

Optimal Stack Filtering and Classical Bayes Decision

Bing Zeng, Moncef Gabbouj and Yrjö Neuvo

Signal Processing Laboratory
Tampere University of Technology
P.O. Box 527, SF-33101 Tampere, Finland

Abstract

The optimal stack filtering under the mean absolute error (MAE) criterion is studied in this paper. It is first shown that this problem is equivalent to the classical *a priori* Bayes minimum-cost decision. Generally, a linear program (LP) with $O(b2^b)$ variables and constraints (b is the window width) is required for finding the best filter. Instead, we develop a suboptimal routine which renders the use of the LP obsolete, but yields “reasonably good” filters. Sufficient conditions under which the proposed routine results in optimal solutions are provided and shown to hold in most practical cases. Several design examples are given.

I. INTRODUCTION

Stack filters [1] belong to the class of nonlinear digital filters. The number of stack filters grows faster than $2^{2^{b/2}}$ as the window width b . The optimal stack filtering theory under the MAE criterion has recently been developed [3]. It was shown that finding the minimum MAE stack filter requires the use of a LP with $O(b2^b)$ variables and constraints. This exponentially increasing complexity of the LP will become unsurmountable when the window width increases even slightly. For example, there are more than 10^6 variables and constraints for a 4×4 window mask! In practice, it is almost impossible to solve a LP with such a large number of variables and constraints. Seeking suboptimal solutions thus becomes very important.

This paper is an attempt in this direction. We will develop a suboptimal routine which requires NO LP for finding “reasonably good” filters after we show that optimal stack filtering under the MAE criterion is equivalent to the classical *a priori* Bayes minimum-cost decision. We then seek sufficient conditions under which the proposed routine will result in optimal solutions. These conditions will be demonstrated to be rather natural and therefore hold in most practical cases. They will also be confirmed by several design examples.

II. THRESHOLD DECOMPOSITION AND STACK FILTERS

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

$N_l), \dots, X(n), \dots, X(n + N_r)\}$ where $N_l + N_r + 1 = b$.

Definition 2.1 ([2]): Threshold decomposing a window process $\vec{X}(n)$ produces a set of M binary sequences, called threshold signals, $\vec{x}_1(n), \vec{x}_2(n), \dots, \vec{x}_M(n)$ whose elements are defined by

$$x_l(k) = T_l(X(k)) = \begin{cases} 1 & \text{if } X(k) \geq l \\ 0 & \text{otherwise,} \end{cases} \quad (2.1)$$

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

Consider two sequences (of the same length) \vec{u} and \vec{v} . A property called the *stacking property* holds between \vec{u} and \vec{v} if and only if $u(k) \geq v(k)$ ($v(k) \geq u(k)$) for all k ; \vec{v} (\vec{u}) is said to stack on top of \vec{u} (\vec{v}), denoted as $\vec{v} \leq \vec{u}$ ($\vec{u} \leq \vec{v}$). In the threshold decomposition architecture, it is easy to see that the binary threshold signal on level l stacks on top of all signals on levels $l - 1, l - 2, \dots, 1$.

Definition 2.2 ([1]): A window width b stack filter $S_f(\cdot) : \Omega_b \rightarrow \Omega$ is based on a b -variable positive Boolean function $f(\cdot) : \{0, 1\}^b \rightarrow \{0, 1\}$ operating on the binary threshold signals. The output of the stack filter is obtained by adding up all the binary outputs:

$$\begin{aligned} S_f(\vec{X}(n)) &= S_f\left(\sum_{l=1}^M T_l(\vec{X}(n))\right) = \sum_{l=1}^M S_f\left(T_l(\vec{X}(n))\right) \\ &= \sum_{l=1}^M f\left(T_l(\vec{X}(n))\right) = \sum_{l=1}^M f(\vec{x}_l(n)). \end{aligned} \quad (2.2)$$

In practice, one can implement a stack filters directly in the multi-level domain, since the logical *AND* and *OR* in the binary domain are equivalent to the *MIN* and *MAX* functions in the multi-level domain, respectively.

III. OPTIMAL STACK FILTERING AND CLASSICAL BAYES DECISION

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

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

Generally, the samples in the window process are composed of a signal component and a noise component. In this paper, we always assume that the signal and noise processes

are jointly stationary. Therefore, the MAE is the same at all time.

Assuming that the desired signal $D(n)$ is also discrete and takes values in the set $\{0, 1, \dots, M\}$, we can reduce $C(S_f)$ to (see [3]):

$$C(S_f) = \sum_{j=1}^{2^b} \left[P_f(0|\vec{w}_j) \sum_{l=1}^M \pi_l(1|\vec{w}_j) \pi_l(\vec{w}_j) \right. \\ \left. + P_f(1|\vec{w}_j) \sum_{l=1}^M \pi_l(0|\vec{w}_j) \pi_l(\vec{w}_j) \right]. \quad (3.2)$$

Various notations used in the above equation are explained as follows. \vec{w}_j denotes a binary state of length b . $P_f(0|\vec{w}_j)$ and $P_f(1|\vec{w}_j)$, which take one value of either 1 or 0 and meet

$$P_f(0|\vec{w}_j) + P_f(1|\vec{w}_j) = 1 \quad \forall j, \quad (3.3)$$

are the decision probabilities corresponding to the outputs of the Boolean function $f(\cdot)$ for \vec{w}_j . $\pi_l(\vec{w}_j)$ is the limiting probability of the event that the binary state \vec{w}_j is observed on level l , and the conditional probability

$$\pi_l(0|\vec{w}_j) = 1 - \pi_l(1|\vec{w}_j) = \text{Prob}\{\text{signal value is less than } l|\vec{w}_j \text{ is observed on level } l\}. \quad (3.4)$$

Obviously, Eq. (3.2) can be interpreted as a Bayes decision. As probabilities $\pi_l(0|\vec{w}_j)$ and $\pi_l(\vec{w}_j)$ depend on the prior probability models of the signal, the noise, and the input processes, this Bayes decision belongs to the *a priori* decision. If we interpret stack filtering a multi-level window process by S_f as a decision made upon binary threshold signals, we can see that $C(S_f)$ is the cost function of the decision where the cost coefficients for correct and wrong decisions are 0 and 1, respectively, while $P_f(0|\vec{w}_j)$ and $P_f(1|\vec{w}_j)$ are the outputs of the decision. Following standard decision procedures, we can also incorporate different cost coefficients:

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

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

Then, the cost function is generalized as

$$C(S_f) = \sum_{j=1}^{2^b} \left\{ P_f(0|\vec{w}_j) E_j(0) + P_f(1|\vec{w}_j) E_j(1) \right\} \quad (3.5)$$

where

$$E_j(0) = \sum_{l=1}^M \left[C_l(0, 0, \vec{w}_j) \pi_l(0, \vec{w}_j) + C_l(1, 0, \vec{w}_j) \pi_l(1, \vec{w}_j) \right] \quad (3.6)$$

and

$$E_j(1) = \sum_{l=1}^M \left[C_l(0, 1, \vec{w}_j) \pi_l(0, \vec{w}_j) + C_l(1, 1, \vec{w}_j) \pi_l(1, \vec{w}_j) \right], \quad (3.7)$$

which are the expected costs of deciding a 0 or 1, respectively, for each binary state observed by the filter.

Now, considering the fact that a stack filter upholds the stacking property in its outputs, we can state the design procedure of minimum MAE stack filters as follows [3].

Optimization #3.1:

minimize :

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

subject to :

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

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

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

It has been shown [3] that Optimization #3.1 can be solved by a LP which contains $O(b2^b)$ variables and constraints. As a consequence, one can see that a slight increase in b amounts to a large increase in the complexity and thus prohibits the use of the LP. This unsurmountable (and unfortunate) obstacle appears even in relatively small window sizes, e.g., for a 4×4 window where the number of variable and constraints is more than 10^6 (a million)! In the following section, we will seek suboptimal solutions without resorting to the use of a LP.

IV. A SUBOPTIMAL ROUTINE TO THE STACK FILTER DESIGN

In this suboptimal routine, we first separate the stacking constraints from the programming itself and then solve the optimization in several stages. For window width b stack filters, the 2^b binary window process states are divided into $b+1$ groups according to the number of ones in the states: $\Gamma_i = \{\vec{w}_j : \vec{w}_j \cdot \vec{1}^i = i\}$, $i = 0, 1, \dots, b$, while $\Gamma = \bigcup_{i=0}^b \Gamma_i = \{0, 1\}^b$. Now, at each stage of the optimization, only one among these $b+1$ groups is selected and a LP is solved over it. At the end of each stage, the stacking property is used to decide the outputs corresponding to some binary states in other groups. For instance, if the LP performed over Γ_1 ($b=3$) says that $P_f(1|010) = 1$, then we know, by the stacking property, that $P_f(1|110) = 1$, $P_f(1|011) = 1$, and $P_f(1|111) = 1$. If there still remain some states whose outputs have not been determined yet, we select another group and continue to do a LP over it. A similar procedure is repeated until all 2^b binary states have been assigned output values and thus the positive Boolean function representing the stack filter is completely determined.

The most intuitive starting point of this routine is to select the middle level binary window process state group Γ_B with

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

In the following stages, we choose a group according to Rule A stated below.

Rule A: Set B equal to i which is closest to mid and not all states in Γ_i have been assigned outputs. If there are more than one such a group, select the one with the

largest number of binary states whose outputs are not yet determined.

Based on the above description, the whole optimization procedure can be stated as follows:

Optimization #4.1:

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

Step 2: Minimize :

$$C(S_f) = \sum_{\bar{w}_j \in \Gamma_B} [P_f(0|\bar{w}_j)E_j(0) + P_f(1|\bar{w}_j)E_j(1)]$$

subject to :

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

$$P_f(0|\bar{w}_j) + P_f(1|\bar{w}_j) = 1, \quad \forall \bar{w}_j \in \Gamma_B$$

Step 3: For each $\bar{w}_j \in \Gamma_B$,

$$\text{if } P_f(0|\bar{w}_j) = 0, \text{ then } P_f(0|\bar{w}_i) = 0$$

$$\text{for all } \bar{w}_i \geq \bar{w}_j, \bar{w}_i \in \Gamma,$$

$$\text{if } P_f(0|\bar{w}_j) = 1, \text{ then } P_f(0|\bar{w}_i) = 1$$

$$\text{for all } \bar{w}_i \leq \bar{w}_j, \bar{w}_i \in \Gamma.$$

Step 4: If all states have been decided, then goto Step

5, else, reset B according to Rule A and goto

Step 2.

Step 5: Stop.

The following observation summarizes the computation requirements and the complexity of Optimization #4.1.

Observation 4.1: The LPs in Optimization #4.1 are, in fact, trivial ones in the sense that all computations needed are exclusively data comparisons between the expected costs defined in (3.6) and (3.7). Moreover, an upper bound on the number of comparisons is 2^b .

Proof: Consider a binary state group Γ_i . Any element in the group is neither larger nor smaller than the other elements in the group. So, the minimum value of $C(S_f)$ on group Γ_i can be reached by deciding individually whether $P_f(0|\bar{w}_j)$ or $P_f(1|\bar{w}_j)$ should be equal to one for each $\bar{w}_j \in \Gamma_i$. It is easy to see that this reduces to checking if

$$E_j(0) < E_j(1). \quad (4.1)$$

If the above inequality holds, we choose $P_f(0|\bar{w}_j) = 1$. If $E_j(0) > E_j(1)$, we choose $P_f(1|\bar{w}_j) = 1$. If both sides of (4.1) are equal, one strategy is to choose arbitrarily $P_f(0|\bar{w}_j)$ or $P_f(1|\bar{w}_j)$ to be 1. However, a better way to deal with this case is to leave this state as a *don't care* state temporarily. Its output can be decided later by the stacking property after the outputs of binary window process states in adjacent groups have been decided. This way, the resulting filter would be closer to the optimal filter. For other binary state groups, we do the same data comparisons and accordingly decide their outputs. The proof of the second part of Observation 4.1 is trivial since there are only 2^b different binary states for window width b .

Design Examples: In order to demonstrate the efficiency of the proposed suboptimal routine, let us consider an example which is the same as was given in [4] (Example WW3). The transition matrices of the signal and noise Markov chains are

$$P_S = \begin{matrix} & \begin{matrix} (0, a) & (0, b) & (1, a) & (1, b) \end{matrix} \\ \begin{matrix} (0, a) \\ (0, b) \\ (1, a) \\ (1, b) \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0 & 1 \\ 0.4 & 0 & 0 & 0.6 \end{pmatrix} \end{matrix}$$

and

$$P_N = \begin{matrix} & \begin{matrix} X & Y & Z \end{matrix} \\ \begin{matrix} X \\ Y \\ Z \end{matrix} & \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.8 & 0.1 & 0.1 \\ 0.6 & 0.3 & 0.1 \end{pmatrix}, \end{matrix}$$

respectively. In order to produce observation values for the signal and input process, two nonlinear functions operating on the signal Markov chain and the input process which is composed of the signal and noise Markov chains are defined as

$$f_S(b(n), q(n)) = b(n) \quad (4.2)$$

and

$$f_R(b(n), q(n), N(n)) = \begin{cases} b(n) & \text{if } N(n) = X; \\ 1 & \text{if } N(n) = Y; \\ 0 & \text{if } N(n) = Z. \end{cases} \quad (4.3)$$

Here, the signal and the noise processes are assumed to be independent. Note that any signal generated by (4.2) based on the Markov model with transition matrix P_S is a root signal of the window width three median filter.

Table 1 presents the Boolean functions and corresponding MAE of stack filters designed by using the proposed suboptimal routine. Surprisingly, they are completely the same as those obtained by solving the LP, see Table 2.2 of [4]. This suggests that the suboptimal routine yields the optimal solutions for this case. This is, of course, not always true. However, in the following section we will show that it is true under some natural conditions which hold in most practical cases.

$C(1, 0)$	$f(x_1, x_2, x_3)$	MAE
0.1	$x_1 x_2 x_3$	0.044
0.2	$x_1 x_2$	0.082
0.5	$x_2(x_1 + x_3)$	0.127
1.0	x_2	0.178
2.0	$x_2 + x_1 x_3$	0.239
5.0	$x_2 + x_3$	0.365
6.0	$x_1 + x_2 + x_3$	0.375

V. OPTIMAL CONDITIONS

First, we present an intuitive theorem concerning the case when the solution to Optimization #4.1 is optimal in the minimum MAE sense.

Theorem 5.1: Suppose that Optimization #4.1 is solved ignoring Step 3 (that is, the stacking property is not used to set the outputs for binary states in other groups). If this procedure results in a positive Boolean function, then this solution is optimal in the MAE sense.

See [6] for the proof. The above Theorem is the key Theorem to the more explicit sufficient conditions we are seeking. To this goal, let us consider the binary case first,

where we assume that the cost coefficients for correct decisions are zeros and those for wrong decisions are the same for all binary states. Under these assumptions, Eq. (3.2) reduces to

$$C(f) = \sum_{j=1}^{2^b} \left[P_f(0|\bar{w}_j)C(1,0)\pi(1|\bar{w}_j) + P_f(1|\bar{w}_j)C(0,1)\pi(0|\bar{w}_j) \right] \pi(\bar{w}_j). \quad (5.1)$$

Definition 5.1: The conditional probabilities $\pi(0|\bar{w}_j)$'s are said to possess the stacking property if the following inequality holds for all binary states:

$$\pi(0|\bar{w}_i) \leq \pi(0|\bar{w}_j) \quad \text{whenever} \quad \bar{w}_j \leq \bar{w}_i. \quad (5.2)$$

Based on this definition, we have the following theorem.

Theorem 5.2: If the conditional probabilities $\pi(0|\bar{w}_j)$'s of the input binary window process possess the stacking property, then the stack filter designed by solving the suboptimal routine is optimal under the MAE criterion.

Proof: Consider two binary states \bar{w}_i and \bar{w}_j . According to the suboptimal routine, we can obtain the outputs for \bar{w}_i and \bar{w}_j by comparing both sides of (4.1) which now reduce to

$$C(1,0)\pi(1|\bar{w}_j) < C(0,1)\pi(0|\bar{w}_j). \quad (5.3)$$

If these two binary states are incomparable, there is no need to compare their outputs. The reason is that, in the optimization procedure, decision outputs of two binary states are constrained by the stacking property if and only if one of the binary states stacks on top of the other. Therefore, let us focus on binary state pairs in which the two states possess the stacking property. Without loss of generality, we assume that \bar{w}_j stacks on top of \bar{w}_i , i.e., $\bar{w}_i \geq \bar{w}_j$. Now, $\pi(0|\bar{w}_j) \geq \pi(0|\bar{w}_i)$ (and therefore $\pi(1|\bar{w}_j) \leq \pi(1|\bar{w}_i)$) guarantees that if (5.3) holds for \bar{w}_i producing a 0 decision output, then (5.3) must also hold for \bar{w}_j producing a 0 decision output. This means that $P_f(1|\bar{w}_i) \geq P_f(1|\bar{w}_j)$. Similarly, one can show that if $\bar{w}_j \geq \bar{w}_i$, then $P_f(1|\bar{w}_j) \geq P_f(1|\bar{w}_i)$. (Note that here the stacking property was not used to set the output for any binary state). This is exactly equivalent to the stacking property. Therefore, according to Theorem 5.1, the suboptimal routine produces optimal solutions.

The stacking property in the conditional probabilities defined in Definition 5.1 is a natural condition. It means that the more ones in a binary window process state, the less probable it is that the true signal value is zero. Therefore, it is easy to understand that this property should hold in most practical cases. During the calculation of $\pi(0|\bar{w}_j)$'s for the examples presented in the last section, we have found that the stacking property of Definition 5.1 holds. Therefore, it is a consequence of Theorem 5.2 that the results in Table 1 are optimal in the MAE sense.

The above result can easily be extended to multi-level signals as follows.

Definition 5.2: The expected costs $E_j(0)$ and $E_j(1)$ are said to possess the stacking property if, for \bar{w}_j we have $E_j(0) < E_j(1)$, then the inequality $E_i(0) < E_i(1)$ must hold for all binary states $\bar{w}_i < \bar{w}_j$.

Based on this definition, we have the following theorem.

Theorem 5.3: Suppose that the expected costs $E_j(0)$ and $E_j(1)$ possess the stacking property. Then the stack filter designed by solving the suboptimal routine is optimal under the MAE criterion.

The proof is analogous to that of Theorem 5.2 and it is therefore omitted. The stacking property defined for the expected costs has a similar interpretation as that for the conditional probabilities in the binary case. It says that as the number of zeros in a binary state increases, the cost of deciding a 0 or 1 output decreases or increases, respectively. Therefore, if, for a given binary state, the cost of deciding a 0 is less than the cost of deciding a 1, then for any other binary state stacking on top of this state, the cost of deciding a 0 is also less than that of deciding a 1.

For the binary case, we can generalize and weaken the sufficient condition. This is presented in the following theorem (see [6] for the proof).

Theorem 5.4: For each binary state \bar{w}_j such that

$$\pi(0|\bar{w}_j) < \frac{C(1,0,\bar{w}_j)}{C(0,1,\bar{w}_j) + C(1,0,\bar{w}_j)}, \quad (5.4)$$

if there does not exist \bar{w}_i such that $\bar{w}_i \geq \bar{w}_j$ and \bar{w}_i violates (5.4), then the suboptimal solution produced by Optimization #4.1 is optimal in the minimum MAE sense.

Remark: Theorem 5.4 is more general than Theorem 5.2 since we use different cost coefficients for wrong decisions for different binary states. On the other hand, the condition of Theorem 5.4 is looser than that of Theorem 5.2 because (5.2) need not hold for those \bar{w}_j 's which violate or do not violate (5.4). That is, it is possible that the conditional probabilities for certain stacking binary states do not possess the stacking property and yet the suboptimal solution is optimal.

VI. CONCLUSION

This paper showed, among other things, how the complexity of finding optimal stack filters can be reduced and thus allowing larger, i.e., more interesting, optimization problems to be solved. Although, all stack filter examples presented in the paper are based on Markov chain processes, which would exclude many practical cases, attempts to synthesize optimal stack filters for general image processing applications have been investigated, see [5], where the suboptimal routine proposed here was successfully applied to find optimal filters.

REFERENCES

- [1] P. D. Wendt, E. J. Coyle, and N. C. Gallagher, "Stack filters," *IEEE Trans. ASSP*, pp. 898-911, Aug. 1986.
- [2] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Median filtering by threshold decomposition," *IEEE Trans. ASSP*, pp. 1183-1188, Dec. 1984.
- [3] E. J. Coyle and J.-H. Lim, "Stack filters and the mean absolute error criterion," *IEEE Trans. ASSP*, pp. 1244-1254, Aug. 1988.
- [4] M. Gabbouj, "Optimal stack filter examples and positive Boolean functions," M.S. Thesis, School Elec. Eng., Purdue Univ., West Lafayette, IN, Dec. 1986.
- [5] B. Zeng, H. Zhou, and Y. Neuvo, "Synthesis of optimal detail-restoring stack filters for image processing," *Proc. of ICASSP'91*, this conference.
- [6] 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. CAS*, submitted.