

國立交通大學

電機學院 電機與控制學程

碩士論文

數位相機彩色濾鏡陣列影像雜訊消除方法

Color Filter Array Denoising Method for Digital Cameras



研究生：唐學用

指導教授：林昇甫 博士

中華民國九十七年一月

數位相機彩色濾鏡陣列影像雜訊消除方法

Color Filter Array Denoising Method for Digital Cameras

研究生：唐學用

Student : Hsueh-Yung Tang

指導教授：林昇甫 博士

Advisor : Dr. S. F. Lin



Submitted to College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electrical and Control Engineering

January 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年一月

數位相機彩色濾鏡陣列影像雜訊消除方法

學生：唐學用

指導教授：林昇甫 博士

國立交通大學 電機學院 電機與控制學程碩士班

摘 要

當今時下，數位相機已幾乎完全取代傳統底片相機，舊有的底片也有產品將之翻拍成數位照片，這個趨勢已如潮水般滾滾而來，不只是數位相機，舉凡能數位化的產品都逃不過數位化的命運，只因數位資訊便於處理、儲存以及傳遞。

數位相機中的影像處理是經過了很多道的程序而成，幾乎所有的程序都會增強雜訊的作用，所以降低雜訊的影響最好的辦法就是在任何影像處理的程序之前就先將影像來源本身具有的雜訊濾除掉，而影像雜訊濾除最困難的就是在濾除雜訊的同時也保存了影像的輪廓和高頻訊號。本篇論文提出了另一種消除影像雜訊的架構，由三個想法所組成，一個是在影像區塊中將正在處理的像素歸類給某一預設圖樣，既然可被歸類給某一圖樣，就可以對這圖樣中的元素平均以降低雜訊，另一個是獲取校正過後的相機雜訊特性，掃描出其雜訊特性並以標準差來表示，以便於用來判別正在處理的影像區塊是否可以判定為均勻影像，進而做出最強的濾除效果。最後一個是運用肉眼視覺系統不易察覺到那些藏在多紋路、高反差區域的影像雜訊，所以不去濾除那些不易被察覺的雜訊以保存更多的影像輪廓訊號。

Color Filter Array Denoising Method for Digital Cameras

Student : Hsueh-Yung Tang

Advisors : Dr. S. F. Lin

Degree Program of Electrical and Computer Engineering

National Chiao Tung University

ABSTRACT

Nowadays, traditional film camera has almost been replaced by digital camera in commercial market. This trend is not only on camera, but also on any product in which the signal can be digitalized, since digital information would be much convenience to be processed, stored, and transmitted.

Image processing in digital camera consists of many processes. Almost all of the processes would enhance noise added in images. The best way to make noise strength be minimized is to reduce noise in front of any image processing. The difficulty of image denoising is always to preserve edge information, and filters out noise in flat area simultaneously. In this paper, we have presented a denoising method which consists of three ideas. One is to filter noisy pixel based on nearest pattern to keep edge information, another one is to use camera noise characteristic to judge the uniformity of current processed area and the last one is to make use of the property of spatial masking to keep edge information again on the highly texture area.

誌 謝

這篇論文的完成要感謝的人很多，有些是實質在論文上的指導，有些則是精神上的體貼和關心。首先最要感謝的是幫助我很多的指導教授 林昇甫博士，感謝林教授的多方幫助，包含論文的研究、生活處事哲理上的一切耐心教導，讓我受用無窮，有了林教授的體諒和鼓勵，我們這些學生們才有辦法工作、家庭與學業兼顧，並完成此論文。也深深感受到口試委員廖德誠教授、潘晴財教授、張翔教授等口試委員的用心和一絲不苟的作論文態度，他們訂正了我許多地方，所提供的寶貴意見讓我受益良多。

感謝亞洲光學集團-世界宏景主管的體諒，同意讓我繼續進修，感謝同仁們分擔掉我的工作量，特別感謝同仁林璟憲修改了相機的程式，讓我可以運用真實相機方便的比較不同的濾雜訊方法。能進到數位相機這領域也要感謝前一家公司華晶科技，華晶科技提供的舞台讓我和數位相機結下不解之緣，也感謝學長彭永薰和呂龍騰，由於他們的引進才讓我可以進入這行。還有多位同仁彼此間多年下來建立出來的信任和情感，無形中也成為我的精神支柱，感謝他們。

感謝我的父母、大哥、大嫂及家人們的關心和幫忙，也感謝恩師王瑞材教授多年來一再的關心和噓寒問暖，每次接到王教授的電話都會有暖上心房的感覺。最後要感謝我的太太，五年前她幫我偷偷買了簡章，在不知情的情況下，幫我把大部分的報考資料準備好了，雖然一路上在工作、家庭和學業都要兼顧很辛苦，但她總會在需要時給我一整段的時間讓我好好準備課業，才有完成學業的可能！

要感謝的親朋好友很多，不及備載，儘致上最深的謝意，謝謝您們！

Contents

Abstract in Chinese	i
Abstract in English	ii
Acknowledgements in Chinese	iii
Contents	iv
List of Tables	v
List of Figures	vi
Chapter 1 Introduction	1
1.1 Motivation and Contribution	1
1.2 Outline	2
Chapter 2 Review of Related Works.....	3
2.1 Color Filter Array	3
2.2 Bilinear Method	5
2.3 Human Visual System	5
2.4 Bosco-and-Mancuso Filter for Image Denoising	6
2.5 Peak Signal to Noise Ratio	11
2.6 Structural Similarity	12
Chapter 3 Proposed Image Noise Reduction System	13
3.1 Overview of Image Processing on Digital Camera	13
3.2 Block Diagram of Proposed Denoising Method	17
3.3 Algorithm of Proposed Denoising Method.....	18
Chapter 4 Experiments	24
4.1 Test Results of the Images with Additive Noise:.....	24
4.2 Test Results of the CFA Raw Images:	53
Chapter 5 Conclusions	61
References	62

List of Tables

Table 1	Proposed SSIM and PSNR compare with other filters for the images with additive Gaussian random noise.....	25
Table 2	Proposed SSIM and PSNR compare with other filters for the images with additive Rayleigh random noise.....	26
Table 3	Proposed SSIM and PSNR compare with other filters for the images with additive Gamma random noise.....	27
Table 4	Proposed SSIM and PSNR compare with other filters for the images with additive Exponential random noise.....	28
Table 5	Proposed SSIM and PSNR compare with other filters for the images with additive Uniform random noise.....	29
Table 6	Proposed SSIM and PSNR compare with other filters for the images with additive Pepper-and-Salt random noise.....	30
Table 7	Test result of filtering image “4.1.08-jellybeansg” with additive $\sigma=7.1875$ and $\sigma=3.125$ Gaussian noise.....	41
Table 8	Use image “1.2.03-straw” as an example to explain the proposed filter has worst SSIM on texture images under high Gamma noise condition.....	41
Table 9	Proposed SSIM and PSNR compare with other filters for the pattern images with additive Gaussian noise.....	47
Table 10	Proposed SSIM and PSNR compare with other filters for the pattern images with additive Rayleigh noise.....	48
Table 11	Proposed SSIM and PSNR compare with other filters for the pattern images with additive Gamma noise.....	49
Table 12	Proposed SSIM and PSNR compare with other filters for the pattern images with additive Exponential noise.....	50
Table 13	Proposed SSIM and PSNR compare with other filters for the pattern images with additive Uniform noise.....	51
Table 14	Proposed SSIM and PSNR compare with other filters for the pattern images with additive Pepper-and-Salt noise.....	52

List of Figures

Fig. 1	Capture an image by three image sensors structure.....	4
Fig. 2	Capture an image by one image sensor structure.....	4
Fig. 3	Reproduced an unknown value $C(u + du, v + dv)$ from its neighboring pixels by using bilinear method.....	5
Fig. 4	Spatial Masking effect.....	6
Fig. 5	System block diagram of Bosco-and-Mancuso filter.....	7
Fig. 6	Two kinds of operating windows established from Bayer pattern.....	8
Fig. 7	A candidate of K_n curve.....	9
Fig. 8	K_n curve for G channel in Bosco and Mancuso's patents.....	10
Fig. 9	K_n curve for R/B channel in Bosco and Mancuso's patents.....	10
Fig. 10	Similarity derived from distance D_i of neighboring pixel to C_0	11
Fig. 11	Typical block diagram of image processing in digital camera.....	14
Fig. 12	Noise distribution illustration from real camera.....	15
Fig. 13	Block diagram of proposed denoising method.....	17
Fig. 14	Two kinds of operating windows.....	18
Fig. 15	Pre-set twelve patterns when current processed pixel is green.....	19
Fig. 16	Pre-set twelve patterns when current processed pixel is red or blue.....	21
Fig. 17	Curve fitting for noise characteristic in terms of standard deviation.....	22
Fig. 18	An example illustrated that noisy texture image has better SSIM and PSNR than filtered images.....	33
Fig. 19	Another example illustrated that noisy aerial image has better SSIM and PSNR than filtered images.....	34
Fig. 20	An example illustrated that bilinear filter has better SSIM than proposed filter.....	35
Fig. 21	Proposed filter has worse PSNR even if it has better SSIM than others.....	37
Fig. 22	Proposed filter has better filtering result under $\sigma=7.1875$ Gaussian noise.....	39
Fig. 23	Proposed filter has better filtering result under $\sigma=3.125$ Gaussian noise.	40
Fig. 24	$\sigma=18.75$ Gamma noise yielded too much edge information which is not able to be recovered by proposed filter.....	42
Fig. 25	Proposed filter is not able to filter Pepper-and-Salt noise.....	45
Fig. 26	4 kinds of step wedge test pattern in USC web site.....	46
Fig. 27	CFA image captured by K1003 under ISO 100 condition.....	54
Fig. 28	CFA image captured by K1003 under ISO 400 condition.....	56
Fig. 29	CFA image captured by K1003 under ISO 1600 condition.....	58
Fig. 30	Images with fine tuned parameter.....	60

Chapter 1

Introduction

Nowadays, traditional film camera has almost been replaced by digital camera in commercial market. This trend is not only on camera, but also on any product in which the signal can be digitalized, since digital information would be more convenient to be processed, stored, and transmitted. The forecast volume of digital camera will be more than 130 million in year 2008 by statistical investigation. In general, once the camera has a good image quality under high ISO condition, it will have a big sale, since the high ISO performance is one of the important terms to attract user to buy it. The phenomenon is more obvious for high-end model. Nonetheless, once users buy it and take picture with high ISO condition, usually a lot of random noise can be seen all over the photo even though the camera had been claimed as a high ISO camera.



1.1 Motivation and Contribution

The trend of image resolution of digital camera is increasing nonstop and whereas the camera size tends to slim-down, so that the area of CCD photodiode is getting smaller. In this case, the sensitivity of CCD is getting down and down in consequence. That is to say, low SNR CCD has been made use of producing high ISO camera. Nonetheless, users won't accept grainy image when taking a high ISO picture. To tackle this difficult problem, an advanced denoising method of digital image is required.

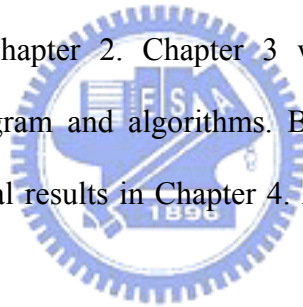
Bosco and Mancuso [1]-[3] provided an adaptive filtering for image denoising in front of image interpolation. Their paper and patents provided a good denoising method. They invented a *local feature detector* to compute texture degree of the area so that it is feasible to

determine the strength of the filter. If the area under processing is highly textured then the filtering strength has to be low, whereas the filtering strength is high when the area is almost uniform. However, we have found Bosco and Mancuso's method [1]-[3] didn't filter out noise as we expected when we have implemented their algorithm.

According to above situation we mentioned, it's valuable to develop a new denoising method which is to reduce more random noise and preserve more edge information of image simultaneously.

1.2 Outline

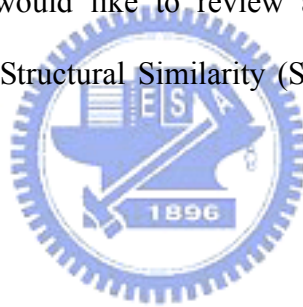
This thesis is structured as following: An overview of related works about random noise reduction would be given in Chapter 2. Chapter 3 would introduce proposed method represented in system block diagram and algorithms. Based on the proposed method, we would like to discuss experimental results in Chapter 4. At the end of this work, we draw a conclusion in Chapter 5.



Chapter 2

Review of Related Works

Image acquisition devices made up of many kinds of different ways. In this chapter, some specific prior works related to this proposed denoising method will be reviewed. In Section 2.1, we would like to explain why Color Filter Array (CFA) is necessary and what kind of CFA we have taken. In Section 2.2, two of HVS models which have been used in the proposed denoising method would be reviewed. Then, Two kinds of image denoising methods in spatial domain such as bilinear, texture detection based denoising algorithm would be brought up in Section 2.3 and 2.4 respectively since we used them to compare the performance. Finally, what we would like to review are the methods of image quality assessment. They are PSNR and Structural Similarity (SSIM) stated in Section 2.5 and 2.6 respectively.



2.1 Color Filter Array

In general, there are two types of images sensing structure for commercial digital camera. One is to use 3 or 4 image sensors to capture image of a scene as shown in Fig. 1. In order to make this structure work, optical paths, optical filters and image sensors for each color channel have to be separated to sense the image of a scene. Once image sensors got R, G and B color information, what we need to reproduce color is just picking up R, G and B color information for each corresponding coordinate without doing color interpolation. It works well! However, it's a very expensive approach, so that only professional digital cameras use this structure.

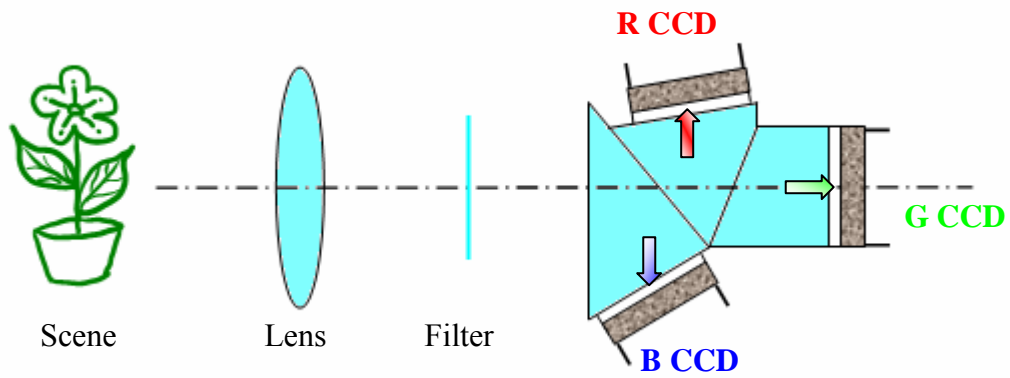


Fig. 1. Capture an image by three image sensors structure.

The other one, a single image sensor is used to capture image of a scene as shown in Fig. 2. The structure is much simple so as to reduce camera cost. In order to make this structure work, a color filter with mosaic pattern in front of image sensor to separate color information is necessary. So definitely, the resolution is reduced. To reproduce original color and resolution, interpolation is needed accordingly. At the right hand side of Fig. 2, each small square is representing a pixel. Basically, the pixels beneath color filters are light sensitive cells to respond the intensity of light falling on them. Color filter is aligned to sub-sample color information of a scene. There are many kinds of CFA. In digital camera, Bayer pattern CFA [4] with 3 primary colors R, G and B (as known as Bayer pattern) is widely used. As mention before, this thesis is going to discuss noise reduction method mainly based on this Bayer pattern domain.

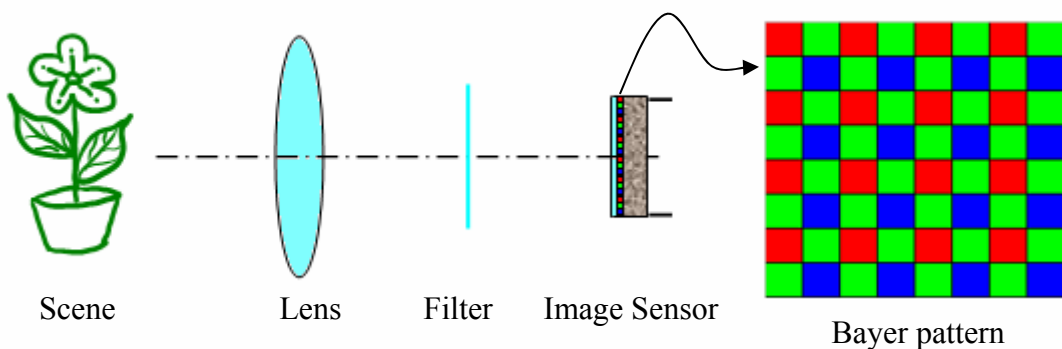


Fig. 2. Capture an image by one image sensor structure.

2.2 Bilinear Method

Bilinear is widely used in smoothing edge, typically in the application of re-scaling image size. Also, it can be used in denoising application. The method is described as following equation,

$$C(u + du, v + dv) = dudv * C(u + 1, v + 1) + du(1 - dv) * C(u + 1, v) + (1 - du)dv * C(u, v + 1) + (1 - du)(1 - dv) * C(u, v), \quad (1)$$

and illustrated as Fig. 3.

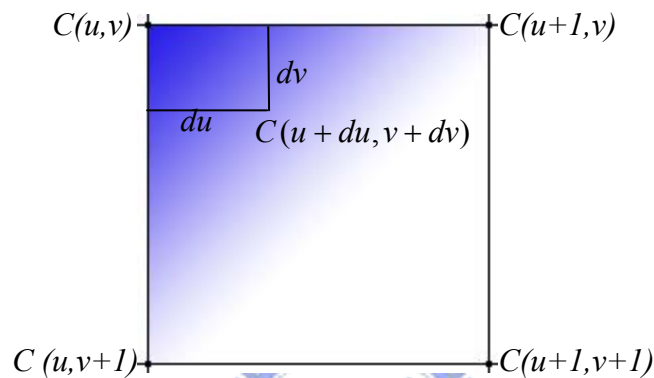


Fig. 3. Reproduced an unknown value $C(u + du, v + dv)$ from its neighboring pixels by using bilinear method.

2.3 Human Visual System

Human Visual System (HVS) is complex and not fully understood yet. It's difficult to use a mathematical function to represent it. It would be more realistic to get some useful information by experiments. There are two important observations on monochrome image which can be used for noise reduction. The first one is Just Noticeable Difference (JND) which is the minimum amount of stimulus intensity must be changed to cause a noticeable difference in sensory experience.

Ernst Weber (1795~1878), an experimental psychologist in 19th century, observed that the size of the JND threshold is related to initial stimulus magnitude. This relationship, had been simplified as Weber's Law by Gustav Theodor Fechner(1801~1887), can be expressed as

$\Delta I/I = k$. where ΔI represents JND threshold, I represents the initial stimulus intensity and k stands for the proportional constant. In other words, Weber's Law states that the size of the JND (i.e., ΔI) is a constant which is proportional to the original stimulus value.

The second one is spatial masking. Natural images contain large changes in luminance, and these changes suppress the ability of the eyes to detect distortions spatially adjacent to them, this is so-called spatial masking. As a result of masking, noises in images are less detectable along strong edges and in highly textured areas, than in smooth areas of the image as illustrated in Fig. 4.

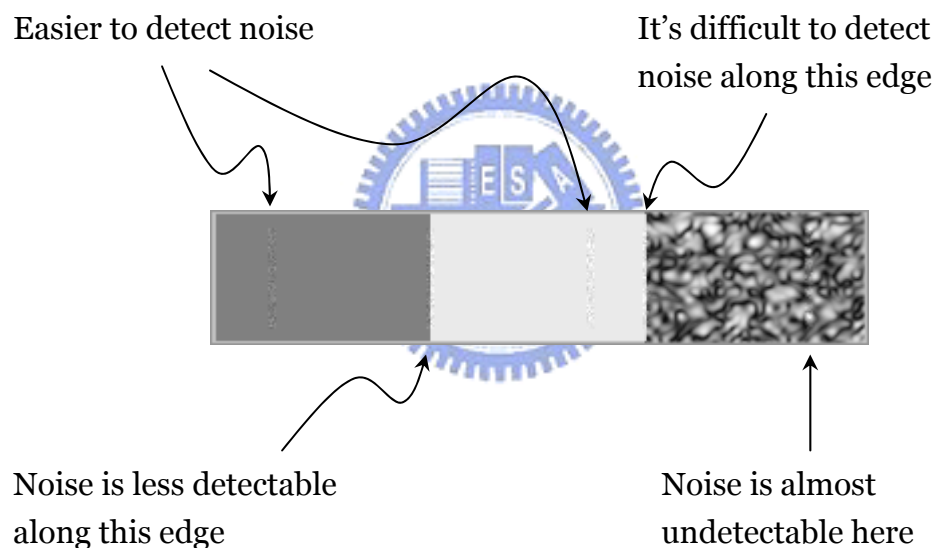


Fig. 4. Spatial Masking effect.

2.4 Bosco-and-Mancuso Filter for Image Denoising

Bosco and Mancuso [1]-[3] invented an adaptive image filter which is used to reduce the amount of noise in images captured by sensors in Bayer pattern format. The concept of this filter acts mainly on smoothing the high spatial frequency components which are hardly perceived by the HVS.

In Bosco and Mancuso's paper [1] and patent [2],[3], they made use of the Weber's Law to determine the JND, ΔI , which could be differentiated between intensity I and $I + \Delta I$. Bosco and Mancuso's HVS model assume that the uniform areas are the ones with details amplitude under JND. Having these considerations in mind they designed an algorithm that can distinguish if the current processed area is uniform area or not. Once the current processed area is not uniform, the algorithm will go to detect how textural the area is and adapts its filtering strength for the noisy pixel.

The system block diagram they designed is illustrated in Fig. 5. As mentioned above, the *local feature detector* is to compute texture degree of the area. The estimation is based on the information of distances, noise level, JND and exposure condition from distance and noise level estimator, HVS evaluator and exposure controller respectively. Once texture degree of this current processed area is decided, it would be able to determine the strength of the filtering strength.

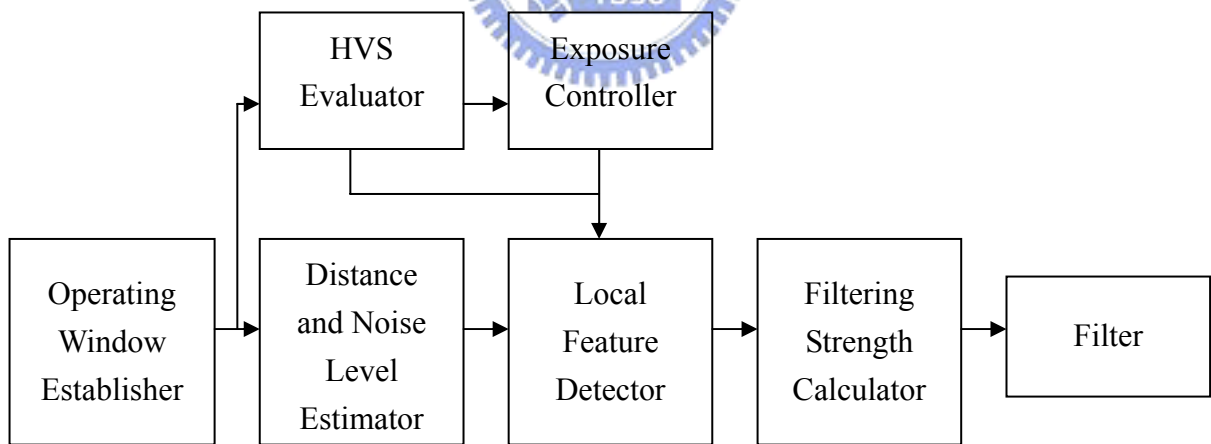


Fig. 5. System block diagram of Bosco-and-Mancuso filter.

The algorithm they proposed is to use two different filter masks, depending on which color is current processed pixel. One mask is for green pixels exclusively, the other one is for red/blue pixels, but not operated simultaneously. Fig. 6 is to illustrate those two kinds of operating windows established from Bayer pattern through 2 masks.

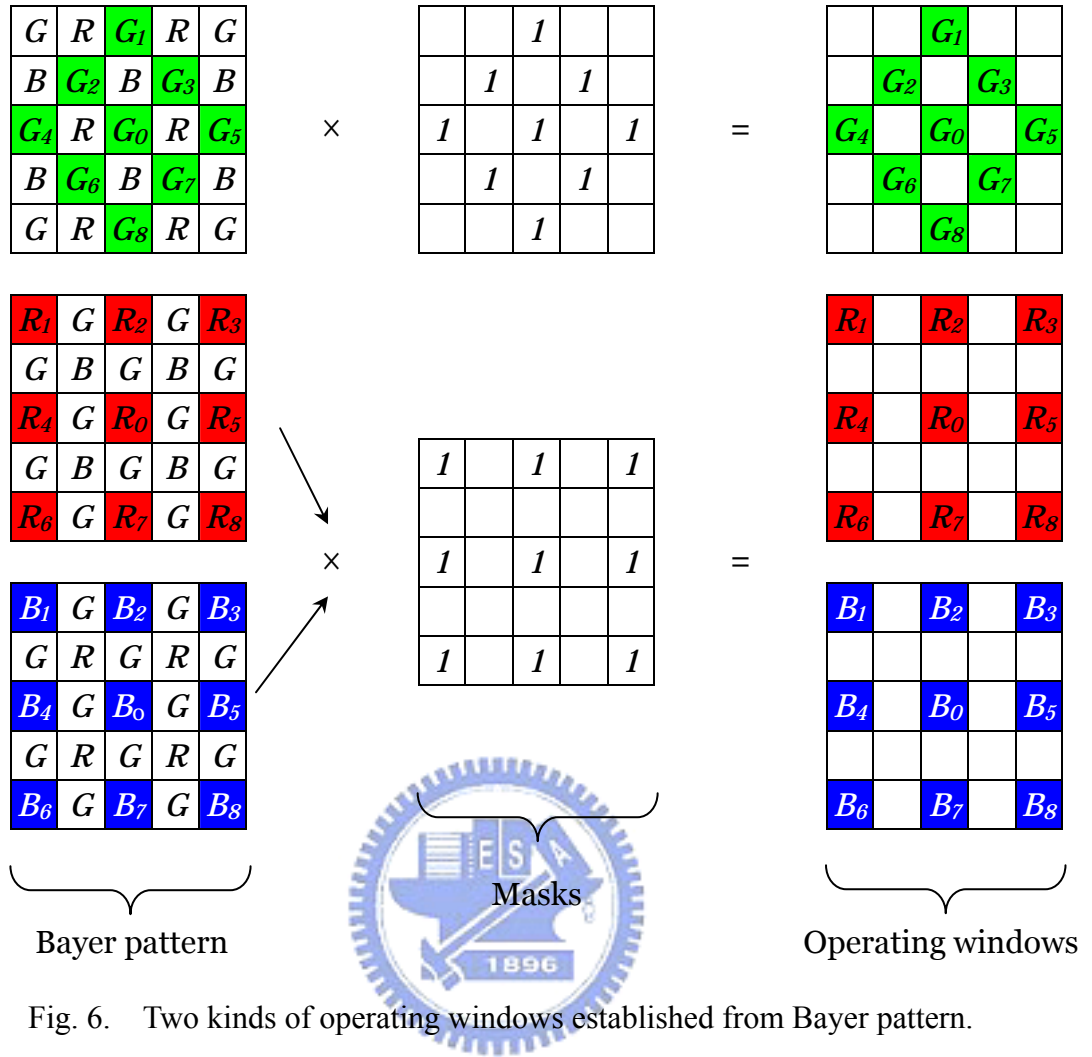


Fig. 6. Two kinds of operating windows established from Bayer pattern.

Definitely, green operating window is established when current pixel is green. Red and blue operating window will also be established once current pixel is red or blue respectively. The green operating window is different from red and blue, since the green channel has double information comparing with either red or blue channel in Bayer pattern format.

In each case of operating window for red, green and blue, let's define the current pixel C_0 and eight neighboring pixels C_1 to C_8 respectively. C_i will also represent for C_1 to C_8 to describe easier in many cases. So now we have symbol C_0 to C_8 or C_0 and C_i to stand for elements of operating window such as C_0 represent for G_0 when current color channel is green, C_1 represent for G_1 , etc., up to C_8 . For red and blue channel, definitely, C_0 to C_8 represent for R_0 to R_8 and B_0 to B_8 respectively.

C_0 is the current pixel with noise needed to be filtered. As described in Bosco and Mancuso's paper and patent, C_0 will be filtered and replaced with a weighted average of C_1 to C_8 . However, how much is the weight of C_1 to C_8 ? That will depend on how similar of them to C_0 . In the design, the higher similarity degree of neighboring pixels, the bigger weight of them to C_0 . The similarity degree is calculated based on the brightness of current processed pixel and takes into account of the predicted noise level NL of current area as following:

$$NL_{c_0}(t) = K_n * D_{\max} + [1 - K_n] * NL_{c_0}(t-1), \quad (2)$$

where, D_{\max} is the maximum distance derived from calculating each distance D_i of neighboring pixel C_i to C_0 and K_n is a parameter to determine the strength of filtering. In the definition, $K_n=1$ stands for almost flat area since this area could be filtered strongly, and $K_n=0$ stands for highest texture area as this area could not be filtered too much.

Recall the assumption of uniform area by using Weber's Law and refers to the K_n definition as above, we can assume the curve of K_n versus D_{\max} can be drawn as Fig. 7. That's to say, filter current pixel strongly if D_{\max} not greater than HVS threshold Th_{HVS} and filter current pixel lighter depending on how noisy the current area is. If D_{\max} is greater than $Th_{HVS} + NL$, then the area has to look as a highly texture area without strongly filter needed. However, they used Figs. 8 and 9 instead.

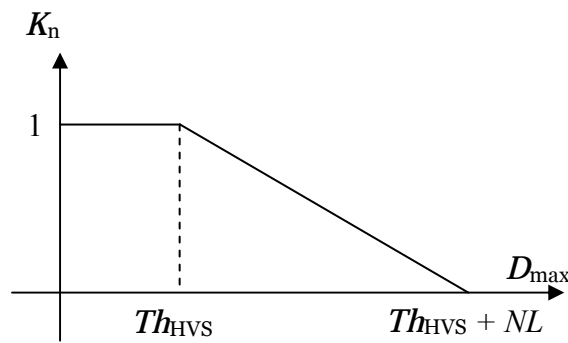


Fig. 7. A candidate of K_n curve.

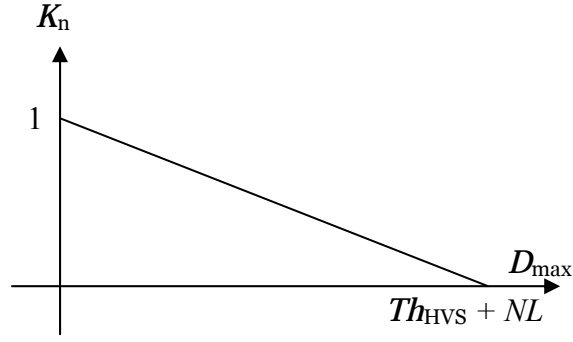


Fig. 8. K_n curve for G channel in Bosco and Mancuso's patents.

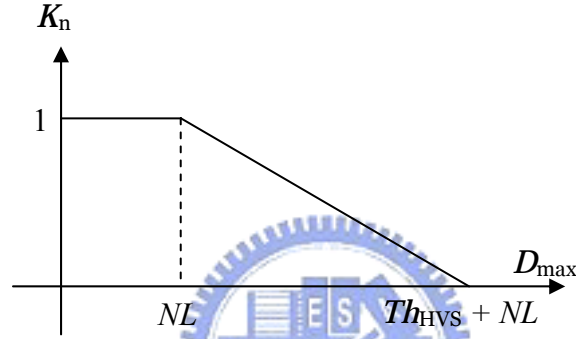


Fig. 9. K_n curve for R/B channel in Bosco and Mancuso's patents.

K_n is the overall filtering strength of current processed area. Although the filtering strength is given, we still need to determine the filtering strength of each neighboring pixel C_i to target pixel C_0 . Therefore, in order to evaluate similarity of each neighboring pixel C_i to the target pixel C_0 , two boundary thresholds refer to Th_1 and Th_2 are used to stand for most similar and least similar according to following equations:

$$Th_1 = K_n * D_{\max} + (1 - K_n) * D_{\min}, \quad (3)$$

$$Th_2 = K_n * D_{\max} + (1 - K_n) * \left(\frac{D_{\max} + D_{\min}}{2} \right), \quad (4)$$

where, D_{\min} is the minimum distance between C_0 to C_i . The value of similarity K_i can be determined by

$$K_i = \begin{cases} 1, & \text{if } D_i \leq Th_1, \\ 0, & \text{if } D_i \geq Th_2, \\ \frac{Th_2 - D_i}{Th_2 - Th_1}, & \text{for } Th_1 < D_i < Th_2, \end{cases} \quad (5)$$

and Fig. 10 is the illustration for it. At the end, the result of pixel out is expressed as

$$PixelOut = \frac{1}{8} \sum_1^8 [K_i * C_i + (1 - K_i) * C_0]. \quad (6)$$

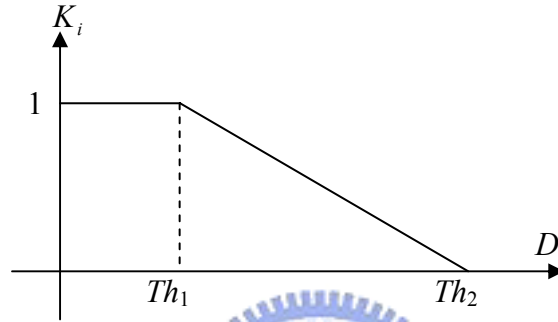


Fig. 10. Similarity derived from distance D_i of neighboring pixel to C_0 .

2.5 Peak Signal to Noise Ratio

Generally, Peak Signal to Noise Ratio (PSNR) is the most regular way used in the metric of distortion level. It is defined as a ratio between possible maximum power of image intensity and the power of distorted difference in terms of logarithmic decibel scale. In our application, given an original image O_{M*N} and processed image P_{M*N} with dimension of $M*N$, the PSNR is defined as

$$PSNR = 10 \log_{10} \left(\frac{I_{\max}^2}{MSE} \right), \quad (7)$$

where, I_{\max} represents for the possible maximum power of image intensity and MSE is defined as

$$MSE = \frac{\sum_{n=1}^N \sum_{m=1}^M [P(m,n) - O(m,n)]^2}{M * N}. \quad (8)$$

2.6 Structural Similarity

In many cases, the objective metric of PSNR and MSE could not correlate well with subjective perception. Therefore, Wang [5] developed Structural Similarity (SSIM) which is based on an assumption that HVS is highly sensitive to structural information of a scene, and is defined as

$$SSIM(O, P) = [l(O, P)]^\alpha \cdot [c(O, P)]^\beta \cdot [s(O, P)]^\gamma, \quad (9)$$

where α , β and γ are the parameters to define the relative importance of the three components.

The equation $l(O, P)$, $c(O, P)$ and $s(O, P)$ are defined as

$$l(O, P) = \frac{2\mu_O\mu_P + k_1}{\mu_O^2 + \mu_P^2 + k_1}, \quad (10)$$

$$c(O, P) = \frac{2\sigma_O\sigma_P + k_2}{\sigma_O^2 + \sigma_P^2 + k_2}, \quad (11)$$

$$s(O, P) = \frac{\sigma_{OP} + k_3}{\sigma_O\sigma_P + k_3}, \quad (12)$$

where k_1 , k_2 and k_3 are small constants.



Chapter 3

Proposed Image Noise Reduction System

The proposed denoising method would be introduced in this chapter. At the beginning, we would like to have brief description of the image processing on digital camera and the noise category we encountered in Section 3.1. Section 3.2 brings up system block diagram of proposed denoising method. At the end, Section 3.3 explains the detail algorithms.

3.1 Overview of Image Processing on Digital Camera

In general, the image processing flow of digital camera is shown as Fig. 11. Once user pressed the shutter button to take an image, the scene in front of camera would be captured by an image sensor, and converted to digital information by analog front end. Normally, black level compensation must be done in front of any processing to calibrate optical black. At the second, lens shading compensation sometimes could be applied. Meanwhile, statistic information for AE, AF and AWB is calculated and stored. And then, save the raw image to memory. What we are going to handle is this raw image. The format of input image is Bayer pattern, and output format of filtered image is Bayer pattern again without any formatting change.

Succeeding to the step of image denoising, color interpolation is normally applied to reproduce the missing color components for each pixel. Since every image sensor has a unique electrical response to light, color matching block is to compensate such kind of deviation. Then, gamma correction is to compensate the non-linearity of display device. YCC conversion block is to transform RGB color domain to YCbCr color domain. Usually, the Y channel will be used to enhance the edge of image. The final step is to perform JPEG compression to output JPEG image.

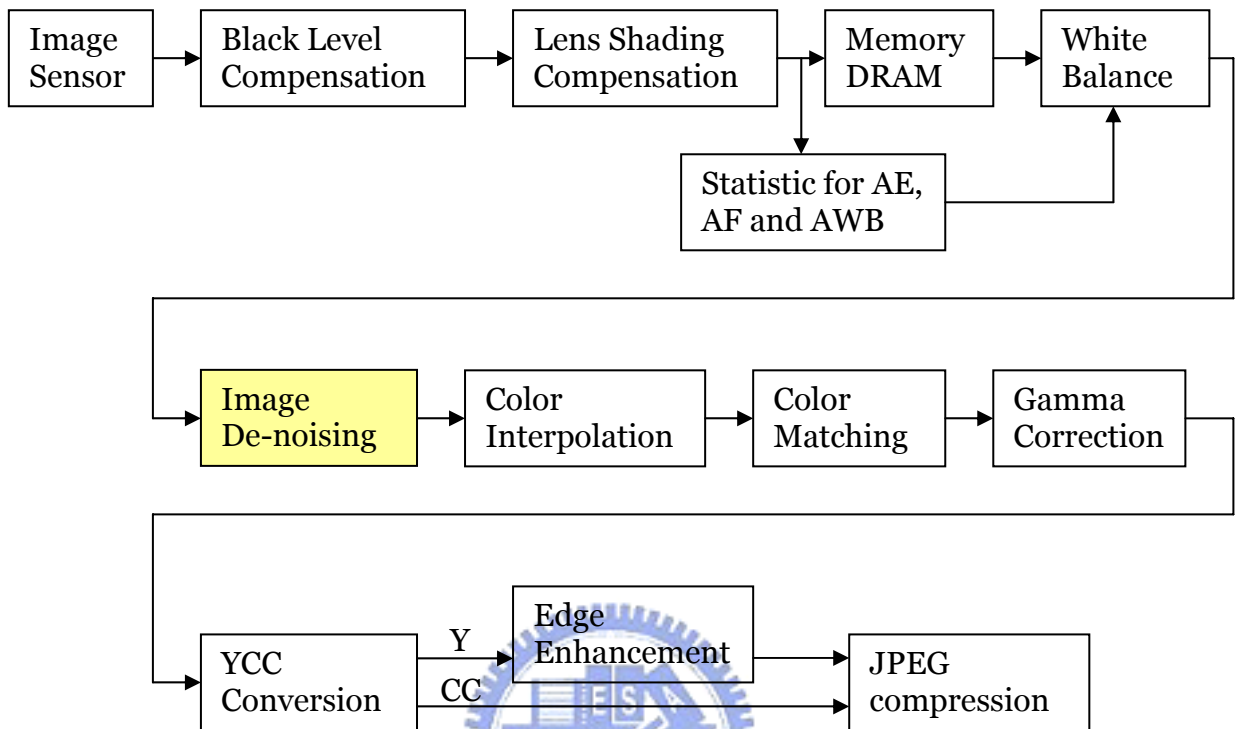


Fig. 11. Typical block diagram of image processing in digital camera.

As stated in Chapter 1 that we are interested in random noise. Fig. 12 is an example showing the noisy image captured by a digital camera. Gaussian noise distribution is shown in the histogram as Figs. 12 (b), (d), and (f).

Basically, not only random noise but also fix pattern noise in digital camera need to be handled. However fix pattern noise is much smaller than random noise if the camera made used of CCD sensor to capture image. Therefore, for the noise problem of digital camera, the most headache thing is dealing with Gaussian random noise.

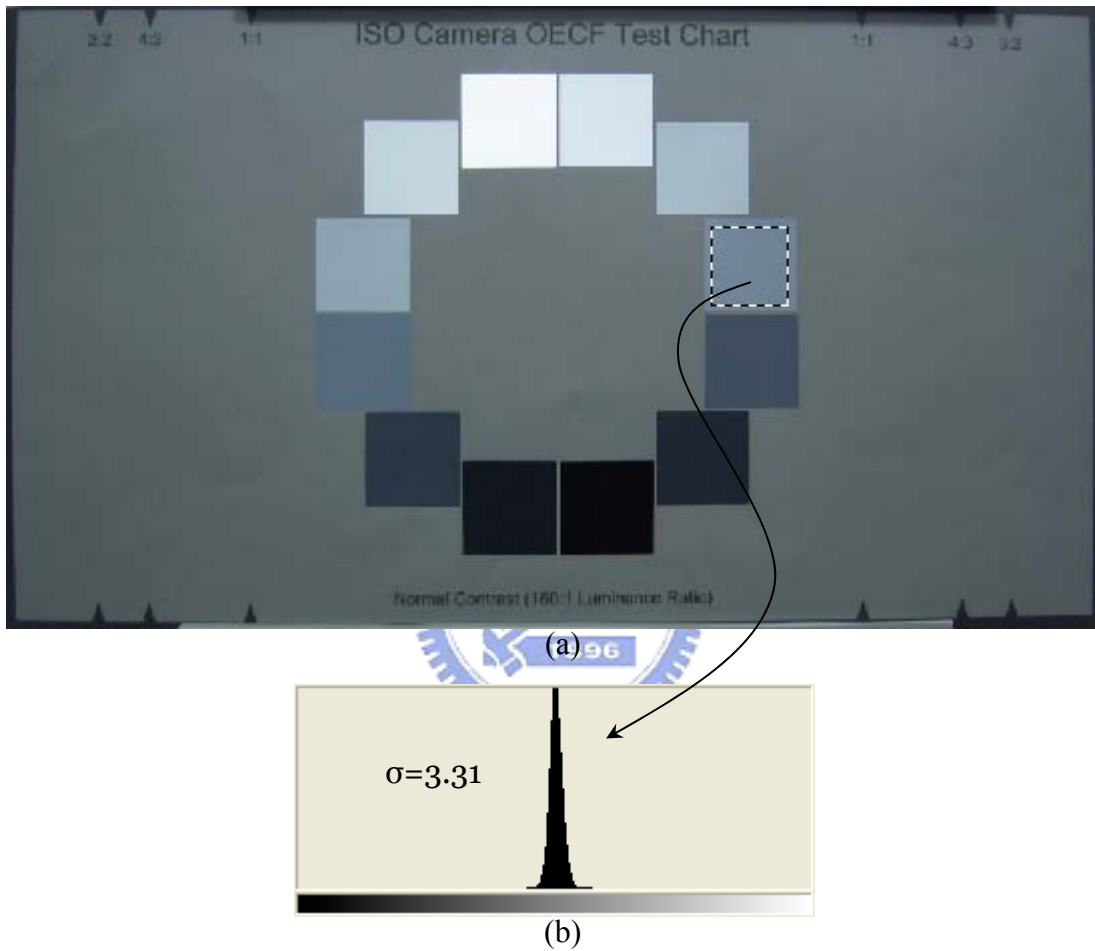
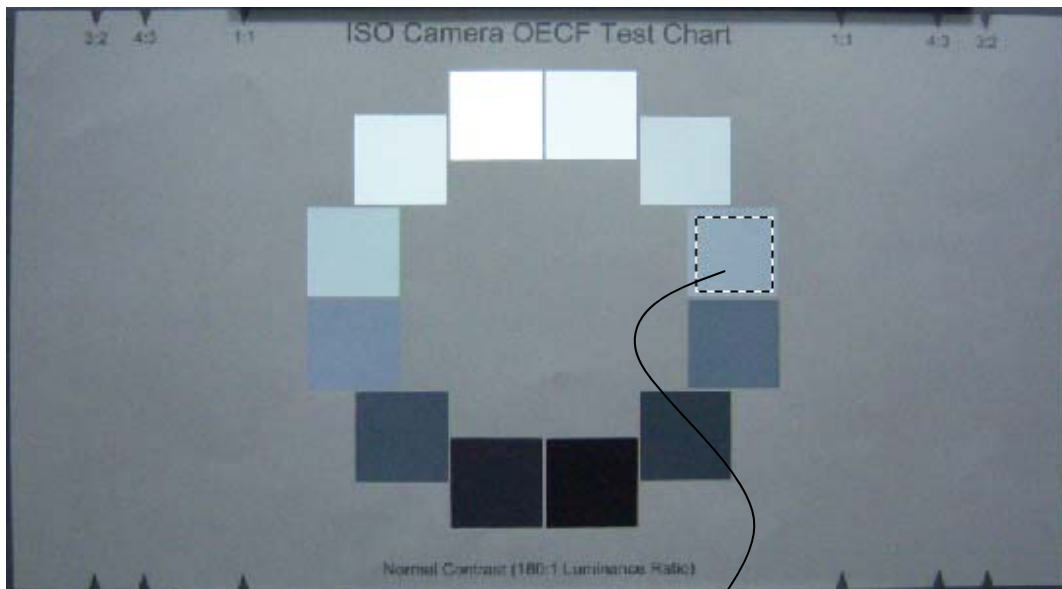
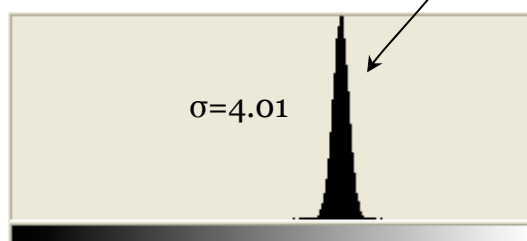


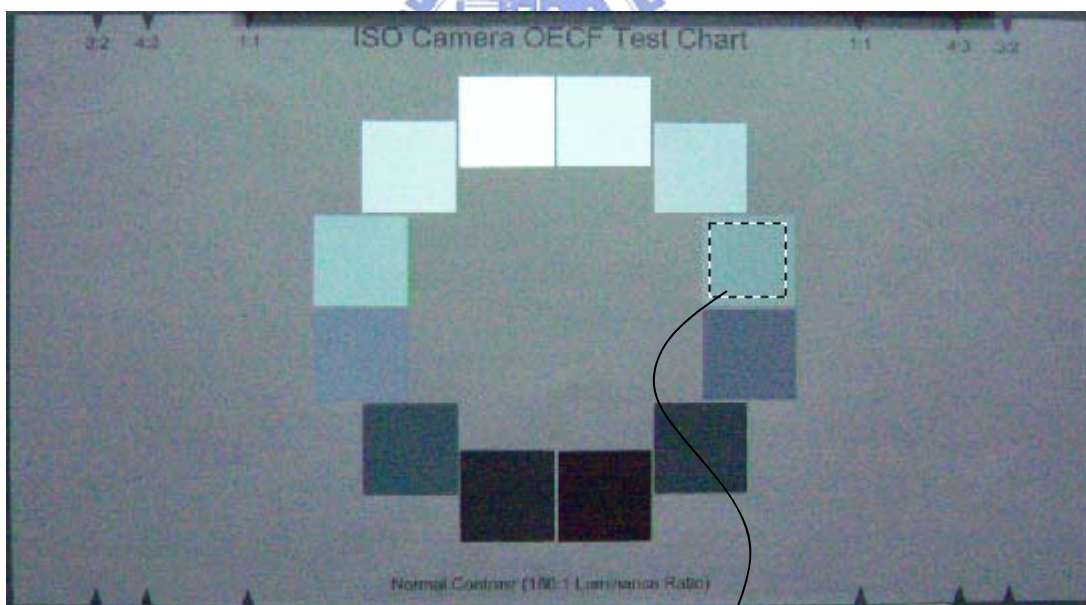
Fig. 12. Noise distribution illustration from real camera. (a), (b) Image “OECF” be taken under ISO 100 condition, and its noise distribution; (c), (d) Image “OECF” be taken under ISO 400 condition, and its noise distribution; (e), (f) Image “OECF” be taken under ISO 1600 condition, and its noise distribution.



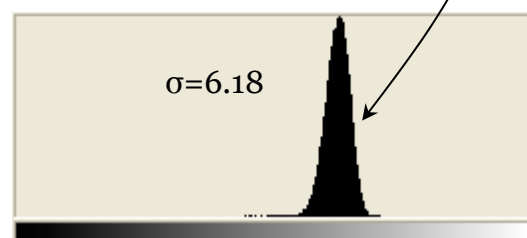
(c)



(d)



(e)



(f)

Fig.12. (Continued)

3.2 Block Diagram of Proposed Denoising Method

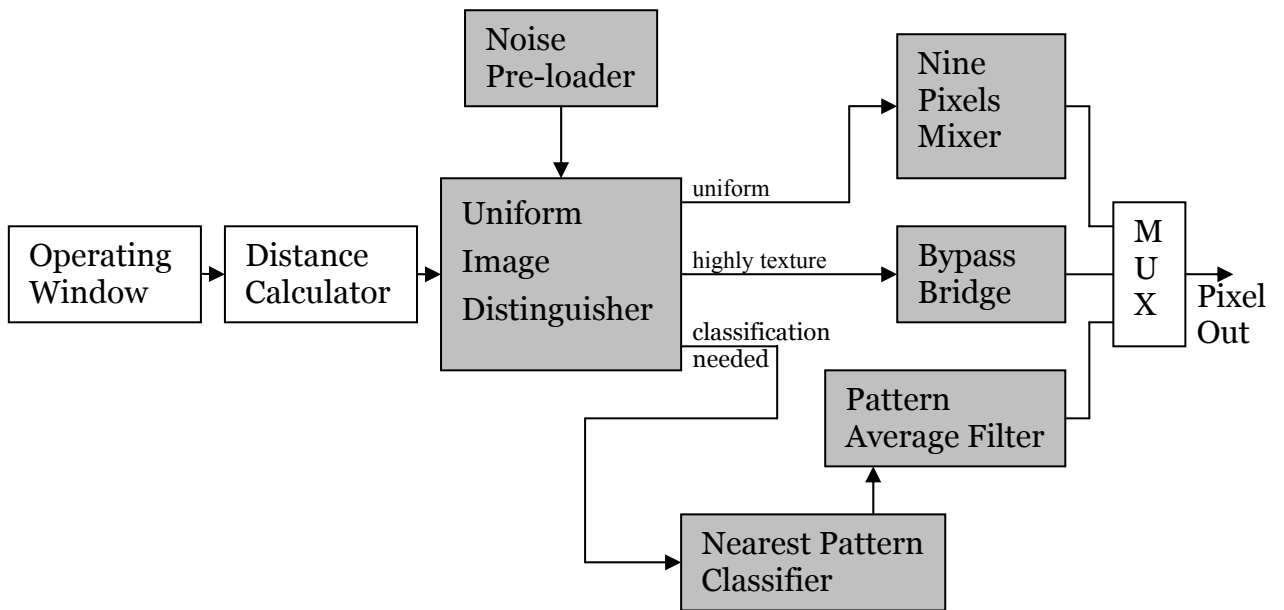


Fig. 13. Block diagram of proposed denoising method.

In the same way, the first step is to establish operating window. Since the 5x5 operating window as shown in Fig. 6 is very common, so that we just use it without change. Once the operating window is determined, a distance calculator is to calculate distance information based on operating window. By feeding the distance information to uniform image distinguisher, the current processed area can be judged as a uniform area, highly texture area or pattern classification needed area based on pre-loaded noise information. Once the processed area has been judged as a uniform area, the succeeding nine pixel mixer will be in charge to output final result and whereas the succeeding bypass bridge will be in charge to output final result while the processed area has been judged as a highly texture area without filtering needed. In the case of processed area is neither a uniform area nor a highly texture area, the succeeding nearest pattern classifier with pattern average filter will be used to output final result. Following Section 3.3 is going to explain the detail about how do they work.

3.3 Algorithm of Proposed Denoising Method

The same as Chapter 2, the elements $C_0 \sim C_8$ of operating window are generically represented for $G_0 \sim G_8$, $R_0 \sim R$, and $B_0 \sim B_8$ respectively depending on which color is the current pixel to be processed. Fig. 14 is to illustrate the two operating windows which are referred to $C_0 \sim C_8$.

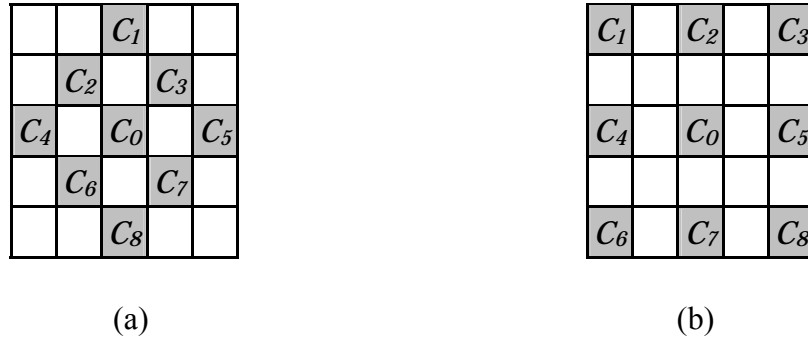


Fig. 14. Two kinds of operating windows. (a) Operating window when current processed color channel is green; (b) Operating window when current processed color channel is red or blue.

Once the operating window is established, the next step is to calculate distance. Distance calculator is going to calculate the distances D_i of each neighboring pixel to current pixel C_0 defined as

$$D_i = C_i - C_0, i = 1, 2, \dots, 8. \quad (13)$$

Also, D_{\max} and D_{\min} respectively represent for maximum and minimum distances which can be found once D_i derived.

There are twelve pre-set patterns as shown in Fig. 15. Eqs. (14)-(25) are the equations to calculate the distance of those twelve pre-set patterns to C_0 when current processed pixel is green.

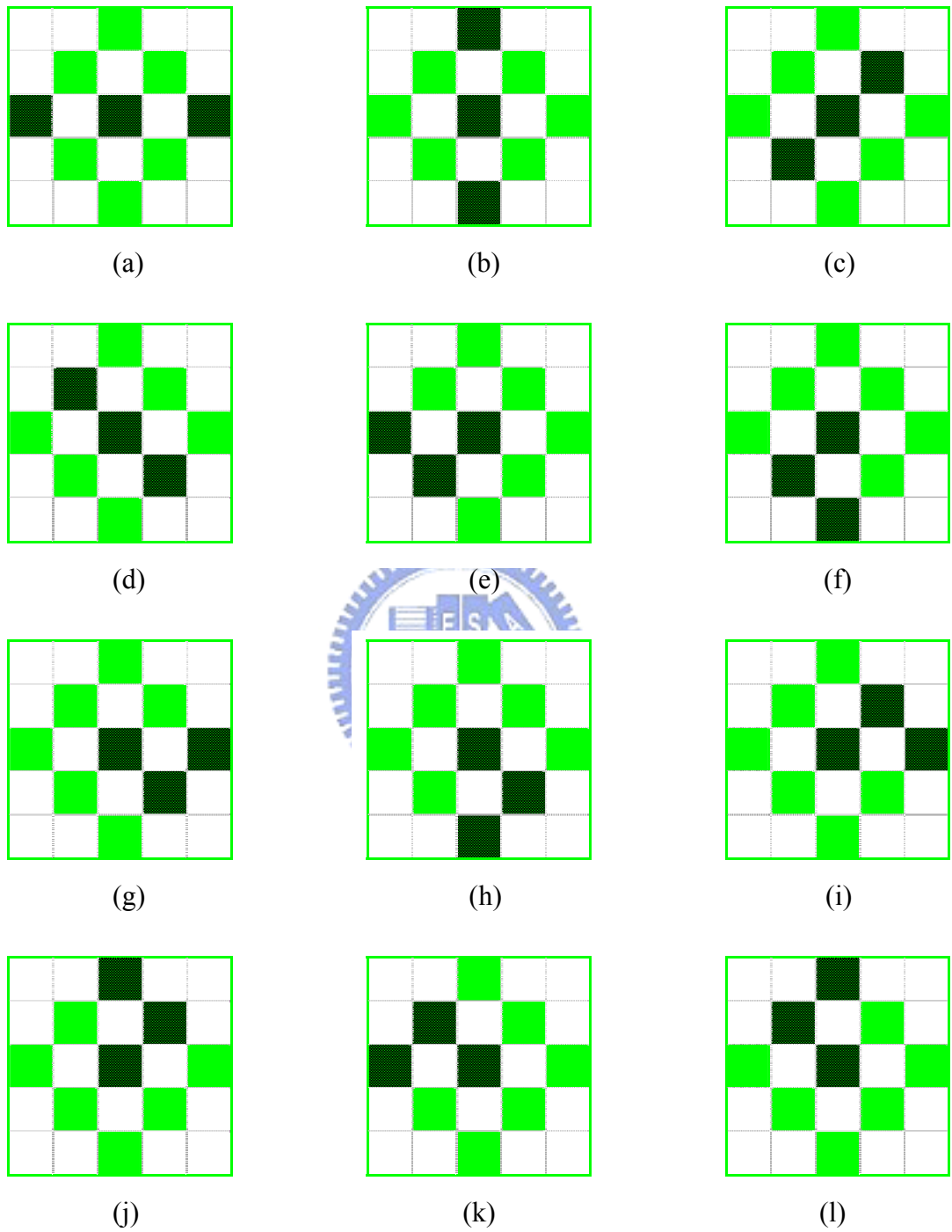


Fig. 15. Pre-set twelve patterns when current processed pixel is green. (a) Horizontal line; (b) Vertical line; (c) Rising line; (d) Falling line; (e) Left-Bottom corner; (f) Bottom-Left corner; (g) Right-Bottom corner; (h) Bottom-Right corner; (i) Right-Top corner; (j) Top-Right corner; (k) Left-Top corner; (l) Top-Left corner.

$$D_{HL} = D_4 + D_5 + |C_4 - C_5|, \quad (14)$$

$$D_{VL} = D_1 + D_8 + |C_1 - C_8|, \quad (15)$$

$$D_{FL} = D_2 + D_7 + |C_2 - C_7|, \quad (16)$$

$$D_{RL} = D_3 + D_6 + |C_3 - C_6|, \quad (17)$$

$$D_{LB} = D_4 + D_6 + |C_4 - C_6|, \quad (18)$$

$$D_{BL} = D_8 + D_6 + |C_8 - C_6|, \quad (19)$$

$$D_{BR} = D_8 + D_7 + |C_8 - C_7|, \quad (20)$$

$$D_{RB} = D_5 + D_7 + |C_5 - C_7|, \quad (21)$$

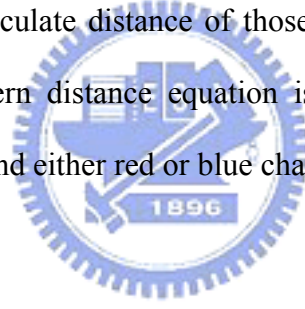
$$D_{RT} = D_5 + D_3 + |C_5 - C_3|, \quad (22)$$

$$D_{TR} = D_1 + D_3 + |C_1 - C_3|, \quad (23)$$

$$D_{TL} = D_1 + D_2 + |C_1 - C_2|, \quad (24)$$

$$D_{LT} = D_4 + D_2 + |C_4 - C_2|. \quad (25)$$

When current pixel is red or blue, the pre-set patterns are shown in Fig. 16 and Eqs. (26)-(37) are the equations to calculate distance of those twelve pre-set patterns to C_0 . The reason of having 2 sets of pattern distance equation is because the index of element is different between green channel and either red or blue channel.



$$D_{HL} = D_4 + D_5 + |C_4 - C_5|, \quad (26)$$

$$D_{VL} = D_2 + D_7 + |C_2 - C_7|, \quad (27)$$

$$D_{FL} = D_1 + D_8 + |C_1 - C_8|, \quad (28)$$

$$D_{RL} = D_3 + D_6 + |C_3 - C_6|, \quad (29)$$

$$D_{LB} = D_4 + D_6 + |C_4 - C_6|, \quad (30)$$

$$D_{BL} = D_7 + D_6 + |C_7 - C_6|, \quad (31)$$

$$D_{BR} = D_7 + D_8 + |C_7 - C_8|, \quad (32)$$

$$D_{RB} = D_5 + D_8 + |C_5 - C_8|, \quad (33)$$

$$D_{RT} = D_5 + D_3 + |C_5 - C_3|, \quad (34)$$

$$D_{TR} = D_2 + D_3 + |C_2 - C_3|, \quad (35)$$

$$D_{TL} = D_2 + D_1 + |C_2 - C_1|, \quad (36)$$

$$D_{LT} = D_4 + D_1 + |C_4 - C_1|. \quad (37)$$

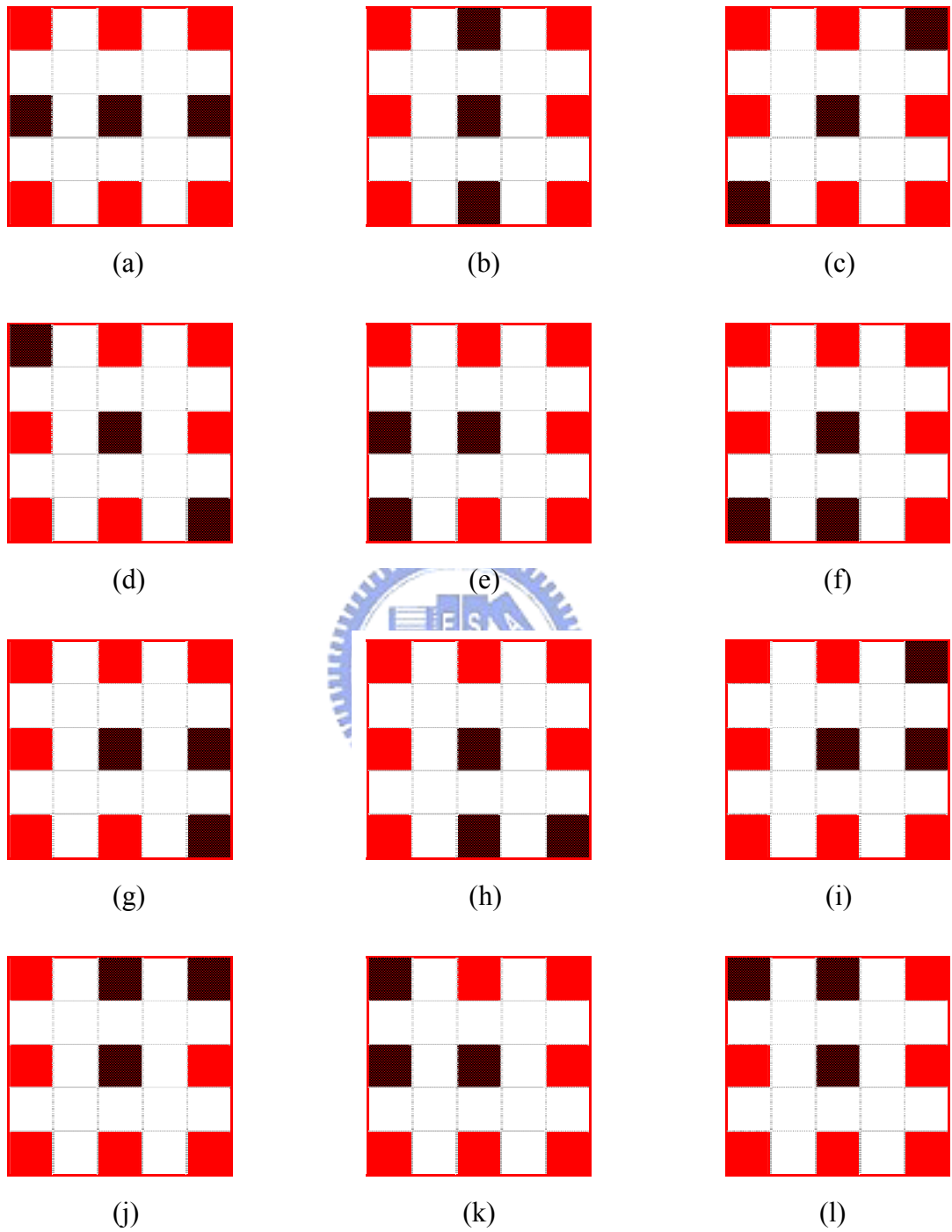


Fig. 16. Pre-set twelve patterns when current processed pixel is red or blue. (a) Horizontal line; (b) Vertical line; (c) Rising line; (d) Falling line; (e) Left-Bottom corner; (f) Bottom-Left corner; (g) Right-Bottom corner; (h) Bottom-Right corner; (i) Right-Top corner; (j) Top-Right corner; (k) Left-Top corner; (l) Top-Left corner.

In each case of pattern sets, once the distances of those patterns to C_0 are derived, $D_{\max\text{Pattern}}$ and $D_{\min\text{Pattern}}$ which represent for maximum and minimum distances of those patterns to C_0 can be found at the same time.

In addition to the distance information, we still need camera noise characteristic which can be obtained by doing noise scanning as a curve shown in Fig. 17.

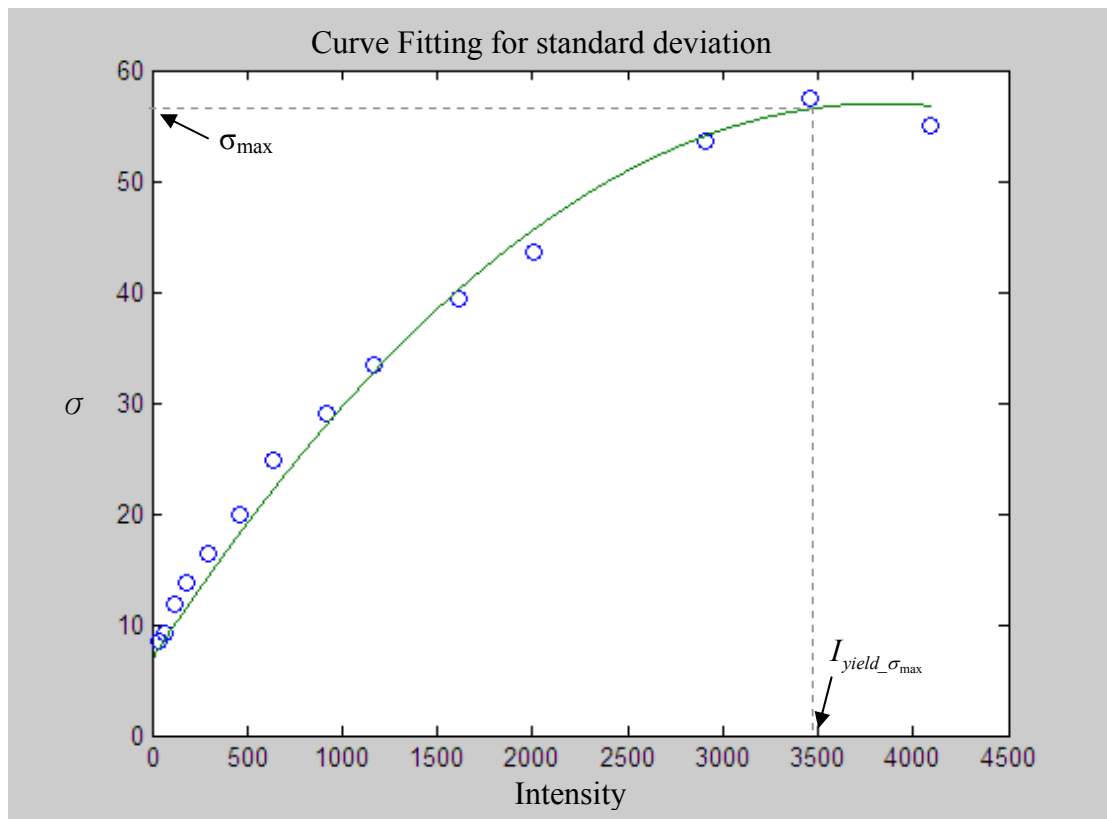


Fig. 17. Curve fitting for noise characteristic in terms of standard deviation.

Noise pre-loader is doing the job of loading pre-scanned noise level in terms of standard deviation σ which could be obtained by calculating on several flat areas from dark to bright as the small circles shown in Fig. 17. However we would like to use a curve fitted model which has the similar value to the reality noise instead of loading pre-scanned value. Curve fitting model expressed as

$$\sigma(C_0) = \left\{ 1 - \left[\frac{(C_0 - I_{yield_ \sigma_{max}})}{4096} \right]^2 \right\} * \sigma_{max}, \quad (38)$$

where, $I_{yield_ \sigma_{max}}$ is intensity of the flat area which has the maximum standard deviation there as shown in Fig. 17. In the sense, σ_{max} will be different under different ISO speed. Therefore, σ_{max} has to be found for each ISO speed from pre-scanned noise information.

Having both distance information and camera noise characteristic information, we now can distinguish if the current area is uniform or not by using multiple of σ to be the threshold as

$$Th_{uniform} = m * \sigma(C_0), \quad (39)$$

where, m is a different constant depend on color channel, ISO speed, and also how much captured images like to be filtered. Generally, if $D_{maxPattern}$ is less than $Th_{uniform}$, we would consider current area as a uniform image so that the result of output pixel is the average of C_0 to C_8 . On the other hand, if both $D_{maxPattern}$ and $D_{minPattern}$ are greater than $Th_{uniform}$, the current area must be a highly texture area without noise reduction needed since the spatial masking of HVS will work fine here. While the $Th_{uniform}$ is between $D_{maxPattern}$ and $D_{minPattern}$, the value of $D_{minPattern}$ will help us to find out which pattern is the nearest pattern. By averaging the element in nearest pattern, the end result of output pixel can be derived. Eq. (40) can help us understand criteria easier.

$$PixelOut = \begin{cases} \frac{1}{9} \cdot \sum_{n=1}^9 C_n, & \text{if } D_{maxPattern} \leq Th_{uniform}, \\ C_0, & \text{if } D_{maxPattern} \geq Th_{uniform} \text{ and } D_{minPattern} \geq Th_{uniform}, \\ \frac{1}{3} \cdot \sum_{n=1}^3 C_n, & \text{if } D_{maxPattern} \geq Th_{uniform} \text{ and } D_{minPattern} \leq Th_{uniform}. \end{cases} \quad (40)$$

Chapter 4

Experiments

In this chapter, we would like to discuss the experiment results. The proposed method is implemented in matlab language. There are 148 pieces of test images downloaded from USC, and have been classified to miscellaneous, texture and aerial images. Some of them are monochrome, and some are colored. Those color images have been separated into R, G and B raw information. Hence, 254 pieces of images we have tested in total. Also, the testing images have been classified to miscellaneous, texture and aerial images. Section 4.1 is going to discuss the test results of those images. As for Bayer pattern test images, they were captured by real camera, model: K1003, from market, and will be discussed in Section 4.2.

4.1 Test Results of the Images with Additive Noise:

Tables 1~6 show the noise filtering results for those images with additive Gaussian, Rayleigh, Gamma, Exponential, Uniform and Pepper-and-Salt noises. As shown in the tables, we use 3.125, 7.1875 and 18.75 instead of 5, 10 and 20 of the standard deviation to be the additive random noise. That is because the real camera K1003 we tested has a corresponding relation of ISO 100, 400 and 1600 induce standard deviation $\sigma=3.125$, 7.1875 and 18.75 Gaussian noise respectively. It will be more convenient to compare test results of downloaded images and Bayer pattern images by using the same standard deviation.

In order to compare the testing result easier, once the testing result is the maximum value in its own comparing block, the word will be shown in blue. For example, the proposed filter has the better SSIM and PSNR for the miscellaneous images with additive $\sigma=3.125$ Gaussian noise, so that 0.95522 and 38.8279 have been shown in blue in Table 1.

Table 1.

Proposed SSIM and PSNR compare with other filters for the images with additive Gaussian random noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Gaussian Noise ($\sigma=3.125$)			Adding Gaussian Noise ($\sigma=7.1875$)			Adding Gaussian Noise ($\sigma=18.75$)		
Method		SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
Average Performance of Misc images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9345215	2635.3187	38.078523	0.7602482	13063.999	31.117336	0.4132095	85827.13	22.940767
	Bilinear Filter	0.9458909	8751.3833	35.985395	0.8668262	12459.015	32.961046	0.598386	37903.299	26.78399
	BoscoMancuso	0.94303	3211.7889	37.512465	0.8396472	9438.6921	32.709541	0.7069787	25403.612	28.47603
	Proposed Filter	0.955219	2369.0048	38.82792	0.877256	7998.7579	33.55646	0.721585	26745.156	28.351483
Average Performance of Texture images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.977307	2692.0342	37.94795	0.9054758	13256.236	31.02415	0.68008	86799.959	22.86681
	Bilinear Filter	0.9511116	19645.974	32.04139	0.921281	23277.948	30.299642	0.789722	48656.289	25.76058
	BoscoMancuso	0.96658	7787.917	34.328642	0.9108355	20131.951	30.055265	0.7844042	65511.752	25.445342
	Proposed Filter	0.9710045	3577.3602	36.745723	0.9145903	14094.103	30.941432	0.7851884	67157.284	25.322661
Average Performance of Aerial images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.944267	2735.3849	37.87538	0.7844696	13493.13	30.944135	0.4271327	90031.875	22.701765
	Bilinear Filter	0.9299304	6294.7093	35.401446	0.859433	9859.6851	32.759052	0.6002676	35215.982	26.819875
	BoscoMancuso	0.9407586	4001.8892	36.596138	0.8356556	11066.248	32.068922	0.6667525	29015.028	28.068872
	Proposed Filter	0.9442624	3093.7263	37.452481	0.8583594	9603.6531	32.76298	0.669239	30138.787	28.11235
Average Performance for above 3 categories	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9520319	2687.5793	37.96728	0.8167312	13271.122	31.028542	0.5068074	87552.988	22.836447
	Bilinear Filter	0.942311	11564.022	34.476077	0.8825132	15198.883	32.00658	0.662792	40591.856	26.454814
	BoscoMancuso	0.9501229	5000.5317	36.145748	0.8620461	13545.63	31.611243	0.7193785	39976.797	27.33008
	Proposed Filter	0.956829	3013.3638	37.675376	0.883402	10565.505	32.42029	0.725338	41347.076	27.262164

Table 2.

Proposed SSIM and PSNR compare with other filters for the images with additive Rayleigh random noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Rayleigh Noise ($\sigma=3.125$)			Adding Rayleigh Noise ($\sigma=7.1875$)			Adding Rayleigh Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
Average Performance of Misc images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9524053	2108.0467	39.203924	0.8195329	10084.949	32.405071	0.5185017	64174.225	24.362159
	Bilinear Filter	0.9544726	8332.0962	36.322186	0.8977145	11236.724	33.56601	0.6872944	32053.961	27.608477
	BoscoMancuso	0.9553635	2893.3807	38.059051	0.8745135	8262.0186	33.37031	0.7635932	23782.496	28.78649
	Proposed Filter	0.967256	2026.9335	39.56338	0.914593	6627.2968	34.40967	0.776909	25925.48	28.528432
Average Performance of Texture images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.982769	2329.5273	38.61549	0.9254763	11027.233	31.86248	0.7323481	69047.683	23.899324
	Bilinear Filter	0.9558581	18219.152	32.364742	0.935453	19827.575	30.958405	0.83327	37271.335	26.81695
	BoscoMancuso	0.9713345	7188.3697	34.694235	0.9257176	17597.343	30.623742	0.8211497	53774.549	26.181915
	Proposed Filter	0.9752206	3273.8703	37.157302	0.9271713	12606.569	31.447376	0.8156782	62887.65	25.727853
Average Performance of Aerial images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.952311	2392.2183	38.49954	0.8129201	11556.016	31.661924	0.481988	75853.428	23.490104
	Bilinear Filter	0.934878	5985.4082	35.599208	0.876497	8877.6263	33.18623	0.6493042	30406.011	27.458247
	BoscoMancuso	0.9468747	3707.5422	36.90747	0.8533796	9989.9041	32.478467	0.7012338	26447.659	28.45958
	Proposed Filter	0.9496021	2841.6673	37.817415	0.873894	8619.5344	33.182683	0.702278	27782.268	28.452171
Average Performance for above 3 categories	Original Image	1	0		1	0		1	0	
	Noisy Image	0.962495	2276.5974	38.77298	0.8526431	10889.399	31.976493	0.5776126	69691.779	23.917196
	Bilinear Filter	0.9484029	10845.552	34.762045	0.9032214	13313.975	32.570213	0.7232894	33243.769	27.294557
	BoscoMancuso	0.9578576	4596.4309	36.553585	0.8845369	11949.755	32.157506	0.7619922	34668.235	27.80933
	Proposed Filter	0.964026	2714.157	38.179364	0.90522	9284.4666	33.01324	0.764955	38865.133	27.569485

Table 3.

Proposed SSIM and PSNR compare with other filters for the images with additive Gamma random noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Gamma Noise ($\sigma=3.125$)			Adding Gamma Noise ($\sigma=7.1875$)			Adding Gamma Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
Average Performance of Misc images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9468771	3923.0955	36.601812	0.8055776	18585.55	29.843401	0.5023506	108338.14	22.119836
	Bilinear Filter	0.9526165	9373.675	35.066067	0.8889569	17880.457	30.807949	0.6684268	71676.381	24.061314
	BoscoMancuso	0.9517971	4601.662	36.09254	0.8661901	15798.758	30.695568	0.7468097	62465.894	24.86763
	Proposed Filter	0.962456	3819.0176	36.86021	0.901636	15091.346	31.05836	0.750736	71338.921	24.371313
Average Performance of Texture images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.980834	4048.3634	36.45442	0.9202432	18567.251	29.779769	0.7246066	104345.14	22.168423
	Bilinear Filter	0.9587397	16716.207	32.455294	0.939066	19828.859	30.53282	0.834452	53357.365	25.34136
	BoscoMancuso	0.971281	7982.279	34.189189	0.9263285	21341.208	29.764013	0.8317934	67193.132	25.040787
	Proposed Filter	0.9739903	4980.7568	35.473742	0.9238756	20166.031	29.720223	0.8120095	99742.349	23.943844
Average Performance of Aerial images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.951178	3053.7849	37.50087	0.810479	14725.317	30.660279	0.4845545	94198.712	22.559213
	Bilinear Filter	0.937584	6051.2398	35.561203	0.880459	10539.637	32.63553	0.6566235	44080.417	26.146491
	BoscoMancuso	0.9476738	4107.58	36.572273	0.8571346	12033.662	31.883421	0.714269	38748.237	27.29767
	Proposed Filter	0.949818	3454.768	37.094764	0.8747839	11548.832	32.264883	0.7111025	42946.461	27.181801
Average Performance for above 3 categories	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9596297	3675.0813	36.85237	0.8454333	17292.706	30.094483	0.5705039	102294	22.282491
	Bilinear Filter	0.9496467	10713.707	34.360854	0.902827	16082.984	31.32543	0.7198342	56371.387	25.183055
	BoscoMancuso	0.9569173	5563.8403	35.618001	0.8832178	16391.209	30.781	0.764291	56135.754	25.73536
	Proposed Filter	0.962088	4084.8475	36.476239	0.9000984	15602.07	31.014488	0.7579495	71342.577	25.165652

Table 4.

Proposed SSIM and PSNR compare with other filters for the images with additive Exponential random noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Exponential Noise ($\sigma=3.125$)			Adding Exponential Noise ($\sigma=7.1875$)			Adding Exponential Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
Average Performance of Misc images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9538932	1916.0539	39.703041	0.8278349	9171.7644	32.892586	0.5404806	55881.464	25.003254
	Bilinear Filter	0.9542979	8318.3242	36.413495	0.8999395	10783.027	33.927708	0.7056303	27208.719	28.445277
	BoscoMancuso	0.9554814	2767.8087	38.328801	0.8711072	7869.327	33.648945	0.7638481	21312.695	29.324212
	Proposed Filter	0.964009	1934.0347	39.82872	0.903121	6360.8482	34.60461	0.78714	21304.67	29.45893
Average Performance of Texture images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.983237	2137.9798	39.01697	0.9282737	10101.145	32.27136	0.7469824	60810.245	24.473434
	Bilinear Filter	0.95507	18765.966	32.283742	0.934426	20676.762	30.896156	0.838341	36237.881	27.04295
	BoscoMancuso	0.971111	7228.5954	34.70439	0.9233719	17785.997	30.61	0.8138668	55504.168	26.107744
	Proposed Filter	0.9748681	3110.5723	37.401785	0.9235726	12031.845	31.592684	0.815686	58407.968	26.017606
Average Performance of Aerial images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.952762	2328.7978	38.64061	0.8182523	11233.554	31.806479	0.5019272	69021.956	23.901791
	Bilinear Filter	0.9340914	6044.9248	35.569447	0.876264	8876.268	33.19371	0.663491	27849.38	27.845443
	BoscoMancuso	0.9458779	3711.0621	36.897107	0.8462719	10219.802	32.352077	0.6965737	26525.128	28.454309
	Proposed Filter	0.9458406	2879.239	37.741935	0.8583505	9119.3632	32.819789	0.705502	26401.613	28.69633
Average Performance for above 3 categories	Original Image	1	0		1	0		1	0	
	Noisy Image	0.963298	2127.6105	39.12021	0.8581203	10168.821	32.323475	0.5964634	61904.555	24.459493
	Bilinear Filter	0.9478198	11043.072	34.755561	0.903543	13445.352	32.672525	0.7358207	30431.994	27.77789
	BoscoMancuso	0.9574901	4569.1554	36.643433	0.8802503	11958.375	32.203674	0.7580962	34447.33	27.962088
	Proposed Filter	0.9615725	2641.282	38.324145	0.8950146	9170.6856	33.0057	0.769443	35371.417	28.05762

Table 5.

Proposed SSIM and PSNR compare with other filters for the images with additive Uniform random noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Uniform Noise ($\sigma=3.125$)			Adding Uniform Noise ($\sigma=7.1875$)			Adding Uniform Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
Average Performance of Misc images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9466853	3924.5979	36.604222	0.8036968	18604.176	29.836767	0.4964588	109637.05	22.066244
	Bilinear Filter	0.9526122	9375.4442	35.066494	0.8884286	17887.981	30.80699	0.6631888	72237.356	24.021099
	BoscoMancuso	0.950285	4613.0272	36.065946	0.8594214	15968.515	30.635285	0.7373354	63346.948	24.77497
	Proposed Filter	0.963068	3811.8075	36.87933	0.90416	15010.803	31.07655	0.748584	71594.946	24.33555
Average Performance of Texture images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.980778	4049.9905	36.45284	0.9196255	18622.458	29.766729	0.7200411	105759.46	22.105561
	Bilinear Filter	0.9587221	16716.134	32.455076	0.938936	19847.905	30.52765	0.831692	53892.557	25.28707
	BoscoMancuso	0.9713056	7983.0211	34.189932	0.9267439	21345.365	29.766276	0.8291931	67717.684	24.973077
	Proposed Filter	0.9740785	4979.8103	35.475321	0.9247218	20171.318	29.727771	0.8110182	100198.21	23.91431
Average Performance of Aerial images	Original Image	1	0		1	0		1	0	
	Noisy Image	0.951126	3050.9628	37.5055	0.8089396	14727.829	30.659513	0.4789032	95197.107	22.513315
	Bilinear Filter	0.9375922	6051.0579	35.563036	0.880124	10542.482	32.63464	0.6525323	44444.054	26.105319
	BoscoMancuso	0.9474257	4111.8302	36.562772	0.8545456	12118.549	31.832587	0.7059513	39464.875	27.148317
	Proposed Filter	0.9504938	3440.4087	37.120414	0.8776947	11453.156	32.33459	0.709854	43117.236	27.15497
Average Performance for above 3 categories	Original Image	1	0		1	0		1	0	
	Noisy Image	0.9595298	3675.1837	36.85419	0.8440873	17318.155	30.08767	0.5651344	103531.2	22.228373
	Bilinear Filter	0.9496422	10714.212	34.361535	0.902496	16092.789	31.32309	0.7158043	56857.989	25.137828
	BoscoMancuso	0.9563388	5569.2928	35.606217	0.880237	16477.477	30.744716	0.757493	56843.169	25.63212
	Proposed Filter	0.962547	4077.3421	36.491687	0.9021921	15545.092	31.046305	0.7564855	71636.796	25.134943

Table 6.

Proposed SSIM and PSNR compare with other filters for the images with additive
Pepper-and-Salt random noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Pepper-and-Salt Noise ISO 100 and ISO 400 (didn't handle singular point)			Adding Pepper-and-Salt Noise ISO 1600 (handle singular point)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
Average Performance of Misc images	Original Image	1	0		1	0	
	Noisy Image	0.6280432	106620.26	22.010358	0.6280432	106620.26	22.010358
	Bilinear Filter	0.71665	44444.204	26.06897	0.7166504	44444.204	26.068965
	BoscoMancuso	0.7002945	59926.768	24.5426	0.7002945	59926.768	24.5426
	Proposed Filter	0.6122379	107459.05	21.974556	0.937776	15220.672	30.73973
Average Performance of Texture images	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0		1	0	
	Noisy Image	0.7812048	103674.45	22.1105	0.7812048	103674.45	22.1105
	Bilinear Filter	0.82964	54786.666	25.25988	0.8296402	54786.666	25.259881
	BoscoMancuso	0.8118458	70802.997	23.869474	0.8118458	70802.997	23.869474
Proposed Filter	0.7686513	105006.78	22.053915	0.948625	23049.538	29.30574	
Average Performance of Aerial images	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0		1	0	
	Noisy Image	0.6385427	98627.618	22.32257	0.6385427	98627.618	22.32257
	Bilinear Filter	0.71295	38241.843	26.46561	0.7129502	38241.843	26.46561
	BoscoMancuso	0.6858612	58501.829	24.594154	0.6858612	58501.829	24.594154
Proposed Filter	0.6100298	100111.22	22.256986	0.915285	13365.4	31.08356	
Average Performance for above 3 categories	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0		1	0	
	Noisy Image	0.6825969	102974.11	22.147809	0.6825969	102974.11	22.147809
	Bilinear Filter	0.75308	45824.237	25.93149	0.7530802	45824.237	25.931485
	BoscoMancuso	0.7326672	63077.198	24.335409	0.7326672	63077.198	24.335409
Proposed Filter	0.6636397	104192.35	22.095152	0.933895	17211.87	30.37634	

In general, the proposed filter has better SSIM and PSNR than Bosco-and-Mancuso filter as the test results shown in the row named “Average Performance for above 3 Categories” in Tables 1~6. However, there are some special cases such as noisy images have better SSIM and PSNR than filtered images as the results shown in the row named “Average Performance of Texture Images” and “Average Performance of Aerial Images” in Tables 1~5, bilinear filter has better SSIM than proposed filter as the results shown in the row named “Average Performance of Texture Images” and “Average Performance of Aerial Images” in Tables 1~5, proposed filter has worse PSNR even if it has better SSIM than others as the results shown in the row named “Average Performance of Misc Images” in Tables 1, 2, 3, and 5, proposed filter has worst SSIM on texture images under high Gamma and Uniform noise condition as the results shown in the row named “Average Performance of Texture Images” in Tables 3 and 5, proposed filter is not able to filter out Pepper-and-Salt noise under ISO 100 and 400 conditions as the results shown in the column named “Adding Pepper-and-Salt Noise for ISO 100 and ISO 400 (didn't handle singular point)” in Table 6, and proposed filter performs not badly in each kind of test patterns as shown in Tables 9~14. The phenomena mentioned as above have been brought up for case study as following:

Case 1: Noisy images have better SSIM and PSNR than filtered images:

The reason of noisy images have better SSIM and PSNR than filtered images is because original images themselves have had much random noise scattered on themselves already without additive noise. We have chosen a worst case for an example as shown in Fig. 18. The original image “1.3.08-Water” is shown in Fig. 18(a) and its 400% enlarged sub-image is shown in Fig. 18(b). By calculating a block in the sub-image, that $\sigma=7$ random noise at flat area has been found as shown in Fig. 18(b). No wonder that Fig. 18(c) which is the image “1.3.08-Water” with additive $\sigma=3.125$ Gaussian noise looks very similar to the original image by our eyes' perception. As for filtered images shown in Figs. 18(d)-(f), not only additive

noise have been filtered, but also a lot of original images have been handled as noisy signals, so that filtered images are not so similar as noisy images to the original images. In the category of aerial images, it has the same situation as texture images so that noisy images have better SSIM and PSNR than filtered images. Fig. 19 is another example which is an aerial image to explain this phenomenon, and the situation of this image is totally the same as Fig. 18. So it could be understood by consulting the above analysis for this example.

Case 2: Bilinear filter has better SSIM than proposed filter:

In our evaluation, the bilinear filter is implemented as

$$PixelOut = \frac{(C_2 + C_7 + C_4 + C_5) + C_0}{2}. \quad (41)$$

Therefore the filtering strength of bilinear filter is less than Bosco-and-Mancuso filter and proposed filter, when Bosco and Mancuso's algorithm and proposed algorithm handle the current image block as a uniform area.



The worst case happen on this case is still the image named “1.3.08-Water”. So we use 2nd worst case, image “1.5.06-BrickWall”, for an example as shown in Fig. 20. The original image is shown in Fig. 20(a) and its 400% enlarged sub-image is shown in Fig. 20(b). It's not difficult to observe that the original image is very noisy already. The standard deviation of each brick is around 6.5. Fig. 20(c) shows the image “1.5.06-BrickWall” with additional $\sigma=7.1875$ Gaussian noise. Also, the filtered images have been shown in Figs. 20(d)-(f). From subjective observation, we would say, the image filtered by bilinear filter as shown in Fig. 20(d) has the highest similarity to the original images, the same as objective metric. It's a reasonable result since original image is very noisy already, so that denoising filter with less filtering strength will output better SSIM images.

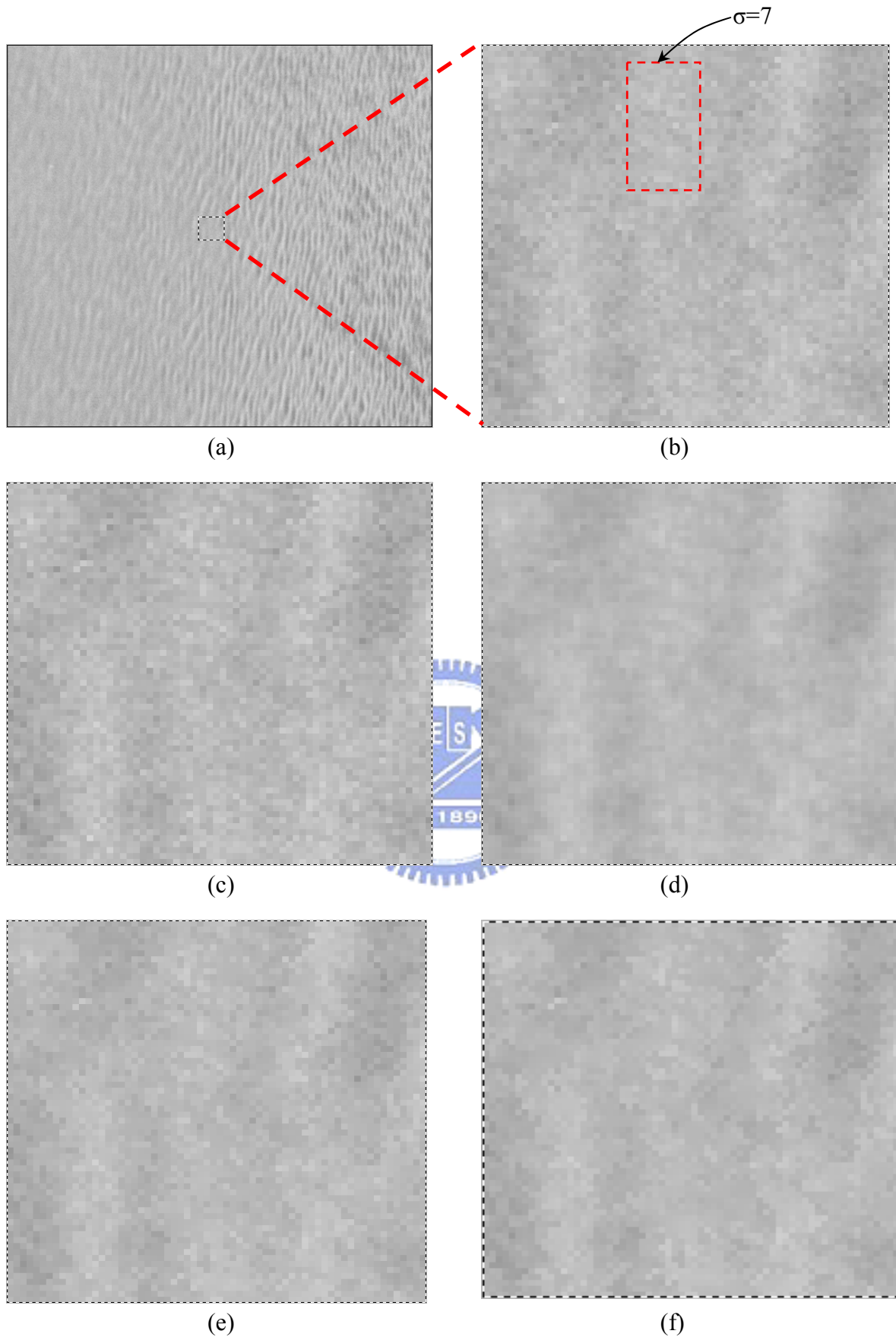


Fig. 18. An example illustrated that noisy texture image has better SSIM and PSNR than filtered images. (a) Original image “1.3.08-Water”; (b) 400% enlarged; (c) Adding $\sigma=3.125$ Gaussian noise; (d) Filtered by Bilinear filter; (e) Filtered by Bosco-and-Mancuso filter; (f) Filtered by Proposed filter.

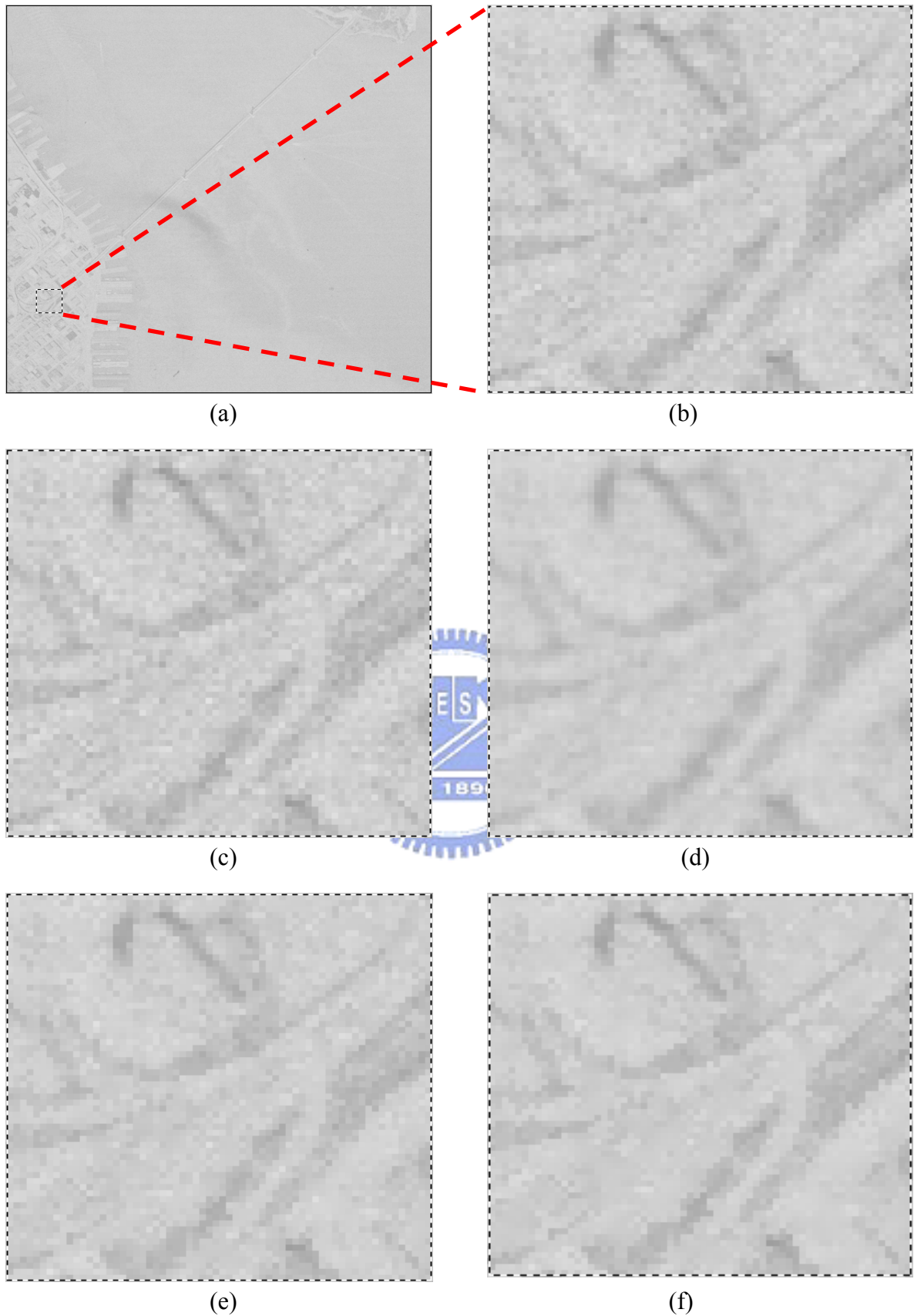


Fig. 19. Another example illustrated that noisy aerial image has better SSIM and PSNR than filtered images. (a) Original image “2.2.06-SanFran-cisco(BayBridge)B”; (b) 400% enlarged; (c) Adding $\sigma=3.125$ Gaussian noise; (d) Filtered by Bilinear filter; (e) Filtered by Bosco-and-Mancuso filter; (f) Filtered by proposed filter.

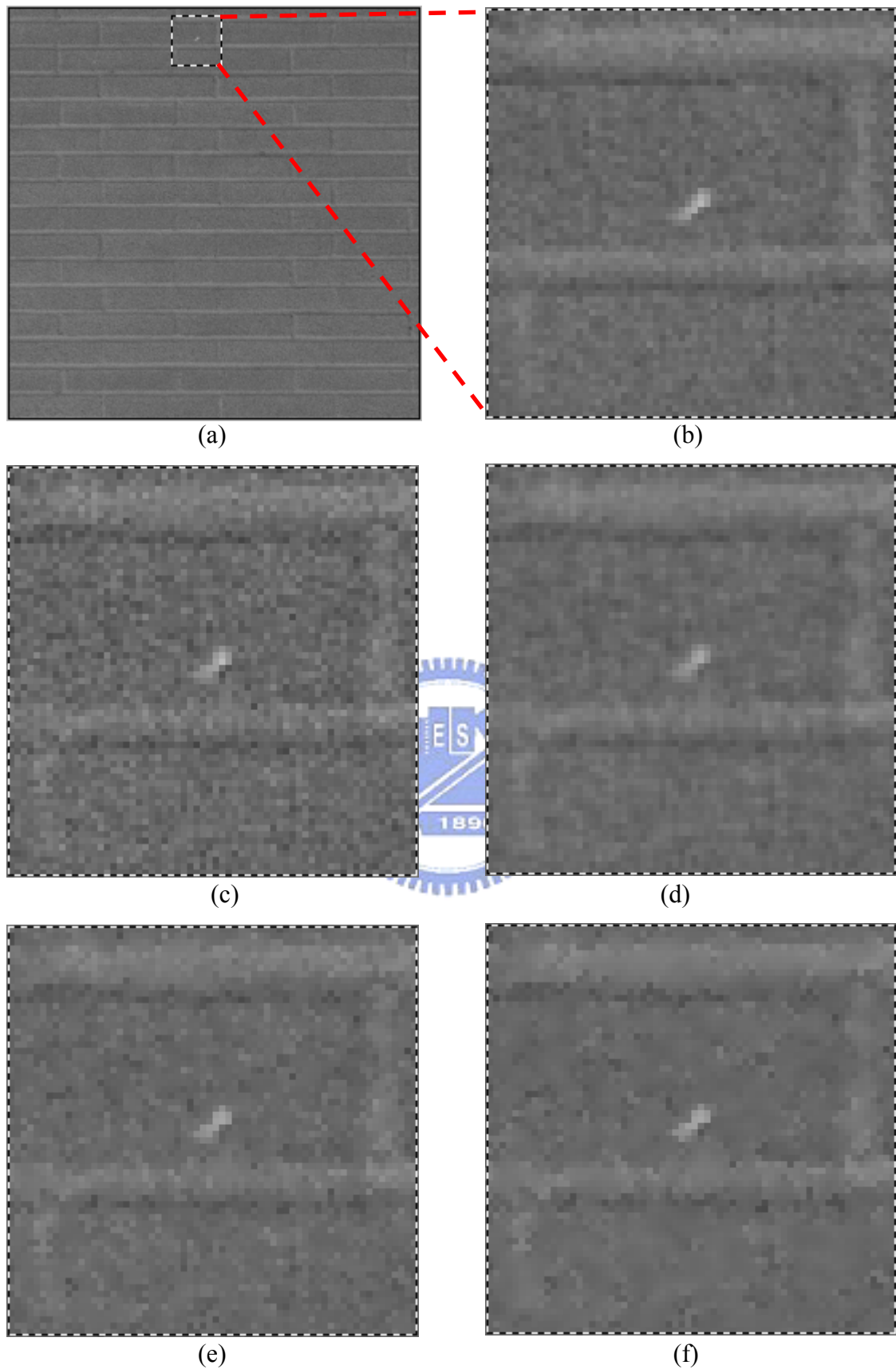


Fig. 20. An example illustrated that bilinear filter has better SSIM than proposed filter. (a) Original image “1.5.06-BrickWall”; (b) 400% enlarged; (c) Adding $\sigma=7.1875$ Gaussian noise; (d) Filtered by Bilinear filter; (e) Filtered by Bosco-and-Mancuso filter; (f) Filtered by Proposed filter.

Case 3: Proposed filter has worse PSNR even if it has better SSIM than others:

Basically, case 3 illustrates the reason why Wang developed SSIM for image assessment [5]-[7]. In some cases, images look very un-acceptable even if those images have the same PSNR as acceptable images [5]-[8]. So in general, case 3 is not a special case needed to bring up for a discussion, since proposed filter keep more structural information such as edge information.

However, when we surveyed the worst case of images, there is a drawback found in the proposed algorithm. That is proposed algorithm will yield contour seriously if the image has been added too much random noise such as $\sigma=18.75$ shown in Fig. 21. The original image “4.1.08-JellybeansG” is shown in Fig. 21(a) and its 400% enlarged sub-image is shown in Fig. 21(b). Besides, the image with additive $\sigma=18.75$ Gaussian noise is shown in Fig. 21(c) and its 400% enlarged sub-image is shown in Fig. 21(d). The reason to yield contour under big noise condition is because current processed pixel will handle the neighboring pixels as noisy pixel when pixel signal is under transient position either from bright to dark or dark to bright, so that current processed pixel will chose a nearest pattern to be its filtering elements instead of applying spatial masking on those transient pixels. By choosing nearest pattern to filter noisy pixel, the value of processed pixel will close to the nearest pattern. In the case of transient pixels changed slowly so that they consist of many pixels. Then the value of one side of them will join bright area and whereas the other side of them will join dark area. Therefore contour effect is enhanced as shown in Figs. 21(i)-(j). Comparing with Figs. 21(e)-(h) which are the images outputted from bilinear filter and Bosco-and-Mancuso filter respectively, they have no this kind of contour problem. This is the case that the proposed algorithm doesn't like to see, and unable to filter this kind of transient pixels well under big noise condition. Nevertheless, once the additive noise is not big enough to cause proposed algorithm handle neighboring pixels as noisy pixel in transient area, the proposed algorithm performs well as shown in Figs.

22, 23 and Table 7. Figs. 22(a) and 23(a) show the image “4.1.08-JellybeansG” with additive Gaussian noise and that Figs. 22(b) and 23(b) show their 400% enlarged sub-image. Both two sets of Figs. 22(c)-(e) and 23(c)-(e) show the images outputted from bilinear filter, Bosco-and-Mancuso filter, and proposed filter respectively. As subjective observation, bilinear filter looks too blue and whereas proposed filter has strongest edge information. Table 7 is also to illustrate that proposed filter has best filtering result on flat area.

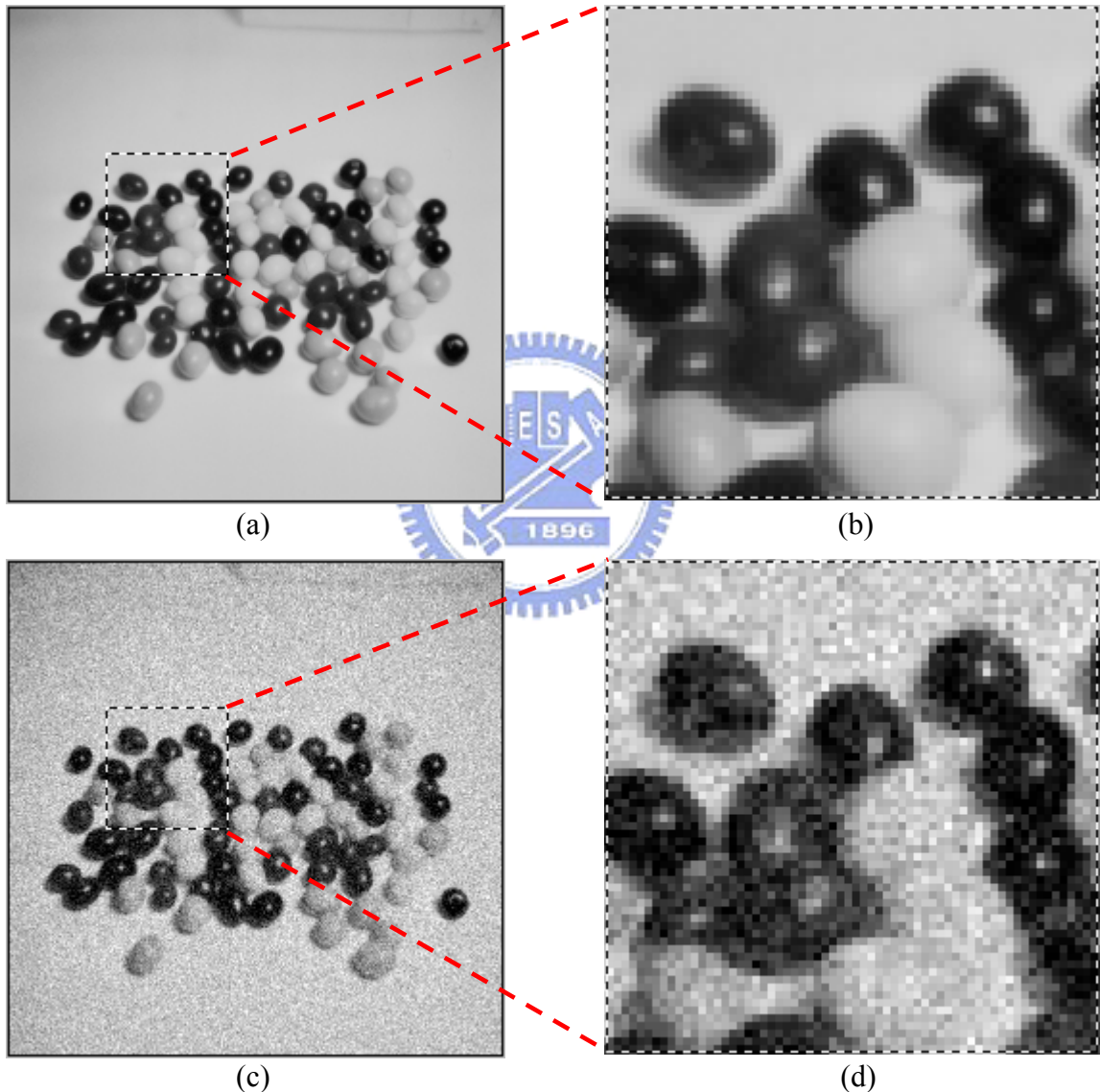


Fig. 21. Proposed filter has worse PSNR even if it has better SSIM than others. (a) Original image “4.1.08-JellybeansG”; (b) 400% enlarged; (c) adding $\sigma=18.75$ Gaussian noise; (d) 400% enlarged; (e) Filtered by Bilinear filter; (f) 400% enlarged; (g) Filtered by Bosco-and-Mancuso filter; (h) 400% enlarged; (i) Filtered by proposed filter; (j) 400% enlarged.

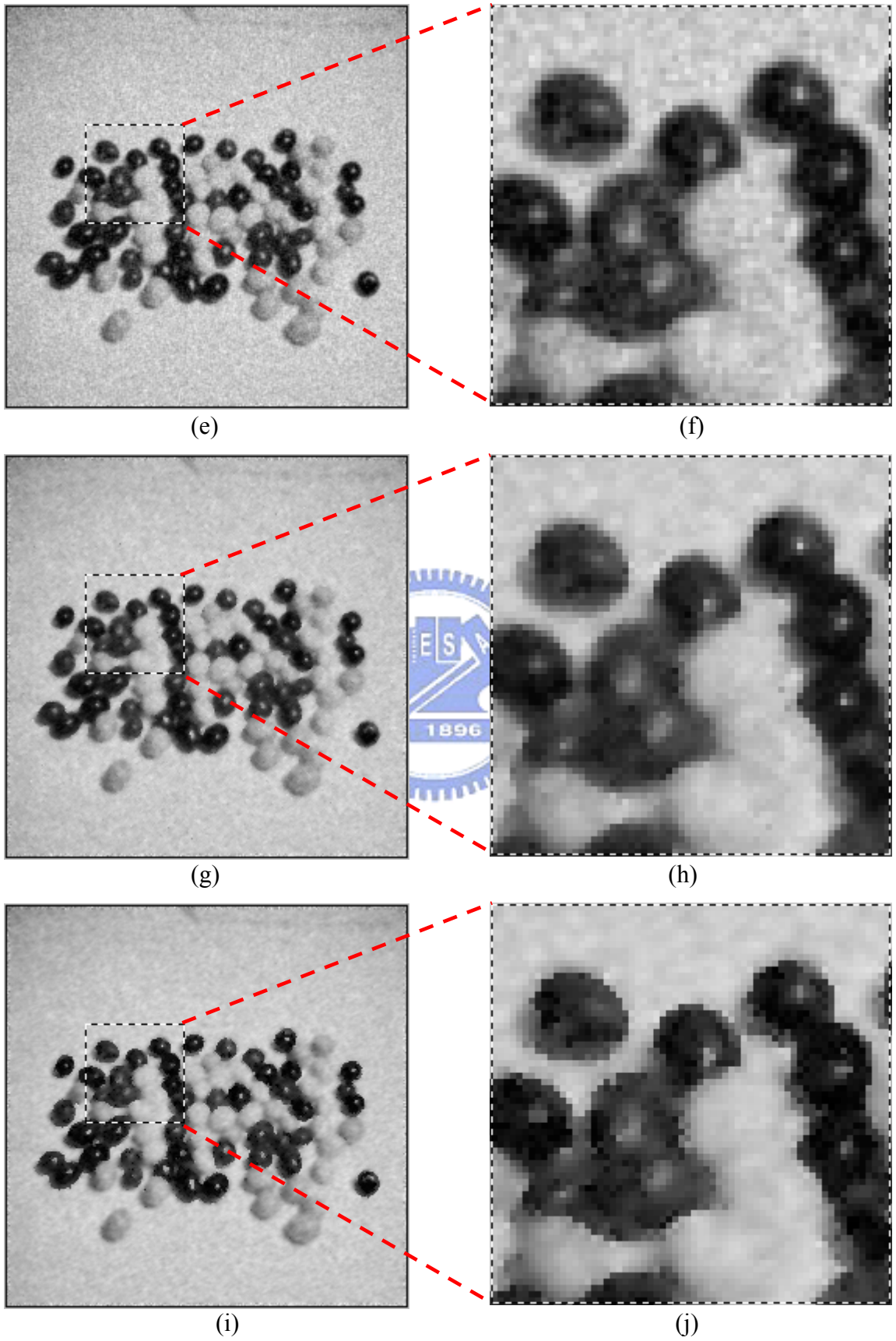


Fig. 21. (Continued)

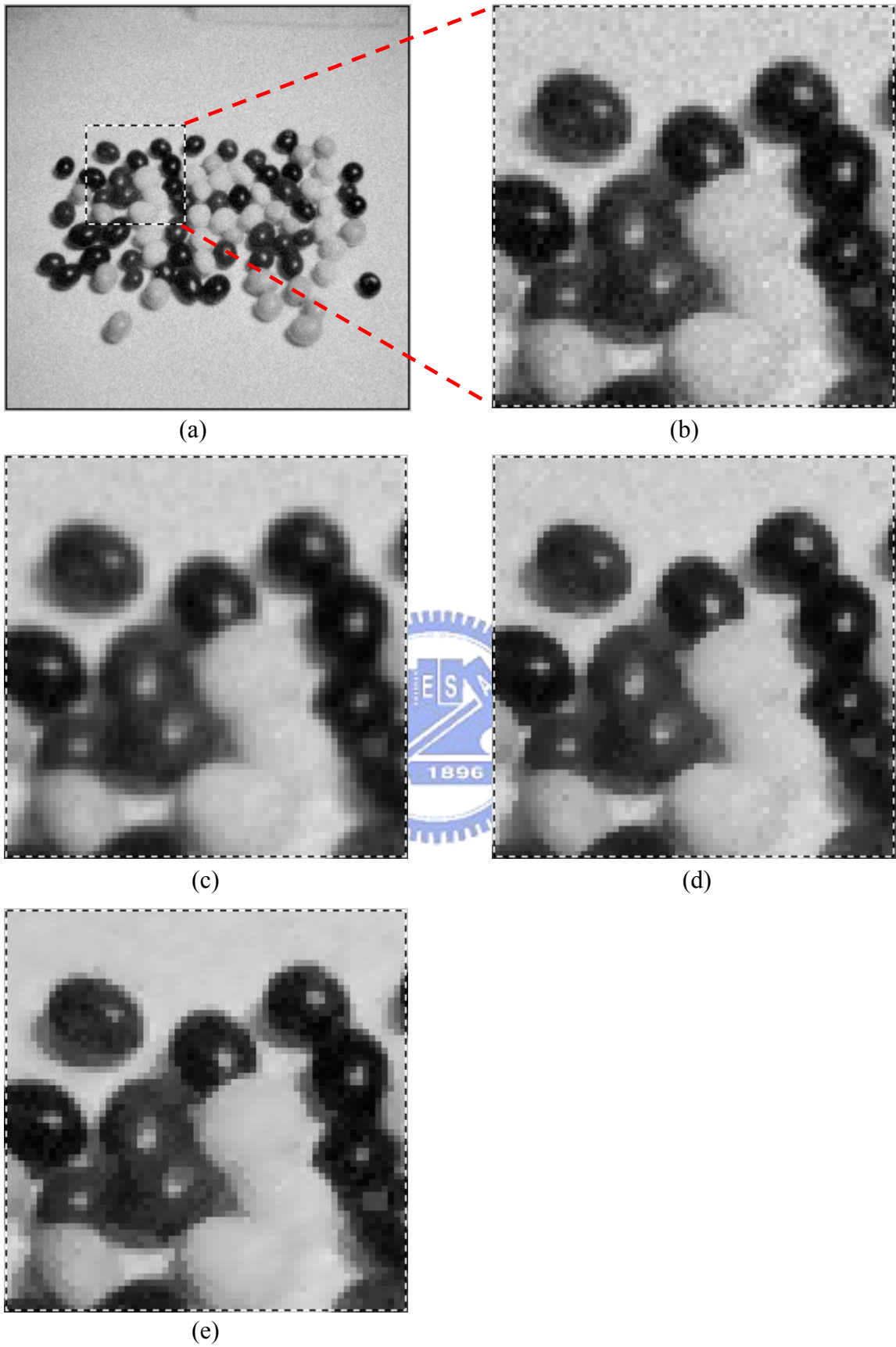


Fig. 22. Proposed filter has better filtering result under $\sigma=7.1875$ Gaussian noise. (a) Adding $\sigma=7.1875$ Gaussian noise; (b) 400% enlarged; (c) Filtered by Bilinear filter; (d) Filtered by Bosco-and-Mancuso filter; (e) Filtered by proposed filter.

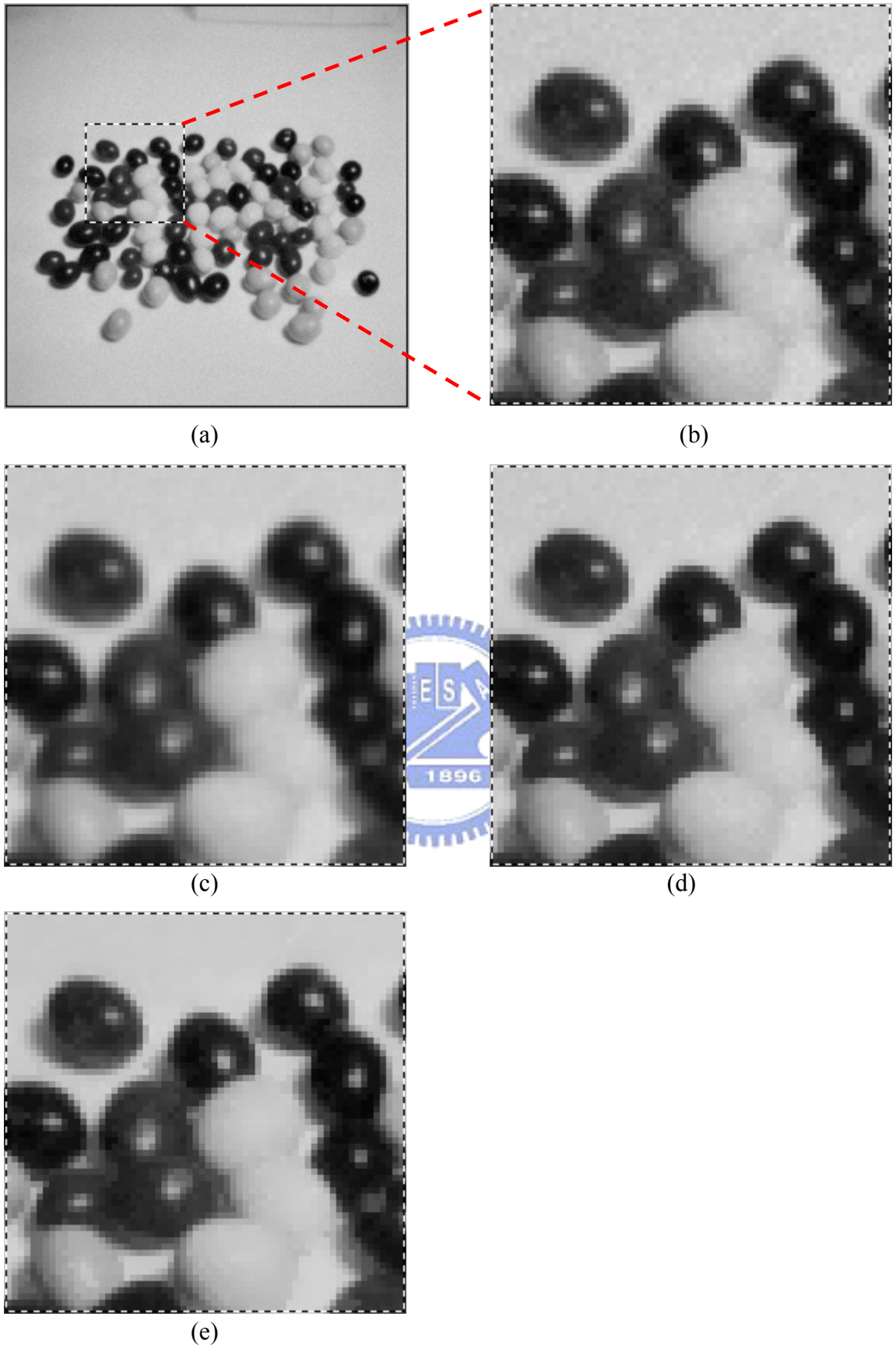


Fig. 23. Proposed filter has better filtering result under $\sigma=3.125$ Gaussian noise. (a) Adding $\sigma=3.125$ Gaussian noise; (b) 400% enlarged; (c) Filtered by Bilinear filter; (d) Filtered by Bosco-and-Mancuso filter; (e) Filtered by proposed filter.

Table 7.

Test result of filtering image “4.1.08-jellybeansg” with additive $\sigma=7.1875$ and $\sigma=3.125$ Gaussian noise.

Standard deviation of noisy image and filtered image on the flat area	$\sigma=7.1875$	$\sigma=3.125$
Noisy image	8.32	5.48
Bilinear	5.83	4.84
Bosco-and-Mancuso	6.26	5.14
Proposed	4.78	4.61

Case 4: Proposed filter has worst SSIM on texture images under high Gamma and Uniform noise condition:

Since there is a similar analysis result for filtering Gamma noise and Uniform noise in this case, we just use Gamma noise condition to explain this particular case. To explain the phenomenon, let’s bring up the image “1.2.03-Straw” for discussion. Proposed filter performs better with this image under $\sigma=18.75$ Gaussian noise, but performs worst under $\sigma=18.75$ Gamma noise condition. Table 8 is the test result of this image.

Table 8.

Use image “1.2.03-straw” as an example to explain the proposed filter has worst SSIM on texture images under high Gamma noise condition

$\sigma=18.75$	Gaussian		Gamma	
	SSIM	PSNR	SSIM	PSNR
Bosco-and-Mancuso	0.85456	20.0698	0.90466	20.8664
Proposed	0.89005	20.4736	0.88052	18.5804

By comparing texture images under $\sigma=18.75$ Gamma noise and Gaussian noise conditions. We found the histogram of image was expanded a lot by high Gamma noise, so that many noisy pixels fewer than zero or over saturated value to become edge information as shown in Figs. 24(a)-(c). Fig. 24(a) shows original image “1.2.03-straw” as well as its histogram. Figs. 24(b)-(c) show the image with additive $\sigma=18.75$ Gamma noise as well as its histogram and image with additive $\sigma=18.75$ Gaussian noise as well as its histogram respectively. As mentioned before, proposed algorithm filters noisy pixels based on nearest

pattern, so that it will keep edge information in most of cases as the histogram result shown in Fig. 24(e). On the contrary, Bosco-and-Mancuso filter won't keep edge information as much as proposed filter, so that it can recover damaged pixels as the histogram result shown in Fig. 24(d).

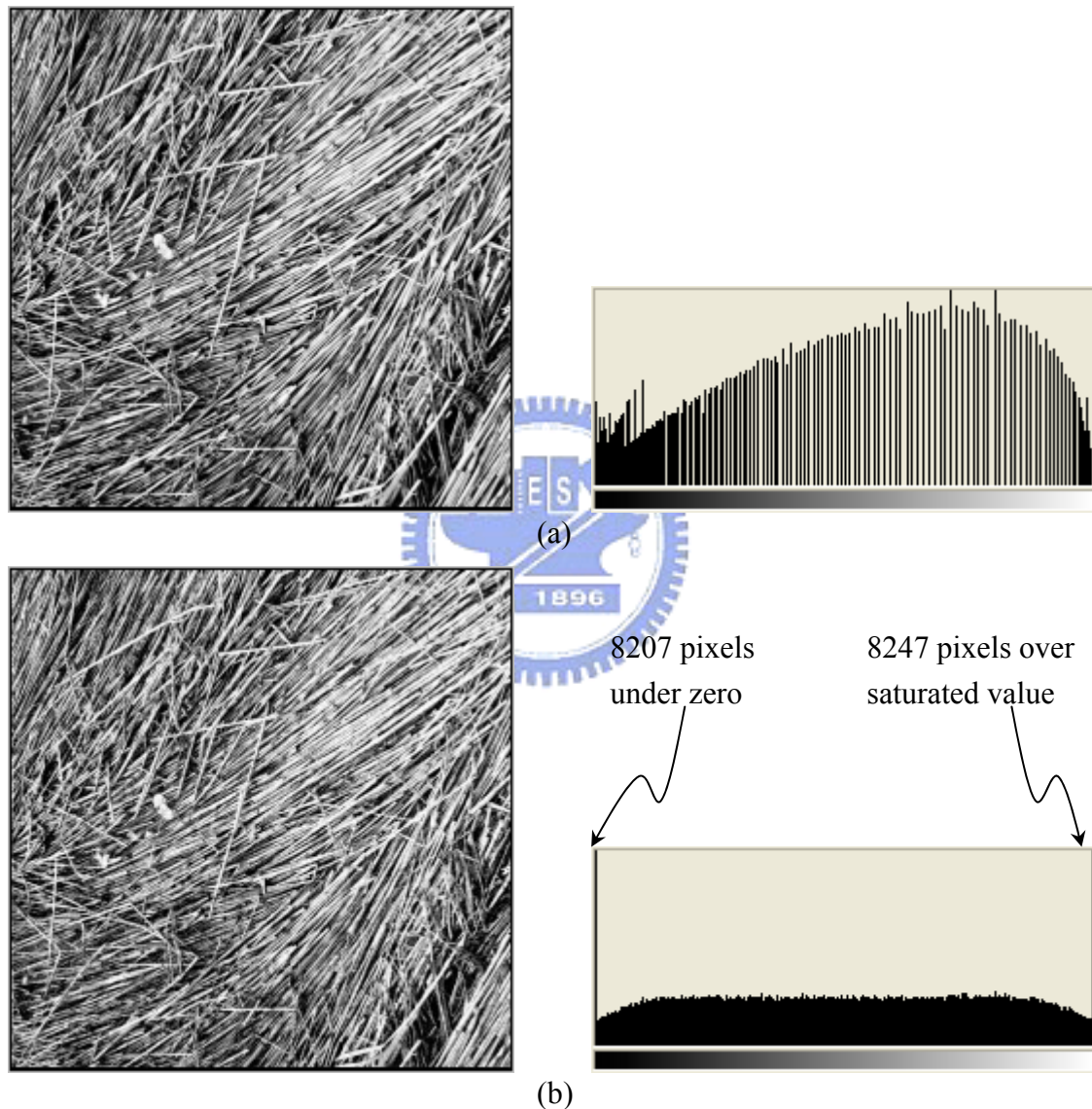


Fig. 24. $\sigma=18.75$ Gamma noise yielded too much edge information which is not able to be recovered by proposed filter. (a) Original image “1.2.03-Straw” and its histogram; (b) Image “1.2.03-Straw” with $\sigma=18.75$ Gaussian noise and its histogram; (c) Image “1.2.03-Straw” with $\sigma=18.75$ Gamma noise and its histogram; (d) Image “1.2.03-Straw” with $\sigma=18.75$ Gamma noise filtered by Bosco-and-Mancuso filter and its histogram; (e) Image “1.2.03-Straw” with $\sigma=18.75$ Gamma noise filtered by proposed filter and its histogram.

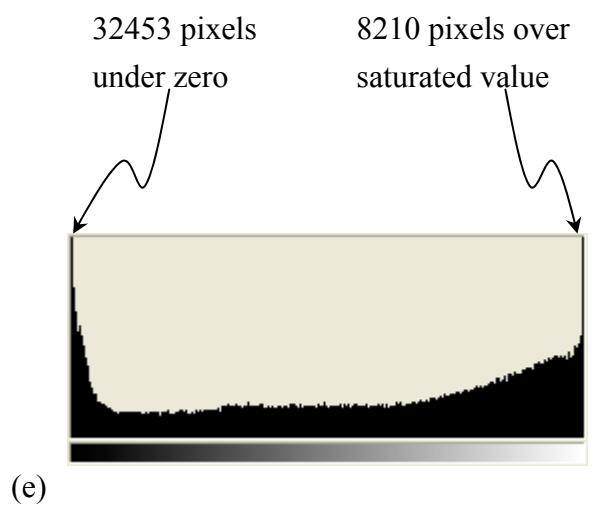
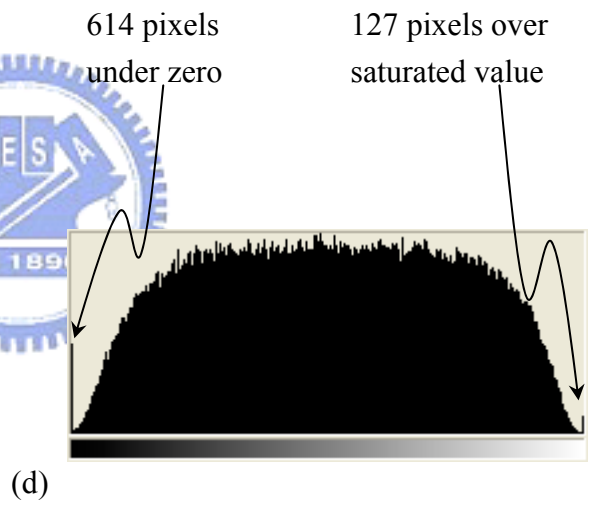
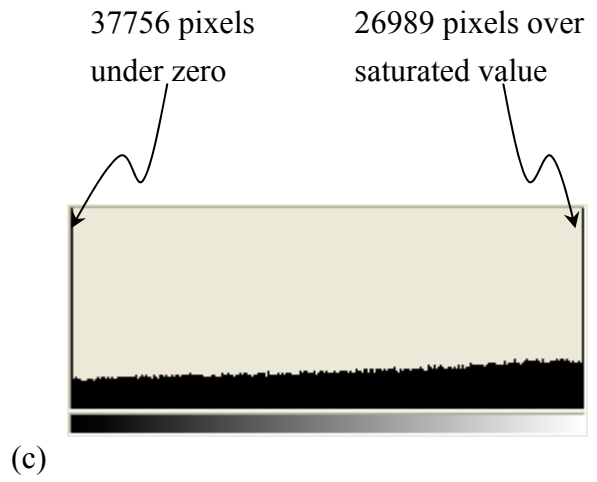


Fig. 24. (Continued)

Case 5: Proposed filter is not able to filter out Pepper-and-Salt noise:

In order to keep as much edge information as possible, proposed denoising method has been designed to not handle singular point such as Pepper-and-Salt noise, unless the image is captured under ISO 1600 condition. The reason that proposed denoising method handle Singular point under ISO 1600 condition is because standard deviation under such kind of condition is very huge, up to $\sigma=18.75$. In order to filter out most of noise, we also use 4σ to be the threshold to judge if current processed area is uniform or not. However, it still has 0.00633% possibility that noisy pixel in flat region won't be handled as uniform area. These un-handled pixels they all have big distance from neighboring pixels, so that they will look like Pepper-and-Salt noise very much. Image with Pepper-and-Salt-like noise is very unacceptable. This is the reason why the proposed denoising method pays attention to keep any edge information but ISO 1600 condition. This is also the reason why the testing result shown in the column named "Adding Pepper-and-Salt Noise for ISO 100 and ISO 400 (didn't handle singular point)" in Table 6 so bad. Fig. 25(a) shows the original image "testpat" with additive Pepper-and-Salt noise and Fig. 25(b) shows its 400% enlarged sub-image. Fig. 25(c) is used to illustrate that proposed filter is not able to filter out Pepper-and-Salt noise under ISO 100 condition as well ISO 400 condition. On the other hand, Fig. 25(d) is used to illustrate that proposed filter will filter out Pepper-and-Salt noise as well as any singular point in test image under ISO 1600 condition.

Case 6: Test patterns:

There are 4 kinds of step wedge patterns in USC web site, as shown in Fig. 26. The proposed filter on those patterns performs not badly except filtering Pepper-and-Salt noise as the testing results shown in Tables 9~14.

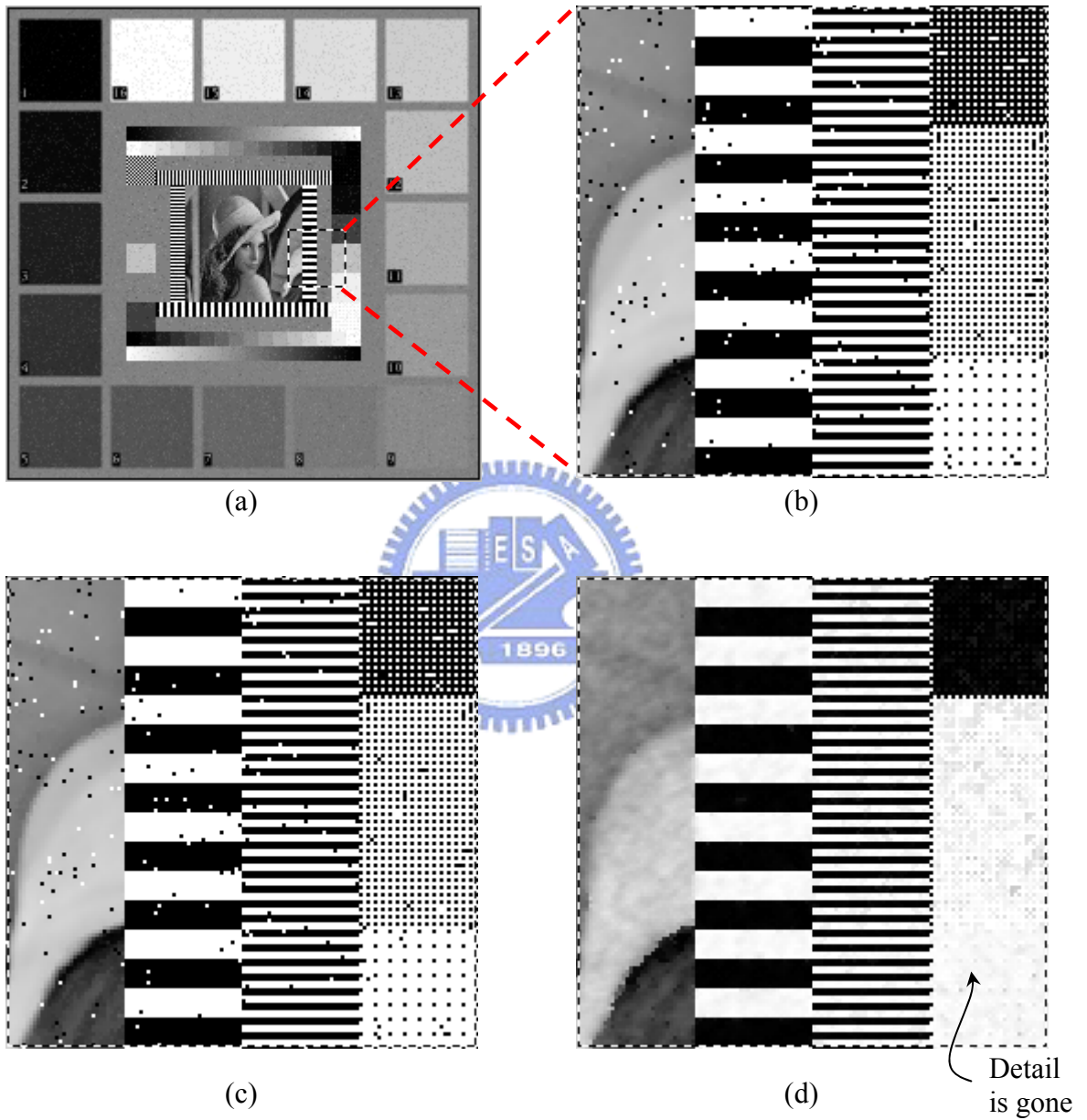


Fig. 25. Proposed filter is not able to filter Pepper-and-Salt noise. (a) Image “testpat” with additive Pepper-and-Salt noise; (b) 400% enlarged; (c) Filtered by proposed filter; (d) Image “testpat” with additive $\sigma=18.75$ Gaussian noise filtered by proposed filter. The detail is gone.

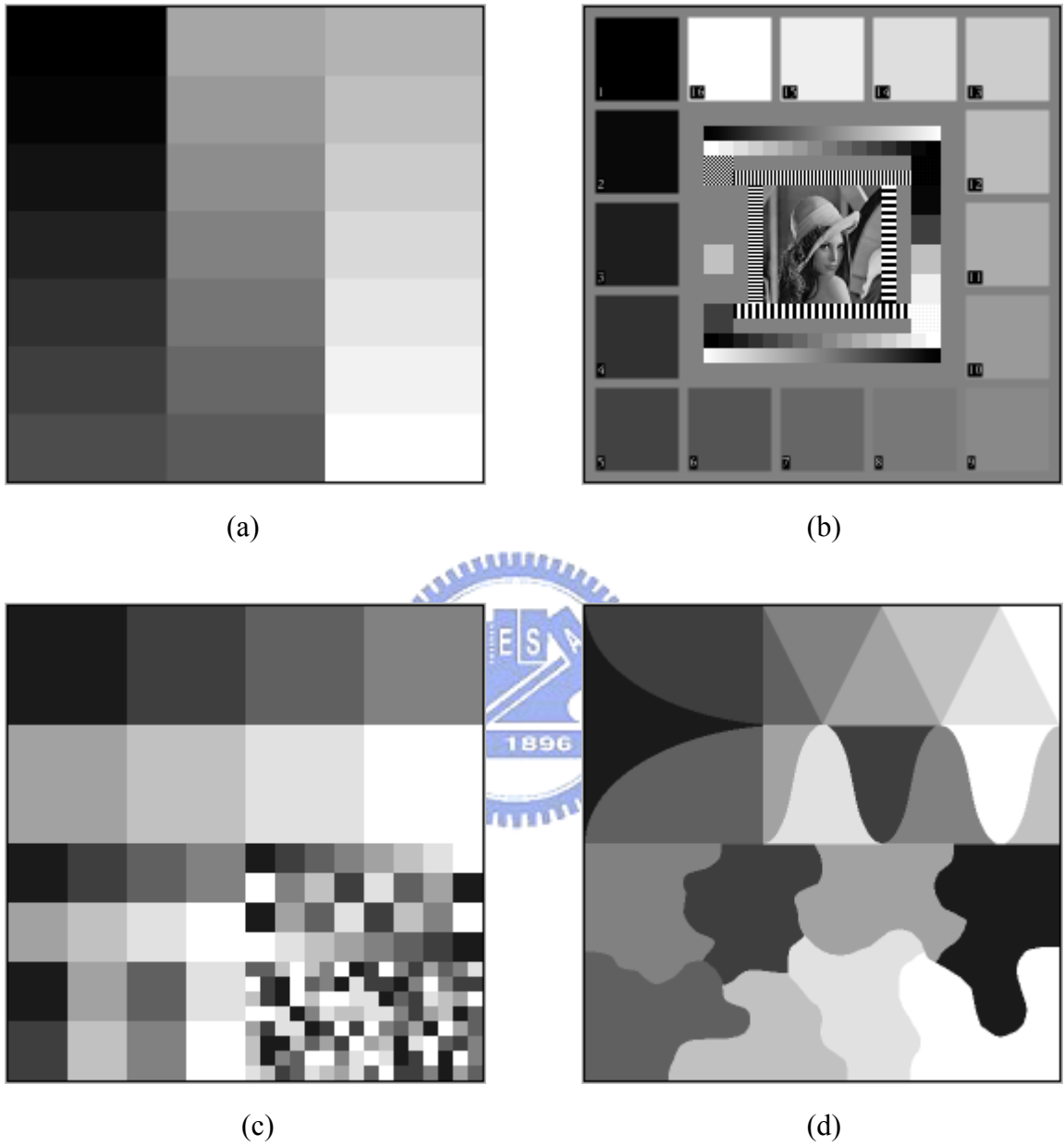


Fig. 26. 4 kinds of step wedge test pattern in USC web site. (a) Test pattern: “gray21”; (b) Test pattern: “testpat”; (c) Test pattern: “texmos2.s512-USC”; (d) Test pattern: “texmos3.s512-USC”.

Table 9.

Proposed SSIM and PSNR compare with other filters for the pattern images with additive Gaussian noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Gaussian Noise ($\sigma=3.125$)			Adding Gaussian Noise ($\sigma=7.1875$)			Adding Gaussian Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
gray21.raw	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.86169	2581.1482	38.1269	0.56981	12639.548	31.2278	0.18943	81271.003	23.1457
	Bilinear Filter	0.94226	1365.9309	40.8908	0.794	4793.8503	35.4382	0.4082	28548.825	27.6892
	BoscoMancuso	0.90577	1687.4743	39.9727	0.75005	5717.8466	34.6728	0.63608	13198.406	31.0399
	Proposed Filter	0.95675	810.0874	43.1598	0.85495	3421.6767	36.9027	0.67809	11678.162	31.5713
testpat.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.89196	2491.5535	38.2804	0.66265	12195.812	31.383	0.34891	79343.124	23.25
	Bilinear Filter	0.92933	78857.258	23.2767	0.81169	84246.344	22.9896	0.50223	113163.23	21.708
	BoscoMancuso	0.92384	4023.4569	36.1991	0.7991	8449.1149	32.977	0.69747	20102.204	29.2126
Proposed Filter	0.96842	877.6523	42.8119	0.89474	3594.5876	36.6886	0.72138	48793.78	25.3614	
texmos2.s512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.89677	2490.1966	38.2827	0.6558	12492.351	31.2786	0.3015	82135.833	23.0998
	Bilinear Filter	0.95839	4434.5423	35.7766	0.8424	7912.047	33.2622	0.50788	32417.001	27.1373
	BoscoMancuso	0.93261	2012.5697	39.2076	0.80868	6716.9773	33.9733	0.7007	20050.254	29.2239
Proposed Filter	0.9782	692.3014	43.8421	0.90685	3204.0857	37.188	0.73498	15370.955	30.3781	
texmos3.s512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.88374	2514.2197	38.241	0.61683	12607.146	31.2389	0.24088	83343.038	23.0364
	Bilinear Filter	0.95547	3001.1416	37.4722	0.8257	6473.6508	34.1336	0.45923	30800.013	27.3596
	BoscoMancuso	0.92485	1704.4474	39.9292	0.78819	5839.4434	34.5814	0.67857	15203.374	30.4257
Proposed Filter	0.97471	685.4412	43.8854	0.89252	3184.7793	37.2143	0.70546	14647.196	30.5875	

Table 10.

Proposed SSIM and PSNR compare with other filters for the pattern images with additive Rayleigh noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Rayleigh Noise ($\sigma=3.125$)			Adding Rayleigh Noise ($\sigma=7.1875$)			Adding Rayleigh Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
gray21.raw	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.91361	2238.071	38.7463	0.69756	10375.783	32.0849	0.31715	61014.651	24.3907
	Bilinear Filter	0.96808	1544.6401	40.3568	0.86727	5411.3963	34.912	0.54072	28367.354	27.7169
	BoscoMancuso	0.94185	1687.5055	39.9726	0.83042	6086.5879	34.4013	0.72437	18668.392	29.534
	Proposed Filter	0.98795	891.6189	42.7433	0.94725	3534.5486	36.7617	0.77221	17656.987	29.7759
testpat.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92969	2007.004	39.2196	0.75724	9207.7902	32.6035	0.45216	56297.669	24.7402
	Bilinear Filter	0.9492	77626.061	23.345	0.87071	81157.093	23.1518	0.61144	102436.71	22.1405
	BoscoMancuso	0.95011	4474.9716	35.7372	0.8609	8484.9208	32.9586	0.776	20737.724	29.0775
Proposed Filter	0.98943	818.7609	43.1135	0.95944	3009.4023	37.4603	0.81788	16502.128	30.0697	
texmos2.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92824	2052.4968	39.1223	0.74454	9907.8039	32.2853	0.39556	62924.589	24.2569
	Bilinear Filter	0.96951	4244.3173	35.967	0.88764	7515.0727	33.4857	0.60906	30626.527	27.3841
	BoscoMancuso	0.95163	1827.0844	39.6275	0.85882	6400.0638	34.1832	0.7719	23168.566	28.5961
Proposed Filter	0.99256	694.9146	43.8258	0.96315	3032.9956	37.4264	0.81861	19282.503	29.3934	
texmos3.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92215	2019.8463	39.1919	0.72111	9742.0132	32.3586	0.34582	61163.715	24.3801
	Bilinear Filter	0.96902	2955.0478	37.5394	0.87957	6276.1618	34.2681	0.57787	28990.273	27.6226
	BoscoMancuso	0.94865	1541.1599	40.3666	0.84884	5598.4131	34.7644	0.7632	19307.198	29.3879
Proposed Filter	0.992	679.7585	43.9215	0.96045	2918.4346	37.5936	0.80989	17778.75	29.7461	

Table 11.

Proposed SSIM and PSNR compare with other filters for the pattern images with additive Gamma noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Gamma Noise ($\sigma=3.125$)			Adding Gamma Noise ($\sigma=7.1875$)			Adding Gamma Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
gray21.raw	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.89513	5945.2439	34.5034	0.65784	25668.281	28.1511	0.31751	125246.05	21.2674
	Bilinear Filter	0.949	5195.2567	35.089	0.81937	20774.858	29.0697	0.51109	96290.129	22.4093
	BoscoMancuso	0.92476	5371.009	34.9445	0.78959	21391.241	28.9427	0.67148	88370.86	22.782
	Proposed Filter	0.96782	4633.2458	35.5862	0.89436	19198.495	29.4124	0.71314	87603.682	22.8199
testpat.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92551	4447.078	35.7643	0.73343	20334.955	29.1626	0.4484	103479.42	22.0965
	Bilinear Filter	0.94504	79472.553	23.2429	0.84361	91026.462	22.6534	0.59156	148609.87	20.5246
	BoscoMancuso	0.9433	15450.724	30.3556	0.83609	25017.668	28.2626	0.74418	72470.676	23.6435
Proposed Filter	0.98404	3294.6763	37.067	0.92675	14500.26	30.6313	0.78001	67367.289	23.9606	
texmos2.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92651	4839.1627	35.3974	0.73261	24107.841	28.4235	0.38041	124239.45	21.3025
	Bilinear Filter	0.96743	6479.376	34.1297	0.87053	20366.476	29.1559	0.55788	92469.871	22.5851
	BoscoMancuso	0.95101	4589.5552	35.6274	0.8478	20358.403	29.1576	0.69587	87669.907	22.8166
Proposed Filter	0.98932	3515.7801	36.7849	0.93934	17648.791	29.7779	0.73268	86594.175	22.8702	
texmos3.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92085	4708.5954	35.5162	0.70991	23605.885	28.5149	0.33295	124563.88	21.2912
	Bilinear Filter	0.96699	5398.7012	34.9222	0.86319	19532.364	29.3375	0.52744	94326.465	22.4987
	BoscoMancuso	0.9485	4172.4869	36.0411	0.83877	19227.176	29.4059	0.68552	87006.945	22.8495
Proposed Filter	0.9891	3404.3442	36.9247	0.93848	17116.827	29.9108	0.72268	86800.768	22.8598	

Table 12.

Proposed SSIM and PSNR compare with other filters for the pattern images with additive Exponential noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Exponential Noise ($\sigma=3.125$)			Adding Exponential Noise ($\sigma=7.1875$)			Adding Exponential Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
gray21.raw	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.91702	1847.8236	39.5785	0.71914	8504.4393	32.9486	0.35558	49761.755	25.2761
	Bilinear Filter	0.9693	1192.7866	41.4795	0.87887	3764.3945	36.4881	0.58632	19288.884	29.392
	BoscoMancuso	0.94269	1324.6821	41.024	0.82942	4823.1974	35.4117	0.74667	11748.592	31.5452
	Proposed Filter	0.98067	663.1393	44.029	0.93411	2461.9199	38.3323	0.80734	9532.1628	32.4532
testpat.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.93144	1739.3328	39.8413	0.76952	7950.1568	33.2413	0.47606	48446.51	25.3925
	Bilinear Filter	0.94942	77526.464	23.3506	0.87571	80363.355	23.1945	0.63797	96713.559	22.3902
	BoscoMancuso	0.94983	4904.7123	35.3389	0.85594	8215.4593	33.0988	0.77913	16931.361	29.9582
Proposed Filter	0.98255	701.0975	43.7873	0.94427	2468.8725	38.3201	0.83954	10677.424	31.9604	
texmos2.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.93244	1756.3942	39.7989	0.75637	8610.7713	32.8947	0.43219	50687.111	25.1961
	Bilinear Filter	0.97061	4115.3352	36.101	0.89166	6558.7002	34.0769	0.64289	22049.092	28.8112
	BoscoMancuso	0.95321	1565.0766	40.2997	0.85394	5672.8048	34.7071	0.7826	16902.492	29.9656
Proposed Filter	0.98578	597.2347	44.4836	0.94637	2417.7251	38.411	0.84091	11053.546	31.8101	
texmos3.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92642	1731.0555	39.862	0.73449	8413.8305	32.9951	0.38092	50148.344	25.2425
	Bilinear Filter	0.9703	2760.3671	37.8354	0.88474	5181.9574	35.1001	0.61081	20640.077	29.098
	BoscoMancuso	0.9502	1298.6756	41.1101	0.84314	4770.9046	35.4591	0.76952	13270.307	31.0163
Proposed Filter	0.98485	579.6294	44.6136	0.94326	2267.023	38.6905	0.83028	10211.132	32.1543	

Table 13.

Proposed SSIM and PSNR compare with other filters for the pattern images with additive Uniform noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Uniform Noise ($\sigma=3.125$)			Adding Uniform Noise ($\sigma=7.1875$)			Adding Uniform Noise ($\sigma=18.75$)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
gray21.raw	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.8952	5956.6032	34.4951	0.65441	25619.033	28.1595	0.31529	126347.02	21.2294
	Bilinear Filter	0.94931	5209.3554	35.0772	0.8184	20685.768	29.0884	0.50671	96815.345	22.3856
	BoscoMancuso	0.917	5530.6117	34.8173	0.76423	21953.196	28.8301	0.65714	89256.298	22.7387
	Proposed Filter	0.96932	4629.4072	35.5898	0.89801	18981.989	29.4617	0.71221	87606.548	22.8197
testpat.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92495	4447.1449	35.7643	0.73041	20383.915	29.1522	0.44547	103871.92	22.0801
	Bilinear Filter	0.94508	79432.207	23.2451	0.84265	91014.91	22.654	0.5877	148477.76	20.5285
	BoscoMancuso	0.93727	13735.503	30.8666	0.81612	24197.026	28.4075	0.73208	72262.13	23.656
Proposed Filter	0.98527	3271.645	37.0974	0.93003	14437.245	30.6502	0.77934	67156.893	23.9742	
texmos2.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.92627	4820.3966	35.4143	0.73086	24058.324	28.4324	0.37639	125603.2	21.2551
	Bilinear Filter	0.96739	6461.7512	34.1416	0.8701	20326.11	29.1645	0.55147	93048.245	22.558
	BoscoMancuso	0.94423	4656.992	35.564	0.82649	20868.826	29.0501	0.68145	88011.203	22.7997
Proposed Filter	0.99032	3478.0076	36.8318	0.9438	17424.23	29.8335	0.73027	86669.226	22.8664	
texmos3.512-USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf	1	0	Inf
	Noisy Image	0.91993	4728.1647	35.4982	0.70841	23474.807	28.5391	0.33004	125584.08	21.2557
	Bilinear Filter	0.96682	5399.4689	34.9216	0.86287	19439.865	29.3581	0.52153	94904.267	22.4722
	BoscoMancuso	0.94051	4339.6822	35.8705	0.81526	19726.343	29.2946	0.66954	87882.846	22.806
Proposed Filter	0.99005	3393.9243	36.9381	0.94294	16835.479	29.9828	0.72173	86751.026	22.8623	

Table 14.

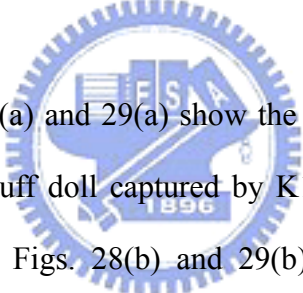
Proposed SSIM and PSNR compare with other filters for the pattern images with additive
Pepper-and-Salt noise.

Noisy Image and Filtered Image Comparing with Original Image		Adding Pepper-and-Salt Noise ISO 100 and ISO 400 (didn't handle singular point)			Adding Pepper-and-Salt Noise ISO 1600 (handle singular point)		
	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
gray21.raw	Original Image	1	0	Inf	1	0	Inf
	Noisy Image	0.50308	114175.72	21.6693	0.50308	114175.72	21.6693
	Bilinear Filter	0.62531	38989.987	26.3355	0.62531	38989.987	26.3355
	BoscoMancuso	0.92294	14950.934	30.4984	0.92294	14950.934	30.4984
	Proposed Filter	0.50223	114197.97	21.6685	0.9609	11514.065	31.6328
testpat.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf
	Noisy Image	0.59009	112266.33	21.7426	0.59009	112266.33	21.7426
	Bilinear Filter	0.67419	116489.57	21.5822	0.67419	116489.57	21.5822
	BoscoMancuso	0.90166	69264.489	23.84	0.90166	69264.489	23.84
Proposed Filter	0.58888	112328.59	21.7402	0.94542	62281.193	24.3015	
texmos2.s 512- USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf
	Noisy Image	0.55229	114588.5	21.6537	0.55229	114588.5	21.6537
	Bilinear Filter	0.66482	42267.897	25.985	0.66482	42267.897	25.985
	BoscoMancuso	0.91555	19041.044	29.4482	0.91555	19041.044	29.4482
Proposed Filter	0.55229	114588.5	21.6537	0.96419	12353.364	31.3272	
texmos3.s 512- USC.raw	Method	SSIM	MSE(12bit)	PSNR(dB)	SSIM	MSE(12bit)	PSNR(dB)
	Original Image	1	0	Inf	1	0	Inf
	Noisy Image	0.52099	110549.17	21.8095	0.52099	110549.17	21.8095
	Bilinear Filter	0.64158	39021.718	26.332	0.64158	39021.718	26.332
	BoscoMancuso	0.92496	14915.49	30.5087	0.92496	14915.49	30.5087
Proposed Filter	0.52099	110549.17	21.8095	0.9662	9754.6044	32.353	

4.2 Test Results of the CFA Raw Images:

In this section, we would like to compare the proposed denoising method with real camera K1003. CFA raw images were captured by K1003. The only difference of processed images is denoising filter only in order to compare the testing result conveniently.

Fig. 27(a) shows the image which is a scene consists of resolution chart and fluff doll captured by K1003 under ISO 100 condition. Fig. 27(b) shows the sub-image filtered by K1003 as well as standard deviation calculated on flat area. Fig. 27(c) shows the sub-image filtered by Bosco-and-Mancuso filter as well as standard deviation calculated on flat area and that Fig. 27(d) shows the sub-image filtered by proposed filter as well as standard deviation calculated on flat area.

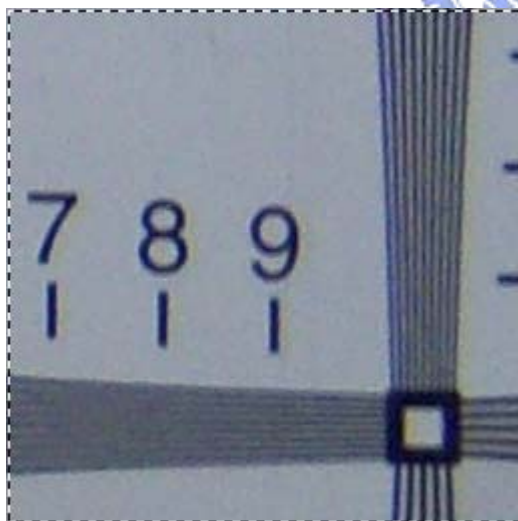


In the same construction, Figs. 28(a) and 29(a) show the images which used the same scene consists of resolution chart and fluff doll captured by K1003 under ISO 400 condition and ISO 1600 condition respectively. Figs. 28(b) and 29(b) show the sub-images filtered by K1003 as well as standard deviation calculated on flat area. Figs. 28(c) and 29(c) show the sub-images filtered by Bosco-and-Mancuso filter as well as standard deviation calculated on flat area and that Figs. 28(d) and 29(d) show the sub-images filtered by proposed filter as well as standard deviation calculated on flat area.

By comparing the image with subjective perception, proposed denoising method slightly reduced the resolution. However, it gains the benefit of reducing standard deviation a lot as the images shown in Figs. 27(d), 28(d), and 29(d). We also found there is a discontinuous pattern problem as shown in Figs. 28(d) and 29(d), since the filtering strength of proposed denoising method didn't change smoothly.



(a)



(b)

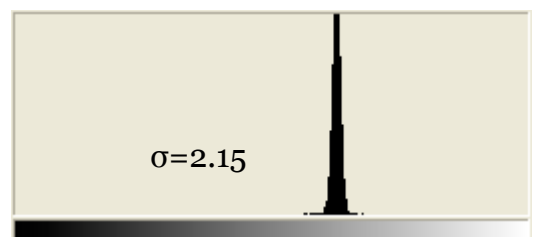
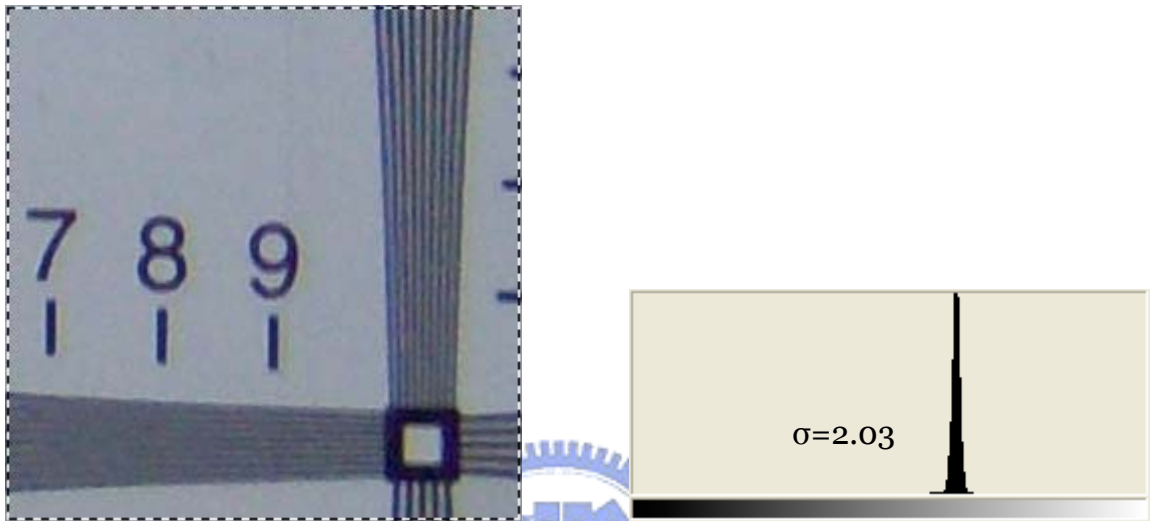
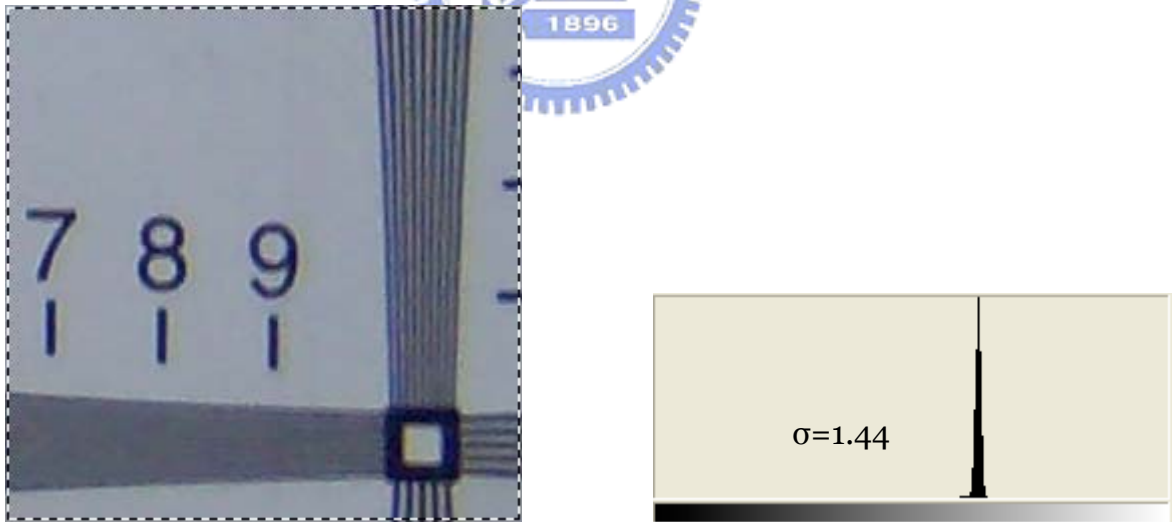


Fig. 27. CFA image captured by K1003 under ISO 100 condition. (a) Resolution chart with fluff doll captured by K1003 under ISO 100 condition; (b) ISO 100 noise filtered by K1003 and standard deviation of flat area; (c) ISO 100 noise filtered by Bosco-and-Mancuso filter and standard deviation of flat area; (d) ISO 100 noise filtered by proposed filter and standard deviation of flat area.



(c)

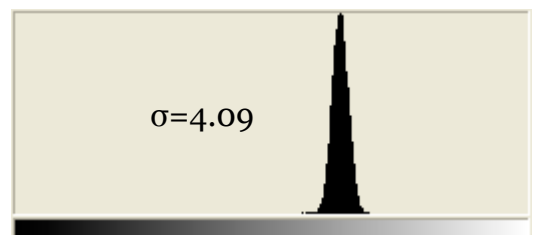
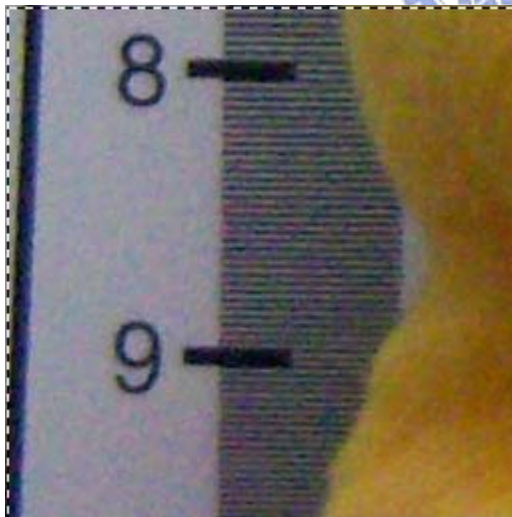


(d)

Fig. 27. (Continued)

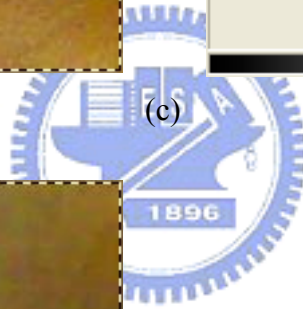
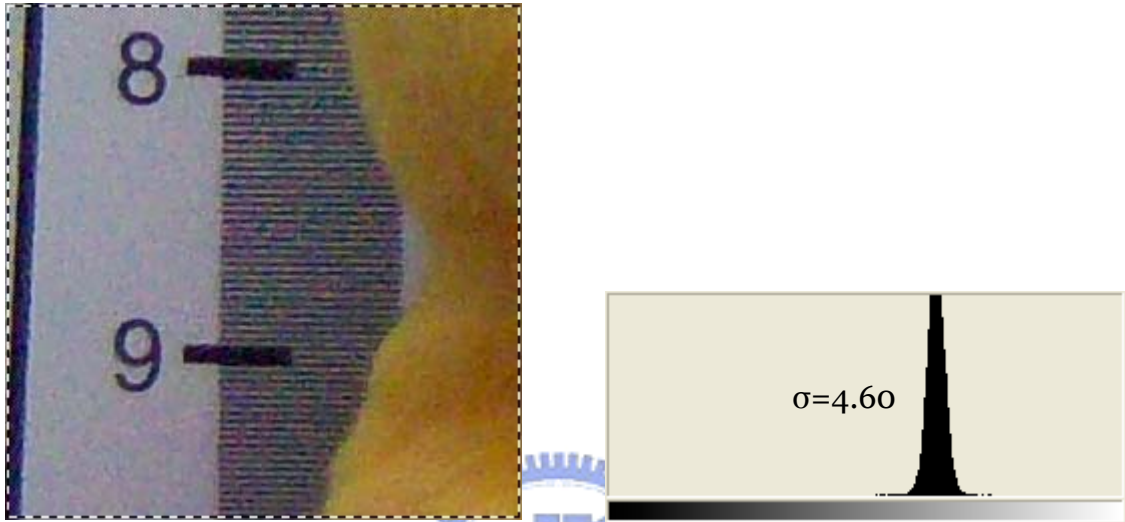


(a)

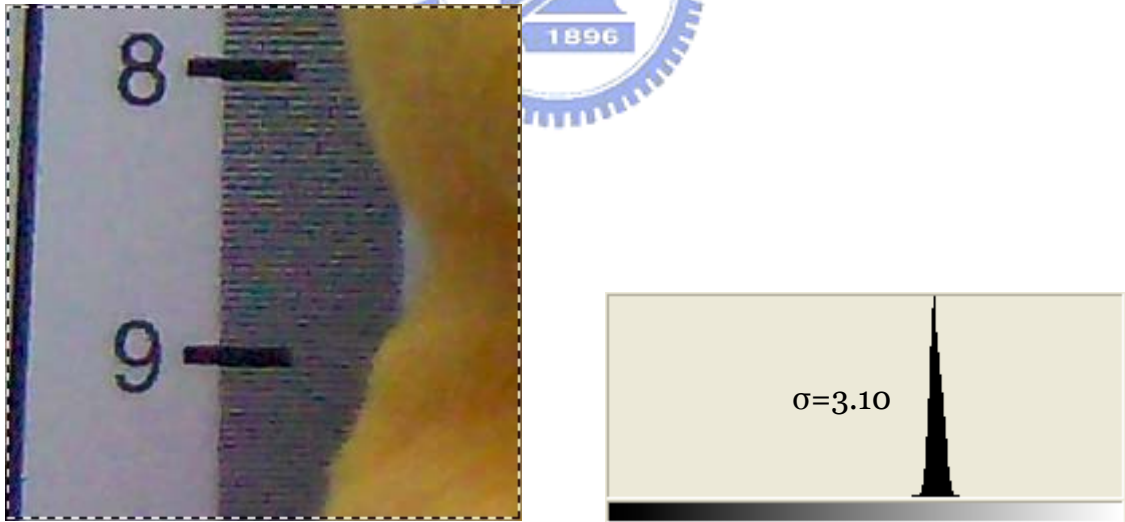


(b)

Fig. 28. CFA image captured by K1003 under ISO 400 condition. (a) Resolution chart with fluff doll captured by K1003 under ISO 400 condition; (b) ISO 400 noise filtered by K1003 and standard deviation of flat area; (c) ISO 400 noise filtered by Bosco-and-Mancuso filter and standard deviation of flat area; (d) ISO 400 noise filtered by proposed filter and standard deviation of flat area.



(c)

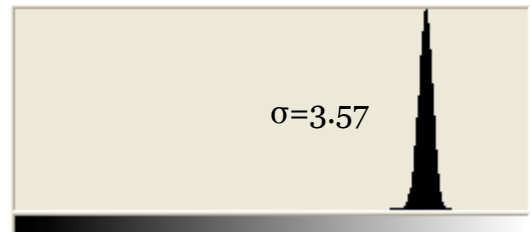
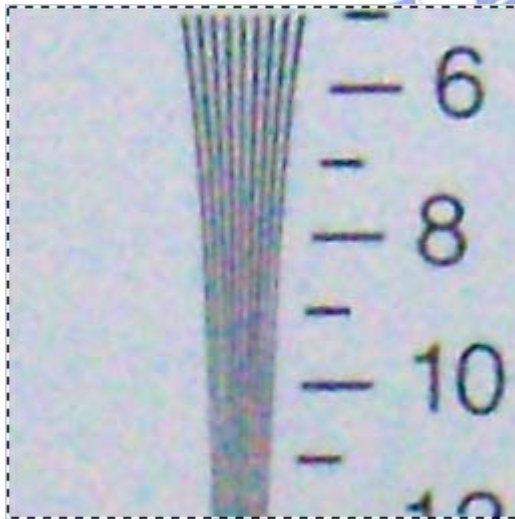


(d)

Fig. 28. (Continued)



(a)



(b)

Fig. 29. CFA image captured by K1003 under ISO 1600 condition. (a) Resolution chart with fluff doll captured by K1003 under ISO 1600 condition; (b) ISO 1600 noise filtered by K1003 and standard deviation of flat area; (c) ISO 1600 noise filtered by Bosco-and-Mancuso filter and standard deviation of flat area; (d) ISO 1600 noise filtered by proposed filter and standard deviation of flat area.

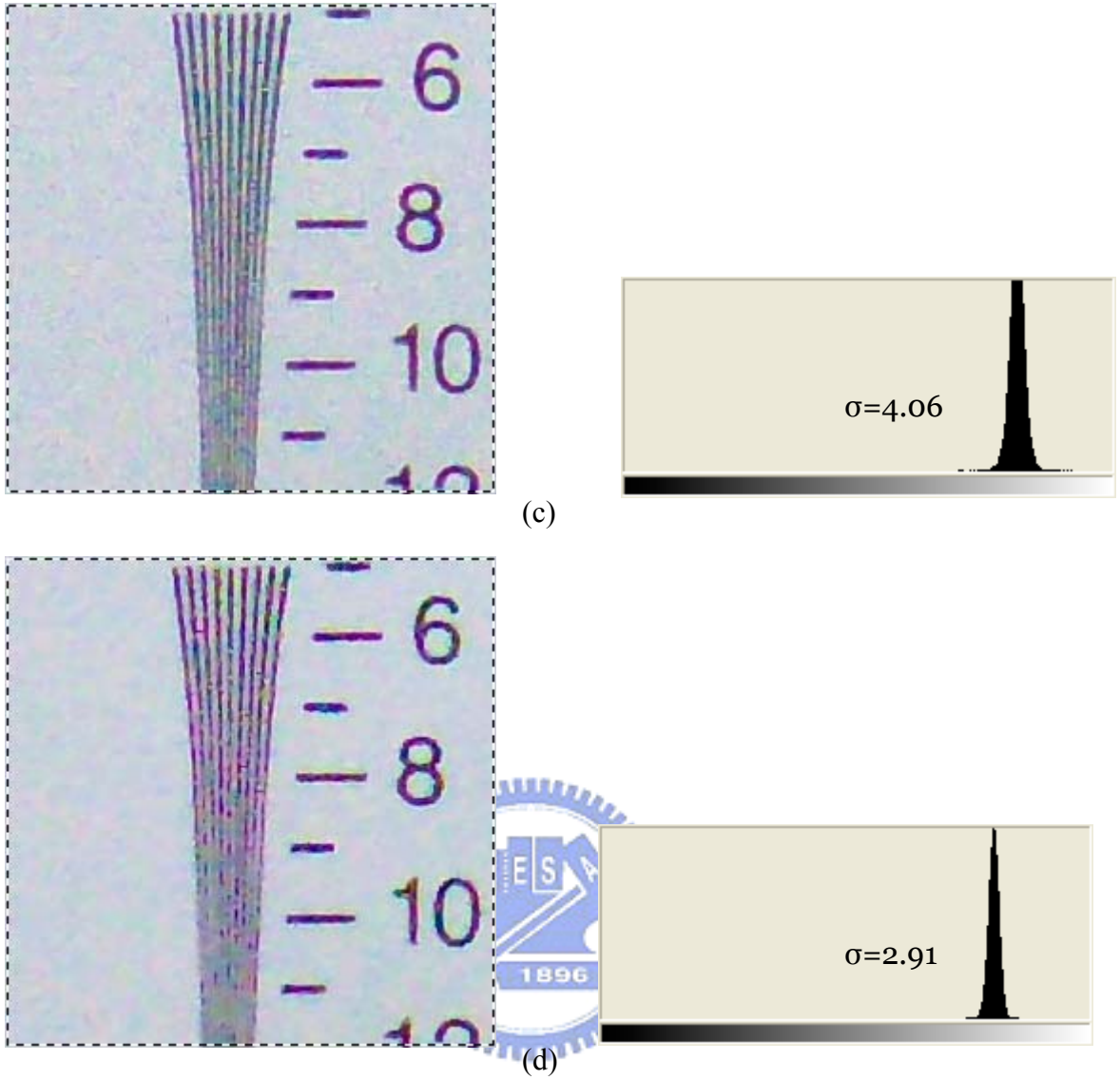


Fig. 29. (Continued)

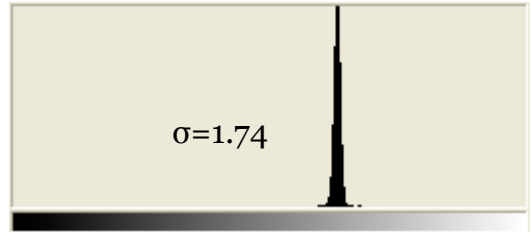
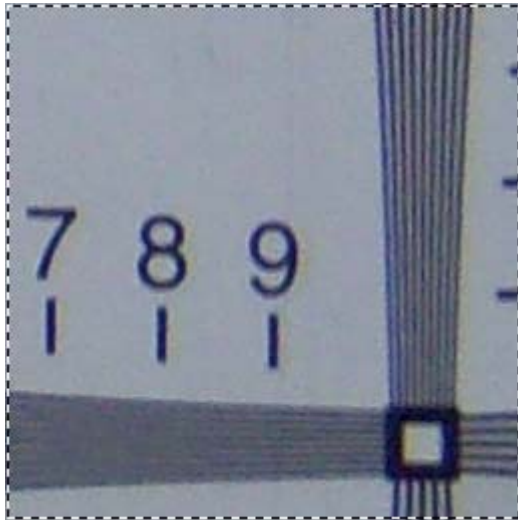
In order to yield better result, the parameter $Th_{uniform}$ of green channel has been fine tuned as

$$Th_{uniform} = \begin{cases} 1 * \sigma(C_0), & \text{if } ISO = 100, \\ 3 * \sigma(C_0), & \text{if } ISO = 1600, \\ 2 * \sigma(C_0), & \text{else,} \end{cases} \quad (42)$$

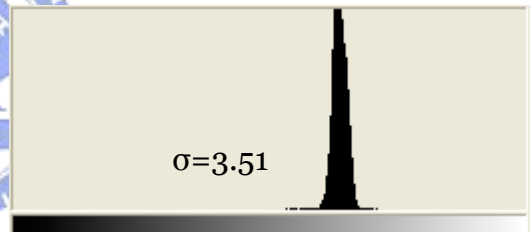
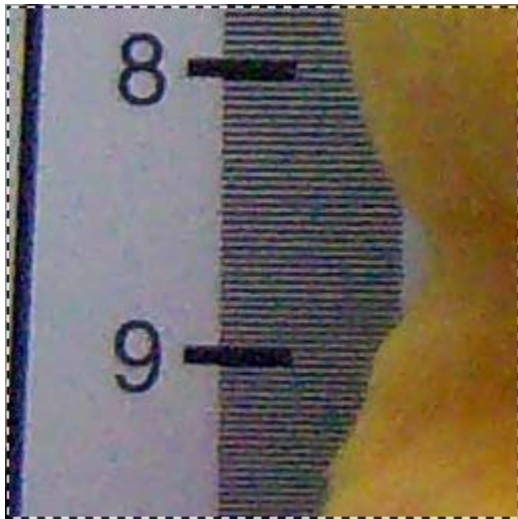
and fine tuned

$$Th_{uniform} = \begin{cases} 3 * \sigma(C_0), & \text{if } ISO = 1600, \\ 2 * \sigma(C_0), & \text{else,} \end{cases} \quad (43)$$

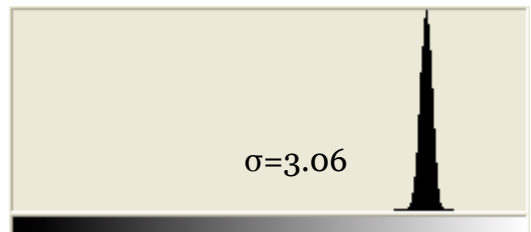
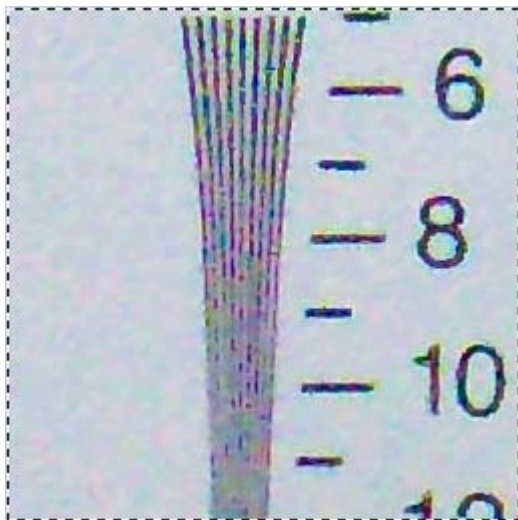
for red/blue channel. As the images shown in Figs. 30(a)-(b), resolution preserved more accordingly and the standard deviation on the flat area still less than other filters. Nonetheless, Fig. 30(c) shows that the discontinuous pattern problem is still exist.



(a)



(b)



(c)

Fig. 30. Images with fine tuned parameter. (a) ISO 100; (b) ISO 400; (c) ISO 1600.

Chapter 5

Conclusions

In this thesis, we have presented a denoising method which consists of three ideas. One is to filter noisy pixel based on nearest pattern to keep edge information, another one is to use noise characteristic of camera to judge the uniformity of current processed area and the last one is to make use of spatial masking to keep edge information again on the highly texture area.

The results from above experiments indicate that proposed denoising method performs better under ISO 100 and ISO 400 condition, since there is a discontinuous patterns problem generated in few specific image areas under ISO 1600 condition. The root cause of discontinuous patterns problem is that filtering strength of proposed denoising method didn't change smoothly, so that discontinuous patterns happened under ISO 1600 condition. Not only this issue is the next step we would like to overcome, but also how to judge current processed area is uniform without sacrificed real signal too much is the topic we will be interested in. In addition, a method to fine tune the filtering strength automatically under any ISO condition would be another valuable research to optimize noise filtering strength.

References

- [1] A. Bosco and M. Mancuso, "Adaptive filtering for image denoising," ICCE. International Conference on Consumer Electronics, pp. 208-209, Jun. 2001.
- [2] A. Bosco and M. Mancuso, "Noise filter for Bayer pattern image data," Europe Patent EP1289309, May 2003.
- [3] M. Mancuso, "Non-linear image filter for filtering noise," United States Patent 6108455, Aug. 2000.
- [4] B. Bayer, "Color Imaging Array," U. S. Patent 3 971 065, Jul. 1976.
- [5] Z. Wang and A. C. Bovik, "A universal image quality index," IEEE Signal Processing Letters, Vol. 9, no. 3, pp. 81-84, Mar. 2002.
- [6] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment," Invited Paper, IEEE Asilomar Conference on Signals, Systems and Computers, Vol. 2, pp.1398-1402, Nov. 2003.
- [7] Z. 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, Vol. 13, no. 4, pp. 600-612, Apr. 2004.
- [8] G. H. Chen, C. L. Yang, L. M. Po, and S. L. Xie, "Edge-based structural similarity for image quality assessment," IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 2, pp. 933-936, May 2006.