A training framework for stack and Boolean filtering – Fast optimal design procedures and robustness case study

3 authors:

Ioan Tabus
Tampere University
193 PUBLICATIONS **2,022** CITATIONS

SEE PROFILE

Doina Petrescu
Motorola Mobility
8 PUBLICATIONS **69** CITATIONS

SEE PROFILE

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

SEE PROFILE

A Training Framework for Stack and Boolean Filtering—Fast Optimal Design Procedures and Robustness Case Study

Ioan Tăbuş, Doina Petrescu, and Moncef Gabbouj, *Senior Member, IEEE*

Abstract— A training framework is developed in this paper to design optimal nonlinear filters for various signal and image processing tasks. The targeted families of nonlinear filters are the Boolean filters and stack filters. The main merit of this framework at the implementation level is perhaps the absence of constraining models, making it nearly universal in terms of application areas. We develop fast procedures to design optimal or close to optimal filters, based on some representative training set. Furthermore, the training framework shows explicitly the essential part of the initial specification and how it affects the resulting optimal solution. Symmetry constraints are imposed on the data and, consequently, on the resulting optimal solutions for improved performance and ease of implementation.

The case study is dedicated to natural images. The properties of optimal Boolean and stack filters, when the desired signal in the training set is the image of a natural scene, are analyzed. Specifically, the effect of changing the desired signal (using various natural images) and the characteristics of the noise (the probability distribution function, the mean, and the variance) is analyzed. Elaborate experimental conditions were selected to investigate the robustness of the optimal solutions using a sensitivity measure computed on data sets. A remarkably low sensitivity and, consequently, a good generalization power of Boolean and stack filters are revealed.

Boolean-based filters are thus shown to be not only suitable for image restoration but also robust, making it possible to build libraries of "optimal" filters, which are suitable for a set of applications.

I. INTRODUCTION

THE last few decades have brought to the attention of the signal processing community a new field of research—nonlinear digital filtering—which has proved to be very effective when dealing with data corrupted by noise having a heavy tailed distribution. This field is continuously growing, mainly in two directions: finding new classes of filters, which possess useful properties, and developing new

I. Tăbuş is with the Signal Processing Laboratory, Tampere University of Technology, SF-33101 Tampere, Finland, on leave from the Department of Control and Computers, Polytechnic University of Bucharest, R-77202 Bucharest, Romania.

D. Petrescu is with the Signal Processing Laboratory, Tampere University of Technology, SF-33101 Tampere, Finland, on leave from the Department of Electronics and Telecommunication, Polytechnic University of Bucharest, R-77202 Bucharest, Romania.

M. Gabbouj is with the Signal Processing Laboratory, Tampere University of Technology, SF-33101 Tampere, Finland (e-mail: Gabbouj@fosco.pori.tut.fi).

design techniques. The class of stack filters [18] is one of the recent and major classes of nonlinear filters that is large enough to contain filters with various types of behavior from which one can select the best suited for the given application.

Several optimal design techniques were proposed for selecting the best filter in the class of stack filters or in other related classes, according to the MAE criterion [3]–[5], [6], [20], founding a well-established theory, which will be quoted in the sequel as the classical theory. In almost all of these classical papers, the approach taken was an analytical one, based on the availability of signal and noise models. Although this approach gives insight into the way in which the modification of some model parameters affects the optimal solution, the link between the final solution and the initial data of the filtering problem is made only in an indirect way, requiring some model assumptions and setting of some parameters that are not natural in some practical applications. For example, in image filtering applications, the mean absolute error (MAE) is not well defined since data and noise can hardly be supposed jointly stationary.

Over the past few years in the engineering literature, the training framework for the "optimal design" problem has become increasingly popular due to the availability and widespread use of "neural network" type solutions for many practical problems. This framework is mainly the same as the one in which estimation theory is developed. There, an optimal problem is roughly expressed as follows: "Given the examples in the form of a training data set, find the model that matches best the examples." However, the "training" concept emphasizes more some practical meaningful ingredients as the sufficiency of the training set and the generalization power of the optimal solution (there will be more about these concepts latter).

The estimation approach was used in [21] to implement an optimal solution starting from the given data. However, the solutions obtained are merely approximations of some "ideal optimal solution" defined in the classical sense, i.e., minimizing the *expectation* of the estimation error. An adaptive approach to optimal stack filtering was proposed in [11], where the procedure acts similarly to LMS algorithms, training the filter at each new pair of (input, target) data such as to preserve the stacking property and to advance toward an optimum solution.

Our approach to optimal filter design is similar with respect to the optimization criterion to the adaptive approach [11], but

the optimal solution is derived, as in "training" methods, after preprocessing the whole training set to extract the relevant information (in the form of cost coefficients). Based on this compressed information, we show that the optimal Boolean filter, which can be readily computed, can be used in a projection procedure for the very fast derivation of the optimal stack filter.

In the first part of this paper, we shall define the training framework for the general class of Boolean filters (stack filters in particular).

The definition and architectures of stack filters are first reviewed in the next section. In Section III, we first define the training $(\mathcal{T})$ approach to optimal design. We transform the integer domain optimization problem into a linear programming (LP) problem, where the costs coefficients are functions of the training set. We establish properties of the cost coefficients, and in order to compute their mean, we introduce statistical assumptions leading to a new formulation of the optimal problem, which has been denoted the $(\mathcal{D}, \Phi)$ approach, based on training data (where $\mathcal{D}$ is the target image) and on statistical assumptions (Φ-probability distribution functions of the independent noise). The connections between the optimal Boolean filter and the optimal stack filter are derived, resulting in a projection rule that allows a very fast design of the stack filter starting from the optimal Boolean filter. We describe in Section IV new and fast suboptimal and optimal design procedures and exemplify one implementation in a matrix-based computing environment. Optimal filter design under symmetry constraints is considered in Section V. Section VI presents robustness case studies dealing entirely with natural images. The properties of $\mathcal{T}$- and $(\mathcal{D}, \Phi)$-optimal Boolean and stack filters are analyzed when $\mathcal{D}$, which is the desired signal, is the image of a natural scene. Assuming that the noise is stationary, we analyze the effect of changing the desired signal (using various natural images) and the characteristics of the noise (the probability distribution function, the mean, and the variance). Finally, the computational performances of the FASTAF procedure are compared with other well-known procedures for optimal stack filter design.

II. STACK FILTERS: DEFINITIONS AND ARCHITECTURES

In this section, we review the stack filtering architecture and introduce the basic notation that will be used in the sequel.

We will assume that the signals to be processed take integer values in the set $\{0, \cdots, M\}$. When referring to a signal value, the following conventions will be used: capital letters denote integer valued signals, lowercase letters denote binary valued signals, and underlined letters stand for vector-valued variables.

The thresholding at level m operator will be denoted by T_m, and the binary value obtained by thresholding an integer variable at level m will be denoted by the variable name with small letters, with superscript m, i.e.,

$$i^m = T_m(I) = \begin{cases} 1, & \text{if } I \geq m; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Using the threshold decomposition, one can accomplish a (redundant) binary codification of an integer value I into the

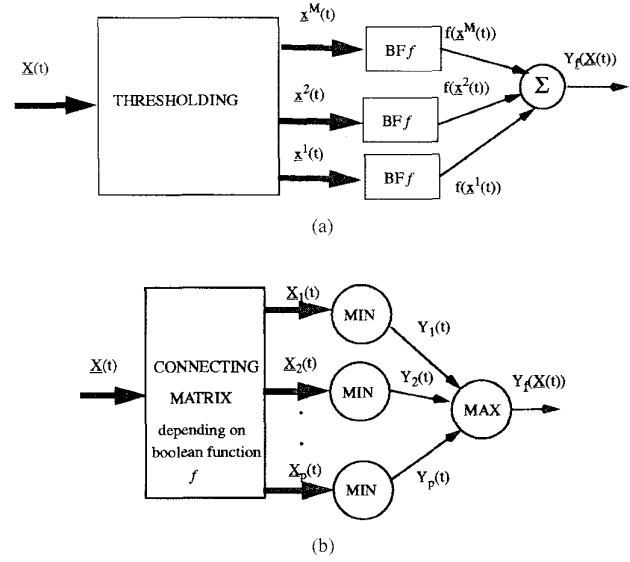Fig. 1. Implementation of stack filtering in (a) the binary domain, and in (b) the integer domain.

column binary vector $[i^M \ i^{M-1} \cdots i^1]^T$, which can be decoded back into the integer value by simply adding its elements: $I = \sum_{m=1}^{M} i^m$.

Let $X(t)$ denote the signal that must be filtered and $D(t)$ the original signal (i.e., the signal $D(t)$ is not known during the filtering stage but only a noisy version of it—$X(t)$—is available to be processed by the filter; however, during the design stage, we must consider as known the "ideal" signal $D(t)$ as well). At time t, the filter has available a small number N of samples of the signal $X(\cdot)$. The shape of the window $\underline{X}(t)$ around the current input sample $X(t)$, which is processed by the filter, is also considered to be given. For 1-D signals, the window is built in the following way:

$$\underline{X}(t)$$
$$= [X(t - N_1) \cdots X(t - 1) \ X(t) \ X(t + 1) \cdots X(t + N_2)]^T$$
$$= [X_1(t) \cdots X_N(t)]^T \tag{2}$$

where the length of the vector $\underline{X}(t)$, in this case, is $N = N_1 + N_2 + 1$. In image processing applications, we consider that the way to arrange the samples inside the vector $\underline{X}(t)$ is known and fixed.

The output of a stack filter can be computed in two equivalent ways. The binary domain computation is represented in Fig. 1(a): First, the input window $\underline{X}(t)$ is decomposed (applying to its elements the threshold operator at levels $1, \cdots, M$) into M binary windows $\underline{x}^1(t), \cdots, \underline{x}^M(t)$; then, each binary window is processed by a Boolean filter $f(\underline{x})$, which satisfies the stacking constraint [18]

$$\underline{x} \geq \underline{z} \Rightarrow f(\underline{x}) \geq f(\underline{z}) \tag{3}$$

and the binary values obtained $f(\underline{x}^1(t)), \cdots, f(\underline{x}^M(t))$ are summed up to obtain the output of the filter $Y(t) = Y_f(\underline{X}(t))$.

The inequality $\underline{x} \geq \underline{z}$ between two vectors holds if all entries in $\underline{x}$ are greater than or equal to the corresponding entries in $\underline{z}$.

A Boolean function satisfying the stacking constraint is called a positive Boolean function and has the additional property that it can be expressed in a unique sum of products form (disjunctive normal form) with all the variables appearing only uncomplemented [8]. The processing inside a stack filter can be represented in an equivalent form in the integer domain as shown in Fig. 1(b). The equivalence between the binary domain and the integer domain processing can be easily derived using the correspondence between the min and max operations in the integer domain with the AND and OR operations in the binary domain. It becomes clear that the connecting matrix in Fig. 1(b) selects the variables that appear in the minimal disjunctive form of the positive Boolean function. As an example, for the PBF

$$f(x_1, x_2, x_3) = x_1 x_2 + x_2 x_3 + x_1 x_3 \qquad (4)$$

the connecting matrix in Fig. 1(b) must select from its integer input window $\underline{X}(t) = [X(t-1) \ X(t) \ X(t+1)]^T$ the output windows $\underline{X}_1(t) = [X(t-1) \ X(t)]^T, \underline{X}_2(t) = [X(t) \ X(t+1)]^T$, and $\underline{X}_3(t) = [X(t-1)X(t+1)]^T$.

All N-length binary vectors $\underline{v}_i = [v_{i_1} \cdots v_{i_N}]$, $i = 1, \cdots, 2^N$ can be enumerated in the natural order (such that $(i-1) = v_{i_1} 2^{N-1} + v_{i_2} 2^{N-2} + \cdots + v_{i_N} 2^0$—the vector indexed with i is the binary representation of $i - 1$).

A Boolean function $f(x_1, x_2, \cdots, x_N)$ that depends on N logical variables $x_1, x_2, \cdots, x_N$ can be defined in the form of a vector (or, similarly, a truth table form)

$$\underline{f} = [f(\underline{v}_1) f(\underline{v}_2) \cdots f(\underline{v}_{2^N})]^T. \qquad (5)$$

The vectors $\underline{v}_i$, for which $f(\underline{v}_i) = 1$, are called "units of $\underline{f}$." The Hamming weight of a vector $\underline{v}_i$ (the number of 1's in $\underline{v}_i$) is denoted $w_H(\underline{v}_i)$. The set of vectors $\underline{v}_i$ with $f(\underline{v}_i) = 1$ is denoted $V_1(\underline{f})$ (onset), whereas the set of vectors with $f(\underline{v}_i) = 0$ is denoted $V_0(\underline{f})$ (offset). We will usually define a Boolean function by specifying the way to construct its onset $V_1(\underline{f})$.

The set of all vectors that are greater than or equal to (stacks under) a given vector $\underline{v}_i$ is denoted $W_{\text{down}}(\underline{v}_i) = \{\underline{v}_j \mid \underline{v}_j \geq \underline{v}_i\}$; similarly, the set of vectors less than or equal to (stacks on top of) $\underline{v}_i$ is denoted $W_{\text{up}}(\underline{v}_i) = \{\underline{v}_j \mid \underline{v}_j \leq \underline{v}_i\}$. Restricting the above sets to vectors differing of $\underline{v}_i$ in only one position, we obtain $H_{\text{down}}(\underline{v}_i) = \{\underline{v}_j \mid \underline{v}_j \geq \underline{v}_i$ and $w_H(\underline{v}_j) = w_H(\underline{v}_i) + 1\}$, $H_{\text{up}}(\underline{v}_i) = \{\underline{v}_j \mid \underline{v}_j \leq \underline{v}_i$ and $w_H(\underline{v}_j) = w_H(\underline{v}_i)-1\}$. A minimal term of the positive Boolean function $\underline{f}^+$ is a vector $\underline{v}_i \in V_1(\underline{f}^+)$ such that there is no vector $\underline{v}_j \in V_1(\underline{f}^+)$ such that $W_{\text{down}}(\underline{v}_i) \subset W_{\text{down}}(\underline{v}_j)$. One possible parametrization of the stack filter class is through the vector $\underline{W} = [i \quad j \quad \cdots \quad k]$, where $i, j, \cdots, k$ are indexes of the vectors $\underline{v}_i, \underline{v}_j, \cdots, \underline{v}_k$, which are minimal terms of the positive Boolean function, e.g., the parameters for (4) are $\underline{W} = [4 \ 6 \ 7]$.

Armed with this notation and the filter architecture, we shall, in the next section, present the training approach to optimal design.

III. TRAINING APPROACH TO OPTIMAL DESIGN

A. Problem Statement

We shall consider a training framework in which "representative" sets for input signal and desired signal are considered known and state the problem to refer to a generic family of nonlinear filters $\mathcal{C}$:

Problem 1) Optimal Design of Nonlinear Filters—Training Framework: Given

- The training set $\mathcal{T} = (\mathcal{D}, \mathcal{X})$:
  - the input set $\mathcal{X} = \{\underline{X}(t)\}_{t=1}^T$ (the integer signal $X(t)$ is already arranged in windows)
  - the desired output set $\mathcal{D} = \{D(t)\}_{t=1}^T$
- the filter class $\mathcal{C}$ (a member of this class being identified by the parameter vector $\underline{W}$)

find the filter parameters $\underline{W}^*$ that minimize the criterion

$$J_T(\underline{W}) = \frac{1}{T} \sum_{t=1}^T \mid D(t) - Y_{\underline{W}}(\underline{X}(t)) \mid. \qquad (6)$$

No restrictive hypothesis concerning the data was imposed in this training formulation (e.g., stationarity is not required). However, in order to obtain results useful for other sets of data, we require that the selected set of data be "representative." This issue will be elaborated upon in the following.

The same type of criterion was considered in [10] and [11], but there, in order to define an optimal solution, some statistical assumptions about the data were imposed.

The criterion MAE

$$J_s(\underline{W}) = E[|D(t) - Y_{\underline{W}}(\underline{X}(t))|] \qquad (7)$$

can be seen as a particular case of the criterion (6), which is obtained, e.g., under ergodicity hypothesis, for large training sets

$$J_T(\underline{W}) \xrightarrow[T \to \infty]{} J_s(\underline{W}).$$

B. Reducing the Optimal Design Problem to a Linear Programming Problem

We want to retain in our framework the least restriction regarding the filter class and then, after dealing with the most general problem, to adapt it to different particular cases. Hence, we introduce the following definition.

Definition 3.1: A filter belonging to a class $\mathcal{C}$, with parameters $\underline{W}$, possesses the *threshold decomposition (TD) property* if there is a Boolean function $f_{\underline{W}}$ such that for every window $\underline{X}(t)$

$$Y_{\underline{W}}(\underline{X}(t)) = \sum_{m=1}^M f_{\underline{W}}(\underline{x}^m(t)). \qquad (8)$$

All filters possessing the TD property will be called *Boolean filters* for the obvious reason that their processing can be expressed in the decomposed form presented in Fig. 1(a) and that they are completely specified given the function $f_{\underline{W}}$, which must be a Boolean function. The class of Boolean filters contains as subclasses stack filters, weighted-order statistic

filters, weighted median filters, and morphological filters with flat structuring elements, but it does not reduce to any of them (for a comprehensive picture of the representation properties (including integer domain representations) and implementation aspects for this filter class, see [9]). There are, however, some important classes of filters that do not possess the TD property such as linear filters, L-filters, and variable rank order statistic filters [16] for which other methods different from the one presented here must be considered for the optimal design.

We next state and prove the result that relates the integer domain expression of criterion (6) to a binary domain expression. First, define for all binary vectors $\underline{v}_i$ the following:

$\mathcal{M}_0(\underline{v}_i)$—the set of all pairs (t, m)

for which $d^m(t) = 0$ and $\underline{x}^m(t) = \underline{v}_i$;   (9)

$\mathcal{N}_0(\underline{v}_i) = \mathrm{Card}(\mathcal{M}_0(\underline{v}_i))$

    × (Card denotes the cardinality of a set);   (10)

$\mathcal{M}_1(\underline{v}_i)$—the set of all pairs (t, m)

for which $d^m(t) = 1$ and $\underline{x}^m(t) = \underline{v}_i$;   (11)

$\mathcal{N}_1(\underline{v}_i) = \mathrm{Card}(\mathcal{M}_1(\underline{v}_i));$ (12)

$$C_0 = \frac{1}{T} \sum_{i=1}^{2^N} \mathcal{N}_1(\underline{v}_i) \tag{13}$$

$$c(\underline{v}_i) = \frac{1}{T}(\mathcal{N}_0(\underline{v}_i) - \mathcal{N}_1(\underline{v}_i)). \tag{14}$$

Lemma 3.1: If the filter with parameters $\underline{W}$ possesses the threshold decomposition property, then we have the following:

1)

$$J_T(\underline{W}) \le J_T^b(\underline{W}) = C_0 + \sum_{i=1}^{2^N} c(\underline{v}_i) f_{\underline{W}}(\underline{v}_i) \tag{15}$$

where J_T^b denotes the cost function defined in the binary domain.

2) The equality

$$J_T(\underline{W}) = J_T^b(\underline{W}) \tag{16}$$

holds for *any* training set $\mathcal{T} = \{\underline{X}(t), D(t)\}_{t=1}^T$ *iff* the Boolean filter is a stack filter.

Proof: See the Appendix.   □

Using the above Lemma, we can restate the optimization Problem 1 in the following form:

Problem 2): Given

- the coefficients $\{c(\underline{v}_i)\}_{i=1}^{2^N}$ and C_0, compressing the information from $\{\underline{X}(t)\}_{t=1}^T$, and $\{D(t)\}_{t=1}^T$, according to (13) and (14)
- the filter class $\mathcal{C}$ with a fixed parametrization in terms of the vector $\underline{W}$

find the filter parameters $\underline{W}^*$ that minimize the criterion

$$J_T^b(\underline{W}) = C_0 + \sum_{i=1}^{2^N} c(\underline{v}_i) f_{\underline{W}}(\underline{v}_i) = C_0 + \underline{C}^T \underline{f}_{\underline{W}}. \tag{17}$$

In the rest of the paper, we denote *optimal Boolean filter* to be the filter minimizing (17) (which may be suboptimal with respect to criterion (6)), whereas the *optimal stack filter*

will be optimal with respect to both (17) and (6)). Note that the compression rate achieved using the coefficient set instead of the original training data depends on N and T. For small values of N, the compression is effective (true for most practical cases), whereas for large window sizes, there is no more compression.

C. Properties of the Cost Coefficients and the $(\mathcal{D}, \Phi)$ Approach to Optimal Design

1) Fast Computation of the Cost Coefficients: The computation of the cost coefficients using (9)–(14) is very simple, but it is inefficient in that at each t, one must perform the threshold decomposition at all levels $m = 1, \cdots, M$. Since inside the processing window $\underline{X}(t)$ there are at most N different values, let us denote them in increasing order $X_{(1)}(t), \cdots, X_{(N)}(t)$ and denote by convention $X_{(0)} = 0$ and $X_{(N+1)} = M$. It is obvious that the thresholded input window $\underline{x}^m(t)$ is the same for all thresholding levels $m \in \{X_{(i)}, X_{(i)} + 1, \cdots, X_{(i+1)}\}$. Thus, if one performs the ordering of the integer values in the input window $\underline{X}(t)$, one can save in the required number of threshold operations $\underline{x}^m(t) = T_m(\underline{X}(t))$. We obtain the following fast implementation of the computations in (9)–(12), which will serve as a first stage in the optimal design procedure:

Stage I: For all $t = 1, \cdots, T$, do the following.

I.1) Order the entries of $\underline{X}(t)$ to obtain the increasing sequence $X_{(1)}, \cdots, X_{(N)}$, and find r, which is the index such that $X_{(r)} \le D(t) \le X_{(r+1)}$.

I.2) For $i = 1, \cdots, r$, compute $\underline{v} = T_{X_{(i-1)}}(\underline{X}(t))$, and update

$$\mathcal{N}_1(\underline{v}) \leftarrow \mathcal{N}_1(\underline{v}) + X_{(i)} - X_{(i-1)}.$$

For $i = (r + 2), \cdots, (N + 1)$, compute $\underline{v} = T_{X_{(i-1)}}(\underline{X}(t))$, and update

$$\mathcal{N}_0(\underline{v}) \leftarrow \mathcal{N}_0(\underline{v}) + X_{(i)} - X_{(i-1)}.$$

For $i = r+1$, compute $\underline{v} = T_{X_{(i-1)}}(\underline{X}(t))$, and update

$$\mathcal{N}_0(\underline{v}) \leftarrow \mathcal{N}_0(\underline{v}) + X_{(i)} - D(t);$$
$$\mathcal{N}_1(\underline{v}) \leftarrow \mathcal{N}_1(\underline{v}) + D(t) - X_{(i-1)}.$$

The above loops will not be performed if the first limit is greater than the second. $\mathcal{N}_0$ and $\mathcal{N}_1$ are initialized with 0 for all $\underline{v}_i$. The computation of the cost coefficients is the most costly stage in the procedures for optimal stack filter design, and thus, we must improve the efficiency by all means, e.g., by keeping track at time t of the entries already sorted increasingly at step $t - 1$.

2) Properties of the Cost Coefficients: In this section, we derive an integer domain expression for the cost coefficients, and then, we compute the statistical mean of the cost coefficients for some precise assumptions about the generation of the training set, leading to a new formulation of the optimality problem.

Denote $\min_{\underline{v}_i}$, which is the stack filter that selects the minimum of those pixels in the processing window corresponding

	v_1 [0 0 0]	v_2 [0 0 1]	v_3 [0 1 0]	v_4 [0 1 1]	v_5 [1 0 0]	v_6 [1 0 1]	v_7 [1 1 0]	v_8 [1 1 1]
v_1 [0 0 0]	1	0	0	0	0	0	0	0
v_2 [0 0 1]	-1	1	0	0	0	0	0	0
v_3 [0 1 0]	-1	0	1	0	0	0	0	0
v_4 [0 1 1]	1	-1	-1	1	0	0	0	0
v_5 [1 0 0]	-1	0	0	0	1	0	0	0
v_6 [1 0 1]	1	-1	0	0	-1	1	0	0
v_7 [1 1 0]	1	0	-1	0	-1	0	1	0
v_8 [1 1 1]	-1	1	1	-1	1	-1	-1	1

Fig. 2. (Left) Recursion from n to $n+1$ for the matrix $K(L)$; (Right) matrix L for $N=3$ (note that $K(i,j) = |L(i,j)|$).

to nonzero entries in v_i. The PBF of this filter is g_{v_i} with the onset $V_1(g_{v_i}) = W_{\text{down}}(v_i)$ (and, by convention, we take $g_{v_1} \equiv \underline{1}$).

Lemma 3.2: The cost coefficients have the following representation in the integer domain:

$$C_0 = \frac{1}{T}\sum_{t=1}^{T} D(t) \tag{18}$$

$$c(v_i) = \frac{1}{T}\sum_{t=1}^{T}\sum_{v_j \in W_{\text{down}}(v_i)} (-1)^{w_H(v_i)+w_H(v_j)}$$
$$\times \left(\left| D(t) - \min_{v_j}(\underline{X}(t)) \right| - D(t) \right). \tag{19}$$

Proof: Using (16) and (17), we can write

$$J_T(g_{v_i}) = \frac{1}{T}\sum_{t=1}^{T} |D(t) - \min_{v_i}(\underline{X}(t))| \tag{20}$$

$$= C_0 + \underline{C}^T g_{v_i} = C_0 + \sum_{v_j | v_j \geq v_i} c(v_j). \tag{21}$$

If we arrange the vectors g_{v_i} for all $i = 1, \cdots, 2^N$ as the columns of a matrix, we obtain the matrix K

$$K = \begin{bmatrix} g_{v_1} & \cdots & g_{v_{2^N}} \end{bmatrix} \tag{22}$$

which is shown, for $N = 3$, in Fig. 2. The elements of this matrix have the property

$$K(i,j) = 1 \text{ iff } v_i \geq v_j \tag{23}$$

and thus, K is the conjunctive matrix ([1], p. 13), which is defined by the recursion $K_{N+1} = \begin{bmatrix} K_N & 0 \\ K_N & K_N \end{bmatrix}$, initialized with $K_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, [2]. From (21) and (22), it follows that

$$\begin{bmatrix} (J_T(g_{v_1}) - C_0) & \cdots & (J_T(g_{v_{2^N}}) - C_0) \end{bmatrix}$$
$$= \underline{C}^T \begin{bmatrix} g_{v_1} & \cdots & g_{v_{2^N}} \end{bmatrix} = \underline{C}^T K. \tag{24}$$

This formula can be inverted to give the values of the cost coefficients in terms of (20)

$$\underline{C}^T = \begin{bmatrix} (J_T(g_{v_1}) - C_0) & \cdots & (J_T(g_{v_{2^N}}) - C_0) \end{bmatrix} K^{-1}. \tag{25}$$

However, the matrix $L = K^{-1}$ also has a simple structure and can be computed recursively ([1], p. 13) as

$$L_{N+1} = \begin{bmatrix} L_N & 0 \\ -L_N & L_N \end{bmatrix} \tag{26}$$

starting from $L_1 = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$. We show now that the element $L_N(j,i)$ is nonzero, $L_N(j,i) = (-1)^{w_H(v_i)+w_H(v_j)}$, iff $v_j \geq v_i$. The N-dimensional binary vectors $v_j \geq v_i$ can be obtained starting from $(n = 1)$-dimensional vectors, $v_j^n \geq v_i^n$ by concatenating recursively $v_j^{n+1} = [0 \ v_j^n]$ or $v_j^{n+1} = [1 \ v_j^n]$. From Fig. 2, one can see that the inversion of sign occurs for $L_{n+1}([1 \ v_j^n], [0 \ v_i^n]) = -L_n(v_j^n, v_i^n)$ when a unit enters the first position of v_j^{n+1} and a zero enters the first position of v_i^{n+1}. Thus, the value of $L_N(j,i)$ will be given by the number of units in v_j exceeding those of v_i, $L_N(j,i) = (-1)^{(w_H(v_j)-w_H(v_i))} = (-1)^{w_H(v_j)+w_H(v_i)}$. Thus, from (25), the inversion formula for (20) results:

$$c(v_i) = \sum_{j=1}^{2^N} (J_T(g_{v_j}) - C_0)L(j,i)$$
$$= \sum_{v_j | v_j \geq v_i} (J_T(g_{v_j}) - C_0)(-1)^{w_H(v_i)+w_H(v_j)}. \tag{27}$$

This formula immediately gives the computational expression of $c(v_i)$ in the integer domain (19). $\square$

The cost coefficients we introduced in (13) and (14) are functions of the training set. If we assume that this training set was generated by a statistical model—more precisely, by a Markov chain—the statistical mean of the cost coefficients has been computed in [3] and has been shown to be identical to the asymptotic values of the cost coefficients [13]. In the following, we propose a different statistical scenario, assuming that we have given as initial specification of the design problem the desired image and the probability distribution

of the independent errors [15]. In this scenario, one can compute the mean of the cost coefficients starting from the integer representation of the cost coefficients (19) since the output distribution of $\min_{\underline{v}_j}$ for independent (but possibly not identically) distributed inputs is well known [19]. However, we will deduce the mean coefficients in an indirect but simpler manner, where we will also have the merit to emphasize the new setting of the optimal problem, which is different from Problem 1. Making full use of the statistical assumptions, the optimal stack filter design based on the mean cost coefficients is expected to be very robust in that the optimal filter will perform close to optimal for many noise realizations. This property is further illustrated in the experimental section.

Problem 3) $(\mathcal{D}, \Phi)$—*Optimal Stack Filter Design:* (The target signal is deterministic) *Given*

- the target deterministic signal $\mathcal{D} = \{D(t)\}_{t=1}^{T}$ (with integer values $D(t) \in \{0, \cdots, M\}$)
- the set of distribution functions (which may depend on t) of the discrete random variables $e(t)$, $\Phi = \{\text{Prob}(e(t) \leq u)\}_{t=1}^{T} = \{\phi_{e(t)}(u)\}_{t=1}^{T}$ (where $e(t)$ is the additive noise process)

find the positive Boolean function f associated with the stack filter that minimizes the criterion

$$J_{\mathcal{D}, \Phi}(\underline{f}) = \frac{1}{T} \sum_{t=1}^{T} E|D(t) - Y_{\underline{f}}(\underline{X}(t))| \qquad (28)$$

where $\underline{X}(t) = \underline{D}(t) + \underline{e}(t)$ denotes the input of the stack filter at "moment" t, $\underline{X}(t) = [X(t - N_1), \cdots, X(t + N_2)]^{T}$ is a sliding window of length $N = N_1 + N_2 + 1$, and $Y_{\underline{f}}(\underline{X}(t))$ denotes the output of the stack filter.

Lemma 3.3: If the random variables $e(t)$ and $e(s)$ are independent for all $t \neq s$, then criterion (28) can be computed as follows:

$$J_{\mathcal{D}, \Phi}(\underline{f}) = C_0 + \sum_{i=1}^{2^N} c^{(\mathcal{D}, \Phi)}(\underline{v}_i) f(\underline{v}_i) \qquad (29)$$

with C_0 given by (18) and

$$c^{(\mathcal{D}, \Phi)}(\underline{v}_i) = \frac{1}{T} \sum_{t=1}^{T} \left[\sum_{m=D(t)}^{M-1} \prod_{j=1}^{N} \phi_{X(t+j-1-N_1)}(m)^{1-v_{i_j}} \right.$$
$$\times \left[1 - \phi_{X(t+j-1-N_1)}(m)\right]^{v_{i_j}}$$
$$- \sum_{m=0}^{D(t)-1} \prod_{j=1}^{N} \phi_{X(t+j-1-N_1)}(m)^{1-v_{i_j}}$$
$$\left. \times \left[1 - \phi_{X(t+j-1-N_1)}(m)\right]^{v_{i_j}} \right] \qquad (30)$$

and the distribution $\phi_{X(t)}(u) = \text{Prob}(X(t) \leq u) = \text{Prob}(e(t) \leq u - D(t)) = \phi_{e(t)}(u - D(t))$ denotes the distribution of the input signal.

Proof: Since $Y_{\underline{f}}(\underline{X}(t))$ is the output of a stack filter, then

$$|D(t) - Y_{\underline{f}}(\underline{X}(t))| = \sum_{m=1}^{M} |d^m(t) - f(\underline{x}^m(t))| \qquad (31)$$

and (28) can be rewritten as

$$J_{\mathcal{D}, \Phi}(\underline{f}) = \frac{1}{T} \sum_{t=1}^{T} E|D(t) - Y_{\underline{f}}(\underline{X}(t))|$$
$$= \frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} E|d^m(t) - f(\underline{x}^m(t))|. \qquad (32)$$

However, $\varepsilon = |d^m(t) - f(\underline{x}^m(t))|$ is a random variable taking on only the values 0 and 1; hence, $E(\varepsilon) = \text{Prob}(\varepsilon = 1)$, and it follows that

$$J_{\mathcal{D}, \Phi}(\underline{f}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} \text{Prob}(|d^m(t) - f(\underline{x}^m(t))| = 1)$$
$$= \frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} (\text{Prob}(d^m(t) = 1, f(\underline{x}^m(t)) = 0)$$
$$+ \text{Prob}(d^m(t) = 0, f(\underline{x}^m(t)) = 1))$$
$$= \frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} (\text{Prob}(D(t) \geq m, f(\underline{x}^m(t)) = 0)$$
$$+ \text{Prob}(D(t) < m, f(\underline{x}^m(t)) = 1))$$
$$= \frac{1}{T} \sum_{t=1}^{T} \left[\sum_{m=1}^{D(t)} \text{Prob}(f(\underline{x}^m(t)) = 0) \right.$$
$$\left. + \sum_{m=D(t)+1}^{M} \text{Prob}(f(\underline{x}^m(t)) = 1) \right]$$
$$= \frac{1}{T} \sum_{t=1}^{T} \left[\sum_{m=1}^{D(t)} \sum_{\underline{v} \in V_0(\underline{f})} \text{Prob}(\underline{x}^m(t) = \underline{v}) \right.$$
$$\left. + \sum_{m=D(t)+1}^{M} \sum_{\underline{v} \in V_1(\underline{f})} \text{Prob}(\underline{x}^m(t) = \underline{v}) \right]$$
$$= \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{2^N} \left[\sum_{m=1}^{D(t)} (1 - f(\underline{v}_i))(\text{Prob}(\underline{x}^m(t) = \underline{v}_i) \right.$$
$$\left. + \sum_{m=D(t)+1}^{M} f(\underline{v}_i)\text{Prob}(\underline{x}^m(t) = \underline{v}_i) \right]$$
$$= C_0^{(\mathcal{D}, \Phi)} + \sum_{i=1}^{2^N} c^{(\mathcal{D}, \Phi)}(\underline{v}_i) f(\underline{v}_i),$$

where

$$c^{(\mathcal{D}, \Phi)}(\underline{v}_i) = \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{m=D(t)+1}^{M} \text{Prob}(\underline{x}^m(t) = \underline{v}_i) \right.$$
$$\left. - \sum_{m=1}^{D(t)} \text{Prob}(\underline{x}^m(t) = \underline{v}_i) \right),$$

$$C_0^{(\mathcal{D}, \Phi)} = \frac{1}{T} \sum_{i=1}^{2^N} \sum_{t=1}^{T} \sum_{m=1}^{D(t)} \text{Prob}(\underline{x}^m(t) = \underline{v}_i)$$
$$= \frac{1}{T} \sum_{t=1}^{T} D(t) = C_0.$$

The probabilities in the above expressions can be evaluated as follows:

$$\text{Prob}(\underline{x}^m(t) = \underline{v}_i)$$
$$= \text{Prob}(x^m(t - N_1) = v_{i_1}, \cdots, x^m(t + N_2) = v_{i_N})$$
$$= \prod_{j=1}^{N} \text{Prob}(X(t + j - 1 - N_1) \geq m)^{v_{i_j}}$$
$$\times \text{Prob}(X(t + j - 1 - N_1) < m)^{1 - v_{i_j}}$$
$$= \prod_{j=1}^{N} [1 - \phi_{X(t+j-1-N_1)}(m - 1)]^{v_{i_j}}$$
$$\times \phi_{X(t+j-1-N_1)}(m - 1)^{1 - v_{i_j}}$$

finally yielding (30). $\qquad\square$

Corollary 3.1: If $\mathcal{T} = (\mathcal{D}, \mathcal{X})$ is a training set where $\mathcal{X} = \{D(t) + e(t)\}_{t=1}^{T}$ has been generated as described in the assumptions of Problem 3, then the cost coefficients computed using (9)–(14) have the mean

$$Ec(\underline{v}_i) = c^{(\mathcal{D}, \Phi)}(\underline{v}_i). \tag{33}$$

Proof: From (24), we have

$$E\left[\left(J_{\mathcal{T}}(\underline{g}_{\underline{v}_1}) - C_0\right) \quad \cdots \quad \left(J_{\mathcal{T}}(\underline{g}_{\underline{v}_{2N}}) - C_0\right)\right] = E\underline{C}^T K$$
$$= \left[\left(J_{(\mathcal{D}, \Phi)}(\underline{g}_{\underline{v}_1}) - C_0\right) \quad \cdots \quad \left(J_{(\mathcal{D}, \Phi)}(\underline{g}_{\underline{v}_{2N}}) - C_0\right)\right]$$
$$= (\underline{C}^{(\mathcal{D}, \Phi)})^T K. \tag{34}$$

and since K is nonsingular (its inverse being the matrix L), it results that $E\underline{C} = \underline{C}^{(\mathcal{D}, \Phi)}$, and hence, (33) is true. $\quad\square$

D. Connections Between the Optimal Boolean Filter and the Optimal Stack Filter and Derivation of Projection Rules

In order to find the Boolean function $\underline{f}^{b*}$ that makes (17) or (29) minimum, it is very easy to see that the following assignment [9], [11], [20] will provide the optimal solution:

$$\underline{f}_i^{b*} = f^{b*}(\underline{v}_i) = \begin{cases} 1, & \text{if } c(\underline{v}_i) < 0; \\ 0 & \text{if } c(\underline{v}_i) > 0; \\ *(\text{don't care}), & \text{if } c(\underline{v}_i) = 0. \end{cases} \tag{35}$$

The don't care positions can be set to either 1 or 0, without affecting the performance cost and, therefore, will be assigned, taking into account different criteria, e.g., the lowest complexity of implementation of the resulting $\underline{f}^{b*}$. The algorithm (35) for finding $\underline{f}^{b*}$ is very fast since it contains only trivial comparisons and assignment operations. It will serve as a basis for finding the optimal stack filter using a projection approach. In order to find the projection mechanism, we analyze, in the following, the connections between the optimal Boolean filter and the optimal stack filter.

Lemma 3.4: For any Boolean function $\underline{f}^b$, there are unique PBF's $\underline{f}_{\min}^+$ and $\underline{f}_{\max}^+$ that satisfy

$$\underline{f}_{\min}^+ = \arg\min_{\underline{f}^+ | \underline{f}^+ \leq \underline{f}^b} \|\underline{f}^+ - \underline{f}^b\|_2 \tag{36}$$

$$\underline{f}_{\max}^+ = \arg\min_{\underline{f}^+ | \underline{f}^+ \geq \underline{f}^b} \|\underline{f}^+ - \underline{f}^b\|_2 \tag{37}$$

and can be computed as

$$V_1(\underline{f}_{\min}^+) = I_1(\underline{f}^b) \text{ with } I_1(\underline{f}^b)$$
$$= \{\underline{v}_i | W_{\text{down}}(\underline{v}_i) \subset V_1(\underline{f}^b)\} \tag{38}$$
$$V_0(\underline{f}_{\max}^+) = I_0(\underline{f}^b) \text{ with } I_0(\underline{f}^b)$$
$$= \{\underline{v}_i | W_{\text{up}}(\underline{v}_i) \subset V_0(\underline{f}^b)\}. \tag{39}$$

Proof: We prove only (36) and (38) here; the proof for (37) and (39) results simply by changing $V_1 \rightarrow V_0$ and $W_{\text{down}} \rightarrow W_{\text{up}}$.

We will repeatedly use the following statements that are obviously true:

$$\underline{f}^+ \text{ is positive iff } \forall \underline{v}_i \in V_1(\underline{f}^+), \quad W_{\text{down}}(\underline{v}_i) \subset V_1(\underline{f}^+);$$
$$\tag{40}$$

$$\underline{f}^+ \text{ is positive iff } \forall \underline{v}_i \in V_0(\underline{f}^+), \quad W_{\text{up}}(\underline{v}_i) \subset V_0(\underline{f}^+). \tag{41}$$

We first check that $\underline{f}_{\min}^+$, given by $V_1(\underline{f}_{\min}^+) = I_1(\underline{f}^b)$, is a PBF indeed. For any $\underline{v}_i \in I_1(\underline{f}^b)$ and any $\underline{v}_j \in W_{\text{down}}(\underline{v}_i)$, we have $W_{\text{down}}(\underline{v}_j) \subset W_{\text{down}}(\underline{v}_i) \subset V_1(\underline{f}^b)$, which implies $\underline{v}_j \in I_1(\underline{f}^b)$, and thus, $W_{\text{down}}(\underline{v}_i) \subset I_1(\underline{f}^b) = V_1(\underline{f}_{\min}^+)$, which, using (40), shows that $\underline{f}_{\min}^+$ is PBF.

Now, we check the minimality property (36) of $\underline{f}_{\min}^+$. If $\underline{f}_{\min}^+$ is not minimal in the sense of (36), then there is one PBF $\underline{f}^+ \leq \underline{f}^b$ with $f^+(\underline{v}_i) = 1$ and $f_{\min}^+(\underline{v}_i) = 0$ for some $\underline{v}_i$. However, $W_{\text{down}}(\underline{v}_i) \subset V_1(\underline{f}^+) \subset V_1(\underline{f}^b)$ (the last inclusion is due to $\underline{f}^+ \leq \underline{f}^b$), and now, $\underline{v}_i \in I_1(\underline{f}^b)$, and $f_{\min}^+(\underline{v}_i) = 1$, which contradicts the previous statement, and therefore, $\underline{f}_{\min}^+$ is the minimal in (36). $\qquad\square$

Lemma 3.5: Let $\underline{f}^{b*}$ be the optimal Boolean function and $\underline{f}^{+*}$ an optimal *positive* Boolean function, minimizing (17), where all $c(\underline{v}_i) \neq 0$. Denote $\underline{f}_{\min}^+$, the minimal PBF and $\underline{f}_{\max}^+$ the maximal PBF of the optimal Boolean function $\underline{f}^{b*}$ (36), (37). Then

i) $f^{+*}(\underline{v}_i) = 1 \quad \forall \underline{v}_i \in V_1(\underline{f}_{\min}^+)$ $\tag{42}$

ii) $f^{+*}(\underline{v}_i) = 0 \quad \forall \underline{v}_i \in V_0(\underline{f}_{\max}^+)$ $\tag{43}$

iii) $\underline{f}_{\min}^+ \leq \underline{f}^{+*} \leq \underline{f}_{\max}^+.$ $\tag{44}$

Proof: Suppose that i) is not true and that there is $\underline{v}_i \in V_1(\underline{f}_{\min}^+)$ such that $f^{+*}(\underline{v}_i) = 0$. Define the set $I_{\text{fake}} = W_{\text{down}}(\underline{v}_i) \cap V_0(\underline{f}^{+*})$, which is obviously included in $V_1(\underline{f}_{\min}^+)$, and therefore, $\forall \underline{v}_j \in I_{\text{fake}}, f^b(\underline{v}_j) = 1$, and from (35), $c(\underline{v}_j) < 0$. $\underline{v}_j \in I_{\text{fake}}$ also implies $f^{+*}(\underline{v}_j) = 0$. Define $\underline{f}^+$ such that $f^+(\underline{v}_k) = f^{+*}(\underline{v}_k)$, $\forall \underline{v}_k \notin I_{\text{fake}}$ and $f^+(\underline{v}_j) = 1$, $\forall \underline{v}_j \in I_{\text{fake}}$. First, we show the positivity of $\underline{f}^+$, using (40), splitting $V_1(\underline{f}^+)$ into $(V_1(\underline{f}^+) \backslash I_{\text{fake}})$ and I_{fake}. We take first into consideration an arbitrary $\underline{v}_k$ in the set $V_1(\underline{f}^+) \backslash I_{\text{fake}}$ for which (40) immediately follows: $W_{\text{down}}(\underline{v}_k) \subset V_1(\underline{f}^{+*}) \subset V_1(\underline{f}^+)$. For arbitrary $\underline{v}_j \in I_{\text{fake}}$, we have $(W_{\text{down}}(\underline{v}_j) \cap I_{\text{fake}}) \subset I_{\text{fake}} \subset V_1(\underline{f}^+)$, and $(W_{\text{down}}(\underline{v}_j) \backslash I_{\text{fake}}) \subset V_1(\underline{f}^{+*}) \subset V_1(\underline{f}^+)$. Now, the positivity of $\underline{f}^+$ is proved, and we show that $\underline{f}^+$ will provide a better value of criterion (17) than $\underline{f}^{+*}$: $J(\underline{f}^{+*}) - J(\underline{f}^+) = -\sum_{\underline{v}_j \in I_{\text{fake}}} c(\underline{v}_j) > 0$. Since now, it would result that $\underline{f}^{+*}$ is not a minimum of (17), we have a contradiction, and therefore,

i) is true. The proof of ii) follows closely that for i) with the substitution $V_1 \rightarrow V_0$ and $W_{\text{down}} \rightarrow W_{\text{up}}$. iii) results immediately from i) and ii). $\square$

We now define the undecided set $I_u(\underline{f}^{b*}) = \mathcal{B}^N \backslash (I_0(\underline{f}^{b*}) \cup I_1(\underline{f}^{b*}))$, where $\mathcal{B} = \{0, 1\}$. For $\underline{v}_i \in I_u(\underline{f}^{b*})$, the value of the optimal $f^{+*}(\underline{v}_i)$ will be decided by the *value* of $c(\underline{v}_i)$, whereas for $\underline{v}_j \in (I_0(\underline{f}^{b*}) \cup I_1(\underline{f}^{b*}))$, the value of the optimal $f^{+*}(\underline{v}_j)$ (see (42) and (43)) is decided only by the sign of the cost coefficients (which determined the values for $\underline{f}^{b*}, \underline{f}^+_{\min}$ and $\underline{f}^+_{\max}$).

Corollary 3.2: Any of

i) "$c(\underline{v}_i) < 0$ implies $c(\underline{v}_j) < 0$, $\forall \underline{v}_j \in W_{\text{down}}(\underline{v}_i)$" $\quad$ (45)

ii) $I_1(\underline{f}^{b*}) = V_1(\underline{f}^{b*})$ $\qquad\qquad\qquad\qquad\qquad\qquad$ (46)

iii) $I_0(\underline{f}^{b*}) = V_0(\underline{f}^{b*})$ $\qquad\qquad\qquad\qquad\qquad\qquad$ (47)

implies that $I_u(\underline{f}^b) = \emptyset$ and that $\underline{f}^{b*} = \underline{f}^+_{\max} = \underline{f}^+_{\min} = \underline{f}^{+*}$.

Lemma 3.6—Indetermination Case: Let $\underline{f}^+$ be an optimal PBF with respect to (17) and $\underline{v}_i$ such that $c(\underline{v}_i) = 0$.

i) If

$$W_{\text{down}}(\underline{v}_i) \subset V_1(\underline{f}^+) \qquad (48)$$

then $\underline{f}^{+\text{new}}$ with $V_1(\underline{f}^{+\text{new}}) = V_1(\underline{f}^+) \cup \{\underline{v}_i\}$ is also a PBF optimal with respect to (17).

ii) If

$$W_{\text{up}}(\underline{v}_i) \subset V_0(\underline{f}^+) \qquad (49)$$

then $\underline{f}^{+\text{new}}$ with $V_0(\underline{f}^{+\text{new}}) = V_0(\underline{f}^+) \cup \{\underline{v}_i\}$ is also a PBF optimal with respect to (17).

Therefore, $I_0(\underline{f}^{b*})$ and $I_1(\underline{f}^{b*})$ can be extended with the vectors corresponding to zero cost coefficients when either (48) or (49) is fulfilled. This will reduce the cardinality of the undecided set and will also lead in the case of (48) to a PBF with a reduced number of minimal terms.

The formal results we presented in this section establish the rules for the projection of the Boolean filter into the stack filter class, i.e., fixing the values of $f^+(\underline{v}_i)$ for all vectors $\underline{v}_i \in (I_0(\underline{f}^{b*}) \cup I_1(\underline{f}^{b*}))$ to the values taken by $f^{b*}(\underline{v}_i)$.

In the next section, we present techniques that can be used to decide the values of $\underline{f}^+$ for vectors in the undecided set $I_u(\underline{f}^{b*})$.

E. Optimization of the Stack Filter After the Projection Stage

Denote $\underline{h}_r$ to be the $(n_u = \text{Card}(I_u(\underline{f}^{b*})))$-dimensional vector grouping the values $f^+(\underline{v}_i)$ for $\underline{v}_i \in I_u(\underline{f}^{b*})$.

Lemma 3.7: The values $f^+(\underline{v}_i)$ of optimal PBF for the vectors in the undecided set $I_u(\underline{f}^{b*})$ are obtained by solving the LP problem

$$\min_{\underline{h}_r} \sum_{\underline{v}_i \in I_u(\underline{f}^{b*})} c(\underline{v}_i) f^+(\underline{v}_i) \qquad (50)$$

$$f^+(\underline{v}_i) - f^+(\underline{v}_j) \leq 0 \quad \forall \underline{v}_i \in I_u(\underline{f}^{b*}),$$
$$\forall \underline{v}_j \in (H_{\text{down}}(\underline{v}_i) \cap I_u(\underline{f}^{b*})) \qquad (51)$$

$$f^+(\underline{v}_i) \leq 1 \ \forall \underline{v}_i \text{ with } (H_{\text{down}}(\underline{v}_i) \cap I_u(\underline{f}^{b*})) = \emptyset \qquad (52)$$

where (51) and (52) select only a nonredundant set of stacking constraints for the undecided variables.

Proof: Define now a directed graph known as the window process digraph in [6], with the vectors in $I_u(\underline{f}^{b*})$ as nodes and with arcs leaving $\underline{v}_j$ and entering $\underline{v}_i$ iff both $\underline{v}_i \leq \underline{v}_j$ and $w_H(\underline{v}_j) = w_H(\underline{v}_i) + 1$ are true (e.g., the directed graph formed by the nodes with tones of gray and the bold arrows in Fig. 3). The incidence matrix of the graph M with rows and columns indexed by arcs and nodes, respectively, has nonzero elements of the form $M(k, i) = 1, M(k, j) = -1$ for each arc k entering $\underline{v}_i$ and leaving $\underline{v}_j$, and therefore, we can use M to write the system of inequalities (51) as $M\underline{h}_r \leq 0$. However, for a directed graph, the incidence matrix is totally unimodular (see e.g., [12], p. 274). When adding one by one the inequalities (52) to the system (51) (each time a row is added, which has the only nonzero entry with value 1), the matrix of the system remains totally unimodular [12, p. 280]. Since the free vector of the inequality system (51)–(52) is an integer and the matrix of the system is totally unimodular, all basic feasible solutions of the linear programmming problem (50)–(52) must be integer valued vectors[3]. $\square$

The number of variables and constraints in the reduced LP problem depends on the cardinality of $I_u(\underline{f}^{b*})$ as well as on the Hamming weight of the undecided vectors. For illustration in Fig. 3, from the original LP problem with $2^N = 32$ variables and $N2^{N-1} + 1 = 81$ constraints, only 10 variables and 17 constraints remain in the reduced LP problem. In image restoration applications, one typical situation for $N = 13$ is the reduction from $2^{13} = 8192$ variables and $N2^{N-1} + 1 = 53\,249$ constraints to 250 undecided variables and 470 constraints. However, the tractability of the LP problem remains an application-dependent issue, and we need to look for a more efficient alternative to solve the problem, even if sometimes only suboptimally. However, even when the LP problem is tractable, it may advance slowly toward the global optimum solution because (50)–(52) is a degenerate LP problem, (e.g., hundreds of iterations can be done until an improvement of the cost criterion is achieved). One way to accelerate the convergence of LP procedures is to initialize them as close as possible to the optimum solution, e.g., using a (sub)optimal procedure. Next, we present an iterative procedure that advances toward the optimum, checking if adding or deleting groups of vectors from the onset of the PBF would produce an improvement.

Let $\underline{f}^+$ be the current PBF, and check the following ways to improve its performance criterion:

a) extending the onset with a minimal term corresponding to $\underline{v}_i$, $V_1(\underline{f}^+_{\text{improv}}) \leftarrow V_1(\underline{f}^+) \cup W_{\text{down}}(\underline{v}_i)$
b) extending the offset with the vectors stacking over $\underline{v}_i$, $V_0(\underline{f}^+_{\text{improv}}) \leftarrow V_0(\underline{f}^+) \cup W_{\text{up}}(\underline{v}_i)$.

The improvement in performance can be evaluated adding only a small number of cost coefficients (there is no need to evaluate the costs over all $\underline{v}_j \in \mathcal{B}^N$) as follows: For case a)

$$J(\underline{f}^+) - J(\underline{f}^+_{\text{improv}}) = - \sum_{\underline{v}_j \in I_u \cap W_{\text{down}}(\underline{v}_i) \cap V_0(\underline{f}^+)} c(\underline{v}_i) \qquad (53)$$
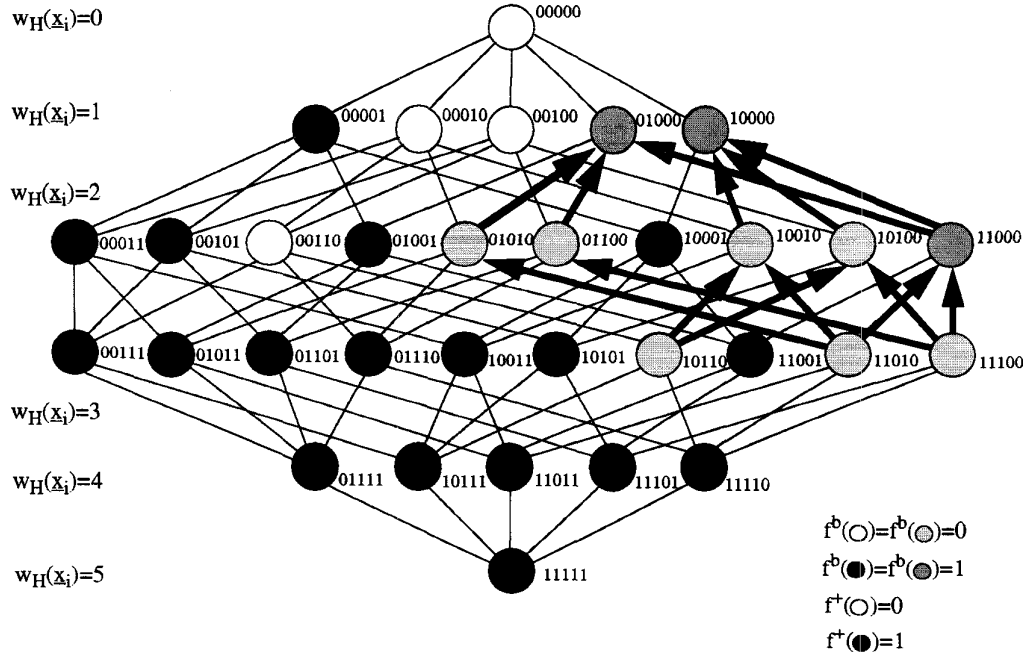
Fig. 3. Three classes of nodes in a Hasse diagram, decided by the *sign of the cost coefficients* or, equivalently, by the associated optimal Boolean function $\underline{f}^{b*}$. White: the class $I_0(\underline{f}^{b*})$, where $f_{\mathrm{OPT}}^+(\underline{v}_i)$ *must* be 0; Black: the class $I_1(\underline{f}^{b*})$, where $f_{\mathrm{OPT}}^+(\underline{v}_i)$ *must* be 1; Gray (light and dark) the class $I_u(\underline{f}^{b*})$, where $f_{\mathrm{OPT}}^+(\underline{v}_i)$ *may* be 0 or 1, according to the *values* of cost coefficients for these nodes. Bold arrows are the arcs in a directed graph specifying the stacking constraints to be imposed in the reduced LP problem.

and for case b)

$$J(\underline{f}^+) - J(\underline{f}_{\text{improv}}^+) = \sum_{\underline{v}_j \in I_u \cap W_{\text{up}}(\underline{v}_i) \cap V_1(\underline{f}^+)} c(\underline{v}_i). \quad (54)$$

If the result in (53) or (54) is positive, $\underline{f}_{\text{improv}}^+$ effectively improves the performance.

IV. FAST PROCEDURES FOR STACK FILTER DESIGN

A. Fast Optimal Stack Filter Design for a Fixed Order of the Filter

As direct consequences of the properties and connections between optimal Boolean filters and optimal stack filters derived in the preceding sections, we present in Table I the new procedure for stack filter design.

The procedure formed by Stages I–III has proven very effective in the extensive simulations reported in the experimental section. In nearly 90% of the cases, the procedure reached the global optimum. In the remaining 10% of the cases, the suboptimal solutions were only a few tenths of percent higher than the global optimum. Thus, Stage IV has been used mainly to check that $\underline{f}_{\mathrm{OPT.III}}^+$ is indeed the global optimum.

The variant of simplex that we selected for implementing Step IV.1 has $n_u = \mathrm{Card}(I_u(\underline{f}^{b*}))$ principal variables and m slack variables (where m is the number of inequalities in (51) and (52)).

The initialization in Step IV.2 is realized by iterating the simplex algorithm as follows: We start with the basic extreme solution, where all nonbasic variables are the principal

variables, and at each step, we enforce pivoting, such that each variable $\underline{v}_i$ with $f_{\mathrm{OPT.III}}^+(\underline{v}_i) = 1$ enters the basis. In order not to violate the LP constraints, the sequence of insertion of such $\underline{v}_i$ is in decreasing order of their Hamming weight. When Step IV.2 is accomplished, the LP solution will be identical to $\underline{f}_{\mathrm{OPT.III}}^+$, only with a cost of $\mathrm{Card}(I_u(\underline{f}^{b*}) \cap V_1(\underline{f}_{\mathrm{OPT.III}}^+)) < n_r$ simplex iterations, which is much faster than letting the simplex iterate with the classical rule of column selection (namely, selection of the nonbasic variable with maximal positive cost) because (51) and (52) is a degenerate LP problem.

B. Fast Optimal Stack Filter Design in a Matrix Computing Environment

The optimal stack filter design procedure developed in the previous section can be efficiently implemented in a general purpose programming language, e.g., C. It is possible to describe the steps of the FASTAF procedure using an alternative formulation in a matrix environment with the main operation being the product of a matrix with a vector. This will provide an immediate and compact implementation of FASTAF in a Matlab environment but also will shed additional light into the simplicity of the procedure.

We present in Table II the implementation of stages II and III of FASTAF. The connections between the formulations in Tables II and I can be readily established, given the significance of the conjunctive matrix K (23) (see also Fig. 2 and a simplified procedure in [14]). Note that checking for improvements of f_{red} in Table II is slightly different

TABLE I
PROCEDURE FASTAF

Procedure FASTAF:

Stage I Compute the cost coefficients;

Stage II Compute the optimal Boolean function $\underline{f}^{b*}$,(35);

Stage III Find a (sub)optimal positive Boolean function $\underline{f}^+_{OPT.III}$:

> **III.1** Compute the set of decided units, $I_1(\underline{f}^{b*}) = \{\underline{v}_i | \forall \underline{v}_j \in W_{down}(\underline{v}_i), f^{b*}(\underline{v}_j) = 1\}$;
>
> **III.2** If $I_1(\underline{f}^{b*}) = V_1(\underline{f}^{b*})$ then $\underline{f}^{b*}$ is PBF. The global optimal PBF is $\underline{f}^+_{OPT.IV} = \underline{f}^{b*}$, Exit.
>
> **III.3** Compute the set of decided zeros, $I_0(\underline{f}^{b*}) = \{\underline{v}_i | \forall \underline{v}_j \in W_{up}(\underline{v}_i), f^{b*}(\underline{v}_j) = 0\}$ and the undecided set $I_u(\underline{f}^{b*}) = \mathcal{B}^N - (I_0(\underline{f}^{b*}) \cup I_1(\underline{f}^{b*}))$;
>
> **III.4** Find $\underline{f}^+_{min}$ and $\underline{f}^+_{max}$ where $V_1(\underline{f}^+_{min}) \leftarrow I_1(\underline{f}^{b*})$ and $V_0(\underline{f}^+_{max}) \leftarrow I_0(\underline{f}^{b*})$;
>
>> Compute the criterion J for the PBF's $\underline{f}^+_{min}, \underline{f}^+_{max}$ and select the PBF
>>
>> with the lowest J for initializing $\underline{f}^+_{OPT.III}$
>
> **III.5** Iterate until no more improvements are obtained:
>
>> For all $\underline{v}_i \in I_u(\underline{f}^{b*})$
>>
>>> **III.5.1** Compute $\underline{f}^+_{improv}$ as follows:
>>>
>>>> if $f^+_{OPT.III}(\underline{v}_i) == 0$, then $V_1(\underline{f}^+_{improv}) \leftarrow (V_1(\underline{f}^+_{OPT.III}) \cup W_{down}(\underline{v}_i))$
>>>>
>>>> $J(\underline{f}^+_{OPT.III}) - J(\underline{f}^+_{improv}) = -\sum_{\underline{v}_j \in I_u \cap W_{down}(\underline{v}_i) \cap V_0(\underline{f}^+_{OPT.III})} c(\underline{v}_i)$
>>>>
>>>> if $f^+_{OPT.III}(\underline{v}_i) == 1$, then $V_0(f^+_{improv}) \leftarrow (V_0(\underline{f}^+_{OPT.III}) \cup W_{up}(\underline{v}_i))$
>>>>
>>>> $J(\underline{f}^+_{OPT.III}) - J(\underline{f}^+_{improv}) = \sum_{\underline{v}_j \in I_u \cap W_{up}(\underline{v}_i) \cap V_1(\underline{f}^+_{OPT.III})} c(\underline{v}_i)$
>>>
>>> **III.5.2** Check if $\underline{f}^+_{improv}$ improves the cost:
>>>
>>>> if $J(\underline{f}^+_{OPT.III}) - J(\underline{f}^+_{improv}) > 0$ then $\underline{f}^+_{OPT.III} \leftarrow \underline{f}^+_{improv}$;

Stage IV (Optional) Find the exact solution starting from $\underline{f}^+_{OPT.III}$:

> **IV.1** Extract the linear programming problem for undecided variables (50),(51),(52);
>
>> If the dimension of LP problem is too large, Exit with $\underline{f}^+_{OPT.III}$.
>
> **IV.2** Initialize the extreme basic solution such that $f^+(\underline{v}_i) = f^+_{OPT.III}(\underline{v}_i)$
>
> **IV.3** Iterate the simplex, updating $f^+(\underline{v}_i)$, $\forall \underline{v}_i \in I_u(\underline{f}^{b*})$ until no more improvements are obtained;
>
>> Exit with
>>
>> $f^+_{OPT.IV}(\underline{v}_i) = f^+(\underline{v}_i), \quad \forall \underline{v}_i \in I_u(\underline{f}^{b*})$
>>
>> $f^+_{OPT.IV}(\underline{v}_i) = f^+_{OPT.III}(\underline{v}_i), \qquad \forall \underline{v}_i \in (\mathcal{B}^N - I_u(\underline{f}^{b*}))$

---

algorithmically than Step III.5 in Table I in order to use efficiently the facilities of the Matlab language.

V. OPTIMAL DESIGN UNDER SYMMETRY CONSTRAINTS

In some applications, there may exist some regularities (such as symmetry) in the data to be processed, which induce some regularities in the solution of the optimal design problem. If in the training set selected to be learned the regularities are slightly violated (e.g., in the cases of natural data sets), the optimal solution may deviate from the regular type of solution, which is expected. These cases of overfitting unimportant (particular) information in the training set must usually be avoided for two reasons: First, the solution must be near optimal for more data sets than the specific one used in the design stage, and second, by imposing additional constraints (e.g., symmetry), less complex solutions may result.

Since the filter perceives data through the template that slides along the signal, the symmetry of the data will be defined, taking into account the shape and the "spatial" order defined inside the window. The data that is the initial specification of the optimal design problem is the set (in fact, the multiset, since we allow repetition of the same elements in the set) $\mathcal{T} = \{(\underline{X}(t), D(t))\}^T_{t=1}$ or, equivalently, the set of binary windows and signals $\mathcal{T}^b = \{(\underline{x}(i), d(i))\}^{TM}_{i=1}$. In the process of optimal filter design, the order in which the elements of this set are indexed is not relevant.

Consider that some symmetry transformation can be defined, which transforms the window $\underline{X}$ to $\underline{X}_s = s(\underline{X})$.

Example: Consider a 2-D signal and a template of size 3×3. We define next several types of symmetry that arise naturally: left-right, s_{l-r}; up-down, s_{u-d}; symmetry with respect to the origin s_o; and the identity s_{id}, as shown in Fig. 4. The four symmetry functions form a group w.r.t. the composition of

TABLE II
STAGES II AND III OF FASTAF IMPLEMENTED IN MATLAB

```
k1=[ 1 0;1  1]; k2=kron(k1,k1); k4=kron(k2,k2);
k8=kron(k4,k4); k9=kron(k8,k1); K=k9;

f= (c<0) ;                                       % Stage II
I1 = find( (K'*f)==sum(K)');                     % Step III.1
if ( sum(f(I1))==sum(f==1) ),                    % Step III.2
      f_optimal = f;
   else
        Kf=K*f;
        I0=find((K*f)==0);                       % Step III.3
        Id=[I0;I1];
        Iu=(1:LG)'; Iu(Id)= [];
        K_red=K(Iu,Iu); K_red_neg=1-K_red;c_red=c(Iu);
        f_red_0=zeros(size(Iu));f_red=ones(size(Iu));   % Step III.4
        J0= c_red'*f_red_0; J1= c_red'*f_red;
        if(J0<J1), f_red=f_red_0; end
        [NN, Mn]=size(K_red);                    % Step III.5
        while (1)
            J0= c_red'*f_red;
            J_improv =J0 + K_red'*(c_red.*(1-f_red));
            J_improv_2=K_red_neg*(c_red.*f_red);
            [J_min_2,index_min_2]= min( J_improv_2 );
            [J_min , index_min]= min( J_improv(2:NN) );
            if .(J_min_2<J_min)
                if (J_min_2>=J0), break, end      % no more improvement
                   f_red = f_red.* K_red_neg(index_min_2,:)' ;
              else
                if (J_min>=J0), break, end         % no more improvement
                   f_red = f_red  | K_red(:,index_min+1) ;
              end
        end
  f_optimal(Id)=f(Id); f_optimal(Iu)=f_red;
end
```
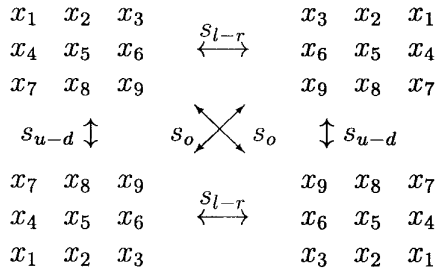

$$
\begin{array}{ccc}
x_1 & x_2 & x_3 \\
x_4 & x_5 & x_6 \\
x_7 & x_8 & x_9
\end{array}
\quad \overset{S_{l-r}}{\longleftrightarrow} \quad
\begin{array}{ccc}
x_3 & x_2 & x_1 \\
x_6 & x_5 & x_4 \\
x_9 & x_8 & x_7
\end{array}
$$

$$
S_{u-d} \updownarrow \qquad S_o \diagdown\!\!\diagup S_o \qquad \updownarrow S_{u-d}
$$

$$
\begin{array}{ccc}
x_7 & x_8 & x_9 \\
x_4 & x_5 & x_6 \\
x_1 & x_2 & x_3
\end{array}
\quad \overset{S_{l-r}}{\longleftrightarrow} \quad
\begin{array}{ccc}
x_9 & x_8 & x_7 \\
x_6 & x_5 & x_4 \\
x_3 & x_2 & x_1
\end{array}
$$

Fig. 4. Symmetry functions defined for the 3×3 template.

functions (the composition of the symmetries can be followed in the diagram in Fig. 4).

Definition 5.1: The data in the training set is *symmetric* w.r.t. the symmetry s if

$$
\{(\underline{X}(t), D(t))\}_{t=1}^T = \{(s(\underline{X}(p)), D(p))\}_{p=1}^T. \tag{55}
$$

The simplest way to enforce the symmetry constraint for an asymmetric training set $\mathcal{T} = \{(\underline{X}(t), D(t))\}_{t=1}^T$ is to build an extended set $\mathcal{T}^e$ having twice the cardinality of $\mathcal{T}$

$$
\mathcal{T}^e = \mathcal{T} \bigcup \mathcal{T}^s
$$
$$
= \{(\underline{X}(t), D(t))\}_{t=1}^T \bigcup \{(s(\underline{X}(p)), D(p))\}_{p=1}^T. \tag{56}
$$

(all operations are multiset operations).

Denote by $c(\underline{x})$ the cost coefficients obtained for the set $\mathcal{T}$ and $c^s(\underline{x})$ that for $\mathcal{T}^s$. From the definition of the cost

coefficients, we get

$$
c^s(\underline{x}) = c(s(\underline{x})). \tag{57}
$$

After the preprocessing stage of the optimal design procedure applied to the extended training set $\mathcal{T}^e$, we get

$$
c^e(\underline{x}) = c(\underline{x}) + c^s(\underline{x}) = c(\underline{x}) + c(s(\underline{x}))
$$
$$
= c^e(s(\underline{x})) \quad \forall \underline{x}
$$
$$
C_0^e = 2C_0 \tag{58}
$$

which shows that extended cost coefficients for symmetric windows are equal. Furthermore, this relation gives an alternative way to compute the extended set of cost coefficients: There is no need to process the extended training set $\mathcal{T}^e$, but it is enough to apply (58) after processing the original training set $\mathcal{T}$.

Due to (58), the Boolean filter designed using the extended cost coefficients will be symmetric, i.e.,

$$
f^e(\underline{x}) = f^e(s(\underline{x})) \quad \forall \underline{x}. \tag{59}
$$

The resulting minimal and maximal PBF's $f_{\min}^{e+}$, $f_{\max}^{e+}$ will also be symmetric since

$$
f_{\min}^{e+}(\tilde{\underline{v}}) = 1 \Leftrightarrow f^e(\underline{v}_i) = 1, \ \forall \underline{v}_i \ \text{with} \ \underline{v}_i \geq \tilde{\underline{v}}
$$
$$
\Leftrightarrow f^e(s(\underline{v}_i)) = 1, \ \forall \underline{v}_i \ \text{with} \ s(\underline{v}_i) \geq s(\tilde{\underline{v}})
$$
$$
\Leftrightarrow f_{\min}^{e+}(s(\tilde{\underline{v}})) = 1.
$$

It is straightforward [17] to impose the symmetry constraint in Stage III of the FASTAF procedure, i.e., when looking for the

improvement of $\underline{f}^+$ by adding or deleting a group of vectors from $V_1(\underline{f}^+)$ to select this group of vectors such that it is closed with respect to the symmetry property. Thus, we prove that the final $\underline{f}^+_{\text{OPT.III}}$ will respect the symmetry constraints

$$f^+_{\text{OPT.III}}(\underline{v}) = f^+_{\text{OPT.III}}(s(\underline{v})) \quad \forall \underline{v} \tag{60}$$

which will be shown in the case studies section to increase the robustness of the stack filter.

VI. CASE STUDY: TRAINING SET SIGNALS ARE NATURAL IMAGES CORRUPTED BY ADDITIVE NOISE

The general optimality theory developed in the previous sections for stack filtering is now applied to more specific cases in order to gain further analytical and experimental insights into the problem of optimal design of stack filters. The frame of the experiments is built carefully, setting at fixed values some filter parameters as the window length and focusing on the different experimental situations, which can be at least roughly characterized in practice by a limited number of parameters, e.g., the mean and SNR of the signal.

In this case study, we analyze the properties of optimal Boolean and stack filters when the desired signal in the training set is the image of a natural scene. We examine the effect of changing the desired signal (using various natural images) and the characteristics of the noise (the probability distribution function, the mean, and the variance). The corrupting process is assumed to be stationary and i.i.d. The experiments we performed cover the following situations:

- changing the noise PDF and its variance
- changing the variance and the mean of the noise
- changing the desired signal and the noise variance (One additional situation considered is imposing symmetry constraints.)
- changing the training set size and realizations of the noise. (The optimal design is carried out in both $\mathcal{T}$ and $(\mathcal{D}, \Phi)$ approaches.)

Finally, the computational performance for the new optimal stack filter design procedure is compared with those of other optimal stack filter design procedures. The general experimental conditions are set as follows. One training set contains the desired image $[\mathbf{D}(i,j)]_{i=1:512, j=1:512}$ and the perturbed image resulting after corrupting the desired image with i.i.d. noise [7]

$$\nu \sim (1-\lambda)\mathcal{N}\left(0, \sigma_n^2\right) + \lambda\mathcal{N}\left(0, \frac{\sigma_n^2}{\lambda}\right) \tag{61}$$

with mean $E(\nu) = 0$ and variance

$$\sigma_\nu^2 = \sigma_n^2\left(1 - \lambda + \frac{1}{\lambda}\right). \tag{62}$$

The nonlinear perturbation process is

$$\mathbf{X}(i,j)$$
$$= \begin{cases} \mathbf{D}(i,j) + \nu(i,j), & \text{if } 0 \le \mathbf{D}(i,j) + \nu(i,j) \le 255 \\ 0, & \text{if } \mathbf{D}(i,j) + \nu(i,j) < 0 \\ 255, & \text{if } 255 < \mathbf{D}(i,j) + \nu(i,j). \end{cases} \tag{63}$$

The template function of the filter is a window of size 3×3, including all the neighbors (in 8-connectivity) of the current pixel.

Note that due to the nonlinear effect of the noise on the original image, given by (63), the PDF. of the effective noise deviates slightly from the PDF. in (61) (the large values of the outliers are truncated), and the effective noise variance is smaller than in (62).

A. Experiment (λ-SNR): Changing the PDF and the Variance of the Noise

The experiment consists of designing optimal Boolean and stack filters for 120 training sets, generated with various pairs (λ, σ_ν), with the picture "harbor" as the desired image. The parameter λ varies between 0.1 (corresponding to a heavy tailed distribution) and 1 (corresponding to a Gaussian distribution). Equivalently, the independent variables will be considered to be $(\lambda, \text{SNR}_{\text{eff}})$, where SNR_{eff} is *estimated* from the training set (and not computed using (62)).

1) Stack Filter Design: The condition (45) was found to hold in roughly one fourth of the cases. The resulting stack filters are thus identical to the Boolean filters in those cases. In the other cases, the stack filter solutions are different from the Boolean ones. As can be seen from Fig. 5, a larger number of residual terms (vectors $\underline{v}_i$, where $f^{b*}(\underline{v}_i) = 1$ and $f^+_{\min}(\underline{v}_i) = 0$) implies a larger relative difference between optimal *Boolean* and optimal *stack* filter criteria. However, the largest value of the relative difference is 1.23%, suggesting that the optimal stack and Boolean filters have very close performances. This is yet another revealing property of the importance of stack filters as a major subclass of Boolean filters, at least as far as image restoration is concerned.

2) Sensitivity Measure for Robustness Evaluation: For the window width considered in this experiment ($N = 9$), the number of stack filters is well over 4 billion. It is thus conceivable that the 120 stack filter solutions be all different. This is actually the case; there is a large number of filters that can fit well within the particularities in the training set. Robustness, on the other hand, will not be considered in the sense of *invariance* of the filters resulting for problems with close initial specification, but rather in the sense of a *low sensitivity* of the solution due to changes in the initial specifications. We define the following sensitivity measure.

Definition—Relative Criteria Difference: The sensitivity (or the relative criteria difference) of the optimal solution $\underline{f}^*_{\mathcal{S}_0}$ induced by the training set $\mathcal{S}_0$ and evaluated over a training set $\mathcal{S}_i$ is

$$\text{RCD}\left(\underline{f}^*_{\mathcal{S}_0}, \mathcal{S}_0, \mathcal{S}_i\right) = \frac{J\left(\mathcal{S}_i, \underline{f}^*_{\mathcal{S}_0}\right) - J\left(\mathcal{S}_i, \underline{f}^*_{\mathcal{S}_i}\right)}{J\left(\mathcal{S}_i, \underline{f}^*_{\mathcal{S}_i}\right)} \tag{64}$$

where $J(\mathcal{S}_i, \underline{f}^*_{\mathcal{S}_i})$ is the optimal criterion value for the training set $\mathcal{S}_i$, and $J(\mathcal{S}_i, \underline{f}^*_{\mathcal{S}_0})$ is the value of the performance criterion obtained with the filter $\underline{f}^*_{\mathcal{S}_0}$ over the training set $\mathcal{S}_i$.

Using the above definition of relative criteria difference, we shall introduce now the notion of filter covering.
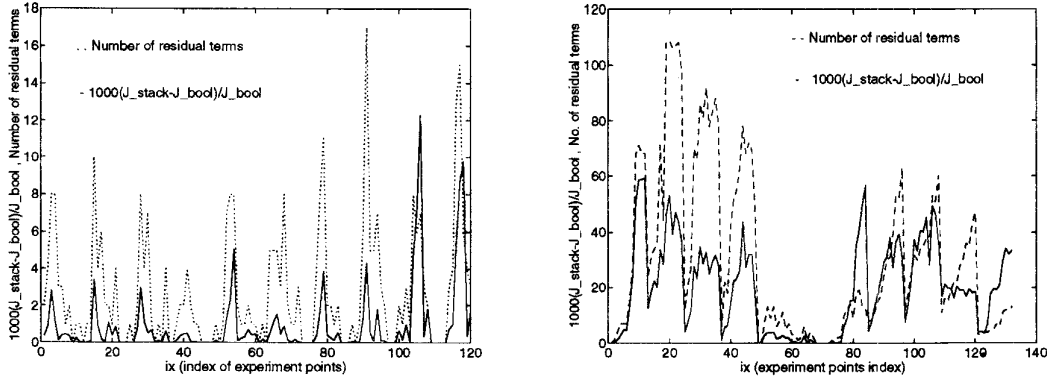
Fig. 5. (-) Relative difference between optimal *stack* and optimal *Boolean* filter criteria (the values are scaled by 1000); (...) number of residual terms after extracting the maximum positive Boolean function. (Left) Experiment (λ-SNR); (Right) experiment ($E(\nu)$-SNR).

Definition—Filter Covering: A filter $\underline{f}^*_{\mathcal{S}_0}$ is said to cover filter $\underline{f}^*_{\mathcal{S}_1}$ at threshold ε if

$$\text{RCD}\left(\underline{f}^*_{\mathcal{S}_0}, \mathcal{S}_0, \mathcal{S}_1\right) \leq \varepsilon. \tag{65}$$

Those "essentially" different filters among the 120 obtained in the optimal design experiment can now be determined by finding the minimum number of filters that cover at a threshold ε all the other filters. That is, we subdivide the experiment space under the above sensitivity criterion into regions yielding a reduced number of optimal solutions.

Table III(a) shows the partition of the experiment space into merely seven different regions using the covering at a threshold $\varepsilon = 1\%$. In each region, the performance of the covering filter is less than 1% worse than that of the corresponding optimal filter. It is quite remarkable that only seven filters succeeded in covering all the experiment points, given the very large spread in the values of RCD sensitivity: Sometimes, it varies between 0 and 1%, but very often, it ranges between 30 and 50%.

The main feature that discriminates between the seven covering filters is what we define as *filter activity*, i.e., the number of binary vectors $\underline{v}_i$ for which $f(\underline{v}_i)$ is different from the identity Boolean function ($f(x_1, x_2, \cdots, x_N) = x_{N_1+1}$ where $N_1 + 1$ is the output reference index). For high SNR values, the activity of the filter is low (e.g., filter 1 and filter 3) and for low SNR values, the activity is high (e.g., filter 6 and filter 7).

At a fixed SNR value, the filter activity should be higher in the presence of Gaussian noise than impulsive noise. For example, at 7.5 dB, the number of active terms is decreasing for filters 6, 5, 4, and 3; see Table III(a). This follows the intuition that, at the same SNR level, Gaussian noise disturbs a larger number of binary windows (from the overall 512 windows) than impulsive noise. On the other hand, the same filter is seen to be (sub)optimal at high SNR values in the presence of Gaussian noise and low SNR values in impulsive noise. As an example, filter 3 is (sub)optimal in the interval (13.5 and 16.5 dB) for the Gaussian distribution and in the interval (7.5 and 13.5 dB) for the impulsive distribution.

B. Experiment ($E(\nu)$-SNR): Changing the Mean and the Variance of Noise

1) Robustness Analysis: The classical assumptions on the noise process concerning its mean being zero is relaxed in this section. The aim is to study the effect on the optimal solutions in terms of robustness when changing the mean and the variance of the noise.

In this experiment, we analyze the effect of changing simultaneously the mean and the variance of the noise affecting the desired image "harbor." We use 132 different training sets with the noise mean ranging from -20 to $+20$ and SNR from 3 to 19.5 dB.

Fifteen optimal stack filters were found to cover (sub)optimally all the 132 training sets at a threshold of $\varepsilon = 0.02$. These are displayed in Table III(b).

The differences between the performance criteria for the optimal Boolean and stack filters over all the training sets are plotted in Fig. 5. The number of residual terms, after extracting the maximal positive Boolean functions, is also plotted in the same figure. The differences are the highest observed during these experiments but still to a level lower than 6%. The number of residual terms is large and for almost all the points in the experiment, the constraint (45) is violated.

An important conclusion of Experiment (λ-SNR) can be expressed as follows: *For zero mean noise, the properties of the optimal filter are mainly decided by the SNR of the application and only as a secondary factor by the distribution of the noise.* In this experiment, where the mean of the noise is allowed to vary, it can be seen from Table III(b) that the most important factor becomes the value of the mean. Only for very low absolute values of the mean (near zero-mean assumption), the SNR is taken into account. For high values of the mean, the optimal filter is decided according to the mean value, irrespective of the SNR levels.

Motivated by these findings, one can draw the following heuristic picture: The mean of the noise decides what order statistics filter is the best for the training set for reducing the mean of the filter output. The SNR level in the training set (the variance of the noise) decides to what extent a stack filter must differ from an order statistics filter to take into account

TABLE III

PARTITION OF TRAINING SETS SPACE IN A SMALL NUMBER OF REGIONS. THE OPTIMAL FILTER FOR ONE TRAINING SET BEHAVES (SUB)OPTIMALLY (AT A THRESHOLD ε IN THE RELATIVE CRITERIA DIFFERENCE) OVER ALL THE TRAINING SETS IN THE SAME REGION. THE NUMBERS IN EACH TABLE REPRESENT INDEXES OF COVERING FILTERS. THE GRAY LEVEL OF EACH REGION IS PROPORTIONAL TO THE ACTIVITY PARAMETER OF THE FILTER COVERING THAT REGION (WHITE IS FOR HIGH ACTIVITY.) (a) EXPERIMENT (λ-SNR,) $\varepsilon = 0.01$, NUMBER OF COVERING FILTERS #c.f. $= 7$; (b) Experiment ($E(\nu)$-SNR), $\varepsilon = 0.02$, #c.f. $= 15$; (c) Experiment ($\mathcal{D}$-SNR), $\varepsilon = 0.02$, #c.f. $= 11$; (d) Experiment ($\mathcal{D}$-SNR, symmetry), $\varepsilon = 0.02$, #c.f. $= 10$

(a) λ

SNR	1/10	1/9	1/8	1/7	1/6	1/5	1/4	1/3	1/2	1/1
3 dB	6	6	6	6	6	6	7	7	7	7
4.5 dB	6	6	6	6	6	6	6	6	7	7
6 dB	4	4	4	6	6	6	6	6	6	7
7.5 dB	3	3	4	4	5	6	6	6	6	6
9 dB	3	3	3	4	4	4	6	6	6	6
10.5 dB	3	3	3	3	3	3	4	6	6	6
12 dB	3	3	3	3	3	3	3	3	4	6
13.5 dB	3	3	3	3	3	3	3	3	3	3
15 dB	1	1	1	3	3	3	3	3	3	3
16.5 dB	1	1	1	1	3	3	3	3	3	3
18 dB	1	1	1	1	1	1	1	3	2	2
19.5 dB	1	1	1	1	1	1	1	1	1	1

(b) E(ν) (Noise mean)

SNR	-20	-10	-7	-4	-1	0	1	4	7	10	20
3 dB	14	11	11	9	6	6	6	10	12	12	15
4.5 dB	14	11	11	11	7	6	6	12	12	12	15
6 dB	14	8	8	11	6	6	6	12	12	12	15
7.5 dB	14	8	8	11	4	4	5	12	12	12	15
9 dB	14	8	8	8	4	4	5	12	12	12	15
10.5 dB	14	8	8	8	4	4	4	12	12	12	15
12 dB	14	8	8	8	4	4	4	12	12	12	15
13.5 dB	14	8	8	8	4	4	4	12	12	12	15
15 dB	14	8	8	8	3	3	3	12	12	12	15
16.5 dB	14	8	8	8	3	3	3	12	13	15	
18 dB	14	8	8	8	3	3	3	12	13	15	
19.5 dB	14	8	8	8	3	3	3	13	13	15	

(c) and (d) SNR (dB)

	3	4	6	8	9	10	12	14	15	16	18	20
harbour 1/4												
harbour 2/4												
harbour 3/4												
harbour 4/4												
airfield 1/4												
airfield 2/4												
airfield 3/4												
airfield 4/4												
bridge 1/4												
bridge 2/4												
bridge 3/4												
bridge 4/4												
lenna 1/4												
lenna 2/4												
lenna 3/4												
lenna 4/4												

a	b
c	d

the small details in the signal for reducing the variance of the output.

Since for high absolute values of the noise mean the optimal decision is based mainly on order statistics actions, the SNR affects the filter decision only marginally.

C. Experiment ($\mathcal{D}$-SNR): Changing the Desired Signal and the Noise Variance

The experiment aims at characterizing the performance of the optimal solutions for 192 different training sets, with various desired signals and with SNR's in the interval (3 and 19.5 dB). We used 16 images as desired signals, each of size 256×256 (all the quarters of the following 512×512 images: "harbor," "airfield," "bridge," and "lena").

The relative difference in the performance criterion for optimal stack filters and optimal Boolean filters is actually zero in more than three fourths of the experiment points,

suggesting remarkably that the constraint (45) holds for many training sets, which are natural images. When residual terms are present, some loss in performance is unavoidable due to additionally imposing the stacking constraint. This loss is, however, still very low (less than 1%) comparing with the benefits obtainable by the use of stack filters (in terms of analysis, implementation, or other aspects).

The number of optimal stack filters which cover, at a threshold of 1%, all the points of the experiment is now 31. This is somewhat larger than in the preceding experiments, but note also that the size of the experiment is larger. For the sake of illustrating the results, the covering threshold was selected to be 2%. Now, only 11 optimal stack filters were found to cover the experiment; see Table III(c). It is interesting to note the high number of occurrences of filter 2 (72 times) in contrast to the less significant number of occurrences of filter 5 (only once). The robustness expressed in the very low number of

filters needed for (nearly) optimally covering *very different* images leads naturally to the following conjecture.

Conjecture: It is possible to build a library (catalog) of optimal stack filters from which some potential candidates can be selected for a set of applications at hand. The best filter for a particular task can then be selected by comparing the performances obtained for this low number of candidates.

The same training sets of Experiment ($\mathcal{D}$-SNR) were also used for designing optimal *symmetric* stack filters. Three symmetry constraints were imposed: left-right s_{l-r}; up-down s_{u-d}, and symmetry with respect to the origin s_o. Only 10 optimal symmetric stack filters were needed to cover, with minimum number of terms, all 192 experiment points (at threshold $\varepsilon = 0.02$; see Table III(d).

D. Experiment (T, ν) Generalization Power of Optimal Stack Filters Designed in $\mathcal{T}$ and $(\mathcal{D}, \Phi)$ Approaches

We analyze, using the sensitivity measure RCD, how well the filter designed for a given training set generalizes to other training sets differing only in the noise realization used. We compare the generalization power of the $\mathcal{T}$ approach to that of $(\mathcal{D}, \Phi)$ approach.

It is natural that the generalization of an optimal filter to other training sets improves as the size T of the training set increases, and we illustrate this aspect considering six different values of training set size T: 256, 1024, 4096, 16384, 65536, 262144 (for each T, $\mathcal{D}$ is a square subimage from "harbor" image). For each size of the training set T, we generate 100 different noise realizations (generated as described in Experiment (λ-SNR), using $\lambda = 0.1, E(\nu) = 0, \text{SNR} = 9$ dB), compute the optimal stack filter for each training set, and then compute the sensitivities over the other training sets. We also compute for each T the optimal stack filter using $(\mathcal{D}, \Phi)$ approach, and evaluate the sensitivities of this solution when applied to each of the 100 training sets:

$$\text{RCD}\left(\underline{f}^*_{\mathcal{D},\Phi}, \mathcal{S}_i\right) = \frac{J\left(\mathcal{S}_i, \underline{f}^*_{\mathcal{D},\Phi}\right) - J\left(\mathcal{S}_i, \underline{f}^*_{\mathcal{S}_i}\right)}{J\left(\mathcal{S}_i, \underline{f}^*_{\mathcal{S}_i}\right)}. \quad (66)$$

When T is large enough (e.g., $T \geq 128 \times 128$), both the $\mathcal{T}$ and $(\mathcal{D}, \Phi)$ approaches provide optimal filters that are very close in performance, and thus very robust, as can be seen from Table IV.

When T is small, the $(\mathcal{D}, \Phi)$ approach is significantly superior to the $\mathcal{T}$ approach with respect to both worst and average sensitivities, and thus provides better generalization capabilities.

Further experimental results have shown a more general fact: The robustness of the optimal stack filter increases with the ratio training set size/processing window size T/N.

The conclusions in the preceding subsections were drawn based on a single realization of the noise, but it is clear that they hold true, irrespective of the particular noise realization used (compare the sensitivities in the last columns of Table IV, with the threshold $\varepsilon = 1$ or 2% used in the preceding section).

TABLE IV
GENERALIZATION POWER OF OPTIMAL STACK FILTERS: WORST CASE AND AVERAGE SENSITIVITIES FOR $\mathcal{T}$-OPTIMAL BOOLEAN FILTERS, $\mathcal{T}$-OPTIMAL STACK FILTERS, AND $(\mathcal{D}, \Phi)$-OPTIMAL STACK FILTERS. THE SENSITIVITIES ARE EVALUATED FOR 100 TRAINING SETS THAT DIFFER ONLY W.R.T. NOISE REALIZATION

Worst case sensitivity average sensitivity	$T = 16 \times 16$	$T = 32 \times 32$	$T = 64 \times 64$	$T = 128 \times 128$	$T = 256 \times 256$	$T = 512 \times 512$
$\mathcal{T}$-optimal Boolean filter $RCD(\underline{f}^b_{\mathcal{S}_i}, \mathcal{S}_i, \mathcal{S}_j)$ $\forall i, j = 1, \ldots, 100$	108.7% 49.0%	43.8% 20.9%	9.1% 3.3%	2.8% 0.8%	0.92% 0.29%	0.34% 0.078%
$\mathcal{T}$-optimal stack filter $RCD(\underline{f}^+_{\mathcal{S}_i}, \mathcal{S}_i, \mathcal{S}_j)$ $\forall i, j = 1, \ldots, 100$	67.4% 12.7%	23.0% 7.7%	6.0% 1.4%	1.9% 0.4%	0.71% 0.13%	0.30% 0.036%
$(\mathcal{D}, \Phi)$-optimal stack filter $RCD(\underline{f}^+_{\mathcal{D},\Phi}, \mathcal{S}_i)$ $\forall i = 1, \ldots, 100$	17.1% 7.8%	10.9% 4.8%	2.8% 0.8%	1.0% 0.3%	0.47% 0.10%	0.05% 0.011%

TABLE V
PERFORMANCES OBTAINED AND TIME REQUIRED FOR EACH STAGE OF FASTAF PROCEDURE, FOR THE TRAINING SETS IN EXPERIMENT $(E(\nu)$-SNR,) FOR WINDOW SIZE $N = 9$. THE LAST COLUMN GIVES THE TIME REQUIRED BY A COSTUMIZED LINEAR PROGRAMMING PROCEDURE (WHICH TAKES INTO ACCOUNT THE SPECIAL STRUCTURE OF THE CONSTRAINT MATRIX)

	Performances			Time		
	MAE f_{bool}	MAE $f_{OPT.III}$	MAE $f_{OPT.IV}$	CPU time[s] $f_{OPT.III}$	CPU time[s] $f_{OPT.IV}$	CPU time[s] f_{LP}
average over 132 sets	1.5717	1.5792	1.5768	0.200	0.02	6.55
worst case over 132 sets	1.8728	1.8952	1.8951	0.22	0.13	7.87

E. Evaluation of Computational Performances for the New Fast Design Procedures

The computational performances of FASTAF procedure when finding the optimal solutions in Experiment $(E(\nu)$-SNR) are illustrated in Table V. In this experiment, we have the largest number of training sets where the optimal Boolean filter was not identical to the stack filter, and therefore, the performances presented here are "worst case performances."

From 132 training sets, only in 11 cases (one such case is shown in the last row of the table), the filter after the third stage $f_{OPT.III}$ was different from the optimal one $f_{OPT.IV}$. However, all MAE's differences are obviously insignificant.

All simulations in this section are performed on a DEC 3000-700 AXP workstation. We also report in Table V the computational times required for running the various stages of FASTAF design procedure and compare them with the time for running one LP (simplex) procedure that was costumized to take into account the special structure of the stacking constraints [3] (the time for computing the cost coefficients is not shown since we illustrate here only differences between methods that start from the set of cost coefficients). We conclude that for window size $N = 9$ and images of size 512×512, the time cost for finding the optimal stack filter, starting from the cost coefficients, is usually 10 or 100 times faster for FASTAF, compared with a traditional simplex method. For larger window sizes ($N > 10$), the traditional LP can no longer solve the optimal design problem, at least using common memory and time resources, whereas FASTAF (the first three stages) can be used for orders as large as 20.

The CPU time required by the Matlab program described in Section IV-B for implementing stages II and III of the FASTAF procedure is on average 0.9 s, compared with the 0.22 s (see
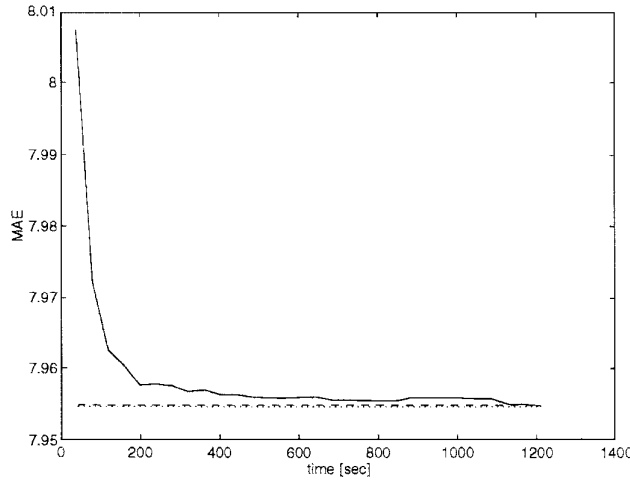
Fig. 6. Learning curves when designing an optimal stack filter of order $N = 13$: Top: (-) Fast–Lin algorithm; middle: (—) FASTAF algorithm, three stages (MAE($\{f_{OPT.III}\}$)); bottom: (···) FASTAF algorithm, four stages (MAE ($f_{OPT.IV}$) $\equiv$ exact LP solution).

TABLE VI

COMPARISON OF PERFORMANCES OBTAINED WITH FAST-LIN PROCEDURE AND WITH FASTAF PROCEDURE, FOR TWO WINDOW SIZES, $N = 9$ AND $N = 13$, USING THE 512×512 IMAGE "AIRFIELD" CORRUPTED AT $SNR = 6$ DB WITH GAUSSIAN CONTAMINATED NOISE ($\lambda = 0.1$)

	N=9		N=13	
	CPU time [sec]	MAE	CPU time [sec]	MAE
FASTAF	12.3	8.17293	25.1	7.95474
Fast–Lin 1 epoch	19.2	8.17306	39.1	8.0073
Fast–Lin 3 epochs	57.4	8.17293	117.3	7.96255
Fast–Lin 30 epochs	573.9	8.17293	1210.1	7.95474

Table V) required by its C language counterpart but still is very fast compared with the C implementation of LP or even the Fast–Lin procedure [10].

Table VI shows the computational performances (precision (MAE) and time required) for the newly introduced procedure (FASTAF) and the procedure Fast–Lin. Since the Fast–Lin procedure does not start from the set of cost coefficients but only uses the data in the training set, we also include, in the time reported for FASTAF, the time to compute the cost coefficients from the data. It can be seen that FASTAF is faster than Fast–Lin, when we consider the window sizes up to $N = 13$, for which the exact solution can be computed by both methods. In Fig. 6, we illustrate, for the case $N = 13$, how quickly FASTAF reaches the exact solution, compared with the slower convergence of the Fast–Lin algorithm to the same solution. A detailed description of the implementation aspects for our costumized LP procedure, which is a new fast algorithm that is recursive in order, and a comparative study of the existing procedures for optimal stack filter design are under preparation.

VII. CONCLUSION

This paper has introduced a *training* framework for the optimal nonlinear filter design problem. Fast procedures for obtaining optimal solutions for two classes of nonlinear filters,

Boolean filters, and stack filters were derived under the new framework.

We presented one major case study pertaining to the optimal design of Boolean and stack filters, where natural images corrupted by additive noise with different characteristics are considered. Elaborate experimental conditions were selected to investigate the robustness of the optimal solutions using a sensitivity measure computed on data sets.

The case study led us to conjecture that it is possible to build a library of optimal stack filters suitable for a set of applications. It is, of course, easier to disprove, if possible, this conjecture than to prove it. If it stands, the optimization efforts in nonlinear Boolean-based algorithms should be focused on building such filter libraries for different signal and image processing tasks.

APPENDIX
PROOF OF LEMMA 3.1

$$J_T(\underline{W}) = \frac{1}{T} \sum_{t=1}^{T} |D(t) - Y_{\underline{W}}(\underline{X}(t))|$$

$$= \frac{1}{T} \sum_{t=1}^{T} \left| \sum_{m=1}^{M} d^m(t) - \sum_{m=1}^{M} f_{\underline{W}}(\underline{x}^m(t)) \right|$$

$$\leq \frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} |d^m(t) - f_{\underline{W}}(\underline{x}^m(t))|. \qquad (67)$$

Now, the right-hand side of the above inequality can be rewritten as

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} |d^m(t) - f_{\underline{W}}(\underline{x}^m(t))|$$

$$= \frac{1}{T} \sum_{i=1}^{2^N} \left[\sum_{(t,m) \in \mathcal{M}_0(\underline{v}_i)} |d^m(t) - f_{\underline{W}}(\underline{x}^m(t))| \right.$$

$$\left. + \sum_{(t,m) \in \mathcal{M}_1(\underline{v}_i)} |d^m(t) - f_{\underline{W}}(\underline{x}^m(t))| \right]$$

$$= \frac{1}{T} \sum_{i=1}^{2^N} \left[\sum_{(t,m) \in \mathcal{M}_0(\underline{v}_i)} |f_{\underline{W}}(\underline{v}_i)| \right.$$

$$\left. + \sum_{(t,m) \in \mathcal{M}_1(\underline{v}_i)} |1 - f_{\underline{W}}(\underline{v}_i)| \right]$$

$$= \frac{1}{T} \sum_{i=1}^{2^N} [\mathcal{N}_1(\underline{v}_i) + (\mathcal{N}_0(\underline{v}_i) - \mathcal{N}_1(\underline{v}_i)) f_{\underline{W}}(\underline{v}_i)]$$

$$= C_0 + \sum_{i=1}^{2^N} c(\underline{v}_i) f_{\underline{W}}(\underline{v}_i) = J_T^b(\underline{W}) \qquad (68)$$

and part a) of the Lemma results.

To show the "if" part of b), suppose that f is a stack filter, and rewrite J_T as

$$J_T(\underline{W}) = \frac{1}{T} \sum_{t=1}^{T} \left| \sum_{m=1}^{M} (d^m(t) - f_{\underline{W}}(\underline{x}^m(t))) \right|. \qquad (69)$$

The nonzero terms of the expression inside the inner summation all have the same sign since for any two integer numbers D and Y

$$D \geq Y \Rightarrow \forall m, d^m \geq y^m, \quad D \leq Y \Rightarrow \forall m, d^m \leq y^m \quad (70)$$

and consequently $|D - Y| = \sum_{m=1}^{M} |d^m - y^m|$. Due to the stacking property of $f_{\underline{W}}$, we have $y^m = f_{\underline{W}}(\underline{x}^m(t))$, and thus

$$J_{\mathcal{T}}(\underline{W}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} |d^m(t) - f_{\underline{W}}(\underline{x}^m(t))|$$
$$= J_{\mathcal{T}}^b(\underline{W}). \quad (71)$$

We now have the "only if" part: If $J_{\mathcal{T}} = J_{\mathcal{T}}^b$ for all training sets $\mathcal{T}$, then is f a stack filter? The proof is by contradiction. Suppose $J_{\mathcal{T}} = J_{\mathcal{T}}^b$ for all $\mathcal{T}$, but f is not a stack filter. Hence, there exist $\underline{x}^{*1} = [x_1^{*1} \cdots x_N^{*1}] > \underline{x}^{*2} = [x_1^{*2} \cdots x_N^{*2}]$ such that $f_{\underline{W}}(\underline{x}^{*1}) = 0$, and $f_{\underline{W}}(\underline{x}^{*2}) = 1$. Suppose also that $\mathcal{T}$ contains, at time s, the data pair

$$\left(\underline{X}(s) = \left[\left(x_1^{*1} + x_1^{*2} \right) \cdots \left(x_N^{*1} + x_N^{*2} \right) \right], D(s) = 1 \right).$$

Since $\underline{x}^{*1} > \underline{x}^{*2}$, the thresholded versions of $\underline{X}(s)$ are $T_1(\underline{X}(s)) = \underline{x}^{*1}, T_2(\underline{X}(s)) = \underline{x}^{*2}$ and $T_m(\underline{X}(s)) = \underline{x}^{*m} = \underline{0}$, for $m > 2$. If $f_{\underline{W}}(\underline{0}) = 0$ (which is almost always the case), the data pair, at time s, contributes 0 to $J_{\mathcal{T}}$ since

$$|D(s) - Y_{\underline{W}}(\underline{X}(s))| = |D(s) - (f_{\underline{W}}(\underline{x}^{*1}) + f_{\underline{W}}(\underline{x}^{*2}))|$$
$$= |1 - (0 + 1)| = 0. \quad (72)$$

On the other hand, its contribution to $J_{\mathcal{T}}^b$ is

$$\sum_{m=1}^{M} |d^m(s) - f_{\underline{W}}(\underline{x}^{*m})| = |d^1(s) - f_{\underline{W}}(\underline{x}^{*1})|$$
$$+ |d^2(s) - f_{\underline{W}}(\underline{x}^{*2})|$$
$$= |1 - 0| + |0 - 1| = 2. \quad (73)$$

Now, rewrite $J_{\mathcal{T}}$ as

$$J_{\mathcal{T}}(\underline{W}) = \frac{1}{T} \sum_{t=1, t \neq s}^{T} |D(t) - Y(\underline{W}, \underline{X}(t))|$$
$$\leq \frac{1}{T} \sum_{t=1, t \neq s}^{T} \sum_{m=1}^{M} |d^m(t) - f_{\underline{W}}(\underline{x}^m(t))|$$
$$< \frac{1}{T} \sum_{t=1, t \neq s}^{T} \sum_{m=1}^{M} |d^m(t) - f_{\underline{W}}(\underline{x}^m(t))| + 2 = J_{\mathcal{T}}^b(\underline{W})$$

leading to a contradiction. In the case $f_{\underline{W}}(\underline{0}) = 1$, the proof is similar, that is, the contribution of the data pair at time s to criterion $J_{\mathcal{T}}$ being $M - 2$ is less than the contribution to criterion $J_{\mathcal{T}}^b$, which is M. This completes the proof of part b) of the Lemma. $\square$

REFERENCES

[1] S. Agaian, J. Astola, and K. Egiazarian, *Binary Polynomial Transforms and Nonlinear Digital Filters.* New York: Marcel Dekker, 1995.

[2] N. Aizenberg and O. Trofimluk, "The conjunctive transform of discrete signals and their applications to finding tests and recognizing monotony of functions of Boolean algebra," *Cybern.*, vol. 5, pp. 138–139, Jan. 1981.

[3] 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.

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

[5] 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. 37, pp. 2037–2066, Dec. 1989.

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

[7] M. Gabbouj and I. Tăbuş, "TUT noisy image database," Tech. Rep. ISBN 951-722-281-5, Signal Processing Lab., Tampere Univ. of Technol., Tampere, Finland, Dec. 1994.

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

[9] K. D. Lee and Y. H. Lee, "Threshold Boolean filters," *IEEE Trans. Signal Processing*, vol. 42, pp. 2022–2036 Aug. 1994.

[10] J.-H. Lin and Y.-T. Kim, "Fast algorithms for training stack filters," *IEEE Trans. Signal Processing*, vol. 42, pp. 772–781, Apr. 1994.

[11] 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. 38, pp. 938–954, June 1990.

[12] A. Schrijver, *Theory of Linear and Integer Programming.* Chichester: Wiley, 1987.

[13] I. Tăbuş, "Training and model based approaches for optimal stack and Boolean filtering with applications in image processing," Ph.D. Dissertation, Tampere Univ. of Technol., Mar. 1995.

[14] I. Tăbuş and M. Gabbouj, "Stacking matrix based fast procedure for optimal stack filter design," in *Proc. SPIE Image Algebra Morphological Image Processing*, San Diego, July 1994.

[15] ———, "Image restoration using Boolean and stack filters," in *Proc. IEEE Workshop Nonlinear Signal Image Processing*, Greece, June 20–22, 1995.

[16] I. Tăbuş, M. Gabbouj, and L. Yin, "Real domain WOS filtering using neural network approximations," in *Proc. IEEE Winter Workshop Nonlinear Digital Signal Processing*, Tampere, Finland, Jan. 1993, pp. 7.2-1.1–6.

[17] I. Tăbuş, D. Petrescu, and M. Gabbouj, "Optimal stack filter design with symmetry constraints," in *Proc. Eusipco-94, VII Euro. Signal Processing Conf.*, Edinburgh, UK, Sept. 1994, pp. 295–298.

[18] 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.

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

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

[21] B. Zeng, H. Zhou, and Y. Neuvo, "Synthesis of optimal detail-restoring stack filters for image processing," in *Proc. 1991 IEEE Int. Conf. Acoust., Speech, Signal Processing*, May 1991, pp. 147–150.
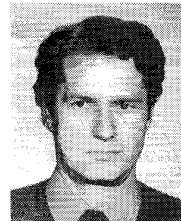
Ioan Tăbuş was born on January 6, 1957 in Romania. He received the M.S. degree in electrical engineering in 1982 and the Ph.D degree in automatic control in 1993 from "Politehnica" University of Bucharest. He received the Ph.D. degree (with honors) in signal and image processing from Tampere University of Technology, Tampere, Finland, in 1995.

In the academic years 1992–1993 and 1994–1995, he was senior researcher at Tampere University of Technology. He was a Teaching Assistant from 1984 to 1990 and a Lecturer from 1990 to 1993 in the Department of Control and Computers at "Politehnica" University of Bucharest, where he currently holds an Associate Professor position. His research interests include neural networks for signal processing applications, system identification, and adaptive nonlinear filtering. He is coauthor of two books and of numerous papers in the fields of automatic control, system identification, and signal processing.

Dr. Tăbuş was the co-recipient of the "Traian Vuia" Award of the Romanian Academy for the year 1989.

Doina Petrescu was born on September 10, 1959 in Romania. She received the M.S. degree in electrical engineering in 1983 and the Ph.D. degree in electronics in 1995 from the "Politehnica" University of Bucharest.

Between January and August 1993, she was a visiting researcher at Tampere University of Technology, Tampere, Finland. Between June 1995 and September 1995, she was a senior researcher at Tampere University of Technology. She was a Researcher from 1983 to 1987 and a Senior Researcher from 1987 to 1991 at the Research Institute for Computer Techniques, Bucharest, Romania. From 1991 to 1995, she was a Teaching Assistant in the Department of Applied Electronics at "Politehnica" University of Bucharest, where she currently holds a Lecturer position. Her research interests include image processing and analysis and adaptive nonlinear filtering.

Moncef Gabbouj (S'85–M'90–SM'95) was born in Monastir, Tunisia, in 1962. He received the B.S. degree in electrical engineering in 1985 from Oklahoma State University, Stillwater, USA. He received the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1986 and 1989.

He is currently a Professor with the Department of Information Technology, Pori, at Tampere University of Technology, Tampere, Finland. From 1994 to 1995, he was Associate Professor with the Signal Processing Laboratory of Tampere University of Technology. From 1990 to 1993, he held the position of Senior Research Scientist with the Research Institute for Information Technology, Tampere, Finland. He is the Director of the International University Program in Digital Signal Processing in the Signal Processing Laboratory at Tampere University of Technology. His research interests include nonlinear signal and image processing and analysis, mathematical morphology, and neural networks.

Dr. Gabbouj is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and was Guest Editor of the Special Issue on Nonlinear Digital Signal Processing (August 1994) of the European journal *Signal Processing*. He is a member and past Chairman, Chairman-Elect and Secretary of the IEEE Circuits and Systems Society, the Technical Committee on Digital Signal Processing, and Chairman of the IEEE SP/CAS Chapter of the IEEE Finland Section. He is also the DSP Track Chair of the 1996 IEEE ISCAS and the Program Chair of NORSIG'96. He is a member of Eta Kappa Nu, Phi Kappa Phi, and the IEEE Signal Processing and Circuits and Systems societies. He and Dr. E. J. Coyle were co-recipients of the Myril B. Reed Best Paper Award of the 32nd Midwest Symposium on Circuits and Systems. He was also co-recipient of the NORSIG'94 Best Paper Award of the 1994 Nordic Signal Processing Symposium.