Circuits, Systems, and Signal Processing

Hardware-efficient VLSI architecture for feedback decision-based trimmed mean filter to eliminate high-density salt and pepper noise

--Manuscript Draft--

Manuscript Number:	CSSP-D-23-00823
Full Title:	Hardware-efficient VLSI architecture for feedback decision-based trimmed mean filter to eliminate high-density salt and pepper noise
Article Type:	Original Research
Keywords:	Salt and pepper noise; mean filter; peak signal to noise ratio; structural similarity; decision based feedback trimmed mean filter
Abstract:	This paper proposes a hardware-efficient VLSI architecture for the feedback decision-based trimmed mean filter that eliminates high-density salt and pepper noise in the images. The pixel is modified only if it a noisy pixel. In such cases, the neighbouring pixels in a 3×3 window corresponding to the center pixel are considered for computing the output pixel. Either the mean of the horizontal and vertical noisy pixels or the noise-free pixels in the window is computed. This mean value is fed back and the center noisy pixel is updated immediately. The updated pixel value is used henceforth for correcting the remaining corrupted pixels. This procedure helps in removing the noisy pixels effectively even if the noise density is high. The VLSI architecture designed is efficient since algorithm does not require sorting process and the computing resources required are less when compared to other state-of-the-art algorithms.

Hardware-efficient VLSI architecture for feedback decision-based trimmed mean filter to eliminate high-density salt and pepper noise

Midde Venkata Siva[1*] and Jayakumar E. P.[2]

[1, 2] Electronics and Communication Engineering, National Institute of Technology Calicut, Kozhikode, 673601, Kerala, India.

*Corresponding author(s). E-mail(s): siva375937@gmail.com;
Contributing authors: jay@nitc.ac.in;

Abstract

This paper proposes a hardware-efficient VLSI architecture for the feedback decision-based trimmed mean filter that eliminates high-density salt and pepper noise in the images. The pixel is modified only if it a noisy pixel. In such cases, the neighbouring pixels in a 3×3 window corresponding to the center pixel are considered for computing the output pixel. Either the mean of the horizontal and vertical noisy pixels or the noise-free pixels in the window is computed. This mean value is fed back and the center noisy pixel is updated immediately. The updated pixel value is used henceforth for correcting the remaining corrupted pixels. This procedure helps in removing the noisy pixels effectively even if the noise density is high. The VLSI architecture designed is efficient since algorithm does not require sorting process and the computing resources required are less when compared to other state-of-the-art algorithms.

Keywords: Salt and pepper noise, mean filter, peak signal to noise ratio, structural similarity, decision based feedback trimmed mean filter

1 Introduction

Impulse noise, which results from defects in the image capture device or noise in the transmission channel, significantly reduces the image quality. To eliminate the impulse noise, nonlinear filters are employed. It is well known that the Median Filter (MF) may

remove impulsive noise while maintaining the edges. Before determining the middle-ranked value while implementing a median filter, a sorting procedure must be carried out. To sort the data provided, two numbers must be compared and swapped. The kind of sorting algorithm employed determines the time complexity of the median filter, which may be done in software. To improve computing speed and meet user demands, hardware implementation of median filters on Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASIC) is necessary and unavoidable.

By including a design error in the implementation of the 2-bit comparator logic, Monajati and Kabir [2] proposed an Inexact Median Filter (IMF) architecture. The 8-bit comparator is produced by the 2-bit comparator and used to compare and swap the two 8-bit integers [2]. In this case, the median value is not exactly the median value. A design for a high-efficiency median filter was proposed by Lin et al. [3]. In this design, the median value is obtained by cascading a number of partial median units. The cascading of numerous comparison-free median units results in the presentation of a Comparison Free Median Filter (CFMF) [4]. Without any comparisons, this design generates the median value. The above standard median filters [2–4] architectures produces high denoised image quality at lower noise densities and low denoised image quality when the Noise Density (ND) is high [5].

To remove noisy pixel in a image with high noise density, the adaptive median filter [6] extends the window size and computes the median value. The denoised images are still affected with blurring by using this technique. A switching-based median filter was presented by Pei-Eng Ng and Kai-Kuang Ma [7, 8], in which the noisy pixel is correctly identified using a predetermined threshold. A quick and effective decision-based approach to remove high-density impulse noise was suggested by Srinivasan and Ebenezer [9]. In this approach, if the median value is noisy, the noisy pixel is replaced by the neighbourhood value of the median value; otherwise, the median value does so.

The iterative adaptive alpha trimmed mean filter introduced by Ahmed and Das [10] replaces the noisy pixel with a weighted mean operation after using fuzzy detection to identify the noisy pixel. High-quality denoised images are produced by this method. Esakkirajan et al. [11] suggested a technique in which the output pixel that corrects the noisy pixel is computed using either the median of all noise-free pixels in the selected window size or the average of all accessible noisy pixels. With more noise present, this algorithm exhibits more output value failures. The Direction Weighted Median (DWM) filter, developed by Dong and Xu [12], computes the weighted average of the absolute differences in four directions. A weighted median directed towards closely adjacent pixels also restores the noisy pixel. The DWM filter employs a detection process continually with a dropping threshold in order to enhance detection accuracy, however this leads to a high level of computer complexity. A modified DWM filter described by Lu and Chou [13] analyses a 7×7 window and classifies pixels as noisy, noise-free, or edge pixels in all possible orientations. Additionally, [14] shows an expansion of the DWM filter where different window sizes are chosen based on ND.

Habib *et al.*'s [15] adaptive fuzzy inference system-based DMF accurately classifies pixels as being noisy or noise-free. The Fuzzy Directional Median (FDM) filter boosts the detection accuracy by differentially recognising noisy pixels in these regions by

differentiating between smooth and non-smooth areas [16]. The median of the 3×3 window is also used to remove noisy pixels from areas that are smooth and those that are not. FDM filters may, however, eliminate up to 30% ND of impulsive noise. Using either a patch median or a trimmed median, Balasubramanian *et al.* introduced the probabilistic decision-based filter that substitutes the noisy pixel depending on noise density [17]. A threshold-based linear prediction-based adaptive MF was introduced by Roy and Laskar [18] and uses an adaptive window's median to evaluate if a pixel is noisy before denoising it. In [19], a unique applied MF is provided that takes nearby pixels' values into account and uses a movable window to remove salt and pepper noise. A pixel-based density filter [20] determines if the current pixel is noise-free or noisy before selecting on an adaptive window size where the most repeated NFP returns the noise-free pixel.

Four stage Median Average (FoMA) filtering technique by Garg and Arya was introduced in [21]. The FoMA algorithm has four stages: first, second, third, and fourth. The first stage uses median filtering with several window sizes—smaller, bigger, running average, and replacement by previously processed pixels, respectively. The algorithm moves on to the following steps if the noisy pixel in the candidate cannot be denoised at the current level. An image with low noise levels can be efficiently denoised using the FoMA algorithm's first two phases. To reduce higher noise levels in the image, all phases are necessary.

If the processing pixel is noisy, the Decision-Based Neighbourhood Referred Asymmetrically Trimmed Modified Trimean (DBNRATMTF) [22] method replaces it with an mean of 4 pixels (horizontal and vertical directional pixels of 3×3 window) if these 4 pixels are likewise noisy. If not, a modified trimean with asymmetrical trimming takes the place of the treated pixel. In order to denoise the picture that has excessive noise density, Garg [23] introduced the Adaptive Trimmed Median (ATM) filter. This filter applies an interpolation-based technique for low and high noise densities, respectively, and corrects noisy pixels by computing the median of Noise Free Pixels (NFPs) inside of a window of customizable size. An asymmetric trimmed/winsorized mean filter with two stages for noise reduction was introduced by Goel [24]. In the first step, an unsymmetrical trimmed mean filter is applied when the processing window is entirely noisy and contains both outliers (0 and 255). By changing the processing pixel with an asymmetrically trimmed mean in one version, the second phase removes any unremoved noisy pixels. The procedure is additionally modified by selecting the unsymmetrical winsorized mean rather than the unsymmetrical trimmed mean in the second step. Modif_A1 and Modif_A2, respectively, are the designations for the algorithm with an asymmetrical mean and an asymmetrical winsorized mean. A three-stage approach, consisting of pre-processing, main processing, and post-processing, was presented by Bindal and Garg [25] to remove salt and pepper noise. The primary step of processing deals with high-density noises and noisy pixels near the edge of the picture, while the pre-processing and post-processing stages eliminate low-density noise. The presented approach also uses non-corner pixels to calculate the value of denoised corner pixels. The works presented in [6–25] discusses only the algorithms to remove high density of salt and pepper noise, whereas the hardware architectures for the efficient implementation of such algorithms are not discussed.

In this paper, an algorithm is proposed that is designed such that it is computationally less complex and hence, hardware friendly. Additionally, an efficient VLSI architecture for realizing the algorithm in the hardware is also proposed. The key contributions of this work are as follows

- In the literature, only algorithms suitable for software implementation are available in plenty to denoise images with high density salt and pepper noise. Here, we propose hardware friendly algorithm and its efficient architecture which works better to eliminate high density salt and pepper noise without adversely affecting the image quality.
- In the existing algorithms, after sorting the input pixels, the noisy pixels are discarded and the mean of noise-free pixels is computed. The proposed architecture is designed in an optimized manner such that there is no requirement of such a sorting process.
- In general, while calculating the mean of noise-free pixels, the divisor which is the number of noise-free pixels may vary across windows. Complex hardware is required for this divide operation. This hardware complexity is reduced by using approximate division in the proposed architecture that uses simple right-shift operations and additions.

The remaining sections of the paper are as follows: Section 2 presents the proposed method with VLSI architecture, results are discussed in the Section 3. Finally Section 4 concludes the paper.

2 Proposed Method

In this section, the pseudocode, internal VLSI architecture, and implementation of the proposed method are discussed.

2.1 Pseudocode

The proposed method's pseudocode is described in Algorithm 1. The algorithm receives an image with salt and pepper noise as input and returns a denoised image as output. In the noisy image, the noisy pixel is first found, and then the nearby pixels around the noisy pixel are examined for computing the output pixel. If P5 is the first noisy pixel in Fig. 1, then P1, P2, P3, P4, P6, P7, P8, and P9 create a 3×3 window of surrounding pixels, with P4 and P6 representing horizontal pixels, P2 and P8 representing vertical pixels, and the remaining P1, P3, P7, and P9 representing diagonal pixels. If all of the horizontal and vertical pixels are noisy, the noisy pixel (P5) will be replaced with the mean of these four pixels. The average of the noise-free pixels from the window is used to replace P5 if any one of these four pixels is not noisy. This output pixel substitutes the noisy pixel in a noisy image and is given as feedback. This output pixel is sent as feedback and replaces the noisy pixel in a noisy image. If P8 is next noisy pixel, then the neighbouring pixels in 3×3 window represented as W which is shown in the Fig. 1 contains both processed pixels (P4, P5, P6, and P7) and unprocessed pixels (P9, P10, P11, and P12).

Algorithm 1 Algorithm

Input *NI*: Noisy Image
Output *DI*: Denoised Image
for each pixel $NI_{(s,p)}$ of *NI* **do**
  if ($NI_{(s,p)}$=0 | $NI_{(s,p)}$=255) **then**
    Get 3×3 window size of pixels as P1=$NI_{(s-1,p-1)}$, P2=$NI_{(s,p-1)}$, P3=$NI_{(s+1,p-1)}$, P4=$NI_{(s-1,p)}$, P5=$NI_{(s,p)}$, P6=$NI_{(s+1,p)}$, P7=$NI_{(s-1,p+1)}$, P8=$NI_{(s,p+1)}$, and P9=$NI_{(s+1,p+1)}$.

    if ((P2=0|P2=255)&(P4=0|P4=255)&(P6=0|P6=255)&(P8=0|P8=255)) **then**
      Final = mean (P2, P4, P6, P8);
    else
      Final = mean (noise-free pixels)
    end if
  else
    Final = P5;
  end if
  $NI_{(s,p)}$ = Final;
  $DI_{(s,p)}$ = Final;
end for
return *DI*;

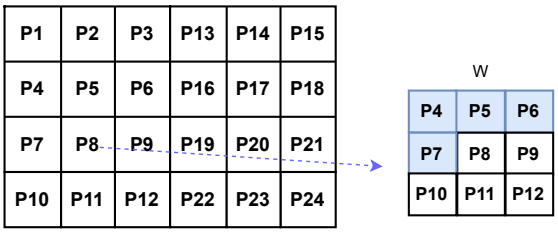

Fig. 1: Representation of centre noisy pixel surrounded by processed and unprocessed pixels

The working of the proposed algorithm is shown in Fig. 2 with an example of 4×6 noisy input image, with zero padding across the border. Here the first pixel is noisy and hence the neighbouring pixels of 3x3 window i.e., *W1* (0, 0, 0, 0, 255, 255, 0, 255, 35) are considered for correcting it. Since the horizontal and vertical pixels in 3×3 window are noisy, the mean of these pixels is computed and it is sent as feedback to update in the position of the noisy center pixel. After detecting the second noisy pixel, then the window *W2* is considered, and *W2* contains (0, 0, 0, 128, 255, 0, 255, 35, 27) pixels. Here, 128 is the previously processed pixel, i.e., the output pixel from the *W1* window. All the horizontal and vertical pixels are not noisy in the current window, so the mean is computed only for noise-free pixels (i.e (128+35+27)/3 = 63). Again this processed output pixel is sent as feedback and replaced in the second noisy pixel location. The same procedure is carried on the rest of the noisy pixels, and the denoised output window is obtained, as shown in Fig. 2.

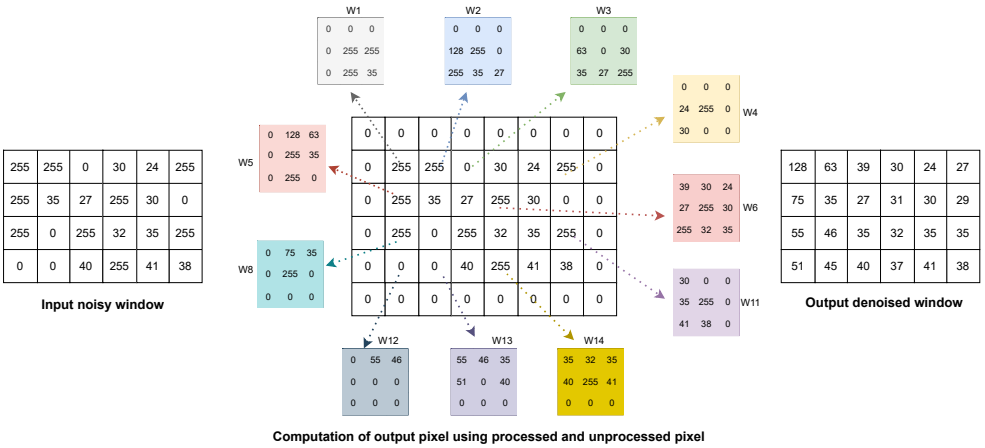

Fig. 2: Working of proposed feedback decision based mean filter

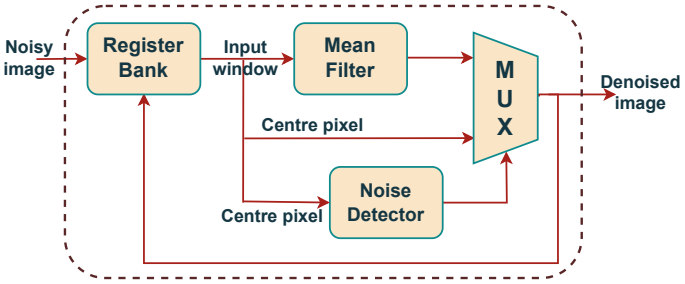

Fig. 3: Block diagram of the proposed method

2.2 VLSI Architecture

The proposed architecture's block diagram is seen in Fig. 3, and it includes a register bank, mean filter, noise detector, and multiplexer (MUX). The mean filter and noise detector both get the necessary pixels from the register bank. The mean filter determines the mean of noise-free pixels or the mean of the horizontal and vertical pixels in a chosen window. The noise detector detects whether or not the centre pixel is noisy. The output of the mean filter or the centre pixel is chosen using the MUX. If the centre pixel is noisy, MUX chooses the mean filter's output value; otherwise, MUX chooses the centre pixel itself. The register bank receives the output from the MUX as feedback and updates it.

2.2.1 Register Bank

The mean filter requires nine pixels as input. A noise detector requires a centre pixel from every window Wi to detect whether it is noisy or not. The register bank is used to provide nine pixels to the mean filter and one pixel i.e. centre pixel, to the noise

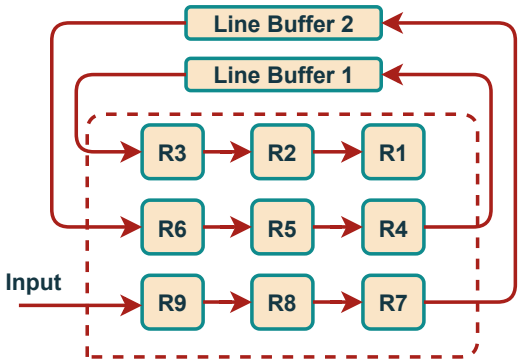

Fig. 4: Architecture of register bank

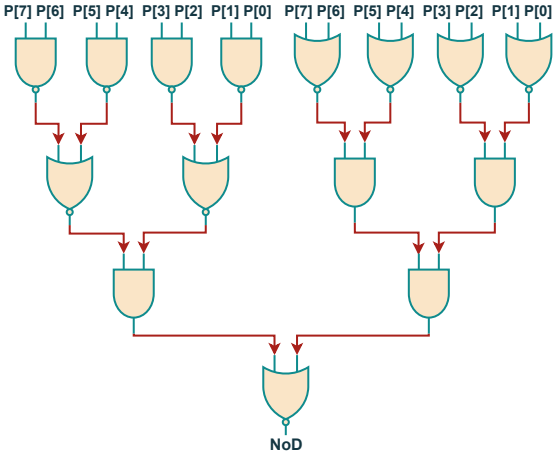

Fig. 5: Internal architecture of salt and pepper noise detector [26]

detector. The architecture of the register bank is shown in Fig. 4. Two line buffers and nine shift registers are used to provide nine pixels to the mean filter. Line buffer 1 stores the processed pixels, and line buffer 2 stores the unprocessed pixels. When a new pixel enters register R9, registers R9, R8, R6, R5, R3, and R2 are, accordingly, right shifted to registers R8, R7, R5, R4, R2, and R1. Line buffer 1 receives the pixel from the R4 register while line buffer 2 receives the pixel from the R7 register. The R3 and R6 registers, respectively, read the leftmost pixels from line buffers 1 and 2.

2.2.2 Salt and Pepper Noise Detector

The gate-level architecture of the Salt and Pepper Noise Detector (SAPND) is shown in Fig. 5. The centre pixel from the designated input window serves as the input for the SAPND. The output of SAPND is represented by NoD. When the centre pixel

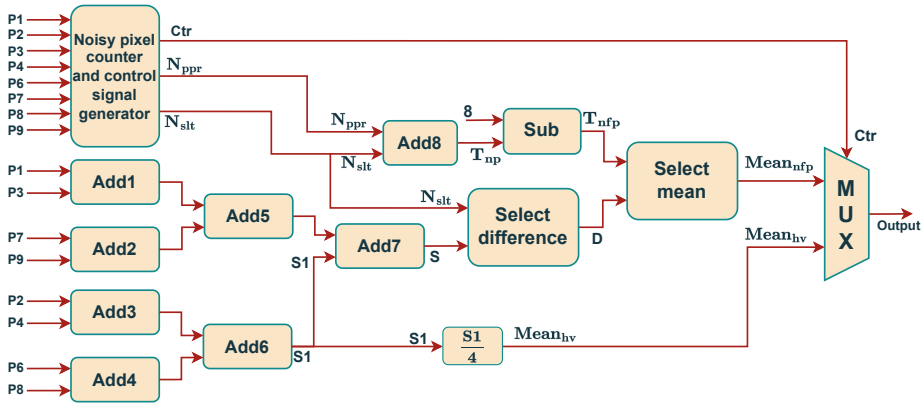

Fig. 6: VLSI architecture for proposed decision based trimmed mean filter

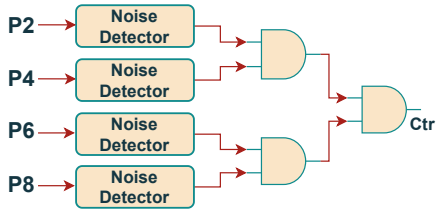

Fig. 7: Internal architecture of control signal generator

is either 0 or 255, NoD becomes 1 and remains at 0 for all other values. The MUX utilises this NoD as as the select line.

2.2.3 Mean filter architecture

If every horizontal and vertical pixel surrounding the centre noisy pixel is noisy, the proposed feedback decision-based trimmed mean technique computes the mean of these pixels. The mean of the noise-free pixels throughout the whole window *Wi* is computed if none of the horizontal or vertical pixels are noisy. The number of noise-free pixels will vary from one window to another. For the proposed algorithm, a hardware-friendly VLSI architecture is proposed to find the mean for variable number of noise-free pixels, it is seen in Fig. 6. It contains a noisy pixel counter, control signal generator, adders, subtractors, select difference, select mean, and multiplexer (MUX) blocks.

In the proposed mean filter architecture, the mean is computed by excluding the center pixel. The working of each block in the proposed VLSI architecture is discussed by considering the sixth noisy pixel from the Fig. 2. Here, P1= 39, P2= 30, P3= 24, P4= 27, P5= 255, P6= 30, P7= 255, P8= 32, and P9= 35 are the pixels inside the window *W6*, and P5 is the noisy pixel. The noisy pixel counter and control signal generator block produce three outputs, and they are named *ctr*, N_{ppr}, and N_{slt}. The

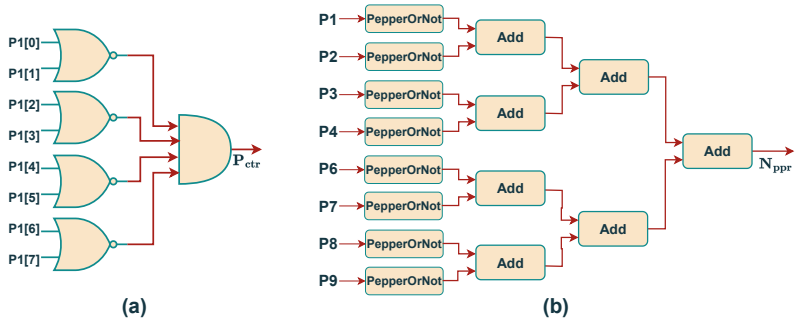

Fig. 8: Internal architecture for (a) Pepper noisy pixel or not for pixel P1 (b) Number of Pepper noisy pixels

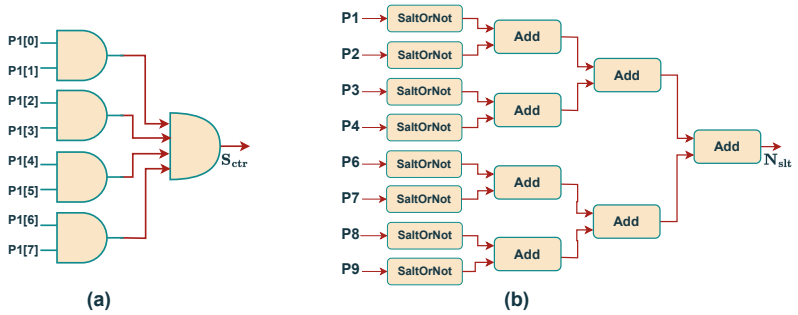

Fig. 9: Internal architecture for (a) Salt noisy pixel or not for pixel P1 (b) Number of Salt noisy pixels

ctr is generated by verifying whether P2, P4, P6, and P8 are noisy. If all the pixels are noisy, then the *ctr* is 1 otherwise, it is 0. In the considered case, P2= 30, P4= 27, P6= 30, and P8= 32 are not noisy, so *ctr* is 0. The internal architecture that generates *ctr* is shown in Fig. 7 and it requires four noise detectors, and three *AND* gates.

The internal architecture that counts the amount of pepper noise pixels is seen in Fig. 8, and the number of pepper noise pixels present in *W6* is represented by N_{ppr}. As shown in Fig. 8 (a), the control signal P_{ctr} is initially created by determining whether or not the pixel is pepper noisy. All of these P_{ctr} signals are then summed to determine the amount of pepper noise pixels, as shown in Fig. 8 (b). Here, N_{ppr} is equal to 0. The number of salt noise pixels present in the *W6* is denoted by N_{slt}, and the internal architecture that counts the number of salt noise pixels is shown in Fig. 9. Initially, the control signal S_{ctr} is generated by checking if the pixel is salt noisy or not, as shown in Fig. 9 (a), and all these P_{ctr} signals are added to get the number of salt noise pixels as shown in Fig. 9 (b). Here, N_{slt} value is 1. The maximum bit length of N_{ppr} and N_{slt} is four as the total number of pixels is eight, which requires 4 bits in binary representation. The total number of noisy pixels, abbreviated as T_{np}, is obtained by adding the N_{ppr} and N_{slt} values. To find the total number of noise-free

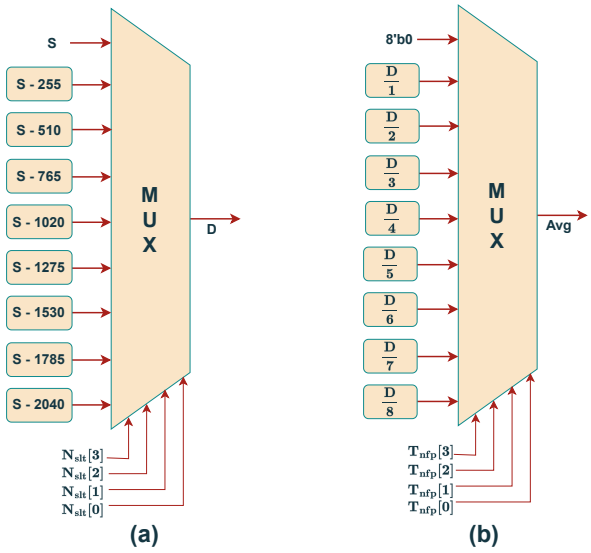

Fig. 10: Internal architecture for (a) select difference term (b) select mean term

pixels in W6, subtract this T_{np} with 8, which is denoted as T_{nfp}. Here, the T_{np} and T_{nfp} are 1 and 7, respectively. The maximum bit length of both T_{np} and T_{nfp} is 4.

If the centre pixel is noisy, the proposed algorithm selects the output value between the mean of noise-free pixels and the mean of horizontal and vertical pixels. The sum of horizontal and vertical pixels will be obtained at the end of *Add6* block, which is denoted as *S1* (i.e. 119), and its mean obtained by using right shift operation and performing the rounding off on the right shifted bit values as shown in Fig. 11. The mean of horizontal and vertical pixels is obtained after the $\frac{S1}{4}$ block that is denoted as $Mean_{hv}$. The maximum bit length of *S1* and $Mean_{hv}$ are 10 and 8, respectively.

The sum of noise-free pixels needs to be computed initially to get the mean of noise-free pixels. At the end of *Add7* block, the sum of all eight pixels is obtained, which is denoted as S (i.e 472); this includes the noisy salt/pepper pixels also. To get the sum of noise-free pixels, the value S must be subtracted from the sum of all salt noisy pixels. The select difference block generates the sum of noise-free pixels with N_{slt} and S as inputs. The internal architecture of the select difference block is shown in Fig. 10 (a) which consists of subtraction blocks and a 9×1 multiplexer. As the number of salt noisy pixels will vary for different 3×3 windows, the S will be subtracted from all possible summation of salt noisy pixels and one of the precalculated difference is selected as output using MUX with N_{slt} control signal. These subtraction units are implemented using adders. In the example taken, as N_{slt} is 1, output from the S - 255 block is selected as the output, which is represented as D (i.e 217). The maximum bit length of S and D is 11.

For computing the mean of noise-free pixels, the sum of noise-free pixels value needs to be divided by the total number of noise-free pixels. The select mean block generates a mean of noise-free pixels with T_{nfp} and D as inputs, and its internal architecture is

Fig. 11: Calculation of dividing a number by 4 using right shift operation with rounding off

Table 1: Approximate division with right shifting a number

Exact division	Approximate division
$1/3 = 0.333$	$1/4 + 1/16 + 1/64 = 0.328125$
$1/5 = 0.2$	$1/4$ -$1/16 + 1/128 = 0.19531$
$1/6 = 0.1666$	$1/8+1/32+1/128 = 0.1640625$
$1/7 = 0.1428$	$1/8 + 1/64 = 0.140625$

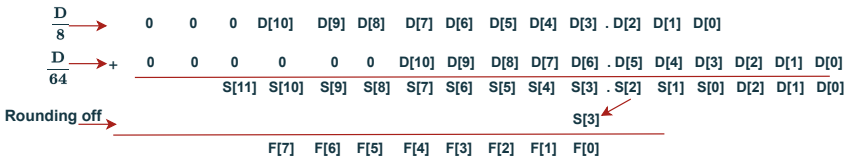

Fig. 12: Internal calculation of dividing a number by 7 with rounding off

shown in Fig. 10 (b) that consist of division blocks and 9×1 multiplexer. As the total number of noise-free pixels will vary for different 3×3 windows, the D will be divided with all possible T_{nfp} and one of the precalculated division term is selected as output using MUX with T_{nfp} control signal. The direct implementation of any division block consumes more area except if the divisor terms are 2, 4, or 8. The division of any number with 2, 4, and 8 can be implemented by right-shifting the input data and performing round off. Fig. 11 shows the division of a number by 4. The hardware cost will increase where the divisor is 3, 5, 6 or 7. So, an approximate division is performed using the right shift operation on input data and additions as shown in Table 1. This approximate division is implemented with basic adders, which require less hardware. Here, the total number of noise-free pixels (T_{nfp}) is 7, so the output from the $\frac{D}{7}$ block will be selected as output. The approximate calculation of $\frac{D}{7}$ block with the right shifting of D, performing addition and rounding off, is shown in Fig.12. The output from the select mean block is 31, represented as $Mean_{nfp}$.

The final output of the proposed mean filter is obtained using the ctr signal, $Mean_{nfp}$, and $Mean_{hv}$ values. Here, the ctr is 0, and the output is selected as $Mean_{nfp}$ (i.e 31). The proposed architecture is implemented with basic gates, adders, subtractors, and multiplexers that consume very less hardware.

Fig. 13: List of images in set12 dataset

3 Results and Comparisons

3.1 Image quality assessment

To compare the quality of the denoised images, metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are utilised. To examine the denoised visual quality, 68, 12, and 24 images from the BSD68 [27], Set12 [28], and 24 datasets are used. Fig. 13 shows the list of images in the Set12 [28] dataset. Figure 14 compares the input noisy images and output denoised images for the existing and suggested method. It reveals that, the image quality using the proposed method is comparable to that of the existing methods.

The PSNR and SSIM metric is computed for each image in the three datasets mentioned. The average PSNR and SSIM are obtained for each dataset and they are tabulated in Table 2 and Table 3, respectively. The proposed method produces similar image quality compared to DBNRATMTF [22] and Modif_A1[24] at all noise densities. The proposed method produces similar image quality at 10% and 20% noise densities and produces 0.42dB, 0.82dB, 1.19dB, 1.59dB, 2dB, 2.47dB, and 3.12dB improvement in denoised image quality at 30%, 40%, 50%, 60%, 70%, 80%, and 90% noise densities, respectively compared to ATM [23]. Our method produces the same image quality till 70% noise densities, and it loses 0.35dB and 0.5dB denoised image quality at 80% and 90% noise densities, respectively, compared to Modif_A2[24]. Our method is capable of improving the denoised image quality by 0.83dB, 0.70dB, 0.57dB, 0.40dB, and 0.15dB at 10%, 20%, 30%, 40%, and 50%, respectively, compared to ThreeStage [25] and it

Fig. 14: Input and output images of existing and proposed algorithms from 40% to 90% ND(left to right) (a) noisy images, (b) DBNRATMTF [22], (c) ATM[23], (d) Modif_A1[24], (e) Modif_A2[24], (f) ThreeSta.[25], (g) Proposed

Table 2: Average PSNR

Noise Density	Dataset	DBNRA.[22]	ATM[23]	Modif_A1[24]	Modif_A2[24]	ThreeSta.[25]	Proposed
10ND	Vasieck	38.4037	38.4909	38.3683	38.2587	37.9990	38.3749
	Set12	36.5095	37.0825	36.4898	36.3866	36.6334	36.4902
	BSD68	36.3919	36.2957	36.3556	36.2364	34.1693	36.4125
	Average	37.1017	37.2897	37.0712	36.9606	36.2672	37.0926
20ND	Vasieck	35.1065	34.9577	35.1077	34.9462	34.8496	35.1123
	Set12	33.4911	33.6563	33.4920	33.3299	33.6132	33.4924
	BSD68	33.2194	32.9200	33.1984	33.0331	31.2925	33.2546
	Average	33.9390	33.8446	33.9327	33.7698	33.2518	33.9531
30ND	Vasieck	32.9741	32.5163	33.0162	32.8540	32.8681	33.0129
	Set12	31.4438	31.2565	31.4843	31.3115	31.6567	31.4854
	BSD68	31.2032	30.7168	31.2292	31.0704	29.5215	31.2575
	Average	31.8737	31.4965	31.9099	31.7453	31.3488	31.9186
40ND	Vasieck	31.2170	30.3465	31.2846	31.1878	31.3349	31.2850
	Set12	29.8696	29.2875	29.9307	29.8081	30.2409	29.9287
	BSD68	29.6803	28.8563	29.6886	29.5659	28.1677	29.7424
	Average	30.2556	29.4968	30.3013	30.1873	29.9145	30.3187
50ND	Vasieck	29.6755	28.3854	29.7365	29.7198	30.0678	29.7382
	Set12	28.4629	27.5167	28.5270	28.4664	29.0301	28.5273
	BSD68	28.2771	27.1324	28.2574	28.1946	27.0503	28.3321
	Average	28.8052	27.6782	28.8403	28.7936	28.7161	28.8659
60ND	Vasieck	28.1004	26.4030	28.1436	28.2425	28.9734	28.1427
	Set12	26.9348	25.4827	26.9729	27.0412	27.9304	26.9725
	BSD68	26.8292	25.3373	26.8382	26.8774	26.1174	26.8649
	Average	27.2881	25.7410	27.3182	27.3871	27.6737	27.3267
70ND	Vasieck	26.4626	24.2823	26.4712	26.7126	27.8416	26.4704
	Set12	25.3840	23.4816	25.3907	25.6052	26.8111	25.3931
	BSD68	25.3793	23.4988	25.3615	25.5123	25.1022	25.3854
	Average	25.7420	23.7542	25.7412	25.9434	26.5849	25.7496
80ND	Vasieck	24.4711	21.7386	24.4424	24.8456	26.3839	24.4427
	Set12	23.3875	20.9473	23.3560	23.7497	25.3457	23.3570
	BSD68	23.6123	21.3031	23.5679	23.8513	23.8337	23.5858
	Average	23.8236	21.3297	23.7888	24.1489	25.1878	23.7952
90ND	Vasieck	21.7129	18.2687	21.6410	22.1979	22.3748	21.6470
	Set12	20.7469	17.6174	20.6737	21.2098	21.6585	20.6801
	BSD68	21.2440	18.2640	21.1685	21.5978	20.6985	21.1877
	Average	21.2346	18.0500	21.1611	21.6685	21.5773	21.1716

loses 0.35dB, 0.84dB, 1.39dB, and 0.41dB image quality at 60%, 70%, 80%, and 90% respectively compared to ThreeStage [25]. In Fig. 15, the average PSNR and SSIM of three datasets are represented graphically for each noise density. It is clear that the proposed method consistently generates good image quality at higher noise densities.

3.2 Hardware analysis

For the existing methods, such as DBNRATMTF [22], ATM[23], Modif_A1[24], Modif_A2[24], and ThreeSta.[25] only software implementation is presented. If these algorithms are to be mapped into architectures, the number of computing resources that might be required to correct one corrupted pixel is listed in Table 4. The major computing resources are line buffers, counter required to count the number of noise pixel in the choosen window, Compare and Swap (CS) block that sort the two numbers

Table 3: Average SSIM

Noise Density	Dataset	DBNRA.[22]	ATM[23]	Modif_A1[24]	Modif_A2[24]	ThreeSta.[25]	Proposed
10ND	Vasieck	0.9873	0.9873	0.9873	0.9871	0.9872	0.9873
	Set12	0.9843	0.9849	0.9844	0.9842	0.9846	0.9844
	BSD68	0.9828	0.9823	0.9828	0.9824	0.9710	0.9830
	Average	0.9848	0.9848	0.9848	0.9846	0.9809	0.9849
20ND	Vasieck	0.9728	0.9722	0.9730	0.9723	0.9730	0.9730
	Set12	0.9698	0.9699	0.9700	0.9692	0.9704	0.9700
	BSD68	0.9642	0.9627	0.9644	0.9635	0.9526	0.9646
	Average	0.9690	0.9683	0.9691	0.9683	0.9654	0.9692
30ND	Vasieck	0.9557	0.9527	0.9561	0.9550	0.9566	0.9561
	Set12	0.9523	0.9506	0.9528	0.9514	0.9538	0.9528
	BSD68	0.9425	0.9388	0.9431	0.9418	0.9318	0.9431
	Average	0.9502	0.9474	0.9506	0.9494	0.9474	0.9507
40ND	Vasieck	0.9347	0.9267	0.9355	0.9345	0.9377	0.9355
	Set12	0.9320	0.9262	0.9328	0.9312	0.9354	0.9328
	BSD68	0.9173	0.9085	0.9180	0.9167	0.9077	0.9182
	Average	0.9280	0.9205	0.9288	0.9275	0.9269	0.9288
50ND	Vasieck	0.9086	0.8917	0.9095	0.9092	0.9158	0.9095
	Set12	0.9066	0.8933	0.9076	0.9063	0.9138	0.9076
	BSD68	0.8860	0.8685	0.8866	0.8859	0.8798	0.8870
	Average	0.9004	0.8845	0.9012	0.9005	0.9032	0.9014
60ND	Vasieck	0.8743	0.8425	0.8752	0.8769	0.8909	0.8752
	Set12	0.8722	0.8444	0.8732	0.8733	0.8889	0.8732
	BSD68	0.8453	0.8140	0.8461	0.8473	0.8492	0.8462
	Average	0.8639	0.8336	0.8648	0.8658	0.8763	0.8649
70ND	Vasieck	0.8270	0.7719	0.8275	0.8330	0.8591	0.8275
	Set12	0.8262	0.7772	0.8267	0.8302	0.8574	0.8267
	BSD68	0.7921	0.7393	0.7922	0.7969	0.8085	0.7923
	Average	0.8151	0.7628	0.8154	0.8200	0.8416	0.8155
80ND	Vasieck	0.7548	0.6661	0.7540	0.7661	0.8121	0.7540
	Set12	0.7553	0.6723	0.7545	0.7649	0.8111	0.7545
	BSD68	0.7153	0.6347	0.7142	0.7245	0.7521	0.7144
	Average	0.7418	0.6577	0.7409	0.7518	0.7918	0.7410
90ND	Vasieck	0.6354	0.5013	0.6326	0.6528	0.6874	0.6326
	Set12	0.6371	0.5031	0.6339	0.6538	0.6862	0.6340
	BSD68	0.5968	0.4804	0.5939	0.6109	0.6237	0.5941
	Average	0.6231	0.4950	0.6201	0.6392	0.6658	0.6203

which is then used to sort all the input values in the window Wi, adders, multiplexers which is used to select the final value based on control signals, noise detector, shift operations which are used to multiply or divide a number with rounding off, and dividers that are used to get the mean of noise-free pixels (number of noise-free pixels may change for different window).

The DBNRATMTF [22] algorithm requires 25 CS blocks, 13 adders, 3 MUXs, 14 noise detectors, whereas ATM[23] algorithm requires 124 CS blocks, 10 adders, 12 MUXs, and 50 noise detectors. The Modif_A1[24] and Modif_A2[24] algorithms consume the same amount of 50 CS blocks, 28 adders, 5 MUXs computing resources, and performing winsorization in the Modif_A2[24] algorithm require more hardware resources than the Modif_A1[24]. The ThreeSta.[25] algorithm produces denoised image in three stages, in which the entire image is processed and the output image is

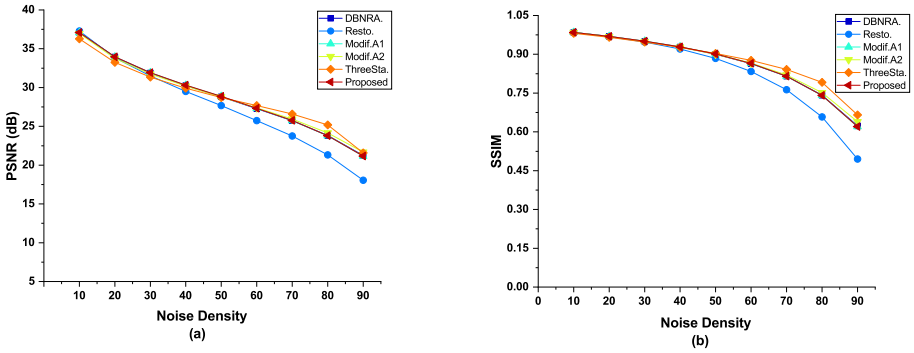

Fig. 15: Average PSNR and SSIM of three datasets

Table 4: Comparison of computing resource requirements for different high noise density denoising algorithms

Design	Line buffers	Counter required ?	Compare swap	Adders	Multiplexers	Noise detectors	Shift operations	Dividers
DBNRATMTF [22]	2	Yes	25	13	3	14	4	Yes
ATM [23]	4	Yes	124	10	12	50	6	Yes
Modif_A1[24]	2	Yes	50	28	5	16	4	Yes
Modif_A2[24]	2	Yes	50	28	5	32	4	Yes
ThreeSta.[25]	4	Yes	199	37	7	60	5	Yes
Proposed	2	No	0	22	4	21	4	No

obtained in stage 1, which is then followed by stage 2 and then the stage 3 operation. These three stages make the algorithm consume more computing resources and cause delay in obtaining the final denoised output image. All existing algorithms require counter and dividers, further increasing hardware resources.

In the proposed architecture, the necessity to have counters to count the noise pixels and noise-free pixels is eliminated. In the existing algorithms, all the pixels in the chosen window are sorted in ascending or descending order before computing unsymmetrical mean that uses CS block in the existing algorithms. The proposed architecture doesn't require any CS block to find the mean of noise-free pixels that reduces the hardware cost. The proposed architecture requires only 22 adders, 4 MUXs, and 21 noise detectors (including detecting salt, pepper, either salt or pepper noise). To determine the mean of a number, division operations need to be performed. The divisor will vary from window to window in the proposed and existing methods. The direct implementation of dividers consumes more hardware, and this is avoided in the proposed architecture by selecting the precalculated division which is implemented with simple shift and addition operations.

The proposed feedback decision-based trimmed mean filter architecture is designed in Verilog hardware description language and ASIC synthesis is carried out in Cadence genus compiler with GPDK 90nm processing technology. The synthesis details of the proposed architecture is provided in Table 5. The proposed architecture requires 14465 μm^2 of area, consumes 0.988 mW of power, has a critical path delay of

Table 5: Synthesis result of proposed architecture

Architecture	Gate Count	Area (μm^2)	Power (mW)	Critical path delay (ps)	Max. clock freq. (MHz)
Proposed	1729	14465	0.988	1937	516

1937 ps, and operates at a maximum frequency of 516 MHz. From Table 4, it is evident that the direct implementation of existing algorithms DBNRATMTF [22], ATM[23], Modif_A1[24], Modif_A2[24], and ThreeSta.[25] into hardware requires more computing resources than the proposed architecture. The proposed architecture delivers similar image quality compared to existing algorithms with a huge reduction in hardware.

4 Conclusion

The amount of noise-free pixels in the selected window varies from window to window, making it difficult to design a hardware architecture for the removal of high-density salt and pepper noise in images. In contrast to state-of-the-art algorithms, a feedback decision-based trimmed mean filter architecture is described that uses a very less hardware resources. The counter that counts the number of noise-free pixels and the sorting procedure on the input pixels are eliminated in the proposed architecture. The division operation in the proposed architecture is carried out with right shift and addition operations, which results in a reduction in hardware resource requirements. The denoised image quality of the proposed architecture is similar to the existing methods. Based on our literature survey, this appears to be the first VLSI architecture that is successful in removing the salt and pepper noises from images even at higher noise densities.

Acknowledgement. The authors would like to thank SMDP-C2SD Project, NIT Calicut, funded by MeitY, Govt. of India, for providing research facility and technical support.

Data availability. Data will be made available on reasonable request.

Declaration of competing interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R.C. Gonzalez and R.E. Woods, *Digital Image Processing.* Pearson Education, 2007

[2] M. Monajati and E. Kabir, "A Modified Inexact Arithmetic Median Filter for Removing Salt-and-Pepper Noise From Gray-Level Images," *IEEE Trans. Circuits Syst. II,* vol. 67, no. 4, pp. 750-754, April 2020, doi: 10.1109/TCSII.2019.2919446.

[3] S.-H. Lin, P.-Y. Chen, and C.-K. Hsu, "Modular design of high-efficiency hardware median filter architecture," *IEEE Trans. Circuits Syst. I*, vol. 65, no. 6, pp. 1929–1940, June 2018, doi: 10.1109/TCSI.2017.2770216.

[4] C. Lin, W.-T. Chen, Y.-C. Chou, and P.-Y. Chen, "A novel comparison-free 1-d median filter," *IEEE Trans. Circuits Syst. II*, vol. 67, no. 7, pp. 1329–1333, July 2020, doi: 10.1109/TCSII.2019.2936230.

[5] J. Astola and P. Kusmanen. *Fundamentals of non linear digital filtering.* FL:CRC press, 1997.

[6] Hwang, H.,Haddad, R.A."Adaptivemedian filters: new algorithms and results." *IEEE Trans. Image Process.* 4(4), 499–502 (1995)

[7] Zhang, S., Karim, M.A.: "A new impulse detector for switching median filters." *IEEE Signal Process. Lett.* 9(11), 360–363 (2002)

[8] 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, pp. 1506-1516, June 2006, doi: 10.1109/TIP.2005.871129.

[9] K. S. Srinivasan and D. Ebenezer, "A New Fast and Efficient Decision-Based Algorithm for Removal of High-Density Impulse Noises," *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 189-192, March 2007, doi: 10.1109/LSP.2006.884018.

[10] F. Ahmed and S. Das, "Removal of High-Density Salt-and-Pepper Noise in Images With an Iterative Adaptive Fuzzy Filter Using Alpha-Trimmed Mean," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 5, pp. 1352-1358, Oct. 2014, doi: 10.1109/TFUZZ.2013.2286634.

[11] Esakkirajan, S.,Veerakumar, T., Subramanyam, A.N., PremChand, C. "Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter." *IEEE Signal Process. Lett.* 18(5), 287–290 (2011)

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

[13] Lu, C.-T., Chou, T.-C. "Denoising of salt-and-pepper noise corrupted image using modified directional-weighted-median filter." *Pattern Recogn. Lett.* 33(10), 1287–1295 (2012)

[14] Ma, H., Nie, Y. "A two-stage filter for removing salt-and-pepper noise using noise detector based on characteristic difference parameter and adaptive directional mean filter." *PLoS ONE* 13(10), 1–24 (2018)

[15] Habib, M., Hussain, A., Rasheed, S., Ali, M.: "Adaptive fuzzy inference system based directional median filter for impulse noise removal." *AEU Int. J. Electr. Commun.* 70(5), 689–697 (2016)

[16] Kiani,V.,Zohrevand, A. "Afuzzy directionalmedian filter for fixedvalue impulse noise removal. In: 7th Iranian Joint Congress on Fuzzy and Intelligent Systems" *(CFIS). IEEE*, vol. 2019, pp. 1–4 (2019)

[17] Balasubramanian, G., Chilambuchelvan, A., Vijayan, S., Gowrison, G.: "Probabilistic decision based filter to remove impulse noise using patch else trimmed median." *AEU Int. J. Electr. Commun.* 70(4), 471–481 (2016)

[18] Roy, A., Laskar, R.H.: "Non-casual linear prediction based adaptive filter for removal of high density impulse noise from color images." *AEU Int. J. Electron. Commun.* 72, 114–124 (2017)

[19] Erkan, U., ökrem, L.G., Engino˘glu, S.: "Different applied median filter in salt and pepper noise" *Comput. Electr. Eng.* 70, 789–798 (2018)

[20] Erkan, U., ökrem, L.G.: "A new method based on pixel density in salt and pepper noise removal" *Turk. J. Electr. Eng. Comput. Sci.* 26(1), 162–171 (2018)

[21] B. Garg and K. V. Arya "Four stage median-average filter for healing high density salt and pepper noise corrupted images." *Multimed. Tools Appl.*, vol. 79(43), pp. 32305–32329, 2020.

[22] K. Vasanth, "A Decision Based Neighbourhood Referred Asymmetrically Trimmed Modified Trimean for the Removal of High Density Salt and Pepper Noise in Images and Videos," *Wireless, Pers. Commun,* vol. 120, pp. 2585–2609, 2021, https://doi.org/10.1007/s11277-021-08547-4

[23] B. Garg, "Restoration of highly salt-and-pepper-noise-corrupted images using novel adaptive trimmed median filter," *Signal, Image and Video Processing*, vol 14(8), pp. 1555-1563, 2020.

[24] Goel, N. "Modified decision based two-phase unsymmetrical trimmed/winsorized mean filter for removal of very high density salt and pepper noise from images and videos," *Multimed. Tools Appl.* 81, 32953–32979 (2022). https://doi.org/10.1007/s11042-022-12876-2

[25] Bindal, N., Garg, B. "Novel three stage range sensitive filter for denoising high density salt & pepper noise," *Multimed. Tools Appl.* 81, 21279–21294 (2022). https://doi.org/10.1007/s11042-022-12574-z

[26] K. N. Vijeyakumar, P. T. N. K. Joel, S. H. S. Jatana, N. Saravanakumar, and S. Kalaiselvi, "Area efficient parallel median filter using approximate comparator and faithful adder," *IET Circuits, Devices & Syst.*, vol. 14, no. 8, pp. 1318–1331,

November 2020. doi: https://doi.org/10.1049/iet-cds.2020.0059

[27] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc Eighth IEEE Int. Conf. Comput. Vision,* 2001, pp. 416-423 vol.2, doi: 10.1109/ICCV.2001.937655.

[28] K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142-3155, July 2017, doi: 10.1109/TIP.2017.2662206.

[29] *vasicek image dataset*, accessed Aug. 11, 2022. [Online]. Available: http://www.fit.vutbr.cz/~vasicek/imagedb/?lev=20&knd=original

[30] B. K. Mohanty, A. Mahajan and P. K. Meher, "Area- and Power-Efficient Architecture for High-Throughput Implementation of Lifting 2-D DWT," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 7, pp. 434-438, July 2012, doi: 10.1109/TCSII.2012.2200169.