SIViP (2016) 10:575–584
DOI 10.1007/s11760-015-0780-6

ORIGINAL PAPER

A study on genetic expression programming-based approach for impulse noise reduction in images

Vivek Singh Bhadouria[1] · Dibyendu Ghoshal[1]

Received: 20 October 2013 / Revised: 30 November 2014 / Accepted: 10 May 2015 / Published online: 27 May 2015
© Springer-Verlag London 2015

Abstract Existing impulse noise reduction techniques perform well at low noise densities; however, their performance drops sharply at higher noise densities. In this paper, we propose a two-stage scheme to surmount this problem. In the proposed approach, first stage consists of impulse detection unit followed by the filtering operation in the second stage. We have employed a genetic expression programming-based classifier for the detection of impulse noise-corrupted pixels. To reduce the blurring effect caused due to filtering operation on the noise-free pixels, we filter the detected noisy pixels only by using a modified median filter. Better peak signal-to-noise ratio, structural similarity index measure, and visual output imply the efficacy of the proposed scheme for noise reduction at higher noise densities.

Keywords Filtering · Genetic expression programming · Impulse detection · Impulse noise · Noise reduction

1 Introduction

Digital images are often corrupted by the impulse noise during the acquisition or transmission stage. Therefore, it is essential to reduce the noise before further processing to extract any relevant information from the image, e.g., image segmentation, object detection etc. [1]. Depending upon the intensity distribution, impulse noise can be classified into two categories, namely: (1) salt-and-pepper noise and (2) random-valued impulse noise. Salt-and-pepper noise-corrupted pixels can take either 0 or 255 gray value in any 8-bit gray-scale image; however, random-valued impulse noise-corrupted pixels can take any random value in the allowed gray-scale range of the image. Various efficient approaches have been proposed for the salt-and-pepper noise reduction [2,3]. However, developing a robust approach for random-valued impulse noise reduction still remains a challenge [4–6]. Therefore, in this paper, we will restrict our discussion to the random-valued impulse noise-corrupted images only.

Various methods have been proposed for the impulse noise reduction among which standard median filter (SMF) holds a prominent place due to its mathematical simplicity in operation. However, when the noise density is high, SMF fails to preserve the image details during the filtering operation [3]. To overcome the drawbacks of SMF, another class of impulse noise filters, i.e., switching median filters were proposed. Such filters consist of two stages in which the first stage deals with the detection of noisy pixels with a follow-up filtering stage. In such methods, filtering operation is applied only to the detected noisy pixels. Adaptive center-weighted median filter (ACWMF) [7], alpha-trimmed mean-based method (ATMBM) [1], differential rank-order impulse detector (DRID) [8], nonlinear adaptive lower-upper-middle-based filter (NALUMBF) [9], decision-tree-based denoising method (DTBDM) [10] are some of the efficient methods belonging to this class. These schemes perform efficiently at low noise density, but their performance degrades sharply at high noise densities. In this paper, we discuss an experiment in which we investigated the performance of genetic expression programming (GEP)-based impulse detector, followed by a modified median filtering operation in the second stage. The proposed method has been found to be robust in performance, especially at high noise densi-

✉ Vivek Singh Bhadouria
vivek@hotmail.de

1 Department of Electronics and Communication Engineering, National Institute of Technology, Agartala 799055, Tripura (West), India

ties. As per our knowledge, no such GEP-based impulse reduction scheme exists in the published or online literature.

The organization of the paper is as follows: A brief introduction to GEP has been presented in Sect. 2. Section 3 describes the feature vectors considered in the present study to train the GEP-based classifier. In Sect. 4, we present the proposed impulse noise reduction scheme, followed by results and discussion in Sect. 5. Finally, conclusions are drawn in Sect. 6.

2 Genetic expression programming

GEP is a variant of genetic programming in which tree-based structures evolve that models the relationship between the input and output. GEP consists of linear chromosomes of fixed length similar to genetic algorithms and complex structures of different sizes and shape similar to parse trees in genetic programming. In GEP, computer program evolves to find the solution of the candidate problem based on Darwin' s theory of reproduction, crossover, and mutation. These processes form the basis to evolve a genetic expression tree from the GEP. In order to reach to the optimal solution, trees with worst fitness are 'killed.' After killing the trees with worst fitness, remaining population comprises of surviving trees based on accepted selection mechanism.

In GEP, chromosomes are made up of multiple genes where each gene comprises of a head and a tail part. Head part contains the detailing symbols specific to the functions and terminal operators, e.g., $+$, $-$, / etc., and tail contains the symbols specific to terminal operators only. Mathematical relation exhibiting the relationship between the head length and tail length can be written as:

$$t_c = (n-1)\,h_c + 1 \qquad (1)$$

where h_c is the head length, t_c is the tail length of the chromosome, and n is the number of argument within the function. The flowchart illustrating the architectural flow has been depicted in Fig. 1 along with its optimization flow diagram in Fig. 2. Various functional steps from Fig. 1 have been explained as follows.

2.1 Define the initial population

Similar to other evolutionary algorithms, GEP starts with an initial population of randomly generated chromosomes; all future succeeding populations evolve from this initially generated population. This initial random population is also the first solution to the candidate problem at hand. As these initial populations are randomly generated, therefore they are not still the best solution (adapted according to the problem environment). In the evolution of new generations, genetic operators evolve each of the individual by 'mating' them

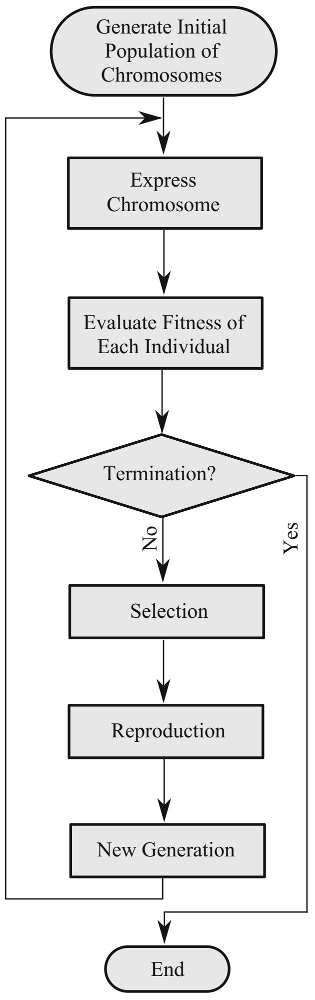
Generate Initial Population of Chromosomes
Express Chromosome
Evaluate Fitness of Each Individual
Termination?
No
Yes
Selection
Reproduction
New Generation
End

Fig. 1 Flowchart of the GEP

with other individual in the population. These genetic operators are selected before running the problem, and it largely depends upon the complexity of the problem. For instance, genetic operators for mathematical model may include '+,' '−,' '×,' etc., and 'AND,' 'OR,' 'NOT,' etc., for Boolean logical expressions.

2.2 Express chromosome

In this stage, chromosomes are expressed in terms of expression trees. The structure of each expression tree is designed in a way that first node corresponds to the beginning of a new gene. The evolved offspring from the first node is dependent upon the number of arguments. In this process, functions can have many arguments, however, the terminal takes 0 arguments.

2.3 Fitness evaluation

It is crucial to choose the proper fitness function according to the desired objectives of the problem set. To evaluate the

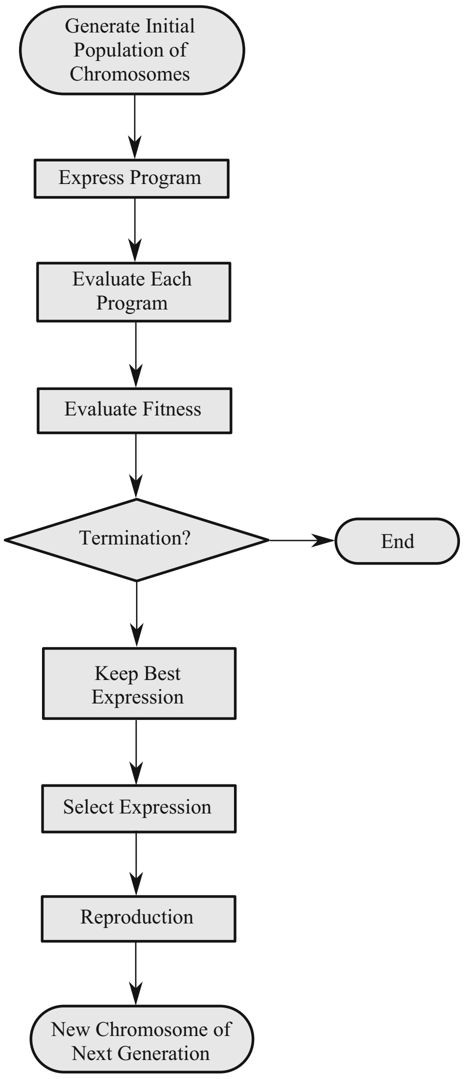

Fig. 2 GEP optimization scheme

fitness of any chromosome population, we have resorted to maximizing the sensitivity and specificity because they are widely used metric to determine the performance of two-class classifiers. To explain the sensitivity and specificity, let us assume any two-class classification problem with class-label $\{L_1, L_2\}$. For any input instance, candidate classifier can have four following possible types of output.

1. If any object from class L_1 has been assigned label L_1. This input instance is called as true positive, T_p.
2. If any object from class L_2 has been assigned label L_2. This input instance is called as true negative, T_n.
3. If any object from class L_1 has been assigned label L_2. This input instance is called as false positive, F_p.
4. If any object from class L_2 has been assigned label L_1. This input instance is called as false negative, F_n.

Therefore, depending upon the correct classification, sensitivity and specificity can be defined as:

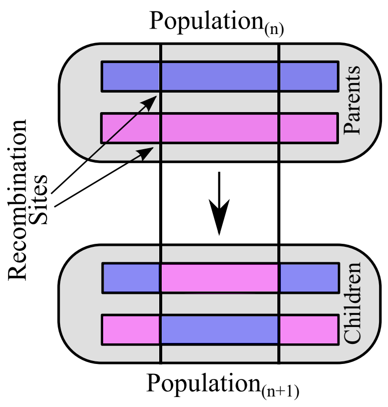

Fig. 3 Illustration of chromosome recombination

$$\text{Specificity} = \frac{T_\mathrm{n}}{T_\mathrm{n} + F_\mathrm{p}} \tag{2}$$

$$\text{Sensitivity} = \frac{T_\mathrm{p}}{T_\mathrm{p} + F_\mathrm{n}} \tag{3}$$

Therefore, initial populations evolve with a target to maximize the sensitivity and specificity. During the evolution, generation yielding the maximum sensitivity and specificity is considered as the optimal generations. Expression tree corresponding to the optimal generations is carried forward for remaining calculations.

2.4 Storing the best generation

In GEP, the fundamental process of evolution is the generation of offsprings from the two superior parent individuals to achieve 'etilism.'[1] The best individuals from the parent generation produce offsprings in future generations with most desirable features; however, the individuals with less desirable features are removed. On the basis of this fact, our model maximizes the sensitivity and specificity of the classifier and yields superior classification capabilities.

2.5 Selection

This step assures that the best individuals are used to produce the offsprings for the future generations. The selection of chromosomes for their fitness is carried out during the 'tournament' selection for reproduction and mutation. The competition between the chromosomes depends upon the tournament size that is adjusted and defined by the user. Summarily, greater tournament size results in more competitive selection procedure, as a result weak chromosomes are less likely to compete for the survival.

[1] 'Etilism' is the cloning of best chromosomes to next population or generation.

2.6 Reproduction

In the reproduction phase, new offsprings are born from the parent chromosome. Here, we have to consider the genetic operators that are responsible for evolution. In particular, we apply the genetic operators known as mutation and recombination explained as follows:

2.6.1 Mutation

It is the process of creating new model that is mutated randomly from an existing model. For mutation, a parent chromosome is selected with a probability defined by its fitness. After the selection of parent chromosome, mutation point is defined on the parent chromosome which alters one or more gene representing the new mutated individual that is added to the population.

2.6.2 Recombination

In the recombination, parent chromosomes are matched and split up at identical points in order to determine the recombination points, shown in Fig. 3. The spliced parts of each genes are exchanged between the two preselected parent chromosomes on the basis of probability. This process results in the birth of two new offsprings.

2.7 Prepare new expressions of next generation

In this step, tournament losers are replaced with the new individuals by the reproduction in population.

2.8 Termination criteria

Defining a termination criteria for evolutionary algorithm is essential in order to stop the evolution. In the present study, we terminate our program when there is no change in the either sensitivity or specificity over 1000 generations.

3 Feature vectors

Selection of appropriate training feature vectors plays an important role in determining the ultimate performance of any classifier. For training purpose, we have extracted the below-mentioned feature vectors from the scaled Lena image of dimension (128×128) shown in Fig. 4a, corrupted with 50 % noise density, shown in Fig. 4b. Corrupting the image with 50 % noise density makes number of noisy pixels equal to the noise-free pixels in the input dataset, thereby avoiding any bias that is likely to arise during the training of the classifier. The target image for the GEP-based classifier has been shown in Fig. 4c.

3.1 Difference of the median and central pixel

Median value is a good estimator of the pixel distribution in any local window of dimension 3×3. If the central pixel is noise free, then its intensity value will lie close to the median value (of the pixels contained in the local window) or the difference between them will be less. On the other hand, if the central pixel is noisy, then there will be a large difference in between the median of the local window and the central pixel value. As a result, this difference can play an important role in determining the pixel' s nature for any local window W_3, and the difference between the central pixel $X(i, j)$ and the median of the pixels $\text{Median}(W_3(i, j))$ can be calculated as:

$$D\left(W_3\left(i, j\right)\right) = \text{abs}\left(\text{Median}\left(W_3\left(i, j\right)\right) - X\left(i, j\right)\right) \quad (4)$$

where X is the noisy version of original image I and W_3 represents any arbitrary local window of dimension 3×3.

3.2 Rank-ordered information

Position of the central pixel within a local window in any sorted series (i.e., arranging the pixel values in row major form followed by the sorting operation) carries vital information regrading its nature [8]. If the central pixel is noise free, then its location, R, will lie close to the median position. However, if the pixel is noisy or an edge pixel, then it will lie at the extreme locations (or close to extreme location) in the sorted series. Therefore for a 3×3 window, $1 \le R \le 9$.

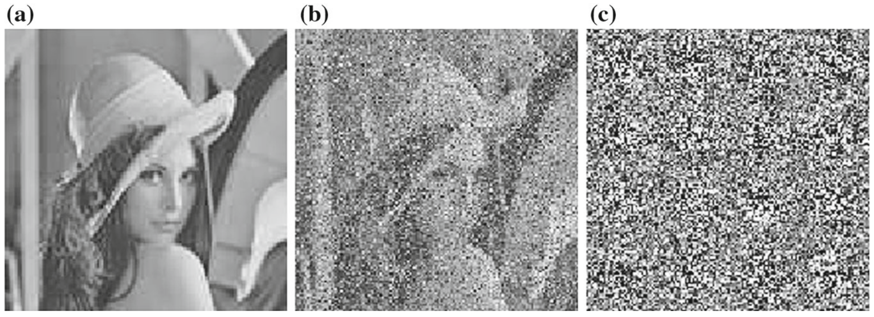

Fig. 4 Training dataset for GEP-based classifier. **a** Original noise-free Lena image, **b** Lena image corrupted with 50 % random-valued impulse noise, and **c** target image

3.3 Robust outlyingness ratio, ROR

Local statistical information within the window can yield important information regarding the pixel distribution. Robust outlyingness ratio proposed by Bo and Zhouping provides a robust estimation of the pixel' s nature [11]. ROR value is high for the noisy pixels, compared to noise-free pixels. To calculate ROR, we have considered a window of dimension (5×5), W_5. ROR can be calculated as:

$$\mathrm{ROR}\left(W_5\left(i,j\right)\right) = \left|\frac{W_5\left(i,j\right) - \mathrm{Median}\left(W_5\left(i,j\right)\right)}{\mathrm{MADN}\left(W_5\left(i,j\right)\right)}\right| \quad (5)$$

where MADN represents normalized median absolute deviation and can be calculated by:

$$\mathrm{MADN}\left(W_5\left(i,j\right)\right) = \frac{\mathrm{Median}\left(\left|W_5 - \mathrm{Median}\left(W_5\left(i,j\right)\right)\right|\right)}{0.6457} \quad (6)$$

4 Proposed approach

GEP is a class of evolutionary algorithm based on the principle of natural selection. GEP works by evolving population of fixed length of chromosomes through genetic operators, i.e., selection, crossover, and mutation. For detailed explanation of GEP, we refer to the work of Ferreira [12], Zhou et al. [13] and Sermpinis et al. [14]. The fitness function used for the GEP classifier was to maximize the sensitivity and specificity of the classifier, discussed in Sect. 2.3 [15].

A classification problem can be seen as $F : r^u \rightarrow s^v$, where u-dimensional signal is fed to GEP framework and v-dimensional vector is obtained as the output. Input feature vectors can be written as $r = (\mathrm{D}, \mathrm{R}, \mathrm{ROR})_{16384\times 3}$ where D, R and ROR represent the difference in between the central pixel intensity value and median of the local window, rank-ordered information and ROR, respectively. Target vector can be expressed as $s = (s)_{16384\times 1} \mid s \in \{0, 1\}$.

In the first stage, feature vectors are fed to GEP-based classifier which determines the nature of the central pixel depending upon the input values. In the second stage, if the pixel is found out to be noisy, then the detection map and noisy image are fed to the filtering unit in order to apply the filtering operation on the detected noisy pixels, shown in Fig. 5. If the pixel is found to be noise free, then it is left intact. Performance of the noise reduction algorithm largely depends upon the efficiency of the detection unit. If the detection unit performs poorly in the detection of noisy pixels, i.e., low correct classification rate, then the restored image will contain noisy pixels, even after the filtering operation. On the other hand, if impulse detection stage yields high false classification rate, then the restored image will be blurred in nature

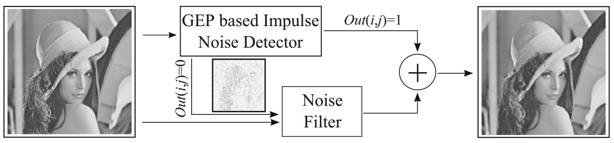

Fig. 5 Flowchart of the proposed impulse noise reduction scheme

due to filtering operation on the noise-free pixels. Standard median filter is a widely used approach for the reduction in impulse noise. But it performs poorly at high noise densities (i.e., $P > 30\,\%$, where P is the noise density) because of the following reasons. (1) Generally, SMF is employed across the whole image without considering the nature of the central pixel. As a result, this approach performs filtering operation on the noise-free pixels as well which leads to the blurring of the image details; (2) SMF considers all of the elements of the local window to estimate the noise-free value for the noisy central pixel. This approach can again yield a noisy estimate in the case if the number of noisy pixels in the window is more than 4 (for a 3×3 window).

In the present study, we have employed a variant of SMF on the lines of [16] in which we consider noise-free pixels only in order to estimate a value for the noisy central pixel. To perform this operation, we need noise-free pixels which can be obtained by the element by element matrix multiplication of the local image window with the corresponding detection map window. Detection map is binary in our case, i.e., a location can have a value of either 0 or 1. If we assign 0 values to the noisy pixels, then in the multiplied output we will obtain noise-free pixel values along with some zero values. This operation can be written as:

$$\mathrm{Out}_{\mathrm{Mult}} = \left(\left(X\left(W_3\right)\right) \times \mathrm{Map}\left(W_3\right)\right) \cap X\left(W_3\right) \quad (7)$$

Finally, estimate for noisy central pixel can be made by using:

$$\hat{X}\left(i,j\right) = \mathrm{Median}\left(\mathrm{Out}_{\mathrm{Mult}}\right) \quad (8)$$

5 Results and discussion

5.1 Performance of the GEP-based impulse classifier

Various GEP simulation parameters considered during the study have been listed in Table 1. The final expression tree (ET) obtained from the GEP has been shown in Fig. 6. In Fig. 6, $D0$, $D1$, and $D2$ represent D, R, and ROR, respectively. The net expression to detect an impulse noise-corrupted pixel can be written as:

$$\mathrm{MAP}_{\mathrm{GEP}} = (\mathrm{ET})_1 + (\mathrm{ET})_2 + (\mathrm{ET})_3 \quad (9)$$

Values of various constants used in Fig. 6 are as follows: (1) For $\mathrm{ET}_1 : C_3 = 2.97,\ C_8 = 6.45$, (2) For $\mathrm{ET}_2 : C_5 = -9.85,\ C_9 = 9.56$, (3) For $\mathrm{ET}_3 : C_5 = 9.46,\ C_9 = 8.57$.

Table 1 GEP simulation parameters

Parameters	Value
Function set	$+, -, /, \times, \leq, <, \geq, >, ==, !=$, atan
Terminal set	D, R, ROR
Head length	7
Gene length	10
Number of genes	3
Chromosome length	30
Mutation rate	0.0051
Gene recombination rate	0.1

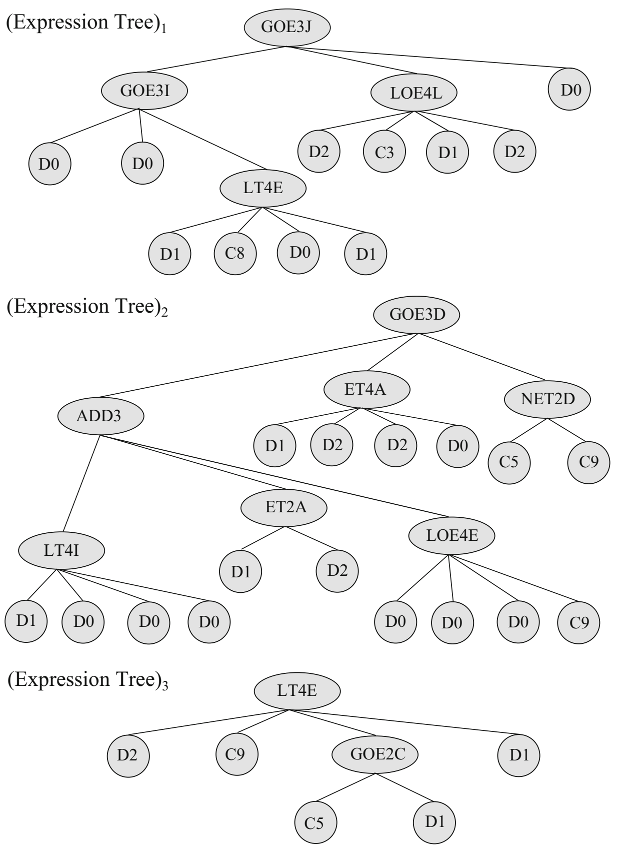

Fig. 6 Expression tree for the GEP-based impulse classifier

Floating point numerical constants are crucial in designing the evolutionary technique-based mathematical model for any problem [17]. We have also used the numerical constants to model the GEP-based classifier for automated noisy pixel classification. These random numeric constants (RNC) are randomly generated by the GEP during the run and are fine-tuned with the progressing population. Before starting the simulation, a range of values is required to be provided to the GEP for RNC generation and RNCs are generated randomly and are fine-tuned during the run [18]. There is no strict rule for determining the optimal range of the RNC values [18]; however, experimentally we have determined to set the range to adequately produce positive and negative random numbers. For the present work, we set the range to [−10, 10]. For a detailed information on RNC generation and fine-tuning, we refer to chapter 5 of the Ferreira' s work [18].

Various operators of ETs, shown in Fig. 6, have been listed as following[2]:

$$\text{GOE2C}(x, y) = \begin{cases} (x + y) & \text{if } x \geq y \\ (x - y), & \text{otherwise} \end{cases} \tag{10}$$

$$\text{ET2A}(x, y) = \begin{cases} (x) & \text{if } x == y \\ (y), & \text{otherwise} \end{cases} \tag{11}$$

$$\text{NET2D}(x, y) = \begin{cases} (x \times y) & \text{if } x! = y \\ (x/y), & \text{otherwise} \end{cases} \tag{12}$$

$$\text{GOE3D}(x, y, z) = \begin{cases} (x + y) & \text{if } (x + y) \geq z \\ (x - z), & \text{otherwise} \end{cases} \tag{13}$$

$$\text{GOE3I}(x, y, z) = \begin{cases} (x \times y) & \text{if } (x + y) \geq z \\ (x \times z), & \text{otherwise} \end{cases} \tag{14}$$

$$\text{GOE3J}(x, y, z) = \begin{cases} (x \times y) & \text{if } (x + y) \geq z \\ (x/z), & \text{otherwise} \end{cases} \tag{15}$$

$$\text{ET4A}(a, b, c, d) = \begin{cases} (c) & \text{if } (a == b) \\ (d), & \text{otherwise} \end{cases} \tag{16}$$

$$\text{LT4E}(a, b, c, d) = \begin{cases} (a + b) & \text{if } ((a + b) < (c + d)) \\ (c \times d), & \text{otherwise} \end{cases} \tag{17}$$

$$\text{LOE4E}(a, b, c, d) = \begin{cases} (a + b) & \text{if } ((a + b) \leq (c + d)) \\ (c \times d), & \text{otherwise} \end{cases} \tag{18}$$

$$\text{LT4I}(a, b, c, d) = \begin{cases} (a \times b) & \text{if } ((a + b) < (c + d)) \\ (c \times d), & \text{otherwise} \end{cases} \tag{19}$$

$$\text{LOE4L}(a, b, c, d) = \begin{cases} \text{atan}\,(a \times b) & \text{if } ((a + b) \leq (c + d)) \\ \text{atan}\,(c \times d), & \text{otherwise} \end{cases} \tag{20}$$

The output obtained from the above expression is continuous and therefore to convert into binary logic output, a two-state threshold operation can be employed on MAP_{GEP}

$$\text{Out}\,(i, j) = \begin{cases} 0, & \text{if } 0 \leq PMAP_{\text{GEP}} < 0.5 \\ 1, & \text{if } 0.5 \leq MAP_{\text{GEP}} \leq 1 \end{cases} \tag{21}$$

The proposed classifier has been found to be robust in performance from its high area under the curve (AUC) of 0.8466 which implies the probability with the an instance of noisy pixel will be classified as noisy. The specificity and sensitivity of the proposed detector have been found to be 92.51 and 77.99 %, respectively. The correct classification rate was found to be 85.26 %.

To evaluate the performance of the proposed algorithm, various standard test images viz. Lena, Elaine, Goldhill,

[2] For the simplicity of expressing the mathematical equations, we have used arbitrary variables for 2, 3, and 4 variable operators by (x, y), (x, y, z), and (a, b, c, d), respectively. We have considered a post-order traversing scheme to represent the actual variables (represented by leaves in the ET) by the arbitrary variables mentioned above.

Bridge, and Aerial were artificially corrupted with the impulse noise, varying from 10 to 70 % noise density with the step increment of 20 %. Performance of the various approaches has been evaluated on the basis of quantitative superiority (i.e., peak signal-to-noise ratio, PSNR and structural similarity index measure, SSIM) and qualitative superiority (i.e., visual output). PSNR between original image, I, and restored image, K, of dimension $(M \times N)$ can be calculated by using the following expression:

$$\mathrm{PSNR}\,(I, K) = 10\log_{10}\left(\frac{255^2}{\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\{I\,(m,n) - K\,(m,n)\}^2}\right) \tag{22}$$

SSIM between the original image and restored image can be calculated as:

$$\mathrm{SSIM}\,(I, K) = \frac{(2\mu_I\mu_K)\,(2\sigma_{IK} + C_2)}{\left(\mu_I^2 + \mu_K^2 + C_1\right)\left(\sigma_I^2 + \sigma_K^2 + C_2\right)} \tag{23}$$

where μ and σ represent mean intensity and standard deviation from the peak intensity. C_1 and C_2 being the constants, $\sigma_{I,K}$ (i.e., standard deviation at any pixel) can be calculated as:

$$\sigma_{IK} = \frac{1}{J-1}\sum_{j=1}^{J}\left(I_j - \mu_I\right)\left(K_j - \mu_K\right) \tag{24}$$

Tables 2, 3, 4, and 5 list the qualitative results for various images at 10, 30, 50, and 70 % noise density, respectively. It can be observed from Tables 2, 3, and 4 that the performance of ACWM, ATMBM, and DRID drops sharply with the increasing noise density. Therefore, these approaches are inefficient for noise reduction at high noise densities. DTBDM involves decision-tree-based impulse detector

Table 2 PSNR and SSIM values of various algorithms for numerous standard test images at 10 % noise density

Approach	Attribute	Image				
		Lena	Elaine	Goldhill	Bridge	Aerial
ACWM	PSNR (dB)	33.93	32.25	30.88	26.14	27.29
	SSIM	0.9109	0.7781	0.8443	0.7771	0.8782
ATMBM	PSNR (dB)	36.79	37.44	33.73	**27.77**	28.54
	SSIM	0.9703	**0.9503**	0.9478	0.8841	0.9325
DRID	PSNR (dB)	34.55	32.66	31.33	26.70	28.01
	SSIM	0.9216	0.7954	0.8607	0.8104	0.8954
NALUMBF	PSNR (dB)	30.81	31.49	28.65	23.97	23.94
	SSIM	0.8569	0.7427	0.7434	0.6046	0.7172
DTBDM	PSNR (dB)	**37.05**	**36.65**	**34.21**	27.70	**29.01**
	SSIM	**0.9713**	0.9419	**0.9522**	**0.8863**	**0.9393**
Proposed	PSNR (dB)	32.92	33.41	30.04	24.64	25.04
approach	SSIM	0.9330	0.8446	0.8540	0.7356	0.8188

Table 3 PSNR and SSIM values of various algorithms for various standard test images at 30 % noise density

Approach	Attribute	Image				
		Lena	Elaine	Goldhill	Bridge	Aerial
ACWM	PSNR (dB)	23.93	23.58	26.82	23.84	23.49
	SSIM	0.7663	0.7306	0.7191	0.6673	0.7283
ATMBM	PSNR (dB)	28.53	29.05	27.60	**23.88**	23.96
	SSIM	0.8273	0.7907	0.7928	**0.7360**	0.7663
DRID	PSNR (dB)	27.56	27.32	26.34	23.42	23.44
	SSIM	0.7760	0.6658	0.7107	0.6816	0.7206
NALUMBF	PSNR (dB)	29.19	30.10	27.61	23.02	22.98
	SSIM	0.8328	0.7222	0.7126	0.5648	0.6766
DTBDM	PSNR (dB)	27.26	27.10	26.42	23.15	23.24
	SSIM	0.7747	0.7264	0.7549	0.7086	0.7430
Proposed	PSNR (dB)	**31.25**	**32.05**	**29.24**	23.64	**24.02**
approach	SSIM	**0.9132**	**0.8294**	**0.8401**	0.7184	**0.7978**

Table 4 PSNR and SSIM values of various algorithms for various standard test images at 50 % noise density

Approach	Attribute	Image				
		Lena	Elaine	Goldhill	Bridge	Aerial
ACWM	PSNR (dB)	21.63	21.68	20.79	19.13	18.51
	SSIM	0.4881	0.4210	0.4506	0.4428	0.4327
ATMBM	PSNR (dB)	21.66	21.72	20.94	19.17	18.55
	SSIM	0.4989	0.4683	0.4795	0.4744	0.4513
DRID	PSNR (dB)	20.94	21.08	20.27	18.78	18.10
	SSIM	0.4484	0.3922	0.4268	0.4234	0.4132
NALUMBF	PSNR (dB)	25.54	26.41	24.41	20.85	20.60
	SSIM	0.7259	0.6276	0.5931	0.4679	0.5227
DTBDM	PSNR (dB)	20.11	20.24	19.54	18.16	17.54
	SSIM	0.3933	0.3654	0.3940	0.4124	0.3940
Proposed	PSNR (dB)	**28.69**	**29.70**	**27.76**	**21.98**	**21.89**
approach	SSIM	**0.8574**	**0.7534**	**0.7564**	**0.5940**	**0.6740**

Table 5 PSNR and SSIM values of various algorithms for various standard test images at 70 % noise density

Approach	Attribute	Image				
		Lena	Elaine	Goldhill	Bridge	Aerial
ACWM	PSNR (dB)	16.77	16.92	16.20	15.27	14.33
	SSIM	0.2159	0.1982	0.2148	0.2219	0.2113
ATMBM	PSNR (dB)	16.70	16.93	16.21	15.26	14.35
	SSIM	0.2149	0.2092	0.2234	0.2363	0.2130
DRID	PSNR (dB)	16.38	16.54	15.94	15.10	14.14
	SSIM	0.1932	0.1781	0.1975	0.2185	0.1991
NALUMBF	PSNR (dB)	19.71	19.78	18.87	17.20	16.20
	SSIM	0.4602	0.4025	0.3760	0.2970	0.2742
DTBDM	PSNR (dB)	15.43	15.60	15.03	14.20	13.48
	SSIM	0.1470	0.1335	0.1560	0.1753	0.1736
Proposed	PSNR (dB)	**23.80**	**23.12**	**22.40**	**19.25**	**17.76**
approach	SSIM	**0.7166**	**0.6121**	**0.5897**	**0.4246**	**0.4040**

which performs efficiently at low noise density, yielding high PSNR and SSIM values evident from Table 2. However, at higher noise densities, decision-tree-based approach falls short due to the intricate detection environment evident from low PSNR and SSIM values in Tables 3, 4 and 5. NALUMBF performs satisfactorily over the wide range of noise but fails to preserve the structural content of the image, evident from the low SSIM values in Tables 4 and 5. Proposed approach performs satisfactorily at low noise density which may be due to the fact that GEP classifier has been trained with the high noise density data. Therefore, classifier may be biased toward the high noise density performance. At high noise densities (i.e., $P > 30\,\%$), proposed approach has been found to yield higher PSNR and SSIM values, compared to existing state-of-art noise reduction schemes.

Qualitative results in terms of the visual output for cropped Elaine image have been shown in Fig. 7. It can be observed from Fig. 7c–e that ACWM, ATMBM, and DRID, respectively, fail to remove the noise present in the image apparent from the noise patches present in the restored image. NALUMBF suppresses the noise to a certain extent but also blurs the image detail during the filtering operation, shown in Fig. 7f. DTBDM also performs poorly at higher noise densities as the restored image still contains the noisy pixels, shown in Fig. 7g. Proposed approach performs well at high noise densities, evident from the restored image, shown in Fig. 7h. A similar observation has also been made with the standard Lena image results, illustrated in Fig. 8. From the fine image details present in the restored image (e.g., hairs on the top right corner in Fig. 7), it can be inferred that the

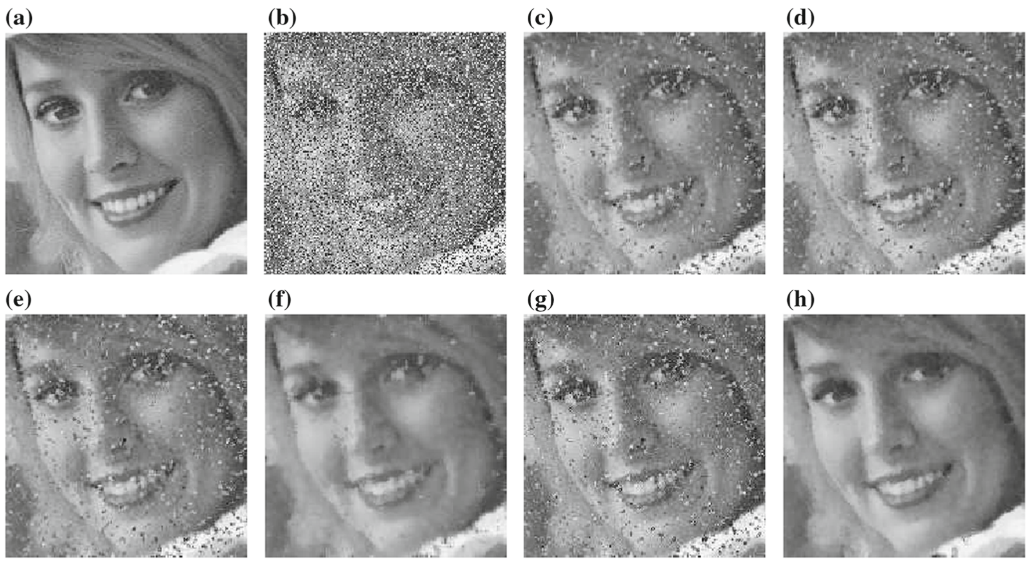

Fig. 7 Simulation results of various algorithms for cropped Elaine image, corrupted with 50 % random-valued impulse noise. **a** Original noise-free Elaine image; **b** Noisy Elaine image; **c** Output of ACWM; **d** Output of ATMBM; **e** Output of DRID; **f** Output of NALUMBF; **g** Output of DTBDM; **h** Output of the proposed approach

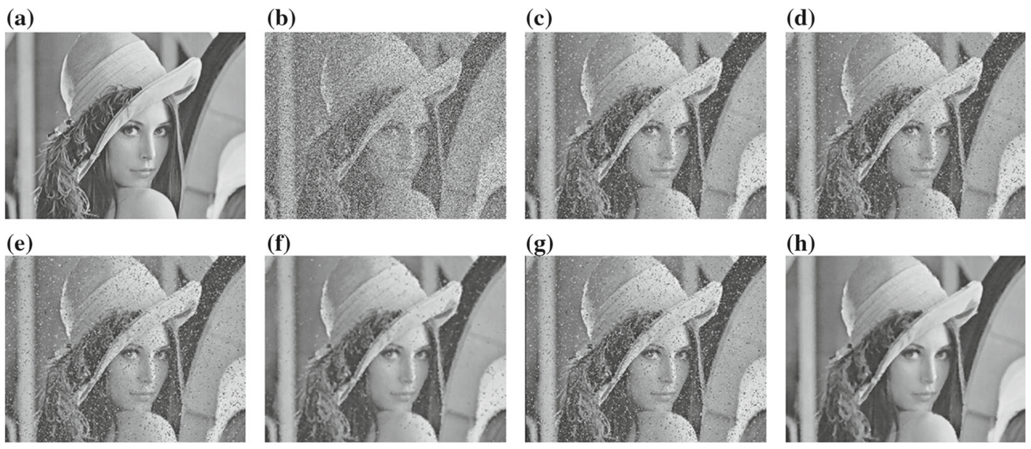

Fig. 8 Simulation results of various algorithms for cropped Lena image, corrupted with 50 % random-valued impulse noise. **a** Original noise-free Elaine imageb) Noisy Elaine image; **c** output of ACWM; **d** output of ATMBM; **e** output of DRID; **f** output of NALUMBF; **g** output of DTBDM; **h** output of the proposed approach

proposed algorithm is capable of preserving the image details while reducing the noise.

During the experimentation, it has been found out that at higher noise density (i.e., $P > 30\,\%$), proposed method requires more than one iteration in order to clean the noise present in the image. At lower noise density, single pass of the proposed scheme reduces the noise present in image. Experimentally, we have observed that for higher noise density (i.e., $P > 30\,\%$), PSNR and SSIM of the candidate noisy image improve with the certain number of iterations of the proposed scheme. PSNR and SSIM variation was found to be purely depending upon the noise present in the image and number of times we applied the filtering operation on the candidate noisy image, shown in Fig. 9 for the Lena image. For 50 % noise density, 2–3 iterations were found to be optimal on variety of images; we have also observed that performing more than three iterations result in drop in the PSNR and SSIM values concluding the blurring phenomena caused by the over-filtering operation. Similarly, for $\approx$70 % noise-corrupted image, $\approx$4–5 iterations are sufficient to reduce the noise present in the image. For the ease of readers, we have enlisted the optimal number of iterations required for various noise densities in Table 6.

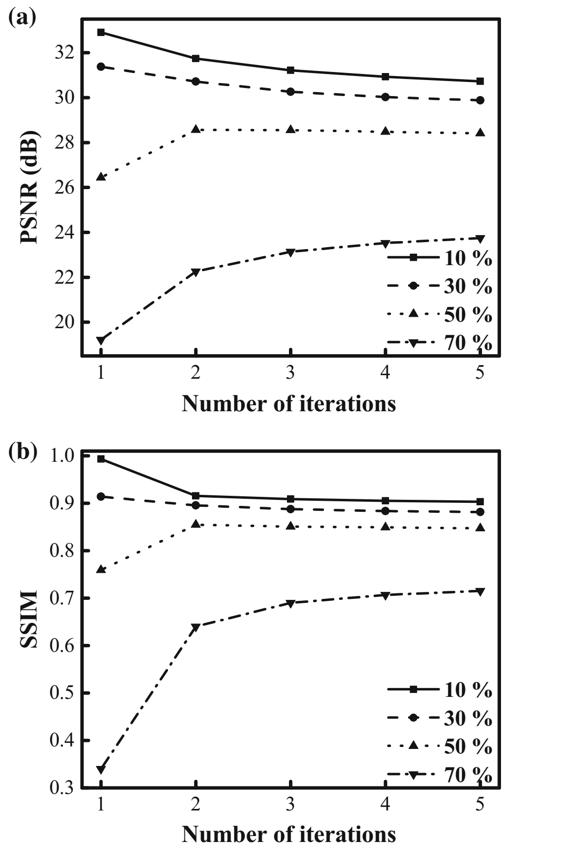

Fig. 9 Variation of PSNR (dB) and SSIM with the iterations for Lena image corrupted with 50 % random-valued impulse noise

Table 6 Optimal number of iterations for various noise densities

Noise density (%)	No. of iterations
$0 < P \leq 30$	1–2
$30 < P \leq 50$	2–3
$50 < P < 70$	3–4
$P \geq 70$	4–6

6 Conclusion

In this paper, a novel GEP-based classifier has been proposed for the impulse noise detection. High sensitivity, specificity, and AUC of 77.91, 92.51, and 0.8466 %, respectively, imply its robust behavior. To evaluate the performance of the proposed method, we performed extensive simulations on standard test images and compared the results with existing state-of-art methods. Experimental results indicate that the proposed scheme performs better, compared to existing approaches at the high noise densities.

References

1. Luo, W.: An efficient detail-preserving approach for removing impulse noise in images. IEEE Signal Process. Lett. **13**(7), 413–416 (2006)
2. Bhadouria, V.S., Ghoshal, D., Siddiqi, A.H.: A new approach for high density saturated impulse noise removal using decision-based coupled window median filter. Signal Image Video Process. **8**(1), 71–84 (2014)
3. Esakkirajan, S., Veerakumar, T., Subramanyam, A., 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)
4. Nair, M.S., Raju, G.: A new fuzzy-based decision algorithm for high-density impulse noise removal. Signal Image Video Process. **6**(4), 579–595 (2012)
5. Zhu, Z., Zhang, X., Wan, X., Wang, Q.: A random-valued impulse noise removal algorithm with local deviation index and edge-preserving regularization. Signal Image Video Process. pp 1–8 (2013). doi:10.1007/s11760-013-0426-5
6. Wu, J., Tang, C.: Random-valued impulse noise removal using fuzzy weighted non-local means. Signal, Image Video Process. **8**(2), 349–355 (2014)
7. Chen, T., Wu, H.R.: Adaptive impulse detection using center-weighted median filters. IEEE Signal Process. Lett. **8**(1), 1–3 (2001)
8. Aizenberg, I., Butakoff, C.: Effective impulse detector based on rank-order criteria. IEEE Signal Process. Lett. **11**(3), 363–366 (2004)
9. Fischer, V., Lukac, R., Martin, K.: Cost-effective video filtering solution for real-time vision systems. EURASIP J. Appl. Signal Process. **13**, 2026–2042 (2005)
10. Lien, C.Y., Huang, C.C., Chen, P.Y., Lin, Y.F.: An efficient denoising architecture for removal of impulse noise in images. IEEE Trans. Comput. **62**(4), 631–643 (2013)
11. Xiong, B., Yin, Z.: A universal denoising framework with a new impulse detector and nonlocal means. IEEE Trans. Image Process. **21**(4), 1663–1675 (2012)
12. Ferreira, C.: Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst. **13**(2), 87–129 (2001)
13. Zhou, C., Xiao, W., Tirpak, T., Nelson, P.: Evolving accurate and compact classification rules with gene expression programming. IEEE Trans. Evol. Comput. **7**(6), 519–531 (2003)
14. Sermpinis, G., Laws, J., Karathanasopoulos, A., Dunis, C.L.: Forecasting and trading the EUR/USD exchange rate with gene expression and psi sigma neural networks. Expert Syst. Appl. **39**(10), 8865–8877 (2012)
15. Fawcett, T.: ROC graphs: notes and practical considerations for researchers. Technical. report, HP Laboratories, Palo Alto, USA (2004)
16. Forouzan, A., Araabi, B.: Iterative median filtering for restoration of images with impulsive noise. In: Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems, vol 1, 232–235 (2003)
17. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, Cambridge. MIT Press, Cambridge (1992)
18. Ferreira, C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence (Studies in Computational Intelligence). Springer, New York (2006)