Gregori Gregori, V.; Morillas, S.; Roig, B.; Sapena Piera, A. (2018). Fuzzy averaging filter for impulse noise reduction in colour images with a correction step. Journal of Visual Communication and Image Representation. 55:518-528. doi:10.1016/j.jvcir.2018.06.025

Fuzzy Averaging Filter For Impulse Noise Reduction In Colour Images With A Correction Step

Valentín Gregori[a], Samuel Morillas[b], Bernardino Roig[a], Almanzor Sapena[a]

[a] *Instituto de Investigación para la Gestión Integrada de Zonas Costeras, Universitat Politècnica de València, Campus de Gandia, Spain*
[b] *Instituto Universitario de Matemática Pura y Aplicada, Universitat Politècnica de València, Campus de Gandia, Spain*

Abstract

In this paper we propose a fuzzy detection and reduction method for impulse noise in colour images. Detection is based on the fuzzyfication of a well-known statistic called ROD. The noise degrees obtained are used to reduce impulses by employing a fuzzy averaging between the input colour vector and a robust estimate of noise-free colour vector within the input neighbourhood. Fuzzy averaging has some advantages in terms of both noise reduction and detail preservation in front of detect and replace approaches because of threshold based decisions of the latter. However, robustness of the former is lower. We solve this problem by including a correction mechanism that checks the fuzzy noise degree of the output and replaces it with a robust colour vector either when noise has not been properly reduced or when a colour artefact has been introduced. We carry out a thorough study of the method parameter setting and give a convenient and robust setting. Experimental results show that our approach is very robust in front of four different types of impulse noise.

Keywords: Color Image Filter, Correction Step, Fuzzy Filter, Impulse Noise
2010 MSC: 68U10, 94A08, 94D05

*Corresponding author: smorillas@mat.upv.es

1. Introduction

Recently, detect and replace approaches, also called switching filters, have become the most popular techniques to reduce impulse noise in colour images because of their simplicity, computational efficiency and high performance [1]-[12]. The most advanced approaches in this family have evolved to a little less efficient methods that try to improve performance by adapting the local region under processing [13, 14]. All these methods use a crisp threshold based decision and their performance is critically influenced by the threshold setting. Furthermore, as it is common in crisp methods, even in optimal setting the decision may be inappropriate near the threshold: either because noise may not be detected or because small image details could be smoothed.

The use of a fuzzy approach can solve these drawbacks. Indeed, because of fuzzy models capability to deal with uncertainty and nonlinearity in color images, many fuzzy methods have been proposed in the literature either to find robust filtering methods or in the detection and decision steps of switching methods [15, 16, 17]. But no fuzzy restoration step has been introduced for impulse noise removal in color images. Only a few approaches for gray-scale images have been published [18, 19, 20, 21, 22, 23] but no one has been properly extended to colour images. [24] is a variant of the hard threshold method [25] where the statistic used in the threshold condition is fuzzified and later uses two thresholds to replace the pixel, keep it as it is or perform an intermediate operation. [26] is a component-wise application of a fuzzy median filter to colour images but it is well known that this is not a good option for filtering colour images.

Also, other important families of filters are the Total Variation filters [27, 28] and, in particular, the machine learning and deep learning approaches [29, 30, 31, 33]. Among the latter, we can find the use of deep convolution neural networks (CNN) for image restoration in color images [29], the combination of deep CNN and residual learning for Gaussian noise removal, the application of semi-supervised learning on big image data for impulse noise removal in gray-

scale images [13, 32], and a new strategy for building adaptive neuro-fuzzy systems for impulse noise removal [33].

The advantage of the proposed fuzzy soft-switching filtering in front of crisp switching approaches lies in two points: (i) The sensitivity of parameters setting is lower than in crisp methods. We will see that for the method that we propose is pretty easy to find a general parameter setting able to appropriately process any image contaminated with a variety of noise types and intensities; (ii) The intensity of the reduction operation adapts to the pixel under processing characteristics which makes the filter effective for different image noise types and intensities and able to yield a high performance in many different situations.

However, fuzzy averaging has a serious disadvantage which regards its robustness: since it combines different colour vectors which may belong to different image regions, the result of the fuzzy averaging may be a colour which is different from both regions, generating a colour artefact or a less noisy pixel but far from a noise-free one. This is the reason why gray-scale soft switching filters have not been properly extended to colour. To solve this issue, we apply a correction mechanism after obtaining the output from the fuzzy averaging: if the noise degree of the output has not improved significantly with respect to the input, we consider that it is because of the lack of robustness of the fuzzy averaging and, consequently, we disregard the result and we replace the input with a robust estimate of noise-free colour vector in the neighbourhood of the input.

The structure of the paper is as follows. In the following section we detail the proposed filter: in Section 2.1 we describe the fuzzy noise detection based on the fuzzyfication of ROD statistic [34]; in Section 2.2 we explain the restoration employing fuzzy averaging between input and robust estimator of noise-free colour vector; in Section 2.3 we propose the correction mechanism to solve the lack of robustness of the averaging; computational complexity is studied in Section 2.4, and parameter setting in Section 2.5. Section 3 shows the experimental results and comparisons and Section 4 presents the conclusions.

2. Fuzzy Averaging Filter

Denote by $\mathcal{F}$ a colour image to be processed, and let W be a sliding filtering window, of size $n \times n$ ($n = 3, 5, \ldots$), centered at the pixel $\mathbf{F}_0$ under processing. The colour vectors in $\mathcal{F}$ are denoted as $\mathbf{F}_i = (F_i^R, F_i^G, F_i^B)$, as usual in the RGB colour space. The details of the proposed method that we call *Corrected Fuzzy Averaging Filter* (CFAF) are given in the following.

2.1. Noisiness of image pixels

First, we are interested in evaluating how noisy is $\mathbf{F}_0$. For this, we use the noise detector in [34]. We assign a certainty degree $\delta(\mathbf{F}_0)$ for the vague statement "$\mathbf{F}_0$ *is noisy*" as follows.

We order the pixels $\mathbf{F}_i$ in a window W' centered at $\mathbf{F}_0$ that is also taken, for simplicity, of size $n \times n$ in the way $\mathbf{F}_{(0)}, \mathbf{F}_{(1)}, \ldots, \mathbf{F}_{(n^2-1)}$ according to a similarity measure ρ, so that $\rho(\mathbf{F}_0, \mathbf{F}_{(0)}) \leq \rho(\mathbf{F}_0, \mathbf{F}_{(1)}) \leq \cdots \leq \rho(\mathbf{F}_0, \mathbf{F}_{(n^2-1)})$, where obviously $\mathbf{F}_{(0)} = \mathbf{F}_0$. As the similarity measure ρ we use the metric L_∞, defined by

$$L_\infty(\mathbf{F}_i, \mathbf{F}_j) = \max\{|F_i^R - F_j^R|, |F_i^G - F_j^G|, |F_i^B - F_j^B|\}, \tag{1}$$

because it is specially appropriate to detect noise due to its high sensitivity to differences between any of the pixels components.

Now, we consider the $s+1$ first pixels $\mathbf{F}_{(0)}, \mathbf{F}_{(1)}, \ldots, \mathbf{F}_{(s)}$ and compute the ROD_s statistic [34] for the pixel $\mathbf{F}_0$, as follows:

$$ROD_s(\mathbf{F}_0) = \sum_{j=0}^{s} L_\infty(\mathbf{F}_0, \mathbf{F}_{(j)}). \tag{2}$$

Notice that since $\mathbf{F}_0 = \mathbf{F}_{(0)}$ then $L_\infty(\mathbf{F}_0, \mathbf{F}_{(0)}) = 0$ and ROD_s takes integer values in the interval $[0, 255 \cdot s]$: a low value of $ROD_s(\mathbf{F}_0)$ means that the selected $s+1$ pixels $\mathbf{F}_{(j)}$ in W' are close to $\mathbf{F}_0$ which in turn means that $\mathbf{F}_0$ is expected to be noise-free; on the other hand, higher values of $ROD_s(\mathbf{F}_0)$ indicate a higher noise degree for $\mathbf{F}_0$, since no close pixels are found.

Now, if we put $x = ROD_s(\mathbf{F}_0)$ we define the certainty degree $\delta(\mathbf{F}_0)$ for the vague statement "$\mathbf{F}_0$ *is noisy*" by

$$\delta(\mathbf{F}_0) = f(x) = \begin{cases} 0 & x \leq k_1 \\ \dfrac{x - k_1}{k_2 - k_1} & k_1 < x < k_2 \\ 1 & k_2 \leq x \end{cases}, \tag{3}$$

where the k_1 and k_2 parameters settings will be commented in 2.5.

Finally, we assign to each pixel $\mathbf{F}_i$ of $\mathcal{F}$ a certainty degree of the vague statement "$\mathbf{F}_i$ *is not noisy*". Representing the negation by the fuzzy involutive operator, it will be given by $1 - \delta(\mathbf{F}_i)$. The corresponding fuzzy sets, for both vague statements f and $1 - f$, defined on $[0, 255 \cdot s]$, are shown in Figure 1.

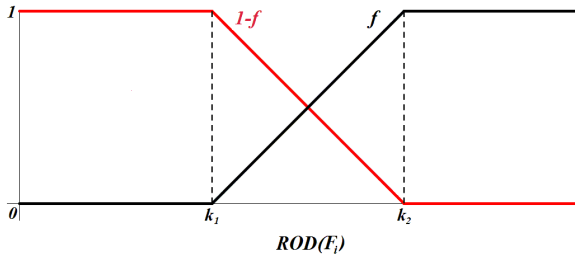

Figure 1: Noise degree of a pixel $\mathbf{F}_i$ as a function f of $ROD(\mathbf{F}_i)$: In black, certainty degree $\delta(\mathbf{F}_i)$ of "$\mathbf{F}_i$ *is noisy*"; in red, certainty degree of "$\mathbf{F}_i$ *is not noisy*".

2.2. Fuzzy averaging for noise reduction

Once the fuzzy noise degree of each pixel is computed, we aim to compute a fuzzy averaging between $\mathbf{F}_0$ and a robust estimator of noise-free colour vector for the pixels in W. As robust estimator we use the *Robust Vector Median Filter* proposed in [35] that has been found to outperform the classical *Vector Median Filter* and whose output we denote by $\mathbf{F}_{\mathrm{RVMF}}$.

So, the fuzzy averaging $\overline{\mathbf{F}_0}$ is obtained as

$$\overline{\mathbf{F}_0} = (1 - \delta(\mathbf{F}_0))\,\mathbf{F}_0 + \delta(\mathbf{F}_0)\,\mathbf{F}_{\mathrm{RVMF}} \tag{4}$$

This soft-switching allows the reduction operation to adapt to the pixel under processing characteristics so that it can properly process pixels heavily corrupted with noise, noise pixels that are similar to the background where they appear and also noise-like image features that are better preserved.

2.3. Correction step

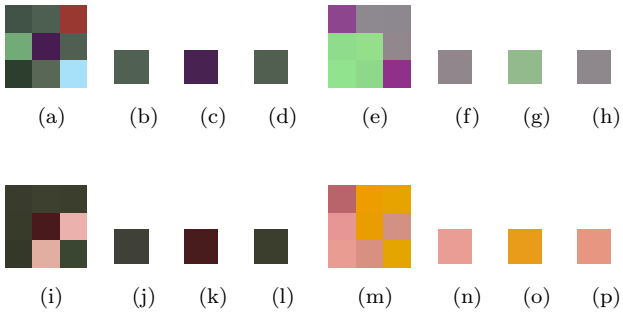

Figure 2: Four examples of pixels $\mathbf{F}_0$ where correction on $\overline{\mathbf{F}_0}$ is needed. (a),(e),(i),(m) show the input 3×3 window W centered at $\mathbf{F}_0$; (b),(f),(j),(n) show the respective original color (before adding noise) of the processed pixel; The color pixel $\overline{\mathbf{F}_0}$ obtained in each case is shown (c),(g),(k),(o), respectively, where we can see that it is not the result of a robust processing and so it needs the correction step; (d),(h),(l),(p) show the corresponding result after applying the correction step $\mathbf{F}_{\mathrm{CFAF}}$. (a)-(d) corresponds to pixel (42,16) in Baboon detail image corrupted by 30% of impulsive noise of type II, (e)-(h) to pixel (31,91) in Boats detail image corrupted by 20% of impulsive noise of type III, (i)-(l) to pixel (20,59) in Goldhill detail image corrupted by 30% of impulsive noise of type II, and (m)-(p) to pixel (25,34) in Lenna detail image corrupted by 20% of impulsive noise of type I.

The problem with $\overline{\mathbf{F}_0}$ is that it is obtained by averaging $\mathbf{F}_0$, and $\mathbf{F}_{\mathrm{RVMF}}$, which may belong to different image regions. Therefore, $\overline{\mathbf{F}_0}$ may be different from both regions, generating a colour artefact, an inapropriate blur, or a less noisy pixel but far from a noise-free one. That is, $\overline{\mathbf{F}_0}$ is not the result of a robust enough operation, as it is shown in Figure 2.

We propose to detect when $\overline{\mathbf{F}_0}$ can be considered as the result of an inappropriate processing by checking on its noisy degree in comparison with that of $\mathbf{F}_0$, as follows: If

$$\delta(\overline{\mathbf{F}_0}) > \gamma\delta(\mathbf{F}_0), \tag{5}$$

we consider that the averaging operation has performed inappropriately because

of its lack of robustness and consequently we disregard $\overline{\mathbf{F}_0}$ and assign as filter output $\mathbf{F}_{\mathrm{RVMF}}$. The setting of γ is studied in Section 2.5. This constitutes the last step of the *Corrected Fuzzy Averaging Filter* (CFAF).

This correction step, despite applied not many times in general, is necessary given that robustness is a key feature in any denoising method. In Figure 2 we show a few examples of the application of this correction step that illustrate its necessity.

2.4. Computational efficiency

The computation efficiency of vector filters is usually characterized in the literature in terms of the number of vector distances computations needed to process each image pixel. This is justified for the computation of vector distances being by far the most costly operation of those performed in the filtering process. Roughly, state-of-the-art impulse noise filters can be classified into two main classes in terms of the vector distances needed to process each pixel: (i) the fastest methods are those that decide the filtering operation to be done over each image pixel by computing just the distances from this pixel to its neighbours, such as [4] and some optimizations found in [3] and [5]. Thus, they save a lot of computations when processing noise-free pixels given that the detection phase is in general less complex than the correction phase where a robust estimation of noise-free vector needs to be computed; (ii) on the contrary, the less efficient methods are those which compute a robust estimation of a noise-free vector to decide the operation to apply over each pixel depending on robust vector statistics, such as the methods in [1], [2], [10], or [27]. Our method lies in efficiency with this latter group of filters.

In terms of distances computed, the CFAF first needs to compute $n^2 - 1$ distances in Eq. (2). Then, if $\delta(\mathbf{F}_0) = 0$ there is no further processing to be done, given that Eq. (4) would equal the identity operation and there is no need to perform Eq. (5) since $\delta(\mathbf{F}_0) = \delta(\overline{\mathbf{F}_0})$; otherwise, the RVMF output is needed for Eq. (4) which means to compute $\frac{n^2}{2}(n^2 - 1)$ vector distances. But these distances include the previous distances computed for Eq. (2) that we do not

need to compute again. Finally, for the correction step in Eq. (5) we need to compute $\delta(\overline{\mathbf{F}_0})$ which involves n^2-1 extra distances. So, the overall number of distances to be computed per pixel in the worst case is $\left(\frac{n^2}{2}+1\right)(n^2-1)$. That is, the computational order in terms of number of vector distances is proportional to the square of the number of pixels in the filtering window $O\left((n^2)^2\right)$.

Furthermore, we can detail the computational complexity of CFAF in terms of number of operations needed, in the worst case, to process each image pixel. First, we have 5 operations per vector distance according to Eq. (1) implying $5\left(\frac{n^2}{2}+1\right)(n^2-1)$ operations due to vector distances. In addition, we have $s-1$ operations in Eq. (2) that we compute twice ($\delta(\mathbf{F}_0)$ and $\delta(\overline{\mathbf{F}_0})$), 5 operations for Eq. (3) in the worst case, 3 for Eq. (4) and 2 for Eq. (5). All this sums up to $5\left(\frac{n^2}{2}+1\right)(n^2-1)+2s+8$. On top of this we need to add all the extra operations within RVMF (not due to vector distances) that include: additions ($n^2(n^2-1)$), ordering operations ($7n^2\log_2(n^2)$), and similarity operations (5). Thus, the overall number of operations needed in the whole filtering process per image pixel is $5\left(\frac{n^2}{2}+1\right)(n^2-1)+n^2(n^2-1)+7n^2\log_2(n^2)+5$, which implies that the computational order in terms of number of operations is also proportional to the square of the number of pixels in the filtering window $O\left((n^2)^2\right)$.

2.5. Adjustment of Parameters

To study the appropriate parameter setting for our method we have made extensive experiments using the four training images in row 1 of Figure 3 contaminated with different densities of four impulsive noise models [37] defined as follows:

Let $\mathbf{F}=\{F_R,F_G,F_B\}$ be the original pixel and let $\mathbf{F}^*$ be the corrupted pixel generated by the noise modelling process with noise appearance probability p. Then, the different models of impulsive noise are described as follows:

I. Fixed value impulsive noise.

Figure 3: Set of images used in the experiments: First row are training images and second are validation images.

All pixels in the colour image are modified following the next scheme:

$$\mathbf{F}^* = \begin{cases} \{d_1, F_G, F_B\} & \text{with probability } p \cdot p_1\,, \\ \{F_R, d_2, F_B\} & \text{with probability } p \cdot p_2\,, \\ \{F_R, F_G, d_3\} & \text{with probability } p \cdot p_3\,, \\ \{d_1, d_2, d_3\} & \text{with probability } p \cdot \left(1 - \sum_{i=1}^{3} p_i\right)\,. \end{cases} \tag{6}$$

where d_1, d_2, d_3 are independent and take the values 0 or 255 with the same probability, and p_i for $i = 1, 2, 3$ determines the probability of noise appearance in the image channels.

II. Correlated random value impulsive noise.

$\mathbf{F}^* = \{d_1, d_2, d_3\}$ with probability p, where d_1, d_2, d_3 are random integer values uniformly distributed in the interval $[0, 255]$.

III. Uncorrelated random value impulsive noise.

All pixels in the colour image are modified following the next scheme:

$$\mathbf{F}^* = \{F_R^*, F_G^*, F_B^*\} \tag{7}$$

where

$$F_R^* = \begin{cases} d_R & \text{with probability } p_R \\ F_R & \text{with probability } 1 - p_R \end{cases}$$

$$F_G^* = \begin{cases} d_G & \text{with probability } p_G \\ F_G & \text{with probability } 1 - p_G \end{cases}$$

$$F_B^* = \begin{cases} d_B & \text{with probability } p_B \\ F_B & \text{with probability } 1 - p_B \end{cases}$$

where d_R, d_G and d_B are random integer values in the interval $[0, 255]$ and p_R, p_G and p_B determine the probability of noise appearance in the red, green and blue channel, respectively. For simplicity we assume $p_R = p_G = p_B = p$.

IV. α-stable distribution

An α-stable distribution can also be used to model impulsive noise [38]. A symmetric α-stable (SαS) random variable is only described by its characteristic function

$$\varphi(t) = \mathrm{e}^{(j\theta t - \gamma|t|^\alpha)}$$

where $j \in \mathbb{C}$ is the imaginary unit, $\theta \in \mathbb{R}$ is the location parameter (centrality), $\gamma \in \mathbb{R}$ is the dispersion of the distribution and $\alpha \in (0, 2]$, which controls the heaviness of the tails, is a parameter that controls the degree of impulsiveness so that impulsiveness increases as α decreases. The Gaussian (α=2) and the Cauchy (α=1) distributions are the only symmetric α-stable distributions that have closed-form probability density functions.

We have used these noise models to contaminate the training images in row 1 of Figure 3. Images in row 2 of Figure 3 are used for validation and comparison with other methods in the next section. We have considered different intensities of noise in the image. For types I, II and III we have considered noise probability equal to 0.1, 0.2, and 0.3. For noise type IV we set α equal to 0.7, 0.6 and 0.5. We have focused in these low to middle densities of noise for being the range

of the most common densities used in the literature. It should be pointed out that for noise type III the probability p is referred for the noise to appear independently in each channel, what makes the global noise level being quite higher. Despite it is also interesting to study higher densities of contaminating noise, the technique described in this paper and those used in the experimental comparison are not specifically designed for such purpose.

For the filtering assessment we have used the objective quality measures MAE, PSNR, and DIS^k (defined from CMSSIM [39]) to objectively compare the performance of a selected group of filters. These measures are defined as follows [37]:

- MAE (*Mean Absolute Error*):

$$MAE = \frac{1}{NMQ} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{q=1}^{Q} \left| F^q(i,j) - \hat{F}^q(i,j) \right| \tag{8}$$

  where M and N are the image dimensions, Q is the channel number of the image ($Q = 3$ for colour images) and $F^q(i,j)$ and $\hat{F}^q(i,j)$ denote the q-th component of a vector in the original image for the pixel situated in position (i,j) in the image, respectively.

- PSNR (*Peak Signal to Noise Ratio*):

$$PSNR = 20 \log \left((2^k - 1) / \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{q=1}^{Q} \frac{\left(F^q(i,j) - \hat{F}^q(i,j)\right)^2}{NMQ}} \right) \tag{9}$$

  where we took $k = 8$ which corresponds with the 8 bits per channel case.

- CMSSIM (*Color Multiscale Structural Similarity index*): The images are divided into different patches of varying size (different scales) and the global similarity is obtained as

$$CMSSIM = (Clr(x,y))^{\delta} (l_M(x,y))^{\alpha_M} \prod^{M} (C_i(x,y))^{\beta_i} \cdot (S_i(x,y))^{\gamma_i} \tag{10}$$

  where Clr is a colour similarity factor, l_M is the luminance factor, contrast

Table 1: Mean and standard deviation (as subindex) of performance of proposed method in its one-step version for different values of s in terms of MAE, PSNR and DIS^k for the training images and the four noise types considered.

Noise type I									
s	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
2	$1.00_{0.6}$	$34.4_{3.2}$	$1.71_{0.6}$	$2.11_{1.0}$	$29.7_{2.1}$	$4.89_{0.6}$	$3.38_{1.2}$	$26.3_{1.2}$	$1.18_{0.1}$
3	$\mathbf{1.10_{0.8}}$	$\mathbf{35.2_{4.2}}$	$\mathbf{1.19_{0.7}}$	$\mathbf{2.13_{1.2}}$	$\mathbf{31.3_{3.2}}$	$\mathbf{3.06_{1.0}}$	$\mathbf{3.15_{1.4}}$	$\mathbf{29.1_{2.5}}$	$\mathbf{0.67_{0.1}}$
4	$\mathbf{1.26_{1.0}}$	$\mathbf{34.7_{4.5}}$	$\mathbf{1.04_{0.6}}$	$\mathbf{2.33_{1.5}}$	$\mathbf{31.2_{3.5}}$	$\mathbf{2.85_{1.1}}$	$\mathbf{3.36_{1.7}}$	$\mathbf{29.3_{2.9}}$	$\mathbf{0.55_{0.1}}$
5	$1.50_{1.3}$	$34.0_{4.7}$	$1.02_{0.7}$	$2.62_{1.7}$	$30.8_{3.6}$	$2.98_{1.4}$	$3.75_{2.0}$	$28.9_{3.0}$	$0.53_{0.1}$
6	$1.83_{1.6}$	$33.2_{4.7}$	$1.06_{0.7}$	$3.08_{2.1}$	$30.2_{3.6}$	$3.18_{1.5}$	$4.32_{2.3}$	$28.5_{3.1}$	$0.55_{0.2}$
Noise type II									
s	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
2	$\mathbf{1.08_{0.6}}$	$\mathbf{34.6_{3.2}}$	$\mathbf{0.59_{0.4}}$	$\mathbf{2.24_{0.9}}$	$\mathbf{30.8_{2.3}}$	$\mathbf{2.24_{0.8}}$	$\mathbf{3.88_{1.4}}$	$\mathbf{27.5_{2.0}}$	$\mathbf{0.78_{0.2}}$
3	$1.19_{0.8}$	$34.5_{3.9}$	$0.66_{0.6}$	$2.35_{1.1}$	$30.8_{2.7}$	$2.13_{0.9}$	$4.06_{1.6}$	$27.3_{2.1}$	$0.82_{0.2}$
4	$1.36_{1.0}$	$34.0_{4.3}$	$0.72_{0.6}$	$2.58_{1.3}$	$30.6_{2.9}$	$2.49_{1.1}$	$4.44_{2.0}$	$27.0_{2.2}$	$0.93_{0.3}$
5	$1.60_{1.3}$	$33.4_{4.4}$	$0.78_{0.7}$	$2.92_{1.6}$	$30.0_{3.0}$	$3.06_{1.5}$	$4.96_{2.3}$	$26.6_{2.3}$	$1.08_{0.3}$
6	$1.95_{1.6}$	$32.5_{4.4}$	$1.09_{0.8}$	$3.47_{2.0}$	$29.3_{3.1}$	$3.99_{1.8}$	$5.68_{2.6}$	$26.2_{2.4}$	$1.30_{0.4}$
Noise type III									
s	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
2	$2.51_{1.0}$	$29.9_{2.0}$	$5.96_{2.0}$	$5.06_{1.5}$	$25.9_{1.3}$	$20.1_{2.3}$	$8.06_{2.0}$	$23.1_{1.3}$	$3.84_{0.4}$
3	$\mathbf{2.57_{1.2}}$	$\mathbf{30.4_{2.7}}$	$\mathbf{4.28_{1.9}}$	$\mathbf{4.99_{1.8}}$	$\mathbf{26.9_{2.1}}$	$\mathbf{15.6_{3.1}}$	$\mathbf{7.89_{2.4}}$	$\mathbf{23.9_{1.7}}$	$\mathbf{3.38_{0.6}}$
4	$\mathbf{2.75_{1.5}}$	$\mathbf{30.4_{3.0}}$	$\mathbf{3.75_{1.9}}$	$\mathbf{5.19_{2.1}}$	$\mathbf{26.9_{2.3}}$	$\mathbf{14.6_{3.8}}$	$\mathbf{8.11_{2.6}}$	$\mathbf{23.9_{1.9}}$	$\mathbf{3.33_{0.7}}$
5	$3.01_{1.8}$	$30.2_{3.3}$	$3.62_{2.0}$	$5.53_{2.4}$	$26.8_{2.5}$	$15.0_{4.4}$	$8.43_{2.9}$	$23.9_{1.9}$	$3.38_{0.7}$
6	$3.39_{2.1}$	$29.8_{3.5}$	$3.86_{2.4}$	$5.99_{2.7}$	$26.6_{2.5}$	$15.7_{5.0}$	$8.79_{3.0}$	$23.8_{2.0}$	$3.45_{0.8}$
Noise type IV									
s	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
2	$4.25_{0.8}$	$29.1_{1.7}$	$9.55_{1.3}$	$4.99_{1.0}$	$28.0_{1.7}$	$13.4_{2.3}$	$6.05_{1.3}$	$26.5_{1.4}$	$1.97_{0.1}$
3	$\mathbf{4.31_{1.1}}$	$\mathbf{29.3_{2.1}}$	$\mathbf{8.63_{1.1}}$	$\mathbf{5.02_{1.3}}$	$\mathbf{28.3_{2.1}}$	$\mathbf{11.8_{2.5}}$	$\mathbf{5.97_{1.6}}$	$\mathbf{27.2_{2.0}}$	$\mathbf{1.61_{0.1}}$
4	$\mathbf{4.43_{1.2}}$	$\mathbf{29.3_{2.4}}$	$\mathbf{7.94_{1.0}}$	$\mathbf{5.13_{1.5}}$	$\mathbf{28.4_{2.3}}$	$\mathbf{11.0_{2.5}}$	$\mathbf{6.07_{1.8}}$	$\mathbf{27.4_{2.3}}$	$\mathbf{1.47_{0.1}}$
5	$4.61_{1.6}$	$29.2_{2.7}$	$7.65_{1.1}$	$5.31_{1.8}$	$28.3_{2.6}$	$10.6_{2.9}$	$6.25_{2.1}$	$27.3_{2.5}$	$1.42_{0.2}$
6	$4.84_{1.9}$	$29.0_{3.0}$	$7.52_{1.5}$	$5.54_{2.1}$	$28.2_{2.8}$	$10.2_{3.0}$	$6.46_{2.4}$	$27.2_{2.7}$	$1.40_{0.2}$

and structure similarity of scale i are C_i and S_i, respectively, and $\beta_1 = \gamma_1 = 0.04448$; $\beta_2 = \gamma_2 = 0.2856$; $\beta_3 = \gamma_3 = 0.3001$; $\beta_4 = \gamma_4 = 0.2363$; $\alpha_5 = \beta_5 = \gamma_5 = 0.1333$, and $\delta = 0.7$. CMSSIM takes values in the interval $[0, 1]$. To better observe the performance differences we will use the value of DIS defined as

$$DIS^k = (1 - CMSSIM) \cdot 10^k.$$

First we deal with the adjustment of the parameter s which is the most important one in terms of influence in performance. To set this parameter we

could choose between a fixed setting or an adaptive setting. Fixed setting is easier to find but adaptive setting should yield better results. However, the s parameter can only take integer values so changing its value makes a kind of crisp switching in the filter behaviour. This makes that it is not easy to decide when to modify the value of s. Also, the switching would affect the other parameters in the method that should be adjusted again. Therefore, we think that it is more practical to find a fixed value for s, which is the strategy followed by other filtering methods like [34, 36]. In these works it has been studied that the setting of s is related to robustness and detail preservation. A lower value is better for preserving details while a higher value makes the filter more robust.

In Table 1 we show the filter performance for different values of s while fixing the rest of the parameters to suboptimal values. We can see that the best performance results are achieved for $s = 3$ and $s = 4$, for noise types I, III and IV. The former is a little better for low noise and the latter for high noise. For noise type II a lower value of s works better due to the much lower probability of clusters of noise appearing for this noise type. Thus, from a numerical value point of view, both settings $s = 3$ and $s = 4$ are reasonable, but we know that $s = 4$ is associated to a more robust performance, therefore, we choose to use this setting in general.

We also consider for our method a two-step version that according to [10, 9] can improve significantly the performance of the one-step version when clusters of noisy pixels appear in the images. In this version, named from now on CFAF2, the first step should use a parameter setting to remove only isolated noise pixels and the second one to remove clusters of noise. We have studied how to set the filter parameter s for the two steps (s_1, s_2). In Table 2 we show the performance for different couples of s_1, s_2. We can see that there is a higher variability on the setting achieving the optimum in each case. Again, for noise type II lower values of s_1, s_2 are preferred. In general, the difference in performance is not much. Following the same reasoning as for setting s in the one-step version, we set s_1, s_2 to the most robust case of the options achieving best overall performance, which now is $s_1 = 4, s_2 = 6$ (similar results are obtained for $s_1 = 3, s_2 = 6$).

Table 2: Mean and standard deviation (as subindex) of performance of proposed method in its two-steps version for different values of s_1, s_2 in terms of MAE, PSNR and DIS^k for the training images and the four noise types considered.

Noise type I									
s_1, s_2	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
$2, 4$	$1.05_{0.7}$	$34.4_{3.4}$	$1.45_{0.5}$	$2.14_{1.0}$	$29.8_{2.1}$	$4.43_{0.6}$	$3.36_{1.2}$	$26.5_{1.2}$	$1.06_{0.1}$
$2, 5$	$1.07_{0.7}$	$34.3_{3.6}$	$1.40_{0.5}$	$2.16_{1.1}$	$29.8_{2.2}$	$4.29_{0.7}$	$3.37_{1.2}$	$26.5_{1.2}$	$1.03_{0.1}$
$3, 5$	$1.17_{0.9}$	$34.9_{4.3}$	$1.08_{0.7}$	$2.18_{1.3}$	$31.5_{3.3}$	$2.84_{1.0}$	$3.16_{1.5}$	$29.4_{2.6}$	$0.57_{0.1}$
$3, 6$	$\mathbf{1.22_{0.9}}$	$\mathbf{34.7_{4.3}}$	$\mathbf{1.03_{0.7}}$	$\mathbf{2.22_{1.3}}$	$\mathbf{31.4_{3.4}}$	$\mathbf{2.78_{1.1}}$	$\mathbf{3.17_{1.5}}$	$\mathbf{29.4_{2.7}}$	$\mathbf{0.54_{0.1}}$
$4, 6$	$\mathbf{1.39_{1.1}}$	$\mathbf{34.3_{4.5}}$	$\mathbf{1.04_{0.8}}$	$\mathbf{2.39_{1.5}}$	$\mathbf{31.4_{3.7}}$	$\mathbf{2.64_{1.3}}$	$\mathbf{3.37_{1.7}}$	$\mathbf{29.8_{3.2}}$	$\mathbf{0.47_{0.1}}$
$4, 7$	$1.46_{1.2}$	$33.9_{4.5}$	$1.06_{0.8}$	$2.45_{1.6}$	$31.3_{3.8}$	$2.65_{1.4}$	$3.41_{1.8}$	$29.7_{3.2}$	$0.45_{0.1}$
Noise type II									
s_1, s_2	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
$2, 4$	$\mathbf{1.12_{0.6}}$	$\mathbf{34.7_{3.4}}$	$\mathbf{0.58_{0.4}}$	$\mathbf{2.23_{0.9}}$	$\mathbf{31.2_{2.6}}$	$\mathbf{1.79_{0.8}}$	$\mathbf{3.67_{1.5}}$	$\mathbf{28.7_{2.5}}$	$\mathbf{0.57_{0.2}}$
$2, 5$	$1.14_{0.7}$	$34.6_{3.6}$	$0.60_{0.4}$	$2.24_{1.0}$	$31.2_{2.6}$	$1.70_{0.8}$	$3.66_{1.5}$	$28.7_{2.5}$	$0.54_{0.2}$
$3, 5$	$1.26_{0.8}$	$34.4_{3.9}$	$0.65_{0.6}$	$2.38_{1.2}$	$31.1_{2.9}$	$1.79_{1.1}$	$3.84_{1.7}$	$28.6_{2.6}$	$0.57_{0.3}$
$3, 6$	$1.30_{0.9}$	$34.3_{4.1}$	$0.66_{0.6}$	$2.42_{1.2}$	$31.1_{3.0}$	$1.78_{1.1}$	$3.86_{1.7}$	$28.7_{2.7}$	$0.56_{0.3}$
$4, 6$	$1.47_{1.1}$	$33.8_{4.3}$	$0.68_{0.7}$	$2.63_{1.4}$	$30.7_{3.1}$	$2.13_{1.2}$	$4.19_{2.0}$	$28.3_{2.8}$	$0.64_{0.3}$
$4, 7$	$1.55_{1.2}$	$33.5_{4.4}$	$0.71_{0.7}$	$2.69_{1.4}$	$30.6_{3.2}$	$2.12_{1.2}$	$4.22_{2.1}$	$28.3_{2.8}$	$0.62_{0.3}$
Noise type III									
s_1, s_2	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
$2, 4$	$2.52_{1.0}$	$30.2_{2.2}$	$4.83_{1.6}$	$4.93_{1.5}$	$26.5_{1.5}$	$17.1_{2.3}$	$7.51_{2.0}$	$24.2_{1.5}$	$3.27_{0.4}$
$2, 5$	$2.53_{1.0}$	$30.2_{2.3}$	$4.51_{1.5}$	$4.92_{1.6}$	$26.6_{1.6}$	$16.4_{2.3}$	$7.43_{2.0}$	$24.4_{1.5}$	$3.17_{0.4}$
$3, 5$	$2.62_{1.3}$	$30.6_{2.8}$	$3.65_{1.8}$	$4.88_{1.8}$	$27.5_{2.3}$	$12.7_{3.1}$	$7.27_{2.3}$	$25.3_{2.0}$	$2.72_{0.6}$
$3, 6$	$\mathbf{2.66_{1.3}}$	$\mathbf{30.6_{2.9}}$	$\mathbf{3.48_{1.8}}$	$\mathbf{4.89_{1.9}}$	$\mathbf{27.6_{2.4}}$	$\mathbf{12.2_{3.1}}$	$\mathbf{7.23_{2.3}}$	$\mathbf{25.4_{2.0}}$	$\mathbf{2.66_{0.6}}$
$4, 6$	$\mathbf{2.81_{1.5}}$	$\mathbf{30.5_{3.1}}$	$\mathbf{3.22_{1.9}}$	$\mathbf{5.08_{2.1}}$	$\mathbf{27.6_{2.6}}$	$\mathbf{11.6_{3.2}}$	$\mathbf{7.42_{2.6}}$	$\mathbf{25.5_{2.2}}$	$\mathbf{2.59_{0.7}}$
$4, 7$	$2.88_{1.6}$	$30.5_{3.2}$	$3.11_{1.9}$	$5.10_{2.2}$	$27.6_{2.6}$	$11.4_{3.2}$	$7.40_{2.6}$	$25.6_{2.3}$	$2.54_{0.7}$
Noise type IV									
s_1, s_2	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
$2, 4$	$4.27_{0.9}$	$29.2_{1.9}$	$9.13_{1.1}$	$4.98_{1.0}$	$28.2_{1.7}$	$12.5_{2.2}$	$6.0_{1.3}$	$26.8_{1.5}$	$1.81_{0.1}$
$2, 5$	$4.29_{1.0}$	$29.3_{1.9}$	$8.88_{1.0}$	$5.00_{1.1}$	$28.3_{1.8}$	$12.1_{2.0}$	$6.0_{1.4}$	$26.8_{1.6}$	$1.75_{0.1}$
$3, 5$	$4.36_{1.1}$	$29.4_{2.2}$	$8.18_{1.0}$	$5.05_{1.3}$	$28.5_{2.1}$	$11.2_{2.3}$	$6.0_{1.6}$	$27.4_{2.1}$	$1.50_{0.1}$
$3, 6$	$\mathbf{4.40_{1.2}}$	$\mathbf{29.3_{2.3}}$	$\mathbf{8.00_{0.9}}$	$\mathbf{5.07_{1.4}}$	$\mathbf{28.5_{2.2}}$	$\mathbf{10.9_{2.2}}$	$\mathbf{6.0_{1.6}}$	$\mathbf{27.5_{2.1}}$	$\mathbf{1.45_{0.1}}$
$4, 6$	$\mathbf{4.53_{1.4}}$	$\mathbf{29.2_{2.5}}$	$\mathbf{7.80_{0.9}}$	$\mathbf{5.18_{1.6}}$	$\mathbf{28.5_{2.4}}$	$\mathbf{10.6_{2.4}}$	$\mathbf{6.1_{1.9}}$	$\mathbf{27.5_{2.4}}$	$\mathbf{1.38_{0.1}}$
$4, 7$	$4.60_{1.4}$	$29.2_{2.6}$	$7.69_{0.9}$	$5.22_{1.6}$	$28.4_{2.4}$	$10.4_{2.4}$	$6.10_{1.9}$	$27.5_{2.4}$	$1.35_{0.2}$

Next, once fixed the setting for s, we deal with the setting of the parameters k_1, k_2, γ. We address this task from two points of view: On the one hand, through extensive experimentation, we find a robust setting of the parameters able to provide a good overall performance. On the other hand, we study how to set the parameters adaptively to the density and type of the noise contaminating the images in each case.

In particular, we first compute the terns (k_1, k_2, γ) that optimize each of the measures for each type and density of noise and for each of the images in

Table 3: Regression lines of the parameters for the training images and the four noise types considered.

Regression lines for Noise type I: $k_1 = (0.1215 + (-0.0938)\cdot p)\cdot 255\cdot s$ $k_2 = (0.2333 + (-0.0625)\cdot p)\cdot 255\cdot s$ $\gamma = 1.1875 + (-0.3542)\cdot p$	Regression lines for Noise type II: $k_1 = (0.1271 + (0.0208)\cdot p)\cdot 255\cdot s$ $k_2 = (0.2236 + (-0.0000)\cdot p)\cdot 255\cdot s$ $\gamma = 1.1472 + (-0.2708)\cdot p$
Regression lines for Noise type III: $k_1 = (0.0882 + (-0.0052)\cdot p)\cdot 255\cdot s$ $k_2 = (0.1826 + (0.0729)\cdot p)\cdot 255\cdot s$ $\gamma = 1.2431 + (-0.7292)\cdot p$	Regression lines for Noise type IV: $k_1 = (0.0531 + (-0.0573)\cdot \alpha)\cdot 255\cdot s$ $k_2 = (0.2160 + (-0.0521)\cdot \alpha)\cdot 255\cdot s$ $\gamma = 1.2486 + (0.0625)\cdot \alpha$

the training set. That is $3 \cdot 3 \cdot 4 \cdot 4 = 144$ terns. From them, and giving more importance to those optimizing the higher densities of noise to obtain a robust performance, we select the general robust setting

$$k_1 = 0.1 \cdot 255 \cdot s,\ k_2 = 0.2 \cdot 255 \cdot s,\ \gamma = 1.2. \tag{11}$$

For the adaptive setting we also use the 144 terns computed above but now we find a regression line using the least squares method to relate each parameter to the density of contaminating noise for each one of the 4 types of noise. Thus, we obtain the 12 regression lines in Table 3 where p, α are the densities of contaminating noise. Through them, we can set the appropriate parameters in each case. It is interesting to note that the slope of each line is quite small, meaning that constant setting of the parameter should not be a bad choice.

The comparison between the two alternatives considered for setting k_1, k_2, γ is shown in Table 4. Here we can see that there is not a clear upside yielded by the regression-based setting with respect to the robust setting. In particular, despite the robust setting is not picked to perform better for low noise densities it is in these cases where it works significantly better than the regression-based setting. Probably this is due to a worse fit of the regression lines for low noise. For higher noise densities, the regression-based setting does not always perform better and the improvement when it does is very small. Furthermore, using the regression-based setting adds the problem of having to estimate the values of p and α which are in general unknown. Therefore, we conclude that it is more

Table 4: Mean and standard deviation of performance of proposed method using the robust setting (Rob.) vs the regression-based (Reg.) setting for k_1, k_2, γ in terms of MAE, PSNR and DIS^k for the training images and the four noise types considered.

Noise type I									
Param.	$p = 10\%$			$p = 20\%$			$p = 30\%$		
setting	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
Reg.	$1.69_{1.6}$	$32.7_{4.8}$	$1.26_{0.8}$	$2.2_{1.3}$	$31.3_{3.4}$	$3.05_{1.1}$	$3.36_{1.6}$	$29.3_{2.8}$	$0.56_{0.1}$
Rob.	$1.26_{1.0}$	$34.7_{4.5}$	$1.04_{0.6}$	$2.3_{1.4}$	$31.2_{3.5}$	$2.85_{1.1}$	$3.36_{1.6}$	$29.3_{2.9}$	$0.55_{0.1}$
Noise type II									
Param.	$p = 10\%$			$p = 20\%$			$p = 30\%$		
setting	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
Reg.	$1.48_{1.2}$	$32.9_{4.5}$	$0.81_{0.7}$	$2.36_{1.0}$	$30.7_{2.6}$	$2.39_{0.9}$	$4.16_{1.6}$	$27.1_{2.0}$	$0.88_{0.2}$
Rob.	$1.36_{1.0}$	$34.0_{4.2}$	$0.72_{0.6}$	$2.58_{1.3}$	$30.5_{2.9}$	$2.49_{1.1}$	$4.44_{1.9}$	$27.0_{2.2}$	$0.93_{0.2}$
Noise type III									
Param.	$p = 10\%$			$p = 20\%$			$p = 30\%$		
setting	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
Reg.	$3.79_{2.6}$	$29.0_{3.7}$	$3.95_{2.9}$	$5.27_{2.2}$	$27.0_{2.4}$	$14.37_{3.9}$	$8.16_{2.6}$	$23.9_{1.9}$	$3.32_{0.7}$
Rob.	$2.75_{1.5}$	$30.4_{3.0}$	$3.75_{1.9}$	$5.19_{2.1}$	$27.0_{2.3}$	$14.61_{3.8}$	$8.11_{2.6}$	$23.9_{1.8}$	$3.33_{0.7}$
Noise type IV									
Param.	$p = 10\%$			$p = 20\%$			$p = 30\%$		
setting	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
Reg.	$6.24_{3.2}$	$28.1_{3.4}$	$7.02_{4.0}$	$5.27_{1.9}$	$28.9_{2.8}$	$9.2_{2.9}$	$6.16_{2.2}$	$27.8_{2.7}$	$1.28_{0.2}$
Rob.	$4.43_{1.3}$	$29.3_{2.5}$	$7.94_{1.0}$	$5.13_{1.5}$	$28.4_{2.4}$	$11.0_{2.6}$	$6.07_{1.9}$	$27.4_{2.3}$	$1.47_{0.1}$

practical to use the robust setting.

3. Experimental Study

In this section we validate and compare the performance of the CFAF. For this, we use the validation image set in row 2 of Figure 3. These images are contaminated with the different noise types and densities considered in Section 2.5 and filtered with the CFAF and a set of recent representative filters from the different families which are shown in Table 5 [37]. It should be noted that method in [29] is just a restoration method that needs a crisp previous noise detection for which we use the FPGF method in [4]. In all cases we have used a 3×3 filtering window since larger filtering windows do not provide significant relative performance differences among the considered methods. Also, for all filters we consider the application of one filtering step and two recursive filtering steps, as for our method. The parameter setting suggested for each method in the corresponding work is used for practical reasons. The experimental results for the comparison are shown in Tables 5-8.

Table 5: Filters used in the comparisons.

Filter Name / Reference	Short Name
Fast Peer Group Filter [4]	FPGF
Sigma Vector Median Filter [2]	SVMF
Impulse Noise Reduction [10]	INR
Local Self-Adaptive Fuzzy Magnitude Impulse Vector Filter [11]	LSAFSVF
Iterative Peer Group Switching Vector Filter [9]	IPGSVF
Fuzzy ROD Filter [3]	FRF
Quaternion Representation Vector Filter [6]	QRVF
Simple Fuzzy Rule Filter [16]	SFRF
Fuzzy Decision Filter [24]	FDF
L0TV: Image Restoration in the Presence of Impulse Noise [27]	L0TV
Learning Deep CNN Denoiser [29]	LDCNND

For noise type I (Table 5) we can see that the CFAF is among the best performing methods along with IPGSVF, INR, L0TV, and LDCNND. L0TV performs specially well in this type of noise given that it is specifically designed for salt and pepper noise. However, it shows a much higher standard deviation of performance, meaning that it is not as robust as the rest. This may be due to componentwise application of the method in the three color channels, that can lead to undesired results. The proposed method is specially good in terms of the DIS measure, where in general it outperforms the rest.

In the case of noise type II (Table 6) the leading methods are FPGF, IPGSVF, FRF, LDCNND and CFAF, which is again specially good in terms of DIS. However, CFAF performs a little worse in comparison to noise type I, probably because of the parameter setting used that, as commented in Section 2.5, is not optimal for this type of noise. We can see here the performance improvement achieved for LDCNND in front of FPGF (notice that they have the same detection mechanism and different restoration step). FPGF is very good at detecting this type of noise, as well as FRF and IPGSVF, because this type of noise is less likely to generate noise clusters. CFAF parameter setting is set to be robust in front of noise clusters so the detail preservation in this type of noise is lower and its performance drops a bit. Moreover, notice that L0TV and LDCNND show a lack of robustness illustrated by the higher standard de-

viation of their performances. For L0TV this can be explained again by the componentwise application of the method in the color channels. For LDCNND the high variability in performance may be related to images matching better or worse the characteristics of the dataset with which LDCNND was trained.

In the case of noise type III (Table 7) we need to take into account that noise is introduced independently in the color channels. This is interesting because this type of noise represents a higher contamination degree for the probability of noise appearance being applied separately in each channel. Here, best performance is achieved by L0TV, which is now taking advantage of its componentwise application in the color channels. Given that this noise type has a componentwise nature, L0TV can detect and correct it better by processing the channels separately. However, it also shows a higher standard deviation than the rest, meaning that there are cases where the performance drops which may be due to generation of color artefacts in the componentwise restoration of the image. CFAF and INR are the next methods in performance. INR processes the images taking account correlation between pairs of channels, which is better in this case because it is very likely that one channel is corrupted and the other two are not. CFAF yields a performance similar to INR due to its robust parameter setting ant its flexible fuzzy processing of the image. This means that CFAF performance on higher noise density levels is good, as well.

Finally, noise type IV (Table 8) CFAF is the best performing method with INR. This type of noise is not a pure impulse model but has also some high frequency contamination component. Thus, crisp switching methods tend to perform worse for this noise and soft switching methods as INR and CFAF are preferred. This means that CFAF is able to yield a good performance in mixed noise scenarios.

As a consequence, our method provides a very solid performance both numerically and visually, as it is shown in Figures 4-5. CFAF is robust in all noise types and densities and it is the only filter that is always in the group of best performing filters for the four noise types. This is due to the flexibility provided by the fuzzy design that allows the filter to adapt performance in different noise

contexts as well as for the correction step that makes sure the method is always robust. The bottleneck of the method is that the noise detection accuracy critically depends on the parameter s. So, when the setting is not good, as happens for noise type II, the performance is affected significantly. However, given that it is a soft switching method this is not as critical as for crisp switching methods like FPGF, FRF, or IPGSVF. Finally, it must be stressed that CFAF is a pure vector filtering method so it inherits the strengths and weakness of this family of vector filters: vector filters provide a simple an efficient way to process multichannel images taking into account the correlation among the image channels but this correlation is modeled in an inflexible, rigid, way, which constitutes the main limitation of this approach.

Table 6: Mean value and standard deviation (as subindex) of performance over validation images using noise type I.

Filter	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
FPGF	$1.01_{0.2}$	$31.7_{1.6}$	$1.64_{0.6}$	$2.19_{0.3}$	$28.5_{1.1}$	$5.06_{1.0}$	$3.62_{0.3}$	$26.4_{1.0}$	$0.96_{0.3}$
SVMF	$4.91_{0.4}$	$19.4_{0.4}$	$6.25_{0.2}$	$9.83_{0.4}$	$16.5_{0.1}$	$9.08_{0.3}$	$15.0_{0.6}$	$14.5_{0.2}$	$7.28_{0.3}$
INR	$\mathbf{0.80_{0.3}}$	$\mathbf{32.5_{1.7}}$	$\mathbf{0.91_{0.1}}$	$\mathbf{1.44_{0.3}}$	$\mathbf{28.9_{1.0}}$	$\mathbf{4.63_{0.1}}$	$\mathbf{2.49_{0.2}}$	$\mathbf{25.2_{0.5}}$	$\mathbf{0.87_{0.1}}$
LSAFSVF	$1.20_{0.2}$	$30.8_{1.6}$	$0.93_{0.1}$	$2.15_{0.3}$	$28.6_{1.2}$	$5.51_{0.2}$	$3.28_{0.2}$	$26.4_{0.9}$	$0.90_{0.2}$
IPGSVF	$\mathbf{0.98_{0.2}}$	$\mathbf{31.1_{1.5}}$	$\mathbf{0.93_{0.2}}$	$\mathbf{1.77_{0.3}}$	$\mathbf{29.7_{1.5}}$	$\mathbf{2.83_{0.2}}$	$\mathbf{2.67_{0.3}}$	$\mathbf{28.2_{1.6}}$	$\mathbf{0.97_{0.2}}$
FRF	$1.00_{0.2}$	$31.8_{1.4}$	$0.91_{0.1}$	$2.12_{0.3}$	$28.7_{1.0}$	$3.70_{0.1}$	$3.75_{0.7}$	$25.9_{0.9}$	$0.94_{0.1}$
QRVF	$4.91_{0.4}$	$19.4_{0.4}$	$6.25_{0.2}$	$9.83_{0.4}$	$16.5_{0.1}$	$9.05_{0.3}$	$15.0_{0.6}$	$14.5_{0.2}$	$7.27_{0.3}$
SFRF	$10.3_{1.2}$	$23.9_{1.0}$	$3.57_{0.3}$	$13.0_{0.5}$	$22.2_{0.4}$	$9.73_{0.5}$	$15.9_{0.6}$	$20.8_{0.4}$	$6.92_{0.3}$
FDF	$4.87_{0.4}$	$19.5_{0.4}$	$4.27_{0.2}$	$9.75_{0.4}$	$16.5_{0.1}$	$6.08_{0.3}$	$14.9_{0.5}$	$14.6_{0.2}$	$7.29_{0.3}$
L0TV	$\mathbf{0.91_{0.3}}$	$\mathbf{32.6_{2.8}}$	$\mathbf{4.41_{1.7}}$	$\mathbf{1.35_{0.4}}$	$\mathbf{30.9_{2.4}}$	$\mathbf{7.02_{1.3}}$	$\mathbf{1.80_{0.4}}$	$\mathbf{29.8_{2.2}}$	$\mathbf{0.93_{0.6}}$
LDCNND	$\mathbf{1.50_{0.2}}$	$\mathbf{32.8_{1.9}}$	$\mathbf{2.34_{1.0}}$	$\mathbf{2.29_{0.3}}$	$\mathbf{29.7_{1.5}}$	$\mathbf{5.12_{1.8}}$	$\mathbf{3.66_{0.5}}$	$\mathbf{27.3_{1.1}}$	$\mathbf{0.73_{0.5}}$
CFAF	$\mathbf{1.07_{0.2}}$	$\mathbf{32.3_{2.1}}$	$\mathbf{0.75_{0.4}}$	$\mathbf{2.03_{0.3}}$	$\mathbf{30.0_{1.7}}$	$\mathbf{2.61_{1.4}}$	$\mathbf{3.12_{0.3}}$	$\mathbf{27.8_{1.2}}$	$\mathbf{0.48_{0.2}}$
FPGF2	$1.02_{0.3}$	$31.7_{1.8}$	$1.31_{0.5}$	$2.16_{0.4}$	$28.9_{1.1}$	$3.82_{1.1}$	$3.66_{0.4}$	$26.7_{1.0}$	$0.90_{0.1}$
SVMF2	$4.91_{0.4}$	$19.4_{0.4}$	$3.75_{0.2}$	$9.83_{0.4}$	$16.5_{0.1}$	$6.11_{0.3}$	$15.0_{0.6}$	$14.5_{0.2}$	$7.22_{0.3}$
INR2	$\mathbf{1.34_{0.7}}$	$\mathbf{31.1_{2.1}}$	$\mathbf{0.80_{0.4}}$	$\mathbf{1.76_{0.7}}$	$\mathbf{29.7_{1.9}}$	$\mathbf{1.64_{1.0}}$	$\mathbf{2.21_{0.6}}$	$\mathbf{28.8_{1.9}}$	$\mathbf{0.26_{0.1}}$
LSAFSVF2	$1.32_{0.3}$	$30.4_{1.6}$	$1.65_{0.5}$	$2.22_{0.3}$	$28.6_{1.2}$	$4.24_{1.8}$	$3.25_{0.3}$	$27.0_{0.9}$	$0.69_{0.2}$
IPGSVF2	$\mathbf{0.98_{0.2}}$	$\mathbf{31.1_{1.7}}$	$\mathbf{1.67_{1.1}}$	$\mathbf{1.76_{0.3}}$	$\mathbf{29.8_{1.5}}$	$\mathbf{2.17_{1.9}}$	$\mathbf{2.67_{0.3}}$	$\mathbf{28.3_{1.7}}$	$\mathbf{0.25_{0.2}}$
FRF2	$1.00_{0.2}$	$31.8_{1.5}$	$0.90_{0.1}$	$2.12_{0.3}$	$28.8_{1.1}$	$2.83_{0.9}$	$3.72_{0.7}$	$26.0_{0.9}$	$0.49_{0.1}$
QRVF2	$4.91_{0.4}$	$19.4_{0.4}$	$3.75_{1.6}$	$9.83_{0.4}$	$16.5_{0.1}$	$8.16_{0.3}$	$15.0_{0.6}$	$14.5_{0.1}$	$7.22_{0.3}$
SFRF2	$10.4_{1.9}$	$24.3_{1.3}$	$3.23_{0.5}$	$12.2_{1.2}$	$23.1_{0.7}$	$7.43_{0.6}$	$14.3_{1.4}$	$21.8_{0.5}$	$6.59_{0.3}$
FDF2	$4.87_{0.4}$	$19.5_{0.4}$	$3.73_{0.2}$	$9.75_{0.4}$	$16.5_{0.1}$	$6.17_{0.3}$	$14.9_{0.5}$	$14.6_{0.1}$	$7.21_{0.3}$
L0TV2	$\mathbf{0.97_{0.3}}$	$\mathbf{32.3_{2.8}}$	$\mathbf{4.29_{1.6}}$	$\mathbf{1.42_{0.4}}$	$\mathbf{30.7_{2.4}}$	$\mathbf{6.62_{2.6}}$	$\mathbf{1.80_{0.4}}$	$\mathbf{29.9_{2.2}}$	$\mathbf{0.89_{0.3}}$
LDCNND2	$\mathbf{1.75_{0.3}}$	$\mathbf{33.2_{2.2}}$	$\mathbf{2.43_{1.1}}$	$\mathbf{2.48_{0.4}}$	$\mathbf{30.6_{1.7}}$	$\mathbf{4.51_{2.5}}$	$\mathbf{3.84_{0.5}}$	$\mathbf{27.6_{1.1}}$	$\mathbf{0.71_{0.5}}$
CFAF2	$\mathbf{1.25_{0.3}}$	$\mathbf{31.8_{1.9}}$	$\mathbf{0.80_{0.5}}$	$\mathbf{2.16_{0.4}}$	$\mathbf{29.8_{1.5}}$	$\mathbf{2.21_{1.3}}$	$\mathbf{3.12_{0.3}}$	$\mathbf{28.1_{1.3}}$	$\mathbf{0.38_{0.2}}$

4. Conclusions

In this paper we have introduced a fuzzy detection and reduction method to remove impulse noise from colour images. The method is based on obtaining fuzzy noise-free degrees of the input colour vectors and use them to reduce

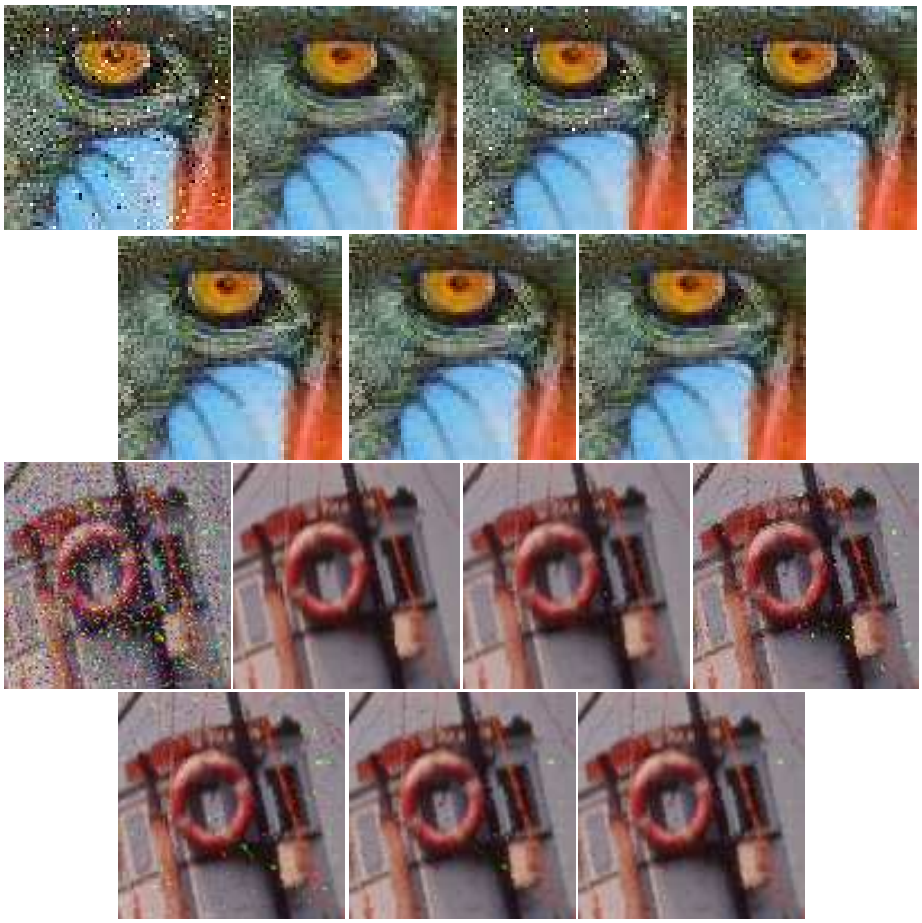

Figure 4: From left to right, noisy image, and images filtered by INR2, QRVF, IPGSVF, FRF, CFAF and CFAF2 for the Baboon image corrupted by 10% of impulsive noise model I in the first and second row. From left to right, noisy image, and images filtered by INR2, QRVF, RWVMF, IPGSVF, CFAF and CFAF2 for the Boats image corrupted by 20% of impulsive noise model III in the third and fourth row.

Figure 5: From left to right, noisy image, and images filtered by INR2, SFRF, RWVMF, QRVF, CFAF and CFAF2 filters for the Goldhill image corrupted by impulsive noise model IV with $\alpha = 0.7$ in the first and second row. From left to right, noisy image, and images filtered by FRF, IPGSVF, QRVF, FPGF, CFAF and CFAF2 filters for the Lenna image corrupted by 30% of impulsive noise model II in the third and fourth row.

Table 7: Mean value and standard deviation (as subindex) of performance over validation images using noise type II.

Filter	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
FPGF	$\mathbf{1.11_{0.2}}$	$\mathbf{31.7_{1.2}}$	$\mathbf{0.61_{0.2}}$	$\mathbf{2.27_{0.4}}$	$\mathbf{29.1_{1.5}}$	$\mathbf{1.63_{0.7}}$	$\mathbf{4.05_{0.6}}$	$\mathbf{26.8_{1.0}}$	$\mathbf{0.42_{0.1}}$
SVMF	$7.21_{0.4}$	$19.1_{0.6}$	$9.82_{0.4}$	$14.5_{0.3}$	$16.1_{0.3}$	$6.62_{0.5}$	$22.1_{0.8}$	$14.2_{0.5}$	$2.43_{0.4}$
INR	$1.54_{0.4}$	$30.4_{1.3}$	$0.96_{0.1}$	$3.19_{0.6}$	$26.6_{1.2}$	$4.45_{0.3}$	$5.34_{0.3}$	$23.8_{0.2}$	$0.85_{0.2}$
LSAFSVF	$1.33_{0.3}$	$30.6_{1.4}$	$0.99_{0.1}$	$2.47_{0.3}$	$28.1_{1.3}$	$3.81_{0.1}$	$4.10_{0.4}$	$25.5_{0.8}$	$0.94_{0.1}$
IPGSVF	$\mathbf{1.18_{0.1}}$	$\mathbf{31.1_{1.1}}$	$\mathbf{0.99_{0.1}}$	$\mathbf{2.37_{0.3}}$	$\mathbf{28.4_{1.6}}$	$\mathbf{1.94_{0.1}}$	$\mathbf{3.77_{0.3}}$	$\mathbf{26.3_{0.6}}$	$\mathbf{0.97_{0.1}}$
FRF	$\mathbf{1.02_{0.1}}$	$\mathbf{32.4_{1.2}}$	$\mathbf{0.96_{0.1}}$	$\mathbf{2.22_{0.3}}$	$\mathbf{28.9_{1.3}}$	$\mathbf{1.80_{0.1}}$	$\mathbf{3.73_{0.6}}$	$\mathbf{26.4_{1.2}}$	$\mathbf{0.98_{0.1}}$
QRVF	$7.21_{0.4}$	$19.1_{0.6}$	$5.81_{0.5}$	$14.5_{0.3}$	$16.1_{0.3}$	$6.67_{0.5}$	$22.1_{0.8}$	$14.2_{0.5}$	$7.57_{0.4}$
SFRF	$11.3_{1.5}$	$23.6_{1.1}$	$4.21_{0.9}$	$15.6_{0.9}$	$21.1_{0.5}$	$6.49_{0.7}$	$19.7_{1.7}$	$19.4_{0.6}$	$7.49_{0.6}$
FDF	$7.14_{0.4}$	$19.1_{0.6}$	$5.85_{0.4}$	$14.4_{0.3}$	$16.2_{0.3}$	$9.66_{0.5}$	$21.9_{0.8}$	$14.3_{0.5}$	$2.46_{0.4}$
L0TV	$1.83_{0.4}$	$29.5_{2.3}$	$6.78_{0.3}$	$3.29_{0.8}$	$27.1_{2.2}$	$4.34_{0.4}$	$4.76_{0.9}$	$25.5_{1.9}$	$1.90_{0.5}$
LDCNND	$\mathbf{1.46_{0.3}}$	$\mathbf{33.5_{2.1}}$	$\mathbf{1.00_{0.4}}$	$\mathbf{2.22_{0.4}}$	$\mathbf{30.9_{2.2}}$	$\mathbf{1.63_{0.7}}$	$\mathbf{3.79_{0.6}}$	$\mathbf{27.3_{1.4}}$	$\mathbf{0.28_{0.2}}$
CFAF	$\mathbf{1.27_{0.2}}$	$\mathbf{31.7_{1.2}}$	$\mathbf{0.60_{0.1}}$	$\mathbf{2.63_{0.5}}$	$\mathbf{28.6_{1.4}}$	$\mathbf{2.71_{0.8}}$	$\mathbf{4.38_{0.6}}$	$\mathbf{25.8_{0.8}}$	$\mathbf{0.72_{0.3}}$
FPGF2	$\mathbf{1.21_{0.2}}$	$\mathbf{31.6_{1.4}}$	$\mathbf{0.61_{0.2}}$	$\mathbf{2.28_{0.4}}$	$\mathbf{29.1_{1.5}}$	$\mathbf{1.53_{0.7}}$	$\mathbf{4.16_{0.6}}$	$\mathbf{26.5_{1.0}}$	$\mathbf{0.42_{0.2}}$
SVMF2	$7.21_{0.4}$	$19.1_{0.6}$	$5.82_{0.4}$	$14.5_{0.3}$	$16.1_{0.3}$	$6.64_{0.5}$	$22.1_{0.8}$	$14.3_{0.5}$	$2.43_{0.4}$
INR2	$1.99_{0.7}$	$29.7_{1.7}$	$0.92_{0.1}$	$3.24_{0.8}$	$27.5_{1.7}$	$5.76_{0.2}$	$4.63_{0.8}$	$25.8_{1.2}$	$0.95_{0.2}$
LSAFSVF2	$1.42_{0.3}$	$30.4_{1.4}$	$0.93_{0.1}$	$2.46_{0.4}$	$28.4_{1.5}$	$3.82_{0.1}$	$3.85_{0.5}$	$26.2_{1.1}$	$0.97_{0.1}$
IPGSVF2	$\mathbf{1.18_{0.1}}$	$\mathbf{31.2_{1.1}}$	$\mathbf{0.93_{0.1}}$	$\mathbf{2.35_{0.4}}$	$\mathbf{28.6_{1.7}}$	$\mathbf{1.94_{0.1}}$	$\mathbf{3.64_{0.3}}$	$\mathbf{26.9_{0.9}}$	$\mathbf{0.98_{0.1}}$
FRF2	$\mathbf{1.02_{0.1}}$	$\mathbf{32.3_{1.1}}$	$\mathbf{0.96_{0.1}}$	$\mathbf{2.22_{0.3}}$	$\mathbf{29.0_{1.3}}$	$\mathbf{9.90_{0.1}}$	$\mathbf{3.73_{0.7}}$	$\mathbf{26.4_{1.2}}$	$\mathbf{0.98_{0.1}}$
QRVF2	$7.21_{0.5}$	$19.1_{0.6}$	$4.19_{0.4}$	$14.5_{0.3}$	$16.1_{0.3}$	$6.41_{0.5}$	$22.1_{0.8}$	$14.2_{0.5}$	$7.57_{0.4}$
SFRF2	$11.5_{1.8}$	$23.7_{1.1}$	$3.98_{1.1}$	$15.1_{1.9}$	$21.7_{1.0}$	$6.17_{0.8}$	$18.6_{2.3}$	$20.2_{0.8}$	$7.27_{0.7}$
FDF2	$7.14_{0.4}$	$19.2_{0.5}$	$5.85_{0.4}$	$14.4_{0.3}$	$16.1_{0.3}$	$3.68_{0.5}$	$21.9_{0.8}$	$14.3_{0.5}$	$2.46_{0.4}$
L0TV2	$1.88_{0.5}$	$29.5_{2.4}$	$6.49_{0.3}$	$3.33_{0.8}$	$27.1_{2.3}$	$1.36_{0.4}$	$4.70_{0.9}$	$25.7_{2.0}$	$1.78_{0.5}$
LDCNND2	$\mathbf{1.71_{0.4}}$	$\mathbf{33.6_{2.1}}$	$\mathbf{1.31_{0.6}}$	$\mathbf{2.43_{0.5}}$	$\mathbf{31.0_{2.3}}$	$\mathbf{1.84_{0.9}}$	$\mathbf{3.99_{0.7}}$	$\mathbf{27.3_{1.4}}$	$\mathbf{0.32_{0.2}}$
CFAF2	$\mathbf{1.42_{0.3}}$	$\mathbf{31.5_{1.1}}$	$\mathbf{0.58_{0.1}}$	$\mathbf{2.70_{0.5}}$	$\mathbf{28.8_{1.4}}$	$\mathbf{2.20_{1.0}}$	$\mathbf{4.30_{0.7}}$	$\mathbf{26.3_{1.0}}$	$\mathbf{0.52_{0.2}}$

noise by means of a fuzzy averaging. Later, a correction step is used to solve a lack of robustness of the averaging. We have thoroughly studied the setting of the method parameters and propose a setting that is very convenient to use in the practice as well as robust. The main advantage of our method in front of popular detect and replace approaches is that it does not use any threshold based decision. This makes that it is very flexible and can process properly different types and densities of impulse noise. Experimental results show that, unlike other filters in the state-of-the-art, our approach is always in the group of best performing filters for different noise types and densities.

References

[1] Lukac, R.: Adaptive vector median filtering. Pattern Recogn. Lett. **24 (12)**, 1889-1899 (2003)

[2] Lukac, R., Smolka, B., Plataniotis, K.N., Venetsanopoulos, A.N.: Vector sigma filters for noise detection and removal in color images. J. Vis. Commun. Image R. **17 (1)**, 1-26 (2006)

Table 8: Mean value and standard deviation (as subindex) of performance over validation images using noise type III.

Filter	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
FPGF	$2.83_{0.4}$	$28.2_{0.6}$	$5.73_{1.2}$	$5.48_{0.4}$	$24.7_{0.6}$	$19.2_{3.0}$	$9.06_{1.0}$	$22.3_{0.8}$	$3.38_{0.4}$
SVMF	$7.34_{0.1}$	$19.0_{0.3}$	$9.44_{1.4}$	$14.5_{0.2}$	$16.0_{0.3}$	$73.1_{4.8}$	$21.8_{0.3}$	$14.3_{0.3}$	$8.16_{0.4}$
INR	$\mathbf{1.55_{0.3}}$	$\mathbf{30.9_{1.2}}$	$\mathbf{3.38_{0.5}}$	$\mathbf{3.24_{0.4}}$	$\mathbf{27.1_{0.9}}$	$\mathbf{12.3_{4.4}}$	$\mathbf{5.98_{0.6}}$	$\mathbf{23.4_{0.8}}$	$\mathbf{3.05_{0.4}}$
LSAFSVF	$2.91_{0.2}$	$27.3_{0.8}$	$0.89_{0.3}$	$5.47_{0.1}$	$24.3_{0.3}$	$22.1_{2.6}$	$8.78_{0.45}$	$21.4_{0.6}$	$4.10_{0.3}$
IPGSVF	$3.03_{0.2}$	$26.4_{0.6}$	$5.19_{0.8}$	$5.05_{0.3}$	$25.0_{0.5}$	$16.3_{1.6}$	$7.8_{0.71}$	$23.1_{0.8}$	$2.68_{0.5}$
FRF	$2.67_{0.2}$	$27.9_{0.6}$	$9.36_{1.4}$	$5.33_{0.5}$	$24.8_{0.7}$	$12.7_{1.3}$	$9.64_{1.2}$	$21.3_{1.0}$	$2.69_{0.3}$
QRVF	$7.34_{0.1}$	$19.0_{0.3}$	$9.44_{1.4}$	$14.5_{0.2}$	$16.1_{0.3}$	$73.1_{4.8}$	$21.8_{0.3}$	$14.3_{0.3}$	$8.16_{0.4}$
SFRF	$11.2_{1.4}$	$23.7_{0.9}$	$9.07_{1.5}$	$15.6_{1.3}$	$21.3_{0.6}$	$69.8_{6.1}$	$19.7_{1.5}$	$19.4_{0.6}$	$7.86_{0.5}$
FDF	$7.29_{0.1}$	$19.1_{0.3}$	$5.43_{0.4}$	$14.4_{0.2}$	$16.1_{0.3}$	$73.2_{4.8}$	$21.7_{0.3}$	$14.3_{0.3}$	$8.15_{0.4}$
L0TV	$\mathbf{1.90_{0.4}}$	$\mathbf{29.4_{2.4}}$	$\mathbf{9.28_{2.0}}$	$\mathbf{3.12_{0.5}}$	$\mathbf{27.5_{1.9}}$	$\mathbf{15.2_{6.0}}$	$\mathbf{4.82_{0.9}}$	$\mathbf{25.4_{1.8}}$	$\mathbf{2.44_{0.7}}$
LDCNND	$\mathbf{3.24_{0.2}}$	$\mathbf{28.6_{0.7}}$	$\mathbf{8.38_{1.4}}$	$\mathbf{5.16_{0.4}}$	$\mathbf{25.8_{0.7}}$	$\mathbf{16.6_{3.9}}$	$\mathbf{8.04_{0.8}}$	$\mathbf{24.1_{0.8}}$	$\mathbf{2.24_{0.4}}$
CFAF	$\mathbf{2.49_{0.3}}$	$\mathbf{29.6_{1.1}}$	$\mathbf{3.14_{0.6}}$	$\mathbf{4.82_{0.4}}$	$\mathbf{26.5_{0.7}}$	$\mathbf{12.0_{4.9}}$	$\mathbf{7.84_{0.8}}$	$\mathbf{23.5_{0.8}}$	$\mathbf{2.95_{0.6}}$
FPGF2	$2.80_{0.3}$	$28.3_{0.9}$	$5.79_{1.1}$	$5.51_{0.5}$	$25.2_{0.8}$	$15.6_{2.8}$	$9.05_{0.9}$	$22.6_{0.8}$	$3.53_{0.4}$
SVMF2	$7.34_{0.1}$	$19.0_{0.3}$	$9.44_{1.4}$	$14.5_{0.2}$	$16.1_{0.3}$	$73.1_{4.8}$	$21.8_{0.3}$	$14.3_{0.3}$	$8.16_{0.4}$
INR2	$\mathbf{1.97_{0.7}}$	$\mathbf{30.3_{1.9}}$	$\mathbf{1.95_{0.7}}$	$\mathbf{3.23_{0.6}}$	$\mathbf{28.2_{1.4}}$	$\mathbf{06.0_{2.4}}$	$\mathbf{4.96_{0.7}}$	$\mathbf{26.1_{1.2}}$	$\mathbf{1.46_{0.4}}$
LSAFSVF2	$2.91_{0.2}$	$27.6_{0.8}$	$0.71_{0.6}$	$5.11_{0.2}$	$25.2_{0.6}$	$15.9_{3.2}$	$7.77_{0.5}$	$22.9_{0.7}$	$3.01_{0.3}$
IPGSVF2	$3.02_{0.2}$	$26.4_{0.6}$	$3.17_{0.3}$	$4.98_{0.3}$	$25.2_{0.5}$	$14.7_{0.9}$	$7.44_{0.7}$	$23.9_{0.9}$	$2.02_{0.3}$
FRF2	$2.63_{0.2}$	$28.2_{0.6}$	$9.62_{0.3}$	$5.21_{0.6}$	$25.1_{1.0}$	$11.2_{1.8}$	$9.00_{1.2}$	$22.2_{1.2}$	$2.14_{0.2}$
QRVF2	$7.34_{0.1}$	$19.0_{0.3}$	$9.44_{0.4}$	$14.5_{0.2}$	$16.1_{0.3}$	$73.1_{4.8}$	$21.8_{0.3}$	$14.3_{0.3}$	$8.16_{0.4}$
SFRF2	$11.6_{1.7}$	$23.8_{1.1}$	$8.68_{0.7}$	$15.2_{2.1}$	$21.7_{1.0}$	$66.9_{7.5}$	$18.7_{2.4}$	$20.2_{0.9}$	$7.62_{0.6}$
FDF2	$7.29_{0.1}$	$19.1_{0.3}$	$9.43_{0.4}$	$14.4_{0.2}$	$16.1_{0.3}$	$73.2_{4.8}$	$21.7_{0.3}$	$14.3_{0.3}$	$8.15_{0.4}$
L0TV2	$\mathbf{1.95_{0.5}}$	$\mathbf{29.4_{2.4}}$	$\mathbf{9.00_{3.8}}$	$\mathbf{3.11_{0.5}}$	$\mathbf{27.6_{2.0}}$	$\mathbf{15.1_{5.3}}$	$\mathbf{4.73_{1.0}}$	$\mathbf{25.6_{2.0}}$	$\mathbf{2.30_{0.7}}$
LDCNND2	$\mathbf{3.53_{0.3}}$	$\mathbf{29.4_{1.0}}$	$\mathbf{7.38_{1.7}}$	$\mathbf{5.27_{0.5}}$	$\mathbf{26.7_{0.9}}$	$\mathbf{15.0_{4.6}}$	$\mathbf{8.16_{0.8}}$	$\mathbf{24.2_{0.9}}$	$\mathbf{2.22_{0.5}}$
CFAF2	$\mathbf{2.61_{0.4}}$	$\mathbf{29.6_{1.1}}$	$\mathbf{2.79_{0.7}}$	$\mathbf{4.73_{0.4}}$	$\mathbf{27.1_{0.7}}$	$\mathbf{09.3_{3.9}}$	$\mathbf{7.32_{0.7}}$	$\mathbf{24.6_{0.8}}$	$\mathbf{2.33_{0.6}}$

[3] Camarena, J.G., Gregori, V., Morillas, S., Sapena, A.: Two-step fuzzy logic-based method for impulse noise detection in colour images. Pattern Recogn. Lett. **31 (13)**, 1842-1849 (2010)

[4] Smolka, B., Chydzinski, A.: Fast detection and impulsive noise removal in color images. Real-Time Imaging. **11 (5-6)**, 389-402 (2005)

[5] Camarena, J.G., Gregori, V., Morillas, S., Sapena, A.: Some improvements for image filtering using peer group techniques. Image Vis. Comput. **28 (1)**, 188-201 (2010)

[6] Jin, L., Liu, H., Xu, X., Song E.: Color impulsive noise removal based on quaternion representation and directional vector order-statistics. Signal Process. **91**, 1249-1261 (2011)

[7] Geng, X. Hu, X., Xiao, J.: Quaternion switching filter for impulse noise reduction in color image. Signal Process. **92**, 150-162 (2012)

[8] Schulte, S., De Witte, V., Nachtegael, M., Van der Weken, D., Kerre, E.E.:

Table 9: Mean value and standard deviation (as subindex) of performance over validation images using noise type IV.

Filter	$p = 10\%$			$p = 20\%$			$p = 30\%$		
	MAE	PSNR	DIS^2	MAE	PSNR	DIS^2	MAE	PSNR	DIS^1
FPGF	$4.29_{0.1}$	$28.2_{0.4}$	$1.09_{0.2}$	$5.26_{0.2}$	$26.8_{0.3}$	$13.7_{2.0}$	$6.75_{0.5}$	$25.2_{0.7}$	$2.01_{0.2}$
SVMF	$7.76_{0.3}$	$20.3_{0.2}$	$4.61_{0.4}$	$10.5_{0.9}$	$18.4_{0.5}$	$58.9_{4.5}$	$14.9_{0.6}$	$16.3_{0.1}$	$7.14_{0.3}$
INR	$\mathbf{3.42_{0.2}}$	$\mathbf{30.3_{1.0}}$	$\mathbf{0.58_{0.2}}$	$\mathbf{3.93_{0.2}}$	$\mathbf{28.9_{0.8}}$	$\mathbf{09.5_{3.0}}$	$\mathbf{4.78_{0.3}}$	$\mathbf{26.8_{0.7}}$	$\mathbf{1.58_{0.4}}$
LSAFSVF	$4.62_{0.1}$	$27.3_{0.5}$	$1.19_{0.3}$	$5.41_{0.2}$	$26.3_{0.4}$	$15.9_{2.7}$	$6.62_{0.3}$	$24.7_{0.6}$	$2.29_{0.3}$
IPGSVF	$4.72_{0.1}$	$26.5_{0.4}$	$1.50_{0.3}$	$5.17_{0.3}$	$26.5_{0.7}$	$14.4_{4.4}$	$6.01_{0.6}$	$25.9_{1.1}$	$1.49_{0.4}$
FRF	$4.26_{0.1}$	$28.2_{0.3}$	$0.92_{0.2}$	$5.21_{0.2}$	$26.7_{0.6}$	$11.3_{1.3}$	$6.87_{0.9}$	$24.8_{1.2}$	$1.51_{0.3}$
QRVF	$7.76_{0.3}$	$20.3_{0.2}$	$4.61_{0.4}$	$10.5_{0.9}$	$18.4_{0.5}$	$58.9_{4.5}$	$14.9_{0.6}$	$16.3_{0.1}$	$7.14_{0.3}$
SFRF	$10.0_{0.9}$	$24.6_{0.9}$	$4.12_{0.6}$	$11.6_{0.5}$	$23.5_{0.5}$	$52.8_{5.1}$	$13.7_{0.7}$	$22.2_{0.6}$	$6.64_{0.4}$
FDF	$7.70_{0.3}$	$20.3_{0.2}$	$4.58_{0.4}$	$10.4_{0.9}$	$18.5_{0.5}$	$58.8_{4.4}$	$14.8_{0.6}$	$16.3_{0.1}$	$7.12_{0.3}$
L0TV	$3.88_{0.3}$	$28.8_{1.2}$	$1.24_{0.4}$	$4.40_{0.4}$	$27.8_{1.5}$	$14.9_{4.5}$	$5.09_{0.8}$	$26.8_{1.9}$	$2.05_{0.7}$
LDCNND	$4.03_{0.3}$	$29.1_{0.8}$	$1.05_{0.2}$	$4.51_{0.3}$	$28.3_{0.7}$	$11.8_{3.0}$	$6.04_{0.9}$	$26.3_{1.4}$	$1.49_{0.6}$
CFAF	$\mathbf{4.08_{0.1}}$	$\mathbf{29.3_{0.5}}$	$\mathbf{0.72_{0.3}}$	$\mathbf{4.75_{0.2}}$	$\mathbf{28.2_{0.6}}$	$\mathbf{09.7_{2.8}}$	$\mathbf{5.71_{0.3}}$	$\mathbf{26.8_{0.8}}$	$\mathbf{1.40_{0.4}}$
FPGF2	$4.41_{0.2}$	$28.1_{0.4}$	$1.07_{0.2}$	$5.26_{0.2}$	$27.1_{0.4}$	$12.9_{1.8}$	$7.55_{0.5}$	$25.3_{0.7}$	$2.03_{0.2}$
SVMF2	$7.76_{0.3}$	$20.3_{0.2}$	$4.61_{0.4}$	$10.5_{0.9}$	$18.4_{0.5}$	$58.9_{4.5}$	$14.9_{0.6}$	$16.3_{0.1}$	$7.14_{0.3}$
INR2	$\mathbf{3.76_{0.5}}$	$\mathbf{29.6_{1.3}}$	$\mathbf{0.52_{0.2}}$	$\mathbf{4.18_{0.5}}$	$\mathbf{28.8_{1.3}}$	$\mathbf{06.9_{2.0}}$	$\mathbf{4.74_{0.6}}$	$\mathbf{27.8_{1.2}}$	$\mathbf{0.98_{0.3}}$
LSAFSVF2	$4.65_{0.2}$	$27.3_{0.5}$	$1.11_{0.3}$	$5.41_{0.2}$	$26.4_{0.4}$	$14.6_{3.2}$	$6.49_{0.4}$	$25.1_{0.7}$	$2.02_{0.4}$
IPGSVF2	$4.72_{0.2}$	$26.5_{0.4}$	$1.49_{0.3}$	$5.16_{0.3}$	$26.5_{0.7}$	$14.4_{4.6}$	$5.99_{0.6}$	$26.0_{1.1}$	$1.43_{0.4}$
FRF2	$4.29_{0.1}$	$28.1_{0.4}$	$0.90_{0.2}$	$5.19_{0.3}$	$26.8_{0.6}$	$11.1_{1.3}$	$6.82_{0.9}$	$24.9_{1.2}$	$1.43_{0.3}$
QRVF2	$7.76_{0.3}$	$20.3_{0.2}$	$4.61_{0.4}$	$10.5_{1.0}$	$18.4_{0.5}$	$58.9_{4.5}$	$14.9_{0.6}$	$16.3_{0.1}$	$7.14_{0.3}$
SFRF2	$10.7_{1.8}$	$24.3_{1.5}$	$3.83_{0.7}$	$11.0_{1.3}$	$24.1_{1.1}$	$47.9_{4.3}$	$12.7_{1.7}$	$22.9_{1.2}$	$6.16_{0.4}$
FDF2	$7.70_{0.3}$	$20.3_{0.2}$	$4.58_{0.4}$	$10.4_{1.0}$	$18.5_{0.5}$	$58.8_{4.4}$	$14.8_{0.6}$	$16.3_{0.1}$	$7.12_{0.3}$
L0TV2	$3.89_{0.3}$	$28.8_{1.2}$	$1.21_{0.4}$	$4.42_{0.4}$	$27.8_{1.6}$	$15.2_{4.6}$	$4.99_{0.6}$	$27.0_{1.8}$	$1.93_{0.6}$
LDCNND2	$4.00_{0.3}$	$30.1_{1.0}$	$0.89_{0.3}$	$4.45_{0.4}$	$29.3_{0.9}$	$10.5_{3.6}$	$6.04_{0.9}$	$26.6_{1.6}$	$1.41_{0.7}$
CFAF2	$\mathbf{4.18_{0.2}}$	$\mathbf{29.2_{0.6}}$	$\mathbf{0.69_{0.3}}$	$\mathbf{4.85_{0.2}}$	$\mathbf{28.1_{0.8}}$	$\mathbf{09.2_{2.9}}$	$\mathbf{5.71_{0.3}}$	$\mathbf{27.0_{0.8}}$	$\mathbf{1.31_{0.3}}$

Fuzzy two-step filter for impulse noise reduction from color images. IEEE Trans. Image Process. **15 (11)**, 3567-3578 (2006)

[9] Morillas S., Gregori V., Peris-Fajarnes, G.: Isolating impulsive noise pixels in color images by peer group techniques, *Computer Vision and Image Understanding* **110 (1)**, 102-116 (2008).

[10] Schulte, S. Morillas, S., Gregori, V., Kerre, E.E.: A New Fuzzy Color Correlated Impulsive Noise Reduction Method. IEEE Trans. Image Process. **16 (10)**, 2565-2575 (2007).

[11] Morillas, S., Gregori, V., Peris-Fajarnés, G., Sapena, A. Local Self-Adaptive Fuzzy Filter for Impulsive Noise Removal in Color Images. Signal Process. **88 (2)**, 390-398 (2008)

[12] Smolka, B., Malik, K., Malik, D. Adaptive rank weighted switching filter for impulsive noise removal in color images, Journal of Real-Time Image Processing 10 (2) pp. 289-311 (2015).

[13] Y. Chen, Y. Zhang, H. Shu, J. Yiang, L. Luo, J.L. Coatrieux, Q. Feng, Structure-Adaptive Fuzzy Estimation for Random-Valued Impulse Noise Suppression, IEEE Transactions on Circuits and systems for video technology **28** (2) 414-427 (2018).

[14] A. Roy, L. Manam, R.H. Laskar, Region-Adaptive Fuzzy Filter: An Approach for Removal of Random-Valued Impulse Noise, IEEE Transactions on Industrial Electronics **65** (9) 7268-7278 (2018).

[15] Meher, S.K.: Recursive and noise-exclusive fuzzy switching median filter for impulse noise reduction. Eng. Appl. Artif. Intel. **30**, 145-154 (2014)

[16] Camarena, J.G., Gregori, V., Morillas, S., Sapena, A.: A Simple Fuzzy Method to Remove Mixed Gaussian-Impulsive Noise From Color Images. IEEE Trans. Fuzzy Syst. **21 (5)**, 971-978 (2013)

[17] Qin, H., Yang, S.X.: Adaptive neuro-fuzzy inference systems based approach to nonlinear noise cancellation for images. Fuzzy Set. Syst. **158 (10)**, 1036-1063 (2007)

[18] Mukhopadhyay, S., Mandal, J.K.: A fuzzy switching median filter of impulses in digital imagery. Circuits Syst. Signal Process. **33**, 2193-2216 (2014)

[19] Pushpavalli, R., Sivarajde, G.: A fuzzy switching median filter for highly corrupted images. International Journal of Scientific and Research Publications **3 (6)**, 1-6 (2013)

[20] Ramakrisnan, E., Karthik, B., Kiran Kumar T.V.U.: Noise reduction and removal using fuzzy based median filter and matrix algorithm. International Journal of Research **2 (4)**, 841-845 (2015)

[21] Kuo, Y, Tai, Ch.: A simple and efficient median filter for removing high-density impulse noise in images. Int. J. Fuzzy. Syst. **17 (1)**, 67-75 (2015)

[22] Aborisade, D.O.: A novel fuzzy logic based impulse noise filtering technique. International Journal of Advanced Science and Technology **32**, 79-87 (2011)

[23] Thirilogasundari, V., Suresh babu, V., Agatha Janet, S.: Fuzzy based salt and pepper noise removal using adaptive switching median filter. Procedia Engineering **38**, 2858-2865 (2012)

[24] Wang, G., Zhu, H., Wang, Y.: Fuzzy decision filter for color images denoising. Optik **126**, 2428-2432 (2015).

[25] Lukac, R.: Adaptive vector median filtering. Pattern Recognition Letters **24**, 1889-1899 (2003).

[26] Rahman, T., Uddin, M.S.: Removal of high density impulse noise from color images using an adaptive fuzzy filter. International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT) (2014).

[27] G. Yuan, B. Ghanem, L0TV: A new method for image restoration in the presence of impulse noise, Computer Vision and Patter Recognition 2015, 5369-5377.

[28] K.H. Jin, J.C. Ye, Sparse and Low-Rank Decomposition of a Hankel Structured Matrix for Impulse Noise Removal, IEEE Transactions on Image Processing **27** (3) 1448-1461 (2018).

[29] Kai Zhang, Wangmeng Zuo, Shuhang Gu, Lei ZhangLearning Deep CNN Denoiser Prior for Image Restoration. Conference on Computer Vision and Pattern Recognition, 2017.

[30] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a Gaussian Denoisier: Residual Learning of Deep CNN for Image Denoising, IEEE Transactions on Image Processing **26** (7) 3142-3155 (2017).

[31] B.H. Chen, J.L. Yin, Y. Li, Image Noise Removing Using Semi-Supervised Learning on Big Image Data, 2017 IEEE Third International Conference on Multimedia Big Data 338-345.

[32] B.H. Chen, J.L. Yin, Y. Li, Highly accurate image reconstruction for multimodal noise suppression using semisupervised learning on Big Data, to appear in IEEE Transactions on Multimedia.

[33] S.D. Nguyen, S.B. Choi, T.I. Seo, Recurrent Mechanism and Impulse Noise Filter for Establishing ANFIS, IEEE Transactions on Fuzzy Systems **26** (2) 985-997 (2018).

[34] Peris-Fajarnés, G., Roig, B., Vidal, A.: Rank-Ordered Differences Statistic Based Switching Vector Filter, *Lecture Notes in Computer Science* **4141**, 74-81 (2006).

[35] Morillas, S. Gregori, V.: Robustifying Vector Median Filter, *Sensors (Basel)* **11 (8)**, 8115-8126 (2011).

[36] Garnett, R., Huegerich, T., Chui, C., He, W.: A universal noise removal algorithm with an impulse detector, *IEEE Transactions on Image Processing* **14 (11)**, 1747-1754 (2005).

[37] Celebi, M.E., Lecca, M., Smolka, B.: *Color Image and Video Enhancement.* Springer, 2015.

[38] Kuruoglu, E.E., Molina, C., Godsill, S.J., Fitzgerald, W.J.: A new analytic representation for the α-stable probability density function. In: The Fifth World Meeting of the International Society for Bayesian Analysis (ISBA), Istanbul, August (1997)

[39] Hassan, M., Bhagvati, C.: Structural Similarity Measure for Color Images. International Journal of Computer Applications **43 (14)**, 7-12 (2012)