Assignment – 1

Elkhan Ismayilzada

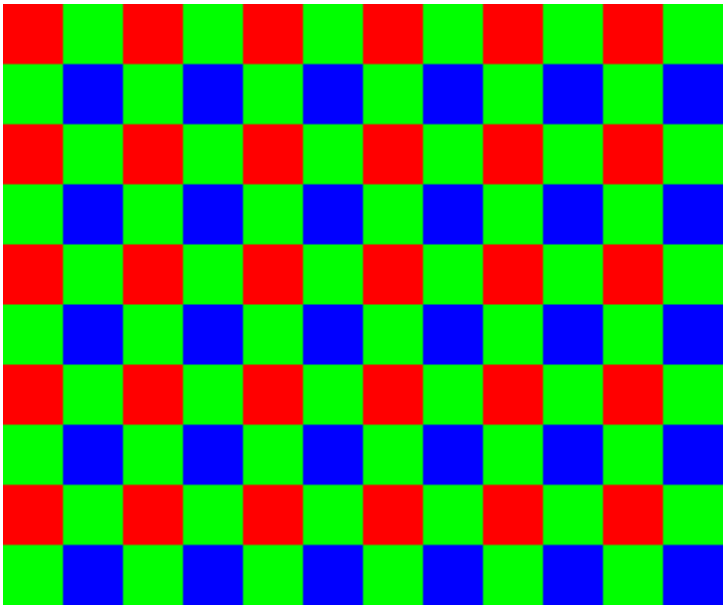
GITHUB LINK

Task 1 – To solve task 1, I tried to extract pattern from Bayer image, and as can be seen from the example Bayer picture, there is a clear pattern to extract channels. Red channel has values in odd columns and rows whereas Blue channel has values in even columns and rows and Green channel has values in even columns when rows are odd and odd columns when rows are even. Overall, following masks array can be deduced per each channel.

repmat([1 0; 0 0],h/2,w/2) – red channel mask

repmat([0 1; 1 0],h/2,w/2) – green channel mask

repmat([0 0; 0 1],h/2,w/2) – blue channel mask



Task 2 – For designing the filter, I fill missing channels by averaging four neighboring known channel values. Therefore, filters are as follows

0.25	0	0.25
0	0	0
0.25	0	0.25

Filter for R

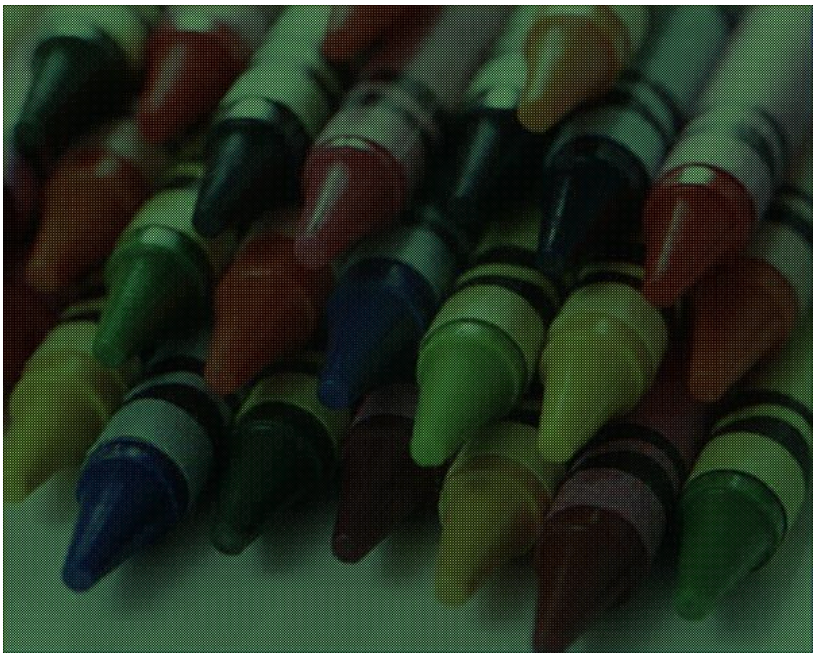
0	0.25	0
0.25	0	0.25
0	0.25	0

Filter for G

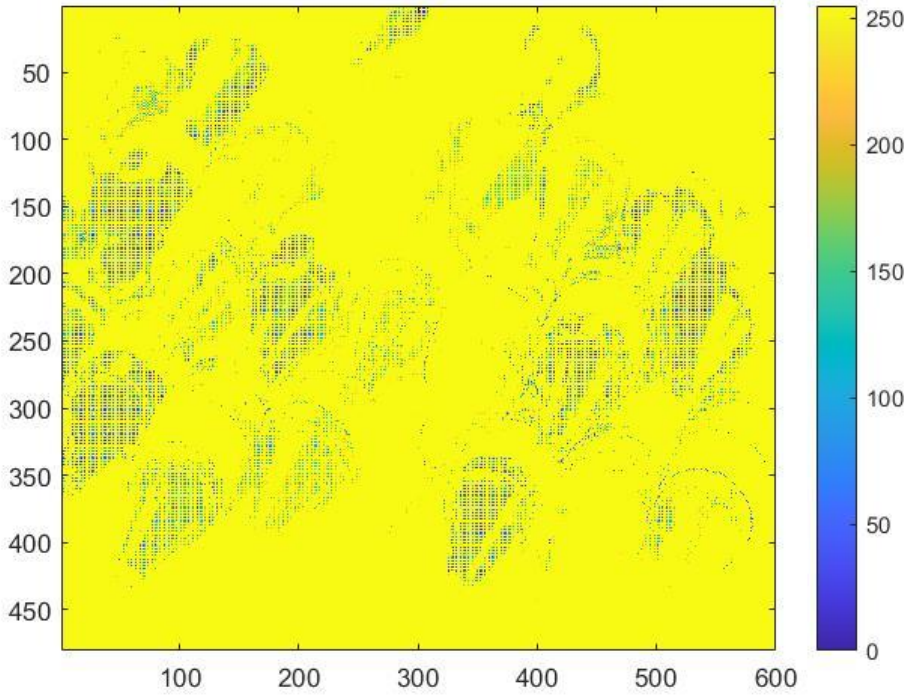
0.25	0	0.25
0	0	0
0.25	0	0.25

Filter for B

The result image is



Task 3 – As expected, there is huge difference between original and reconstructed image and the map of squared differences can prove it.



Mean and max errors are **29286.1826**, **192035** respectively.

To fix this issue, instead of only averaging, it is better to use weighted average where the values close to center pixel have more weight than those that are from it and center pixel has the highest weight. As an example, I used following filters

0.25	**0.5**	**0.25**
0.5	**1**	**0.5**
0.25	**0.5**	**0.25**

Filter for R

0	**0.5**	**0**
0.5	**1**	**0.5**
0	**0.5**	**0**

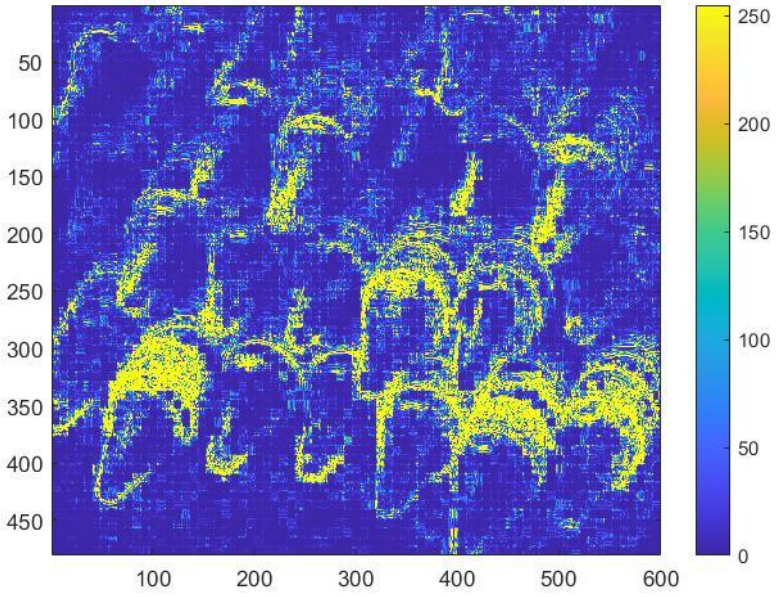
Filter for G

0.25	**0.5**	**0.25**
0.5	**1**	**0.5**
0.25	**0.5**	**0.25**

Filter for B

And I got following reconstructed image.

It is very close to the original image, hence the map of squared differences



And mean, max errors are **203.2237** and **53345.125** respectively.

Task 4 – To get the channels, slicing the image array into 3 parts will do the job.

[h,w] = size(image)

h = h-1

blue_channel = image(1:round(h/3),1:w)

green_channel = image(round(h/3)+1:round(2*h/3),1:w)

red_channel = image(round(2*h/3)+1:h,1:w)

Task 5 – In order to find best alignment, I used NCC score which is given by following formula

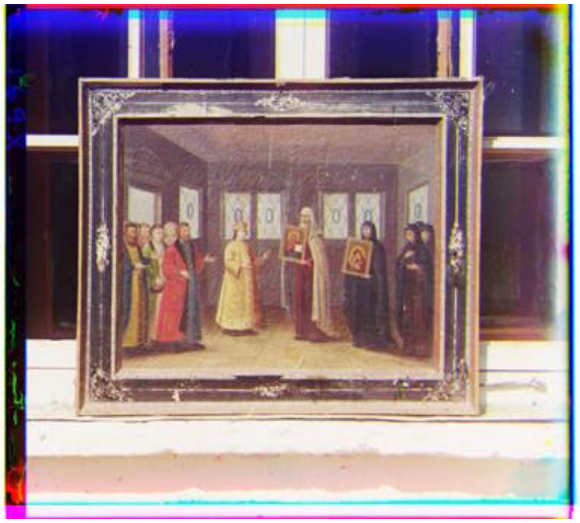
$$NCC(u,v) = \frac{\mathbf{u}}{\|\mathbf{u}\|} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

Where **u** and **v** are the matrices that we find correlation. In our scenario they are the channel matrices. For all images, I aligned blue and green channels to the red channel and here are the result pictures and their NCC scores, x y displacements.

[the channel name] – [NCC score], [x y] (x and y displacements)

blue channel - 0.92162, [-10 2]

green channel - 0.92031, [-4 1]



blue channel - 0.93898, [-9 -1]

green channel - 0.97113, [-5 0]

blue channel - 0.93238, [-11 -3]

green channel - 0.97551, [-13 -2]

blue channel - 0.96213, [-13 1]

green channel - 0.95799, [-9 0]

blue channel - 0.93539, [-8 2]

green channel - 0.9613, [0 1]

blue channel - 0.92243, [-8 -3]

green channel - 0.95676, [-8 -1]

Bonus task (Multiscale) – For this task, I recursively scaled down the image and found the best displacement then updated my estimations with it. Same as before, I aligned blue and green channels with red channel. In the smallest scale I use the [-15,15] range and for the larger ones I use [-2,2] range. Here are the result pictures and their NCC scores, x y displacements.

blue channel - 0.92703, [-50 -7]

green channel - 0.93122, [-48 -12]

blue channel - 0.94894, [-120 17]

green channel - 0.93539, [-43 10]

blue channel - 0.93783, [-134 -46]

green channel - 0.96672, [-77 -23]