Multimed Tools Appl
DOI 10.1007/s11042-017-5216-6

An effective directional interpolation- and inpainting-based algorithm for removing impulse noise

Kuo-Liang Chung[1] · **Yong-Huai Huang**[2]

Received: 6 January 2017 / Revised: 4 August 2017 / Accepted: 8 September 2017

Abstract In this paper, an effective directional interpolation- and inpainting-based impulse noise removal algorithm is proposed. Firstly, each noisy pixel is classified to either the low-density noise or the middle/high density noise. Secondly, a directional interpolation-based noise removal procedure is proposed to denoise the low-density noise. Thirdly, an inpainting-based noise removal procedure is proposed to denoise the middle/high-density noise. Based on ten typical test images, each image with noise level ranging from 30 to 90%, the experimental results demonstrate that in terms of peak-signal-to-noise-ratio (PSNR), structural similarity index (SSIM), and visual effect, the proposed algorithm has the best quality performance when compared with six state-of-the-art noise removal algorithms.

Keywords Highly corrupted images · Directional interpolation · Impulse noise removal · Inpainting · Objective and subjective quality comparison

1 Introduction

Digital images are often corrupted by impulse noise due to electromagnetic interference, bad-quality sensors, interfered channels or environmental noise [9]. Usually, impulse noise is classified as the fixed-valued impulse noise, called the salt and pepper noise which is exhibited as the pixel with the maximum/minimal gray value, and the random-valued impulse noise exhibited as the pixel with gray value within a pre-determined range

✉ Kuo-Liang Chung
klchung01@gmail.com

1 Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, 10672, Taiwan

2 Department of Computer Science and Information Engineering, Jinwen University of Science and Technology, New Taipei City, 23154, Taiwan

Published online: 21 September 2017

[8, 18]. Removing impulse noise and preserving good quality in reasonable execution-time are demanded by signal and image communities.

The standard median filter (SMF) [17] simply replaces the noisy pixel with the median gray value within the window, but suffers from the blurring side-effect for high density noise. Accordingly, several improved SMFs [4, 13, 14, 20] were proposed and they work well for low-density noise, but still suffer from quality degradation for highly corrupted images. Therefore, Hwang and Haddad [10] developed an adaptive window-based median filter (AMF) to improve the quality performance. In [21], Zhang et al. proposed a four-direction based impulse noise detector and then applied the SMF for noise removal.

In [7], Eng and Ma classify all pixels into the noise-free pixels, isolated impulse noisy pixels, and non-isolated impulse noisy pixels. Then the SMF and the Fuzzy weighted MF are adopted to restore isolated impulse noisy pixels and non-isolated impulse noisy pixels, respectively. Srinivasan and Ebenezer [18] proposed an improved decision-based noise removal algorithm (DBA) for highly corrupted images. Unfortunately, when the noise level is up to 80%, the DBA often suffers from the streaking effect. For reducing the streaking effect, Jayaraj and Ebenezer [12] proposed a switching-based median filtering noise removal algorithm for high-density salt and pepper noise. Aiswarya et al. [1] proposed a decision-based nonsymmetrical trimmed median filter (DBUTMF) for noise removal. However, when the noise density ranges from 80% to 90%, its denoising performance is poor. Esakkirajan et al. [8] proposed a modified DBUTMF (MDBUTMF) which adaptively selects the mean or median of the neighboring noise-free pixel values to restore the current noisy pixel.

With the help of four directions, Bai et al. [2] adopted the continued fractions interpolation filter (CFIf) for noise rmoval. Currently, Lin et al. [15] proposed an efficient morphological mean filter (MMF) based noise removal. The MMF first restores the noisy pixels which are adjacent to the noise-free pixels, and then iteratively propagates the denoised results to the remaining noisy pixels. Considering more impulse noise models, Ng and Ma [16] developed a switching median filter with boundary discriminative noise detection (BDND) method for denoising highly corrupted images. Jafar et al. [11] proposed an improved adaptive window-based BDND (IBDND). Recently, Chou et al. [5] proposed the multi-level adaptive switching filter (MASF) for removing noise under different impulse noise models. The MASF has better quality performance than the IBDND.

In this paper, we propose an effective three-stage directional interpolation- and inpainting-based impulse noise removal algorithm. In the first stage, each detected noisy pixel is classified to the low-density noise type or the middle/high density noise type. In the second stage, a directional interpolation-based procedure is proposed to restore the low-density noisy pixels. In the third stage, first, a connected component labeling method is applied to identify and locate all connected middle/high noisy pixels in the denoised image obtained in the second stage. Then, the proposed inpainting-based procedure is proposed to denoise these middle/high density noisy pixels in the selected connected component according to the priority-first rule. Based on ten typical test images, six from the commonly used benchmark, each with size 512 $\times$ 512, and four from the Kodak test set (http://r0k.us/graphics/kodak/), each with size 768 $\times$ 512, each image with noise density ranging from 30% to 90%, the experimental results demonstrate that the proposed noise removal algorithm substantially outperforms six state-of-the-art noise removal algorithms, MDBUTMF, BDND, IBDND, CFIf, MMF, and MASF, in terms of the peak-signal-ratio-noise (PSNR), the structural similarity index (SSIM) [19], and visual effect. The average execution-time required by the proposed algorithm is less than 0.35 second per image, which is middle among the concerned seven algorithms and is promising for real-time demand.

The rest of this paper is organized as follows. In Section 2, the proposed three-stage noise removal algorithm is presented. In Section 3, several experiments are conducted to show the quantitative and qualitative quality superiority of the proposed algorithm when compared with six state-of-the-art algorithms. Some concluding remarks are addressed in Section 4.

2 The proposed three-stage noise removal algorithm

In this section, we first present the procedure for impulse noise detection and classification. Next, we present the proposed directional interpolation-based procedure to restore the low-density noise. Finally, we present the inpainting-based procedure to restore the middle/high-density noisy pixels.

2.1 Noise detection and classification

For the image pixel $I_{x,y}$ at location (x, y), we define a noise detection map $D = (D_{x,y})$, in which '1' denotes a noisy pixel and '0' denotes a noise-free pixel. Therefore, $D = (D_{x,y})$ is defined by

$$D_{x,y} = \begin{cases} 1, & \text{where } I_{x,y} \in [0, \ell_1] \text{ or } I_{x,y} \in [255 - \ell_2, 255], \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where ℓ_1 and ℓ_2 are two pre-determined gray values by the histogram based noise detection method. As mention in Section 3, for the first and second noise models, i.e. the salt-and-pepper noise model, we set $\ell_1 = 0$ and $\ell_2 = 0$; for the third noise model, we set $(\ell_1, \ell_2) \in \{9, 19, 29\}$ and for the fourth noise model, we set $(\ell_1, \ell_2) \in \{(9, 19), (19, 29)\}$.

The number of neighboring noisy pixels of the current noisy pixel is defined by

$$N^{3\times3}_{x,y} = \sum_{(i,j)\in W^{3\times3}_{x,y}} D_{i,j} \tag{2}$$

The noisy pixel is called the low-density noise when the number of its neighboring noise-free noisy pixels is equal to or less than 3, i.e. $N^{3\times3}_{x,y} \leq 3$; otherwise, because the number of its neighboring noise-free ones is equal to or larger than 4, it is called the middle/high-density noise.

2.2 Directional interpolation-based noise removal for low-density noise

For the low-density noisy pixel $I_{x,y}$, for which $D_{i,j} = 1$ and $N^{3\times3}_{x,y} \leq 3$ are held, we further examine its eight neighboring pixels to determine its textural class as either a horizontal edge, a vertical edge, a textural region, or a smooth region. The main idea of the second stage is that $I_{x,y}$ will be restored by the average value of its two horizontal (vertical) neighboring noise-free pixels if there is a horizontal (vertical) line passing through it; by the mean value of the four-directional neighboring noise-free pixels if there is a textural area around $I_{x,y}$; by applying the Gaussian filter with mask $\begin{array}{|c|c|c|}\hline 0.5 & 1 & 0.5 \\ \hline 1 & 0 & 1 \\ \hline 0.5 & 1 & 0.5 \\ \hline \end{array}$ on the eight-directional neighboring noise-free pixels if there is a smooth area around $I_{x,y}$. To determine the texture type of a low-density noisy pixel, we first calculate the horizontal gradient and vertical gradient of $I_{x,y}$, respectively, by

$$G_{x,y}^{h} = \begin{cases} \left| \dfrac{\sum_{(i,j)\in S_{x,y}^{l}} f_{i,j}}{\sum_{(i,j)\in S_{x,y}^{l}} c_{i,j}+\delta} - \dfrac{\sum_{(i,j)\in S_{x,y}^{r}} f_{i,j}}{\sum_{(i,j)\in S_{x,y}^{r}} c_{i,j}+\delta} \right|, & \text{when } c_{i,j} \neq 0, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

and

$$G_{x,y}^{v} = \begin{cases} \left| \dfrac{\sum_{(i,j)\in S_{x,y}^{t}} f_{i,j}}{\sum_{(i,j)\in S_{x,y}^{t}} c_{i,j}+\delta} - \dfrac{\sum_{(i,j)\in S_{x,y}^{b}} f_{i,j}}{\sum_{(i,j)\in S_{x,y}^{b}} c_{i,j}+\delta} \right|, & \text{when } c_{i,j} \neq 0, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $S_{x,y}^{l} = \{(x-1, y-1), (x-1, y), (x-1, y+1)\}$, $S_{x,y}^{r} = \{(x+1, y-1), (x+1, y), (x+1, y+1)\}$, $S_{x,y}^{t} = \{(x-1, y-1), (x, y-1), (x+1, y-1)\}$, and $S_{x,y}^{b} = \{(x-1, y+1), (x, y+1), (x+1, y+1)\}$. We set $\delta = 0.001$ in the experiment to avoid the infinite case of the rational term when the denominator of the rational term tends to zero. The noise-free pixel $f_{i,j}$ and its noise-free status $c_{i,j}$ are defined by

$$f_{i,j} = \begin{cases} I_{i,j}, & \text{when } D_{i,j} = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

and

$$c_{i,j} = \begin{cases} 1, & \text{when } D_{i,j} = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The following procedure is used to determine the texture type of the noisy pixel $I_{x,y}$. In the procedure, if there is a vertical (or horizontal) line passing through the noisy pixel $I_{x,y}$, the gradient $G_{x,y}^{h}$ (or $G_{x,y}^{v}$) must be larger than the threshold T_G, and empirically, the best value of T_G is set to 10 for all test images. The threshold T_R denotes the minimal ratio bound of $G_{x,y}^{h}$ over $G_{x,y}^{v}$ or vice versa, and empirically, the best value of T_R is set to 1.5 for all test images. The similar texture type determination concept for low-density noise can be seen in [16].

Procedure: Texture type determination for low-density noise

```
if $G_{x,y}^{h} > T_G$ or $G_{x,y}^{v} > T_G$ and $\frac{\max(G_{x,y}^{h}, G_{x,y}^{v})}{\min(G_{x,y}^{h}, G_{x,y}^{v})} > T_R$ then
    if $G_{x,y}^{v} > G_{x,y}^{h}$ then
        $I_{x,y}$ is in a horizontal edge.
    else
        $I_{x,y}$ is in a vertical edge.
else
    if $G_{x,y}^{h} > T_G$ and $G_{x,y}^{v} > T_G$ then
        $I_{x,y}$ is in a textural region.
    else
        $I_{x,y}$ is in a smooth region.
```

Based on the determined texture type of the low-density noise, the proposed directional interpolation-based noise removal is presented below.

Procedure: Directional interpolation-based noise removal for low-density noise

```
if $I_{x,y}$ is in a horizontal edge and $c_{x-1,y} + c_{x+1,y} > 1$ then
    $I_{x,y} = \frac{f_{x-1,y}+f_{x+1,y}}{c_{x-1,y}+c_{x+1,y}}$.    (7)
else if $I_{x,y}$ is in a vertical edge and $c_{x,y-1} + c_{x,y+1} > 1$ then
    $I_{x,y} = \frac{f_{x,y-1}+f_{x,y+1}}{c_{x,y-1}+c_{x,y+1}}$    (8)
else
    if $I_{x,y}$ is in a texture region then
        $I_{x,y} = \frac{\sum_{(i,j)\in S^c_{x,y}} f_{i,j}}{\sum_{(i,j)\in S^c_{x,y}} c_{i,j}}$    (9)
        where $S^c_{x,y} = \{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$.
    else
        $I_{x,y} = \frac{\sum_{(i,j)\in S^c_{x,y}} f_{i,j} + 0.5 \sum_{(i,j)\in S^d_{x,y}} f_{i,j}}{\sum_{(i,j)\in S^c_{x,y}} c_{i,j} + 0.5 \sum_{(i,j)\in S^d_{x,y}} c_{i,j}}$    (10)
        where $S^d_{x,y} = \{(x-1, y-1), (x-1, y+1), (x+1, y-1),$
        $(x+1, y+1)\}$.
```

2.3 Inpainting-based noise removal for Middle/High-density noise

After performing the second stage for denoising low-density noise, we have the updated D and $N^{3\times3}_{x,y}$s together with the recovered image in which the low-density noisy pixels have been restored. For restoring middle/high-density noisy pixels, the proposed inpainting-based procedure is presented below. Selecting one middle/high-density noisy pixel as a seed, we apply the connected component labeling algorithm (CCLA) [6] to obtain the seed-based connected noisy component (CNC) in which each node in the CNC is a middle/high-density noisy pixel. By experiment, when the path length of each CNC is at most 20, the inpainting effect, i.e. denoising effect, is better for having more noise-free pixels at the endpoints of these CNCs. We continue CCLA until all the remaining middle/high-density noisy pixels have been processed. Suppose we obtain n CNCs, C_0, C_1 , ..., and C_{n-1}. Then we sort these n CNCs based on their priorities, and the priority of C_k is defined by

$$P_k = \frac{\sum_{I'_{i,j}\in C_k} (9 - N^{3\times3}_{i,j})}{|C_k|} \tag{11}$$

where $|C_k|$ denotes the number of noisy pixels in C_k, $0 \leq k \leq n-1$. The following inpainting-based noise removal procedure begins at the selected CNC with the highest priority and ends at the one with the lowest priority.

In the recovery of each CNC, the proposed inpainting-based procedure considers the noisy pixels between the current CNC and the adjacent CNC in order to avoid the gray value discontinuity between two adjacent recovered CNCs. Thus, for C_k, the rectangular region R_k bounded by $(x_l^k - \Delta, y_t^k - \Delta)$ and $(x_r^k + \Delta, y_b^k + \Delta)$, where (x_l^k, y_t^k) and (x_r^k, y_b^k) denote the top-left corner and bottom-right corner of the minimum bounding box of C_k and empirically, Δ is set to 5 in order to cover the overlapped pixels between C_k and its adjacent CNCs, leading to better denoising effect. Instead of denoising C_k directly, the proposed inpainting-based noise removal procedure will start from denoising the noisy pixels in R_k. Before that, an inpainting status map of R_k, $M = (M_{x,y})$, is defined as follows:

$$M_{x,y} = \begin{cases} -1, & I_{x,y} \notin R_k \\ 0, & I_{x,y} \text{ is unnecessary to be inpainted } (D_{x,y} = 0), \\ 1, & I_{x,y} \text{ is to be inpainted } (D_{x,y} = 1), \\ 2, & I_{x,y} \text{ has been inpainted.} \end{cases} \tag{12}$$

In R_k, the noisy pixels are inpainted by the onion-peel order, i.e. from the noisy pixels on the border to thous on the center of R_k. Different from inpainting holes in video [3], we plug the Navier-stokes formula into the proposed inpainting status map. With the help of the neighboring noise-free pixels $f'_{i,j}$s within a $m \times m$ window $W_{x,y}^{m\times m}$ centered at the location (x, y), by the Navier-stokes formula, the noisy pixel $I_{x,y}$ with $M_{x,y} = 1$ is restored, i.e. inpainted, by

$$I_{x,y} = \frac{\sum\limits_{(i,j)\in W_{x,y}^{m\times m}} \omega_{i,j}^{x,y} f'_{i,j}}{\sum\limits_{(i,j)\in W_{x,y}^{m\times m}} \omega_{i,j}^{x,y} c'_{i,j}} \tag{13}$$

where

$$f'_{i,j} = \begin{cases} I_{i,j}, & M_{i,j} \in \{0, 2\}, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

and

$$c'_{i,j} = \begin{cases} 1, & M_{i,j} \in \{0, 2\}, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

In (13), we set $m = 3$ when there is more than one noise-free pixel in the neighborhood of $I(x, y)$, i.e. $N_{x,y}^{3\times 3} < 8$; otherwise, we set $m = 5$ for $N_{x,y}^{3\times 3} \geq 8$. This adaptive window size-based approach achieves better denosing result. Using the inner production operation, the weight $\omega_{i,j}^{x,y}$ is calculated by

$$\omega_{i,j}^{x,y} = \frac{1}{\|r_{i,j}^{x,y}\|^2} \cdot \frac{|G_{i,j} \cdot r_{i,j}^{x,y}|}{\|G_{i,j}\|\|r_{i,j}\|} \tag{16}$$

where $G_{i,j} = (G_{i,j}^h, G_{i,j}^v)$ denotes the gradient vector and $r_{i,j}^{x,y} = (x - i, y - j)$ denotes the directional vector. The bigger the weight $\omega_{i,j}^{x,y}$ is, the higher the correlation of $G_{i,j}$ and $r_{i,j}^{x,y}$ is. It means that by (16), if the weight the neighboring noise-free pixel $f'_{i,j}$ within the window W is high, based on (13), the neighboring noise-free pixel the noisy pixel $f'_{i,j}$ has more positive influence on the restoration of the current noisy pixel $I_{x,y}$.

In what follows, the inpainted pixels in C_k is further refined. The refinement for $I_{x,y}$ is realized by

$$I_{x,y} = \frac{\sum\limits_{(i,j)\in W_{x,y}^{m\times m}} \omega_{i,j}^{x,y} I_{i,j}}{\sum\limits_{(i,j)\in W_{x,y}^{m\times m}} \omega_{i,j}^{x,y}}. \tag{17}$$

After all $I_{x,y}$s in C_k have been refined, we calculate the difference between the currently restored result $I_{x,y}$, denoted by $I_{x,y}^{C}$, and the previously restored $I_{x,y}$, denoted by $I_{x,y}^{P}$, by

$$E = \sum_{I_{x,y}\in C_k} \frac{|I_{x,y}^{C} - I_{x,y}^{P}|}{|C_k|}. \tag{18}$$

We continue the refinement step until the termination condition $E < T_E$ $(= 1$ empirically) is held. Note that in our experiments, the number of refinement iterations being 2 is enough. After finishing the refinement step for C_k, the noise detection map D is updated by setting $D(x, y) = 0$ for $I_{x,y} \in C_k$.

After performing the above inpainting-based noise removal process on C_0, C_1 , ..., and C_{n-1}, the middle/high-density noisy pixels in I can be restored. The pseudo-code of the above inpainting-based noise removal procedure is listed below.

Procedure: Inpainting-based noise removal for middle/high-density noise

```
Perform CCLA to obtain $N$ CNCs, $C_0$, $C_1$ , ..., and $C_{n-1}$.
Sort CNCs based on their priorities by (11).
repeat
    Select $C_k$ with the highest priority from the remaining CNCs.
    Determine the rectangular region $R_k$ to cover $C_k$.
    Initialize the inpainting status map $M$ by (12).
    repeat
        Select the noisy pixel $I_{x,y}$ from $R_k$ based on onion-peel order.
        Recover $I_{x,y}$ by (13)–(16).
        Update $M_{x,y}$ by setting $M_{x,y} = 2$.
    until all noisy pixels in $R_k$ have been recovered.
    repeat
        repeat
            Select the noisy pixel $I_{x,y}$ from $C_k$ based on onion-peel order.
            Refine $I_{x,y}$ by (17).
        until all noisy pixels in $C_k$ have been refined.
        Calculate $E$ by (18).
    until $E < T_E$.
    Update $D$ by setting $D(x, y) = 0$ for $I_{x,y} \in C_k$.
until all CNCs have been recovered.
```

Because the directional interpolation-based noise removal procedure in the second stage takes the texture type of the low-density noise into account, the second stage achieves a good denoising effect for low-density noisy pixels. In the third-stage, several useful strategies, such as the connected component labeling, priority-first denoising policy, determining rectangular region to smooth the denosing effect, Navier-stokes formula-based inpainting procedure, and 2-stage refinement, are integrated to efficiently denoise the middle/high-density noisy pixels. In the next section, the experimental results demonstrate that the proposed three-stage noise removal algorithm has substantial quality improvement and the edge-preservation effect.

3 Experimental results

In our experiments, four impulse noise models are considered. The first noise model is the salt-and-pepper noise model, i.e. $\ell_1 = 0$, $\ell_2 = 0$, and $p_1 = p_2$ in which p_1 and p_2 denote the probability of noise with gray value 0 and gray value 255, respectively. The second noise model is similar to the salt-and-pepper noise model but with unbalanced noise density, $p_1 \neq p_2$. The third noise model considers the noise with larger and balanced noise range, $\ell_1 > 0$, $\ell_2 > 0$, and balanced noise density, $p_1 = p_2$, while the fourth noise model considers the noise with unbalanced noise density and balanced noise range.

We compare PSNR, SSIM, and the visual effect performance of the proposed algorithm with the six concerned algorithms, MDBUTMF [8], BDND [16], IBDND [11], CFIf [2], MMF [15], and MASF [5]. Since MDBUTMF, CFIf, and MMF only deal with images corrupted by the salt-and-pepper noise model, for fairness, we apply the first noise model to generate the corrupted images. As shown in Fig. 1, the 10 test images are Lena, Boat,

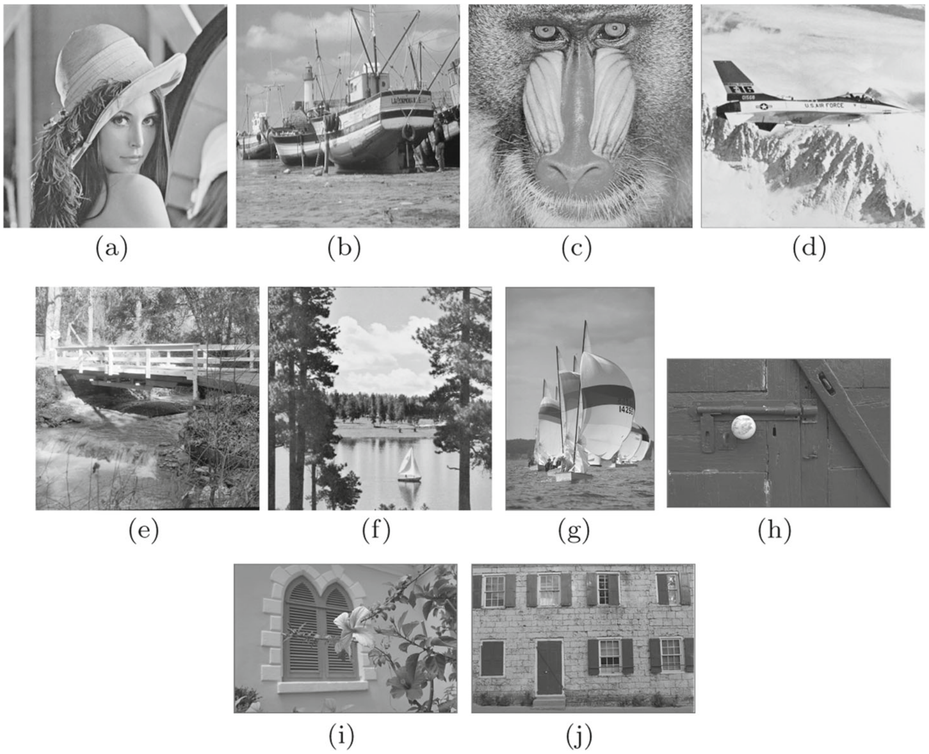

Fig. 1 Ten test images. **a** Lena. **b** Boat. **c** Mandrill. **d** Airplane. **e** Stream and bridge. **f** Sailboat on lake. **g** Sailboard. **h** Door. **i** Window. **j** Wall

Table 1 PSNR comparison for the first noise model

Image	Noise density	Proposed	MDBUTMF [8]	BDND [16]	IBDND [11]	CFIf [2]	MMF [15]	MASF [5]
		PSNR	ΔPSNR(dB)					
Lena	30%	37.64	1.23	1.31	0.66	0.80	0.83	0.35
	50%	34.37	2.41	2.06	0.61	0.76	0.81	0.46
	70%	31.14	1.86	2.32	0.43	0.55	0.81	0.52
	90%	26.66	1.89	1.62	0.83	0.65	0.88	0.46
Boat	30%	34.62	1.86	2.35	1.75	1.52	1.59	1.01
	50%	31.25	2.72	2.71	1.22	1.26	1.23	0.84
	70%	28.11	1.92	2.59	0.72	0.91	0.98	0.65
	90%	24.13	1.64	1.62	0.95	0.86	0.98	0.50
Mandrill	30%	27.71	0.96	1.04	0.30	0.67	0.57	0.26
	50%	25.01	1.56	1.21	0.27	0.56	0.60	0.29
	70%	22.74	1.22	1.17	0.29	0.37	0.78	0.25
	90%	20.26	0.97	0.96	0.89	0.16	1.16	0.35
Airplane	30%	37.46	2.49	2.40	1.44	1.53	1.99	1.21
	50%	33.75	3.46	3.29	1.21	1.33	1.62	1.13
	70%	29.95	2.34	3.06	0.60	0.83	1.14	0.77
	90%	24.97	2.17	1.85	0.99	1.09	0.92	0.48
Stream and	30%	29.84	1.04	1.78	1.04	0.71	0.53	0.28
bridge	50%	27.25	1.93	2.07	0.83	0.75	0.60	0.39
	70%	24.81	1.62	1.99	0.66	0.62	0.74	0.36
	90%	21.65	1.50	1.41	1.04	0.51	1.00	0.41
Sailboat	30%	33.86	1.30	1.36	0.50	0.80	0.71	0.34
on lake	50%	30.88	2.59	2.34	0.62	0.91	0.87	0.57
	70%	27.79	1.90	2.57	0.45	0.76	0.86	0.54
	90%	23.51	1.82	1.73	0.94	0.98	0.88	0.42
Sailboard	30%	36.97	2.37	3.83	3.01	1.8	2.07	1.40
	50%	33.42	3.09	4.04	2.04	1.47	1.50	1.11
	70%	30.10	2.22	3.43	1.32	0.98	1.06	0.87
	90%	25.88	1.67	1.80	1.09	0.87	0.89	0.45
Door	30%	37.21	1.21	1.81	1.25	1.10	0.99	0.58
	50%	34.22	2 .16	1.99	0.92	0.97	0.89	0.54
	70%	31.69	1.50	1.83	0.67	0.63	0.88	0.42
	90%	28.50	1.78	0.95	0.83	0.32	1.00	0.26
Window	30%	37.07	1.52	2.22	1.61	0.77	1.21	0.57
	50%	33.80	2.84	2.93	1.29	0.89	1.14	0.76
	70%	30.27	2.03	2.82	0.68	0.75	0.87	0.67
	90%	25.68	1.68	1.60	0.75	1.03	0.83	0.47
Wall	30%	29.80	1.67	2.36	1.56	1.45	1.33	0.78
	50%	27.05	2.84	2.62	1.22	1.30	1.18	0.84
	70%	24.30	1.98	2.22	0.74	0.87	0.92	0.67
	90%	21.07	1.29	1.25	0.87	0.30	1.10	0.49

Table 2 SSIM comparison for the first noise model

Image	Noise density	Proposed	MDBUTMF [8]	BDND [16]	IBDND [11]	CFIf [2]	MMF [15]	MASF [5]
		SSIM	ΔSSIM					
Lena	30%	0.967	0.005	0.006	0.002	0.003	0.003	0.001
	50%	0.937	0.023	0.018	0.004	0.007	0.006	0.003
	70%	0.890	0.032	0.040	0.006	0.011	0.015	0.007
	90%	0.785	0.079	0.056	0.027	0.017	0.033	0.018
Boat	30%	0.939	0.012	0.014	0.008	0.010	0.007	0.005
	50%	0.884	0.040	0.038	0.013	0.019	0.015	0.010
	70%	0.798	0.051	0.074	0.015	0.031	0.024	0.015
	90%	0.569	0.092	0.081	0.032	0.102	0.034	0.027
Mandrill	30%	0.911	0.017	0.019	0.006	0.014	0.010	0.005
	50%	0.827	0.068	0.047	0.010	0.027	0.017	0.009
	70%	0.694	0.079	0.087	0.007	0.037	0.021	0.014
	90%	0.460	0.081	0.079	0.019	0.040	0.027	0.021
Airplane	30%	0.980	0.006	0.007	0.003	0.005	0.004	0.002
	50%	0.960	0.030	0.025	0.006	0.012	0.007	0.005
	70%	0.920	0.038	0.051	0.006	0.017	0.011	0.009
	90%	0.812	0.102	0.057	0.018	0.029	0.018	0.015
Stream and bridge	30%	0.931	0.014	0.024	0.012	0.012	0.008	0.004
	50%	0.866	0.064	0.056	0.017	0.025	0.016	0.009
	70%	0.756	0.076	0.106	0.016	0.037	0.02	0.017
	90%	0.533	0.101	0.102	0.030	0.050	0.027	0.027
Sailboat on lake	30%	0.946	0.004	0.005	0.001	0.001	0.000	0.000
	50%	0.902	0.028	0.023	0.004	0.006	0.004	0.002
	70%	0.836	0.040	0.056	0.008	0.015	0.015	0.008
	90%	0.694	0.095	0.070	0.028	0.034	0.031	0.019
Sailboard	30%	0.971	0.009	0.01	0.009	0.007	0.008	0.005
	50%	0.942	0.031	0.031	0.013	0.012	0.012	0.008
	70%	0.893	0.04	0.050	0.014	0.016	0.017	0.013
	90%	0.789	0.069	0.051	0.021	0.019	0.023	0.015
Door	30%	0.958	0.011	0.014	0.008	0.011	0.008	0.005
	50%	0.918	0.041	0.029	0.011	0.018	0.014	0.008
	70%	0.855	0.046	0.044	0.010	0.019	0.018	0.011
	90%	0.742	0.074	0.040	0.023	0.011	0.031	0.012
Window	30%	0.981	0.006	0.010	0.007	0.003	0.005	0.002
	50%	0.960	0.029	0.029	0.010	0.008	0.009	0.005
	70%	0.916	0.040	0.059	0.010	0.014	0.013	0.012
	90%	0.790	0.077	0.062	0.015	0.035	0.017	0.017
Wall	30%	0.933	0.024	0.031	0.018	0.022	0.019	0.011
	50%	0.863	0.089	0.067	0 .025	0.038	0.028	0.019
	70%	0.739	0.102	0.111	0.021	0.047	0.027	0.028
	90%	0.503	0.098	0.091	0.023	0.028	0.028	0.028

Mandrill, Airplane, Stream and bridge, Sailboat on lake, Sailboard, Door, Window, and Wall, where the last four images are from the Kodak set. All experiments were performed on a computer with an Intel i7-3770 CPU 3.4 GHz, 4GB RAM, and Microsoft Windows 7 operating system. The programming language and library used are Visual Studio C++ 2008 and the OpenCV library.

Interfering each test image with noise density ranging from 30% to 90%, the PSNR gain and SSIM gain of the proposed algorithm, 'ΔPSNR' and 'ΔSSIM', over the previous six algorithms are shown in Tables 1 and 2, respectively, in which the two columns, PSNR and SSIM, below 'Proposed' denote the PSNR and SSIM performance of the proposed algorithm, respectively. The last rows of Tables 1 and 2 indicate the average quality performance merits of the proposed algorithm. The positive PSNR and SSIM gains indicate that the proposed algorithm has the best quantitative quality performance among the seven concerned algorithms. In detail, the proposed algorithm outperforms MDBUTMF, BDND, ISDND, CFI, MMF, and MASF in 0.96-3.46 dBs, 0.95-4.04 dBs, 0.27-3.01 dBs, 0.16-1.8 dBs, 0.3-2.07 dBs, and 0.25-1.4 dBs, respectively, under the four noise levels. Note that because of employing the inpainitng-and-refinement stage for restoring the middle/high density noise, the proposed algorithm does preserve more edge and texture information

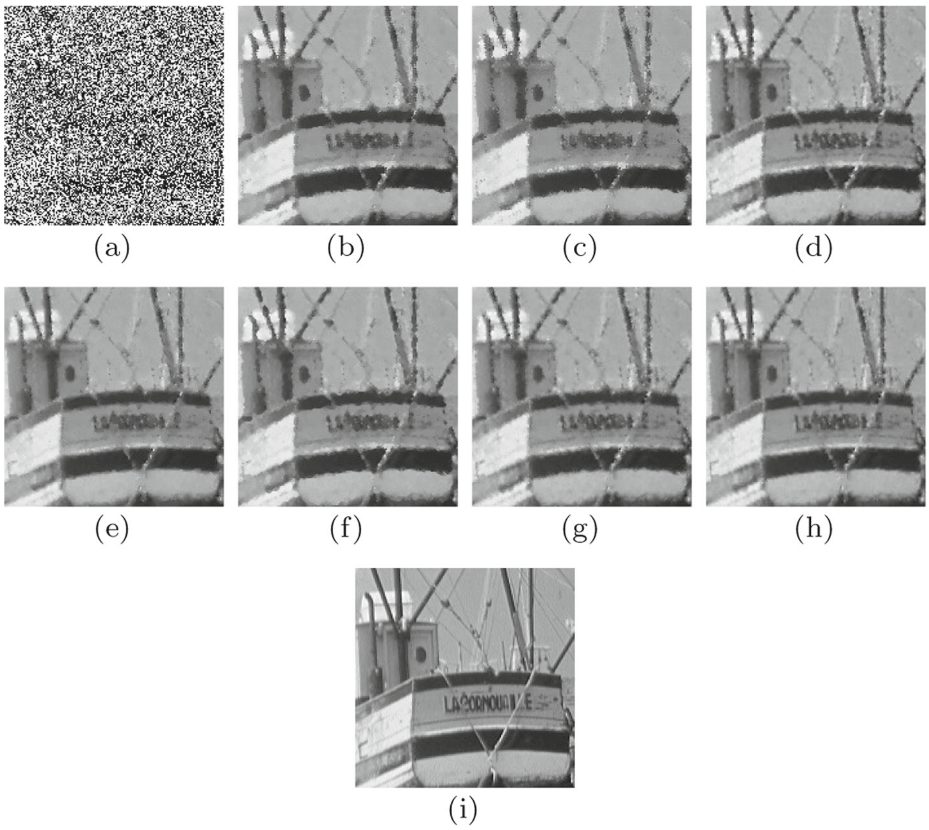

Fig. 2 Denoised results for the Boat image corrupted by 70% noise density. **a** Corrupted image. **b** MDBUTMF. **c** BDND. **d** IBDND. **e** CFIf. **f** MMF **g** MASF. **h** The proposed algorithm. **i** The original image

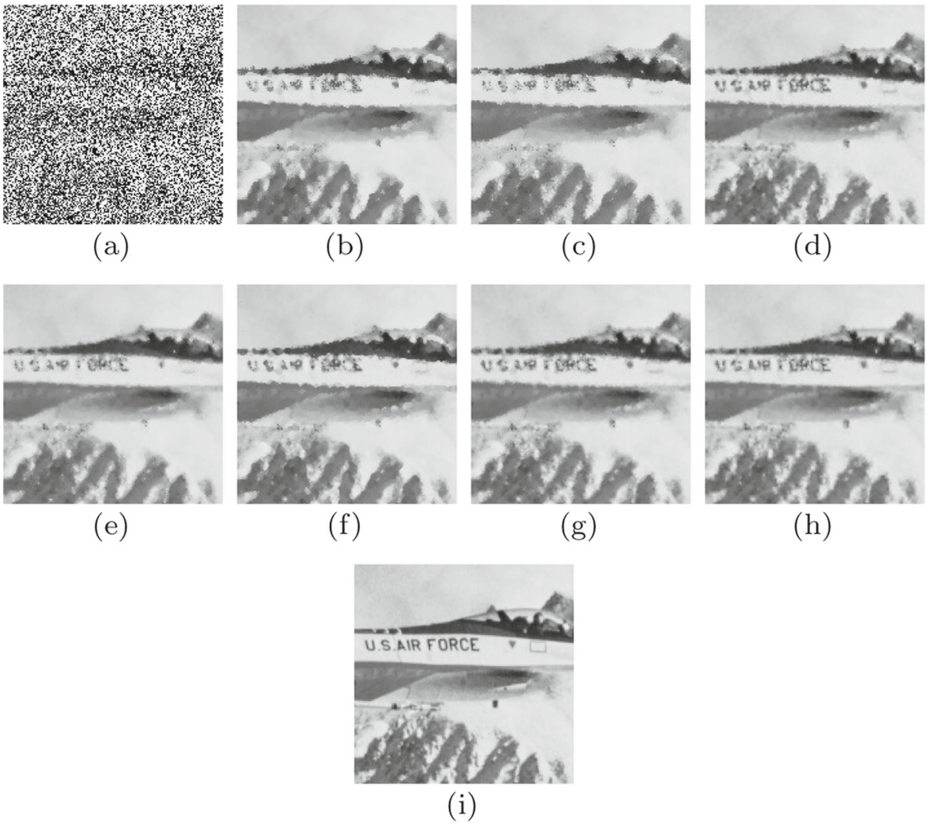

Fig. 3 Denoised results for the Airplane image corrupted by 70% noise density. **a** Corrupted image. **b** MDBUTMF. **c** BDND. **d** IBDND. **e** CFIf. **f** MMF **g** MASF. **h** The proposed algorithm. **i** The original image

for highly corrupted images. The main reason is that the SSIM gain improvement in the proposed algorithm is proportional to the noise level, leading to better quality effect.

We take the two original test images, Boat and Airplane, as shown in Figs. 2i and 3i, as the visual comparison examples. The two images are corrupted by 70% salt-and- pepper noise density and the three corresponding corrupted images are shown in Figs. 2a and 3a. After running the seven algorithms on the two corrupted images, the two sets of resultant denoised images are shown in Figs. 2b–h and 3b–h. It is observed that when compared with the MDBUTMF, BDND, IBDND, CFIf, MMF, and MASF, the proposed algorithm can substantially reduce the blurring and zigzag artifacts in the denoised images.

After demonstrating the quality merits of the proposed algorithm for high-density impulse noise, the related performance comparison among the seven concerned algorithms for the second, third, and fourth noise models are shown in Tables 3 and 4. For simplicity, Tables 3 and 4 only show the average PSNR and SSIM of each test image. The parameters of each noise model are set as below. In the second noise model, four combinations of noise density, $(p_1, p_2) = \{(20\%, 50\%), (30\%, 40\%), (40\%, 30\%), (50\%, 20\%)\}$ are considered. In the third noise model, we consider 70% noise density and $(\ell_1, \ell_2) \in \{9, 19, 29\}$. In the fourth noise model, we consider $(\ell_1, \ell_2) \in \{(9, 19), (19, 29)\}$ and $(p_1, p_2) \in \{(20\%, 50\%), (30\%, 40\%)\}$. From Tables 3 and 4, we observe that the proposed noise removal algorithm

Table 3 PSNR comparison for the second, third, and fourth noise models

Noise model	Noise range		Noise density		Proposed	MDBUTMF [8]	BDND [16]	IBDND [11]	CFIf [2]	MMF [15]	MASF [5]
	ℓ_1	ℓ_2	p_1	p_2	PSNR	ΔPSNR(dB)					
Second model	0	0	20%	50%	29.10	1.98	3.49	2.12	0.83	0.96	0.67
	0	0	30%	40%	29.15	2.01	2.64	0.92	0.85	0.96	0.68
	0	0	40%	30%	29.07	1.91	2.57	0.83	0.80	0.92	0.64
	0	0	50%	20%	29.12	1.97	2.94	1.31	0.81	0.97	0.68
Third Model	9	9	35%	35%	28.99	1.88	14.75	15.66	0.81	0.91	0.63
	19	19	35%	35%	28.66	1.78	14.25	15.08	0.65	0.81	0.61
	29	29	35%	35%	27.79	1.57	12.99	13.82	0.51	0.53	0.50
Fourth Model	9	19	20%	50%	28.98	1.89	18.86	18.59	0.65	0.87	0.63
	9	19	30%	40%	28.90	1.89	15.04	15.78	0.64	0.84	0.63
	19	29	20%	50%	28.52	1.70	18.09	17.80	0.56	0.74	0.60
	19	29	30%	40%	28.50	1.74	14.44	15.14	0.49	0.72	0.56

Table 4 SSIM comparison for the second, third, and fourth noise models

Noise model	Noise range		Noise density		Proposed	MDBUTMF [8]	BDND [16]	IBDND [11]	CFIf [2]	MMF [15]	MASF [5]
	ℓ_1	ℓ_2	p_1	p_2	SSIM	ΔSSIM					
Second model	0	0	20%	50%	0.851	0.057	0.107	0.055	0.024	0.019	0.015
	0	0	30%	40%	0.851	0.057	0.067	0.014	0.024	0.019	0.015
	0	0	40%	30%	0.851	0.056	0.065	0.013	0.024	0.019	0.015
	0	0	50%	20%	0.851	0.057	0.072	0.020	0.024	0.019	0.016
Third model	9	9	35%	35%	0.850	0.056	0.712	0.727	0.024	0.019	0.015
	19	19	35%	35%	0.849	0.055	0.707	0.721	0.023	0.018	0.015
	29	29	35%	35%	0.842	0.055	0.692	0.707	0.023	0.017	0.015
Fourth model	9	19	20%	50%	0.85	0.057	0.781	0.778	0.023	0.019	0.015
	9	19	30%	40%	0.854	0.056	0.721	0.733	0.023	0.018	0.015
	19	29	20%	50%	0.852	0.057	0.776	0.772	0.023	0.018	0.015
	19	29	30%	40%	0.852	0.056	0.716	0.729	0.023	0.018	0.015

still has the best quality performance among the seven concerned algorithms for the second, third, and fourth noise models.

Finally, the execution-time performance comparison of the concerned algorithms is provided in terms of milliseconds (ms) per image. On average, the proposed algorithm, MDBUTMF, CFIf, MMF, MASF, BDND, and IBDND take 350 ms, 62 ms, 259 ms, 22 ms, 342 ms, 2700 ms, and 2772 ms, respectively. Because a sorting operation is required in each 21×21 sliding window, the previous BDND and IBDND have poor execution-time performance. Although the proposed algorithm has modest execution-time performance among the seven concerned algorithms, it has the best quality performance in terms of PSNR, SSIM, and visual effect.

4 Conclusion

The proposed three-stage algorithm for removing impulse noise has been presented. We first classify each noisy pixel into the low-density noise or the middle/high-density noise. Next, according to the texture of each low-density noisy pixel, a directional interpolation-based noise removal method is proposed. Finally, the proposed inpainting-based noise removal method for restoring the middle/high-density noise is presented. Based on four noise models with different noise densities, the experimental results demonstrate that the proposed noise removal algorithm has the best subjective and objective quality performance when compared with the six concerned algorithms. Specifically, the execution-time performance of the proposed algorithm is middle among the concerned seven algorithms and is quite competitive with the two recently published real-time noise removal algorithms, the CFIf and MASF.

Acknowledgments The authors appreciate the proofreading help of Ms. C. Harrington, the valuable comments of the three anonymous referees, and the support under the contracts MOST 104-2221-E-011-118-MY3 and MOST 104-2221-E-228-006.

References

1. Aiswarya K, Jayaraj V, Ebenezer D (2010) A new and efficient algorithm for the removal of high density salt and pepper noise in images and videos. In: Proc International Conference Computer Modeling and Simulation, vol 4, pp 409–413
2. Bai T, Tan J, Hu M, Wang Y (2014) A novel algorithm for removal of salt and pepper noise using continued fractions interpolation. Signal Process 102:247–255
3. Bertalmio M, Bertozzi A, Sapiro G (2001) Navier-stokes, fluid dynamics, and image and video inpainting. In: Proceedings IEEE Conference Computer Vision and Pattern Recognition, vol 1, pp 355–362
4. Brownrigg DRK (1984) The weighted median filter. Commun ACM 27(8):807–818
5. Chou HH, Hsu LY, Hu HT (2015) Multi-level adaptive switching filters for highly corrupted images. J Vis Commun Image Represent 30(7):363–375
6. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to Algorithms. 3rd Edition, Subsection 22. 5: Strongly connected components. The MIT Press, Cambridge
7. Eng HL, Ma KK (2001) Noise adaptive soft-switching median filter. IEEE Trans Image Process 10: 242–251
8. Esakkirajan S, Veerakumar T, Subramanyam AN, PremChand CH (2011) Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter. IEEE Signal Process Lett 18(5):287–290
9. Gonzalez RC, Woods RE (2002) Digital Image Processing. Prentice Hall, Upper Saddle River

10. Hwang H, Haddad RA (1995) Adaptive median filters: new algorithms and results. IEEE Trans Image Process 4(4):499–502
11. Jafar IF, AlNa'mneh RA, Darabkh KA (2013) Efficient improvements on the BDND filtering algorithm for the removal of high-density impulse noise. IEEE Trans Image Process 22(3):1223–1232
12. Jayaraj V, Ebenezer D (2010) A new switching-based median filtering scheme and algorithm for removal of high-density salt and pepper noise in images, EURASIP. J Advances Signal Process 2010:1–11
13. Ko SJ, Lee YH (1991) Center weighted median filters and their applications to image enhancement. IEEE Trans Circuits Syst 38(9):984–993
14. Lin HM, Willson AN (1988) Median filters with adaptive length. IEEE Trans Circuits Syst 35(6):675–690
15. Lin PH, Chen BH, Cheng FC, Huang SC (2015) A Morphological mean filter for impulse noise removal, IEEE J. Display Technology, Accepted for publication
16. Ng PE, Ma KK (2006) A switching median filter with boundary discriminative noise detection for extremely corrupted images. IEEE Trans Image Process 15(6):506–1516
17. Pitas I, Venetsanopoulos AN (1992) Order statistics in digital image processing. Proc IEEE 80(12):1893–1921
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(3):189–192
19. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
20. Yang R, Yin L (1995) Optimal weighted median filtering under structural constrains. IEEE Trans Signal Process 43(3):591–604
21. Zhang S, Karim MA (2002) A new impulse detector for switching median filters. IEEE Signal Process Lett 9(11):360–363

Kuo-Liang Chung received the B.S., M.S., and Ph.D. degrees from National Taiwan University, Taipei, Taiwan, in 1982, 1984, and 1990, respectively. He is currently a Chair Professor with the Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan. His current research interests include video coding and image processing. Dr. Chung was a recipient of the Distinguished Research Award (2004-2007) and Distinguished Research Project Award (2009-1012) from the National Science Council of Taiwan. He is now an Associate Editor of the Journal of Visual Communication and Image Representation.

Yong-Huai Huang received the BS degree in Information Management from Aletheia University, Danshui, Taipei, Taiwan, and the MS and PhD degrees in Computer Science and Information Engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan. He is now an associate professor in the Department of Computer Science and Information Engineering at Jinwen University of Science and Technology, Hsin-Tien Dis., New Taipei City, Taiwan. His research interests include image processing and compression, and algorithms.