that these values are not that bad. On the figure 3, we can see an output given in order to visualize the shape and the values of the matrix of the image.

```
res = run_reconstruction(y, cfa_name)
✓ 1.0s

# Prints some information on the reconstruction
print(f'Size of the reconstruction: {res.shape}.')
print(res)
✓ 0.0s

Size of the reconstruction: (1024, 1024, 3).
[[[0.19920319 0.19920319 0.19920319]
 [0.19521912 0.19521912 0.19521912]
 [0.19521912 0.19521912 0.19521912]
 ...
 [0.20318725 0.20318725 0.20318725]
 [0.21115538 0.21115538 0.21115538]
 [0.21912351 0.21912351 0.21912351]]

 [[0.19123506 0.19123506 0.19123506]
 [0.19920319 0.19920319 0.19920319]
 [0.20717131 0.20717131 0.20717131]
 ...
 [0.21912351 0.21912351 0.21912351]
 [0.21513944 0.21513944 0.21513944]
 [0.20318725 0.20318725 0.20318725]]

 [[0.20717131 0.20717131 0.20717131]
 [0.19920319 0.19920319 0.19920319]
 [0.17928287 0.17928287 0.17928287]
 ...
 [0.20717131 0.20717131 0.20717131]
 [0.19920319 0.19920319 0.19920319]
 [0.19920319 0.19920319 0.19920319]]
```

Figure 3: Visualisation of the matrix of the image

## 4 Conclusion

The results given by this method are not better than these obtained with the naive interpolation and the fact that we can't visualize the colour reconstruction doesn't help to see the performances of this method. Metrics are just way to quantify but it's not the first thing which can describe how a method is efficient.

Here I talked only about different interpolation methods based on some tools. But an other possibility to perform demosaicking is the use of neural network. A method like this would just need a dataset in order to train a model and to perform demosaicking on new images.