Decision Tree Based Denoising Method for removal of Random valued impulse noise

Abstract - Since the inception of Digital image processing, noise removal in images has always been a challenge to researchers and experts of the field. Most noteworthy of these noises are the randomly varying impulse noises produced during image acquisition and transmission. Over the period of time, different methods have been found to tackle different noises in images. However, these conventional methods fail to address the various types of impulse noises. Hence, a need for efficient denoising method has led to extensive research and development of various innovative methods to denoise the impulse noises in images. Some of them are Switching median filter (SMF), alpha-trimmed mean-based method (ATMBM), Rank ordered relative differences based impulse detector (RORD), differential rank impulse detector (DRID), Decision based un-symmetric trimmed median filter (DBUTM) and directional weighted median (DWM). All of which concentrate only on denoising impulse noises. Of the available methods, a need for an efficient and simple denoising mechanism still exists for random valued impulse noise, a noise which is more difficult to denoise than salt and pepper noise. For this, we employ a new method which detects and filters random valued impulse noise in images. The method proposed in this paper uses a decision tree based impulse detector and an edge preserving filter to reconstruct the noise free image. The method requires only less memory for storage due to its lower complexity and is more efficient than the existing low complexity techniques.

Key Terms - Impulse noise, Salt and Pepper noise, Random valued Impulse noise, Effective noise Removal, Decision tree, Edge preserving filter.

I. Introduction

With the advent of digital images, productivity in science and technology has greatly increased due to subsequent better understanding capabilities, analysis, visualization and interpretation which are not only limited to a particular field but can be extended to as many as possible. However, with these advantages came the limitations of using digital images. Prominent among these limitations with usage of digital images include noise and storage problems. Noise in images mainly occurs during acquisition and transmission. [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [15]. Presence of noise in images hinders all the advantages with using digital images as it deteriorates the quality of images making it difficult to analyze, interpret or visualize. This led to the development of effective denoising algorithms which target various types of noise in images. Today, image processing not only concentrates on image interpretation, analysis and visualization but also on denoising of images. Thus, denoising is a key part of image processing which involves noise detection followed by its removal. Among the discovered image noises, most irksome are the impulse noises which affect the images during their acquisition and transmission. Impulse noises generally involve random occurrence or distribution of noisy pixels over the image. However, based on the values of its noisy pixels these are divided into two categories as fixed value impulse noise (or Salt and Pepper noise) and random valued impulse noise, both of which involve random distribution of noisy pixels over the image but have different pixel values. [1], [2], [6], [9], [10]. Fixed value impulse noise (or Salt and Pepper noise) is limited to only

two noisy pixel values (i.e.) salt - 255 and pepper - 0 (for an 8 bit Image) which are distributed randomly over an image. [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. On the other hand, random valued impulse noise involves noisy pixels of any value within the range [0 255] (if 8 bit images are considered) which are distributed randomly over the image values. [1], [2], [6], [9], [10]. It is evident that denoising random valued impulse noise is far more challenging than denoising Salt and Pepper noise due to its random pixel values. Of all the various methods available to denoise impulse noise [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], very few address the random valued impulse noise. Thus, there exists a need for development of an efficient denoising scheme for removal of random valued impulse noise in images.

Mean filters perform denoising effectively but at the cost of heavy information loss due to blurring or smoothening of the image. Median filters are preferred to preserve several important details even after denoising. However, median filters do not serve their purpose when at high noise densities. [7], [8], [9], [10], [13], [14]. Although not as much as mean filters, median filters result in information loss too. [1], [2], [3], [4], [6], [7], [8], [9], [11]. Hence to avoid this we opt for a switching median concept. Here, the median filter is provided with an impulse detector which detects the impulses (noisy pixels) prior to the filter. Thus, by detecting the noisy pixels before filtering we may condition the filter such that it filters only those noisy pixels that are detected, thereby resulting in less information loss and better preservation of details in the image which amount to better quality in terms of image characteristics as well as visual perception.

In today's world digital images and their processing are not only limited to the scientific or defense communities but are used in commercial applications which include entertainment, advertising, etc. as well as fields such as medical applications, education, infotainment, etc. Thus, for an effective implementation of the technology (image denoising) we require efficient denoising schemes which are not just effective in their performance but are of lower complexity too, enabling us to implement it to as many real time applications as possible. The complexity of a denoising scheme can be determined by three important factors that affect its performance viz. a) The size of the window used, b) The size of the code (or memory required to store) and c) The total number of iterations involved. [1]. Algorithms of lower complexity use lesser or smaller versions of the above attributes while those of a relatively higher complexity involve usage of higher attributes (larger window, bigger code and more iterations). This reduced complexity in low complex methods is achieved at a cost of low performance when compared to high complex methods.

Keeping in view of the above requirements, in this paper, we propose a model which is of low complexity but has an output performance comparable to those of the high complexity. The proposed method employs a decision tree based impulse detector and an edge preserving filter to denoise random valued impulse noise in an image. The rest of the paper is organized as follows. In section-II, the related work is discussed followed by the system model in section-III. Section-IV presents the methodology, while, section-V discusses the implementation. Section-VI and section-VII comprise of the results, conclusions and future scope.

II. Related work

Lien et al [1] proposed a method to denoise random valued impulse noise in images with the help of a decision tree based denoising method. The method uses a decision tree based impulse detector to detect the corrupted pixels and an edge preserving filter to denoise them. Yiqiu Dong and Shufang Xu [2] proposed another method to denoise random valued impulse noise in images. This method introduces a new impulse detector which works using the directional indexes within a 5x5 mask to locate the impulses within the image. The detected impulses are now filtered using a directional weighted median filter. Pei-Yin Chen and Chih-Yuan Lien [3] presented a new algorithm for removal of salt and pepper noise in images. The algorithm uses an efficient impulse detector which detects the salt and pepper noise in images by using two thresholds N_{max} & N_{min}. It also introduces an edge preserving filter which works based on directional correlations to denoise the detected impulse noise in the images. Yu et al [4] proposed a new efficient procedure for removing random valued impulse noise in images. The proposed method employs an impulse detector which uses the statistic of rank-ordered relative differences to identify corrupted pixels in an image and a simple weighted mean filter to denoise the located noise. Wenbin Luo [5] introduced an efficient detail preserving approach for removal of impulse noise in images. The introduced method uses an alpha trimmed mean approach for impulse noise detection in an image. It then refines the image and finally cancels the impulse noise by replacing the corrupted pixels with estimated ones within the image. Pei-Eng Ng and Kai-Kuang Ma [6] proposed a method for denoising images corrupted by high density impulse noise. The method introduces four different noise models for impulse noise and aims to denoise images corrupted by each of these impulse noise models. The method employs a step by step classification of pixels into corrupted and uncorrupted pixels making use of windows of sizes 3x3 and 21x21, to detect the impulse noise and denoises the detected noise using a noise adaptive filtering technique. K.Aishwarya et al [7] presented a new and efficient technique for removal of high density salt and pepper noise in images. The proposed method makes use of Shear sorting to find the median of a window and proceeds to filter the corrupted pixels with the help of Decision based un symmetric trimmed median filter (DBUTM). The DBUTM checks for impulses at the extreme ends of an array of sorted pixels and then replaces the corrupted pixels with the computed median of the window. Md. Imrul Jubair et al [8] proposed a new and enhanced decision based adaptive median filter to denoise images corrupted by salt and pepper noise. The proposed method employs an adaptive windowing approach to vary the size of the window used for spatial processing by taking into consideration the local noise densities. By using this approach only noise free pixels are used for median calculation in the local window. Finally, the corrupted pixels are replaced by the calculated median from the adaptive windowing approach. Dr. T. Santhanam and Ms. K. Chithra [9] presented a new algorithm for removal of high density salt and pepper noisy in images. The algorithm uses a decision based un symmetric trimmed median filter using Euclidean distance as a major parameter while denoising images corrupted by high density salt and pepper noise. The algorithm uses a 13x13 window and compares the pixel of interest with 0 and 255 to detect

the corrupted pixels in the image. If the window has all 0's, 255's or both it uses mean to replace the centre pixel. if the window has not all 0's or 255's then it eliminates all of them and replaces the corrupted pixels with the uncorrupted pixels that have the smallest Euclidean distance from the pixel of interest (centre pixel). Arabinda Dash and Sujaya Kumar Sathua [10] introduced an efficient algorithm for removal of high density salt and pepper noise in images using a cascaded system of filters. The algorithm uses a Decision based median filter for preliminary noisy removal and a Modified Decision based Partial trimmed global mean filter or a modified decision based un symmetric trimmed median filter to remove the remaining noise and enhance the image quality. S.Vishaga and Sreejith .L .das [11] presented a survey on various available variants of median filters such as weighted median filter, adaptive switching median filter, tri state switching median filter, modified decision based un symmetric tri state median filter and discussed on each of their advantages and disadvantages.

III. System Model

Figure.1 illustrates the total system model for the proposed algorithm. The figure shows how an image (noisy) goes through various modules and gets reconstructed at the end.

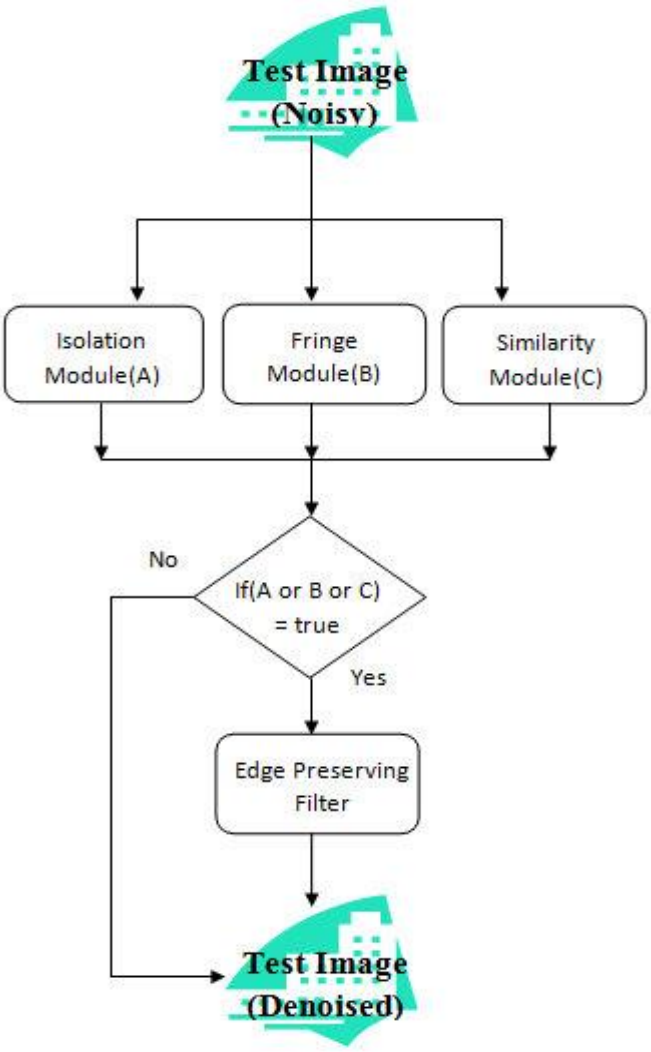


Fig.1 : A Model of the Proposed method

IV. Methodology

4.1 Forming a Mask

In this method, we consider a 3x3 mask using which noise detection and filtering is performed. The mask considered is made up of nine pixels, of which the centre pixel is of the utmost importance as it is only the centre pixel onto which the final filtering is performed. The algorithm is so designed that the mask moves over the entire image, thereby, covering as many pixels as possible. To form a mask (3x3) we always need a centre pixel and its eight neighbours. However, this requirement is not always fulfilled, pixels at the border of the image fail to form masks as they do not posses enough neighbours to form a mask. Thus, only those pixels that are within (next column and row) the border pixels are filtered. To address this problem, we may use border correction techniques wherein the border pixels are replaced with their immediate neighbours that fall under the range of filtering.

Now, if the considered centre pixel is located at a coordinate (i, j) then its neighbours range from (i-1 : i+1, j-1 : j+1). The centre pixel is denoted by '$p_{i,j}$', while, its luminance is denoted by '$f_{i,j}$'. The neighbours are named accordingly in the range of (i-1 : i+1, j-1 : j+1).

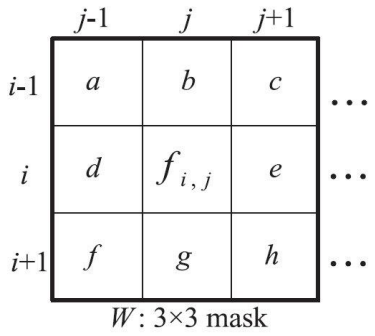


Fig.2 : A 3x3 mask (refer : [1])

4.2 Components of DTBDM

This method involves two main components viz., Impulse Detector and Edge Preserving filter. The impulse detector makes use of a 3x3 mask, to detect the noisy pixels and to activate the filter accordingly. The filter operates on the centre pixel and replaces it with the final obtained value. If the centre pixel is found to be noise free, then, it is used in the reconstruction of the final image.

4.3 Impulse Detector

The impulse detector detects the corrupted pixels so as to activate the filter only for those pixels that are corrupted, thereby, avoiding unnecessary filtering and the resulting information loss. Here, we use an impulse detector designed based on a decision tree. A

decision tree is a way of simplifying a complex decision by resolving it into multiple simpler decisions, used mainly in multiple variable analysis. The complex decision in this context is "Whether the pixel is corrupted or not?". This decision is now resolved into three simpler decisions (i.e.)., (i) "Is the pixel Isolated?", (ii) "Is it an edge pixel?" and (iii) " How is the similarity of the pixel with its neighbourhood?". Thus, the impulse detector comprises of three modules based on the above decisions viz., Isolation module, Fringe module and Similarity module. Each of these modules are independent of each other and can judge whether a pixel is corrupted or not. However, to increase the performance of the detector we use all the three modules in a parallel fashion ('OR' logic), wherein if any of the three detect the pixel to be corrupted it is filtered, while, the pixel is considered noise-free only if all three modules judge it to be un corrupt.

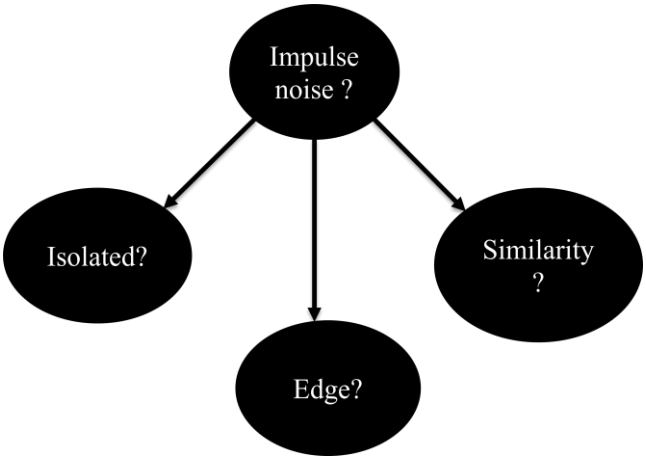
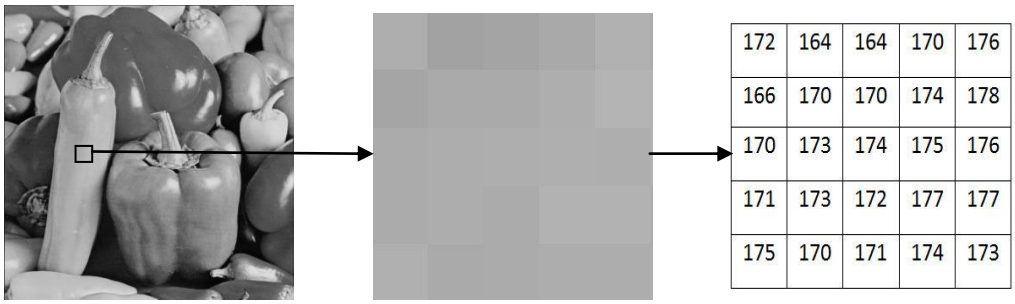


Fig.3 : Decision tree in impulse detector

4.3.1 Isolation Module

The isolation module is the first among the three modules that we use in this algorithm (Although, no sequence is necessarily followed). The isolation module is considered first among the three for the fact that it is the easiest and the most obvious way of determining whether a pixel is corrupted or not. The strength of this algorithm lies in the fact that it is closer to the human perception or simply, closer to the way we would determine an image to be noisy or noise free just by looking at it. When we look at an image, say, a white paper we would call it a clean (noise free) paper only if the entire paper is white, else, in the case of the presence of a black spot we would immediately come to the conclusion that it is dirty (noisy). Here, detecting the presence of noise, which is a spot in this case, is possible from the fact that it has a high difference in intensity when compared with its neighbours (the rest of the paper). This is the basic logic we follow in this module. Any isolated pixel can be identified by computing the intensity differences of the pixel with its neighbours. However, this logic is not always valid. The isolation logic is based on the prime assumption that the centre pixel is located on a smooth surface. To understand this let us go back to the example of the white paper. The black spot would become a noise only if the rest of the paper is completely white, else, if the paper has several such spots or if it has a pattern of such spots, then, it would be

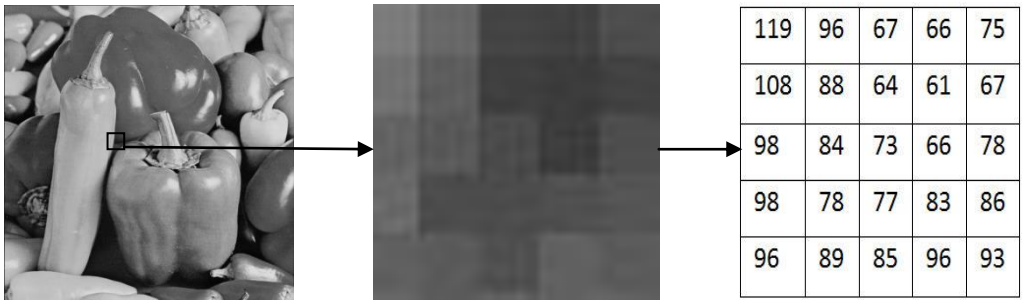
difficult to determine whether the spot found is a noise or a part of the pattern on the paper. Thus, to test the pixel for isolation, first, it has to be on a uniform surface.



172	164	164	170	176
166	170	170	174	178
170	173	174	175	176
171	173	172	177	177
175	170	171	174	173

(4a) Original image (4b) Region in the Square (4c) Gray Scale values

Fig.4 : A uniform surface in Peppers image



119	96	67	66	75
108	88	64	61	67
98	84	73	66	78
98	78	77	83	86
96	89	85	96	93

(5a) Original image (5b) Region in the Square (5c) Gray Scale values

Fig.5 : A non-uniform surface in Peppers image

Practically, the impulse module can be easily implemented using the 3x3 mask. The mask is first split into two halves $W_{TopHalf}$ and $W_{BottomHalf}$. Now, the nine pixels in the mask can be named as 'a, b, c, d, e, f, g, h and $f_{i,j}$', where, '$f_{i,j}$' denotes the centre pixel and the variables 'a to h' denote the eight neighbours of the centre pixel. '$W_{TopHalf}$' comprises of the top four neighbours 'a to d', while, '$W_{BottomHalf}$' comprises of the four neighbours 'e to h', at the bottom of the mask. They are represented as

$$W_{TopHalf} = \{a, b, c, d\}.$$

$$W_{BottomHalf} = \{e, f, g, h\}.$$

Since, it is a must for the pixel to be present in a uniform region, before we test for its isolation, a test for the uniformity of the region is always performed first. This test for uniformity can be performed by finding out the maximum possible difference in intensities of both the top and bottom halves of the mask (i.e.)., the difference between maximum and the minimum intensity values in both the halves. If either of these differences in intensity is very high (i.e.)., greater than a threshold, then, the region is considered as a non uniform region and the test for isolation cannot be performed. The test can be represented using the following equations

$$TopHalf\_diff = TopHalf\_max - TopHalf\_min.$$

$$BottomHalf\_diff = BottomHalf\_max - BottomHalf\_min.$$

$$Decision1 = \begin{cases} true, & if\ (TopHalf\_diff \geq Th\_IM_a) \\ & or\ (BottomHalf\_diff \geq Th\_IM_a) \\ false, & otherwise. \end{cases}$$

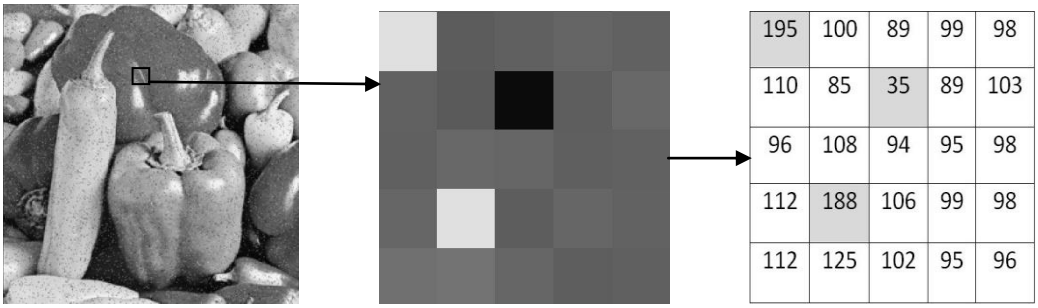
Decision1 determines whether the centre pixel lies in a uniform region or a non uniform region. If the region is found to be uniform then we proceed to the test for isolation. The centre pixel is now taken into consideration and the intensity difference between the centre pixel and the TopHalf_max, TopHalf_min, BottomHalf_max & BottomHalf_min are computed respectively. If any of the differences is found to be greater than a threshold, then, the centre pixel is considered as an isolated pixel. It can be represented as

$$IM\_TopHalf = \begin{cases} true, & if\ (|f_{i,j} - TopHalf\_max| \geq Th\_IM_b) \\ & or\ (|f_{i,j} - TopHalf\_min| \geq Th\_IM_b) \\ false, & otherwise. \end{cases}$$

$$IM\_BottomHalf = \begin{cases} true, & if\ (|f_{i,j} - BottomHalf\_max| \geq Th\_IM_b) \\ & or\ (|f_{i,j} - BottomHalf\_min| \geq Th\_IM_b) \\ false, & otherwise. \end{cases}$$

$$Decision2 = \begin{cases} true, & if\ (IM\_TopHalf = true) \\ & or\ (IM\_BottomHalf = true) \\ false, & otherwise. \end{cases}$$

Decision2 determines whether the centre pixel is isolated or not. It is to be noted that we arrive at Decison2 only when Decision1 is false.



195	100	89	99	98
110	85	35	89	103
96	108	94	95	98
112	188	106	99	98
112	125	102	95	96

(6a) Corrupted image (6b) Region in the Square (6c) Gray Scale values

Fig.6 : A 5x5 mask containing isolated pixels in the Peppers image.

4.3.2 Fringe module

Edges can be regarded as the most important feature of an image as in most of the cases, they tend to give us more information than any other feature of an image. Due to excessive filtering using filters like mean filter, we may end up losing this information. However, this can be avoided by using a proper detecting mechanism that detects the edges in an image and avoids filtering them unnecessarily. The fringe module in this algorithm is one such effective mechanism that detects the edges and avoids their filtering unnecessarily. To avoid unnecessary filtering of edges in an image one has to detect these edges in an image and differentiate them from the noise. Detection of edges in an image and differentiating them from noise is not always easy, which is due to the fact that most of the edge pixels always end up being the high frequency content of an image (i.e.)., they always seem to be isolated when compared to their neighbouring non-edge pixels. But, there also exists a clue to solving this problem in the fact that although edge pixels may seem isolated when compared to the non-edge pixels, they are always of lesser intensity differences when compared to their neighbouring edge pixels (i.e.)., if an intensity difference is computed along the direction of an edge then it always ends up to be less. This can be practically implemented using a 3x3 mask. Since, we are only concerned about the centre pixel, we arrive at four directions E1 to E4 passing through the centre pixel using which we can determine whether the centre pixel is an edge pixel or not.

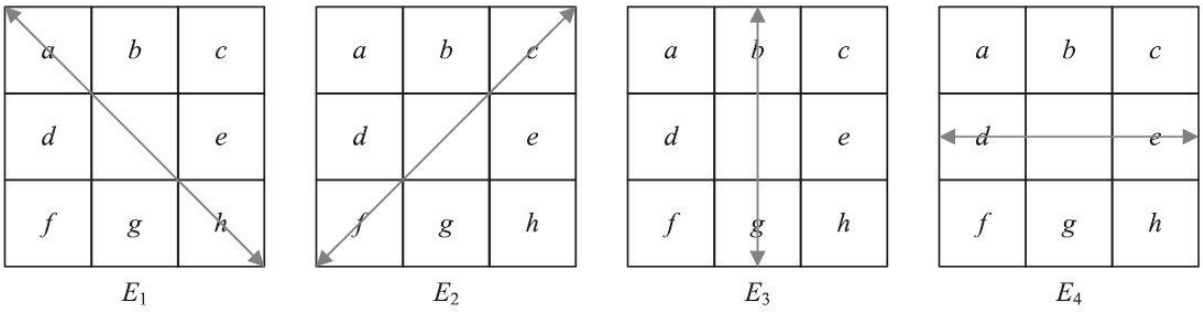


Fig.7 : The Four directions along which directional differences are calculated. (refer[1])

The directional differences are computed as follows

$$FM\_E1 = \begin{cases} false, & \text{if } (|\, a - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, h - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, a - h \,| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases}$$

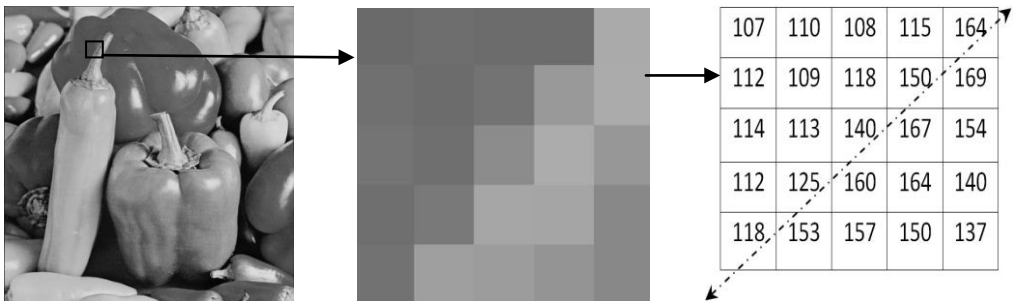
$$FM\_E2 = \begin{cases} false, & \text{if } (|\, c - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, f - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, c - f \,| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases}$$

$$FM\_E3 = \begin{cases} false, & \text{if } (|\, b - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, g - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, b - g \,| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases}$$

$$FM\_E4 = \begin{cases} false, & \text{if } (|\, d - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, e - f_{i,j} \,| \geq Th\_FM_a) \\ & \text{or } (|\, d - e \,| \geq Th\_FM_b) \\ true, & \text{otherwise.} \end{cases}$$

$$Decision3 = \begin{cases} false, & \text{if (FM\_E1) or (FM\_E2)} \\ & \text{or (FM\_E3) or (FM\_E4)} \\ true, & \text{otherwise.} \end{cases}$$

By computing these directional differences we can easily trace the edges and determine whether the centre pixel is an edge pixel or not. Decision3 gives us the final judgment on the centre pixel (i.e.)., whether it is noisy or uncorrupted. As per Decision3, if any of the directional differences FM_E1 to FM_E4 give us a true (i.e.)., shows existence of an edge, then, Decision3 is made false (i.e.)., the centre pixel is uncorrupted, else, the centre pixel is considered as corrupted. Since, we are using a parallel logic each module is expected to be independent and self-sufficient, so as to deliver the judgment onto the centre pixel. As the module's testing is more focused on whether the pixel '$P_{i,j}$' is an edge pixel or not but not much on the presence of noise, the independency and self-sufficiency of the module is questionable. However, this can be answered from the fact that if the centre pixel is found to be a non-edge pixel, then, it also means that it is isolated from its neighbours. This can be understood by observing the four equations from FM_E1 to FM_E4, where, the intensity difference between the centre pixel and each of its neighbours is found. Thus, the centre pixel can be determined as a non edge pixel only if all the directional differences are greater than the thresholds, which in turn means that it is isolated.



(7a) Original image (7b) Region in the Square (7c) Gray Scale values(edge)

Fig.7 : A 5x5 mask containing an edge in the Peppers image.

4.3.3 Similarity Module

The similarity module basically determines whether the centre pixel belongs to the general range of pixel values within the image or not, thereby, it concludes whether the pixel is noisy or uncorrupted. Since, we are dealing with random valued impulse noise a noisy pixel may have any random value in the range of [0, 255]. However, not all values yield noise. For instance, assume a 3x3 mask of a certain image with pixel intensity values in the range [120, 150]. Now, if a noise of intensity 130 is inserted into the centre pixel, then, it is hardly noticeable as it blends with its neighbourhood perfectly. Thus, only those noisy pixels that have intensity values either greater or lesser than the general intensity range are considered as noise. This is the fundamental principle on which the similarity module is built. Now, as the noisy pixels always have intensity values either greater or lesser than the general pixel range, they always end up at the extreme ends when all the pixels of a mask are arranged in either ascending or descending order. Using this logic we may arrive at a conclusion that when all the pixels of a mask, say a 3x3 mask, are arranged in either ascending or descending order any pixels in either of the extreme ends are always noisy. This, however, is a flawed logic for the fact that in the case of absence of noise within the mask the two pixels at the ends are

suspected to be noisy and are unnecessarily filtered, resulting in loss of information. To avoid this we employ an indirect approach of identifying the noisy pixels using thresholds. In this module, to identify the noisy pixels we first consider two sets of thresholds, out of which only those thresholds that are closer to the general pixel range are chosen. This improves the accuracy of the detection mechanism. To determine the thresholds we first sort the nine pixels within 3x3 mask in ascending order. When the nine pixels are arranged in ascending order the fifth value obtained is the median represented as $MedianInW_{i,j}$, the value preceding it is the fourth value represented as $4_{th}inW_{i,j}$ and the value succeeding it is the sixth value represented as $6_{th}inW_{i,j}$. We now use these values to obtain the thresholds as

$$Max_{i,j} = 6_{th}inW_{i,j} + Th\_SM_a,$$

$$Min_{i,j} = 4_{th}inW_{i,j} + Th\_SM_a.$$

These are the first set of thresholds obtained by adding and subtracting a fixed threshold from the 6th and 4th pixels respectively. Now, the following set of equations give us the final set of thresholds

$$N_{max} = \begin{cases} Max_{i,j}, & \text{if } (Max_{i,j} \leq MedianInW_{i,j} + Th\_SM_b) \\ MedianInW_{i,j} + Th\_SM_b, & \text{otherwise.} \end{cases}$$

$$N_{min} = \begin{cases} Min_{i,j}, & \text{if } (Min_{i,j} \geq MedianInW_{i,j} - Th\_SM_b) \\ MedianInW_{i,j} - Th\_SM_b, & \text{otherwise.} \end{cases}$$

Here, N_{max}, N_{min} are the final thresholds that are used to test the centre pixel, while, $MedianInW_{i,j}+Th\_SM_b$ and $MedianInW_{i,j}-Th\_SM_b$ are the second set of thresholds. Only those thresholds that are closer to the general pixel range are chosen here. In the case of maximum threshold, a threshold that has a lesser value among the two set of thresholds is chosen as it is closer to the general pixel range. Similarly, in minimum threshold, a threshold that has a greater value of the two set of thresholds is chosen as it is closer to the general pixel range.

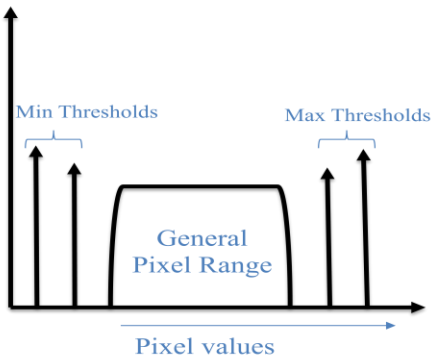


Fig.8 : Shows the explanation of choice of thresholds closer to the pixel range of the image.

Now, the final decision, 'whether the centre pixel is noisy or uncorrupted', is taken by comparing the centre pixel value with the two final thresholds N_{max} and N_{min}. If the centre pixel value lies in between the thresholds N_{max} and N_{min}, then, it is considered as uncorrupted, else, it is considered as corrupted and is filtered. This decision can be expressed as

$$\text{Decision4} = \begin{cases} \text{true}, & \text{if } (f_{i,j} \geq N_{max}) \text{ or } (f_{i,j} \leq N_{min}) \\ \text{false}, & \text{otherwise.} \end{cases}$$

4.4 Pre-Defined Thresholds

Several thresholds viz., Th_IM_a, Th_IM_b, Th_FM_a, Th_FM_b, Th_SM_a and Th_SM_b were used at various places in the modules. These thresholds each posses a predefined value and were found as a result of experimentation and research. The values of Th_IM_a, Th_IM_b, Th_FM_a, Th_FM_b, Th_SM_a and Th_SM_b are 20, 25, 40, 80, 15 and 60 respectively.

4.5 Edge-Preserving Filter

The basic logic behind the working of the edge preserving filter is that if there is an edge in any particular direction passing through the centre pixel in a mask, then, replacing the centre pixel with a mean value of only those pixels that are involved in the edge, will not break the edge but preserves it. To detect the presence of edge and to locate it we use eight directions within the 3x3 mask and their corresponding directional differences D1 to D8.

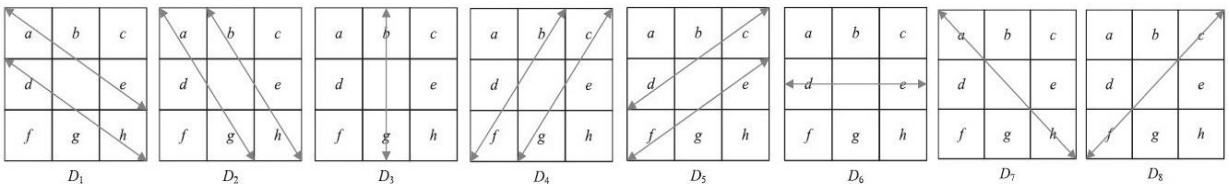


Fig.9 : Shows the eight directions along which directional differences are calculated.(refer[1])

The direction with the least directional difference is more likely to have an edge. Thus, we use the pixels that make up that direction and find the mean, which in turn is used to replace

the centre pixel value. However, this approach fails in the case where the pixels that make up a particular direction are actually corrupted. As it would result in a wrong calculation of directional difference, thereby, resulting in a wrong mean. Thus, to avoid this we omit all those directions from D1 to D8 that include a noisy pixel during the calculation of directional differences. To determine which pixels are noisy we make use of the $Max_{i,j}$ and $Min_{i,j}$ from the similarity module. In the case where all the neighbours of the centre pixel are suspected to be noisy, we find the weighted average of a, b &c. It is to be noted here that our mask is so designed that it moves from one column to the next along a row. Thus, the pixels a, b &c which belong to a previous row must have already been filtered and so are very much eligible to be used to replace the centre pixel. The estimated value of the centre pixel is given by

$$\hat{f}_{i,j} = (a + b \times 2 + c) / 4.$$

Generally, if none of the neighbours were found to be noisy we calculate the edge distances using the equations below

$$D_1 = |d - h| + |a - e|,$$

$$D_2 = |a - g| + |b - h|,$$

$$D_3 = |b - g| \times 2,$$

$$D_4 = |b - f| + |c - g|,$$

$$D_5 = |c - d| + |e - f|,$$

$$D_6 = |d - e| \times 2,$$

$$D_7 = |a - h| \times 2,$$

$$D_8 = |c - f| \times 2.$$

Based on which of the directional differences ends up being the $D_{min,}$ the estimated value of the centre pixel $\hat{f}_{i,j}$ is calculated as follows

$$\hat{f}_{i,j} = \begin{cases} (a + d + e + h) / 4, & \text{if } D_{min} = D_1, \\ (a + b + g + h) / 4, & \text{if } D_{min} = D_2, \\ (b + g) / 2, & \text{if } D_{min} = D_3, \\ (b + c + f + g) / 4, & \text{if } D_{min} = D_4, \\ (c + d + e + f) / 4, & \text{if } D_{min} = D_5, \\ (d + e) / 2, & \text{if } D_{min} = D_6, \\ (a + h) / 2, & \text{if } D_{min} = D_7, \\ (c + f) / 2, & \text{if } D_{min} = D_8. \end{cases}$$

It is to be noted that whenever the median of the pixels b, d, e and g is computed, it always results in a value equal to that of $\hat{f}_{i,j}$. This however happens only when the $\hat{f}_{i,j}$ is calculated correctly. In the case of a wrong detection of an edge or any other causes of error, the $\hat{f}_{i,j}$ is wrongly calculated. Thus, in such cases the obtained $\hat{f}_{i,j}$ is not equal to the median value of b, d, e and g, failing to satisfy the spatial relation it is meant to follow. To prevent this, we compute median of the pixels b, d, e &g along with $\hat{f}_{i,j}$ and use the resultant median to replace the centre pixel value. This not only prevents the error but also acts as a checking mechanism for the proper functioning of the edge preserving filter. The final median value obtained is now represented as $\bar{f}_{i,j}$. The final value which replaces the centre pixel is given by

$$\hat{f}_{i,j} = \text{Median}(\hat{f}_{i,j}, b, d, e, g).$$

It is to be noted that the entire process starting from mask formation to that of filtering using edge preserving filter is to be performed for every pixel present in the image (excluding boundaries), as the operations in the proposed method are only focused on the centre pixel within the mask. Thus, the entire process is iterated both row wise and column wise thereby covering all the pixels in the image.

4.6 Extension to color images

The proposed algorithm works extremely well when applied on grayscale images, however, it can also be applied on color images. For instance, to apply it on a 24 bit color image based on the RGB model affected by random valued impulse noise, we would first have to separate the three main components that are present in the image (i.e.)., Red, Green and Blue. This results in the generation of three variations of the original image. The three obtained images are gray scale images and these images actually represent the Red, Green and Blue components of the original image. Thus, by separating a color image into three gray scale images and applying the proposed method to each of these images separately, denoising a color image affected by random valued impulse noise is achieved. The final denoised image is obtained by combining the three denoised gray scale images, resulting in a denoised color image.

V. Implementation

The proposed algorithm has been implemented using MATLAB R2013a in a PC equipped with a 64-bit Intel i5 processor running at a clock speed of 2.5 GHz and 8 GB of RAM. Results were obtained using MATLAB and bar charts for the obtained results were plotted using Microsoft Office Excel 2007.

VI. Experimental Results

The following are the tabulated experimental results of the proposed method, compared with existing techniques when applied on various images such as Lena, Boat, Gold Hill and Peppers using Peak signal ratio (PSNR) expressed in dB (Decibels)as the comparison parameter.

Table.1 : Shows Comparison of Restored Lena image using PSNR in dB for various noise densities

Method	LENA			
	5%	10%	15%	20%
Noisy	23.47	17.12	15.56	14.16
Median	33.19	32.53	31.81	30.95
ACWM	39.96	37.29	35.15	32.81
MSM(3:T)	36.93	33.65	31.43	29.30
MSM(5:T)	36.82	33.42	30.97	28.66
MSM(7:T)	35.79	31.42	28.33	25.59
RVNP	36.29	32.72	32.25	31.14
AMF	37.50	33.96	33.47	31.70
NAVF	37.83	34.08	33.80	31.91
LCNR	38.67	36.76	35.38	34.07
ATMBM	38.72	36.82	34.56	32.23
DRID	39.39	36.84	34.91	32.86
RORD-WMF	37.52	36.23	35.19	34.87
DTBDM	41.15	38.22	36.17	34.43
Proposed	42.15	35.62	33.80	31.79

Table.2 : Shows Comparison of Restored Boat image using PSNR in dB for various noise densities

Method	BOAT			
	5%	10%	15%	20%
Noisy	23.33	19.37	17.35	15.07
Median	30.25	29.76	29.07	28.48
ACWM	35.58	34.06	32.37	30.50
MSM(3:T)	35.16	32.46	30.39	28.35
MSM(5:T)	35.31	32.38	30.03	27.82
MSM(7:T)	34.83	30.69	27.60	24.97
RVNP	32.17	31.26	29.45	28.92
AMF	33.19	32.55	30.66	29.30
NAVF	33.45	32.58	30.72	29.43
LCNR	34.10	33.02	31.95	30.95
ATMBM	34.61	33.26	31.63	30.13
DRID	35.98	34.20	32.41	30.87
RORD-WMF	32.34	31.70	31.16	30.57
DTBDM	36.83	34.48	32.89	31.38
Proposed	35.50	33.13	31.14	29.83

Table.3 : Shows Comparison of Restored Gold Hill image using PSNR in dB for various noise densities

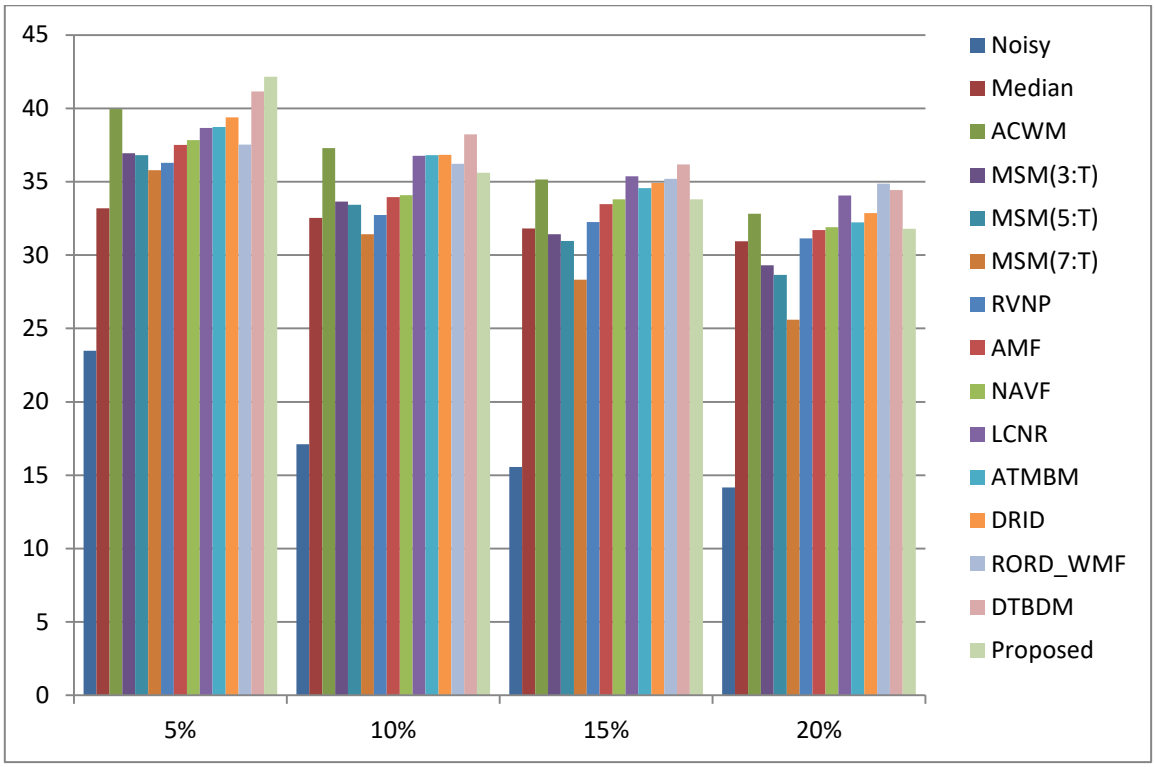
Method	GOLD HILL			
	5%	10%	15%	20%
Noisy	21.42	19.67	17.11	15.21
Median	30.74	30.30	29.85	29.30
ACWM	35.67	34.39	32.92	31.26
MSM(3:T)	35.28	32.54	30.46	28.56
MSM(5:T)	35.38	32.43	30.14	28.02
MSM(7:T)	34.92	30.70	27.79	27.76
RVNP	32.45	31.55	30.03	29.94
AMF	33.66	32.59	31.28	30.22
NAVF	33.72	32.72	31.41	30.74
LCNR	34.90	33.82	32.86	31.81
ATMBM	35.28	33.76	32.47	30.90
DRID	35.84	34.33	33.01	31.40
RORD-WMF	33.91	33.07	32.41	31.76
DTBDM	37.26	35.14	33.60	32.21
Proposed	36.22	34.60	32.29	31.79

Table.4 : Shows Comparison of Restored Peppers image using PSNR in dB for various noise densities

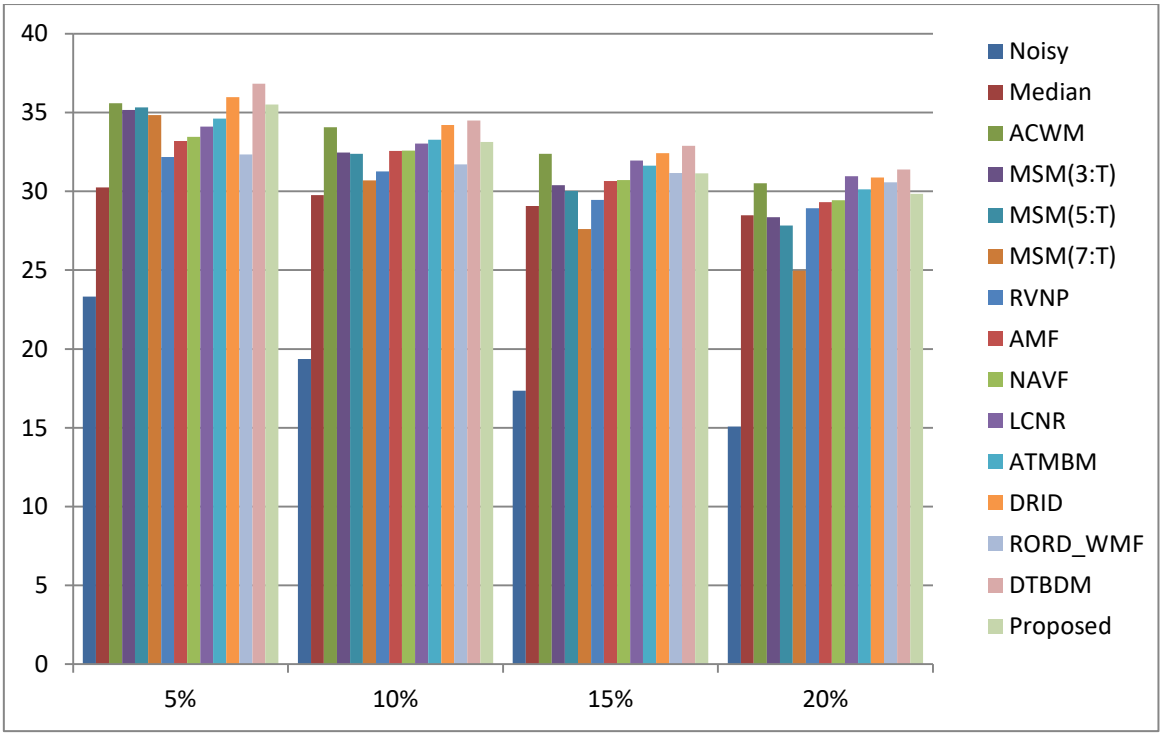
Method	PEPPERS			
	5%	10%	15%	20%
Noisy	22.33	19.45	17.74	15.40
Median	33.81	33.04	32.13	31.29
ACWM	39.34	36.93	34.57	32.30
MSM(3:T)	36.88	33.42	31.15	29.11
MSM(5:T)	36.78	33.22	30.60	28.41
MSM(7:T)	35.94	31.22	27.79	25.08
RVNP	36.23	33.82	32.72	31.42
AMF	37.30	34.33	33.19	31.49
NAVF	37.65	34.57	33.43	31.73
LCNR	39.08	36.94	35.25	32.96
ATMBM	39.43	36.81	33.90	31.62
DRID	38.92	36.55	34.13	31.80
RORD-WMF	36.69	35.70	34.74	33.91
DTBDM	39.84	37.18	35.47	33.70
Proposed	39.16	35.39	32.92	31.11

The following bar charts show the performance of the proposed algorithm when compared with existing techniques of high complexity as well as low complexity.

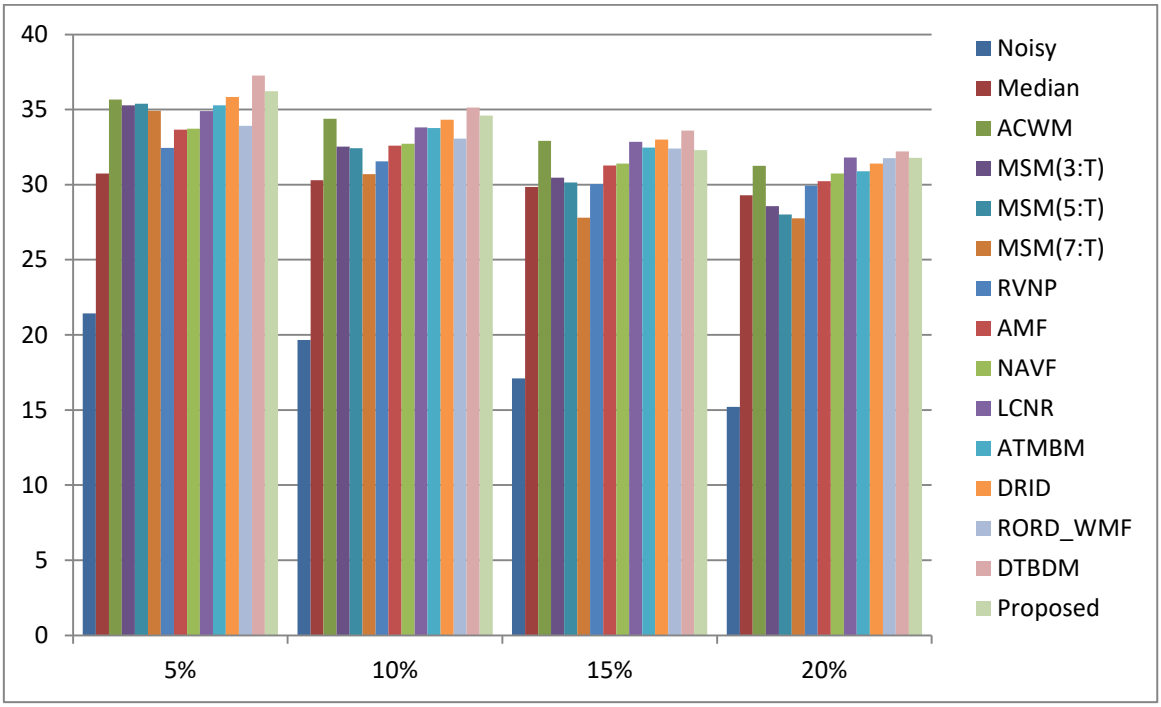
Comparison of restoration results (PSNR in dB vs. Noise density) for Lena image



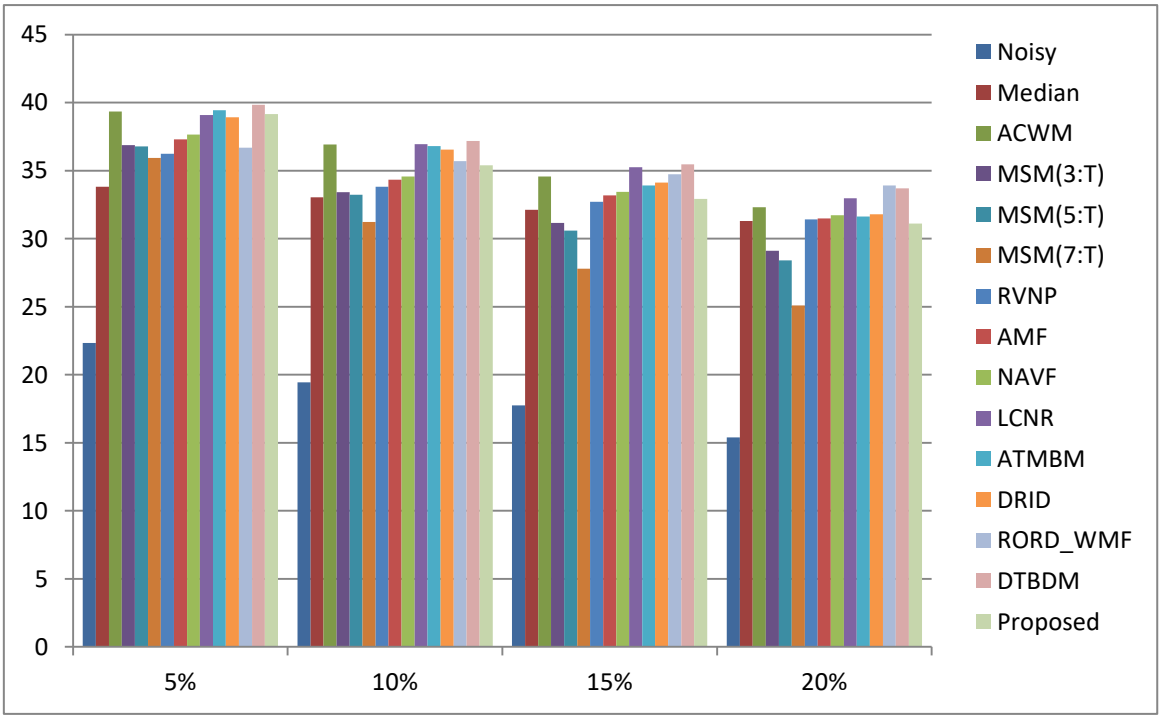
Comparison of restoration results (PSNR in dB vs. Noise density) for Boat image



Comparison of restoration results (PSNR in dB vs. Noise density) for Gold Hill image



Comparison of restoration results (PSNR in dB vs. Noise density) for Peppers image



Here, the existing low complexity techniques are median, ACWM, MSM(3:T), MSM (5:T), MSM (7:T), RVNP, AMF, NAVF, LCNR, DTBDM, Proposed. While, the high complex techniques are ATMBM, DRID, RORD-WMF [1].

VII. Conclusions and future work

In this paper, an efficient algorithm to denoise random valued impulse noise in images has been proposed. The proposed algorithm uses an impulse detector implemented based on a decision tree and uses an edge preserving filter to denoise the corrupted pixels. The algorithm uses simple mathematical expressions to detect and denoise impulse noise in images. Due to this simple approach the proposed algorithm falls under the category of low complexity techniques and yet provides a performance similar to that of the high complexity techniques. This is can be observed from the obtained results in section-VI. Apart from these, the advantage of using this algorithm also lies in the fact that it has a small code size as well as a fast response time, making it very suitable for real time application. The algorithm can be further extended to denoise high density impulse noise by cascading the proposed filter with any existing filters that focus on high density noise removal alone.

References

[1] Lien et al "An Efficient Denoising Architecture for Removal of Impulse Noise in Images", IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 4, APRIL 2013.

[2] Yiqiu Dong and Shufang Xu,"A New Directional Weighted Median Filter for Removal of Random – Valued Noise", IEEE SIGNAL PROCESSIGN Letters, VOL.14, NO.3, MARCH 2007.

[3] Pei-Yin Chen and Chih-Yuan Lien, "An efficient Edge-Preserving Algorithm for Removal of Salt and Pepper Noise", IEEE SIGNAL PROCESSIGNG Letters,VOL.15,2008.

[4] Yu et al "An efficient Procedure for Removing Random Valued Impulse Noise in Images", IEEE SIGNAL PROCESSIGNG Letters,VOL:15,2008.

[5] Wenbin Luo, "An Efficient Detail-Preserving Approach for Removing Impulse Noise in Images", IEEE SIGNAL PROCESSING LETTERS, VOL. 13, NO. 7, JULY 2006.

[6] Pei-Eng Ng and Kai-Kuang Ma, "A Switching Median Filter With Boundary Discriminative Noise Detection for Extremely Corrupted Images", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 15, NO. 6, JUNE 2006.

[7] K. Aiswarya et al, "A New and Efficient algorithm for the removal of High Density Salt and Pepper Noise in images and videos", Second International Conference on Computer Modeling and Simulation, 2010.

[8] Md. Imrul Jubair et al, "An Enhanced Decision Based Adaptive Median Filtering Technique to Remove Salt and Pepper Noise in Digital Images", Proceedings of 14th International Conference on Computer and Information Technology (ICClT 2011) 22-24 December, 2011.

[9] Dr. T. Santhanam and Ms. K. Chithra, "A New Decision Based Un-symmetric Trimmed Median Filter using Euclidean Distance Measure for Removal of High Density Salt and Pepper Noise from Images", International Conference on Information Communication & Embedded Systems (ICICES2014), 2014.

[10] Arabinda Dash and Sujaya Kumar Sathua, "High Density Noise Removal By Using Cascading Algorithms", Fifth International Conference on Advanced Computing & Communication Technologies, 2015.

[11] S.Vishaga and Sreejith .L .das, "A Survey on Switching Median Filters for Impulse Noise Removal", International Conference on Circuit, Power and Computing Technologies (ICCPCT), 2015.
[12] R.H. Chan, C.W. Ho, and M. Nikolova, “Salt-and-Pepper Noise Removal by Median-Type Noise Detectors and Detail-Preserving Regularization,” IEEE Trans. Image Processing, vol. 14, no. 10, pp. 1479-1485, Oct. 2005.
[13] T. Sun and Y. Neuvo, “Detail-Preserving Median Based Filters in Image Processing,” Pattern Recognition Letters, vol. 15, pp. 341-347, Apr. 1994.
[14] H. Hwang and R.A. Haddad, “Adaptive Median Filters: New Algorithms and Results,” IEEE Trans. Image Processing, vol. 4, no. 4, pp. 499-502, Apr. 1995.
[15] S. Zhang and M.A. Karim, “A New Impulse Detector for Switching Median Filter,” IEEE Signal Processing Letters, vol. 9, no. 11, pp. 360-363, Nov. 2002.