UNIVERSITÀ DEGLI STUDI DI CATANIA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

DIPARTIMENTO DI MATEMATICA ED INFORMATICA

Dottorato di Ricerca in Informatica -XXIII Ciclo

Rosetta Rizzo

Image Noise Removal for Embedded Devices

Characterization, estimation and filtering techniques

TESI DI DOTTORATO

Tutor:
Chiar.mo Prof. SEBASTIANO BATTIATO

Coordinatore:
Chiar.mo Prof. Domenico Cantone

ANNO ACCADEMICO 2009 - 2010

To my husband.

Acknowledgements

Writing this thesis, it is natural to think back of those three years of work that have brought me so far, and obviously places, events and especially people who have accompanied me come to mind.

It is difficult to draw up an ordered list of people who I would like to thank, and especially deciding in what order to make the list, because each of them has been important as well. But, somehow, it needs to be done...

I'll start, then, by thanking the two groups that have allowed me to complete this PhD, which are: STMicroelectronics AST Imaging Group and the IPLab of the University of Catania.

Starting from the latter, I would stress that it was a privilege for me to be able to collaborate with the IPLab group, having professors Sebastiano Battiato and Giovanni Gallo as guides, and qualified researchers like Giovanni Puglisi and Giovanni Farinella. Obviously proper thanks go to them and in particular to Prof. Sebastiano Battiato, who followed me throughout this period and led me in my research. But thinking back to moments spent at the University I cannot forget all the other students, and PhD colleagues who collaborate with the group, with whom I shared wonderful moments of work and leisure during the summer schools.

At ST, on the other hand, I spent the most significant part of these three years, therefore, my memories are particularly related to that place and the people who work there. Let me begin, then, by thanking Angelo Bosco, which more closely followed my activities, guiding me in all the phases of my work, from the analysis of problems to the check of the results. But a big thanks goes also to all other group members: Arcangelo, Giuseppe M., Alessandro, Mirko, Valeria, Ivana, Daniele, Davide, Filippo, Antonio, Salvo, Alfio, Giuseppe S. and Mauro. Each of them gave me a bit of his experience in the workplace, contributing to my professional growth. A great esteem and friendship binds me to every one of them, which is also due to the positive and friendly atmosphere that exists in the group.

A special mention, however, is to be dedicated to my PhD colleagues: Giuseppe Messina and Tony Meccio, with whom I shared the experience in the company. Giuseppe, being already an ST employee and having a long experience in research, was certainly an important guide to my work, as well as a valuable friend; Tony, with whom I shared many moments of work (but also coffee breaks), was a presence, irreplaceable and pleasant, thanks to which it was easier to spend even the most intense and stressful days with a smile.

A special thought, of course, also goes to my family and especially to my husband. They have always supported and encouraged me, patiently enduring even my moments of greatest stress.

Finally, I want to thank again in particular : the Prof., Giovanni P., Angelo, Arcangelo e Tony, for their support in preparing this thesis.

Contents

List of Figures

List of Tables

Glossary

AG	Analog Gain
AWGN	Additive White Gaussian Noise
CFA	Color Filter Array
CLT	Central Limit Theorem
DCT	Discrete Cosine Transform
FPN	Fixed Pattern Noise
pdf	Probability Density Function
PRNU	Pixel Response Non Uniformity
PSN	Photon Shot Noise
RMSE	Root Mean Squared Error
SDN	Signal Dependent Noise
SF	Sigma Fixed
SNR	Signal to Noise Ratio
WB	White Balance

Published Works

1. A. Bosco, S. Battiato, A. Bruna, and R. Rizzo *Texture sensitive denoising for single sensor color.* In proceedings of 2009 Computational Color Imaging Workshop (CCIW09), Lecture Notes in Computer Science, vol.5646, pp.130-139, 2009.

2. A. Bosco, S. Battiato, A. Bruna, and R. Rizzo, *Noise reduction for CFA image sensors exploiting HVS behaviour.* Sensors Journal, MDPI Open Access - Special Issue on Integrated High-Performance Imagers, vol.9 (3), pp.1692-1713, 2009.

3. A. Bosco, A. R. Bruna, D. Giacalone, S. Battiato, R. Rizzo, *Signal Dependent Raw Image Denoising Using Sensor Noise Characterization Via Multiple Acquisitions.* In proceedings of SPIE Electronic Imaging, vol.7537, 2010.

4. S. Battiato, G. Puglisi, R. Rizzo, *Characterization of Signal Perturbation Using Voting Based Curve Fitting For Multispectral Images.* In proceedings of Image Processing (ICIP) 2010, pp.545-548, 2010.

5. A. Bosco, R. Rizzo, Chapter 6 - *Noise Removal.* In Image Processing for Embedded Devices, vol.1. Bentham, ISSN.1879-7458, 2010.

6. A. Bosco, A. R. Bruna, D. Giacalone, R. Rizzo - *A System for Image Texture and Flatness Estimation Based on CFA Raw Noise Analysis*, European Patent Pending, November 2010.

Other Published Works

1. S. Battiato, G. Messina and R. Rizzo, *Image Forensics: Contraffazione Digitale e Identificazione della Camera di Acquisizione: Status e Prospettive*. In IISFA Memberbook 2009 - Digital Forensics, Chapter 1, pp.1-48, 2009.

2. S. Battiato, G.M. Farinella, G.C. Guarnera, T. Meccio, G. Puglisi, D. Ravì, R. Rizzo, *Bags of Phrases with Codebooks Alignment for Near Duplicate Image Detection*. In Proceedings of ACM Workshop on Multimedia in Forensics, Security and Intelligence (MiFor 2010), in conjunction with the 2010 ACM Multimedia (ACM-MM).

Introduction

Among the many factors contributing to image quality degradation, noise is one of the most recurrent and difficult elements to deal with. Camera phones and low-end digital still cameras are particularly subject to noise degradation, especially when images are acquired in low light. This issue has been the main focus of my work, with the aim of find solutions to the problem of noise on digital images acquired with low cost devices.

My research was funded by STMicroelectronics AST Imaging Catania Lab, which is also the place where it was entirely performed, and it is part of the activities of the joint laboratory between STMicroelectronics and the Image Processing Lab of the University of Catania.

There is not just one single noise source, rather, different elements contribute to signal degradation. Since the noise is generated by the sum of different sources, which overlap with a Gaussian distribution, typically it is defined additive white Gaussian noise *(AWGN)*, and its intensity is provided by the standard deviation σ of the underlying distribution.

Many noise removal filters rely on σ to adaptively change their smoothing effects; for this reason have a good estimate of the amount of noise contaminating an image is crucial to allow the noise filters to work properly. Anyway, due to the intrinsic difficulty in discriminating noise from actual image signal, achieving a correct noise level estimation is a complex operation that usually requires the localization of low textured areas in the image, where the oscillations of the signal level are mainly caused by random noise and not by image content.

The quality of a filtered image, however, depends, not only on a reliable estimate of the noise level, but also by the adopted filtering method. Smart filters capable to remove noise without affecting the tiny details of a digital image are, in fact, of primary importance to produce pleasant pictures. There is a large literature on image denoising techniques. Over the years have been implemented a wide variety of methods, that use different approaches and models in order to eliminate signal fluctuations while preserving image details. Sophisticated denoising methods perform multiresolution analysis and processing in wavelet domain [27, 34]; other techniques implement texture discrimination using fuzzy logic [23,36]. There are also algorithms based on: anisotropic non-linear diffusion equation [32], bilateral filtering [39], non-local mean [8], etc.. Most of these methods, anyway, are often too complex to be run on low cost imaging devices.

Typically, denoising algorithms assume that the noise is AWGN and use a standard deviation constant with varying intensity value. However, due to the different sources of noise involved in the acquisition process and the quantum nature of light itself, it was proved that the noise level mainly depends on the signal intensity [14]. Hence, a reliable noise reduction filter cannot be based on a single standard deviation value but it must take into account its variation as a function of the underlying signal intensity. Recent literature suggests, therefore, different and quite complex frameworks that are capable to estimate the intensity-based noise standard deviation [10, 14].

This thesis aims to provide an overview of the complex problem of noise in digital images, yielding an excursus on the principal noise sources and analyzing some classical noise estimation and filtering algorithms. Useful and innovative methods for the characterization of the imager signal dependent noise are then introduced, jointly with some reliable and effective signal dependent noise reduction algorithms. In addition, a novel denoising technique based on the study of the human visual system is analyzed.

This work is organized as follows. In Chapter 1 the main noise sources at sensor level are described, distinguishing fixed pattern noise from temporal random noise and introducing *Gaussian* noise model and *signal dependent* noise model. A section is also devoted to analyze the image pipeline stages in which noise has a key impact on image quality. Chapter 2 reports the main classical techniques used to estimate and filter image noise, while in Chapter 3 an innovative noise removal technique based on Human Visual System (*HVS*) behaviour is detailed. Chapter 4 exploits some reliable methods to characterize and model the image sensor noise, considering its signal dependency, and finally, in Chapter 5 some innovative filtering algorithms that use signal dependent noise evalutation are proposed.

Chapter 1. Noise Model

1.1 Introduction

Noise formation on a digital image is a quite complicated process, since there are many factors that determine it. During acquisition, the light, passing through the lens, hits the sensor and is converted into a digital signal. This analog to digital conversion creates electronic and physical conditions on the image sensor that corrupt the acquired data, contributing to the formation of image noise.

This chapter discusses the main elements that allow us to better understand this phenomenon of noise on digital images. The image generation pipeline is introduced by analyzing the steps necessary to acquire an image and describing the noise sources and how these are combined and overlap. The difference between a fixed pattern noise and random noise is also explained.

A large part of the chapter deals with the AWGN noise model. In particular, the analysis concentrates on how the noise distribution changes during the execution of the image pipeline and how these changes affect the performance of each algorithm; this allows having a clear picture of how noise impacts the final quality of the image.

Finally the signal-dependent noise is introduced, highlighting how it is possible shape the noise level through a function, whose slope is closely related to the image sensor and its acquisition settings.

1.2 Image Generation Pipeline

An image sensor (or *imager*) uses an electronic sensor to acquire the spatial variations in light intensity and then uses image processing algorithms (called *image generation* *pipeline*) to reconstruct a color picture from the data provided by the sensor (see Fig. 1.1).

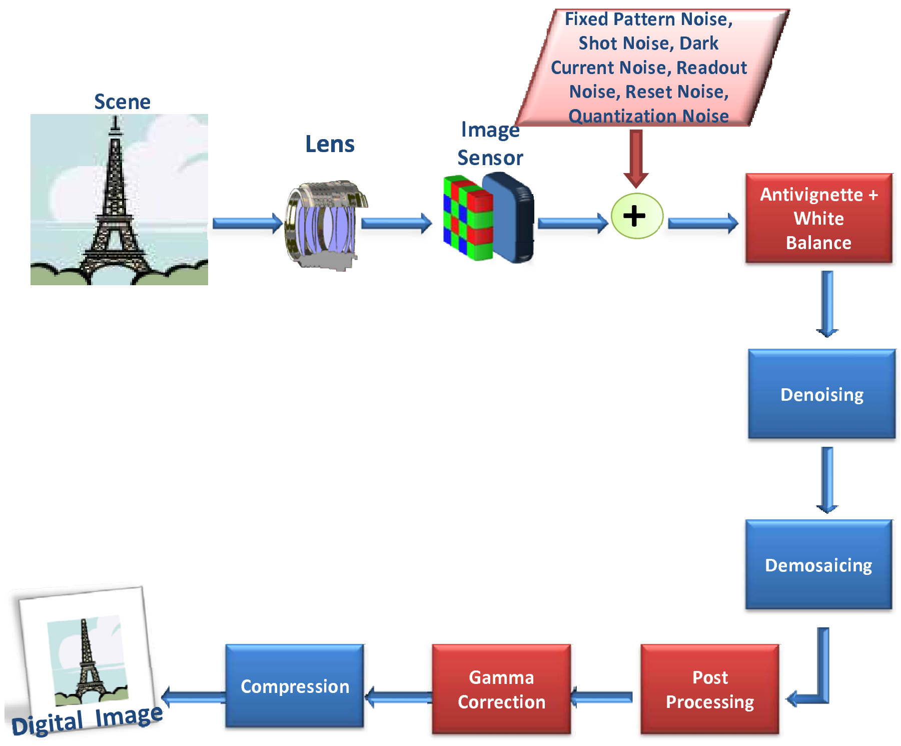

Fixed Pattern Noise, Shot Noise, Dark Current Noise, Readout Noise, Reset Noise, Quantization Noise
Scene
Lens
Image Sensor
+
Antivignette + White Balance
Denoising
Demosaicing
Post Processing
Gamma Correction
Compression
Digital Image

Figure 1.1 : Image processing pipeline and noise sources. Pipeline stages in red indicate the algorithms contributing to increase image noise, while blue stages represent the algorithms that cause a reduction of noise levels.

The sensor is composed by a 2D array of thousands or millions of light-sensitive diodes (photosites) which convert photons (light) into electrons (electrical charge). Due to the deposition of color filters (CFA) on top of a monochrome sensor, each photosite is sensitive to one color component only. The CFA allows you to capture the red, green or blue color component, and is typically arranged into a pattern known as Bayer pattern (Fig.1.2), where number of green elements is twice the number of red and blue pixels due to the higher sensitivity of the human eye to the green light. Starting from the CFA data, image pipeline algorithms (such as white balance, color interpolation, sharpening, etc.) are used to obtain an RGB high quality version of the acquired scene.

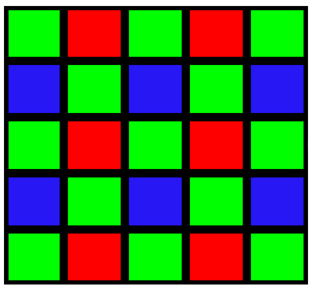


Figure 1.2 : Bayer pattern.

1.3 Noise Types and Models

There is not just one single noise source, rather, during image generation process, many factors contribute to signal degradation (Fig.1.1). Each noise source injects extra-information in the ideal noise-free image signal; eventually unpleasant images are generated if noise is not properly treated. Noise in a digital raw image can be classified into two main categories:

1. Fixed Pattern Noise (*FPN*);

2. Temporal (Random) Noise.

Next sections analyze this two different types of noise and explain how these impact on the images generation process.

1.3.1 Fixed Pattern Noise

In FPN, the term *fixed* refers to the the fact that this noise has a pattern which is invariant with respect to time. FPN has two main components, one at dark and one under illumination. The dark component is known as *dark-FPN* and it is present even in absence of illumination. The FPN under illumination is called Pixel Response Non Uniformity, (*PRNU*) and is caused by different sensitivity of the pixels to light. If the image sensor contains column amplifiers, dark-FPN may appear as vertical stripes in the image *(column-FPN)* that are very annoying and easily detected by the human eye (see Fig.1.3). A technique for removing column-FPN is described in Section 2.4.1.

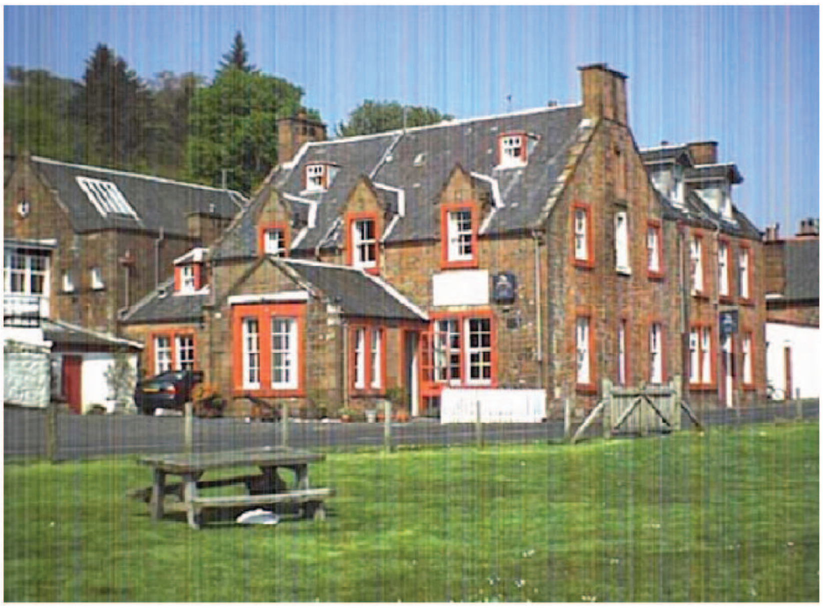


Figure 1.3 : Column-FPN.

1.3.2 Temporal Random Noise

Temporal (random) noise is the part of the noise that fluctuates over time. It changes its pattern frame by frame, even if the same acquisition settings are used, and causes a random variation of brightness or color information. Temporal noise is the sum of different noise sources generated on the imager during the acquisition process. The main temporal noise sources are:

- ***Photon shot noise***: image sensors record light by capturing photons into the photodiodes, eventually converting them into numbers. During the integration time, the arrival rate of photons at each photosite is not constant; rather, there is an intrinsic uncertainty caused by the oscillations of the number of photons that reach the imager.

  These oscillations can be modeled by Poisson distribution. [15, 16].

- ***Dark current noise***: represents the temperature dependent noise generated on the surface of the image sensor. Noise is introduced by the sum of electrons freed by the thermal energy plus electrons generated by the photons hitting the imager.

- ***Readout noise***: is the electronic noise generated during the sensor readout process.

- ***Reset noise***: is generated by residual electrons left in sensors capacitor after the reset operation, which is performed before a new scene acquisition occurs.

- ***Quantization noise***: is due to conversion of photons into a digital number performed by an A/D converter. The errors introduced in the conversion of an analog signal to a set of discrete digital values are known as quantization errors. In particular, quantization noise significantly affects image quality when the bit-depth of the digital conversion process is small.

1.4 Additive Noise Model

Consider an ideal image I with size $M_1 \times M_2$, denoted as $I = [i(x,y)]_{M_1 \times M_2}$, such that $i(x,y) \in \{0,\dots,L-1\}$, $0 \le x \le M_1 - 1$, $0 \le y \le M_2 - 1$. Ideal image I contains no noise, every pixel being the exact representation of the light intensity perfectly recorded and converted by the sensor. In the additive noise model, each pixel of the ideal image is contaminated by a random value drawn from a certain underlying noise distribution Z_d; this random quantity adds to the original ideal signal, generating the noisy observed image $N(x,y)$:

$$N(x,y) = I(x,y) + \eta(x,y) \tag{1.1}$$

The term $\eta(x,y)$ which is added to the ideal value $I(x,y)$ is generated by the contribution of many overlapping noise sources. Because of the central limit theorem, a common assumption is to model the contribution of all noise sources as zero mean *Additive White Gaussian Noise (AWGN)*. Eventually, the noisy term $N(x,y)$ is then observed and recorded.

1.5 Central Limit Theorem

Before proceeding, we recall the Central Limit Theorem (*CLT*). Consider n independent and identically distributed (i.i.d.) random variables $X_1, X_2, \dots, X_n$, each one having a certain mean μ and variance $\sigma^2 > 0$. Let S_n be the sum of each X_i, $i = 1,\dots,n$:

$$S_n = \sum_{i=1}^{n} X_i \tag{1.2}$$

Consider the new variable:

$$Z_n = \frac{S_n - n\mu}{\sigma\sqrt{n}} \tag{1.3}$$

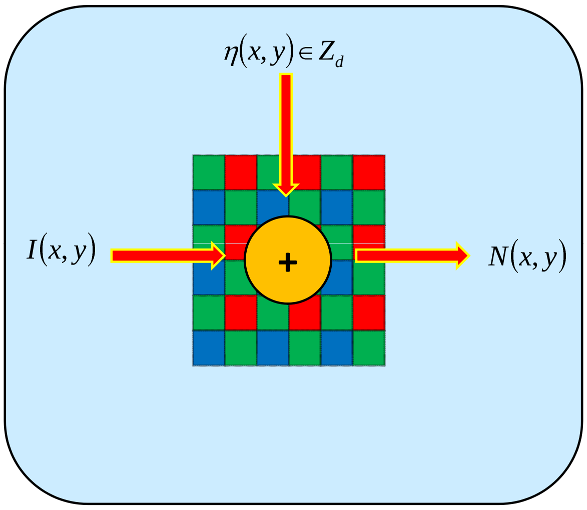


Figure 1.4 : Additive noise model. Ideal image signal $I(x,y)$ is contaminated by a noisy signal $\eta(x,y)$ whose intensities are drawn from an underlying noise distribution Z_d.

The *CLT* states that the distribution of the sample average of the random variables converges to the normal distribution with mean μ and variance σ^2/n even if the X_i have different distributions. In other words the distribution of Z_n converges in distribution to the normal standard distribution $\mathcal{N}(0,1)$ as the number of added i.i.d. X_i approaches infinity:

$$Z_d \rightarrow \mathcal{N}(0,1) \tag{1.4}$$

1.6 Additive White Gaussian Noise Model

AWGN is the most widely adopted noise model; this assumption arises from the central limit theorem: all noise sources overlap, finally producing a zero mean Gaussian distributed noise. More specifically, the theorem states that the sum of a large number of independent random variables is Gaussian distributed. In order to correctly apply the *CLT*, the following properties must be satisfied:

- each single random variable must be independent;
- each term in the sum must be small compared to the overall sum;
- there must be a large number of random variables contributing to the sum.

These assumptions fit well with the fact that not all noise sources have Gaussian distribution. Probability Density Function *(pdf)* of the Gaussian distribution is shown in Fig.1.5 and is modeled as:

$$f(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{1.5}$$

where x is the signal intensity, μ and σ are respectively the mean and standard deviation of the signal x [15]. Some key properties of the normal distribution often used in

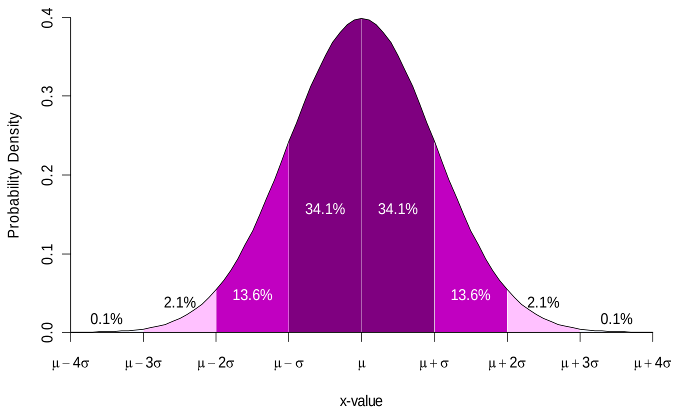


Figure 1.5 : Probability density function of the Gaussian (normal) distribution.

noise reduction algorithms are given below. The probability that a randomly selected value of a variable x falls between the values a and b is defined as:

$$P(a \le x \le b) = \int_a^b f(x)\,dx \tag{1.6}$$

Let z be the *z-score* defined as:

$$z = \frac{(x - \mu)}{\sigma} \tag{1.7}$$

The Chebychev theorem states that for any population or sample, the proportion of observations, whose *z-score* has an absolute value less than or equal to k, is no less than $(1 - (1/k^2))$:

$$P(x \leq k) \geq 1 - \frac{1}{k^2} \tag{1.8}$$

In case of Gaussian distribution, the Chebychev theorem can be further refined. In particular the following properties hold:

$$\begin{array}{l} P(-1 \leq z \leq 1) = \int_{-1}^{1} f(z)\,dz = 68.27\% \\ P(-2 \leq z \leq 2) = \int_{-2}^{2} f(z)\,dz = 95.45\% \\ P(-3 \leq z \leq 3) = \int_{-3}^{3} f(z)\,dz = 99.73\% \end{array} \tag{1.9}$$

In other words:

- 68% of the samples fall within -1 and +1 standard deviations from the mean;
- 95% of the samples fall within -2 and +2 standard deviations from the mean;
- 99% of the samples fall within -3 and +3 standard deviations from the mean.

This implies that there is a small probability that a normally distributed variable falls more than two times standard deviations away from its mean. This noise model is representative of the *small* oscillations that are observed in the pixel values. It must be observed however, that for high levels of noise, the Gaussian distribution bell becomes significantly wide, eventually degenerating to a fat tailed distribution which causes increase of color noise and leaky pixels.

1.7 Impulse Noise Model

Image sensors are composed of millions of photodiodes collecting photons. Faulty elements in the sensor array may occur, generating pixels that do not record correct information. The single isolated defective pixels located in random spatial positions of the imager are referred as *impulse noise*.

The defective nature of a pixel can be classified into two main classes: *fixed*-valued and *random*-valued impulse noise. The following definition shows the *fixed*-valued impulse noise *pdf*:

$$f(x) = \begin{cases} f_a & if \quad x = a \\ f_b & if \quad x = b \\ 0 & \text{otherwise} \end{cases} \tag{1.10}$$

For a 8-bit image, $a = 0$ yields black pixels in the image *(dead pixels)*, and $b = 255$, produces clipped values *(spikes)*. Pixels affected by fixed-valued impulse noise always appear defective unless they are masked by texture, and they can be corrected using a defect map, which stores the position of the faulty elements. The correction stage uses information from the neighboring pixels. Fig.1.6 shows an image contaminated by fixed-valued impulse noise.

Leaky pixels do not respond well to light, rather, their response is uncertain, causing *random*-valued impulse noise (i.e., impulse noise with variable amplitude). The behavior of leaky pixels is not constant and varies according to external factors such as temperature; this extra uncertainty makes leaky pixels position almost unpredictable. For an image contaminated with impulse noise, the impulse noise ratio Q can be defined as:

$$Q = \frac{Number\ of\ impulse\ defective\ pixels}{Total\ number\ of\ pixels} \tag{1.11}$$

The position of the defects and their amplitude are two independent quantities, hence, the map of defects D is defined as the point by point multiplication between D_{POS} and

(a) Clean Image

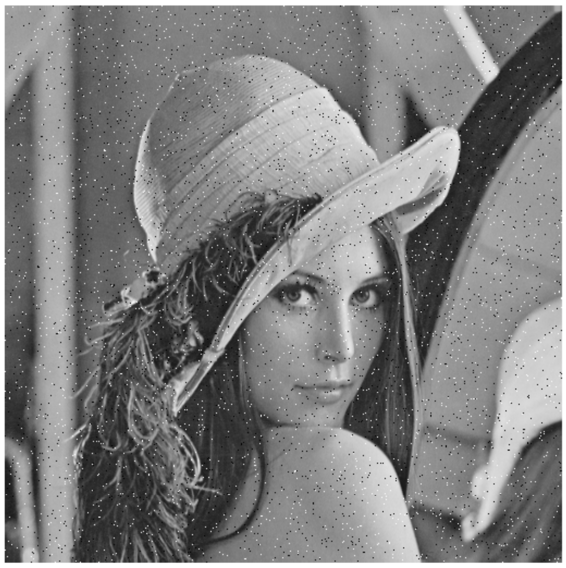

(b) Fixed-Valued impulse noise

Figure 1.6 : Impulse noise.

D_{AMP} [43]:

$$D = D_{POS} \cdot D_{AMP} \tag{1.12}$$

where:

- D_{POS} is $M \times N$ binary matrix mapping the positions of the impulse noise;
- D_{AMP} is $M \times N$ representing the amplitudes of the impulse noise at each pixel position;

The following probabilities can be then defined:

$$\begin{array}{c} P\{D_{POS}\ (x,y) = 1\} = n \\ P\{D_{POS}\ (x,y) = 0\} = 1 - n \end{array} \tag{1.13}$$

with $x = 1, \ldots, M, \quad y = 1, \ldots, N$ and $0 \leq n \leq 1$. Binary distribution (1.13) indicates that position (x, y) is faulty with probability n and correct with probability $1 - n$. The correction of impulse noise can incur into three classes of errors:

- Type I: this type of errors simply refer to the case in which a defective element is not detected (false negative); this error causes a visible, not corrected, defect in the final image unless it is masked by texture.
- Type II: a pixel not affected by impulse noise is erroneously classified as defective and corrected (false positive).Type II errors occurring in textured areas of the image cause loss of detail because important information related to sharpness is lost after correction. False positives in homogeneous areas are not a problem because overcorrecting a homogeneous area does not produce visible artifacts.
- Type III: a defective pixel is correctly classified and corrected, nonetheless its correction augments defectivity (overcorrection problem). This category of errors is more subtle and refers to the case in which the correction of the defect produces a new value which is more defective and visible than the previous one.

As the pixel size decreases and the operating conditions of the imager become critical (e.g., high temperature, low light, etc.) the probability of occurrence of adjacent defective pixels augments. For example, adjacent leaky pixels, in certain conditions behave as couplets of defective pixels, particularly visible and annoying in uniform and dark backgrounds (heavy tailed noise).

Couplets are difficult to remove because two adjacent defective elements may be considered as part of an edge and not corrected. To cope with this problem, ad-hoc defect correction algorithms must be used or properly tuned defect maps have to be built [6]. Fig.1.7 shows the defective and filtered version of a CFA Bayer image (see Section 1.3) in false color. Strong defect correction, such as heavy median filtering, can cause significant resolution loss generating unpleasant blurred images.

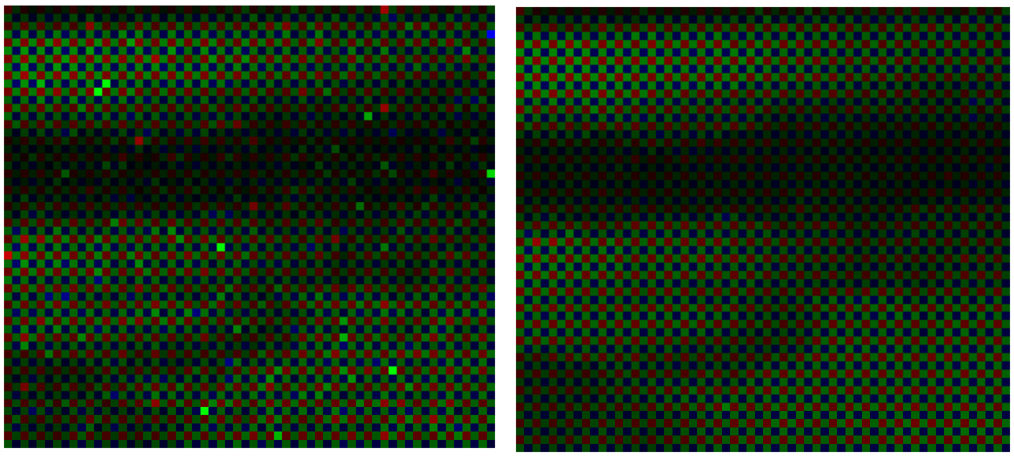

(a) Colorized defective Bayer image. (b) Colorized Filtered Bayer image.

Figure 1.7 : Defective Bayer image.

1.8 Noise in Image Pipeline

Noise can change significantly its intensity and statistics in the different stages of the pipeline. In fact, noise which is superimposed on the image signal during the acquisition phase, has a standard deviation that changes because of the influence of each processing algorithm (see Fig.1.1). Despite the efforts in reducing noise introduced during the acquisition process, the residual unfiltered noise may be amplified in the subsequent image pipeline processing steps. This is a problem especially in low light conditions, when amplification gains are used in order to produce an acceptable picture.

Noise reduction algorithm can take place in different stages of the image processing pipeline. In order to keep noise levels low, it may be necessary to perform more than a single noise reduction stage. Unfiltered sensor noise can also introduce artifacts in colors that are difficult to recover after demosaicing, because color interpolation mixes noises of different color channels, increasing signal and noise correlation. Algorithms in Fig.1.8 [26] and their side-effects on the noise distribution [4] are described in detail in the following subsections.

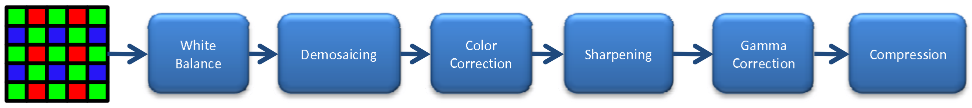


Figure 1.8 : Image pipeline .

1.8.1 White Balance Effects

The first block having a high impact in noise amplification is the White Balance *(WB)*. Before *WB* application, the noise levels basically depend only on the pixel values, as shown in Fig.1.10. However the *WB* algorithm typically applies three different global gains (one for each CFA channel) in order to compensate the amounts of red, green and blue such that the neutral colors are represented correctly [4]. Let I_R, I_G, I_B be the red, green and blue pixels of the CFA image respectively. Let g_R^{WB}, g_G^{WB}, g_B^{WB} be the gains applied to each CFA color channel according to:

$$\begin{array}{c} I_R^{WB} = g_R^{WB} I_R \\ I_G^{WB} = g_G^{WB} I_G \\ I_B^{WB} = g_B^{WB} I_B \end{array} \tag{1.14}$$

Hence, the noise variance (σ_n^2) in each CFA plane is modified in the following ways:

$$\begin{array}{c} \sigma_n^2\left(I_R^{WB}\right) = g_R^{WB}\sigma_n\left(I_R\right)^2 \\ \sigma_n^2\left(I_G^{WB}\right) = g_G^{WB}\sigma_n\left(I_G\right)^2 \\ \sigma_n^2\left(I_B^{WB}\right) = g_B^{WB}\sigma_n\left(I_B\right)^2 \end{array} \tag{1.15}$$

1.8.2 Demosaicing Effects

The demosaicing process allows recovering the color image from the interspersed samples of the Bayer pattern. The algorithm chosen to reconstruct the color image impacts the noise levels because of changes in the spatial correlation of data. To show the effects of demosaicing on noise level a simple algorithm which recovers the color component at location (x,y) by averaging the available color components in the neighborhood is employed. For example, if the current (x,y) is the site of a green sample (I_G), the missing red $(I^*{}_R)$ and blue $(I^*{}_B)$ components are recovered by using:

$$\begin{array}{c} I_R^*\left(x,y\right) = \frac{I_R^{WB}(x-1,y) + I_R^{WB}(x+1,y)}{2} \\ I_B^*\left(x,y\right) = \frac{I_B^{WB}(x-1,y) + I_B^{WB}(x+1,y)}{2} \end{array} \tag{1.16}$$

The noise variance related to the red and blue components interpolated at pixel (x,y) are defined as ([4]):

$$\begin{array}{c} \sigma_n^2\left(I_R^*\left(x,y\right)\right) \cong \frac{\sigma_n^2\left(I_R^{WB}(x-1,y)\right)}{2} \\ \sigma_n^2\left(I_B^*\left(x,y\right)\right) \cong \frac{\sigma_n^2\left(I_B^{WB}(x-1,y)\right)}{2} \end{array} \tag{1.17}$$

σ_n^2 is scaled by a factor of two because of the average operation on data. The spatial correlation of noise also increases.

1.8.3 Color Correction Effects

Color correction is necessary because the response of the color filters placed on top of the imager do not match the one of the human eye; consequently, the RGB values must be corrected using a proper 3×3 matrix that adjusts the values accordingly. This multiplication changes the pixel values but, meanwhile, increases noise and reduces the SNR, especially in the blue channel:

$$\begin{bmatrix} I_R^c \\ I_G^c \\ I_B^c \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} I_R^* \\ I_G^* \\ I_B^* \end{bmatrix} \tag{1.18}$$

noise variance σ_n^2 for each color channel change in the following way ([4]):

$$\begin{array}{l} \sigma_n^2\left(I_R^c\right) = c_{11}\sigma_n^2\left(I_R^*\right) + c_{12}\sigma_n^2\left(I_G^*\right) + c_{13}\sigma_n^2\left(I_B^*\right) \\ \sigma_n^2\left(I_G^c\right) = c_{21}\sigma_n^2\left(I_R^*\right) + c_{22}\sigma_n^2\left(I_G^*\right) + c_{23}\sigma_n^2\left(I_B^*\right) \\ \sigma_n^2\left(I_B^c\right) = c_{31}\sigma_n^2\left(I_R^*\right) + c_{32}\sigma_n^2\left(I_G^*\right) + c_{33}\sigma_n^2\left(I_B^*\right) \end{array} \tag{1.19}$$

1.8.4 Sharpening, Gamma Correction and Compression Effects

The demosaicing process reconstructs the full RGB color image starting from the Bayer samples; this process is essentially a low-pass operation, hence the output image is weak in terms of sharpness and it looks blurred; therefore a sharpening algorithm is mandatory in order to obtain an acceptable image. Subsequently a gamma correction algorithm is also applied, to align the linear response to light intensity of the imager to the nonlinear response of the human visual system.

Sharpening and gamma correction algorithms improve image quality, but increase noise as well. The sharpening algorithm amplifies the high frequencies, consequently increasing the image noise. Gamma correction modifies luminance values to enhance contrast in the dark regions; due to its nonlinearity, it makes the noise distribution for each signal level even more complex to describe. Compression is used to reduce the size of image files. There are two types of compression: *lossless*, where the amount of image data is reduced without loss of information, and *lossy* (e.g., *JPEG*) where image file is reduced allowing the lost of a part of data.

JPEG converts the *RGB* image in the *YCbCr* color space; the luminance plane *(Y)* is used for recognizing structures and details while the chrominance planes *(CbCr)* are subsampled without significant loss of image information for the observer. The *Y* plane is divided in 8×8 blocks that are compressed separately and transformed in the frequency domain using *DCT (Discrete Cosine Transform)* . *DCT* coefficients are then quantized using a quantization table. Compression rate depends on the used quantization table; the higher the compression rate the lower the image quality, due to presence of artifacts. Lossy image compression is similar to a noise reduction algorithm, that maintains the main image structures and suppresses fine textures and details. Anyway, artifacts introduced by compression reduce global image quality.

1.9 Noise Reduction Block Position

According to the previous considerations about the image pipeline, the position of the noise reduction stage strongly affects the quality of the final image. Basically, in order to avoid false colors and increment of noise spatial correlation, it is important to implement noise reduction before demosaicing. Nonetheless, as discussed above, not all noise can be removed before (or jointly to [31]) demosaicing; the residual noise is further amplified by the color correction and sharpening algorithms, hence a new application of noise reduction is generally required at the end of the pipeline. Fig.1.9 shows a possible image processing pipeline with two noise reduction stages. The first denoising stage is applied in the *CFA* domain, before demosaicing; the second noise filtering stage works in the luminance domain and is positioned at the end of pipeline before compression.

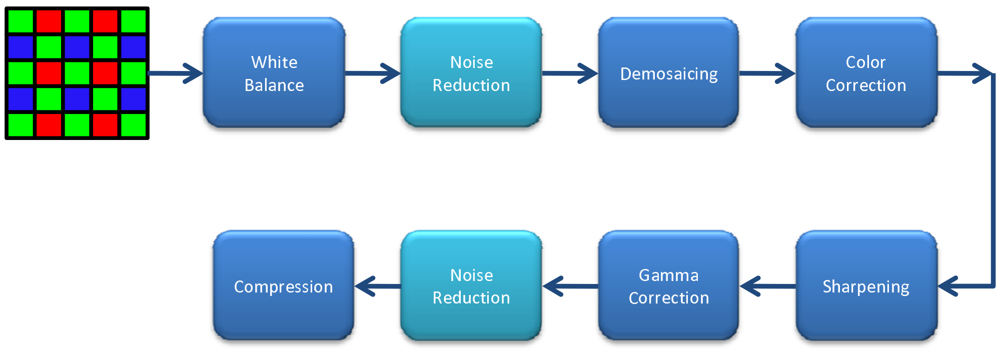


Figure 1.9 : Noise reduction blocks in Image pipeline.

1.10 Signal Dependent Noise Model

In previous sections we have dealt with the noise, assuming that is additive Gaussian (AWGN), as well as most of the literature dealing with this topic. We have also seen that noise contaminating raw images is the juxtaposition of many different sources that overlap the ideal signal. In particular we could distinguish the noise sources into two groups:

1. Noise sources that are dependent on irradiance (Poisson distributed);

2. Noise sources caused by the electric and thermal noise of the imager (Gaussian distributed).

Photon shot noise belongs to the first group, because it is closely tied to the number of photon that reaches the sensor. Therefore, the standard deviation of photon shot noise is Poissonian and depends on signal intensity. The other noise sources, such as read-out noise, thermal noise, amplifiers noise, quantization noise, etc. are usually modeled by considering that their overlap is Gaussian distributed.

Hence, the sum of all sources of noise has a distribution that varies with the signal intensity. The standard deviation of the signal dependent noise (SDN) can be modeled using an equation of the form [14]:

$$\sigma(x) = \sqrt{a \cdot x + b} \tag{1.20}$$

where: x is signal intensity and $a, b \in \Re^+$, are constants related to the slope of the curve; these parameters change depending on quantum efficiency, pedestal and analog gain [14]. Therefore , given a specific image sensor, its intrinsic noise contribution can be estimated considering its behaviour under different illumination conditions. This implies computing a and b coefficents in (1.20) for each sensor operating analog gain, starting from observed noise samples. Fig.1.10 also shows that signal dependent noise standard deviation is quite similar for each Bayer color channel.

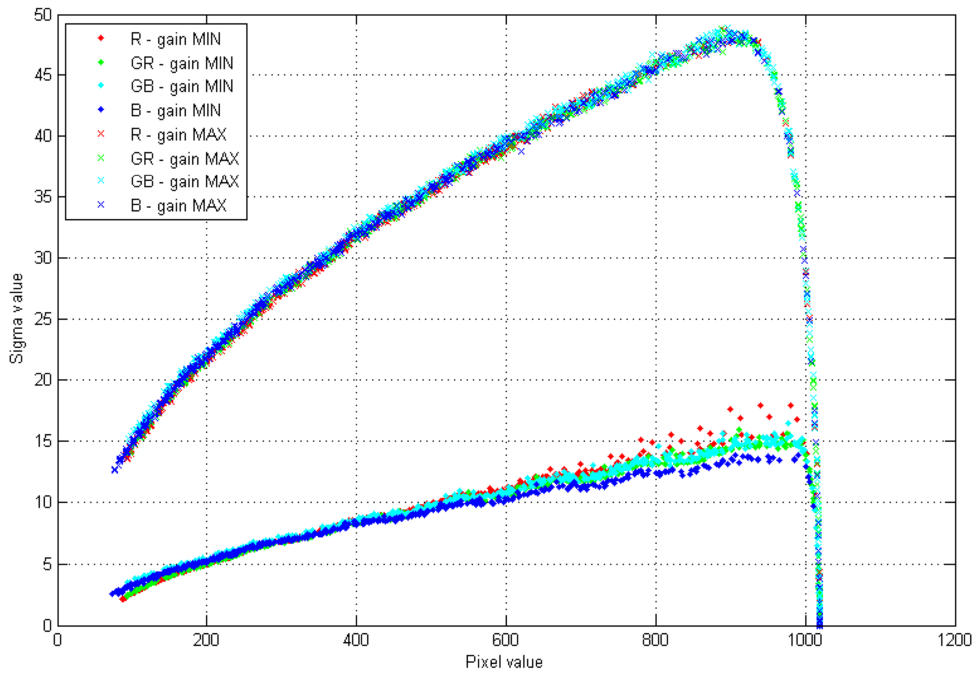


Figure 1.10 : Noise curves in the Bayer domain for a 10 bit image acquired by a sensor operating at two extreme analog gains. Lower curve represents noise levels at minimum analog gain; upper curve represents noise levels at maximum analog gain. It is clear how noise increases with the signal level and the operating analog gain.

Chapter 2. Noise Removal Techniques

2.1 Introduction

The main purpose of an image denoising algorithm is to preserve image details as much as possible while eliminating noise. Typically, denoising filter based on AWGN noise model, requires the estimation of standard deviation (σ) in order to calibrate the filtering intensity. Therefore, a noise removal procedure requires that two different processing algorithms are performed on the image: one to estimate the noise level and the actual filtering process.

In this chapter we first introduce a metric to compute numerically the quality of an image, focusing subsequently our attention on the algorithms for noise estimation and filtering. In particular, the basic concepts for the implementation of some noise estimation algorithms are given, showing in detail the description of a simple procedure [5]. An overview of the most well-known filtering techniques is then presented, providing a detailed explanation of two algorithms such as: *Sigma Filter* [6] and *Bilateral Filter* [39] which are also reported in the Chapter 5 adapted to use signal dependent noise estimation.

2.2 Noise Metrics

An important measures dealing with noise filtering is *SNR* (Signal to Noise Ratio), which is usually adopted as a simple reference measure to numerically define the image quality. It is computed as the ratio between the signal and the underlying noise and is expressed in decibel [15]:

$$SNR(S) = 20\log_{10}\left(\frac{S}{\sigma_N}\right)(dB) \tag{2.1}$$

All quantities are measured in electrons. The term S represents the signal level while σ_N represents the noise standard deviation. More specifically σ_N is defined as sum of different kinds of noise: $\sigma_N = \sigma_S + \sigma_R + \sigma_{DSNU} + \sigma_{PRNU}$ where σ_S, σ_R, σ_{DSNU}, σ_{PRNU} are shot noise, read noise, dark signal non uniformity and photon response non uniformity respectively.

After acquisition and digital conversion, the image is coded into L levels, where L depends on the bit depth of the Analog to Digital *(A/D)* conversion process. Hence, the *SNR* of an image $I(x,y)$, in this case, is defined as:

$$SNR(I) = 20\log_{10}\left(\frac{E(I)}{\sigma(I)}\right)(dB) \tag{2.2}$$

where $E(I)$ and $\sigma(I)$ are the average value and the standard deviation of the image I respectively. The higher the SNR, the better the image.

2.3 Noise Estimation

As discussed in Chapter 1, the zero mean *AWGN* noise model requires the estimation of the standard deviation of the underlying Gaussian noise distribution. Pixels deviate from their correct value by some value which is drawn from a Gaussian distribution; usually pixel fluctuations are small, but greater fluctuations are also possible. Nonetheless, in 99% of the cases, the deviations do not exceed 3-times σ in absolute value. Large noise amplitudes generated by the distribution tails are possible; in this case the pixel of interest might appear as a spike or dead element. The knowledge of a good σ estimation allows filtering the image data properly, significantly reducing the unpleasant effects of Gaussian noise. Furthermore, σ can also be a reference value for detecting outliers.

Compared to the wide literature on image denoising, the literature on noise estimation is very limited. Noise can be estimated from multiple images or a single image. Estimation from multiple image is an over-constrained problem [17], while the estimation asedo on a single image, is an under-constrained problem and requires further assumptions for the noise.

Olsen [29] analyzed six methods for noise standard deviation estimation and showed that the best was the average method, which is also the simplest. Average method consists of filtering the data I with the average filter (a simple box function) and subtracting the filtered image from I. Then a measure of the noise at each pixel is computed. To avoid contribution of image edges to the estimate, the noise measure is rejected if the magnitude of the intensity gradient is greater than a fixed threshold, T.

Estimation of noise standard deviation is based on the following general ideas:

- Locate homogeneous areas in the image, because in flat areas pixel fluctuations are supposed to be caused exclusively by random noise.
- Compute the local variance in the detected flat areas.
- Repeat the previous two steps until the whole image has been processed.
- Estimate the signal variance using the accumulated noise samples.

Therefore, noise estimation algorithms often rely on texture detection techniques, among these are: Amer-Dubois method [3], that uses a set of highpass filters to detect signal activity in the horizontal, vertical, diagonal and corner directions; Kim-Lee technique [44] that is based on a more sophisticated approach in that it tries to differentiate image areas with same standard deviations but generated by patterns not related to random signal fluctuations; finally, Staelin-Nachlieli [11] which propose to estimate noise in areas where cross channel color correlation is low. In the next subsection we present in detail a simple iterative method for estimating the noise [6].

2.3.1 Fast Noise Estimation

A rough approximation of the noise level in an image can be obtained by exploiting the statistical properties of the Gaussian noise distribution (1.9). It is reasonable to suppose that the image cannot contain an arbitrary high noise level [6]; hence initially noise level is set to the maximum (σ_{max}); this value is obtained using a tuning phase in which the behavior of the image sensor is characterized under different illumination conditions.

Assuming a 3×3 filter support, the absolute differences δ_0, δ_1, $\ldots$,δ_7 between the central pixel P_c and its neighborhood are computed:

$$\delta_i = |P_c - P_i| \qquad i = 0, \ldots, 7 \tag{2.3}$$

If $\delta_i \in [0, \sigma_{max}], i = 0, \ldots, 7$ then the assumption of having localized a homogeneous area can be made. The idea is to build a noise histogram Ψ that accumulates the collected noise samples in its bins. Let γ_j be the value of the j-th absolute difference $\delta_j, j \in [0, \ldots, 7]$ when $\delta_i \in [0, \sigma_{max}], i = 0, \ldots, 7$; in this case the bin γ_i in Ψ is incremented:

$$\Psi(\gamma_j) = \Psi(\gamma_j) + 1 \tag{2.4}$$

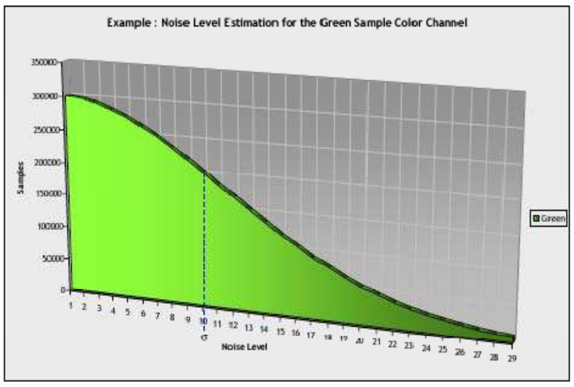
After processing the entire frame, the absolute differences accumulated in the histogram will be Gaussian-like shaped. Because of the absolute values, only the positive side of the x-axis is filled; this is not a problem because the normal distribution is symmetric around its mean value, which is zero in our case.

The noise standard deviation is determined considering the property of the Gaussian distribution stating that 68% of its samples fall in the interval $[\mu - \sigma, \mu + \sigma]$. The histogram of the absolute differences is integrated until the 68% of the total samples has been accumulated. As soon as the histogram integrations stops, the value on the x-axis which attains the 68% of the total samples, represents the estimated noise standard deviation (Fig.2.1):

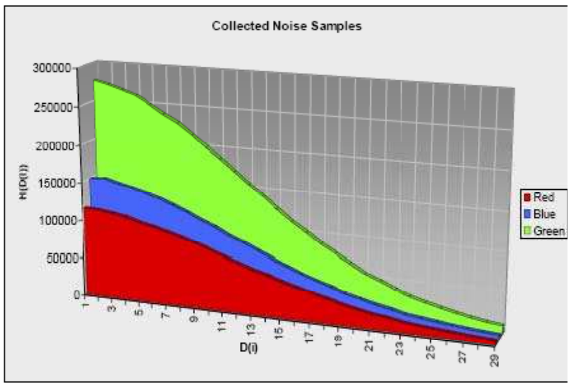
$$\sigma_{est} = \{max\ k | \sum_{i=1}^{k} \Psi(i) \leq \lceil 0.68 \cdot \Sigma_{samples} \rceil \} \tag{2.5}$$

This solution is strongly based on the value originally chosen for σ_{max}. This number has to be carefully generated by performing a tuning phase which consists in testing the image sensor under different light conditions and determining the typical worst case noise situations. These noise levels will set an upper bound for $3\sigma_{max}$. Nonetheless, the gathered sample of the noise population could be contaminated by the real signal. A possible solution that can minimize the bias problem is to allow σ_{max} to change over time; if σ_{max} is initially overestimated, then σ_{max} can be decreased for the next iteration. This allows progressively reducing the sample bias and converging to the optimal estimation. The method can be further refined using a more sophisticated texture detector, like the one described in [3].

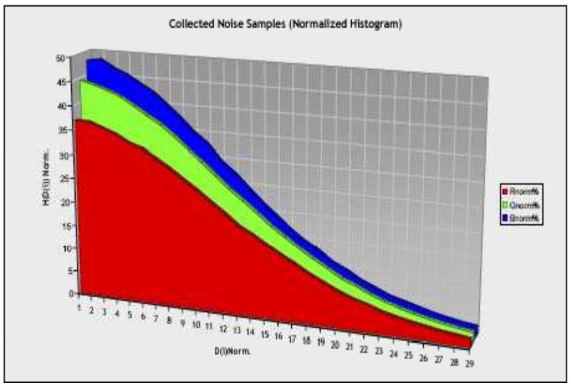
Though the estimation is not perfect and may be biased, it is anyway an approximation, indicating the overall noise level. On a CFA image this method will generate a single σ value for each color channel.



(a)



(b)



(c)

Figure 2.1 : Noise level estimation using noise histogram Gaussian-like distributed.

2.4 Noise Filtering

The noise filtering problem can be described as the process of effectively removing the unwanted noisy component from the acquired signal, restoring the original ideal data, without sacrificing the image sharpness and features (i.e., color component distances, edges, sharpness, etc.). In particular, a robust filtering algorithm should satisfy the following conditions [10]:

- *Flat* regions should be as smooth as possible. Noise should be completely removed from these regions.
- *Image boundaries* should be well preserved. This means the boundary should not be either blurred or sharpened.
- *Texture details* should not be lost. This is one of the hardest criteria to match. Since image denoising algorithm always tends to smooth the image, it is very easy to lose texture details in denoising.
- The *global contrast* should be preserved, or the low-frequencies of the denoised and input images should be identical.
- No *artifacts* should be produced in the denoised image.

The global contrast is probably the easiest to match, whereas some of the rest principles are almost incompatible. For instance, (a) and (c) are very difficult to be tuned together since most denoise algorithms could not distinguish flat and texture regions from a single input image.

Filtering techniques differ in the choice of image prior models and there is a very large part of literature that deals with this topic. Some of the most important denoising approaches are:

- *Wavelet* ([27, 34]): image is decomposed into multiscale oriented sub-bands. The filtering process typically acts applying an (hard or soft) threshold on wavelet coefficients.
- *Anisotropic Diffusion* ([32]): is a non-linear and space-variant transformation of the original image. This method remove noise solving an isotropic heat diffusion equation (a second order linear partial differential equations).
- *Non-Local Mean* ([8]): uses multiple pictures and take the mean to remove the noise. This method is unpractical for a single image, but a temporal mean can be computed from a spatial mean as long as there are enough similar patterns in the single image.
- *Bilateral Filtering* ([39]): is an adaptive Gaussian filtering, that, to preserve edges, takes into account both space and range distances.

The following sections describe some techniques for the removal of some specific noise disturbs such as: fixed pattern noise [7] and temporal random noise by using spatial filtering methodology [6, 39].

2.4.1 Column-FPN Filtering

The column-Fixed Pattern Noise *(FPN)* is caused by column amplifiers, and appears as vertical stripes in the image (see Section 1.3.1). Since FPN is equal in all acquisitions, for its effective cancellation, it is necessary to estimate its signature that, once learned, can be subtracted from the image data.

FPN estimation can be performed using supplementary data provided by the image sensor. As Fig.2.2 depicts, a series of black and dark lines are placed at the top of the imager, that are not shown in the final color pictures. Black lines have zero integration time while dark lines have the same exposure time as the image lines but they are shielded from the incident light. These considerations imply that:

- black lines contain very little noise (specifically, FPN noise only);
- dark lines accumulate almost the same random noise as the image, because they have the same integration time of the image lines.

The FPN cancellation is achieved by continuously averaging the black sampled data, according to the following equation:

$$FPN_{Est} = FPN_{Est}(FPN_{Est}/Leak_C) + (FPN_{CurSample}/Leak_C) \qquad (2.6)$$

where:

- $Leak_C$: is a constant to weight the previous estimation.
- FPN_{Est}: is the estimation of the FPN signature.
- $FPN_{CurSample}$: is the FPN signature, extracted from the current frame.

Denoting with nb the number of black lines and with W the image width, the current estimation, $FPN_{CurSample}$, for the FPN of image I is obtained by averaging each column j of the black lines:

$$M_j = \frac{\sum_{i=0,1,\dots,nb} I(i,j)}{nb} \qquad j = 1,\dots,W \qquad (2.7)$$

FPN_{Est} is initialized to zero and is updated by means of equation (2.6), each time a new frame arrives. The first estimation, computed on the first frame, is merely a coarse approximation of the real FPN signature. After some iterations the estimation

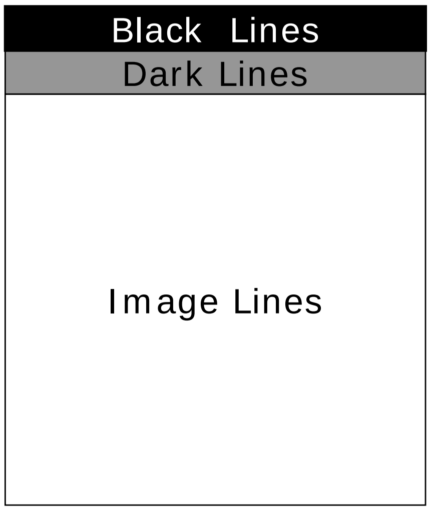


Figure 2.2 : Black lines are used for FPN estimation, Dark lines for random noise estimation.

converges towards the correct signature that must be row-wise subtracted from the image data in order to get rid of the FPN. The $Leak_C$ value defines how much weight is attributed to the previous estimations; by changing this value, the speed of convergence can be modulated.

The number of black lines used to learn the signature is a key element of the algorithm. If a low number of black lines is used, the estimation would be not reliable, as noise would generate uncertain approximations. On the other hand, using more lines than necessary is a useless waste of resources, both on the sensor and from a computational point of view. Thus, a trade-off between the number of black lines and the leak factor value must be found.

2.4.2 Spatial Filtering

Spatial filters are based on low pass filtering of neighboring pixels under the assumption that the noisy part of the signal is located in its high frequencies. Spatial filters can be partitioned into two main classes: linear and non-linear filters [15]. Linear filters, such as the mean filter, are weak in terms of image details preservation and cannot be successfully adopted for removing noise without blurring the image. Other simple non-linear filters such as the median filter are also weak in terms of detail preservation, basically because this filter applies the same processing without explicitly identifying noise. Nonetheless a median filter has good response in cases in which the noise distribution has long tails.

A vast variety of spatial filters exist and covering them is out of the scope of this Section [6]. Follows a short description of two widely used filtering methods, known as *Sigma Filter* [6] and *Bilateral Filter* [39]. In particular, these algorithms are also reported in the Chapter 5 in a modified version, adapted to use a signal dependent noise estimation.

2.4.3 Sigma-Filter

If a reliable noise estimator is available, the *Sigma-Filter* [6] represents a fast solution for reducing noise. The filtering process is based on the assumption that the observed pixel value *N(x,y)* is a good estimate of the local signal mean. The observed pixel value $N(x,y)$ can be expressed as the sum of its representative mean η plus a Gaussian noise term Γ:

$$N(x,y) = \eta(x,y) + \Gamma(x,y) \tag{2.8}$$

We then consider a value $\delta = 3\sigma$ and consider all the pixels in the range delimited by the central pixel value $\pm\delta$. Under the assumption of zero mean *AWGN*, this range includes ˜99% of the distribution from the same class as the central pixel.

Let M be a $m_1 \times m_2$ filter mask and P_c the value of its central pixel. The final output is a weighted average of the pixels having value close to one of the mask central pixel. Weights decrease as the distance in intensity between the central pixel and the neighborhood augments. Under the assumption of Gaussian noise model, the Sigma Filter averages all the pixels whose values fall in the range $[P_c - 3\sigma, P_c + 3\sigma]$. In particular, pixels whose distance falls in the range $[P_c - \sigma, P_c + \sigma]$ receive maximum weight W_{max}. Pixels whose value falls in the range $[(P_c - \sigma) - \sigma, (P_c + \sigma) + \sigma]$ are weighted with medium weight W_{mid}. Finally, pixels whose intensity falls in the range$[(P_c - 2\sigma) - \sigma, (P_c + 2\sigma) + \sigma]$ are weighted with minimum weight W_{min}. Pixels outside of the range $[(P_c - 3\sigma), (P_c + 3\sigma)]$ are considered outliers having zero weight in the weighted average. Clearly, a reliable noise estimate is necessary, otherwise blurring or lack of noise reduction effectiveness can occur, depending on sigma over- or under estimation respectively. The final weighted average P_f can be expressed as the sum of the mask pixels multiplied by their respective weights and divided by the sum of the weights:

$$P_f = \frac{\sum_{i=0}^{i \leq (m_1 \times m_2) - 1} W_i \cdot P_i}{\sum_{i=0}^{i \leq (m_1 \times m_2) - 1} W_i} \tag{2.9}$$

The selection of the range $[(P_c - 3\sigma), (P_c + 3\sigma)]$ excludes shot noise pixels and pixels outside a local edge, maintaining sharp edges and allowing effective noise suppression in homogeneous areas. On the other hand, the preservation of sharp edges and strong filtering strength in flat areas also becomes a weakness of this filter. The main problem is that the *Sigma-Filter* has a strong inclusion/exclusion rule in the average

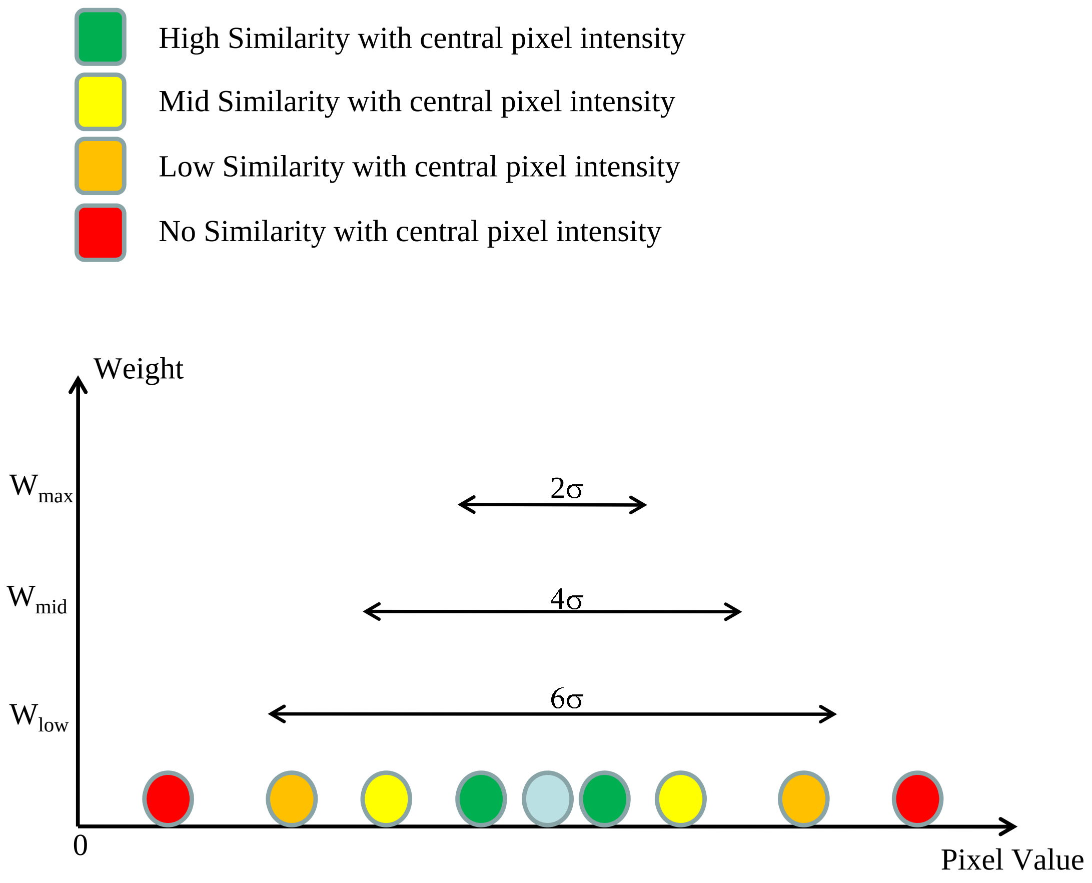

High Similarity with central pixel intensity
Mid Similarity with central pixel intensity
Low Similarity with central pixel intensity
No Similarity with central pixel intensity
Weight
W_{max}
W_{mid}
W_{low}
2σ
4σ
6σ
0
Pixel Value

Figure 2.3 : Sigma filter.

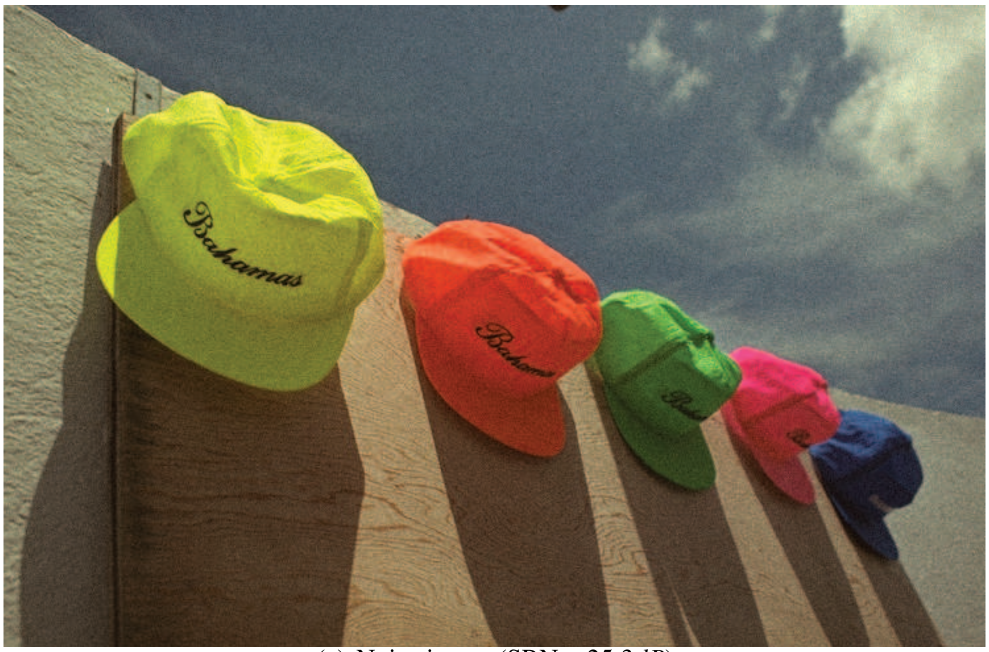
process; this, if not well controlled, adds a cartoon-like appearance to the filtered image because transitions become too abrupt [9]. In fact, if a strong edge separating two regions is present and if the grey level difference between both regions is larger than a threshold, the algorithm computes averages of pixels belonging to the same region as the reference pixel creating artificial shocks. In conclusion, the *Sigma-Filter* can create large flat zones and spurious contours inside smooth regions [9]. Not only noise must be reduced but, at the same time, it is necessary to retain a sense of sharpness, depth and focus which manifests through gradual and smooth edge transitions. The bilateral filter satisfies this requirement by applying a smooth weighting scheme in both spatial and intensity domains.

2.4.4 Bilateral Filter

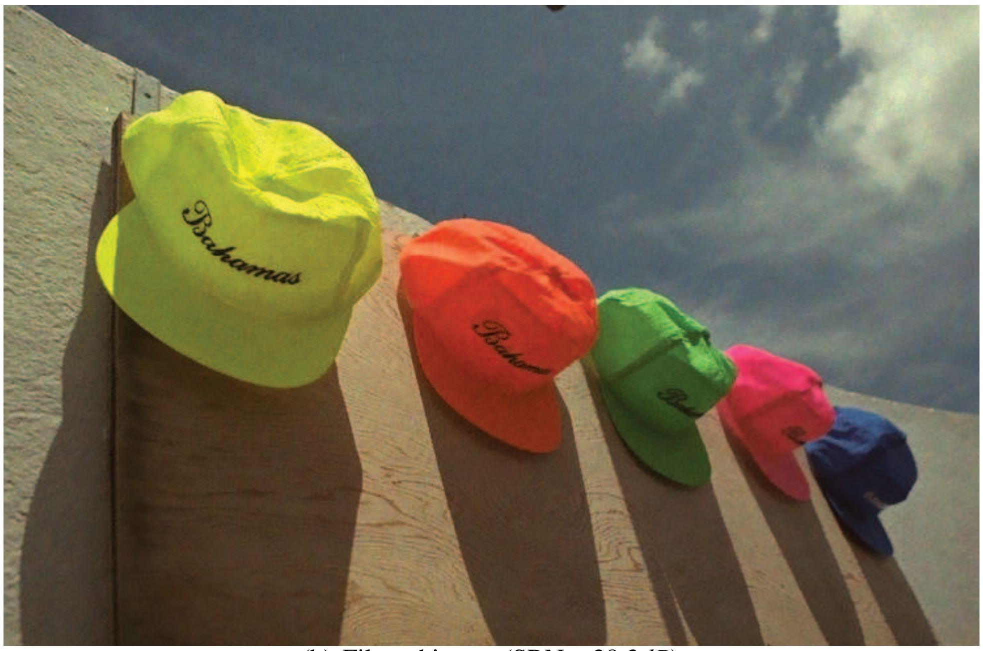
Bilateral filtering [39, 45] can be seen as an extension of the *Sigma-Filter*. Again, the noise reduction process is based on a weighted average of local samples, but in this case the filter is driven by two different standard deviation values: the intensity related σ_i and the spatial related σ_s. In analogy with the sigma filter, σ_i represents the effective noise level which depends on the pixel intensity values. The additional spatial σ_s is used to weight the pixels in the mask depending on their distance from the center of the mask. Hence, if a low σ_s is used, the pixels far from the central pixel are assigned a low weight and have less importance in the final weighted average.

An example of bilateral filtering on a 7×7 mask is shown in Fig. 2.5. At a pixel location $\overrightarrow{x}$ the output of the filter is given by:

$$I(\overrightarrow{x}) = \frac{1}{C} \sum_{y \in N(\overrightarrow{x})} e^{\frac{-\|\overrightarrow{y}-\overrightarrow{x}\|}{2\,\sigma_s^2}} \; e^{\frac{-|I(\overrightarrow{y})-I(\overrightarrow{x})|}{2\,\sigma_i^2}} \; I(\overrightarrow{y}) \tag{2.10}$$



(a) Noisy image (SRN = 25.3dB)

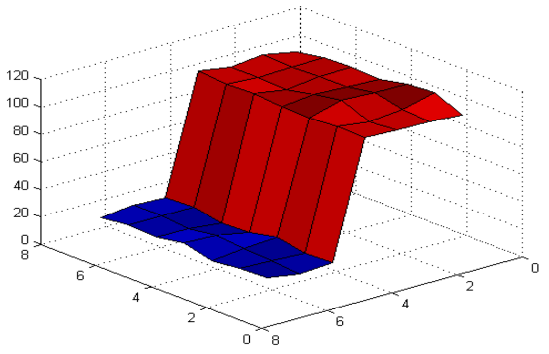


(b) Filtered image (SRN = 28.3dB)

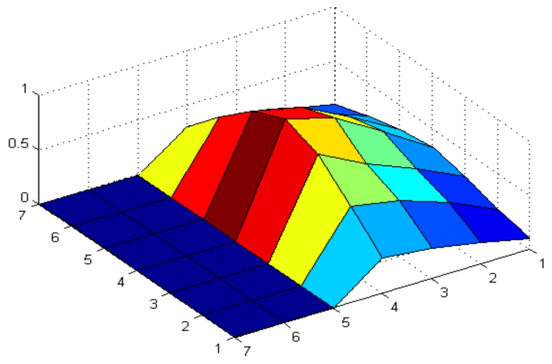
Figure 2.4 : Sigma-filter output.

where $N(\overrightarrow{x})$ is a spatial neighborhood of pixel $I(\overrightarrow{x})$ and C represents the normalization costant:

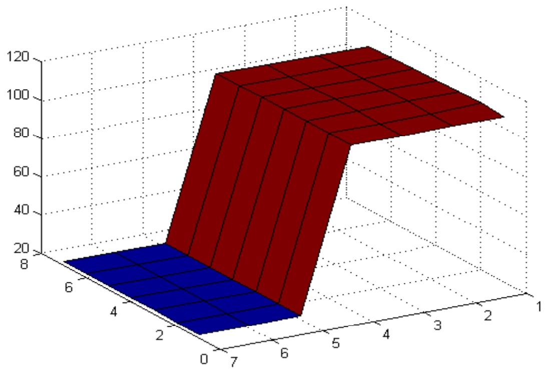
$$C = \sum_{y\ \in N(\overrightarrow{x})} e^{\frac{-\|\overrightarrow{y}-\overrightarrow{x}\|}{2\ \sigma_s^2}} \quad e^{\frac{-|I(\overrightarrow{y})-I(\overrightarrow{x})|}{2\ \sigma_i^2}} \tag{2.11}$$



(a) Noisy input

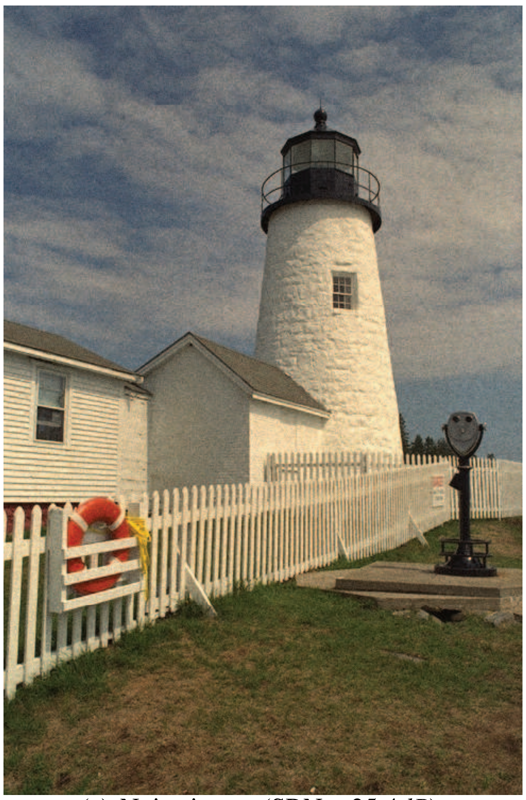


(b) Filter



(c) Filtered output

Figure 2.5 : Bilateral filter.


(a) Noisy image (SRN = 25.4dB)

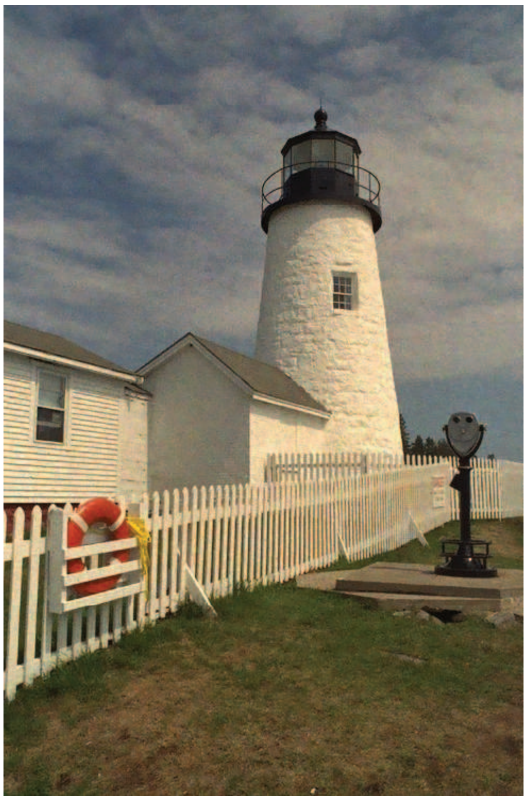

(b) Filtered image (SRN = 29.0dB)

Figure 2.6 : Bilateral filter output.

Chapter 3. Noise Reduction Exploiting HVS Behaviour

This Chapter presents a novel spatial noise reduction method [2] that directly processes the raw *CFA* (Color Filter Array) data acquired by CCD/CMOS image sensors, combining together *HVS* (Human Visual System) heuristics, texture/edges preservation techniques and sensor noise statistics, in order to obtain an effective adaptive denoising.

The proposed algorithm introduces the concept of the usage of HVS peculiarities directly on the CFA raw data from the sensor, and aims to keep low the complexity by using only spatial information and a small fixed-size filter processing window, to obtain real-time performance on low cost imaging devices (e.g., mobile phones, PDAs). The HVS properties, able to characterize or isolate unpleasant artifacts, are a complex phenomenon (highly nonlinear) not yet completely understood involving a lot of complex parameters [21, 28]. Several studies in literature are trying to simulate and code some known aspects, in order to find out reliable image metrics [30, 41, 42] and heuristics to be applied also for demosaicing [25]. The filter processes raw Bayer data, providing the best performance if executed as the first algorithm of the Image Generation Pipeline (Fig.1.8), and adapts its smoothing capability to local image characteristics yielding effective results in terms of visual quality.

3.1 Basic Concepts about the Human Visual System

It is well known that the HVS has a different sensitivity at different spatial frequencies [40]. In areas containing mean frequencies the eye has a higher sensitivity. Furthermore, chrominance sensitivity is weaker than the luminance one. HVS response does not entirely depend on the luminance value itself, rather, it depends on the luminance local variations with respect to the background; this effect is described by the Weber-Fechners law [15,28], which determines the minimum difference DY needed to distinguish between Y (background) and Y+DY. Different values of Y yield to different values of DY.

The aforementioned properties of the HVS have been used as a starting point to devise a CFA filtering algorithm. Luminance from CFA data can be extracted as explained in [24], but for our purposes it can be roughly approximated by the green channel values before gamma correction. The filter changes its smoothing capability depending on the CFA color of the current pixel and its similarity with the neighborhood pixels. More specifically, in relation to image content, the following assumptions are considered:

- If the local area is homogeneous, then it can be heavily filtered because pixel variations are basically caused by random noise.
- If the local area is textured, then it must be lightly filtered because pixel variations are mainly caused by texture and by noise to a lesser extent; hence only the little differences can be safely filtered, as they are masked by the local texture.

3.2 The Proposed Technique

3.2.1 Overall filter block diagram

A block diagram describing the overall filtering process is illustrated in Fig.3.1. Each block will be separately described in detail in the following sections.

The fundamental blocks of the algorithm are:

- **Signal Analyzer Block:** computes a filter parameter incorporating the effects of human visual system response and signal intensity in the filter mask.
- **Texture Degree Analyzer:** determines the amount of texture in the filter mask using information from the Signal Analyzer Block.
- **Noise Level Estimator:** estimates the noise level in the filter mask taking into account the texture degree.
- **Similarity Thresholds Block:** computes the fuzzy thresholds that are used to determine the weighting coefficients for the neighborhood of the central pixel.
- **Weights Computation Block:** uses the coefficients computed by the Similarity Thresholds Block and assigns a weight to each neighborhood pixel, representing the degree of similarity between pixel pairs.
- **Filter Block:** actually computes the filter output.

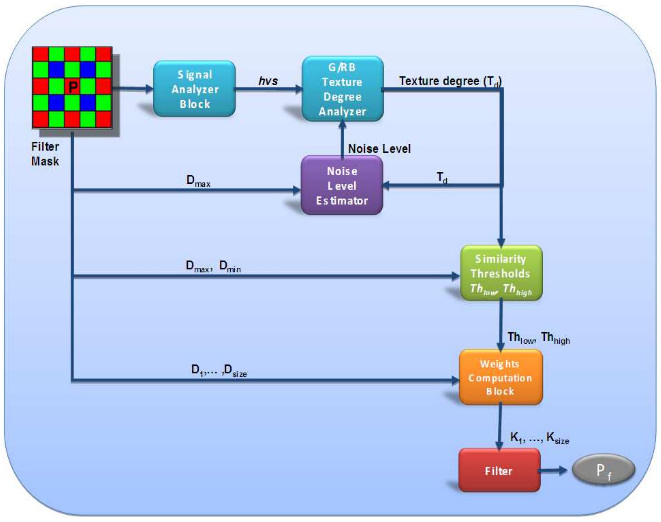


Figure 3.1 : Overall Filter Block Diagram.

The data in the filter mask passes through the Signal Analyzer block that influences the filter strength in dark and bright regions (Section 3.3 for further details). The HVS value is used in combination with the output of the Texture Degree Analyzer (Section 3.5) and Noise Level Estimator (Section 3.6) to produce the similarity thresholds used to finally compute the weights assigned to the neighborhood of the central pixel (Section 3.6). The final filtered value is obtained by a weighted averaging process (Section 3.8).

3.3 Signal Analyzer Block

As noted [12, 19] and [46] it is possible to approximate the minimum intensity gap that is necessary for the eye to perceive a change in pixel values. The base sensitivity thresholds measure the contrast sensitivity in function of frequency while fixing the background intensity level. In general, the detection threshold varies also with the background intensity. This phenomenon is known as luminance masking or light adaptation. Higher gap in intensity is needed to perceive a visual difference in very dark areas, whereas for mid and high pixel intensities a small difference in value between adjacent pixels is more easily perceived by the eye [19].

It also crucial to observe that in data from real image sensors, the constant AWGN (1.6) model does not fit well the noise distribution for all pixel values. In particular, as discussed in [13], the noise level in raw data is predominantly signal-dependent and increases as the signal intensity raises; hence, the noise level is higher in very bright areas. In [13] and [14] it is also illustrated how clipping in data is the cause of noise level underestimation; e.g., noise level for pixels close to saturation cannot be robustly tracked because the signal reaches the upper limit of the allowed bitdepth encoding.

We decided to incorporate the above considerations of luminance masking and sensor noise statistics into a single curve as shown in Fig.3.2. The shape of this curve allows compensating for lower eye sensitivity and increased noise power in the proper areas of the image, allowing adaptive filter smoothing capability in relation to the pixel values.

A high HVS value (HVS_{max}) is set for both low and high pixel values: in dark areas the human eye is less sensitive to variations of pixel intensities, whereas in bright areas noise standard deviation is higher. HVS value is set low (HVS_{min}) at mid pixel intensities. As stated in Section 3.3, in order to make some simplifying assumptions, we use the same HVS curve for all CFA colour channels taking as input the pixel intensities directly from the sensor. The HVS coefficient computed by this block is used by the Texture Degree Analyzer that outputs a degree of texture taking also into account the above considerations (Section 3.5).

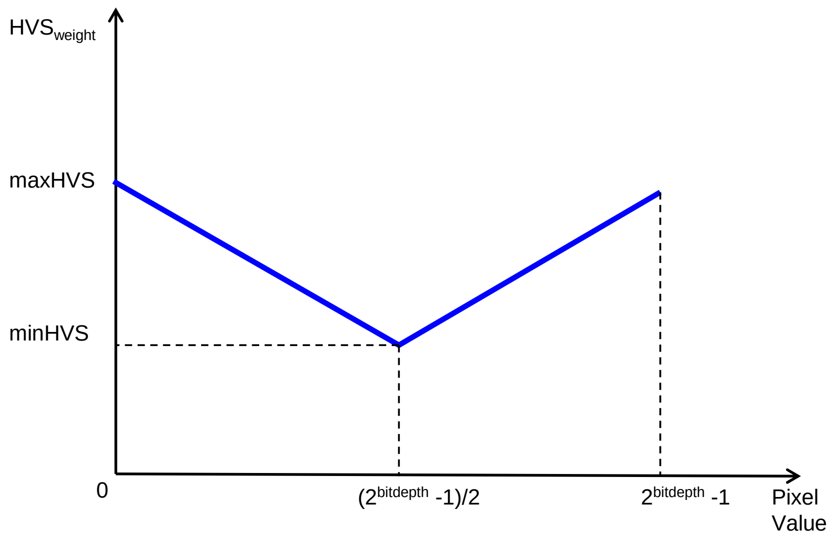


Figure 3.2 : HVS curve used in the proposed approach.

3.4 Filter Masks

The proposed filter uses different filter masks for green and red/blue pixels to match the particular arrangement of pixels in the CFA array. The size of the filter mask depends on the resolution of the imager: at higher resolution a small processing window might be unable to capture significant details. For our processing purposes a 5×5 window size provided a good trade-off between hardware cost and image quality, allowing us to process images up to 5 megapixels, a resolution that is typical of high end mobile phones. Typical Bayer processing windows are illustrated in Fig.3.3.

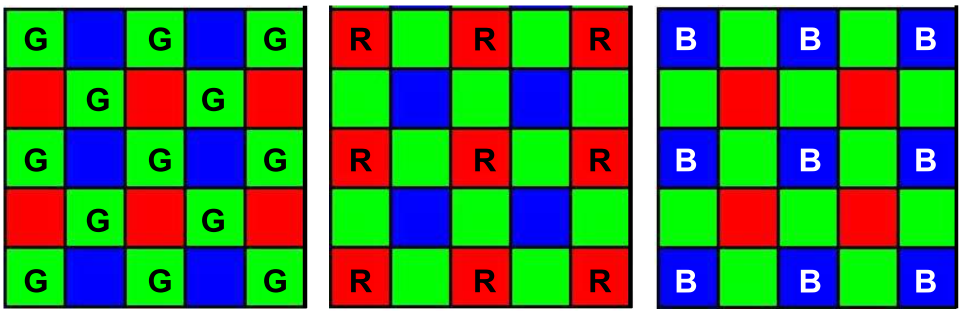


Figure 3.3 : Filter Masks for Bayer Pattern Data.

3.5 Texture Degree Analyzer

The texture analyzer block computes a reference value T_d that is representative of the local texture degree. This reference value approaches 1 as the local area becomes increasingly flat and decreases as the texture degree increases (Fig.3.4). The computed coefficient is used to regulate the filter smoothing capability so that high values of T_d correspond to flat image areas in which the filter strength can be increased.

Depending on the color of the pixel under processing, either green or red/blue, two different texture analyzers are used. The red/blue filter power is increased by slightly modifying the texture analyzer making it less sensitive to small pixel differences (Fig.3.5). The texture analyzer block output depends on a combination of the maximum difference between the central pixel and the neighborhood D_{max} and $Texture_{Threshold}$, a value that is obtained by combining information from the HVS response and noise level, as described below (3.1).

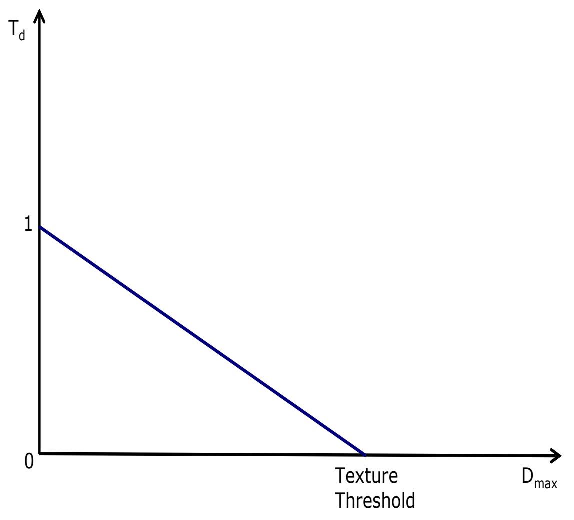


Figure 3.4 : Green Texture Analyzer.

The green and red/blue texture analyzers are defined as follows:

$$T_d(green) = \begin{cases} 1 & D_{max} = 0 \\ -\frac{D_{max}}{Texture_{Threshold}} + 1 & 0 < D_{max} \leq Texture_{Threshold} \\ 0 & D_{max} > Texture_{Threshold} \end{cases} \tag{3.1}$$

$$T_d(red/blue) = \begin{cases} 1 & D_{max} \leq Th_{R/B} \\ -\frac{(D_{max} - Th_{R/B})}{(Texture_{Threshold} - Th_{R/B})} + 1 & Th_{R/B} < D_{max} \leq Texture_{Threshold} \\ 0 & D_{max} > Texture_{Threshold} \end{cases} \tag{3.2}$$

hence:

- if $T_d = 1$ the area is assumed to be completely flat;

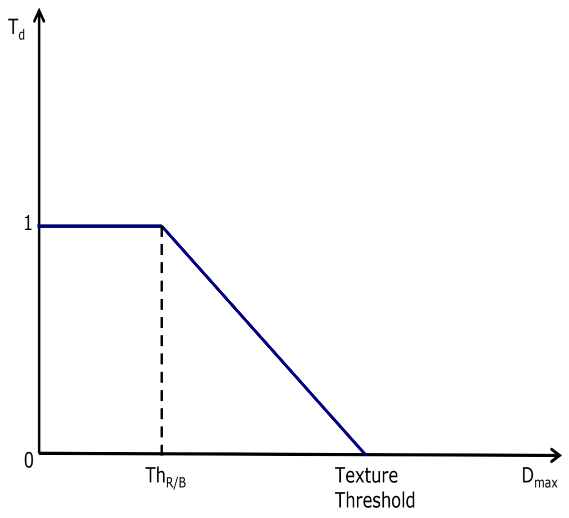


Figure 3.5 : Red/Blue Texture Analyzer.

- if $0 < T_d < 1$ the area contains a variable amount of texture;
- if $T_d = 0$, the area is considered to be highly textured.

The texture threshold for the current pixel, belonging to Bayer channel c $(c = R, G, B)$, is computed by adding the noise level estimation to the HVS response:

$$Texture_{Threshold_c}(k) = HVS_{weight}(k) + NL_c(k-1) \tag{3.3}$$

where NL_c denotes the noise level estimation on the previous pixel of the same Bayer color channel c (see Section 3.6) and HVS_{weight} (Fig.3.2) can be interpreted as a *jnd* (*just noticeable difference*); hence an area is no longer flat if the D_{max} value exceeds the jnd plus the local noise level NL.

The green texture analyzer (Fig.3.4) uses a stronger rule for detecting flat areas, whereas the red/blue texture analyzer (Fig.3.5) detects more flat areas, being less sensitive to small pixel differences below the $Th_{R/B}$ threshold. The gray-scale output of the texture detection is shown in Fig.3.6: bright pixels are associated to high texture, dark pixels to flat areas.

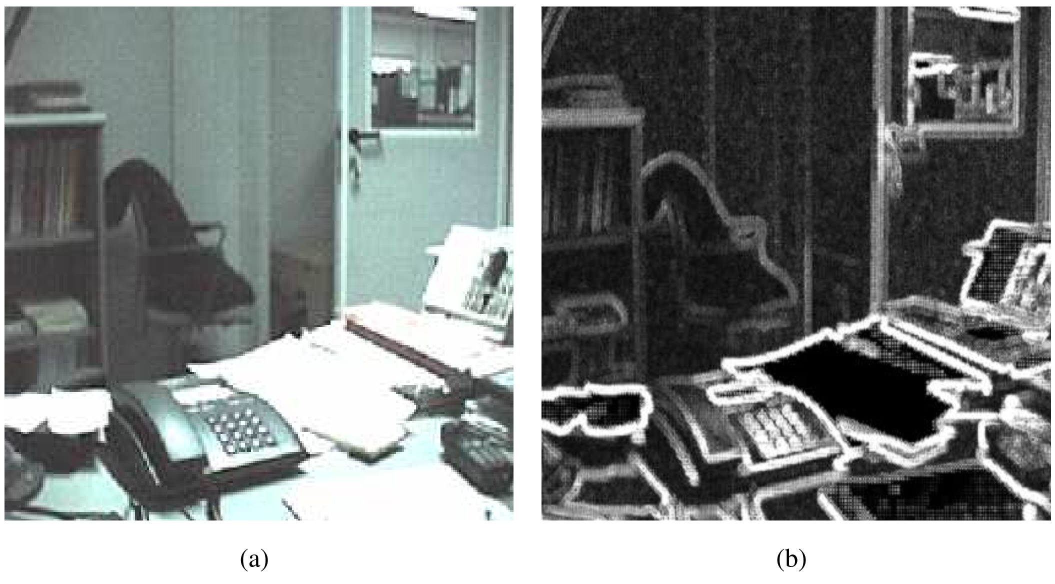


Figure 3.6 : Texture Analyzer output: (a) input image after colour interpolation, (b) gray-scale texture degree output: bright areas correspond to high frequency, dark areas correspond to low frequencies.

3.6 Noise Level Estimator

In order to adapt the filter smoothing capability to the local characteristics of the image, a noise level estimation is required. The proposed noise estimation solution is pixel based and is implemented taking into account the previous estimation to calculate the current one. The noise estimation equation is designed so that:

1. if the local area is completely flat ($T_d = 1$), then the noise level is set to D_{max};
2. if the local area is highly textured ($T_d = 0$), the noise estimation is kept equal to the previous region (i.e., pixel);
3. otherwise a new value is estimated.

Each color channel has its own noise characteristics hence noise levels are tracked separately for each color channel. The noise level for each channel is estimated according to the following formulas:

$$\begin{aligned} NL_R(k) = & \ T_d(k) * D_{Max}(k) + [1 - T_d(k)] * NL_R(k-1) \\ NL_G(k) = & \ T_d(k) * D_{Max}(k) + [1 - T_d(k)] * NL_G(k-1) \\ NL_B(k) = & \ T_d(k) * D_{Max}(k) + [1 - T_d(k)] * NL_B(k-1) \end{aligned} \tag{3.4}$$

where $T_d(k)$ represents the texture degree at the current pixel and $NL_c(k-1)$ ($c = R, G, B$) is the previous noise level estimation, evaluated considering pixel of the same colour, already processed. For $k = 1$ the values $NL_R(k-1), NL_G(k-1)$ and $NL_B(k-1)$ are set to an initial low value depending on the pixel bit-depth. These equations satisfy requirements 1), 2) and 3). The raster scanning order of the input image is constrained by global HW architecture. Starting from different spatial locations the noise level converges to the same values due to the presence of homogeneous areas that are, of course, prominent in almost all natural images.

3.7 Similarity Thresholds and Weighting Coefficients computation

The final step of the filtering process consists in determining the weighting coefficients W_i to be assigned to the neighboring pixels of the filter mask. The absolute differences D_i between the central pixel and its neighborhood must be analyzed in combination with the local information (noise level, texture degree and pixel intensities) for estimating the degree of similarity between pixel pairs (see Fig.3.7).

As stated in section 3.1, if the central pixel P_c belongs to a textured area, then only small pixel differences must be filtered. The lower degree of filtering in textured areas allows maintaining the local sharpness, removing only pixel differences that are not perceived by the HVS. The process for determining the similarity thresholds and the W_i coefficients can be expressed in terms of fuzzy logic (Fig.3.8). Let:

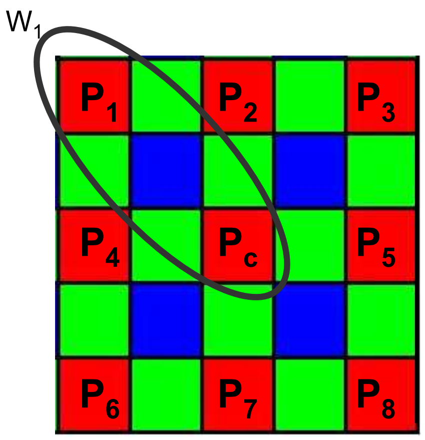


Figure 3.7 : The W_i coefficients weight the similarity degree between the central pixel and its neighborhood.

- P_c be the central pixel of the working window;
- P_i, $i = 1, \ldots, 7$ be the neighborhood pixels;
- $D_i = \|P_c - P_i\|, i = 1, \ldots, 7$ the set of absolute differences between the central pixel and its neighborhood.

In order to obtain the W_i coefficients, each absolute difference D_i must be compared against two thresholds Th_{low} and Th_{high} that determine if, in relation to the local information, the i-th difference D_i is:

1. small enough to be heavily filtered;
2. big enough to remain untouched;
3. an intermediate value to be properly filtered.

The two thresholds can be interpreted as fuzzy parameters shaping the concept of similarity between pixel pairs. In particular, the associated fuzzy membership function

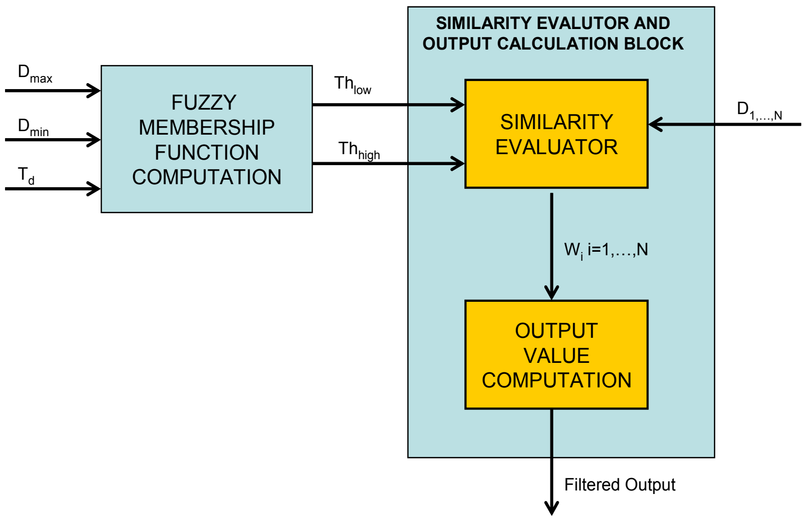


Figure 3.8 : Block diagram of the fuzzy computation process for determining the similarity weights between the central pixel and its N neighborhoods.

computes the similarity degree between the central and a neighborhood pixel. By properly computing Th_{low} and Th_{high}, the shape of the membership function is determined (Fig.3.9).

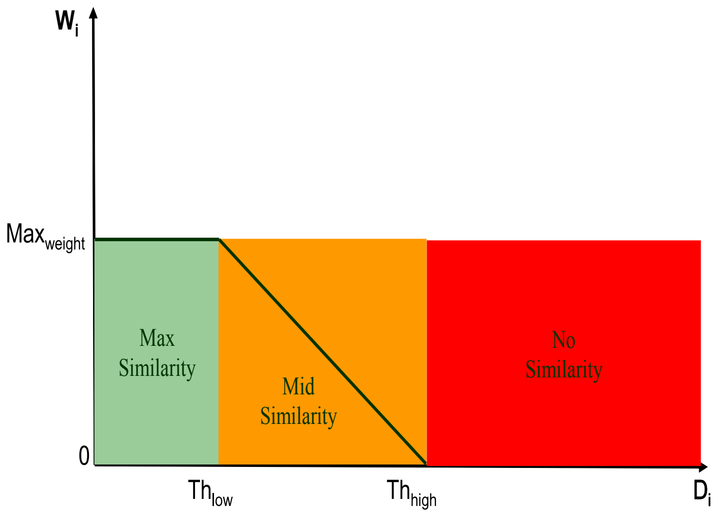


Figure 3.9 : Weights assignment (Similarity Evaluator Block). The i-th weight denotes the degree of similarity between the central pixel in the filter mask and the i-th pixel in the neighborhood.

To determine which of the above cases is valid for the current local area, the local texture degree is the key parameter to analyze. It is important to remember at this point that, by construction, the texture degree coefficient (T_d) incorporates the concepts of dark/bright and noise level; hence, its value is crucial to determine the similarity thresholds to be used for determining the W_i coefficients. In particular, the similarity thresholds are determined to obtain maximum smoothing in flat areas, minimum smoothing in highly textured areas, and intermediate filtering in areas containing medium texture; this can be obtained by using the following rules (3.5).

$$\begin{cases} Th_{low} = Th_{high} = D_{max} & \text{if } T_d = 1 \\ Th_{low} = D_{min} & \text{if } T_d = 0 \\ Th_{high} = \frac{D_{min}+D_{max}}{2} & \text{if } T_d = 0 \\ D_{min} < Th_{low} < Th_{high} & \text{if } 0 < T_d < 1 \\ \frac{D_{min}+D_{max}}{2} < Th_{high} < D_{max} & \text{if } 0 < T_d < 1 \end{cases} \tag{3.5}$$

Once the similarity thresholds have been fixed, it is possible to finally determine the filter weights by comparing the D_i differences against them (Fig.3.9).

To summarize, the weighting coefficient selection is performed as follows. If the i-th absolute difference D_i is lower than Th_{low}, it is reasonable to assume that pixels P and P_i are very similar; hence the maximum degree of similarity Max_{weight} is assigned to P_i. On the other hand, if the absolute difference between P and P_i is greater than Th_{high}, it is reasonable that this difference is due to texture details, hence P_i is assigned a null similarity weight. In the remaining cases, i.e., when the i-th absolute difference falls in the interval $[Th_{low}, Th_{high}]$, a linear interpolation between Max_{weight} and 0 is performed, allowing determining the appropriate weight for P_i.

3.8 Final Weighted Average

Let $W_1, \ldots, W_N$ (N: number of neighborhood pixels) be the set of weights computed for the each neighboring element of the central pixel P_c. The final filtered value P_f is obtained by weighted average as follows 3.6:

$$P_f = \frac{\sum_{i=1}^{N} [W_i P_i + (1 - W_i) P_c]}{N} \tag{3.6}$$

In order to preserve the original bitdepth, the similarity weights are normalized in the interval $[0, 1]$, and chosen according to equation 3.7:

$$W_i \begin{cases} 1 & \text{if } D_i \leq Th_{low} \\ L(Th_{low}, Th_{high}) & \text{if } Th_{low} < D_i < Th_{high} \\ 0 & \text{if } D_i \geq Th_{high} \end{cases} \tag{3.7}$$

Where $L(Th_{low}, Th_{high})$ performs a simple linear interpolation between Th_{low} and Th_{high} as depicted in Fig.3.9.

3.9 Experimental Results

The following sections describe the tests performed to assess the quality of the proposed algorithm. First, a test computing the noise power before and after filtering is reported. Next some comparisons between the proposed filter and other noise reduction algorithms [22, 23, 38] are described.

3.9.1 Noise Power Test

A synthetic image was used to determine the amount of noise that the algorithm is capable to remove. Let us denote:

- I_{NOISY}: Noisy CFA Pattern;
- $I_{FILTERED}$: Filtered CFA Pattern;

- $I_{ORIGINAL}$: Original noiseless CFA Pattern.

According to these definitions we have:

- I_{NOISY} - $I_{ORIGINAL} = I_{ADDED-NOISE}$;
- $I_{FILTERED}$ - $I_{ORIGINAL} = I_{RESIDUAL-NOISE}$.

Where $I_{ADDED-NOISE}$ is the image containing only the noise artificially added to $I_{ORIGINAL}$, whereas $I_{RESIDUAL-NOISE}$ is the image containing the residual noise after filtering. The noise power is computed for both $I_{ADDED-NOISE}$ and $I_{RESIDUAL-NOISE}$ according to the following formula (3.8):

$$P = 20\log_{10}\left(\frac{1}{MN}\sum_{n=0}^{N-1}\sum_{m=0}^{M} I(m,n)^2\right) \tag{3.8}$$

To modulate the power of the additive noise, different values of the standard deviation of a Gaussian distribution are used. Noise is assumed to be AWGN, with zero mean.

A synthetic test image has been generated having the following properties: it is composed by a succession of stripes having equal brightness but different noise power. Each stripe is composed of 10 lines and noise is added with increasing power starting from the top of the image and proceeding downwards (Fig.3.10). In Fig.3.11 is illustrated the filtering effects in terms of noise power; the *x*-axis represents the noise standard deviation; the *y*-axis shows the corresponding noise power decibels before and after filtering.

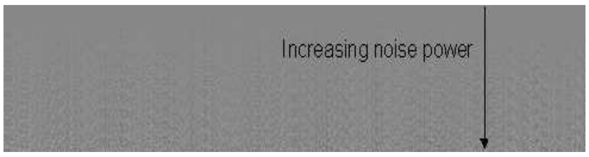


Figure 3.10 : Synthetic image test.

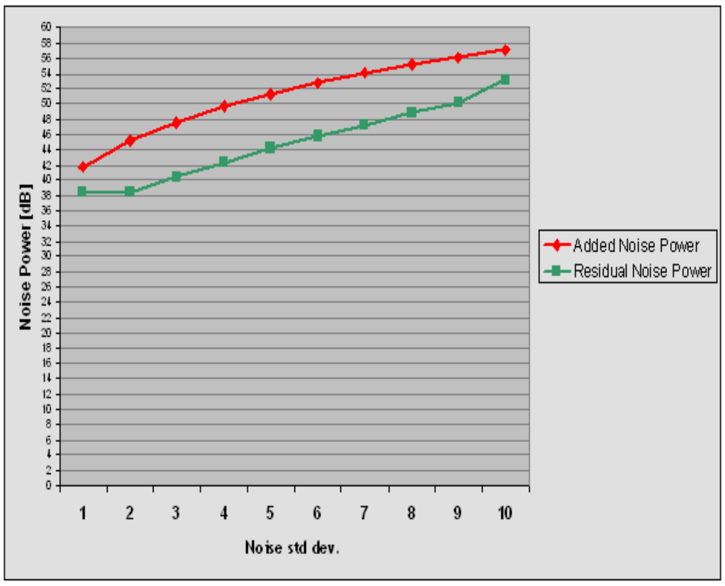


Figure 3.11 : Noise power test. Upper line: noise level before filtering. Lower line: residual noise power after filtering.

The filter significantly reduces noise and gains up to $6-7dB$ can be obtained in terms of noise power reduction.

3.9.2 Visual Quality Test

In order to assess the visual quality of the proposed method, we have compared it with the SUSAN *(Smallest Univalue Segment Assimilating Nucleus)* [38] and MMF *(Multistage Median Filters)* [22] classical noise reduction algorithm. This choice is motivated by considering the comparable complexity of these solutions. Though more complex recent methods for denoising image data exist [18, 31, 33, 40] achieving very good results, they are not yet suitable for real-time implementation.

The tests were executed using two different approaches. In the first approach, the original noisy Bayer data were interpolated obtaining a noisy color image, which was splitted in its color channels; each color plane was filtered independently using SU-

SAN. Finally, the filtered color channels were recombined to obtain the denoised color image as sketched in Fig.3.12 The second approach consists in slightly modifying the

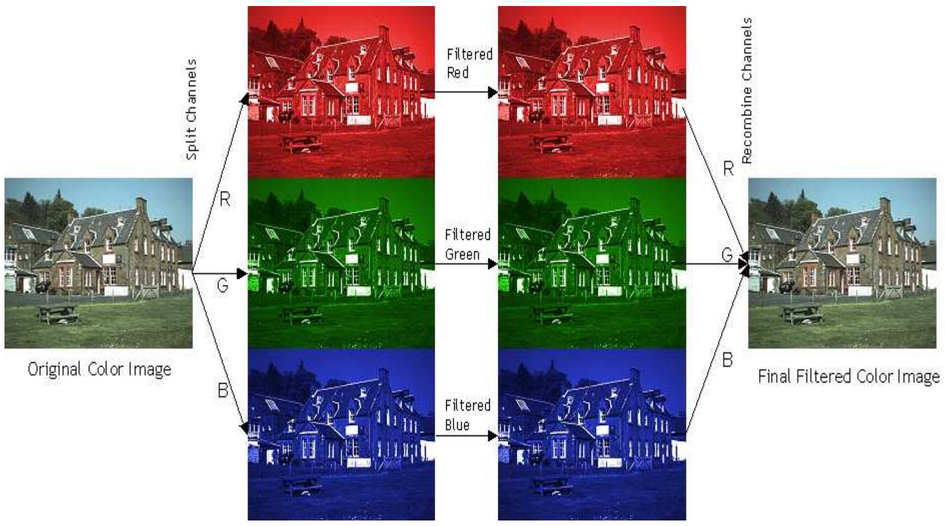


Figure 3.12 : Overall scheme used to compare the SUSAN algorithm with the proposed method. The noisy color image is filtered by processing its color channels independently. The results are recombined to reconstruct the denoised color image.

SUSAN algorithm so that it can process Bayer data. In both cases, the results of SUSAN were compared with the color-interpolated image obtained from a denoised Bayer pattern produced by the proposed method. Fig.3.13 shows two of test noisy reference images acquired by a CFA image sensor (2 megapixels) after colour interpolation. Original SNR values for the two images are 30.2dB and 47.2dB respectively. After filtering, the corresponding SNR values became comparable and higher for both, SUSAN and our filtering. In the first comparison test, both algorithms show very good performances; the proposed method, anyway, is capable to preserve some small details that are lost by SUSAN independent R/G/B filtering. Furthermore, processing is very fast because the method processes only one plane of image information, i.e., the CFA data. Fig.3.14 shows a magnified detail of Fig.3.13(a) and the filtering results with

SUSAN and our method. Fig.3.15 shows how the proposed method significantly retains texture and sharpness after filtering. Fig.3.16 shows two different details of the noisy image in Fig.3.13(b) and their filtered counterparts. The homogeneous areas are heavily filtered 3.16(a) , 3.16(b); on the other hand, in textured areas, the detail is well preserved 3.16(c), 3.16(d).

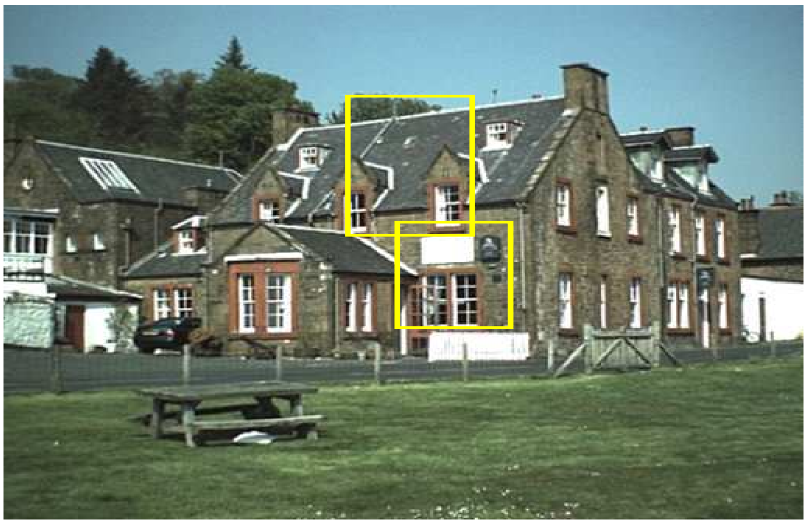
Finally, Fig.3.17 and 3.18 illustrate the results of the multistage median filters described in [22] compared with the proposed filter. Specifically, the multistage median-1 and multistage median-3 filter outputs were considered. The three methods work on CFA data. Fig.3.18(d) shows, again, that the proposed filtering technique is able to preserve texture and sharpness very well.

3.9.3 PSNR Test

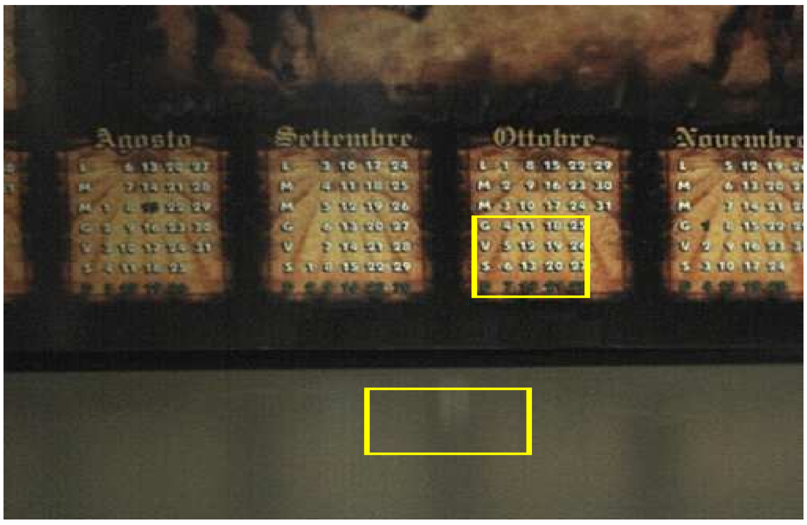
In order to numerically quantify the performance of the filtering process, the standard Kodak 24 (8-bpp) [39] images have been processed with the proposed method comparing them with the outputs of SUSAN [38], Multistage Median-1, Multistage Median-3 algorithms [22] and the following fuzzy approaches from [23]:

1. $GMED$: Gaussian Fuzzy Filter with Median Center;
2. $GMED$: Gaussian Fuzzy Filter with Median Center;
3. $GMAV$: Gaussian Fuzzy Filter with Moving Average Center;
4. $ATMED$: Asymmetrical Triangular Fuzzy Filter with Median Center;
5. $ATMAV$: Asymmetrical Triangular Fuzzy Filter with Moving Average. Center

After converting each image of the set to Bayer pattern format, the simulation was performed by adding noise with increasing standard deviation to each CFA plane. In

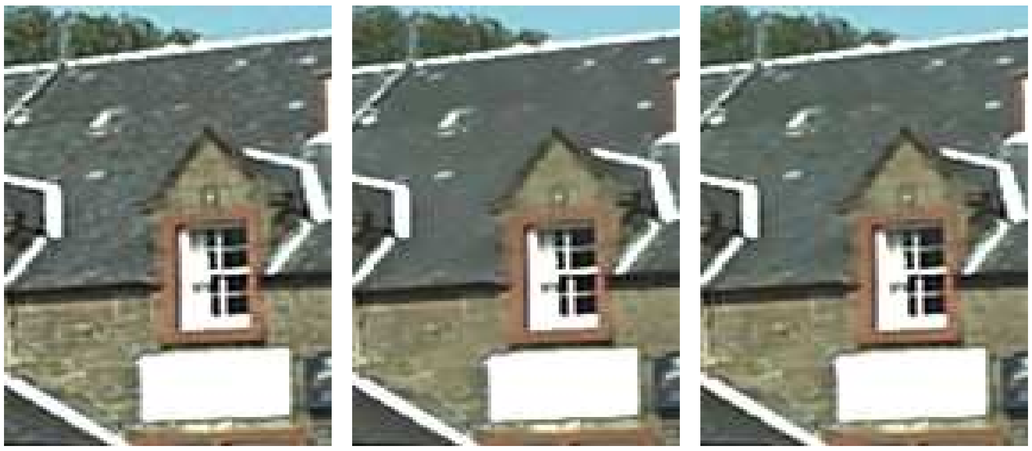


(a)



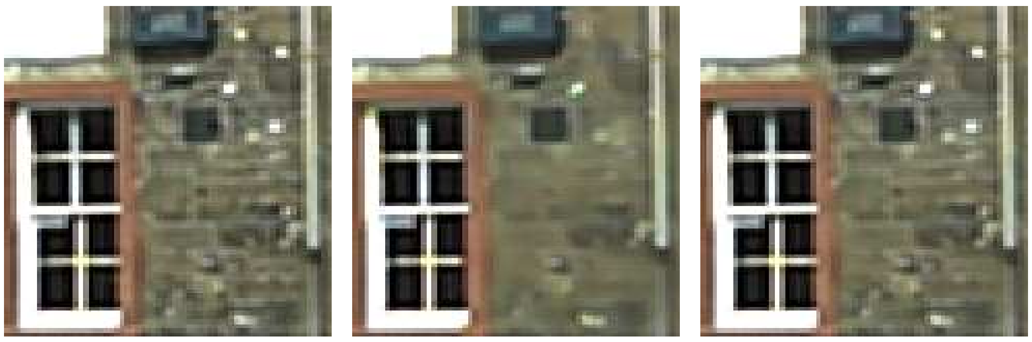
(b)

Figure 3.13 : Images acquired by a CFA sensor. (a) SNR value 30.2dB. (b) SNR value 47.2dB. The yellow crops represent the magnified details contained in the following figures.



(a) Noisy (SNR 30.2dB) (b) SUSAN applied to R/G/B separately (SNR 30.5dB) (c) Proposed Method (SNR 31.2dB)

Figure 3.14 : A magnified detail of Fig.(3.13), to better evaluate the comparison between the proposed filter and the SUSAN algorithm applied on R/G/B channels separately. Both methods preserve details very well, although the proposed technique is capable to better preserve texture sharpness; the enhancement is visible by looking at the wall and the roof texture. The proposed method uses fewer resources as the whole filtering action takes place on one plane of CFA data.

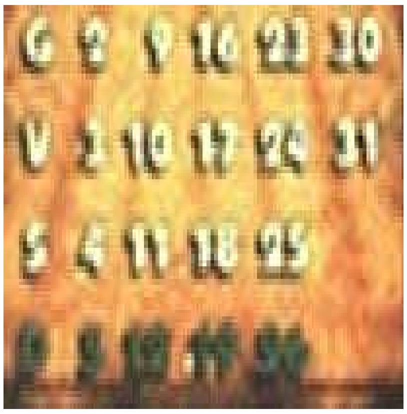


(a) Noisy (SNR 30.2dB) (b) SUSAN adapted to CFA (SNR 30.5dB) (c) Proposed Method (SNR 31.2dB)

Figure 3.15 : Comparison test at CFA level (magnified details of Fig.3.13(a)). The original SUSAN implementation was slightly modified so that it can process Bayer data. The efficiency of the proposed method in retaining image sharpness and texture is clearly visible.

(a) Cropped part of noisy image 3.13(b), 200% zoomed (pixel resize)

(b) Filtered 200% zoomed (pixel resize) counterpart


(c) Cropped part of noisy image 3.13(b), 200% zoomed (pixel resize)

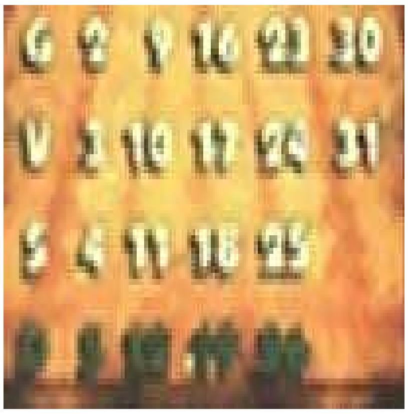
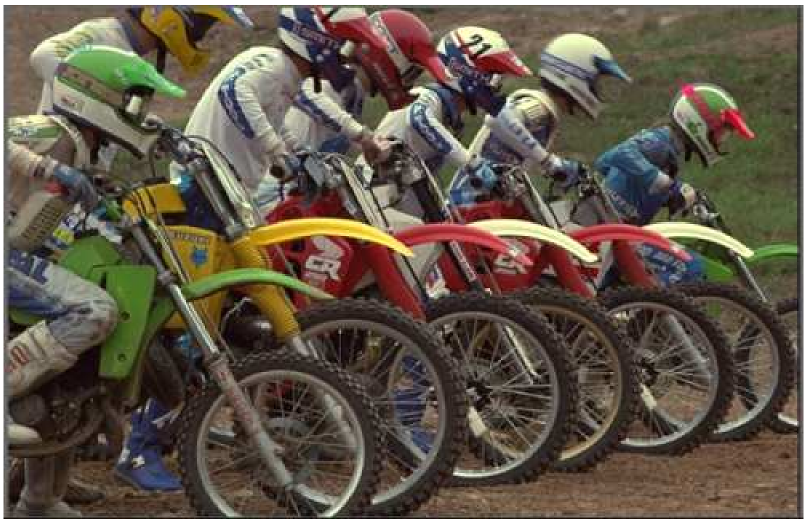

(d) Filtered 200% zoomed (pixel resize) counterpart

Figure 3.16 : Magnified details (noisy and filtered) of Fig.3.13(b). The effects of the proposed method over flat (a), (b) and textured (c), (d) areas are shown. The noisy images are obtained by color interpolating unfiltered Bayer data (a), (c). The corresponding color images produced by demosaicing filtered Bayer data (b), (d). SNR values are: 47.2dB for noisy image and 51.8dB for filtered image.


(a) Original

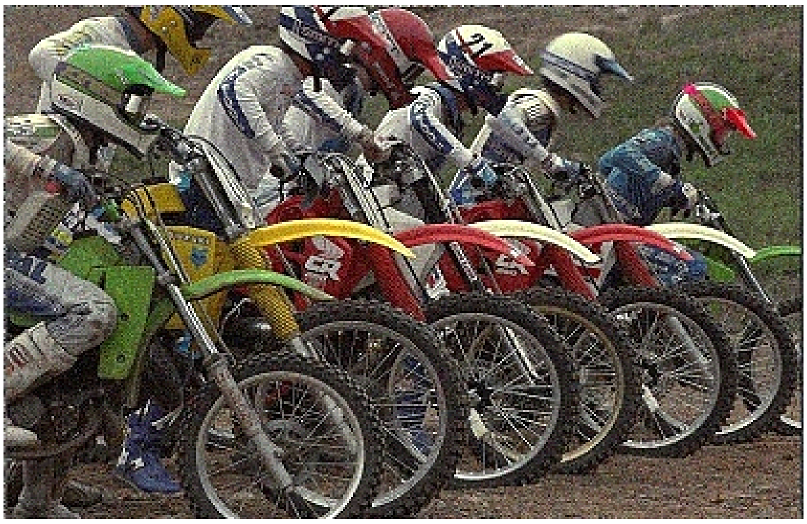
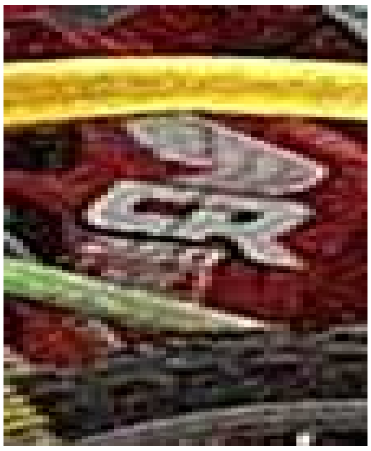
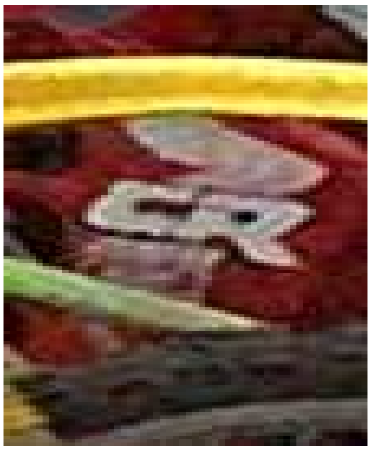
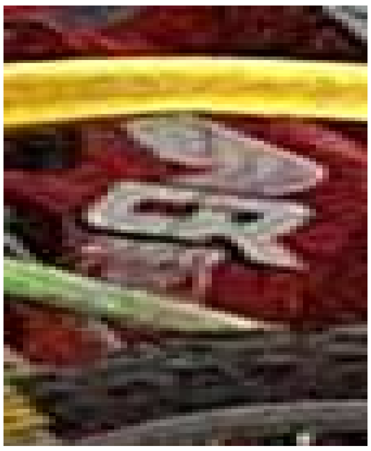

(b) Noisy (SNR 26.1 dB)

Figure 3.17 : Kodak image (*kodim05*). Original and noisy version.


(a) Cropped and zoomed noisy image (SNR 26.1 dB)


(b) Multistage median-1 filter. (SNR 26.5 dB)


(c) Multistage median-3 filter. (SNR 26.8 dB)

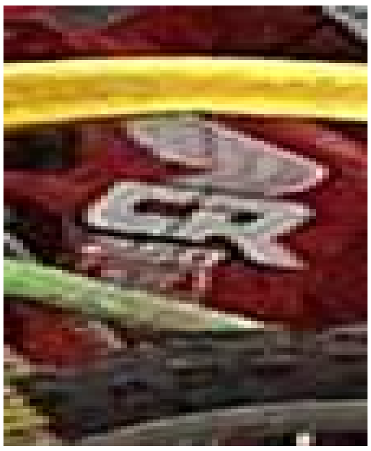

(d) Proposed method. (SNR 27.2 dB)

Figure 3.18 : (a) Cropped and zoomed detail of noisy image in Fig.3.17(b), filtered with: Multistage median-1 filter (b), Multistage median-3 filter (c) and the proposed method (d).

particular the following values have been used: $\sigma = 5, 8, 10$. More specifically, the aforementioned values of σ refer to the noise level in the middle of the dynamic range. To simulate a more realistic sensor noise, in fact, we followed the model described in [13, 14], that allows obtaining lower noise values for dark areas and higher noise values for bright areas, according to a square root characterization of the noise. In order to exclude the effects of different color interpolations from the computation of the PSNR, the reference images were obtained following the procedure described in Fig.3.19; in this way, both images (i.e., clean and noisy) are generated using the same color interpolation algorithm.

Experiments show that the proposed method performs better in terms of PSNR compared to different spatial algorithms [21, 38] and fuzzy approaches [23]. The results are shown in Figg.3.20 and 3.21

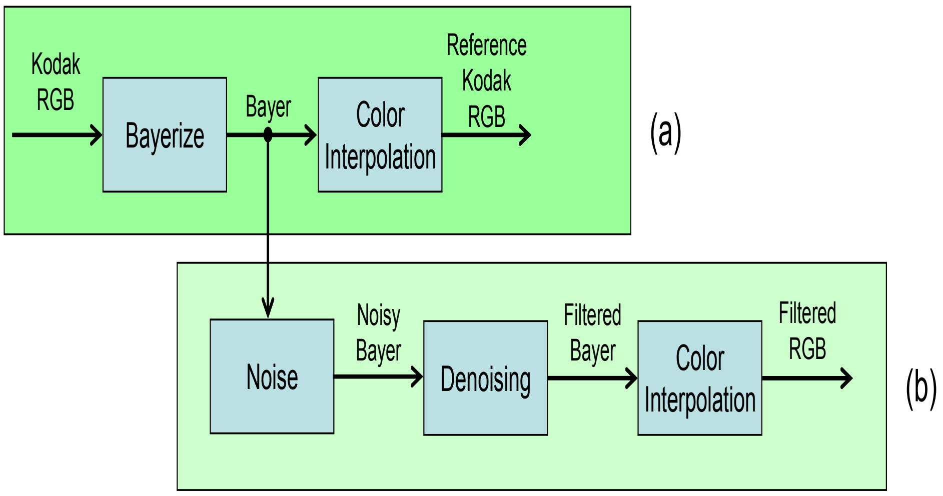
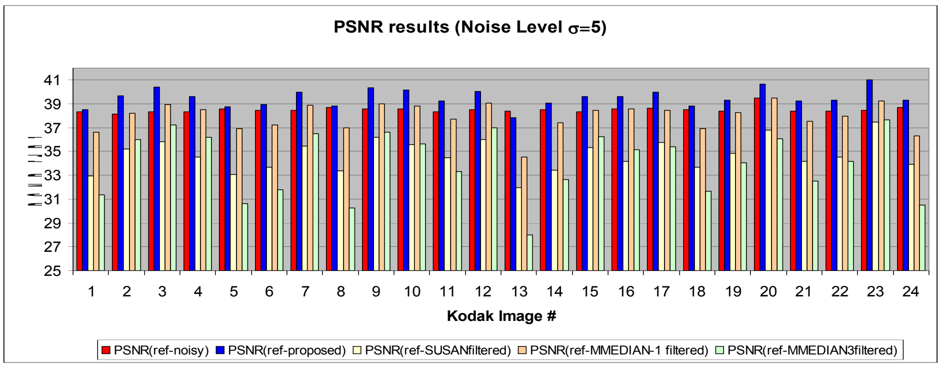
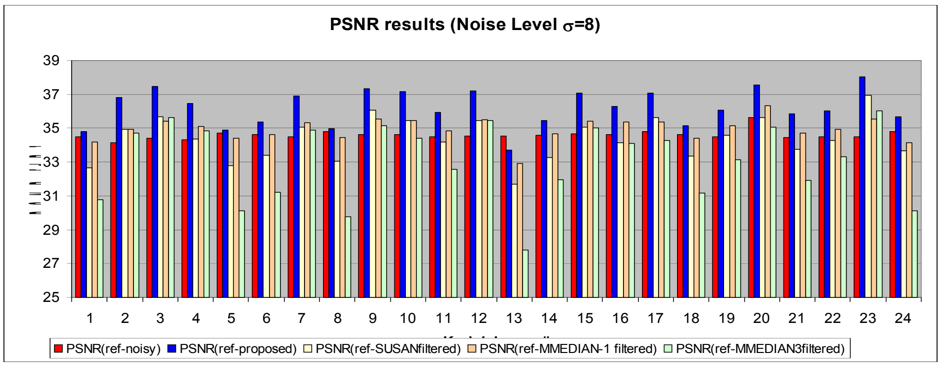


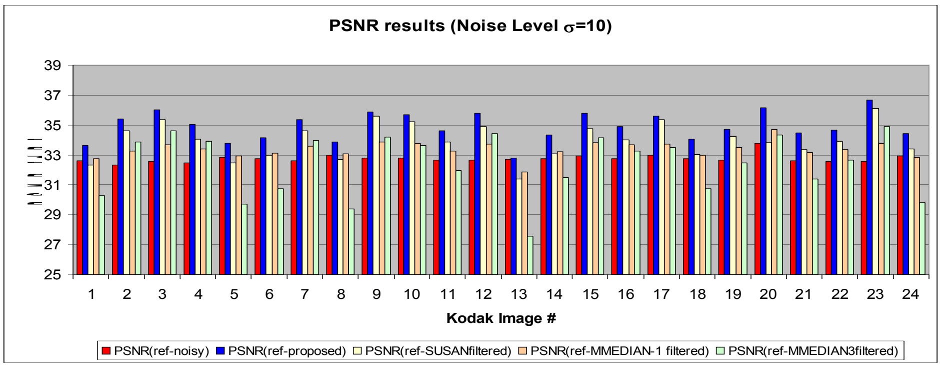
Figure 3.19 : Testing procedure. (a) The original Kodak color image is converted to Bayer pattern format and demosaiced. (b) Noise is added to the Bayer image, filtered and color interpolated again. Hence, color interpolation is the same for the clean reference and the denoised images.



(a) Kodak noisy images set with standard deviation 5

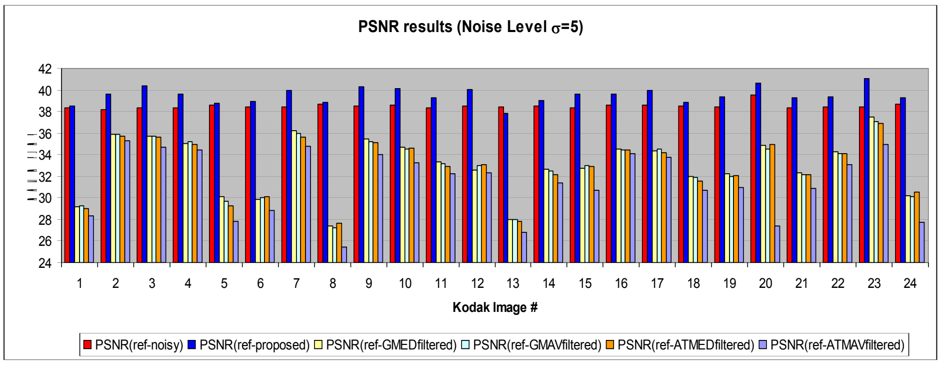


(b) Kodak noisy images set with standard deviation 8

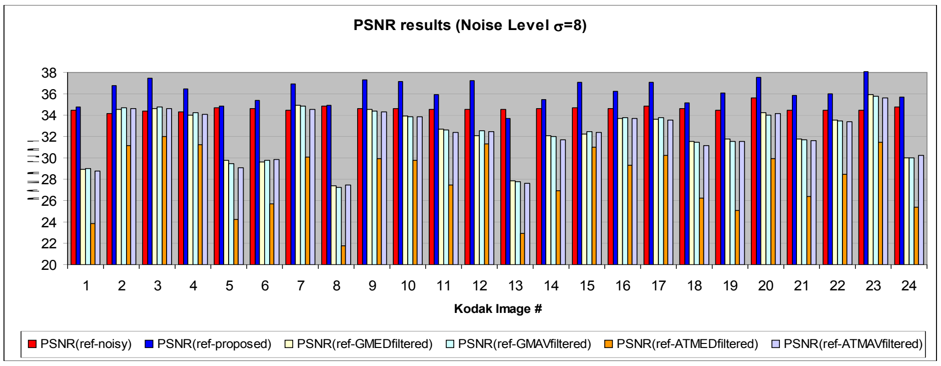


(c) Kodak noisy images set with standard deviation 10

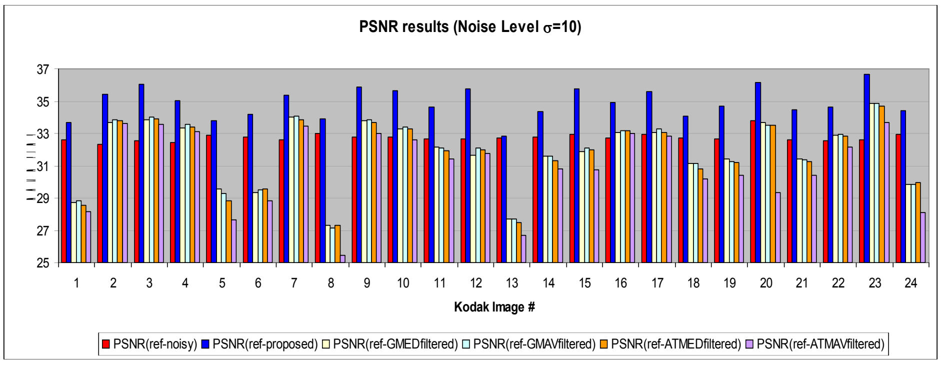
Figure 3.20 : PSNR comparison between proposed solution and other spatial approaches for the Standard Kodak Images test set.



(a) Kodak noisy images set with standard deviation 5



(b) Kodak noisy images set with standard deviation 8



(c) Kodak noisy images set with standard deviation 10

Figure 3.21 : PSNR comparison between proposed solution and other fuzzy approaches for the Standard Kodak Images test set.

Chapter 4. Signal Dependent Noise Estimation

4.1 Introduction

In the Chapter 2 the importance of noise level estimation phase to obtain an effective noise filtering, is described in detail. Anyway, as introduced in Section 1.10, an accurate noise model must take into account the noise dependency from the signal intensity. Usually the standard deviation of the *Signal Dependent Noise (SDN)* is defined using equation (1.20), as described in Section 1.10 [14]. However, through a series of analyses that will be introduced in the next section, we will show that sensor noise level at the various acquisition settings can be modeled by using a more precise fitting equation. Therefore, to model the SDN noise, in this chapter and in the next one, the following model is adopted:

$$\sigma(i) = \sqrt{a \cdot i^2 + b \cdot i + c} \tag{4.1}$$

where: $a, b, c \in \Re^+$ and i is signal intensity. Hence, the problem of characterizing the image sensor noise can be formulated as the problem of finding the proper coefficients a, b and c for any operating condition of the imager (e.g., low light, normal, etc.). To this end, we observe that the noise characteristics of an image sensor are strictly related to the amount of light in the scene being captured. More specifically, as illumination decreases, amplification of the signal is mandatory in order to obtain an acceptable image. Signal amplification at sensor level is obtained by means of *analog gains* that boost the detected intensities [14]. Unfortunately, one cannot amplify image signal without boosting noise levels as well: the higher the analog gain applied at sensor level, the higher the amplification of signal and noise.

An accurate sensor noise characterization requires measurements in a controlled environment. The acquisition device must be analyzed at all its operating conditions; this is usually realized by acquisitions of homogeneous images obtained by means of an integrating sphere, which provides a uniform light field of variable intensity. The sensor is then used to acquire the flat images at varying intensities and using various amplification factors. For each homogeneous image its average value is computed and is chosen as representative of the intensity i; the associated standard deviation is also calculated. We call *noise sample* the pair $(i, \sigma(i))$, where i is the image average value and $\sigma(i)$ is the standard deviation associated to i.

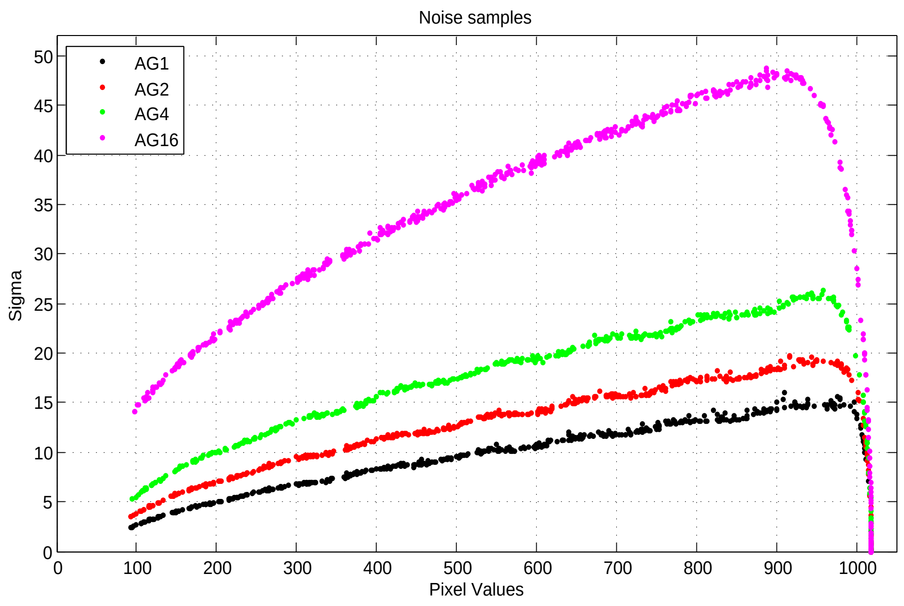


Figure 4.1 : Acquired noise samples at different analog gain (AG = 1, 2, 4, 16) for a 3Mp ST Sensor.

As Fig.4.1 shows, once the analog gain factor is set, the noise samples lie in a specific locus that can be identified by a specific equation after a proper fitting process is performed.

This Chapter is structured as follows: in Section 4.2 the equations (1.20), so far used to describe the signal-dependent noise [14], and the new model (4.1) are analyzed, and compared with others fitting equations, in order to show the performance of each model and detect the more accurate equation which provides the best fitting of the noise samples at each illumination condition. An effective way to obtain noise curves for an image sensor using heterogeneous images acquired without a controlled environment [1] is introduced in Section 4.3. This noise estimation framework operate by fitting and interpolating the noise samples using multiple acquisitions to achieve a more robust estimation. Finally is introduced a study-case that applying the signal-dependent noise estimation procedure to multispectral images, obtain interesting results.

4.2 SDN Model Analysis

The behavior of the signal dependent noise is typically described using Eq.(1.20) as in [14]. However, starting from a set of noise samples acquired in laboratory under controlled conditions, we discovered that an improved fitting equation may provide better approximation of the noise samples in terms of RMSE under all sensor operating conditions. In particular, experiments show that in some cases the fitting curve (1.20) deviates significantly from the noise samples that we obtained in a controlled environment, especially for low signal values (Fig.4.2). Notice that the noise samples associated to signal intensities near saturation (clipped values) are excluded from the fitting process.

In order to find an improved fitting equation, we analyzed Eq.(1.20), together with

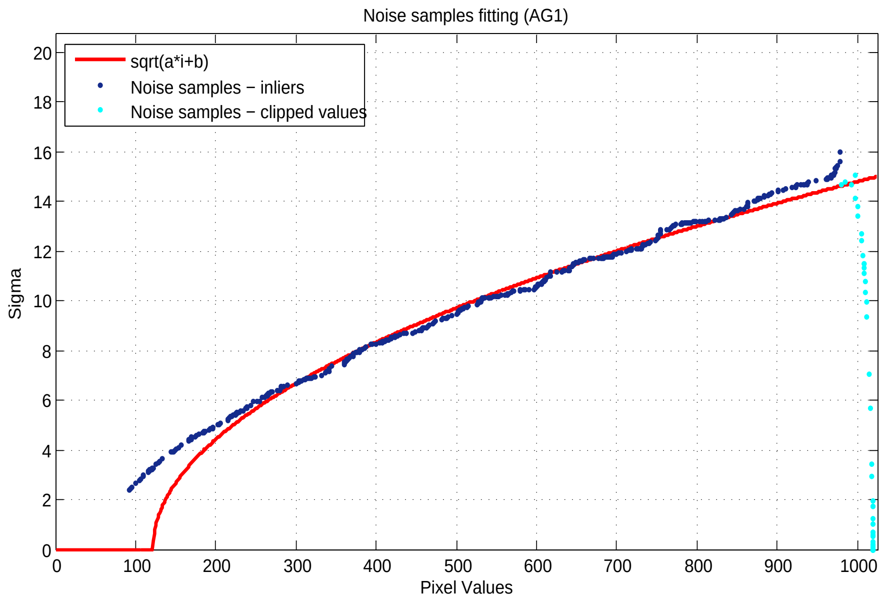


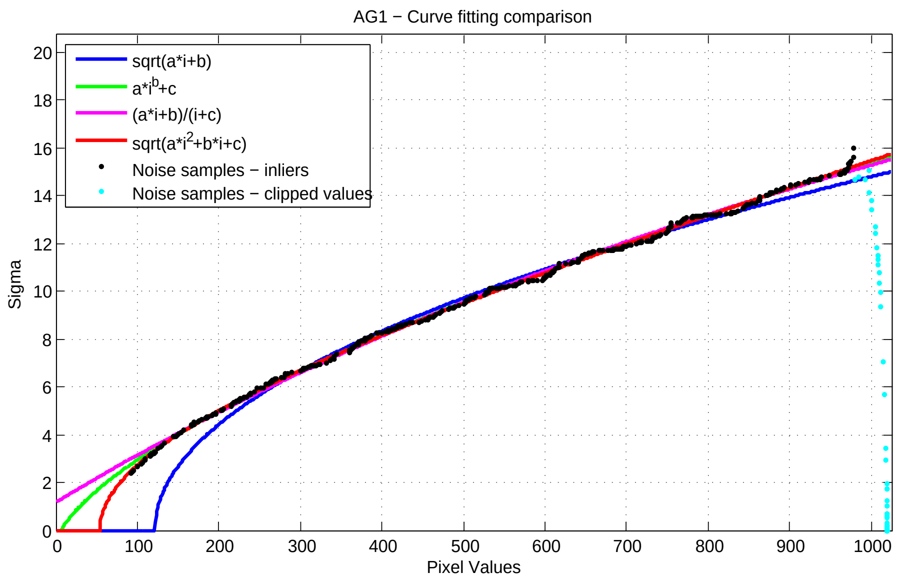
Figure 4.2 : Fitting of noise samples acquired at low analog gain (AG1), using Eq.(1.20).

the Eq.(4.1) and the following alternative functions:

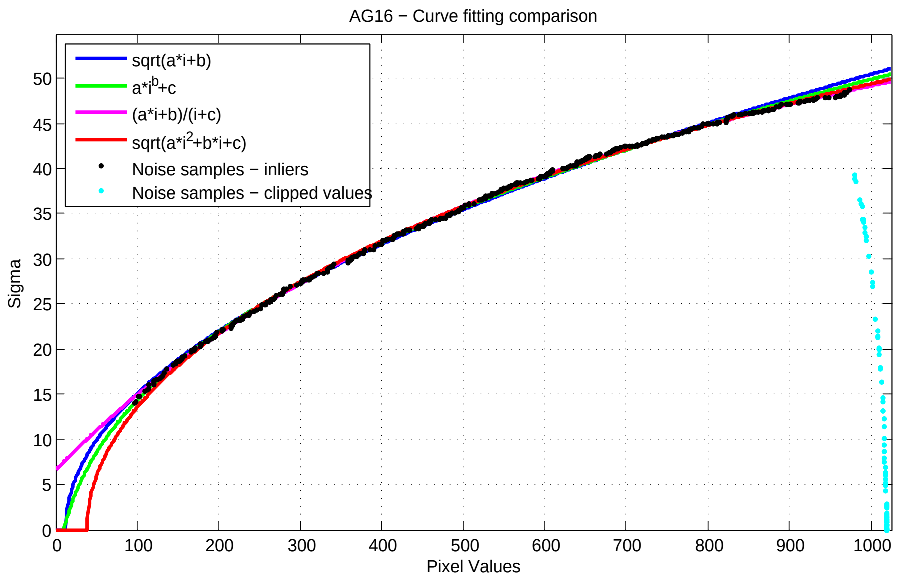
$$\sigma(i) = a \cdot i^b + c \tag{4.2}$$

$$\sigma(i) = \frac{a \cdot i + b}{i + c} \tag{4.3}$$

Table 4.1 and Fig.4.3 show the fitting results and associated root mean squared error (*RMSE*) for the analyzed models at different AG. As Table 4.1 proves, the lowest error is associated to the fitting model (4.1); this equation, in fact, allows a more accurate fitting of the noise samples for each considered analog gain especially in the lower part of the dynamic range. For this reason, the Eq.(4.1) is used as reference for all SDN estimation and SDN removal algorithms, which will be covered in these chapters.



(a) AG1



(b) AG16

Figure 4.3 : Fitting noise samples at AG1 (a) and AG16 (b).

Table 4.1 : RMSE comparisons among equations (1.20), (4.1), (4.2) and (4.3) at different analog gains.

	Eq. (1.20)	Eq. (4.1)	Eq. (4.2)	Eq. (4.3)
AG1	0.4512	**0.1346**	0.1548	0.1801
AG2	0.3291	**0.1634**	0.1775	0.2210
AG4	0.2068	**0.1785**	0.1853	0.2419
AG16	0.1556	0.1553	**0.1517**	0.2184
average	0.2857	**0.1579**	0.1673	0.2154

4.3 SDN Estimation by Multiple Images

This section presents the signal dependent noise estimation framework, through which the noise profile of an image sensor under any illumination condition can be obtained. The initial phase of this procedure consists in the collection of noise samples from images acquired at every analog gain. To obtain a reliable estimate, it is necessary to collect noise samples from a certain amount of images acquired at each analog gain; these images must be partitioned into sets, classified according to the corresponding analog gain, and processed separately with the same methodology.

The proposed solution [1] is based on the analysis of raw CFA data generated before the application of any data processing, using the first image outputted from the imager. The following Sections describe in detail: the methodology for the extraction of the noise samples (Sections 4.3.1 and 4.3.2), and the technique to obtain the sigma-curves from the fitted samples (Sections 4.3.3 and 4.3.4). Finally, Section 4.3.5 describes an alternative voting-based fitting approach.

4.3.1 Homogeneous Area Detection

Noise can be estimated in image areas where the signal activity is very low: tipically, texture detection algorithms are applied to separate homogeneous areas from textured ones. Many methods for texture detection exist: the Amer-Dubois method [3], for example, uses a set of highpass filters that detect signal activity in the horizontal, vertical, diagonal and corner directions; the Kim-Lee technique [44] is based on a more sophisticated approach which tries to differentiate image areas with the same standard deviations but generated by patterns not related to random signal fluctuations; finally, Staelin-Nachlieli [11] propose to estimate noise in areas where cross channel color correlation is low.

The proposed methodology can be used with any texture detector algorithm; we adopted the Kim-Lee texture detector [44] because of its reliability and reduced outlier amount.

4.3.2 Noise Statistics Accumulation

An highpass filter is used to reject image areas containing structure; as stated above, we use the texture detector *(TD)* described in [44]. The basic idea is to accumulate a significant amount of noise level samples for any available intensity; the noise samples sets are then analyzed to characterize the noise sensor behaviour. Let:

- W_i : $m \times m$ i-th processing window over the input image I;
- $Th_{texture}$: texture threshold depending on the used texture detector and desired sensitivity;
- N: number of $m \times m$ blocks in the image I;

The set W^{flat}, composed by the flat areas of the image I, is obtained as follows:

$$W^{flat} = \{W_i \mid TD(W_i) < Th_{texture}, i = 1, \dots, N\} \tag{4.4}$$

For all the elements in W^{flat} the standard deviation (σ) and the average value (μ) are computed:

$$(\mu(w), \sigma(w)), \quad \forall w \in W^{flat} \tag{4.5}$$

A list of σ values is collected for each image intensity $l \in [0, \ldots, 2^{bpp} - 1]$ such that at least a flat mask exists:

$$\Theta(l) = \bigcup_{w \in W^{flat}} \{\sigma(w) \mid \mu(w) = l\} \tag{4.6}$$

$\forall l \in [0, \ldots, 2^{bpp} - 1]$; granularity in each set $\Theta(l)$ is reduced approximating to the nearest integer or, for better accuracy, cutting precision to one digit after the decimal point.

4.3.3 Noise Statistics Fitting for a Single Image

All the noise samples gathered in the previous step are analyzed to retrieve a noise level for each signal intensity. To obtain a robust estimate for the noise level at a certain intensity l we consider the most frequently computed standard deviation in $\Theta(l)$. Furthermore, to reject outliers, a lower bound Ω on the number of occurrences of the most recurrent element in $\Theta(l)$ is considered. Then, let τ represent the frequency of the most recurrent value in $\Theta(l)$:

$$\tau = freq(mode(\Theta(l))) \tag{4.7}$$

The noise level Φ for intensity l is obtained as:

$$\Phi(l) = \begin{cases} mode(\Theta(l)) & if \;\; \tau \geq \Omega \\ 0 & if \;\; \tau < \Omega \end{cases} \tag{4.8}$$

It should be noticed that:

1. the image may contain portions of the dynamic range for which noise statistics cannot be retrieved. For example, the image may not contain suitable flat areas for a given signal level, even if that intensity exists in the image (Fig.4.4).

2. $\Theta(l)$ may contain outliers and null values that must be removed.

In order to solve the above two issues, a robust linear least squares (*LLS*) fitting is applied to the points where $\Phi(l)$ is defined, i.e., $\Phi(l) > 0$, finally obtaining the interpolating curve $C(l)$:

$$C(l) = LLS(\Phi(l)|\Phi(l) > 0), \quad l \in [0,\ldots,2^{bpp}-1] \tag{4.9}$$

using the square root approximating function (4.1) as fitting curve model.

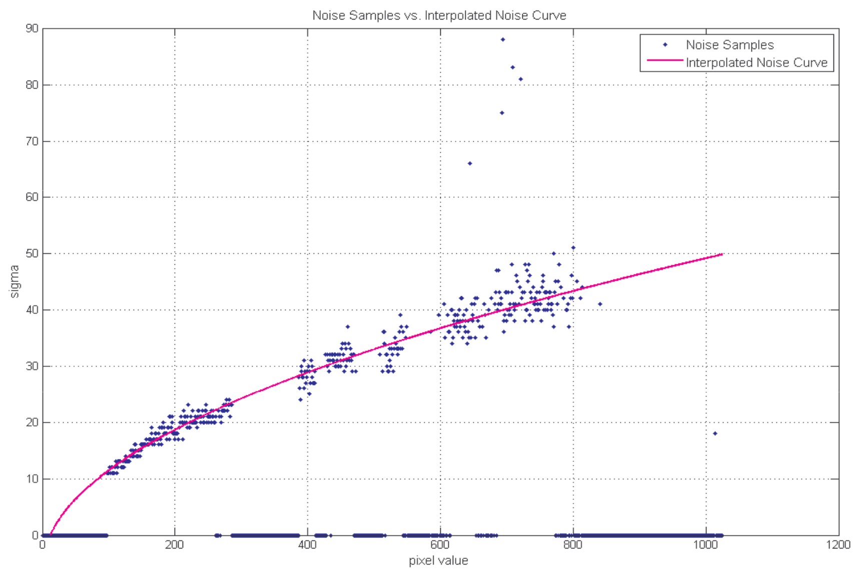


Figure 4.4 : Images can generate noise plots containing missing data for some signal levels. The missing points are obtained via interpolation.

4.3.4 Global Analysis of the Fitted Data

The fitting process is repeated for every image of all the analog gains sets. Given a set of images at a specific analog gain Γ, each image $\gamma \in \Gamma$ is processed finally generating a set of noise curves $C_{\gamma}^{\Gamma}(l)$ according to (4.9). Starting from the set of available noise curves $C_{\gamma}^{\Gamma}(l)$, we generate the unique noise characterization curve for the analog gain of the images in Γ. A non linear smoothing operator is applied along the y-axis to reject data outliers; a median operator represents a robust outlier rejecter for this application:

$$C_{est}^{\Gamma}(l) = F\left(median\left(\left\{C_{\gamma}^{\Gamma}(l)\right\}_{\gamma=1,\ldots,|\Gamma|}\right)\right) \tag{4.10}$$

where $F(\cdot)$ is a low pass smoothing operator applied to reduce small outliers. The final a and b coefficients for the images at analog gain Γ are obtained by *LLS* interpolation of the points in (4.10). As the number of images in Γ increases, the estimation errors decrease, especially at high analog gains.

4.3.5 Voting-based Sensor Noise Characterization

Now we introduce a voting-based fitting technique, that can be used to make more robust the fitting process discussed in previous sections [1]. In fact, due to the complexity of real scenes, the texture detector can misclassify textured areas as flat, yielding to the generation outlier pairs $(i, \sigma(i))$. Simple curve fitting methods (e.g., Least Squares) could be not able to cope well with outliers, hence an ad hoc robust fitting techniques have to be used, and a voting-based approach may be a reliable choice [35]. Equation (4.1), by means of some simple mathematical manipulations, can be rewritten as:

$$\sigma^2(i) = a \cdot i^2 + b \cdot i + c \tag{4.11}$$

Equation (4.11) represents a plane in the three-dimensional parameter space (a,b,c). For each analog gain Γ, all sets $\Theta(l)$ obtained during the noise statistics accumulation (Section 4.3.2), can be used directly for the voting process. Each pair $(i,\sigma(i)) \in \Theta(l)$ votes for a specific plane in the parameter space.

The fitting parameter values $(\overline{a},\overline{b},\overline{c})$ can be then obtained considering the densest region of the space (Fig.4.5) making use of a 3D histogram (generalized Hough Transform [37]). In order to obtain higher accuracy (limited by the binning process), these estimated parameters are only used to filter out noise sample outliers; this filtering is performed in the 3D histogram by considering the pairs voting for the bins close to the best one. Finally the selected inliers are used as input in the Least Squares estimation (Fig.4.6).

4.4 SDN Sensor Characterization Framework Tests

The last proposed noise profile estimator has been validated using sets of 20 images per each analog gain, which are acquired using a *ST 3MP* sensor generating 10bit raw images, and processed adopting the Kim-Lee texture detector [44]. The tests analyze the validity of the proposed framework, comparing, in particular, the results achieved by the voting-based approach [35], and those obtained by the standard technique based on Least Squares fitting method [1].

For the voting method, all tests have been performed using the same parameters, a 3D histogram (a, b, c) of $291 \times 12001 \times 11$ bins with a from 0 to 0.0001, b from 0.01 to 3, c from -60 to 60.

Table 4.2shows the performance of the two fitting methods in terms of RMSE, i.e., the differences between estimated function and the sigma samples acquired in laboratory at different AG. Then, we can see that with both technique the error in the

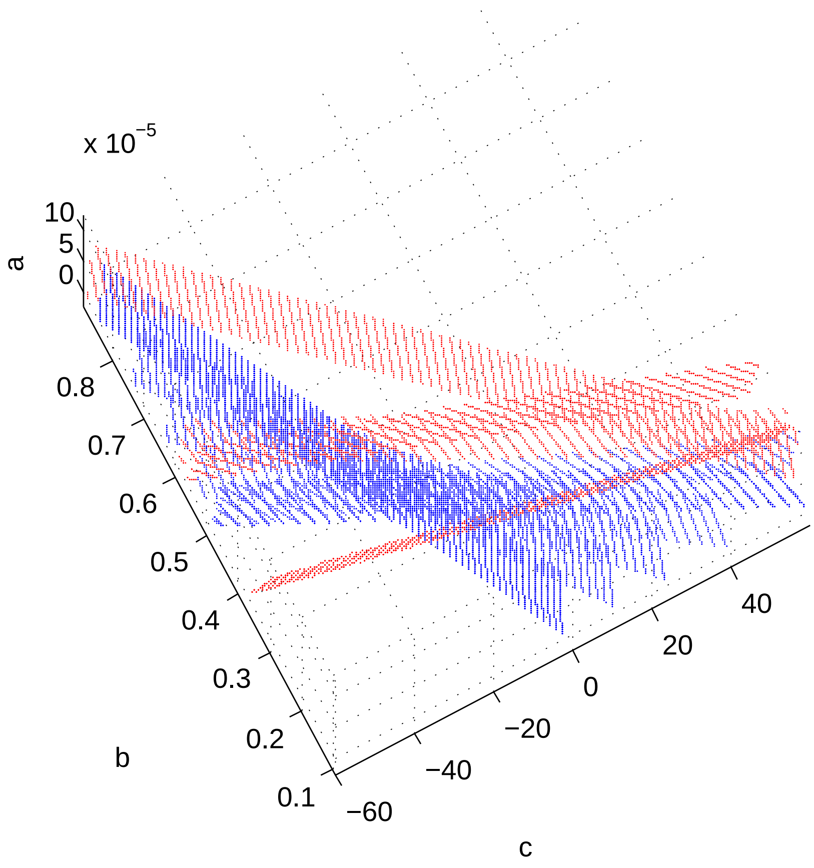


Figure 4.5 : Each $(i, \sigma(i))$ pair votes for a plane in the parameter space. Planes corresponding to inliers (blue), intersecting in a single point, produce the desiderate triplet. The outlier planes (red) do not disturbs this process.

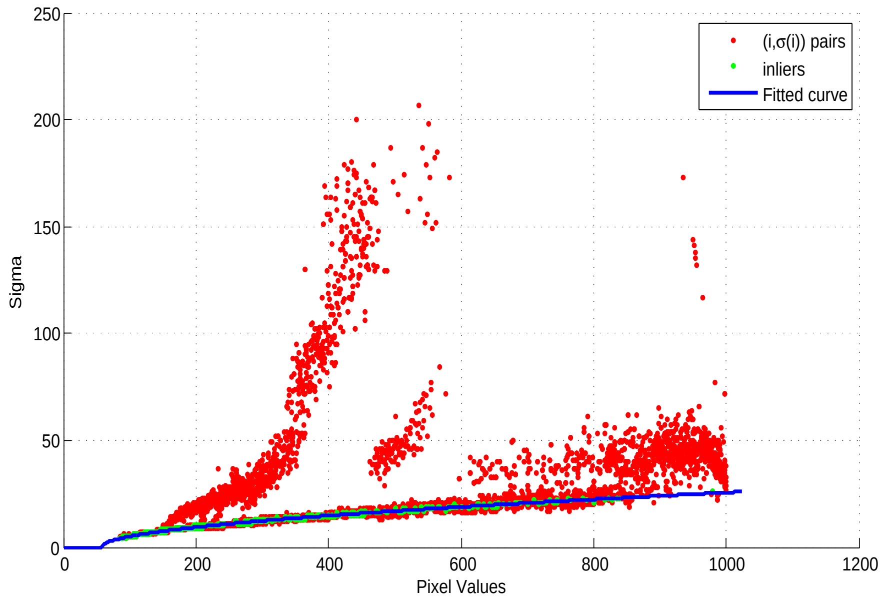


Figure 4.6 : Voting approach selects inliers (green) from the $(i, \sigma(i))$ pair set (red). This inliers are then used to estimate (through Least Squares) the model parameters. The fitting curve is represented in blue.

estimate is low enough, and, in general, the voting approach seems to be the most accurate.

Table 4.2 : RMSE comparison between voting approach and standard approach [1].

	voting approach	[1]
AG 1	0.2470	0.4801
AG 2	0.3648	0.6463
AG 4	0.6981	0.5296
AG 16	2.7514	3.0885
average	**1.0153**	**1.1861**

To visually assess the performances of the proposed system both voting approach [35] and [1] are shown in Fig.4.7 together with the $(i, \sigma(i))$ pairs measured by the integrating sphere. Estimation error increases as the analog gain augments, but it is possible to reduce the errors by increasing the number of processed images.

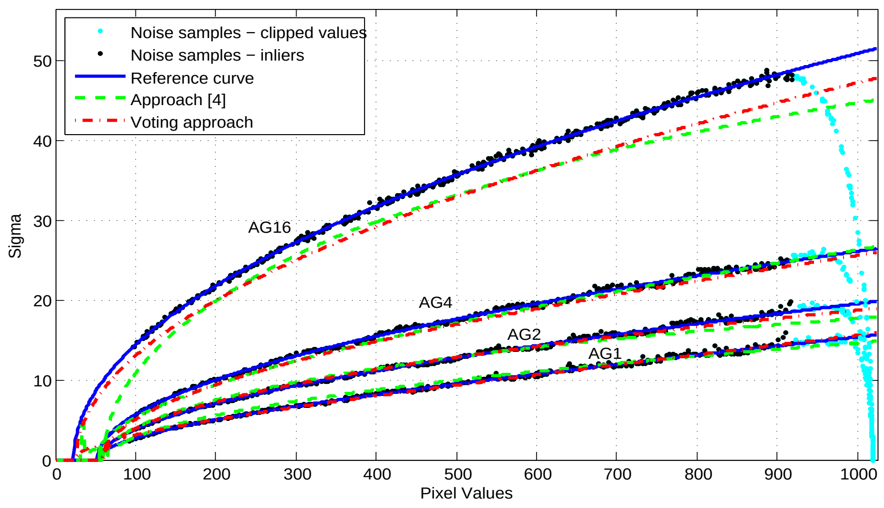


Figure 4.7 : Visual comparison between the voting approach (red) and the method proposed in [1]. The reference curve (blue) is obtained by fitting the noise samples obtained in lab, using Eq.4.1.

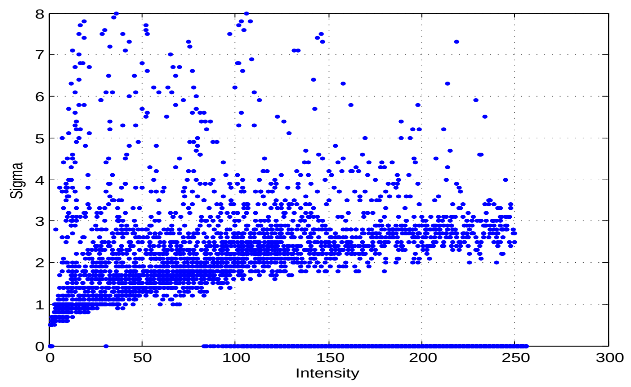
4.5 Application: Signal Dependent Noise Estimation for Multispectral Images

The proposed voting-based SDN profile estimator [35] has been also validated using a database of high spatial and spectral resolution reflectance images captured under calibrated viewing conditions [20]. These images have been acquired using Applied Spectral Imaging Spectracube camera[1].

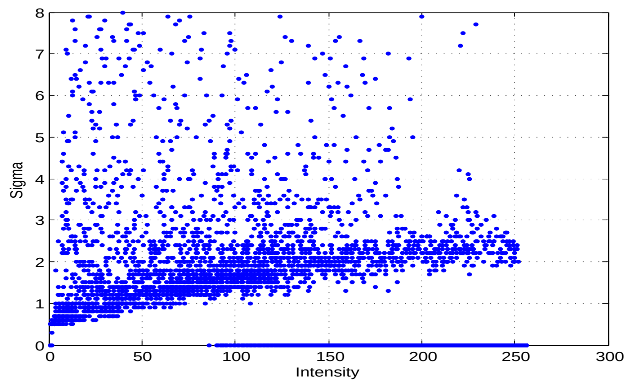
In order to profiling multispectral image perturbation, data at every band have been considered. The images are partitioned into sets and classified according to the corresponding band, obtaining noise samples as show in Fig.4.8:

Multispectral noise samples are then fitted using a voting-based appoach, where, the square of the intensity standard deviation is modeled as a 2th order polynomial function of the signal intensity (4.1).

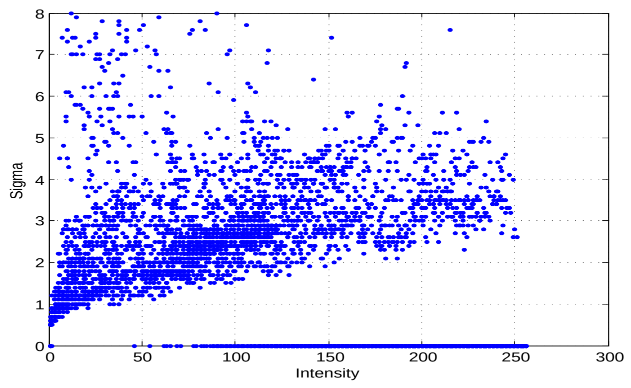
[1]http://www.spectral-imaging.com/products/spectral-imaging



(a) 500nm



(b) 550nm

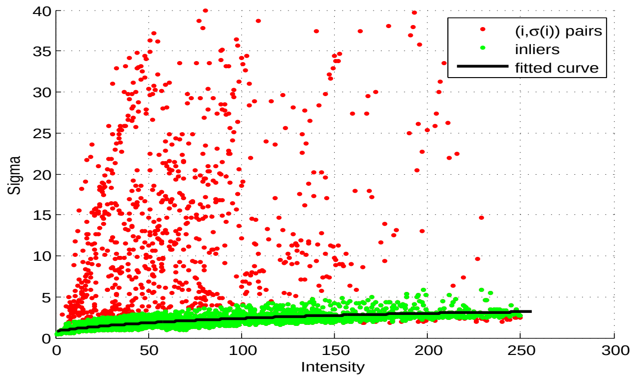


(c) 600nm

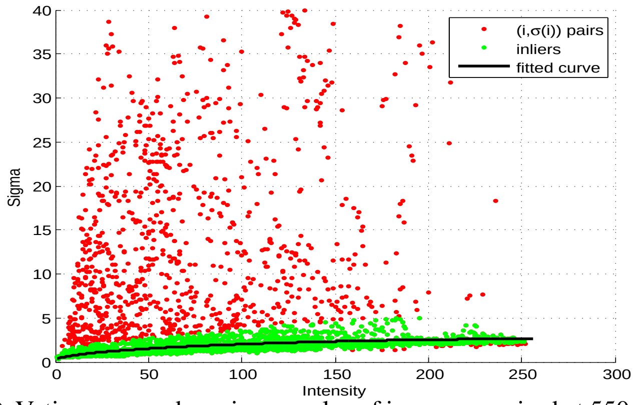
Figure 4.8 : Noise samples estimated at 500, 550 and 600nm respectively.

In Fig.4.9 the $(i, \sigma(i))$ pairs are shown together with the obtained interpolation curve at 500, 550 and 600nm respectively.

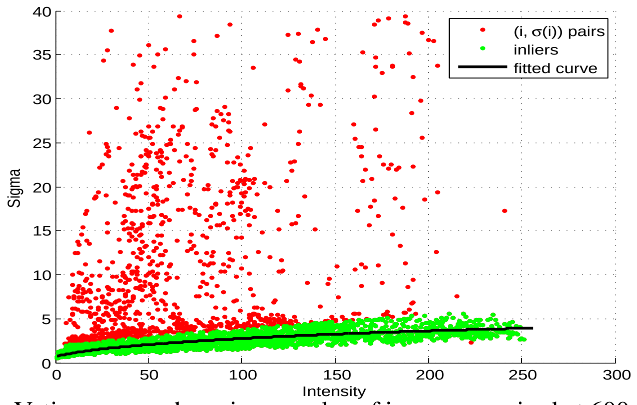
The fitted curves have been then used to improve the quality of the original multispectral images, by making use of a modified signal dependent bilateral filter (see Chapter 5). As can be seen in Figg.4.10 and 4.11 the noise in the homogeneous regions has been considerably reduced without affecting image details.



(a) Voting approach: noise samples of images acquired at 500nm

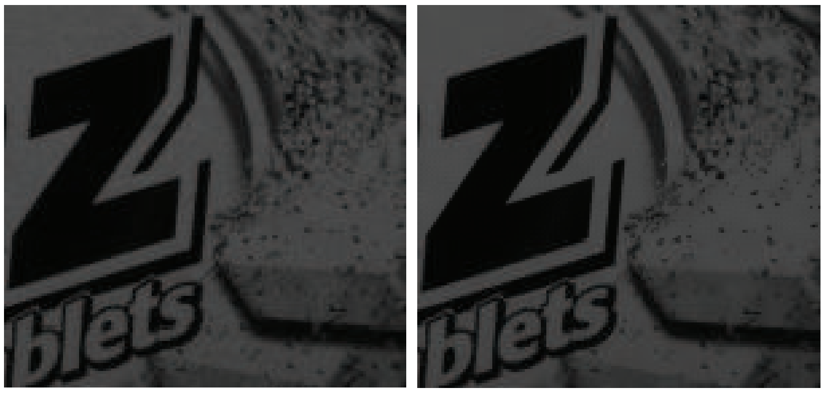


(b) Voting approach: noise samples of images acquired at 550nm

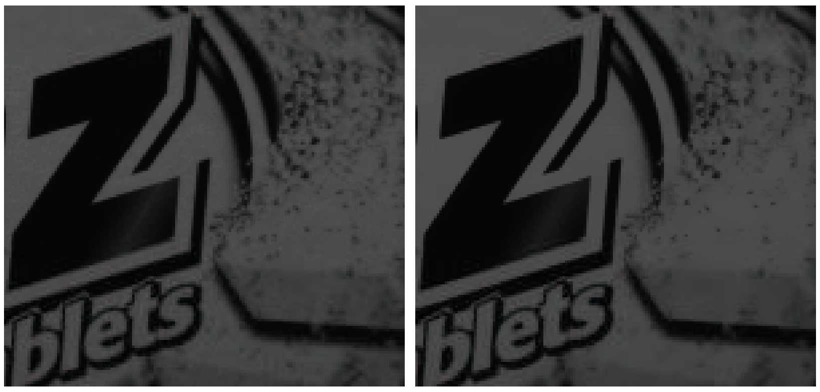


(c) Voting approach: noise samples of images acquired at 600nm

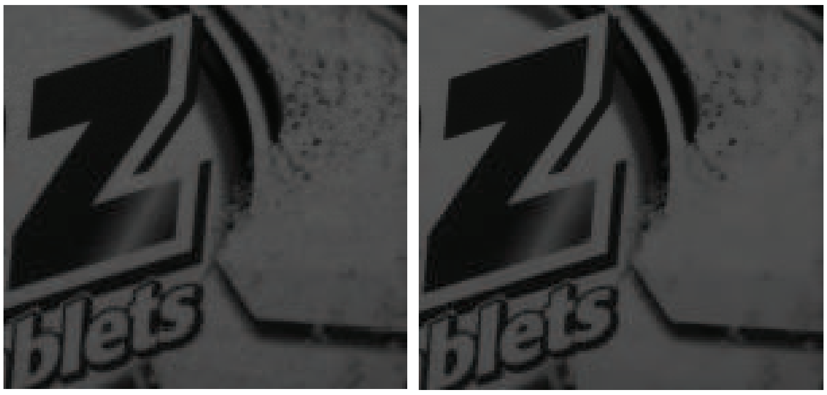
Figure 4.9 : Voting approach selects inliers (green) from the $(i, \sigma(i))$ pair set (red) at 500nm, 550nm, 600nm. These inliers are then used to estimate (through Least Squares) the model parameters. The fitting curve is represented in black.



(a) Noisy and filtered image at 500nm

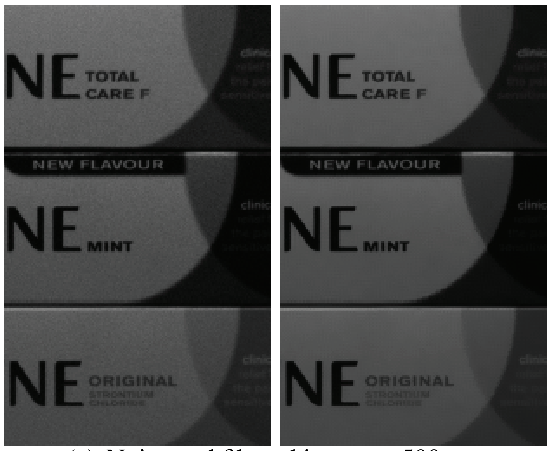


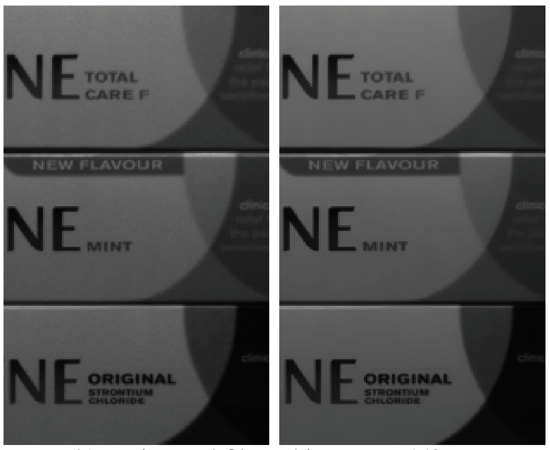
(b) Noisy and filtered image at 550nm



(c) Noisy and filtered image at 600nm

Figure 4.10 : Test figure 1. Crop of noisy and filtered images at 500, 550 and 600nm, obtained using a modified bilateral filter [39].


(a) Noisy and filtered image at 500nm


(b) Noisy and filtered image at 550nm

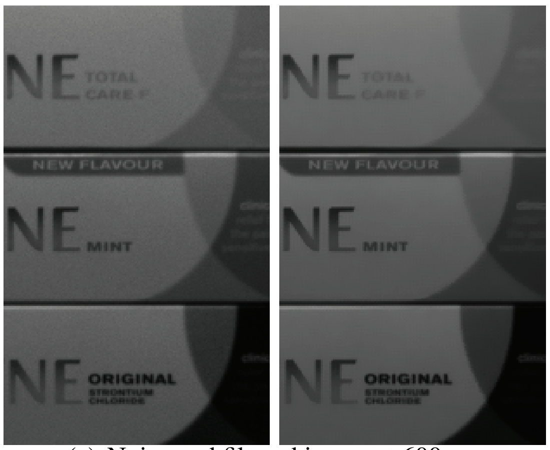

(c) Noisy and filtered image at 600nm

Figure 4.11 : Test figure 2. Crop of noisy and filtered images at 500, 550 and 600nm, obtained using a modified bilateral filter [39].

Chapter 5. Signal Dependent Noise Filtering

5.1 Signal Dependent Sigma Filter

A slightly modified version of the Sigma-Filter [6], detailed in Section 2.4.3, can be used as noise reduction method to remove signal dependent noise; filtering process works by means of SDN curves, that can be estimated as explained in Chapter 4.

Recall that the Sigma Filter process is based on the assumption that the observed pixel value $I(x,y)$ is a good estimate of the local signal mean. Let M be a $m_1 \times m_2$ filter mask and P_c the value of its central pixel, the final filtered output is a weighted average of the pixels having value close to one of the mask central pixel. Weights decrease as the distance in intensity between the central pixel and the neighborhood augments. Under the assumption of Gaussian noise model the Sigma Filter averages all the pixels whose value fall in the range $[P_c - 3\sigma, P_c + 3\sigma]$. Anyway σ value, according to the SDN model (4.1), have to be expressed as a function of the pixel value P_c as follows:

$$\sigma(P_c) = \sqrt{a(\Gamma) \cdot P_c^2 + b(\Gamma) \cdot P_c + c(\Gamma)} \tag{5.1}$$

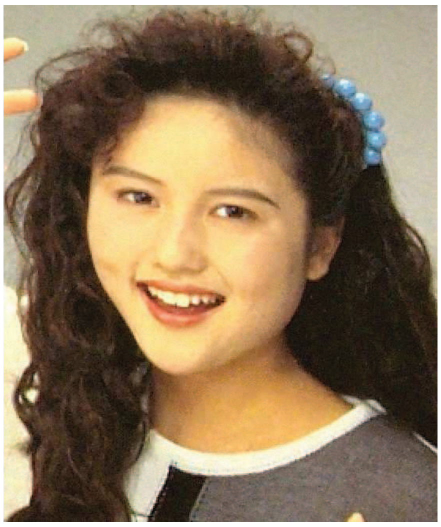
where $a(\Gamma)$, $b(\Gamma)$ and $c(\Gamma)$ are estimated coefficients for the image I with analog gain $AG = \Gamma$.

The Sigma Filter output P_f is computed as the sum of the mask pixels multiplied by their respective weights and divided by the sum of the weights:

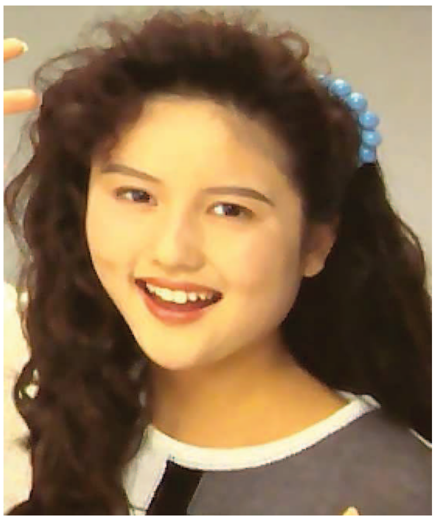
$$P_f = \frac{\sum_{i=0}^{i<(m_1 \times m_2)} w_i \cdot P_i}{\sum_{i=0}^{i<(m_1 \times m_2)} w_i} \tag{5.2}$$

Fig.5.1 illustrates the validity of signal dependent noise filtering 5.1(c) compared to the constant noise level model [6] 5.1(b) using a cropped part of a 10bit raw image acquired using a 3Mp sensor and processed using the *ST image pipeline*. The sigma value used to process the image with the sigma-fixed *(SF)* version of the algorithm [6], was determined by taking the average value of image intensities, and using the corresponding value on the sigma-curves estimated at the same AG used to capture the image, obtaining $\sigma = 18$.

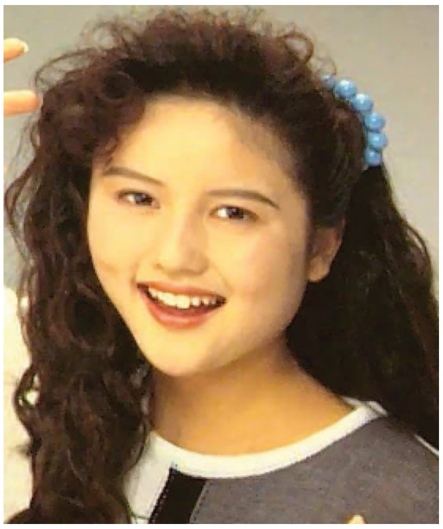
It is evident how the signal dependent approach allows preserving details in a more effective way, generating more pleasant images also in terms of perceptive sharpness. Figg.5.2(a) and 5.2(d) show two zoomed and cropped details of image 5.1(a); it can be noticed how the signal noise dependent model allows better filtering results in terms of sharpness and detail preservation. In fact, while the SF model 5.2(b) and the SDN model 5.2(c) in bright areas have similar performance, in 5.2(e) and 5.2(f) the image portion filtered with the signal dependent noise model retains more details compared to the image processed with the sigma-fixed model; this is caused by lower signal intensity of the considerate area, where a lower sigma is required for the filtering process (Fig.5.3).



(a) Noisy Image

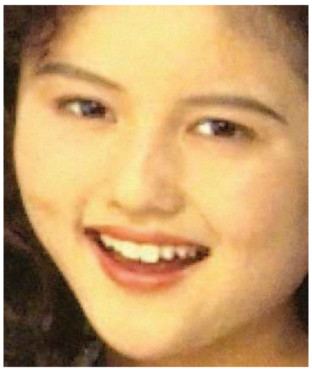


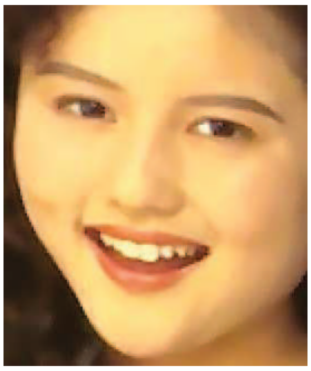
(b) SF Sigma-Filter denoising with ($\sigma =$ 18)

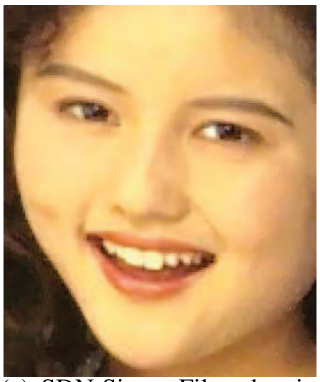


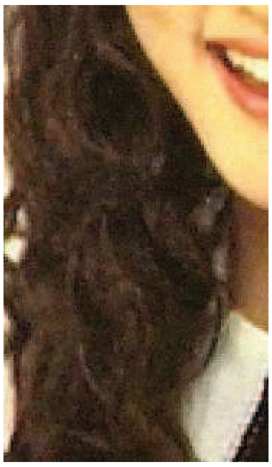
(c) SDN Sigma-Filter denoising

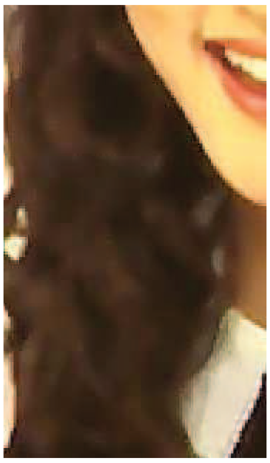
Figure 5.1 : Comparison between SDN filtering approach and SF approach, using a real image acquired with a 3Mp sensor. It is evident how the signal-dependent approach (c) better preserves details and sharpness compared to the fixed noise model filtering (b). At this scale of resolution, the differences may not be evident, depending on the media used to show these images (screen, printer, etc.). Detailed crops of these samples are shown in Fig.5.2.


(a) Crop of noisy image 5.1(a)


(b) SF Sigma-Filter denoising ($\sigma = 18$)


(c) SDN Sigma-Filter denoising


(d) Crop of noisy image 5.1(a)


(e) SF Sigma-Filter denoising based on fixed noise model ($\sigma = 18$)

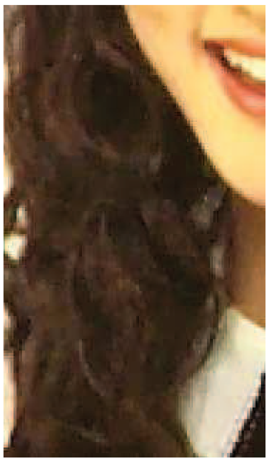

(f) SDN Sigma-Filter denoising

Figure 5.2 : Magnified detail of images in Fig.5.1. In (b) and (c), the SF model and the SDN have similar performances because the sigma-fixed values ($\sigma = 18$) is similar to the values used by the signal dependent model for processing this area. In (e) and (f) the different performances of the two models show up; in this dark crop, the constant noise model uses a fixed sigma which is too high compared to the values taken from the noise curve 5.3.

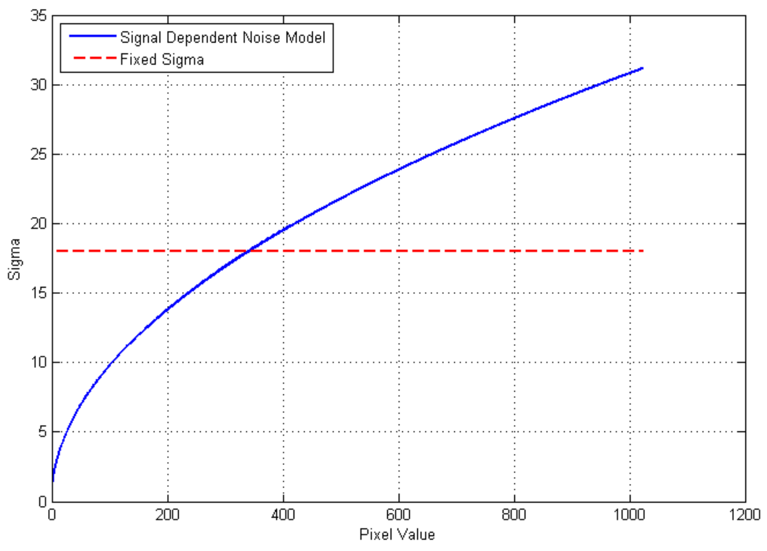


Figure 5.3 : Sigma-curve and costant sigma value (σ = 18) used to filter the image shown in Fig.5.1(a).

5.2 Signal Dependent Bilateral Filter

Bilateral filter [39, 45], presented in Section 2.4.4, can be also adapted to use an SDN estimation. This noise reduction process uses again, a weighted average of local samples, and is based on two different standard deviation values: the intensity related σ_i and the spatial related σ_s.

$$I(\overrightarrow{x}) = \frac{1}{C} \sum_{y \in N(\overrightarrow{x})} e^{\frac{-\|\overrightarrow{y}-\overrightarrow{x}\|}{2\,\sigma_s^2}} \quad e^{\frac{-|I(\overrightarrow{y})-I(\overrightarrow{x})|}{2\,\sigma_i^2}} \quad I(\overrightarrow{y}) \tag{5.3}$$

where $N(\overrightarrow{x})$ is a spatial neighborhood of pixel $I(\overrightarrow{x})$ and C represents the normalization costant:

$$C = \sum_{y \in N(\overrightarrow{x})} e^{\frac{-\|\overrightarrow{y}-\overrightarrow{x}\|}{2\,\sigma_s^2}} \quad e^{\frac{-|I(\overrightarrow{y})-I(\overrightarrow{x})|}{2\,\sigma_i^2}} \tag{5.4}$$

In this equation we can notice that, while σ_s is used to weight the pixels in the mask depending on their distance from the center of the mask, while σ_i, in analogy with the

Sigma Filter, represents the effective noise level which depends on the pixel intensity values. For this reason, likewise Sigma Filter (Eq.5.3), σ_i can be modeled using an SDN model (such as Eq.(4.1)) as follow:

$$\sigma_i(P_c) = \sqrt{a(\Gamma) \cdot P_c^2 + b(\Gamma) \cdot P_c + c(\Gamma)} \tag{5.5}$$

where $a(\Gamma)$, $b(\Gamma)$ and $c(\Gamma)$ are estimated coefficients for the image I with analog gain $AG = \Gamma$.

5.3 Test Results

The algorithms based on a signal dependent noise model *(SDN)*, described in previous Sections, are tested using the Kodak's images test set [1] and the results are compared with the sigma-fixed *(SF)* versions of the related algorithms.

We have added signal dependent noise with variable intensity to the test images following the model (4.1). The noise added is the same as what we would have achieved if we had acquired the images with a 3MP ST sensor and analog gain $AG = 1, 2, 4, 8, 16$. The sigma values used to process each image with the SF version of the algorithms Sigma Filter [6] and Bilateral Filter [39], was determined by using the average value of image intensities, and using the corresponding value on sigma-curves at the related AG.

We calculated the PSNR for each filtered image and the corresponding noisy, obtaining the results described in the following tables: 5.1, 5.3, 5.5, 5.7, 5.9. We also measured the SSIM value [42], i.e., the index of similarity between the filtered image and the clean original, to better observe the advantages of various algorithms (see tables: 5.2, 5.4, 5.6, 5.8, 5.10).

[1] Standard Kodak test images - http://r0k.us/graphics/kodak/

As can be see from the tables, the results obtained with the signal dependent algorithms are certainly superior in terms of PSNR and SSIM respect their counterparts SF. SDN technics, in fact, by using different noise level at at varying intensities, provide an effective filtering, having the advantage of actually responding better to the edges and details of the images and filtering optimally flat areas. Only with very high analog gain (AG=16), however, the SF algorithms seem to obtain better results in terms of SSIM; this is because the images are heavily filtered, then seem to be more similar to the original, although many details are lost.

The effectiveness of SDN filtering methods can also be determined by visual comparison (see Figg. 5.5, 5.6, 5.7, 5.8,5.9), where the advantage of these algorithms is considerable, especially in the way in which the edges and fine details are preserved.

Table 5.1 : AG1 - *PSNR* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	34.74959	35.24183	35.26959	35.16709	35.19698
kodim02	37.08459	37.99202	37.96968	37.64828	37.67370
kodim03	34.88670	36.64730	36.55227	36.42570	36.30483
kodim04	36.95896	38.52280	38.34436	38.02619	37.93403
kodim05	35.61580	35.99400	36.02466	35.81485	35.86205
kodim06	32.84290	34.84649	34.76991	34.41956	34.31775
kodim07	35.57861	38.08058	37.86075	37.77753	37.55052
kodim08	35.45245	35.98176	36.00265	35.81906	35.85969
kodim09	33.96313	37.82626	37.40592	37.70076	37.26373
kodim10	34.97469	37.70385	37.41296	37.58252	37.30110
kodim11	36.60744	38.27543	38.19516	37.86457	37.76396
kodim12	33.20269	35.78670	35.54419	35.45422	35.20424
kodim13	37.36304	37.57261	37.66409	37.35617	37.45777
kodim14	34.88872	35.31289	35.32094	35.20390	35.21244
kodim15	34.52629	37.98665	37.73629	36.55551	36.30336
kodim16	34.44982	37.21392	36.89351	37.07024	36.75801
kodim17	38.66623	39.40549	39.34746	39.00804	38.96821
kodim18	37.47080	37.61644	37.70459	37.54217	37.63532
kodim19	35.31433	38.11314	37.74395	38.07954	37.70599
kodim20	31.40890	36.00091	35.55803	35.72922	35.26558
kodim21	34.21362	37.53664	37.19758	37.36449	37.03256
kodim22	34.53973	35.76404	35.71290	35.50917	35.45657
kodim23	34.67054	36.82005	36.49245	36.43052	36.15563
kodim24	34.26248	36.06770	35.98733	35.68559	35.58252
$GAIN_{avg}$ (dB)		**1.85906**	**1.70913**	**1.56429**	**1.41977**

Table 5.2 : AG1 - *SSIM* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	0.97669	0.93968	0.97956	0.93947	0.97931
kodim02	0.99613	0.96915	0.99680	0.96889	0.99657
kodim03	0.96106	0.94170	0.98040	0.94309	0.98224
kodim04	0.97765	0.95190	0.98532	0.94951	0.98299
kodim05	0.98051	0.94976	0.98354	0.94895	0.98290
kodim06	0.97348	0.94388	0.98462	0.94177	0.98240
kodim07	0.93862	0.92683	0.96744	0.92470	0.96489
kodim08	0.96704	0.93229	0.97213	0.93099	0.97114
kodim09	0.82925	0.87215	0.91498	0.87061	0.91305
kodim10	0.91681	0.90832	0.95301	0.90771	0.95253
kodim11	0.92394	0.92238	0.96019	0.91552	0.95191
kodim12	0.94797	0.92759	0.96968	0.92512	0.96683
kodim13	0.98371	0.94535	0.98583	0.94378	0.98436
kodim14	0.96706	0.93402	0.97025	0.93338	0.96963
kodim15	0.96636	0.94556	0.98384	0.93747	0.97580
kodim16	0.91201	0.91507	0.95261	0.91460	0.95230
kodim17	0.96252	0.92262	0.96898	0.91886	0.96519
kodim18	0.97841	0.94824	0.98166	0.94750	0.98121
kodim19	0.86897	0.87763	0.92134	0.87629	0.91967
kodim20	0.77596	0.85246	0.89705	0.84708	0.89015
kodim21	0.94965	0.92958	0.97273	0.92887	0.97188
kodim22	0.94322	0.92250	0.96477	0.91929	0.96099
kodim23	0.94893	0.92959	0.96710	0.92566	0.96361
kodim24	0.91800	0.91261	0.95399	0.90349	0.94355
$SSIM_{avg}$	**0.94016**	**0.92587**	**0.96533**	**0.92344**	**0.96271**

Table 5.3 : AG2 - *PSNR* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	29.62765	30.85200	30.77186	30.76670	30.69148
kodim02	31.50375	33.31532	33.09713	33.09154	32.99279
kodim03	29.78230	32.27954	31.93061	32.05679	31.84142
kodim04	31.35997	33.93780	33.64227	33.62681	33.38482
kodim05	30.44175	31.33794	31.19074	31.18001	31.14023
kodim06	28.09800	30.42199	30.28046	30.09981	29.93752
kodim07	30.27507	33.32510	33.02930	33.10235	32.80436
kodim08	30.17987	31.32580	31.26461	31.22667	31.17920
kodim09	28.96382	33.49083	33.02008	33.39348	32.91197
kodim10	29.84105	33.50207	33.11873	33.42333	33.04540
kodim11	31.12757	33.40287	33.05409	33.14482	32.95472
kodim12	28.39436	31.75453	31.42617	31.48776	31.14046
kodim13	31.64802	32.41455	32.34049	32.26384	32.24992
kodim14	29.77718	31.01623	30.86855	30.86335	30.74453
kodim15	29.60406	33.20814	32.84645	32.17699	31.84785
kodim16	29.33813	33.05879	32.63329	32.89586	32.46970
kodim17	32.74560	34.64551	34.34593	34.37132	34.18037
kodim18	31.73876	32.50051	32.47821	32.44985	32.44231
kodim19	30.01755	33.93060	33.47874	33.93037	33.46962
kodim20	26.91008	31.77383	31.28375	31.55978	31.06197
kodim21	29.14270	33.22598	32.81210	33.11058	32.69320
kodim22	29.43847	31.43940	31.27931	31.22483	31.06580
kodim23	29.57268	32.78791	32.33579	32.40474	31.99829
kodim24	29.29853	31.38437	31.25926	31.17807	31.04020
$GAIN_{avg}$ (dB)		**2.56270**	**2.29004**	**2.34178**	**2.10255**

Table 5.4 : AG2 - *SSIM* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	0.93204	0.87185	0.94937	0.87160	0.94920
kodim02	0.99018	0.93808	0.99382	0.93766	0.99363
kodim03	0.86201	0.84719	0.92310	0.85312	0.93133
kodim04	0.87561	0.87251	0.93287	0.86671	0.92629
kodim05	0.92734	0.88092	0.95134	0.87978	0.95081
kodim06	0.87734	0.87535	0.95010	0.87049	0.94473
kodim07	0.74926	0.80359	0.88500	0.80243	0.88406
kodim08	0.84595	0.81540	0.89653	0.81453	0.89649
kodim09	0.56292	0.69694	0.77122	0.69681	0.77030
kodim10	0.83800	0.83646	0.91842	0.83652	0.91833
kodim11	0.66409	0.73962	0.81208	0.73219	0.80291
kodim12	0.82142	0.83364	0.91924	0.83197	0.91692
kodim13	0.94573	0.87797	0.95235	0.87754	0.95260
kodim14	0.85126	0.80987	0.88499	0.80802	0.88313
kodim15	0.85825	0.86296	0.93759	0.85711	0.93052
kodim16	0.78127	0.81250	0.89982	0.81682	0.90556
kodim17	0.90511	0.88773	0.94880	0.89216	0.95432
kodim18	0.89009	0.86348	0.92351	0.85968	0.91948
kodim19	0.72002	0.77824	0.85491	0.77546	0.85172
kodim20	0.47201	0.62805	0.70417	0.62483	0.70133
kodim21	0.89991	0.87599	0.96015	0.87560	0.95970
kodim22	0.83598	0.80874	0.89275	0.80601	0.88904
kodim23	0.90060	0.87695	0.95264	0.87829	0.95481
kodim24	0.74826	0.77131	0.84515	0.75997	0.83177
$SSIM_{avg}$	**0.82311**	**0.82772**	**0.90250**	**0.82605**	**0.90079**

Table 5.5 : AG4 - *PSNR* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	28.35616	29.78968	29.69447	29.68951	29.59767
kodim02	30.23546	32.26786	32.09242	32.03121	31.89678
kodim03	28.58153	31.30715	31.08785	31.03151	30.78785
kodim04	30.22814	33.09524	32.77256	32.77976	32.50211
kodim05	29.15596	30.20327	30.10877	30.00799	29.94871
kodim06	26.90997	29.38241	29.23112	29.05648	28.88137
kodim07	29.00144	32.21598	31.90091	31.95230	31.63599
kodim08	28.92227	30.27435	30.17516	30.14487	30.06932
kodim09	27.74002	32.41003	31.93707	32.32915	31.83329
kodim10	28.51690	32.31768	31.90970	32.22382	31.81550
kodim11	29.93086	32.36931	32.16449	32.07065	31.85749
kodim12	27.09325	30.67393	30.31142	30.39177	30.01900
kodim13	30.37236	31.33987	31.28202	31.16120	31.12571
kodim14	28.46847	29.96079	29.80830	29.77703	29.62785
kodim15	28.37972	32.14720	31.73975	31.07685	30.71822
kodim16	28.09992	31.98699	31.56194	31.82547	31.39726
kodim17	31.58448	33.73035	33.40346	33.41045	33.18176
kodim18	30.50684	31.41376	31.37228	31.35032	31.31754
kodim19	28.81394	32.94863	32.47906	32.96116	32.48454
kodim20	25.58448	30.45077	29.95101	30.23536	29.74024
kodim21	27.93544	32.18134	31.76022	32.06810	31.63405
kodim22	28.23403	30.47613	30.27803	30.23905	30.03771
kodim23	28.38024	31.85509	31.37583	31.44905	31.00432
kodim24	28.02604	30.23791	30.07835	30.02715	29.85602
$GAIN_{avg}$ (dB)		**2.74908**	**2.4758**	**2.50968**	**2.24635**

Table 5.6 : AG4 - *SSIM* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	0.91332	0.86159	0.93741	0.86118	0.93704
kodim02	0.98706	0.93660	0.99234	0.93604	0.99191
kodim03	0.82737	0.82835	0.90120	0.83539	0.91134
kodim04	0.84725	0.86150	0.91974	0.85414	0.91151
kodim05	0.90858	0.87198	0.94000	0.87086	0.93959
kodim06	0.84121	0.86157	0.93386	0.85510	0.92668
kodim07	0.68948	0.77755	0.85220	0.77632	0.85101
kodim08	0.80480	0.79489	0.87236	0.79419	0.87276
kodim09	0.48999	0.65376	0.71714	0.65413	0.71657
kodim10	0.79773	0.82120	0.89828	0.82125	0.89812
kodim11	0.60737	0.70221	0.76720	0.69418	0.75725
kodim12	0.77705	0.81774	0.89854	0.81579	0.89589
kodim13	0.92984	0.86652	0.93971	0.86567	0.93992
kodim14	0.81514	0.78932	0.86055	0.78698	0.85810
kodim15	0.82125	0.84844	0.91994	0.84048	0.91058
kodim16	0.72815	0.79019	0.87238	0.79569	0.87969
kodim17	0.88388	0.87887	0.93874	0.88574	0.94680
kodim18	0.86125	0.84773	0.90549	0.84313	0.90007
kodim19	0.66263	0.74747	0.82034	0.74494	0.81727
kodim20	0.40810	0.58215	0.64751	0.57864	0.64395
kodim21	0.87164	0.86695	0.94909	0.86673	0.94862
kodim22	0.80377	0.79004	0.87019	0.78668	0.86569
kodim23	0.87730	0.86982	0.94397	0.87187	0.94683
$SSIM_{avg}$	**0.7872**	**0.80880**	**0.87961**	**0.80698**	**0.87774**

Table 5.7 : AG8 - *PSNR* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	24.92398	26.93885	26.78827	26.82943	26.67537
kodim02	26.70580	29.45190	29.01574	29.18544	28.96265
kodim03	25.01730	28.25151	27.37712	27.93873	27.60716
kodim04	26.63413	30.08995	29.69976	29.85800	29.50627
kodim05	25.68645	27.23099	26.94983	27.01877	26.86982
kodim06	23.41486	26.27248	26.06368	25.97654	25.73854
kodim07	25.59096	29.10075	28.75056	28.86818	28.53869
kodim08	25.45677	27.33971	27.17264	27.24314	27.08877
kodim09	24.24915	29.03409	28.55505	28.99521	28.48200
kodim10	25.09847	29.25057	28.81694	29.20665	28.75767
kodim11	26.49293	29.34799	28.83583	29.12110	28.84441
kodim12	23.70148	27.77245	27.36346	27.52998	27.09338
kodim13	26.89983	28.50779	28.29914	28.36253	28.22794
kodim14	25.06194	27.26666	26.97930	27.06526	26.81487
kodim15	24.91582	28.89497	28.04866	27.93696	27.52302
kodim16	24.66960	28.98974	28.52734	28.84515	28.36448
kodim17	27.92703	30.71782	30.12445	30.39632	30.08031
kodim18	26.97623	28.46026	28.17465	28.38877	28.27281
kodim19	25.30910	29.75916	29.29225	29.83573	29.34490
kodim20	22.18379	27.10491	26.59447	26.94154	26.42817
kodim21	24.49327	28.94967	28.51564	28.90678	28.44650
kodim22	24.78112	27.62589	27.36195	27.43588	27.14944
kodim23	24.90001	28.92622	28.41813	28.49428	28.00189
kodim24	24.53935	27.04813	26.83993	26.88377	26.66752
$GAIN_{avg}$ (dB)		**3.19596**	**2.78897**	**2.98478**	**2.66072**

Table 5.8 : AG8 - *SSIM* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	0.83746	0.81739	0.88658	0.81664	0.88559
kodim02	0.97102	0.93054	0.98452	0.92963	0.98413
kodim03	0.71323	0.76204	0.82310	0.77396	0.83951
kodim04	0.73844	0.80959	0.85984	0.79991	0.84876
kodim05	0.82916	0.82940	0.88658	0.82802	0.88949
kodim06	0.72090	0.80919	0.87242	0.79939	0.86156
kodim07	0.50996	0.67947	0.73480	0.67906	0.73417
kodim08	0.66078	0.72219	0.78435	0.72437	0.78844
kodim09	0.30680	0.50915	0.54921	0.51198	0.55059
kodim10	0.65907	0.75783	0.81945	0.75962	0.82115
kodim11	0.45386	0.58352	0.78301	0.57652	0.62009
kodim12	0.60928	0.74225	0.80603	0.73942	0.80178
kodim13	0.85357	0.81800	0.88357	0.81821	0.88637
kodim14	0.66984	0.70686	0.76311	0.70363	0.75938
kodim15	0.68412	0.78531	0.84462	0.77337	0.83062
kodim16	0.55745	0.70046	0.76545	0.71180	0.77865
kodim17	0.75701	0.81526	0.85717	0.84374	0.89740
kodim18	0.74924	0.78124	0.82551	0.77734	0.82445
kodim19	0.48820	0.65025	0.70523	0.64984	0.70503
kodim20	0.25684	0.43668	0.48291	0.43512	0.48208
kodim21	0.75494	0.82317	0.89590	0.82408	0.89634
kodim22	0.68683	0.71357	0.77943	0.70949	0.77441
kodim23	0.77021	0.82470	0.88941	0.82974	0.89548
kodim24	0.63087	0.66191	0.71812	0.65062	0.70666
$SSIM_{avg}$	**0.66121**	**0.73625**	**0.8000**	**0.73606**	**0.79426**

Table 5.9 : AG16 - *PSNR* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	21.77130	24.32551	24.07544	24.28335	24.04779
kodim02	23.38835	26.71072	26.28839	26.69253	26.37244
kodim03	21.96400	25.65707	25.15957	25.43183	25.03442
kodim04	23.32953	27.25089	26.82941	27.32749	26.91100
kodim05	22.46622	24.45851	23.55486	24.34071	24.11478
kodim06	20.47102	23.72794	23.44164	23.51427	23.20344
kodim07	22.28903	25.99799	25.59276	25.90020	25.53774
kodim08	22.26718	24.74117	24.46728	24.73779	24.51045
kodim09	21.22578	26.10801	25.58291	26.13307	25.61488
kodim10	21.91767	26.27675	25.81266	26.33222	25.84389
kodim11	23.07271	26.28096	25.06844	26.20690	25.86916
kodim12	20.68932	25.03977	24.59592	24.93056	24.44989
kodim13	23.49033	25.88105	25.58928	25.89004	25.63679
kodim14	21.90099	24.75514	24.23716	24.62087	24.27942
kodim15	21.81081	25.90361	24.75050	25.28931	24.82901
kodim16	21.56759	26.09366	25.62567	26.05976	25.55264
kodim17	24.45252	27.76267	27.07776	27.73907	27.36133
kodim18	23.60495	25.67968	25.14752	25.73193	25.52218
kodim19	22.14410	26.78340	26.29124	26.98417	26.47820
kodim20	19.40421	24.37649	23.79873	24.28627	23.75541
kodim21	21.39857	25.98234	25.48065	26.01198	25.52655
kodim22	21.69124	25.08361	24.70294	24.99824	24.64525
kodim23	21.76497	26.06192	25.57326	25.80797	25.29563
kodim24	21.49540	24.40909	24.13935	24.35116	24.07413
$GAIN_{avg}$ (dB)		**3.57376**	**3.05440**	**3.50100**	**3.12036**

Table 5.10 : AG16 - *SSIM* comparison between Signal Dependent and Sigma-Fixed Noise Removal Approaches

Filename	Noisy	SDF		SF	
		$SDN_{SIGMAfilt}$	$SDN_{BILATfilt}$	$SF_{SIGMAfilt}$	$SF_{BILATfilt}$
kodim01	0.73394	0.75392	0.81333	0.75414	0.81370
kodim02	0.93838	0.91812	0.96932	0.91815	0.97021
kodim03	0.58870	0.67631	0.72307	0.69016	0.74084
kodim04	0.60937	0.72810	0.76991	0.72065	0.76060
kodim05	0.71524	0.76215	0.79990	0.76375	0.81367
kodim06	0.58958	0.73638	0.78736	0.72579	0.77679
kodim07	0.33408	0.54513	0.57934	0.54844	0.58179
kodim08	0.49500	0.61584	0.65904	0.62187	0.66702
kodim09	0.18070	0.36663	0.38397	0.37115	0.38702
kodim10	0.49341	0.65901	0.69871	0.66510	0.70475
kodim11	0.32376	0.44873	0.47928	0.44838	0.47841
kodim12	0.45655	0.65252	0.69846	0.65089	0.69502
kodim13	0.72299	0.73713	0.79108	0.74433	0.80171
kodim14	0.51363	0.60853	0.65023	0.60704	0.64770
kodim15	0.52471	0.68925	0.72705	0.67893	0.72003
kodim16	0.39169	0.57233	0.61804	0.58847	0.63516
kodim17	0.50823	0.65658	0.67043	0.72762	0.76346
kodim18	0.59340	0.67653	0.69896	0.68462	0.71752
kodim19	0.32559	0.51583	0.55251	0.52341	0.55982
kodim20	0.16420	0.31476	0.34502	0.31453	0.34643
kodim21	0.60438	0.75338	0.81146	0.75700	0.81441
kodim22	0.56922	0.63720	0.68788	0.63552	0.68541
kodim23	0.62735	0.74935	0.80040	0.76003	0.81232
kodim24	0.55323	0.58508	0.62922	0.57788	0.62534
$SSIM_{avg}$	**0.52322**	**0.63995**	**0.68100**	**0.64491**	**0.68830**

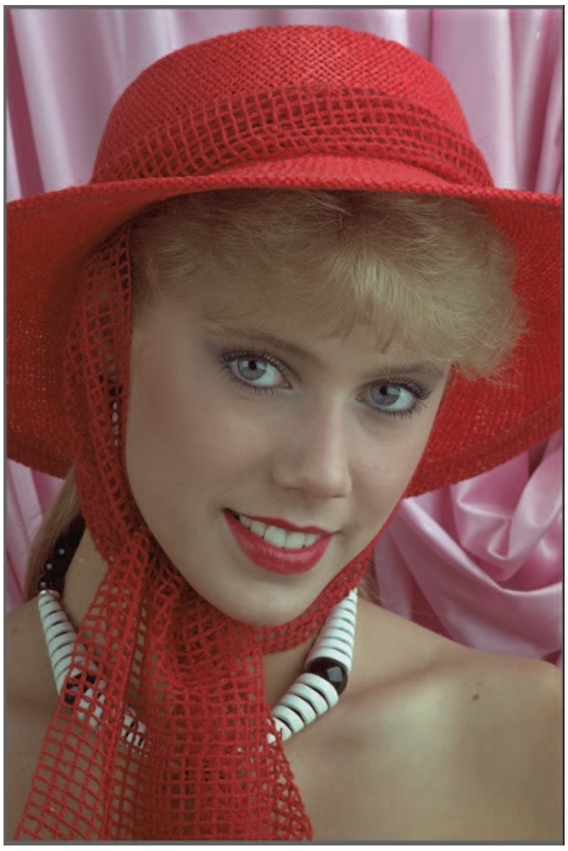
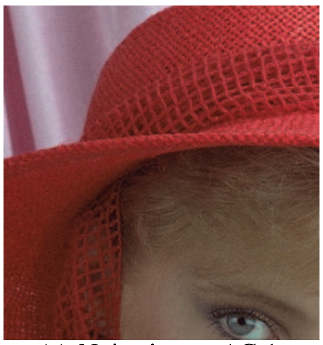
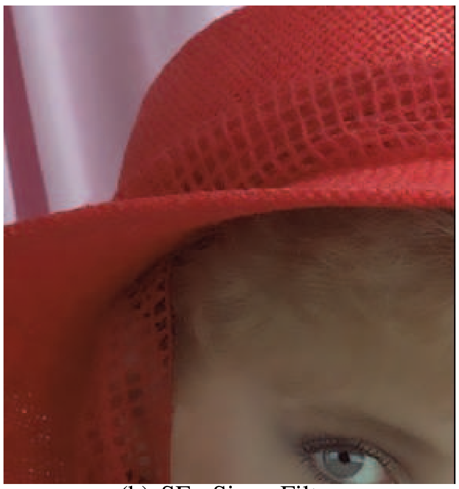
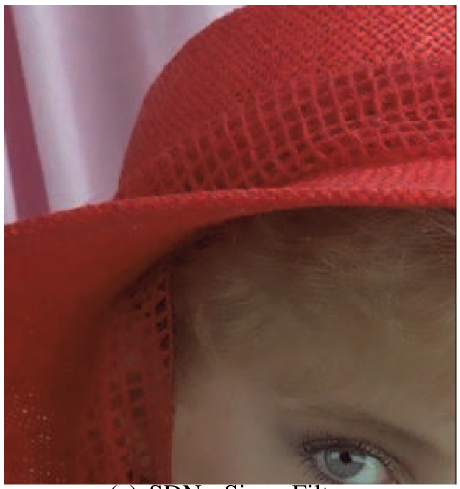


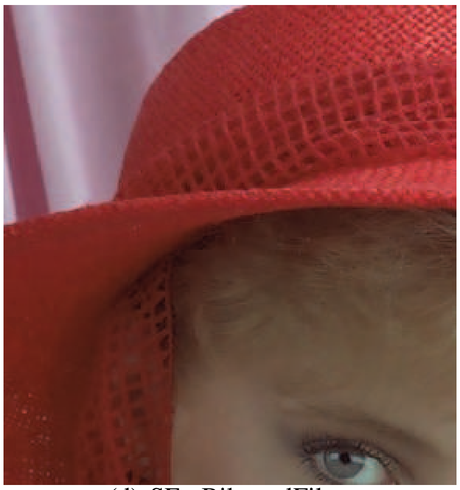
Figure 5.4 : Reference test image *kodim04* used for visual comparison at different AG in Figg. 5.5, 5.6, 5.7, 5.8, 5.9

.


(a) Noisy image AG 1


(b) SF - SigmaFilter


(c) SDN - SigmaFilter


(d) SF - BilateralFilter

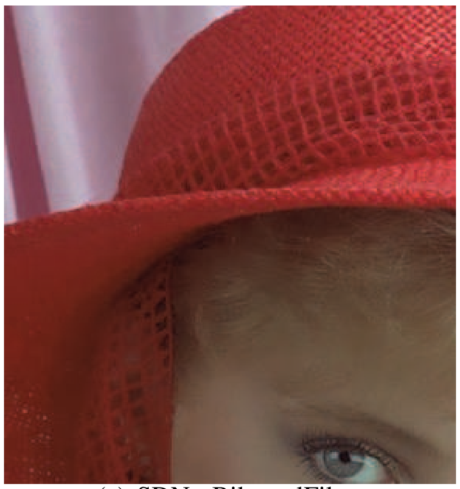
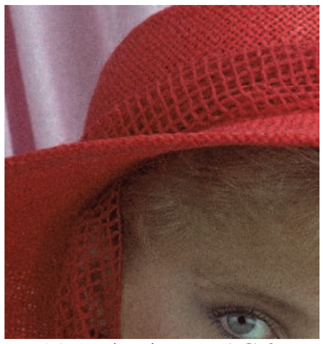
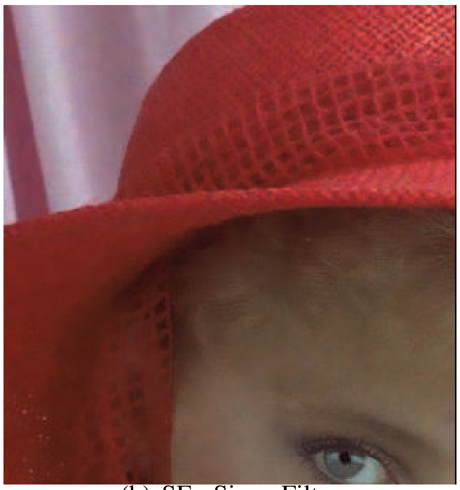
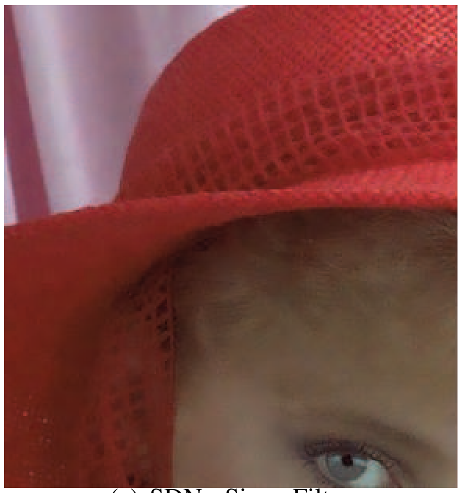
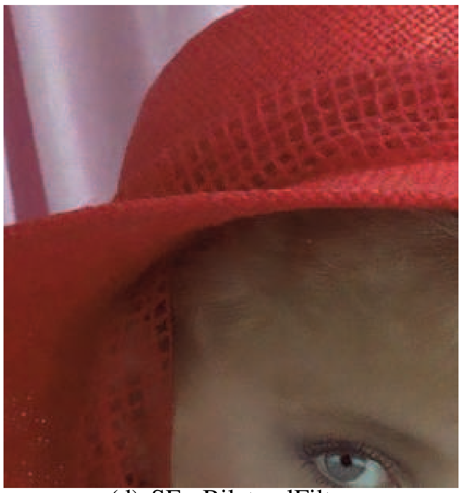

(e) SDN - BilateralFilter

Figure 5.5 : AG1. Cropped part of image *kodim04* shown in Fig.5.4 - Visual comparison between sigma fixed and signal dependent noise removal approaches.


(a) Noisy image AG 2


(b) SF - SigmaFilter


(c) SDN - SigmaFilter


(d) SF - BilateralFilter

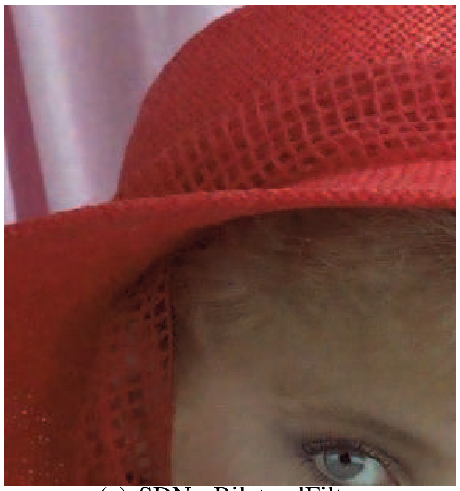
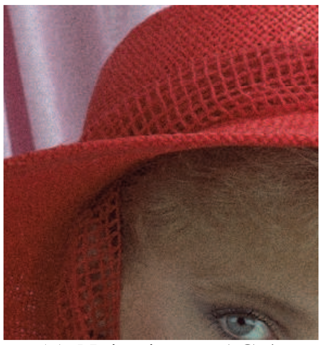
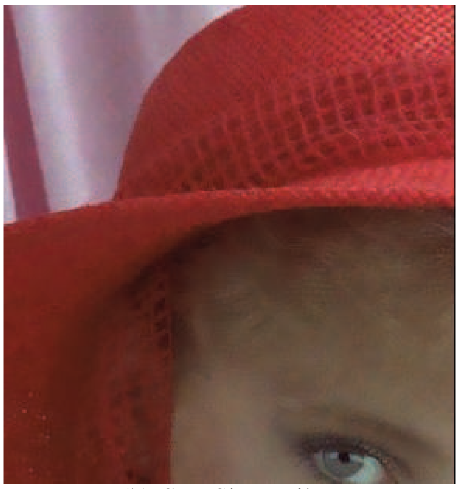
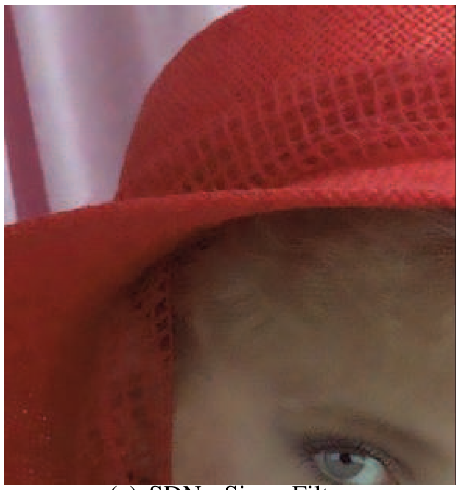
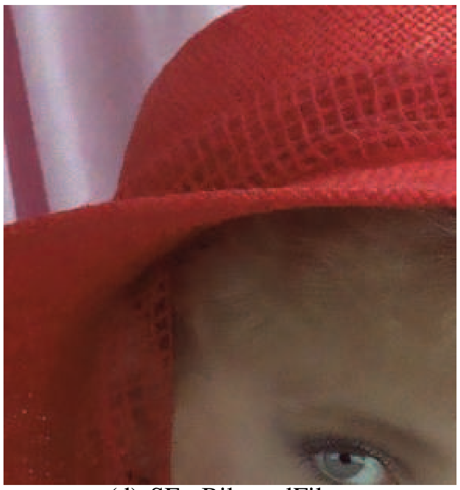

(e) SDN - BilateralFilter

Figure 5.6 : AG2. Cropped part of image *kodim04* shown in Fig.5.4 - Visual comparison between sigma fixed and signal dependent noise removal approaches.


(a) Noisy image AG 4


(b) SF - SigmaFilter


(c) SDN - SigmaFilter


(d) SF - BilateralFilter

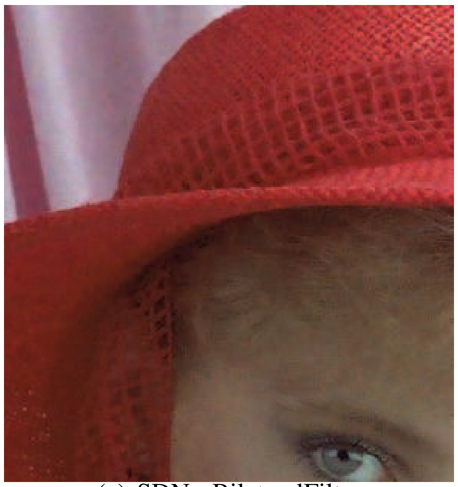
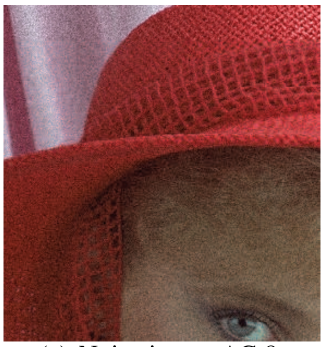
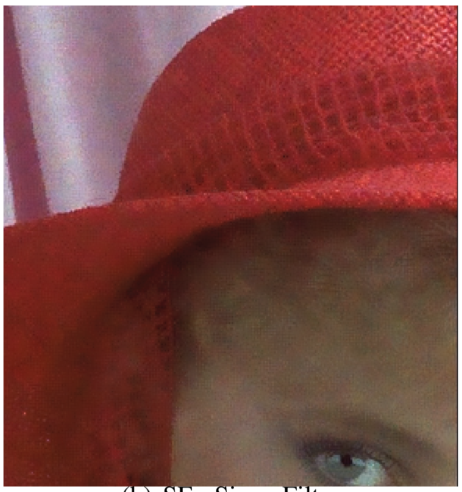
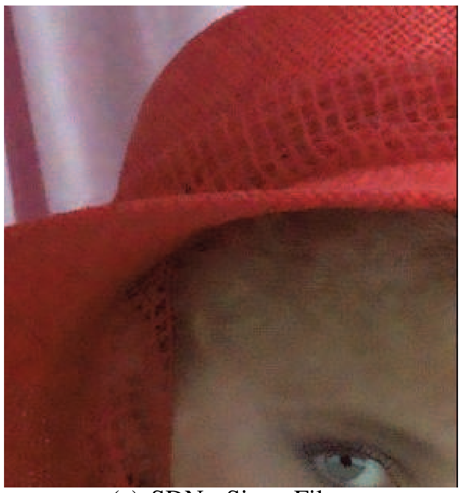
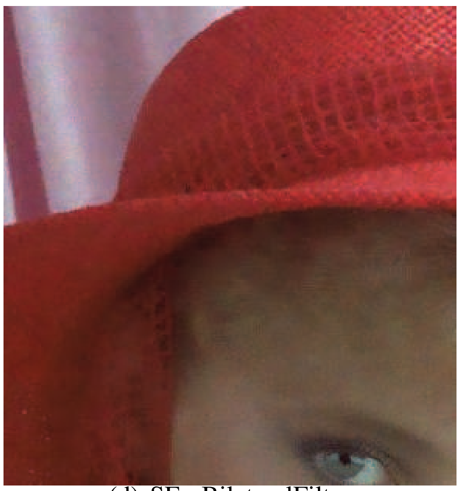

(e) SDN - BilateralFilter

Figure 5.7 : AG4. Cropped part of image *kodim04* shown in Fig.5.4 - Visual comparison between sigma fixed and signal dependent noise removal approaches.


(a) Noisy image AG 8


(b) SF - SigmaFilter


(c) SDN - SigmaFilter


(d) SF - BilateralFilter

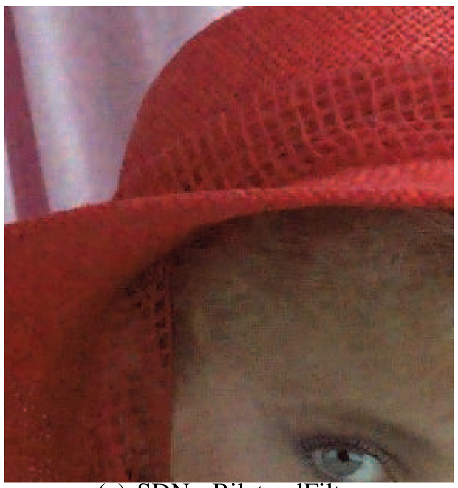
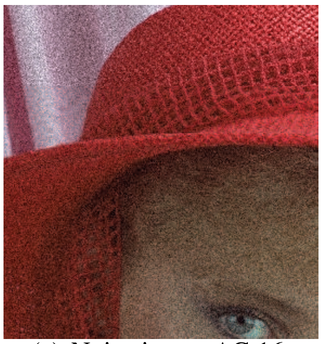
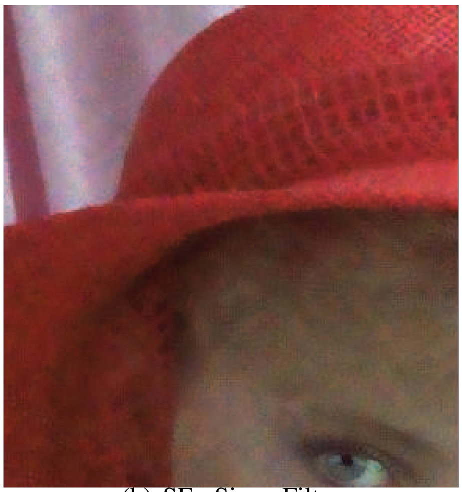
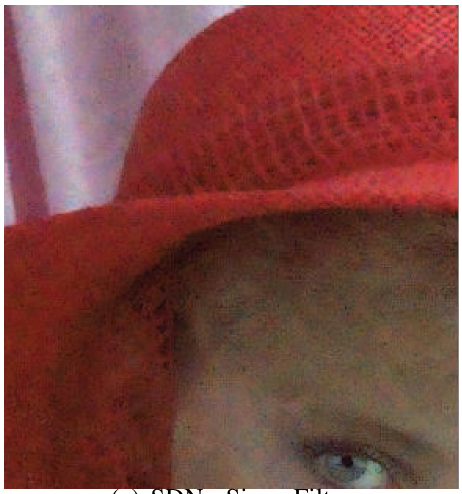
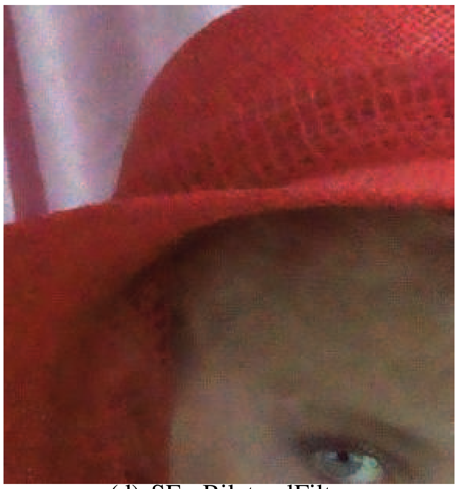

(e) SDN - BilateralFilter

Figure 5.8 : AG8. Cropped part of image *kodim04* shown in Fig.5.4 - Visual comparison between sigma fixed and signal dependent noise removal approaches.


(a) Noisy image AG 16


(b) SF - SigmaFilter


(c) SDN - SigmaFilter


(d) SF - BilateralFilter

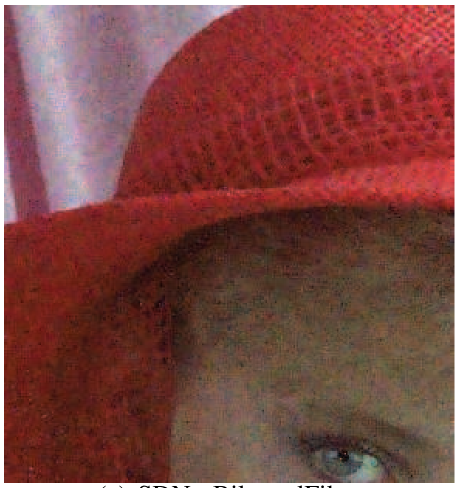

(e) SDN - BilateralFilter

Figure 5.9 : AG16. Cropped part of image *kodim04* shown in Fig.5.4 - Visual comparison between sigma fixed and signal dependent noise removal approaches

Bibliography

[1] A. Bosco, S. Battiato, A. Bruna, and R. Rizzo. **Noise Reduction for CFA Image Sensors Exploiting HVS Behaviour**. *Sensors*, **9**(3):1692–1713, 2009.

[2] A. Amer and E. Dubois. **Fast and Reliable Structure-Oriented Video Noise Estimation**. *IEEE Transactions on Circuits and Systems for Video Technology.*, **15**(1):113–118, 2005.

[3] N. Azzabou, N. Paragios, F. Guichard, and F. Cao. **Variable Bandwidth Image Denoising Using Image-based Noise Models**. In *proceedings of Computer Vision and Pattern Recognition (CVPR-07)*, pages 1–7, 2007.

[4] A. Bosco, D. Giacalone, S. Battiato, and R. Rizzo. **Signal dependent raw image denoising using sensor noise characterization via multiple acquisitions**. *proceedings of IS&T/SPIE Electronic Imaging, Digital Photography VI*, **7537**:753705, 2010.

[5] A. Bosco, A. Bruna, S. Smith, and V. Tomaselli. **Fast Noise Level Estimation using a Convergent Multiframe Approach**. In *proceedings of International Conference on Image Processing (ICIP-06)*, pages 2621–2624. IEEE, 2006.

[6] A. Bosco, A. Bruna, G. Spampinato, and G. Messina. **Fast Method for Noise Level Estimation and Integrated Noise Reduction**. *IEEE Transactions on Consumer Electronics*, **51**(3):1028–1033, 2005.

[7] A. BOSCO, K. FINDLATER, S. BATTIATO, AND A. CASTORINA. **A noise reduction filter for full-frame data imaging devices**. *IEEE Transactions on Consumer Electronics*, **49**(3):676–682, 2003.

[8] A. BUADES, B. COLL, AND J. M. MOREL. **A review of image denoising algorithms, with a new one**. *Journal on Multiscale Modeling and Simulation*, **4**:490–530, 2005.

[9] A. BUADES, B. COLL, AND J.M. MOREL. **The Staircasing Effect in Neighborhood Filters and its Solution**. *IEEE Transaction on Image Processing*, **15**(6):1499–1505, 2006.

[10] C. LIU, R. SZELISKI, S. B. KANG, C. L. ZITNICK, AND W. T. FREEMAN. **Automatic Estimation and Removal of Noise from a Single Image**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**(2):299–314, 2008.

[11] C. STAELIN AND H. NACHLIELI. **Image Noise Estimation Using Color Information**. Technical Report 2005-2R1, Hewlett-Packard Development Company, 2007.

[12] C.H. CHOU AND Y.C. LI. **A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile**. *IEEE Transaction on Circuits Systems and Video Technology*, **5**(6):467–476, 1995.

[13] A. FOI, S. ALENIUS, V. KATKOVNIK, AND K. EGIAZARIAN. **Noise Measurement for Raw-Data of Digital Imaging Sensors by Automatic Segmentation of Nonuniform Targets**. *IEEE Sensors Journal*, **7**(10):1456–1461, 2007.

[14] A. Foi, M. Trimeche, V. Katkovnik, and K.O. Egiazarian. **Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data**. *IEEE Transactions on Image Processing*, **17**(10):1737–1754, 2008.

[15] R.C. Gonzalez and R.E. Woods. *Digital Image Processing 2nd Edition*. Prentice Hall, 2002.

[16] Hewlett-Packard Components Group. **Noise Sources in CMOS Image Sensors**. Technical report, Hewlett-Packard Company, 1998.

[17] G.E. Healey and R. Kondepudy. **CCD camera calibration and noise estimation**. In *proceedings of Computer Vision and Pattern Recognition (CVPR-92)*, pages 90–95, 1992.

[18] K. Hirakawa and T. W. Parks. **Joint demosaicing and denoising**. *IEEE Transactions on Image Processing*, **15**:2146–2157, 2006.

[19] I. Hontsch and L.J. Karam. **Locally Adaptive Perceptual Image Coding**. *IEEE Transactions on Image Processing*, **9**(9):1472–1483, 2000.

[20] S. Hordley, G. Finalyson, and P. Morovic. **A Multi-Spectral Image Database and Its Application to Image Rendering across Illumination**. In *proceedings of the Third International Conference on Image and Graphics*, pages 394–397, 2004.

[21] N. Jayant, J. Johnston, and R. Safranek. **Signal compression based on models of human perception**. In *proceedings of the IEEE*, 10, pages 1385–1422, 1993.

[22] O. KALEVO AND H. RANTANEN. **Noise reduction techniques for Bayer-matrix images**. In *proceedings of IS&T/SPIE Electronic Imaging, Society of Photo-Optical Instrumentation Engineers Conference*, **4669**, pages 348–359, 2002.

[23] H.K. KWAN AND Y. CAI. **Fuzzy filters for image filtering**. In *proceedings of the IEEE 45th Midwest Symposium on Circuits and Systems*, pages 672–5, 2002.

[24] N.X. LIAN, L. CHANG, AND Y.P. TAN. **Improved Color Filter Array Demosaicking by Accurate Luminance Estimation**. In *proceedings of IEEE International Conference on Image Processing (ICIP-05)*, pages 41–44, 2005.

[25] P. LONGERE, X.M. ZHANG, P.B. DELAHUNT, AND D.H. BRAINARD. **Perceptual assessment of demosaicing algorithm performance**. In *proceedings of the IEEE*, 1, pages 123–132, 2002.

[26] R. LUKAC. **Single-Sensor Imaging in Consumer Digital Cameras: A survey of Recent Advances and Future Directions**. *Journal of Real-Time Image Processing*, **1**(1):45–52, 2006.

[27] STEPHANE G. MALLAT. **A theory for multiresolution signal decomposition: the wavelet representation**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**:674–693, 1989.

[28] M. J. NADENAU, S. WINKLER, D. ALLEYSSON, AND M. KUNT. **Human Vision Models for Perceptually Optimized Image Processing – A Review**. In *In procedings of the IEEE*, 2000.

[29] S. I. OLSEN. **Noise Variance Estimation in Images**. In *proceedings of 8th Scandinavian Conference on Image Analysis (SCIA-93)*, pages 25–28, 1993.

[30] T. N. PAPPAS AND R. J. SAFRANEK. **Perceptual Criteria for Image Quality Evaluation**. *Handbook of Image and Video Processing*, pages 669–684, 2000.

[31] T.W. PARKS AND K. HIRAKAWA. **Joint Demosaicing and Denoising**. In *proceedings of International Conference on Image Processing (ICIP-05)*, pages 309–312, 2005.

[32] PIETRO PERONA AND JITENDRA MALIK. **Scale-space and edge detection using anisotropic diffusion**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**:629–639, 1990.

[33] A. PIZURICA, R. PIZURICA, V. ZLOKOLICA, AND W. PHILIPS. **Combined Wavelet Domain and Temporal Video Denoising**. In *proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 334–341, 2003.

[34] JAVIER PORTILLA, VASILY STRELA, MARTIN J. WAINWRIGHT, AND EERO P. SIMONCELLI. **Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain**. *IEEE Transaction on Image Processing*, **12**:1338–1351, 2003.

[35] S. BATTIATO, G. PUGLISI, AND R. RIZZO. **Characterization Of Signal Pertubation Using Voting Based Curve Fitting For Multispectral Images**. In *proceedings of International Conference on Image Processing (ICIP-10)*, pages 545–548, 2010.

[36] S. SCHULTE, V. DE WITTE, AND E.E. KERRE. **A Fuzzy Noise Reduction Method for Color Images**. *IEEE Transactions on Image Processing*, **16**(5):1425–1436, 2007.

[37] L. SHAPIRO AND G. STOCKMAN. *Computer Vision.* Prentice-Hall, 2001.

[38] S. M. SMITH AND J. M. BRADY. **SUSAN - A New Approach to Low Level Image Processing**. *International Journal of Computer Vision*, **23**:45–78, 1995.

[39] C. TOMASI AND R. MANDUCHI. **Bilateral Filtering for Gray and Color Images**. In *proceedings of International Conference on Computer Vision (ICCV-98)*, pages 839–846, 1998.

[40] B. A. WANDELL. *Foundations of Vision.* Sinauer Associates, Inc., 1995.

[41] Z. WANG, A. C. BOVIK, AND L. LU. **Why Is Image Quality Assessment So Difficult?** In *proceedings of IEEE International Conference on Acoustic, Speech, and Signal Processing*, **4**, pages 3313–3316, 2002.

[42] ZHOU WANG, A. C. BOVIK, H. R. SHEIKH, AND E. P. SIMONCELLI. **Image quality assessment: from error visibility to structural similarity**. *IEEE Transactions on Image Processing*, **13**(4):600–612, 2004.

[43] X.Y. XU, E.L. MILLER, D.B. CHEN, AND M. SARHADI. **Adaptive two-pass rank order filter to remove impulse noise in highly corrupted images**. *IEEE Transactions on Image Processing*, **13**(2):238–247, 2004.

[44] Y.-H. KIM AND J. LEE. **Image feature and noise detection based on statistical hypothesis tests and their applications in noise reduction**. *IEEE Transactions on Consumer Electronics*, **51**(4):1367–1378, 2005.

[45] M. ZHANG AND B. GUNTURK. **A new image denoising method based on the bilateral filter**. In *proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP-08)*, pages 929–932, 2008.

[46] X. H. ZHANG, W. S. LIN, AND P. XUE. **Improved estimation for just-noticeable visual distortion**. *IEEE Transaction on Signal Processing*, **85**(4):795–808, 2005.