Project Report

On

Adaptive Type-2 Fuzzy Approach for Filtering Salt and Pepper Noise in Grayscale Images

Vikas Singh, Raghav Dev , Narendra K. Dhar , Pooja Agrawal and Nishchal K. Verma

To be submitted by

Nalin Kumar Giri (18104063)

Ujjwal Kumar(18104118)

Sathyanarayanan(18204276)

Gaurav(18204403)

Shubham Ghotkar(18204403)

Under the supervision of

Dr. Nishchal K. Verma
Professor
Dept. of Electrical Engineering
Indian Institute of Technology Kanpur, India

Contents

Chapter 1

Adaptive Type-2 Fuzzy Approach for Filtering Salt and Pepper Noise in Grayscale Images

Vikas Singh , Raghav Dev , Narendra K. Dhar , Pooja Agrawal , and Nishchal K. Verma

1. Introduction

This paper proposes a novel adaptive Type-2 fuzzy filter for removing salt and pepper noise from the images. The filter removes noise in two steps.

Step 1. The pixel are categorised as good or bad based on their primary membership function (MF) values in the respective filter window.

Step 2. The pixel categorised as bad are denoised.

Two approaches have been proposed for finding threshold between good or bad pixels by designing primary MFs.

(a). MFs with distinct Means and same Variance.
(b). MFs with distinct Means and distinct Variances.

- The primary MFs of the Type-2 fuzzy set is chosen as Gaussian membership functions.
- For denoising, a novel Type-1 fuzzy approach based on a weighted mean of good pixels is presented in the paper.
- The proposed filter is validated for several standard images with the noise level as low as 20% to as high as 99%.

Removing noise from images is an essential task since good quality images are required in various applications such as medical imaging, satellite imaging, recognition, etc. There are several types of noises that can degrade the quality of images. One such type is salt and pepper (SAP) noise. This noise can be represented as randomly occurring white (1) and black (0) pixels in the image. The main sources of this noise are electrical conditions, light intensity, imperfection in imaging sensors, transmission errors, etc.

2. Previous related Work and Problems assosiated

Median filter and adaptive median filter are popularly used for SAP noise removal. In these approaches noisy pixel intensity is replaced by the median of intensities of neighbourhood pixels. Although these filters are efficient to remove noise but fails to preserve details of the image due to blurring at the edges.

Ahmed *et al.* [9] have proposed an iterative adaptive fuzzy filter for removal of high-density SAP noise. The drawback of this approach seems to be assignment of weight to good pixels in window during denoising using inverse distance weighting function. Therefore, this method fails to preserve the image details. The other problem of this method is use of many heuristic parameters such as $K1$, and $K2$ that are not consistent to give best result for different noise levels

Liang and Mendel [13] proposed Type-2 adaptive filter using an unnormalized Type-2 Takagi Sugeno Kang fuzzy logic system for the application of equalization of a nonlinear time-varying channel. John *et al.* [14] have applied neurofuzzy clustering techniques for classifying images where an image is represented by Type-2 fuzzy set. Yldrm *et al.* [15] have proposed a Type-2 fuzzy filter for suppressing noise in the image while at the same time preserving thin lines, edges, texture, and other useful features within the image.

3.PRELIMINARIES

A. Type-2 Fuzzy Set

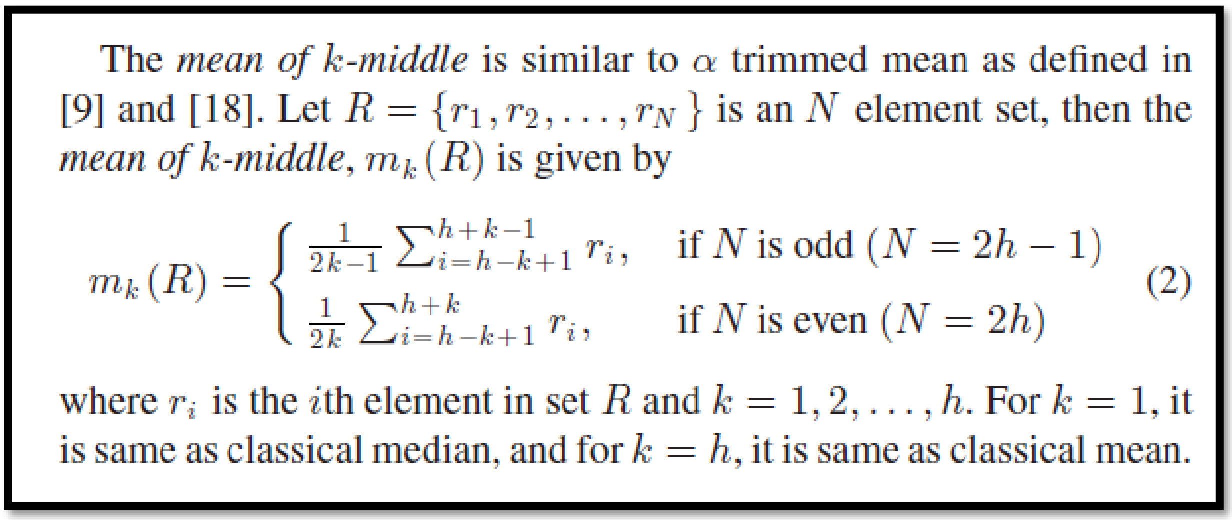
Type-2 fuzzy set is an extension of Type-1 fuzzy set, which was originally introduced by Zadeh [10]. The set theoretic operations and properties of membership grades are evaluated in the form of algebraic product and sum given by Mizumoto and Tanaka [11]. Karnik and Mendel in [12] have extended the concept of Type-2 set for performing union, intersection, and complement.

The Type-2 fuzzy system characterized by membership function and is defined as

$$\tilde{M}_{ij}^{H} = \{ \, (p_{ij}, \mu_{M_{ij}^{H}}), \;\; \mu_{\tilde{M}_{ij}^{H}}(p_{ij}, \mu_{M_{ij}^{H}}) \;\; \forall\, p_{ij} \in I \;\; \forall\, \mu_{M_{ij}^{H}} \in J_{p_{ij}} \subseteq [0\,1] \, \} \qquad (1)$$

where $0 \le \mu_{M_{ij}^{H}}, \mu_{\tilde{M}_{ij}^{H}}(p_{ij}, \mu_{M_{ij}^{H}}) \le 1$ and I is the universe of discourse.

B. Mean of k-Middle

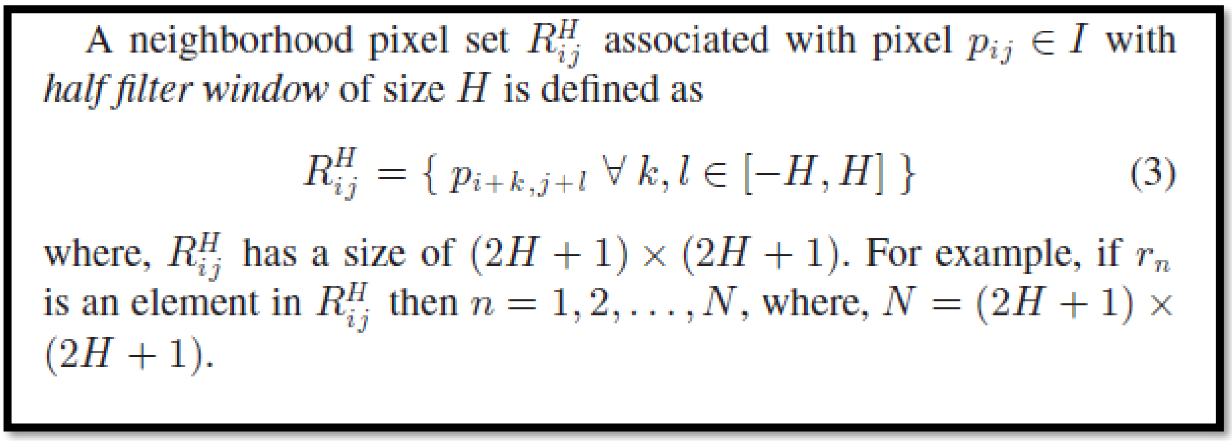


The *mean of k-middle* is similar to α trimmed mean as defined in [9] and [18]. Let $R = \{r_1, r_2, \ldots, r_N\}$ is an N element set, then the *mean of k-middle*, $m_k(R)$ is given by

$$m_k(R) = \begin{cases} \frac{1}{2k-1} \sum_{i=h-k+1}^{h+k-1} r_i, & \text{if } N \text{ is odd } (N = 2h-1) \\ \frac{1}{2k} \sum_{i=h-k+1}^{h+k} r_i, & \text{if } N \text{ is even } (N = 2h) \end{cases} \quad (2)$$

where r_i is the ith element in set R and $k = 1, 2, \ldots, h$. For $k = 1$, it is same as classical median, and for $k = h$, it is same as classical mean.

C. Neighborhood Pixel Set



A neighborhood pixel set R_{ij}^H associated with pixel $p_{ij} \in I$ with *half filter window* of size H is defined as

$$R_{ij}^H = \{\, p_{i+k,j+l} \ \forall\, k, l \in [-H, H] \,\} \quad (3)$$

where, R_{ij}^H has a size of $(2H+1) \times (2H+1)$. For example, if r_n is an element in R_{ij}^H then $n = 1, 2, \ldots, N$, where, $N = (2H+1) \times (2H+1)$.

4. PROPOSED ADAPTIVE TYPE-2 FUZZY FILTER

The proposed Type-2 adaptive fuzzy filter for SAP noise removal has been discussed in this section. It has two steps. In the first step, pixels are categorized as “good” or “bad.” Two approaches have been proposed for this categorization. Both have their own advantages. Either of them can be used. In the second step, an approach for denoising bad pixels is proposed. The schematic steps of methodology is shown in Fig. 1.

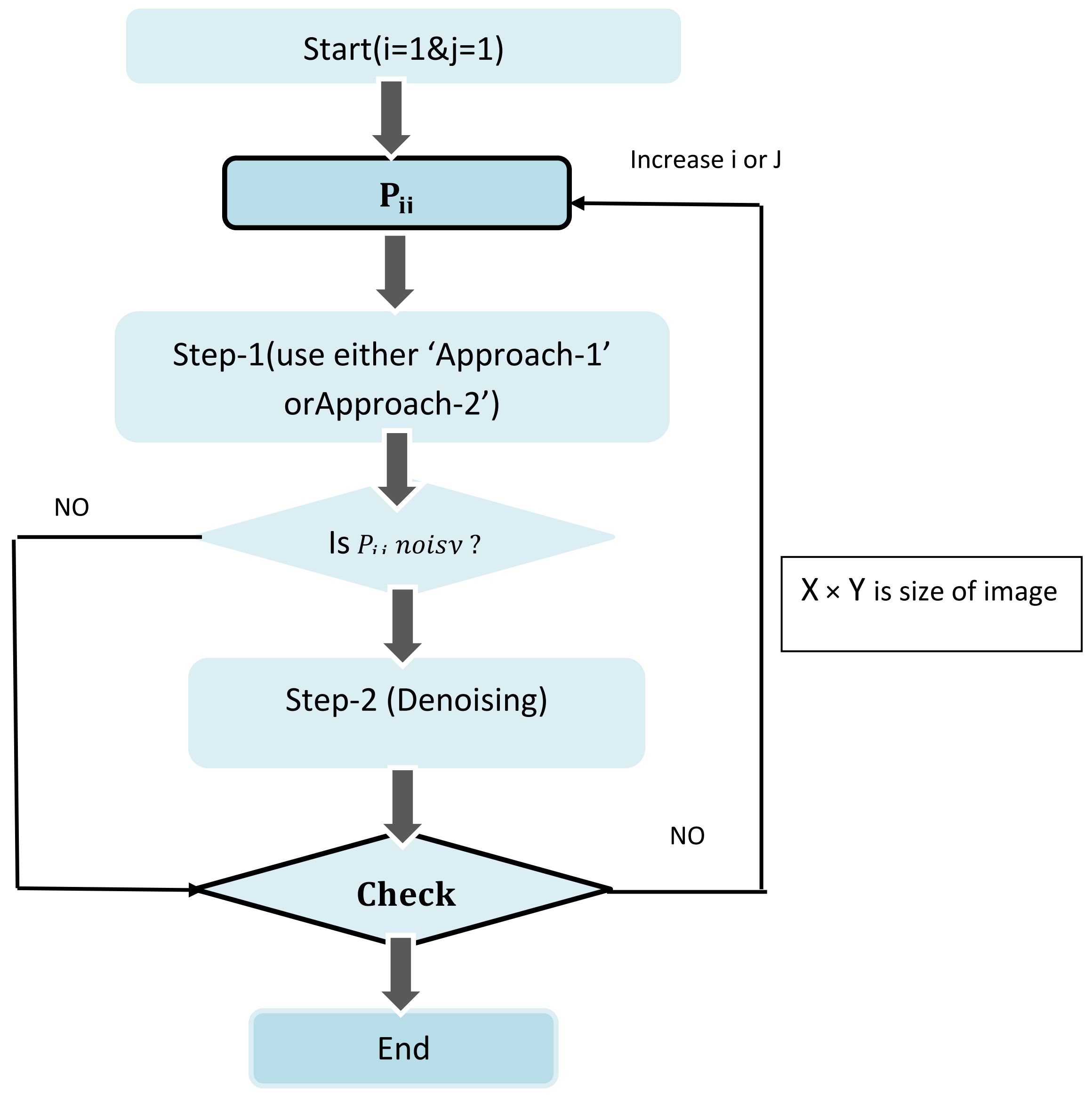


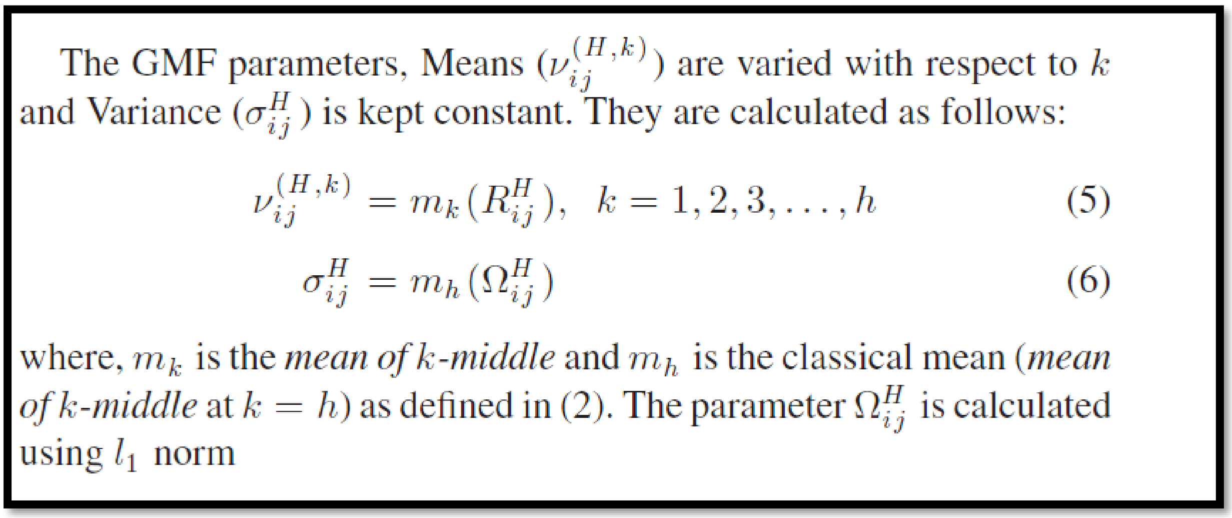
Fig- Schematic diagram of proposed filter

Both the approaches of first step comprise of three parts. In the first part, set *RH ij* is evaluated using (3) for pixel *pij* . The primary MF values of Type-2 fuzzy set are evaluated for each element of *RH ij* in the second part. The upper membership function (UMF) and lower membership function (LMF) values of Type-2 fuzzy set are used to decide the threshold. In the third part, elements of *RH ij* are categorized as “good” and “bad” pixels by comparing their primary MF values with the threshold. The pixel (within filter window) having primary MF value greater than the threshold is considered as “good” otherwise “bad.” If the center pixel *pij* is a good pixel, then its value is retained and filtered in next iteration. If the center pixel *pij* is bad, then it is denoised using good pixels of *RH ij* in the second step that is Type-1 fuzzy set .

A. MFs With Distinct Means and Same Variance

In this approach, UMF and LMF of Type-2 fuzzy set are designed to obtain a threshold that will ensure proper categorization of pixels in a window. The pixel having intensity value greater than the threshold is considered as good, whereas one with value lesser than the threshold is a bad or noisy pixel. The paper deals with SAP noise whose intensity values are either 0 or 1. Hence the pixels having intensity values *pij* /∈*{*0, 1*}* are retained. For each pixel *pij* ∈*{*0, 1*}* at the location (*i, j*) in an image *I*, a filter window of size (2*H* + 1) × (2*H* + 1) is considered for the evaluation of neighborhood set *RH ij* . The pixel *pij* is at the center of the window and *H* is its half-length. A Type-1 fuzzy set *MH ij* is defined where each element *rn* ∈*RH ij* is associated with a membership value using a Gaussian membership functions (GMF) *µM H i j* (*rn*) : *RH ij* → [0, 1]. It is known as primary MF. The membership grade of primary MF is again a fuzzy set in the unit interval [0,1]. It is known as secondary MF. The Type-2 fuzzy set ˜*M H ij* is characterized by MF *µ* ˜*M H i j* (*rn* , *µM H i j*) as defined in (1). Every element *rn* in the set *RH ij* belongs to Type-1 fuzzy set *MH ij* and it is associated with a GMF

$$\boxed{\mu_{M_{ij}^{(H,k)}}(r_n) = e^{-(r_n - \nu_{ij}^{(H,k)})^2 / 2(\sigma_{ij}^{H})^2}. \qquad (4)}$$

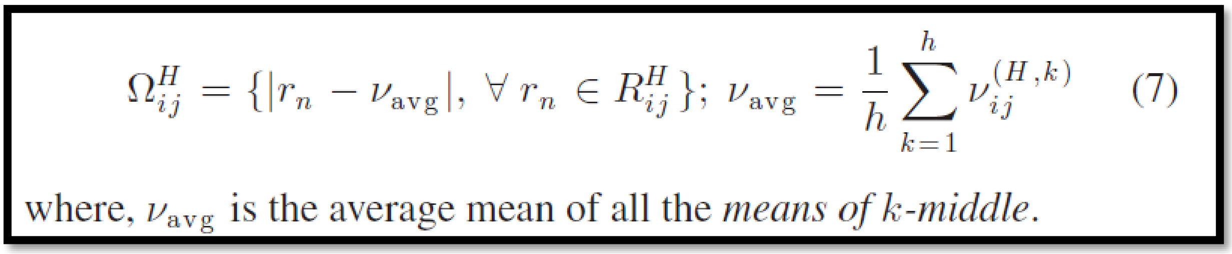


The GMF parameters, Means ($\nu_{ij}^{(H,k)}$) are varied with respect to k and Variance (σ_{ij}^{H}) is kept constant. They are calculated as follows:

$$\nu_{ij}^{(H,k)} = m_k(R_{ij}^{H}), \quad k = 1, 2, 3, \dots, h \tag{5}$$

$$\sigma_{ij}^{H} = m_h(\Omega_{ij}^{H}) \tag{6}$$

where, m_k is the *mean of* k*-middle* and m_h is the classical mean (*mean of* k*-middle* at $k = h$) as defined in (2). The parameter Ω_{ij}^{H} is calculated using l_1 norm



$$\Omega_{ij}^{H} = \{|r_n - \nu_{\text{avg}}|, \ \forall\, r_n \in R_{ij}^{H}\}; \ \nu_{\text{avg}} = \frac{1}{h}\sum_{k=1}^{h} \nu_{ij}^{(H,k)} \tag{7}$$

where, ν_{avg} is the average mean of all the *means of* k*-middle*.

The (4)–(7) are described below. A neighborhood vector $RH\ ij$ for a $pij \in I$ of a half filter window of size H using (3) is created. Then, using (5), h different means are evaluated for all possible values of k. Using (6) and (7), $\sigma H\ ij$ and $\Omega H\ ij$ are evaluated. Based on the values of $\sigma H\ ij$ and $v\ (H,k)\ ij$, we can plot h number of GMFs for a filter window using (4).

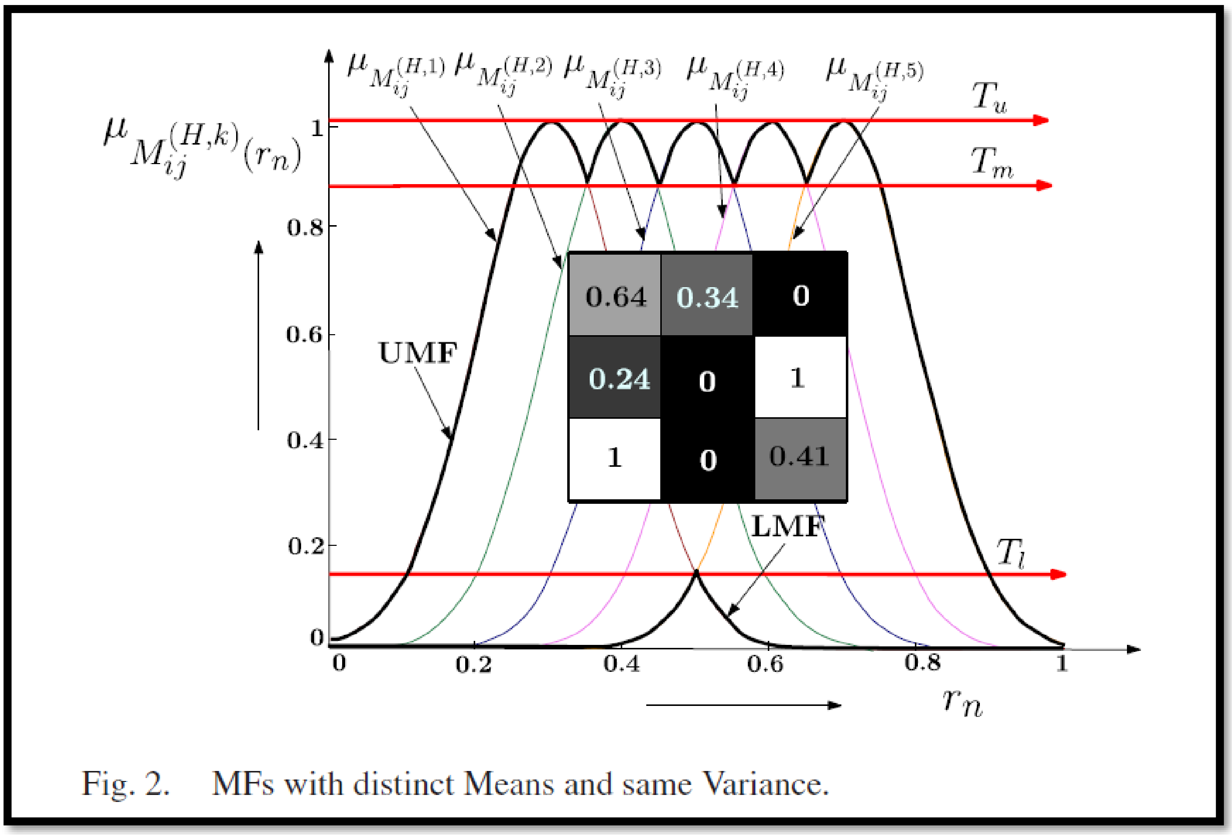


Fig. 2. MFs with distinct Means and same Variance.

For example, if $H = 1$, then $N = 9$ and $h = 5$. Five GMFs are plotted using *mean of k-middle* ($k = 1, 2, \ldots, h$) for a 3×3 window as shown in Fig. 2. Now, let Δij is a matrix consisting membership values of elements $rn \in RH\ ij$ evaluated using (4), (5), and (6). Basically, Δij contains h membership values of each element of filter window corresponding to h GMFs. Hence, the size of matrix is $h \times N$ and can be written as

$$\Delta_{ij} = \begin{bmatrix} \mu_{M_{ij}^{(H,1)}}(r_1) & \mu_{M_{ij}^{(H,1)}}(r_2) & \cdots & \mu_{M_{ij}^{(H,1)}}(r_N) \\ \mu_{M_{ij}^{(H,2)}}(r_1) & \mu_{M_{ij}^{(H,2)}}(r_2) & \cdots & \mu_{M_{ij}^{(H,2)}}(r_N) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{M_{ij}^{(H,h)}}(r_1) & \mu_{M_{ij}^{(H,h)}}(r_2) & \cdots & \mu_{M_{ij}^{(H,h)}}(r_N) \end{bmatrix} \tag{8}$$

A column-wise S-norm (max operation) is performed on the matrix Δ_{ij} followed by T-norm (min operation) on the outcomes to obtain a MF value T_m as shown in Fig. 2 for a 3×3 window. It is the minimum MF value of UMF. In the designed algorithm, T_m is used as threshold for categorizing a pixel. Mathematically, it can be expressed as

$$T_m = \wedge(\vee(\Delta_{ij})) \tag{9}$$

where, $\wedge$ and $\vee$ are the min and max operators, respectively. It is adaptive in nature as compared to threshold based on Type-1 that is heuristic [9]. In the matrix Δ_{ij} , h number of MF values are associated with each pixel intensity. A set of MF values $\mu_{M_{ij}^{H}}$ associated with neighborhood vector R_{ij}^{H} is the column-wise mean value of the matrix Δ_{ij} . Mathematically, it can be expressed as

$$\mu_{M_{ij}^{H}} = \frac{\sum_{k=1}^{h} \Delta_{ij}}{h} \quad \forall \ i = 1 \ldots N \tag{10}$$

2nd Way of approach

B. MFs With Distinct Means and Variances

The second approach is similar to the first one, with only difference in determining the MF values of each pixel.The MF values are obtained by varying both Mean and Variance. Every element r_n in the set R_{ij}^H belongs to fuzzy set M_{ij}^H similar to the first approach with a GMF,

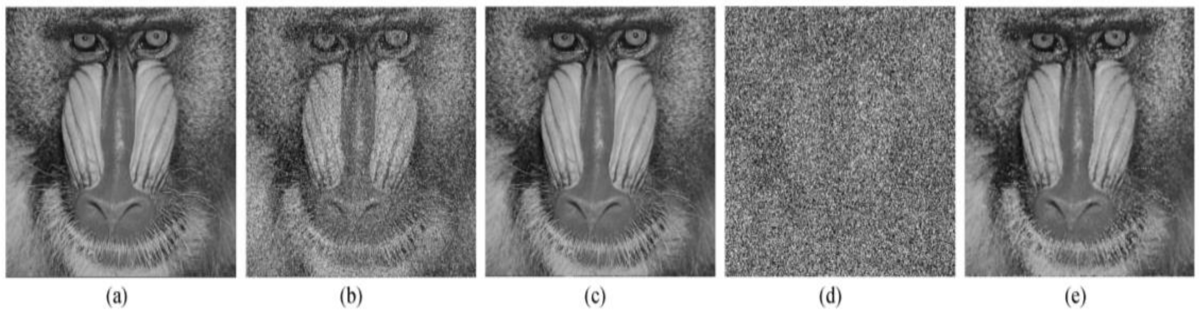


Fig. 5. Baboon image: (a) actual image; (b) at 20% noise; (c) filtered image; (d) at 80% noise; and (e) filtered image.

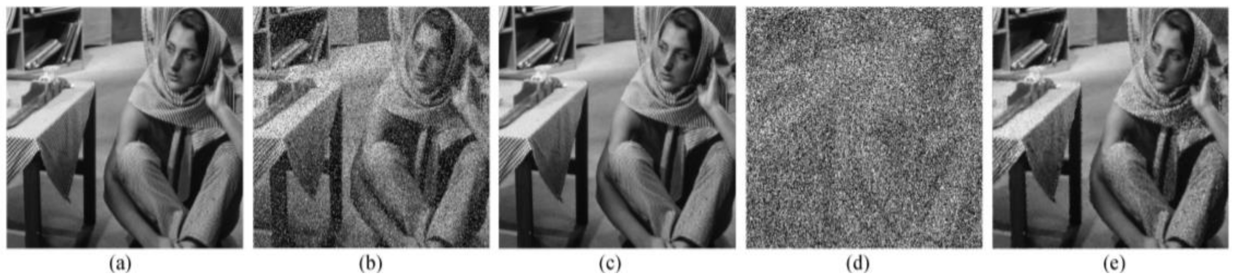


Fig. 6. Barbara image: (a) actual image; (b) at 20% noise; (c) filtered image; (d) at 80% noise; and (e) filtered image.

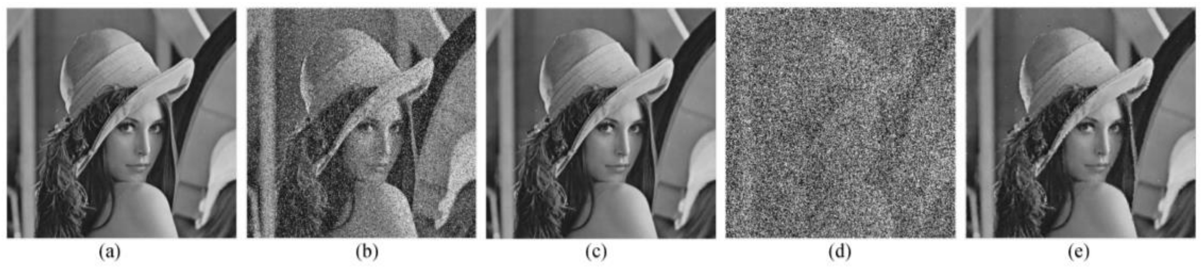


Fig. 7. Lena image: (a) actual image; (b) at 20% noise; (c) filtered image; (d) at 80% noise; and (e) filtered image.

$$\mu_{M_{ij}^{(H,k)}}(r_n) = e^{-(r_n - \nu_{ij}^{(H,k)})^2 / 2(\sigma_{ij}^{H})^2} \tag{11}$$

Both the GMF parameters Means ($\nu_{ij}^{(H,k)}$) and Variance (σ_{ij}^{H}) are varied with respect and are calculated as follows:

$$\nu_{ij}^{(H,k)} = m_k(R_{ij}^{H}), \quad k = 1, 2, 3, \cdots, h \tag{12}$$

$$\sigma_{ij}^{(H,k)} = m_k(\Omega_{ij}^{(H,k)}), \quad k = 1, 2, 3, \cdots, h \tag{13}$$

where, m_k is the mean of k-middle defined in (2). The parameter ΩH ij is defined using l1 norm as follows:

$$\Omega_{ij}^{(H,k)} = \{ \, |r_n - \nu_{ij}^{(H,k)}|, \ \forall \ r_n \in R_{ij}^{H} \, \} \tag{14}$$

The matrix Δ_{ij}^{H} defined in (8) are calculated using (12), (13), and (14). The operations given in (9) and (10) are performed on matrix Δ_{ij}^{H} to obtain the threshold Tm and MF value for 3×3 window. They are shown in Fig. 3. The categorization of pixels are done in similar way as the first approach as mentioned in Algorithm 1 with one change in step 8.

C. Denoising Noisy Pixels Using Type-1 Fuzzy Logic

The pixels categorized as bad (noisy) in step-1 are denoised in this step. The set of good pixels G is considered to be a fuzzy set and each element in it is mapped to [0,1] by MF . μ_G Using this approach, each pixel in the fuzzy set G has a different MF value.GMF is considered for the set of good pixel sin G. First ,mean of GMF is computed by applying mean of k-middle for all the good pixels in G and then an average is taken for all the k-means. The variance is found out using l1 norm of the good pixels with respect to the average mean.The MF plot for the good pixels across a 3×3 window is shown in Fig. 4. The mean of k-middle for good pixels in a particular window is computed using (2). The average mean m avg and variance . σ_G of MF . μ_G are determined as follows:

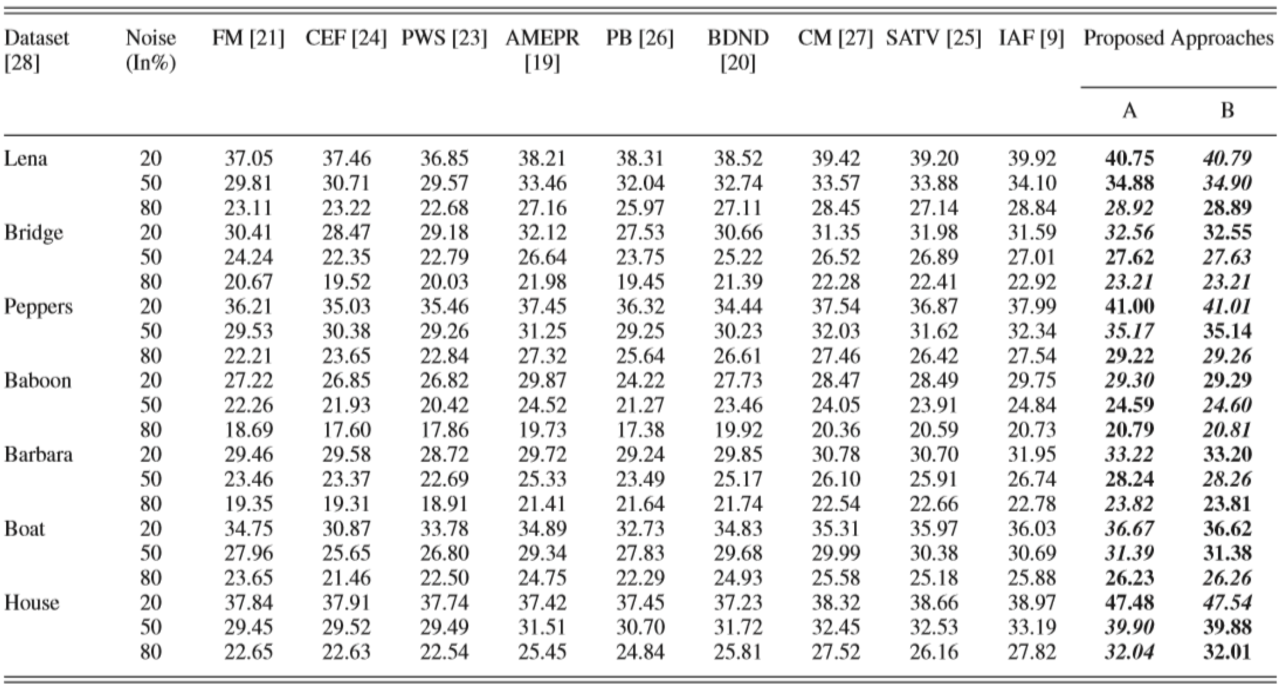
$$m_{\text{avg}} = \frac{\sum_{k=1}^{h} m_k}{h}; \quad \sigma_G = |G - m_{\text{avg}}| \tag{15}$$

$$\mu_G(g_i) = e^{-(g_i - m_{\text{avg}})^2 / 2\sigma_G^2} \tag{16}$$

where, mk is the kth (k =1 ,2,...,h) middle mean of good pixels in G. Finally, the denoised pixel intensity pnew is computed in (17).

$$p^{\text{new}} = \frac{\sum_{\forall g_i \in G} w_i g_i}{W}; \quad W = \sum_{i=1}^{\rho} w_i \qquad (17)$$

wi ∈ μG is the weight corresponding to the ith good pixel, ρ is the number of good pixels in a filter window, and W is the normalizing term. In case of high noise level, there is a chance that ρ (Cardinality of GH ij) will become zero. In such cases, H is increased by 1 and the whole process is repeated. If the deviation σG is below a very small threshold , i.e., the window consists of pixels with intensities very near to that of pij, then the value of pij is simply replaced by m avg. This will restrict the division by zero that may arise due to uniform intensity and make σG zero.



Dataset [28]	Noise (In%)	FM [21]	CEF [24]	PWS [23]	AMEPR [19]	PB [26]	BDND [20]	CM [27]	SATV [25]	IAF [9]	Proposed Approaches	
											A	B
Lena	20	37.05	37.46	36.85	38.21	38.31	38.52	39.42	39.20	39.92	**40.75**	***40.79***
	50	29.81	30.71	29.57	33.46	32.04	32.74	33.57	33.88	34.10	**34.88**	***34.90***
	80	23.11	23.22	22.68	27.16	25.97	27.11	28.45	27.14	28.84	***28.92***	**28.89**
Bridge	20	30.41	28.47	29.18	32.12	27.53	30.66	31.35	31.98	31.59	***32.56***	**32.55**
	50	24.24	22.35	22.79	26.64	23.75	25.22	26.52	26.89	27.01	**27.62**	***27.63***
	80	20.67	19.52	20.03	21.98	19.45	21.39	22.28	22.41	22.92	***23.21***	***23.21***
Peppers	20	36.21	35.03	35.46	37.45	36.32	34.44	37.54	36.87	37.99	**41.00**	***41.01***
	50	29.53	30.38	29.26	31.25	29.25	30.23	32.03	31.62	32.34	***35.17***	**35.14**
	80	22.21	23.65	22.84	27.32	25.64	26.61	27.46	26.42	27.54	**29.22**	***29.26***
Baboon	20	27.22	26.85	26.82	29.87	24.22	27.73	28.47	28.49	29.75	***29.30***	**29.29**
	50	22.26	21.93	20.42	24.52	21.27	23.46	24.05	23.91	24.84	**24.59**	***24.60***
	80	18.69	17.60	17.86	19.73	17.38	19.92	20.36	20.59	20.73	**20.79**	***20.81***
Barbara	20	29.46	29.58	28.72	29.72	29.24	29.85	30.78	30.70	31.95	***33.22***	**33.20**
	50	23.46	23.37	22.69	25.33	23.49	25.17	26.10	25.91	26.74	**28.24**	***28.26***
	80	19.35	19.31	18.91	21.41	21.64	21.74	22.54	22.66	22.78	***23.82***	**23.81**
Boat	20	34.75	30.87	33.78	34.89	32.73	34.83	35.31	35.97	36.03	***36.67***	**36.62**
	50	27.96	25.65	26.80	29.34	27.83	29.68	29.99	30.38	30.69	***31.39***	**31.38**
	80	23.65	21.46	22.50	24.75	22.29	24.93	25.58	25.18	25.88	**26.23**	***26.26***
House	20	37.84	37.91	37.74	37.42	37.45	37.23	38.32	38.66	38.97	**47.48**	***47.54***
	50	29.45	29.52	29.49	31.51	30.70	31.72	32.45	32.53	33.19	***39.90***	**39.88**
	80	22.65	22.63	22.54	25.45	24.84	25.81	27.52	26.16	27.82	***32.04***	**32.01**

This table showing comparison our proposed approach with several state of the art approaches existing.

D. Stopping Criterion

The stopping criteria for algorithm is defined by percentage of denoised pixels *α* given in (18). The number of noisy pixels detected in succeeding iterations is denoted by *dm* . The algorithm terminates when *α* falls below 0.05%.

$$\alpha = \frac{d_m^{\text{new}} - d_m^{\text{old}}}{X \times Y} \tag{18}$$

where, *X* × *Y* represents the dimension of 8-bit images.

5. RESULTS AND VALIDATIONS

The proposed filter has been used for removing SAP noise from seven standard grayscale images of resolution 512 × 512. The validations were carried out on a system with Intel Core i3, 1.7 Ghz processor and 4 GB RAM. The performance of proposed filter has been compared with various state-of-the-art algorithms. For all these images, the parameter *k* is varied form 1, 2, . . . , *h* in both the approaches of step 1 of filter to compute the mean and variance. The MFs are plotted for both the approaches to decide respective threshold. The threshold is adaptive in nature that depends on SAP noise level. The SAP noise level is varied from 20% to 99%. For comparative analysis, the results for three different noise levels viz. 20%, 50%, and 80% are shown in Table I. For the noise level of 97%,
the results are shown (Fig. 8) with the images of Lena, Baboon, House, and Peppers. The filter operation at noise level of 98% and 99% are tested only for peppers. The proposed approaches preserve significant image details even for 99% of noise level (shown in Fig. 8). To perform the experimentation, we have set a minimum number of good pixel (*ρ*min) as 1 in filter window to estimate the pixel intensity of bad pixels. This is because a large number of good pixels are required for denoising bad pixel. Having *ρ*min more than 1, the results are poor for high noise level. The performance of proposed approaches are numerically evaluated using PSNR. The PSNR is defined using filtered image (*If*) with respect to original image (*Io*) as follows:

$$\mathrm{PSNR}(I_o, I_f) = 10\log_{10} \frac{255^2}{\frac{1}{XY}\sum_{i,j}(I_o(i,j) - I_f(i,j))^2}$$

where, 255 is the maximum pixel intensity of 8-bit images. Table I shows the performance of proposed approaches A and B with respect to various state-of-the-art algorithms, namely, iterative adaptive fuzzy filter (IAF), adaptive median with edge-preserving regularization (AMEPR), boundary discriminative noise detection (BDND), fast median (FM), wavelet neural network, pixel-wise S-estimate of variance (PWS), contrast enhancement based filter (CEF), spatially adaptive total variation filter (SATV), patch-based (PB), and the cloud model (CM) filter. The PSNR values given in Table I are averaged over 20 experiments for each image. The computational time of the proposed approaches are comparable to state-of-the-art algorithms for low noise levels, as shown in Table II. With the increase in noise level, the filter window size increases. For large windows, more MF's are drawn to compute the threshold. The computational time is relatively higher in such cases.

Appendix A

MATLAB CODE :-

K – MIDDLE FUNCTION:-

```
%% K-Middle Mean Evaluation
function mean=K_middle_mean(k,Vector)
Lenght_of_Vector = numel(Vector);
Vector=sort(Vector);
if mod(Lenght_of_Vector,2) == 1
    half_lengh = 0.5*(Lenght_of_Vector+1);
    factor = 1/(2*k - 1);
    K_middle_vector=Vector((half_lengh-k+1):(half_lengh+k-1));
    sum_of_element=sum(K_middle_vector);
else
    half_lengh=0.5*Lenght_of_Vector;
    factor=1/(2*k);
    K_middle_vector=Vector((half_lengh-k+1):(half_lengh+k));
    sum_of_element=sum(K_middle_vector);
end
mean=factor*sum_of_element;
end
```

MEMBERSHIP TYPE 2 FUNCTION :-

```
function
[T_min,T_max,T_min_max,PI,H,sigma,average_mu]=membership_type_2(vector)
%%
p=numel(vector);
H=(p+1)/2;

mu=zeros(1,H);
for q=1:H
    mu(1,q)=K_middle_mean(q,vector);
    sigma(1,q)=1.5*(sum(abs(vector-mu(1,q))))^2;

if sigma(1,q) < 0.0001
    sigma(1,q)=0.0001;
end
end

average_mu=((sum(mu))/H)*ones(1,p);

mu_Mat=repmat(mu,p,1)';

lembda_vector=repmat(vector,H,1);


for i=1:length(mu_Mat(:,1))
    for j=1:length(mu_Mat(1,:))
        PI(i,j)= exp(-0.5*((lembda_vector(i,j)-mu_Mat(i,j)).^2/sigma(i)));
    end
end
```

```
T_max=max(max(PI));

T_min_max=min(max(PI));

T_min=max(min(PI)); %%

end
```

VECTOR Rij FUNCTION:-

```
 function [vector,index, Window]=R_ij_M(image,i,j,M)
x=(i-M):(i+M);
y=(j-M):(j+M);
neighborhood_length = (2*M+1)^2;
Window=image(x,y);
vector=reshape(Window,[1,neighborhood_length]);  % vector of 1*9
index=combvec(x,y);
end
```

MAIN PROGRAM:-

```
close all;
clear;
clc;

%% Salt and Pepper Noise removal Using type 2 fuzzy system
input_image=imread('Lenna.png');
figure;
imshow(input_image);
title('Input image');
%%
im_gray=rgb2gray(input_image);
figure;
imshow(im_gray);
title('Input image after rgb 2 gray');
%%
im_gray_1=im2double(im_gray);
figure;
imshow(im_gray_1);
title('Input gray image in double');

%% Inialization Of Parameters
Noise_density = 0.50;
im_noised=imnoise(im_gray_1,'salt & pepper',Noise_density);
figure;
imshow(im_noised);

title(sprintf('Input noisy image with %d noise density',Noise_density));
[p,q]=size(im_noised);
%% Padding
im_denoised=0.63*ones(p+16,q+16);
im_denoised(9:p+8,9:q+8)=im_noised;
epsilon=0.0001;
count0=0;
count1=0;
count2=0;
count3=0;
M=1;  % M is half window size %% Inialization
```

```
N_init = 8;    % for low noise image N can be low
im_denoised_pixels = zeros(p+16,q+16);
im_noised_pixels = zeros(p+16,q+16);
%%
time_elapsed_per_itteration = zeros(1,10);
tic
e=1;
for z=1:2
    im_denoised_pixels = zeros(p+16,q+16);
    im_noised_pixels = zeros(p+16,q+16);

    for j=9:q+8 % To scan rows
        for i=9:p+8 % To scan col.
            M = 1;  % M is half window size %% Inialization
            N_init = 6;    % for low noise image N can be low
            %
            S_max = 2; % upper bound of 'M'
            N = N_init; % Inialisation of number of good pixel required to
evaluate the pixel value of a currupted pixel

            while (im_denoised(i,j)==0)||(im_denoised(i,j)==1)

                [R_ij_M_matrix,index] = R_ij_M(im_denoised,i,j,M);

                [T_min,T_max,T_min_max,PI,H,sigma,average_mu] =
membership_type_2(R_ij_M_matrix);

                lenght_R_ij_M_matrix = length(R_ij_M_matrix);
                ave_PI = sum(PI)/H;
                T_Threshold = T_max;
                o = 1;
                if ave_PI(H)>T_Threshold
                    count0=count0+1;
                    break
                elseif sigma==epsilon

                    im_denoised_pixels(i,j) = average_mu(1);
                    im_noised_pixels(i,j) = im_denoised(i,j);
                    count1=count1+1;
                    break
                end
                G=zeros(1,N);
                for x=1:lenght_R_ij_M_matrix
                    if  ave_PI(x)>=T_Threshold
                        count2=count2+1;
                        G(o)=R_ij_M_matrix(x);
                        o=o+1;

                    elseif  (R_ij_M_matrix(x)~=0)&&(R_ij_M_matrix(x)~=1)
                        count3=count3+1;
                        G(o)=R_ij_M_matrix(x);
                        o=o+1;
                     end
                    if o==N+1
                        break
                    end
                end

                neta=length(find(G));
                if (neta<N) && (M< S_max)
                    M=M+1;
                    continue
                elseif (neta<N) && (M== S_max)
                    N=N-1;
                    if N<1
                        S_max=S_max+1;
                        N=1;
                    end
                    continue
                end
                for k=1:(length(G)/2)
                    mean(k)=K_middle_mean(k,G);
                end
                mean_G=((sum(mean))/length(G));
                var_G=2.5*abs(G-mean_G);
                var_G=max(var_G);
                if var_G<=.01
                    var_G=.01;
                end
                w=gaussmf(G,[var_G,mean_G]);
                W=sum(w);
                weighted_G=w*G';

                im_denoised_pixels(i,j) = weighted_G/W;
                im_noised_pixels(i,j) = im_denoised(i,j);
                break
            end
        end
    end
    time_elapsed_per_itteration(z)=toc;
    im_denoised=(im_denoised-im_noised_pixels)+im_denoised_pixels;
end
time_elapsed=toc;

%%

im_denoised=im_denoised(9:p+8,9:q+8);

im_denoised_1 = im2uint8(im_denoised);

im_denoised_1=int16(im_denoised_1);

im_gray=int16(im_gray);

PSNR=10*log10((255*255)/((1/((p-10)*(q-10)))*sum(sum((im_denoised_1(6:p-
5,6:q-5)-im_gray(6:p-5,6:q-5)).^2))));

fprintf('PSNR is %d\n',PSNR);

figure;
imshow(im_denoised);
title(sprintf('Denoised image with %d noise density',Noise_density));
```