A Unified Design Method for Rank Order, Stack, and Generalized Stack Filters Based on Classical Bayes Decision

Article *in* IEEE Transactions on Circuits and Systems · October 1991

DOI: 10.1109/31.83872 · Source: IEEE Xplore

CITATIONS
31

READS
34

3 authors:

Bing Zeng
University of Electronic Science and Technology of China
309 PUBLICATIONS **6,438** CITATIONS
SEE PROFILE

Moncef Gabbouj
Tampere University
1,029 PUBLICATIONS **28,699** CITATIONS
SEE PROFILE

Y. Neuvo
Aalto University
314 PUBLICATIONS **8,950** CITATIONS
SEE PROFILE

A Unified Design Method for Rank Order, Stack, and Generalized Stack Filters Based on Classical Bayes Decision

Bing Zeng, Moncef Gabbouj, *Member, IEEE*, and Yrjö Neuvo, *Fellow, IEEE*

Abstract—**This paper presents a unified method for designing optimal rank order filters (ROF's), stack filters, and generalized stack filters (GSF's) under the mean absolute error (MAE) criterion. The method is based on classical Bayes minimum-cost decision. Both the *a priori* and the *a posteriori* approaches are considered. It is shown that designing the minimum MAE stack filters and GSF's is equivalent to the *a priori* Bayes decision. To solve the problem, linear programming (LP), whose complexity increases faster than exponentially as a function of the filter window width, is required. This renders the use of this approach extremely impractical. In this paper, we shall develop suboptimal routines to avoid the huge complexity of the LP for designing stack filters and GSF's. It is shown that the only required computations then reduce to data comparisons exclusively, and the number of comparisons needed increases only exponentially (for GSF's) or even slower than exponentially (for stack filters) as a function of the filter's window width. The most important feature of the design routines is perhaps the fact that, for most practical cases, they yield optimal solutions under the MAE criterion.**

When the *a posteriori* approach is employed, it is shown that the optimal solutions become ROF's with appropriately chosen orders that do not depend on the prior probability model of the input process. Moreover, it is shown that the *a posteriori* Bayes minimum-cost decision reduces to the median filter in frequent practical applications.

The filters produced by the *a priori* and the *a posteriori* approaches are subjected to a sensitivity analysis to quantify their dependency upon the cost coefficients. Several design examples of ROF's, stack filters, and GSF's will be provided, and an application of stack filters and GSF's to image recovery from impulsive noise will be considered.

I. Introduction

STACK filters are nonlinear digital filters that were developed as an alternative to linear filters when the latter failed to accomplish certain tasks in many applications in signal and image processing [1]–[6]. These filters are based on two defining properties: the threshold decomposition and the stacking property [7], [8]. The former is a weak superposition property and the latter is an ordering property. Since these are the defining properties of stack filters, they will be discussed in more detail in the next section.

This large class of digital filters includes all rank order filters (ROF's); all compositions of ROF's; all compositions of morphological filters composed of opening and closing operations [9], [10]; Ξ filters [11]; and, allowing randomization of outputs and repetition of samples in the window, FIR filters with non-negative weights and order statistic filters in which the linear section has non-negative weights [12]. Generalized stack filters (GSF's) are an extension of stack filters and include all finite window width digital operations [13].

Some techniques were needed to select one filter among this large class of filters that is best suited for the application at hand. The theory of minimum mean absolute error (MAE) estimation over the class of stack filters and GSF's [2]–[5], [13] was developed for this purpose. Adaptive stack filtering under the MAE criterion was later developed [6]. Another error criterion, the minimax, was used for selecting the best filter [14]; however, the current paper deals exclusively with the MAE criterion.

In [2], it was shown that it is possible to determine the stack filter that minimizes the MAE between its output and a desired signal, given noisy observations of this desired signal. Specifically, it was shown that an optimal window width b stack filter can be determined using a linear program (LP) with $O(b2^b)$ variables and constraints. This algorithm was claimed to be efficient since the number of different inputs to a window width b stack filter is M^b if the filter's input signal is M-valued and since the number of stack filters grows faster than $2^{2^{b/2}}$ [2]. In [13], it was shown that the optimal GSF under the MAE criterion can be found via an LP with $O((2I+3)^b)$ variables and constraints for the homogeneous case and $O(M(2I+3)^b)$ for the inhomogeneous case (M is as before and $2I+1$ threshold levels are fed into the Boolean function at each level). Now, for a "small" window of width seven, $I=1$, and eight-bit samples, the LP that finds the best GSF in minimum MAE sense has 20 million variables! Therefore, finding an optimal GSF with a 3×3 window size (considered to be a small mask size for image processing) using this approach is quite impossible at the present time. Now, the only alternative left is

Manuscript received July 31, 1990. This paper was recommended by Associate Editor E. J. Coyle.

The authors are with the Signal Processing Laboratory, Tampere University of Technology, SF-33101 Tampere, Finland.

IEEE Log Number 9101485.

0098-4094/91/0900-1003$01.00 ©1991 IEEE

to use the adaptive algorithm [6] assuming that training sequences exist. However, there are two problems associated with this approach: (1) training sequences may not be available; and (2) even if such sequences were available, the algorithm in question may take an extremely long time to produce an optimal filter.

What is lacking here is an "acceptable" suboptimal approach that would considerably reduce the complexity of the required LP and thus allow larger optimization problems to be solved. Furthermore, it would be quite an improvement if such a suboptimal approach would yield optimal solutions under some "reasonable" conditions.

This paper is an attempt in this direction. First, we shall develop a suboptimal routine which requires no LP for finding "reasonably good" filters. Secondly, we shall determine sufficient conditions (not too restrictive, hopefully) under which the suboptimal routine produces optimal solutions. The above procedure is developed separately for ROF's, stack filters, and GSF's.

This paper is organized as follows. In Section II, we briefly review ROF's, stack filters, and GSF's based on the two fundamental properties: the threshold decomposition and the stacking property. The optimal theory developed earlier for stack filters is reviewed in Section III, where it is given an equivalent interpretation in the context of the classical *a priori* Bayes minimum-cost decision. Based on this interpretation, we shall develop, in Sections IV and V, suboptimal algorithms for designing stack filters and GSF's, respectively. Conditions under which these suboptimal algorithms result in optimal filters are also discussed in these sections. We will show that these conditions are natural conditions that hold in many practical applications.

Section VI gives the optimality theory a new meaning in terms of the *a posteriori* Bayes minimum-cost decision. It will be shown that this always leads to an ROF as the optimal filter.

Variations of the solutions, produced by the above algorithms, upon the associated cost coefficients are investigated in Section VII. This analysis shows, among other things, the robustness of the solutions produced by these algorithms.

In Section VIII, some design examples for stack filters, GSF's, and ROF's will be presented, and the application of stack filters and GSF's to image recovery from impulsive noise will be considered. Section IX gives some conclusions.

Finally, the major result in this paper shows how the complexity of finding optimal stack filters and GSF's can be reduced, thus allowing larger, i.e., more interesting, optimization problems to be solved.

II. ROF's, Stack Filters, and GSF's

Suppose that we have a discrete-time input space Ω whose elements $X(n)$ (n is a time index) take values in the set $\{0,1,\cdots,M\}$. Based on Ω, we construct another space called the *input window* space Ω_b (b is the window width) by concatenating b successive samples in the input space Ω. The elements in Ω_b are called (input) window process states and denoted by $\vec{X}(n) = \{X(n-n_1),\cdots,X(n),\cdots,X(n+n_2)\}$ where $n_1+n_2+1=b$.

With this notation, the rth window width b ROF [15], $R_{b,r}(\cdot)$: $\Omega_b \to \Omega$, can be defined as

$$Y(n) = R_{b,r}\left(\vec{X}(n)\right) = \text{the } r\text{th largest sample in } \vec{X}(n) \tag{2.1}$$

for $r=1,2,\cdots,b$. The following are several special cases: $r=1$ corresponds to the max filter, $r=b$ is the min filter, and $r=(b+1)/2$ (for odd b) is the median filter.

Next, we shall invoke the weak superposition property, the threshold decomposition [7], [8], in order to define stack filters and GSF's.

Definition 2.1: Threshold decomposing a window process $\vec{X}(n)$ produces a set of M binary sequences, called threshold signals, $\vec{x}_1(n),\vec{x}_2(n),\cdots,\vec{x}_M(n)$ whose elements are defined by

$$x_l(k) = T_l(X(k)) = \begin{cases} 1, & \text{if } X(k) \geqslant l \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

for $l=1,2,\cdots,M$.

Consider two sequences (of the same length) $\vec{u}$ and $\vec{v}$. A property called the *stacking property* holds between $\vec{u}$ and $\vec{v}$ if and only if $u(k) \geqslant v(k)$ ($v(k) \geqslant u(k)$) for all k; $\vec{v}(\vec{u})$ is said to stack on top of $\vec{u}(\vec{v})$, denoted as $\vec{v} \leqslant \vec{u}$ ($\vec{u} \leqslant \vec{v}$). Otherwise, we say that $\vec{u}$ and $\vec{v}$ are *incomparable*. In the following, we will first define what it means for a Boolean function to possess the stacking property and then give the formal definition of stack filters.

Definition 2.2: A Boolean function $f(\cdot)$ operating on binary sequences is said to possess the stacking property if

$$f(\vec{v}) \leqslant f(\vec{u}) \text{ whenever } \vec{v} \leqslant \vec{u}. \tag{2.3}$$

It has been shown [16], [17] that a necessary and sufficient condition for a Boolean function to satisfy the stacking property is that the Boolean function be *positive*, i.e., no complementation on any of the input variables in the minimum-sum-of-products (MSP) form of the function.

Definition 2.3: A window width b stack filter $S_f(\cdot)$: $\Omega_b \to \Omega$ is based on a b-variable positive Boolean function $f(\cdot)$: $\{0,1\}^b \to \{0,1\}$ operating on the binary threshold signals. The output of the stack filter is obtained by adding all the binary outputs:

$$S_f\left(\vec{X}(n)\right) = S_f\left(\sum_{l=1}^{M} T_l\left(\vec{X}(n)\right)\right) = \sum_{l=1}^{M} S_f\left(T_l\left(\vec{X}(n)\right)\right)$$

$$= \sum_{l=1}^{M} f\left(T_l\left(\vec{X}(n)\right)\right) = \sum_{l=1}^{M} f\left(\vec{x}_l(n)\right). \tag{2.4}$$

Fig. 1 illustrates how the threshold decomposition and the stacking property can be used to filter a 4-valued signal by a window width three asymmetric median filter (which is also a stack filter).

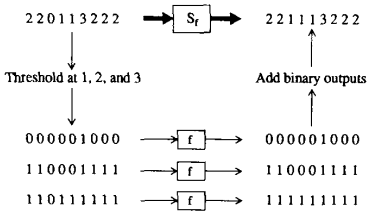


Fig. 1. An example showing how threshold decomposition and stack filtering are performed on a 4-valued signal by a window width three asymmetric median filter (which is also a stack filter). The operations performed by the stack filter and the Boolean filter shown above are $S_f = \max(\min(X_1, X_3), X_2)$ and $f = x_2 + x_1 x_3$, respectively, in which X_i (x_i), $i = 1,2,3$ are the samples (bits) appearing in the filter's window.

In [13], a generalized form of stack filters, called generalized stack filters (GSF's) was defined. GSF's operate on some binary threshold arrays involving several threshold signals. The following notation is used to express these arrays (with dimensions $(2I+1)\times b$):

$$\vec{x}_l^I(n) = \begin{pmatrix} T_{l+I}(\vec{X}(n)) \\ \vdots \\ T_l(\vec{X}(n)) \\ \vdots \\ T_{l-I}(\vec{X}(n)) \end{pmatrix} = \begin{pmatrix} \vec{x}_{l+I}(n) \\ \vdots \\ \vec{x}_l(n) \\ \vdots \\ \vec{x}_{l-I}(n) \end{pmatrix} \tag{2.5}$$

where I is an integer and $l = 1,2,\cdots,M$. To define the above expression at the boundary levels, we have to define

$$\vec{x}_l(n) = \begin{cases} \vec{0}, & \text{if } l > M \\ \vec{1}, & \text{if } l < 1 \end{cases} \tag{2.6}$$

where $\vec{0}$ and $\vec{1}$ are row vectors of length b in which every entry is 0 and 1, respectively. From (2.5) it is easy to show that the rows of any binary threshold array obey the *internal* stacking property, i.e.,

$$\vec{x}_{l+I}(n) \leqslant \cdots \leqslant \vec{x}_l(n) \leqslant \cdots \leqslant \vec{x}_{l-I}(n). \tag{2.7}$$

Now, let us denote by B_b^I the set of all $(2I+1)\times b$ binary arrays possessing the internal stacking property and, for each l, by $f_l(\cdot)$: $B_b^I \to \{0,1\}$ the Boolean function on level l. In the following, we first define a stacking set of Boolean functions and then give the definition of GSF's [13].

Definition 2.4: A set of M Boolean functions $f_1(\cdot), f_2(\cdot), \cdots, f_M(\cdot)$, each operating on B_b^I, is said to possess the stacking property if

$$f_{l+1}(\vec{x}_{l+1}^I(n)) \leqslant f_l(\vec{x}_l^I(n)), \qquad l = 1,2,\cdots,M-1. \tag{2.8}$$

Definition 2.5: A window width b GSF $F_{I,b}(\cdot)$: $\Omega_b \to \Omega$ is based on a stacking set of M Boolean functions $f_l(\cdot)$: $B_b^I \to \{0,1\}$, $l = 1,2,\cdots,M$. The output of the GSF is expressed as

$$F_{I,b}(\vec{X}(n)) = \sum_{l=1}^{M} f_l(\vec{x}_l^I(n)). \tag{2.9}$$

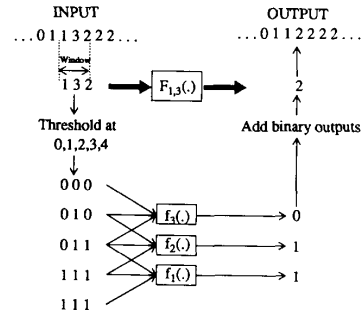


Fig. 2. A GSF $F_{I,b}(\cdot)$ with window width $b = 3$ and with $2I+1 = 3$ binary threshold signals fed into each Boolean filter in the threshold decomposition architecture.

An example of a GSF with $b = 3$ and $I = 1$ is shown in Fig. 2. It is worth noting that, in GSF's, Boolean functions are allowed to be different at different threshold levels and each Boolean function operates on possibly more than one threshold signal. Therefore, the Boolean functions in a GSF are not necessarily positive; see [13] for a detailed discussion.

III. Design of Minimum MAE Stack Filters and Classical Bayes Decision

The theory of optimal stack and generalized stack filtering under the MAE criterion [2]–[5], [13] allows us to find the best filter in the class of stack filters or GSF's. This is of great importance, since the number of stack filters increases double-exponentially in the filter's window width, while the number of GSF's is even much larger than that of stack filters. In this section, we review this optimality theory in a new equivalent context of Bayes minimum-cost decision.

Bayes decision [18], [19], which is a classical minimum-cost decision method, has a long history. It has found applications in radar and sonar signal processing; a modified Bayes decision, called the Neyman–Pearson criterion, led to the widely employed *constant false alarm rate* (CFAR) processors. Applications of Bayes decisions in communications include detection of signals embedded in Gaussian or non-Gaussian white noise, using, for example, the maximum likelihood rate criterion. Bayes decisions were also applied in pattern recognition, etc. In this section, we first review the design of optimal stack filters under the MAE criterion and then we shall show that such a design procedure is completely equivalent to the *a priori* Bayes minimum-cost decision with some consistency constraints. Modifications to GSF's and ROF's will be discussed in Section V and VI, respectively.

Given a window width b stack filter $S_f(\cdot)$ operating on Ω_b, the MAE at time n between a desired signal $D(n)$ and the output produced by the stack filter on a multilevel window process $\vec{X}(n) \in \Omega_b$ can be expressed as

$$C(S_f) = E\{|D(n) - S_f(\vec{X}(n))|\}. \tag{3.1}$$

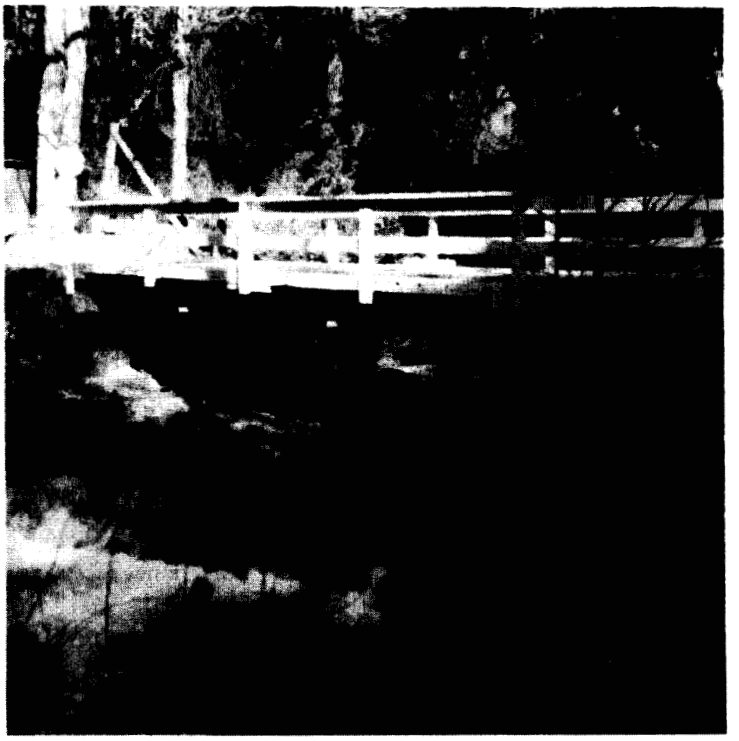


Fig. 3. Original image of bridge over stream.

Generally, the samples in the window process are composed of a signal component and a noise component. The type of interaction between the signal and the noise can be additive or multiplicative, or can have other characteristics. However, in this paper we shall always assume that the signal and the noise processes are jointly stationary. Therefore, the MAE is the same all the time.

Next, let us further assume that both the signal space and the input space are discrete and take on values in the set $\{0,1,\cdots,M\}$. This assumption makes possible the use of the threshold decomposition technique. With this technique, we can derive the following formulas as a basis for the design of optimal stack filters under the MAE criterion [2]–[5]:

$$C(S_f) = E\left\{\left|D(n) - S_f\left(\vec{X}(n)\right)\right|\right\}$$

$$= \sum_{l=1}^{M} E\left\{\left|d_l(n) - f\left(\vec{x}_l(n)\right)\right|\right\} \qquad (3.2)$$

where $d_l(n)$ is the thresholded value of $D(n)$ on level l; see (2.2).

Knowing the probability models of the signal, the noise, and the input window processes and considering the fact that there are only 2^b different states in the binary domain for window width b stack filters, we can further modify (3.2) (see [2]) to

$$C(S_f) = \sum_{j=1}^{2^b} \left[P_f\left(0|\vec{w}_j\right) \sum_{l=1}^{M} \pi_l\left(1,\vec{w}_j\right) + P_f\left(1|\vec{w}_j\right) \sum_{l=1}^{M} \pi_l\left(0,\vec{w}_j\right) \right]. \qquad (3.3)$$

The notation used in the above equation is explained as follows. $\vec{w}_j$ denotes a length b binary state observed by the filter. $P_f(0|\vec{w}_j)$ and $P_f(1|\vec{w}_j)$ are the filter decision probabilities corresponding to the outputs of the Boolean function $f(\cdot)$ when $\vec{w}_j$ is observed in the filter's window. They take on values in $\{0,1\}$ and are complementary to each other about 1, i.e.,

$$P_f\left(0|\vec{w}_j\right) + P_f\left(1|\vec{w}_j\right) = 1 \qquad \forall j. \qquad (3.4)$$

$\pi_l(0,\vec{w}_j)$ and $\pi_l(1,\vec{w}_j)$ are some joint probabilities that depend on the given probability models mentioned above and, by Bayes rule, each can be factored into two terms as

$$\pi_l\left(0,\vec{w}_j\right) = \pi_l\left(0|\vec{w}_j\right)\pi_l\left(\vec{w}_j\right)$$

$$\pi_l\left(1,\vec{w}_j\right) = \pi_l\left(1|\vec{w}_j\right)\pi_l\left(\vec{w}_j\right) \qquad (3.5)$$

where $\pi_l(\vec{w}_j)$ is the limiting probability of the event that the binary state $\vec{w}_j$ is observed on level l, and the condi-

tional probability

$$\pi_l(0|\vec{w}_j) = 1 - \pi_l(1|\vec{w}_j)$$
$$= \text{Prob}\{\text{signal value is less than } l|\vec{w}_j \text{ is observed on level } l\}. \quad (3.6)$$

Equation (3.3) can be interpreted as a Bayes decision. As probabilities $\pi_l(0,\vec{w}_j)$'s and $\pi_l(1,\vec{w}_j)$'s depend on the prior probability models of the signal, the noise, and the input processes, this Bayes decision is an *a priori* decision. If we interpret stack filtering of a multilevel window process by $S_f(\cdot)$ as a decision made upon binary threshold signals, we can see that $C(S_f)$ is the cost function of the decision where the cost coefficients for correct and wrong decisions are zero and one, respectively; $P_f(0|\vec{w}_j)$ and $P_f(1|\vec{w}_j)$ are the outputs of the decision. Following standard decision procedures, we can also use different cost coefficients (as in [2]):

$C_l(0,0,\vec{w}_j)$ ($C_l(0,1,\vec{w}_j)$): the cost of deciding the input signal is 0 (1) on level l when it is actually 0 and the binary state $\vec{w}_j$ is observed;

$C_l(1,0,\vec{w}_j)$ ($C_l(1,1,\vec{w}_j)$): the cost of deciding the input signal is 0 (1) on level l when it is actually 1 and the binary state $\vec{w}_j$ is observed.

Then, the cost function is generalized as follows:

$$C(S_f) = \sum_{j=1}^{2^b} \left[P_f(0|\vec{w}_j) E_j(0) + P_f(1|\vec{w}_j) E_j(1) \right] \quad (3.7)$$

where

$$E_j(0) = E(\text{cost} \mid 0 \text{ output and } \vec{w}_j \text{ is observed})$$
$$= \sum_{l=1}^{M} \left[C_l(0,0,\vec{w}_j)\pi_l(0,\vec{w}_j) + C_l(1,0,\vec{w}_j)\pi_l(1,\vec{w}_j) \right] \quad (3.8)$$

$$E_j(1) = E(\text{cost} \mid 1 \text{ output and } \vec{w}_j \text{ is observed})$$
$$= \sum_{l=1}^{M} \left[C_l(0,1,\vec{w}_j)\pi_l(0,\vec{w}_j) + C_l(1,1,\vec{w}_j)\pi_l(1,\vec{w}_j) \right] \quad (3.9)$$

which are the expected costs of deciding a zero or one, respectively, for each binary state observed by the filter [2].

Finally, recalling that a stack filter is represented by a positive Boolean function whose outputs obey the stacking property, we can describe the design procedure of optimal stack filters under the MAE criterion (or making the *a priori* Bayes minimum-cost decision) by the following optimization problem.

Optimization #3.1 ([2]):

minimize:

$$C(S_f) = \sum_{j=1}^{2^b} \left[P_f(0|\vec{w}_j) E_j(0) + P_f(1|\vec{w}_j) E_j(1) \right]$$

subject to:

$$P_f(0|\vec{w}_j), P_f(1|\vec{w}_j) = 0 \text{ or } 1$$

$$P_f(0|\vec{w}_j) + P_f(1|\vec{w}_j) = 1$$

$$P_f(1|\vec{w}_i) \geq P_f(1|\vec{w}_j) \quad \text{if } \vec{w}_i \geq \vec{w}_j.$$

The following theorem summarizes the solution and the complexity of Optimization #3.1.

Theorem 3.1 ([2]): There is always a solution for Optimization #3.1 that gives the optimal stack filter under the MAE criterion. This filter can be found by solving an LP with $O(b2^b)$ variables and constraints.

The complexity of the LP, $O(b2^b)$, increases faster than exponentially. A slight increase in b amounts to a large increase in the complexity and prohibits the use of the LP. This insurmountable (and unfortunate) obstacle appears even in relatively small window sizes, e.g., for a 4×4 window, the required LP contains over a million variables and constraints! In practice, it is almost impossible to perform an LP with such a large number of variables and constraints. Seeking suboptimal solutions thus becomes very important.

IV. Design of Suboptimal Stack Filters

In this section, we shall propose a suboptimal solution for the LP in Optimization #3.1. First, observing Optimization #3.1, one can find that "if" all the stacking constraints between binary states were removed, the resulting problem would be easy to solve. That is, the solution can be obtained by comparing $E_j(0)$ with $E_j(1)$ for each $\vec{w}_j \in \{0,1\}^b$. Of course, such a solution will in general violate the stacking property and thus will not be a stack filter. However, this raises the following question: can we somehow incorporate the stacking constraints into this approach? The answer is yes! We can solve Optimization #3.1 in an iterative way; at each stage, the optimization considers only a group of binary states that are all *incomparable* with respect to each other, whereas the stacking constraints are taken into account at the end of each stage.

With this idea, we need first partition 2^b binary states (for window width b stack filters) into groups, each group consisting of binary states that are incomparable. There are several ways to partition 2^b binary tuples. We select to perform this partition according to the number of 1's in a binary tuple. The reason for this partition choice will be apparent later. The resulting groups are $\Gamma_q = \{\vec{w}_j: \vec{w}_j \cdot \vec{1}^t = q\}$, $q = 0,1,\cdots,b$. Let $\Gamma = \cup_{i=0}^{b} \Gamma_i = \{0,1\}^b$, the set of all binary window process states. Now, at each stage of the LP, only one among these $b+1$ groups is selected and an LP is solved over it. At the end of each stage, the

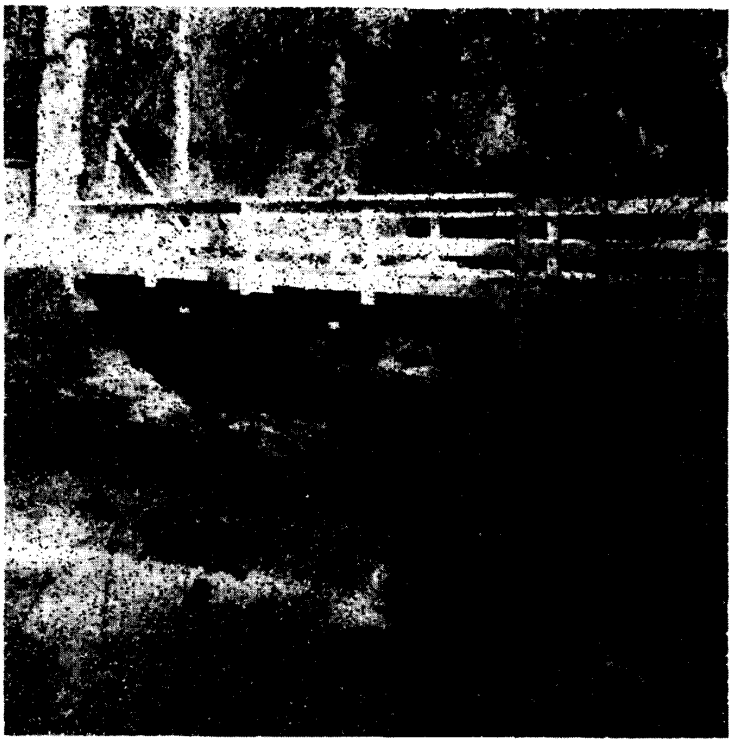


Fig. 4. Impulsive noise (10%) corrupted image of bridge over stream.

stacking property is used to decide the outputs corresponding to some window process states in other groups. For instance, if the LP performed over Γ_1 ($b = 3$) says that $P_f(1|\ 010) = 1$, then we know, by the stacking property, that $P_f(1|110) = 1$, $P_f(1|011) = 1$, and $P_f(1|111) = 1$. If there still remain some states whose outputs have not been determined yet, we select another group and continue to do an LP over it. (Note that the outputs corresponding to some states in this group might have already been decided by the stacking property, so the number of variables involved in the LP over the qth group is less than or equal to the number of states in the group which is $|\Gamma_q| = C_q^b$, where C_q^b is the binomial number.) A similar procedure is repeated until all 2^b binary states have been assigned output values, and thus the positive Boolean function representing the stack filter is completely determined.

Next, let us discuss how to select the starting group, which, obviously, seems to be very important to the proposed iterative algorithm. First, it is easy to see that if a group Γ_q with q being very close to 0 or b is selected, either the resulting solution is rather far from the optimal filter, assuming that the optimal solution is not trivial, or the number of binary states in other groups whose outputs will be set by the stacking property is small, leading to a slow iterative algorithm (i.e., the number of comparisons needed is large). To make this clearer, let us suppose, for instance, that Γ_0 is chosen, which contains only one element $\overbrace{0 \cdots 0}^{b \text{ times}}$. Now, if its output is decided to be 1, the stacking property will set outputs of all other binary states to 1, thus resulting in the trivial 1 filter, which is far away from the best filter, by the above assumption. On the other hand, if its output is decided to be 0 (must be), the outputs of all other binary states still remain undecided.

With the above reasoning, the best starting point of the proposed routine is thus to select the middle level binary window process state group Γ_{mid} with

$$\text{mid} = \begin{Bmatrix} b/2, & \text{for even } b \\ \lfloor b/2 \rfloor \text{ or } \lceil b/2 \rceil, & \text{for odd } b \end{Bmatrix}$$

where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x and $\lceil x \rceil$ is the smallest integer larger than or equal to x. The same reason suggests the following strategy to choose a group at the second stage, the third stage, and so on, in the iterative algorithm.

Rule A: Set q equal to an integer which is closest to mid and not all states in Γ_q have been assigned outputs. If there is more than one such group, select the one with the largest number of binary states whose outputs are not yet determined.

Clearly, the less frequent the stacking property is used to set other outputs, the closer the suboptimal solution is to the optimal one. On the other hand, the more the

stacking property is used to set other outputs, the faster the solution is obtained. The above selection rule with the chosen starting group presents a biased compromise, favoring less frequent use of the stacking property.

Summarizing the above description, the whole optimization procedure can be stated as follows.

Optimization #4.1:

Step 1: Set $q = \text{mid}$

Step 2: Minimize:

$$C(S_f) = \sum_{\vec{w}_j \in \Gamma_q} \left[P_f(0|\vec{w}_j) E_j(0) + P_f(1|\vec{w}_j) E_j(1) \right]$$

subject to:

$P_f(0|\vec{w}_j), P_f(1|\vec{w}_j) = 0$ or 1

$P_f(0|\vec{w}_j) + P_f(1|\vec{w}_j) = 1 \ \forall \vec{w}_j \in \Gamma_q$

Step 3: For each $\vec{w}_j \in \Gamma_q$,

if $P_f(0|\vec{w}_j) = 0$, then $P_f(0|\vec{w}_i) = 0$ for all $\vec{w}_i \geq \vec{w}_j, \vec{w}_i \in \Gamma$,

if $P_f(0|\vec{w}_j) = 1$, then $P_f(0|\vec{w}_i) = 1$ for all $\vec{w}_i \leq \vec{w}_j, \vec{w}_i \in \Gamma$.

Step 4: If all states have been decided, then go to Step 5,

else, reset q according to Rule A and go to Step 2.

Step 5: Stop.

Since all elements in Γ_q are incomparable (implying that no stacking constraints exist among them), we can immediately conclude, by the reasoning presented at the beginning of this section, that the LP's in Optimization #4.1 are, in fact, trivial ones in the sense that all computations needed are exclusively data comparisons between the expected costs defined by (3.8) and (3.9). Specifically, if $E_j(0) < E_j(1)$, we choose $P_f(0|\vec{w}_j) = 1$; and if $E_j(0) > E_j(1)$, we choose $P_f(1|\vec{w}_j) = 1$. If both costs are equal, one strategy is to choose arbitrarily $P_f(0|\vec{w}_j)$ or $P_f(1|\vec{w}_j)$ to be 1. However, a better way to deal with this case is to leave this state as a *don't care* state temporarily. Its output can be decided later by the stacking property after the outputs of binary window process states in adjacent groups have been decided. This way, the resulting filter would be closer to the optimal filter.

Moreover, it is easy to see that an upper bound on the number of comparisons needed is 2^b since there are only 2^b different binary states for window of width b. The exact number of comparisons is difficult to obtain; nevertheless, a tighter bound is desired. First, it is worth pointing out that this number will usually be much smaller than 2^b, since the outputs of many binary states would have been already set by the stacking property. For instance, with our choice of starting point and if the states in Γ_{mid} are not decided to be all 0's or all 1's, at least $\sum_{q=1}^{\text{mid}} C_q^{\text{mid}} + \sum_{q=1}^{b-\text{mid}} C_q^{b-\text{mid}}$ states will be set at the end of the first stage in the algorithm. Actually, the outputs of many more states will be set at the end of the first stage. The given lower bound corresponds merely to the consequence of the output of one state in Γ_{mid} being 1 and one state in Γ_{mid} being 0. To make this clearer, let us consider an example with $b = 4$. For this example, the starting group is $\Gamma_2 = \{0011, 0101, 0110, 1001, 1010, 1100\}$. Suppose that $P_f(1|0011) = 1$ and $P_f(1|0101) = 0$. By the stacking property, we can then set the outputs for six other binary states in $\{0,1\}^4$, i.e., $P_f(1|0111) = P_f(1|1011) = P_f(1|1111) = 1$ and $P_f(1|0001) = P_f(1|0100) = P_f(1|0000) = 0$. More states could be set by using the decision outputs of binary states in Γ_2 other than 0011 and 0101. It is even possible that all the other binary states will be set after the first stage. For instance, let us consider the above example again. If the outputs of 0011 and 1100 are decided to be 0 and the outputs of 0101, 0110, 1001, and 1010 are decided to be 1, then all the other binary states in $\{0,1\}^4$ will be set by the stacking property. Thus only 6 comparisons are needed. For this example, one can find that the worst case happens when the binary states selected at the first stage are all decided to be 1 or 0; and this repeats at subsequent stages. For this case, the above example will require 11 comparisons. However, the worst case will result in the max or the 1 filter for the 1 case and the min or 0 filter for the 0 case, respectively, which are rare cases in practice.

Actually, it can be shown that the above worst case holds also for other window widths, and the corresponding number of comparisons required is $\text{MAX}(\sum_{q=0}^{\text{mid}} C_q^b, \sum_{q=\text{mid}}^{b} C_q^b) = \beta C_{\text{mid}}^b$ which increases slower than exponentially as a function of the filter window width. For window widths less than or equal to 25, we found that $1 \leq \beta < 4.20$. In practice, $\beta = 2$ is a reasonable estimate.

In summary, the above discussion shows two facts: 1) the proposed suboptimal routine completely avoids the use of an LP and the required computations reduce to merely data comparisons, and 2) the number of comparisons is smaller than the number of variables and constraints involved in the LP for finding the optimal stack filter. Hence the suboptimal solution can be obtained efficiently. More importantly, with the proposed rule to choose the starting point and successive groups, the resulting suboptimal filter will be *reasonably* close to the optimal one. However, deeper insights on the goodness of this suboptimal algorithm (i.e., how close the suboptimal solution is to the optimal one produced by an LP) have not been made clear yet. Of course, it is always possible in practice to find this difference by comparing the MAE's of the resulting filters. However, this defeats the purpose of using a suboptimal approach. A better (and, of course, wiser) alternative would be to seek sufficient conditions under which the suboptimal approach would yield optimal solutions. This alternative approach would be a great success if the condition is not very restrictive and holds in most practical cases. The following is an attempt towards this direction.

First, let us suppose that Optimization #4.1 is solved ignoring Step 3, that is, the stacking property is not used to set the outputs for binary states in other groups. Then it is easy to show that if this procedure results in a

positive Boolean function, then this solution is optimal in the minimum MAE sense.

The above statement is the key to the sufficient conditions we are seeking. What we need are explicit conditions on the coefficients of the objective function to be minimized. Let us consider the binary case first (i.e., the input space Ω is binary). To this end, we assume that the cost coefficients for correct decisions are zeros and those for wrong decisions are the same for all binary states. Therefore, the two cost coefficients for wrong decisions are now denoted by $C(1,0)$, the cost of deciding the output to be 0 when it is actually 1, and $C(0,1)$, the cost of deciding the output to be 1 when it is actually 0. Under these assumptions, (3.7) reduces to

$$C(f) = \sum_{j=1}^{2^b} \left[P_f\left(0|\vec{w}_j\right) C(1,0)\pi\left(1|\vec{w}_j\right) + P_f\left(1|\vec{w}_j\right) C(0,1)\pi\left(0|\vec{w}_j\right)\right]\pi\left(\vec{w}_j\right). \quad (4.1)$$

Before stating the first theorem, the following definition explains what it means for the conditional probabilities introduced earlier to possess the stacking property.

Definition 4.1: The conditional probabilities $\pi(0|\vec{w}_j)$'s are said to possess the stacking property if the following inequality holds for all binary states:

$$\pi\left(0|\vec{w}_i\right) \leqslant \pi\left(0|\vec{w}_j\right) \qquad \text{whenever } \vec{w}_j \leqslant \vec{w}_i. \quad (4.2)$$

Based on this definition, we have the following theorem.

Theorem 4.1: If the conditional probabilities $\pi(0|\vec{w}_j)$'s of the input binary window process possess the stacking property, then the stack filter designed by solving the suboptimal routine is optimal under the MAE criterion.

See Appendix A for the proof. The constraint that the conditional probabilities possess the stacking property, as defined in Definition 4.1, is natural and not so restrictive. It means that the more ones there are in a binary window process state, the less probable it is that the true signal value is zero. Therefore, it is easy to understand that this property should hold in most practical cases.

The above result can easily be extended to multilevel signals. To do this, we shall first define the following stacking property between the expected costs $E_j(0)$'s and $E_j(1)$'s.

Definition 4.2: The expected costs $E_j(0)$'s and $E_j(1)$'s are said to possess the stacking property if $E_j(0) < E_j(1) \Rightarrow E_i(0) < E_i(1)$, $\forall(\vec{w}_i < \vec{w}_j)$.

Based on this definition, we have the following theorem.

Theorem 4.2: Suppose that the expected costs $E_j(0)$'s and $E_j(1)$'s possess the stacking property. Then the stack filter designed by solving the suboptimal routine is optimal under the MAE criterion.

The proof is analogous to that of Theorem 4.1 and it is therefore omitted. The stacking property defined for the expected costs has an interpretation similar to that for the condition probabilities in the binary case. It says that as the number of zeros in a binary state increases, the cost of deciding a 0 (1) output decreases (increases). Therefore, if, for a given binary state, the cost of deciding a 0 is less than the cost of deciding a 1, then for any other binary state stacking on top of this state, the cost of deciding a 0 is also less than that of deciding a 1.

Comparing the sufficient conditions of Theorems 4.1 and 4.2, one can find that the condition of Theorem 4.2 is looser than that of Theorem 4.1. Any attempt to further weaken this condition would make the suboptimal solution closer to the optimal one, and might even reduce the complexity of the suboptimal approach (i.e., reduce the number of comparisons). This does not seem to be such an easy task, however. This is especially true for the multilevel case, although one might argue that the sufficient condition of Theorem 4.2 is not required at least for those binary states whose outputs were set by the stacking property. However, this does not lead to any significant improvement in the optimization procedure and more importantly, it does not make the difference between the optimal and suboptimal solutions any easier to compute.

A second attempt to weaken these sufficient conditions, which might be useful in certain applications, is to relax Condition (4.2) of Definition 4.1 as follows. Assuming again here that correct decisions cost nothing, we have the following modification of Theorem 4.1.

Theorem 4.3: For each binary state $\vec{w}_j$ such that

$$\pi\left(0|\vec{w}_j\right) < \frac{C\left(1,0,\vec{\omega}_j\right)}{C\left(0,1,\vec{w}_j\right) + C\left(1,0,\vec{w}_j\right)} \quad (4.3)$$

if there does not exist $\vec{w}_i$ such that $\vec{w}_i \geqslant \vec{w}_j$ and $\vec{w}_i$ violates (4.3), then the suboptimal solution produced by Optimization #4.1 is optimal in the minimum MAE sense.

See Appendix A for the proof.

Remark: The condition of Theorem 4.3 is looser than that of Theorem 4.1 because (4.2) need not hold for any stacking pair of binary states that violate or do not violate (4.3) simultaneously and yet the suboptimal solution is optimal. In addition, the condition of Theorem 4.3 is more general than that of Theorem 4.1 since different cost coefficients are allowed to be used for different binary states.

V. Design of GSF's

In this section, we continue to discuss the *a priori* Bayes minimum-cost decision, but the decision problem is now divided into M sub-decisions, on the binary threshold arrays defined in Section II, which are constrained to be consistent with respect to each other. This approach will produce M Boolean functions. The constraint that the M sub-decisions are consistent with respect to each other guarantees the stacking property in the resulting set of Boolean functions. Therefore, this set of Boolean functions forms a GSF.

From the definition of GSF's, we know that the input, at each time instance, to the Boolean function at each level is the binary threshold array with dimensions $(2I +$

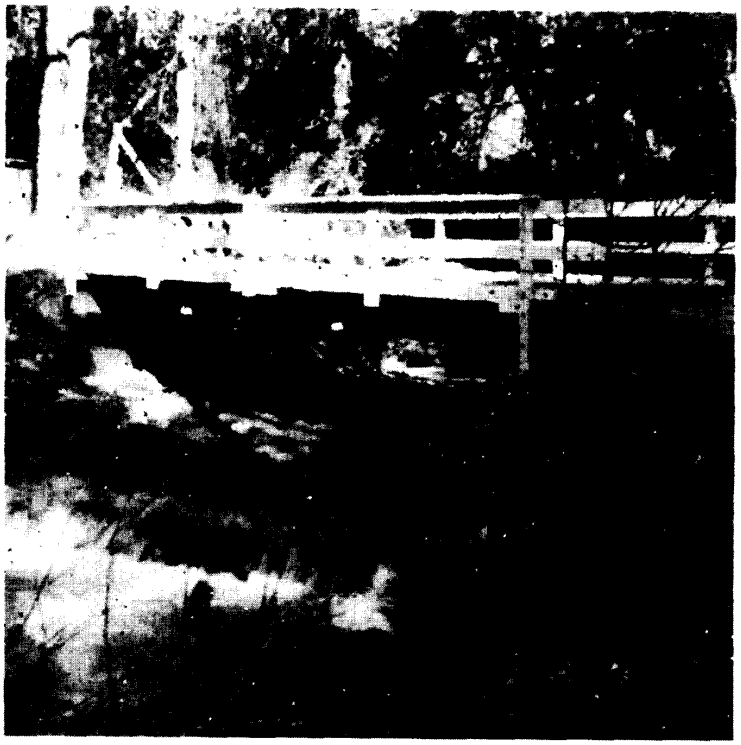


Fig. 5. Output of the optimal stack filter with a 3×3 rectangular window with image in Fig. 4 as input. The upper-left quarter parts of images in Figs. 3 and 4 are used to estimate the probabilities.

$1)\times b$ in which all rows obey the internal stacking property with respect to each other. Using the notation B_b^I introduced earlier and denoting by $\vec{w}_j^I$ the elements in B_b^I, we can describe the design procedure of optimal GSF's as follows.

Optimization #5.1 ([13]):

minimize:

$$C(F_{I,b}) = \sum_{l=1}^{M} C(S_{f_l}) = \sum_{l=1}^{M} \sum_{\vec{w}_j^I \in B_b^I} \left[P_{f_l}\left(0|\vec{w}_j^I\right) E_j^{(l)}(0) + P_{f_l}\left(1|\vec{w}_j^I\right) E_j^{(l)}(1) \right]$$

subject to:

$$P_{f_l}\left(0|\vec{w}_j^I\right), P_{f_l}\left(1|\vec{w}_j^I\right) = 0 \text{ or } 1$$

$$P_{f_l}\left(0|\vec{w}_j^I\right) + P_{f_l}\left(1|\vec{w}_j^I\right) = 1$$

$$P_{f_l}\left(0|\vec{w}_j^I\right) + P_{f_l}\left(1|\vec{w}_j^I\right) = 1$$

$$P_{f_l}\left(1|\vec{w}_j^I\right) \geq P_{f_m}\left(1|\vec{w}_i^I\right) \text{ whenever } l < m \text{ and } \vec{w}_i^I \leq \vec{w}_j^I$$

where

$$E_j^{(l)}(0) = C_l\left(0,0,\vec{w}_j^I\right)\pi_l\left(0,\vec{w}_j^I\right) + C_l\left(1,0,\vec{w}_j^I\right)\pi_l\left(1,\vec{w}_j^I\right) \tag{5.1}$$

and

$$E_j^{(l)}(1) = C_l\left(0,1,\vec{w}_j^I\right)\pi_l\left(0,\vec{w}_j^I\right) + C_l\left(1,1,\vec{w}_j^I\right)\pi_l\left(1,\vec{w}_j^I\right). \tag{5.2}$$

The following theorem presents the solution and the complexity of Optimization #5.1.

Theorem 5.1 ([13]): There is always a solution for Optimization #5.1 that gives the optimal GSF under the MAE criterion. The filter can be found by solving an LP with $O(M(2I+3)^b)$ variables and constraints.

From this theorem, it is easy to see that, again, for a large size window, the complexity of the LP required in Optimization #5.1 is insurmountable, and one must seek a suboptimal approach to reduce this huge complexity problem. This will be discussed in the following.

The idea of the suboptimal routine is to separate the stacking constraints in the set of Boolean functions from the programming itself and solve the LP in several stages. This is the same as the one used to design suboptimal stack filters, but the actual implementation is quite different.

To do this, we first select one threshold level l and then solve the corresponding optimization problem on this level only. After this step, we use the stacking property to set the outputs for some binary threshold arrays at other threshold levels. For instance, if the LP on level l says that $P_{f_l}(0|\vec{w}_j^I) = 1$, then we must set $P_{f_m}(0|\vec{w}_i^I) = 1$ for

$m = l+1, l+2, \cdots, M$ and $\vec{w}_i^I \leqslant \vec{w}_j^I$. If there remain some elements in B_b^I whose outputs are still undecided for some Boolean function in the GSF, we select another threshold level and continue to solve an LP on this level. A similar procedure is repeated until all elements in B_b^I on all levels have been assigned outputs. It is easy to see that such an approach guarantees the stacking property in the resulting set of M Boolean functions and therefore a GSF can be obtained.

It is rather straightforward to see that with the same reasoning as for the stack filter case, the most reasonable starting level is the middle level, i.e., $l = (M+1)/2$ for odd M and $l = M/2 \pm 1$ for even M. (Let $[M/2]$ denote this middle level.) Also, the strategy of level selection at subsequent stages is similar to that proposed for stack filters.

Rule B: Reset l to a level that is closest to the middle level and that contains binary arrays whose outputs have not been determined yet. If there are more than one such level, choose the one with the largest number of binary arrays whose outputs are not found yet.

Note that the same principle following Rule A in the previous section also applies here. Now, the whole optimization procedure can be stated as follows.

Optimization #5.2:

Step 1: Set $l = [M/2]$

Step 2: Minimize:

$$C(S_{f_l}) = \sum_{\vec{w}_j^I \in B_b^I} \left[P_{f_l}(0|\vec{w}_j^I) E_j^{(l)}(0) + P_{f_l}(1|\vec{w}_j^I) E_j^{(l)}(1) \right]$$

subject to:

$P_{f_l}(0|\vec{w}_j^I),\ P_{f_l}(1|\vec{w}_j^I) = 0 \text{ or } 1, \quad \forall \vec{w}_j^I \in B_b^I$

$P_{f_l}(0|\vec{w}_j^I) + P_{f_l}(1|\vec{w}_j^I) = 1, \ \forall \vec{w}_j^I \in B_b^I$

Step 3: For each $\vec{w}_j^I \in B_b^I$,

if $P_{f_l}(0|\vec{w}_j^I) = 0$, then $P_{f_m}(0|\vec{w}_i^I) = 0$,

for $m = l-1, l-2, \cdots, 1$, and $\vec{w}_i^I \geqslant \vec{w}_j^I$;

if $P_{f_l}(0|\vec{w}_j^I) = 1$, then $P_{f_m}(0|\vec{w}_i^I) = 1$,

for $m = l+1, l+2, \cdots, M$, and $\vec{w}_i^I \leqslant \vec{w}_j^I$.

Step 4: If all $\vec{w}_j^I \in B_b^I$ have been decided at all levels, then go to Step 5; else, reset l according to Rule B and go to Step 2.

Step 5: Stop.

In the same way as was done for the stack filter case, it is easy to show that to solve Optimization #5.2, the use of an LP becomes unnecessary and the required computations are merely data comparisons. Moreover, all the outputs of the Boolean functions on all levels can be decided by the same compare-and-select algorithm described in the previous section.

Due to the complicated architecture of GSF's, the exact number of comparisons needed for the current case is even more difficult to determine than for the stack filter case. However, it is easy to see that this number is proportional to $|B_b^I|$, the number of binary arrays in B_b^I that equals $(2I+2)^b$. Let γ denote this proportionality factor. Obviously, the worst case is $\gamma = M$. However, such a case will never be met, since, at the end of the first stage, we can always set the outputs of at least one binary array by using the stacking property. It is worth noting that even in the worst case, the number of comparisons needed is still smaller than the number of variables and constraints involved in the LP: $M(2I+2)^b$ versus $M(2I+3)^b$. As in the stack filter case, it is even possible that after the first stage, all Boolean functions on other levels will be determined by the stacking property. For instance, let us consider the case $b = 3$, $M = 3$, and $I = 0$. The starting level is then $l = 2$. Now, if the outputs of binary states 000, 001, 010, and 100 on this level are decided to be 1, while the outputs of the other four states are decided to be 0, then the Boolean functions on levels 1 and 3 will all be decided by the stacking property. The resulting three functions are $f_1 = 1$, $f_2 = \bar{x}_1\bar{x}_2 + \bar{x}_2\bar{x}_3 + \bar{x}_3\bar{x}_1$, and $f_3 = 0$. This set of three Boolean functions, though it may not be met in practice, does form a GSF, although f_2 is not positive. In practice, $\gamma \approx 3$ is a reasonable estimate. This estimate is obtained by assuming that at each stage half of the binary arrays in B_b^I whose outputs have not been determined yet are decided to be 1 and the outputs of the other half are decided to be 0.

For the goodness of this suboptimal algorithm, the same remark as in the stack filter case applies. This, however, has a positive impact on this development since we are again compelled to seek sufficient conditions (which we hope would be not so restrictive and hold in most practical cases) under which the suboptimal algorithm would yield optimal solutions.

Let us first focus on the effect of Step 3 in Optimization #5.2 on the solution of the minimization procedure. Suppose that Optimization #5.2 is solved ignoring Step 3 (i.e., the stacking constraints are not used to set the outputs for binary threshold arrays at other levels). Then, it is easy to see that if the resulting set of M Boolean functions satisfies the stacking property along the threshold levels, then the solution is optimal in the minimum MAE sense.

To make the above statement more explicit and hence derive some more useful results, let us define in the following what it means for the expected costs to possess the stacking property along the threshold levels.

Definition 5.1: The expected costs $E_j^{(l)}(0)$'s and $E_j^{(l)}(1)$'s are said to possess the stacking property along the threshold levels if $E_j^{(l)}(0) < E_j^{(l)}(1) \Rightarrow E_i^{(m)}(0) < E_i^{(m)}(1), m > l$, $\forall (\vec{w}_i^I \leqslant \vec{w}_j^I)$.

Notice that Definitions 4.2 and 5.1 define two different types of stacking properties between two different sets of expected costs. Based on Definition 5.1, we have the following theorem.

Theorem 5.2: If the expected costs $E_j^{(l)}(0)$'s and $E_j^{(l)}(1)$'s possess the stacking property along the threshold levels, then the set of Boolean functions obtained by solving Optimization #5.2 for M threshold levels automatically

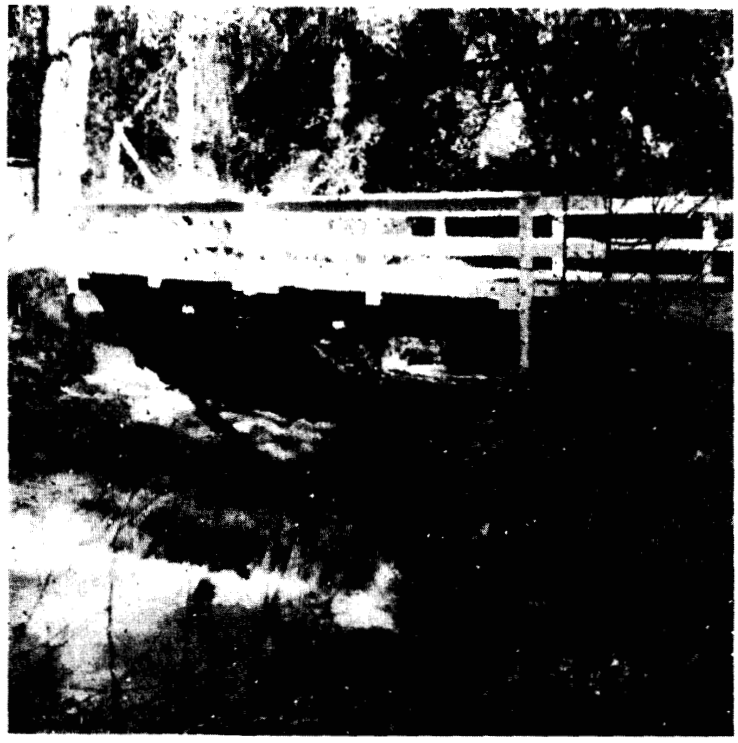


Fig. 6. Output of the optimal GSF with a 3×3 rectangular window and one threshold level fed into each filter ($I=0$). The input image to this GSF is Fig. 4. The upper-left quarter parts of images in Figs. 3 and 4 are used to estimate the probabilities.

possesses the stacking property, and hence, a GSF is obtained that is optimal under the MAE criterion.

See Appendix A for the proof. For the special case $I=0$ mentioned in Section II where each Boolean function $f_l(\cdot)$ is fed with the threshold signal on level l only, if we assume that the cost coefficients for correct decisions are zero, then the expected costs $E_j^{(l)}(0)$ and $E_j^{(l)}(1)$ reduce to

$$E_j^{(l)}(0) = C_l\left(1,0,\vec{w}_j\right)\pi_l\left(1|\vec{w}_j\right)\pi_l\left(\vec{w}_j\right)$$
$$E_j^{(l)}(1) = C_l\left(0,1,\vec{w}_j\right)\pi_l\left(0|\vec{w}_j\right)\pi_l\left(\vec{w}_j\right) \qquad (5.3)$$

where $\vec{w}_j$ was used rather than $\vec{w}_j^I$ because for this case, the binary threshold arrays reduce to binary vectors. Accordingly, the stacking property of Definition 5.1 can be modified for this case as follows.

Definition 5.2: The conditional probabilities $\pi_l(0|\vec{w}_j)$'s are said to possess the stacking property along the threshold levels if for any $\vec{w}_i \leqslant \vec{w}_j$,

$$\pi_l\left(0|\vec{w}_j\right) \leqslant \pi_m\left(0|\vec{w}_i\right) \qquad \text{whenever } l<m. \qquad (5.4)$$

Note that the same remark following Definition 5.1 also applies here, which differentiates the stacking property defined above from that in Definition 4.1. Now, let us further assume that for a binary threshold state $\vec{w}_j$, the cost coefficients $C_l(1,0,\vec{w}_j)$ and $C_l(0,1,\vec{w}_j)$ are the same for all threshold levels. Then, for the case $I=0$, we have the following theorem.

Theorem 5.3: If the conditional probabilities $\pi_l(0|\vec{w}_j)$'s possess the stacking property along the threshold levels, then the GSF based on the set of Boolean functions obtained by solving Optimization #5.2 is optimal under the MAE criterion.

The proof is analogous to that of Theorem 5.2 and it is therefore omitted. The stacking property defined in both Definitions 5.1 and 5.2 has a natural interpretation. Threshold decomposition preserves order, that is, higher binary threshold signals stack on top of lower ones and hence they are likely to contain more zeros. As a consequence, when l approaches to M, the highest level, the probability for the true signal value exceeding the threshold level decreases and accordingly the cost of deciding a zero output decreases. Therefore, it is easy to understand that this property should also hold in most practical cases.

Finally, when $I=0$ and $C_l(0,1,\vec{w}_j)$ and $C_l(1,0,\vec{w}_j)$ are not the same for all threshold levels, we have the following theorem.

Theorem 5.4: Let

$$G_l = \left\{ \vec{w}_j \colon \pi_l\left(0|\vec{w}_j\right) < \frac{C_l\left(1,0,\vec{w}_j\right)}{C_l\left(0,1,\vec{w}_j\right) + C_l\left(1,0,\vec{w}_j\right)} \right\}.$$

If $G_M \subset G_{M-1} \subset \cdots \subset G_1$, then the suboptimal routine will result in optimal solutions.

Proof: For $\vec{w}_j \in G_l$, it can easily be shown that the suboptimal routine will choose $P_{fl}(1|\vec{w}_j) = 1$. Therefore, the condition $G_M \subset G_{M-1} \subset \cdots \subset G_1$ guarantees the stacking property in the set of M Boolean functions. According to our previous statement, the suboptimal solution is optimal in the minimum MAE sense. □

VI. Posterior Decisions and ROF's

In this section, we give the optimality theory a new interpretation in terms of the *a posteriori* Bayes minimum-cost decision. Specifically, we will show that this will always lead to an ROF as the optimal solution. The aim of this approach is also to minimize the MAE between a desired signal $D(n)$ and the output of a stack filter S_f operating on a window process $\vec{X}(n)$. In the binary domain, this MAE can be written as (assuming in this section that the cost coefficients for correct decisions are always zero):

$$C(S_f) = \sum_{j=1}^{2^b} \left\{ P_f(0|\vec{w}_j) \sum_{l=1}^{M} C_l(1,0,\vec{w}_j)\pi_l(1|\vec{w}_j)\pi_l(\vec{w}_j) + P_f(1|\vec{w}_j) \sum_{l=1}^{M} C_l(0,1,\vec{w}_j)\pi_l(0|\vec{w}_j)\pi_l(\vec{w}_j) \right\}. \quad (6.1)$$

In Section III, we defined $\pi_l(0|\vec{w}_j)$ based on the prior probability models of the signal, the noise, and the input processes, and, therefore, the resulting decision belongs to the *a priori* Bayes decision. The solution of this decision gives a stack filter or GSF depending on whether the decision is solved as a whole decision or M consistent sub-decisions.

In this section, we introduce another definition of $\pi(0|\vec{w}_j)$ which provides the *a posteriori* interpretation. Specifically, we define

$$\pi_l(0|\vec{w}_j) = \text{Prob}\{\text{filtered output value is less than } l|\vec{w}_j \text{ is observed}\} = \frac{N_b^{(0)}(\vec{w}_j)}{N_b} \quad (6.2)$$

where N_b is the number of positive Boolean functions with b variables and $N_b^{(0)}(\vec{w}_j)$ is the number of positive Boolean functions, with b variables, which produce a zero for binary state $\vec{w}_j$ at level l. Similarly, $\pi_l(1|\vec{w}_j)$ is defined by

$$\pi_l(1|\vec{w}_j) = \frac{N_b^{(1)}(\vec{w}_j)}{N_b} \quad (6.3)$$

where $N_b^{(1)}(\vec{w}_j)$ is the number of positive Boolean functions, with b variables, which produce a 1 for binary state $\vec{w}_j$ on level l. It is easy to show that $\pi_l(0|\vec{w}_j) + \pi_l(1|\vec{w}_j) = 1$ for all $\vec{w}_j \in \{0,1\}^b$.

It should be pointed out that the posterior meaning defined above is different from the standard one where the decision is based on the probability model of the outputs. However, such a definition does give the *a posteriori* meaning because it is based on observations after the filtering pass.

By (6.2), it is easy to see that $\pi_l(0|\vec{w}_j)$ is independent of level l, i.e.,

$$\pi_l(0|\vec{w}_j) = \pi_m(0|\vec{w}_j) \qquad \forall \vec{w}_j, l, \text{ and } m. \quad (6.4)$$

With the probabilities defined by (6.2) and (6.3), the optimization procedure is greatly simplified, as will be shown in the following.

Three lemmas are given first. Their proofs can be found in Appendix B.

Lemma 6.1: Probabilities $\pi(0|\vec{w}_j)$'s possess the stacking property, i.e.,

$$\pi(0|\vec{w}_i) \leqslant \pi(0|\vec{w}_j) \qquad \text{whenever } \vec{w}_j \leqslant \vec{w}_i. \quad (6.5)$$

Lemma 6.2: Let $\vec{w}_i$ and $\vec{w}_j$ denote two binary states. If $\vec{w}_i \cdot \vec{1}^t = \vec{w}_j \cdot \vec{1}^t$, then

$$\pi(1|\vec{w}_i) = \pi(1|\vec{w}_j). \quad (6.6)$$

Lemma 6.3: Given two binary states $\vec{w}_i$ and $\vec{w}_j$. If they satisfy the condition $\vec{w}_i \cdot \vec{1}^t + \vec{w}_j \cdot \vec{1}^t = b$, then

$$\pi(0|\vec{w}_i) = \pi(1|\vec{w}_j). \quad (6.7)$$

With these lemmas and assuming that the cost coefficients $C(1,0,\vec{w}_j)$ and $C(0,1,\vec{w}_j)$ are the same for all $\vec{w}_j$'s, we present the theorem concerning the solution of the *a posteriori* Bayes minimum-cost decision as follows.

Theorem 6.1: The solution of the *a posteriori* Bayes minimum-cost decision is an ROF, which can be obtained without solving an LP.

Also, see Appendix B for the proof. It should be pointed out that from the proof, one can see that the solution actually does not depend on the prior probability models. This is because $C(0,1)$, $C(1,0)$, and $\pi_l(0|\vec{w}_j)$ are all independent of the probability models.

Finally, an important special case of Theorem 6.1 occurs, often in practice, when $C(1,0)$ equals $C(0,1)$. (Assume that the filter's window width b is odd.)

Consequence 6.1: If the cost coefficients $C(0,1)$ and $C(1,0)$ are equal, then the solution of the *a posteriori* Bayes minimum-cost decision is the median filter.

Proof: When $C(1,0)$ and $C(0,1)$ are equal, we need only check if $\pi(1|\vec{w}_j) < \pi(0|\vec{w}_j)$ in order to decide the output value for $\vec{w}_j$. This, however, is equivalent to checking if $\pi(1|\vec{w}_j) < 0.5$. If this is true, then the decision output for $\vec{w}_j$ is 0; otherwise the output is 1. But, based on Lemma 6.3 and Lemma 6.1, it is easy to show that if $\vec{w}_j \cdot \vec{1}^t < (b+1)/2$, $\pi(1|\vec{w}_j)$ is always less than 0.5, implying a zero output. As long as $\vec{w}_j \cdot \vec{1}^t \geqslant (b+1)/2$, $\pi(1|\vec{w}_j)$ becomes larger than 0.5, implying a one output. Therefore, the solution is the median filter. □

Remark: The case where $C(0,1)$ and $C(1,0)$ are equal is encountered in many practical applications. Thus Consequence 6.1 provides a complementary interpretation to the optimality of the median filter.

Several design examples will be presented in Section VIII.

VII. Sensitivity Analysis

In the optimization procedures discussed earlier, it is possible that a small and sometimes even a large variation in the cost coefficients does not change the solution. This section is an attempt to quantify these variations. The main question we will try to answer is: how much is each cost coefficient allowed to vary and still produce the same solution? These variations are directly related to deviations in the stochastic models assumed for the signal and noise. The larger these acceptable variations are, the more robust the solution of the optimization procedure is.

In the optimal LP approach, this analysis is known as sensitivity analysis or cost ranging analysis [20]. It is used to determine the range within which each cost coefficient in the cost vector is allowed to vary without changing the optimal solution. This was used in [4], [5], and [22] where the objective of the LP was modified to design optimal stack filters with structural constraints. To simplify the analysis, let us assume in this section (except for the results presented in Theorem 7.2 and 7.3) that the cost coefficients for correct decisions are zero and those for wrong decisions are the same for all binary states for stack filters (and all binary threshold arrays for GSF's).

Starting with ROF's, we would like to determine the range in which the cost coefficients $C(0,1)$ and $C(1,0)$ are allowed to vary while the optimal ROF remains unchanged. The following theorem does just that.

Theorem 7.1: The rth ROF is optimal under the MAE criterion iff

$$\pi\left(1|\vec{w}_i\right) < \frac{C(0,1)}{C(0,1)+C(1,0)} < \pi\left(1|\vec{w}_j\right) \tag{7.1}$$

for any $\vec{w}_i \in \Gamma_{r-1}$ and $\vec{w}_j \in \Gamma_r$.

See Appendix C for the proof. For the median case ($r=(b+1)/2$), since $\pi(1|\vec{w}_i)+\pi(1|\vec{w}_j)=1$ for any $\vec{w}_i \in \Gamma_{r-1}$ and $\vec{w}_j \in \Gamma_r$, it can easily be shown that the median filter is optimal (in the *a posteriori* sense) under the MAE criterion iff

$$\max\left(\frac{C(0,1)}{C(0,1)+C(1,0)}, \frac{C(1,0)}{C(0,1)+C(1,0)}\right) < \pi\left(1|\vec{w}_j\right). \tag{7.2}$$

Next, let us consider the sensitivity analysis of stack filters and GSF's. For these cases, the factors affecting the solution are the expected costs $E_j(0)$'s and $E_j(1)$'s for stack filters or $E_j^{(l)}(0)$'s and $E_j^{(l)}(1)$'s for GSF's. To simplify the analysis, let us further assume that these costs satisfy the stacking property (see Definition 4.2 and 5.1.) Therefore, we need only consider the sensitivity analysis of the two suboptimal routines, proposed in Sections IV and V, which produce optimal solutions under the above assumptions.

First, define

$$\Delta_j = E_j(1) - E_j(0) \tag{7.3}$$

and

$$\Delta_j^{(l)} = E_j^{(l)}(1) - E_j^{(l)}(0). \tag{7.4}$$

The following theorem presents the sensitivity of a stack filter as a function of the expected costs.

Theorem 7.2: The expected costs $E_j(0)$'s and $E_j(1)$'s for a binary state $\vec{w}_j$ can vary independently of the costs corresponding to other binary states. As long as these variations do not change the polarity of Δ_j for all $\vec{w}_j$'s, the optimal stack filter remains unchanged.

The proof of this theorem is quite simple. Because the condition stated in the theorem directly implies that the decision output for each binary state is not changed by the variations in the expected costs, the same stack filter will result. A similar result concerning the sensitivity of a GSF can be stated as follows.

Theorem 7.3: The expected costs $E_j^{(l)}(0)$'s and $E_j^{(l)}(1)$'s for a binary array $\vec{w}_j^l$ on threshold level l can vary independently of the costs corresponding to the same binary array $\vec{w}_j^l$ but on different levels. As long as these variations do not change the polarity of $\Delta_j^{(l)}$ for all $\vec{w}_j^l \in \Omega_b^l$ on all levels, the optimal GSF remains unchanged.

In the following, two special cases, the binary case for stack filters and the case $l=0$ for GSF's, are considered in more detail.

First, let us consider stack filters. For the sensitivity analysis, we assume, without loss of generality, that $C(0,1)$ is fixed and only $C(1,0)$ varies by an amount α. Next, we shall determine the range of values over which α can vary while the resulting stack filter remains unchanged. To do this, suppose that the optimal stack filter $f(\cdot)$ has been obtained by the proposed suboptimal routine and let G_0 and G_1 denote two sets of binary states such that the stack filter produces a 0 output for all states in G_0 and a 1 output for all states in G_1, respectively. Then, we have the following theorem.

Theorem 7.4: The optimal stack filter designed by the suboptimal routine remains the same iff α varies in the range

$$\max_{\vec{w}_j \in G_1}\left\{C(0,1)\frac{\pi\left(0|\vec{w}_j\right)}{\pi\left(1|\vec{w}_j\right)} - C(1,0)\right\} < \alpha < \min_{\vec{w}_j \in G_0}\left\{C(0,1)\frac{\pi\left(0|\vec{w}_j\right)}{\pi\left(1|\vec{w}_j\right)} - C(1,0)\right\}. \tag{7.5}$$

See Appendix C for the proof. Similarly, for GSF's with $l=0$, we fix all $C_l(0,1)$'s and let each $C_l(1,0)$ vary by an amount α_l. Then, we have the following theorem.

Theorem 7.5: The optimal GSF designed by the suboptimal routine remains the same iff α_l varies in the range

$$\max_{\vec{w}_j \in G_1^{(l)}}\left\{C_l(0,1)\frac{\pi_l\left(0|\vec{w}_j\right)}{\pi_l\left(1|\vec{w}_j\right)} - C_l(1,0)\right\} < \alpha_l < \min_{\vec{w}_j \in G_0^{(l)}}\left\{C_l(0,1)\frac{\pi_l\left(0|\vec{w}_j\right)}{\pi_l\left(1|\vec{w}_j\right)} - C_l(1,0)\right\} \tag{7.6}$$

for $l=1,2,\cdots,M$.

In (7.6), $G_0^{(l)}$ and $G_1^{(l)}$ denote sets of binary threshold signals such that $G_0^{(l)} = f_l^{-1}(0)$ and $G_1^{(l)}(1) = f_l^{-1}(1)$, where $f_l(\cdot)$ is the Boolean function used on level l. The proof of this theorem is similar to that of Theorem 7.4 and it is therefore omitted.

Finally, Theorem 7.4 can be generalized to the multilevel case as follows (assuming that the same $C(0,1)$ and $C(1,0)$ are used on all threshold levels and for all $\vec{w}_j \in \{0,1\}^b$).

Theorem 7.6: The optimal stack filter designed by the suboptimal routine remains the same iff α varies in the range

$$\max_{\vec{w}_j \in G_1} \left\{ C(0,1) \frac{\sum_{l=1}^{M} \pi_l(0,\vec{w}_j)}{\sum_{l=1}^{M} \pi_l(1,\vec{w}_j)} - C(1,0) \right\} < \alpha < \min_{\vec{w}_j \in G_0} \left\{ C(0,1) \frac{\sum_{l=1}^{M} \pi_l(0|\vec{w}_j)}{\sum_{l=1}^{M} \pi_l(1|\vec{w}_j)} - C(1,0) \right\}. \quad (7.7)$$

Some numerical bounds for α and α_l's will be given in the next section when we present several design examples.

VIII. Design Examples and Applications

In this section, several design examples of ROF's, stack filters, and GSF's (with $I = 0$) will be presented using the proposed optimization procedures, accompanied by some applications of stack filters and GSF's to image recovery from impulsive noise.

Let us start with the design examples of stack filters and GSF's. To do this, assume that irreducible discrete-time Markov chains with finite state spaces are used to model the signal and noise processes. In these examples, the Markov chain of the signal has six states in its state space, denoted by

$$Q_S = \{(0,a),(0,b),(1,a),(1,b),(2,a),(2,b)\}$$

and a state transition matrix given by

$$P_S = \begin{array}{c} \\ (0,a) \\ (0,b) \\ (1,a) \\ (1,b) \\ (2,a) \\ (2,b) \end{array} \begin{array}{c} \begin{array}{cccccc} (0,a) & (0,b) & (1,a) & (1,b) & (2,a) & (2,b) \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0.4 & 0 & 0 & 0 & 0 & 0.6 \end{pmatrix} \end{array}. \quad (8.1)$$

A nonlinear function operating on Q_S to produce the observed values for the signal is defined as $H_S((x,y)) = x, (x,y) \in Q_S$. Note that any signal generated by this Markov chain is a root signal to the window width three median filter. Note also that (8.1) does not generate all 3-valued roots of the window width three median filter.

TABLE I
Limiting and Conditional Probabilities

$\vec{w}$	000	001	010	011	100	101	110	111
$\pi_1(\vec{w})$	.0690	.1004	.0340	.1321	.1004	.0657	.1321	.3664
$\pi_1(0\|\vec{w})$	.9672	.8650	.3417	.0921	.8853	.3533	.0766	.0064
$\pi_2(\vec{w})$	.2732	.1263	.0606	.1109	.1263	.0452	.1109	.1467
$\pi_2(0\|\vec{w})$	.9928	.9142	.6756	.1391	.9303	.6013	.1207	.0184

The noise that corrupts the signal is assumed to be independent of the signal and is modeled by another irreducible discrete-time Markov chain [23] with a state space containing three states: $Q_N = \{X, Y, Z\}$ and a state transition matrix

$$P_N = \begin{array}{c} \\ X \\ Y \\ Z \end{array} \begin{array}{c} \begin{array}{ccc} X & Y & Z \end{array} \\ \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.8 & 0.1 & 0.1 \\ 0.6 & 0.3 & 0.1 \end{pmatrix} \end{array}. \quad (8.2)$$

The received process is generated by a nonlinear function $H_R(\cdot)$ operating on the signal and the noise Markov chains as follows:

$$H_R(((x,y);N)) = \begin{cases} x, & \text{if } N = X \\ 2, & \text{if } N = Y \\ 0, & \text{if } N = Z. \end{cases} \quad (8.3)$$

In (8.3), $((x,y);N)$ denotes a state in the state space of the received process, which is $Q_R = Q_S \times Q_N$. From (8.3), it is clear that the noise corrupting the signal is of impulsive nature.

Finally, we shall construct the window process whose states will be observed in the filter's window. For a window width b stack filter (or GSF with $I = 0$), there is a sequence of b samples in the filter's window at any time. The process which specifies the transitions between all the possible b-sample sequences (there are M^b of them for M-valued samples) is constructed from the elementary single sample definition of the received process.

Let Q_W denote the state space of the window process; then $Q_W = (Q_R)^b$. An expression for the state transition matrix P_W can be found in [2] but will not be used here to generate the limiting and conditional probabilities required to compute the cost coefficients. Since the signal and the noise are assumed to be independent, these probabilities can be computed using the procedure developed in [22]. These results are presented in Table I (the window width b is equal to 3).

From this table, we see that the conditional probabilities $\pi_l(0|\vec{w}_j)$'s ($l = 1$ and 2) satisfy the stacking properties at each level and from one level to another as well (see Definitions 4.1 and 5.2). It can also be checked that after adding up the probabilities $\pi_l(0,\vec{w}_j)$'s and $\pi_l(1,\vec{w}_j)$'s weighted by the corresponding cost coefficients, the resulting costs $E_j(0)$'s and $E_j(1)$'s also obey the stacking property (see Definition 4.2). Therefore, by Theorems 4.2, 4.3, and 5.3, we know that: 1) the optimal stack filters under the MAE criterion can be obtained by solving the suboptimal routine, 2) the two Boolean functions obtained at levels 1 and 2 are actually positive, and 3) these

TABLE II
WINDOW WIDTH THREE OPTIMAL STACK FILTERS AND GSF'S ($C(0,1)=1.0$)

$C(1,0)$	$f(\cdot)$	$f_1(\cdot)$	$f_2(\cdot)$	SFMAE	GSFMAE	MMAE
0.1	$x_1x_2x_3$	x_1x_2	$x_1x_2x_3$	.0636	.0617	.1156
0.5	$x_1x_2+x_2x_3$	$x_1x_2+x_2x_3$	$x_1x_2+x_2x_3$	.1318	.1318	.1520
1.0	$x_1x_2+x_2x_3+x_3x_1$	$x_2+x_1x_3$	$x_1x_2+x_2x_3+x_3x_1$	.1975	.1776	.1975
2.0	$x_2+x_1x_3$	$x_2+x_1x_3$	$x_2+x_1x_3$	.2569	.2553	.2884
5.0	$x_2+x_1x_3$	$x_2+x_1x_3$	$x_1+x_2+x_3$	.4036	.4036	.5613
10.0	$x_1+x_2+x_3$	$x_1+x_2+x_3$	$x_2+x_1x_3$	.6066	.5732	1.0161

SFMAE: MAE of stack filters
GSFMAE: MAE of GSF's
MMAE: MAE of median filters

TABLE III
NUMERICAL BOUNDS FOR α AND α_i'S ($C(0,1)=1.0$)

$C(1,0)$	α	$\%\cdot\alpha/C(1,0)$	α_1	$\%\cdot\alpha_1/C(1,0)$	α_2	$\%\cdot\alpha_2/C(1,0)$
0.1	−0.0902, 0.0071	−90.2%, 7.1%	−0.0170, 0.0014	−17.0%, 1.4%	−0.0813, 0.0373	−81.3%, 37.3%
0.5	−0.3719, 0.3331	−74.2%, 66.6%	−0.4170, 0.0191	−83.4%, 38.2%	−0.3627, 1.0082	−72.5%, 201.6%
1.0	−0.1669, 0.2524	−16.7%, 25.2%	−0.4537, 5.4074	−45.4%, 540.7%	−0.8627, 0.5082	−86.3%, 50.8%
2.0	−0.7476, 6.2910	−37.4%, 314.6%	−1.4537, 4.4074	−72.7%, 220.4%	−0.4918, 0.0826	−24.6%, 4.13%
5.0	−3.7476, 3.2910	−75.0%, 65.8%	−4.4537, 1.4074	−89.1%, 28.2%	−2.9174, 5.6550	−58.3%, 113.1%
10.0	−1.7090, 70.4762	−17.1%, 704.8%	−2.2816, 19.4878	−22.8%, 194.9%	−7.9174, 0.6550	−79.2%, 6.55%

two Boolean functions automatically form a stacking set, and therefore a GSF is produced that is optimal in the MAE sense.

Table II lists the designed stack filters and GSF's and their corresponding MAE's. For the sake of comparison, we also given in Table II the MAE's corresponding to the median filter.

The Boolean functions in Table II are shown in their MSP form (this can be done since these functions are all positive Boolean functions), where x_1, x_2, and x_3 denote the left bit, the middle bit, and the right bit in the filter's window, respectively. The output of the filter is assigned to the middle bit at each window position.

From Table II, we observe that, for all cases considered here, GSF's have smaller MAE's than stack filters, except for $C(1,0)=0.5$ and 5.0 where the GSF's reduce to a stack filter (which means that for these cases, the GSF's are homogeneous), while the median filter has the largest MAE's.

Before presenting some design examples of ROF's, let us give some numerical bounds for the α and α_l's (the ranges within which the cost coefficient is allowed to vary without changing the optimal solution, see the previous section) for the above examples. The results are presented in Table III.

Note that the ranges given in the above table are consistent with Table II. For instance, $C(1,0)=0.1$ and $C(1,0)=0.5$ produce different $f_1(\cdot)$'s in Table II; therefore, the difference $0.5-0.1=0.4$ must be outside the range of α_1 corresponding to $C(1,0)=0.1$ in Table III; another example, $C(1,0)=1$, 2, and 5 all produce the same $f_1(\cdot)$ in Table II; therefore, the difference $5.0-1.0=4.0$ must be inside the range of α_1 corresponding to $C(1,0)=1.0$ in Table III; and so on. Furthermore, note that the two α ranges in Table III corresponding to $C(1,0)=0.1$ and $C(1,0)=0.5$ are disjoint. This means

TABLE IV
WINDOW WIDTH THREE OPTIMAL ROF'S ($C(0,1)=1.0$)

$C(1,0)$	Filters
0−1/19	$r=4$
1/19−3/7	$r=3$
3/7−7/3	$r=2$
7/3−19	$r=1$
>19	$r=0$

TABLE V
WINDOW WIDTH FOUR OPTIMAL ROF'S ($C(0,1)=1.0$)

$C(1,0)$	Filters
0−1/167	$r=5$
1/167−5/37	$r=4$
5/37−1	$r=3$
1−37/5	$r=2$
37/5−167	$r=1$
>167	$r=0$

that there exists at least one stack filter (different from those corresponding to $C(1,0)=0.1$ and $C(1,0)=0.5$ in Table II) which is optimal in the minimum MAE sense and corresponds to a cost coefficient range in (0.1, 0.5). On the other hand, if two α ranges meet (the upper limit of one is the lower limit of the other, such as in the case of $C(1,0)=0.5$ and $C(1,0)=1.0$), then there does not exist another optimal stack filter different from the other two.

Next, some design examples of ROF's will be presented. To do this, we must first find the number of positive Boolean functions for a given window width and also the number of subsets of these functions that produce a 1 output for some specified binary states. This task is, however, not such an easy one for relatively large window widths [23]. Here, we only consider cases $b=3$ and $b=4$. Tables IV and V contain the designed minimum MAE ROF's for $b=3$ and $b=4$, respectively. Note

Lg=1.0 um

that all filters given in these tables are independent to the prior probability models of the signal and the noise processes, and therefore, they might be used directly in certain practical cases.

In Tables III and IV, $r = 0$ and $r = b + 1$ correspond to the 1 and the 0 filters, respectively.

Finally, a less artificial application is considered. In this application, stack filters and GSF's are designed for image recovery from impulsive noise. For this case, as the prior statistics of the signal and the noise processes are not available, the probabilities needed in the optimization procedure must thus be estimated assuming that training sequences exist. A detailed procedure for this estimation has been presented elsewhere; see, for instance, [26] and [27]. Here, we consider a stack filter and a GSF ($I = 0$) with 3×3 window mask. Using the method in [26] and [27], we found that the estimated probabilities obey the stacking properties defined in Definitions 4.2 and 5.1. Therefore, our suboptimal algorithms will produce optimal solutions. Figs. 3–6 show the original image, the noisy image, and the restored images by the optimal stack filter and GSF, respectively. Compared to the adaptive approach [6], which takes several tens of hours to perform the filtering task, our filtering process takes about half an hour for stack filters and about two hours for GSF's.

IX. Conclusion

In this paper, a method based on classical Bayes minimum-cost decision, providing a unified approach to the design of optimal ROF's, stack filters and GSF's under the MAE criterion was presented. It was shown that designing optimal ROF's or stack filters and GSF's is equivalent to the classical *a posteriori* or *a priori* Bayes minimum-cost decision, respectively. None of the design procedures presented in this paper requires the use of linear programming. The only computation involved is data comparisons whose number increases slower than exponentially as a function of the filter window width, thus rendering the optimization algorithm very efficient and suitable for large scale optimization problems in image and signal processing.

Sufficient conditions under which the suboptimal routines result in optimal solutions were given, and it was demonstrated that these conditions are natural and hold in most practical cases. These conditions have also been confirmed by the presented design examples.

The sensitivity of the designed ROF's, stack filters, and GSF's upon the cost coefficients was analyzed. The results indicate a source of robustness in the resulting optimal filters.

Design examples of optimal ROF's, stack filters, and GSF's using the proposed algorithms were given where the sufficient conditions given in the text have been confirmed. An application of stack filters and GSF's to image recovery from impulsive noise was also considered where the suboptimal routines proposed in this paper were successfully applied to find the optimal filters.

Appendix A
Proofs of Theorems 4.1, 4.3, and 5.2

Proof of Theorem 4.1

Consider two binary states $\vec{w}_i$ and $\vec{w}_j$. According to the suboptimal routine, we can obtain the outputs for $\vec{w}_i$ and $\vec{w}_j$ by checking whether or not the following inequality holds:

$$C(1,0)\pi\left(1|\vec{w}_j\right) < C(0,1)\pi\left(0|\vec{w}_j\right). \qquad \text{(A.1)}$$

If these two binary states are incomparable, there is no need to compare their outputs, since the decision outputs of two binary states are constrained by the stacking property if and only if one of the binary states stacks on top of the other. Therefore, let us focus on binary state pairs in which the two states possess the stacking property. Without loss of generality, we assume that $\vec{w}_j$ stacks on top of $\vec{w}_i$, i.e., $\vec{w}_i \geqslant \vec{w}_j$. Now, $\pi(0|\vec{w}_j) \geqslant \pi(0|\vec{w}_i)$ (and therefore $\pi(1|\vec{w}_j) \leqslant \pi(1|\vec{w}_i)$) guarantees that if (A.1) holds for $\vec{w}_i$ producing a 0 decision output, then (A.1) must also hold for $\vec{w}_j$ producing a 0 decision output. This means that $P_f(1|\vec{w}_i) \geqslant P_f(1|\vec{w}_j)$! The stacking property is automatically satisfied. Note that here the stacking property was not used to set the output for any binary state. Therefore, according to our previous statement, the suboptimal routine produces optimal solutions. □

Proof of Theorem 4.3

First, one can easily manipulate (4.3) to get

$$C\left(1,0,\vec{w}_j\right)\pi\left(1|\vec{w}_j\right) > C\left(0,1,\vec{w}_j\right)\pi\left(0|\vec{w}_j\right). \qquad \text{(A.2)}$$

Therefore, the suboptimal routine chooses $P_f(1|\vec{w}_j) = 1$ for any $\vec{w}_j$ satisfying (4.3). Let G_1 denote the set of these binary states. Similarly, one can show that for any binary state $\vec{w}_j$ violating (4.3), the suboptimal routine chooses $P_f(0|\vec{w}_j) = 1$. Let us denote by G_0 the set of these binary states. The hypothesis of Theorem 4.3 states that there are no binary states in G_1 stacking on top of any binary state in G_0; therefore, the stacking property holds among the decision outputs of all binary states. According to our previous statement, the suboptimal solution is optimal in the minimum MAE sense. □

Proof of Theorem 5.2

Suppose that for a binary threshold array $\vec{w}_j^I$, it is decided on level l, by comparing the expected costs associated with this array, that $P_{f_l}(0|\vec{w}_j^I) = 1$. This means that $E_j^{(l)}(0) < E_j^{(l)}(1)$. Because the expected costs along the threshold levels obey the stacking property, the following inequality must also hold:

$$E_i^{(m)}(0) < E_i^{(m)}(1)$$

$$\text{for } m = l+1, l+2, \cdots, M \text{ and } \forall \vec{w}_i^I \leqslant \vec{w}_j^I. \qquad \text{(A.3)}$$

This implies directly that $P_{f_m}(0|\vec{w}_i^I) = 1$. Here, we do not use the stacking constraints to set these outputs. Similarly, one can prove the other case, that is, if for some

$\vec{w}_j^I, P_{f_l}(1|\vec{w}_j^I) = 1$, then the stacking property of the expected costs implies that $P_{f_m}(1|\vec{w}_i^I) = 1$ for $m = l-1, l-2, \cdots, 1$ and for all $\vec{w}_i^I \geq \vec{w}_j^I$! The set of M Boolean functions automatically satisfies the stacking property. Therefore, according to our previous statement, the suboptimal solution is optimal in the minimum MAE sense. □

Appendix B
Proofs of Lemmas 6.1, 6.2, and 6.3 and Theorem 6.1

Proof of Lemma 6.1

Let us show that $\pi(1|\vec{w}_i) \geq \pi(1|\vec{w}_j)$, which is equivalent to (6.5). Suppose that we find a positive Boolean function $f(\cdot)$ such that $f(\vec{w}_j) = 1$. Since $\vec{w}_i \geq \vec{w}_j, f(\vec{w}_i) = 1$. This implies that $N_b^{(1)}(\vec{w}_j) \leq N_b^{(1)}(\vec{w}_i)$. Now, N_b is constant for a fixed b, hence, $\pi(1|\vec{w}_i) \geq \pi(1|\vec{w}_j)$. □

Proof of Lemma 6.2

Since $\vec{w}_i \cdot \vec{1}^t = \vec{w}_j \cdot \vec{1}^t$, both states contain the same number of 1's. Assume, without loss of generality, that $\vec{w}_i$ contains K 1's located at bit positions $n_1, n_2, \cdots, n_K$, and $\vec{w}_j$ contains K 1's at bit positions $m_1, m_2, \cdots, m_K$. We must show that the number of positive Boolean functions that produce a 1 for $\vec{w}_i$ is equal to the number of positive Boolean functions that produce a 1 for $\vec{w}_j$. Define the following one-to-one mapping Ψ: $\Psi(n_k) = m_k \forall k = 1, 2, \cdots, K$. Now, any positive Boolean function (in its MSP form) that produces a 1 for $\vec{w}_i$ must contain at least one term whose literals are some or all of the bits n_k's. Using the one-to-one mapping Ψ, we obtain another positive Boolean function which produces a 1 for $\vec{w}_j$. Therefore, for each positive Boolean function that produces a 1 for $\vec{w}_i$, there exists a positive Boolean function that produces a 1 for $\vec{w}_j$, there exists a positive Boolean function that produces a 1 for $\vec{w}_j$. □

Proof of Lemma 6.3

First, let us show that $\pi(0|\vec{w}_i) = \pi(1|\vec{w}_i^c)$ where $\vec{w}_i^c = \vec{1} - \vec{w}_i$ (i.e., the 0 bits of $\vec{w}_i$ become 1's and vice versa). This is equivalent to showing that $\pi(1|\vec{w}_i) + \pi(1|\vec{w}_i^c) = 1$. Using the definition given by (6.3), this equation becomes

$$N_b^{(1)}(\vec{w}_i) + N_b^{(1)}(\vec{w}_i^c) = N_b. \tag{B.1}$$

However, from the definition of $N_b^{(1)}(\vec{w}_i)$, it is easy to show that (A.1) is true for any $\vec{w}_i$. Therefore, we proved that $\pi(0|\vec{w}_i) = \pi(1|\vec{w}_i^c)$. Next, it is easy to see that $\vec{w}_j$ and $\vec{w}_i^c$ have the same number of ones, so we can apply the result of Lemma 6.2. This completes the proof. □

Proof of Theorem 6.1

As the conditional probabilities $\pi_l(0|\vec{w}_j)$'s and $\pi_l(1|\vec{w}_j)$'s are independent of l, the expected costs $E_j(0)$ and $E_j(1)$ for any binary state $\vec{w}_j$ reduce to

$$E_j(0) = C(1,0)\pi(1|\vec{w}_j)\sum_{l=1}^{M}\pi_l(\vec{w}_j) \tag{B.2}$$

and

$$E_j(1) = C(0,1)\pi(0|\vec{w}_j)\sum_{l=1}^{M}\pi_l(\vec{w}_j). \tag{B.3}$$

It is easy to check that these costs satisfy the stacking property defined in Definition 4.2. Therefore, the optimal solution under the MAE criterion can be obtained using the suboptimal routine proposed in Section IV, which does not require an LP and whose computational requirements are merely comparisons between the costs, i.e., checking if

$$C(1,0)\pi(1|\vec{w}_j) < C(0,1)\pi(0|\vec{w}_j). \tag{B.4}$$

On the other hand, according to Lemma 6.2, we know that all $\pi(0|\vec{w}_j)$'s or $\pi(1|\vec{w}_j)$'s are equal for the binary states having an equal number of ones. Using the notation $\Gamma_q (q = 0, 1, \cdots, b)$ introduced earlier, this means that $\pi(0|\vec{w}_j)$'s are the same for all $\vec{w}_j \in \Gamma_q$. Therefore, the decision outputs for the binary states in each Γ_q must also be the same. Obviously, such a solution is an ROF. □

Appendix C
Proofs of Theorems 7.1 and 7.4

Proof of Theorem 7.1

To prove necessity, suppose that, for some cost coefficients $C(0,1)$ and $C(1,0)$, the optimal ROF is the rth ROF. Then, for any $\vec{w}_i \cap \Gamma_{r-1}$, according to the optimization procedure, we have $C(1,0)\pi(1|\vec{w}_i) < C(0,1)\pi(0|\vec{w}_i)$, which is equivalent to

$$\pi(1|\vec{w}_i) < \frac{C(0,1)}{C(0,1)+C(1,0)}.$$

Similarly, we can show that for any $\vec{w}_j \in \Gamma_r$, we have

$$\pi(1|\vec{w}_j) > \frac{C(0,1)}{C(0,1)+C(1,0)}.$$

These two results imply that (7.1) holds.

To prove sufficiency, assume that $C(0,1)$ and $C(1,0)$ vary, but (7.1) still holds. According to the optimization procedure, (7.1) implies that the Boolean function representing the ROF will produce a 0 output for any $\vec{w}_i \in \Gamma_{r-1}$ and a 1 output for any $\vec{w}_j \in \Gamma_r$. Extending this result to the multilevel case, one can see that the solution is still the rth ROF. □

Proof of Theorem 7.4

To prove sufficiency, assume that (7.5) holds. Then, it is easy to check that for any $\vec{w}_i \in G_0$ and any $\vec{w}_j \in G_1$, the inequalities

$$(C(1,0)+\alpha)\pi(1|\vec{w}_i) < C(0,1)\pi(0|\vec{w}_i)$$

and

$$(C(1,0)+\alpha)\pi(1|\vec{w}_j) > C(0,1)\pi(0|\vec{w}_j)$$

hold. Therefore, the suboptimal routine proposed in Sec-

tion IV will produce the same solution as α varies according to (7.5). By just reversing the above proof process, one can prove the necessary part. □

REFERENCES

[1] 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.

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

[3] E. J. Coyle, "Rank order operators and the mean absolute error criterion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 63–76, Jan. 1988.

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

[5] M. Gabbouj and E. J. Coyle, "Minimum mean absolute error stack filtering with structural constraints," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 955–968, June 1990.

[6] J.-H. Lin, T. M. Sellke, and E. J. Coyle, "Adaptive stack filtering under the mean absolute error criterion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 938–954, June 1990.

[7] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, Jr., "Median filtering by threshold decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1183–1188, Dec. 1984.

[8] ___, "Threshold decomposition of multidimensional rank order operators," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 445–450, May 1985.

[9] J. Serra, *Image Analysis and Mathematical Morphology*. New York: Academic, 1982.

[10] P. A. Maragos and R. W. Schafer, "Morphological filters—Part I: Their set theoretic analysis and relations to linear shift invariant filters," and "Morphological filters—Part II: Their relations to median, order statistic and stack filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 1153–1184, Aug. 1987.

[11] K. Preston, Jr., "Ξ-filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 861–876, Aug. 1983.

[12] A. C. Bovik, T. S. Huang, and D. C. Munson, Jr., "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1342–1350, Dec. 1983.

[13] J.-H. Lin and E. J. Coyle, "Minimum mean absolute error estimation over the class of generalized stack filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 663–678, Apr. 1990.

[14] M. Gabbouj and E. J. Coyle, "Minimax optimization over the class of stack filter," in *Proc. SPIE Visual Commun. Image Processing '90*, pp. 143–154, Oct. 1990.

[15] T. A. Nodes and N. C. Gallagher, Jr., "Median filters: some modifications and their properties," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 739–746, Oct. 1982.

[16] E. N. Gilbert, "Lattice-theoretic properties of frontal switching functions," *J. Math. Phys.*, vol. 33, pp. 57–67, Apr. 1954.

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

[18] A. Wald, *Statistical Decision Functions*. New York: Wiley, 1947.

[19] T. S. Ferguson, *Mathematical Statistics: A Decision Theoretic Approach*. New York: Academic, 1967.

[20] L. Cooper and D. Steinberg, *Methods and Applications of Linear Programming*. Philadelphia, PA: W. B. Saunders, 1974.

[21] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[22] M. Gabbouj, "Estimation and structural based approach for the design of optimal stack filters," Ph.D. dissertation, School Elec. Eng., Purdue Univ., West Lafayette, IN, Dec. 1989.

[23] ___, "Optimal stack filter examples and positive Boolean functions," M.S. thesis, School Elec. Eng., Purdue Univ., West Lafayette, IN, Dec. 1986.

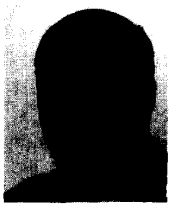
[24] B. Zeng and Y. Neuvo, "Design of suboptimal/optimal stack filters under the mean absolute error criterion," presented at the IASTED Int. Conf. Signal Process. Digital Filtering, June 1990.

[25] B. Zeng, M. Gabbouj, and Y. Neuvo, "Design of optimal generalized stack filters under the mean absolute error criterion," in *Proc. IEEE Workshop Visual Signal Process. and Commun.*, June 1991.

[26] B. Zeng, H. Zhou, and Y. Neuvo, "Synthesis of optimal detail-restoring stack filters for image processing," in *Proc. 1991 IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 1991.

[27] ___, "Synthesis of optimal stack and parallel stack filters under the mean absolute error criterion," submitted to *IEEE Trans. Signal Processing*.

Bing Zeng, for a photograph and biography, please see page 601 of the June 1991 issue of this TRANSACTIONS.



Moncef Gabbouj (S'85–M'90) received the B.S. degree in electrical engineering in 1985 from Oklahoma State University, Stillwater, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN in 1986 and 1989, respectively.

Since 1990, he has been with the Research Institute of Information Technology, Tampere, Finland, where he is currently a Senior Research Scientist. He also holds a teaching position in the Signal Processing Laboratory at Tampere University of Technology, Tampere, Finland. His research interests include nonlinear signal and image processing, mathematical morphology, neural networks, and artificial intelligence.

Dr. Gabbouj is the Director of the International University Program in Digital Signal Processing in the Signal Processing Laboratory at Tempere University of Technology. He is a member of Eta Kappa Nu and Phi Kappa Phi. Dr. Gabbouj was co-recipient of the Myril B. Reed Best Paper Award from the 32nd Midwest Symposium on Circuits and Systems.

Yrjö Neuvo (S'70–M'74–SM'82–F'89), for a photograph and biography, please see page 601 of the June 1991 issue of this TRANSACTIONS.