Knowl Inf Syst (2012) 32:505–526
DOI 10.1007/s10115-011-0456-7

REGULAR PAPER

Impulse noise filtering based on noise-free pixels using genetic programming

Abdul Majid · Choong-Hwan Lee · Muhammad Tariq Mahmood · Tae-Sun Choi

Received: 3 November 2010 / Revised: 21 June 2011 / Accepted: 22 October 2011 /
Published online: 12 November 2011

Abstract Generally, the impulse noise filtering schemes use all pixels within a neighborhood and increase the size of neighborhood with the increase in noise density. However, the estimate from all pixels within neighborhood may not be accurate. Moreover, the larger window may remove edges and fine details as well. In contrast, we propose a novel impulse noise removal scheme that emphasizes on few noise-free pixels and small neighborhood. The proposed scheme searches noise-free pixels within a small neighborhood. If at least three pixels are not found, then the noisy pixel is left unchanged in current iteration. This iterative process continues until all noisy pixels are replaced with estimated values. In order to estimate the optimal value of the noisy pixel, genetic programming-based estimator is developed. The estimator (function) is composed of useful pixel information and arithmetic functions. Experimental results show that the proposed scheme is capable of removing impulse noise effectively while preserving the fine image details. Especially, our approach has shown effectiveness against high impulse noise density.

A. Majid (✉)
Department of Information and Computer Sciences,
Pakistan Institute of Engineering and Applied Sciences, Nilore, Islamabad, Pakistan
e-mail: abdulmajiid@pieas.edu.pk

C.-H. Lee
Visual Information Processing Laboratory, Digital Aria Co.,
Ltd., 6, Jungja-dong, Bundang-gu, Seongnam-si 463-811, Gyenggi-do, Korea
e-mail: chlee@digitalaria.com

M. T. Mahmood
School of Computer Science and Engineering,
Korea University of Technology and Education, 1600 Chungjeolno,
Byeogchunmyun, Cheonan 330-708, Chungnam, Korea
e-mail: tariq@kut.ac.kr

A. Majid · T.-S. Choi
School of Information and Mechatronics, Gwangju Institute of Science and Technology,
261 Cheomdan Gwagiro, Buk-Gu, Gwangju 500-712, Korea
e-mail: tschoi@gist.ac.kr

Keywords Impulse noise · Genetic programming · Image restoration · Noise detection · Noise filtering

1 Introduction

Digital images, during acquisition or transmission, are often corrupted with impulse noise. Restoring noise-free images is carried out as a preprocessing task in a wide range of imaging applications including medical and astronomical imaging [1]. The impulsive noise, generally, exhibits two major properties: (1) Certain percentage of image pixels is corrupted with noise and (2) The intensity of the corrupted pixels is largely different from the noise-free pixels. An image X corrupted with impulse noise can be described as follows:

$$x_{i,j} = \begin{cases} n_{i,j} \text{ for } p \\ f_{i,j} \text{ for } 1 - p \end{cases} \tag{1}$$

where $x_{i,j}$ is the noisy image pixel, $f_{i,j}$ denotes a noise-free image pixel, $n_{i,j} \in [I_{\min}, I_{\max}]$ is the noisy impulse at the location (i, j), and p is the probability of pixels contaminated with noise. In literature, commonly two impulse noise models namely salt-and-pepper noise and random-valued noise are used. The noisy pixels of image, which are contaminated with salt-and-pepper noise, have two values—the minimum and the maximum values within the dynamic range. In case of salt-and-pepper noise model, the noisy impulse $n_{i,j} \in [I_{\min}, I_{\max}]$ has one of two values: minimum $I_{\min}$ or maximum $I_{\max}$. To remove this kind of noise is simple because the noisy pixels can be easily detected. However, noisy pixels of image, corrupted with random-valued noise, have any value from the dynamic range i.e., $n_{i,j} \in \{I_1, I_2, \ldots, I_K\}$. With the increase in noise density, in the image, the numbers of noisy pixels are increased. If the numbers of noisy pixels are greater than noise-free pixels, then noise filtering becomes crucial.

In literature, conventional linear filtering techniques have been used to suppress noise impulses. However, due to the non-linear nature of impulsive noise, linear filtering techniques are unable to suppress it properly [1]. To address this problem, many non-linear and adaptive filters are proposed. Among these filtering techniques, standard median filter and its variants are considered more robust and effective [2,3]. These techniques replace every pixel with the median value computed over a small window without considering the pixel corrupted or noise-free. Consequently, the important image details such as texture information and edges are lost. To overcome this problem, switching concept has been introduced [4–7]. In the switching concept, first step is to detect the noisy pixels. In noise detection techniques, noisy pixels are determined using some criterion that is based on the similarity or difference to the central pixel within neighborhood. Absolute difference in the median or mean of the neighboring pixels is the simplest noise detection criterion. If this difference is greater than a certain threshold value, then the central pixel is declared to be corrupted with noise. For accurate detection of noisy pixel image statistics [8,9], more sophisticated approaches like genetic programming [10], neural network [11], fuzzy logic [12–14], genetic algorithm [15], and self-organizing map [16] have also been proposed. The main difference between the current work and the proposed work in [10] is the utilization of genetic programming(GP) for different purposes in filtering process. In [10], GP technique is proposed to develop noise detector and the noise pixel estimation is carried out using a conventional α-trimmed mean filter. However, in our work, the noise estimation is carried out using GP-based developed filter.

Once detection is performed, noisy pixels are filtered in the second stage. Among the variety of filtering techniques, adaptive median filter (AMF) has gained considerable attention [17]. AMF algorithm is based on data-dependent varying window size. Its performance is satisfactory at low noise density. However, AMF is computationally expensive for high noise density. In order to minimize cost, progressive switching median filter (PSMF) has been developed to suppress salt-and-pepper noise [4]. In this scheme, the noise impulse is detected and the noisy pixel is filtered progressively. Decision-based algorithm (DBA) has also been proposed to suppress the impulse noise [18]. This algorithm is effective to remove the salt-and-pepper noise. However, in case of random-valued noise, it performs poorly. Recently, new impulse detection and filtering (NIDF) algorithm is proposed [6] that uses Laplacian masks to detect noisy pixels.

In this work, we compare the performance of the proposed scheme with four well-known impulse noise removal algorithms proposed in [4,6,17,18]. Mostly, these conventional approaches estimate noisy pixels by taking into account all pixels within the neighborhood. In other words, these estimates are taken from noise-free and noisy pixels. In case of high-density noise, the number of noisy pixels within the neighborhood is increased and accurate estimate becomes more difficult. Therefore, the performance of the conventional approaches is declined remarkably. Some approaches enlarge the window size to deal with high-density noise. However, a larger window size is more likely to remove edges and image fine details. As a result, quality of the restored images is degraded. In contrast to these approaches, we propose a scheme that emphasizes on noise-free pixels and small neighborhood.

In the proposed scheme, first, noisy pixels are detected using directional derivative. The estimate of the detected impulse is based on noise-free pixels within a small neighborhood. If there are not at least three noise-free pixels in the neighborhood, then the noisy pixel is left unchanged and is estimated in a later iteration. This iterative process continues iteratively until all noisy pixels are estimated. To estimate the noisy pixel, we develop a generalized estimator through GP. It has been effectively used in developing mathematical models in the applications of pattern recognition, information systems, and computer vision [19–23]. GP approach efficiently searches solution in the defined problem space. This evolutionary optimization approach is based on the principles of natural selection and recombination to search all possible solutions [24]. During GP evolution cycle, the most optimal solution is developed in the form of a generic estimator function. This estimator optimal combines the useful information of local clean pixels along with arithmetic operators. In the current work, we summarize the main contributions as:

(a) We developed impulse noise removal scheme that emphasizes on clean pixels and small local neighborhood, in contrast to the estimate noisy pixel from all noisy/noise-free pixels and large neighborhoods.
(b) GP-based generic estimator is developed to estimate the value of noisy pixels.

The performance of the proposed scheme is investigated using several standard images. Our comparative analysis highlights the effectiveness of the proposed scheme to remove impulse noise while preserving the fine image details. The organization of the rest of the paper is as follows. In Sect. 2, GP-based noise removal proposed scheme is explained in detail. Experimental results and comparative analysis are presented in Sect. 3. Finally, Sect. 4 concludes this study.

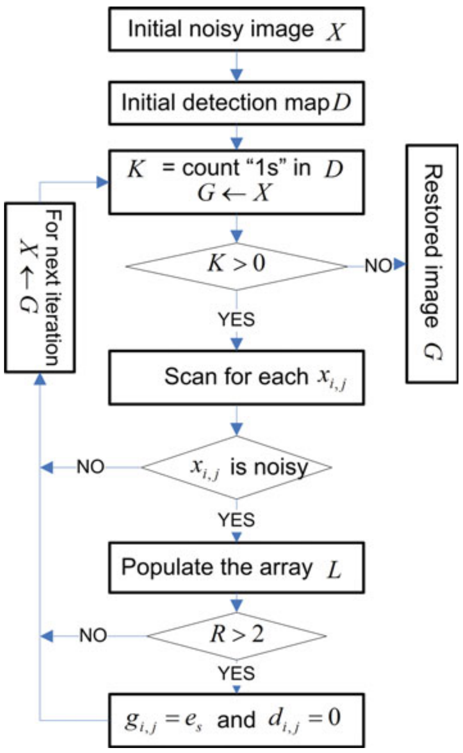

Fig. 1 Block diagram of the proposed impulse noise removal scheme

2 Proposed scheme

The block diagram of the proposed scheme is shown in Fig. 1. The proposed approach is divided into two main stages: noise detection and filtering. First, noisy pixels are detected, and then, values of detected pixels are estimated through GP estimator incorporating noise-free pixels. This scheme iteratively cleans the corrupted image until all noisy pixels are replaced with the estimated values.

2.1 Noise detection

In proposed scheme, the first step is to construct the detection map D from the noisy image. In case of salt-and-pepper noise, the maximum and the minimum intensity values of the image dynamic range $[I_{\min}, I_{\max}]$ can be used to detect noisy pixels. Thus, in a 8 bit gray scale image, if the pixel value is 0 or 255, then, most probably, it is corrupted with salt-and-pepper noise. In noise-free images, only a very small number of pixels can have these two values. A detection map is constructed as:

$$d_{i,j} = \begin{cases} 1 : & x_{i,j} = I_{\max} \\ 1 : & x_{i,j} = I_{\min} \\ 0 : & \text{otherwise} \end{cases} \tag{2}$$

The values of "1" and "0" represent the noisy and the noise-free pixels in the detection map. In the work, this approach proves to be very effective for salt-and-pepper noise. However, in case of random-valued noise, the dynamic range of noise consists of more than one

Fig. 2 Four 5×5 convolution kernels

0	0	0	0	0
0	0	0	0	0
-1	-1	4	-1	-1
0	0	0	0	0
0	0	0	0	0

0	0	-1	0	0
0	0	-1	0	0
0	0	4	0	0
0	0	-1	0	0
0	0	-1	0	0

0	0	0	0	-1
0	0	0	-1	0
0	0	4	0	0
0	-1	0	0	0
-1	0	0	0	0

-1	0	0	0	0
0	-1	0	0	0
0	0	4	0	0
0	0	0	-1	0
0	0	0	0	-1

minimum and maximum intensity values, and the detection approach mention by Eq. (2) may not be useful. Therefore, the corrupted pixels by the random-valued noise are detected using directional derivative [6,25]. Commonly, the second derivative-based methods perform better than gradient-based approaches. In our approach, we use four directional Laplacian masks for noise detection. The pixels of noisy image X are convolved with a set of four directional Laplacian masks shown in Fig. 2. These derivative-based four kernels are sensitive to edges in different orientations [25]. For impulse noise detection, the minimum absolute value $m_{i,j}$ is computed as follows:

$$m_{i,j} = \min \left\{ \left| x_{i,j} * k_l \right| |: l = 1 \text{ to } 4 \right\} \tag{3}$$

where k_l is the lth kernel and $*$ denotes a convolution operation. The absolute numerical value of $m_{i,j}$ is compared with a threshold T to determine whether a pixel is corrupted with noise. If $d_{i,j} = 1$, the pixel $x_{i,j}$ is decided as a noisy; otherwise, the pixel $x_{i,j}$ is declared as the noise-free. Thus, the noise detection map D is computed as:

$$d_{i,j} = \begin{cases} 1, & \text{if } m_{i,j} > T \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where T is the threshold and its value may affect the selection of noisy pixel. For simplicity, we adopted the same heuristic-based selection procedure as reported in [25]. During simulation, we found suitable threshold range 70–90 empirically, which provides precise detection map.

2.2 Noise filtering

In this step, noise-free image G is restored from its noisy version X. Let W_{ij} be a window of size $w \times w$ centered at (i, j), i.e.,

$$W_{i,j} = \{(s,t) : |s - i| \leq w \wedge |t - j| \leq w\} \tag{5}$$

Once, we compute the detection map from the noisy image, we apply this window on each pixel location (i, j) of both noisy image X and detection map. We use a small neighborhood of size 3×3. We prefer to use small window size because the larger window size may not be too efficient and effective. This is because correlation between pixels decreases as the pixels are separated apart. Further, the larger window size will also remove the edges and fine details. By applying this window on noisy image and detection map, patches obtained

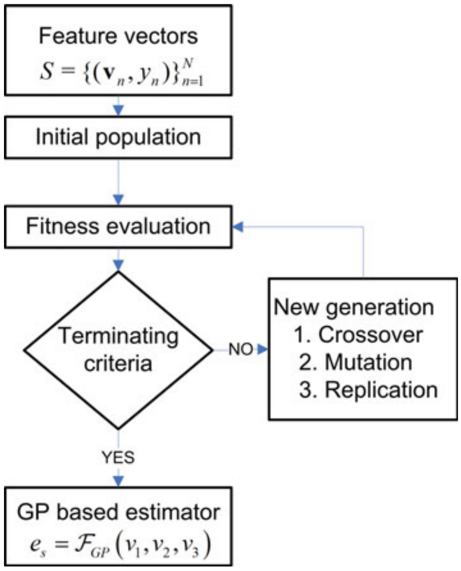

Fig. 3 Block diagram of developing GP-based estimator

Table 1 Selection of three features from sorted noise-free array

Length of array	$L(1)$	$L(2)$	$L(3)$	$L(4)$	$L(5)$	$L(6)$	$L(7)$	$L(8)$
Case 1: $R = 3$	1	1	1	–	–	–	–	–
Case 2: $R = 4$	0	1	1	1	–	–	–	–
Case 3: $R = 5$	0	1	1	1	0	–	–	–
Case 4: $R = 6$	0	0	1	1	1	0	–	–
Case 5: $R = 7$	0	0	1	1	1	0	0	–
Case 6: $R = 8$	0	0	0	1	1	1	0	0

are as follows:

$$W_{i,j}^{x} = \begin{bmatrix} x_{i-1,j-1} & x_{i,j-1} & x_{i+1,j-1} \\ x_{i-1,j} & x_{i,j} & x_{i+1,j} \\ x_{i-1,j+1} & x_{i,j+1} & x_{i+1,j+1} \end{bmatrix} \tag{6}$$

$$W_{i,j}^{d} = \begin{bmatrix} d_{i-1,j-1} & d_{i,j-1} & d_{i+1,j-1} \\ d_{i-1,j} & d_{i,j} & d_{i+1,j} \\ d_{i-1,j+1} & d_{i,j+1} & d_{i+1,j+1} \end{bmatrix} \tag{7}$$

If $d_{i,j} = 1$, then the pixel $x_{i,j}$ is noisy candidate; otherwise, the pixel is noise-free. In case, pixel is noisy then its correct value is to be estimated. To collect the information about the noise-free pixels only, we further examine the remaining eight pixels in the window $W_{i,j}^{x}$ using the detection window $W_{i,j}^{d}$. An array $L(r)$ is populated with noise-free pixels. The length of this array $L(r)$ varies from zero to eight depending upon noise density within the patch. The minimum length zero shows that all pixels are noisy in this window whereas maximum value eight indicates that all eight pixels are noise-free pixels. To filter the noisy pixel, we emphasize noise-free pixels and put additional constraint of minimum three noise-free pixels i.e., length$(L) > 2$. In other words, there must be at least three noise-free pixels in the window to estimate the value of noisy pixel. If this condition is not satisfied, then the

Table 2 Unary and binary functions used in the proposed GP-based scheme

Unary functions	Description	Binary functions	Description
Sin (v_i)	Returns the sin of the input	Plus (v_i, v_j)	Adds two inputs
cos (v_i)	Returns the cos of the input	Minus (v_i, v_j)	Subtracts the second input from the first input
Sqrt (v_i)	Computes square root of the given input	Divide (v_i, v_j)	Divides the first input by the second input
Exp (v_i)	Returns the exponential value of the input	Times (v_i, v_j)	Multiplies first input with the second input
Log (v_i)	Returns the log value of the input	Power (v_i, v_j)	Raises the first input to the power of the second input

Table 3 GP parameters setting under minimum MSE fitness criterion

Parameters	Set values
Terminals set	Feature vector $\mathbf{v} = (v_1, v_2, v_3)$ along with random constants in the range $[-1, 1]$
Functions set	Lest in Table 1
Fitness criterion and selection method	Minimization of MSE & generational, respectively
Pop. size and generations	50 & 500, respectively
Pop. initialization and sampling	Ramped half and half and tournament, respectively
Expected offspring	rank85
Operators probabilities	Variable crossover/mutation ratio
Survival criterion	Keep the best individual

noisy pixel is left unchanged. Thus, the replacement of noisy pixel with the estimated value is carried out as:

$$g_{i,j} = \begin{cases} e_s & \text{if } d_{i,j} = 1 \text{ and } R > 2 \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{8}$$

where e_s is the estimated value for the noisy pixel and $R = length(L)$ is the length of the array L. In our case, we develop a GP-based function $\mathcal{F}_{\mathcal{GP}}$ to estimate the correct pixel value (Sect. 2.3). If the noisy pixel is estimated, then the detection map D is also updated using the following function as:

$$d_{i,j} = \begin{cases} 0 & \text{if } d_{i,j} = 1 \text{ and } R > 2 \\ d_{i,j}, & \text{otherwise} \end{cases} \tag{9}$$

This process is furnished for each noisy pixel in image X. As a result, a refined image G and an updated detection map D are obtained. Due to this constraint, many noisy pixels in X are left unchanged. Particularly, when noise density is very high, more noisy pixels will not satisfy the condition. This leads to iterate the filtering process until all noisy pixels are corrected. The number of iterations depends upon the noise density and the constraint imposed.

2.3 GP-based estimator

Block diagram for the development of GP estimator is shown in Fig. 3. In this process, the first step is to generate a set of feature vectors. The proposed GP-based function uses feature

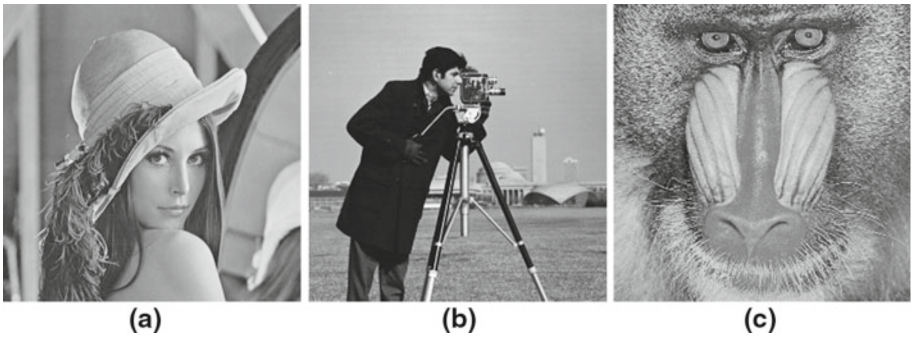

Fig. 4 Standard test images: **a** LENA, **b** CAMERAMAN, and **c** BABOON

vector to compute the estimated pixel value. We use the constructed array $L(r)$ for feature selection. This array contains only noise-free pixels as discussed in Sect. 2.2. The length of array $L(r)$ varies from zero to eight. First, elements of this array are sorted in the ascending order. We compute the feature vector that contains three elements v_1, v_2, v_3 as:

$$v_{l+1} = L\left(\left\lfloor \frac{R}{2} \right\rfloor + l\right), \quad l = 0, 1, 2. \tag{10}$$

If the length of the sorted array is odd number, then this feature vector contains the median and its neighboring pixels. Thus, these features contain characteristics of ordered statistics. Experimentally, we found that more than three features have very slight effect on the GP-based estimator. Therefore, we use feature vector consisting of only three pixels. For lucid explanation of the feature selection, Table 1 shows all possible cases. Selected elements are shown by “1s” and “0s” which indicate non-selected element in different cases.

For the development of GP-based estimator, a set of N training data points is constructed using a standard test image. The standard test image is corrupted with 50% salt-and-pepper noise, and a feature vector $\mathbf{v} = (v_1, v_2, v_3)$ is formed for those pixels which are noisy and have more than two noise-free pixels in the neighborhood. It is possible to pick corresponding target value y for each feature vector from the original noise-free image. Thus, a training data set $S = \{(\mathbf{v}_n, y_n)\}_{n=1}^{N}$ is obtained for the development of GP estimator.

The main goal of GP module is to estimate optimal model $\mathcal{F}_{\mathcal{GP}} : \mathbf{v} \to y$ from the training examples. The numerical function $\mathcal{F}_{\mathcal{GP}}$ is represented in a tree-like data structure with selected variables, random constants, and arithmetic operators. Initial population of candidate solutions is created randomly as a possible solution space. The fitness score of each candidate is computed in terms of mean square error (MSE). On the basis of the survival of fittest rule, the best candidates are ranked and then selected for the creation of the next generation. During evolution cycle, new generation has higher average fitness score. In this way, the solution space is refined and the best solution converges to the optimal/near-optimal solution. The developed function optimally combines the useful local information along with some other parameters. After the development, this function is ready to estimate the optimal value of noisy images. Now, we will explain briefly the main steps involved in developing GP function.

2.3.1 GP parameters

GP parameters consist of terminals and non-terminal parameters. A list of these functions is provided in Table 2. Terminal parameters (variables and random constants) contain the

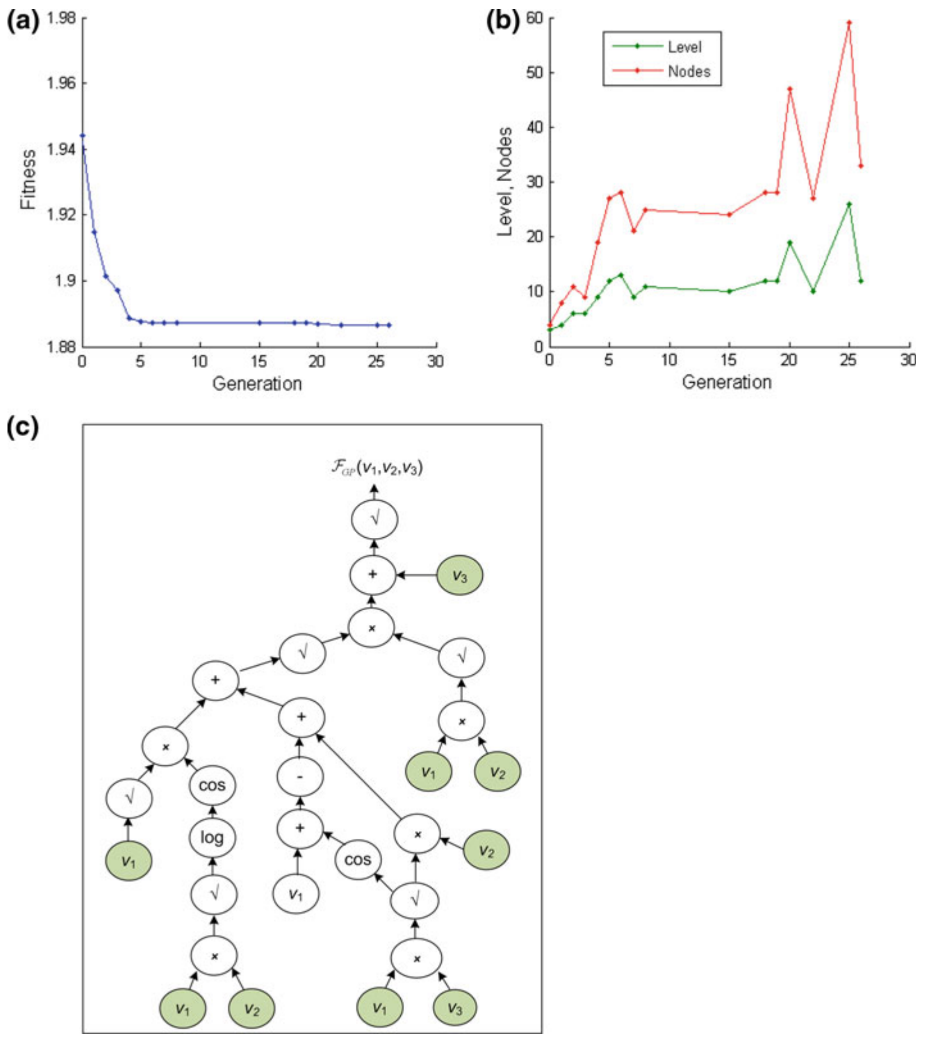

Fig. 5 Accuracy versus complexity. **a** Improvement in accuracy/fitness score exhibited by the best individual in each generation. **b** Increase in complexity in terms of level of tree depth and number of nodes. **c** Graphical representation of the GP-based developed filter

useful input information. Currently, for each pixel, a feature vector **v** is given. This set of gray values is provided as variable terminals for each noisy pixel. Moreover, some random numbers in the range $[-1, 1]$ are generated from uniform distribution.

2.3.2 Initial population

Population initialization is the first step to start GP cycle. Each individual is represented in the form of randomly generated GP tree that is constituent of variables, random constants, and mathematical functions. The initial population is created using ramped half-and-half method. This method requires that half of the randomly generated trees must be generated

Table 4 Performance comparison of the proposed method with different algorithms in terms of PSNR (db) for Lena image corrupted with salt-and-pepper (SPN) and random-valued (RVN) noise

Noise density	Type	SMF	PSMF	DBA	NIDF	Proposed
10%	SPN	33.74	38.31	41.54	37.45	42.89
	RVN	33.82	35.10	19.30	35.11	35.64
20%	SPN	29.47	34.79	37.33	34.04	39.18
	RVN	31.65	31.68	16.26	32.25	32.99
30%	SPN	24.03	30.74	34.82	31.37	37.03
	RVN	28.35	28.56	14.55	29.87	30.71
40%	SPN	19.15	26.09	32.52	29.55	35.02
	RVN	24.67	25.24	13.25	27.53	28.46
50%	SPN	15.37	21.13	30.19	27.42	33.31
	RVN	21.50	22.26	12.31	24.74	25.71
60%	SPN	12.41	12.33	28.19	23.53	31.76
	RVN	19.00	19.69	11.51	22.15	22.85
70%	SPN	10.00	9.95	25.66	17.84	29.86
	RVN	16.78	17.24	10.83	19.13	19.69
80%	SPN	8.15	8.12	23.30	16.50	27.74
	RVN	15.01	15.24	10.25	16.60	17.28
90%	SPN	6.66	6.64	19.67	10.25	24.16
	RVN	13.51	13.52	9.73	14.57	15.35
Mean	SPN	17.66	20.90	30.36	25.33	33.44
	RVN	22.70	23.17	13.11	24.66	25.41
Overall mean		20.18	22.03	21.73	25.00	29.40

by a random process that ensures all branches of the maximum initial depth. The remaining randomly generated trees require branches whose lengths do not exceed this depth. These constraints have been found to generate a good initial sample of trees. Now, for each tree depth level, half of the individuals are initialized using Grow method and the other half individuals using Full method.

2.3.3 Fitness criterion

In second step, the fitness of each individual is measured through a suitable fitness function. This plays an important role to obtain the optimal solution within a large search space. The performance of each individual is assessed by this function. We use mean square error (MSE) as fitness criteria. The value of fit(Q^j) measures how effectively jth individual minimizes the MSE:

$$\text{fit}(Q^j) = \frac{1}{N}\sum_{n=1}^{N}(Q^j(\mathbf{v}_n) - y_n)^2, \quad n = 1, 2, \ldots, N. \tag{11}$$

where y_n is the nth target values taken from the original noise-free image corresponding to the nth training pattern $(\mathbf{v}_n, y_n)$.

Table 5 Performance comparison of the proposed method with different algorithms in terms of PSNR (db) for the Cameraman image corrupted with salt-and-pepper (SPN) and random-valued (RVN) noise

Noise density	Type	SMF	PSMF	DBA	NIDF	Proposed
10%	SPN	34.63	36.76	41.70	34.55	43.47
	RVN	34.36	33.53	18.42	34.02	34.53
20%	SPN	29.22	33.17	37.19	32.15	39.30
	RVN	30.77	29.87	15.42	30.91	31.16
30%	SPN	23.67	29.92	34.34	30.52	36.98
	RVN	26.53	26.68	13.68	28.82	29.19
40%	SPN	18.74	26.19	31.67	28.66	34.90
	RVN	22.58	23.34	12.46	25.92	26.67
50%	SPN	14.92	21.39	29.41	26.30	33.11
	RVN	19.45	20.23	11.46	22.51	23.50
60%	SPN	11.94	11.93	26.93	22.56	31.16
	RVN	16.75	17.29	10.66	18.81	20.11
70%	SPN	9.67	9.66	24.55	17.46	28.97
	RVN	14.74	15.01	9.98	16.11	17.13
80%	SPN	7.81	7.80	22.06	12.37	26.56
	RVN	13.15	13.26	9.42	14.13	14.93
90%	SPN	6.25	6.25	18.37	7.94	22.46
	RVN	11.85	11.85	8.90	12.57	13.29
Mean	SPN	17.43	20.34	29.58	23.61	32.99
	RVN	21.13	21.23	12.27	22.64	23.39
Overall mean		19.28	20.78	19.7	23.13	28.19

2.3.4 Creation of new population

To create new population, based on fitness scores, GP individuals are selected from the whole population. Crossover, mutation, and replication operators are applied. Crossover creates offspring by exchanging genetic material between two individual parents. It tries to mimic recombination and sexual reproduction. For crossover, a method namely tournament selection is used. Tournament selection works by first selecting trees at random from the current generation. Two trees with the highest fitness values exchanged their sub-trees resulting in two new possible solutions. Crossover helps in converging to optimal/near-optimal solution. In mutation, a small part of an individual is changed that often brings diversity in the solution space. In this case, an individual is selected and a mutation point picked (a sub-tree of the individual). The sub-tree blow the mutation point is replaced with a randomly generated sub-tree. A variable ratio of crossover to mutation is adapted. Replication is the copying of an individual into the next generation without any change. The probability of jth individual Q_q^j being selected is given by:

$$\Pr(Q_q^j) = \frac{\mathrm{fit}(Q_q^j)}{\sum_{q=1}^{N_q} \mathrm{fit}(Q_q^j)}. \tag{12}$$

where $\sum_{q=1}^{N_q} \mathrm{fit}(Q_q^j)$ is the accumulative fitness of the population of size N_q.

Table 6 Performance comparison of the proposed method with different algorithms in terms of PSNR (db) for the Baboon image corrupted with salt-and-pepper (SPN) and random-valued (RVN) noise

Noise density	Type	SMF	PSMF	DBA	NIDF	Proposed
10%	SPN	28.99	31.93	37.12	31.66	38.19
	RVN	29.00	30.50	19.66	30.59	30.37
20%	SPN	26.78	30.25	33.05	29.63	34.57
	RVN	27.58	28.30	16.70	28.20	28.14
30%	SPN	22.88	27.68	30.33	27.89	32.39
	RVN	25.90	26.29	14.98	26.50	26.58
40%	SPN	18.78	24.37	28.14	26.37	30.63
	RVN	23.77	24.14	13.72	24.87	25.05
50%	SPN	15.25	20.56	26.11	24.72	29.04
	RVN	21.60	21.99	12.75	23.13	23.47
60%	SPN	12.44	12.47	24.24	22.08	27.30
	RVN	19.41	19.77	11.94	21.19	21.65
70%	SPN	10.15	10.15	22.48	17.73	25.63
	RVN	17.56	17.84	11.27	19.21	19.75
80%	SPN	8.29	8.28	20.53	12.78	23.66
	RVN	15.85	15.98	10.68	17.33	18.03
90%	SPN	6.77	6.77	18.45	8.65	21.14
	RVN	14.43	14.41	10.16	15.60	16.44
Mean	SPN	16.70	19.16	24.83	22.39	29.17
	RVN	21.68	22.14	13.54	22.96	23.28
Overall mean		19.19	20.65	19.18	22.67	26.22

2.3.5 Termination criterion

The algorithm is stopped if the number of generations reaches the maximum limit or fitness value (MSE) approaches the minimum set value. At the end, the best solution in the form of optimal composite function is developed. GP simulations and experiments are carried out using GPLAB [26] toolbox on a PC equipped with processor Intel Pentium IV 2.88-GHz and 1 GB RAM. In order to represent a possible solution in the form of an optimal function, all necessary parameters including arithmetic functions, variables, and constants are provided in Table 3.

3 Results and discussion

Several experiments have been carried out to analyze the performance of proposed scheme. Experiments are reported using three commonly used standard images namely LENA, CAMERAMAN, and BABOON. These images are shown in Fig. 4. The test images were corrupted by two types of noises: salt-and-pepper noise and random-valued noise. The level of noise density is varied from 10 to 90%. The experimental results obtained through the proposed scheme and several well-known filtering schemes SMF [27], PSMF [4], DBA [28], and NIDF [6] have been compared.

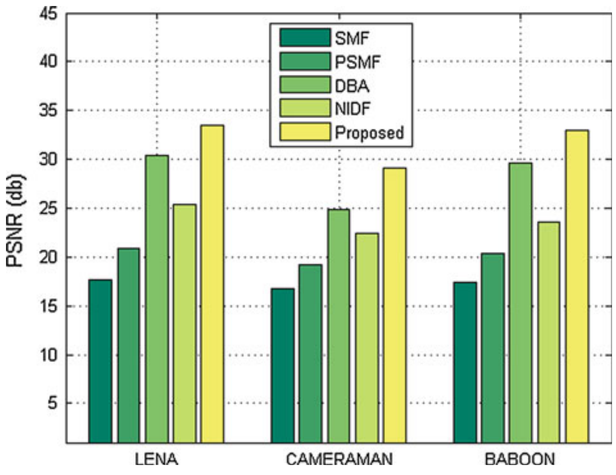

Fig. 6 Overall performance comparison using different images

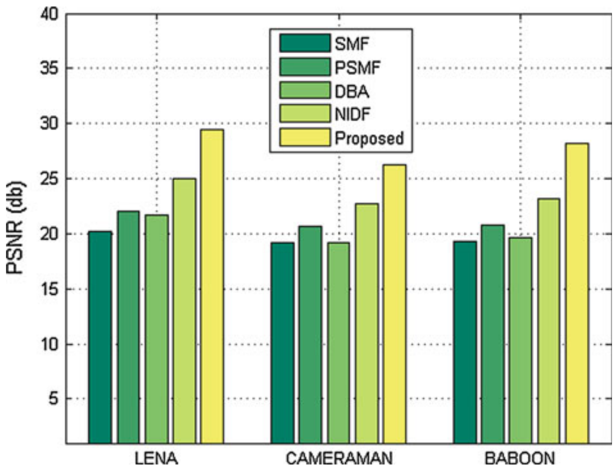

Fig. 7 Overall performance comparison using different images corrupted with salt-and-pepper noise

3.1 Experimental setup

To develop GP-based filter, the training data have been prepared from the LENA image. The original LENA image corrupted with salt-and-pepper noise with 50% density. A set of feature vectors of the form $\mathbf{v} = (v_1, v_2, v_3)$ is formed as explained in Sect. 2.2. There are total pixels ($511 \times 511 = 261{,}121$) in LENA image by excluding one pixel from boundary of the image. By adding 50% salt-and-pepper noise, a total of 129,717 pixels have been corrupted out of total pixels 261,121. Among the noisy pixels, 110,986 pixels have satisfied the constraints. From the original LENA image, corresponding true values have been taken. In this way, the training data set consisting of 110,986 has been used to develop the GP estimator.

Figure 5a shows the improvement in the best fit individual in each generation. The complexity is expressed as a function of tree depth level and the number of nodes. During GP evolution, constructive blocks are created that try to minimize the destruction of useful

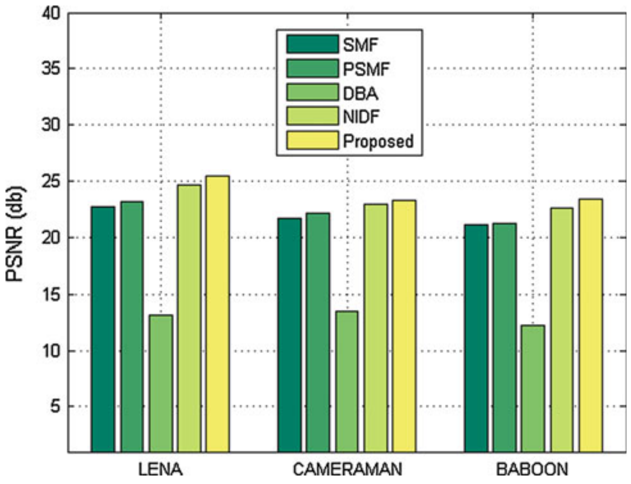

Fig. 8 Overall performance comparison using different images corrupted with random-valued noise

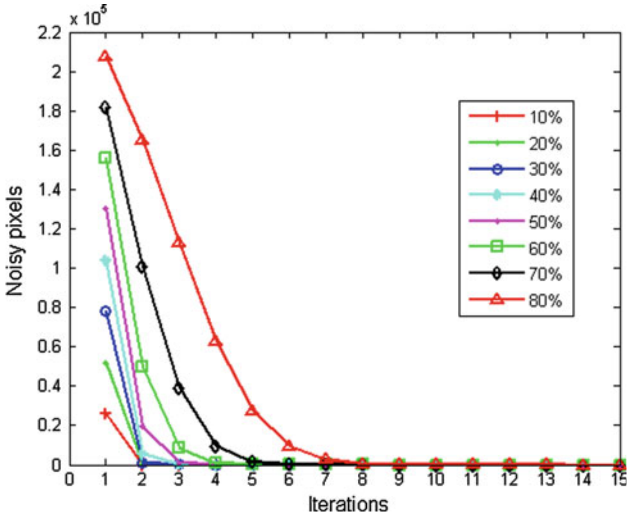

Fig. 9 Number of noisy pixels at each iteration computed for the CAMERAMAN image corrupted with different levels of salt-and-pepper noise

Table 7 Temporal cost comparison using 10 and 90% salt-and-pepper noise

Method/cost	Lena		Baboon		Cameraman	
	10%	90%	10%	90%	10%	90%
SMF	0.5262	0.5052	0.5219	0.5119	0.5123	0.4881
PSMF	1.8488	0.8630	1.8383	0.8704	1.6146	0.8470
DBA	5.9977	5.9998	6.0519	6.0104	6.0136	5.9915
NIDF	2.9094	5.1900	3.0059	5.2488	2.8804	5.1931
Proposed	5.9389	51.2127	6.2709	47.4326	5.8900	53.2468

Table 8 Detailed temporal cost of the proposed approach using different levels of salt-and-pepper noise

Noise density (%)	Lena	Baboon	Cameraman
10	5.9389	6.2709	5.8900
20	10.1998	10.2037	10.1667
30	11.3426	11.3507	11.4423
40	12.5127	15.5479	12.5259
50	16.8946	17.1032	16.8242
60	18.2422	18.0106	21.2084
70	22.8480	25.7079	25.8530
80	34.3977	34.5850	30.9486
90	51.2127	47.4326	53.2468

building blocks [29]. As a result, in several regions of Fig. 5b, the size of GP individual grows exponentially without appreciable improvement in performance curve of the best individual. This is due to the bloating phenomenon occurring during GP cycle [30]. Many branches do not contribute in improving its performance. Therefore, the best genome's total number of nodes increases, and its average tree depth becomes very large. Therefore, with the increase in complexity, performance curve of the best individual approaches toward the optimal solution. During the training phase, several GP runs were carried out and the best filter function is reported. Each GP simulation took considerable computational time that depends on several input parameters i.e., input data size, population size, and number of generations. At the end of GP simulation, the developed filter is given, in prefix form as well as in algebraic form, as follows:

$$\begin{aligned}\mathcal{F}_{\mathrm{GP}}(v_1, v_2, v_3) = {} & \mathrm{sqrt}(\mathrm{plus}(v_3, \mathrm{times}(\mathrm{sqrt}(\mathrm{times}(v_2, v_1)), \mathrm{sqrt}(\mathrm{plus}(\mathrm{minus}(\\ & \mathrm{times}(\cos(\log(\mathrm{sqrt}(\mathrm{times}(v_2, v_1)))), \mathrm{sqrt}(v_1)), \mathrm{plus}(v_1, \cos(\\ & \mathrm{sqrt}(\mathrm{times}(v_1, v_3))))), \mathrm{times}(\mathrm{sqrt}(\mathrm{times}(v_1, v_3)), v_2)))))). \qquad (13)\end{aligned}$$

$$\begin{aligned}&\mathcal{F}_{\mathrm{GP}}(v_1, v_2, v_3) \\ &= \sqrt{\left(v_3 + \left\{\sqrt{(v_2 \times v_1)} \times \sqrt{\left[\left\{\left(\cos\left(\log\left(\sqrt{v_2 \times v_1}\right)\right) \times \sqrt{v_1}\right)\right\} - \left(v_1 + \cos\left(\sqrt{v_1 \times v_3}\right)\right) + \left(v_2 \times \sqrt{v_1 \times v_3}\right)\right]}\right\}\right)}. \qquad (14)\end{aligned}$$

The above empirical expression highlights a generic structure of the filter function, which may not be easily understood by human being [30]. However, such empirical function can be easily used by providing input parameters v_1, v_2, and v_3. The tree structure of the generated filter function is demonstrated in Fig. 5c. This graphical representation highlights the functional dependency of the GP-based empirical expression on the input features v_1, v_2, and v_3 along with some selected arithmetic and trigonometric functions.

3.2 Quantitative analysis

To measure the quality of the restored image and compare the results quantitatively, quality measure peak signal-to-noise ratio (PSNR) is used. It can be defined as:

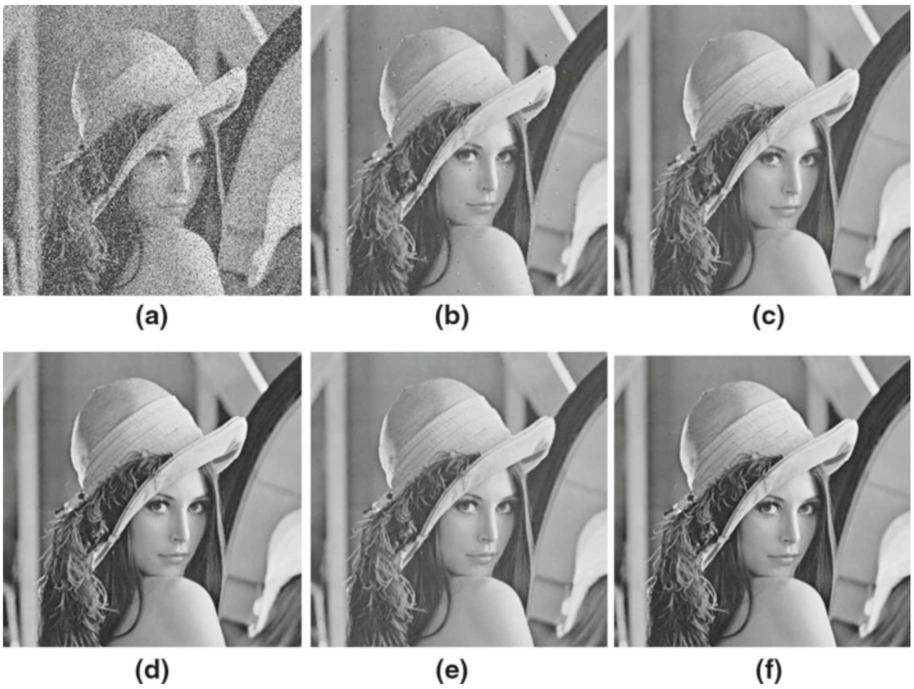

Fig. 10 Performance comparison of image restoration results obtained from the LENA image using different algorithms. **a** Image corrupted with 20% salt-and-pepper noise, **b** SMF (29.47 db), **c** PSMF (34.79 db), **d** DBA (37.33 db), **e** NIDF (34.04 db), and **f** proposed method (39.18 db)

$$\mathrm{PSNR} = 10 \log 10 \frac{(255)^2}{\mathrm{MSE}} \tag{15}$$

$$\mathrm{MSE} = \frac{1}{\mathrm{MN}} \sum_{i=1}^{M} \sum_{j=1}^{M} (X'_{i,j} - G_{i,j})^2 \tag{16}$$

where $X'_{i,j}$ is the original noise-free image, $G_{i,j}$ is the de-noised image, and M is the number of pixels in rows and columns of the original and de-noised images. Tables 4, 5 and 6 show PSNR values computed through proposed algorithm along with other well-known methods. Table 4 shows the PSNR for LENA image. For salt-and-pepper noise, our scheme has provided the best mean PSNR value 33.44, whereas PSNR for other four algorithms SMF, PSMF, DBA, and NIDF has given 17.66, 20.90, 30.36, and 25.33, respectively. The improved performance of the proposed scheme is also obtained for random-valued noise. The overall performance of the proposed scheme is calculated to be 29.40. Other four schemes gave the overall mean values of 20.18, 22.03, 21.73, and 25.00. These values are considerably lower as compared with the proposed scheme. From Table 5, it can be observed that the proposed scheme provides higher mean value (32.99) as compared with other schemes (17.43, 20.34, 29.58, and 23.61) for salt-and-pepper noise. Our scheme has given the improved performance for random-valued noise as well. Experimental results on high-textured image BABOON are listed in Table 6. In this table, for salt-and-pepper noise, our method gives a considerable margin of improvement (29.17) as compared to other four filtering schemes (16.70, 19.16, 24.83, and 22.39).

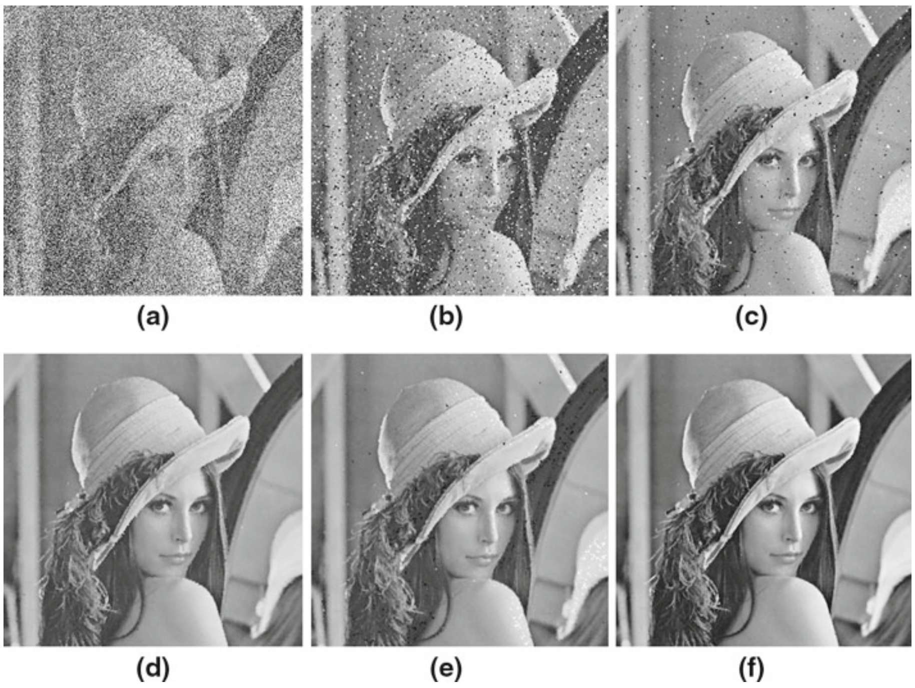

Fig. 11 Performance comparison of image restoration results obtained from the LENA image using different algorithms. **a** Image corrupted with 50% salt-and-pepper noise, **b** SMF (15.37 db), **c** PSMF (21.13 db), **d** DBA (30.19 db), **e** NIDF (27.42 db), and **f** proposed method (33.31 db)

The value of overall performance of the proposed scheme (26.22) is better than other filtering approaches (19.19, 20.65, 19.18, and 22.67). Our scheme provided better result for random-valued noise as well. For lucid comparison, Figs. 6, 7 and 8 show the overall performance for salt-and-pepper-noise, random-valued noise, and including both types of noises using different images. The proposed scheme has outperformed these existing techniques.

While using GP filter for noise filtering, it is found that the number of iterations increases with the increase in noise density. Figure 9 shows this character of the generated filter. It highlights the number of noisy pixels left after each iteration with increasing impulse noise density. The number of iterations also depends on the window size and the minimum length of noise-free array. For experimentation, we have used a fix window of size 3×3 and the minimum length of noise-free array is adjusted to three pixels.

Table 7 shows the comparison of computational time consumed by SMF, PSMF, DBA, and NIDF, and the proposed approach for images is corrupted with 10 and 90% salt-and-pepper noise. It is observed that, for low noise density, the temporal coast of our approach is competitive with the previous schemes. However, In case of high-density noise, our approach is computationally expensive. Further, Table 8 highlights the detailed temporal cost of the proposed approach for test images corrupted with various levels of salt-and-pepper noise. This table shows that the temporal cost increases with the increase in noise density. This is due to the fact that the proposed approach iteratively estimates the noisy pixel using noise-free pixels. More iteration is needed to estimate all noisy pixels for high-density noise.

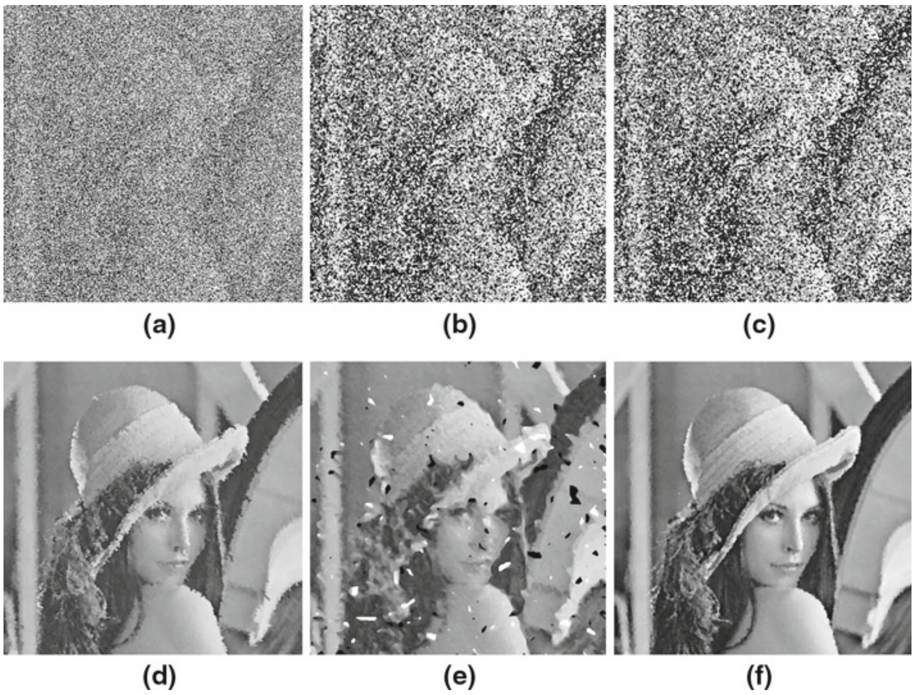

Fig. 12 Performance comparison of image restoration results obtained from the LENA image using different algorithms. **a** Image corrupted with 80% salt-and-pepper noise, **b** SMF (8.15 db), **c** PSMF (8.12 db), **d** DBA (23.30 db), **e** NIDF (16.50 db), and **f** proposed method (27.74 db)

3.3 Qualitative analysis

Figures 10, 11, 12, 13, and 14 show restored images using proposed algorithm and existing approaches (NIDF, SMF, PSMF, and DBA) from LENA, BABOON, and CAMERAMAN images corrupted with different noise densities. The restored images results on low (20%), middle (50%), and high (80%) range of salt-and-pepper noise for LENA images are shown in Figs. 10, 11 and 12. For low category of noisy, among all other restorations methods, our proposed algorithm gives the best quality image. DBA is the nearest competitor to our scheme. Similarly, from Fig. 11, it can be observed that our scheme has restored images of good quality for 50% noise density. For this category of noise, DBA is again the nearest competitor. In case of LENA image corrupted with 80% salt-and-pepper noise, Fig. 12 shows the effect of high noise that suppressed by DBA filter. The distortion effect near the lines and edges of LENA image is clearly visible. On the other hand, it can be observed that the proposed approach has preserved edges and fine details of restored LENA image.

Figure 13 shows the restoration results on different filtering algorithms for BABOON image corrupted with 80% salt-and-pepper noise. For this high-textured image, all other algorithms, except DBA, could not recovered visible BABOON image. Among the existing approaches, DBA is performing better. However, in case of high-density noise, this method could not preserve the fine details near the high-contrast region and the blurring effect is visible. This is because the existing approaches estimate noisy pixels

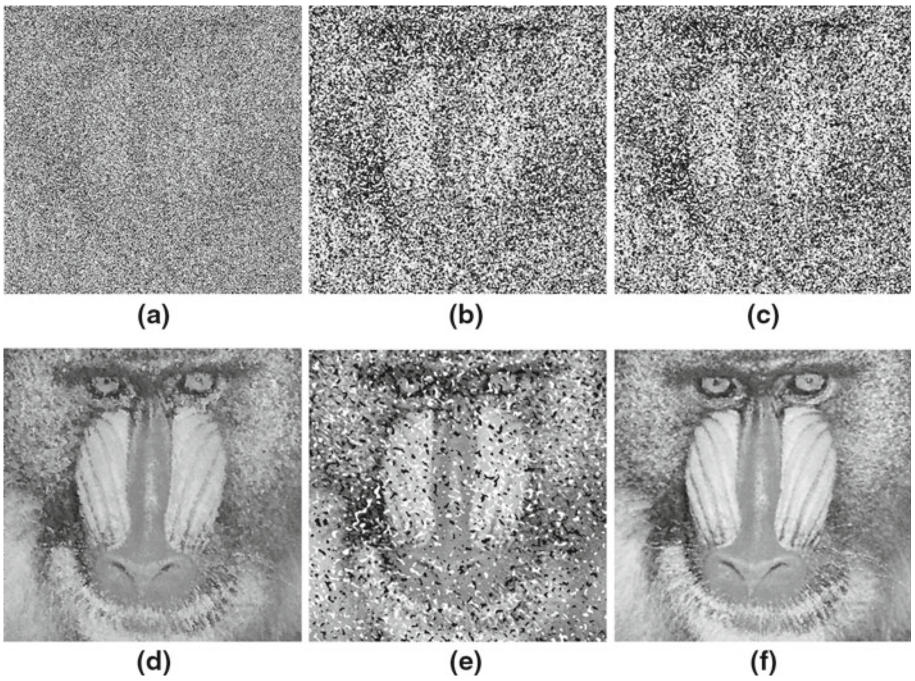

Fig. 13 Performance comparison of image restoration results obtained from the BABOON image using different algorithms. **a** Image corrupted with 80% salt-and-pepper noise, **b** SMF (8.29 db), **c** PSMF (8.28 db), **d** DBA (20.53 db), **e** NIDF (12.78 db), and **f** proposed method (23.66 db)

by taking into account all neighbors (noisy/noise-free pixels) without any distinction. However, our filtering scheme estimates the noisy image by taking only noise-free neighbors. Therefore, the quality of restored BABOON image through our scheme is superior.

Figure 14 shows the restored CAMERAMAN images from corrupted with 30% random-valued noise. In this case, NIDF is the nearest competitor to our scheme, whereas DBA could not yield the same image quality. DBA performed well for salt-and-pepper noise; however, its performance for the random-valued noise is poor. The effect of distortion in restored image is clearly visible. In summary, our proposed approach has successfully suppressed noise and able to preserve fine details as well.

4 Conclusion

In this work, we have proposed a novel impulse noise removal scheme that emphasizes on few noise-free pixels and small neighborhood instead of using all pixels within a neighborhood. This scheme iteratively cleans the corrupted image until all noisy pixels are replaced with the estimated values. To estimate the optimal value of noisy pixel, we have developed GP-based robust estimator that combined the useful local clean pixel and arithmetic functions. The proposed scheme was tested using standard images, and the simulated results are compared with four well-known techniques. Our scheme has provided better performance as compared to existing approaches. Moreover, this scheme is capable to restore the corrupted images

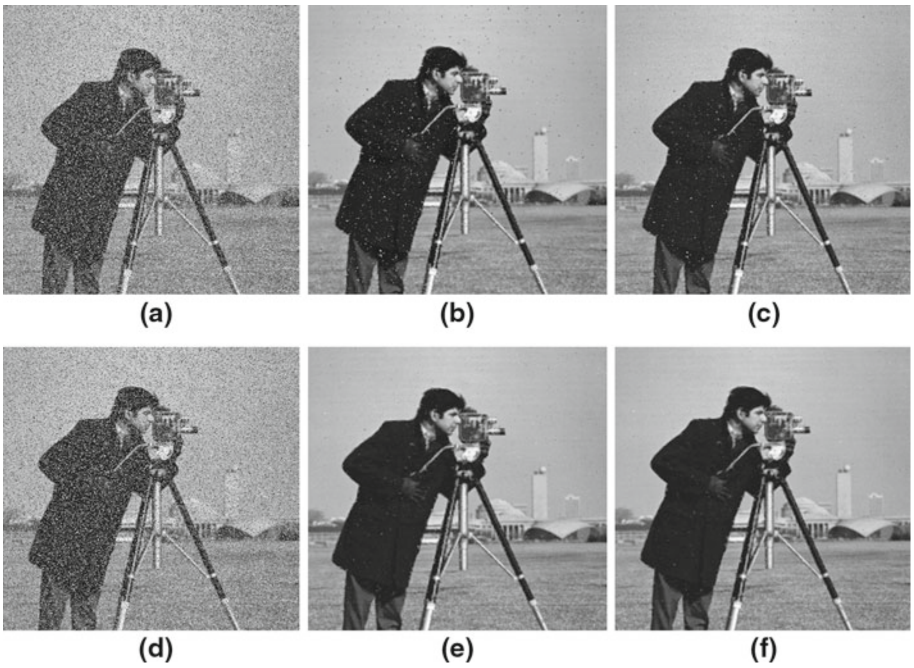

Fig. 14 Performance comparison of image restoration results obtained from the CAMERAMAN using different algorithms. **a** Image corrupted with 30% random-valued impulse noise, **b** SMF (23.67 db), **c** PSMF (29.92 db), **d** DBA (34.34 db), **e** NIDF (30.52 db), and **f** proposed method (36.98 db)

while preserving edges and fine details. Especially, in the presence of high-density impulse noise, our scheme outperformed the previous approaches.

Acknowledgments This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. 2009-0083733) and, by the Korea University of Technology and Education 2010 Research Support for New Professors.

References

1. Gonzalez RC, Woods RE (2007) Digital imageProcessing, 3rd edn. Prentice Hall PTR, New Jersey
2. Huang T, Yang G, Tang G (1979) A fast two-dimensional median filtering algorithm. IEEE Trans Acoust Speech Signal Process 27:13–18
3. Lin T-C (2007) A new adaptive center weighted median filter for suppressing impulsive noise in images. Inf Sci 177:1073–1087
4. Zhou W, Zhang D (1999) Progressive switching median filter for the removal of impulse noise from highly corrupted images. IEEE Trans Circuits Syst II Analog Digit Signal Process 46:78–80
5. Lin T-C (2010) Switching-based filter based on Dempster's combination rule for image processing. Inf Sci 180:4892–4908
6. Wang S-S, Wu C-H (2009) A new impulse detection and filtering method for removal of wide range impulse noises. Pattern Recogn 42:2194–2202
7. Sheng-Fu L, Shih-Mao L, Jyh-Yeong C, Chin-Teng L (2008) A novel two-stage impulse noise removal technique based on neural networks and fuzzy decision. IEEE Trans Fuzzy Syst 16:863–873
8. Hussain A, Jaffar MA, Mirza AM (2009) Detail preserving fuzzy filter for impulse noise removal. Int J Innov Comput Inf Control 5:3583–3591
9. Zhengya X, Hong Ren W, Bin Q, Xinghuo Y (2009) Geometric features-based filtering for suppression of impulse noise in color images. IEEE Trans Image Process 18:1742–1759

10. Petrovic NI, Crnojevic V (2008) Universal impulse noise filter based on genetic programming. IEEE Trans Image Process 17:1109–1120
11. Kaliraj G, Baskar S (2010) An efficient approach for the removal of impulse noise from the corrupted image using neural network based impulse detector. Image Vis Comput 28:458–466
12. Hussain A, Jaffar M, Mirza A (2009) A hybrid image restoration approach: fuzzy logic and directional weighted median based uniform impulse noise removal. Knowl Inf Syst 24:77–90
13. Becerikli Y, Karan TM, Okatan A (2009) A new fuzzy based edge detection for noisy images using modified WFM filter. Int J Innov Comput Inf Control 5:1725–1733
14. Schulte S, Nachtegael M, De Witte V, Vander Weken D, Kerre EE (2006) A fuzzy impulse noise detection and reduction method. IEEE Trans Image Process 15:1153–1162
15. El Akadi A, Amine A, El Ouardighi A, Aboutajdine D (2010) A two-stage gene selection scheme utilizing MRMR filter and GA wrapper. Knowl Inf Syst 24:77–90
16. Suetake N (2001) Self-organizing maps based impulse detector for switching median filters. In: International conferences on In Info-tech and Info-net, 2001. Proceedings. ICII 2001—Beijing, vol 24, pp 20–25
17. Xuming Z, Youlun X (2009) Impulse noise removal using directional difference based noise detector and adaptive weighted mean filter. IEEE Signal Process Lett 16:295–298
18. Srinivasan KS, Ebenezer D (2007) A new fast and efficient decision-based algorithm for removal of high-density impulse noises. IEEE Signal Process Lett 14:189–192
19. Rodríguez-Vázquez K, Fleming PJ (2005) Evolution of mathematical models of chaotic systems based on multiobjective genetic programming. Knowl Inf Syst 8:235–256
20. Pappa GL, Freitas AA (2009) Evolving rule induction algorithms with multi-objective grammar-based genetic programming. Knowl Inf Syst 19:283–309
21. Majid A (2006) Optimization and combination of classifiers using Genetic Programming. In: Faculty of Computer Science, GIK institute, Swabi
22. Kouchakpour P, Zaknich A, Brnl T (2009) Dynamic population variation in genetic programming. Inf Sci 179:1078–1091
23. Mahmood MT, Majid A, Choi TS (2011) Optimal depth estimation by combining focus measures using genetic programming. Inf Sci 181:1249–1263
24. Kouchakpour P, Zaknich A, Bräunl T (2009) A survey and taxonomy of performance improvement of canonical genetic programming. Knowl Inf Syst 21:1–39
25. Zhang S, Karim MA (2002) A new impulse detector for switching median filters. IEEE Signal Process Lett 9:360–363
26. Silva S, Almeida J (2003) GPLAB-a genetic programming toolbox for MATLAB
27. Pitas I, Venetsanopoulos AN (1992) Order statistics in digital image processing. Proc IEEE 80:1893–1921
28. Srinivasan KS, Ebenezer D (2007) A new fast and efficient decision-based algorithm for removal of high-density impulse noises. IEEE Signal Process Lett 14:189–192
29. Majid A, Khan A, Mirza AM (2006) Combination of support vector machines using genetic programming. Int J Hybrid Intell Syst 3:109–125
30. Langdon WB (2000) Size fair and homologous tree genetic programming crossovers. Genet Program Evol Mach 1:95–119

Author Biographies

Abdul Majid received his M.Sc. degree in Electronics from Quaid-i-Azam University, Islamabad, Pakistan, in 1991. He received his M.S. and Ph.D. degrees in Computer Systems Engineering from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Pakistan, in 2003 and 2006, respectively. He has more than 16 years of research and development experience. Currently, he is working as Associate Professor in Department of Computer and Information Sciences at PIEAS. He has completed his Post-Doc Research in the Department of Mechatronics, GIST, South Korea, in 2010. His research areas include pattern recognition, image processing, machine learning, and computational material science.

Choong-Hwan Lee received the B.S. degree in electrical engineering from the Hanyang University, Seoul, Korea, in 1991, the M.S and Ph.D. degrees in electrical engineering and Computer Science from the Korea Advanced Institute of Science and Technology, Daejon, Korea, in 1993 and 1997, repectivly. He served for many years as a senior researcher at Electrical Telecommunication Research Institute (ETRI) in Daejon, Korea. He is currently CTO in Digital Aria Co., Ltd, Seoul, Korea. His research interests include computer vision, graphics, embedded systems, and software engineering.

Muhammad Tariq Mahmood received the MCS degree in computer science from AJK University of Muzaffarabad, Pakistan, in 2004, and the M.S. degree in intelligent software systems from Blekinge Institute of Technology, Sweden, in 2006, and the Ph.D. degree in information and mechatronics from Gwangju Institute of Science and Technology, Korea, in 2011. He is currently a full-time lecturer at School of Computer Science and Engineering, Korea University of Technology and Education, Korea. His research interests include image processing, 3D shape recovery from image focus, computer vision, pattern recognition, and machine learning.

Tae-Sun Choi received the B.S. degree in electrical engineering from the Seoul Nation University, Seoul, Korea, in 1976, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Seoul, Korea, in 1979, and the Ph.D. degree in electrical engineering from the State University of New York at Stony Brook, 1993. He is currently a Professor in the School of Information and Mechatronics at Gwangju Institute of Science and Technology, Gwangju, Korea. His research interests include image processing, machine/robot vision, and visual communications.