Rearranging the inequality results in

$$\mathbf{P}_{ii} > \sum_{k=1}^{n} \mathbf{A}_{ki}^2 \mathbf{P}_{kk}$$

and, finally, moving the $\mathbf{P}_{ii}$ term from the sum to the left-hand side gives us

$$(1 - \mathbf{A}_{ii}^2)\mathbf{P}_{ii} > \sum_{k=1, k\neq i}^{n} \mathbf{A}_{ki}^2 \mathbf{P}_{kk}.$$

The right-hand term is greater than zero since the diagonal entries of $\mathbf{P}$ are positive and at least one entry in the ith column of $\mathbf{A}$ is nonzero (from the nonzero columns corollary). This means that $(1 - \mathbf{A}_{ii}^2)\mathbf{P}_{ii} > 0$. Since $\mathbf{P}_{ii}$ must be positive, the inequality can be reduced to $1 - \mathbf{A}_{ii}^2 > 0$ which implies that $|\mathbf{A}_{ii}| < 1$, a contradiction. Q.E.D.

The maximum diagonal zeroes corollary places a bound on the number of zero-valued diagonal entries of $\mathbf{A}$.

Maximum Diagonal Zeroes Corollary: Let $(\mathbf{A}, \mathbf{B}, \mathbf{C}, d)$ be a stable system of order $n > 1$ with $\mathbf{A} \in D$. Let $p = n - |\mathrm{Tr}\mathbf{A}|$ and z be the number of zero-valued diagonal entries of $\mathbf{A}$. Then $z \leq p$.

Proof: There are two cases to consider: $n = z$ and $n > z$. For the first case, we have n diagonal zeros, which means that $p = n - |\mathrm{Tr}\mathbf{A}| = n - 0 = n = z$.

For the second case, define $\alpha_i, i = 1 \cdots n - z$, to be the index of the ith nonzero diagonal entry of $\mathbf{A}$. From the definition of the trace of a matrix and by applying the triangle inequality, we have

$$|\mathrm{Tr}\mathbf{A}| = \left|\sum_{i=1}^{n-z} \mathbf{A}_{\alpha_i \alpha_i}\right| \leq \sum_{i=1}^{n-z} |\mathbf{A}_{\alpha_i \alpha_i}|.$$

By the diagonal entries theorem, $|\mathbf{A}_{\alpha_i \alpha_i}| < 1$, which bounds the right-hand term by $n - z$. This means $|\mathrm{Tr}\mathbf{A}| < n - z$, and therefore, $z < n - |\mathrm{Tr}\mathbf{A}| \equiv p$. Q.E.D.

III. Conclusion

It has been shown here that the class of state-space digital filters whose $\mathbf{A}$ matrix is diagonally stable (and therefore free of zero-input overflow oscillations) has the following two structural properties:

- The diagonal entries of $\mathbf{A}$ must be bounded in magnitude by 1.
- The number of zero-valued diagonal entries of $\mathbf{A}$ is bounded by $n - |\mathrm{Tr}\mathbf{A}|$.

Acknowledgment

The author would like to especially thank Reviewer #2 whose insightful comments greatly improved the quality of this correspondence.

References

[1] C. Chen, *Linear System Theory and Design*. New York: Holt, 1984.

[2] K. J. Hebel, "An environment for the development of digital signal processing software," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 1989.

[3] E. Kaszkurewicz and A. Bhaya, "Comments on overflow oscillations in state-space digital filters," *IEEE Trans. Circuits, Systems-II: Analog, Digital Signal Processing*, vol. 39, no. 9, pp. 675–676, Sept. 1992.

[4] W. L. Mills, C. T. Mullis, and R. A. Roberts, "Digital filter realizations without overflow oscillations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 334–338, 1978.

Design of Weighted Order Statistic Filters Using the Perceptron Algorithm

Byeongjang Jeong and Yong Hoon Lee

***Abstract*—We observe that the design of optimal weighted order statistic (WOS) filters under the mean absolute error criterion can be thought of as a two-class linear classification problem. Based on this observation, the perceptron algorithm is applied in designing WOS filters. It is shown experimentally that the perceptron algorithm can find optimal or near-optimal WOS filters in practical situations.**

I. Introduction

The *weighted order statistic* (WOS) filter is a nonlinear digital filter effective in suppressing noise superimposed on signals with sharp discontinuities [1]. The class of WOS filters encompassing *weighted median* [1], [2], median, and rank order filters [3] is a subclass of stack filters [4].

In [5] and [6], it is shown that an optimal stack filter can be designed under the mean absolute error (MAE) criterion by using linear programming (LP). Although the WOS filter is a special case of stack filters, it cannot be optimized through LP. In [7]–[9], algorithms for designing WOS filters are developed through some approximations of MAE and mean square error (MSE) criteria. These algorithms can provide WOS filters quite useful for signal restoration, but cannot give optimal WOS filters under the MAE or MSE criterion.

In this correspondence, we observe that the problem of designing optimal WOS filters minimizing the MAE can be thought of as a two-class linear classification problem [10]. Based on this observation, the *perceptron* algorithm is applied in designing WOS filters. This method can find optimal WOS filters under the MAE criterion when the classification problem has linearly separable training samples. Experimental results in image enhancement demonstrate that this new method can give optimal or nearly optimal WOS filters in practical situations.

The organization of this correspondence is as follows: in Section II, we briefly review the definitions and properties of stack and WOS filters and introduce our notation. The proposed design method for WOS filters is described in Section III. Finally, in Section IV, WOS filters are designed using the proposed method and applied to enhance noisy images.

II. Review of Stack and WOS Filters

Let $X(n)$ be an input signal, $Y(n)$ an output signal, and $\underline{X}(n)$ a vector consisting of b samples lying within the window at time index n: $\underline{X}(n) = [X(n - b_1), \ldots, X(n), \ldots, X(n + b_2)]^t \triangleq [X_1(n), \ldots, X_j(n), \ldots, X_b(n)]^t$ where $b = b_1 + b_2 + 1$, $X_j(n) = X(n - b_1 - 1 + j)$ and the superscript t denotes transposition. From now on, the time index n will be dropped from $\underline{X}(n), X_j(n)$, and $Y(n)$ to simplify notation.

Manuscript received May 5, 1993; revised March 25, 1994. The associate editor coordinating the review of this paper and approving it for publication was Prof. Gonzalo Arce.

The authors are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Taejon 305-701, South Korea.

IEEE Log Number 9404544.

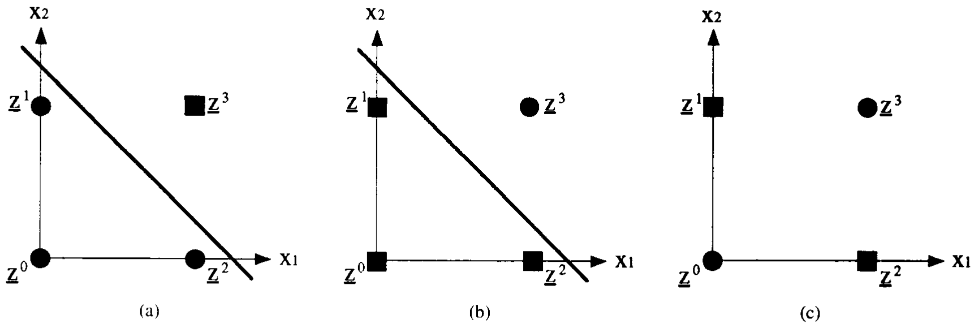


Fig. 1. Possible cases that may occur when the window size $b = 2$. Here, $\underline{z}^0 = [0,0,-1]^t, \underline{z}^1 = [0,1,-1]^t, \underline{z}^2 = [1,0,-1]^t, \underline{z}^3 = [1,1,-1]^t$, and the elements of $\mathcal{Z}^+$ and $\mathcal{Z}^-$ are depicted as • and ■, respectively. (a) Case 1: separable and nonnegative; (b) Case 2: separable and negative; and (c) Case 3: nonseparable.

TABLE I

Weight Vectors of WOS Filters and the Number of Vectors which Cannot Be Separated Linearly by the Vector $\underline{a}$ (For All $\underline{a}, T$ Is Set to 1).

Pe	0.0	0.0125	0.025	0.05	0.1	0.2
Weight vectors	0.060 0.228 0.044 0.048 1.033 0.040 0.095 0.308 0.100	0.459 0.408 0.057 0.003 0.997 0.001 0.006 0.065 0.005	0.081 0.240 0.112 0.058 0.895 0.047 0.113 0.321 0.104	0.124 0.259 0.125 0.124 0.754 0.123 0.123 0.246 0.123	0.173 0.276 0.174 0.102 0.624 0.101 0.101 0.275 0.174	0.180 0.274 0.182 0.181 0.456 0.182 0.182 0.274 0.182
The number of vectors violating linear separability	0	0	0	5	24	38

A. Stack Filters

The class of stack filters encompasses all filters that can be expressed as a composition of local MIN/MAX operations. For example, the filter represented by $Y = \text{MAX}\{\text{MIN}(X_1, X_2), \text{MIN}(X_2, X_3)\}$ is a stack filter. Let $X_i, 1 \le i \le b$, take an integer value from $\{0, 1, \ldots, M-1\}$ and $F(\underline{X})$ denote the output of a stack filter. Then

$$Y = \sum_{m=1}^{M-1} T_m[F(\underline{X})] = \sum_{m=1}^{M-1} F[T_m(X_1), \ldots, T_m(X_b)] \quad (1)$$

where $T_m(X_i)$ is a function which takes the value 1 if $X_i \ge m$ and 0, otherwise. The second equality in (1) comes from the fact that a composite function of MIN/MAX operations commutes with nondecreasing functions. Since $T_m(X_i)$ is binary, $F[T_m(X_1), T_m(X_2), \ldots, T_m(X_b)]$ is also binary and this function is a Boolean function. Moreover, this Boolean function is a positive Boolean function (PBF), because the MIN/MAX operations for binary signals are equivalent to logical AND/OR operations. The class of stack filters encompasses all filters expressed as PBF's in the binary domain.

B. WOS Filters

The output Y of a WOS filter is written as

$$Y = T^{\text{th}}\text{largest}\{\overbrace{X_1, \ldots, X_1}^{W_1\text{times}}, \overbrace{X_2, \ldots, X_2}^{W_2\text{times}}, \ldots, \overbrace{X_b, \ldots, X_b}^{W_b\text{times}}\} \quad (2)$$

where $W_i, 1 \le i \le b$, is a positive integer. Let $X_{(i)}$ be the ith largest $\{X_1, X_2, \cdots, X_b\}$ and $W_{(i)}$ the corresponding weight. Then a necessary and sufficient condition for $X_{(k)}, 1 \le k \le b$, being the output of a WOS filter, is

$$k = \min\left\{ j \,\middle|\, \sum_{i=1}^{j} W_{(i)} \ge T \right\}. \quad (3)$$

WOS filters can be defined by using (3): in such a definition $W_1, W_2, \ldots, W_b$ and T are not necessarily limited to positive integers but can take arbitrary nonnegative real numbers. Using (3), the output $f(\underline{x})$ of a WOS filter for a binary input vector $\underline{x} = [x_1, x_2, \cdot, x_b]^t$ is written as

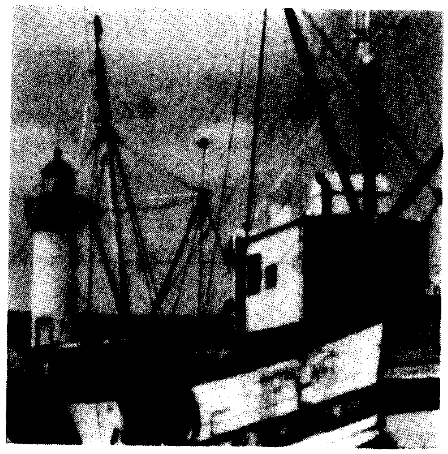
$$f(\underline{x}) = \begin{cases} 1, & \text{if } \sum_{i=1}^{b} W_i x_i \ge T \\ 0, & \text{otherwise} \end{cases}$$
$$\triangleq U(\underline{a}^t \underline{z}) \quad (4)$$

where $\underline{a} = [W_1, W_2, \ldots, W_b, T]^t, \underline{z} = [x_1, x_2, \ldots, x_b, -1]^t$ and $U(\cdot)$ is the unit step function. The function $f(\underline{x})$ in (4) is a special case of Boolean functions, and is called the *threshold* function [11]. A threshold function becomes a PBF if $W_i \ge 0$ and $T \ge 0$. Since WOS filters have nonnegative W_i's and T, they are stack filters.

III. Design of WOS Filters

In this section, an algorithm for designing WOS filters is developed through some modifications of the procedure for designing optimal stack filters.

Suppose that a stack filter $F(\underline{X})$ is used to estimate a desired signal S, where the time index n of the desired signal is dropped. In [5] and [6] it is shown that the MAE between S and $F(\underline{X})$ is given by $\text{MAE} \triangleq E\{|S - F(\underline{X})|\} = \sum_{m=1}^{M-1} E\{|T_m(S) - F[T_m(X_1), T_m(X_2), \ldots, T_m(X_b)]|\}$ and that this equation becomes $\text{MAE} = \sum_{j=0}^{2^b-1} c_j f(\underline{x}^j) + C$ where $f(\underline{x}^j)$is a filter output for the jth



(a)

(b)

Fig. 2. Original image and noisy image: (a) Boat; (b) noisy image when $P_e = 0.1$.

binary input vector $\underline{x}^j$, and c_j's and C are constants determined by

$$c_j = \sum_{m=1}^{M-1} \{\text{Prob}[s_m = 0, \underline{x}_m = \underline{x}^j \quad \text{at level} m] - \text{Prob}[s_m = 1, \underline{x}_m = \underline{x}^j \quad \text{at level} m]\}.$$

$$C = \sum_{j=0}^{2^b-1} \sum_{m=1}^{M-1} \text{Prob}\{s_m = 1, \underline{x}_m = \underline{x}^j \quad \text{at level} m\}. \tag{5}$$

Here, $s_m = 1$ if $S \geq m$ and 0, otherwise, $\underline{x}_m$ is the binary input vector at level m and $\{\underline{x}^j \mid 0 \leq j \leq 2^b - 1\}$ is the set of all possible binary vectors of dimension b. Now, the optimal stack filter can be obtained through the following optimization:

Find $f(\cdot)$ minimizing

$$J(f) = \sum_{j=0}^{2^b-1} c_j f(\underline{x}^j) \tag{6a}$$

subject to the constraints

$$f(\underline{x}^j) \geq f(\underline{x}^i) \text{ if } \underline{x}^j \geq \underline{x}^i. \tag{6b}$$

TABLE II
MAE'S BETWEEN THE ORIGINAL AND FILTERED IMAGES

Pe	MAE			
	Median filter	Optimal stack filters	LMA WOS filters	WOS filters designed by proposed algorithm
0.0	3.1962	0.0000	0.0000	0.0000
0.0125	3.2549	0.4090	0.4293	0.4090
0.025	3.3314	0.6458	0.6783	0.6458
0.05	3.4742	1.0435	1.0756	1.0443
0.1	3.7833	1.8878	1.9198	1.9120
0.2	4.5890	3.4784	3.3679	3.3674

Here, the inequality consists of the stacking constraints restricting $f(\cdot)$ to a PBF. This optimization problem can be solved by LP, but the complexity of LP increases exponentially as b increases.

From (4) and (6) the optimization of WOS filters is described as follows:

Find a vector $\underline{a}$ minimizing

$$J(\underline{a}) = \sum_{j=0}^{2^b-1} c_j U(\underline{a}^t \underline{z}^j) \tag{7a}$$

subject to the constraints

$$W_i \geq 0 \quad \text{for all } i, \quad 1 \leq i \leq b$$
$$T \geq 0. \tag{7b}$$

Due to the nonlinear function $U(\cdot)$, the optimization in (7) cannot be solved through LP. Next we shall see that the problem in (7) can be thought of as a two-class linear classification problem. The objective function $J(\underline{a})$ in (7) is minimized if $U(\cdot)$ takes the value $1(0)$ whenever c_j is negative(positive). That is, $J(\underline{a})$ is minimized if

$$U(\underline{a}^t \underline{z}^j) = \begin{cases} 1, & \text{if } c_j < 0 \\ 0, & \text{if } c_j > 0 \end{cases}. \tag{8}$$

Therefore, if a vector $\underline{a}$ satisfying (8) exists, the optimization in (7) is equivalent to the following:

Find a vector $\underline{a}$ such that

$$\begin{cases} \underline{a}^t \underline{z}^j \geq 0 & \text{if } c_j < 0 \\ \underline{a}^t \underline{z}^j < 0 & \text{if } c_j > 0 \end{cases} \tag{9}$$

under the constriants in (7b).

The problem in (9) is essentially a two-class linear classification problem. To be more precise, define $\mathcal{Z}^+ \triangleq \{\underline{z}^j \mid c_j > 0\}$ and $\mathcal{Z}^- \triangleq \{\underline{z}^j \mid c_j < 0\}$. Here, the vectors with $c_j = 0$ are neglected. Finding a vector $\underline{a}$ satisfying (8) is equivalent to obtaining a linear discriminant function separating $\mathcal{Z}^+$ and $\mathcal{Z}^-$. If $\mathcal{Z}^+$ and $\mathcal{Z}^-$ are linearly separable, then a solution vector for (9) exists and can be obtained by the *perceptron* algorithm [10]. In this case, if the solution vector satisfies the nonnegativity constraints in (7b), it specifies an optimal WOS filter minimizing the MAE; the resulting WOS filter is equivalent to an optimal stack filter.

When either nonnegativity constraints are violated or $\mathcal{Z}^+$ and $\mathcal{Z}^-$ are not linearly separable, the optimization in (7) cannot be solved through (9). In many practical situations, however, $\mathcal{Z}^+$ and $\mathcal{Z}^-$ tend to be linearly separable and the solution vector separating them tends to have nonnegative entries. The following observation addresses this issue.

Observation 1: If an input signal X is equal to the desired signal S, there exists a vector $\underline{a}$ with nonnegative elements separating $\mathcal{Z}^+$ and $\mathcal{Z}^-$.

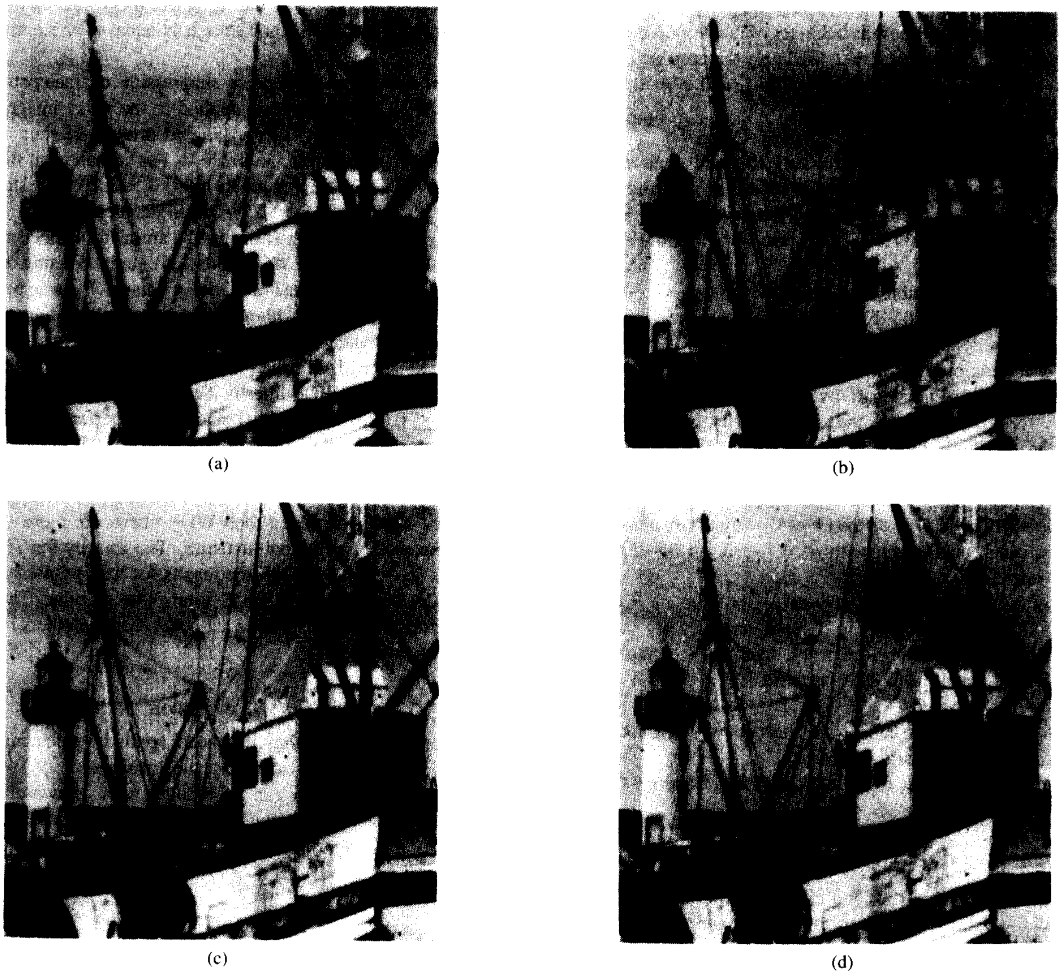


Fig. 3. Images for $P_e = 0.1$: (a) Median filtered image; (b) optimal stack filtered image; (c) LMA WOS filtered image; (d) proposed WOS filtered image.

Proof: If $X = S$, then $x_m(b_1+1) = s_m$ for any binary vector $\underline{x}_m = [x_m(1), \ldots, x_m(b_1+1), \ldots, x_m(b)]^t$. Therefore for $\underline{x}^j = [x^j(1), x^j(2), \ldots, x^j(b)]^t$ with $x^j(b_1+1) = 1$, $\text{Prob}\{s_m = 0, \underline{x}_m = \underline{x}^j \textit{ at level } m\} = 0$. Hence we have $c_j = -\sum_{m=1}^{M-1} \text{Prob}\{s_m = 1, \underline{x}_m = \underline{x}^j \textit{ at level } m\} \le 0$ for all j for which $x^j(b_1+1) = 1$. Similarly we can show that $c_j \ge 0$ for all j for which $x^j(b_1+1) = 0$. Therefore, $\mathcal{Z}^- = \{\underline{z}^j \mid c_j < 0\} \subset \{\underline{z}^j \mid x^j(b_1+1) = 1\}$ and $\mathcal{Z}^+ = \{\underline{z}^j \mid c_j > 0\} \subset \{\underline{z}^j \mid x^j(b_1+1) = 0\}$. A vector separating these two sets always exists. For instance $\mathcal{Z}^+$ and $\mathcal{Z}^-$ are linearly separated by $\underline{a} = [0, \ldots, 0, 1, 0, \ldots, 0, 1]^t$, which corresponds to an identity filter. Note that this $\underline{a}$ has nonnegative entries. □

From this observation, it is deduced that $\mathcal{Z}^+$ and $\mathcal{Z}^-$ tend to become linearly separable and to have nonnegative entries as the observation X closes with the desired signal S. This is verified through the computer simulation in Section IV.

Now we will describe the proposed algorithm for designing WOS filters. This algorithm is the same as the standard perceptron algorithm except for the constraints in (7b). The perceptron criterion function that will be used is $J_p(\underline{a}) = \sum_{\underline{z}^j \in \mathcal{Z}_m} c_j \underline{a}^t \underline{z}^j$ where $\mathcal{Z}_m = \{\underline{z}^j \mid c_j \underline{a}^t \underline{z}^j \ge 0\}$ is the set of misclassified vectors.

Proposed Algorithm Let $\underline{a}(k)$ be the weight vector at the k^{th} iteration.

Step 1: Set $\underline{a}(1)$ to an arbitrary nonnegative $(b+1) \times 1$ vector and $k = 1$.

Step 2: Replace $\underline{a}(k)$ with

$$\underline{a}(k+1) = \underline{a}(k) - \rho_k \sum_{\underline{z}^j \in \mathcal{Z}_m(\underline{a}(k))} c_j \underline{z}^j$$

where $\mathcal{Z}_m(\underline{a}(k)) = \{\underline{z}^j \mid c_j \underline{a}^t(k) \underline{z}^j \ge 0\}$ and ρ_k is a sequence satisfying the limit conditions in [10, p. 146].

Step 3: Go to Step 4 if $\sum_{i=1}^{b+1} |a_i(k+1) - a_i(k)| < \epsilon$. Otherwise, go to Step 2 after increasing k by 1. Here ϵ is a positive number which is sufficiently small, and $a_i(k)$ is the ith element of $\underline{a}(k)$.

Step 4: If $a_i(k+1) < 0$, then replace $a_i(k+1)$ with zero (0) for all i.

Step 5: Stop.

This algorithm is identical to the perceptron algorithm classifying the vectors $\{c_j \underline{z}^j\}$, with the exception of Step 4. Note that the nonnegativity constraint in Step 4 is applied at the end of the algorithm instead of being applied at each iteration. When there exists a solution vector with negative elements, both approaches to enforcing the nonnegativity will eventually yield 0's instead of negative values, but the convergence speed of the latter should be considerably slower than that of the former. (In our simulation, the nonnegativity constraint was redundant because all the weights and threshold values that appeared at the end of Step 2 were nonnegative.) The example below illustrates the behavior of the proposed algorithm.

Example 1: Suppose that $b = 2$. We apply the proposed algorithm with $\underline{a}(1) = [0, 0, 0]'$, $\rho_k = 1/(1 + 0.1k)$, and $\epsilon = 0.01$ to the following three cases.

Case 1 (linearly separable, nonnegative): Given $\{c_0 = 1.0, c_1 = 0.5, c_2 = 0.5, c_3 = -1.0\}$. $\underline{z}^j$s for this case are depicted in Fig. 1(a). After 6 iterations, the algorithm separates $\mathcal{Z}^+$ and $\mathcal{Z}^-$ and $\underline{a}(6) = [1.5, 1.5, 2.0]'$, consisting of nonnegative elements. The parameters of the optimal WOS filter are $W_1 = 1.5, W_2 = 1.5$, and $T = 2.0$.

Case 2 (linearly separable, negative): Given $\{c_0 = -1.0, c_1 = -0.5, c_2 = -0.5, c_3 = 1.0\}$. Note that these c_js are minus the c_js in Case 1. Thus, the solution vector separating $\mathcal{Z}^+$ and $\mathcal{Z}^-$ is again obtained after 6 iterations, but in this case $\underline{a}(6) = [-1.5, -1.5, -2.0]'$ completely violates the nonnegativity constraints; the algorithm yields zero($\underline{0}$) vector. If we enforce the nonnegativity constraint at each iteration, the algorithm stops after 490 iterations yielding $\underline{a}(490) = \underline{0}$. It should be pointed out that WOS filtering is inadequate this case. An extension of the WOS filter, called the *linearly separable threshold Boolean filter* [12] which is defined by (4) without the nonnegativity constraints, can be optimized with $\underline{a} = [-1.5, -1.5, -2.0]'$ under the MAE criterion.

Case 3 (not linearly separable): Given $\{c_0 = -0.5, c_1 = 0.5, c_2 = 0.5, c_3 = -1.0\}$. As seen from Fig. 1(c), $\mathcal{Z}^+$ and $\mathcal{Z}^-$ are not linearly separable. The algorithm stops after 14 iterations yielding $\underline{a}(14) = [0.27, 0.27, 0.46]'$ and the WOS filter with $W_1 = 0.27, W_2 = 0.27$, and $T = 0.46$ is obtained. □

Before concluding this section, we shall show that Case 2 in the above example rarely occurs in practice.

Observation 2: Consider two binary vectors $\underline{x}^i$ and $\underline{x}^j$ for which $x^j(k) \geq x^i(k)$ for all $1 \leq k \leq b$. If

$$\text{Prob}(s_m = 1 \mid \underline{x}_m = \underline{x}^j) \geq \text{Prob}(s_m = 1 \mid \underline{x}_m = \underline{x}^i) \qquad (10)$$

then $c_j < 0$ whenever $c_i < 0$, and $c_i > 0$ whenever $c_j > 0$.

The proof of this observation is simple, and is omitted. The condition in (10) is called the *stacking* property of the conditional probabilities [13], which holds in many practical situations. Note that the c_js in Case 2 of Example 1 violates this property in Observation 2.

IV. Application to Image Enhancement

We consider the design of a 2-D 3×3 WOS filter for enhancing a noisy image corrupted by additive impulses. The original image shown in Fig. 2(a) is the 256×256 "boat" image. The image is corrupted by positive and negative impulses with values of ± 200, respectively. Following the approach in [7]–[9], we assume that the upper left quarter of the original and the noisy images are given; c_js in (5) are estimated from the given data. Suppose that $N_{m,0}^j(N_{m,1}^j)$ is the number of times that binary vector $\underline{x}^j$ is observed in the noisy image and at the same time, $s = 0(1)$ in the original image on level m. Then, $\hat{c}_j = \sum_{m=1}^{M-1}(N_{m,0}^j - N_{m,1}^j)/(I \times J)$ where $I \times J$ is the size of the upper left quarter ($I \times J = 128 \times 128$ in this experiment) and $M = 256$. Using the perceptron algorithm with $\epsilon = 10^{-5}$, $\rho_k = 1/(1 + 0.1k)$ and $\underline{a}(1)$ which is equal to the zero vector, we obtained the vectors $\underline{a}$ and counted the number of binary vectors that cannot be separated linearly by vector $\underline{a}$ while changing the probability of occurrence of impulses, P_ϵ. The results are summarized in Table I. When $P_\epsilon = 0$, the resulting WOS filter is the identity filter, and the WOS filters approach the median filter as P_ϵ increases. It is interesting to note that the number of vectors which are not linearly separable increases as P_ϵ increases. In this experiment, for $P_\epsilon \leq 0.025$ the designed WOS filter is the optimal one minimizing the MAE, and thus it is equal to the optimal stack filter.

In order to examine the noise suppression characteristics of the designed WOS filters, they are applied to the noisy images and the MAE's between the original and filtered images are thus evaluated. The results are summarized in Table II. For comparison, the MAE's associated with the 3×3 median filter, 3×3 optimal stack filters designed by LP, and 3×3 WOS filters designed by the *least mean absolute* (LMA) algorithm in [8] are also obtained and listed in Table II. (The LMA WOS filters are trained using the upper left quarter of the original and noisy images.) As expected, the designed WOS and stack filters outperform the median filter. The WOS filters designed by the proposed algorithm consistently yield somewhat smaller MAE's than the LMA WOS filters. As mentioned above, the proposed WOS filter is identical to the optimal stack filter when $P_\epsilon \leq 0.025$. The optimal stack filters perform slightly better than the proposed WOS filters when $P_\epsilon = 0.05$ and 0.1, but the former yields a larger MAE than the latter when $P_\epsilon = 0.2$. One possible explanation for this is that stack filters, which have more parameters to be determined than WOS filters, are more vulnerable to the inaccuracies of signal statistics. The images for $P_\epsilon = 0.05$ are shown in Fig. 3. Visually, the results for the optimal stack filter, LMA WOS, and proposed WOS filters look similar. The median filter suppressed impulses somewhat better than the others but caused severe blurring.

V. Conclusion

The perceptron algorithm has been applied in designing WOS filters based on the observation that the optimization of WOS filters under the MAE criterion can be thought of as a two-class linear classification problem. Through computer experiments, it has been shown that the perceptron algorithm can find optimal or near-optimal WOS filters.

References

[1] O. Yli-Harja, J. Astola, and Y. Neuvo, "Analysis of the properties of median and weighted median filters using threshold logic and stack filter representation," *IEEE Trans. Signal Processing*, vol. 39, pp. 395–410, Feb. 1991.

[2] M. K. Prasad and Y. H. Lee, "Analysis of weighted median filters based on inequalities relating the weights," *Circuits, Syst., Signal Processing*, vol. 11, no. 1, pp. 115–136, Jan. 1992.

[3] T. K. Nodes and N. C. Gallagher Jr., "Median filters: Some modifications and their properties," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 739–746, Oct. 1982.

[4] P. D. Wendt, E. J. Coyle, and N. C. Gallagher Jr., "Stack filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 898–911, Aug. 1986.

[5] E. J. Coyle and J. H. Lin, "Stack filters and the mean absolute error criterion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1244–1254, Aug. 1988.

[6] E. J. Coyle, J. H. Lin, and M. Gabbouj, "Optimal stack filtering and structural approaches to image processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 2037–2066, Dec. 1989.

[7] L. Yin, J. Astola, and Y. Neuvo, "Adaptive stack filtering with application to image processing," *IEEE Trans. Signal Processing*, vol. 41, pp. 162–184, Jan. 1993.

[8] ——, "A New Class of Nonlinear Filters-Neural Filters," *IEEE Trans. Signal Processing*, vol. 41, pp. 1201–1222, Mar. 1993.

[9] L. Yin and Y. Neuvo, "Fast adaptation and performance characteristics of FIR-WOS hybrid filters," to appear in *IEEE Trans. Signal Processing*.

[10] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

[11] S. Muroga, *Threshold Logic and Its Applications*. New York: Wiley, 1971..

[12] K. D. Lee and Y. H. Lee, "Threshold Boolean Filters," to appear in *IEEE Trans. Signal Processing*.

[13] B. Zeng, M. Gabbouj, and Y. Neuvo, "A unified design methods for rank order, stack and generalized stack filters based on classical Bayes decision," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1003–1020, Sept. 1991.

Representation of Discrete Sequences with Three-Dimensional Iterated Function Systems

David S. Mazel

***Abstract*— Several recent papers have shown the ability of iterated function system (IFS) models to represent discrete sequences where the attractor of an IFS is self-affine either in R^2 or R^3 or is piece-wise self-affine in R^2. In this correspondence, we extend the piece-wise self-affine IFS model from R^2 to R^3 and we call this model the piece-wise hidden-variable fractal model. This extension permits the representation of discrete sequences with a fractal model that is general and flexible. This new model is presented and a constrained inverse algorithm for identification of the model parameters is given. Examples of this model to seismic data and speech data are then presented.**

I. Introduction

Three recent papers have introduced fractal interpolation of discrete sequences with an Iterated Function System (IFS) where the IFS operated in R^2 for both the self-affine fractal model and for the piece-wise self-affine fractal model [1]–[3]. In fact, the self-affine fractal model is a special case of the piece-wise self-affine fractal model. It was found that the piece-wise self-affine fractal model was more flexible and capable of representing a wider variety of data types than the self-affine fractal model.

One attempt, however, to use a self-affine fractal model that would be capable of representing a wide variety of data types led to hidden-variable fractal interpolation [4], [5] where the model itself operates in R^3, yet it produces a single-valued function in the $x \times y$-plane and is then able to represent discrete sequences [6]. The ability of the hidden-variable model to operate in R^3 made it flexible since the resulting attractor of the IFS was 3–D and self-affine, yet the projection of that attractor to the $x \times y$-plane need not be self-affine.

In this correspondence, we extend the hidden-variable fractal model so that the resulting attractor of our new model is piece-wise self-affine in R^3; we call this model the piece-wise hidden-variable fractal model. This extension permits the representation of discrete sequences with a fractal model that is general and flexible. In addition, we are able to use the algorithms developed for the previous fractal models for this new model. Thus many of the ideas and properties presented in earlier papers are extendible to this new model.

This correspondence is organized as follows. In Section II we present the piece-wise hidden-variable fractal model and use it to produce single-valued functions in R^2. Then, in Section III, we present an inverse algorithm that allows use of the model to represent given data. Examples of this model to seismic data and speech data are given in Section IV and conclusions are presented in Section V.

Manuscript received May 26, 1993; revised April 21, 1994. The associate editor coordinating the review of this paper and approving it for publication was Dr. Thomas F. Quatieri, Jr.

The author was with the IBM Federal Systems Company, Manassas, VA 22110 USA. He is now with the CNA Corporation, Alexandria, VA 22302.

IEEE Log Number 9404781.

II. Background

In this section, we present the piece-wise hidden-variable fractal model. We begin with the IFS and its associated maps and then we show how to synthesize data with the model.

The model is composed of three sets of parameters. The first set consists of $M+1$-interpolation points in the space $\mathbf{X} = \mathcal{N} \times R \times R$, where $\mathcal{N}$ is the set of nonnegative integers and R is the set of real numbers. Since we are concerned with discrete sequences in this paper, we use $\mathcal{N}$ as a dimension of $\mathbf{X}$ to represent the sample indices of our sequences. In general, however, this restriction is not required as the IFS produces continuous functions and we could have $\mathbf{X} = R^3$. The second set of parameters consists of the affine maps that comprise the IFS: $\{\mathbf{X}; \mathbf{w}_i, i = 1, \dots, M\}$, where each affine map is given by

$$\mathbf{w}_i \begin{pmatrix} n \\ x[n] \\ y[n] \end{pmatrix} = \begin{pmatrix} a_i & 0 & 0 \\ c_i & d_i & h_i \\ k_i & l_i & m_i \end{pmatrix} \begin{pmatrix} n \\ x[n] \\ y[n] \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ g_i \end{pmatrix}; \quad i = 1, 2, \dots, M. \tag{1}$$

There are M-maps for the $M+1$ interpolation points and the interpolation points are required to be single-valued with respect to their first index.

The third set of parameters are address points assigned to each consecutive pair of interpolation points. These address points are used to constrain the affine maps so that

$$\mathbf{w}_i \begin{pmatrix} p' \\ x[p'] \\ y[p'] \end{pmatrix} = \begin{pmatrix} p \\ x[p] \\ y[p] \end{pmatrix} \text{ and } \mathbf{w}_i \begin{pmatrix} q' \\ x[q'] \\ y[q'] \end{pmatrix} = \begin{pmatrix} q \\ x[q] \\ y[q] \end{pmatrix}; \quad i = 1, 2, \dots, M \tag{2}$$

where $(p, x[p], y[p])$ and $(q, x[q], y[q])$ are two consecutive interpolation points with $q > p$, and $(p', x[p'], y[p'])$ and $(q', x[q'], y[q'])$ are the address points associated with that pair of interpolation points with $q' > p'$ and $q' - p' > q - p$. All the address points must lie within the interval of support of the interpolation points along $\mathcal{N}$.

Lastly, each map in (1) has the restriction that the contraction matrix, given by

$$\begin{bmatrix} d_i & h_i \\ l_i & m_i \end{bmatrix}$$

must have all eigenvalues of modulus less than unity. Once the interpolation points and address points are chosen, the free parameters of each affine map are the elements of the contraction matrix. We show how to choose those elements in Section III-A. Once the contraction matrices are chosen, the other affine map parameters are found with (2). Synthesis of the piece-wise hidden-variable fractal interpolation function is given in [7].

III. An Inverse Algorithm

In this section we describe an inverse algorithm to identify the parameters associated with the piece-wise hidden-variable fractal model. We begin with the equations necessary to solve for computation of the contraction matrix associated with a map when the address points and interpolation points for the map are assumed known. Then, we present an algorithm which identifies all the map parameters so that piece-wise hidden-variable fractal interpolation may be applied to given data.