# CSE472 Assignment 1

Yunhoe, Ku

Due date: October 6th, 2021

## 1 Intensity interpolation

### 1.1 Task 1

In task 1, we should extract each channel values from a bayer pattern image which consists of red, green and blue values. The figure 1 is the picture of bayer pattern which should be divided into each channel.

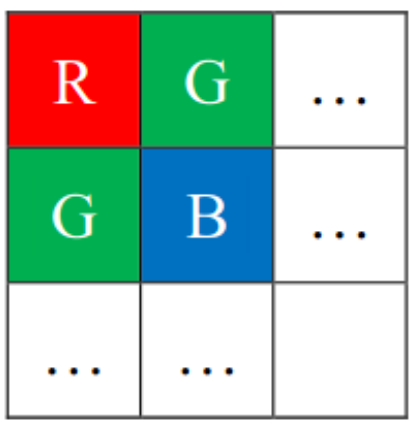



Figure 1: Given bayer pattern in this problem

So I apply a function with matlab for loop to **crayons_mosaic** file to extract each channel values like below:

$$f(x) = \begin{cases} red, & \text{if } rows, cols \text{ are odd} \\ blue, & \text{if } rows, cols \text{ are even} \\ green, & \text{otherwise} \end{cases}$$

where $x$ is pixel of image.

## 1.2 Task 2

At the task 1, we can extract each channel value from a nested image which has bayer pattern. However, despite of our effort, each channel matrix has so many hollow element so that we have to interpolate those part to get fine result. In this task, we interpolate that empty element with using filter way. The below matrices are filters which is used to interpolate vacant pixels.

$$R = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} \tag{1}$$

$$G = \begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 1 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix} \tag{2}$$

$$B = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} \tag{3}$$

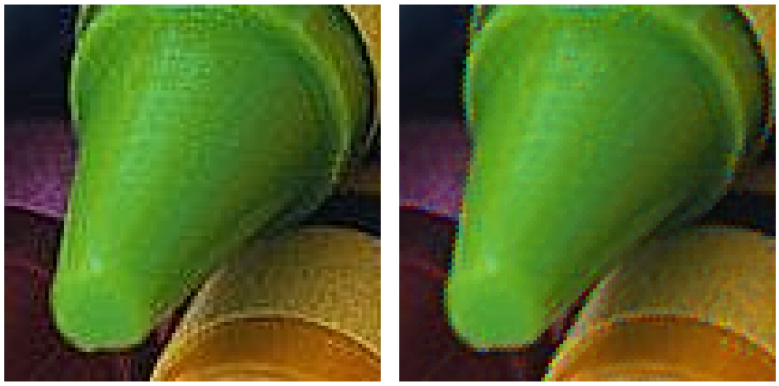
Equation (1) is for interpolating red pixels, (2) is for green pixels and (3) is for blue pixels. The reason why I design that way is only because bayer pattern makes some special pattern to each channel so that I should use two or four pixel to interpolate remaining pixels. Using that way can make average pixels for every zero-value pixel. The result is like this:

Figure 2: The result using interpolated R, G, B pixels in task 2

## 1.3 Task 3

In the above result, at first glance, it looks fine. But it is not real when we see the details of images. First we should zoom in to check the degree of completion. In fig. (3), there are two images. One is ground truth, and the other is interpolated

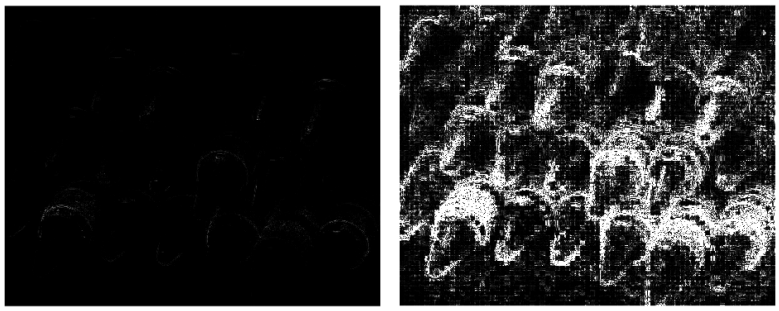
result using mosaic image. If you look closely, you can see some artifacts like RGB raw dotted pixels at the tip of color pencil.



(a) ground truth (b) interpolation

Figure 3: Zoomed figure of ground truth and interpolated result. Left one has smooth edge line at the tip of color pencil, but interpolated result has some artifact which is RGB dot clusters at the edge.

If you cannot see it really well, here is sum of squared error between ground truth and interpolated images at every RGB channel. The error is like this:



(a) sum of squared error (b) sum of squared error with threshold

Figure 4: Sum of squared error between (a) and (b) in figure 2 and normalized result with a threshold value.

In the above figure, we can calculate average error and maximum error. Before I get the result image, I normalized the value in range from 0 to 1 because I want to use double data type, but matlab cannot support some various image showing method without normalized image matrix. So you should know about that. The average error of normalized value is 0.0022 and the maximum value is 0.8204, which are quite small values.

The reason why the interpolation is not perfect is due to limitation of linear interpolation method. When we use linear interpolation, we just average nearby pixels to get result. In this process, we do **NOT** consider some conditions which affect the quality of result.

First, average method has **NOT** robustness. In other words, it is too sensitive to operands of average. When we want to get something which should have some correlation between nearby things, we usually use average method because it is quite simple and quick method. But if there is a outlier, which has no correlation among nearby elements, the average method easily loses its power due to only one fake element. That is one negative aspect of linear interpolation as the average method in this task.

Second, linear interpolation is **not** precise. Look at example figure (5). There is only a assumption like the intermediate value follows linear function when we use linear interpolation. But in the wild area, there are some values following curve function, not linear function. The difference between in below example is not really huge, but what if the function is more curvy so that the difference will be huge? For this reason, we can see some artifacts in our result even we use interpolation method.

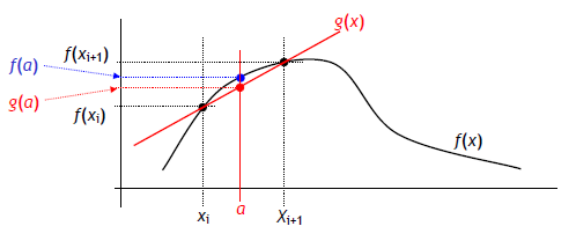



Figure 5: Explanation for linear interpolation.

# 2 Image Alignment

## 2.1 Task 4

In this task, we should divide one vertically concatenated image into three RGB parts. I just measure the height of image and divide it into equally 3 parts. The result is like this:

(a) Blue part (b) Green part (c) Red part

Figure 6: Equally divided result. From left to right, blue, green and red one.

In the pictures, we can see some black and white borders(whites cannot be easily seen due to background color is white), which is out of harmony. This is a problem which we should deal with in task 5.

## 2.2 Task 5

In this task, I use **norm2xcorr2** matlab method which can find a maximum correlation point. Using this point, firstly I set base picture from R, G and B channel matrix and aligned to that picture for rest of matrix. Then I choose one of them based on good quality. The result is like this:

(a) Blue base 00125v.jpg (b) Green base 00149v.jpg (c) Blue base 00153v.jpg

Figure 7: Aligned result with normxcorr2 matlab method for each input image

From left to right, the displacement of each channel is like this:

- 00125v.jpg: Blue channel fixed result. Red channel displacement is (-1, 9) and Green channel displacement is (-1, 10).
- 00149v.jpg: Green channel fixed result. Red channel displacement is (0, 5) and Blue channel displacement is (0, -4).
- 00153v.jpg: Blue channel fixed result. Red channel displacement is (3, 17) and Green channel displacement is (1, 11).

(a) Blue base 00351v.jpg (b) Green base 00398v.jpg (c) Blue base 01112v.jpg

Figure 8: Aligned result with normxcorr2 matlab method for each input image

Also, result is like this:

- 00351v.jpg: Blue channel fixed result. Red channel displacement is (0, 13) and Green channel displacement is (0, 4).
- 00398v.jpg: Green channel fixed result. Red channel displacement is (-1, 12) and Blue channel displacement is (-1, 6).
- 01112v.jpg: Blue channel fixed result. Red channel displacement is (1, -67) and Green channel displacement is (-1, 0).

# 3 Bonus

## 3.1 Option 2

In this section, I just add a process which cut white border and black border of image. The result is like this:

(a) Just concatenated (b) Blue base

Figure 9: Just concatenated result and blue base aligned result.

(a) Without white border (b) Without white and black border

Figure 10: Aligned without white result and aligned without white and black result.

You can check a better result when we crop white border. Because that can disrupt finding correlation between image, so removing that process can help make a better result. But if you see a result of removing white and black border, you can see that removing any border can't guarantee improving performance. But this kind of trend can't always apply to all files. I also test with problem 2 files, but some files can achieve good result when dropping the white and black border. But generally removing white border can make better result. So depending on the image, it can make good result or not when you removing black border of image.

Unfortunately, because the results have huge file size, I can't show you all the result in this report. So if you want to see, you should better to access my github repository and see those results. Please see carefully below explanation:

In **result** directory you can find two directories, **problem 2** and **problem 3**. In each of directory, you will see direcotires:

- not_aligned: You can see concatenated results that each channel is equally divided from original file.
- aligned: You can see results which are aligned with using matlab method **normxcorr2**.
- aligned_no_white: You can see **aligned** directory results with removing white border.
- aligned_no_white_black: You can see **aligned** directory results with removing white border and black border.

You can see there is no result of **aligned_no_white_black** for some tif files,

because I can't run for those files with my removing border method. So you should know about that.