FORTIN Côme

SICOM 3A

Image Analysis report : Demosaicking

January 2024

GRENOBLE INP - PHELMA : Ecole nationale supérieur de physique, électronique, matériaux
Bât. Grenoble INP - Minatec, 3 Parvis Louis Néel - CS 50257, F-38016 Grenoble Cedex 01
Tél +33 (0)4 56 52 91 00 , Fax +33 (0)4 56 52 91 03
http ://phelma.grenoble-inp.fr

Table des matières

Table des figures

Liste des tableaux

1 Problem statement

Red, green, and blue are the three colour values that typically correspond to each pixel in a digital colour image. However, the majority of everyday cameras employ a CCD sensor, which measures one colour per pixel. In the process known as demosaicking, the other two components need to be interpolated from nearby pixels. There are two different sensor layout configurations : the Bayer pattern and the Quad Bayer pattern [1]. These two configurations are shown in figure 1

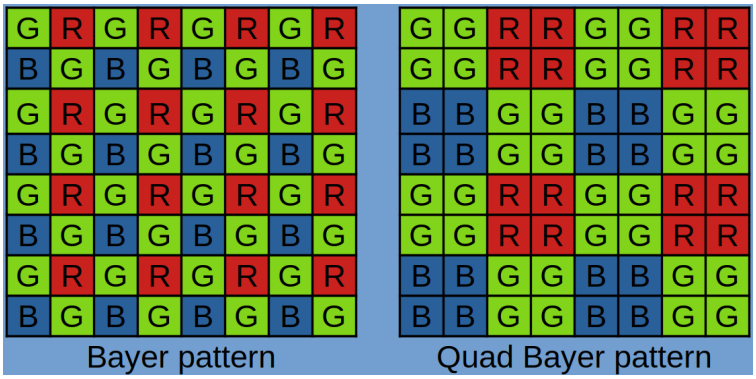

FIGURE 1. Color Filters Array patterns

2 Selected solution

The method proposed to respond to this project's problematic is the local directional interpolation with adaptive inter-channel correlation method described in [3]. The goal is to implement this method in Python. This method is based on the observation that, for a given image, there is a more or less correlation between channels. It is therefore proposed to introduce a parameter $0 < \beta \leqslant 1$, with 0 meaning that there is no correlation among channels, and 1 a strong correlation. The authors of the paper have experimentally verified that a minimum value of $\beta = 0.7$ can be used.

This method was set up by the authors by considering only the filter with the Bayer pattern. It doesn't work with the Quad Bayer pattern. In what follows, let Ω_R, Ω_G and Ω_B represent the portions of the picture grid that contain the red, green, and blue channel values, respectively.

2.1 Interpolation of the green channel

Because of the Bayer pattern configuration, green pixels are found in greater quantities. So, it's better to interpolate first the green channel. For $(i, j) \notin \Omega_G$, the green value can be estimated on the uper, lower, right and left pixels. The gradient along the north (n), south (s), east (e), and west (w) directions can be accurately calculated using these values. Let $G_{i,j}$ be the intensity of the green channel and $C_{i,j}$, $C \in \{R, B\}$, be intensity of the red or blue channel. The missing green value can be interpolated by :

$$\widehat{G}^n_{i,j} = G_{i,j-1} + \frac{\beta}{2}\left(C_{i,j} - C_{i,j-2}\right), \quad \widehat{G}^s_{i,j} = G_{i,j+1} + \frac{\beta}{2}\left(C_{i,j} - C_{i,j+2}\right),$$
$$\widehat{G}^e_{i,j} = G_{i+1,j} + \frac{\beta}{2}\left(C_{i,j} - C_{i+2,j}\right), \quad \widehat{G}^w_{i,j} = G_{i-1,j} + \frac{\beta}{2}\left(C_{i,j} - C_{i-2,j}\right)$$

2.2 Interpolation of the Red and Blue Channels

Now that the missing green values have been interpolated, it is possible to do the same for the red and blue channels. The authors made the observation that the red (or blue) and green difference images can be interpolated more correctly than the red and blue channels themselves because they mostly include low-frequency components. Let $m \in \{n,s,e,w\}$ be the direction along which green channel has been previously interpolated. Let $(i,j) \notin \Omega_C$, (with $C \in \{R,B\}$). First, let's define $CG^m = C_{i,j} - \beta G^m_{i,j}$. Then, let's do a bilinear interpolation as follows :

$$\widehat{CG}^m_{i,j} = \begin{cases} \frac{CG^m_{i-1,j}+CG^m_{i+1,j}}{2} & \text{if } (i,j) \in \Omega_G \text{ and } (i+1,j) \in \Omega_C \\ \frac{CG^m_{i,j-1}+CG^m_{i,j+1}}{2} & \text{if } (i,j) \in \Omega_G \text{ and } (i+1,j) \notin \Omega_C \\ \frac{CG^m_{i-1,j-1}+CG^m_{i+1,j-1}+CG^m_{i-1,j+1}+CG^m_{i+1,j+1}}{4} & \text{if } (i,j) \notin \Omega_G \end{cases}$$

Finally, the missing red or blue value is interpolated as $\widehat{C}^m_{i,j} = \widehat{CG}^m_{i,j} + \beta G^m_{i,j}$.

2.3 Pixel-level Fusion of Full Color Interpolated Image

Now we have four fully color images thanks to the interpolation of the four directions. It's important to weight them. For this, let's compute the variation of the chrominance components in the YUV space at each pixel along the four direction. We switch from RGB space to YUV space as follows, for $m \in \{n,s,e,w\}$:

$$Y^m = 0.299R^m + 0.587G^m + 0.114B^m, \quad U^m = R^m - Y^m, \quad V^m = B^m - Y^m$$

Then, the gradient of the chromatic components by using a local neighborhood of L pixels in the same direction of interpolation is computed as follows :

$$\nabla^n_{i,j} = \frac{1}{L} \sum_{X \in \{U,V\}} \left(\sum_{l=1}^{L} (X^n_{i,j-l} - X^n_{i,j})^2 \right)^{\frac{1}{2}}, \quad \nabla^s_{i,j} = \frac{1}{L} \sum_{X \in \{U,V\}} \left(\sum_{l=1}^{L} (X^s_{i,j+l} - X^s_{i,j})^2 \right)^{\frac{1}{2}}$$

$$\nabla^e_{i,j} = \frac{1}{L} \sum_{X \in \{U,V\}} \left(\sum_{l=1}^{L} (X^e_{i+l,j} - X^e_{i,j})^2 \right)^{\frac{1}{2}} \quad \nabla^w_{i,j} = \frac{1}{L} \sum_{X \in \{U,V\}} \left(\sum_{l=1}^{L} (X^w_{i-l,j} - X^w_{i,j})^2 \right)^{\frac{1}{2}}$$

A high gradient along one direction means there is an edge. So, it's not interesting to average in that direction. The weights are firstly compute as $\forall m \in \{n,s,e,w\}, \quad \widetilde{\omega}^m_{i,j} = \frac{1}{\nabla^m_{i,j}+\varepsilon}$, with $\varepsilon > 0$ here to avoid dividing by zero. Then, the weights are normalized :

$$\forall m \in \{n,s,e,w\}, \quad \omega^m_{i,j} = \frac{\widetilde{\omega}^m_{i,j}}{\widetilde{\omega}^n_{i,j} + \widetilde{\omega}^s_{i,j} + \widetilde{\omega}^e_{i,j} + \widetilde{\omega}^w_{i,j}}$$

Finally, the RGB components are computed as follows :

$$\widehat{R}_{i,j} = \omega^n_{i,j} R^n_{i,j} + \omega^s_{i,j} R^s_{i,j} + \omega^e_{i,j} R^e_{i,j} + \omega^w_{i,j} R^w_{i,j}$$
$$\widehat{G}_{i,j} = \omega^n_{i,j} G^n_{i,j} + \omega^s_{i,j} G^s_{i,j} + \omega^e_{i,j} G^e_{i,j} + \omega^w_{i,j} G^w_{i,j}$$
$$\widehat{B}_{i,j} = \omega^n_{i,j} B^n_{i,j} + \omega^s_{i,j} B^s_{i,j} + \omega^e_{i,j} B^e_{i,j} + \omega^w_{i,j} B^w_{i,j}$$

3 Results

3.1 Visual results

The figure 2 shows the result of the implemented method. The "reconstructed image" is the demosaicked one. For each image, the β that give the best result was selected. For the first and third images, $\beta = 0.9$ and for the second and forth images, $\beta = 1$. The fact that the betas are so close is consistent. Indeed, since the images come from the same image bank, the correlation between the different channels is almost the same. Visually, the result is satisfactory but not perfect. Indeed, we can see that the colors are blander than the original image. Then, there are block artifacts. Finally, there are some borders effects. Pixels on the edges of the image are miscalculated, giving false colors.

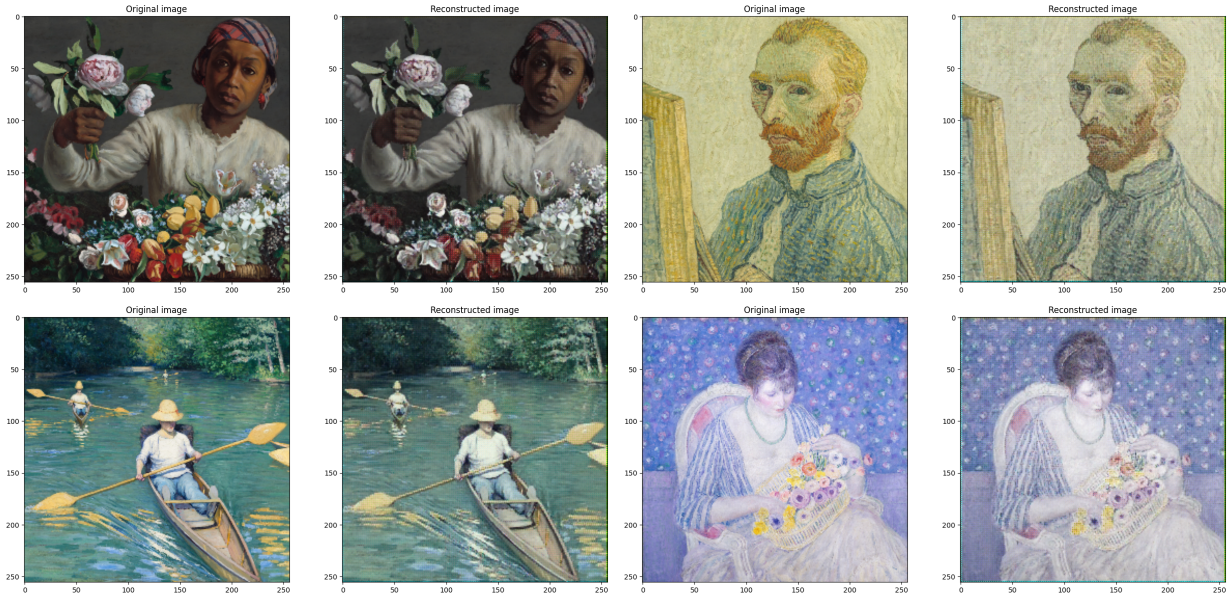

FIGURE 2. Results of the demosaicking algorithm

3.2 Performance measurements

The table 1 shows the performances of the algorithm. Two quality indexes were used. First the PSNR is the ratio of signal power to noise power. Then, the SSIM is a perceptual metric that quantifies image quality degradation. As we can see, the results are good. For each case, the PSNR is higher than 20 dB and the SSIM is close to 1. However, for the second image, the SSIM is 0,7873. It can be improved.

image n°	PSNR [dB]	SSIM
1	27,16	0,9367
2	21,97	0,7873
3	24,51	0,8401
4	22,7	0,8068

TABLE 1. Algorithm performance measurements on supplied images

4 Conclusion

The method described in [3] is satisfactory, but can be improved. First of all, the beta has been entered by hand, whereas we cannot know it in advance. We need a method for calculating it based on the working image. The article suggests one. A first attempt at implementation has been made, but it doesn't give satisfactory results and degrades image quality. Then there are artifacts. Interpolation should be smoothed. These artifact can be removed by involving image self-similarity and redundancy, as discussed in [2]. Adding this step would maybe increase the performance of the algorithm. However, due to lack of time, I wasn't able to implement this improvement.

Références

[1] B.E. Bayer. Color Imaging Array. US Patent 3 971 065, 1976.

[2] A. Buades, B. Coll, JM. Morel, and S. Catalina. Self-similarity Driven Demosaicking. Image Processing on Line, 1 :51–56, 2011.

[3] J. Duran and A. Buades. A Demosaicking Algorithm with Adaptive Inter-Channel Correlation. Image Processing on Line, 5 :311–327, 2015.