COMPENG 3SK3 - Project 3

Color Demosaicing for Digital Cameras with Linear Regression

Instructor: Dr. Xiaolin Wu

Glen Tsui – tsuig – 400201284

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **[Glen Tsui, tsuig, 400201284]**

Implementation

Mosaicing

```
function [RGGB,GBRG,GRBG,BGGR] = mosaic(original_image)
    [x,y,~] = size(original_image);
    %disp([x,y,z]);
    %imshow(original_image(:,:,3));
    RGGB = zeros(x,y);
    GBRG = zeros(x,y);
    GRBG = zeros(x,y);
    BGGR = zeros(x,y);

    for i = 1:x
        for j = 1:y
            if mod(i,2)==0
                if mod(j,2)==0
                    RGGB(i,j) = original_image(i,j,3);
                    GBRG(i,j) = original_image(i,j,2);
                    GRBG(i,j) = original_image(i,j,2);
                    BGGR(i,j) = original_image(i,j,1);
                else
                    RGGB(i,j) = original_image(i,j,2);
                    GBRG(i,j) = original_image(i,j,1);
                    GRBG(i,j) = original_image(i,j,3);
                    BGGR(i,j) = original_image(i,j,2);
                end
            else
                if mod(j,2)==0
                    RGGB(i,j) = original_image(i,j,2);
                    GBRG(i,j) = original_image(i,j,3);
                    GRBG(i,j) = original_image(i,j,1);
                    BGGR(i,j) = original_image(i,j,2);
                else
                    RGGB(i,j) = original_image(i,j,1);
                    GBRG(i,j) = original_image(i,j,2);
                    GRBG(i,j) = original_image(i,j,2);
                    BGGR(i,j) = original_image(i,j,3);
                end
            end
        end
    end
end
```

This function converts a regular image file into 4 bayer pattern equivalents: RGGB, GBRG, GRBG, and BGGR. The dimensions of the original image is first taken as a reference point for the bayer patterns. Each Bayer pattern is indicative of the order of the colour channels, and this function extracts the specific colour channel from the image and stores it into the mosaic matrix based on the coordinate of the pixel. For example, an RGGB mosaic displays the red channel value in the top-left coordinate (both rows and columns are odd), so the red channel value at the original image at that coordinate is stored at that location in the RGGB mosaic respectively. This method allows for quick conversion of an image into a bayer pattern for training the demosaicing model.

Linear Regression - Calculate coefficient matrices

```
function [A_RGGB_G,A_RGGB_B,A_GBRG_R,A_GBRG_B,A_GRBG_R,A_GRBG_B,A_BGGR_R,A_BGGR_G] = LLSPDIV2K()
    img_dir = 'C:\Users\User\Downloads\DIV2K_train_HR\DIV2K_train_HR\';
    img_files = dir(fullfile(img_dir, '*.png'));
    pad = 2;
    X_RGGB = zeros(25,0);
    X_GBRG = zeros(25,0);
    X_GRBG = zeros(25,0);
    X_BGGR = zeros(25,0);
    GT_G1 = zeros(1,0);
    GT_G2 = zeros(1,0);
    GT_R1 = zeros(1,0);
    GT_R2 = zeros(1,0);
    GT_R3 = zeros(1,0);
    GT_B1 = zeros(1,0);
    GT_B2 = zeros(1,0);
    GT_B3 = zeros(1,0);

    window_x = 150;
    window_y = 150;
    for file = 1:length(img_files)
        img_path = fullfile(img_dir, img_files(file).name);
        img = imread(img_path);
        [x,y,~] = size(img);
        [RGGB,GBRG,GRBG,BGGR] = mosaic(img);
        ximg_min = 1+pad;
        ximg_max = x-pad;
        yimg_min = 1+pad;
        yimg_max = y-pad;
        startx = randi([ximg_min (ximg_max - window_x)]);
        starty = randi([yimg_min (yimg_max - window_x)]);
        for i=startx:ximg_max
            for j=starty:yimg_max
                kernel=RGGB(i-2:i+2,j-2:j+2);
                row_vector = im2col(kernel,[5 5]).';
                if mod(i,2)==0
                    if mod(j,2)==0
                        %BGGR
                        X_BGGR = [X_BGGR;row_vector];
                        GT_R3 = [GT_R3;img(i,j,1)];
                        GT_G2 = [GT_G2;img(i,j,2)];
                    else
                        %GBRG
                        X_GBRG = [X_GBRG;row_vector];
                        GT_R1 = [GT_R1;img(i,j,1)];
                        GT_B2 = [GT_B2;img(i,j,3)];
                    end
                else
                    if mod(j,2)==0
                        %GRBG
                        X_GRBG = [X_GRBG;row_vector];
                        GT_R2 = [GT_R2;img(i,j,1)];
                        GT_B3 = [GT_B3;img(i,j,3)];
                    else
                        %RGGB
                        X_RGGB = [X_RGGB; row_vector];
                        GT_G1 = [GT_G1; img(i,j,2)];
                        GT_B1 = [GT_B1; img(i,j,3)];
                    end
                end
                if j > window_x
                    break
                end
            end
            if i > window_y
                break
            end
        end
    A_RGGB_G = ((X_RGGB.')*X_RGGB)\(X_RGGB.')*GT_G1;
    A_RGGB_B = ((X_RGGB.')*X_RGGB)\(X_RGGB.')*GT_B1;
    A_GBRG_R = ((X_GBRG.')*X_GBRG)\(X_GBRG.')*GT_R1;
    A_GBRG_B = ((X_GBRG.')*X_GBRG)\(X_GBRG.')*GT_B2;
    A_GRBG_R = ((X_GRBG.')*X_GRBG)\(X_GRBG.')*GT_R2;
    A_GRBG_B = ((X_GRBG.')*X_GRBG)\(X_GRBG.')*GT_B3;
    A_BGGR_R = ((X_BGGR.')*X_BGGR)\(X_BGGR.')*GT_R3;
    A_BGGR_G = ((X_BGGR.')*X_BGGR)\(X_BGGR.')*GT_G2;
end
```

The training model utilizes a large number of RGB images from the DIV2K dataset to learn the statistical model. The regression model is trained by generating the bayer pattern for each 2-pixel padded image and sliding a 3-by-3 kernel to check each pixel for the pattern. The same pattern determination from the mosaic method is used, but at each iteration it appends the row vector of X values as well as the ground truth values for the two missing colours each. On initial testing, the linear regression training takes an ample amount of time looping every pixel of 800 images, and setting a row and column limit (which mainly consists of edge pixel data) may have a lower accuracy. My method creates a window at a randomly generated coordinate within the image for the kernel to iterate through. This solution ensures that there are large amounts of varying sample data for the model to train with. With the mosaic patches and ground truth components accumulated, the 8 demosaicing coefficient matrices are obtained.

Output - Use coefficients to compute result

```
function output_image = computeOutput(A_RGGB_G,A_RGGB_B,A_GBRG_R,A_GBRG_B,A_GRBG_R,A_GRBG_B,A_BGGR_R,A_BGGR_G, mosaic, x, y)
    pad = 2;
    ximg_min = 1+pad;
    ximg_max = x-pad;
    yimg_min = 1+pad;
    yimg_max = y-pad;
    output_image = zeros(x,y,3);
    for i=ximg_min:ximg_max
        for j=yimg_min:yimg_max
            kernel=mosaic(i-2:i+2,j-2:j+2);

            if mod(i,2)==0
                if mod(j,2)==0
                    %BGGR
                    output_image(i,j,3) = mosaic(i,j);
                    output_image(i,j,1) = A_BGGR_R.'*im2col(kernel,[5 5]);
                    output_image(i,j,2) = A_BGGR_G.'*im2col(kernel,[5 5]);
                else
                    %GBRG
                    output_image(i,j,2) = mosaic(i,j);
                    output_image(i,j,1) = A_GBRG_R.'*im2col(kernel,[5 5]);
                    output_image(i,j,3) = A_GBRG_B.'*im2col(kernel,[5 5]);
                end
            else
                if mod(j,2)==0
                    %GRBG
                    output_image(i,j,2) = mosaic(i,j);
                    output_image(i,j,1) = A_GRBG_R.'*im2col(kernel,[5 5]);
                    output_image(i,j,3) = A_GRBG_B.'*im2col(kernel,[5 5]);
                else
                    %RGGB
                    output_image(i,j,1) = mosaic(i,j);
                    output_image(i,j,2) = A_RGGB_G.'*im2col(kernel,[5 5]);
                    output_image(i,j,3) = A_RGGB_B.'*im2col(kernel,[5 5]);
                end
            end
        end
    end
end
```

Using the 8 demosaicing coefficients, the output image can be computed by iterating a 3-by-3 kernel across the mosaic image and applying the demosaicing coefficient for the missing colour values in each pattern. The corresponding RGB values are stored back into a 3-D output matrix at each pixel, thus fully generating the resulting demosaiced image.

Calculate RMSE

```
function rmse = calcRMSE(im1,im2)
    im1 = double(im1);
    im2 = double(im2);
    squared_diff = (im1-im2).^2;
    rmse = sqrt(mean(squared_diff(:)));
end
```

The RMSE calculation is calculated with the square root of the mean squared error.

Results

The final results of the experiment visually matched the appearance of MATLAB's demosaic function as well as the original image. The linear regression model does take a long time to train with the DIV2K dataset and window size of 300-by-300 which encapsulates the kernel. The workspace values were saved as matlab.mat file in order to load the model without having to go through the training process again. In terms of RMSE, the custom demosaic function performed well, but does not compare to MATLAB's demosaic function, even while there are no noticeable artifacts or discolourations in the resulting image.

Mosaicing Method

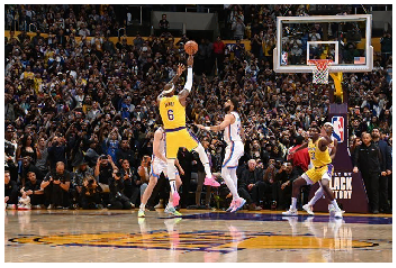
RGGB Mosaic

GRBG Mosaic

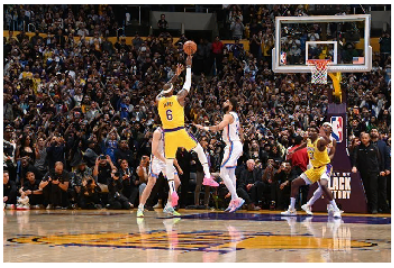
GRBG Mosaic

BGGR Mosaic

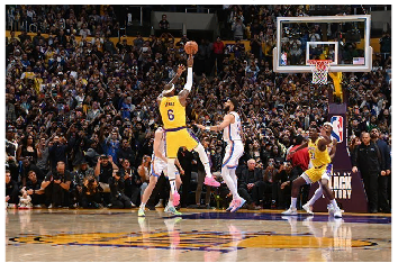
RGGB MATLAB Demosaic, RMSE: 7.3053



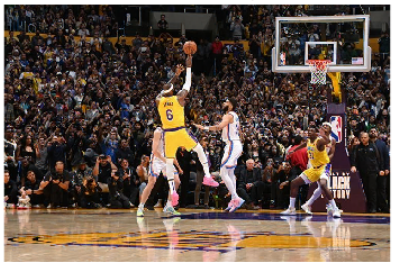
GRBG MATLAB Demosaic, RMSE: 7.3524



GBRG MATLAB Demosaic, RMSE: 7.3314



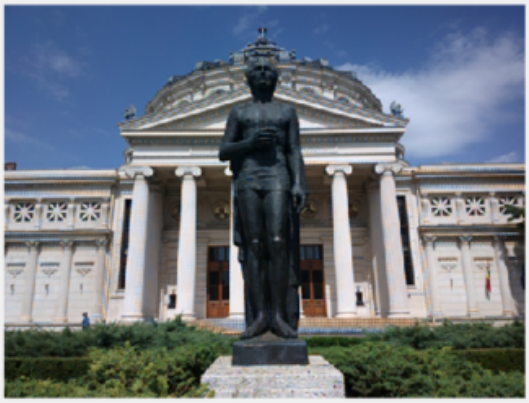
BGGR MATLAB Demosaic, RMSE: 7.2966



Dr. Wu’s Test Image 1 Results

RGGB Custom Demosaic

RGGB MATLAB Demosaic



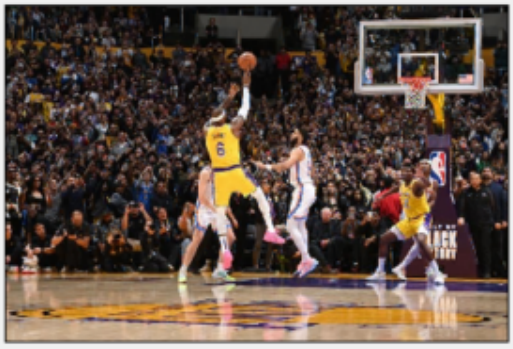
Dr. Wu’s Test Image 2 Results

RGGB Custom Demosaic

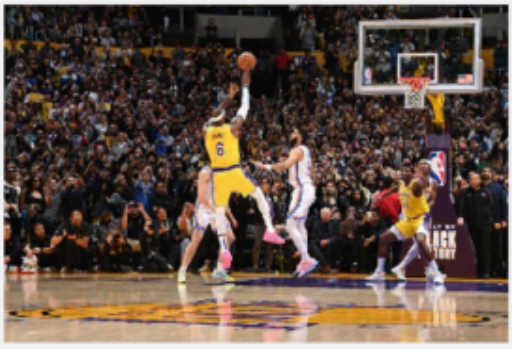
RGGB MATLAB Demosaic

Personal Test Image 1

RGGB Custom Demosaic, RMSE: 14.4803

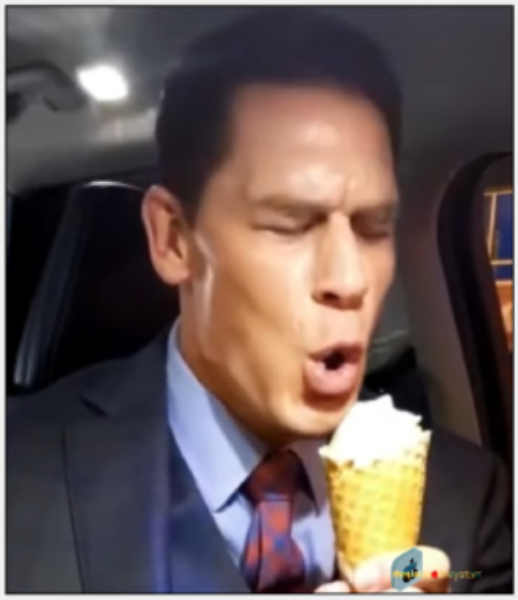


RGGB MATLAB Demosaic, RMSE: 7.3053

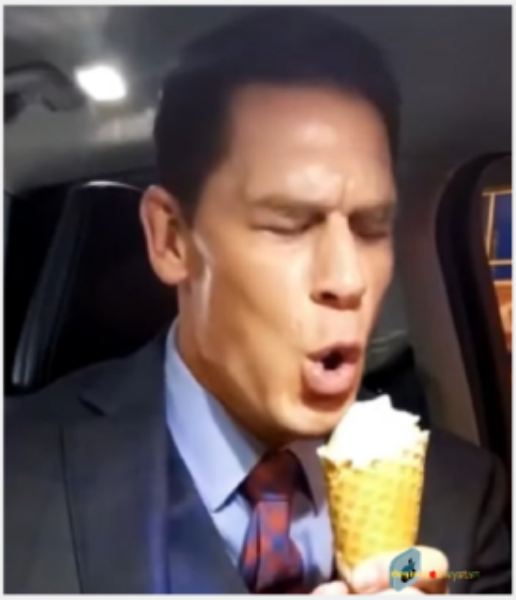


Personal Test Image 2

RGGB Custom Demosaic, RMSE: 7.4998

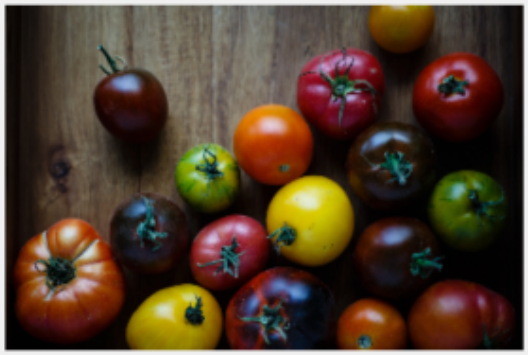


RGGB MATLAB Demosaic, RMSE: 1.4603



Personal Test Image 3

RGGB Custom Demosaic, RMSE: 4.0104



RGGB MATLAB Demosaic, RMSE: 1.411