Removal of High-Density Salt-and-Pepper Noise in Images With an Iterative Adaptive Fuzzy Filter Using Alpha-Trimmed Mean

Faruk Ahmed and Swagatam Das

***Abstract*—Suppression of impulse noise in images is an important problem in image processing. In this paper, we propose a novel adaptive iterative fuzzy filter for denoising images corrupted by impulse noise. It operates in two stages—detection of noisy pixels with an adaptive fuzzy detector followed by denoising using a weighted mean filter on the "good" pixels in the filter window. Experimental results demonstrate the algorithm to be superior to state-of-the-art filters. The filter is also shown to be robust to very high levels of noise, retrieving meaningful detail at noise levels as high as 97%.**

***Index Terms*—Alpha-trimmed mean, fuzzy filter, high-density impulse noise removal.**

I. Introduction

Digital images may be contaminated by salt-and-pepper noise due to factors such as imperfections in imaging sensors, channel transmission errors, nonideal medium between the scene and the imaging system (factors such as random scattering and absorption), and faulty memory locations in hardware [1].

For most image processing and analysis applications, it is highly desirable to remove impulse noise from the images. It is desired that while removing impulse noise from the image, there is a minimal loss of the useful image detail in the process.

A huge number of techniques have been proposed for solving this problem. Nonlinear filtering techniques, which are based on filters utilizing rank-order information of pixels in a window-wise fashion, are in general better performers than linear filtering techniques. The classical nonlinear filtering technique in the context of impulse noise removal in images is the standard median (SM) filter [2]. The SM filter is not good at preserving image detail, and moreover fails at high levels of corruption. To overcome these limitations, a number of modifications to the filter have been proposed over the years, including the weighted median filter (WMF) [3], [4] and the center-weighted median filter (CWMF) [5]. These filters give more weight to some pixels in the filtering window, but while being more detail preserving than the traditional median filter, they do not deal efficiently with higher levels of noise.

In later years, the single-stage filtering approach has evolved to a two-stage approach—the first step involves the detection of pixels corrupted by impulse noise, followed by the second step of denoising those pixels. The superiority of the switching filter over single-stage approaches was shown in [6]. Wang and Zhang, [7], demonstrated that a progressive switching filter is more effective than the single-iteration algorithms for images where the corruption level was high. In [8], Crnojevic *et al.* proposed using a robust estimator of variance, the median of absolute deviations from the median.

Manuscript received February 12, 2013; revised May 3, 2013 and July 26, 2013; accepted September 24, 2013. Date of publication October 21, 2013; date of current version October 2, 2014.

F. Ahmed is with the Department of Electronics and Commmunication Engineering, Institute of Engineering and Management, Kolkata 700091, India (e-mail: faruk.ahmed.91@gmail.com).

S. Das is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: swagatam.das@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2013.2286634

Two-staged soft computing approaches have been formulated as well. Russo *et al.* [9] developed a two-step fuzzy filter with better detail-preserving capabilities. Fuzzy inference rules by else-action filters were proposed in [10] to better maintain edge details. Choi *et al.* [11] proposed a technique where three filters (based on fuzzy least squares method) were combined based on a set of fuzzy rules. In [12], a switching median filter is proposed on a fuzzy-set framework. Schulte *et al.* introduced a two-stage nonlinear filtering technique based on fuzzy logic [13]. In [14], a filter based on adaptive neuro-fuzzy inference systems was proposed that was effective for the high levels of noise.

The fuzzy filters are usually simpler and quite efficient, especially when used in an adaptive setting. In [27], an adaptive fuzzy mean filter is proposed for impulse noise denoising based on the Cloud Model [15] that performs well at removing high levels of impulse noise. Apart from applications in image proccessing, there has been recent interest in applying fuzzy filters to more general signal processing problems as well [16]–[18].

II. Proposed Algorithm

We propose a two-stage iterative adaptive fuzzy filter for denoising images corrupted by impulse noise.

First, we briefly review the well-known *alpha(α)-trimmed mean filter* [32]. The α-trimmed mean is more effective as a measure of central tendency than the classical mean or median measures in the context of impulse noise removal [33]. We rewrite the measure in a form more relevant to our algorithm and call this the *mean of k-middle*.

The α-trimmed mean computes the mean of a set of elements after trimming the top and bottom $\alpha/2$ elements of the set. The α-trimmed mean of a set $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ of n elements is thus given by

$$\mu_\alpha = \frac{1}{n-\alpha} \sum_{i=(\alpha/2)+1}^{n-(\alpha/2)-1} a_i \tag{1}$$

where a_i is the ith order statistic of the elements of $\mathcal{A}$.

A. Mean of k-Middle

We define the *mean of k-middle*, denoted by $\mathcal{M}_k(\mathcal{A})$:

$$\mathcal{M}_k(A) = \begin{cases} \dfrac{1}{2k-1} \displaystyle\sum_{i=h-k+1}^{h+k-1} a_i, & \text{if } n \text{ is odd } (n = 2h-1) \\ \dfrac{1}{2k} \displaystyle\sum_{i=h-k+1}^{h+k} a_i, & \text{if } n \text{ is even } (n = 2h) \end{cases} \tag{2}$$

where a_i is the ith order statistic of the n elements of $\mathcal{A}$.

For $k = 1$, the measure is the same as the classical median, and for $k = h$, the measure takes the form of the mean.

B. Algorithm for Detection and Denoising of Noisy Pixels

Since we approach the particular problem of salt-and-pepper noise, if a pixel intensity is not fully dark or fully bright, we need not run further checks and its current intensity value is retained as it is. Otherwise, we need to check if it is a corrupt pixel.

For every pixel p_{ij} at location (i, j) in the image, we take a $(2M + 1) \times (2M + 1)$ neighborhood region with p_{ij} at the center. We call the

set of pixel intensities in this region R_{ij}^M

$$R_{ij}^M = \{p_{i+k,j+l} | -M \leq k,l \leq M\}. \quad (3)$$

Then, we define a fuzzy set $\mathcal{U}_{ij}^M$, where every element p_k in the set R_{ij}^M is associated with the following Gaussian membership function:

$$m_{\mathcal{U}_{i,j}^M}(p_k) = e^{-(p_k - \mu_{ij}^M)/2(\sigma_{ij}^M)^2}. \quad (4)$$

The parameters μ_{ij}^M and σ_{ij}^M for the membership function are calculated in the following way:

$$\mu_{ij}^M = \mathcal{M}_{K_1}(R_{ij}^M) \quad (5)$$

$$\sigma_{ij}^M = \mathcal{M}_{K_2}(\Lambda_{ij}^M) \quad (6)$$

where $\mathcal{M}_k$ is the mean of k-middle measure (2) as defined in the previous section and the set Λ_{ij}^M is defined as

$$\Lambda_{ij}^M = \{(p_k - \mu_{ij}^M)^2, \forall p_k \in R_{ij}^M\}. \quad (7)$$

The key stages of our algorithm are as follows.

1) *Initialization of parameters:* For denoising, we shall require N "good" pixels in a window. The number of noisy pixels being detected per iteration will be used for the stopping criterion of our iterative algorithm, and so we set a variable d_m to hold this value for the mth run. The window half-size parameter M is set initially to 1, which implies a window size of 3×3. $T_{\max}$ and $T_{\min}$ are the initial upper and lower bounds for T, an adaptive threshold for the Gaussian membership function.
2) *Detection of noisy and "good" pixels:* If the center pixel in a window p_{ij} is not at an extreme intensity, we retain its value. Otherwise, we need to determine whether p_{ij} is a corrupted pixel or not.

   a) Safe distance: We expect that pixels lying far away from p_{ij} do not have much influence on the intensity of p_{ij}, so we introduce a variable $S_{\max}$, which serves as an upper bound on M. As a last resort, we will increase our window size beyond the safe distance, but our priority is to avoid going beyond this size. This will ensure our preference for using nearer pixels for estimating denoised values as far as possible.

   b) Gaussian membership function: We compute the parameters μ_{ij}^M and σ_{ij}^M of the Gaussian membership function using (5)–(7). If it turns out that the deviation σ_{ij}^M is below a very low threshold (ϵ), i.e., the window consists of pixels with intensities very close to that of p_{ij}, then we simply set the value of p_{ij} to μ_{ij}^M. This also avoids division by zero errors that may arise if the region is of a uniform intensity which would result in σ_{ij}^M being 0. If the degree of membership of p_{ij}, $m_{\mathcal{U}_{ij}^M}(p_{ij})$, is above the threshold T, then p_{ij} is deemed to be uncorrupt, and we retain that value of p_{ij} (see Fig. 1).

   c) "Good" Set $\mathcal{G}$: If $m_{\mathcal{U}_{ij}^M}(p_{ij}) \leq T$, then p_{ij} is likely to be noisy. Hence, we need to estimate an intensity value from the surrounding pixels. For this, we need to develop a set $\mathcal{G} \subseteq R_{ij}^M$ of neighboring uncorrupted pixels. $\mathcal{G}$ is computedas follows:

$$\mathcal{G} = \{p_k | (m_{\mathcal{U}_{ij}^M}(p_k) > T) \vee (p_k \notin \{0,1\}), \forall p_k \in R_{ij}^M\} \quad (8)$$

   d) Adaptive slackening of parameters: If the cardinality of the set of "good" pixels $|\mathcal{G}|$ is below N, the minimum number of required "good" pixels, we slacken our initially demanding parameters. The choices of which parameters to slacken and in which order are guided by a set of rules. Our contention here is that when it comes to adaptive window sizing for denoising, it makes sense to stay close to the center pixel while searching for good pixels. Therefore, while slackening parameters, we first slacken the threshold $T_{\min}$ (with the window size fixed at 3×3). If such a slackening results in insufficient number of "good" pixels, we next enhance the window size, but with an upper bound that prevents us from going too far. If that is insufficient as well, we start to relax our demand for N "good" pixels, which reduces the number of samples from which we estimate the denoised intensity, but is still preferred to increasing the window size too far from the center pixel and thus using irrelevant samples. In the case, where we are still unable to find any "good" pixels, we are forced to increase the window size beyond the upper bound. Thus, we now enumerate the three slackening rules.

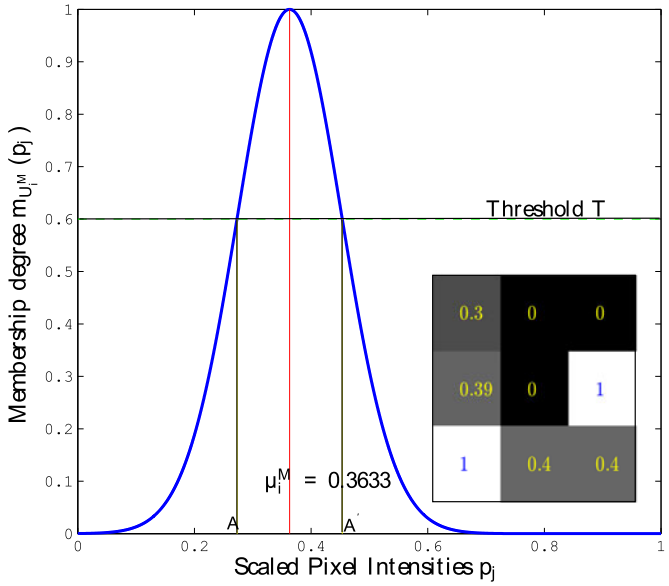

Fig. 1. Gaussian membership function for a sample 3×3 window (inset). Pixel intensity values lying within A and A' (with a Gaussian membership degree above T) or are not in $\{0, 1\}$ are "good" pixels.

Rule 1: If we are unable to find N uncorrupted pixels in the window, we first slacken the membership threshold T from its initial value of $T_{\max}$ down to $T_{\min}$ in a stepwise manner. (The stepsize α may be varied.) We stop if the criterion $|\mathcal{G}| \geq N$ is reached, as we then have our minimum number of "good" pixels required to perform denoising of p_{ij}.

Rule 2: If $|\mathcal{G}|$ is still less than N, we increase the window half-size parameter M such that the window size increases from 3×3 upto $(2S_{\max}+1) \times (2S_{\max}+1)$ in a stepwise manner, stopping if the criterion $|\mathcal{G}| \geq N$ is reached.

Rule 3: If $|\mathcal{G}|$ is still below N, we decrease our parameter N in a stepwise manner. If the iterative process results in N falling below 1, we increase $S_{\max}$ by unity, setting N permanently to 1 for denoising p_{ij}. We could alternatively set N back to 8, or some lower number, but the very fact that we are forced to go beyond $S_{\max}$ implies that there is considerable ambiguity in the current region either due to extremely high noise or the very nature of the image itself. Setting N higher will not be particularly helpful,

Algorithm 1 : Detection and Denoising of Noisy Pixels

procedure DENOISENOISYPIXELS(Image I)
1. **for** all pixels $p_{i,j}$ in I **do**
    1. **if** $p_{ij} \notin \{0, 1\}$ **then**
        1. retain value of p_{ij}
        2. **continue;**
    2. $T \leftarrow T_{init}$
    3. $N \leftarrow N_{init}$
    4. **while** true:
        1. Compute R_{ij}^M using Eq. (3).
        2. Compute μ_{ij}^M and σ_{ij}^M using Eqs. (5-7).
        3. **if** $\sigma_{ij}^M \leq \epsilon$ **then**
            1. $p_{ij} \leftarrow \mu_{ij}^M$
            2. **break;**
        4. **if** $m_{\mathcal{U}_{ij}^M}(p_{ij}) > T$ **then**
            1. retain value of p_{ij}
            2. **break;**
        5. Compute $\mathcal{G}$ using Eq. (7).
        6. $\eta \leftarrow |\mathcal{G}|$.
        7. **if** $\eta < N$ **AND** $T > T_{min}$ **then**
            1. $T \leftarrow T - \alpha$
            2. **continue;**
        8. **if** $\eta < N$ **AND** $M < S_{max}$ **then**
            1. $M \leftarrow M + 1$
            2. **continue;**
        9. **if** $\eta < N$ **AND** $M \geq S_{max}$ **AND** $T \leq T_{min}$ **then**
            1. $N \leftarrow N - 1$
            2. **if** $N \leq 1$ **then**
                1. $S_{max} \leftarrow S_{max} + 1$
                2. $N \leftarrow 1$
        10. $d_m \leftarrow d_m + 1$
        11. Compute denoised pixel intensity of p_{ij} from $\mathcal{G}$ using Eqs. (9-11)
        12. **break;**
    end while
end for
end procedure

and may lead to further increase in $S_{\max}$ which would result in pixels farther away exerting undue influence upon denoising p_{ij}.

3) *Denoising noisy pixels:* We estimate the denoised pixel intensity from the pixel intensities in the "good" set $\mathcal{G}$. We weight the elements of $\mathcal{G}$ by a simple inverse distance weighting function that assigns lower weights to pixels far from p_{ij} and higher weights to pixels that are closer. The weight w_k for a "good" pixel p_k at location $(i^{'}, j^{'})$ in the image is given by

$$w_k = \frac{1}{((i - i')^2 + (j - j')^2)^p} \tag{9}$$

where p is a tunable parameter. We empirically find that $p = 2$ works best overall. The weights w_k are normalized so that they sum to unity.

Thus, the denoised pixel intensity p_{ij}^{new} is computed by the formula

$$p_{ij}^{\text{new}} = \frac{1}{W} \sum_{\forall p_k \in \mathcal{G}} w_k p_k \tag{10}$$

where W is a normalizing term, given as

$$W = \sum_{k=1}^{|\mathcal{G}|} w_k . \tag{11}$$

We describe the entire procedure in a pseudocode in Algorithm 1.

4) *Iterations and stopping criterion:* Algorithm I is run iteratively, with the following stopping criterion. We find the difference of the number of noisy pixel detections d_m in succeeding iterations of the algorithm. We terminate iterations when d_m falls below a small threshold. We set the threshold at 0.05% of the total number of pixels in the image.

III. RESULTS

A. Effect of Varying Parameters K_1 and K_2

We first evaluate the performance of our algorithm with variation in the values of the parameters K_1 and K_2 for different noise levels. The selected test images are *Baboon (Mandrill)*, *Lena*, *Peppers,* and *Bridge*. We run our algorithm setting values of K_1 and K_2 to 1, 2, 3, 4, or h [see (2)]. For every test image, we perform the experiment for three levels of corruption by salt-and-pepper noise: 10%, 50%, and 90%. All the values reported in Table I are averages of ten trials. From the experiments, it is observed that:

1) For high noise, $K_1 = h$ seems a good choice, and for low noise, $K_1 = 1$ gives good results.
2) $K_1 = 1$ (median measure) leads to bad performance for corruption by high noise levels. It follows from the fact that the median is a bad estimator of denoised pixel intensity value in the case of high noise.
3) Overall, it seems that varying K_2 does not have much effect, however, values of K_2 lying within 4 provide better results than for $K_2 = h$ (mean measure), except in the case of *Lena*.

In general, we observe that the best selections of the parameters K_1 and K_2 are highly dependent on the specific characteristics of the image. We note that the parameter settings of $K_1 = 3$ and $K_2 = 3$ might be used to achieve good performance overall, and for a comparative analysis, we use this setting.

B. Effect of $T_{\min}$, $T_{\max}$, and $S_{\max}$

In Table II, we report the average PSNR values for a set of nine standard images at varying values of $S_{\max}$, $T_{\max}$, and $T_{\min}$. Every image was corrupted with three levels of noise—10%, 50%, and 90%, and results were averaged for all noise levels over ten trials.

From these experiments, we observe that the best settings are $S_{\max} = 2$, $T_{\min} = 0.8$, and $T_{\max} = 0.999$. We also note that increasing $S_{\max}$ results in poorer performance, which is confirms that increasing window size results in poor performance.

C. Selecting N_{init}

N_{init} is the initial number of "good" pixels to look for in the window. From Table III, (all values are averages of ten trials), we observe that

1) In general, increasing the value of N_{init} leads to poorer image denoising, where the noise is low (20%).
2) Conversely, higher values of N_{init} results in better performance for cases when the noise is high (80%).
3) The runtime also increases with increase in the value of N_{init}. This is because a higher number of "good" pixels are required per pixel denoising, and consequently Algorithm I takes longer to run.

We choose $N_{\text{init}} = 1$ for a comparative analysis of denoising performance and run time.

D. Comparison With Other State-of-the Art Filters

Table IV shows how our algorithm compares with other state-of-the-art algorithms—*Adaptive median with edge preserving regularization*

TABLE I
PSNR (dB) FOR THE VARYING LEVELS OF NOISE WITH RESPECT TO DIFFERENT PARAMETER SETTINGS OF K_1 AND K_2

(a) Baboon, 10% noise

$K_1\backslash K_2$	1	2	3	4	h
1	32.81	32.82	32.76	32.75	32.81
2	32.82	32.85	**32.88**	32.84	32.86
3	32.75	32.79	32.74	32.77	32.80
4	32.66	32.61	32.65	32.62	32.71
h	32.60	32.60	32.60	32.57	32.65

(b) Baboon, 50% noise

$K_1\backslash K_2$	1	2	3	4	h
1	24.74	24.22	24.05	24.03	23.92
2	24.84	24.85	24.62	24.60	24.27
3	24.85	24.82	**24.88**	24.81	24.42
4	24.83	24.85	24.84	24.84	24.61
h	24.82	24.84	23.86	24.87	24.61

(c) Baboon, 90% noise

$K_1\backslash K_2$	1	2	3	4	h
1	07.60	08.39	08.68	08.71	08.40
2	19.58	**19.76**	12.61	11.20	09.03
3	19.47	19.43	19.72	13.77	10.17
4	19.15	19.10	19.14	19.69	10.70
h	19.64	19.63	19.64	19.66	19.67

(d) Lena, 10% noise

$K_1\backslash K_2$	1	2	3	4	h
1	42.87	42.81	42.82	42.74	42.99
2	42.83	42.81	42.79	42.86	43.00
3	42.82	42.81	42.82	42.70	43.05
4	42.64	42.72	42.75	42.80	43.04
h	42.30	42.42	42.41	42.50	**43.07**

(e) Lena, 50% noise

$K_1\backslash K_2$	1	2	3	4	h
1	33.40	30.38	29.74	29.58	28.99
2	34.08	34.13	32.64	32.11	30.59
3	34.03	34.09	34.18	33.75	30.94
4	34.01	34.08	34.03	34.16	31.88
h	34.11	34.16	34.14	**34.19**	34.16

(f) Lena. 90% noise

$K_1\backslash K_2$	1	2	3	4	h
1	07.58	08.40	08.70	08.78	08.54
2	26.45	26.60	13.12	11.63	09.22
3	25.31	25.33	**26.80**	14.44	10.58
4	24.43	24.43	24.47	26.79	11.09
h	26.76	26.65	26.73	26.75	26.73

(g) Peppers, 10% noise

$K_1\backslash K_2$	1	2	3	4	h
1	**41.47**	41.13	41.12	41.02	41.05
2	41.45	41.30	41.33	41.37	41.23
3	41.30	41.24	41.24	41.31	41.16
4	41.26	41.20	41.24	41.22	41.14
h	40.74	40.86	40.92	40.84	41.19

(h) Peppers, 50% noise

$K_1\backslash K_2$	1	2	3	4	h
1	32.35	29.68	29.10	28.84	28.37
2	32.44	32.52	31.41	31.17	30.02
3	32.53	32.50	32.59	32.13	30.24
4	32.55	32.56	32.51	32.64	31.14
h	32.56	32.61	32.56	32.63	**32.69**

(i) Peppers, 90% noise

$K_1\backslash K_2$	1	2	3	4	h
1	07.45	08.31	08.60	08.65	08.40
2	23.61	24.12	13.07	11.48	09.11
3	23.95	24.23	25.09	14.05	10.49
4	24.22	24.11	24.31	25.44	11.09
h	25.80	25.88	**25.91**	25.82	25.79

(j) Bridge, 10% noise

$K_1\backslash K_2$	1	2	3	4	h
1	**35.45**	35.37	35.36	35.41	35.43
2	35.29	35.26	35.18	35.29	35.36
3	35.18	35.17	35.08	34.91	35.27
4	34.96	34.95	34.92	34.24	34.80
h	33.87	33.82	33.90	33.87	33.63

(k) Bridge, 50% noise

$K_1\backslash K_2$	1	2	3	4	h
1	27.13	26.24	25.94	25.80	25.57
2	27.24	**27.29**	26.90	26.80	26.17
3	27.24	27.23	27.24	27.14	26.39
4	27.28	27.28	27.26	27.13	26.70
h	27.01	27.03	27.00	27.00	27.16

(l) Bridge, 90% noise

$K_1\backslash K_2$	1	2	3	4	h
1	7.33	8.15	8.49	8.48	8.26
2	20.16	20.75	12.69	11.09	8.98
3	20.15	20.17	21.14	13.52	10.21
4	20.33	20.34	20.39	21.24	10.76
h	**21.36**	21.25	21.34	21.33	21.33

TABLE II
VARYING $S_{\max}$, $T_{\max}$, AND $T_{\min}$. N_{init} IS FIXED AT 8 TO ENSURE THAT THE ADAPTIVE STAGES OF THE ALGORITHM ARE RUN MULTIPLE TIMES

$T_{min}\backslash T_{max}$	0.2	0.4	0.6	0.8	0.999
0.2	-	24.6129	24.6168	24.6119	24.6390
0.4	-	-	27.4616	27.3907	27.5251
0.6	-	-	-	29.8964	29.8985
0.8	-	-	-	-	**29.9178**

$T_{min}\backslash T_{max}$	0.2	0.4	0.6	0.8	0.999
0.2	-	24.4535	24.4131	24.4502	24.4813
0.4	-	-	26.9924	26.8205	27.0244
0.6	-	-	-	29.8967	29.8852
0.8	-	-	-	-	29.8777

$T_{min}\backslash T_{max}$	0.2	0.4	0.6	0.8	0.999
0.2	-	24.4029	24.3932	24.3863	24.4316
0.4	-	-	26.6629	26.5822	26.6450
0.6	-	-	-	29.8282	29.7829
0.8	-	-	-	-	29.8132

$T_{min}\backslash T_{max}$	0.2	0.4	0.6	0.8	0.999
0.2	-	24.3800	24.3706	24.4102	24.4035
0.4	-	-	26.4991	26.5005	26.6105
0.6	-	-	-	29.7522	29.7995
0.8	-	-	-	-	29.7551

N_{init} is fixed at 8 to ensure that the adaptive stages of the algorithm are run multiple times.

(AM-EPR) [19], *boundary discriminative noise detection (BDND)* method [20], *fast median (FM)* method [21], *wavelet neural network (WNN)* method [22], a filter using a *pixel-wise S-estimate of variance (PWS)* [23], a *contrast enhancement-based filter (CEF)* [24], *spatially-adaptive total variation* filter [25], a *patch-based* method [26], and the *cloud model (CM) filter* [27] In Figs. 2 and 3, we show qualitative performance of our algorithm on grayscale images.

The FM filter considers all pixels with an intensity of 0 and 255 in a fixed-size 3×3 detection window as noisy pixels. It then uses median values or the left neighborhood values to replace the noise pixels. The WNN filter uses a wavelet neural network for the detection of noisy pixels. For denoising, a coefficient (C_{ij}) is calculated, which is a measure of the confidence with which the pixel is deemed to be noisy. Finally, the pixel is replaced with a linear combination of the original value and the median value, where the weights for each term are C_{ij} and $(1 - C_{ij})$. In the CEF filter, first a nonlinear transform is carried out on pixel intensities within a window (of size $m \times m$). This is followed by a detection step, where the p lowest transformed values are summed and thresholded by T to produce a binary flag indicating if the pixel is noisy, followed by a filtering step with a WMF. For a comparative analysis, we set parameters: $m = 5, p = 13,$ and $T = 35$. In the PWS filter, the detection step uses an affine classifier (parameters set according to the recommendations by the authors, i.e. $\delta = 12$, and $s^{(0)} = 2.6, s^{(k+1)} = s^{(k)} - 0.3$). The authors have varied the window size K according to the noise level; however, we have fixed it to 5×5 for comparison. An edge-preserving potential function ($\psi(x) = |x|^{\alpha}$) is used for denoising, with $\alpha = 1.3$. The AM-EPR filter is a combination of the AM filter [28] with a variational method [29] and exhibits better detail preservation than the AM filter. As recommended by the authors, $W_{\max}$ is set to 39, α is set to 1.3, and β is set to 5. The PB filter is a patch-based filter that relies upon using information from similar patches in an image. The

TABLE III
PSNR AND RUNTIME FOR A SET OF NINE IMAGES WITH VARYING VALUES OF N_{init}

	N_{init}	20% noise		50% noise		80% noise	
		PSNR(dB)	Time(Sec)	PSNR(dB)	Time(Sec)	PSNR(dB)	Time(Sec)
Baboon	1	29.75	5.25	24.84	12.76	20.73	28.05
	4	29.70	5.59	24.87	23.60	21.09	69.49
	8	29.57	13.73	24.88	40.62	21.14	110.10
Barbara	1	31.95	4.86	26.74	12.58	22.78	28.31
	4	31.68	4.97	26.83	23.11	23.10	69.27
	8	31.52	11.81	26.85	39.75	23.18	110.17
Boat	1	36.03	4.77	30.69	12.67	25.88	28.04
	4	36.01	4.88	30.71	23.05	26.12	69.10
	8	35.74	11.32	30.68	39.56	26.13	110.31
Bridge	1	31.59	8.99	27.01	16.88	22.92	32.88
	4	32.11	10.22	27.27	28.35	23.03	74.62
	8	32.13	16.95	27.54	44.86	23.15	115.25
Fingerprint	1	35.71	5.25	28.97	12.78	22.85	28.36
	4	35.61	5.45	29.12	23.55	23.16	69.77
	8	35.06	14.08	29.16	40.80	23.21	110.77
Flintstones	1	32.39	4.79	26.41	12.45	20.79	28.04
	4	32.32	4.96	26.38	22.36	20.75	66.46
	8	31.89	11.36	26.36	37.09	20.67	102.52
House	1	38.97	1.09	33.19	2.98	27.82	6.70
	4	38.88	1.12	33.28	5.49	28.11	16.21
	8	38.57	2.39	33.35	9.19	28.23	25.48
Lena	1	39.92	4.61	34.10	12.65	28.84	28.41
	4	39.80	4.75	34.19	23.10	29.19	69.29
	8	39.40	10.47	34.24	39.59	29.31	109.98
Peppers	1	37.99	4.60	32.34	12.58	27.54	28.07
	4	37.86	4.84	32.56	22.93	27.87	68.76
	8	37.83	10.68	32.59	39.41	27.98	108.76

(a) (b) (c) (d) (e) (f)

Fig. 2. Denoising grayscale *Barbara* and *Boat* from 20% and 80% salt-and-pepper corruption, respectively. (a) Barbara. (b) 20% noise. (c) Result (d) Boat. (e) 80% noise. (f) Result.

TABLE IV
COMPARISON OF THE PERFORMANCE OF OUR ALGORITHM WITH SOME STATE-OF-THE-ART FILTERS IN TERMS OF PSNR (IN dB)

		FM	CEF	PWS	WAV	AM-EPR	PB	BDND	CM	SA-TV	Our Algorithm
Lena	20%	37.05	37.46	36.85	38.12	38.21	38.31	38.52	39.42	39.20	**39.92**
	50%	29.81	30.71	29.57	30.27	33.46	32.04	32.74	33.57	33.88	**34.10**
	80%	23.11	23.22	22.68	23.44	27.16	25.97	27.11	28.45	27.14	**28.84**
Bridge	20%	30.41	28.47	29.18	29.38	32.12	27.53	30.66	31.35	**31.98**	31.59
	50%	24.24	22.35	22.79	24.72	26.64	23.75	25.22	26.52	26.89	**27.01**
	80%	20.67	19.52	20.03	20.01	21.98	19.45	21.39	22.28	22.41	**22.92**
Peppers	20%	36.21	35.03	35.46	37.27	37.45	36.32	34.44	37.54	36.87	**37.99**
	50%	29.53	30.38	29.26	30.49	31.25	29.25	30.23	32.03	31.62	**32.34**
	80%	22.21	23.65	22.84	22.95	27.32	25.64	26.61	27.46	26.42	**27.54**
Baboon	20%	27.22	26.85	26.52	27.49	29.87	24.22	27.73	28.47	28.49	**29.75**
	50%	22.26	21.93	20.42	22.31	24.52	21.27	23.46	24.05	23.91	**24.84**
	80%	18.69	17.60	17.86	18.35	19.73	17.38	19.92	20.36	20.59	**20.73**
Barbara	20%	29.46	29.58	28.72	30.14	29.72	29.24	29.85	30.78	30.70	**31.95**
	50%	23.46	23.37	22.69	24.84	25.33	23.49	25.17	26.10	25.91	**26.74**
	80%	19.35	19.31	18.91	19.96	21.41	21.64	21.74	22.54	22.66	**22.78**
Boat	20%	34.75	30.87	33.78	34.53	34.89	32.73	34.83	35.31	35.97	**36.03**
	50%	27.96	25.65	26.80	27.34	29.34	27.83	29.68	29.99	30.38	**30.69**
	80%	23.65	21.46	22.50	22.87	24.75	22.29	24.93	25.58	25.18	**25.88**
Fingerprint	20%	34.14	33.89	33.68	34.18	33.58	29.50	33.26	34.85	34.53	**35.71**
	50%	25.44	24.25	24.79	25.32	27.70	23.36	27.07	28.31	27.80	**28.97**
	80%	19.82	18.54	19.08	19.47	21.49	18.75	21.48	22.39	20.39	**22.85**
Flintstones	20%	30.85	31.13	30.48	31.25	31.38	31.14	31.26	32.00	32.19	**32.39**
	50%	23.53	23.18	22.99	23.90	25.62	21.92	25.39	26.03	25.70	**26.41**
	80%	17.75	17.66	17.39	18.33	19.20	18.39	19.71	20.56	19.54	**20.79**
House	20%	37.84	37.91	37.74	38.15	37.42	37.45	37.23	38.32	38.66	**38.97**
	50%	29.45	29.52	29.49	29.94	31.51	30.70	31.72	32.45	32.53	**33.19**
	80%	22.65	22.63	22.54	22.88	25.45	24.84	25.81	27.52	26.16	**27.82**

TABLE V
COMPARISON OF THE MEAN RUN TIME OF OUR ALGORITHM WITH STATE-OF-THE-ART FILTERS IN SECONDS FOR THE SET OF IMAGES USED IN TABLE IV

	FM	CEF	PWS	WAV	AM-EPR	PB	BDND	CM	SA-TV	Our Algorithm
20%	2.31	10.34	18.52	30.96	1826.75	360.38	226.83	10.32	13.60	4.91
50%	2.31	18.59	26.48	56.37	3957.94	895.72	219.36	12.58	23.96	12.04
80%	2.31	37.09	34.63	72.51	6486.45	1804.51	220.47	17.02	28.78	26.32

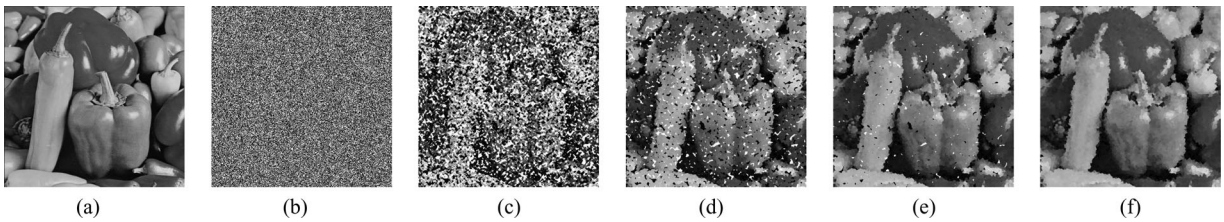

Fig. 3. Denoising *Peppers* corrupted with a high 97% level of impulse noise. The final PSNR (iteration 12) is 21.95dB. (a) Peppers. (b) 97% noise. (c) Iteration 1. (d) Iteration 3. (e) Iteration 5. (f) Iteration 12.

Fig. 4. Results of applying our algorithm to the separate color channels (RGB) of noisy color images and recombining them. (a) Lena. (b) 50% noise. (c) Result. (d) Baboon. (e) 80% noise. (f) Result.

filter uses rank ordered absolute differences (ROAD) to estimate noise ratio. The parameter settings are set to the values recommended by the authors: ROAD threshold (τ) = 70, half-size of patches = 3, half-size of research neighborhood = 7, and iterations = 2. The BDND filter uses two fixed size windows (of sizes 3 × 3 and 21 × 21) for the detection of noisy pixels, and an adaptive window for noise removal. The CM filter uses an uncertainty-based detector to detect noisy pixels, followed by a fuzzy mean filter to suppress noise. The parameter δ, in the CM filter is the minimum number of "good" pixels, similar to the parameter η in our algorithm. For comparison purposes, we fixed the value of δ to 1 in our trials, as recommended by the author. The SA-TV algorithm is based on the iteratively reweighted norm (IRN) [30] [31] method and solves the *l*-1 total variation problem for noisy pixels. The parameters are set according to the recommendations by the authors. We used the code supplied by the authors at http://www.mathworks.com/matlabcentral/fileexchange/37161-spatially-adaptive-irn-algorithm for simulating this algorithm.

The test images are *Lena*, *Bridge*, *Peppers*, *Baboon*, *Barbara*, *Boat*, *Fingerprint*, *Flintstones*, and *House*. The performance measure is the peak-signal-to-noise ratio (PSNR), defined for the restored image (I_r) with respect to the original image (I_o) as

$$\text{PSNR}(I_o, I_r) = 10\log_{10} \frac{255^2}{\frac{1}{MN}\sum_{i,j}(I_r(i,j) - I_o(i,j))^2} \tag{12}$$

where 255 is the maximum pixel intensity for 8-bit images.

The experiments were carried out on a system with an Intel Core i7 processor at 3.2 GHz, equipped with 24GB RAM, running MATLAB 2012a.

TABLE VI
NUMBER OF ITERATIONS REQUIRED FOR DIFFERENT IMAGES AT DIFFERENT NOISE LEVELS

	20%	50%	80%		20%	50%	80%
Lena	2	3	5	Boat	2	3	5
Bridge	8	9	10	Fingerprint	2	3	5
Peppers	2	3	5	Flintstones	2	5	9
Baboon	2	3	5	House	2	3	4
Barbara	2	3	5				

TABLE VII
PSNR (IN DB) FOR DENOISING COLOR IMAGES CORRUPTED BY IMPULSE NOISE OF LEVELS 20%, 50%, AND 80%

	20%	50%	80%
Lena	35.33	29.58	23.27
Peppers	30.36	26.89	21.81
Baboon	26.13	21.34	17.69

Settings for K_1 and K_2 were kept at $K_1 = K_2 = 3$.

E. Number of Iterations

Table VI shows the number of iterations required per image for varying the levels of noise. In general, a more noisy image requires more iterations, but characteristics particular to specific images play an important role.

F. Denoising Color Images

Our algorithm may be extended for use in color images as well. We apply it to the three color channels—Red, Green, and Blue—separately,

and then combine the results for each of the channels to acquire the final denoised image. In Table V, we tablulate the performance of our algorithm for denoising color images in terms of PSNR (dB). For three-channel color images, the PSNR is given by

$$10\log_{10}\frac{255^2}{\frac{1}{3MN}\sum_{c\in\{\mathrm{R,G,B}\}}\sum_{i,j}(I_r^c(i,j)-I_o^c(i,j))^2}. \quad (13)$$

We show visual results for Lena and Baboon in Fig. 4.

IV. Conclusion

In this paper, we presented a novel two-stage filter for denoising images corrupted with salt-and-pepper noise. In the first stage, an adaptive fuzzy filter is used for the detection of noisy pixels. In the second stage, denoising is performed on noisy pixels, by performing a weighted mean filtering operation on nearby uncorrupt pixels. Experimental results show that the proposed filter is superior to state-of-the-art filters, and moreover, can restore meaningful image detail at levels of corruption as high as 97%.

References

[1] A. Bovik, *Handbook of Image and Video Processing*. New York, NY, USA: Academic, 2000.

[2] T. A. Nodes and N. C. Gallagher, "Median filters: Some modifications and their properties," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-30, no. 5, pp. 739–746, Oct. 1982.

[3] O. Yli-Harja, J. Astola, and Y. Neuvo, "Analysis of the properties of median and weighted median filters using threshold logic and stack filter representation," *IEEE Trans. Signal Process.*, vol. 39, no. 2, pp. 395–410, Feb. 1991.

[4] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: A tutorial," *IEEE Trans. Circuits Syst. II*, vol. 43, no. 3, pp. 157–192, Mar. 1996.

[5] S. J. Ko and Y. H. Lee, "Center weighted median filters and their applications to image enhancement," *IEEE Trans. Circuits Syst.*, vol. 38, no. 9, pp. 984–993, Sep. 1991.

[6] T. Sun and Y. Neuvo, "Detail-preserving median based filters in image processing," *Pattern Recognit. Lett.*, vol. 15, pp. 341–347, 1994.

[7] Z. Wang and D. Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 1, pp. 78–80, Jan. 1999.

[8] V. Crnojevic, V. Senk, and Z. Trpovski, "Advanced impulse detection based on pixel-wise MAD," *IEEE Signal Process. Lett.*, vol. 11, no. 7, pp. 589–592, Jul. 2004.

[9] F. Russo and G. Ramponi, "A fuzzy filter for images corrupted by impulse noise," *IEEE Signal Process. Lett.*, vol. 3, no. 6, pp. 168–170, Jun. 1996.

[10] F. Russo, "FIRE operators for image processing," *Fuzzy Sets Syst.*, vol. 103, no. 2, pp. 265–275, 1999.

[11] Y. S. Choi and R. Krishnapuram, "A robust approach to image enhancement based on fuzzy logic," *IEEE Trans. Image Process.*, vol. 6, no. 6, pp. 808–825, Jun. 1997.

[12] H. L. Eng and K. K. Ma, "Noise adaptive soft-switching median filter," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 242–251, Feb. 2001.

[13] S. Schulte, M. Nachtegael, V. De Witte, D. Van der Weken, and E. E. Kerre, "A fuzzy impulse noise detection and reduction method," *IEEE Trans. Image Process.*, vol. 15, no. 5, pp. 1153–1162, May 2006.

[14] E. Beşdok, P. Çivicioğlu, and M. Alçı, "Using an adaptive neuro-fuzzy inference system based interpolant for impulsive noise suppression from highly distorted images," *Fuzzy Sets Syst.*, vol. 150, no. 3, pp. 525–543, 2005.

[15] H. J. Wang and Y. Deng, "Spatial clustering method based on cloud model," in *Proc. IEEE Int. Conf. Fuzzy Syst. Knowl. Discov.*, Aug. 2007, vol. 2, pp. 272–276.

[16] H. Zhang, H. Zhong, and C. Dang, "Delay-dependent decentralized H_∞ filtering for discrete-time nonlinear interconnected systems with time-varying delay based on the TS fuzzy model," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 3, pp. 431–443, Jun. 2012.

[17] B.-S. Chen, W.-H. Chen, and W. Zhang, "Robust filter for nonlinear stochastic partial differential systems in sensor signal processing: Fuzzy approach," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 5, pp. 957–970, Oct. 2012.

[18] X.-J. Li and G.-H. Yang, "Switching-type H∞ filter design for TS fuzzy systems with unknown or partially unknown membership functions," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 2, pp. 385–392, Apr. 2013.

[19] R. H. Chan, C.-W. Ho, and M. Nikolova, "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization," *IEEE Trans. Image Process.*, vol. 14, no. 10, pp. 1479–1485, Oct. 2005.

[20] P.-E. Ng and K.-K. Ma, "A switching median filter with boundary discriminative noise detection for extremely corrupted images," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1506–1516, Jun. 2006.

[21] K. S. Srinivasan, D. Ebenezer, "A new fast and efficient decision-based algorithm for removal of high-density impulse noises," *IEEE Signal Process. Lett.*, vol. 14, no. 3, pp. 189–192, Mar. 2007.

[22] C. Deng and J. Y. An, "An impulse noise removal based on a wavelet neural network," in *Proc. 2nd Int. Conf. Inf. Comput. Sci.*, May 21–22, 2009, vol. 2, pp. 71–74.

[23] V. Crnojević and P. NemanjaI, "Impulse noise filtering using robust pixel-wise S-estimate of variance," *EURASIP J. Adv. Signal Process.*, Feb. 2010, DOI: http://dx.doi.org/10.1155/2010/830702.

[24] U. Ghanekar, A. K. Singh, and R. Pandey, "A contrast enhancement-based filter for removal of random valued impulse noise," *IEEE Signal Process. Lett.*, vol. 17, no. 1, pp. 47–50, Jan. 2010.

[25] R. Rojas and P. Rodrguez, "Spatially adaptive total variation image denoising under salt and pepper noise," in *Proc. the Eur. Signal Process. Conf.*, Barcelona, Spain, Aug. 2011, pp. 278–282.

[26] J. Delon and A. Desolneux, "A patch-based approach for random-valued impulse noise removal," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 25–30, 2012, pp. 1093–1096.

[27] Z. Zhou, "Cognition and removal of impulse noise with uncertainty," *IEEE Trans. Image Process.*, vol. 21, no. 7, pp. 3157–3167, Jul. 2012.

[28] H. Hwang and R. A. Haddad, "Adaptive median filters: New algorithms and results," *IEEE Trans. Image Process.*, vol. 4, no. 4, pp. 499–502, Apr. 1995.

[29] M. Nikolova, "A variational approach to remove outliers and impulse noise," *J. Math. Imag. Vis.*, vol. 20, no. 1/2, pp. 99–120, 2004.

[30] P. Rodriguez and B. Wohlberg, "Efficient minimization method for a generalized total variation functional," *IEEE Trans. Image Process.*, vol. 18, no. 2, pp. 322–332, 2009.

[31] P. Rodriguez and B. Wohlberg, "A generalized vector-valued total variation algorithm," in *Proc. IEEE Int. Conf. Image Process.*, Nov. 2009, pp. 1309–1312.

[32] J. Bednar and T. Watt, "Alpha-trimmed means and their relationship to median filters," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 32, no. 1, pp. 145–153, Feb. 1984.

[33] R. Oten and R. J. P de Figueiredo, "Adaptive alpha-trimmed mean filters under deviations from assumed noise model," *IEEE Trans. Image Process.*, vol. 13, no. 5, pp. 627–639, May 2004.