Scientific Research and Essays Vol. 6(26), pp. 5523-5533, 9 November, 2011
DOI: 10.5897/SRE11.856
Available online at http://www.academicjournals.org/SRE
ISSN 1992-2248 © 2011 Academic Journals

Full Length Research Paper

An efficient implementation of switching median filter with boundary discriminative noise detection for image corrupted by impulse noise

Haidi Ibrahim*, Theam Foo Ng and Sin Hong Teoh

School of Electrical and Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, 14300 Nibong Tebal, Penang, Malaysia.

Accepted 18 October, 2011

Switching Median Filter with Boundary Discriminative Noise Detection (BDND) is one of the useful methods that are capable to restore digital images which have been extremely corrupted by universal impulse noise. Following the fundamental framework of the switching median filter, the construction of BDND can be divided into two stages. The first stage classifies the pixels into either "noise" or "noise-free" pixels, while the second stage restores the image by changing only the intensity values of the "noise" pixels. Unfortunately, the originally proposed BDND employs sorting operations in both of its stages. This condition makes the originally proposed BDND computationally expensive. Therefore, in this paper, an implementation of BDND with reduced computational time is suggested. This reduction is achieved mainly by manipulating the local histograms' properties. Experimental results show that the proposed implementation successfully produces the same results as the originally proposed BDND, but with much shorter processing time.

Key words: Digital image processing, impulse noise, median filter.

INTRODUCTION

One of the noises that are commonly corrupting digital images is the impulse noise. This type of noise can be contributed by many factors such as by faulty imaging sensors and from malfunction memory cells in storage devices. Besides, impulse noise also can be created during the transmission of the signal through noisy channel (Chan et al., 2005). In general, the impulse noise can be considered as an additive noise. This noise changes the value of some pixels at random locations into either relatively high or relatively low intensity value. Noise pixels with high intensity values appear as white dots on the image (that is salt), while noise pixels with low intensity values appear as black dots on the image (that is pepper). Therefore, impulse noise is also named as the salt-and-pepper noise (Petrou and Bosdogianni, 2000). As the impulse pixels are having a relatively high contrast toward their surrounding, even at low percentage of corruption, the impulse noise can degrade the appearance of the image significantly (Ibrahim et al., 2008). Therefore, it is crucial for us to remove the impulse noise before any subsequent image processing operations such as image segmentation and pattern recognition. Commonly, median filter which is a nonlinear filter is employed to reduce the impulse noise in digital images due to its sensitivity towards outliers (Eng and Ma, 2001). The standard median filter (SM) operates by defining a contextual region by using a sliding window of size $W \times W$. It replaces the intensity value of the centre

*Corresponding author. E-mail: haidi_ibrahim@ieee.org.

pixel with the median value calculated from the samples within the defined contextual region. However, there are a few drawbacks associated with SM. SM is not able to filter the image with an extremely high level of impulse noise. Besides, SM also does not differentiate uncorrupted pixels from corrupted pixels, and thus changes all the pixel intensity values of the image. Furthermore, SM also cannot perform very well in preserving lines and edges. Therefore, several variations of median filter have been introduced such as the detail-preserving median filter (DMF) (Sun and Neuvo, 1994), tri-state median filter (TSMF) (Chen et al., 1999), progressive switching median filter (PSMF) (Wang and Zhang, 1999), classifier-augmented median filter (Chang and Chen, 2004), switching median filter with boundary discriminative noise detection (BDND) (Ng and Ma, 2006), fuzzy median filter (FMF) (Luo, 2006), multichannel weighted median filter (MWM) (Li et al., 2006), difference-type noise detector for adaptive median filter (DNDAM) (Yuan and Tan, 2006), 3D median filter (Jiang and Crookes, 2006), directional weighted median filter (DWM) (Dong and Xu, 2007), simple adaptive median filter (SAMF) (Ibrahim et al., 2008), fuzzy switching median filter (FSM) (Toh et al., 2008), switching median filter with highly effective impulse noise detection algorithm (Duan and Zhang, 2010), new adaptive switching median filter (Akkoul et al., 2010) and modified decision based un-symmetric trimmed median filter (MDBUTMF) (Esakkirajan et al., 2011). Among these new median filtering techniques, BDND has been claimed to have a good performance in reducing the universal impulse noise. Although BDND can reduce four types of impulse noise; this method requires a long processing time due to a few factors. Therefore, in this paper, an alternative and efficient implementation of BDND which requires much shorter processing time is suggested.

The rest of this paper is organized as follows: first, we explain the impulse noise model used in this paper; after that we describe the construction of the originally proposed BDND; then, we present our approach in implementing BDND; next, we compare the performance of this new implementation towards the originally proposed BDND and finally, we conclude our finding.

NOISE MODEL

Although the work by Ng and Ma (2006) used four impulse noise models; in this paper only one impulse noise model has been implemented. This is sufficient for this research because the main problem that we want to tackle in this paper is the high computational time associated with BDND and not the noise filtering ability. To simplify the explanations of this noise model, lets consider $p(i)$ as the probability density function of intensity i in an image of size $M \times N$ pixels and the level of impulse noise that corrupting the image is 100 P%, where $0 \le P \le 1$. The noise model that we used is the "noise Model 1" in Ng and Ma (2006). Some examples of image corrupted by this type of noise are shown in Figure 1. If the image is an eight-bit depth grayscale image, this noise model assumes that the impulse noise is being presented by only two intensity values which is either intensity 0 or intensity 255. Pepper is being presented by intensity 0, while salt is being presented by intensity 255. In this noise model, the number of salts is assumed to be equal to the number of peppers that degraded the image; this model can be described by the following equation:

$$p(i) = \begin{cases} 0.5P & : \quad \text{pepper}; i = 0 \\ 1-P & : \quad \text{noise free pixels}; 0 \le i \le 255 \\ 0.5P & : \quad \text{salt}; i = 255 \end{cases} \quad (1)$$

ORIGINALLY PROPOSED BDND

Similar to other switching median methods, the construction BDND can be divided into two stages. The first stage is the detection stage where the pixels are classified into either "noise" or "noise free" pixels. The second stage of the method is the noise correction stage where the "noise" pixels are fixed. In the originally proposed BDND by Ng and Ma (2006), the detection stage has eight steps:

Step 1

Define a contextual region around the current pixel by using a local window of size 21 × 21.

Step 2

Sort the pixels in the window in ascending order to create the sorted vector v_o. Find the median value of v_o, *med*.

Step 3

Create vector v_D which contains the different value between each pair of adjacent pixels in v_o.

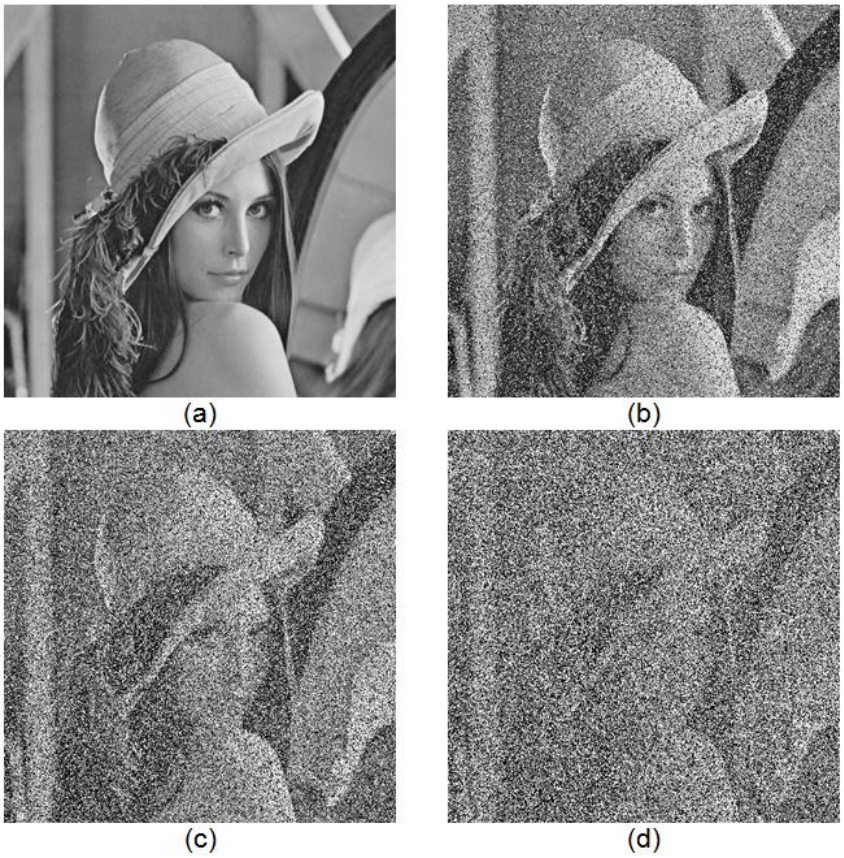

Figure 1. The original, clean Lena image (a), Lena image corrupted by 25% of impulse noise (b), Lena image corrupted by 50% of impulse noise (c) and Lena image corrupted by 75% of impulse noise (d).

Step 4

For the intensity value between 0 and *med*, find the maximum value of v_D. The corresponding pixel in v_o is defined as the boundary b_1.

Step 5

For the intensity value between *med* and 255, find the maximum value of v_D. The corresponding pixel in v_o is defined as the boundary b_2.

Step 6

The middle cluster is defined as the range from b_1 to b_2. If the current pixel falls within this range, it is considered as "noise free" pixel and the classification process for this pixel is completed. Else, the pixel will be further investigated by using Step 7.

Step 7

Define a contextual region around the current pixel by using a local window of size 3 × 3 and repeat Steps 2 to 5.

Step 8

The middle cluster is defined as the range from b_1 to b_2. If the current pixel falls within this range, it is considered as "noise free" pixel. Else, the pixel is considered as "noise" pixel.

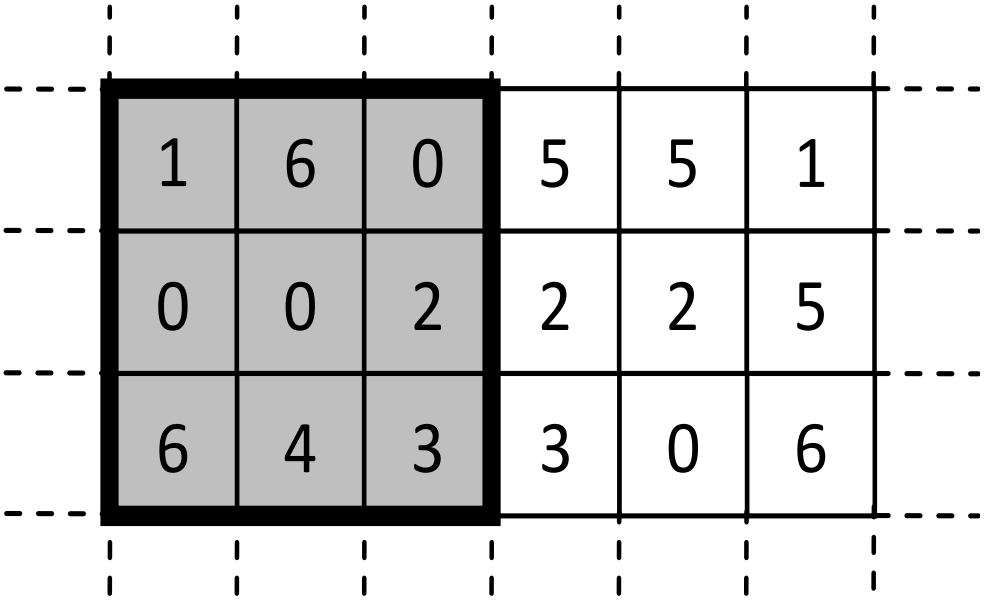

Figure 2. A portion of an image that is being filtered by using a median filter of size 3 × 3.

The noise correction stage of BDND consists of two steps. In the first step, the maximum filtering window size, $W_M \times W_M$ has to be determined based on the global noise density estimation. The maximum window size is limited to the size of 7 × 7 pixels. In the second step, only "noise" pixels will be filtered while the "noise free" pixels are remain unaffected. For each "noise" pixel starting by using a filtering window of size 3 × 3, the number of "noise free" pixels N is calculated. If N is less than half of the area covered by the current filtering window, the size of the filtering window is extending one pixel in all the four sides of the window. The expansion of size of the filtering window will stop when N is greater or equal to the half of the filtering window area or when the filtering window has a size of $W_M \times W_M$. However, if there is still not a single "noise free" sample exists within the filtering window, although the size of the filtering window already reach $W_M \times W_M$, BDND will then continue to expand until at least it has one "noise free" pixel within its contextual region. After that, the filtering process is done by finding the median value from the sorted "noise free" pixels within the window area.

NEW IMPLEMENTATION OF BDND

A long processing time is required by the originally proposed BDND mainly due to the following three factors:

1) In both noise detection and noise correction stages, the sorting operations are used to find the median value (some of the sorting algorithms are presented in Moses (2009).
2) In noise detection stage, BDND uses two detection windows to classify the input pixels to either "noise" or "noise free" pixels. The primary detection window is a big window which is 21 × 21 pixels and thus requires long processing time.
3) In noise correction stage, BDND employs an adaptive filtering. The filtering window is not fixed but changing accordingly to the local noise density.

In order to reduce the problems as pointed by points 1 and 2, a similar work to Kong and Ibrahim (2010) is used in this paper. With the intention of avoiding the usage of sorting operations, the method employed in Huang et al. (1979) which exploits local histogram is utilized here to find the median value. To ease our understanding, lets consider the portion of the image as shown in Figure 2 which is being filtered by a median filter of size 3 × 3. The black thick square region presents the contextual region defined by this sliding window. The values shown in the figure present the pixel intensity values. At this filter's position, a local histogram $h(i)$ as shown in Figure 3a is created. Histogram $h(i)$ presents the number of occurrences of intensity value i within the contextual region. Then, a local cumulative density function (cdf) as

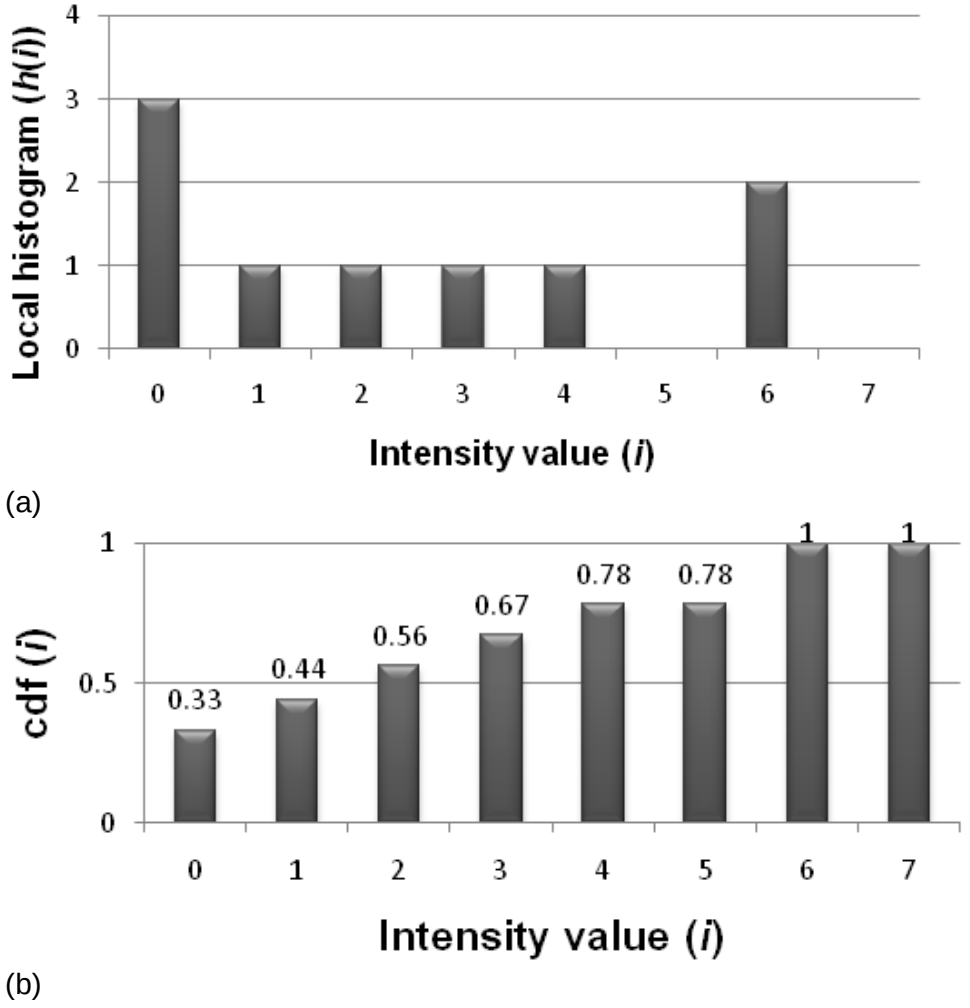

Figure 3. The local histogram defined by the window shown in Figure 2 (a) and the corresponding cumulative density function cdf (b).

shown in Figure 3b is created. The cdf is defined as:

$$\mathrm{cdf}(i) = \frac{1}{T}\sum_{j=0}^{i} h(j) \qquad (2)$$

Where T is the number of samples within the contextual region that contribute to the calculation of the median value (in this example, $T = 3 \times 3 = 9$). The median value is identified as the intensity value i where the cdf reaching the value is equal or greater than 0.5. For the example shown in Figure 3b, the median value is equal to 2.

The usage of local histograms can avoid us from using sorting operations and thus can save some of our processing time, especially if the image is being filtered by a filter with big dimensions (for example 21 × 21 pixels). Besides, the creation of local histogram is simple if the filter is moved one pixel per time. Figure 4 shows the same image portion as shown in Figure 2, but when the filter is moved one pixel to the right. Figure 5 shows the common region and the differences between Figures 2 and 4. Using the example shown by these figures, the samples defined by the contextual region in Figure 4 actually can be initiated base on the samples from Figure 2. The samples in Figure 4 are defined as the samples in Figure 2 with added samples from Figure 5c and deducted samples from Figure 5b. The samples in Figure 2 are (0, 0, 0, 1, 2, 3, 4, 6, 6). Therefore, the samples in

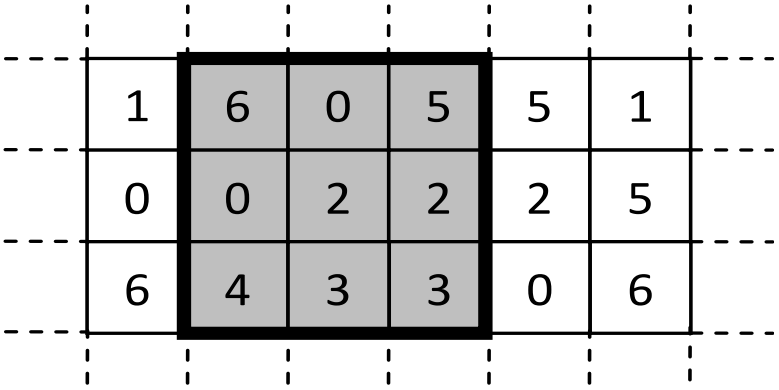

Figure 4. The image in Figure 2 when the filter is shifted one pixel to the right.

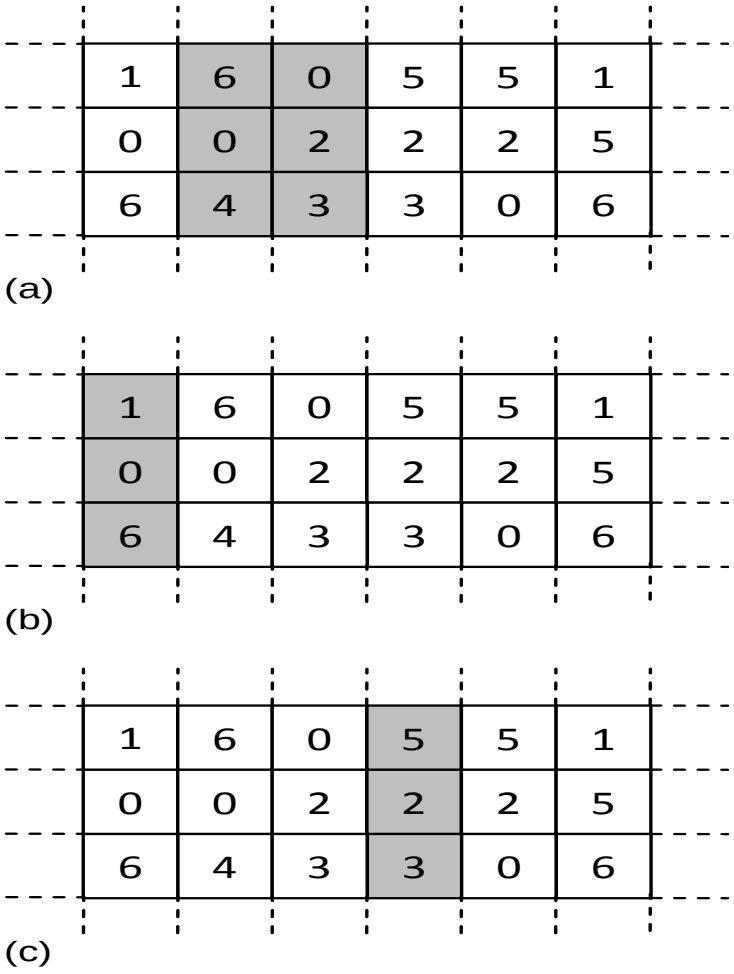

Figure 5. The common region between Figures 2 with 4 (a), the portion of the contextual region that only belongs to Figure 2 (b) and the portion of the contextual region that only belongs to Figure 4 (c).

Table 1. Suggested number of local histograms based on global noise density estimation η.

Noise density	Number of local histograms
$0\% \le \eta \le 20\%$	3 local histograms (corresponding to windows of size 3 × 3, 5 × 5 and 7 × 7)
$20\% < \eta \le 40\%$	4 local histograms (corresponding to windows of size 3 × 3, 5 × 5, 7 × 7 and 9 × 9)
$40\% < \eta \le 70\%$	5 local histograms (corresponding to windows of size 3 × 3, 5 × 5, …, 11 × 11)
$\eta > 70\%$	13 local histograms (corresponding to windows of size 3 × 3, 5 × 5, …, 27 × 27)

Figure 4 can be defined as (0, 0, 0, 1, 2, 3, 4, 6, 6) + (5, 2, 3) – (1, 0, 6) = (0, 0, 2, 2, 3, 3, 4, 5, 6). Thus, the histogram can be updated by considering only two columns at one time and therefore saving a lot of processing time. The processing time can be reduced further if the window slides continuously, where the current local histogram can be created by the previous local histogram. Therefore, in our implementation, the window slides from left to right for the odd rows and from right to left for the even rows. The calculation of median value by using this method is faster than when we create new local histogram from the whole contextual region. Therefore, in our implementation of BDND, the detection stage has been reduced into seven steps as follows:

Step 9

Create two local histograms h_1 and h_2 which are defined by contextual regions around the current pixel, corresponds to window of size 21 × 21 and window of size 3 × 3 by using the aforementioned method (histogram h_2 is still need to be created, although it might not be used in certain condition in order to ensure the continuity of the window's sliding path).

Step 10

Find the median value, *med* from h_1.

Step 11

For the intensity value between 0 and *med* in h_1, find the largest histogram's bins gap. The intensity value corresponds to this condition is defined as the boundary b_1.

Step 12

For the intensity value between *med* and 255 in h_1, find the largest histogram's bins gap. The intensity value corresponds to this condition is defined as the boundary b_2.

Step 13

The middle cluster is defined as the range from b_1 to b_2. If the current pixel falls within this range, it is considered as "noise free" pixel and the classification process for this pixel is completed. Otherwise, the pixel will be further investigated by using Step 14.

Step 14

Repeat Steps 10 to 13 by using h_2.

Step 15

The middle cluster is defined as the range from b_1 to b_2. If the current pixel falls within this range, it is considered as "noise free" pixel; otherwise, the pixel is considered as "noise" pixel.

It is worth noting that the originally proposed BDND which employs sorting operations requires eight steps in its noise detection stage. On the other hand, our implementation of BDND by using local histogram only requires seven steps. This suggests that our proposed implementation is faster than the original BDND. To tackle the third problem associated with BDND which is regarding to the expansion of filter at each "noise" pixel locations, in our implementation of BDND, regardless whether the current pixel is "noise" or "noise free" pixel, a few local histograms are created. These local histograms are associated together with their corresponding total number of "noise free" samples, *T*. The total number of local histograms used is decided based on the global noise density estimation η, this is given in Table 1.

Following the rules set by the original BDND, in our

Figure 6. Image Lena corrupted by 25% of impulse noise (that is P = 0.25) (a), output from SM with size 7 × 7 (b) and output from BDND using sorting operations (c) and output from BDND using local histograms (d).

implementation, the local histogram that is used for the filtering process is selected based on T.

EXPERIMENTAL RESULTS

Here, the originally proposed BDND by Ng and Ma (2006) is referred as BDND (sorting) while our implementation of BDND is referred as BDND (histogram). The performance of BDND (histogram) was compared with BDND (sorting) and SM (with filter size 7 × 7 pixels). For both BDND (sorting) and SM, the median values were found by using the sorted pixel values. In this paper, the bubble sort which is a well known sorting technique has been implemented in both BDND (sorting) and SM. The grayscale image of "Lena" with size 512 × 512 pixels as shown in Figure 1a was used for the evaluation purpose. This image was then corrupted by impulse noise from level 0 to 95% with the incremental step in corruption is set to 1%. Next, each corrupted version was filtered by using SM, BDND (sorting) and BDND (histogram). The filtered images are compared based on visual inspection, root mean square error (RMSE) value and processing time. Figures 6 to 8 shows some of the filtered outputs. As shown by these figures, the appearance of the outputs obtained from BDND (histogram) is same as the outputs from BDND (sorting). Unlike SM, both implementations are able to reduce impulse noise level even when the noise level is high. Furthermore, these figures also show that BDND produces sharper images as compared with SM. In this paper, the size of the input image is 512 × 512 pixels. Thus, the RMSE value is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{512^2}\sum_{x=0}^{511}\sum_{y=0}^{511}\left|f(x,y)-g(x,y)\right|^2} \quad (3)$$

Where g is the original clean image, f is the output from the restoration process and (x, y) are the spatial coordinates. The smaller the value of RMSE, the better

Figure 7. Image Lena corrupted by 50% of impulse noise (that is P = 0.50) (a), output from SM with size 7 × 7 (b), output from BDND using sorting operations (c) and output from BDND using local histograms (d).

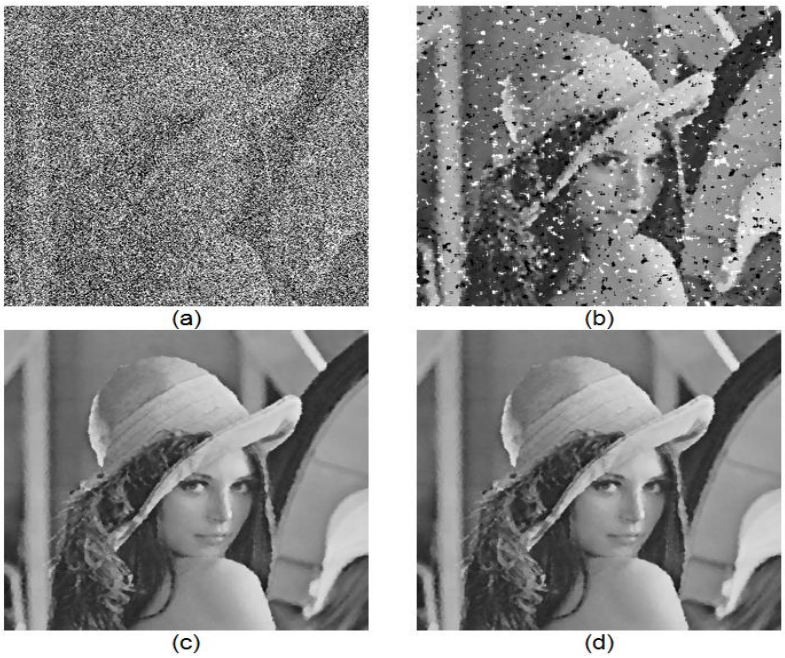

Figure 8. Image Lena corrupted by 75% of impulse noise (that is P = 0.75) (a), output from SM with size 7 × 7 (b), output from BDND using sorting operations (c) and output from BDND using local histograms (d).

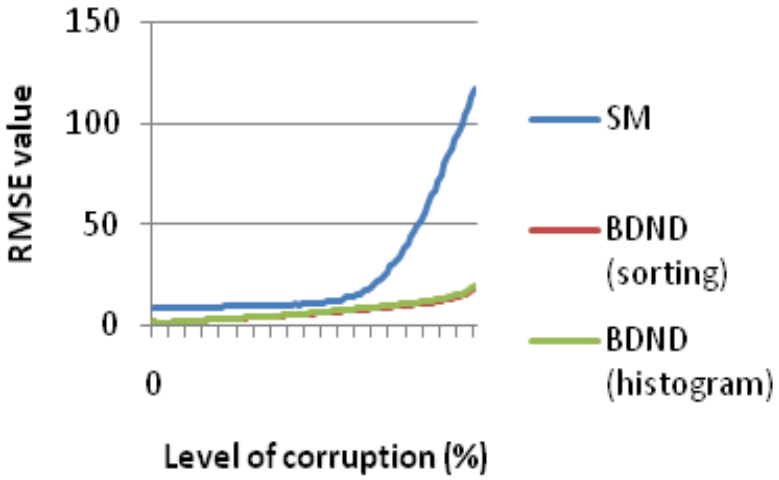

Figure 9. The graph of RMSE value versus level of corruption in Lena image.

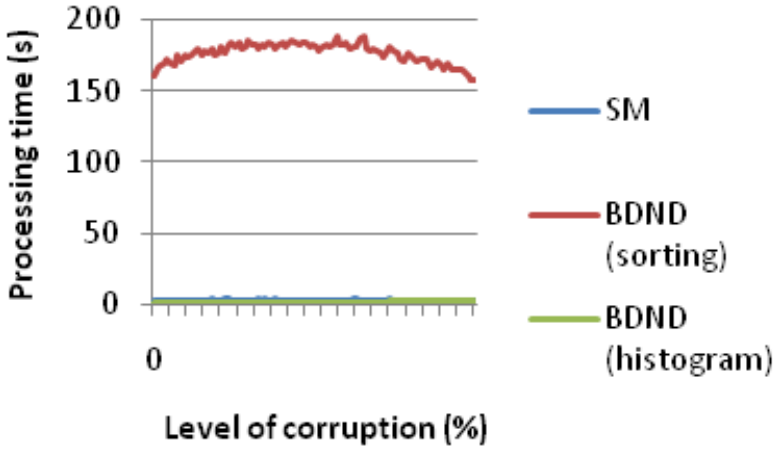

Figure 10. The graph of processing time (in seconds) versus level of corruption in Lena image.

the restoration is. Figure 9 shows the graph of RMSE value versus the level of corruption. As shown by this graph, the plots of BDND (sorting) and BDND (histogram) are overlapped with each other. This suggests that both implementations produce the same outputs. We also can see that the RMSE value of BDND is lower than the RMSE value of SM. This shows that BDND has better restoration ability as compared with SM.

In order to compare the output from BDND (sorting) with the output from BDND (histogram), we use the image similarity measure (ISM) as follows:

$$\mathrm{ISM} = \sum_{x=0}^{511}\sum_{y=0}^{511} \left| f_1(x,y) - f_2(x,y) \right| / (512^2) \quad (4)$$

Where f_1 is the output from BDND (sorting) and f_2 is the output from BDND (histogram). In this paper, we found that ISM is always equal to zero; indicates that the BDND (histogram) produce exactly the same outputs as what have been produced by BDND (sorting).

Figure 10 shows the graph of processing time versus the level of corruption. As shown by this graph, the

processing time required by BDND (sorting) is much higher than what is needed by BDND (histogram) and SM. BDND (sorting) requires between two to three minutes to completely process the image. On the other hand, BDND (histogram) requires only less than two seconds to process and produce exactly the same result. Therefore, this shows that the usage of local histograms in finding the median values can significantly reduce the processing time.

Conclusion

In this paper we presented an effective implementation of BDND. This implementation obtained the median values in both noise detection and noise cancellation stages through local histograms. The reduction in the processing time is achieved by manipulating the creation of local histograms and the window's movement during filtering process. Although several versions of local histogram are needed in this implementation, this approach significantly reduces the computational time needed by BDND.

ACKNOWLEDGEMENTS

The authors would like to thanks the reviewers for their constructive comments. The authors also would like to thank Mr Nicholas Sia Pik Kong and Dr Derek Chan Juinn Chieh for proof reading this manuscript. This work was supported in part by the Universiti Sains Malaysia's Short Term Research Grant with account number 304/PELECT/60311013.

REFERENCES

Akkoul S, Ledee R, Leconge R, Harba R (2010). A new adaptive switching median filter. IEEE Signal Process. Lett., 17(6): 587-590.

Chan RH, Ho CW, Nikolova M (2005). Salt-and-pepper noise removal by median-type noise detectors and details preserving regulation. IEEE Trans. Image Process., 14(10): 1479-1485.

Chang JY, Chen JL (2004). Classifier-augmented median filters for image restoration. IEEE Trans. Instrum. Meas., 53(2): 351-356.

Chen T, Ma KK, Chen LH (1999). Tri-state median filter for image de-noising. IEEE Trans. Image Process., 8(12): 1834-1838.

Dong Y, Xu S (2007). A new directional weighted median filter for removal of random-valued impulse noise. IEEE Signal Process. Lett., 14(3): 193-196.

Duan F, Zhang YJ (2010). A highly effective impulse noise detection algorithm for switching median filters. IEEE Signal Process. Lett., 17(7): 647-650.

Eng HL, Ma KK (2001). Noise adaptive soft-switching median filter. IEEE Trans. Image Process., 10(2): 242-251.

Esakkirajan S, Veerakumar T, Subramanyam AN, PremChand CH (2011). Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter. IEEE Signal Process. Lett., 18(5): 287-290.

Huang TS, Yang GJ, Tang GY (1979). A fast two-dimensional median filtering algorithm. IEEE Trans. Acoust. Speech, Signal Process., 27(1): 3-18.

Ibrahim H, Kong NSP, Ng TF (2008). Simple adaptive median filter for the removal of impulse noise from highly corrupted images. IEEE Trans. Consum. Elect., 54(4): 1920-1927.

Jiang M, Crookes D (2006). High-performance 3D median filter architecture for medical image despeckling. Electron. Lett., 42(24): 1379-1380.

Kong NSP, Ibrahim H (2010). Multiple layers block overlapped histogram equalization for local content emphasis. Comput. Electr. Eng. (Article in Press) Doi:10.1016/j.compeleceng.2010.12.001.

Li Y, Arce GR, Bacca J (2006). Weighted median filters for multichannel signals. IEEE Trans. Signal Process., 54(11): 4271-4281.

Luo W (2006). Efficient removal of impulse noise from digital images. IEEE Trans. Consum. Electron, 52(2): 523-527.

Moses OO (2009). Improving the performance of bubble sort using a modified diminishing increment sorting. Sci. Res. Essays, 4(8): 740-744.

Ng PE, Ma KK (2006). A switching median filter with boundary discriminative noise detection for extremely corrupted images. IEEE Trans. Image Process, 15(6): 1506-1516.

Petrou M, Bosdogianni P (2000). Image Processing: The Fundamentals. England, Wiley.

Sun T, Neuvo Y (1994). Detail-preserving median based filters in image processing. Pattern Recognit. Lett., 15(4): 341-347.

Toh KKV, Ibrahim H, Mahyuddin MN (2008). Salt-and-pepper noise detection and reduction using fuzzy switching median filter. IEEE Trans. Consum. Elect., 54(4): 1956-1961.

Wang Z, Zhang D (1999). Progressive switching median filter for the removal of impulse noise from highly corrupted images. IEEE Trans. Circuits Sys. II., Analog Digit. Signal Process., 46(1): 78-80.

Yuan SQ, Tan YH (2006). Difference-type noise detector for adaptive median filter. Elect. Lett., 42(8): 454-455.