COE 3SK3 Course Project 2:
Demosaicing with Linear Regression

Alexis Ng

April 10, 2021

1 Problem Description

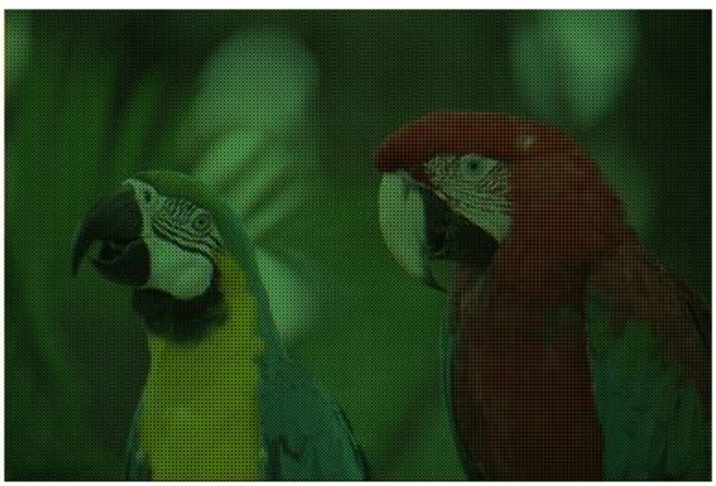
Photo sensors cannot differentiate colours and the only way to achieve colour imaging is through three-sensor camera, or Colour filter array (CFA). Raw images are captured by this type of camera and creates a mosaic of colours laid out in a Bayer Pattern. Full colour images are digitally restored using demosaicing algorithms. In this project, demosaicing with linear regression is experimented and analyzed.

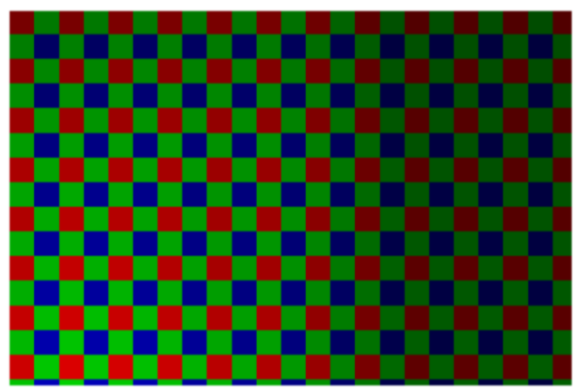
2 Tasks and Requirements

The following steps were used to implement the linear regression based demosaicing algorithm.

Training:

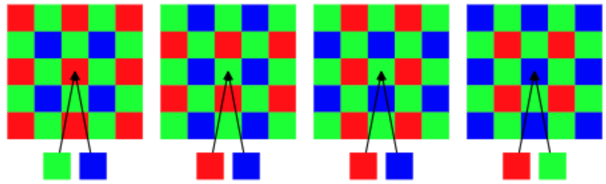
1. The 'Kodim23.png' (parrot) image from the McMaster dataset is chosen to train the algorithm due to its variety of colour in the images. The Bayer CFA pattern program from Project 1 is used to produce the raw Bayer mosaic image. The pattern used is RGGB.





2. The 4 types of mosaic patches from full-colour patches are simulated by first using the matlab function, im2col to make 5x5 patches throughout the image. A for-loop is used to go through each patch and checks to see the type of colour patch (rggb, gbrg, grbg, or bggr) and is then assigned and appended to its respective matrices (rggb, gbrg, grbg, and bggr), making up the 4 X matrices. It is noted that every odd column holds rggb and gbrg patches and every even column holds grbg and bggr pattern. In each column, every odd row holds rggb and grbg pattern and

every even row holds gbrg and bggr pattern. A for-loop going through each column and row is used to index through each patch, simulating the 4 types of mosaic patches shown below.



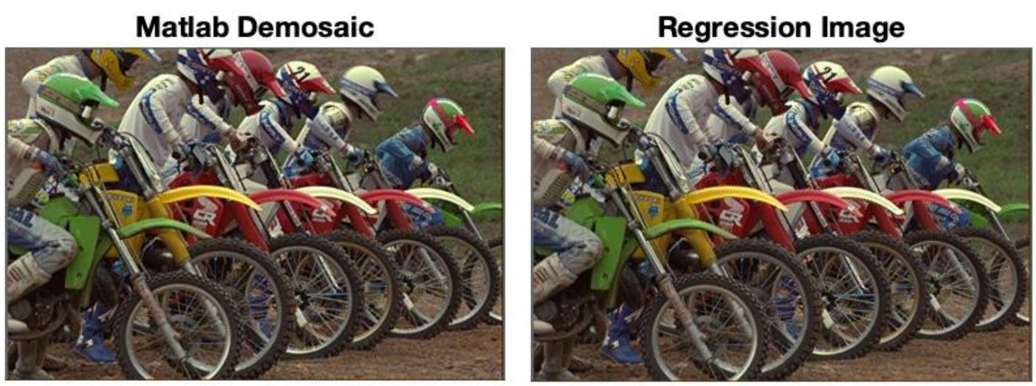
3. The linear least square problem is solved for each case (4 cases with 2 missing channels in each case). To do this, the number of columns chosen for training is 250 columns of the image (i.e. 125 for each patch). This number was chosen from trial and error and it had resulted as the optimal number to reduce underfitting and overfitting. To achieve better performance, quadratic terms were added. A total of 8 ground truth matrices are built using a nested for-loop that goes through each column and row. The following equation is then used to solve for the 8 A coefficients matrices.

$$A = (X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}R = X^{+}R$$

4. Matlab took overnight to generate the A matrices. The A matrices are saved and is then ready to use for testing.

Testing:

1. Using new testing images, the images are converted to its mosaic image and is split into 5x5 patches using im2col. Similar to training, the X matrices were created by looping through the columns knowing the colour patches based on odd/even columns and odd/even rows. A for-loop is used to go through each missing colour for each case and the ground truth is solved by multiplying its corresponding colour A matrix with the X matrix.
2. A sample image (Kodim21.png) is shown below.



3. Below show some artifacts in the images.

4. The following table shows the MSE and PSNR output when tested on different images.

MSE

Image	**Matlab**	**Linear Regression**
Lenna.png	19.511	19.3295
Kodim05.png	60.222	54.0533
Kodim07.png	22.946	26.2198
Kodim21.png	34.297	34.6498
Kodim23.png	17.609	17.5984
Lion.png	17.322	28.9881
Tree.png	78.496	67.9744

PSNR

Image	**Matlab**	**Linear Regression**
Lenna.png	35.2279	35.2686
Kodim05.png	30.3333	60.2215
Kodim07.png	34.5237	33.9445
Kodim21.png	32.7782	32.7338
Kodim23.png	35.6735	35.6761
Lion.png	35.7449	33.5086
Tree.png	29.1823	29.8074

<u>With no edge cases</u>

MSE

Image	**Matlab**	**Linear Regression**
Lenna.png	18.7774	18.9310
Kodim05.png	71.0162	59.0964
Kodim07.png	21.9081	19.7557
Kodim21.png	40.3613	34.4772
Kodim23.png	16.0184	12.9509
Lion.png	20.6596	16.7220
Tree.png	89.0136	68.2315

PSNR

Image	Matlab	Linear Regression
Lenna.png	35.3944	35.3591
Kodim05.png	29.6172	30.4152
Kodim07.png	34.7248	35.1739
Kodim21.png	32.0712	32.7555
Kodim23.png	36.0846	37.0078
Lion.png	34.9796	35.8979
Tree.png	28.6362	29.7910

In conclusion, demosaicing with linear regression was experimented in this project. The algorithm is designed using a 5x5 patch, trained up to 250 columns of the image and quadratic terms were used. We can see that for most images, the linear regression model has a slightly better performance than the matlab built-in demosaic(…) function. It's difficult to see the differences in the output image and the artifacts but it's possible to compare based on the mean square error and the PSNR. The reason for a slightly better performance is that because most natural images are piecewise smooth with high correlations with adjacent pixels and colour channels, it is possible to find missing component using linear combination of the surrounding known pixels. Using machine learning, the algorithm was trained well enough to produce a better performance than the matlab built-in function. For better performance, one can employ more images/patches in training, using a larger patch, e.g. 7x7, or using directional interpolation.

3 Image Credits

McMaster Dataset