ISSN: 0975-833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 7, Issue, 01, pp.11475-11485, January, 2015

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

REMOVAL OF NOISE IN IMAGES USING MDBUT FILTER

[1]Satti Babu, V. and [2]Vasavi Sridevi, C.H.

[1]Department of ECE, Avanthi Institute of Engineering and Technology, Tamaram, Makavarapalem, Narsipatnam, Visakapatnam Dist, A.P, India

[2]Department of ECE, Avanthi Institute of Engineering and Technology, Tamaram, Makavarapalem, Narsipatnam, Visakapatnam Dist, A.P, India

ARTICLE INFO

Article History:
Received 15th October, 2014
Received in revised form
21st November, 2014
Accepted 18th December, 2014
Published online 23rd January, 2015

Key words:
Embedded Controlled Sensor Networks, Environment Monitoring System

ABSTRACT

It is important to remove or minimize the degradations, noises in valuable ancient blurred color images. The traditional available filtering methodologies are applicable for fixed window dimensions only these are not applicable for varying scale images. In our project we propose a new technique for digital image restoration, in this the noise free and noisy pixels are classified based on empirical multiple threshold values. Then the median filtering technique is applied. So that noise free pixels are getting preserved and only noisy pixels get restored. In this project, median filter, called the multiple thresholds switching (MTS) filter, is proposed to restore images corrupted by salt-pepper impulse noise. The filter is based on a detection-estimation strategy. The impulse detection algorithm is used before the filtering process, and therefore only the noise-corrupted pixels are replaced with the estimated central noise-free ordered mean value in the current filter window. The new impulse detector, which uses multiple thresholds with multiple neighborhood information of the signal in the filter window, is very precise, while avoiding an undue increase in computational complexity. For impulse noise suppression without smearing fine details and edges in the image, extensive experimental results demonstrate that our scheme performs significantly better than many existing, well-accepted decision-based methods.

INTRODUCTION

During image acquisition or transmission, digital images are often contaminated by impulse noise due to a number of non-idealities in the imaging process. The noise usually corrupts images by replacing some of the pixels of the original image with new pixels having luminance values near or equal to the minimum or maximum of the allowable dynamic luminance range. In the most applications, it is very important to remove impulse noise from image data, since the performances of subsequent image processing tasks are strictly dependent on the success of image noise removal operation. However, this is a difficult problem in any image processing system because the restoration filter must not distort the useful information in the image and preserve image details and texture while removing the noise. The intensity of impulse noise has the tendency of being either relatively high or relatively low. Thus, it could severely degrade the image quality and cause great loss of information details. So it is important to eliminate noise in the images before some subsequent processing, such as edge detection, image segmentation and object recognition. Two types of impulse noise can be modelled: i) Fixed valued impulse noise (salt and pepper) and ii) Random valued impulse noise. Various filtering techniques have been proposed for removing impulse noise in the past, and it is well-known that linear filters could produce serious image blurring. As a result, nonlinear filters have been widely exploited due to their much improved filtering performance, in terms of impulse noise attenuation and edge/details preservation. One of the most popular and robust nonlinear filters is the *standard median* (SM) filter, which exploits the rank-order information of pixel intensities within a filtering window and replaces the center pixel with the median value.

Due to its effectiveness in noise suppression and simplicity in implementation, various modifications of the SM filter have been introduced, such as the *weighted median* (WM) filter and the *center weighted median* (CWM) filter. Conventional median filtering approaches apply the median operation to each pixel unconditionally, that is, without considering whether it is uncorrupted or corrupted. As a result, even the uncorrupted pixels are filtered, and this causes image quality degradation. An intuitive solution to overcome this problem is to implement an impulse-noise detection mechanism prior to filtering; hence, only those pixels identified as "corrupted"

Corresponding author:Satti Babu, V.
Department of ECE, VLSI, Avanthi Institute of Engineering and Technology, Tamaram,Makavarapalem,Narsipatnam, Visakapatnam dist, A.P, India.

would undergo the filtering process, while those identified as "uncorrupted" would remain intact. By incorporating such noise detection mechanism or "intelligence" into the median filtering framework, the so-called *switching median filters* had shown significant performance improvement.

Existing Method

Mean Filter

Process

- It checks for the pixels that are noisy in input image i.e. pixels with the values 0 or 255 are to be considered.
- For each such noisy pixel P, a window size of 3x3 neighbouring the pixel P is taken.
- Find the absolute differences between the pixel P and the neighbouring pixels of P.
- The arithmetic mean of the differences for a given pixel P is calculated.
- The arithmetic mean is then compared with the threshold value to detect whether the pixel P is signal pixel or corrupted by noise.
- If the arithmetic mean is greater than or equal to the threshold value the pixel P is considered as noisy.
- Otherwise the pixel P is considered as signal pixel.

Proposed Method

Median Filter

Process

- Noise is detected by the noise detection algorithm mentioned above.
- Filtering is applied only at those pixels that were detected as noisy.
- Once a given pixel P is found to be noisy the following steps are applied
- A 3x3 mask is centred at the pixel P and founds if there exists at least one signal pixel around the pixel P.
- If found, the pixel P is replaced by the median of the signal pixels found in 3x3 neighbourhood of P.
- The above steps are repeated if noise still there in the output image for better results.

MDBUT Median filter

NOISE

In common use the word noise means unwanted sound or noise pollution. In electronics noise can refer to the electronic signal corresponding to acoustic noise (in an audio system) or the electronic signal corresponding to the (visual) noise commonly seen as 'snow' on a degraded television or video image. In signal processing or computing it can be considered data without meaning; that is, data that is not being used to transmit a signal, but is simply produced as an unwanted by product of other activities. In information theory, however, noise is still considered to be information.

In a broader sense, film grain or even advertisements in web pages can be considered noise. Noise can block, distort, or change the meaning of a message in both human and electronic communication. In many of these areas, the special case of thermal noise arises, which sets a fundamental lower limit to what can be measured or signalled and is related to basic physical processes at the molecular level described by well known simple formulae.

General Concept of Salt and Pepper Noise

In the following examples, images have been corrupted with various kinds and amounts of drop-out noise. In, pixels have been set to 0 or 255 with probability p=1%. In pixel bits were flipped with p=3%, and in 5% of the pixels (whose locations are chosen at random) are set to the maximum value, producing the snowy appearance. For this kind of noise, conventional low pass filtering, *e.g.* mean filtering or Gaussian smoothing is relatively unsuccessful because the corrupted pixel value can vary significantly from the original and therefore the mean can be significantly different from the true value.

Median filter removes drop-out noise more efficiently and at the same time preserves the edges and small details in the image better. Conservative smoothing can be used to obtain a result which preserves a great deal of high frequency detail, but is only effective at reducing low levels of noise. The proposed Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) algorithm processes the corrupted images by first detecting the impulse noise. The processing pixel is checked whether it is noisy or noisy free. That is, if the processing pixel lies between maximum and minimum gray level values then it is noise free pixel, it is left unchanged. If the processing pixel takes the maximum or minimum gray level then it is noisy pixel which is processed by MDBUTMF. Unsymmetric Trimmed Median Filter (MDBUTMF) algorithm removes this drawback at high noise density.

If the processing pixel value is 0 or 255 it is processed or else it is left unchanged. At high noise density the median value will be 0 or 255 which is noisy. In such case, neighboring pixel is used for replacement. This repeated replacement of neighboring pixel produces streaking effect. In order to avoid this drawback, Decision Based Unsymmetric Trimmed Median Filter (DBUTMF) is proposed. At high noise densities, if the selected window contains all 0's or 255's or both then, trimmed median value cannot be obtained. So this algorithm does not give better results at very high noise density that is at 80% to 90%. The proposed Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) algorithm removes this drawback at high noise density.

MDBUT MEDIAN FILTER

MDBUT Median Filter can be called as Modified Decision Based Unsymmetric Trimmed. It processes the corrupted images by first detecting the impulse noise. Whether it is noisy or noisy free can be checked by using the processing pixel. It can say it is noise free pixel if the processing pixel lies between maximum and minimum gray level values otherwise

it can say that it is noisy pixel. In many practical cases of image processing, only a noisy image is available. This circumstance is known as the blind condition. Many denoising methods usually require the exact value of the noise distribution as an essential filter parameter. So, the noise estimation methods in the spatial domain use the variance or standard deviation to estimate the actual added noise distribution. But it is found that the mean deviation provides better results than the variance or standard deviation to estimate the noise distribution. The advantage of this approach is that the mean deviation is actually more efficient than the standard deviation in practical situations. The standard deviation emphasizes a larger deviation; squaring the values makes each unit of distance from the mean exponentially (rather than additively) larger. The larger deviation will cause overestimation or underestimation of the noise. So, we assume that use of the mean deviation may contribute to more accurate noise estimation. Keeping these points in view, the authors have used the mean deviation parameter in deciding the noise pixel and replaced the central pixel by its mean deviation instead of its mean. The steps in the proposed algorithm are given below.

Step1: Select 2-D window of size 3x3.

Step2: If this pixel value lies between 0 and 255, pixel. So, no processing is required and its value is left unchanged

Step 3: If Pij = 0 or 255, it indicates that the pixel is corrupted by salt and pepper noise .Here two cases are considered

Case i: The selected window contains few 0 or 255 elements and other elements lie between 0 and255.Thenthe 0 and 255 elements are discarded and the median of the remaining elements is found. The Pij pixel is replaced with this median value.

Case ii: Suppose the window under consideration has all the elements either 0 or 255 .Then median of the elements may also be either 0 or 255 which is again a noisy element. Now, find the mean deviation or absolute mean deviation of the window which can never be 0 or 255.Replace the pixel Pij with this mean deviation value

Step 4: Apply the steps 1 to 3 for all the pixels in the image for complete processing.

Case* (I)*: If the selected window contains salt/pepper noise as processing pixel
(i.e., 255/0 pixel value) and neighbouring pixel values contains all pixels that adds salt and pepper noise to the image:

Where 255 is processing pixel P (i,j)

0	255	0
0	255	255
255	0	255

Since all the elements surrounding are 255's and 0's. It will be either 0 or 255 which is again noisy, if one takes the median value. To solve this problem, the processing pixel is replaced by the mean value and the mean of the selected window is found. Here the mean value is 170.
Replace the processing pixel by 170.

Case (ii): If the selected window contains salt or pepper noise as processing pixel (i.e., 255/0 pixel value) and neighbouring pixel values contains some pixels that adds salt (i.e., 255 pixel value) and pepper noise to the image:

Where 0 is processing pixel P (i,j)

78	90	0
120		255
97	255	73

Now eliminate the salt and pepper noise from the selected window. That is, elimination of 0's and 255's. The 1-D array of the above matrix is [78 90 0 120 0 255 97 255 73]. After elimination of 0's and 255's the pixel values in the selected Window will be [78 90 120 97 73]. Here the median value is 90. Hence replace the processing pixel by 90.

Case (iii): If the selected window contains a noise free pixel as a processing pixel, it does not require further processing. For example, if the processing pixel is 90 then it is noise free pixel: Since "90" is a noise free pixel it does not require further processing.

90	95	90
45	65	21
32	90	101

Each and every pixel of the image is checked for the presence of salt and pepper noise. Different cases are illustrated in this Section. If the processing pixel is noisy and all other pixel values are either 0's or 255's is illustrated in Case (i). If the processing pixel is noisy pixel that is 0 or255 is illustrated in Case (ii). If the processing pixel is not noisy pixel and its value lies between 0 and 255 is illustrated in Case (iii).

A Modified Decision Based Unsymmetrical Trimmed Median Filter

A Modified Decision Based Unsymmetrical Trimmed Median Filter (MDBUTMF) is proposed for the restoration of color images that are highly corrupted by salt and pepper noise.
The proposed filter (MDBUTMF) replaces the noisy pixel by trimmed median value when some of the elements with values 0"s and 255"s are present in the selected window.

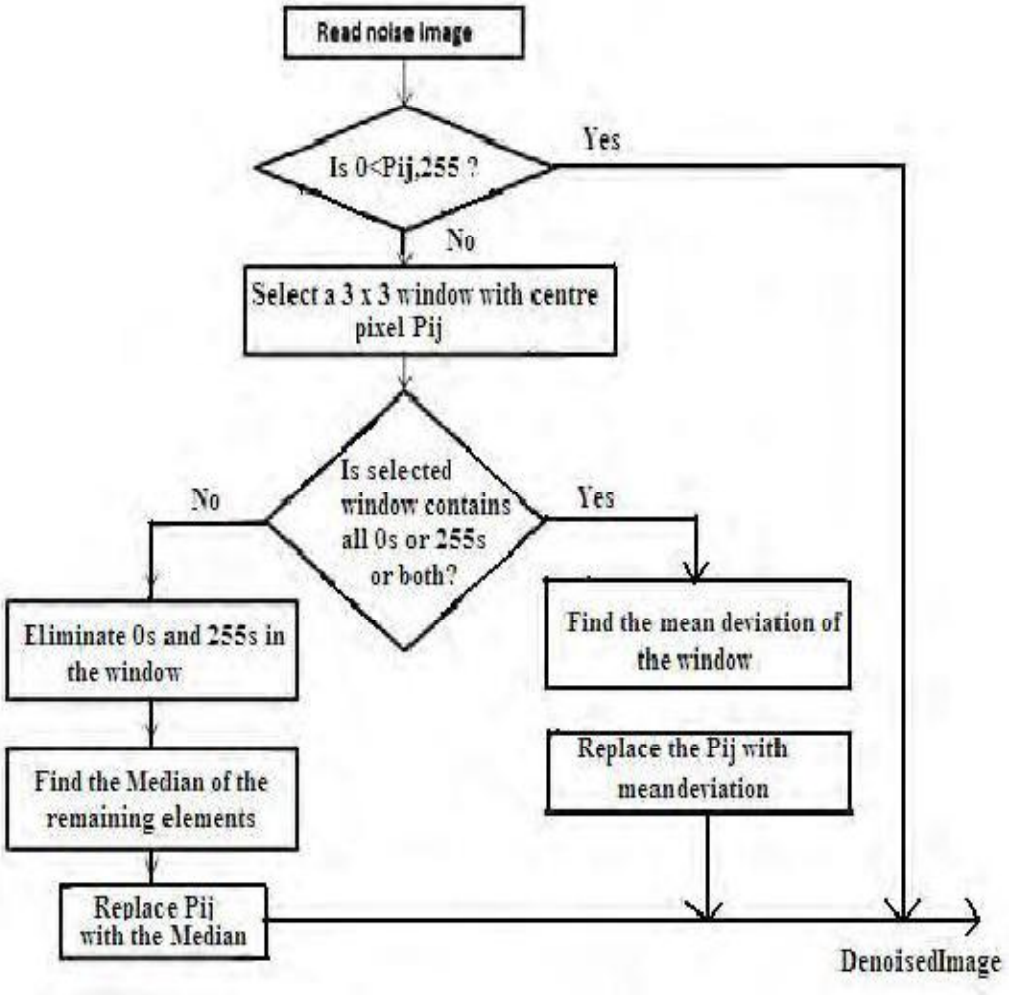


Fig.1. Flow chart of the proposed algorithm

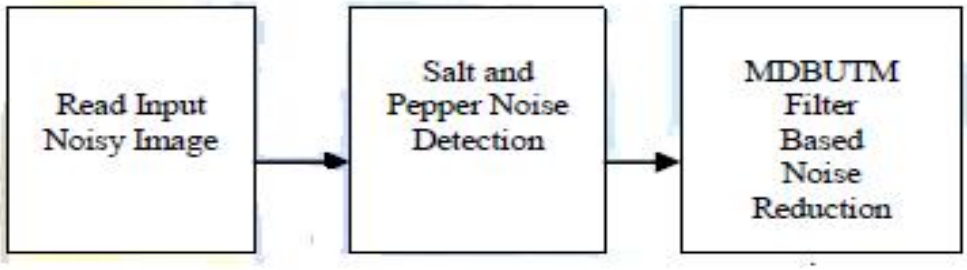


Fig.2. data flow of the proposed method

If all the pixel values in the selected window are 0‟s and 255‟s means then the noisy pixel is replaced by mean value of all the elements present in that selected window. In the proposed method first the noisy image is read then based on some decision salt and pepper noise detection takes place. At the end of the detection stage the noisy and noise-free pixels get separated. The noise-free pixel is left unchanged and the noisy pixel is given to the Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF). The MDBUTMF produces an image as its throughput that is a noise removed one. The data flow of the proposed method is shown in Fig.2. The flow goes in the way that, first the noisy image is given to a noisy image reader.

Followed by this is the salt and pepper noise detection. After this, based on the state of the elements the corrupted pixel is either replaced by Type-I or replaced by Type-II. Type-I): If the selected window contains all the elements as 0‟s and 255‟s means, then replace the processing pixel by the mean value of the elements present in that window. Type-II): If the selected window contains not all elements as 0‟s and 255‟s. Then eliminate 0‟s and 255‟s and find the median value of the remaining elements. Replace the processing pixel with the median value. The clear explanation of Type-I and Type-II with examples is given in this Section.

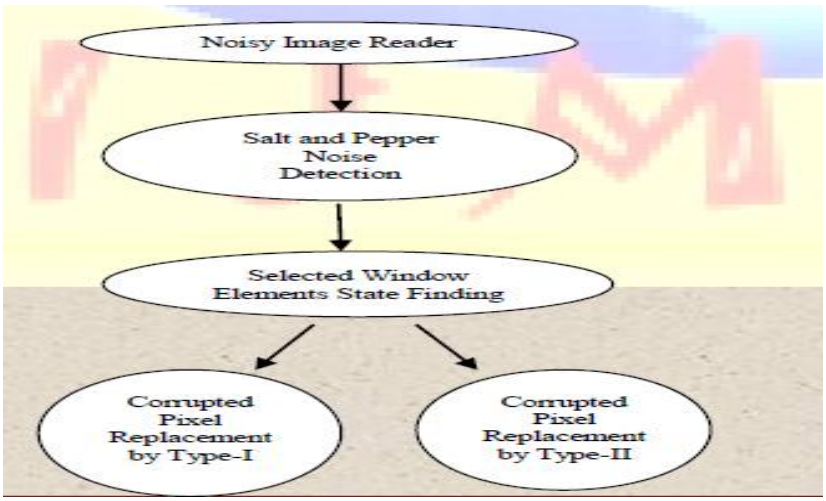


Fig. 3. Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF)

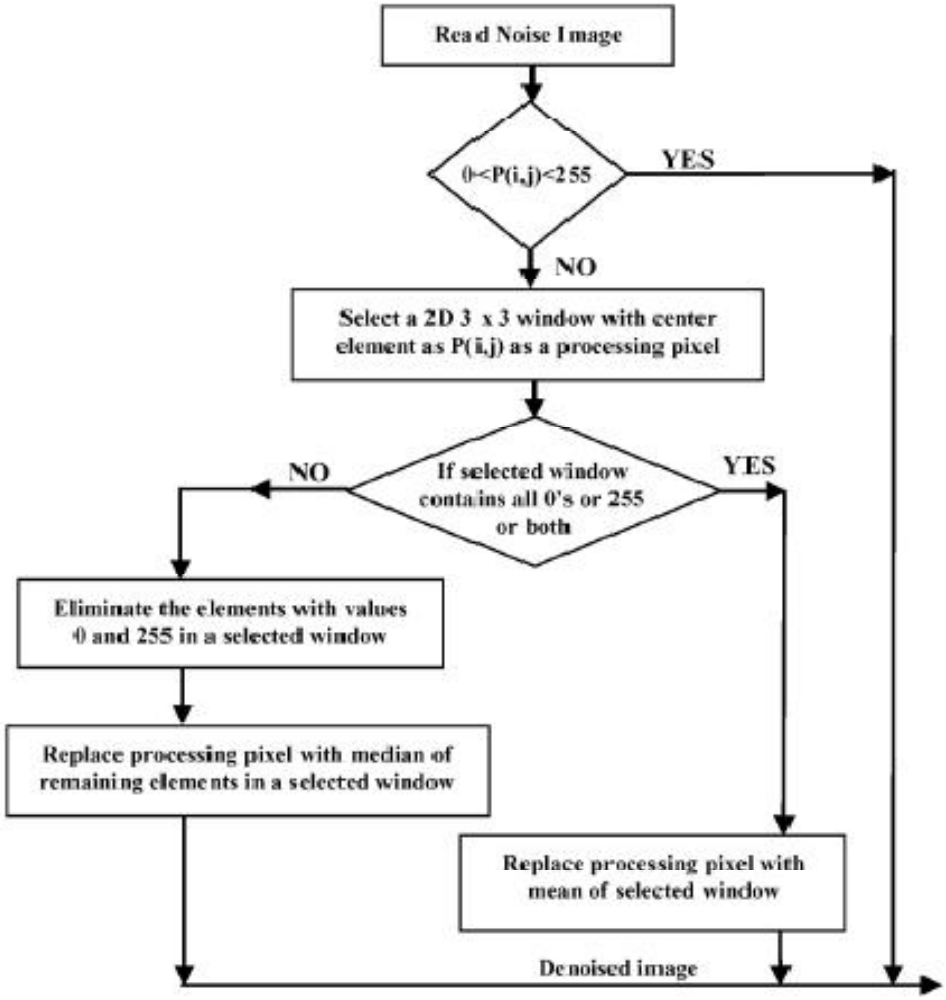


Fig. 4. Illustration of Mdbutm Filter

The output images produced by the combination of Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) contain excellent Result than the existing methods.

The proposed Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) algorithm processes the corrupted images by first detecting the impulse noise. The processing pixel is checked whether it is noisy or noisy free.

That is, if the processing pixel lies between maximum and minimum gray level values then it is noise free pixel, it is left unchanged. If the processing pixel takes the maximum or minimum gray level then it is noisy pixel which is processed by MDBUTMF.

ALGORITHM

Step 1: Select 2-D window of size 3X 3. Assume that the pixel being processed is□□□
Step 2: If *0* < □□□ < 255 then is an uncorrupted pixel and its value is left unchanged. This is illustrated in Case iii) of Section Illustration of MDBUTM Filter.

Step 3: if is a corrupted pixel then two cases are possible as given in Case i) and ii).

Case i): If the selected window contains all the elements as 0's and 255's. Then replace □□□ with the mean of the element of window. Case ii): If the selected window contains not all elements as 0's and 255's .Then eliminate 255's and 0's and find the median value of the remaining elements. Replace with the median value.
Step 4: Repeat steps 1 to 3 until all the pixels in the entire image are processed. The pictorial representation of each case of the proposed algorithm is shown in Fig. The detailed description of each case of the flow chart shown in Fig.1 is illustrated through an example in Section Illustration of MDBUTM Filter.

ILLUSTRATION OF MDBUTM FILTER

Each and every pixel of the image is checked for the presence of salt and pepper noise. Different cases are illustrated in this Section. If the processing pixel is noisy and all other pixel values are either 0's or 255's is illustrated in Case i). If the processing pixel is noisy pixel that is 0 or 255 is illustrated in Case ii). If the processing pixel is not noisy pixel and its value lies between 0 and 255 is illustrated in Case iii). *Case i):* If the selected window contains salt/pepper noise as processing pixel (i.e., 255/0 pixel value) and neighboring pixel values contains all pixels that adds salt and pepper noise to the image:

$$\begin{bmatrix} 0 & 255 & 0 \\ 0 & (255) & 255 \\ 255 & 0 & 255 \end{bmatrix}$$

Where "255" is processing pixel, i.e., . P_{ij}

Since all the elements surrounding □□□ are 0's and 255's. If one takes the median value it will be either 0 or 255 which is again noisy. To solve this problem, the mean of the selected window is found and the processing pixel is replaced by the mean value. Here the mean value is 170. Replace the processing pixel by 170.

Case ii): If the selected window contains salt or pepper noise as processing pixel (i.e., 255/0 pixel value) and neighboring pixel values contains some pixels that adds salt (i.e., 255 pixel value) and pepper noise to the image:

$$\begin{bmatrix} 78 & 90 & 0 \\ 120 & (0) & 255 \\ 97 & 255 & 73 \end{bmatrix}$$

Where "0" is processing pixel, i.e., □□□ . Now eliminate the salt and pepper noise from the selected window. That is, elimination of 0's and 255's. The 1- D array of the above matrix is [78 90 0 120 0 255 97 255 73]. After elimination of 0's and 255's the pixel values in the selected window will be [78 90 120 97 73]. Here the median value is 90. Hence replace the processing pixel by 90. *Case iii):* If the selected window contains a noise free pixel as a processing pixel, it does not require further processing. For example, if the processing pixel is 90 then it is noise free pixel:

$$\begin{bmatrix} 43 & 67 & 70 \\ 55 & (90) & 79 \\ 85 & 81 & 66 \end{bmatrix}$$

where "90" is processing pixel, i.e. □□□, Since "90" is a noise free pixel it does not require further processing.

FPGA

Field Programmable Gate Arrays (FPGAs) correspond to reconfigurable technology, which is in most ways supremely apt for video processing application. Reconfigurable devices are processors that can be programmed with a the desired design specification, and then can be reprogrammed or reconfigured with almost infinite designs as the designer's needs alter. FPGAs in general consists of a system of logic blocks typically look up tables and flip-flops and a few amounts of Random Access Memory (RAM), all hardwired collectively using a enormous collection of interconnects. All of the design logic in an FPGA can be rewired reprogrammed, with a diverse designs as frequently as the designer desired.

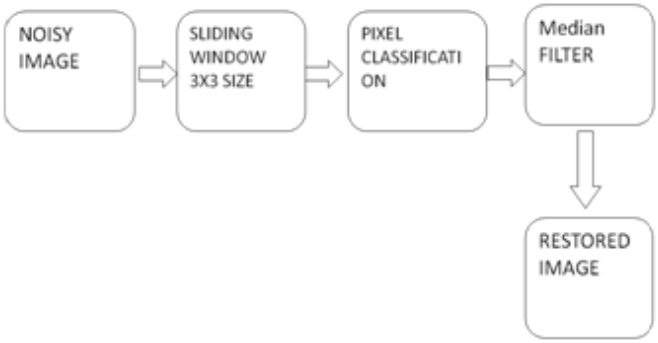


Fig. 5. Block Diagram& Description

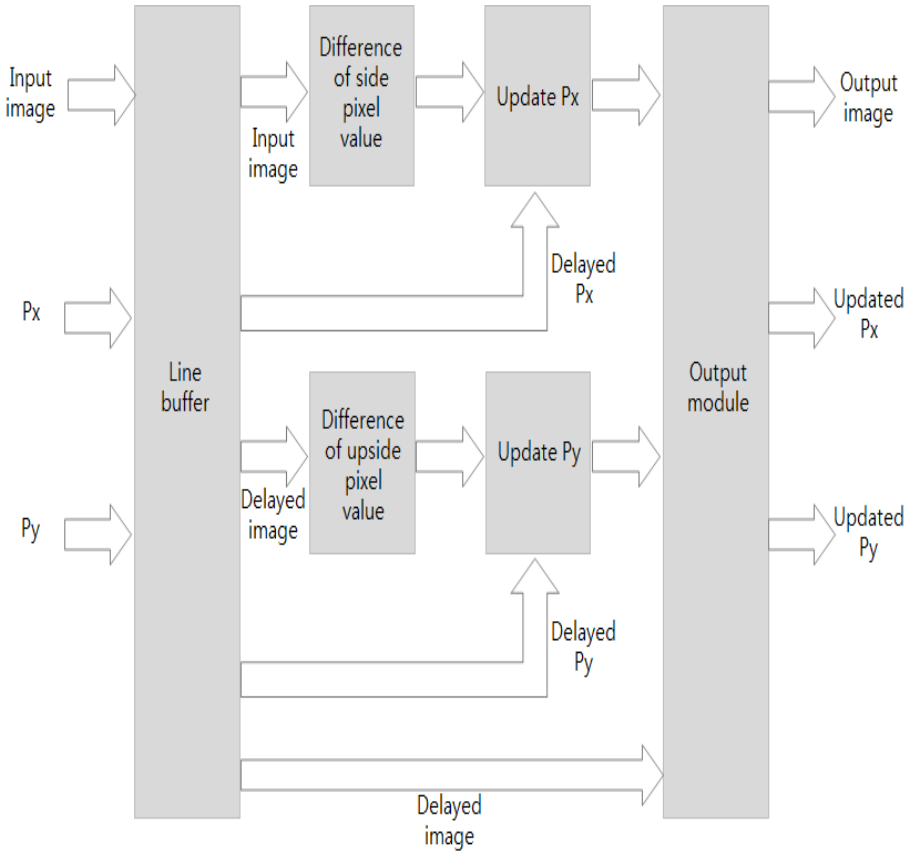


Fig. 6. Hardware architecture of denoising system

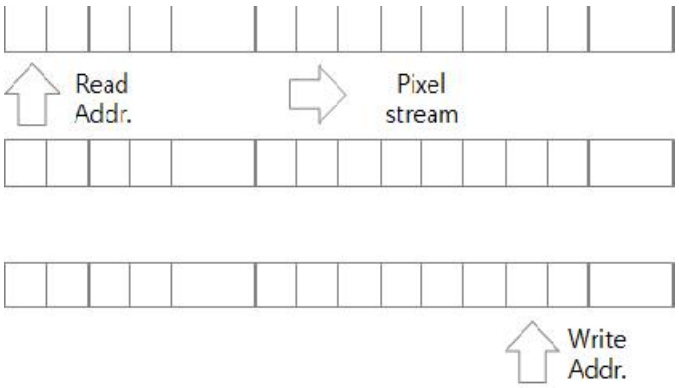


Fig.7. Line buffer

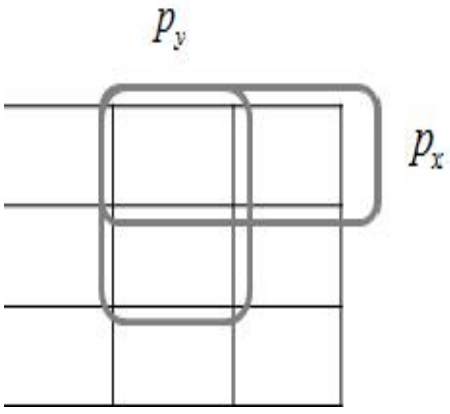


Fig. 8. Obtaining the pixel difference.

This sort of architecture allows a huge assortment of logic designs reliant on the processor's possessions, which can be interchanged for a new-fangled design as rapidly as the device can be reprogrammed. In the present day, FPGAs can be urbanized to implement parallel design style, which is not achievable in dedicated DSP designs. ASIC design methodology can be applied for FPGA designs, prompting the designer to implement designs at gate level. However, frequently engineers use a hardware language such as VHDL or Verilog, which fovours for a design tactic comparable to software design.

ALGORITHM

we introduce the Total Variation algorithm implemented in this paper. The Total Variation algorithm aims to provide the minimum value of the MSE (mean square error) repeat of the image to remove noise. Rudin, Osher and Fatemi applied the Total Variation algorithm for image processing in computervision (Beghdadi and Khellaf, 1997). The ROF model was used to remove unwanted scale pixel values and remove the noise in an image. The ROF model has the following form.

$$\min\left[\max_{\|p\|\le 1}\left\{\int_{\Omega}\frac{1}{2\lambda}(u-f)^2+p\cdot\nabla u d\Omega\right\}\right]$$

By optimization, we derive formula (2), since formula (1) is difficult to implement. Formulas (2) and (3) can be obtained and expressed by dividing.

$$u=f+\lambda\nabla\cdot p=f+\lambda\left(\frac{\partial p_x}{\partial x}+\frac{\partial p_y}{\partial y}\right)$$

$$p'=p+\frac{\tau}{\lambda}\left(\nabla\left(f+\lambda\nabla\cdot p\right)\right)$$

Ω is the domain of an image, and f is the noisy image. u is the image after the noise removal process.

The order derivative term consists of obtaining the difference between the center pixel and pixel below and the difference between the center pixel and the right pixel. τ and λ are parameters. One advantage of the ROF model is the problem does not occur, even if the pixel value of the removed noisy image is zero, because the difference between the pixels continues to be saved. The condition $\|p\|\le 1$ must be satisfied to process this formula.

We must meet the constraint of optimizing the formula. The order derivative terms in formula (4) can be expressed as the following equation.

$$\frac{\partial p_x}{\partial x}=\begin{cases}p_x^{\,i,j+1}-p_x^{\,i,j}\\0\end{cases}$$

$$\frac{\partial p_y}{\partial y}=\begin{cases}p_y^{\,i,j+1}-p_y^{\,i,j}\\0\end{cases}$$

This algorithm is processed in the following order. First, we obtain the difference between the center pixel and the pixel below, and the difference between the center pixel and the right pixel. Second, we update the image using the parameters (τ divided by λ) multiplied obtained in the previous process. In the third step, we add the noisy image and update p. An image with noise removed is obtained by repeating these steps. The range of λ is adjusted to regulate the effect of the surrounding pixels on the center pixel.

HARDWARE ARCHITECTURE

Fig. 6 presents the proposed hardware architecture. The proposed system consists of 10 hardware modules. Each module consists of an input part, a processing part, and an output part. In the input part, an image with 640 □ 480 resolution and *px*, *py* obtained from the previous module are input. In the processing part, the input image is updated by *px*, *py*, and *px*, *py* are updated. The output part generates delayed video sync signals and updated images. Fig. 7 presents the line buffer in the proposed system.

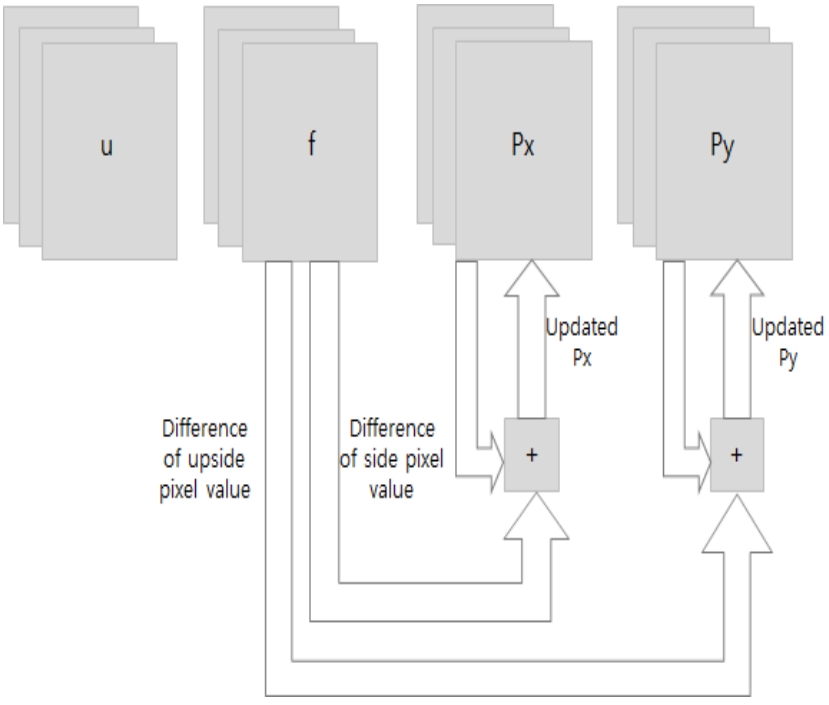


Fig. 9. Processing part of denosing system

The processing time is calculated in the proposed system, and a pixel can be stored to adjust the position of a pixel. The difference between the write address and the read address is processing time of the proposed system. And we use this buffer for obtaining the difference of pixel value. Fig. 8 presents the process of obtaining difference of each pixel. In the pixel buffer, we can calculate the difference of pixel using value in buffer. We already know the location of the pixels. Fig. 9 presents the processing part of the proposed system. The processing procedure is represented by the block diagram. The processing unit is divided into two parts: the processing part consists of obtaining the difference between the pixel values of the center pixel and the surrounding pixels. We obtain the difference to update the part of the right and the center pixels in the input image, and we adjust the image using the updated difference between the pixel values. Fig. 9 indicates the process to obtain the difference between the center pixel and its surrounding pixels. The process is as follows. We store three lines of the input image *f*, then

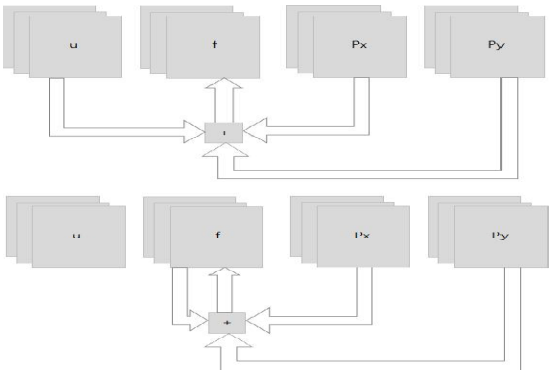


Fig. 10. Applying p_x, p_y to the image

we convert them to floating point values in order to ease the operation before storing the image. We can update *px*, *py* using the parameters (τ / λ) multiplied by the image stored in the buffer. If we want to update *px*, *py*, it should be confirmed whether the condition introduced in the algorithm part is met. The condition is introduced in the algorithm part. The checking process is separated. If the conditions $||p|| \leq 1$ are not met, the following formula should be applied

Fig. 11. Experimental environment

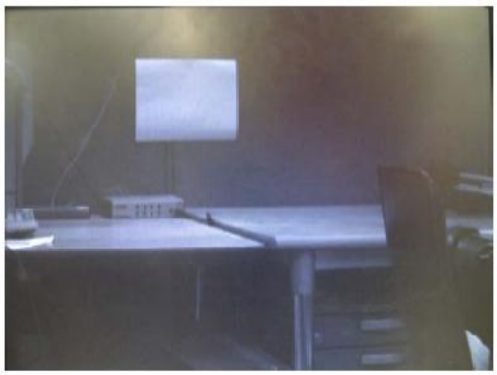
$$p'_x = \frac{p_x}{\sqrt{p_x^2 + p_y^2}}$$

$$p'_y = \frac{p_y}{\sqrt{p_x^2 + p_y^2}}$$

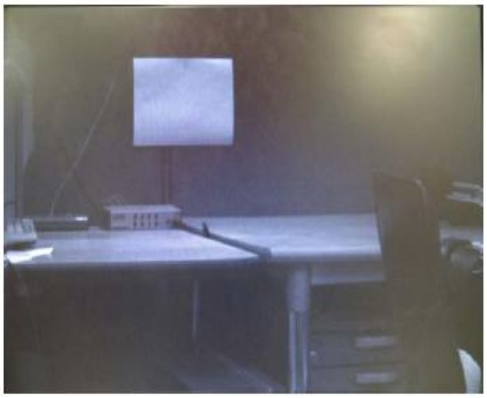
The initial states of p_x, p_y are zero. Fig. 11 indicates the process for applying p_x, p_y to the image. The image is adjusted as follows: we add p_x, p_y and parameter (λ) and multiplied by the image of the current corresponding to the location of the pixel u. We create a buffer for updating, to delay the u add timing corresponds to the location of the pixel in the same p_x, p_y. The images f and u have the same pixel value in the initial state.

EXPERIMENTAL RESULT

We proposed a dedicated pipelined hardware architecture for noise removal. The proposed hardware architecture is designed with the Verilog hardware description language. We obtained input images using a VCC-8350CL camera. This camera provides an image resolution of 640 □480 and 60 frames per second. The clock of this camera is 24.5 Mhz. Fig. 6.



Original image



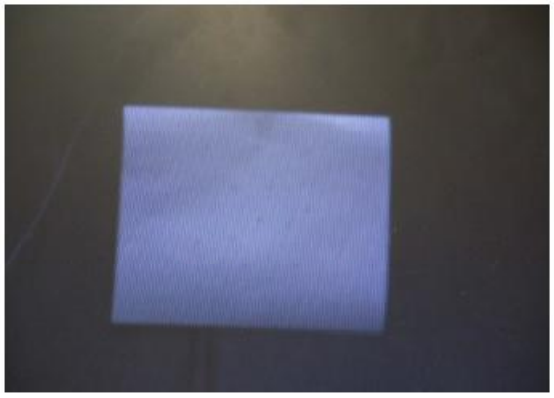
λ = 0.2

Slice Logic Utilization	Used	Available	Util
Number of Slice Registers	90,728	207,360	43%
Number of Slice LUTs	68,657	207,360	33%
Number used as logic	62,313	207,360	30%
Number used as Memory	5,626	54,720	10%
Number of occupied Slices	31,395	51,840	60%
Number of LUT Flip Flop pairs used	98,216		
Number with an unused Flip Flop	7,488	98,216	7%
Number with an unused LUT	29,559	98,216	30%
Number of fully used LUT-FF pairs	61,169	98,216	62%
Number of bonded IOBs	73	1,200	6%
Number of LOCed IOBs	73	73	100%
Number of BlockRAM/FIFO	139	288	48%
Total Memory used(KB)	4,428	10,368	42%

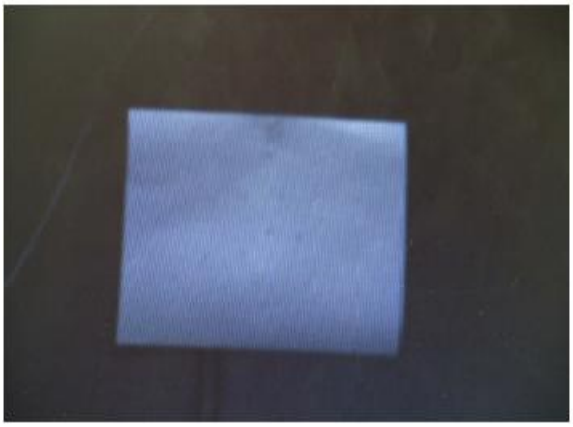
λ = 0.1

λ = 0.5

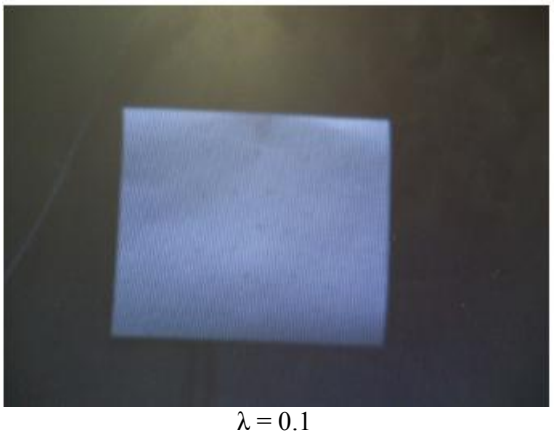
Fig. 12. Result images



Original image



λ = 0.2



λ = 0.1

λ = 0.5

Fig. 13. Expanded result images

Presents the experiment environment. We implemented the system using an XC5VLX330. We implemented the proposed system with internal logic of this system. Table I shows the implementation result of the proposed system. The proposed system can operate at up to 250Mhz processing speed with no loss of accuracy. A pipeline structure is designed operations that occurs processing delay in each module. Our proposed system does not show a frameby- frame delay, even when the number of iterations was increased. Fig. 12 shows the output images according to parameter variations, and Fig.13 is the result of the expanded image. Fig. 12 and 13 indicate the output images according to variation of the parameter (λ).

Conclusion

In this project, we have presented a new efficient decision-based filter, the multiple thresholds switching filter, for image restoration. Because the new impulse detection mechanism can accurately tell where noise is, only the noise-corrupted pixels are replaced with the estimated central noise-free ordered mean value. As a result, the restored images can preserve perceptual details and edges in the image while effectively suppressing impulse noise. The experimental results included in this project have demonstrated that the proposed filter significantly outperforms a number of well-accepted decision-based filters.

REFERENCES

Abreu, E. and Mitra, S. K. 1995. “A signal-dependent rank ordered mean (SD-ROM) filter: a new approach for

removal of impulses from highly corrupted images," in *Proceedings of the IEEE International Conference on Acoustics*, *Speech*, *and Signal Processing*, pp. 2371-2374.

Abreu, E., Lightstone, M., Mitra, S. K. and Arakawa, K. 1996. "A new efficient approach for the removal of impulse noise from highly corrupted images," *IEEE Transactions on Image Processing*, Vol. 5, pp. 1012-1025.

Arakawa, K. 1996. "Median filters based on fuzzy rules and its application to image restoration," *Fuzzy Sets and Systems*, Vol. 77, pp. 3-13.

Beghdadi, A. and Khellaf, A. 1997. "A noise-filtering method using a local information measure," *IEEE Transactions on Image Processing*, Vol. 6, pp. 879-882.

Chen, T. and Hong, R. W. 2001. "Application of partition-based median type filters for suppressing noise in images," *IEEE Transactions on Image Processing*, Vol. 10, pp. 829-836.

Chen, T. and Wu, H. R. 2001. "Adaptive impulse detection using center-weighted median filters," *IEEE Signal Processing Letters*, Vol. 8, pp. 1-3.

Chen, T., Ma, K. K. and Chen, L. H. 1999. "Tri-state median filter for image denoising," *IEEE Transactions on Image Processing*, Vol. 8, pp. 1834-1838.

Kim, J. S. and Park, H. W. 2001. "Adaptive 3-D median filtering for restoration of an image sequence corrupted by impulse noise," *Signal Processing: Image Communication*, Vol. 16, pp. 657-668.

Ko, S. J. and Lee, Y. H. 1991. "Center weighted median filters and their applications to image enhancement," *IEEE Transactions on Circuits Systems*, Vol. 38, pp. 984-993.

Lee, K. C., Song, H. J. and Sohn, K. H. 1998. "Detection-estimation based approach for impulsive noise removal," *Electronics Letters*, Vol. 34, pp. 449-450.

Lin, T. C. and Yu, P. T. 2004. "Adaptive two-pass median filter based on support vector machine for image restoration," *Neural Computation*, Vol. 16, pp. 333-354.

Pok, G. Liu, J. C. and Nair, A. S. 2003. "Selective removal of impulse noise based on homogeneity level information," *IEEE Transactions on Image Processing*, Vol. 12, pp. 85-92.

Russo, F. 1999. "FIRE operators for image processing," *Fuzzy Sets and Systems*, Vol. 103, pp. 265-275.

Sun, T. and Neuvo, Y. 1994. "Detail-preserving median based filters in image processing," *Pattern Recognition Letters*, Vol. 5, pp. 341-347.

Wang, J. H. and Lin, L. D. 1997. "Improved median filter using minmax algorithm for image processing," *Electronics Letters*, Vol. 33, pp. 1362-1363.

Wang, Z. and Zhang, D. 1999. "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Transactions on Circuits and Systems- II: Analog and Digital Signal Processing*, Vol. 46, pp. 78-80.

****