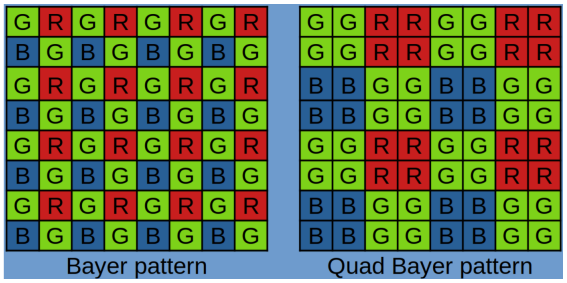
CASA NOVA Vincent
SICOM 3A EEH

Final_project Image Analysis:

I. Introduction:

In this project we will study a common issue existing with the Colors Filters Array technology used for RGB cameras.
That technology will make an acquisition by using filters to acquire one color at a time in a mosaic way.



Example of the mosaic array used.

That will lead to a raw acquisition being a gray-scale image. Thus, the aim of that project is to go from that gray scale image to recover the full RGB image.

Example of the raw acquisition.

To obtain the full RGB image we will apply an edge preservation demosaïcking method.
I chose that method after looking at several methods such as

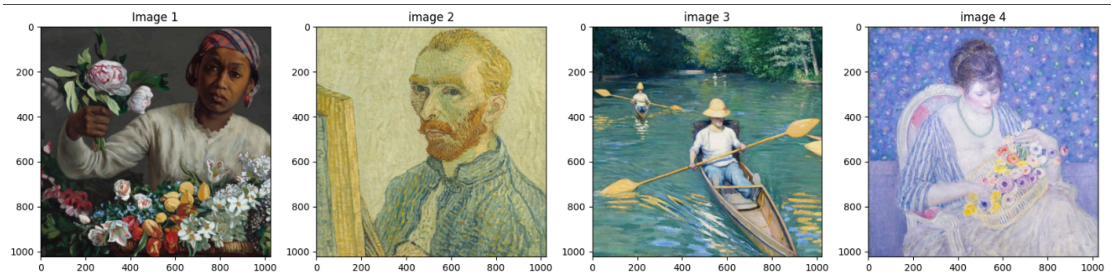
- jsp
- jsp
- jsp

Because it seemed to be an interesting approach, with a bit of complexity to implement it, and in that project my aim was to focus on applying one method a bit complex, rather than trying several simplest methods.

A. Initial data:

For the project we got a set of 4 images to reconstruct, with each time:

- The original RGB picture (To compare our results)
- The raw acquisition (CFA)



Original pictures

II. Methods:

A. Edge preservation method:

For that first method I study the technic describe in that document:

https://core.ac.uk/download/pdf/14933975.pdf

The originality of that method is to take care of the edge information, by using an orientation matrix.

In brief the method follow these steps:

- Compute the orientation matrix
- Estimate the Green Channel by using data from the orientation matrix
- Estimate the Blue and Red channels by using the previous green channel estimation.

1. Compute the orientation matrix:

The aim of the orientation matrix is to indicate if the pixel we want to estimate is on a vertical or a horizontal edge.
If the edge is vertical we will estimate the pixel by extrapolating from the horizontal pixels.
If the edge is horizontal we will estimate the pixel by extrapolating from the vertical pixels.
The values in the orientation matrix will be:

- **1**, if the **vertical gradient > horizontal gradient**.
- **0**, otherwise

2. Estimation of the green channel:

The green channel is the first channel we want to estimate because we have twice as much information about him. Then the red and blue channel will be estimated using values the estimation of this one. (that can lead to errors).

To estimate the green channel we are going to compute the extrapolation value in the four directions (top, bottom, right and left).
Then to obtain the best estimate we use a classifier using the information about the surrounding pixels.

Formulas to extrapolate in the four directions: (we took and example of a pixel at a position (M,N))

G_extra_top(M,N) = G(M-1,N) + ¾ * (B(M,N) - B(M-2,N)) - ¼*(G(M-1,N)-G(M-3,N)
G_extra_left(M,N) = G(M,N-1) + ¾ * (B(M,N) - B(M,N-2)) - ¼*(G(M,N-1)-G(M,N-3)
G_extra_right(M,N) = G(M,N+1) + ¾ * (B(M,N) - B(M,N+2)) - ¼*(G(M,N+1)-G(M,N+3)
G_extra_down(M,N) = G(M+1,N) + ¾ * (B(M,N) - B(M+2,N)) - ¼*(G(M+1,N)-G(M+3,N)

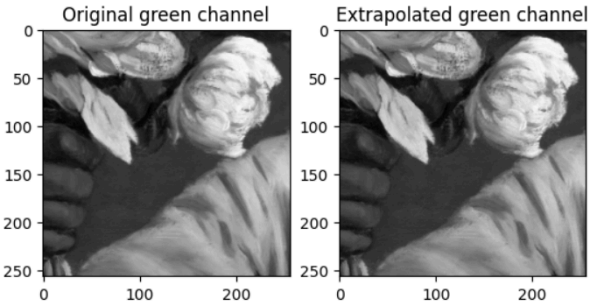
From those extrapolations we will compute the final estimation by using the median values. But, first we have to select one estimation to remove. Indeed to avoid averaging effect when computing the median, **we have to compute it on an ode number of values.**
So the aim of the classifier is to find the less interesting extrapolation.

Classifier steps:

- First we use the **orientation matrix**, if the value is 1 (vertical gradient < horizontal gradient) there is potentially a vertical edge thus the less interesting extrapolation will be the left and the right ones. (And it's the contrary if the orientation matrix value is 0)
- Second, we compute the **root mean square error** of both remaining extrapolations and the extrapolation with the biggest *rmse* will be the one selected as the least interesting one.

Finally we compute the median of the tree extrapolating values and we obtain our estimation.

Example: psnr = 41.51



3. Estimation of the red and blue channels:

The estimation of the blue and the red channels will follow the same steps:

Estimate a red pixel on a blue emplacement: (blue pixel on a red emplacement)

1. We compute the extrapolation from the top_left, the top_right, the bottom_left and the bottom_right. With the following formulas:

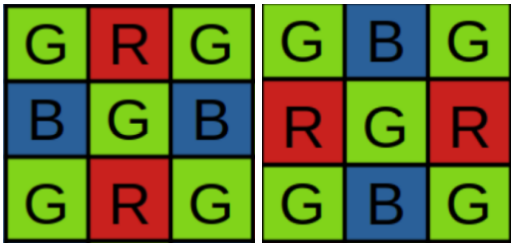
$$\mathbf{R\_extra\_top\_left(N,M)} = R(N-1,M-1) + (G\_extra(N,M) - G\_extra(N-1,M-1))$$
$$\mathbf{R\_extra\_top\_right(N,M)} = R(N-1,M+1) + (G\_extra(N,M) - G\_extra(N-1,M+1))$$
$$\mathbf{R\_extra\_bottom\_left(N,M)} = R(N+1,M-1) + (G\_extra(N,M) - G\_extra(N+1,M-1))$$
$$\mathbf{R\_extra\_bottom\_right(N,M)} = R(N+1,M+1) + (G\_extra(N,M) - G\_extra(N+1,M+1))$$

2. We drop the less interesting one (the one with the bigger *rmse*)
3. We compute the median between the tree's remaining extrapolation.

For the blue channel the steps will be the same but adapted to the blue pixels.

Estimate a red pixel on a green pixel emplacement:

1. We found the direction where there are the red pixels (blue pixels), vertical or horizontal.



Examples of the two cases, we want to extrapolate red in the center, on the left red pixels are at the vertical, on the right the red pixels are at the horizontal.

2. Compute the *rmse* of the extrapolation of red in the direction opposite to the ones with red, and the extrapolation with the highest *rmse* will be the drop out.
3. we compute the extrapolation in the 3 other directions and we compute the median to obtain the final estimation.

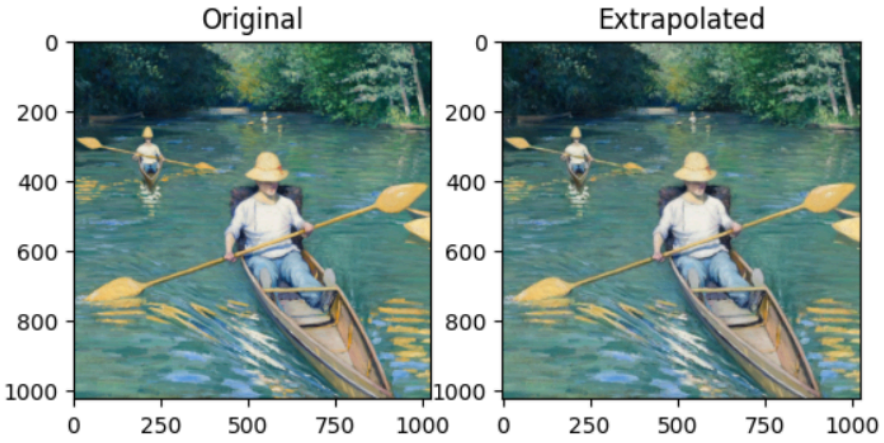
Formulas to compute the four extrapolations:

R_extra_top(N,M) = R(N-1,M) + (G(N,M) - G_extra(N-1,M)
R_extra_left(N,M) = R(N,M-1) + G(N,M) - G_extra(N,M-1)
R_extra_right(N,M) = R(N,M+1) + G(N,M) - G_extra(N,M+1)
R_extra_bottom(N,M) = R(N+1,M) + (G(N,M) - G_extra(N+1,M)

To the blue channel we'll have to follow the same steps as for the red channel. Then we obtained our estimation.

III. Conclusion:

Finally we obtained estimation for all three channels, and get the following results:
example:



CFA = "Bayer"

Images	Basic Interpolation	Edge Preservation
IMG_1	psnr = 34.63 | ssim = 0.9502	psnr = 36.23 | ssim = 0.9703
IMG_2	psnr = 30.31 | ssim = 0.8430	psnr = 32.30 | ssim = 0.9299
IMG_3	psnr = 31.98 | ssim = 0.8941	psnr = 32.69 | ssim = 0.9314
IMG_4	psnr = 29.88 | ssim = 0.8145	psnr = 31.74 | ssim = 0.9082

CFA = “Quad_bayer”

Images	Basic Interpolation	Edge Preservation
IMG_1	psnr = 30.98 | ssim = 0.9108	psnr = 31.61 | ssim = 0.9211
IMG_2	psnr = 26.96 | ssim = 0.7577	psnr = 28.47 | ssim = 0.8184
IMG_3	psnr = 28.61 | ssim = 0.8280	psnr = 28.73 | ssim = 0.8401
IMG_4	psnr = 26.65 | ssim = 0.7235	psnr = 28.79 | ssim = 0.8136

Consequently the results obtained are satisfactory because they give better results than the basic interpolation method.

How to improve results ?
The method used seemed pretty efficient, and even if I tried to change some steps of the methods (such as changing the median to mean, or selecting only the best extrapolation) it didn’t improve my results.
There is some examples of method I seen that could be interesting to try:

- Local covariance based methods (as a method to preserve the edge without using gradient)
- tried the Linear Minimum Mean-Square-Error Estimation proposed by Zhang–Wu
- Demosaicking using the frequency domain. as the Lian and al method

IV. Bibliographie:

- https://core.ac.uk/download/pdf/14933975.pdf (edge preservation method)
- https://hal.science/hal-00683233/document (comparison of methods)
- https://www.ipol.im/pub/art/2011/g_zwld/article.pdf (Zhang-Wu method)
- https://ivrlwww.epfl.ch/research/past_topics/demosaicing.html (Example in the frequency domain)
- https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ca3db12f4fb7e56c6bbfb7658fea0e164ab8a81d (Lian and al method)