Demosaicing using Nonlinear Anisotropic Diffusion

Bachelor Thesis

Timo Florian Adam
tfadam@googlemail.com

Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Mathematical Image Analysis Group
Saarbrücken, Germany, June 26, 2011

Supervisor:

Professor Joachim Weickert

Reviewers:

Professor Joachim Weickert

Assistant Professor Michael Breuß

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, June 26, 2011

Signature

Abstract

In this work, we are going to explore new possibilities of demosaicing algorithms for Bayer-Pattern images, using edge- and coherence enhancing anisotropic diffusion. Demosaicing is simply a specialized kind of interpolation, which is applied to so called color-filtered images (images with missing color values in a fixed pattern). Diffusion as an image processing method is closely related to the physical process of diffusion. In image processing, diffusion algorithms equilibrate concentration differences of color-values across the image plane. In the case of demosaicing, one wants to fill the gaps on the image grid with the aid of this process. However, to preserve the structure of the image, one has to prevent propagation of this flux across edges, thus a good edge estimation is substantial for the diffusion algorithm to deliver adequate results. By exploiting different channels from various color models, we will try to achieve a stable edge detection.

In the first section, we will give a short introduction to the basics of digital image capturing and the kind of post-processing that is necessary to represent images digitally. We are also going to take a look at the challenges, that arise with such methods.

Next we will illustrate the concept of diffusion from an image processing point of view. This will lead us to the structure tensor, a tool for edge description. We will examine several channels from different color models to find a suitable edge detector, to control the diffusion process.

The next section will introduce ways of measuring the quality of demosaicing algorithms. This will allow us to judge the quality of our method compared to some traditional demosaicing algorithms and some commercial applications. We will present the results obtained with this method.

In the last section we are going to draw conclusions of diffusion-based demosaicing in general, look at applicability to other color-filter-patterns and give an outlook at what can be expected by such algorithms in the future.

Appendix A gives a short introduction to already available linear and nonlinear demosaicing algorithms.

The diffusion algorithms used in this paper were developed by the Mathematical Image Analysis Chair at Saarland University. This paper is partially based on the previous work on this topic from [Rus10].

Contents

List of Figures

List of Tables

1 Introduction

In this section, we will give a short overview of how digital cameras record images and what kind of post-processing has to be performed to obtain the final full-color image.

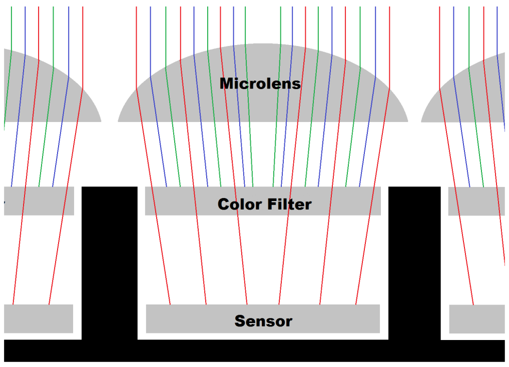
1.1 Representation of Digital Images

Digital images are usually represented using a two-dimensional grid of single picture elements, called *pixels*. Each pixel comprises data that are necessary for displaying an image (such as color-, intensity-, lightness- or luminosity-values, ...). Digital photo and video cameras employ the same principle to record images. Most digital image sensors consist of a rectangular grid of sensor elements, also called *sensels*. Each sensor element responds to incoming photons. An electric charge is created which is then transformed into electronic signals. Those signals are digitized and stored. Since the sensor is not able to differentiate between wavelengths of incoming light, we only end up with a monochrome image, which means that no color information is available at first. The incoming light has to be color-filtered before it hits the sensor, to obtain color information. This process of capturing images digitally from an engineering standpoint is explained in [Nak06].

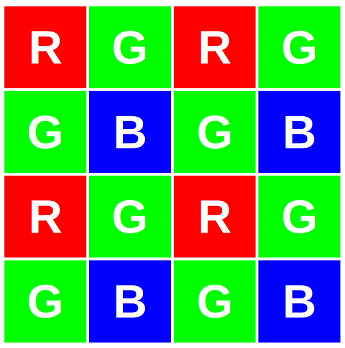
One of the most common representations for displaying color images is the *RGB color model*. In this format, every pixel contains three color-coded values (for red, green and blue). All other colors in this color-space are derived from combinations of those three values. Since a color-filtered sensor only records one color information per pixel, this means that two color values are therefore missing for a proper representation in this format. In some camera designs, this problem is approached by installing three different sensors in a camera and fork the lightwaves through a prism in the camera lens. This solution is of course very expensive and sophisticated, and therefore only used in some professional environments. Another solution is pixel-binning, in which each pixel is merged from several subpixels. This would of course decrease the resolution three or four times or make sensors and camera optics much bigger and more expensive.

1.2 The Bayer Color Filter Array

Another more feasible approach is to use only one sensor with a combination of different color-filtered sensor elements. These color filters have to be distributed in an even pattern across the whole sensor. An additional array of tiny microlenses in front of the filter-array concentrates the light and amplifies the electric charge. Such a construction is referred to as *Color Filter Array* (CFA). Figure 1a illustrates this concept. Now, each sensel only receives light from a certain frequency band of the visible spectrum, which enables the distinction of colors. One of the most common filter patterns is the *Bayer Pattern* as seen in figure 1b.This filter spreads the three different color values evenly among the whole grid. The number of green pixels is twice the number of red and blue values. Therefore only half as many green values have to be interpolated, which in effect means that the green color will be represented more accurately in the final image. The reason for this choice is that the cone-cells in the human eye are much better fitted to distinguish color-nuances of light in the green wavelength spectrum.



(a) Array of microlenses and color filters in front of the sensor



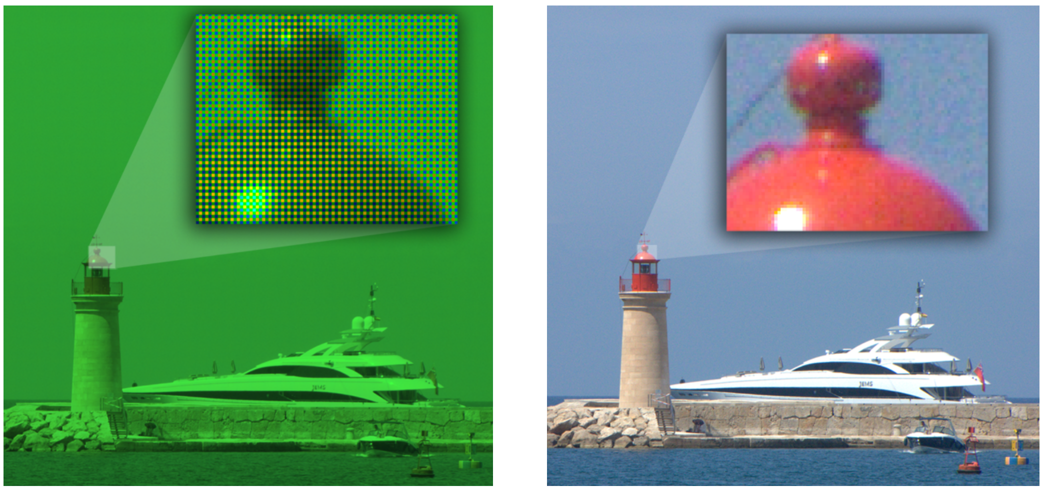
(b) The Bayer Pattern

Figure 1: Bayer Color Filter Array

This has, as already mentioned, the unwanted effect that actually two thirds of the image data (two of the three color-values in each pixel) are missing. To obtain color information, which is necessary to correctly[1] display an image, the missing data have to be recovered by interpolation. This kind of interpolation is commonly referred to as *demosaicing* (also found in literature: demosaicking) and is illustrated in figure 2.

The Bayer Pattern was introduced by Bryce E. Bayer from Eastman Kodak Company in 1976 [Bay75] and can be found in virtually every camera on the market today (Webcams, Smartphones, Digital Still- and Video Cameras (Consumer and Professional), home, scientific and industrial appliances, . . .). This work will focus mostly on demosaicing techniques for Bayer CFAs.

[1]in our sense, correctly means that we want the image to match the original scene as closely as possible



(a) Bayer Pattern image before demosaicing (the two non-available values in each pixel are set to 0)

(b) Full-color image after interpolation

Figure 2: Demosaicing Process

Applicability to other color-filter-designs will also be discussed briefly.

1.3 Shortcomings and Problems

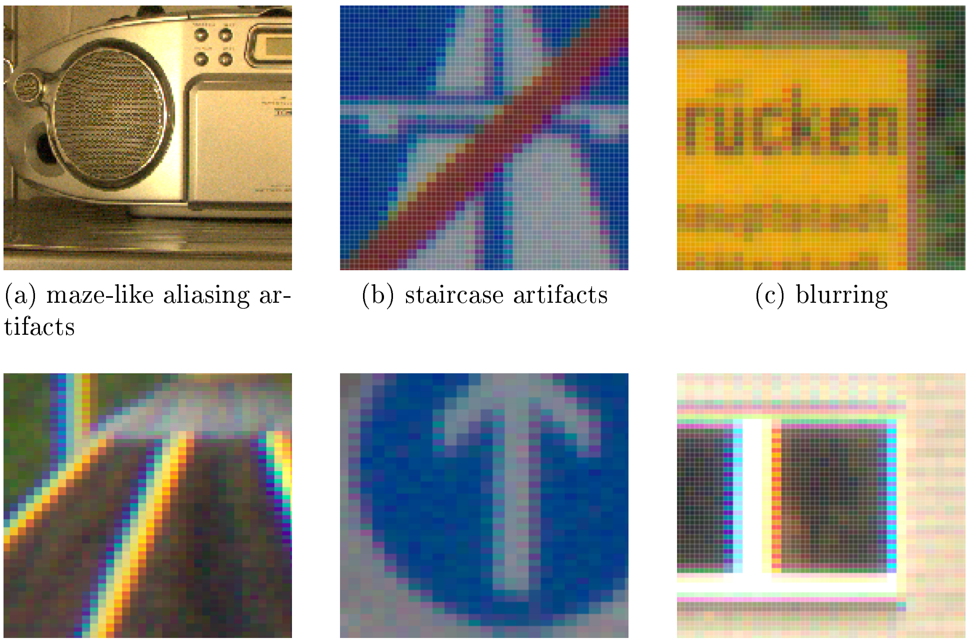
Since interpolation is the process of reconstructing missing data points in between a known set of data, it always involves a certain degree of guessing. The main criterion for assessing the quality of a demosaicing alogrithm is the ability to reconstruct the missing color values as closely as possible to the way the camera would have recorded them in each sensor element. In addition to this lack of precision, demosaicing can even introduce new artifacts if not treated properly.

1.3.1 Blurring

Many demosaicing algorithms rely on some type of averaging across a certain neighborhood. Averaging has the same effect as applying a low-pass filter. This means that high frequency details (which provide important edge information) can get lost. On a low-pixel-level, the image appears *blurred* as seen in figure 3c.

1.3.2 False-Color and Water-Color Effect

The green channel of the Bayer Pattern is different from the red or blue channel in that it has twice as many pixels. This difference in frequency causes color information to be scattered unevenly along edges. These artifacts are referred to as *false-color artifacts* (yellow- or green-blue lines around edges) or so called *water-color effects* (when edges appear as if they were observed through water). Since all colors of the RGB color-space have to be assembled from only three color values, different frequencies in the color channels can end up in wrong chrominance information and new colors can inadvertently pop up. This effect is illustrated in figure 3d and can also be seen in 3f



(a) maze-like aliasing artifacts (b) staircase artifacts (c) blurring

(d) false-color artifacts (e) isolated dots (f) zippering artifacts

Figure 3: Demosaicing artifacts

1.3.3 Aliasing

Another unwanted effect that occurs in interpolation is aliasing. According to Shannon's Sampling theorem [Sha49], if the sampling frequency is not at least twice as high as the image frequency, *aliasing artifacts* can arise. Since in our case, the sampling frequency of the interpolation algorithm is restricted to two times the pixel size (caused by the nature of the Bayer Pattern), aliasing can hardly be prevented. Maze or pattern like structures as seen in in figure 3a are the result. In some cases, even riddle-like patterns can arise. The only remedy is to lower the image frequency by applying a low-pass filter, which however leads to blurring as explained in 1.3.1.

1.3.4 Zippering and Strips

Because of the characteristics of the Bayer Pattern, adjacent pixels have to be treated differently by an algorithm. At otherwise straight edges in horizontal and vertical direction, this leads to an effect called *zippering*, seen in 3f. Diagonal edges also suffer from these kinds of artifacts called *strips* as seen in 3b.

1.3.5 Isolated Dots

Even with digital cameras, the recording of images still remains an analogue process, as mentioned in section 1.1. Due to the physical characteristics of the image sensor, there is always a certain amount of noise present (such as photon shot noise, thermal noise, analog to digital conversion, ...). If not handled carefully, noise in an otherwise unifrom area can occasionally be amplified by an interpolation algorithm as seen in 3e. Diffusion-based algorithms are very capable of dealing with noise, as explained later in section 2.

Thin lines can also cause such dots in locations where they intersect, since edge information may be misleading or blurred in such spots.

2 Demosaicing using Anisotropic Diffusion

2.1 Basic Idea of Diffusion-Based Demosaicing

Diffusion in image processing originates from the idea of the physical process of diffusion. Diffusion is a process in which concentration differences in a body are equilibrated. The same principal can be applied in image processing, where instead of the dispersion of matter in a physical body, the color values are dispersed across the image plane. An important characteristic of diffusion is that no mass is created or destroyed. This guarantees us that the average grayvalues will remain fairly stable during this process. The structure tensor is a tool that allows us to steer this process in certain directions to prevent blurring across edges, as explained below. The process of diffusion is illustrated in figure 4b.

Our demosaicing algorithm includes the following steps:

- Pre-interpolation using a simple and efficient algorithm (such as bilinear interpolation)
- converting the image to a more suitable representation for edge-extraction
- computing the structure tensor on the basis of the data from the preceeding step
- computing the diffusion tensor, based on the structural information from the preceeding step
- perform diffusion iteratively on the image

A comprehensible introduction to anisotropic diffusion (and partial differential equations in general) in the context of image processing is given by [Wei98].

2.2 Pre-Interpolation

Our diffusion algorithm has to be initiated with a full color image to start. For this purpose, we will use a simple and efficient linear interpolation method. Of course this process will introduce unwanted artifacts, which will be propagated by the diffusion algorithm. Unfortunately this is inevitable. In any case, based on its characteristics, the diffusion algorithm should be able to handle such artifacts pretty well.

As pre-interpolation algorithm we choose High Quality Linear Interpolation (HQLI) in this paper, which is presented in Appendix A.3. HQLI uses

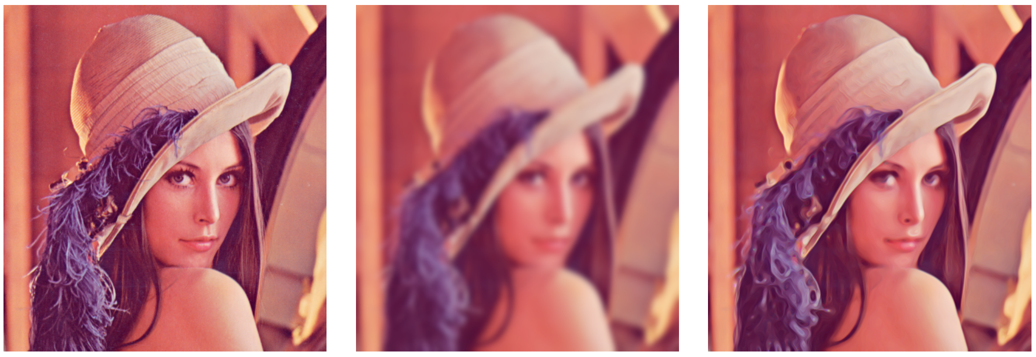

(a) Original image (b) Result of uncontrolled diffusion (c) Diffusion with structure tensor

Figure 4: Diffusion result after 60 iterations

linear $5 * 5$ stencils to interpolate the missing pixel values. In our own testing, HQLI resulted in an almost 5 dB increase in peak signal-to-noise ratio (explained in 3.1.1) and a 1.475 % gain in structral similarity (explained in 3.1.2) to the original image compared to bilinear interpolation[2]. HQLI also offered a much improved visual quality compared to bilinear interpolation (high frequency structures don't look as washed-out as they do in bilinear interpolation), with only an 70 - 80% increase in computational effort. The results are almost on par with nonlinear algorithms, which should suffice for our purposes.

2.3 The Structure Tensor

The *structure tensor* (also called second moment matrix, or interest operator) was first introduced by [FG87]. It is a matrix that contains the calculated partial derivatives of the underlying image and therefore contains important edge information. To prevent noise from distorting edge information, the original image data from which the structure tensor is derived is low-pass-filtered with a gaussian kernel of size σ, in this case called the noise scale. We achieved the best results with $\sigma = 0.4$.

An important advantage of the structure tensor, compared to other edge detectors is that it gives us information about direction and strength (gradient and magnitude) of edges and can prevent cancellation effects, when gradients of opposite direction coincide.

The structure tensor is defined as a gaussian-filtered matrix representa-

[2]tested on a set of very popular demosaicing test images provided by Eastman Kodak Company and on a random set of high-resolution photos

tion of the gradients in horizontal and vertical direction:

$$J_\rho(\nabla u) := K_\rho * (\nabla u \nabla u^T) = \begin{pmatrix} K_\rho * \partial_x f_{i,j}^2 & K_\rho * (\partial_x f_{i,j} \partial_y f_{i,j}) \\ K_\rho * (\partial_x f_{i,j} \partial_y f_{i,j}) & K_\rho * \partial_y f_{i,j}^2 \end{pmatrix}$$

The gradients in the above formula can be calculated as follows:

$$\partial_x f_{i,j} = \frac{1}{2} (f_{i+1,j} - f_{i,j} + f_{i+1,j+1} - f_{i,j+1})$$
$$\partial_y f_{i,j} = \frac{1}{2} (f_{i,j+1} - f_{i,j} + f_{i+1,j+1} - f_{i+1,j})$$

The eigenvalues of this $2 * 2$ Matrix yield important edge information. By manipulating the eigenvalues of this matrix, we can steer the diffusion process. If both eigenvalues are zero, this area of the image is homogenous in grayvalues. If only one eigenvalue is zero, we have a straight edge, if both eigenvalues are larger than zero, this hints at a corner. The squared difference between the two eigenvalues can be interpreted as a measure of anisotropy. These kinds of algorithms were first introduced by Perona and Malik [PM87][PM88]. The subsequent work of several researchers, including [WRV98] has led to notable improvements in this field.

Afterwards, the structure tensor itself can be low-pass filtered with a gaussian kernel of size ρ, in this case called the integration scale. This causes the second eigenvalue to attain positive values and creates flow like structures across edges later in the diffusion process. In our scenario, we are mainly interested in enhancing and preserving edges, so the integration scale ρ should be kept below 0.5.

2.4 Exploring different Color Model Channels for Edge Detection

In the following, we will study various data channels of different color-models to find a good basis for the computation of the structure tensor.

2.4.1 Intensity Channel

Simply adding all the single channels for edge detection is problematic, since gradients of same orientation but opposite direction cancel out, and the structural information gets corrupted. This channel still achieves good results, as seen in table 1.

$$\text{Intensity} = \frac{1}{3} * (R + G + B)$$

Table 1: Comparison of different channels for the structure tensor

Channel	Average	Red	Green	Blue
Intensity	35.8369	34.8945	39.6598	34.6172
Luma Y'_{601}	35.8700	34.8865	39.3159	34.8172
Luma Y'_{709}	35.8678	34.9313	39.1535	34.8309
Lightness	35.4080	34.2749	38.7577	34.5289
$Y'_{0.25,0.6,0.15}$	35.8775	34.9361	39.3320	34.7827
$Y'_{0.25,0.5,0.25}$	35.8719	34.9585	39.4764	34.6986
Green	35.7991	34.9765	38.7415	34.7951
Hue	30.3577	29.8487	31.3446	30.0561
Saturation	32.2135	31.6898	34.8076	31.0513

2.4.2 Lightness Channel

Because of the alternating pattern of the Bayer-CFA, *lightness* information is not very useful for edge detection, since it leads to a zippering effect in the structure tensor itself.

$$\text{Lightness} = \frac{1}{2} * (\text{Maximum}(R, G, B) + \text{Minimum}(R, G, B))$$

2.4.3 Luma Channel

Luma is the weighted average of gamma corrected RGB values. It corresponds more to human perception, since it shifts the weight more towards the green channel. In our tests, the luma channels achieved the best results. This outcome was expected, since the green values are also represented more accurate in the Bayer pattern, as already explained in section 1.2. Several definitions of Luma exist. We could improve the results with a slightly adjusted combination of the weights, as seen in table 1. The different luma channels are defined as follows:

$$Y'_{601} = 0.3 * R + 0.59 * G + 0.11 * B$$

$$Y'_{709} = 0.21 * R + 0.72 * G + 0.07 * B$$

$$Y'_{0.25,0.6,0.15} = 0.25 * R + 0.6 * G + 0.15 * B$$

$$Y'_{0.25,0.5,0.25} = 0.25 * R + 0.5 * G + 0.25 * B$$

2.4.4 Hue and Saturation Channels

As expected *hue* and *saturation* channels are of little use for edge detection, since they don't contain enough structural information. Uniform shapes in an image usually only vary in lightness and do not contain noticable changes in hue, but this information is too coarse to be of much use in this context.

2.4.5 Green Channel

The *green channel* showed acceptable results, but could not exceed luma or intensity channels. Neglecting red and blue in the structure tensor completely is therefore not recommended.

2.5 Diffusion

The diffusion process is described by the so called *diffusion equation* (also known as *heat equation* in physics):

$$\partial_t u = \operatorname{div}(D\nabla u) \tag{1}$$

In this formula, t is just the time parameter, div is the divergence in form of a vector field. The flux $D\nabla u$ consists of ∇u, the concentration gradient and D, the diffusivity. To transform the diffusivity in a form, so that we can apply it to a two-dimensional image grid, we have to use a standard discretization method as further explained in [Wei98]. The resulting matrix representation is called the *diffusion tensor*. The diffusion tensor is an evolving mask that adapts itself to the local image structure after each step of the diffusion process. This allows us to control the diffusion process, by steering it towards interesting image features, in our case edges, as shown in figure 4c. The previously available color-values from the Bayer Pattern are of course not altered during this procedure.

3 Results

3.1 Quality Measures

3.1.1 Peak Signal-to-Noise Ratio

The *peak signal-to-noise ratio* (PSNR, also SNR) is one of the most widely used quality measures in image processing and -reconstruction. The PSNR is calculated as follows:

Let I be the original image that is used as reference and let K be the interpolated image. Let m and n be the width and height of the image.

First, the *mean squared error* term (MSE) is calculated:

$$\mathrm{MSE} = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2}{m * n}$$

Let MAX be the maximum possible value. Then the peak signal-to-noise ratio is defined as follows:

$$\mathrm{PSNR} = 10 * \log_{10} \left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}} \right)$$

The unit of PSNR is deciBel. The higher the value, the higher the similarity between the two images. If the two images are identical, the PSNR is undefined (division with zero in MSE term) and usually set to the maximum value.

3.1.2 Structural Similarity

The *structural similarity* (SSIM) is a measure for the assessment of similarities between images, proposed by [WBSS04]. Compared to PSNR, the strucutral similarity is supposed to be more consistent with human perception of image quality. The structural similarity takes three different weighted components into account:

The luminance term:

$$l(x,y) = \frac{2\mu_x \mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

The contrast term:

$$c(x,y) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

The structural comparison:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{(\sigma_x \sigma_y + C_3)}$$

Some of the terms cancel out, so that structural similarity is defined as:

$$\mathrm{SSIM} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}$$

where

$$C_1 = (K_1 L)^2$$

and

$$C_2 = (K_2 L)^2$$

with $K_1 = 0.01$, $K_2 = 0.03$ as weights and L as the maximum value ($2^{bitrange} - 1$), which in case of 8 Bit images is 255. μ is the mean value

$$\mu_x = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij}$$

σ_x the standard deviation, is defined as

$$\sigma_x = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (x_{ij} - \mu_x)^2$$

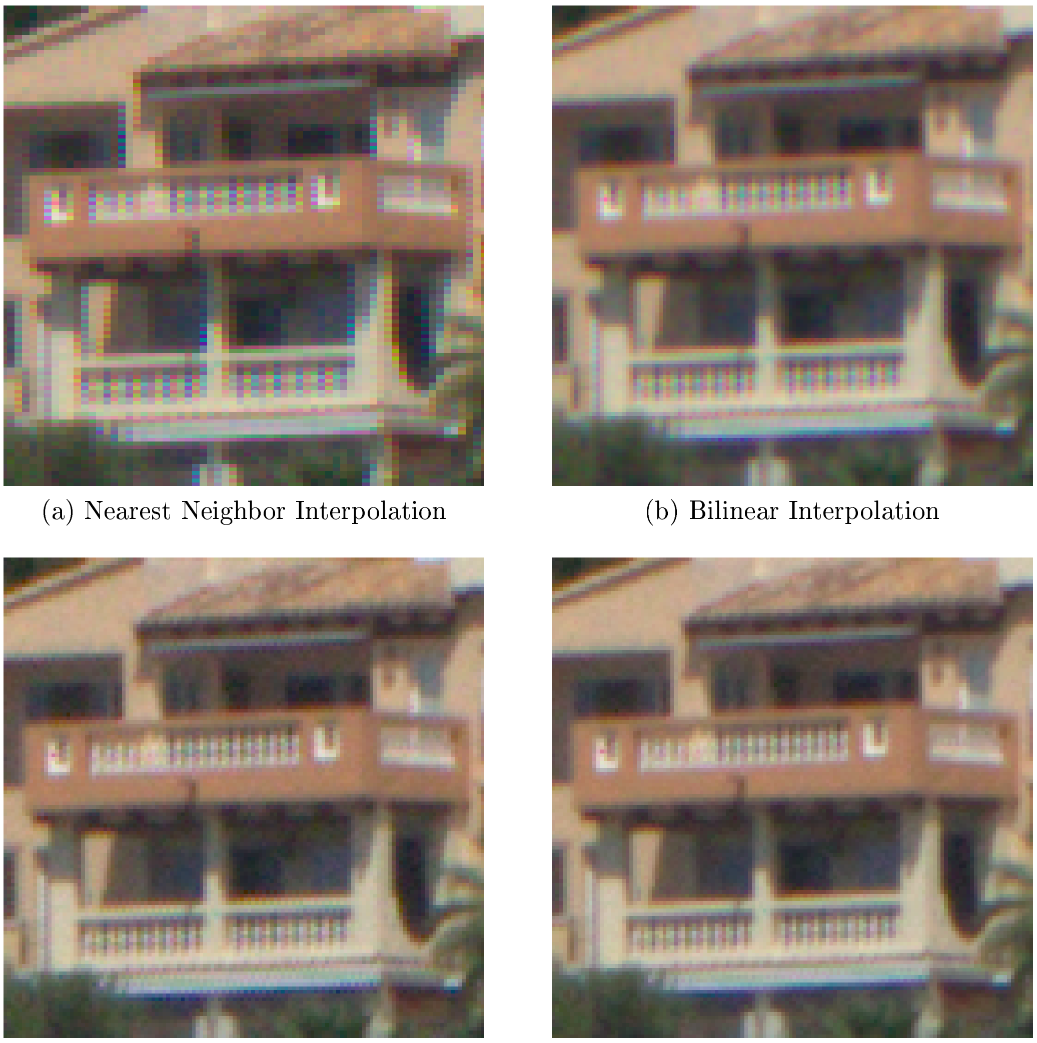
σ_{xy} is the covariance

$$\sigma_{xy} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (x_{ij} - \mu_x)(y_{ij} - \mu_y)$$

The structural similarity can attain values in the range of -1 to 1 It is exactly 1 if the two images are identical.

3.2 Comparison to other Demosaicing Algorithms

In this section, we are going to compare our algorithm to the demosaicing algorithms presented in Appendix A. Nearest Neighbor Interpolation is, as expected, inferior to all other algorithms in every sense. Zippering and false-color artifacts in the balcony on picture 5a are clearly visible. All edges look heavily pixelated. Bilinear interpolation suppresses most of the pixel level artifacts of nearest neighbor interpolation at the expense of losing high-frequency details. The overall image looks blurred as seen in figure 5b.



(a) Nearest Neighbor Interpolation

(b) Bilinear Interpolation

(c) High Quality Linear Interpolation

(d) Diffusion-based algorithm

Figure 5: Comparison to other algorithms

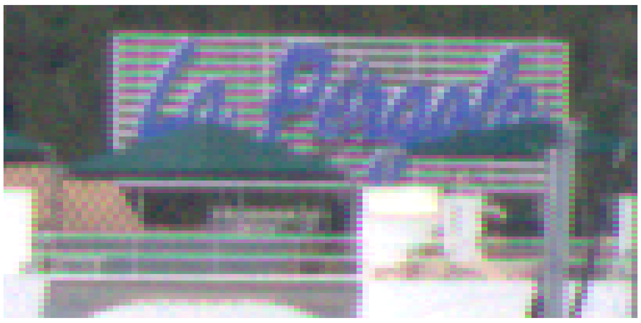
High quality linear interpolation offers a good tradeoff between conservation of high-frequency details and the prevention of edge destruction. The algorithm handles zippering and aliasing effects pretty well. Although, aliasing and slight zippering artifacts are still present in the area below the balcony and the marquee in the upper part of the image in figure 5c. The advantage of diffusion based algorithms are clearly visible. The image doesn't noticably lose detail, but the remaining strips, aliasing and zippering artifacts are almost completely eliminated after diffusion. The PSNR validates the visual impressions of figure 5d. The average PSNR values in table 2 are based on a measurement of 24 popular demosaicing test images from Kodak (figures 15, 15 and 15. The original images were taken on film and digitally scanned.

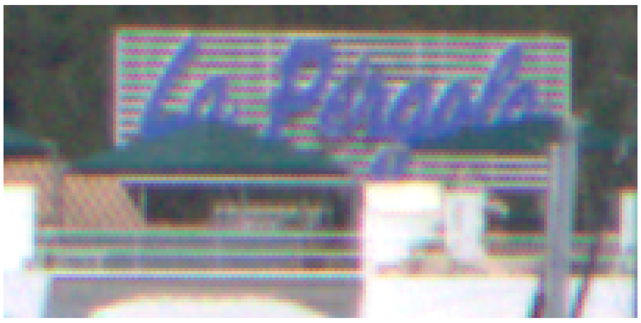
Table 2: Overall comparison of results, all images combined, SNR values in dB

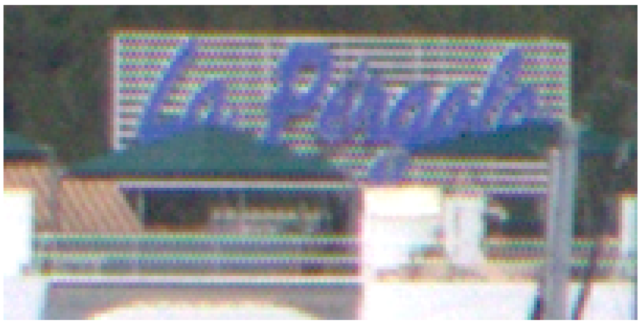
Channel	Near Nbr	Bilinear	HQLI	VNG	Diffusion
	Peak Signal-to-Noise Ratio				
Red	25.19	26.19	27.19	28.19	34.96
Green	28.71	32.65	37.80	39.48	39.47
Blue	25.35	28.66	33.41	33.03	34.70
Average	26.12	29.46	34.47	34.20	35.87
	Structural Similarity				
Red	94.66%	97.26%	99.24%	99.06%	99.45%
Green	97.58%	99.01%	99.71%	99.81%	99.80%
Blue	94.19%	97.11%	99.10%	98.98%	99.33%
Average	95.65%	97.90%	99.38%	99.32%	99.55%

3.3 Comparison to commercial RAW processing applications

Apart from the in-camera processing of color filtered images, many camera manufacturers also offer solutions to save the data in so called *RAW format* and postpone the demosaicing to the postprocessing stage on a desktop computer. Since most commercial software companies only distribute closed source applications and hardly ever provide some insight to the algorithms used, we are only able to compare the resulting images. For comparison purposes, we used two of the most widely used post-processing applications for RAW images, “Canon Digital Photo Professional” and “Adobe Photo RAW for Adobe Photoshop”. In this test scenario, a SNR comparison to an original image is of course not available, since we use actual RAW images from a


(a) Nearest Neighbor Interpolation


(b) Bilinear Interpolation


(c) High Quality Linear Interpolation

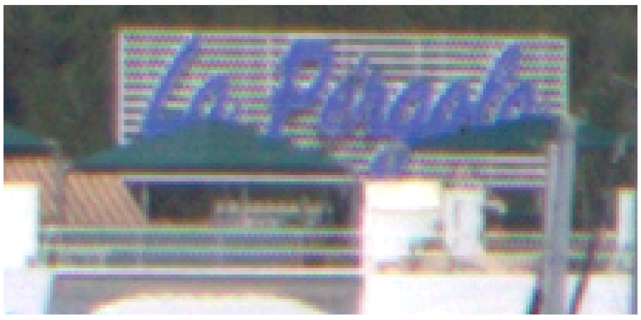
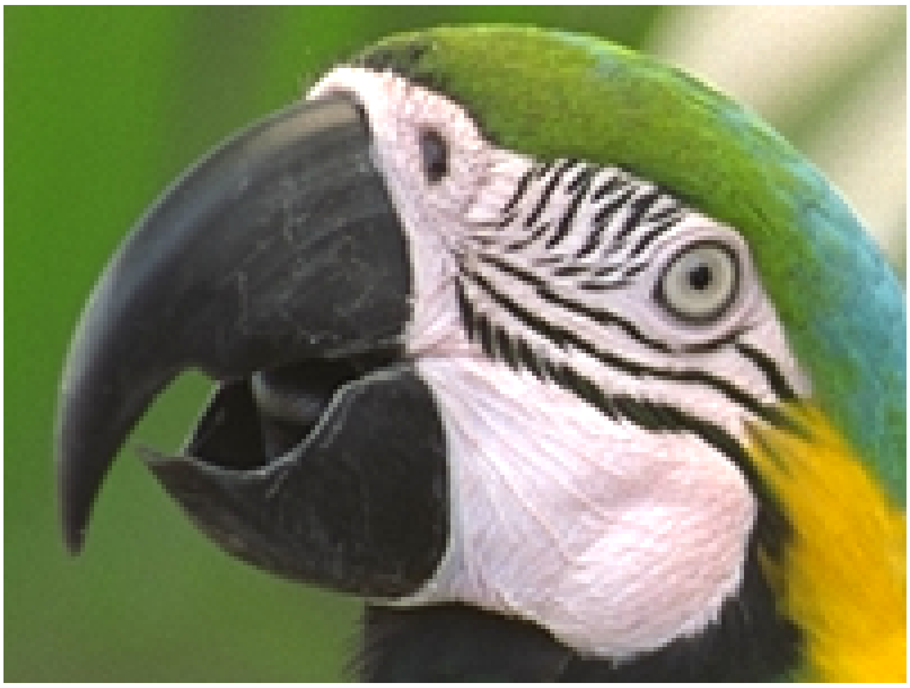

(d) Diffusion - based algorithm

Figure 6: Comparison to other algorithms


(a) Original full-color image

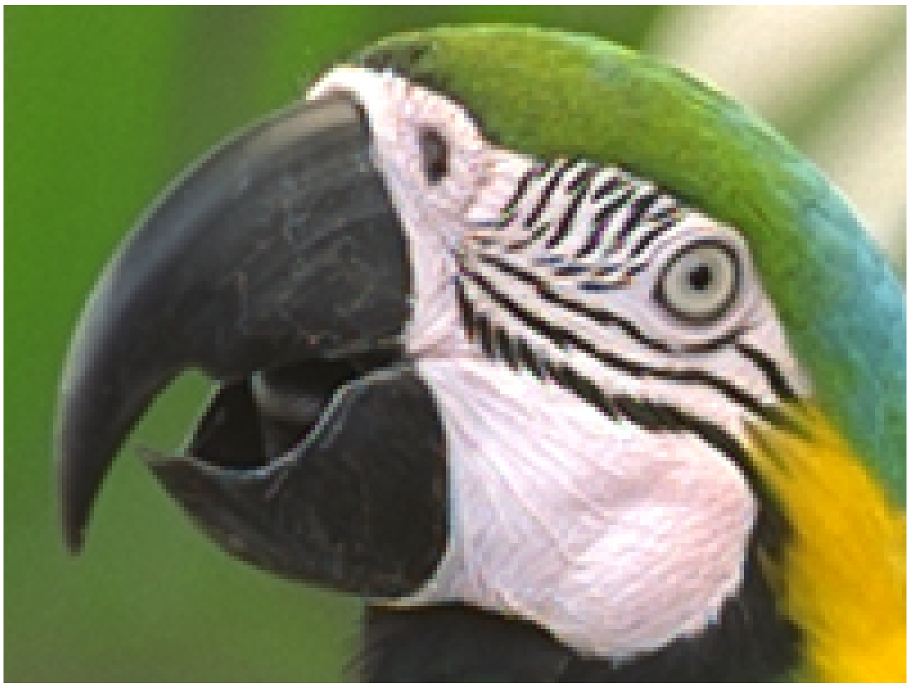
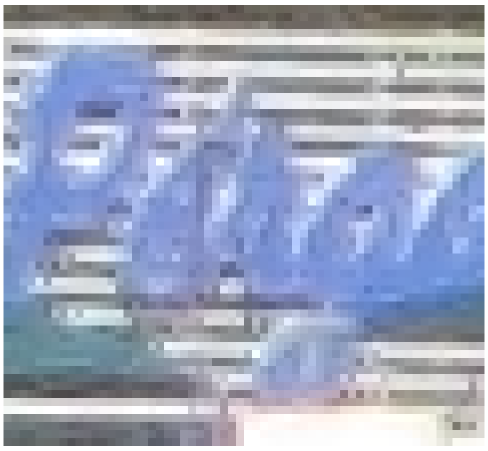
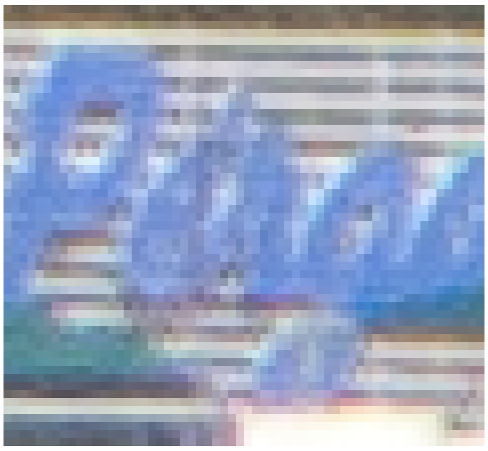
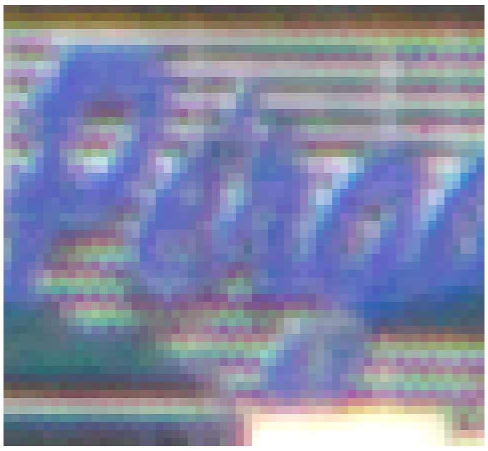

(b) Diffusion-based Demosaicing result

Figure 7: Result of diffusion-based demosaicing algorithm

digital camera instead of scanned full color images.


(a) Canon - Digital Photo Professional


(b) Adobe Photo RAW


(c) Diffusion-based algorithm

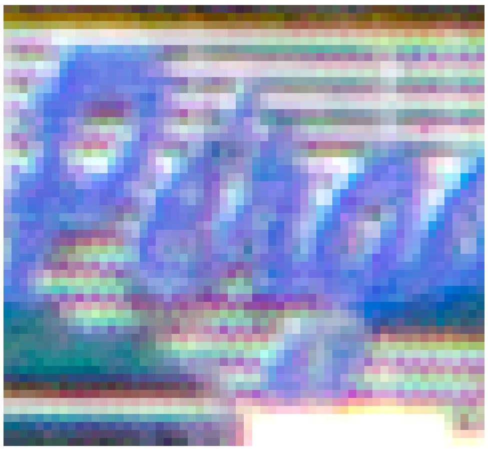

(d) Level-adjusted diffusion result

Figure 8: Comparison to commerical software

It is clearly visible that both commercial algorithms apply heavy postprocessing to the image (including color and gamma correction). This kind of postprocessing leads to an improved perception of sharpness, but also introduces new artifacts and goes to the expense of image quality and precision. Color hue is artificially changed to make the images look more natural. In all three algorithms, zippering artifacts are almost nonexistent. False-color artifacts are also handled very well by all three algorithms. The Canon algorithm especially suffers from isolated dots, since it uses extensive image sharpening. Commercial algorithms are geared towards offering the best perception of sharpness, by bluntly neglecting color- and structural details on a low pixel level.

4 Conclusions and Outlook

The methods presented in this thesis should be easily applicable to other types of filter patterns, since the diffusion algorithm is working on a pre-interpolated full-color image in the first place. When using a different color-filter-pattern, different pre-interpolation algorithms must of course be used. Minor adjustments to the structure tensor might also be necessary, to take advantage of the respective characteristics of other filter patterns.

Diffusion has several very useful properties when used as an interpolation method. It can preserve edges and average grayvalues and even enhance image quality by suppressing noise or aliasing artifacts. However, there are also certain limitations. Diffusion has a high computational complexity and can not easily be applied in multithreading environments since it is a (temporally) nonlinear process. Because of the number of parameters involved, such algorithms are better suited for post-processing. Diffusion can be applied in addition to other demosaicing algorithms to improve quality, as the results have shown.

Handling lots of parameters can be tricky but also gives many opportunities. Adjusting parameters depending on image content (many high- or low frequency structures, ...) could lead to improvements. Since the green channel is inherently different from the red and blue channels, it could also be useful to give the structure tensor of the green channel more attention (for example with weights).

A Available Demosaicing Algorithms

In the following, we will take a brief look at available linear and nonlinear demosaicing algorithms. The techniques presented in this section are used for comparison wiht our own algorithm. We are also going to utilize these algorithms in the approach presented in this paper, since, for the diffusion process to operate correctly, it needs a full-color image to start with.

In the following, we will refer to a green value on a red Bayer Pattern pixel at location (i, j) in the image as $g_{i,j}^{red}$, a red value on a blue Bayer Pattern pixel is denoted as $r_{i,j}^{blue}$, and so forth. For a Bayer Pattern image, this means that only the values $r_{i,j}^{red}$, $b_{i,j}^{blue}$ and $g_{i,j}^{green} \forall i \in (0 \ldots \text{height}), j \in (0 \ldots \text{width})$ are known from the start. When referring to red (blue) Bayer Pattern rows, we are talking about rows that contain red (blue) Bayer Pattern pixels[3]. Figure 9 illustrates the notations used when referring to pixels on the grid.

We will not go into the details of boundary treatment in this work, since it is of little interest to us in this context. Extending the image plane by several pixels in each direction and applying e.g. Neumann- or Dirichlet boundary conditions is sufficient for our purposes and enables us to ignore boundaries in the algorithms of this section.

i-2 j-2	i-2 j-1	i-2 j	i-2 j+1	i-2 j+2
i-1 j-2	i-1 j-1	i-1 j	i-1 j+1	i-1 j+2
i j-2	i j-1	i j	i j+1	i j+2
i+1 j-2	i+1 j-1	i+1 j	i+1 j+1	i+1 j+2
i+2 j-2	i+2 j-1	i+2 j	i+2 j+1	i+2 j+2

Figure 9: $5 * 5$ Neighborhood of pixel (i, j). i denotes height/columns, j denotes width/rows.

A.1 Nearest Neighbor Interpolation

Nearest Neighbor Interpolation (also known as Pixel Doubling Interpolation, Proximal Interpolation or Point Sampling) is the most straightforward way of demosaicing a Bayer Pattern image. This method simply selects one of the available values from one of its direct neighbors. Since several of the 9 neighbor pixels are possible candidates, the algorithm can be implemented

[3]we assume the image to start with an RGGB square pattern in the top left corner (first row is a red Bayer Pattern row), as the one in Figure 1b (without loss of generality)

in several slightly different variations. In our case we (arbitrarily) decided to always select the values from the neighboring pixels with priority to the right and then to the bottom.

The green values on red and blue Bayer Pattern pixels are calculated as follows:

$$g_{i,j}^{red} = g_{i,j+1}^{green}$$

$$g_{i,j}^{blue} = g_{i,j+1}^{green}$$

For a missing red value we have:

on a green Bayer Pattern pixel: $$r_{i,j}^{green} = \begin{cases} r_{i,j+1}^{red} & \text{in a red row} \\ r_{i+1,j}^{red} & \text{in a blue row} \end{cases}$$

on a blue Bayer Pattern pixel: $$r_{i,j}^{blue} = r_{i+1,j+1}^{red}$$

and for a missing blue value:

on a green Bayer Pattern pixel: $$b_{i,j}^{green} = \begin{cases} b_{i+1,j}^{blue} & \text{in a red row} \\ b_{i,j+1}^{blue} & \text{in a blue row} \end{cases}$$

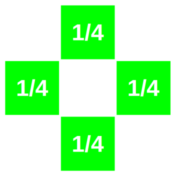
on a red Bayer Pattern pixel: $$b_{i,j}^{red} = b_{i+1,j+1}^{blue}$$

A.2 Bilinear Interpolation

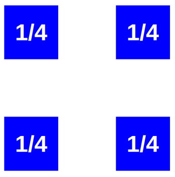
A slightly more sophisticated and one of the more popular algorithms is *bilinear interpolation*. Instead of picking one value from the direct neighborhood, we calculate the average of all the available neighbor values of corresponding color. This gives us a more precise estimate of the missing value and, to a certain extent, also inhibits the zippering effect from section 1.3.4.

For a missing green value in red and blue Bayer Pattern pixels, four neighboring values are available:

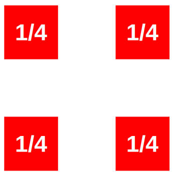
$$g_{i,j}^{red} = g_{i,j}^{blue} = \frac{g_{i-1,j}^{green} + g_{i+1,j}^{green} + g_{i,j-1}^{green} + g_{i,j+1}^{green}}{4}$$



(a) G at R and B locations

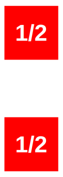


(b) B at R locations

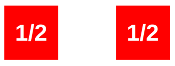


(c) R at B locations

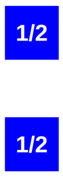
Figure 10: Bilinear Interpolation - Stencils on red and blue Bayer Pattern pixels



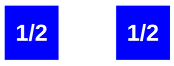
(a) R at G locations (blue row)



(b) R at G locations (red row)



(c) B at G locations (red row)



(d) B at G locations (blue row)

Figure 11: Bilinear Interpolation - Stencils on green Bayer Pattern pixels

For missing blue or red values on a green Bayer Pattern pixel, two neighboring values are averaged:

$$r_{i,j}^{green} = \begin{cases} \frac{r_{i,j-1}^{red}+r_{i,j+1}^{red}}{2} & \text{in a red row} \\ \frac{r_{i-1,j}^{red}+r_{i+1,j}^{red}}{2} & \text{in a blue row} \end{cases}$$

$$b_{i,j}^{green} = \begin{cases} \frac{b_{i-1,j}^{blue}+b_{i+1,j}^{blue}}{2} & \text{in a red row} \\ \frac{b_{i,j-1}^{blue}+b_{i,j+1}^{blue}}{2} & \text{in a blue row} \end{cases}$$

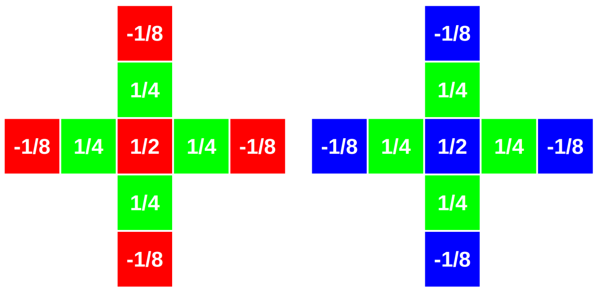
For blue pixels on a red Bayer grid point and vice versa, four neighbors are available:

$$b_{i,j}^{red} = \frac{b_{i-1,j-1}^{blue} + b_{i+1,j-1}^{blue} + b_{i-1,j+1}^{blue} + b_{i+1,j+1}^{blue}}{4}$$

$$r_{i,j}^{blue} = \frac{r_{i-1,j-1}^{red} + r_{i+1,j-1}^{red} + r_{i-1,j+1}^{red} + r_{i+1,j+1}^{red}}{4}$$

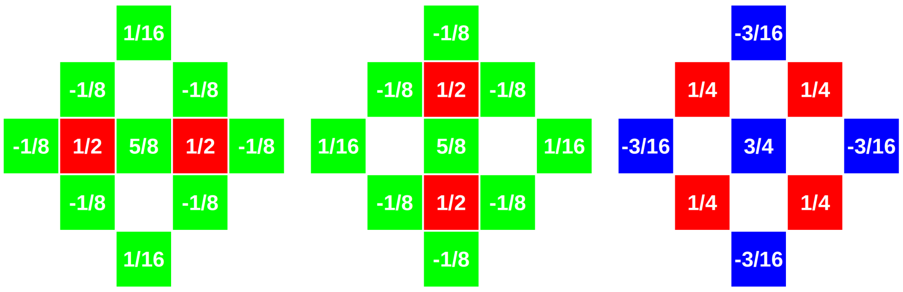
A.3 High Quality Linear Interpolation

High Quality Linear Interpolation was proposed by [MHC04]. It is based upon bilinear interpolation, but with slightly more sophisticated masks. The



(a) G at R locations (b) G at B locations

Figure 12: High Quality Linear Interpolation - Green Stencil



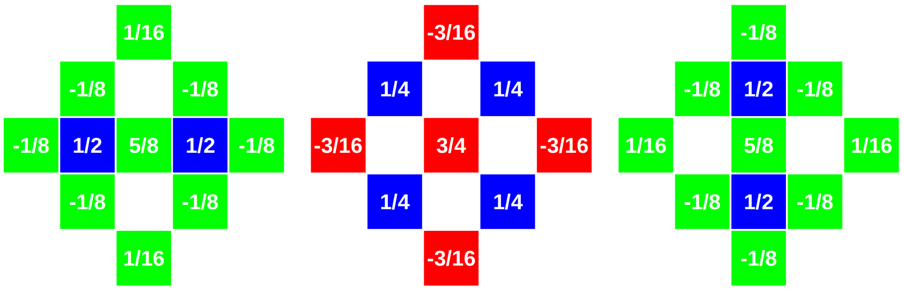
(a) R at G locations, red row (b) R at G locations, blue row (c) B at R locations, blue row

Figure 13: High Quality Linear Interpolation - Red Stencil

algorithm takes into account that luminance information is much more important for edges than chrominance information, so it doesn't discard the other two respective color values of each pixel in the stencil completely, but instead incorporates the gradient information they contain. According to our tests in section 2.2, High Quality Linear Interpolation offers much better quality than Bilinear Interpolation. High Quality Linear Interpolation can keep up with several nonlinear demosaicing algorithms, without requiring much computational power. The stencils for this method are illustrated in figures 12, 13 and 14.

A.4 Variable Number of Gradients Interpolation

Variable Number of Gradients Interpolation (VNG) is a popular technique that found its way into several open-source photo editing applications. It was introduces by [CCP99]. VNG first calculates the gradients in different



(a) B at G locations, blue row (b) B at R locations, red row (c) B at G locations, red row

Figure 14: High Quality Linear Interpolation - Blue Stencil

directions of a pixel and takes only those values into consideration which lie in directions with low gradient. This prevents color information to diffuse across edges.

A.4.1 Gradient Calculation

In the first step, the gradients (∇) in all 8 cardinal directions of pixel (i, j) are calculated.

The gradients of green pixels in red Bayer Pattern rows are calculated as follows (for green pixels in blue Bayer Pattern rows, just switch all the appearances of “red” with “blue” and “r” with “b” in the following equations):

$$\nabla N = \left|b_{i-1,j}^{blue} - b_{i+1,j}^{blue}\right| + \left|g_{i-2,j}^{green} - g_{i,j}^{green}\right| + \frac{\left|g_{i-1,j-1}^{green} - g_{i+1,j-1}^{green}\right|}{2} + \frac{\left|g_{i-1,j+1}^{green} - g_{i+1,j+1}^{green}\right|}{2} + \frac{\left|r_{i-2,j-1}^{red} - r_{i,j-1}^{red}\right|}{2} + \frac{\left|r_{i-2,j+1}^{red} - r_{i,j+1}^{red}\right|}{2}$$

$$\nabla E = \left|r_{i,j-1}^{red} - r_{i,j+1}^{red}\right| + \left|g_{i,j}^{green} - g_{i,j+2}^{green}\right| + \frac{\left|g_{i-1,j-1}^{green} - g_{i-1,j+1}^{green}\right|}{2} + \frac{\left|g_{i+1,j-1}^{green} - g_{i+1,j+1}^{green}\right|}{2} + \frac{\left|b_{i-1,j}^{blue} - b_{i-1,j+2}^{blue}\right|}{2} + \frac{\left|b_{i+1,j}^{blue} - b_{i+1,j+2}^{blue}\right|}{2}$$

$$\nabla S = \left|b_{i-1,j}^{blue} - b_{i+1,j}^{blue}\right| + \left|g_{i,j}^{green} - g_{i+2,j}^{green}\right| + \frac{\left|g_{i-1,j-1}^{green} - g_{i+1,j-1}^{green}\right|}{2} + \frac{\left|g_{i-1,j+1}^{green} - g_{i+1,j+1}^{green}\right|}{2} + \frac{\left|r_{i,j-1}^{red} - r_{i+2,j-1}^{red}\right|}{2} + \frac{\left|r_{i,j+1}^{red} - r_{i+2,j+1}^{red}\right|}{2}$$

$$\begin{aligned}\nabla W = &\left|r^{red}_{i,j-1} - r^{red}_{i,j+1}\right| + \left|g^{green}_{i,j} - g^{green}_{i,j-2}\right| + \frac{\left|g^{green}_{i-1,j-1} - g^{green}_{i-1,j+1}\right|}{2} \\ &+ \frac{\left|g^{green}_{i+1,j-1} - g^{green}_{i+1,j+1}\right|}{2} + \frac{\left|b^{blue}_{i-1,j-2} - b^{blue}_{i-1,j}\right|}{2} + \frac{\left|b^{blue}_{i+1,j-2} - b^{blue}_{i+1,j}\right|}{2}\end{aligned}$$

$$\begin{aligned}\nabla NE = &\left|g^{green}_{i-1,j+1} - g^{green}_{i+1,j-1}\right| + \left|g^{green}_{i-2,j+2} - g^{green}_{i,j}\right| \\ &+ \left|r^{red}_{i,j-1} - r^{red}_{i-2,j+1}\right| + \left|b^{blue}_{i-1,j+2} - b^{blue}_{i+1,j}\right|\end{aligned}$$

$$\begin{aligned}\nabla SE = &\left|g^{green}_{i+1,j+1} - g^{green}_{i-1,j-1}\right| + \left|g^{green}_{i+2,j+2} - g^{green}_{i,j}\right| \\ &+ \left|r^{red}_{i+2,j+1} - r^{red}_{i,j-1}\right| + \left|b^{blue}_{i-1,j} - b^{blue}_{i+1,j+2}\right|\end{aligned}$$

$$\begin{aligned}\nabla NW = &\left|g^{green}_{i+1,j+1} - g^{green}_{i-1,j-1}\right| + \left|g^{green}_{i-2,j-2} - g^{green}_{i,j}\right| \\ &+ \left|r^{red}_{i-2,j-1} - r^{red}_{i,j+1}\right| + \left|b^{blue}_{i-1,j-2} - b^{blue}_{i+1,j}\right|\end{aligned}$$

$$\begin{aligned}\nabla SW = &\left|g^{green}_{i-1,j+1} - g^{green}_{i+1,j-1}\right| + \left|g^{green}_{i+2,j-2} - g^{green}_{i,j}\right| \\ &+ \left|r^{red}_{i+2,j-1} - r^{red}_{i,j+1}\right| + \left|b^{blue}_{i+1,j-2} - b^{blue}_{i-1,j}\right|\end{aligned}$$

The gradients of red pixels of the Bayer Pattern are calculated as follows (for blue pixels of the Bayer Pattern, just switch all the appearances of “red” with “blue” and “r” with “b” in the following equations):

$$\begin{aligned}\nabla N = &\left|g^{green}_{i-1,j} - g^{green}_{i+1,j}\right| + \left|r^{red}_{i-2,j} - r^{red}_{i,j}\right| + \frac{\left|b^{blue}_{i-1,j-1} - b^{blue}_{i+1,j-1}\right|}{2} \\ &+ \frac{\left|b^{blue}_{i-1,j+1} - b^{blue}_{i+1,j+1}\right|}{2} + \frac{\left|g^{green}_{i-2,j-1} - g^{green}_{i,j-1}\right|}{2} + \frac{\left|g^{green}_{i-2,j+1} - g^{green}_{i,j+1}\right|}{2}\end{aligned}$$

$$\begin{aligned}\nabla E = &\left|g^{green}_{i,j-1} - g^{green}_{i,j+1}\right| + \left|r^{red}_{i,j} - r^{red}_{i,j+2}\right| + \frac{\left|b^{blue}_{i-1,j-1} - b^{blue}_{i-1,j+1}\right|}{2} \\ &+ \frac{\left|b^{blue}_{i+1,j-1} - b^{blue}_{i+1,j+1}\right|}{2} + \frac{\left|g^{green}_{i-1,j+2} - g^{green}_{i-1,j}\right|}{2} + \frac{\left|g^{green}_{i+1,j+2} - g^{green}_{i+1,j}\right|}{2}\end{aligned}$$

$$\begin{aligned}\nabla S = &\left|g^{green}_{i-1,j} - g^{green}_{i+1,j}\right| + \left|r^{red}_{i,j} - r^{red}_{i+2,j}\right| + \frac{\left|b^{blue}_{i-1,j-1} - b^{blue}_{i+1,j-1}\right|}{2} \\ &+ \frac{\left|b^{blue}_{i-1,j+1} - b^{blue}_{i+1,j+1}\right|}{2} + \frac{\left|g^{green}_{i,j-1} - g^{green}_{i+2,j-1}\right|}{2} + \frac{\left|g^{green}_{i,j+1} - g^{green}_{i+2,j+1}\right|}{2}\end{aligned}$$

$$\nabla W = \left|g^{green}_{i,j-1} - g^{green}_{i,j+1}\right| + \left|r^{red}_{i,j} - r^{red}_{i,j-2}\right| + \frac{\left|b^{blue}_{i-1,j-1} - b^{blue}_{i-1,j+1}\right|}{2}$$
$$+ \frac{\left|b^{blue}_{i+1,j-1} - b^{blue}_{i+1,j+1}\right|}{2} + \frac{\left|g^{green}_{i-1,j-2} - g^{green}_{i-1,j}\right|}{2} + \frac{\left|g^{green}_{i+1,j-2} - g^{green}_{i+1,j}\right|}{2}$$

$$\nabla NE = \left|b^{blue}_{i-1,j+1} - b^{blue}_{i+1,j-1}\right| + \left|r^{red}_{i-2,j+2} - r^{red}_{i,j}\right| + \frac{\left|g^{green}_{i-1,j} - g^{green}_{i,j-1}\right|}{2}$$
$$+ \frac{\left|g^{green}_{i+1,j} - g^{green}_{i,j+1}\right|}{2} + \frac{\left|g^{green}_{i-1,j} - g^{green}_{i-2,j+1}\right|}{2} + \frac{\left|g^{green}_{i,j+1} - g^{green}_{i+1,j+2}\right|}{2}$$

$$\nabla SE = \left|b^{blue}_{i-1,j-1} - b^{blue}_{i+1,j+1}\right| + \left|r^{red}_{i+2,j+2} - r^{red}_{i,j}\right| + \frac{\left|g^{green}_{i-1,j} - g^{green}_{i,j+1}\right|}{2}$$
$$+ \frac{\left|g^{green}_{i,j-1} - g^{green}_{i+1,j}\right|}{2} + \frac{\left|g^{green}_{i,j+1} - g^{green}_{i+1,j+2}\right|}{2} + \frac{\left|g^{green}_{i+1,j} - g^{green}_{i+2,j+1}\right|}{2}$$

$$\nabla NW = \left|b^{blue}_{i-1,j-1} - b^{blue}_{i+1,j+1}\right| + \left|r^{red}_{i-2,j-2} - r^{red}_{i,j}\right| + \frac{\left|g^{green}_{i,j+1} - g^{green}_{i-1,j}\right|}{2}$$
$$+ \frac{\left|g^{green}_{i+1,j} - g^{green}_{i,j-1}\right|}{2} + \frac{\left|g^{green}_{i-2,j-1} - g^{green}_{i-1,j}\right|}{2} + \frac{\left|g^{green}_{i,j-1} - g^{green}_{i-1,j-2}\right|}{2}$$

$$\nabla SW = \left|b^{blue}_{i-1,j+1} - b^{blue}_{i+1,j-1}\right| + \left|r^{red}_{i,j} - r^{red}_{i+2,j-2}\right| + \frac{\left|g^{green}_{i,j-1} - g^{green}_{i-1,j}\right|}{2}$$
$$+ \frac{\left|g^{green}_{i+1,j} - g^{green}_{i,j+1}\right|}{2} + \frac{\left|g^{green}_{i+1,j-2} - g^{green}_{i,j-1}\right|}{2} + \frac{\left|g^{green}_{i+2,j-1} - g^{green}_{i+1,j}\right|}{2}$$

A.4.2 Threshold Calculation

Now we compute a threshold value, based on the gradient results. All the gradient values that are above the threshold will be discarded in the following step.

$$T = (k_1 * \text{MIN}) + (k_2 * (\text{MAX} - \text{MIN})) \qquad \text{with } k_1 = 1.5 \text{ and } k_2 = 0.5$$

where MIN and MAX are the minimum and maximum of the 8 gradient values.

A.4.3 Mean Value Calculation

We now only consider those pixels, that lie within the remaining gradient directions of our pixel neighborhood, since we want to prevent averaging across edges. We compute the average across those remaining pixel values for each color and store them as three variables R, G and B.

A.4.4 Interpolation Step

We use the color values from the step before to determine a color difference between the known color value at this location and the other two color values to be recovered. The following formulae are used to compute the missing values at our respective pixel location:

on a green pixel:
$$b = g + BG_{diff} = g + \tfrac{(B_{sum} - G_{sum})}{4}$$
$$r = g + RG_{diff} = g + \tfrac{(R_{sum} - G_{sum})}{4}$$

on a blue pixel:
$$g = b + GB_{diff} = b + \tfrac{(G_{sum} - B_{sum})}{4}$$
$$r = b + RB_{diff} = b + \tfrac{(R_{sum} - B_{sum})}{4}$$

on a red pixel:
$$b = r + BR_{diff} = r + \tfrac{(B_{sum} - R_{sum})}{4}$$
$$g = r + GR_{diff} = r + \tfrac{(G_{sum} - R_{sum})}{4}$$

For a more detailed description of the algorithm, please refer to [CCP99].

References

[Bay75] Bryce E. Bayer. Color Imaging Array. United States Patent US003971065, Eastman Kodak Company, United States Patent and Trademark Office, 5 May 1975.

[Bec07] Tobias Becker. Image interpolation methods and enhancements. Technical report, Saarland University, Department of Computer Science, Mathematical Image Analysis Group, Saarbrücken, Germany, 2007.

[CCP99] Ed Chang, Shiufun Cheung, and Davis Pan. Color Filter Array Recovery Using a Threshold-based Variable Number of Gradients. Technical report, Compaq Computer Corporation, Cambridge Research Laboratory, Cambridge, Massachusetts, USA, 1999.

[Chi10] Hoo Tang Ching. Using the (1+1) EA for Optimizing Image Interpolation with Homogenous Diffusion. Technical report, Saarland University, Department of Computer Science, Mathematical Image Analysis Group, Saarbrücken, Germany, 2010.

[FG87] Wolfgang Förstner and Eberhard Gülch. A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centers of Circular Features. *Proceedings of the ISPRS Intercommission Workshop on Fast Processing of Photogrammetric Data*, pages 281–305, June 1987.

[Jea04] Rémi Jean. Demosaicing with The Bayer Pattern. Technical report, Department of Computer Science, University of North Carolina, 2004.

[MHC04] Henrique S Malvar, Li Wei He, and Ross Cutler. High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. In *International Conference of Acoustic, Speech and Signal Processing*, One Microsoft Way, Redmond, Washington 98058, USA, May 2004. Institute of Electrical and Electronics Engineers, Inc.

[MW09] Markus Mainberger and Joachim Weickert. Edge-Based Image Compression with Homogeneous Diffusion. Technical report, Saarland University, Department of Computer Science, Mathematical Image Analysis Group, Saarbrücken, Germany, 2009.

[Nak06] Junichi Nakamura. *Image Sensors and Signal Processing for Digital Still Cameras.* CRC Press, Taylor and Francis Group, 6000 Broken Sound Parkway NW Suite 300, Boca Raton FL, 33487-2742, 2006.

[PM87] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. In *Proceedings of IEEE Computer Society Workshop on Computer Vision*, pages 16–22, November 1987.

[PM88] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. Technical report, Electrical Engineering and Computer Science department, University of California at Berkeley, Berkeley, CA 94720, USA, 20 December 1988.

[Rei06] Andreas Reifschneider. Texture generation, Image analysis and enhancement using reaction-diffusion systems and their enhancement using nonlinear anisotropic diffusion algorithms. Technical report, Saarland University, Department of Computer Science, Mathematical Image Analysis Group, Saarbrücken, Germany, 2006.

[Rus10] Kaloyan Rusev. Demosaicing Using the Structure Tensor. Technical report, Saarland University, Department of Computer Science, Mathematical Image Analysis Group, Saarbrücken, Germany, 10 January 2010.

[Sha49] Claude E Shannon. Communication in the presence of noise. In *Proceedings of the Institute of Radio Engineers*, volume 37, pages 10–21, January 1949.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.

[Wei98] Joachim Weickert. *Anisotropic Diffusion in Image Processing.* ECMI Series. B. G. Teubner Verlag, Stuttgart, Germany, 1998.

[WRV98] Joachim Weickert, Bart M ter Haar Romeny, and Max A Viergever. Efficient and Reliable Schemes for Nonlinear Diffusion Filtering. *IEEE Transactions on Image Processing*, 7(3):398–410, March 1998.

Table 3: Comparison of demosaicing methods; the values respresent the signal-to-noise ratio between the original full color image and the resulting image of the respective demosaicing algorithms (in deciBel); images a) to h)

Img	Chan	Near Nbr	Bilinear	HQLI	VNG	Diffusion
a	R	22.07	24.68	30.41	28.83	31.89
	G	25.08	29.36	35.13	37.39	36.73
	B	21.88	25.29	30.50	29.64	31.90
	RGB	22.79	26.01	31.53	30.66	33.00
b	R	27.69	30.55	34.81	33.73	35.86
	G	32.12	35.46	38.58	40.94	39.82
	B	28.44	31.41	34.60	34.58	35.38
	RGB	29.04	32.01	35.65	35.46	36.62
c	R	28.83	31.12	35.58	35.06	37.10
	G	33.18	36.08	40.33	41.88	42.77
	B	29.45	32.67	35.97	35.89	37.23
	RGB	30.11	32.84	36.84	36.75	38.37
d	R	27.94	32.33	36.17	36.60	37.04
	G	31.24	35.74	40.74	41.90	42.02
	B	27.77	30.89	36.16	35.01	37.42
	RGB	28.72	32.55	37.23	37.00	38.32
e	R	21.84	25.17	31.83	31.27	33.84
	G	24.25	29.08	36.06	37.88	37.61
	B	22.05	25.99	31.43	32.10	33.10
	RGB	22.58	26.45	32.67	32.93	34.45
f	R	23.13	25.99	31.40	30.51	32.94
	G	28.70	30.73	36.13	38.09	38.06
	B	23.45	26.73	31.25	30.87	32.67
	RGB	24.46	27.38	32.42	32.07	33.96
g	R	26.97	30.72	36.07	35.13	37.63
	G	31.04	35.54	40.32	41.69	42.31
	B	26.93	32.01	36.31	36.58	37.32
	RGB	27.94	32.33	37.18	37.03	38.57
h	R	19.22	22.29	28.09	25.91	29.60
	G	21.36	27.28	32.87	34.69	35.11
	B	19.36	22.71	27.77	26.10	29.21
	RGB	19.87	23.59	29.04	27.48	30.62

Table 4: Comparison of demosaicing methods; the values respresent the signal-to-noise ratio between the original full color image and the resulting image of the respective demosaicing algorithms (in deciBel); images i) to p)

Img	Chan	NN	Bilinear	HQLI	VNG	Diffusion
i	R	26.87	31.33	36.23	35.94	37.48
	G	30.53	35.14	40.25	41.92	42.08
	B	27.21	29.94	35.46	33.81	36.75
	RGB	27.92	31.64	36.87	36.11	38.22
j	R	27.42	31.50	36.71	36.75	38.03
	G	30.31	34.78	40.86	42.45	42.56
	B	27.44	29.75	35.62	34.25	37.16
	RGB	28.19	31.55	37.22	36.68	38.70
k	R	24.64	27.51	33.04	32.17	34.60
	G	28.05	32.01	37.34	39.25	38.98
	B	24.98	28.39	33.20	33.30	34.64
	RGB	25.64	28.91	34.13	34.01	35.65
l	R	27.75	30.17	35.81	34.22	37.15
	G	31.68	35.78	40.91	42.53	43.17
	B	27.44	32.05	36.03	36.19	37.14
	RGB	28.58	32.10	37.04	36.48	38.40
m	R	20.06	22.84	28.69	29.14	30.31
	G	23.47	26.40	32.50	34.41	33.52
	B	20.01	23.11	28.19	29.14	29.57
	RGB	20.91	23.85	29.41	30.30	30.83
n	R	23.91	27.44	32.72	32.44	34.12
	G	27.81	31.68	36.51	37.78	38.10
	B	24.74	28.21	32.03	32.68	33.22
	RGB	25.19	28.76	33.36	33.71	34.69
o	R	26.95	28.89	33.92	32.30	34.99
	G	29.56	34.09	38.22	39.58	40.07
	B	27.12	30.39	33.99	33.71	35.24
	RGB	27.73	30.63	34.97	34.26	36.23
p	R	26.68	29.01	33.32	32.73	34.88
	G	33.31	34.02	38.40	39.46	40.90
	B	27.07	29.69	34.20	33.21	35.64
	RGB	28.17	30.42	34.81	34.26	36.45

Table 5: Comparison of demosaicing methods; the values respresent the signal-to-noise ratio between the original full color image and the resulting image of the respective demosaicing algorithms (in deciBel); images q) to x)

Img	Chan	NN	Bilinear	HQLI	VNG	Diffusion
q	R	27.62	31.43	36.74	37.72	38.41
	G	29.60	34.47	40.37	42.26	41.50
	B	27.37	30.66	36.59	36.76	37.97
	RGB	28.09	31.90	37.58	38.34	39.04
r	R	23.73	27.13	32.79	33.22	34.17
	G	26.35	30.40	36.38	37.35	37.04
	B	23.89	26.70	32.56	32.70	33.86
	RGB	24.50	27.79	33.60	33.98	34.81
s	R	23.71	26.82	32.71	30.05	34.27
	G	27.25	31.77	37.31	39.76	39.38
	B	24.24	26.99	32.74	30.27	34.26
	RGB	24.81	28.01	33.79	31.69	35.41
t	R	25.52	29.08	33.24	32.83	34.64
	G	29.51	33.08	37.45	39.64	39.56
	B	25.82	28.44	32.60	31.80	33.71
	RGB	26.61	29.77	33.97	33.67	35.33
u	R	23.93	27.16	32.79	32.19	34.38
	G	28.38	31.35	37.00	38.78	38.41
	B	24.20	27.54	32.52	32.30	33.82
	RGB	25.09	28.32	33.68	33.55	35.12
v	R	26.20	29.18	34.16	33.39	35.64
	G	28.42	33.11	37.75	39.45	39.12
	B	26.12	29.34	33.71	33.24	34.80
	RGB	26.79	30.21	34.87	34.58	36.16
w	R	28.61	31.33	35.26	34.62	36.46
	G	32.58	36.81	40.51	42.29	42.51
	B	29.16	33.74	38.50	38.39	39.54
	RGB	29.80	33.42	37.54	37.38	38.82
x	R	23.25	26.33	32.14	32.38	33.50
	G	25.31	29.40	35.38	36.20	36.06
	B	22.28	25.26	30.02	30.22	31.17
	RGB	23.44	26.67	31.99	32.29	33.14

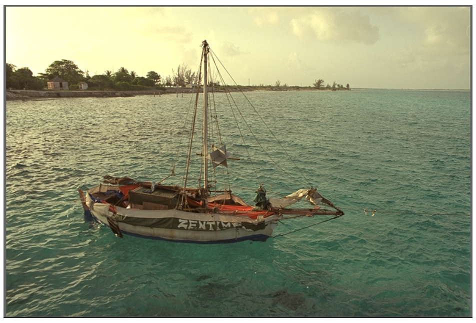
(a)

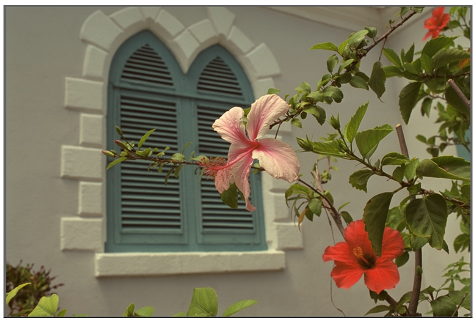
(b)

(c)

(d)

(e)


(f)


(g)

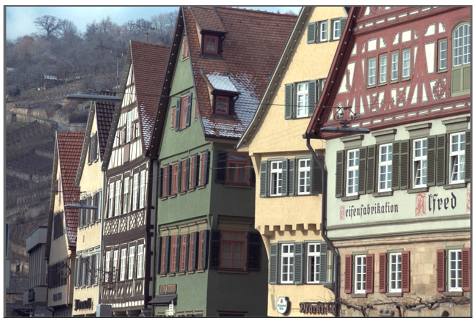

(h)

Figure 15: Set of standard test images, Source: Eastman Kodak Company

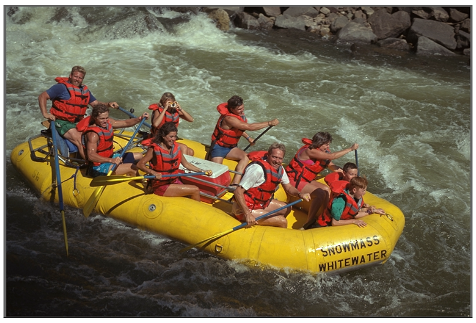
(i)

(j)

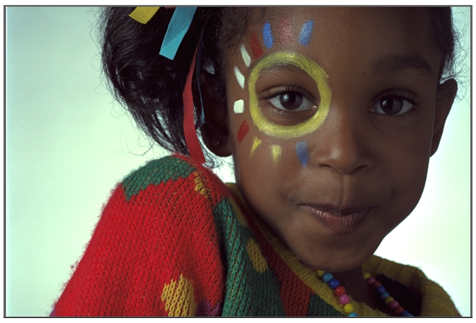
(k)

(l)

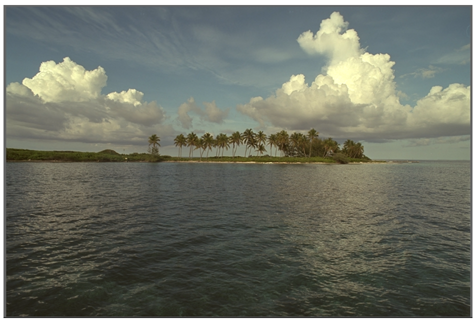
(m)



(n)



(o)



(p)

Figure 15: Set of standard test images, Source: Eastman Kodak Company

(q)

(r)

(s)

(t)

(u)

(v)

(w)

(x)

Figure 15: Set of standard test images, Source: Eastman Kodak Company