A Fuzzy Impulse Noise Detection and Reduction Method

Stefan Schulte, Mike Nachtegael, Valérie De Witte,
Dietrich Van der Weken and Etienne E. Kerre

Department of Applied Mathematics and Computer Science
Fuzziness and Uncertainty Modelling Research Unit
Ghent University, Krijgslaan 281 (Building S9), B-9000 Gent, Belgium
Email: Stefan.Schulte@UGent.be, Etienne.Kerre@UGent.be
Phone: +32 9 264 47 65, Fax: +32 9 264 49 95
http://www.fuzzy.UGent.be/

***Abstract*— Removing or reducing impulse noise is a very active research area in image processing. In this paper we describe a new algorithm that is especially developed for reducing all kinds of impulse noise: FIDRM (Fuzzy Impulse noise Detection and Reduction Method). It can also be applied to images having a mixture of impulse noise and other types of noise. The result is an image quasi without (or with very little) impulse noise so that other filters can be used afterwards. This nonlinear filtering technique contains two separated steps: an impulse noise detection step and a reduction step that preserves edge sharpness. Based on the concept of fuzzy gradient values our detection method constructs a fuzzy set *impulse noise*. This fuzzy set is represented by a membership function that will be used by the filtering method, which is a fuzzy averaging of neighbouring pixels. Experimental results show that FIDRM provides a significant improvement on other existing filters. FIDRM is not only very fast but also very effective for reducing little as well as very high impulse noise.**

***Index Terms*— Image processing, fuzzy filter, membership functions, impulse noise, noise reduction.**

I. Introduction

IN the literature several (fuzzy and non-fuzzy) filters have been studied for impulse noise reduction [1]- [5]. Impulse noise is caused a.o. by errors in the data transmission generated in noisy sensors or communication channels, or by errors during the data capture from digital cameras. Noise is usually quantified by the percentage of pixels which are corrupted. Corrupted pixels are either set to the maximum value or have single bits flipped over. In some cases, single pixels are set alternatively to zero or to the maximum value. This is the most common form of impulse noise and is called salt and pepper noise. Nevertheless other types of impulse noise are possible as well.

In this work we will present a new, faster and more efficient noise reduction method for images corrupted with impulse noise. For clarity we first give the definition of impulse noise. Afterwards we introduce the new filter called **F**uzzy **I**mpulse noise **D**etection and **R**eduction **M**ethod (**FIDRM**). The rest of the paper is structured as follows: The details of the detection phase are given in Section II. In Section III the filtering step is discussed. Experimental results and conclusions are finally presented in Section IV and Section V.

A. Impulse noise for greyscale images

A Monochrome (greyscale) digital image O is often represented by a two-dimensional array where an address (i, j) defines a position in O, called a pixel or picture element. In a greyscale (or greylevel) image the only colours are shades of grey. A 'grey' colour is one in which the red, green and blue components all have equal intensity in the RGB space, so it is only necessary to specify one single intensity value for each pixel, as opposed to the three intensities needed to specify a pixel in a full colour image. Often, the (greyscale) intensity is stored as an 8-bit integer giving 256 possible different shades of grey going from black to white, which can be represented as a $[0, 255]$- integer interval. In this interval we consider several integer values p_1, p_2, ..., p_n with ($p_k \neq p_l$) and $n \leq 255$. If $O(i, j)$ denotes the pixel value of the (two-dimensional) image O at position (i, j), then we can model the occurrence of impulse noise, for greyscale images, as:

$$A(i,j) \quad = \quad \begin{cases} O(i,j) & \text{with probability } 1 - pr \\ p_1 & \text{with probability } pr_1 \\ p_2 & \text{with probability } pr_2 \\ \vdots & \vdots \\ p_n & \text{with probability } pr_n \end{cases} \tag{1}$$

where pr is the probability that a pixel is corrupted and where A is the corrupted image. The value pr_k (with $k \in \{1, .., n\}$) indicates the probability that an original image pixel becomes p_k. Of course the following restriction must be valid: $\sum_{k=1}^{n} pr_k = pr$. In the case of saturated impulse noise (salt and pepper noise) there are only two values p_1 and p_2, which are the maximum and the minimum pixel value of the considered integer interval (in our case respectively 255 and 0). This definition of impulse noise is a simplification of a more general noise model in which a noisy pixel can take on arbitrary values in the dynamic range according to some underlying probability distribution. Let $O(i, j)$ and $\eta(i, j)$

denote the luminance values of the original and the noisy image, respectively, at position (i, j). Then, we have

$$A(i,j) \quad = \quad \begin{cases} O(i,j) & \text{with probability } 1 - pr \\ \eta(i,j) & \text{with probability } pr \end{cases} \tag{2}$$

where $\eta(i, j)$ is an identically distributed, independent random process with an arbitrary underlying probability density function. In this paper we concentrate on the more common discrete impulse noise model from Eq. 1. In case of random impulse noise (Eq. 2) the global detection and reduction method discussed here, can be changed to a local method.

B. A Fuzzy Impulse noise Detection and Reduction Method

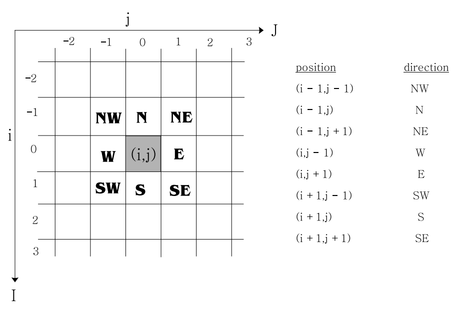
We introduce a new two step filter called "**F**uzzy **I**mpulse noise **D**etection and **R**eduction **M**ethod" (**FIDRM**). This new filter has two separated steps or phases: the detection phase and the filtering phase. The detection phase uses fuzzy rules to determine whether a pixel is corrupted with impulse noise or not. When impulse noise is detected we try to indicate the values p_k ($k \in \{1, ..., n\}$ with $1 \leq n < 255$). Moreover some parameters will be determined which will be passed to the filtering phase. After this detection our fuzzy filtering technique focuses only on the p_k values, i.e. the filtering is concentrated on the real noisy pixels.

II. Fuzzy Impulse Noise Detection

Our noise detection method uses fuzzy gradient values as introduced with the GOA filter [6], [7] to determine if a certain pixel is corrupted with impulse noise or not.

A. Fuzzy gradient values

For each pixel (i, j) of the image (that isn't a border pixel) we use a 3×3 neighbourhood window as illustrated in Fig. 1. Each neighbour with respect to (i, j) corresponds to one direction {NW = North West, N = North, NE = North East, W = West, E = East, SW = South West, S = South, SE = South East}. Each such direction with respect to (i, j) can also be linked to a certain position (also indicated in Fig. 1).



position	direction
(i − 1,j − 1)	NW
(i − 1,j)	N
(i − 1,j + 1)	NE
(i,j − 1)	W
(i,j + 1)	E
(i + 1,j − 1)	SW
(i + 1,j)	S
(i + 1,j + 1)	SE

Fig. 1. The neighbourhood of a central pixel.

If we denote A as the input image, then the gradient $\nabla_{(k,l)}A(i, j)$ is defined as the difference:

$$\nabla_{(k,l)}A(i,j) \quad = \quad A(i+k, j+l) - A(i,j) \tag{3}$$
$$\text{with} \quad k,l \in \{-1, 0, 1\},$$

where the pair (k, l) corresponds to one of the eight directions and (i, j) is called the centre of the gradient. The eight gradient values (according to the eight different directions or neighbours) are called the basic gradient values. One such gradient value with respect to (i, j) can be used to determine if a central pixel is corrupted with impulse noise or not, because if this gradient is quite large then it is a good indication that some noise is present in the central pixel (i, j). But there are two cases in which this conclusion is wrong:

1) If the central pixel isn't noisy but one of the neighbours is, then this can also cause large gradient values.
2) An edge in an image causes some kind of natural large gradient values.

To handle the first case we use not only one gradient value but eight different gradient values (according to the eight different directions) to make a conclusion. And to solve the second case we use not one basic gradient for each direction but one basic and two related gradient values for each direction. The two related gradient values in the **same direction**, are determined by the **centres** making a right-angle with the direction of the first (basic) gradient. For example (Fig. 2), in the NW-direction (i.e. for $(k, l) = (-1, -1)$) we calculate the basic gradient value $\nabla_{(-1,-1)}A(i, j)$ plus the two related gradient values $\nabla_{(-1,-1)}A(i-1, j+1)$ and $\nabla_{(-1,-1)}A(i+1, j-1)$. The two extra gradient values are used for making the separation between noisy pixels and edge pixels: when all these gradients are large, then (i, j) is considered to be not a noisy but an edge pixel. In Table I we give an overview of the involved gradient

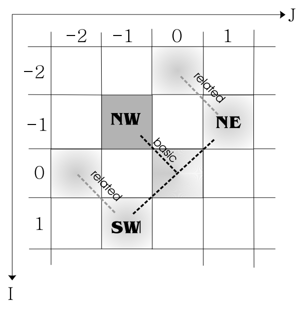


Fig. 2. Involved centres for the calculation of the related gradient values in the NW-direction.

values: each direction R (column 1) corresponds to a position (Fig. 1) with respect to a central position. Column 2 gives the basic gradient for each direction, column 3 gives the two related gradients.

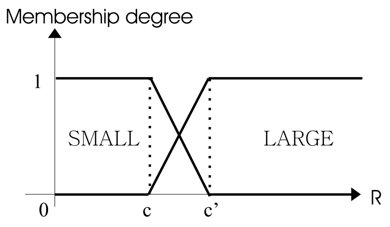
TABLE I
INVOLVED GRADIENT VALUES TO CALCULATE THE FUZZY GRADIENT.

R	basic gradient	related gradients
NW	$\nabla_{NW}A(i,j)$	$\nabla_{NW}A(i+1,j-1), \nabla_{NW}A(i-1,j+1)$
N	$\nabla_{N}A(i,j)$	$\nabla_{N}A(i,j-1), \nabla_{N}A(i,j+1)$
NE	$\nabla_{NE}A(i,j)$	$\nabla_{NE}A(i-1,j-1), \nabla_{NE}A(i+1,j+1)$
E	$\nabla_{E}A(i,j)$	$\nabla_{E}A(i-1,j), \nabla_{E}A(i+1,j)$
SE	$\nabla_{SE}A(i,j)$	$\nabla_{SE}A(i-1,j+1), \nabla_{SE}A(i+1,j-1)$
S	$\nabla_{S}A(i,j)$	$\nabla_{S}A(i,j-1), \nabla_{S}A(i,j+1)$
SW	$\nabla_{SW}A(i,j)$	$\nabla_{SW}A(i-1,j-1), \nabla_{SW}A(i+1,j+1)$
W	$\nabla_{W}A(i,j)$	$\nabla_{W}A(i-1,j), \nabla_{W}A(i+1,j)$

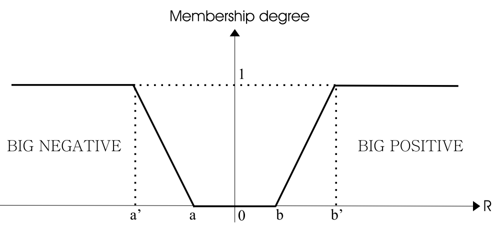
Finally we define eight fuzzy gradient values for each of the eight directions. These values indicate in which degree the central pixel can be seen as an impulse noise pixel. The fuzzy gradient value $\nabla_R^F A(i,j)$ for direction R ($R \in \{NW, N, NE, E, SE, S, SW, W\}$), is calculated by the following fuzzy rule:

```
IF   $|\nabla_R\ A(i,j)|$ is large AND $|\nabla_R'\ A(i,j)|$ is small
       OR
     $|\nabla_R\ A(i,j)|$ is large AND $|\nabla_R''\ A(i,j)|$ is small
       OR
     $\nabla_R\ A(i,j)$ is big positive AND $\Big(\nabla_R'\ A(i,j)$
               AND $\nabla_R''\ A(i,j)\Big)$ are big negative
       OR
     $\nabla_R\ A(i,j)$ is big negative AND $\Big(\nabla_R'\ A(i,j)$
               AND $\nabla_R''\ A(i,j)\Big)$ are big positive
THEN $\nabla_R^F\ A(i,j)$ is large
```

where $\nabla_R A(i,j)$ is the basic gradient value and $\nabla_R' A(i,j)$ and $\nabla_R'' A(i,j)$ are the two related gradient values for the direction R. Because "large", "small", "big negative" and "big positive" are non-deterministic features, these terms can be represented as fuzzy sets [8]. Fuzzy sets can be represented by a membership function. Examples of the membership functions LARGE (for the fuzzy set *large*), SMALL (for the fuzzy set *small*), BIG POSITIVE (for the fuzzy set *big positive*) and BIG NEGATIVE (for the fuzzy set *big negative*) are shown in Fig. 3.



(a)



(b)

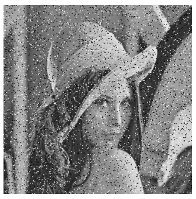
Fig. 3. The membership functions `SMALL` respectively `LARGE` (a), `BIG NEGATIVE` respectively `BIG POSITIVE` (b).

The horizontal axis of these functions (Fig. 3) represents all the possible gradient values (the universe $[-255, 255]$) and the vertical axis represents a membership degree ($\in [0,1]$). A membership degree indicates the degree in which a certain gradient value matches the predicate (e.g. large). If a gradient value has membership degree one, for the fuzzy set *large*, it means that it is large for sure.

The fuzzy rule contains some conjunctions and disjunctions. In fuzzy logic triangular norms and co-norms are used to represent conjunctions [9] (roughly the equivalent of AND operators) and disjunctions (roughly the equivalent of OR operators). Some well-known triangular norms (together with their dual co-norms) are the minimum (maximum) and the product (probabilistic sum) [10]. So we can for example translate the sub-rule "$|\nabla_R\ A(i,j)|$ *is large AND* $|\nabla_R'\ A(i,j)|$ *is small*" as $min\Big(LARGE(|\nabla_R\ A(i,j)|), SMALL(|\nabla_R'\ A(i,j)|)\Big)$, where LARGE and SMALL are the membership functions shown in Fig. 3(a). These two membership functions depend on the two parameters c and c'. According to the following three observations we can derive these values:

1) Gradient values for a given direction R, which are situated in the interval [0,40] are most likely non-edge and non-noisy pixels but very fine detail pixels as illustrated in Fig. 4(b). So these pixels can be categorized as noise free without (or with very little) uncertainty, resulting in a zero membership degree in the fuzzy set *impulse noise*.
2) Gradient values for a given direction R, which are situated in the interval [40,125] are most likely edge pixels or noise pixels as shown in Fig. 4(c). Here we have some kind of uncertainty which is expressed by the membership degrees in the fuzzy set *impulse noise*: these membership degrees are now situated between zero and one.
3) Finally gradient values for a given direction R, which are situated in the interval [125,255] are most likely noise pixels as shown in Fig. 4(d). In this case the membership degrees in the fuzzy set *impulse noise* are one.

Since we are searching for noise pixels, we choose $c \in [50, 80]$ and $c' \in [100, 150]$, which are good choices for our method. The idea behind the usage of the fuzzy sets *big positive* and *big negative* is that if the basic gradient and the two related gradients are both large but have different signs then it is a good indication that noise is present. Therefore we also use



(a)

(b) (c) (d)

Fig. 4. For a noisy Lena image (a) we display (in white): all $|\nabla_{NW}\ A(i,j)| \in [0,40]$ (b), $|\nabla_{NW}\ A(i,j)| \in [40,125]$ (c), $|\nabla_{NW}\ A(i,j)| \in [125,255]$ (d).

the fuzzy sets *big negative* and *big positive*. We represent these fuzzy sets by membership functions pictured in Fig. 3b. Gradient values around zero are seen as more or less unsigned and gradient values above 15 or under -15 become significant to matching the feature big positive respectively big negative. Therefore we choose $a \in [-15, -10]$, $b \in [10, 15]$, $a' \in [-25, -15]$ and $b' \in [15, 25]$. Finally, after choosing a T-norm (triangular norm) and a S-norm (triangular co-norm) we can calculate the activation degree of such an "IF-THEN" rule. This activation degree also indicates the degree in which $\bigtriangledown_R^F$ can be considered as large. So the calculated activation degree will be used as a membership degree for $\bigtriangledown_R^F$ in the fuzzy set *large*.

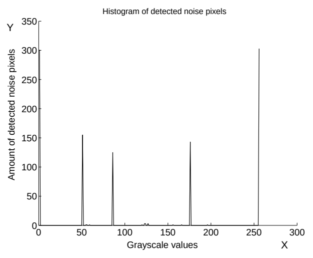
B. Detection method

To decide if a central pixel (a non-border pixel of course) is an impulse noise pixel we use following (fuzzy) rule:

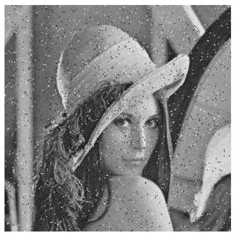
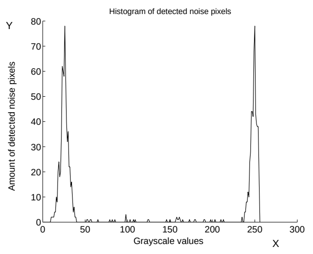
```
IF    most of the eight $\bigtriangledown_R^F A(i,j)$ are large
THEN  the central pixel $A(i,j)$ is an impulse noise pixel
```

We translate this rule by: if for a certain central pixel (i, j) more than half of the fuzzy gradient values (thus more than four) are part of the support of the fuzzy set *large* [8], then we can conclude that this pixel is an impulse noise pixel. The support of a certain fuzzy set F is the crisp set of all points in the Universe of Discourse U such that the membership function of F is non-zero.

If a pixel (i, j) is detected as an impulse noise pixel, then we store the corresponding greyscale value in a histogram (e.g. Fig. 5(b) and (d)). This histogram indicates the amount of noise detections (Y-axis) for each possible greyscale value (X-axis). We use this histogram (which we simply call the noise histogram) to investigate the presence of impulse noise. If the noise histogram contains some peaks, then we conclude that the image contains impulse noise pixels otherwise we conclude that the image is free of impulse noise. Fig. 5 shows two noise histograms for two images. The first image Fig. 5(a) is only corrupted with impulse noise which results in a noise histogram containing only peaks (see Fig. 5(b)). Fig. 5(c) is corrupted with a mixture of impulse noise and Gaussian noise. This mixture of noise will be displayed in the noise histogram which now contains two accumulations around two extreme values.


Histogram of detected noise pixels
Y
Amount of detected noise pixels
0
50
100
150
200
250
300
350
Grayscale values
X

(a) (b)



Histogram of detected noise pixels
Y
Amount of detected noise pixels
0
10
20
30
40
50
60
70
80
Grayscale values
X

(c) (d)

Fig. 5. (a) A Lena image corrupted with 5% impulse noise $\big((p_1, p_2, p_3, p_4, p_5) = (0, 50, 85, 175, 255)\big)$ (b) the corresponding histogram of the detected noisy pixels. (c) A Lena image corrupted with 3% impulse noise $\big((p_1, p_2) = (25, 250)\big)$ plus Gaussian noise with $\sigma = 5$. (d) The corresponding histogram of the detected impulse noisy pixels.

Finally this detection phase ends with the decision procedure pictured in Fig. 6. In this procedure we decide that an image A is impulse noise-free if the maximum value (the maximal y-value) of the noise histogram (for image A) contains less than $THR1$ % of the total detections. $THR1$ is a threshold value, which should be situated in the interval $[2.5; 10]$, because when the maximum value is lower than 2.5% we can't really speak of peaks. On the other hand, in images corrupted with mixed types of noise (incl. impulse noise) a threshold value above 10% could lead to the wrong conclusion that there is no impulse noise present.

```
Input:  $HIST$: the noise histogram matrix
        $HIST(i)$: indicates the amount of selected noise pixels
                with greyscale value $i$
                $if\big((i < 0)$ or $(i > 255)\big)$ $HIST(i) = 0$
        $TOT$: the total quantity of detected noise pixels
        $HIST(m)$: the maximum histogram value assumed to be
                located at greyscale value $m$ ($m \in [0, 255]$)

(1) IF $(HIST(m)/TOT) \times 100 < THR1$
(2)     NO IMPULSE NOISE DETECTED
(3) ELSE
(4)   IF $HIST(i)/(HIST(i-1) + HIST(i) + HIST(i+1)) \geq 0.95$
(5)       ImpulseNoise(TOT, HIST)
(6)   ELSE
(7)       ImpulseNoise&Other(TOT, HIST)
(8)   END IF
(9) END IF
```

Fig. 6. Pseudo-code of the decision procedure.

In cases where impulse noise is detected we distinguish two sub-cases. In the first sub-case, where images are only corrupted with impulse noise, we perform the ImpulseNoise function (Fig. 7). If we call m the greyscale value (x-axis value) where the maximal y value of the noise histogram is reached. Now we execute the ImpulseNoise function if the two closest greyscale values $m - 1$ and $m + 1$ both have very low y values with respect to the y-value of greyscale value m of the noise histogram (Fig. 6). In the other case we take into account that the image is corrupted with a mixture of impulse noise and other noise types (as for example Gaussian noise). In this second sub-case we perform the ImpulseNoise&Other function shown in Fig. 8.

The function ImpulseNoise (Fig. 7) determines the integer values p_k ($k \in \{1, .., n\}$) (used in Eq. 1). We determine maximal five such integer values to make the filtering step more robust against overfiltering. If an image is corrupted with impulse noise, with more than five such p_k values than we restart our complete method when the filtering method has finished. As indicated in Fig. 7 we use a threshold value $THR2$, which is related to $THR1$ ($THR2 \leq THR1/100$, and $THR2 \in$

```
Function ImpulseNoise(real TOT, matrix HIST)
  (1) $cum = 0$
  (2) FOR $k = 1$ to 5
  (3)   $kmax =$ select the $k^{th}$ greatest $y-$value of the noise histogram
  (4)   $kpos =$ the corresponding greyscale value where the maximum
              is reached
  (5)   IF $((kmax/TOT) + cum) \geq THR2$
  (6)      $p_k = kpos$
  (7)      set the 4 parameters $(a_k, b_k, c_k, d_k)$ to $(p_k, p_k, p_k, p_k)$
  (8)      $cum = cum + (kmax/TOT)$
  (9)   ELSE
  (10)      break
  (11)  END IF
  (12) END FOR
END Function
```

Fig. 7. Pseudo-code of the Impulsenoise function.

$[0.025, 0.1]$). This threshold value is used to select the quantity of noise integers p_k, that agrees with the amount of peaks in the histogram. The function ImpulseNoise&Other (Fig. 8)

```
Function ImpulseNoise&Other(real TOT, matrix HIST)
  (1) $cum = 0$
  (2) FOR $k = 1$ to 2
  (3)   $kmax =$ select the $k^{th}$ greatest $y-$value of the noise histogram
  (4)   $kpos =$ the corresponding greyscale value where the maximum
              is reached
  (5)   IF $((kmax/TOT) + cum) \geq THR2$
  (6)      $p_k = kpos$
  (7)      selecting 4 parameters $(a_k, b_k, c_k, d_k)$
  (8)      $cum = cum + (kmax/TOT)$
  (9)   ELSE
  (10)      break
  (11)  END IF
  (12) END FOR
END Function
```

Fig. 8. Pseudo-code of the Impulsenoise&Other function.

determines also the integer values p_k ($k \in \{1, .., n\}$), but here we consider a maximal amount of two such integers (per execution) because the presence of impulse noise mixed with another kind of noise should be solved in several complete iterations. Besides the determination of the integer values p_k we also calculate some parameters (a_k, b_k, c_k, d_k), which we will explain in the next section.

III. Filtering phase

If the detection phase has detected impulse noise then we perform the filtering step explained in this section. Otherwise we leave the image unchanged. This section is structured as follows: First we explain the calculation of the parameters (a_k, b_k, c_k, d_k), which are used to construct the fuzzy set *more or less impulse noise*. Afterwards we construct our first filtering iteration based on the membership function $\mu(.)$ that represents this fuzzy set. The explanation of the other iterations and some stop criteria finally terminate this section.

A. Configuration of the parameters

As shown in line 7 of the functions ImpulseNoise (Fig. 7) and ImpulseNoise&Other (Fig. 8) we calculate for each integer p_k four parameters (a_k, b_k, c_k, d_k). These parameters are used to construct the fuzzy set *more or less impulse noise*, which is represented by a proper membership function $\mu(.)$. An example of such a membership function is illustrated in Fig. 9. The obtained membership function is a simplification

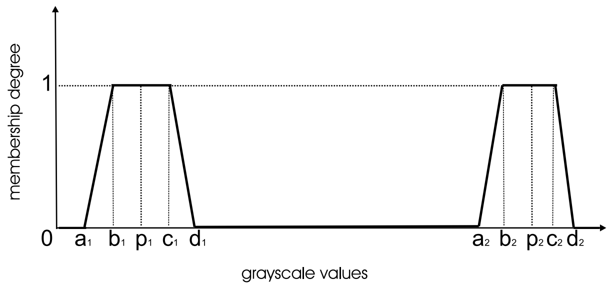


Fig. 9. The membership function $\mu(.)$ representing the fuzzy set "more or less impulse noise".

of the obtained noise histogram (e.g. Fig. 5(d)). Therefore we calculate or select the parameters (a_k, b_k, c_k, d_k) for the function ImpulseNoise&Other by the following rules:

$$\begin{aligned} a_k &= p_k - THRa & b_k &= p_k - THRb \\ c_k &= p_k + THRc & d_k &= p_k + THRd \\ THRb &= \frac{2}{3}THRa & THRc &= \frac{2}{3}THRd \end{aligned} \tag{4}$$

with $THRa = THRd = min(25, \lfloor\sigma\rfloor)$, where $\lfloor\sigma\rfloor$ is the largest integer value smaller than the variance σ. These rules are used to approximate the noisy histogram (as in Fig. 5(d)). Because σ is unknown, we try to make an accurate estimation of this value first. Our noise estimation is based on the edge image produced by the Sobel operator [11]. For more information we refer to the V. Zlokolica work [12]. Other approaches are possible as well (e.g. a statistical approach). When $\lfloor\sigma\rfloor > 25$, then we restrict $THRa$ and $THRd$ to be 25 to prevent overfiltering. Otherwise this large σ-value could lead to an extreme wide membership function which would cause some kind of blurring of the image.

B. First iteration

The filtering step of the first iteration is given in the pseudo-code presented in Fig. 10. This method is based on the

```
Input:          A: the noisy image with impulse noise.
       $\mu(A(i,j))$: the membership degree for the fuzzy set *more or
                 less impulse noise*.
                F: the output image.

(1)  FOR each *non-border* pixel $(i, j) \in A$
(2)    IF $A(i, j) \in$ supp("more or less impulse noise")
(3)         $F(i,j) = \frac{\sum_{h=-1}^{1}\sum_{l=-1}^{1}[1-\mu(A(i+h,j+l))]A(i+h,j+l)}{\sum_{h=-1}^{1}\sum_{l=-1}^{1}1-\mu(A(i+h,j+l))}$
(4)    ELSE
(5)       $F(i, j) = A(i, j)$
(6)    END IF
(7)  END FOR
```

Fig. 10. Pseudo-code of the first filtering iteration.

membership function "more or less impulse noise" (μ). The corresponding membership degree of a certain intensity value

$A(i,j)$ is denoted as $\mu(A(i,j))$. A degree one (zero) indicates that the intensity value is noisy for sure (not noisy for sure). When the degree is between one and zero then there is some kind of uncertainty. We only filter pixels which are part of the support of the fuzzy set *more or less impulse noise* (pixels which have a non-zero membership degree in this fuzzy set). Otherwise we leave the pixel unchanged. As shown in line 3 of Fig. 10 we use, in the first iteration, a 3×3 window around the filtered pixel. In addition, we used the standard negation $1 - \mu(.)$ to express the membership degrees in the fuzzy set *noise free* [9], [10] . If the output image (F) is the same as the input image (A) then the filter method is called recursively otherwise we call it non-recursively.

C. Next iterations

After the first iteration it is possible as a side effect (especially with high initial impulse noise) that the impulse noise is clustered around one or more pixels. To reduce these noisy pixels we will provide some more (recursive) iterations that are quite similar to the first one.

In each iteration we use the modified image of the previous performed iteration and a different window (indicated by eh and el in Fig. 11) around a given pixel. Fig. 12 shows the neighbourhood windows used in the first, second, third and fourth iteration. The changing window is used to avoid future clustering and also speeds up the execution time. We

```
Input:            F: output image of the previous iteration.
      $\mu_e(F(i,j))$: the membership degree for the fuzzy set more or
                  less impulse noise for the $e^{th}$ iteration ($e \geq 2$).

(1) FOR each non-border pixel $(i,j) \in A$
(2)   IF $F(i,j) \in$ supp("more or less impulse noise")
(3)      $F(i,j) = \frac{\sum_{h=-1}^{1}\sum_{l=-1}^{1}[1-\mu_e(F(i+eh,j+el))]F(i+eh,j+el)}{\sum_{h=-1}^{1}\sum_{l=-1}^{1} 1-\mu_e(F(i+eh,j+el))}$
(4)   END IF
(5) END FOR
```

Fig. 11. Pseudo-code of the e^{th} (recursive) filtering iteration.

always use a window of 9 pixels (inclusive the centre). The window change also implies the change of the term **"non-border pixel"**. In the n^{th} iteration non-border pixels are defined as having n left-, n right-, n upper- and n lower-neighbourhood pixels. The e^{th} ($e \geq 2$) iteration is illustrated in Fig. 11. In addition to the different window we also modify the membership function "more or less impulse noise" (denoted as $\mu_e(.)$ for the e^{th} iteration) by changing the parameters as shown in Eq. 5.

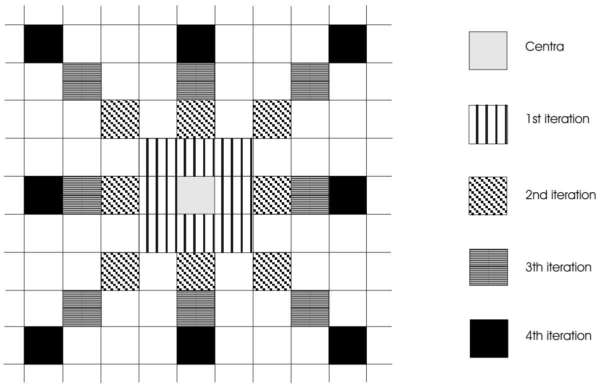


Fig. 12. The used neighbourhood window for the first, second, third and fourth iteration.

$$a_k^e = \frac{1}{2}(a_k^{e-1} + p_k) \qquad b_k^e = \frac{1}{2}(b_k^{e-1} + p_k)$$
$$c_k^e = \frac{1}{2}(c_k^{e-1} + p_k) \qquad d_k^e = \frac{1}{2}(d_k^{e-1} + p_k) \qquad (5)$$

Since this change will reduce the slope of the membership function and therefore also the amount of investigated pixels for an image A, it will speed up the execution time. We reduce the slope of the function, because the amount of noisy pixels was already reduced in the previous iteration.

D. Stop criteria

There are several techniques to improve the efficiency of the filtering phase. We will focus on the stop criteria and the amount of pixels that must be scanned during an iteration. During the first iteration we check every pixel. If the pixel value does not belong to the support of the fuzzy set *more or less impulse noise* then we don't change this pixel value, not only in this iteration neither in the other ones. By remembering only the positions of pixels, whose pixel value is an element of the support of the fuzzy set *more or less impulse noise*, we can drastically reduce the scanning amount in the next iterations.

Of course we also need some stop criteria. If we define $\#_e$ as the amount of pixel values which belong to the support of the fuzzy set *more or less impulse noise* in the e^{th} iteration, then we can apply the following stop criteria:

1) If there are no pixel values in the support of the fuzzy set ($\#_e = 0$) in any iteration e ($e \geq 2$).
2) $\#_e$ is equal to $\#_{e-1}$. This indicates that the resulting pixels (pixels which still are element of the support of the fuzzy set *more or less impulse noise*) aren't noisy even with the non-zero membership degrees.
3) Since $\#_1 \geq \#_2 \geq ... \geq \#_{e-1} \geq \#_e$ hold we can define $\Delta_e = \#_{e-1} - \#_e$. When Δ_e is (very) small then we can decide to stop too.

IV. Simulation Results

In this section we compare the efficiency and performance of FIDRM for greyscale images with other well-known filters for impulse noise reduction. The first group of filters consists of the FIRE filters (fuzzy inference rule by else-action filters). The idea behind these filters is that they try to calculate positive and negative correction terms in order to express the degree of noise for a certain pixel. We distinguish three FIRE filters: the normal FIRE [13] (fuzzy inference rule by else-action filters), the DS-FIRE [14] (dual step FIRE) and the PWLFIRE [15] (piecewise linear FIRE). A second group of filters contains filters which are extensions of classical median (denoted by MED) and weighted filters. We use the FMF [16], [17] (fuzzy median filter), AWFM [18], [19] (adaptive weighted fuzzy mean) and the ATMAV [20] (asymmetrical triangular fuzzy filter with moving average centre). The FCF (fuzzy control based) filters constitute a third group of filters. These filters correct a certain central pixel value according to some features of some luminance (pixel values) differences between the central pixel value and some neighbour pixel values. In the literature we know: the IFCF [21] (iterative fuzzy control based filter), the MIFCF [21] (modified IFCF), the EIFCF [21] (extended IFCF) and the SFCF [22] (smoothing fuzzy control based filter). Furthermore there exist many other types of filters such as e.g. the histogram adaptive fuzzy filter which of course uses the histogram of an image [23] (HAF) or the fuzzy similarity filter [24] (FSB) where the local similarity between some patterns is used. Besides all these fuzzy filters some other popular filters exist. We use the CWM [3] (center weighted median), the TSM [4] (tri-state median filter) and the LUM [5] (lower-upper-middle filter) for comparison.

As a measure of objective dissimilarity between a filtered image and the original one, we use the mean square error (MSE) and the peak signal to noise ratio (PSNR (in decibels dB)):

$$MSE(Img, Org) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\Big[Org(i,j) - Img(i,j)\Big]^2}{NM} \quad (6)$$

$$PSNR(MSE(Img, Org)) = 10\log_{10}\frac{S^2}{MSE(Img, Org)} \quad (7)$$

where Org is the original image, Img the filtered image of size NM and S the maximum possible pixel value (with 8-bit integer values the maximum will be 255). We used the well-known Lena and Mandrill (Baboon) images of size respectively 256×256 and 512×512, to compare FIDRM with the other filters. From Table II and Table III we see that FIDRM is one of the fastest algorithms, which also generates the smallest PSNR values. The IFCF filters are iterative filters, so the execution time must be multiplied with the amount of iterations. For the IFCF filters these amounts are indicated next to the time. The FIDRM filter is iterative too, but here the execution time shown for that filter is the total execution time. After the first iteration the FIDRM filter reinvestigates only those pixels which belong to the support of the fuzzy set *more or less impulse noise*. This causes lower execution time for each further iteration.

FIDRM performs very well even at high noise levels. As an illustration Fig. 13 shows the results for a Mandrill image corrupted with 30% salt and pepper noise. One can observe that the proposed filter preserves edge sharpness and reduces many artefacts in contrast to the other well performing algorithms as AWFM, HAF and ATMAV.

In addition to the previous experiments, where only salt and pepper noise was considered, we also compared our method for other noise situations in Table IV. The first experiment (case 1) shows the numerical results for images corrupted with impulse noise with $(p_1, p_2, p_3, p_4) = (15, 25, 225, 250)$. In the second experiment (case 2) we illustrate the filtering performance for images corrupted with a mixture of Gaussian noise and salt and pepper noise. In this case the proposed method filters out the impulse noise but leaves the Gaussian noise, so that a Gaussian denoising filter can be applied afterwards. Fig. 14 confirms this by showing the visual results for some of these filters. Most of the compared filters can not deal well with the mixture of Gaussian noise and impulse noise (the images tend to blur and they lose detailed information). In a last experiment (case 3) we want to apply FIDRM for an image corrupted with random impulse noise. We applied the FIDRM locally instead of globally in order to handle this random impulse noise (i.e. the proposed fuzzy detection and reduction method is applied on different parts of the image, separately). As one can see in Table IV, FIDRM does not outperform the Median based filters for random impulse noise. Nevertheless the idea behind this filter (especially the detection method) can be used to develop a method that can suppress random impulse noise very well. Our future research will be concentrated on this issue and on the construction of fuzzy filtering methods for colour images.

TABLE II

PSNR & EXECUTION TIME RESULTS FOR THE (512×512-) MANDRILL IMAGE FOR DIFFERENT IMPULSE NOISE LEVELS AND DIFFERENT FILTERS (MATLAB ENVIRONMENT).

	PSNR (dB)			Execution time (seconds)					
	5%	25%	50%	5%		25%		50%	
Original	**18.5**	**11.5**	**8.5**						
CWM (3 × 3)	26.8	21.5	14.3	44.0		43.7		44.1	
CWM (7 × 7)	24.4	21.2	19.4	50.3		50.4		50.5	
TSM (3 × 3)	29.0	21.7	14.3	86.9		88.9		93.8	
TSM (7 × 7)	23.5	22.4	20.0	104.0		104.3		105.7	
LUM	24.9	20.4	14.3	8.9		9.0		8.9	
MED (3 × 3)	23.2	21.0	14.4	8.8		8.8		8.8	
MED (5 × 5)	20.9	20.6	18.9	9.3		9.3		9.3	
MED (7 × 7)	20.2	20.0	19.5	11.7		11.5		11.6	
ATMAV	21.8	21.2	20.6	76.5		88.6		72.8	
FSB	22.1	20.1	14.0	349.0		386.9		313.1	
HAF	22.4	22.1	20.7	277.7		276.9		243.0	
AWFM	22.6	22.3	21.3	356.4		357.0		333.1	
SFCF	22.8	19.7	14.1	436.6		499.5		429.3	
EIFCF	23.5	20.0	15.4	321.3	| 2 |	326.4	| 5 |	297.8	| 5 |
MIFCF	23.6	20.0	15.3	259.2	| 2 |	279.2	| 5 |	240.3	| 5 |
IFCF	23.6	20.0	15.5	250.8	| 2 |	261.3	| 5 |	218.0	| 5 |
FIRE	24.3	18.8	12.8	225.0		227.0		208.8	
FMF	27.7	20.9	14.4	59.5		64.4		55.4	
DSFIRE	30.3	24.6	16.6	660.1		676.4		611.0	
PWLFIRE	32.6	19.6	12.2	62.1		66.7		57.8	
FIDRM	34.4	27.0	23.2	4.9		15.7		39.1	

TABLE III

PSNR & EXECUTION TIME RESULTS FOR THE (256×256-) LENA IMAGE FOR DIFFERENT IMPULSE NOISE LEVELS AND DIFFERENT FILTERS (MATLAB ENVIRONMENT).

	PSNR (dB)			Execution time (seconds)					
	5%	25%	50%	5%		25%		50%	
Original	**18.6**	**11.6**	**8.6**						
CWM (3 × 3)	32.4	25.0	15.2	11.1		10.7		10.7	
CWM (7 × 7)	29.5	26.8	24.3	12.5		12.6		12.5	
TSM (3 × 3)	35.4	25.6	24.8	21.6		21.9		23.2	
TSM (7 × 7)	31.9	28.3	15.5	24.8		25.8		24.5	
LUM	31.9	25.1	18.4	2.2		2.2		2.3	
MED (3 × 3)	29.7	24.9	15.1	2.1		2.2		2.1	
MED (5 × 5)	27.3	26.3	22.4	5.8		5.6		5.7	
MED (7 × 7)	25.9	25.3	23.9	9.3		9.7		9.5	
ATMAV	24.1	23.5	20.2	16.6		18.8		18.7	
FSB	29.4	24.7	14.9	80.9		79.2		75.5	
HAF	29.1	28.4	25.6	61.8		65.4		62.2	
AWFM	30.3	29.5	26.9	84.6		88.2		84.0	
SFCF	29.1	22.3	14.7	87.9		111.4		103.8	
EIFCF	28.9	24.0	17.5	68.5	| 2 |	80.9	| 5 |	71.9	| 5 |
MIFCF	29.4	24.9	17.6	56.3	| 3 |	67.8	| 5 |	61.9	| 5 |
IFCF	29.2	24.1	17.6	53.6	| 2 |	63.7	| 5 |	59.9	| 5 |
FIRE	31.8	21.1	13.5	53.8		54.7		50.5	
FMF	34.3	25.5	15.7	12.7		12.9		14.1	
DSFIRE	35.7	28.5	17.9	153.0		156.5		146.6	
PWLFIRE	36.5	20.4	12.6	14.0		15.2		10.6	
FIDRM	40.7	33.4	29.0	1.0		4.2		8.3	

TABLE IV

PSNR RESULTS FOR THE $(256 \times 256\text{-})$ LENA IMAGE FOR DIFFERENT IMPULSE NOISE CASES WITH CASE 1: IMPULSE NOISE WITH $(p_1, p_2, p_3, p_4) = (15, 25, 225, 250)$; CASE 2: GAUSSIAN NOISE WITH $\sigma = 5$ MIXED WITH SALT & PEPPER NOISE; CASE 3: RANDOM IMPULSE NOISE.

	PSNR Case 1			PSNR Case 2			PSNR Case 3		
	5%	20%	30%	5%	20%	30%	5%	20%	30%
Original	**19.7**	**13.7**	**11.9**	**18.6**	**12.9**	**11.4**	**22.8**	**16.4**	**14.3**
CWM (3×3)	33.0	27.3	23.5	31.0	27.1	23.4	32.9	28.5	26.0
CWM (7×7)	29.4	26.7	26.1	28.7	26.9	25.6	29.4	26.7	25.8
TSM (3×3)	34.7	27.7	23.8	31.6	28.2	24.1	34.9	28.5	26.1
TSM (7×7)	32.7	26.6	23.9	30.6	27.3	23.8	31.8	27.5	25.9
LUM	32.7	26.9	23.8	30.9	26.9	23.8	33.1	27.9	25.9
MED (3×3)	29.7	26.9	23.8	29.1	26.9	23.9	29.5	27.5	25.7
MED (5×5)	27.3	26.5	26.1	27.2	26.8	25.9	27.2	26.3	25.6
MED (7×7)	25.8	25.3	25.2	25.7	25.4	25.0	25.8	25.2	24.7
ATMAV	20.3	21.7	21.3	27.8	26.9	26.3	22.2	21.1	20.3
FSB	29.4	26.9	23.1	29.2	26.9	23.7	29.4	27.3	25.9
HAF	28.6	24.5	22.0	28.9	28.4	27.5	28.5	24.7	22.6
AWFM	29.6	27.4	26.3	30.2	29.7	28.3	29.4	25.6	23.5
SFCF	29.3	25.0	21.2	28.8	24.9	21.7	29.3	26.9	24.8
EIFCF	29.5	26.9	24.3	29.0	26.6	24.0	29.6	27.6	26.1
MIFCF	29.7	26.3	23.1	29.4	25.7	22.8	30.1	27.5	25.7
IFCF	29.6	26.9	24.3	29.2	26.6	24.2	29.7	27.7	26.2
FIRE	31.8	24.3	20.4	30.0	24.0	20.4	31.4	25.9	23.3
FMF	33.9	27.8	23.4	32.0	27.7	23.9	33.1	28.2	25.7
DSFIRE	23.6	22.6	21.5	23.5	23.1	21.9	22.8	20.4	19.7
PWLFIRE	35.2	24.0	19.5	32.2	23.6	19.5	28.4	21.7	19.7
FIDRM	40.7	33.9	31.8	35.3	32.8	31.0	31.1	30.3	26.7

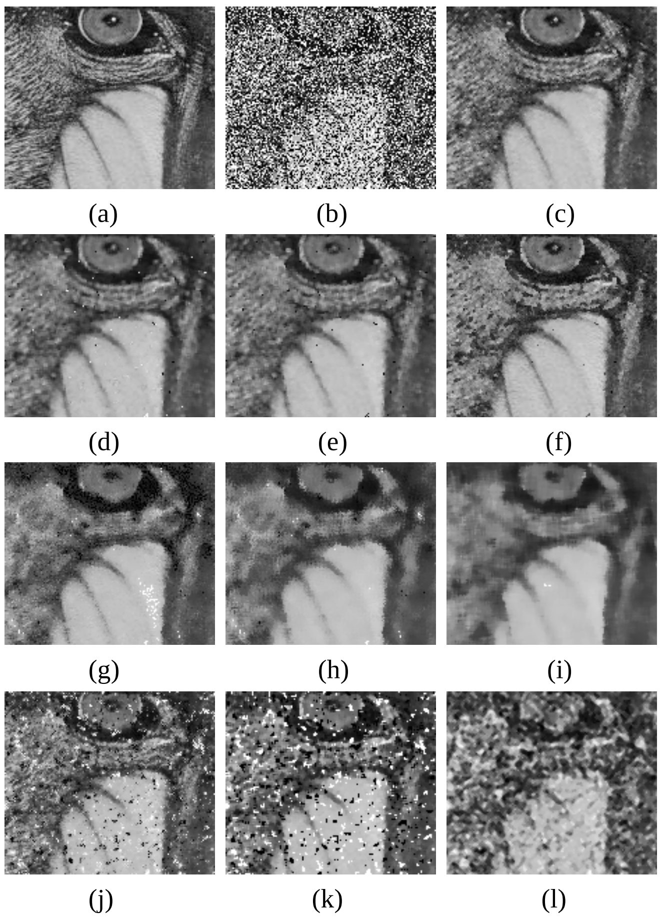


Fig. 13. The restoration of a noisy $(512 \times 512\text{-})$ Mandrill image (a) The increased noise-free Mandrill, (b) Image contaminated with 50% impulse noise (salt & pepper noise), (c) FIDRM, (d) HAF, (e) ATMAV, (f) AWFM, (g) TSM (7×7), (h) CWM (7×7), (i) basic MEDIAN (7×7), (j) DSFIRE, (k) LUM, (l) IFCF.

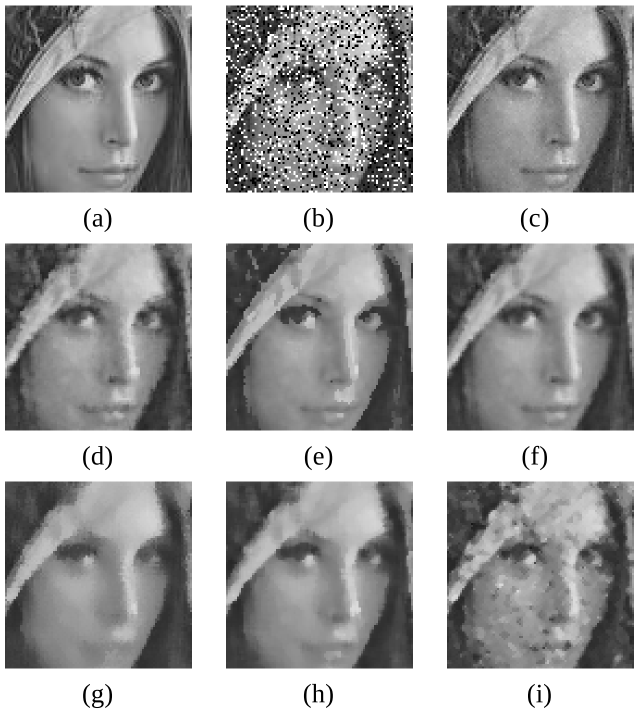


Fig. 14. The restoration of a noisy $(256 \times 256\text{-})$ Lena image (a) The increased noise-free Lena, (b) Image contaminated with $\sigma = 5$ gaussian noise and 30% impulse noise (salt & pepper noise), (c) FIDRM, (d) ATMAV, (e) AWFM, (f) HAF, (g) CWM (7×7), (h) basic MED (5×5), (i) EIFCF.

V. CONCLUSION

A new two step filter (FIDRM), which uses a fuzzy detection and an iterative filtering algorithm, has been presented. This filter is especially developed for reducing all kinds of impulse noise (not only salt and pepper noise). Its main feature is that it leaves the pixels which are noise-free unchanged. Experimental results show the feasibility of the new filter. A numerical measure, such as the PSNR, and visual observations (Fig. 13-14) show convincing results for greyscale images. Finally, this new method is easy to implement and has a very low execution time.

ACKNOWLEDGMENT

This work was financially supported by the GOA-project 12.0515.03 of Ghent University.

REFERENCES

[1] E. E. Kerre and M. Nachtegael, “Fuzzy Techniques in Image Processing,” 1st ed., vol. 52, Heidelberg, Physica Verlag, 2000, 429 pages.

[2] M. Nachtegael, D. Van der Weken, D. Van de Ville and E. E. Kerre, “Fuzzy Filters for Image Processing,” 1st ed., vol. 122, Heidelberg, Physica Verlag, 2003, 386 pages.

[3] S. J. Ko and Y. H. Lee, “Center weighted median filters and their applications to image enhancement,” IEEE Transactions on Circuits and Systems, vol. 38, pp. 984-993, Sep. 1991.

[4] T. Chen, K. K. Ma and L. H. Chen, “Tri-state median filter for image denoising,” IEEE Transactions on Image Processing, vol. 8, pp. 1834-1838, Dec. 1999.

[5] R. C. Hardie and C. G. Boncelet, “LUM filters: a class of rank-order-based filters for smoothing and sharpening,” IEEE Transactions on Signal Processing, vol. 41, pp. 1834-1838, Mar. 1993.

[6] D. Van De Ville, M. Nachtegael, D. Van der Weken, E. E. Kerre and W. Philips, “Noise reduction by fuzzy image filtering,” IEEE Transactions on Fuzzy Systems, vol. 11, pp. 429-436, Aug. 2001.

[7] M. Nachtegael, D. Van der Weken and E. E. Kerre, “Fuzzy Techniques in Image Processing: Three Case Studies,” International Journal of Computing Anticipatory Systems, vol. 12, pp. 89-104, Aug. 2002.

[8] E. E. Kerre, *Fuzzy sets and approximate Reasoning*. Xian Jiaotong University Press, Softcover, 1998.

[9] C. Cornelis, G. Deschrijver and E. E. Kerre, “Classification of intuitionistic fuzzy impicators: an algebraic approch,” in *Proc. of the 6th Joint Conference on Information Sciences*, 2002, pp. 105-108.

[10] C. Cornelis, “Tweezijdigheid in de Representatie en Verwerking van Imprecieze Informatie,” Doctoraatsproefschrift Universiteit Gent, 2004.

[11] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison Wesley, Prentice Hall, 1992.

[12] V. Zlokolica and W. Philips, “Motion and detail adaptive denoising of video,” in *IS&T/SPIE Symposium on Electronic Imaging*, 2004, pp. 1417-1421.

[13] F. Russo and G. Ramponi, “A Fuzzy Filter for Images Corrupted by Impulse Noise,” IEEE Signal Procceedings Letters, vol. 3, pp. 168-170.

[14] F. Russo and G. Ramponi, “Removal of impulse noise using a FIRE filter,” in *Proc. Third IEEE Intern. Conf. on Image Processing*, 1996, pp. 975-978.

[15] F. Russo, “Fire Operators for Image Processing” Fuzzy Sets and Systems, vol. 103, pp. 265-275. April 1999.

[16] K. Arakawa, “Median filter based on fuzzy rules and its application to image restoration,” Fuzzy Sets and Systems, vol. 77, pp. 3-13, Jan. 1996.

[17] K. Arakawa, “Fuzzy rule-based image processing with optimization,” in *Fuzzy Techniques in Image Processing*, 1st ed., vol. 52, E. E. Kerre and M. Nachtegael, Eds. Heidelberg: Physica Verlag, 2000, pp. 222-247.

[18] C. S. Lee, Y. H. Kuo and P. T. Yu, “Weighted fuzzy mean filters for image processing,” Fuzzy Sets and Systems, vol. 89, pp. 157-180, July 1997.

[19] C. S. Lee and Y. H. Kuo, “Adaptive fuzzy filter and its application to image enhancement,” in *Fuzzy Techniques in Image Processing*, 1st ed., vol. 52, E. E. Kerre and M. Nachtegael, Eds. Heidelberg: Physica Verlag, 2000, pp. 172-193.

[20] H. K. Kwan, “Fuzzy filters for noise reduction in images,” in *Fuzzy Filters for Image Processing*, 1st ed., vol. 122, M. Nachtegael, D. Van der Weken, D. Van De Ville and E. E. Kerre, Eds. Heidelberg: Physica Verlag, 2003, pp. 25-53.

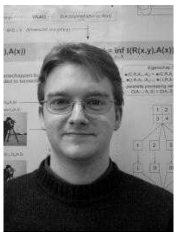
[21] F. Farbiz and M. B. Menhaj, “A fuzzy logic control based approch for image filtering,” in *Fuzzy Techniques in Image Processing*, 1st ed., vol. 52, E.E. Kerre and M. Nachtegael, Eds. Heidelberg: Physica Verlag, 2000, pp. 194-221.

[22] F. Farbiz, M. B. Menhaj and S. A. Motamedi, “Edge Preserving Image Filtering based on Fuzzy Logic,” in *Proc. of the 6th EUFIT conference*, 1998, pp. 1417-1421.

[23] J. H. Wang and H. C. Chiu, “An adaptive fuzzy filter for restoring highly corrupted images by histogram estimation,” Proceedings of the National Science Council -Part A, vol. 23, pp. 630-643, 1999.

[24] I. Kalaykov and G. Tolt, “Real-time image noise cancellation based on fuzzy similarity,” in *Fuzzy Filters for Image Processing*, 1st ed., vol. 122, M. Nachtegael, D. Van der Weken, D. Van De Ville and E. E. Kerre, Eds. Heidelberg: Physica Verlag, 2003, pp. 54-71.

Stefan Schulte was born in Dortmund, Germany, in 1979. After secondary school he studied Computer Science at the University of Ghent. In 2003 he obtained a M.Sc. degree. Since September 2003, he joined the Department of Applied Mathematics and Computer Science, Ghent University, where he is a member of the Fuzziness and Uncertainty Modelling Research Unit working toward the Ph.D. degree with a thesis on fuzzy techniques in image processing and fuzzy filters for image noise removal under the promotorship of Prof. E. E. Kerre.



Mike Nachtegael was born on February 16, 1976. After secondary school he went to Ghent University, where he obtained a M.Sc. in Mathematics in 1998. In the same year he joined the Department of Applied Mathematics and Computer Science, where he is a member of the Fuzziness and Uncertainty Modelling Research Unit. In May 2002 he obtained a Ph.D. in Mathematics, on the topic of Fuzzy Techniques in Image Processing. Currently, he has the position of doctor-assistant in his Department.

Valérie De Witte was born in Ghent, Belgium, in 1981. She received the M.Sc. degree in Mathematics from Ghent University, Ghent, Belgium, in 2003. In September 2003, she joined the Department of Applied Mathematics and Computer Science, Ghent University, where she is a member of the Fuzziness and Uncertainty Modelling Research Unit working toward the Ph.D. degree with a thesis Fuzzy mathematical morphology under the promotorship of Prof. E. E. Kerre.

Dietrich Van der Weken was born in Beveren, Belgium, in 1978. He received the M.Sc. degree in Mathematics from Ghent University, Ghent, Belgium, in 2000. In September 2000, he joined the Department of Applied Mathematics and Computer Science, Ghent University, where he is a member of the Fuzziness and Uncertainty Modelling Research Unit. In May 2004 he obtained a Ph.D. in Mathematics, on the use and construction of similarity measures in image processing. Currently he is working as a post-doctoral researcher in his Department.

Etienne Kerre was born in Zele, Belgium on May 8, 1945. He obtained his M.Sc. degree in Mathematics in 1967 and his Ph.D. in Mathematics in 1970 from Ghent University. Since 1984, he has been a lector, and since 1991, a full professor at Ghent University. He is a referee for more than 30 international scientific journals, and also a member of the editorial board of international journals and conferences on fuzzy set theory. He was an honorary chairman at various international conferences. In 1976, he founded the Fuzziness and Uncertainty Modelling Research Unit (FUM) and since then his research has been focused on the modelling of fuzziness and uncertainty, and has resulted in a great number of contributions in fuzzy set theory and its various generalizations, and in evidence theory. Especially the theories of fuzzy relational calculus and of fuzzy mathematical structures owe a very great deal of him. Over the years he has also been a promotor of 19 Ph.D's on fuzzy set theory. His current research interests include fuzzy and intuitionistic fuzzy relations, fuzzy topology, and fuzzy image processing. He has authored or co-authored 11 books, and more than 100 papers appeared in international refereed journals.