hajabdollahi mohsen (Orcid ID: 0000-0002-1058-9459)
Karimi Nader (Orcid ID: 0000-0001-8904-1607)
Soroushmehr S.M. Reza (Orcid ID: 0000-0001-8417-9260)

Real-Time Removal of Impulse Noise from MR Images for Radiosurgery Applications

Zohreh HosseinKhani[1], Mohsen Hajabdollahi[1], Nader Karimi[1], Kayvan Najarian[2,3], Ali Emami[1,4]
Shahram Shirani[5], Shadrokh Samavi[1,5], S.M. Reza Soroushmehr[2,3]

[1]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran
[2]Michigan Center for Integrative Research in Critical Care, University of Michigan, Ann Arbor, MI 48109 U.S.A
[3]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109 U.S.A
[4]Department of Information Technology and Electrical Engineering, University of Queensland, Brisbane, Australia
[5]Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada

Abstract

In the recent years, image processing techniques are used as a tool to improve detection and diagnostic capabilities in the medical applications. Among these techniques, medical image enhancement algorithms play an essential role in the removal of the noise which can be produced by medical instruments and during image transfer. Impulse noise is a major type of noise, which is produced by medical imaging systems, such as MRI, CT, and angiography instruments. An embeddable hardware module, which can denoise medical images before and during surgical operations, could be very helpful. In this paper, an accurate algorithm is proposed for real-time removal of impulse noise in medical images. Our algorithm categorizes all image blocks into three types of edge, smooth, and disordered areas. A different reconstruction method is applied to each category of blocks for noise removal. The proposed method is tested on MR images. Simulation results show acceptable denoising accuracy for various levels of noise. Also, an FPGA implementation of our denoising algorithm shows acceptable hardware resource utilization. Hence, the algorithm is suitable for embedding in medical hardware instruments such as radiosurgery devices.

Keywords: Medical image processing, MR imaging, real-time implementation, impulse noise.

I. Introduction

Medical images are affected by different types of noise. Presence of impulse noise can produce misleading artifacts in the visual representation of the interior of human organs. These artifacts could mislead the expert in the process of diagnosis or prognosis. Noise can be produced in different types of medical image instruments such as MR, CT, X-ray, ultrasound, etc. In medical applications, the probability of noise creation is increased due to the fast scanning process [1]. Different types of noises in medical imaging instruments, such as impulse, Gaussian, and speckle, can be created during image capture or transmission [1]–[4]. Numerous research works have been conducted in detection and elimination of noise in medical images. Sanches *et al.* in [5], and Toprak *et al*. in [6], studied the problem of impulse noise reduction in medical images. Balafar proposed an adaptive filter with edge preserving property for Rician noise in MRI images [1] . In [2], for Gaussian and impulse noise detection in tomography images, discriminative bilateral filtering is proposed. Sawant *et al*. designed an adaptive median filter for removal of impulse noise in X-ray images and speckle noise in ultrasound images [3]. In [4], medical images which are used for detecting cancer in different parts of human body are considered, and different types of noise affecting such images are reviewed. Impulse noise is investigated in many different medical imaging applications such as MR imaging [7], [8], mammogram images [9], etc.

Impulse noise is a common type of noise, which is randomly distributed throughout the image. Impulse noises are divided into two main types, according to the range of the injected values. The first type is the fixed-value impulse noise (FVIN). As shown in equation (1), in grayscale images the randomly injected values could belong to one of the two constant ranges [10]. In equation (1), m is a constant value, $x_{i,j}$ and $y_{i,j}$ are the original and noisy value of the pixel respectively, and $p1$ and $p2$ are the probabilities that the pixel gets the noisy value, where $p = p1 + p2$. Also, $(L-1)$ shows the maximum possible intensity value of a pixel.

$$p(y_{i,j} = y') = \begin{cases} p1 & 0 \le y' \le m \\ p2 & 0 \le (L-1) - y' \le m \\ 1-p & y' = x_{i,j} \end{cases} \quad (1)$$

If $m = 0$ then the induced noise is salt and pepper noise. The second type is random value impulse noise (RVIN). As shown in equation (2) for gray-scale images, a pixel may get noisy with a probability of p, where a value in the range of 0 to $(L-1)$ is randomly chosen to replace the original pixel [10]. In equation (2), r is a random value, $x_{i,j}$ and $y_{i,j}$ are the original and new values of the pixel respectively.

$$p(y_{i,j} = y') = \begin{cases} p & y' = r \\ 1-p & y' = x_{i,j} \end{cases} \quad (2)$$

Impulse noise has been of interest to many research works as a common noise. In [11], mixed impulse noises are removed using an iterative method based on an s-estimator for variance MAD (median of absolute deviations from median). Noisy pixels are detected using a pixel-wise s-estimator and reconstructed using EPR method. In [12], noisy pixels are detected using sparse representation and reconstructed using an image inpainting method. In [13], noisy pixels are detected using Laplacian based second order of difference. To preserve image details, anisotropic diffusion method is used for reconstruction. In [14], partial differential equation (PDE) based method for impulse noise removal is presented. Two controlling function for PDE model is modified to take difference of edge, noisy and interior pixels in to account. In [15], a method based on Dempster–Shafer evidence theory as an alternative of Bayesian theory is used for noise detection. Fuzzy averaging filter is employed for restoration of noise-corrupted pixels.

Most denoising methods that are proposed for impulse noise in natural images are computationally complex. For example, fuzzy methods in colored videos [16], evolutionary algorithms [17], [18], and an uncertainty based detector with a weighted fuzzy filter [19] can be considered as high complexity denoising methods. Bhadouria and Ghoshal proposed a genetic programming approach for detection of noisy pixels is proposed [17]. They avoid the blurring effect by using a modified median based method using genetic programming. *Zhou* used genetic programming for the detection phase[19]. Also for the restoration phase, genetic programming selects the most similar pixel to the original pixel.

On the other hand, some denoising methods have lower complexity. For example, in [20]–[24] a patch oriented approach, based on the image texture, is used for noise detection. *Turkmen* designed a multi-layer perceptron model for detection of noisy pixels [20]. To train this model, two features are extracted from the image patches including rank ordered absolute difference (ROAD) and rank ordered logarithmic difference (ROLD). Edge-Preserving filtering restores noisy pixels. In [21], [22], median operation is used for image reconstruction and in [23], a reconstruction method, based on edge directions, is considered. *Mandal* and Mukhopadhyay proposed a restoration method in which median operation on non-noisy pixels is performed for restoration [24].

Enhancing the MR images is of importance in the segmentation of the gray matter of the brain. With the advancement of the image-guided surgical approaches, segmentation of MR images is becoming an important tool [25]. Therefore, MR image enhancement and denoising play essential roles before and during surgical operations such as radiosurgery. Many studies have tried to enhance and remove the impulse noise in MR images. *Sadri et al.* [26], employed a wavelet network, as a preprocessing stage, and a median operation for removal of noisy pixels in medical images. Differences between gray-scale values and the average of the background are feed to a wavelet network. The

wavelet network detects the location of the noise and replaces the noisy pixel with the median of the neighboring pixels. Bharathi and Govindan in [27], proposed an impulse noise removal method using a hybrid filtering method based on the structure of each image block and median operation. In [7], a fuzzy genetic algorithm is proposed which has relatively high computation complexity. Fuzzy rules are used to modify the cross-over probability which generates proper denoising filters by using a genetic algorithm. Although averaging between multiple images reduces the noise in MR images, but it increases image acquisition time. Therefore, in most cases, a filtering method as a post-processing step is used. The filtering process may involve image blurring and smoothing [28].

Since MR image processing can be a time-consuming task, hardware implementation of these algorithms can be beneficial to obtain a better performance [29]. The need for real-time implementation of some enhancement techniques, such as denoising, makes hardware implementation more appealing. There are numerous studies to accelerate medical image processing algorithms using hardware -accelerators, such as FPGAs and GPUs [29]. *Chen et al.* proposed a denoising method and designed its VLSI architecture [30]. In the noise detection stage of this method, the maximum and minimum intensities in a 3×3 window are calculated. In the restoration stage, edge directions are considered, and noisy pixels are restored in the correct edge direction. Hosseini and Hesar proposed a real-time approach for impulse noise removal [31]. In [32], noisy pixels are detected using decision-tree and amount of similarity between neighboring pixels. In this method, weighted filtering approach is implemented for the reconstruction of the detected noisy-pixels. Similar to the algorithm of [30], the edge-direction is utilized to restore the noisy pixels. In [33], a simple method for detection and restoration of the noisy pixels is proposed which is implemented on FPGA. Hosseinkhani *et al.* proposed a hardware architecture for detection and restoration of the random-valued impulse noise on medical imaged [34]. Image blocks are locally analyzed and divided into four regions including smooth, noisy smooth, edge and noisy edge. Pixels detected as noisy one, are restored based on their regions by different filters.

In the denoising process, different factors such as accuracy, scalability, and complexity must be taken in to account. All of the mentioned factors are important, but in some applications some of these properties become critical. In real-time applications and for embedding an algorithm in a medical imaging instrument, it is essential to decrease complexity and increase the speed of operation [35].

In this paper, we are proposing an accurate and real-time algorithm to detect and remove impulse noises in MR images. For local image blocks different structures, such as edges, smooth, and disordered areas, are considered. Different reconstruction methods are applied for each block depending on its structure. Due to efficient detection and adaptive reconstruction method, noisy pixels are removed while image structures are preserved accurately. This type of performance is essential in the processing of medical images. All steps of the proposed denoising algorithm are designed to have low hardware complexity. For each part of the proposed algorithm, a hardware implementation is designed and optimized which makes the proposed method suitable for denoising of images in medical imaging instruments.

The rest of this paper is organized as follows. The proposed method for removal of random value impulse noise, composed of software algorithm and hardware architecture, is explained in sections 2 and 3, respectively. Section 4 is dedicated to simulation results, and in section 5 concluding remarks are presented.

II. PROPOSED METHOD

In this research, we are considering RVIN, which is more common noise and its removal is more challenging. For random value impulse noise, it is not easy to label a pixel as being noisy because it could have any grayscale value. To overcome this issue in the proposed method, we categorize all image blocks into four block types. These categories are called noisy-smooth, edge, noisy-edge, and disordered blocks.

2.1. The general structure of the algorithm

The block diagram of the proposed method, a sample noisy image, and the restored image, are displayed in Fig. 1. As illustrated in Fig. 1, for each detected structure of a block, different reconstruction methods are used. Our

proposed method categorizes the image blocks to one of four different types and applies a proper reconstruction scheme. These four types are shown by examples in Fig. 2. Different stages of the algorithm are as follows.

2.2. Block Partitioning

Normally, neighboring pixels of an image block have similarities. Similarity diminishes in the presence of noise. We need to detect abnormal variations in pixel values. Therefore, the neighborhood of each pixel is analyzed to find out if the pixel is a normal part of that neighborhood. Hence, the proposed method partitions the image into 3×3 and 5×5 blocks depending on the local structure of the image and the severity of the noise. It is necessary to use variable block size for better noise detection in different block structures. For example, in edge blocks, the larger block size leads to better edge detection and better image restoration. In this paper the 9 pixels of the 3×3 block are called $P_1, P_2, \dots, P_9$, where P_5 is the central pixel. Likewise, in 5×5 blocks the pixels are called as $P_1, P_2, \dots, P_{25}$, where P_{13} is the central pixel. As illustrated in row a) of Fig. 2, in order to find a better detection of block categories, pixel intensities are sorted. In this paper the sorted pixel values are called $F_1, F_2, \dots, F_9$. Partitioning of pixels in different block sizes can be useful for the next stages discussed in the following subsections.

2.3. Edge Detection

In noisy conditions, edges can be affected by noise; hence it is necessary to find out whether an edge is noisy or non-noisy. Here, edge detection is done in two steps including *Edge-Detection A* and *Edge-Detection B*. In the first step, using *Edge-Detection A*, it is determined whether a block contains an edge or not. This step is checked for all image blocks as illustrated in row b) of Fig. 2. In the second step, using a 5×5 block, considering the edge directions, rough and non-noisy edges are separated from noisy edges. These two steps of edge detection are explained as follows:

2.3.1. *Edge-Detection A*

In this step, edge regions are being detected which is very useful to better detection and restoration of noisy pixels. So, the presence of an edge region must be examined, at first. In a region containing edge, two different ranges of values can be observed as two sets of pixels. The difference between two set of pixels in an image region could lead us to detect edge regions. In regions containing edge, almost half of the pixels are located in each set. Hence, a difference could be observed between 4^{th}-5^{th}, or 5^{th}-6^{th} elements of the sorted pixels. By sorting pixel's values, difference between two aforementioned sets become more visible. As a result, a 3×3 block around each pixel is considered, and elements of the block are sorted. The differences between the 5^{th} and 4^{th} sorted elements, $(F_5 - F_4)$, or the 5^{th} and 6^{th} elements, $(F_6 - F_5)$, represent the edge strength in the block. Then using a threshold (T_1) the central pixel is labeled as edge based on Equation (3).

$$P_5 = \begin{cases} Edge & \max(|F_5 - F_4|, |F_6 - F_5|) > T_1 \\ Non-Edge & Otherwise \end{cases} \quad (3)$$

As illustrated in row b) of Fig. 2, *Edge-Detection A* is performed for all image blocks. In each step in row b) of Fig. 2, green and red colors indicate that the criterion for this step is met or not, respectively. For those blocks that this condition is true, the *Edge-Detection B* is also performed and for blocks that *Edge-Detection A* is false, *Disorder-Analysis* is performed. *Edge-Detection B* and *Disorder-Analysis*, which are considered as the 3^{rd} step of the proposed algorithm, are discussed in the following.

2.3.2. *Edge-Detection B*

For all pixels that are labeled as an edge by the *Edge-Detection A*, the second edge detection criterion is also checked. In *Edge-Detection A*, all edges were detected, but it is not known whether these edges are noisy or not. Difference between the central pixel and its neighbors on an edge region, demonstrate whether a pixel is located on edge or is noisy. There is a similarity between pixels located on an edge at least in one edge's direction. In *Edge-Detection B*, presence of this similarity at least in one direction is examined. As a result, noisy pixels are detected by considering major edge directions in a 5×5 block. To do so, as shown in Fig. 3, four main directions of horizontal,

vertical, diagonal, and anti-diagonal, are considered. In each of the four directions the weighted sum of absolute differences, $D_i|_{i=1,\dots,4}$, between the central pixel and the other pixels located on a particular direction, is calculated based on (4).

$$D_i|_{i=1,\dots,4} = \sum_{j=-2,-1,1,2} |I_c - W_j I_j| \qquad (4)$$

where I_c is the central pixel, $I_{\pm1}$ are the two pixels that are closest to the central pixel in each direction. Also, $I_{\pm2}$ are the two pixels that are farthest from the central pixel, in each direction. For the two farthest pixels of $I_{\pm2}$ a weight coefficient of ½ is considered, which means $W_j = ½|_{j=\pm2}$. For the two nearest pixels of $I_{\pm1}$ a weight coefficient of 1 is considered, which means $W_j = 1|_{j=\pm1}$. This operation is done in all four main directions. The minimum value, in all four directions, $D_{min} = min(D_i|_{i=1,\dots,4})$, shows the most probable edge direction. If D_{min} were to be less than a threshold (T_2) there is a high similarity between the central and the edge pixels, and the central pixel is considered as an edge pixel. However, if $D_{min} > T_2$, it would be labeled as a noisy edge pixel. In Fig. 2, two examples of edge and noisy edge blocks are shown in columns a) and c) of row c), respectively. According to *Edge-Detection B*, in noisy edge blocks (in column c) of row c)) D_{min} is greater than T_2, where T_2 is considered equal to 150. This situation of noisy edge block is shown by a red colored block. Noisy edge blocks are labeled to be fed into *Edge-Preserved Filtering B* step, which is discussed later. On the other hand, non-noisy edge occurs (in column a) of row c)) if $D_{min} \leq T_2$. This situation is colored by green. Non-noisy edge blocks are labeled and are fed into the *Similarity-Checking* step which is discussed section 2.6.

2.4. *Disorder Analysis*

When *Edge-Detection A* labels a pixel as non-edge, it is important to know whether the pixel is in a smooth-area or a disordered-area. In the *Disorder-Analysis* step, those non-smooth blocks, and blocks that their central pixels have different values from their surrounding pixels, are considered as disordered blocks. In edge and smooth areas, there are similarity between the central pixel and its neighbors. This similarity can be observed between the central pixel and almost half of its neighbors. The central pixel's value should be at least similar to 4th, 5th, or 6th elements of the sorted pixels around it. In other words, if minimum differences between the central pixel and 4th, 5th, and 6th elements of the sorted pixels is greater than a threshold, it is different from the others. In this case, pixel under consideration is identified as a central pixel located in a disordered-area.

The sorted pixels of the 3×3 window are referred and pixels F_4, F_5 and F_6 are considered. Absolute difference between the central pixel with F_4 , F_5 and F_6 is a measure of disorder within the 3×3 neighborhood.

$$P_5 = \begin{cases} Disordered & \min(|F_6 - P_5|, |P_5 - F_4|, \\ & \quad |P_5 - F_5|) > T_3 \\ smooth & Otherwise \end{cases} \qquad (5)$$

where T_3 is a threshold value. Details of the disorder analysis procedure are visually represented in columns b) and d) of row c) of Fig. 2. The 4th, 5th, and 6th sorted elements are considered in Fig. 2, and their differences with the central pixel is compared with a threshold T_3. If all three absolute differences are greater than T_3, the image block is considered to be disordered. Since the disordered blocks are potentially expected to be noisy, the *Noisy-Pixel-Checking* procedure is applied as the next step. As illustrated in Fig. 2, in columns b) of row c), for disordered blocks, the central pixel as well as three sorted pixels (F_4 ,F_5, F_6) are highlighted with green. If the condition of Equation (5) is not met then the mentioned pixels are highlighted as red (in Fig. 2, in columns d) of row c)).

2.5. *Noisy Pixel Checking*

In the ***Disorder Analysis***, blocks, which were detected as smooth, may contain some noisy pixels. Such a block may contain a noisy pixel which is surrounded by smooth neighboring pixels. Hence, in a smooth area, those pixels

which have different intensity values from their background are determined as noisy pixels. If a central pixel is not a noisy pixel in a smooth area, its difference between at least one of the smallest or highest values in a neighborhood area should not be significant. As shown by Equation 6, differences between the central pixel P_5 and the maximum or minimum pixels, inside the 3×3 window, are used for detection of a noisy pixel.

$$P_5 = \begin{cases} noisy & \min(F_9 - P_5, P_5 - F_1) < T_4 \\ not\ noisy & otherwise \end{cases} \quad (6)$$

If either $(F_9 - P_5)$ or $(P_5 - F_1)$ is less than a threshold T_4 then the pixel is considered as a noisy pixel in a smooth area. In some conditions, all non-noisy block pixels may have nearly maximum or minimum values. In such a case, they are wrongly considered as noisy pixels based on Equation 6. To prevent this wrong decision, the similarity between a noisy pixel and its neighbors is checked by the ***Similarity-Checking*** step.

2.6. *Similarity Checking*

Checking out the block similarity is necessary for two situations. First when we want to leave the pixel without any modification. In row d) of column a) in Fig. 2, non-noisy an edge-pixel is left without any modification. Second situation is when the pixel's intensity indicates that pixel is noisy as illustrated in row e) of column d) in Fig. 2. Hence, in Fig. 2, similarity must be checked when *Edge-Detection B* and *Noisy-Pixel-Checking* have true conditions. Similar pixels can be recognized via comparing them with their neighbors. So absolute differences between the central pixel and its eight neighbors are calculated to determine the similarity amount. Using threshold T_4, these absolute differences determine similarity or non-similarity among these 8 pairs in 3×3 window. If the number of the similar pixel around a pixel becomes less than a threshold (T_5) , then it is considered to be a noisy pixel. In Fig. 2, if central pixel is similar with its 3×3 neighbors, thus all pixel blocks are colored with green (row d) of column a)) otherwise they are colored with red (row e) of column d)).

2.7. Restoration

The restoration mechanisms are different for different block types. Three methods for restoring the original pixel value are proposed including averaging on fourth, fifth and sixth elements of sorted results (the *Average* method), *Edge-Preserved Filtering A* and *Edge-Preserved Filtering B*. Type of the restoration method depends on the block in which the pixel is located in. Three restoration methods are as follows.

2.7.1. Average

In smooth blocks as well as in blocks that a pixel has similar value to its neighbors, restoration is performed by averaging on fourth, fifth and sixth elements of the sorted 3×3 block as depicted in Fig. 2 (row f) of column d)).

2.7.2. *Edge-Preserved Filtering A*

In this step, the noisy pixels are restored using the direction of the edge. To have an efficient reconstruction method which take edge areas into consideration, we employed edge preserving filtering which proposed in [32], as *Edge-Preserved Filtering A*. As illustrated in Fig. 2 (row e), column b)), in *Edge-Preserved Filtering A*, for disorder blocks, two pixels that are used in the averaging process are colored with green.

2.7.3. Edge-Preserved Filtering B

Noisy pixels which are detected in edge areas are restored with *Edge-Preserved Filtering B*. To have better view on the neighboring pixels and provide better restoration for high-density noises, a 5×5 block around the central pixel is considered. Since there are four main directions of the edge, four directions including horizontal, vertical, diagonal, and anti-diagonal, are considered. All pixels corresponding to each direction are considered. Sum of absolute differences between each pixel and their corresponding average is calculated. In this step central pixel isn't

considered in final results because this pixel is a noisy pixel. Next, to determine the possible direction of the edge, the minimum value in four directions is computed. Finally restoring is performed by taking a median operation on directions which is determined in the previous step. As illustrated in Fig. 2 (row d) of column c)), pixels on the possible direction of the edge, which are used for median restoration, are colored green.

2.8. *Image Formation*

Noise-free pixels detected in the previous steps as well as restored pixels are placed back to form the noise-free image.

III. HARDWARE ARCHITECTURE

The proposed noise removal algorithm is designed to be suitable for hardware implementation. As illustrated in Fig. 4, a 3×3 window around each pixel is considered, and a sorting module sorts its elements. Results of the sorting module are feed to four computational modules, including *Disorder-Analyzer*, *Edge-Detection A*, *Noisy-Pixel-Checker* and a module which performs averaging of the three medians of sorted elements (the *Average* method). Different parts of the hardware structure of the proposed algorithm are explained in the followings:

3.1. *Edge-Detection Module A*

Two subtractor modules, two comparator units, and a logical OR gate form the circuit which can be used as the *Edge-Detection A* module as shown in Fig. 5. It could be used for implementation of (6) as a *Noisy-Pixel-Checking* module by changing the inputs of the circuit.

3.2. *Edge-Detection Module B*

In Fig. 6, the hardware structure of this module is illustrated. In *Edge-Detection B*, absolute differences of the central pixel with corresponding pixels located on the edge direction are computed, and weighted sum of them is computed. Weighted results are produced by sixteen absolute difference calculation modules (ABS-DIF) and eight shift registers. After that, an adder and a comparator are utilized to detect the edge direction.

3.3. *Similarity-Checker Module*

In Fig. 7, we are using eight absolute-difference-calculation modules (ABS-DIF). They are used to calculate the absolute differences between the central pixel and its 8 neighbors. Afterward, the results are compared with the threshold T_4, by using eight comparator units. A positive result from each comparator indicates the similarity of that pixel with the central pixel. Finally, sum of similar pixels is added by an adder unit. The output of the adder is compared with a threshold T_5 to produce the similarity output.

3.4. *Disorder-Analysis Module*

In Fig. 8, disorder-analysis-module is shown. Three absolute-difference-calculation modules (ABS-DIF) are used for calculation of the absolute difference between the central pixel and three medians of the sorted elements. Then these values are compared with threshold (T_3) using comparator module. Final result is provided by logical AND operation of the comparator outputs.

3.5. *Edge Preserved Filtering B*

In the variance-calculation-module (VAR) four ABS-DIF units are used for each main direction. These four units are for calculating the absolute difference between each edge pixel and the average value of the pixels in that direction. An adder module adds these four differences. Figure 9, shows one VAR unit for the anti-diagonal direction. At the next step, as illustrated in Fig. 10, the minimum of the four variances selects one of the four inputs of a multiplexer. Multiplexer inputs are the medians of the four directions. Hence, the direction with least variance is selected and the median of that direction replaces the noisy pixel.

3.6. Sorting and restoration blocks

For the sorting module, a simple structure of [36] is employed. Also, for the *Edge-Preserved Filtering A*, we use the hardware architecture of [32]. In the image formation step, the restored and non-noisy pixels are replaced in proper locations based on the type of pixel. Pixel value replacement can be performed by a multiplexer or by simple wiring.

IV. Experimental Results

We perform software simulation to verify the accuracy of the proposed method, and then we perform FPGA implementation to show the low complexity the algorithm.

4.1. Software Simulation

Natural and MR images are used to validate the performance of the proposed methods. Experiments are performed and verified with MATLAB and source code is available in [37]. In this study, 124 standard 8-bit gray-scale MR images with the size of 256×256 are used [38]. Noise densities between 5% and 40% are uniformly injected. Objective testing of peak signal-to-noise ratio (PSNR) is used to assess the quality of the restored images. In our proposed method we set thresholds $T_1 = 20$, T_2=150, T_3=30, T_4=10 and T_5=6 and in order to achieve better results, the algorithm was repeated twice. In the first iteration due to high noise levels, no similarity can be observed by the ***Similarity-Checking*** stage. Hence, the ***Noisy-Pixel-Checking*** module does not function. Two hardware architectures, proposed in references [21], [32], [34], are used for removal of the impulse noises. Also, 3×3 and 5×5 median filters are used for comparison. As shown in the Table I, the proposed method has better results than the compared methods for all noise densities.

To show some visual results of the proposed method, in Fig. 11, four original MR images and their noisy versions with the presence of 20% impulse noise are shown. In Fig. 12, a median filter is used for noise removal, and PSNR (dB) values are reported for each image. In Fig. 13, comparison of the proposed method with [21], [32], [34], are shown. Simulation results, as shown in Fig. 13, indicate that the proposed method produces better image qualities based on PSNR values. Lena, GoldHill, and Peppers are used for more validation of our method and comparison with other methods. All employed images are 512×512 in TIF format. In Fig. 14, three images above and their noisy versions with the presence of 40% impulse noise are shown. In Fig. 15, a median filter is used for noise removal, and PSNR (dB) values are reported for each image.

In Fig. 16, comparison of the proposed method with [21], [32], [34] are shown. For better visualization of denoising in different methods, in Fig. 17, a part of Peppers image is cropped and results of denoising are illustrated. Noise density of %40 is injected on experimented image in Fig. 17. It can be observed that median 3×3 and denoising method in [21], are not able to proper denoising and Median 5×5 creates some blurring effects. Visual results of our denoising method and [32], are better than the method in [34] . Visual results show that, our denoising algorithm can remove noises in natural images with proper visual quality.

For evaluation of edge detection methods utilized in the proposed algorithm, results of ***Edge-Detection A*** and ***Edge-Detection B,*** are illustrated in Fig 18. Edge detection methods are used in two iterations on 20% noisy Lena image. In Fig 18, it can be observed that in ***Edge-Detection A*** and ***Edge-Detection B,*** edges are detected in the first iteration. In the first iteration of the algorithm, due to the high density of noise, some noisy pixels are wrongly detected as edge which are reduced in the second iteration. Results of ***Edge-Detection B*** in Fig 18 show noisy edges which are detected. Detection of these noisy edge pixels leads to better denoising and edge preserving.

For quantitative evaluation, in Table II, III, and IV, proposed method is compared with related methods. It is important to note that, only in [21], [32], [33], and our method, hardware complexity have been considered. Noise densities of 10% to 50% are tested in case of these three images. In Tables II, noise density of 10% and 20%, in Tables III, noise density of 30% and 40%, and in Tables IV, noise density of 50% are used. Although from Table II-IV, in some cases better results are observed, the proposed method has generally suitable denoising efficiency in different noise densities. Simulations show our medial application intensive method can denoise natural images properly.

4.2. Complexity Analysis

Software simulation is conducted using a PC equipped with an Intel(R) Core(TM) i5-480 CPU 2.67 GHz and 4GB of RAM. Denoising algorithm is run on 10 images including Lena, Peppers, Goldhill, Boat, Barbara, Cameraman, Jetplane, Bridge, House, and Mandrill with size 512×512 and TIFF format. Denoising algorithm is repeated 10 times and average processing time for each 512×512 tested image is obtained 95 S. All operations required for denoising, are conducted for each image's pixel in a 3×3 or 5×5 region. The number of operations in each region is not increased with growing image's size and is a constant value. For a sample image with N pixels, the number of all conducted operations is a fixed-coefficient production of N, hence denoising algorithm is with O(N) computational complexity. For complexity analysis of the proposed method on the hardware platform, FPGA implementation is investigated. The proposed architecture is described in VHDL and is implemented on a XILINX virtex4 family xc4vfx12-12-sf363 device. Implementation specifications as well as average PSNR, for noise densities of 5%, 10%, 15% and 20%, are reported and compared with the other studies in Table V (MR images). It clearly can be seen that the proposed method has better image quality and it has acceptable hardware implementation results. As illustrated in Table V, 2761 number of slices are consumed from the total 5,472 number of available slices. No memory modules (BRAM) are used from the total 648Kb available BRAM. Available BRAM can be used to save intermediate denoising results and improve performance of the algorithm. Also available logic slices provide an opportunity to parallel implementation of denoising
algorithm and utilizing techniques such as pipeline to enhance overall processing time. For a 512×512 image, 12.91 ms is required for denoising the entire image at 40 MHz clock frequency. Also average resulted PSNR in Table V. shows that the proposed system has acceptable denoising performance.

V. CONCLUSION

In this paper, a low complexity noise removal system for MR images was implemented. This method was proved to be suitable for hardware implementation. We can implement our proposed method as a part of a medical image capturing instruments for enhancement of MR images that are used before and during of surgical operations. Our proposed method contains two steps of detection and restoration. We improved results of both steps. Highly accurate noisy-pixel detection in the first stage, and proper removal of noisy pixels in the next stage led to a better restoration of noisy images. Simulation results using MATLAB, performed on MR images, demonstrated that the proposed approach removed random value impulse noise with high accuracy. Also, FPGA implementation of the proposed method resulted in low hardware resource utilization and produced high-quality denoised images.

References

[1] M. A. Balafar, "New spatial based MRI image de-noising algorithm," *Artif. Intell. Rev.*, 39(3), pp. 225–235, 2013.

[2] R. S. Pantelic, R. Rothnagel, C. Y. Huang, D. Muller, D. Woolford, M. J. Landsberg, A. McDowall, B. Pailthorpe, P. R. Young, J. Banks, B. Hankamer, and G. Ericksson, "The discriminative bilateral filter: An enhanced denoising filter for electron microscopy data," *J. Struct. Biol.*, 155(3), pp. 395–408, 2006.

[3] A. Sawant, H. Zeman, D. Muratore, S. Samant, And F. Dibianca, "An adaptive median filter algorithm to remove impulse noise in x-ray and CT images and speckle in ultrasound images," *Projection displays. Conference(San Jose CA, United States 01-26)*, pp. 1263–1274, 1999.

[4] A. K. Saini, H. S. Bhadauria, and A. Singh, "A survey of noise removal methodologies for lung cancer diagnosis," in *Proceedings - 2016 2nd International Conference on Computational Intelligence and Communication Technology, CICT 2016*, 2016, pp. 673–678.

[5] J. M. Sanches, J. C. Nascimento, and J. S. Marques, "Medical image noise reduction using the Sylvester-Lyapunov equation," *IEEE Trans. Image Process.*, 17(9), pp. 1522 - 1539, 2008.

[6] A. Toprak and I. Güler, "Impulse noise reduction in medical images with the use of switch mode fuzzy adaptive median filter," *Digit. Signal Process. A Rev. J.*, 17(4), pp. 711-723, 2007.

[7] K. K. Anisha and M. Wilscy, "Impulse noise removal from medical images using fuzzy genetic algorithm," *Int. J. Multimed. Its Appl.*, 3(4), p. 93, 2011.

[8] K. H. Jin, J. Y. Um, D. Lee, J. Lee, S. H. Park, and J. C. Ye, "MRI artifact correction using sparse + low-rank decomposition of annihilating filter-based hankel matrix," *Magn. Reson. Med.*, 78(1), pp.327-340, 2017.

[9] N. Naveed, A. Hussain, M. Arfan Jaffar, and T. S. Choi, "Quantum and impulse noise filtering from breast mammogram images," *Comput. Methods Programs Biomed.*, 108(3), pp.1062-1069, 2012.

[10] H. Hosseini and F. Marvasti, "Fast restoration of natural images corrupted by high-density impulse noise," *Eurasip J. Image Video Process.*, 2013(1), p.15, 2013.

[11] V. Crnojevi´c and N. I. Petrovi´c, "Impulse noise filtering using robust pixel-wise s-estimate of variance," *EURASIP J. Adv. Signal Process.*, pp. 8:1-8:10, 2010.

[12] B. Deka, M. Handique, and S. Datta, "Sparse regularization method for the detection and removal of random-valued impulse noise," *Multimed. Tools Appl.*, 76(5), pp.6355-6388, 2017.

[13] N. U. Khan, K. V. Arya, and M. Pattanaik, "Edge preservation of impulse noise filtered images by improved anisotropic diffusion," *Multimed. Tools Appl.*, 73(1), pp.573-597, 2014.

[14] J. Wu and C. Tang, "PDE-based random-valued impulse noise removal based on new class of controlling functions," *IEEE Trans. Image Process.*, 20(9), pp. 2428-2438, 2011.

[15] T.-C. Lin, "Decision-based fuzzy image restoration for noise reduction based on evidence theory," *Expert Syst. Appl.*,38(7), pp. 8303–8310, 2011.

[16] T. Mélange, M. Nachtegael, and E. E. Kerre, "Fuzzy random impulse noise removal from color image sequences," *IEEE Trans. Image Process.*, 20(4), pp. 959–970, 2011.

[17] V. S. Bhadouria and D. Ghoshal, “A study on genetic expression programming-based approach for impulse noise reduction in images,” *Signal, Image Video Process.*, 10(3), pp. 575–584, 2016.

[18] S. G. Javed, A. Majid, A. M. Mirza, and A. Khan, “Multi-denoising based impulse noise removal from images using robust statistical features and genetic programming,” *Multimed. Tools Appl.*, 75(10), pp. 5887–5916, 2016.

[19] Z. Zhou, “Cognition and removal of impulse noise with uncertainty,” *IEEE Trans. Image Process.*, 21(7), pp. 3157–3167, 2012.

[20] I. Turkmen, “The ANN based detector to remove random-valued impulse noise in images,” *J. Vis. Commun. Image Represent.*, 34(2016), pp. 28–36, 2016.

[21] T. Matsubara, V. G. Moshnyaga, and K. Hashimoto, “A FPGA implementation of low-complexity noise removal,” in *Proceedings of IEEE International Conference on Electronics, Circuits, and Systems(ICECS)*, 2010, pp. 255–258.

[22] C. Y. Lien, C. C. Huang, P. Y. Chen, and H. Y. Yang, “An efficient denoising approach for random-valued impulse noise in digital images,” in *Proceedings of International Conference on Innovations in Bio-Inspired Computing and Applications(IBICA)*, 2011, pp. 13–16.

[23] K. S. Srinivasan and D. Ebenezer, “A new fast and efficient decision-based algorithm for removal of high-density impulse noises,” *IEEE Signal Process. Lett.*, 14(3), pp. 189–192, 2007.

[24] J. K. Mandal and S. Mukhopadhyay, “A novel variable mask median filter for removal of random valued impulses in digital images (VMM),” in *Proceedings - 2011 International Symposium on Electronic System Design(ISED)*, 2011, pp. 302–306.

[25] A. Hamamci, N. Kucuk, K. Karaman, K. Engin, and G. Unal, “Tumor-Cut: Segmentation of brain tumors on contrast enhanced MR images for radiosurgery applications,” *IEEE Trans. Med. Imaging*, 31(3), pp. 790–804, 2012.

[26] A. R. Sadri, M. Zekri, S. Sadri, and N. Gheissari, “Impulse noise cancellation of medical images using wavelet networks and median filters.,” *J. Med. Signals Sens.*, 2(1), pp. 25–37, 2012.

[27] D. Bharathi and S. M. Govindan, “A new hybrid approach for denoising medical images,” *Adv. Intell. Syst. Comput.*,177(2013), pp. 905–914, 2013.

[28] J.V. Manjón, P. Coupé, L. Martí-Bonmatí, D.L. Collins, and M. Robles, “Adaptive non-local means denoising of MR images with spatially varying noise levels, ” *Journal of Magnetic Resonance Imaging*, 31(1), pp.192-203..

[29] J. J. Koo, A. C. Evans, and W. J. Gross, “3-D brain MRI tissue classification on FPGAs,” *IEEE Trans. Image Process.*, 18(12), pp. 2735–2746, 2009.

[30] P.-Y. Chen, C.-Y. Lien, and H.-M. Chuang, “A Low-Cost VLSI implementation for efficient removal of impulse noise,” *IEEE Trans. Very Large Scale Integr. Syst.*, 18(3), pp. 473–481, 2010.

[31] H. Hosseini, F. Hessar, and F. Marvasti, “Real-time impulse noise suppression from images using an efficient weighted-average filtering,” *IEEE Signal Process. Lett.*, 22(8), pp. 1050–1054, 2015.

[32] C. Y. Lien, C. C. Huang, P. Y. Chen, and Y. F. Lin, “An efficient denoising architecture for removal of impulse noise in images,” *IEEE Trans. Comput.*, 62(4), pp. 631–643, 2013.

[33] V. S. Bhadouria, A. Tanase, M. Schmid, F. Hannig, J. Teich, and D. Ghoshal, “A novel image impulse noise removal algorithm optimized for hardware accelerators,” *J. Signal Process. Syst.*, 89(2), pp. 225–242, 2017.

[34] Z. Hosseinkhani, N. Karimi, S. M. R. Soroushmehr, M. Hajabdollahi, S. Samavi, K. Ward, and K. Najarian, “Real-time removal of random value impulse noise in medical images,” in *Proceedings - International Conference on Pattern Recognition*, 2017, pp. 3916–3921.

[35] C. Alonso-Montes, D. L. Vilariño, P. Dudek, and M. G. Penedo, “Fast retinal vessel tree extraction: A pixel parallel approach,” *Int. J. Circuit Theory Appl.*, 36(5-6), pp.641-65,1 2008.

[36] S. A. Fahmy, P. Y. K. Cheung, and W. Luk, “Novel FPGA-based implementation of median and weighted median filters for image processing,” in *Proceedings of International Conference on Field Programmable Logic and Applications, FPL*, 2005, pp. 142–147.

[37] Matlab source code for image impulse noise removal, available:"*https://www.researchgate.net/profile/Zohreh_Hosseinkhan*".

[38] Brain MRI Images, [Online]. Available: “*http://overcode.yak.net/15?size=M*”, [Accessed: March. 10, 2017].

Table I. Comparison between results of different denoising algorithms using PSNR (dB) for different noise densities.

Noise density	5%	10%	15%	20%	30%	40%
Median3×3	34.27	33.17	31.14	28.40	22.89	18.41
Median 5×5	30.18	29.97	29.66	29.28	27.76	23.77
***Matsubara et al* [21]**	38.27	35.65	32.18	28.65	22.67	18.13
***Lien et al* [32]**	36.18	34.93	33.78	32.48	29.62	26.18
***Hosseinkhani et al* [34]**	**38.65**	**37.07**	**35.43**	33.52	28.43	22.86
Proposed Method	38.11	36.61	35.27	**33.96**	**30.90**	**26.44**

Table II Comparison between denoised results in terms of PSNR (dB) for different images in 10% and 20% noise density.

	10%			**20%**		
Method	**Lena**	**Goldhill**	**Peppers**	**Lena**	**Goldhill**	**Peppers**
Bhadouria *et al* [17]	32.92	30.04	--	--	--	--
Bhadouria *et al* [33]	37.89	35.20	--	33.48	31.72	--
Turkmen [20]	--	--	--	33.84	31.55	--
Javed *et al* [18]	36.45	--	35.68	34.06	--	33.03
Lien *et al* [32]	36.49	33.23	35.91	33.58	31.73	33.59
Hosseinkhani *et al* [34]	37.81	34.62	**36.96**	34.89	**32.48**	33.99
Deka *et al* [12]	37.75	--	--	34.46	--	--
Khan *et al* [13]	33.86	--	--	32.88	--	--
Lin [15]	--	--	--	34.27	--	33.81
Wu *et al* [14]	32.8	--	32.4	31.4	--	31.4
Proposed Method	**37.89**	34.58	36.67	**35.01**	**32.48**	**34.13**

Table III Comparison between denoised results in terms of PSNR (dB) for different images in 30% and 40% noise density.

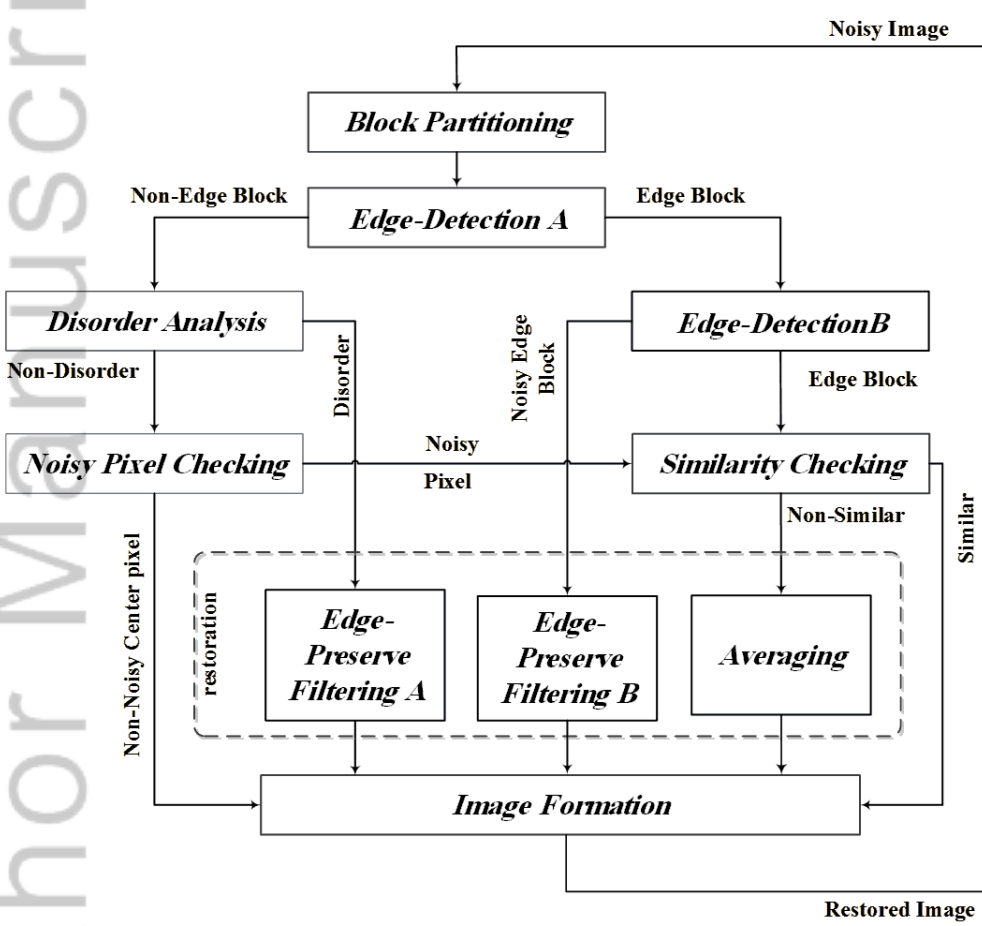
	30%			**40%**		
Method	**Lena**	**Goldhill**	**Peppers**	**Lena**	**Goldhill**	**Peppers**
Bhadouria *et al* [17]	31.25	29.24	--	--	--	--
Bhadouria *et al* [33]	--	--	--	--	--	--
Turkmen [20]	--	--	--	**31.79**	**29.46**	--
Javed *et al* [18]	32.01	--	**31.59**	30.01	--	**29.26**
Lien *et al* [32]	31.16	30.16	31.08	29.00	28.29	28.69
Hosseinkhani *et al* [34]	31.61	30.00	30.91	27.90	26.67	27.17
Deka *et al* [12]	32.20	--	--	29.39	--	--
Khan *et al* [13]	30.74	--	--	29.52	--	--
Lin [15]	**32.16**	--	31.61	29.16	--	28.59
Wu *et al* [14]	29.9	--	29.9	28.35	--	28.89
Proposed Method	32.12	**30.56**	**31.59**	29.35	28.23	28.79

Table IV Comparison between denoised results in terms of PSNR (dB) for different images in 50% noise density.

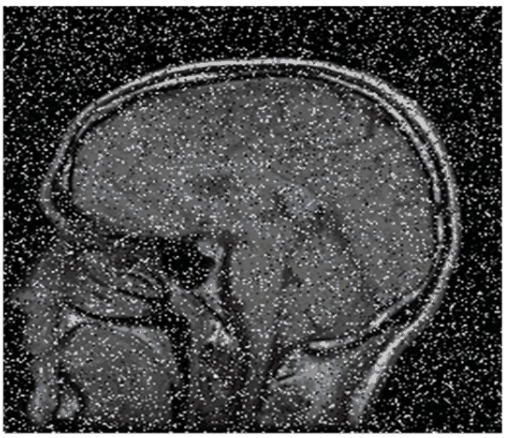
Method	**Lena**	**Goldhill**	**Peppers**
Bhadouria *et al* [17]	**28.69**	**27.76**	
Bhadouria *et al* [33]	--	--	
Turkmen [20]	--	--	
Javed *et al* [18]	27.87		**27.43**
Lien *et al* [32]	26.37	26.22	25.75
Hosseinkhani *et al* [34]	24.08	23.53	23.29
Deka *et al* [12]	26.16		
Khan *et al* [13]	27.96		
Lin [15]	25.63		24.76
Wu *et al* [14]	26.39		27.02
Proposed Method	26.18	25.58	25.53

Table V Comparison of implementation specifications between proposed method and methods of [21], [32], [33]

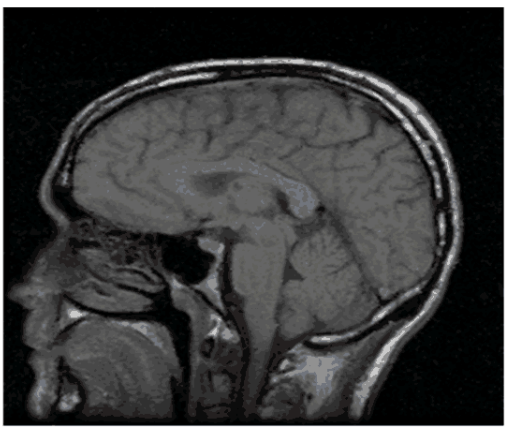
Method	Target Device	Area	Delay (ms)	Average PSNR in 5%, 10%, 15% and 20 % noise
***Matsubara et al* [21]**	Altera Cyclone II EP2C20F484C7N	513 (Logic cell)	7.72	33.68 dB
***Lien et al* [32]**	Altera FLEX10KE EPF10K200-SRC240-1	2166 (Logic cell)	14.90	34.34 dB
***Bhadouria et al* [33]**	Xilinx virtex7 XC7K325T	~10500(LUT) 7520 (FF)	1.22	---
Proposed Method	Xilinx virtex4 xc4vfx12-12-sf363	2761 (Slice)	12.91	**35.98 dB**



(a)

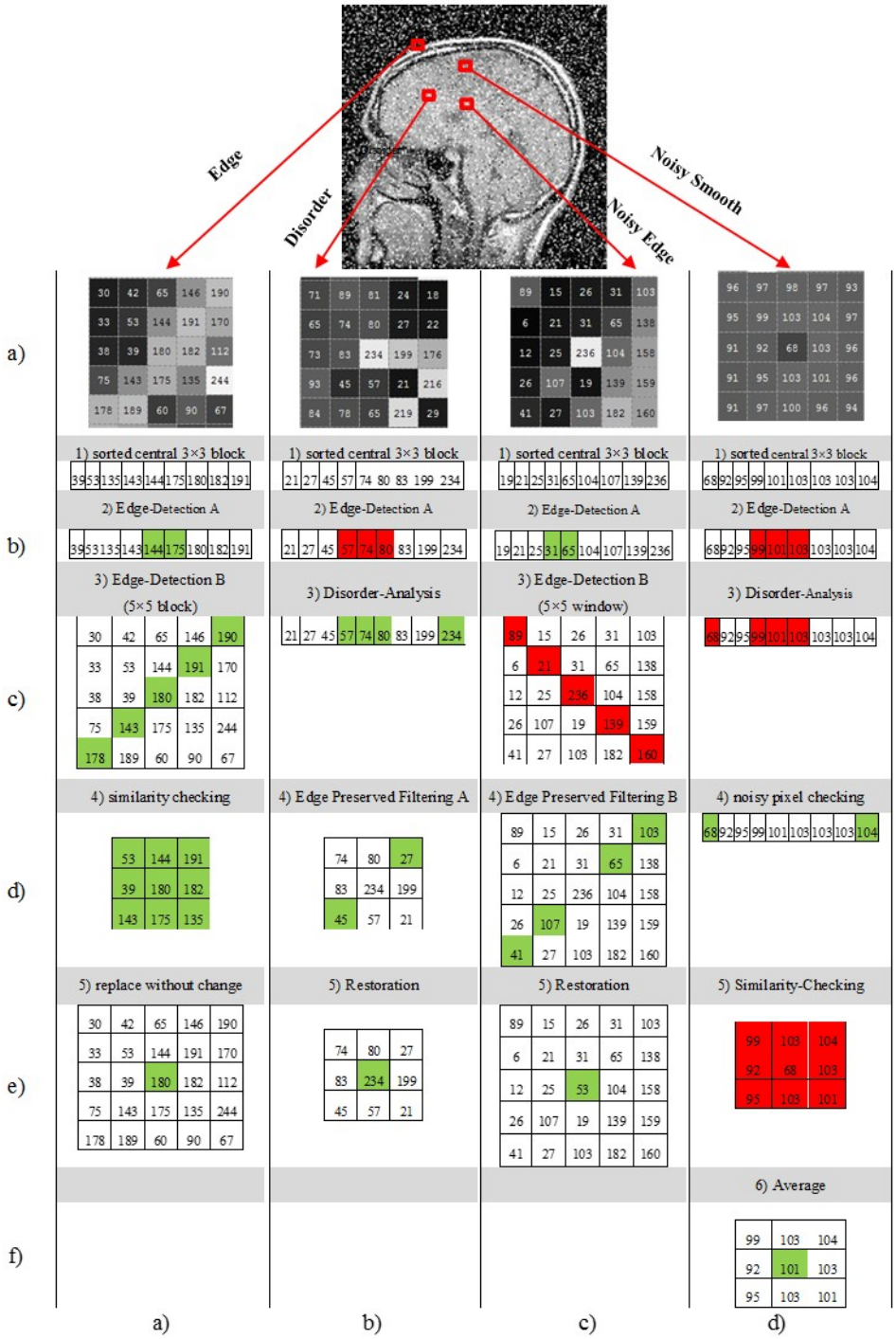


(b)

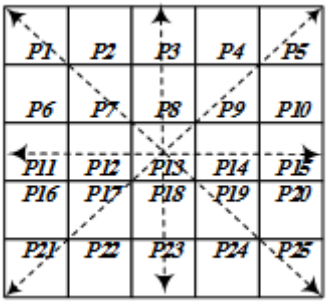


(c)

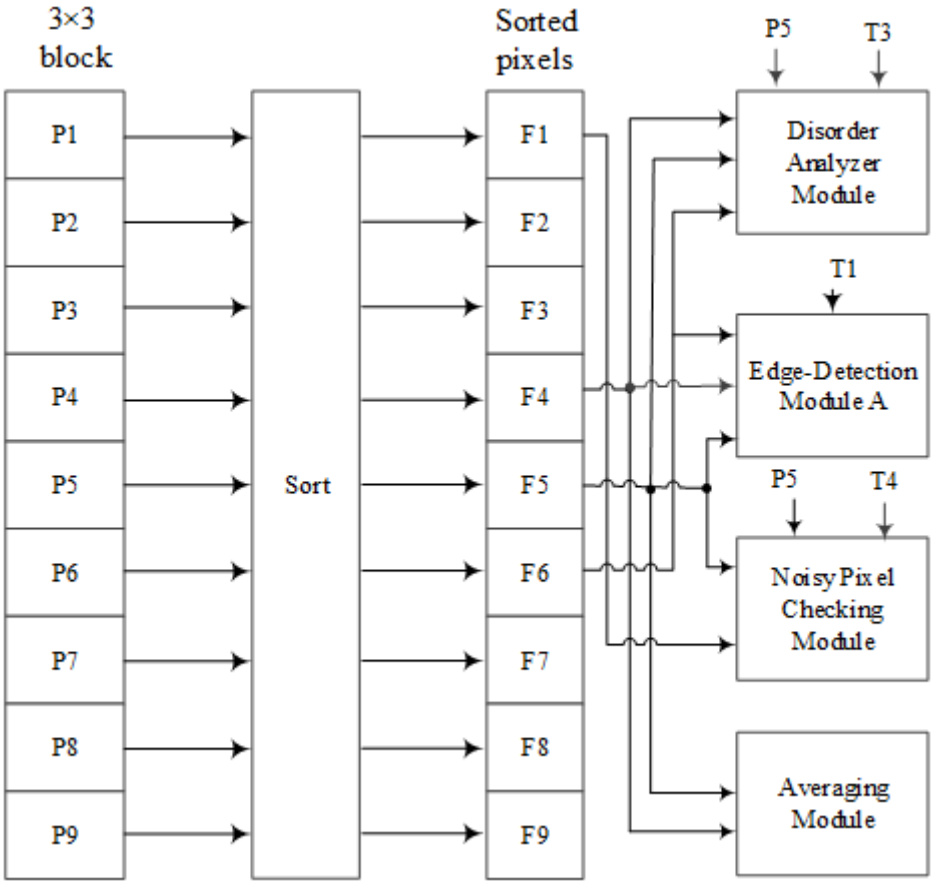
CTA_2591_F1.tif



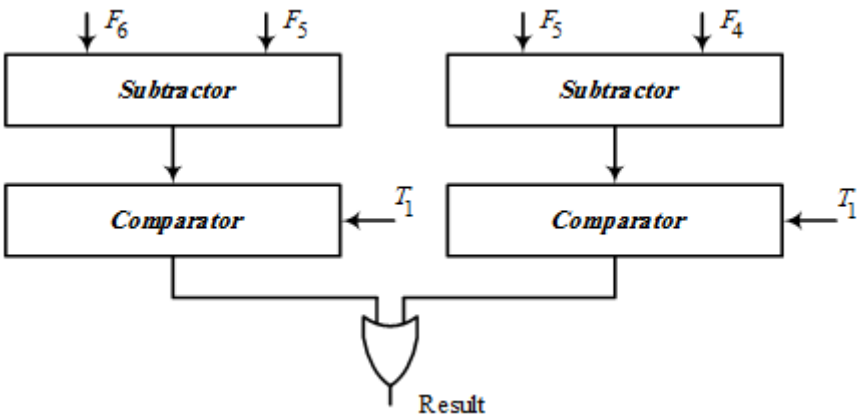
CTA_2591_F2.tif



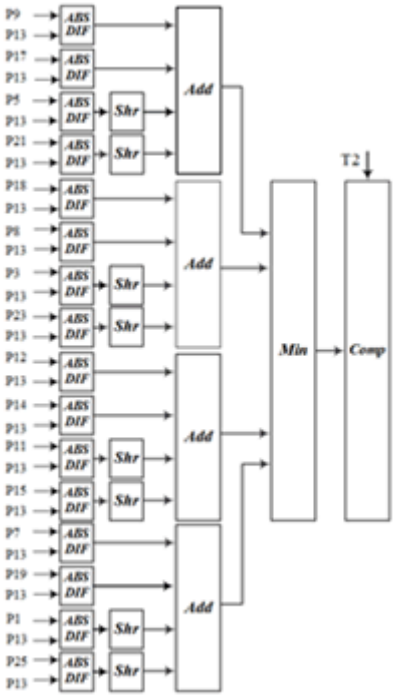
CTA_2591_F3.tif



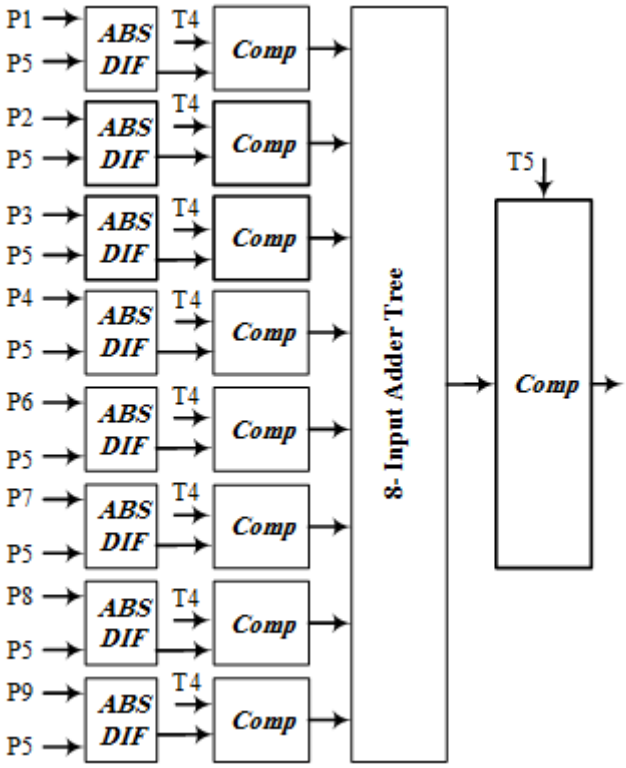
CTA_2591_F4.tif



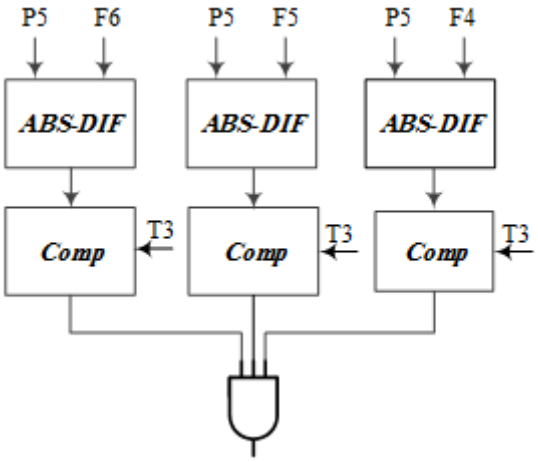
CTA_2591_F5.tif



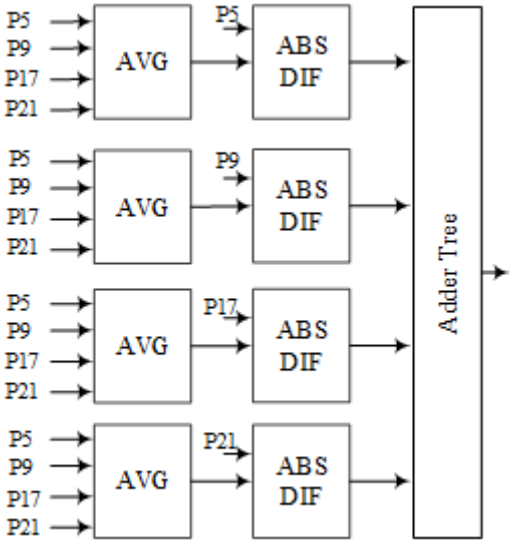
CTA_2591_F6.tif



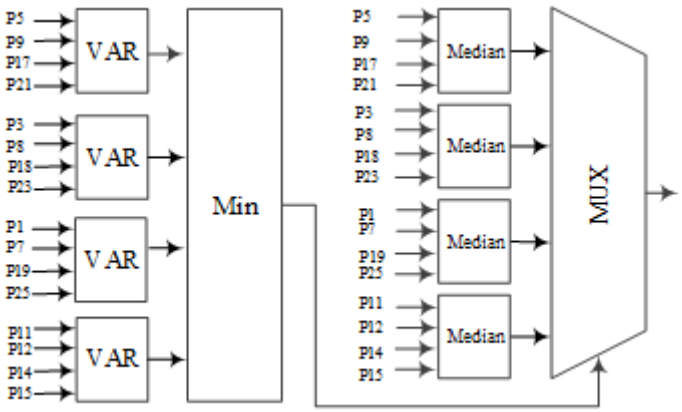
CTA_2591_F7.tif



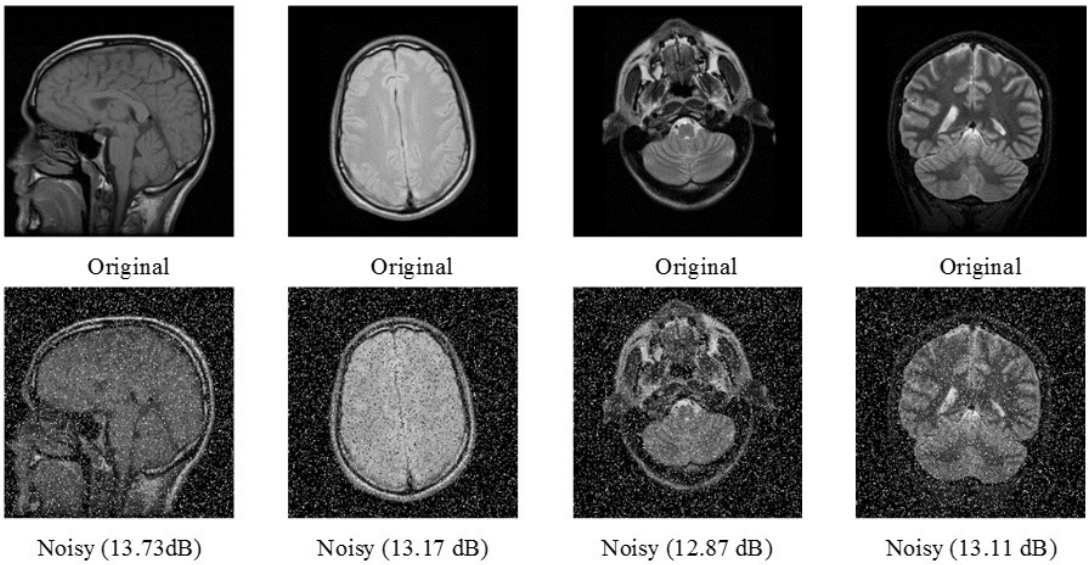
CTA_2591_F8.tif



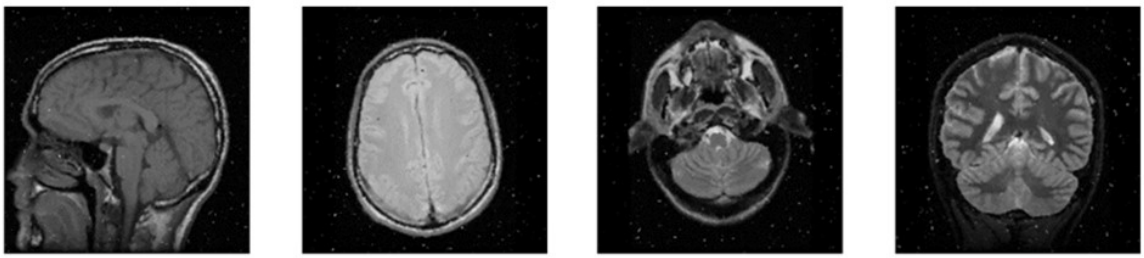
CTA_2591_F9.tif



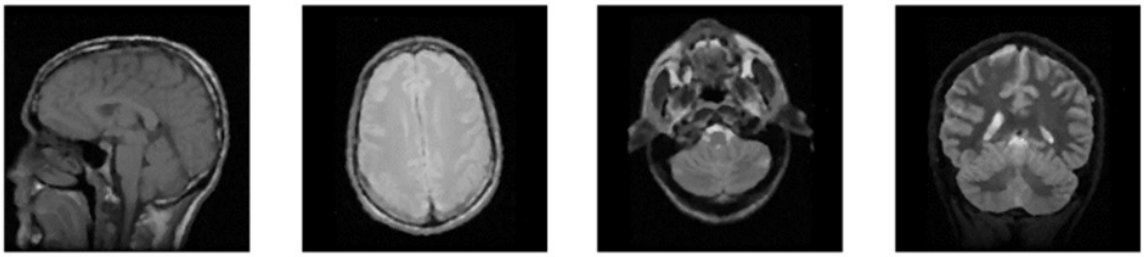
CTA_2591_F10.tif



CTA_2591_F11.tif

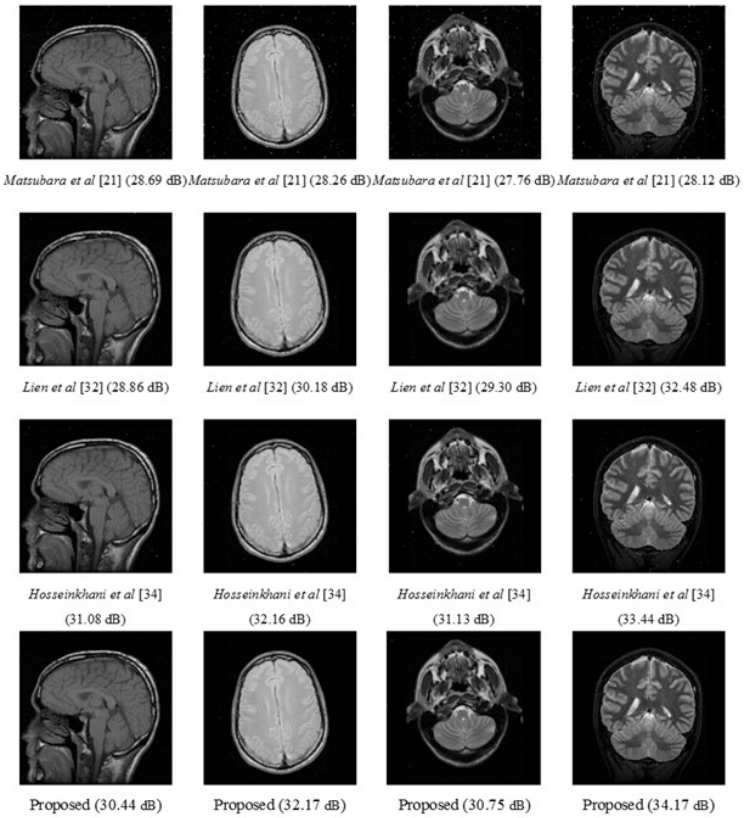


Median 3×3 (27.72 dB) Median 3×3 (27.86 dB) Median 3×3 (27.07 dB) Median 3×3 (28.44 dB)



Median 5×5 (26.07 dB) Median 5×5 (27.25 dB) Median 5×5 (26.78 dB) Median 5×5 (28.97 dB)

CTA_2591_F12.tif



CTA_2591_F13.tif

Original Lena

Original Peppers

Original Goldhill

Noisy Lena (13.17 dB)

Noisy Peppers (12.81 dB)

Noisy Goldhill (13.00 dB)

CTA_2591_F14.tif

Median 3×3 (24.53 dB)

Median 3×3 (23.48 dB)

Median 3×3 (23.54 dB)

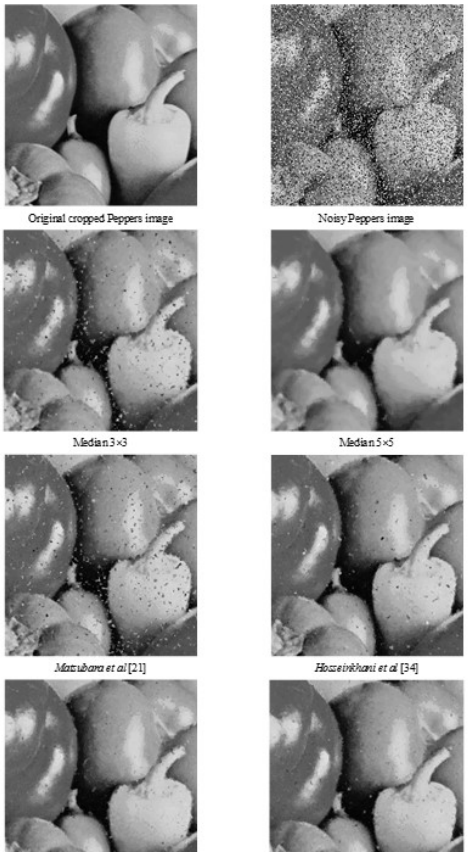
Median 5×5 (27.03 dB)

Median 5×5 (26.18 dB)

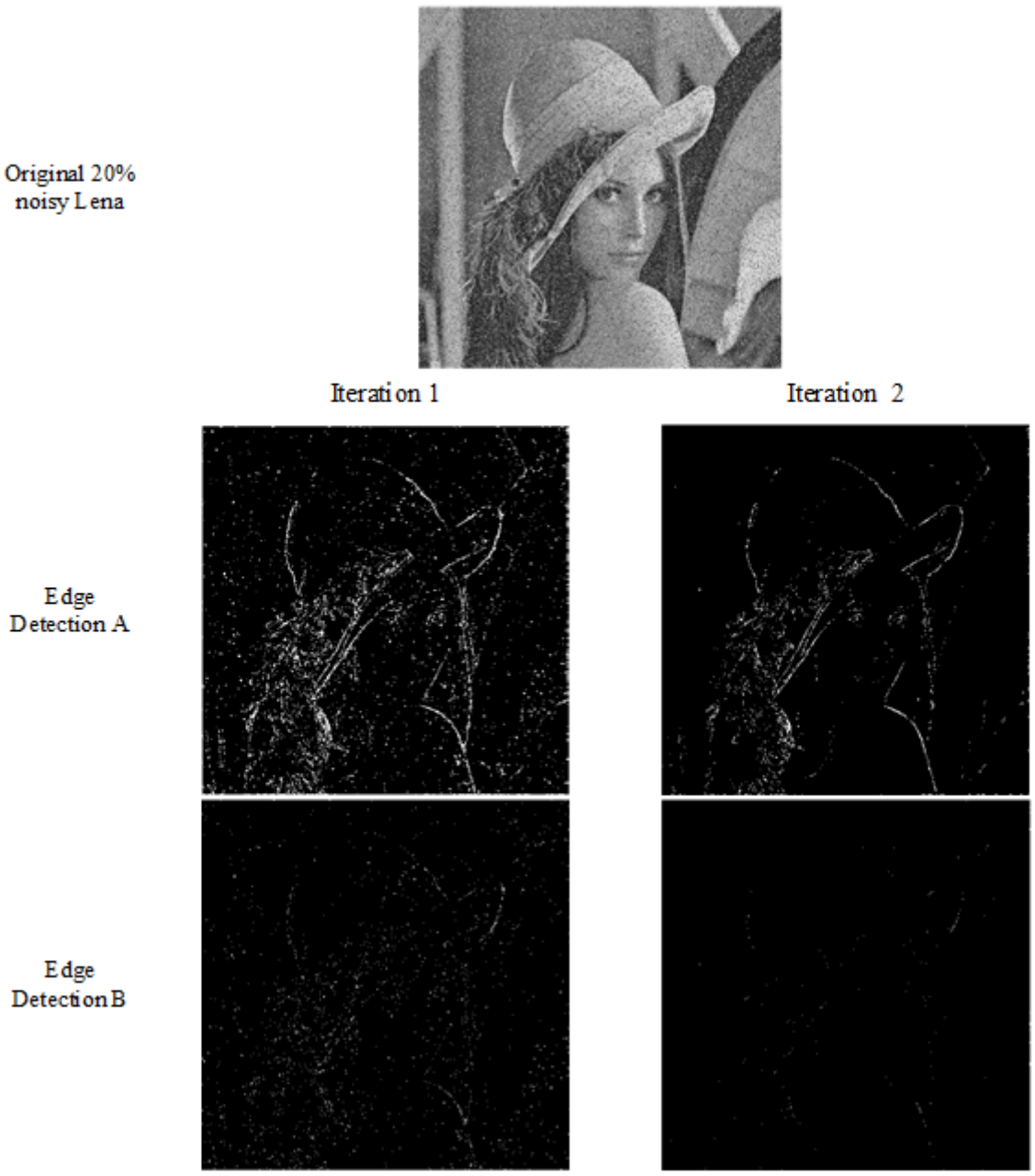
Median 5×5 (25.53 dB)

CTA_2591_F15.tif

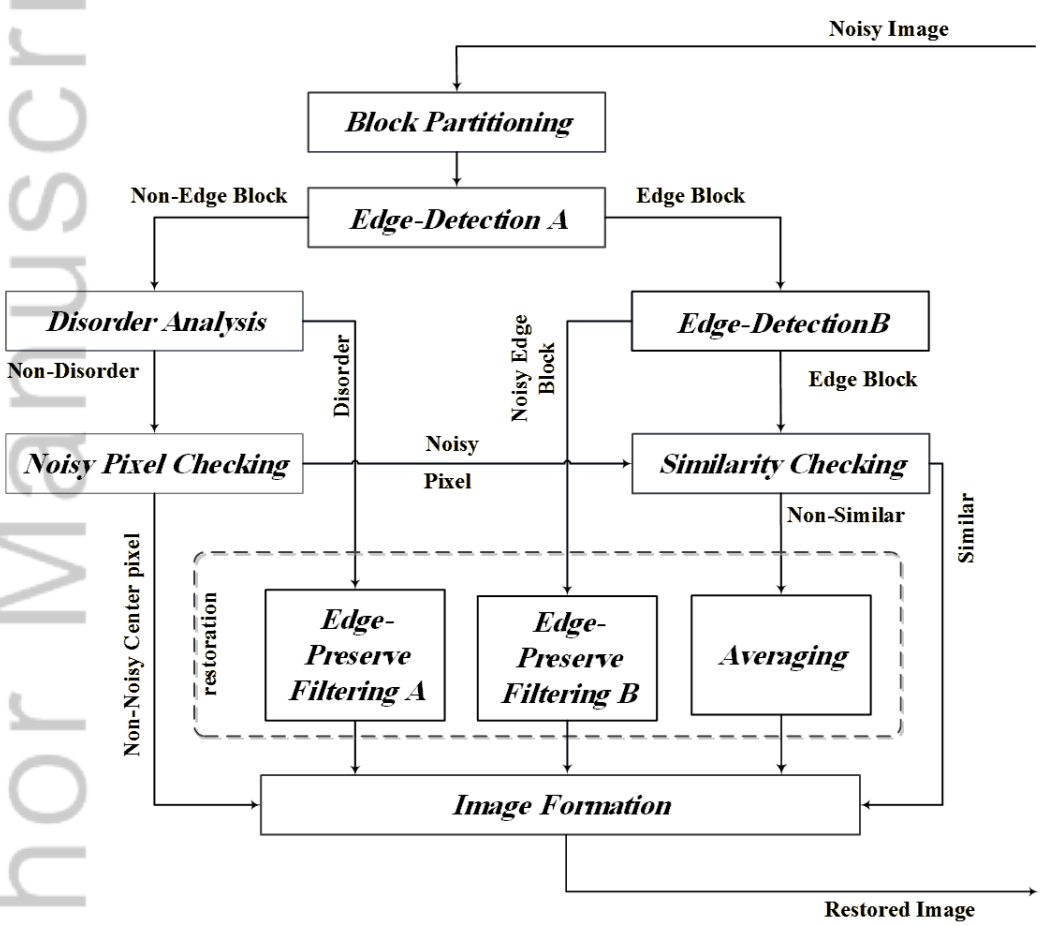
CTA_2591_F16.tif



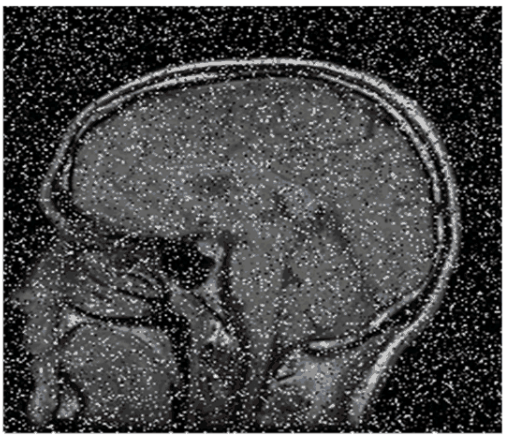
CTA_2591_F17.tif



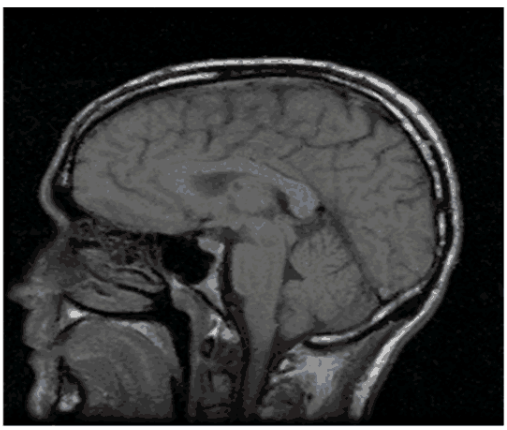
CTA_2591_F18.tif



(a)



(b)



(c)

CTA_2591_Graphical abstract.tif