先去马赛克还原成全彩图像，然后进行坏点矫正，然后针对坏点再进行去马赛克处理的到最终的全彩图像

Adaptive Order-Statistics Multi-Shell Filtering for Bad Pixel Correction within CFA Demosaicking

Jim S. Jimmy Li and Sharmil Randhawa
School of Computer Science, Engineering and Mathematics
Flinders University
Adelaide, Australia
Jimmy.Li@flinders.edu.au, Sharmil.Randhawa@flinders.edu.au

***Abstract*— As today's digital cameras contain millions of image sensors, it is highly probable that the image sensors will contain a few defective pixels due to errors in the fabrication process. While these bad pixels would normally be mapped out in the manufacturing process, more defective pixels, known as hot pixels, could appear over time with camera usage. Since some hot pixels can still function at normal settings, they need not be permanently mapped out because they will only appear on a long exposure and/or at high ISO settings. In this paper, we apply an adaptive order-statistics multi-shell filter within CFA demosaicking to filter out only bad pixels whilst preserving the rest of the image. The CFA image containing bad pixels is first demosaicked to produce a full colour image. The adaptive filter is then only applied to the actual sensor pixels within the colour image for bad pixel correction. Demosaicking is then re-applied at those bad pixel locations to produce the final full colour image free of defective pixels. It has been shown that our proposed method outperforms a separate process of CFA demosaicking followed by bad pixel removal.**

Keywords-bad pixel correction; CFA demosaicking; adaptive order-statistics; multi-shell filtering

I. INTRODUCTION

Demosaicking refers to determining the missing colour values at each pixel when a single-sensor digital camera is used for colour image capture. The Bayer Colour Filter Array (CFA) is the most common colour filter array used [1]. Fig. 1 shows a 5x5 window of a Bayer array neighbourhood. In this pattern, the green colour is sampled at twice the rate of the red and blue values. This is due to the peak sensitivity of the human visual system which lies in the green spectrum [1].

All the sensor values produced by digital sensor must be accurate in order for the demosaicking process to produce a visually pleasing image. However, despite advances in the manufacturing process, digital cameras often contain a few defective pixels as a result of noise or fabrication errors [2]. There are three main types of defective or bad pixels: hot, dead or noisy. A hot pixel produces a brighter than expected spot, while a dead pixel produces a darker than expected spot in the output image. A noisy pixel produces a sensor value which differs from neighbouring pixels by more than a certain amount when exposed to the same light conditions.

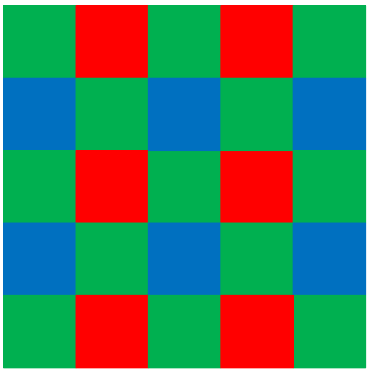

Figure 1. 5x5 Bayer CFA pattern

Bad Pixel Correction (BPC) is the process of detecting and correcting defective pixels. Traditionally BPC and demosaicking have been performed in two separate stages, either in software or in a digital circuit implemented in hardware. Each stage adds to the complexity and expense of processing sensor values for final output on a display device. Our proposed algorithm aims to combine an accurate BPC algorithm with demosaicking in a single stage.

本算法将DPC和demosaic合并为一步

II. ADAPTIVE ORDER-STATISTICS MULTI-SHELL FILTERING

In this section, we describe the adaptive order-statistics multi-shell filtering method which can be used to remove and correct for bad pixels.

The multi-shell median filters have been shown to be effective in removing impulsive noise while preserving image details [3]-[8]. By modifying its filter structure, it can be used for detection of bad pixels effectively.

Let $y_{m/shell}(m,n)$ be the multi-shell median filter output at position (m,n) and $a(m,n)$ be the sample at the centre of the filter window of size $(2N+1)\,\mathrm{x}\,(2N+1)$. Let $S(m,n)$ be the set of samples surrounding the central sample $a(m,n)$. The generalized j^{th} shell [3] is defined as:

978-1-4244-4547-9/09/$26.00 ©2009 IEEE

$$
\begin{aligned}
&S[j](m,n) = \\
&\{a(m-j,n-j), a(m-j,n-j+1), ..., a(m-j,n+j), \\
&\ a(m-j+1,n-j), \qquad\qquad , a(m-j+1,n+j), \\
&\qquad ,..., \qquad\qquad\qquad ,..., \\
&a(m+j-1,n-j), \qquad\qquad , a(m+j-1,n+j), \\
&\ a(m+j,n-j), a(m+j,n-j+1), ..., a(m+j,n+j)\}
\end{aligned} \tag{1}
$$

where $1 \le j \le N$.

The output of the median based multi-shell order-statistics filter [7] with K shells is defined by:

$$
\begin{aligned}
y_{m/shell}(m,n) = median\,[&S[K]_{(r_K)}(m,n), ..., S[1]_{(r_1)}(m,n), \\
&a(m,n), S[1]_{(N_1-r_1+1)}(m,n), ..., S[K]_{(N_K-r_K+1)}(m,n)]
\end{aligned} \tag{2}
$$

where $S[j]_{(r_j)}(m,n) = r_j^{th}$ order statistic of $S[j](m,n)$; $1 \le j \le K$ and its complementary order-statistics operator of the same shell is $S[j]_{(N_j-r_j+1)}(m,n)$. r_j , the rank of the complementary order-statistics operators, is within the range $1 \le r_j \le \frac{1}{2}N_j$ where N_j is the cardinal number of the shell $S[j](m,n)$.

For a 5x5 window, the output of the median based multi-shell order-statistics filter is given by:

$$
\begin{aligned}
y_{m/shell_{(r_1,r_2)}}(m,n) = median\,[&S[2]_{(r_2)}(m,n), S[1]_{(r_1)}(m,n), \\
&a(m,n), S[1]_{(N_1-r_1+1)}(m,n), S[2]_{(N_2-r_2+1)}(m,n)]
\end{aligned} \tag{3}
$$

where

$$
S[1]_{(r_1)}(m,n) = r_1^{th} \text{ order statistic of } S[1](m,n) \tag{4}
$$

$$
S[2]_{(r_2)}(m,n) = r_2^{th} \text{ order statistic of } S[2](m,n) \tag{5}
$$

and $1 \le r_1 \le 4$, $1 \le r_2 \le 8$ and N_1 (=8) & N_2 (=16) are the cardinal numbers of the shell $S[1](m,n)$ and $S[2](m,n)$ respectively.

The basic idea of the adaptive median based multi-shell order-statistics filtering is to adaptively select the values of r_1 and r_2 according to the input impulsive noise density so that r_1 and/or r_2 will move up or down depending on whether the input noise density is high or low respectively. The filtering scheme includes a number of steps; the first is to obtain the median $m_{S[j]}$ and the median of the absolute deviations from the median $MAD_{S[j]}$ [9] estimates from $\{a(m_j,n_j)\}$ where j =1,2 and

$$
\{a(m_j,n_j) \in \{S[j] \cup a(m,n)\}; m_j, n_j \in Z\} \tag{6}
$$

$$
MAD_{S[j]} = median[\{\left|a(m_j,n_j) - m_{S[j]}\right|\}] \tag{7}
$$

The second step involves counting the number of samples, $i_{pos[1]}$, in $\{a(m_1,n_1)\}$ that are larger than $[m_{S[1]} + C \cdot MAD_{S[1]}]$, and the number of samples, $i_{neg[1]}$, in $\{a(m_1,n_1)\}$ that are smaller than $[m_{S[1]} - C \cdot MAD_{S[1]}]$.

$$
i_{pos[1]} = \left|\{a(m_1,n_1) : a(m_1,n_1) > [m_{S[1]} + C \cdot MAD_{S[1]}]\}\right| \tag{8}
$$

$$
i_{neg[1]} = \left|\{a(m_1,n_1) : a(m_1,n_1) < [m_{S[1]} - C \cdot MAD_{S[1]}]\}\right| \tag{9}
$$

Similarly, $i_{pos[2]}$ and $i_{neg[2]}$ are obtained from the second shell, $S[2]$, and r_1 and r_2 are then determined as follows:

$$
r_1 = \begin{cases} i_{pos[1]} & \text{if } i_{pos[1]} \ge i_{neg[1]} \\ i_{neg[1]} & \text{if } i_{pos[1]} < i_{neg[1]} \end{cases}; \tag{10}
$$

$$
r_2 = \begin{cases} i_{pos[2]} & \text{if } i_{pos[2]} \ge i_{neg[2]} \\ i_{neg[2]} & \text{if } i_{pos[2]} < i_{neg[2]} \end{cases}; \tag{11}
$$

As a result, r_1 and r_2 will vary adaptively with the number of estimated bad pixels within the filter window. If both r_1 and r_2 are zero, this implies that no bad pixel has been identified and hence no filtering will be applied. However, if only r_1 is zero, the filter output is given as follows:

$$
\begin{aligned}
&y_{m/shell_{(r_1,r_2)}}(m,n) = \\
&\quad median\,[S[2]_{(r_2)}(m,n), a(m,n), S[2]_{(N_2-r_2+1)}(m,n)]
\end{aligned} \tag{12}
$$

On the other hand, if only r_2 is zero, the filter output is:

$$
\begin{aligned}
&y_{m/shell_{(r_1,r_2)}}(m,n) = \\
&\quad median\,[S[1]_{(r_1)}(m,n), a(m,n), S[1]_{(N_1-r_1+1)}(m,n),]
\end{aligned} \tag{13}
$$

In [6], C was defined as a fixed constant value. To preserve the image better, we propose a variation to C so that C will be adaptively adjusted so that only pixel values which are considerably different from their surroundings will be removed.

One problem with impulse noise removal algorithms is that they will remove a pixel which is different from its neighbours. However, some image features will contain pixels which are different from their neighbours and will mistakenly be removed. To preserve an image well, only a pixel which is

considerably different from its surroundings should be considered as a bad pixel and removed, because a local maximum/minimum pixel should not be simply considered as a bad pixel. To reduce false bad pixel detection, we examine the gradients of the centre pixel to that of the surrounding pixels in 8 directions, and the estimated ratio of the gradients is given by the median of the ratios in all directions.

Let Ω_D be the ratio of the gradients at $a(m,n)$, where D represents one of 8 directions, $D \in \{N,S,E,W,NE,NW,SE,SW\}$.

$$\Omega_N(m,n) = \left|\frac{a(m-1,n)-a(m-2,n)}{a(m,n)-a(m-1,n)}\right| \quad (14)$$

$$\Omega_S(m,n) = \left|\frac{a(m+1,n)-a(m+2,n)}{a(m,n)-a(m+1,n)}\right| \quad (15)$$

$$\Omega_E(m,n) = \left|\frac{a(m,n+1)-a(m,n+2)}{a(m,n)-a(m,n+1)}\right| \quad (16)$$

$$\Omega_W(m,n) = \left|\frac{a(m,n-1)-a(m,n-2)}{a(m,n)-a(m,n-1)}\right| \quad (17)$$

$$\Omega_{NE}(m,n) = \left|\frac{a(m-1,n+1)-a(m-2,n+2)}{a(m,n)-a(m-1,n+1)}\right| \quad (18)$$

$$\Omega_{NW}(m,n) = \left|\frac{a(m-1,n-1)-a(m-2,n-2)}{a(m,n)-a(m-1,n-1)}\right| \quad (19)$$

$$\Omega_{SE}(m,n) = \left|\frac{a(m+1,n+1)-a(m+2,n+2)}{a(m,n)-a(m+1,n+1)}\right| \quad (20)$$

$$\Omega_{SW}(m,n) = \left|\frac{a(m+1,n-1)-a(m+2,n-2)}{a(m,n)-a(m+1,n-1)}\right| \quad (21)$$

When Ω_D is small, this implies that the centre pixel is considerably different from its surroundings, and C should be small so that the bound is small enough to exclude the bad pixel. On the other hand, when Ω_D is large, C should be large so that the bound is big enough to include the centre pixel even though it may differ from its surroundings, and be preserved. We propose C as follows:

$$C = K \cdot median\{\Omega_N,\Omega_S,\Omega_E,\Omega_W,\Omega_{NE},\Omega_{NW},\Omega_{SE},\Omega_{SW}\} \quad (22)$$

where K is a constant which determines how the image will be preserved depending on the ratio of the gradients of the centre pixel to that of the surrounding pixels. It has to be found experimentally. A larger K will preserve an image better, while a smaller K will remove a higher density of bad pixels.

III. Weighted Median Based CFA Demosaicking

Various CFA demosaicking techniques [10]-[14] have been proposed to tackle the problems of colour artefacts in the demosaicked image. In this section, we describe the weighted median based CFA demosaicking which can be used to interpolate the missing colour values in the CFA image with minimal colour artefacts while preserving sharp colour edges.

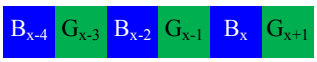

Figure 2. 1D Bayer pattern

By applying the formulas given in [14] to Fig. 2, the missing green pixel value $\hat{G}_x$ at the blue pixel location B_x along the west direction is given by:

$$\hat{G}_x = G_{x-1} + \tfrac{1}{2}(B_x - B_{x-2}) + \tfrac{1}{8}(G_{x+1} - 2G_{x-1} + G_{x-3}) \quad (23)$$

Similarly, the other three estimates for the north, south and east directions can be determined. Likewise, a missing green value at a red pixel position can be evaluated using similar equations.

In order to preserve an edge, a weighted median based classifier is used to process the four estimates because it inherits the robustness and edge preserving capability of a median filter, while offering much greater flexibility in design specifications [15],[16]. For a discrete-time vector $X = [X_1, X_2, ..., X_N]$, the output Y of the WMF of width N associated with the filter weights $W = [W_1, W_2, ..., W_N]$ is given by $Y = \text{MEDIAN}[W_1 \Diamond X_1, W_2 \Diamond X_2, ..., W_N \Diamond X_N]$ where MEDIAN[.] denotes the median operation and $\Diamond$ denotes the duplication operator, i.e. $K \Diamond X = \overbrace{X,...,X}^{K \text{ times}}$ where K is the duplication number.

The weighted median filtering process duplicates each sample according to its corresponding weight, and then selects the median value from the new sequence. As a result the weights of the WMF influence the probability of a particular sample to be chosen as the output. We apply the same criterion in [13] to determine the filter coefficients for the weighted median filter, which is based on the edge orientation map.

IV. Proposed Overall Algorithm

Fig. 3 gives the flowchart of the proposed algorithm. In order for the adaptive filter to work effectively, the CFA image is first demosaicked to produce a full colour image so that the adaptive filter has a full neighbourhood of same colour pixels to process. As only the actual sensor pixels may be defective, the adaptive multi-shell order-statistics filter is not applied to the whole image, but only to actual sensor values in the demosaicked image to detect and remove bad pixels. At these detected bad pixel locations, demosaicking is only re-applied to affected pixels to update the demosaicked image.

V. Results

To assess the performance of our proposed algorithm, we added defective pixels in the form of random impulses of various magnitudes to the CFA image, as shown in Fig. 5b. Our algorithm was tested with CFA images corrupted by bad pixels in order to assess the performance differences in both cases. It was also compared with the algorithm in [2] which is a combined demosaicking and bad pixel correction method.

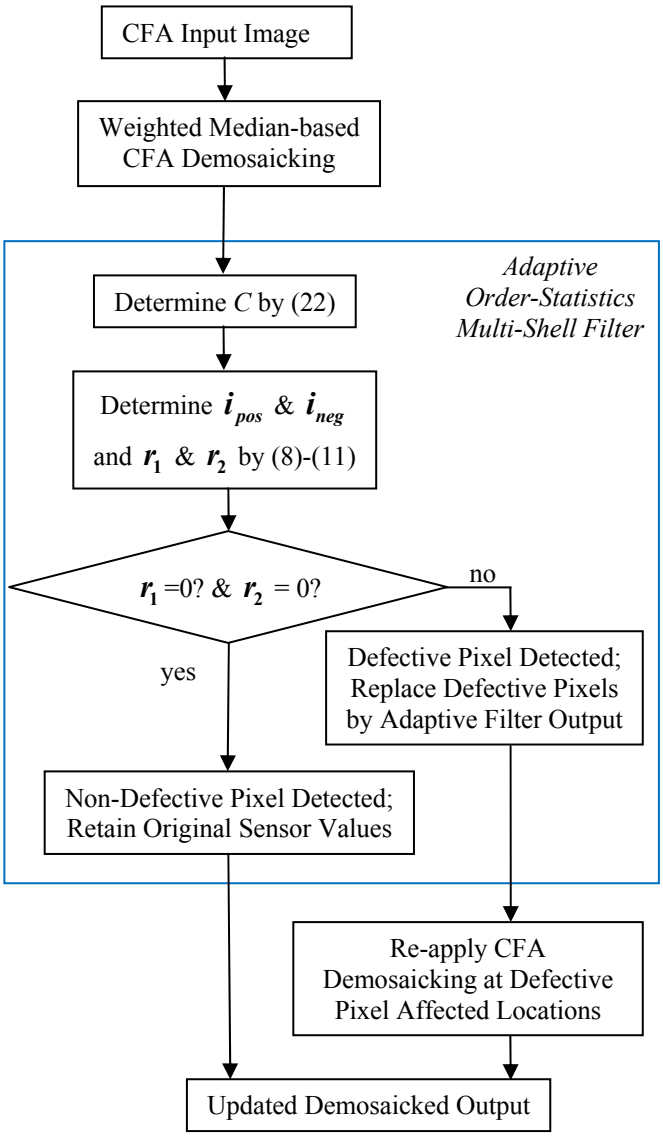

Figure 3. Flowchart of Proposed Algorithm

24 test images with various characteristics in Fig. 4 were selected to evaluate our algorithm. A 5x5 window with 2 shells was used for the adaptive order-statistics filter. In order to give a good balance of feature preservation and bad pixel correction, the value of K was set to 7.

Tables I and II are the PSNR and NCD [17] results when the CFA test images corrupted by 5% bad pixels were used as input. The first column gives the results of our proposed method which combines bad pixel correction within CFA demosaicking. The second column gives the results of the method by Kakarala [2] for comparison. This shows that our proposed algorithm outperforms that of Kakarala's for all the twenty-four test images. Column 3 shows the results of the application of the adaptive multi-shell filter to correct bad pixels after demosaicking. This shows that our proposed method of applying the adaptive multi-shell filter within CFA demosaicking outperforms a separate process of bad pixel correction after CFA demosaicking.

Fig.5(a) is the original image used for visual assessment. Fig.5(b) is the CFA image of Fig.5(a) corrupted with 5% bad pixels. Fig.5(c) is the weighted median based demosaicked output of Fig.5(b) without any filtering. Fig.5(d) gives the demosaicked output of Fig.5(b) using Kakarala's method [2]. Fig.5(e) is the demosaicked output of our proposed algorithm applied to Fig.5(b). It demonstrates that it is capable of removing high density bad pixels whilst preserving details of the image. In comparison, a significant amount of bad pixels still remained in Fig.5(d) produced by Kakarala's method, with colour artifacts and loss in detail. Fig.5(f) is the direct application of the adaptive multi-shell filter to Fig.5(c). While Figs.5(e) and 5(f) show both methods are capable of removing bad pixels very well, however on closer inspection from Figs.6(a) and 6(b), it can be seen our proposed method in Fig.6(a) produced a cleaner image.

VI. Conclusion

Our proposed method combines an adaptive order-statistics multi-shell filter within CFA demosaicking for bad pixel correction. We have shown that this method outperforms a separate process of CFA demosaicking followed by bad pixel removal. The adaptive multi-shell filter changes its bound according to the gradient ratio so that only "bad" pixels which are appreciably different from their surrounding neighbours will be corrected. In this way, our proposed way of combining bad pixel removal within CFA demosaicking has been proved to give superior results in terms of bad pixel removal and detail preservation. In addition, our proposed method is preferred to permanent remapping of hot pixels which only appear in images on a long exposure and/or at high ISO settings. Remapping of hot pixels is indeed sometimes undesirable for those pixels which only appear on a long exposure and/or high ISO settings but still function at normal exposure and ISO settings.

References

[1] B. E. Bayer, "Color Imaging Array," *US Patent 3 971 065*, 1976.
[2] R. Kakarala, "Digital Image System and Method for Combining Demosaicing and Bad Pixel Correction", *US Patent 7,015,961 B2*, 2006.
[3] J.S.J. Li, "A class of multi-shell min/max median filters," *1989 IEEE Symposium on Circuits and Systems Proceedings*, vol. 1, pp. 421-424, 1989.
[4] J.S.J. Li, "Median based feature selective filtering," *SPIE Proceedings of Nonlinear Image Processing*, vol. 1247, pp. 58-69, 1990.
[5] J.S.J. Li and A. Ramsingh, "On the Similarities between the Multistage and the Multi-shell Median filters," *1993 Asia-Pacific Conference on Communications Proceedings*, vol. 2, pp. 544-547, 1993.
[6] J.S.J. Li and A. Ramsingh, "Adaptive median based multi-shell order-statistics filters for the removal of impulsive noise," *Proceedings IEEE/International Symposium on Speech, Image Processing and Neural Networks*, vol. 2, pp. 776-779, 1994.
[7] J.S.J. Li and A. Ramsingh, "Statistical analysis of the median based multi-shell order-statistics filters," *IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings*, vol. 5, pp. V69-V72, 1994.
[8] J.S.J. Li and A. Ramsingh, "The relationship of the Multi-Shell to Multistage and Standard Median Filters," *IEEE Transactions on Image Processing*, vol. 4, no. 8, pp. 1165-1169, 1995.
[9] Y.H. Lee and S.A. Kassam, "Generalised median filtering and related nonlinear filtering techniques," IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-34,pp. 898-911, Aug. 1986.

[10] J. S. J. Li and S. Randhawa, "High Order Extrapolation using Taylor Series for Color Filter Array Demosaicking," *Lecture Notes in Computer Science*, M. Kamel and A. Campilho, Eds., Springer-Verlag, vol. 3656, pp. 703-711, 2005.

[11] S. Randhawa and J. S. J. Li, "CFA Demosaicking with Improved Color Edge Preservation," *Proceedings of IEEE Tencon'05*, 2005.

[12] J. S. J. Li and S. Randhawa, "CFA Demosaicking using Cubic Spline Interpolation," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2007) Proceedings*, pp. I 865-868, 2007.

[13] J. S. J. Li and S. Randhawa, "Weighted Median Based Color Filter Array Demosaicking,", *Image and Vision Computing New Zealand IVCNZ 2008*, IEEE Xplore, pp. 1-6, 2008.

[14] J.S.J. Li and S. Randhawa, "Color Filter Array Demosaicking Using High Order Interpolation Techniques with a Weighted Median Filter for Sharp Color Edge Preservation," IEEE Transactions on Image Processing, vol. 18, no. 9, pp. 1946-1957, 2009.

[15] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted Median Filters: A Tutorial," *IEEE Transactions on Circuits and Systems*, vol. 43, pp.157-192, 1996.

[16] M. K. Prasad, Y. H. Lee. "Analysis of Weighted Median Filters based on inequalities relating the weights", *Circuits Systems Signal Processing*, vol. 11, pp. 115 - 136, 1992.

[17] K. N. Plataniotis and A. N. Venetsanopoulos, *Color Image Processing and Applications*, Springer Verlag, 2000.

TABLE I. IMAGE QUALITY PERFORMANCE MEASURES - PSNR (dB)

Image	Proposed Method	Kakarala's Method	Multishell Filtering
4(a)	31.93	24.66	31.21
4(b)	37.10	31.57	36.57
4(c)	33.39	24.89	32.52
4(d)	34.33	26.16	33.71
4(e)	39.29	32.14	38.56
4(f)	29.09	21.97	27.98
4(g)	38.77	30.96	38.12
4(h)	38.36	30.89	37.42
4(i)	34.70	27.65	33.97
4(j)	39.28	32.30	38.37
4(k)	29.63	22.19	29.25
4(l)	37.59	31.83	37.35
4(m)	37.31	29.96	36.59
4(n)	37.42	30.55	36.90
4(o)	33.42	26.36	33.10
4(p)	31.42	26.30	30.14
4(q)	37.35	30.50	36.80
4(r)	34.57	26.84	34.08
4(s)	34.80	28.76	34.10
4(t)	31.11	24.97	30.65
4(u)	38.73	32.11	38.36
4(v)	33.97	27.53	33.57
4(w)	41.36	33.13	40.98
4(x)	40.41	33.28	39.86

TABLE II. IMAGE QUALITY PERFORMANCE MEASURES - NCD

Image	Proposed Method	Kakarala's Method	Multishell Filtering
4(a)	0.0373	0.0951	0.0402
4(b)	0.0274	0.0489	0.0296
4(c)	0.0470	0.1155	0.0494
4(d)	0.0224	0.0612	0.0239
4(e)	0.0194	0.0414	0.0203
4(f)	0.0368	0.0984	0.0406
4(g)	0.0122	0.0271	0.0128
4(h)	0.0121	0.0254	0.0129
4(i)	0.0321	0.0764	0.0346
4(j)	0.0111	0.0240	0.0118
4(k)	0.0562	0.1245	0.0590
4(l)	0.0277	0.0460	0.0290
4(m)	0.0195	0.0516	0.0210
4(n)	0.0341	0.0668	0.0362
4(o)	0.0509	0.1052	0.0535
4(p)	0.0283	0.0692	0.0309
4(q)	0.0193	0.0358	0.0203
4(r)	0.0264	0.0622	0.0277
4(s)	0.0297	0.0570	0.0313
4(t)	0.0347	0.0728	0.0367
4(u)	0.0274	0.0502	0.0288
4(v)	0.0376	0.0821	0.0395
4(w)	0.0147	0.0245	0.0154
4(x)	0.0149	0.0308	0.0158

Figure 4. Test Images 4(a) – 4(x) from left to right, top to bottom

Figure 5(a). Original Parrot Image

Figure 5(b). CFA Image corrupted with 5% Bad Pixels

Figure 5(c). Demosaicked output without Bad Pixel Correction

Figure 5(d). Demosaicked output using Kakarala's Method [2]

Figure 5(e). Demosaicked output using Our Proposed Method

Figure 5(f). Demosaicked output using CFA Demosaicking followed by Adaptive Order-Statistics Multi-Shell Filter

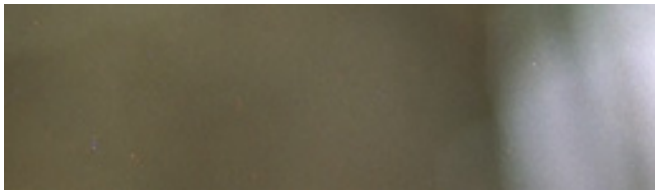

Figure 6(a). Zoom in area above the red parrot of Fig 5(e)

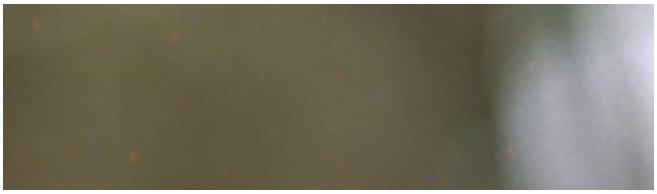

Figure 6(b). Zoom in area above the red parrot of Fig 5(f)