Demosaicking using Adaptive Bilateral Filters

Ekine Akuiyibo
Timothy Atobatele
EE 362, Winter 2006

0. Abstract

Digital cameras sample the continuous spectrum using an array of color filters such that each pixel samples only one color band producing what is commonly referred to as a color mosaic. Demosaicking is then the process of 'filling in' the lost color information to restore the original image. In this project, we explored the novel concept of bilateral filtering [2] and its use for the demosaicking process. We present the results of our experiments.

Figure 1: a) Original Lighthouse Picture, b) Lighthouse Mosaic

1. Introduction

Demosaicking or 'color filter array interpolation' is one of several image processing steps a digital camera performs to provide the user a viewable image. Digital still color cameras produce a color mosaic during image capture as the light sensors sample the image through a color filter array (CFA). There are different CFA's used in different cameras but the most popular till date is the Bayer CFA [1]. The Bayer pattern shown in Figure 2 below typifies the design of most CFA patterns. Luminance (represented by green) occurs at twice the spatial frequency of the chrominance (red and blue), accounting for two facts i) the luminance response of the human vision peaks around the frequency of green light and, ii) the human vision is more sensitive to luminance than chrominance. The challenge becomes the best way to reproduce the original full-color resolution image.

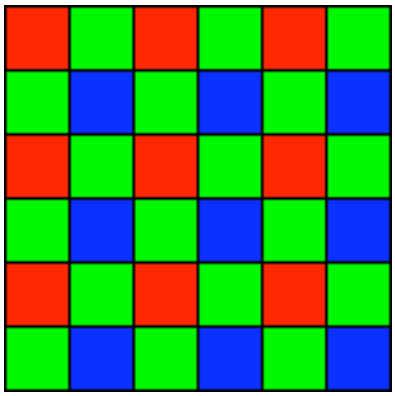


Figure 2: Bayer Color Filter Array

With the advent of cheaper digital camera manufacturing over the last 10 years, research into more efficient ways to demosaic has been a focus of the image processing community. This has resulted in a plethora of techniques being proposed for demosaicking from simple "data replication" techniques to more complex fourier domain algorithms. We will only include references to some of these other methods preferring instead to concentrate on providing background into demosaicking via bilateral filtering.

2. Bilateral Filtering

"Bilateral filtering smoothes images while preserving edges, by means of a nonlinear combination of nearby image values." And even though bilateral filtering is a nonlinear technique it is non-iterative, 'local' and simple. The nonlinearity arises due to the nonlinear relationship of pixel values of an image. Bilateral filters comprise of two component filters: a domain filter and a range filter.

I. Domain: the domain filter component refers to the traditional low-pass filter used to average values of the image that are close in space. Our implementation of the domain filter utilizes a Gaussian blur kernel for filter weights. Figure 3 shows an example 5X5 Gaussian blur kernel.

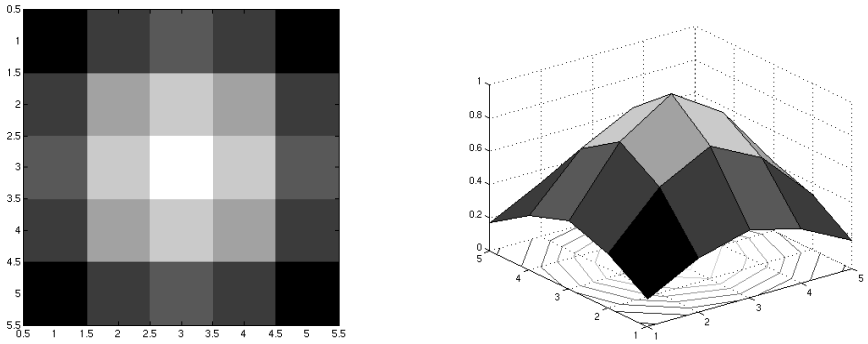


Figure 3: 5X5 Gaussian Blur Example

II. Range Filter: the range component averages pixel values based on similarity (photometric closeness). We say two pixels are similar to each other if their photometric values are close. Unlike the domain component, similarity is dependent on image characteristics. This adaptive quality is illustrated in the figures below.

Figure 4: a) Slanted Line Image, b) Straight Line Image

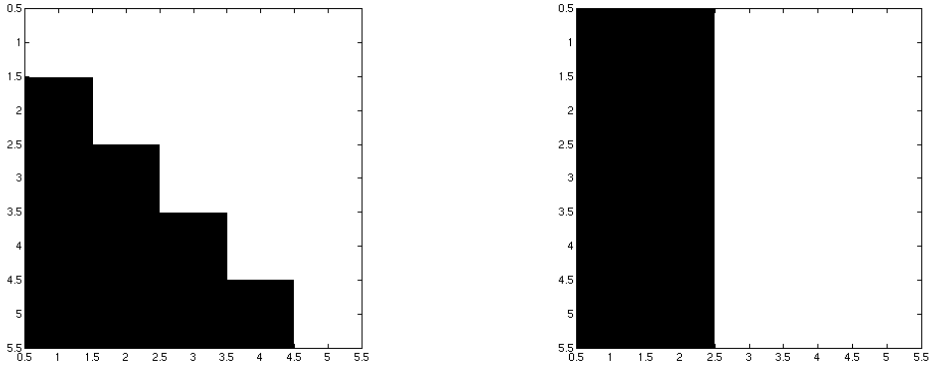


Figure 5: a) 5X5 Slanted Line Kernel, b) 5X5 Straight Line kernel

The Bilateral filter is the resultant product (spatial convolution) of the domain and range filters, which results in an averaging of image pixels based on spatial and photometric closeness. This is the central idea underlying the bilateral filtering interpolation method. Notice that the nonlinearity as a result of pixel values does not necessarily add computational complexity. Figure 4 shows the original images while the corresponding range filter kernels generated along the edge of the image is depicted in Figure 5. Figure 6 below shows the combined Gaussian blur kernel with the slant and straight-line kernels discussed above. As mentioned before the range kernel is generated according to image content and thus the resulting bilateral filter has adaptive weights.

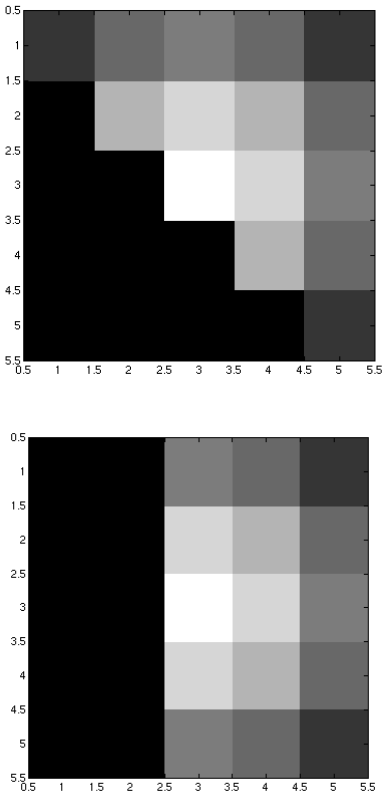


Figure 6: Bilateral Filter kernels for a) Slant line, b) Straight line

Note that range filtering alone does not provide us with the desired averaging [2]. We do not want to average pixels 'close' in value from one corner of the image to the other corner. The domain component thus ensures the averaging is 'local' which is the desired effect we seek.

3. Application to Demosaicking

The principal characteristic of any good demosaicking technique is edge detection. Filtering in general is a fundamental operation that is very well understood: value of the filtered image is some weighted average of nearby image values. The problem is we don't want to average across 'edges' in our filtered image. Thus, we have to find a way to detect edges, and in the case of demosaicking interpolate along the edges and not across them. Bilateral filters provide this capability with the combined domain and range components. Demosaicking via bilateral filtering occurs essentially as a two-step process:

Estimate the bilateral kernel for the desired Image pixel: as we showed in previous examples the bilateral kernel adapts to the image characteristics. Lets say we want to interpolate the missing blue values of a mosaic that utilized the Bayer CFA. Figure 7 shows an m x n array where the blank spaces refer to the missing blue values.

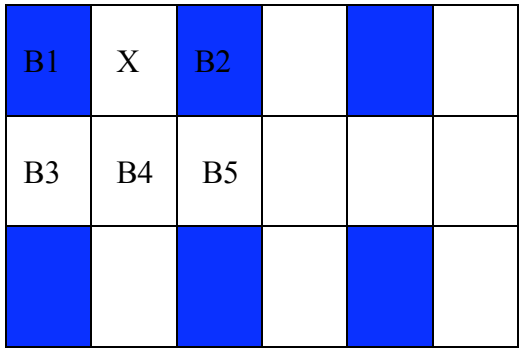


Figure 7: Mosaic showing only blue pixels

To interpolate for a missing blue pixel 'value' say at X, we determine our bilateral kernel (bilateral filter weights) from the surrounding pixels B1-B5. The resulting interpolation thus is the weighted sum of 'blue pixel' weights surrounding pixels used to estimate X. We explain our implementation precisely in the next sections. Current bilateral filter research (for demosaicking) is focused on more efficient penalty functions for weight adaptation.

4. Bilateral Filter Design [3][8]

Reiterating: Bilateral filter adaptively smoothes an input image over neighboring pixels with edge detection capability. Throughout this section the input image is denoted ***f*** while the output image is denoted ***g***.

Consider a pixel of an image with coordinates (x, y) represented as $\vec{z} = \begin{pmatrix} x \\ y \end{pmatrix}$ and a neighboring pixel of the image with coordinates (x+Δx, y+Δy) represented as $\vec{\xi} = \begin{pmatrix} x + \Delta x \\ y + \Delta y \end{pmatrix}$. The contents of the input and output images at the pixel (x, y) are $f(\vec{z})$ and $g(\vec{z})$ respectively.

The output image content at pixel (x, y) is computed as a weighted average of the input image contents of all pixels within a (2W+1)-by- (2W+1) window around the pixel (x, y), where W is predefined. This is expressed mathematically as below

$$g(\vec{z}) = \frac{\sum_{\Delta y=-W}^{W} \sum_{\Delta x=-W}^{W} f(\vec{\xi}) b\left(\vec{z}, \vec{\xi}, f(\vec{z}), f(\vec{\xi})\right)}{\sum_{\Delta y=-W}^{W} \sum_{\Delta x=-W}^{W} b\left(\vec{z}, \vec{\xi}, f(\vec{z}), f(\vec{\xi})\right)} \tag{4.1}$$

where $b\left(\vec{z}, \vec{\xi}, f(\vec{z}), f(\vec{\xi})\right)$ is the bilateral filter kernel used to 'weight' each of the pixels in the averaging process. The bilateral filter kernel is constructed such that the nearby pixels contribute more to the average than the pixels that are father away.

The central idea in the design of any image-smoothing filter is that pixels that are in close geometric proximity have similar contents. Thus, it is assumed safe to average over close pixels. However, this central idea breaks down at the 'edges' of an image. In this context, 'edges' refer to those points on an image where there are discontinuities or sharp contrasts between a pixel's content and its immediate neighbor's. The bilateral filter accounts for the edges by weighting pixels based on their photometric similarity in addition to geometric proximity. Therefore, the bilateral filter kernel is a composite kernel consisting of a domain filter kernel and a range filter kernel. The domain filter kernel weights pixel contents based on geometric proximity to the center pixel while the range filter kernel weights the pixel contents based on photometric similarity to the center pixel. These ideas are expressed mathematically as below

$$b\left(\vec{z},\vec{\xi},f\left(\vec{z}\right),f\left(\vec{\xi}\right)\right)=d\left(\vec{z},\vec{\xi}\right)r\left(f\left(\vec{z}\right),f\left(\vec{\xi}\right)\right) \tag{4.2}$$

where $d\left(\vec{z},\vec{\xi}\right)$ is the domain filter kernel and $r\left(f\left(\vec{z}\right),f\left(\vec{\xi}\right)\right)$ is the range filter kernel.

As mentioned earlier, the domain filter kernel weights pixels based on their geometric proximity to the center pixel while the range filter kernel weights the pixels based on their photometric similarity (proximity) to the center pixel with the near pixels contributing more than the far pixels. The Gaussian function is a natural candidate for implementing such a weighting scheme. The Gaussian has most of its weight at or near the center and exponentially diminishes away from the center. Thus, both the domain filter kernel and the range filter kernel have been designed with the Gaussian as follows

$$d\left(\vec{z},\vec{\xi}\right)=\exp\left(-\frac{s_d\left(\vec{z},\vec{\xi}\right)^2}{2\sigma_d^2}\right) \tag{4.3}$$

$$r\left(f\left(\vec{z}\right),f\left(\vec{\xi}\right)\right)=\exp\left(-\frac{s_r\left(f\left(\vec{z}\right),f\left(\vec{\xi}\right)\right)^2}{2\sigma_r^2}\right) \tag{4.4}$$

The arguments of the Gaussians above express the notion of proximity between the center pixel and its neighbors in the geometric and photometric sense respectively. The terms σ_d^2 and σ_r^2 are the geometric and photometric variances respectively. Roughly speaking, the variances determine how much weight far pixels (geometric or photometric) contribute to the averaging process. The term $s_d\left(\vec{z},\vec{\xi}\right)^2$ is the familiar square of the Euclidean distance between two points in a plane given by

$$s_d\left(\vec{z},\vec{\xi}\right)^2=\left(\vec{z}-\vec{\xi}\right)^T\left(\vec{z}-\vec{\xi}\right) \tag{4.5}$$

The term $s_r\left(f(\vec{z}), f\left(\vec{\xi}\right)\right)^2$ is a measure of the photometric distance between the center pixel and its neighbors. This measure has been defined in more than one way in literature. However, the two definitions that are adopted in this work are the square of the ΔE difference (expressed in Eq. 4.6 below) and the squared difference (SD) in the pixel contents (expressed in Eq. 4.7 below).

$$s_r\left(f(\vec{z}), f\left(\vec{\xi}\right)\right)^2 = \left(f(\vec{z}) - f\left(\vec{\xi}\right)\right)^T \left(f(\vec{z}) - f\left(\vec{\xi}\right)\right) \qquad (4.6)$$

$$s_r\left(f(\vec{z}), f\left(\vec{\xi}\right)\right)^2 = \left(f(\vec{z}) - f\left(\vec{\xi}\right)\right)^2 \qquad (4.7)$$

The ΔE metric is well defined in the CIELAB and HSV color spaces while the SD metric was used in the RGB and YCbCr color spaces in the bilateral filter implementation in this work.

5. Bilateral Filter Implementation (MATLAB) [3][8]

A MATLAB file ($bilateral\_filter3.m$) that implements the above design of the bilateral filter has been written. A sample function call to the bilateral filter file is as below

>> $[g,n] = bilateral\_filter3(f, \sigma_d^2, \sigma_s^2, space1, space2);$

The inputs to the filter are an M-by-N-by-3 RGB image, the geometric and photometric variances, the color space in which the averaging process described in the bilateral filter design section is to take place and the color space in which photometric similarity is defined.

The outputs of the filter are an M-by-N-by-3 RGB image, which is the bilateral-filtered version of the input image and M-by-N-by-3 normalization matrix used in the averaging process but not really useful afterward.

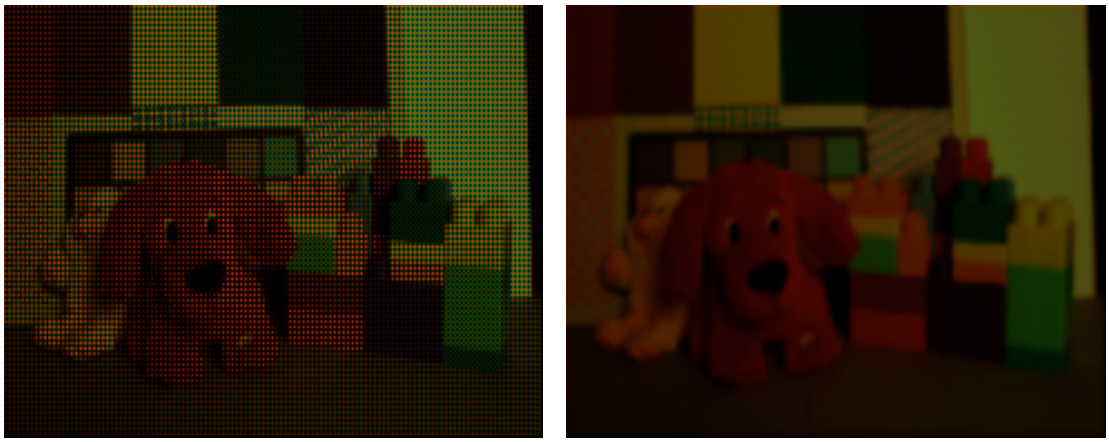
We have attempted to lucidly document the MATLAB code making it easy to use. To see the documentation, you could just type

>>help bilateral_filter3

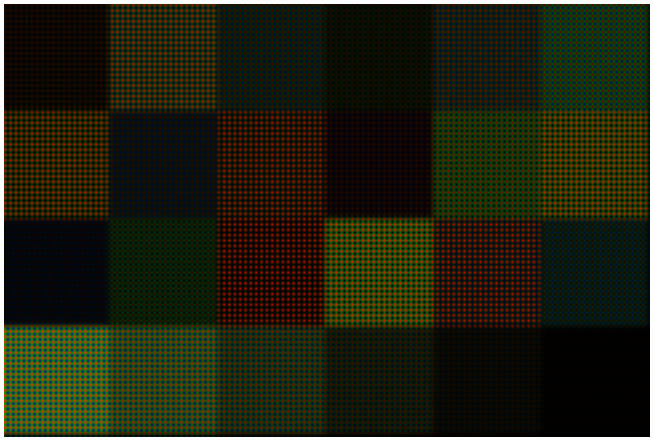
at the MATLAB command prompt.

6. Experiments/Results

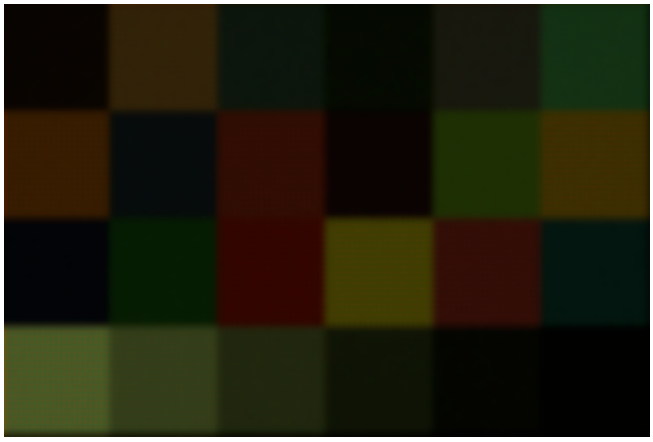
We ran the bilateral filter (bilateral_filter3) on test images to produce a sampling of the results that follow.



Animals: a) Mosaic b) Demosaicked

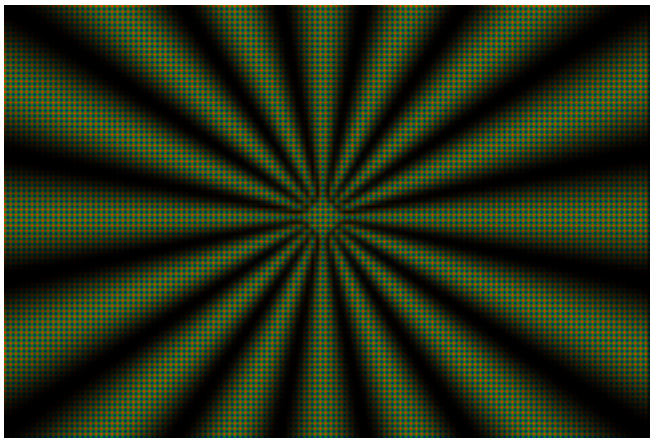


(a)

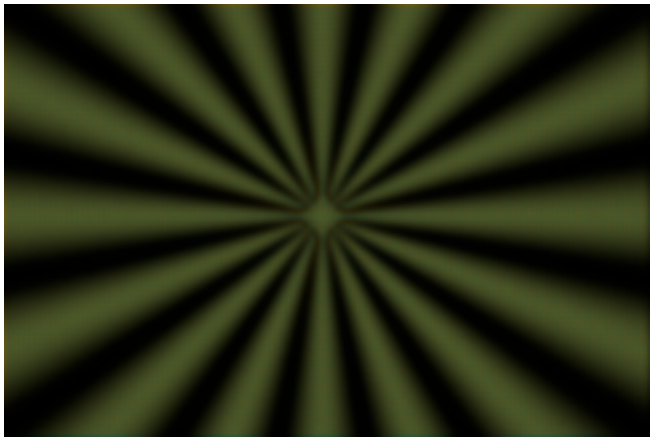


(b)

Macbeth: a) Mosaic, b) Demosaicked

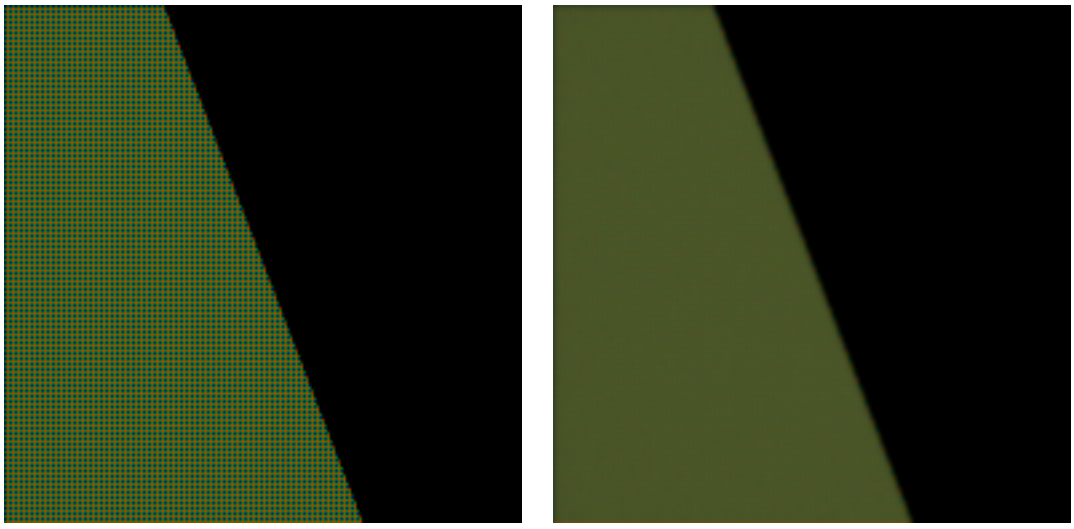


(a)



(b)

Mackay: a) Mosaic, b) Demosaicked



Slanted Bar: a) Mosaic, b) Demosaicked

Lighthouse Image Analysis:

(a)

(b) (c) (d)

Figure 8: a) Original Image, b) Mosaic, c) Bilinear Image, d) Bilateral Image

MSE vs Domain Variance Experiment:

According to our domain filter definition, we expect error of the demosaiced image to increase with increase domain filter variance. Figure 9 shows this trend is true for the example lighthouse image.

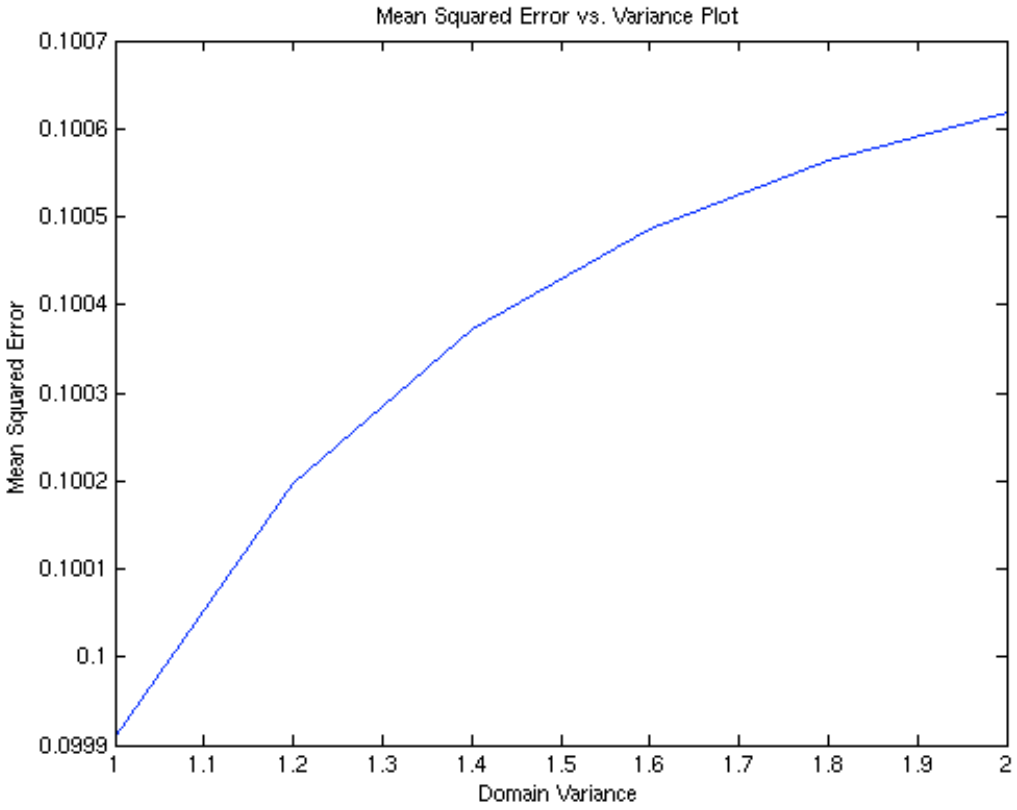


Figure 9: Lighthouse Error Plot

7. De-noising

As a side note, we also like to point out that an added benefit of bilateral filtering is the inherent de-noising capability that occurs due to the averaging process. Since noise in pixel values are mutually less correlated than images values, any averaging process automatically performs a smoothing operation while preserving image values. Due to a lack of time, we did not perform experiments to show this effect although it is well documented in the literature [2].

8. Conclusion

We are happy to show that the relatively simple idea behind the bilateral filtering process is indeed capable of demosaicking. Unfortunately we didn't have enough time during this project to quantify our filter's performance, do a simple PSNR evaluation of the 4 different kernels we implemented or do a comparative study with other standard techniques available today. Nevertheless, we hope that we have shown the exceptional versatility of the concept of bilateral filtering. A natural follow on could be parameter sensitivity analysis testing to determine just how effective bilateral filters can be for demosaicking.

ACKNOWLEDGEMENTS

We would like to thank Prof. Brian Wandell and Greg Ng for their valuable contributions to this project.

APPENDIX I

1. Bilateral Filter: bilateral_filter3.m
2. Experiment scripts: filter_tests.m
3. Images: data

APPENDIX II

Ekine Akuiyibo

1. Bilateral Filter and Demosaicking Research
2. Presentation: Making Slides, Rehearsal
3. Matlab: Example Scripts / Analysis
4. Project write-up

Timothy Atobatele

1. Bilateral Filter and Demosaicking Research
2. Presentation: Making Slides, Rehearsal
3. Matlab: Bilateral Filter Implementation / Analysis
4. Project write-up

REFERENCES

[1] BE Bayer, "Color imaging array," US **Patent**, no.3 971 065, 1976

[2] R. Ramanath and W. E. Snyder, "Demosaicking as a Bilateral Filtering Process," *Proc. SPIE, Image Processing* **4667**, pp 236-244, 2002

[3] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color images," Sixth *International Conference on Computer Vision, 1998.*

[4] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," IEEE Trans., PAMI-12 (7): 629–639, 1990.

[5] M. Gupta and T Chen, "Vector Color Filter Array Demosaicing," *Proc. SPIE, Sensors and Camera Systems* **4306**, pp 374-382, 2001.

[6] EE 362: Tutorial 2: Color Matching, Tutorial 3: Color Metrics, Winter, 2006.

[7] N. Kehtarnavaz, H. Oh, and Y. Yoo, "Color filter array interpolation using color correlations and directional derivatives" *JEI,* 12(4), 621–632, 2003.

[8] D. Barash, "A Fundamental Relationship between Bilateral Filtering, Adaptive Smoothing and the Nonlinear Diffusion Equation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(6), pp 1-5, 2002.