FUZZY STACK FILTERS FOR IMAGE PROCESSING

MARCHEVSKÝ, S. - STUPÁK, Cs.
Department of Electronics and Multimedial Communications
FEI TU of Košice
Park Komenského 13
040 21 Košice
Slovakia

Abstract

This paper is oriented to filtering by fuzzy stack filter of monochromatic images distorted with impulsive noise. Fuzzy stack filter is acquired by extension of stack filters by means of fuzzy logic. Adding some parameters to this filter, that are adjusted by neural adaptation algorithm, is obtained the new class of filters, so-called fuzzy rank-order filters. This class of filters is compared with other well known filters as stack filters and neural stack filters.

Keywords

Image processing, stack filter, fuzzy logic, median filter, neural filter, genetic algorithm.

1. Introduction

During transmission of information over the noisy channel, frequently give out to distortion of data with noise. In nature two types of noise typically occur, *additive Gaussian* and *impulsive noise*. To eliminating the first type of noise are utilised the linear digital filters and for elimination of second type of noise are utilised non-linear digital filters, for instance *median filters*.

Extending the median filter is obtained another class of filters, so-called *stack filters*, that are based on *threshold decomposition* (TD) and *stacking property*. Mentioned properties have only the *positive Boolean functions*.

The design of *optimal stack filter* (OSF) consists of looking for such PBF, that suppress noise but concurrently preserves details of image. OSF should be designed by *linear programming technique*, but such design have great requirement for computation. Consequently are looked for another simplest methods for design, so-called sub-optimal methods.

In case of sub-optimal methods the design of the OSF may be performed by means of *neural network*, *genetic algorithm*, *fuzzy logic*, etc.

2. Stack Filters

The simplest non-linear digital filter, that is used for image reconstruction is the *median filter* (MF) [5]. The output of the MF is acquired by classifying the data from the *operation window* (OW) in ascending order and choosing the middle element. The OW in case of image processing is an square window with odd size, which sliding over the image from left top corner to right bottom corner. Mathematics description of MF is following:

$$x_{vm} \in \{x_i; i = 1,\ldots,N\} \tag{1}$$

and for all $j = 1,\ldots,N$

$$\sum_{i=1}^{N} |x_{vm} - x_i| \le \sum_{i=1}^{N} |x_j - x_i| \tag{2}$$

where x_{vm} is the MF output, x are the input data from operation window, N is the size of OW.

Extending the MF by TD get out the *stack filter* (SF) [3]. The TD is distribution of input data in particular quantization level:

$$X_i = \sum_{m=1}^{M-1} T^m(X_i) = \sum_{m=1}^{M-1} x_i^m \tag{3}$$

where

$X_i = [X_1, X_2, \ldots, X_N]$ are the input data

$$x_j^m = T^m(X_j) = \begin{cases} 1 & \text{if } X_j \ge m \\ 0 & \text{if } X_j < m \end{cases}$$

where M is the number of quantization levels, T is the TD operator, X are the input data from OW.

The structure of the filter given above (Fig. 1) has *stacking property*, that is defined as follows: if the binary value at specific level of TD architecture is equal to zero, then the values are equal to zero at higher level too. Consequently, the input data are decomposed to set of ones on which are set of zeros. This property satisfy only positive Boolean functions, that can be represented by sum of products (*minterms*) or product of sums (*maxterm*).

SF with OW size $N = 3$, for $M = 4$ level input data shown follow:

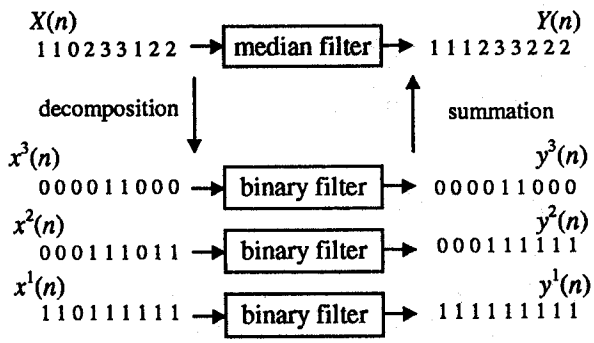

Fig.1 The TD architecture of SF

The binary filters at particular level of TD architecture can be realised in general by PBF, which are set of binary operations (AND, OR). These operations can be simply realised. If all filters in the TD architecture are equal, then it is a *homogenous* SF, otherwise *non-homogenous* SF.

The most frequently used representation of PBF is the so-called *minterm* form (sum of products).

$$f(x_1,\ldots,x_n) = \sum \alpha(x_1,v_1)\ldots\alpha(x_n,v_n) \quad (4)$$

where x are input data from OW with size 3x3, v is an integer {0, 1}, and

$$\alpha(x_i,1) = x_i,\ \alpha(x_i,0) = 1$$

For example: Let v = { 010, 101 }, then

$$f(x_1,x_2,x_3) = x_2 + x_1x_3$$

The PBF can be divided into two types [2], [6]:

- *positive* PBF $on(f) = \{\underline{x}: f(\underline{x}) = 1\}$
- *negative* PBF $off(f) = \{\underline{x}: f(\underline{x}) = 0\}$

The design of OSF is based on searching such PBF, which best suppress the noise and keeps details of image, or other said, has the *minimal mean square error* (MSE) or the *minimal mean absolute error* (MAE). The OSF can be designed by method of linear programming, but such representation has great computation demand and consequently looked for sub-optimal methods. These methods are based on a fact, that PBF in TD structure can be replaced by different structures (filters), as by:

- micro-statistic filters (case of linear filter)
- neural filters
- fuzzy filters

The structures given above considerably reduce the computation effort, therefore are preferred by designers. The OSF designed by method of linear programming for image *Bridge* distorted by 10% impulsive noise is defined in [1].

$$\begin{aligned} y = {} & x_4x_5x_6 + x_2x_5x_8 + x_5x_6(x_1 + x_2 + x_3 + \\ & x_7 + x_8 + x_9) + x_4x_5(x_1 + x_2 + x_3 + x_7 + \\ & x_8) + x_2x_5(x_1 + x_3 + x_7) + x_5x_8(x_3 + \\ & x_7 + x_9) + x_5(x_1x_3x_7 + x_1x_3x_9 + \\ & x_1x_7x_9) + x_1x_3x_6x_7x_8x_9(x_2 + x_4) + \\ & x_1x_2x_3x_4x_7x_9(x_6 + x_8) + x_2x_4x_6x_8 \cdot \\ & (x_1x_3 + x_1x_7 + x_3x_7 + x_3x_9 + x_7x_9) \end{aligned} \quad (5)$$

The simulation results in [1] show, that this filter has the best performance for 10% impulsive noise, however implementation of this filter is relatively complicated.

2.1 Neural Stack Filter

Representation of binary filters by *neural networks* (NN) [10] is preferred because they simple search optimal PBF [1]. Such design is simpler and faster towards linear programming. The realisation of one binary filter in TD architecture of SF (Fig.1) by one neurone is shown below:

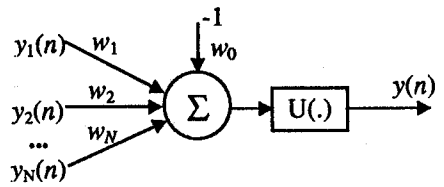

Fig.2 The architecture of neural stack filter

Such architecture of stack filter is also called as weighted order filter. The output of neurone is defined as weighted sum of inputs from the OW of length N and consecutive application of non-linear function U [8].

$$y(n) = U\left(\sum_{i=1}^{N} x_i w_i - w_0\right) \quad (6)$$

where w are the weights of the neurone, x input of the OW, U is a non-linear function, N is the size of OW and in some application w_0 is also called as *bias*.

The non-linear function of equation (6) is so-called activation function of neurone, which is defined as:

- *hard-decision* function

$$U(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (7a)$$

- *linear* function

$$U(x) = \beta x \quad (7b)$$

- *sigmoidal* function

$$U(x) = 1/\left(1+\exp\left(-\beta x\right)\right) \tag{7c}$$

- *tangential* function

$$U(x) = \frac{1-\exp\left(-\beta x\right)}{1+\exp\left(-\beta x\right)} \tag{7d}$$

where x is the input (weighted sum) and β is the coefficient of activation function.

There are two state of NN, the state of *training* (adaptation), where the weights are adjusted for finding the optimal solution, and the state of *activation*, where the NN works as filter. In case of image filtration the most widely used activation function is the *sigmoidal activation function*, and the most frequently used adaptation algorithm is so-called *backpropagation* algorithm defined as:

$$\begin{aligned} w_0(n) &= w_0(n-1) - \mu\,\delta(n) \\ w_i(n) &= w_i(n-1) - \mu\,\delta(n)\,x(n) \\ \delta(n) &= \left(d(n)-y(n)\right)y(n)\left(1-y(n)\right) \end{aligned} \tag{8}$$

where d is the desired output (training sample), w_i and w_0 weights of neurone, y is the output of neurone and μ is the adaptation coefficient (small positive number).

In order to find the optimal PBF is necessary to keep the weights non-negative, failing which the stacking property is not valid. Sometimes, especially in case of complex NN the searching of optimal solution is very difficult. The NN in many time find the *local optimum* instead of *global optimum*. In order to eliminate this problem were developed the other methods, for instance training the NN by *evolution algorithm* also called *genetic algorithm*.

2.2 The genetic algorithm

The *genetic algorithm* (GA) comes out from analogy of evolution process of nature. This algorithm has ability to find the global optimum although the NN is more complex [4], [9]. The GA has some characteristic item as *genes*, that create *chromosomes*, that forth create the *population*. NN can be described by one chromosome, that items are *genes* which represent the weights of NN. In this case each *chromosomes* of *population* describe this one NN, but with different weights. The task of GA is find out the most fitness *chromosome* of *population*, which has the best filtering performance.

Following parameters should be determined for fitting the GA to the given problem:

- genetic representation of possible solutions
- way to compute an evaluation function
- genetic operators realization
- tuning parameters used in GA

The GA is described below:

1) *initialisation* - the basic population created, each chromosome (weights of NN) is chosen accidentally.
2) *evaluation* - the *fitness* of each chromosome computed.
3) *reproduction* - new population (descendent) created from the most fitness chromosome of previous population (parents) by using genetic operators such as *mutation* and *crossover*.
4) *evaluation* - the *fitness* of each new chromosome computed.
5) *selection* - the old chromosomes (parents) replaced with the lowest fitness of previous population by new chromosomes (children) with the highest fitness.
6) repeat from 3, while any condition is reached.

The most frequently used genetic operators:

- *crossover* - arisen by arithmetic average of two randomly chosen chromosomes;
- *mutation* - arisen by addition a Gaussian random number with zero mean to the *gene*.

The listed algorithm is simple. The most acquisition of this algorithm is finding the global optimum unlike other gradient method. However on the other hand this method has an important drawback, the searching of global optimum is rather based on random also called *Monte Carlo searching* and therefore that time of finding the global optima is unpredictable, usually very long.

2.3 Fuzzy Stack Filter

Fuzzy logic (FL) as well as the NN is very useful and widely used [7]. In practical application the FL more preferred as NN. The filter based on FL does not need any training signal and the computation demand is less in contrast of NN. Of course the performance of fuzzy filters is usually worse than NN, because there is no adaptation algorithm or searching the optimal performance. The performance of this filter is given by architecture of the filter. To improve the performance of the filter can be accomplished by combining FL and NN [11].

The fuzzy logic briefly can be described as follows. If D is a set, then a fuzzy set A in D is a set of ordered pairs as follows [2]:

$$A = \left\{\left(\underline{x},\, u(\underline{x})\right) : \underline{x} \in D\right\} \tag{9}$$

where u is called the membership function of x in A and $u(.)$ is a so-called *fuzzy* value or fuzzy degree of x in A. Elements in A with zero degree of membership are normally not listed.

The task of this paper is to find fuzzy representation of the PBF (FPBF). Henceforth the FPBF is derived for the positive PBF. Subsequently for FPBF $F\ \{0,1\}^n \rightarrow \langle 0,1\rangle$ is deal [2], [6]:

- if x belongs to $on(f)$, then $F(\underline{x}) = 1$
- if x does not belong to $on(f)$ (it belongs to $off(f)$, then $0 \le F(\underline{x}) < 1$
- stacking property, if $\underline{x} \le \underline{y}$, then $F(\underline{x}) \le F(\underline{y})$

Such FPBF can be written as follows

$$F(x_1,\ldots,x_n) = \sum_{i=1}^{n}\left(v_i x_i + p_i v_i \bar{x}_i\right) \Big/ \sum_{i=1}^{n} v_i \quad (10)$$

where x and v are defined similarly as in formula (4), p is coefficient of similarity and it is adjusted by adaptation algorithm

For example: Let $v = \{\ 101\ \}$, then

$$F(x_1,x_2,x_3) = \left[\left(x_1 + p_1\bar{x}_1\right) + \left(x_3 + p_3\bar{x}_3\right)\right]/2$$

Table 1 The fuzzy truth table

x_1	x_2	x_3	$F_{101}(x_1, x_2, x_3)$
0	0	0	$(p_1 + p_3)/2$
0	0	1	$(p_1 + 1)/2$
0	1	0	0
0	1	1	$(p_1 + 1)/2$
1	0	0	$(1 + p_3)/2$
1	0	1	1
1	1	0	$(1 + p_3)/2$
1	1	1	1

The equation of FPBF (10) holds only for one term of (4), therefore the complete FPBF is defined as:

$$F(\underline{x}) = \Phi(F_1,\ldots,F_m) \quad (11)$$

where F_i are separated members of formula (10), Φ is *maximum operator*, generally it is so-called *s-norm.*

Similarly can be deduced the FPBF for negative PBF [2].

$$G(x_1,\ldots,x_n) = \sum_{i=1}^{n}\left(v_i x_i + q_i v_i \bar{x}_i\right) \Big/ \left(n - \sum_{i=1}^{n} v_i\right) \quad (12)$$

By means of FPBF the *fuzzy stack filter* (FSF) is defined as follows:

$$Y_n = \sum_{l=1}^{M} F\left(T_l(\underline{X}_n)\right) \quad (13)$$

where T is the operator of TD, X_n is input vector from operation window 3x3, F is a FPBF (10), M is the number of quantization levels (for grey images is 256).

The disadvantage of the filter moreover remains necessity of the optimal FSF deduction. Similarly as in case of OSF the optimal solution can be derived by means of linear programming, but it was be very difficult for computing. Therefore after modification of FPBF (10), the fuzzy rank-order stack filter is defined as:

$$F_k(\underline{x}) = \frac{l + p(i_1) + \ldots + p(i_{k-l})}{k} \quad (14a)$$

$$G_k(\underline{x}) = \frac{l + q(i_1) + \ldots + q(i_{N-k+1-l})}{N-(k-1)} \quad (14b)$$

where l for equation (14a) is number of ones in OW and for equation (14b) is number zeros, N is the size of OW (odd number), p and q are coefficients of filter, ordered by size, x is the input vector and k is the rank of FPBF.

Subsequently the rank-order FSF is defined as:

$$F_{m,n}(\underline{x}) = \begin{cases} 1, & \text{if } l \ge n \\ 0, & \text{if } l < m \\ \text{select}\left(F_n(\underline{x}), G_m(\underline{x})\right) & \text{other} \end{cases} \quad (15)$$

where l is the number of ones in OW and

$$\text{select}\left(F_n(\underline{x}), G_m(\underline{x})\right) = \begin{cases} F_n(\underline{x}), & \text{if } F_n(\underline{x}) \ge G_m(\underline{x}) \\ 1 - G_m(\underline{x}), & \text{if } F_n(\underline{x}) < G_m(\underline{x}) \end{cases} \quad (16)$$

Coefficients p and q are calculated by neural adaptation algorithm, which is mimic as *Hebbian* learning (the win takes all):

- if $F_n(\underline{x}) > G_m(\underline{x})$ and the desired output is zero, then for $j = 1, \ldots, N$:

$$p_j(i) = p_j(i-1) - \alpha_1 \bar{x}_j\left(F_n(\underline{x}) - G_m(\underline{x})\right) \quad (17a)$$

$$q_j(i) = q_j(i-1) + \alpha_2 x_j\left(F_n(\underline{x}) - G_m(\underline{x})\right) \quad (17b)$$

- if $F_n(\underline{x}) < G_m(\underline{x})$ and the desired output is one, then for $j = 1, \ldots, N$:

$$p_j(i) = p_j(i-1) + \alpha_1 \bar{x}_j\left(G_m(\underline{x}) - F_n(\underline{x})\right) \quad (17c)$$

$$q_j(i) = q_j(i-1) - \alpha_2 x_j\left(G_m(\underline{x}) - F_n(\underline{x})\right) \quad (17d)$$

where α_1, α_1 are adaptation coefficients (small positive numbers)

The listed algorithm adjusts the coefficients so that G_m will close to zero and F_n to one. There is seen, that the algorithm is simply and intuitive, however the initiated approximation of PBF doesn't have to be the optimal.

3. Simulation Results

In this part are compared the derived filters given above. The comparison was performed following minimal mean absolute error (MAE), which indicates degree of original image preservation, and following minimal mean square error (MSE), which indicates degree of noise suppressing.

The simulation was accomplished for two types of images, for image *Bridge*, which is more detailed image, and for image *Lena*, which is less detailed image. The size of images was 256x256 and 256 grey levels. The images were distorted by impulsive noise *I* (random numbers from interval 0, 255), and black and white impulsive noise *BW* (values 0, 255).

The neural stack filter (NSF) consists of one neurone with sigmoidal activation function. The learning of NSF was realised by means of backpropagation algorithm (BA) and by genetic algorithm (GA) on the image *Bridge* distorted by 20% black and white impulsive noise. The fuzzy rank-order filter was trained as well as NSF on image *Bridge*. The simulation has shown that the best suppressing of noise and robustness of this filter is reached for rank $m, n = 3, 9$.

Table 2 Simulation results for image *Lena*

Filter	*MF*		*FSF*	
Image	*MAE*	*MSE*	*MAE*	*MSE*
original	4,243	75,8	3,662	54,6
I 5%	4,589	85,0	4,254	67,3
I 10%	4,888	94,3	4,817	80,0
I 20%	5,633	122,9	6,130	122,6
BW 10%	4,851	98,6	5,394	103,4
BW 20%	5,580	147,0	7,524	217,0
BW 40%	8,949	543,1	14,198	849,1

Simulation has shown that the *MF* has the best robustness. The NSF trained by BA has better performance for lesser noise and trained by GA has better performance for higher noise. The fuzzy rank-order filter has excellent performance especially for monotone image *Lena*. The OSF has the best performance for lesser noise, because the filter was trained for 10% impulsive noise and not for 20% black & white impulsive noise.

Table 3 Simulation results for image *Lena*

Filter	*NSF - BA*		*NSF - GA*	
Image	*MAE*	*MSE*	*MAE*	*MSE*
original	3,238	47,2	7,356	131,8
I 5%	3,603	56,0	7,743	144,3
I 10%	3,986	66,3	8,185	159,6
I 20%	5,071	105,0	9,270	201,2
BW 10%	4,120	79,6	8,191	162,6
BW 20%	5,796	180,5	9,524	232,4
BW 40%	12,648	759,1	14,393	590,3

Table 4 Simulation results for image *Bridge*

Filter	*MF*		*FSF*	
Image	*MAE*	*MSE*	*MAE*	*MSE*
original	7,361	148,9	5,996	102,0
I 5%	7,668	159,2	6,570	122,4
I 10%	8,042	173,7	7,160	143,5
I 20%	8,952	212,9	8,544	201,6
BW 10%	7,996	177,1	7,662	165,6
BW 20%	8,897	235,9	9,723	283,3
BW 40%	12,345	605,4	15,963	875,2

Table 5 Simulation results for image *Bridge*

Filter	*NSF - BA*		*NSF - GA*	
Image	*MAE*	*MSE*	*MAE*	*MSE*
original	5,166	77,7	7,356	131,8
I 5%	5,623	93,5	7,743	144,3
I 10%	6,146	112,0	8,185	159,6
I 20%	7,458	165,9	9,270	201,2
BW 10%	6,190	119,7	8,191	162,6
BW 20%	8,000	230,2	9,524	232,4
BW 40%	14,490	782,1	14,393	590,3

Table 6 Simulation results for OSF

OSF	*Lena*		*Bridge*	
Image	*MAE*	*MSE*	*MAE*	*MSE*
original	1,391	18,1	2,074	29,5
I 5%	1,844	36,1	2,800	61,7
I 10%	2,395	61,0	3,616	100,4
I 20%	4,143	172,8	5,789	240,4
BW 10%	2,757	142,2	3,862	160,7
BW 20%	5,912	545,4	7,328	587,2
BW 40%	17,523	2189,8	18,622	2178,8

Fig.3 Original image *Lena* and *Bridge*

Fig.4 Simulation result

a) distorted image - Imp. 10% b) median filter

c) NSF - BA d) fuzzy filter

4. Conclusion

In this paper was described the new class of stack filters, so-called fuzzy stack filter. The presented filter is obtained by extending of PBF by means of fuzzy logic. The structure of filter includes all characteristic of stack filters. In order to simplify the design, the sub-optimal method was used. Consequently was added some control parameters to FSF, which were adjusted by neural adaptation algorithm.

The simulation results of grey-level image filtering by FSF approach to results of NSF. The new architecture of non-selective filter offers a new approach to improve performance of selective filters. The future investigation may be focused to improve the performance of FSF by looking for optimal representation of FPBF and preferable training algorithm with orientation to enhance the speed of convergence.

References

[1] DRUTAROVSKÝ, M.: Neural weighted order statistic filters based on threshold decomposition architecture. Kandidátska dizertačná práca. FEI TU, Jún 1995, Košice, (in Slovak)

[2] PAO-TA YU - RONG-CHUNG CHEN: Fuzzy Stack Filters - Their Definitions, Fundamental Properties, and Application in Image Processing. IEEE Transactions on image processing. Vol.5, No.6, June 1996, pp. 838-854.

[3] WENDT, P. D. - COYLE, E. J. - GALLAGHER, N. C.: Stack Filters. IEEE Trans. on Acoustics, Speech, and Signal Proc., Vol. ASSP-34, No.4, August 1986, pp. 898-911

[4] MARCHEVSKÝ S. - SEGEDA V. - SERRE I. - CHOMAT O.: Neural filter based on genetic algorithm. Proceedings of the Int. Conference „Neural network and their applications", Nov.1996, pp.34-39

[5] MOUCHA V.: Median digital filtering of two dimensional signals distorted by impulsive noise used in air correlation-extremal navigation systems. Habilitačná práca, Vysoká vojenská letecká škola Košice, Júl 1992, (in Slovak)

[6] STUPAK CS. - MARCHEVSKÝ S.: Fuzzy stack filters in image processing. New trends in signal processing IV. Liptovský Mikuláš, Máj. 1998, pp. 142-146, (in Slovak)

[7] MENDEL J. M.: Fuzzy logic systems for engineering: A tutorial, Proceedings of the IEEE, Vol. 83, No.3, March 1995, pp. 345-377

[8] Yin L. - ASTOLA J. - NEUVO Y.: A new class of nonlinear filters - Neural filters. IEEE Transaction on signal processing, Vol.41, No.3, March 1993, pp.1201-1222

[9] ALBRECHT R. F. - REEVES C. R. - STEELE N. C.: Artificial neural nets and genetic algorithms. Proceedings of the international conference in Innsbruck, Austria, 1993

[10] ORAVEC, M. - PODHRADSKÝ, P.: Image Compression Using Neural Networks. Electrical Engineering. 46 (1995), No.9, pp. 307- 317.

[11] BOSSLEY K. M.: Neurofuzzy Modeling Approaches in System Identification. Ph.D. thesis. Faculty of Engineering and Applied Science. Department of Electronics and Computer Science. University of Southampton. UK. May 1997

About authors...

Csaba STUPÁK received the Ing. degree from the Technical University of Košice, Slovak Republic, in Electronics and multimedial communications in 1997. Currently, he is Ph. D. student at the department of Electronics and multimedial communications of Technical University, Košice. His research interest includes image filtering, neural network and fuzzy logic.

Stanislav MARCHEVSKÝ received the M. S. degree in electrical engineering from the Faculty of Electrical Engineering, Czech Technical University in Prague, in 1976 and Ph. D. degree in radioelectronics from Technical University in Košice in 1985. From 1987 he is the associate professor at the FEI TU in Košice.