Ankur Kejriwal
Kejria1
400113206

Project 1

Instructions for running the program

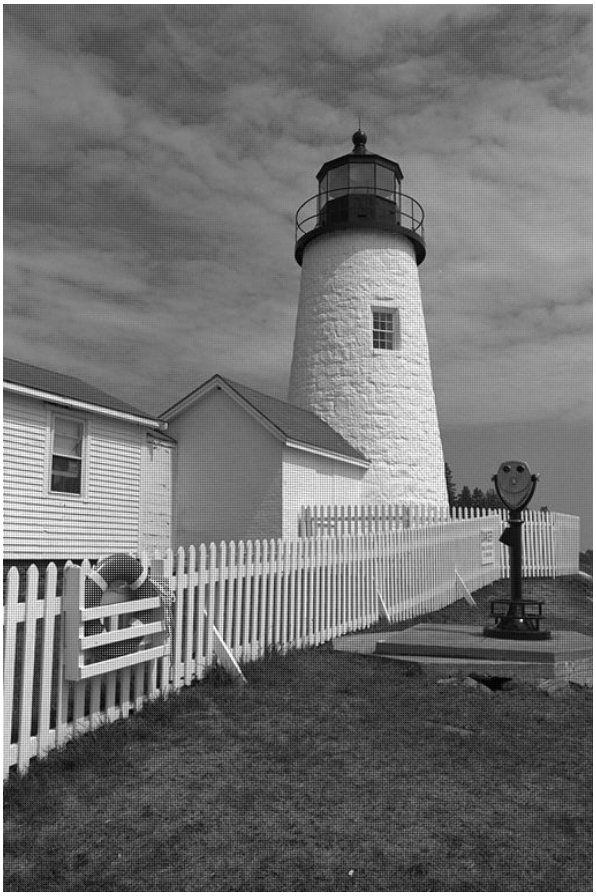
The algorithm is implemented in Matlab with help from the image processing toolkit. The program consists of one file and image and produces a 3 channel Bayer image, 1 Channel Bayer image, and the interpolated image.

The main file is project1.m which contains all the relevant functions and code required to run the project and the image provided is the lighthouse.png image. To change the input image change the filename on line 2. The file outputs three images one of which is the mosaicked image in 3 channel format(mosaic3Channel.png), the other is a 1 channel representation of the same image(mosaick1Channel.png), and the last being the interpolated image (CompleteImage.png)

The mse is outputted to console for both my interpolation implementation and using Matlab's demosaic function and are labeled accordingly.

CFA Image Generation:

Simulation of the camera sensor was performed by downsampling the input image in a similar fashion to the CFA filter on top of conventional cameras. To mosaic the input image I created 2x2 matrices for each channel with a 1 in the position of the sample we want to retain and scaled it up to the size of the input image and multiplied by each channel of the image independently and compressed them to form the Bayer input image(which follows the RGGB pattern) for my interpolation algorithm.



Downsampled Bayer Image

Algorithm Design:

The algorithm that I chose to implement is a gradient-based approach which is based on the algorithm explored by Swirski[2] where the green channel of the image is adaptively interpolated using the horizontal and vertical second derivatives. The green channel in the Bayer pattern contains 2x the samples compared to the red and blue channels so it logically should be the first step in the demosaicing process. Based on the output from the green channel interpolation the red and blue channel was formed. The reason this algorithm was chosen over nonadaptive algorithms is that taking into account the horizontal and vertical derivatives would provide more insight into when we should interpolate locally in the linear direction or in the horizontal direction. The image is symmetrically padded to ensure that we don't index outside of the range of the picture and is trimmed to ensure the Bayer pattern remains.

First I formed a matrix of the Horizontal and Vertical gradients approximated by central difference as shown in the figure below

$$H_{x,y} = \left| \frac{S_{x,y-2} + S_{x,y+2}}{2} - S_{x,y} \right|$$
$$V_{x,y} = \left| \frac{S_{x-2,y} + S_{x+2,y}}{2} - S_{x,y} \right|$$

Figure 1 H and V approximated using central difference[2]

Afterward based on the Horizontal and Vertical 2nd derivative values at the pixel in question the green channel is interpolated as shown in figure 2 below

$$G_{x,y} = \begin{cases} \hat{G}_{x,y} & \text{if } S_{x,y} \text{ is green} \\ \frac{\hat{G}_{x,y-1} + \hat{G}_{x,y+1}}{2} & \text{if } H_{x,y} < V_{x,y} \\ \frac{\hat{G}_{x-1,y} + \hat{G}_{x+1,y}}{2} & \text{if } H_{x,y} > V_{x,y} \\ \frac{\hat{G}_{x,y-1} + \hat{G}_{x,y+1} + \hat{G}_{x-1,y} + \hat{G}_{x+1,y}}{4} & \text{if } H_{x,y} = V_{x,y} \end{cases}$$

Figure 2: Green Channel Interpolation[2]

The remaining red and blue values are then calculated by taking the average of the difference between the green and red and green and blue channels and then adding the interpolated value from the green channel at that point as shown in figure 3

$$R = G + \left(\left(\hat{R} - G \right) * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$
$$B = G + \left(\left(\hat{B} - G \right) * \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

Figure 3: Computation for Red and Blue channels[2]

The 3 2D matrices which hold the interpolated red, green, and blue values are concatenated into one 3D matrix and passed to the image write function to produce the output image. The

matrix convolution seen in the image is a linear interpolation of the color difference that is required to facilitate the interpolation and correction of the red and blue channels based on the interpolated green channel in the previous step. Output images can be seen in the zip folder accompanying this submission.

Error:

The mean squared error for my implementation was calculated using the immse function in Matlab and was found to be **21.9109** and I compared it to the Matlab demosaic function which produced an image with a mse of **27.5216**. Comparing the image produced by Matlab's demosaic function, it is evident that the adaptive approach reduced the zipper effect near the fence region significantly which shows that the algorithm that I implemented reduces interpolation over edges which reduces the resulting zipper artifact that is produced.

Zipper pattern as seen in Matlab's demosaicing

Zipper pattern reduced via adaptive approach

In the 2nd image above you can notice there seems to be an artifact that is produced which is only seen in the adaptive approach. In that region, it seems to be the horizontal and vertical derivatives are changing rapidly so interpolation is happening in both directions in a high frequency in that region which seems to be causing that artifact.

References:

1. I. Pekkucuksen and Y. Altunbasak, "Gradient based threshold free color filter array interpolation," *2010 IEEE International Conference on Image Processing*, Hong Kong, China, 2010, pp. 137-140, doi: 10.1109/ICIP.2010.5654327.

2. Swirski, Leszek. “CFA Interpolation Detection.” (2009). Link: https://www.cl.cam.ac.uk/teaching/0910/R08/work/essay-ls426-cfadetection.pdf