Signal Estimation Techniques Using Lp-Norm Optimal Stack Filters with Applications to Image and Video Processing

Cristian Emanuel Savin

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montréal, Québec, Canada

July 1997

0-612-39787-4

ABSTRACT

Signal Estimation Techniques Using Lp-Norm Optimal Stack Filters with Applications to Image and Video Processing

Cristian Emanuel Savin , Ph.D.
Concordia University , 1997

The class of median-related operators called the stack filters are well-known for outperforming linear filtering in many applications which involve restoration of signals corrupted with impulsive noise, when sharp edges are to be preserved. Various subclasses of stack filters, such as the rank order operators and the weighted median filters, show a great deal of promise in finding commercial applications in image processing and image sequence filtering for advanced television systems. Moreover, the stack filters being closely related to morphological signal processing, the theory of their design and implementation constitutes a very attractive framework for the development of algorithms for low-level machine vision. This thesis, therefore, is concerned with the problems of efficient design and implementation of stack filters. The focus is on exploring the possibility of using the L_p norm optimality criterion for the design of stack filters, and on the development of hardware-oriented algorithms for the implementation of these filters.

The problem of designing optimal stack filters by employing a general objective function given as the L_p norm of the error between the desired signal and the estimated one is addressed. This design problem is formulated as an optimization problem, in which a positive Boolean function is determined such that the L_p norm of the error in signal estimation using stack filters is minimized. It is shown that the L_p norm can be expressed as a linear combination of the responses of the positive Boolean function to all possible binary input vectors. Based on this error formulation, it is established that an L_p-norm optimal stack filter can be determined as the solution of a linear program. It is shown that for the specific problem of restoring images corrupted with impulsive noise, the L_p-optimal stack filters with $p \geq 2$ are capable of removing the noise much more effectively and provide a better visual performance than achieved by the conventional minimum mean absolute error stack filters.

The time-area complexities of the conventional parallel and bit-serial algorithms for stack filtering depend on the number of grey levels of an input image (signal). In this thesis, two new hardware-oriented algorithms for multidimensional stack filtering, for which the time-area complexity depends on the size of the filter, are developed. The new algorithms achieve significantly increased computational efficiency compared to the conventional algorithms for stack filtering by evaluating the Boolean function at thresholds corresponding to the sample-values within the filter-window and by taking advantage of the fact that, in most image processing applications, many pixels appearing in a filter-window assume non-distinct values. In one algorithm, the output of the filter is determined iteratively by employing a divide-and-conquer strategy. This strategy is based on successively partitioning the filter window into two sets with elements larger and smaller than the current threshold level, respectively. The other algorithm uses a binary tree search method for stack filtering in conjunction with a technique of compressing the dynamic range of the window samples.

Motivated by the fact that in typical images, the details of the scene are locally situated, a hardware-oriented design and implementation of locally optimal mean absolute error rank order filters is developed and applied to the problem of intrafield deinterlacing of video signals.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my supervisors, Dr. M. Omair Ahmad and Dr. M.N.S. Swamy, for their guidance, support and encouragement, that made this research possible. I am also grateful for their inspiration during the course of this work, as well as for their careful and critical review of my thesis. I am also thankful to all the members of the examination committee, Dr. A. Antoniou, Dr. T. Fancott, Dr. E.I. Plotkin, Dr. A.K. Elhakeem and Dr. Y. Wang, for their valuable comments and suggestions.

I would also like to express my profound gratitude to my parents, for their love and dedication in shaping up my life. I owe special thanks to my wife, Daniela, and to my son, George, for their patience and understanding for the entire period of this research.

Many thanks are due to all my close relatives and friends for their help, advice and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

$\{\boldsymbol{b}_j\}$	The set of all Boolean vectors of size M ($j = 1, 2, \ldots, 2^{\mathrm{M}}$)
BSWP	Bit-serial window-partitioning (algorithm)
BTS	Binary tree search
$E[\,\lvert e_\ell(n)\rvert\,]$	Mean absolute error incurred at the level ℓ of a stack filter configuration, when the Boolean operator f is used at that level
f	A Boolean function
f_ℓ	Binary value at the level ℓ in the threshold decomposition of the output of stack filter $\mathcal{S}_f$ at the instant n, $\mathcal{S}_f(\mathbf{x}(n))$
F_n	Window sequence coding transformation
$\mathcal{GS}_{\{f\}}$	Generalized stack filter characterized by a set of Boolean functions denoted by $\{f\}$
K	$\mathrm{K} = \log_2 \mathrm{L}$, where L denotes the number of quantization levels of a signal
ℓ	Generic notation designating a level of a stack filter: $\ell = 1, 2, \ldots, \mathrm{L} - 1$
L	Number of quantization levels (grey levels) of a signal. It is assumed that L is a power of 2, i.e., $\mathrm{L} = 2^{\mathrm{K}}$
LSPBF	Linearly separable PBF
M	Size of the filter window
$\tilde{\mathrm{M}}$	Number of distinct grey level values of a sequence $\mathbf{x}(n)$
MAE	Mean absolute error
MMAE	Minimum mean absolute error
MSE	Mean square error
PBF	Positive Boolean function

ROF	Rank order filter
ROSM	Rank order state machine
ROSM-BTS	ROSM-based BTS
s_ℓ	Binary value at the level ℓ in the threshold decomposition of the desired signal S
$\mathcal{S}_f$	Stack filter operator characterized by the PBF f
S	Desired signal in the context of signal estimation using stack filters
WOS	Weighted order statistic (filter)
WSC	Window sequence coding (technique)
WSC-BTS	WSC-based BTS (algorithm)
$\boldsymbol{x}_\ell(n)$	Binary vector (threshold sequence) obtained by applying the thresholding operator δ to the elements of $\mathbf{x}(n)$
$\mathbf{x}(n)$	Input sequence appearing in the filter window at the time instant n, $\mathbf{x}(n) = \{\mathrm{x}_0, \mathrm{x}_1, \ldots, \mathrm{x}_{\mathrm{M}-1}\}$
$\mathbf{x}_o(n)$	A set obtained from $\mathbf{x}(n)$ by sorting the elements of $\mathbf{x}(n)$ in decreasing order; $\mathbf{x}(n) = \{\mathrm{x}_{(0)}, \mathrm{x}_{(1)}, \ldots, \mathrm{x}_{(\mathrm{M}-1)}\}$
$\mathbf{x}^*(n)$	A set obtained from $\mathbf{x}(n)$ by taking each element of $\mathbf{x}(n)$ at its first occurrence, when the elements of $\mathbf{x}(n)$ are scanned through from left to right
$\tilde{\mathbf{x}}(n)$	A sequence obtained from $\mathbf{x}(n)$ by applying the WSC transformation to each element of $\mathbf{x}(n)$
$\delta(\mathrm{x}_i - \ell)$	Threshold operator which decomposes a sample x_i in the multilevel window-sequence $\mathbf{x}(n)$ into $\mathrm{L} - 1$ binary numbers; $\delta(\mathrm{x}_i - \ell)$ assumes a value of 1 if x_i is greater than or equal to ℓ, and it assumes a value of 0 if x_i is smaller than ℓ

Chapter 1

Introduction

1.1 General

Linear filters have long been considered as the primary tools for signal and image processing. They are easy to be implemented and analyzed, and they are well-suited for minimum mean square error estimation of signals in the presence of additive Gaussian noise. Despite the availability of elegant tools of linear systems theory for the analysis and design of linear filters, not all signal processing problems can be satisfactorily addressed through the use of such filters [60], [61].

- *Detection or preservation of edges.* In many image processing problems it is very important to avoid the edge blurring effect associated with linear filtering. This effect can be effectively dealt with by employing nonlinear techniques.

- *Suppression of non-Gaussian noise.* Signals are often corrupted with noise due to bit errors in transmission or processing, and are subjected to interference effects due to the use of coherent imaging systems. None of these signal corruption processes can be adequately modelled as additive Gaussian noise — very often, they are impulsive and/or non-additive in nature. These situations again call for the application of nonlinear tools.

- *Modelling of the human perception system.* The problem of building models for the human perception system is a very difficult task. It has been found that certain parts of this system involve nonlinearities [43]. For instance, it is widely accepted that the human vision system employs logarithmic nonlinearities. Furthermore, it has also been established that many high-level mechanisms of human perception are decision-driven. Some of these decisions appear as hard nonlinearities, e.g., maximum/minimum selection, thresholding, or comparisons. Therefore, suitable modelling and analysis of such systems have to employ suitable nonlinear techniques.

While it is generally true that nonlinear systems are more complex than linear ones, in view of the current advances in the area of microelectronics, the today's technology is more ready than ever before to face the challenges of complex nonlinear problems. Encouraged by these technological advances, a large number of researchers have dedicated their efforts to the development of the underlying theory and techniques of nonlinear signal processing. Some of the widely accepted techniques falling in this domain are rank order based filtering [29], [66], polynomial based algorithms [52], [42], morphological methods [50], and homomorphic processing [55]. Among all these nonlinear techniques, the rank order based filters are especially attractive, since they are expected to play a very important role in the development of the emerging multi-media and communication systems. As it is known, these filters are very effective for signal estimation in the presence of impulsive noise when sharp edges are to be preserved. This thesis is an attempt to broaden the theoretical understanding of rank order based filters and develop new and efficient algorithms and architectures for their implementation.

1.2 An Overview of Rank Order Based Filtering

In the 1970's, Tukey introduced the "running median" as a tool for smoothing discrete data [82], [83]. Since then its name has changed to "median filter" and it has found many applications in speech and image processing [38] [39], [61]. To compute the output of a median filter, M input signal samples in a window are sorted and the middle (median) value is chosen as the filter's output. An immediate extension of the median filter is the class of rank order filters [30]. These filters operate exactly like the median filter except that they output the ith largest sample in a filter window, instead of the sample of the middle rank.

The performance of median filters is often evaluated qualitatively. As it is known, the performance of a filter is an expression of how well it can suppress the undesired parts of a signal and, equally important, how well the desired information can be retained. In the case of linear filters, the two objectives can be easily achieved when the useful and the undesired components of the signal occupy different frequency bands. Unfortunately, in the case of the running median, there is no simple way of formalizing the filtering problem with these objectives and evaluating the performance using the frequency analysis or step and impulse response[1] techniques. As a result, different tools, deterministic and statistical, had to be developed to analyze and characterize the behavior of these nonlinear filters. The basic descriptor of the deterministic properties of a rank order filter is the set of its root signals; a root of a rank order filter is a signal which is invariant to further passes of the filter. The basic statistical descriptor of a rank order filter is the set of its output distributions, that are generally used to study the noise attenuation properties of these filters.

Since the output of the median filter is always one of the input samples, it is conceivable that certain signals could pass through the median filter unaltered. In [83] several examples of signals have been given to demonstrate that successive passes of the median filter of size 3 will eventually force a signal of finite length to a form which is invariant to further passes of the filter. Later, Gallagher and Wise have proved that a signal of

[1] The fact that the step response of a median filter is also a step is nevertheless an indication that a median filter is capable of preserving stepwise abrupt changes (edges) in signals. Similarly, the fact that the impulse response of the median filter is zero is an indication that this filter could be very effective in suppressing impulsive noise.

finite length is a median root, if it consists of constant neighborhoods and edges only [30], [84]. They also proved that repeated median filtering of any finite length signal will result in a root signal after a finite number of passes. This property of median filters is very significant and is called the "convergence property". The understanding of the convergence behavior of rank order filters has continuously evolved, leading to the development of a theory for the "structural" behavior of median-type filters [19], [29]. The goal of this theory is to determine the type and number of root signals, and the structures[2] that are preserved, created or modified by rank order based filters.

The studies of the structural behavior of median filters complemented the results known for quite some time in the statistics literature regarding the median and other order statistics [29], [35]. For instance, the good performance of the median filter with a Laplacian distributed noise is due to an important optimality property of the sample median. Specifically, the sample median is known to be the maximum likelihood estimate for the location parameter of the Laplacian density. Also, as shown in [80], the conditional median at each time instant t is the minimum mean absolute error estimator of the signal value at time t, where the conditioning is on the past history up to time t of the noise-corrupted observations of the signal.

As a result of this investigation on the properties of rank order filters, their success in various signal processing applications is now well understood. However, this investigation has also revealed a number of drawbacks of rank order filters, such as the effects of edge jittering [12] and streaking [10] in image processing applications. The inadequacy of the rank order filters in certain situations encountered in image processing led to the investigation and consequent evolution of different types of rank order based filters. Two main types of rank order based filters have been traditionally given a special attention. The first type includes filters in which rank order operations are combined with linear operations in some fashion [11], [48]. It has been shown in [25], that this type of rank order based filters has a major drawback, in that the optimization over such classes of filters

[2] A structure is any local or global variation of interest in the magnitude of a signal (e.g., constant neighborhoods, edges, impulses, oscillations, etc).

is mainly reduced to optimizing the linear part of the filter. The class of stack filters is the second major type of rank order based filters [17], [86]. They perform nonlinear rank order based operations, and have been shown to overcome the drawback of the first type of filters.

The concept of stack filtering has been introduced as a result of a theoretical investigation on the realization of rank order operators using Boolean functions. In [23] and [24], Fitch et al. have shown that rank order filters possess a limited superposition property, which states that the rank order filtering of an arbitrary multilevel sequence is equivalent to the decomposing of the signal into binary sequences by thresholding, filtering each binary sequence by a binary rank order filter, and then reversing the decomposition process. This equivalence reduces the processing of multilevel sequences to that of binary sequences. The configuration in which a multilevel sequence is reduced to binary sequences has been designated as a threshold decomposition architecture [24].

In addition to the above mentioned limited superposition property of rank order filters in the threshold decomposition architecture, another property, referred to as the stacking property, was identified by Wendt et al. [86]. The stacking property is an ordering property which refers to the fact that in the threshold decomposition architecture, the outputs of the binary rank order filters have the structure of a stack of 0's piled on top of a stack of 1's.

The definition of stack filters was given by abstracting the properties of rank order filters in the threshold decomposition configuration [86]. Specifically, in a stack filter architecture, the binary rank order based operator is characterized by an arbitrary positive Boolean function [3] (PBF). An optimality theory for the design of stack filters under the mean absolute error (MAE) criterion was developed by Coyle and Lin [17]. Based on this theory, it has been possible to determine the stack filter that minimizes the MAE between its output and a desired signal, given a noise-corrupted observation of the latter. As shown in [17], the MAE-optimal stack filter can be determined by employing the

[3] The concept of positive Boolean function is formally defined in Section 2.1.1.

technique of linear programming which in this thesis is referred to as simply linear program [16]. Although the MAE criterion of designing stack filters has been extensively used for image processing applications, higher order errors have been used only in some very restrictive designs, in which the signal is assumed to be a constant embedded in white noise [89], [90]. However, a solution to the general problem of designing stack filters using higher-order error criteria remains an open problem.

Two types of architectures, parallel and bit-serial, have been traditionally used for the implementation of stack filters [13], [49], [86]. The parallel architecture [86] uses PBF circuits at each level in the threshold decomposition configuration, while the bit-serial structure achieves a lower time-area complexity by employing a binary tree search (BTS) technique and by utilizing a single PBF circuit. The time-area complexities of the conventional architectures for stack filtering depend on the number of quantization levels of an input signal (e.g., the number of grey levels of an input image). However, as pointed out in [1] and [46], it is expected that more efficient stack filtering techniques, with time-area complexities determined by the size of the filter, could be developed. Nevertheless, a definitive solution to this problem is still lacking.

1.3 Scope and Organization of the Thesis

As discussed in the previous section, the stack filter theory constitutes a very attractive framework for dealing with nonlinear filtering problems. Motivated by the recent successes of using stack filters in image and video processing applications [94], [21], [65], [59], this investigation is concerned with the problems of efficient design and implementations of stack filters. The focus is on exploring the possibility of extending the MAE design of stack filters to an L_p norm-based design, and on the development of hardware-oriented algorithms for the implementation of the filters designed using this general optimality criterion.

The thesis is organized as follows. In Chapter 2, a review of the fundamentals of stack filter theory and the relevant background material is presented. The problem of designing stack filters that are optimal for signal estimation under the MAE criterion is reviewed, and a new and direct approach for the derivation of the binary-level expression of the MAE is presented. The conventional parallel and bit-serial architectures for stack filtering are also described.

In Chapter 3, the possibility of designing optimal stack filters by employing an L_p norm of the error between the desired signal and the estimated one, is investigated. The design problem is formulated as an optimization problem, in which a positive Boolean function is determined such that the L_p norm of the error in signal estimation employing a stack filter structure is minimized. It is shown that the L_p norm can be expressed as a linear combination of the responses of the positive Boolean function to all possible binary input vectors. Based on this error formulation, an L_p-norm optimal stack filter can be determined as the solution of a linear program.

In Chapters 4 and 5, the problem of developing new and more efficient bit-serial algorithms and architectures for stack filtering is investigated. The proposed solutions reduce the computational complexity of stack filtering by evaluating the Boolean function at threshold levels corresponding to the sample-values within the filter-window, and by taking advantage of the fact that, in most image processing applications, many pixels appearing in a filter-window assume non-distinct values. In Chapter 4, a bit-serial window-partitioning algorithm for stack filtering is proposed, while Chapter 5 is dedicated to the derivation of an input compression-based BTS algorithm.

In Chapter 6, the possibility of employing stack filtering techniques to solve real-world problems in image sequence processing is investigated. The success of median-related operators in image filtering applications has stimulated an intense investigation to understand their relevance in solving image and image sequence processing problems [7],

[8], [66], [96]. As a result, it is now clear that various subclasses of stack filters, such as the rank order filters (ROFs) and the weighted median operators, show a great deal of promise to find commercial applications in image sequence filtering for advanced television systems [64], [85]. Motivated by the fact that in typical images, the details of the scene are locally situated, in Chapter 6, a hardware-oriented design and implementation of locally optimal mean absolute error rank order filters is proposed. The designed locally optimal MAE ROFs are applied to the problem of intrafield deinterlacing of video signals.

Chapter 7 concludes the thesis by highlighting the contributions of this research.

Chapter 2

Stack Filter Theory

A stack filter is a rank order based filter whose output at each window position is the result of a superposition of the outputs of a stack of Boolean functions operating on thresholded versions of the samples appearing in the filter window. In this chapter, some basic concepts of Boolean algebra, frequently encountered in stack filter theory, are briefly reviewed, and the formal definition and the properties of stack filtering, as given in [86], are presented. The problem of signal estimation using minimum mean absolute error (MMAE) stack filters [17] is formulated, and the conventional parallel [86], bit-serial [13], and compression-based [1] architectures for stack filtering are briefly described. In the process of reviewing the fundamental concepts of stack filter theory, several questions concerning some important issues in the design and implementation of stack filters are also raised.

2.1 Background

In this section, employing the switching functions terminology as proposed by Muroga [53], some basic concepts which are used in stack filter theory are briefly reviewed. The definition of a positive Boolean function [53] is stated, and the stacking property of Boolean functions, which is essential to the understanding of stack filtering, is described in terms of the stacking relation between the Boolean vectors [23].

2.1.1 Positive Boolean Functions

To introduce the concept of a positive Boolean function, some basic definitions of Boolean algebra are now recalled [53].

Definition 2.1 A literal is defined as a binary variable x, or its complement $\overline{x}$. A conjunction (logical product) of literals in which a literal for each variable appears at most once is called a term. Similarly, a disjunction (logical sum or alternation) of literals in which a literal for each variable appears at most once is called an alterm. In a special case, a term or an alterm may consist of a single literal. A disjunction of terms is called a disjunctive form or normal form. A conjunction of alterms is called a conjunctive form.

For example, x_0 and $\overline{x}_0$ are the literals of the variable x_0. Both the expression $x_0x_1+x_0+x_1$ and $x_0+\overline{x}_0x_1$ are disjunctive forms which are equivalent to the Boolean function x_0+x_1. However, the expression $x_0+x_1(\overline{x}_0+\overline{x}_1)$ is not a disjunctive form, although it is also equivalent to the same Boolean function, x_0+x_1.

Definition 2.2 If there exists a disjunctive form for a Boolean function $f(x_0, x_1, \ldots, x_j, \ldots, x_{M-1})$, such that the literal $\overline{x}_j$ does not appear in any term of this form, then f is said to be positive in the variable x_j. If a Boolean function f is positive in all its variables $x_0, x_1, \ldots, x_{M-1}$, then f is said to be a positive Boolean function (PBF).

To introduce a result due to Quine [62] suggesting a method to determine whether or not a Boolean function is positive, we recall the concept of minimum sum-of-product form of a binary function as given in [53].

Definition 2.3 Let f and g be two Boolean functions. If for every Boolean vector $(x_0, x_1, \ldots, x_{M-1})$ for which $f(x_0, x_1, \ldots, x_{M-1}) = 1$, we also have $g(x_0, x_1, \ldots, x_{M-1}) = 1$, we write $f \subseteq g$ and say that f implies g. If, in addition, there exists at least one Boolean vector $(b_0, b_1, \ldots, b_{M-1})$ for which $f(b_0, b_1, \ldots, b_{M-1}) = 0$ and $g(b_0, b_1, \ldots, b_{M-1}) = 1$, we write $f \subset g$ and say that f strictly implies g. If an implication relation holds between two binary functions f and g (i.e., $f \subseteq g$, $f \subset g$, $g \subseteq f$, or $g \subset f$), then f and g are said to be implication comparable; otherwise, they are implication incomparable.

Definition 2.4 An implicant of a Boolean function f is a term that implies f. A term P is said to subsume another term Q, if all the literals of Q are literals of P as well. A prime implicant of a binary function f is defined as an implicant of f such that no term subsumed by this implicant can be an implicant of f.

For instance, x_0, x_0x_1, $x_0\overline{x}_1$, and x_1x_2, are examples of implicants of the function $f = x_0 + x_1x_2$. The terms x_1 and x_2 are implication incomparable with f. The term x_1x_2 subsumes both the terms x_1 and x_2. The term x_1x_2 is a prime implicant of f.

Definition 2.5 A minimum sum-of-product form of a Boolean function f is a disjunction of prime implicants, such that the removal of any one of these makes the remaining expression no longer equivalent to the original f.

An important theorem given by Quine [62] shows that a PBF has a unique minimum sum-of-product form. Thus, in order to determine whether or not a Boolean function f is a PBF, one could simply find a minimal sum-of-product form for f: the function f is positive if and only if this minimal sum-of-product form is itself positive.

2.1.2 Stacking Property of Boolean Filters

The stacking (increasing) property of a Boolean filter (function) is stated in terms of an ordering (stacking) relation between Boolean vectors [23], [24]. This ordering relation, which is commonly used in Boolean algebra, is now defined [53].

Definition 2.6 If two Boolean vectors $\boldsymbol{a} = (a_0, a_1, \ldots, a_{M-1})$ and $\boldsymbol{b} = (b_0, b_1, \ldots, b_{M-1})$ satisfy the condition that $a_j \geq b_j$ for all $j = 0, 1, \ldots, M-1$, then we write $\boldsymbol{a} \geq \boldsymbol{b}$ and say that $\boldsymbol{a}$ is greater than or equal to $\boldsymbol{b}$. In particular, if there exists an index k for which $a_k > b_k$, and $a_j \geq b_j$ for all $k \neq j$, then $\boldsymbol{a}$ is said to be greater than $\boldsymbol{b}$, i.e., $\boldsymbol{a} > \boldsymbol{b}$.

Definition 2.7 When for two Boolean vectors, $\boldsymbol{a}$ and $\boldsymbol{b}$, none of the stacking relations, $\boldsymbol{a} \geq \boldsymbol{b}$, $\boldsymbol{b} \geq \boldsymbol{a}$, $\boldsymbol{a} > \boldsymbol{b}$, or $\boldsymbol{b} > \boldsymbol{a}$, holds, then the two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are said to be incomparable. Otherwise they are said to be comparable (by the stacking relation).

The stacking property of a Boolean filter can now be stated as given below [23], [53].

Property 2.1 A Boolean filter f is said to satisfy the stacking (increasing) property if and only if $f(\boldsymbol{a}) \geq f(\boldsymbol{b})$ for all possible pairs of comparable vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with $\boldsymbol{a} \geq \boldsymbol{b}$.

The necessary and sufficient condition for a Boolean filter to satisfy the stacking property was stated by Gilbert [32], [53].

Theorem 2.1 A Boolean filter satisfies the stacking property if and only if it is a PBF.

The stacking property of a PBF is highlighted by employing the Hasse diagram [53]. The derivation of this diagram is now briefly described. Let us denote by $V \triangleq \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{2^{\mathrm{M}}}\}$, the set of all 2^{M} possible binary vectors of size M. In the Hasse diagram, each binary vector $\boldsymbol{b}_j$, $j = 1, 2, \ldots, 2^{\mathrm{M}}$, is represented by a vertex. The set V is partitioned into subsets V_k, $k = 0, 1, \ldots, \mathrm{M} - 1$, such that all Boolean vectors (vertices) belonging to a subset V_k have exactly k entries which are equal to 1, i.e.,

$$V_k \triangleq \{\boldsymbol{b} \mid \boldsymbol{b} \cdot 1_{\mathrm{M}\times 1} = k,\ \boldsymbol{b} \in \mathrm{B}\} \quad , \qquad k = 0, 1, \ldots, \mathrm{M} \ , \tag{2.1}$$

where $\mathbf{1}_{\mathrm{M}\times 1}$ is a column vector of size M whose elements are all 1's. Note that all the vectors which belong to the same subset V_k are incomparable. In the Hasse diagram, the vertices are arranged in $\mathrm{M} + 1$ consecutive lines, such that the jth line, $j = 0, 1, \ldots, \mathrm{M}$, contains the vertices (Boolean vectors) in the set $V_{\mathrm{M}-j}$. Each vertex is represented by an ellipse labeled with the associated Boolean vector. Each pair of stacking comparable vertices $(\boldsymbol{b}_i, \boldsymbol{b}_j)$ placed on consecutive lines (i.e., $\boldsymbol{b}_i \in V_k$ and $\boldsymbol{b}_j \in V_{k+1}$, $k = 0, 1, \ldots, \mathrm{M} - 1$), is represented by an undirected edge conecting $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$. The Hasse diagram of a set of Boolean vectors V can be formally defined as given below [53], [27].

Definition 2.8 The Hasse diagram of the set of all Boolean vectors of size M is the undirected graph $G = (V, E)$, where V is the set of all vertices, with each vertex representing a binary vector, and E denotes the set of all edges, i.e.,

$$E \triangleq \{ (\boldsymbol{b}_i, \boldsymbol{b}_j) \mid \boldsymbol{b}_i \in V_k , \boldsymbol{b}_j \in V_{k+1} , \ \boldsymbol{b}_j > \boldsymbol{b}_i , \ k = 0, 1, \ldots, \mathrm{M} - 1 \} \ . \tag{2.2}$$

The stacking diagram of a Boolean function f of size M is a representation of the truth table of f employing the Hasse diagram of the set of all input Boolean vectors of size M. Specifically, each vertex of the Hasse diagram correponding to an input vector for which the output of f assumes a value of 1 is now represented as a shaded ellipse. The vertices associated with input vectors for which $f = 0$ are left unshaded [34], [27].

Figure 2.1 illustrates the stacking diagram for the function $f(x_0, x_1, x_2, x_3) = x_1 + x_0x_2 + x_2x_3$, which is a PBF, and therefore it satisfies the stacking property. Thus, for each edge $(\boldsymbol{p}1\boldsymbol{q}, \boldsymbol{p}0\boldsymbol{q})$ we have that $f(\boldsymbol{p}1\boldsymbol{q}) \geq f(\boldsymbol{p}0\boldsymbol{q})$. For instance, because $f(0100) = 1$ we also have $f(1100) = 1$, $f(0101) = 1$, and $f(0110) = 1$. Analogously, since $f(1001) = 0$, we also have $f(1000) = 0$ and $f(0001) = 0$.

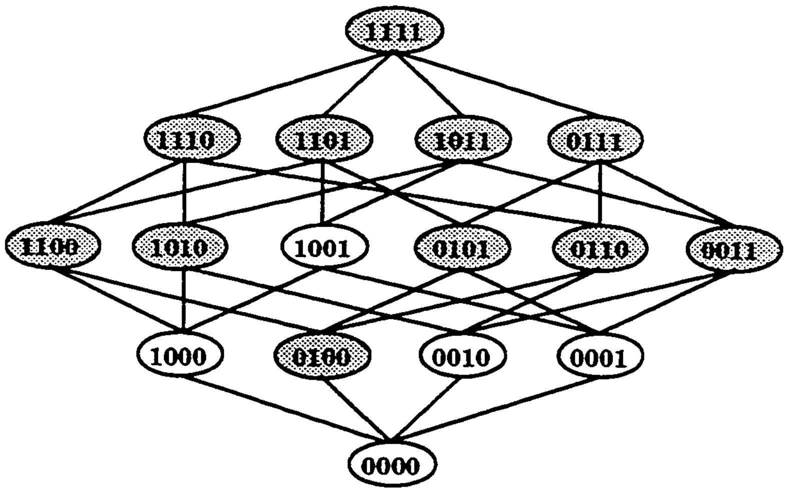


Figure 2.1: Stacking diagram of the Boolean filter $f(x_0, x_1, x_2, x_3) = x_1 + x_0x_2 + x_2x_3$, where a shaded ellipse representing a binary vector $\boldsymbol{b}_j$ signifies that $f(\boldsymbol{b}_j) = 1$, and an unshaded ellipse indicates that $f(\boldsymbol{b}_j) = 0$.

The stacking diagram of a PBF can be efficiently used to determine the minimal sum-of-product form of the PBF [95]. It can also be used to determine whether or not a Boolean function is positive. Most importantly, the stacking diagram representation of Boolean filters is a powerful technique for the design and analysis of stack filters, and it is expected to play a role very similar to that enjoyed by the frequency domain representation of linear filters [26].

2.2 Definition of a Stack Filter

A stack filter architecture has been defined by Wendt et al. in [86], as a filtering configuration in which an L-level sequence appearing in the input window is first decomposed into a set of binary signals by thresholding, then a filtering operation is performed via a positive Boolean function (PBF) f on each of these threshold signals, and finally, all the binary results are added together to yield the filter's multilevel output $\mathcal{S}_f$.

Let $\mathbf{x}(n) = \{\mathrm{x}_0, \mathrm{x}_1, \ldots, \mathrm{x}_{M-1}\}$ denote an L-level sequence appearing at the input of a stack filter of window size M, at the time-instant n.[1] It is convenient to assume that L is a power of 2, i.e., $L = 2^K$. However, when this condition is not fulfilled, it is always possible to assume that $\mathbf{x}(n)$ is an L_1-level sequence where $L_1 = 2^{K_1}$ with K_1 being the smallest integer greater than or equal to $\log_2 L$, i.e., $K_1 = \lceil \log_2 L \rceil$. Let $\delta(\mathrm{x}_i - \ell)$ $(\ell = 1, 2, \ldots, L-1)$ denote a threshold operator which decomposes the sample x_i $(i = 0, 1, \ldots, M-1)$ in the multilevel sequence $\mathbf{x}(n)$ into $L-1$ binary numbers, $\{\delta(\mathrm{x}_i - 1), \delta(\mathrm{x}_i - 2), \ldots, \delta(\mathrm{x}_i - \ell), \ldots, \delta(\mathrm{x}_i - L + 1)\}$. Specifically, $\delta(\mathrm{x}_i - \ell)$ assumes a value of 1 if and only if x_i is greater than or equal to ℓ, i.e.,

$$\delta(\mathrm{x}_i - \ell) \triangleq \begin{cases} 1 & \text{if} \quad \mathrm{x}_i \geq \ell \\ 0 & \text{if} \quad \mathrm{x}_i < \ell\,, \end{cases} \qquad \ell = 1, 2, \ldots, L-1\,. \tag{2.3}$$

[1]In this notation, x_j's are really functions of n, i.e., $\mathrm{x}_j \triangleq \mathrm{x}_j(n)$, and $\mathrm{x}_j \in \{0, 1, \ldots, L-1\}$ for all j's. However, for notational simplicity, dependence on n has not been explicitly shown.

A stack filter $\mathcal{S}_f$ is characterized by a PBF f, and it is defined by the following input-output relation

$$\mathcal{S}_f(\mathbf{x}(n)) = \sum_{\ell=1}^{2^{\mathrm{K}}-1} f(\,\delta(\mathrm{x}_0 - \ell), \delta(\mathrm{x}_1 - \ell) \ldots, \delta(\mathrm{x}_{\mathrm{M}-1} - \ell)\,)\ . \tag{2.4}$$

As an example, Figure 2.2(b) shows a stack filter which performs the operation of median filtering illustrated in Figure 2.2(a). The stack filter of Figure 2.2(b) performs the operation of median filtering by applying the binary function

$$f(x_0, x_1, x_2) = x_0x_1 + x_1x_2 + x_2x_0\ , \tag{2.5}$$

on each threshold level of the input sequence.[2] If the function f given by (2.5) is changed to $f'(x_0, x_1, x_2) = x_0 + x_1 + x_2$, than the multilevel operation becomes one of selecting the largest sample in $\mathbf{x}(n)$. Similarly, when f is changed to $f''(x_0, x_1, x_2) = x_0x_1x_2$, the multilevel operation is changed to that of selecting the minimum element of $\mathbf{x}(n)$.

It can be easily observed that the binary vectors obtained by applying the thresholding operator δ given by (2.3) to the elements of $\mathbf{x}(n)$ for each level-value $\ell = 1, 2, \ldots, \mathrm{L}-1$, i.e.,

$$\boldsymbol{x}_\ell(n) \triangleq (\,\delta(\mathrm{x}_0 - \ell),\ \delta(\mathrm{x}_1 - \ell),\ \ldots,\ \delta(\mathrm{x}_{\mathrm{M}-1} - \ell)\,)\ ,\quad \ell = 1, 2, \ldots, \mathrm{L}-1\ , \tag{2.6}$$

usually referred to as threshold vectors (sequences), are pairwise comparable by the stacking relation. Specifically, we have

$$\boldsymbol{x}_1(n) \geq \boldsymbol{x}_2(n) \geq \ldots \geq \boldsymbol{x}_{\mathrm{L}-1}(n)\ . \tag{2.7}$$

Now, in view of this property of the threshold vectors, and since the Boolean filter f is a PBF (i.e., f has the stacking property), we also have

$$f(\boldsymbol{x}_1(n)) \geq f(\boldsymbol{x}_2(n)) \geq \ldots \geq f(\boldsymbol{x}_{\mathrm{L}-1}(n))\ . \tag{2.8}$$

[2]Recall that the function f given by (2.5) is a positive Boolean function, which refers to the fact that the expression $x_0x_1 + x_1x_2 + x_2x_0$ contains no complements of the input variables (i.e., $\overline{x}_0$, $\overline{x}_1$, or $\overline{x}_2$).

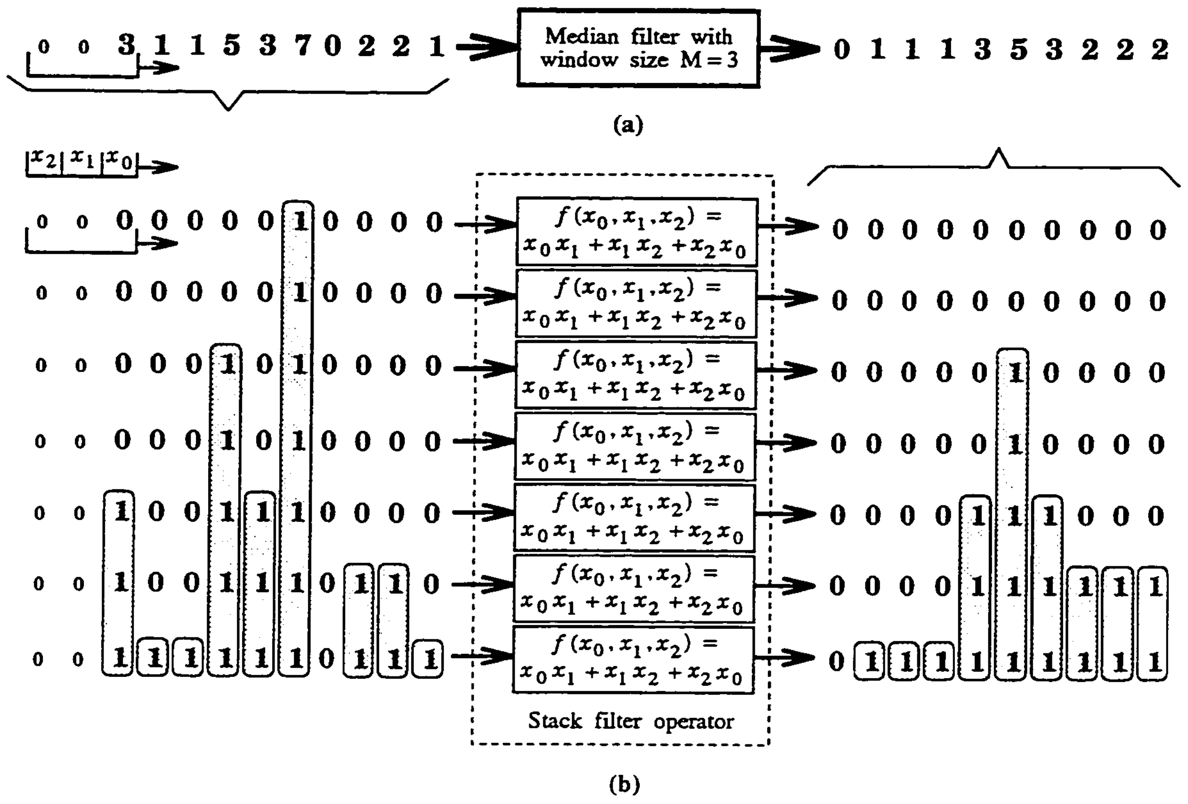


Figure 2.2: An example of a stack filter performing median filtering. (a) A median filter of size M=3 operating on a multilevel signal with L=8. (b) A stack filter that performs the PBF operation corresponding to the median filtering shown in (a).

Thus, when employing a filtering configuration in which the multilevel sequence is decomposed by thresholding and the threshold vectors are placed on top of each other (with $\boldsymbol{x}_1(n)$ at the bottom and $\boldsymbol{x}_{\ell+1}(n)$ above $\boldsymbol{x}_\ell(n)$ for all $\ell = 1, 2, \ldots, \mathrm{L}-2$) to carry out the operation defined by (2.4), the binary outputs of the operator f have a structure of a stack of 0's piled on the top of a stack of 1's.[3] It is due to this property that the operator $\mathcal{S}_f$ defined by (2.4) has been called a stack filter.

Another important property of the filtering operation $\mathcal{S}_f(\mathbf{x}(n))$ defined by (2.4) is that the output of $\mathcal{S}_f$ is always selected from the samples appearing in the input window

[3] This structure can be observed in the example illustrated in Figure 2.2(b).

$\mathbf{x}(n)$. This can be easily proved by observing that in a stack filter configuration, the pile of binary outputs at the instant n could switch from 1 to 0 only for $\ell \in \{\mathrm{x}_0, \mathrm{x}_1, \ldots, \mathrm{x}_{M-1}\}$. As a result, it can be shown that the instantaneous behavior of a stack filter can be interpreted as an operation of selecting the rth largest element of $\mathbf{x}(n)$, with the value of r itself depending on the relative magnitudes of the samples appearing in the filter window $\mathbf{x}(n)$. Thus, we can state that a stack filter performs an operation of signal-dependent rank order filtering.

2.3 Multilevel Representation of Stack Filters

The technique of defining a filter employing unit-weighted signals obtained by the threshold decomposition of a multilevel input signal, allows one to perform the analysis and the hardware design of rank order based filters at the binary level. This could be extremely beneficial when a simple mapping between the set of binary stack filters and that of multilevel filters is available. For instance, for order statistic (rank order) filters, a clear understanding of both their binary and multilevel representations exists. As a result, these filters enjoy increasingly efficient implementations in both hardware and software [67], [63].

Although somewhat more complicated than in the case of order statistic filters, a multilevel representation of the filtering operation of a general binary stack filter has been also established [22], [51]. The next theorem, which is due to Fitch [22], summarizes this multilevel representation of general stack filters.

Theorem 2.2 There is a one to one correspondence between rank order based filters which have a max-min form, and filters defined using the threshold decomposition implementation with a logical sum-of-products as the binary operation. The mapping between the two descriptions is accomplished by interchanging the maximum operator with the logical OR, and the minimum operator with the logical AND.

An illustration of the correspondence between the multilevel and the binary-level implementations of a stack filter as stated by Theorem 2.2 is shown in Figure 2.3 [81]. The stack filter $\mathcal{S}_f$ is assumed to be characterized by a PBF f of size M whose minimum sum-of-product form consists of T terms. The jth term of the minimum sum-of-product form of f, $\mathrm{P}_j = x_{j_1} x_{j_2} \ldots x_{j_J}$ (where $j = 0, 1, \ldots, \mathrm{T}$, and $j_1, j_2, \ldots, j_J$ are distinct numbers in the set $\{0, 1, \ldots, \mathrm{M}-1\}$), determines a set of multilevel variables denoted by $\{\boldsymbol{x}_j\}$. At the multilevel domain, the output of the stack filter $\mathcal{S}_f$ can be obtained as

$$\mathcal{S}_f(\mathbf{x}(n)) = \max \{ \min \{\boldsymbol{x}_0\}, \min \{\boldsymbol{x}_1\}, \ldots, \min \{\boldsymbol{x}_\mathrm{T}\} \} . \tag{2.9}$$

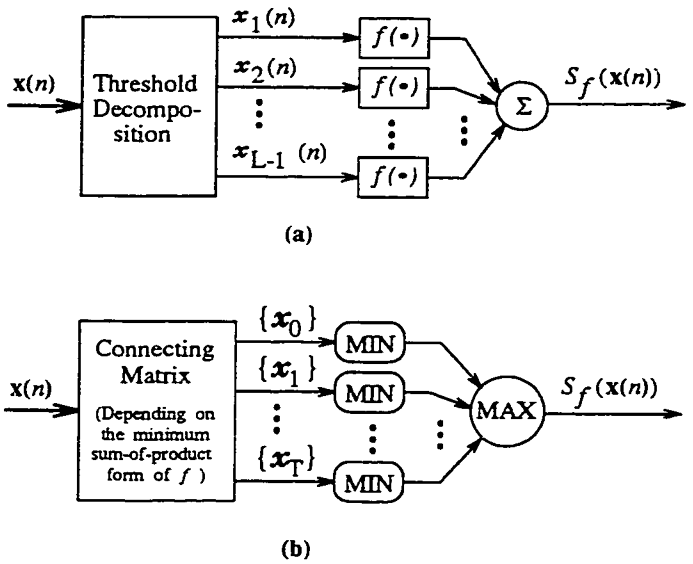


Figure 2.3: An illustration of the correspondence between (a) the binary-level and (b) the multilevel implementations of a stack filter as stated by Theorem 2.2.

As an example, we now determine the multilevel representation as defined by Theorem 2.2, corresponding to the operation of a stack filter characterized by the PBF

$$f(x_0, x_1, x_2, x_3) = x_1 + x_0 x_2 + x_2 x_3 \ , \tag{2.10}$$

whose stacking diagram was illustrated in Figure 2.1. Now, by using Theorem 2.2, the multilevel implementation of the stack filter characterized by (2.10) is given by

$$\mathcal{S}_f(\mathbf{x}(n)) = \max \{ \ \mathrm{x}_1, \ \min\{\mathrm{x}_0, \mathrm{x}_2\}, \ \min\{\mathrm{x}_2, \mathrm{x}_3\} \ \} \ . \tag{2.11}$$

A similar max-min representation can also be derived for the operation of a stack filter characterized by the PBF given by (2.5). However, it should be observed that the max-min representation of the filter characterized by (2.5) is unnecessarily much more complicated than the multilevel representation of this filter as an operation of median filtering. Thus, although the result of Theorem 2.2 gives a general representation of an operation of stack filtering at the multilevel domain of the signal, this representation is not always very useful for the actual multilevel-implementation of stack filters.

2.4 Generalized Stack Filters

In Section 2.2, a stack filter $\mathcal{S}_f$ was characterized by a single PBF f, which is applied to the threshold sequence appearing at each level of the filter configuration (see (2.4)). In [45], the class of stack filters has been generalized by employing different Boolean functions at different levels of the stack filter architecture, and allowing each Boolean function to operate on the threshold sequences appearing at several adjacent levels. The new class of filters has been designated as generalized stack filters [18], [45]. Let us denote the set of Boolean functions that characterizes a generalized stack filter by $\{f\} \triangleq \{f_1, f_2, \ldots, f_{\mathrm{L}-1}\}$. As shown in [18] and [97], in order to preserve the characteristic structure of the binary outputs of a stack filter architecture (i.e., the structure of a stack of 0's piled on top of a stack of 1's), the functions $f_1, f_2, \ldots$, and $f_{\mathrm{L}-1}$ should satisfy the implication relation (see Definition 2.3) in an appropriate order. Specifically, we should have

$$f_1 \supseteq f_2 \supseteq \cdots \supseteq f_\ell \supseteq \cdots \supseteq f_{\mathrm{L}-1} \ . \tag{2.12}$$

Thus, it can be observed that the restriction imposing the positivity of the Boolean function associated with a stack filter has been replaced in the case of generalized stack filtering by the condition (2.12). As a result, in a generalized stack filter, the individual Boolean functions $f_1, f_2, \ldots,$ and f_{L-1} are not necessarily positive, as long as the ordering relation (2.12) is satisfied [18].

A generalized stack filter $\mathcal{GS}_{\{f\}}$, characterized by a set of Boolean functions $\{f\}$ that satisfy the ordering relation given by (2.12), is defined by the following input-output relation

$$\mathcal{GS}_{\{f\}}(\mathbf{x}(n)) = \sum_{\ell=1}^{L-1} f_\ell\,(\,\{\delta(\mathbf{x}(n)-\ell+I)\}\,|\ldots\{\delta(\mathbf{x}(n)-\ell)\}\,|\ldots\{\delta(\mathbf{x}(n)-\ell-I)\}\,)\,, \quad (2.13)$$

where $\{\delta(\mathbf{x}(n) - \ell + k)\}$ designates the threshold sequence at the level $(\ell - k)$, i.e.,

$$\{\delta(\mathbf{x}(n) - \ell + k)\} \triangleq \{\,\delta(\mathbf{x}_0 - \ell + k),\, \delta(\mathbf{x}_1 - \ell + k),\, \ldots,\, \delta(\mathbf{x}_{M-1} - \ell + k)\,\}\,, \quad (2.14)$$

and the notation "|" implies the concatenation of the threshold sequences at the indicated levels. As seen from (2.13), the binary input vector to each Boolean operator f_ℓ is obtained by concatenating the threshold sequence appearing at the level ℓ, with the threshold sequences appearing at $2I$ additional levels, symetrically adjacent to ℓ. Thus, in generalized stack filtering, in addition to thresholding at the regular levels $\ell = 1, 2, L-1$, the operation of thresholding is also carried out at the levels $\ell = -I+1, -I+2, \ldots, 0$, and $\ell = (L-1)+I, (L-1)+(I-1), \ldots, (L-1)+1$. These additional threshold levels are required as boundary conditions. As an example, Figure 2.4 shows a generalized stack filter of size $M = 4$ with $I = 1$, operating on a multilevel signal with $L = 4$. In this case, the size of the Boolean functions is $(2I+1) \cdot M = 12$.

As pointed out in [22], at the multilevel domain of the signal, the operation of generalized stack filtering can be interpreted in terms of a max-min representation similar to that given by Theorem 2.2.

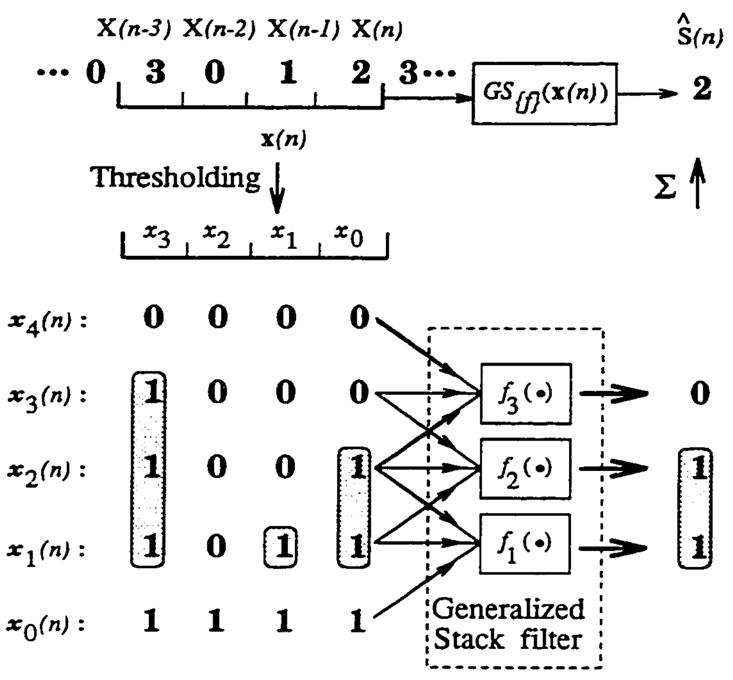


Figure 2.4: An example of a generalized stack filter with M = 4 and $I = 1$ operating on a multilevel signal with L = 4.

2.5 Minimum MAE Design of Stack Filters

The problem of signal estimation using optimal stack filters has been formulated in [17]. Figure 2.5(a) illustrates the procedure of signal estimation using an optimal stack filter S_{f^*}, where X(n) is a noise-corrupted version of S(n), and at each instant n, the output of the stack filter, Ŝ(n), is an optimal estimate of the desired signal S(n). The procedure for designing the optimal filter S_{f^*} is illustrated in Figure 2.5(b). The desired signal S(n) and its noise-corrupted version X(n) are used to determine a PBF f^* which minimizes the MAE between the filter output, $S_f(\mathbf{x}(n))$, and the desired signal S(n). The filter size is M, and $\mathbf{x}(n)$ denotes the sequence appearing in the filter window. The MMAE design of stack filters is based on a binary-level formulation of the mean absolute error between the desired and estimated signals. This formulation is described next.

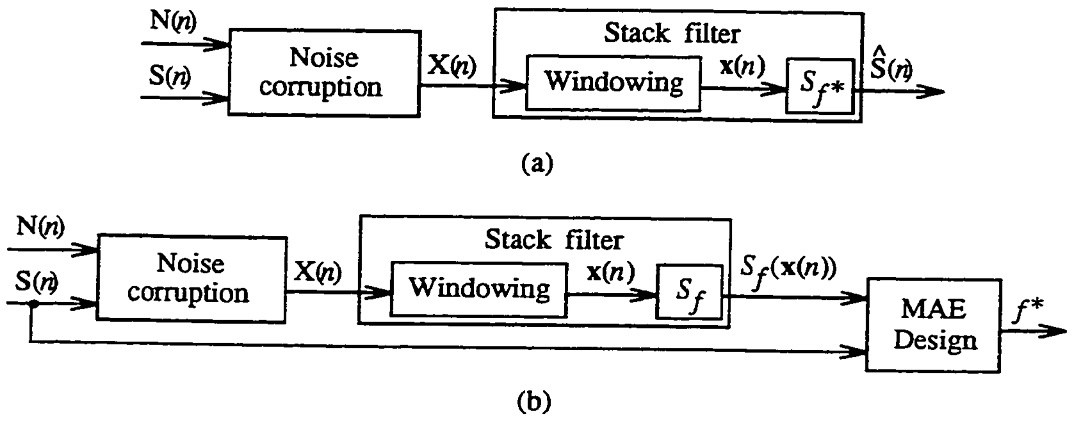


Figure 2.5: An illustration of the procedure of signal estimation using optimal stack filters. (a) Estimation of a desired signal S(n) using an optimal stack filter $\mathcal{S}_{f^*}$. (b) Design of the MAE optimal PBF f^*.

2.5.1 Derivation of a Binary-Level Expression for the MAE

As stated in [16], the reason for the choice of the MAE criterion in the design of stack filters has been primarily its mathematical tractability. Specifically, this criterion has allowed a decomposition of the estimation error of the filter into a sum of decision errors incurred by the Boolean operators at each level of the stack filter architecture [15], [17]. This decomposition of the filter design problem is summarized below. Let MAE($\mathcal{S}_f$) denote the cost of using a stack filter $\mathcal{S}_f$ characterized by the PBF f, i.e.,

$$\mathrm{MAE}(\mathcal{S}_f) = E\left[\left|\ \mathrm{S}(n) - \mathcal{S}_f(\mathbf{x}(n))\ \right|\right]. \tag{2.15}$$

Denoting the threshold sequences in the threshold decomposition of $\mathbf{x}(n)$ by $\boldsymbol{x}_\ell(n)$, $\ell = 1, 2, \ldots, \mathrm{L}-1$, as given by (2.6), and the binary signals in the threshold decomposition of S(n) by $\mathrm{s}_\ell(n)$, i.e., $\mathrm{s}_\ell(n) \triangleq \delta(\mathrm{S}(n) - \ell)$, we have

$$\mathrm{MAE}(\mathcal{S}_f) = E\left[\left|\ \sum_{\ell=1}^{\mathrm{L}-1} \mathrm{s}_\ell(n) - \sum_{\ell=1}^{\mathrm{L}-1} f(\boldsymbol{x}_\ell(n))\ \right|\right]. \tag{2.16}$$

Due to the stacking property of the binary operator f, all the nonzero terms of the sum in (2.16) have the same sign. Figure 2.6 illustrates an example of the errors incurred in signal estimation using a stack filter for the two possible cases, $\hat{\mathrm{S}}(n) \leq \mathrm{S}(n)$ and $\hat{\mathrm{S}}(n) > \mathrm{S}(n)$. Thus, the operations of summation and taking the absolute value may be interchanged as

$$\begin{aligned} \mathrm{MAE}(\mathcal{S}_f) &= E\left[\left|\sum_{\ell=1}^{\mathrm{L}-1}(\mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n)))\right|\right] \\ &= E\left[\sum_{\ell=1}^{\mathrm{L}-1}|\mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))|\right] \\ &= \sum_{\ell=1}^{\mathrm{L}-1} E\left[|\mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))|\right] . \end{aligned} \tag{2.17}$$

The ℓth term of the summation in (2.17) is the mean absolute error $E[\,|\,e_\ell(n)\,|\,]$ incurred at level ℓ, when the Boolean operator f is used at that level. Therefore, (2.17) can be rewritten as

$$\mathrm{MAE}(\mathcal{S}_f) = \sum_{\ell=1}^{\mathrm{L}-1} E[\,|\,e_\ell(n)\,|\,] \ , \tag{2.18}$$

where $e_\ell(n) = \mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))$.

The problem of designing a stack filter which is optimal for signal estimation under the MAE criterion is to minimize the function given by (2.18), under the constraint that f satisfies the stacking property (i.e., it is a PBF). In order to solve the optimization problem associated with the design of stack filters under the MAE criterion, an explicit expression for $E[\,|\,e_\ell(n)\,|\,]$ of (2.18), as a function of the outputs of the Boolean filter f should be derived. The derivation proposed by Coyle and Lin [17] uses the concept of a randomizing Boolean function, and replaces the original MAE cost function with a weighted sum of the decision probabilities at the binary levels. In this section, a simple derivation of an expression for $E[\,|\,e_\ell(n)\,|\,]$, using conventional Boolean functions, is proposed [69], [73].

A Boolean filter f of window size M is completely determined by the vector

$$\mathbf{F} \triangleq (\, f(\boldsymbol{b}_1),\ f(\boldsymbol{b}_2),\ \ldots,\ f(\boldsymbol{b}_j),\ \ldots,\ f(\boldsymbol{b}_{2^\mathrm{M}})\,) \ , \tag{2.19}$$

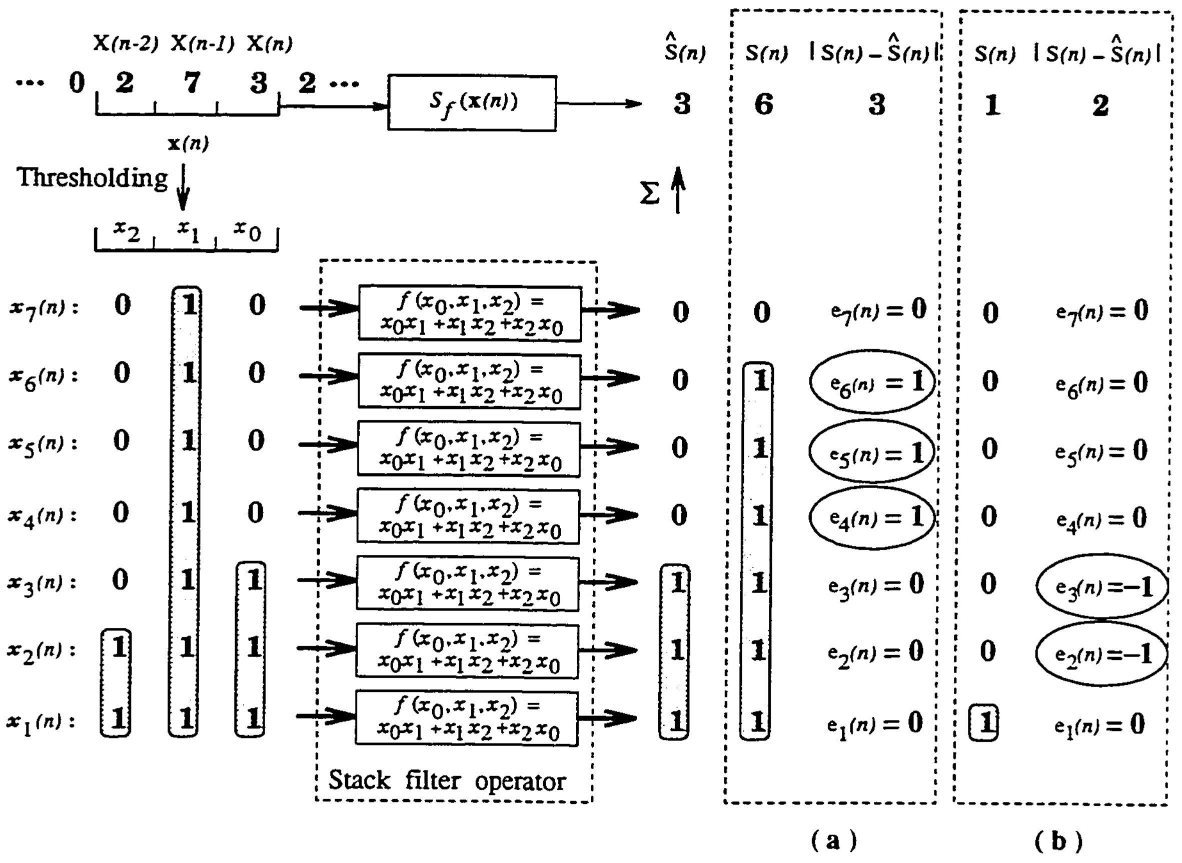


Figure 2.6: An illustration of the errors incurred between the desired and estimated samples in a stack filter architecture, for the two possible cases ($\hat{S}(n) \leq S(n)$ and $\hat{S}(n) > S(n)$). (a) When the desired sample is greater than the estimated one, the binary errors at the various threshold levels can assume values of only 0 or 1. (b) When the desired sample is less than the estimated one, they can be only 0 or -1.

where $\left\{ \boldsymbol{b}_j | j = 1, 2, \ldots, 2^{\mathrm{M}} \right\}$ is the set of all Boolean vectors of size M. The jth entry in (2.19) represents the output of the filter, when the binary vector in the filter's input window is $\boldsymbol{b}_j$. Let us consider the combined experiment of observing different realizations $\mathrm{s}_\ell(n) \in \{0, 1\}$, of the desired threshold process at level ℓ of a stack filter architecture at the time instant n, and the corresponding realizations $\boldsymbol{x}_\ell(n) \in \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{2^{\mathrm{M}}}\}$, of the binary input to the Boolean filter f at level ℓ. The sample space of this experiment is given by

$$\{ (0, \boldsymbol{b}_1), (0, \boldsymbol{b}_2), \ldots, (0, \boldsymbol{b}_{2^M}), (1, \boldsymbol{b}_1), (1, \boldsymbol{b}_2), \ldots, (1, \boldsymbol{b}_{2^M}) \}, \tag{2.20}$$

where a pair $(0, \boldsymbol{b}_j)$ denotes the outcome $(\mathrm{s}_\ell(n) = 0, \boldsymbol{x}_\ell(n) = \boldsymbol{b}_j)$, while a pair $(1, \boldsymbol{b}_j)$ corresponds to $(\mathrm{s}_\ell(n) = 1, \boldsymbol{x}_\ell(n) = \boldsymbol{b}_j)$. The absolute value of the decision error at level ℓ, $|e_\ell(n)|$, can be regarded as a random variable which assigns a numerical quantity given as $|\mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))|$, to each element of the sample space specified in (2.20). The expected value of $|e_\ell(n)|$ is given by[4]

$$\begin{aligned} E\,[\,|e_\ell(n)|\,] &= \sum_{j=1}^{2^{\mathrm{M}}} (\,|0 - f(\boldsymbol{b}_j)| \cdot P\,[\mathrm{s}_\ell(n) = 0, \boldsymbol{x}_\ell(n) = \boldsymbol{b}_j] + \\ &\qquad + |1 - f(\boldsymbol{b}_j)| \cdot P\,[\mathrm{s}_\ell(n) = 1, \boldsymbol{x}_\ell(n) = \boldsymbol{b}_j]) \\ &= \sum_{j=1}^{2^{\mathrm{M}}} [(P(0, \boldsymbol{b}_j|\ell) - P(1, \boldsymbol{b}_j|\ell)) \cdot f(\boldsymbol{b}_j) + P(1, \boldsymbol{b}_j|\ell)], \end{aligned} \tag{2.21}$$

where the notations $P(0, \boldsymbol{b}_j|\ell)$ and $P(1, \boldsymbol{b}_j|\ell)$ imply, respectively, $P\,[\mathrm{s}_\ell(n) = 0, \boldsymbol{x}_\ell(n) = \boldsymbol{b}_j]$ and $P\,[\mathrm{s}_\ell(n) = 1, \boldsymbol{x}_\ell(n) = \boldsymbol{b}_j]$. Therefore, the expected value of the decision error at each level ℓ of a stack filter architecture, $E\,[\,|e_\ell(n)|\,]$, is a linear function of the outputs $f(\boldsymbol{b}_j)$ $(j = 1, 2, \ldots, 2^{\mathrm{M}})$ of the filter f. Using (2.21), the cost function $\mathrm{MAE}(\mathcal{S}_f)$ of (2.18) becomes

$$\mathrm{MAE}(\mathcal{S}_f) = \sum_{j=1}^{2^{\mathrm{M}}} [\alpha_j \cdot f(\boldsymbol{b}_j) + \beta_j], \tag{2.22}$$

[4] The expected value of a discrete random variable X defined on a sample space with K possible outcomes, is given by

$$E\,[\mathrm{X}] = \sum_{j=1}^{\mathrm{K}} \mathrm{x}_j \cdot P(\mathrm{x}_j),$$

where x_j denotes a particular value of X, having the probability of occurrence $P(\mathrm{x}_j)$ [57].

where
$$\alpha_j = \sum_{\ell=1}^{L-1}(P_\ell(0,\boldsymbol{b}_j) - P_\ell(1,\boldsymbol{b}_j)) = P(0,\boldsymbol{b}_j) - P(1,\boldsymbol{b}_j)\,, \tag{2.23}$$
and
$$\beta_j = \sum_{\ell=1}^{L-1} P_\ell(1,\boldsymbol{b}_j) = P(1,\boldsymbol{b}_j)\,. \tag{2.24}$$
Based on the error formulation given by (2.22), a linear program (LP) which finds a minimum mean absolute error stack filter is now derived [17].

2.5.2 Derivation of an LP to Find an MMAE Stack Filter

A stack filter which is optimal for signal estimation in the MAE sense is characterized by a positive Boolean function f^* which minimizes the cost function $\mathrm{MAE}(\mathcal{S}_f)$ given by (2.22), i.e.,
$$\mathrm{MAE}(\mathcal{S}_{f^*}) = \min_{f\in\{PBF_\mathrm{M}\}}\mathrm{MAE}(\mathcal{S}_f)\,, \tag{2.25}$$
where $\{PBF_\mathrm{M}\}$ denotes the set of all PBFs of size M. The solution to this optimization problem can be determined by solving an integer linear program (ILP) which is stated below.

ILP 1 : Minimize $\mathrm{MAE}(\mathcal{S}_f)$, subject to the following constraints in the stacking diagram of a Boolean filter f of size M: *a)* for each vertex $\boldsymbol{b}_j$, $f(\boldsymbol{b}_j) \in \{0,1\}$, and *b)* for each edge $(\boldsymbol{p}1\boldsymbol{q}, \boldsymbol{p}0\boldsymbol{q})$, we have $f(\boldsymbol{p}1\boldsymbol{q}) \geq f(\boldsymbol{p}0\boldsymbol{q})$.

As shown in [17], it can be observed that the constraint equations corresponding to the edges of the stacking diagram of f can be written into a matrix form as
$$\mathbf{A}\cdot\mathbf{F}^\mathrm{T} \leq \mathbf{0}\,, \tag{2.26}$$
where $\mathbf{F}$ is defined as given by (2.19), and the entries of $\mathbf{A}$ are 0, +1 or -1 such that each row of $\mathbf{A}$ has exactly two nonzero entries of opposite signs. Thus, the matrix $\mathbf{A}$ is totally unimodular [27], [56], and as a result, a solution of the *ILP 1* can be obtained by solving the following linear program.

LP 1 : Minimize $\mathrm{MAE}(S_f)$, subject to the following constraints in the Hasse diagram of the set $\left\{\mathrm{b}_j | j = 1, 2, \ldots, 2^{\mathrm{M}}\right\}$: *a)* for each vertex $\boldsymbol{b}_j$, $f(\boldsymbol{b}_j) \in [0,1]$, and *b)* for each edge $(\boldsymbol{p}1\boldsymbol{q}, \boldsymbol{p}0\boldsymbol{q})$, we have $f(\boldsymbol{p}1\boldsymbol{q}) \geq f(\boldsymbol{p}0\boldsymbol{q})$.

Specifically, any solution of *LP 1* is an integer (i.e., $f(\boldsymbol{b}_j) \in \{0,1\}$ for all j's), and consequently, it is a solution of *ILP 1* as well.

As shown in [17] and [27], in spite of the theoretical importance of this result, the number of variables and constraints of *LP 1* increases faster than exponentially with the filter size.[5] Consequently, the exact solution of the LP is generally sought only in the case of filters with a relatively small window size, usually for $\mathrm{M} \leq 12$ (e.g., 3×3 filters). For filters with a larger window size, a number of heuristic (suboptimal) techniques to find an approximately optimal solution of *LP 1* have been used [20], [46], [47], [92].

Although the MAE criterion of designing stack filters has been extensively used for image processing applications, some attempts to obtain more general results, based on higher order errors, have also been reported [89], [90]. Specifically, in [89] and [90], the higher-order error-based design is confined to the case in which the input signal is assumed to be a constant signal embedded in white noise. In Chapter 3, we investigate the possibility to develop a sound mathematical framework for the general problem of designing stack filters employing the L_p norm of the error between the desired and estimated signals.

2.6 Parallel and Bit-Serial Architectures for Stack Filtering

Two types of architectures, parallel and bit-serial, have been traditionally used for the implementation of stack filters. The parallel configuration for stack filtering has been introduced by Wendt et al. in [86], and it is illustrated in Figure 2.7. This configuration

[5] For a filter of size M, the LP has 2^{M} variables and $\mathrm{M} \cdot 2^{\mathrm{M}-1} + 1$ constraints. Thus, for example, for $\mathrm{M} = 13$, there are 8,192 variables and 53,249 constraints [81].

uses a bank of (L − 1) threshold decomposition units and (L − 1) PBF's [5], [31], [86]. However, as pointed out by Chen in [13], the parallel implementation of PBF's at all threshold levels is not an efficient way to use the hardware, since the time-area complexity in this case is $\mathcal{O}(L)$. In [13], Chen has developed a bit-serial binary-tree search (BTS) architecture for stack filtering, which uses only one PBF circuit, and whose time-area complexity is $\mathcal{O}(\lceil \log_2 L \rceil)$.

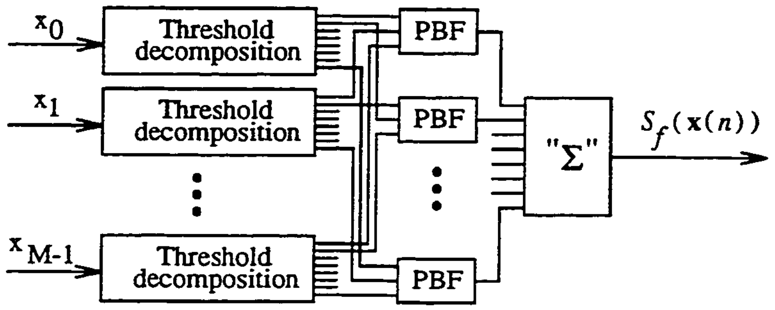


Figure 2.7: Block diagram of a parallel architecture for stack filtering. Note that the operation of "summation" of the binary outputs is implemented as a binary-tree search for the level where the transition from 1 to 0 takes place.

The bit-serial BTS configuration for the implementation of stack filters is illustrated in Figure 2.8, where d_{iJ} denotes $\delta(x_i - \ell_J)$, $(i = 0, 1, \ldots, M-1)$. The architecture of Figure 2.8 realizes the algorithm

$$y = \sum_{J=1}^{K} f(\delta(x_0 - \ell_J), \delta(x_1 - \ell_J), \ldots, \delta(x_{M-1} - \ell_J)) \cdot 2^{K-J}, \tag{2.27}$$

where the threshold levels ℓ_J's are given by

$$\ell_1 = 2^{K-1}, \tag{2.28}$$

$$\ell_J = [\sum_{j=1}^{J-1} f(\delta(x_0 - \ell_j), \delta(x_1 - \ell_j), \ldots,$$
$$\ldots, \delta(x_{M-1} - \ell_j)) \cdot 2^{K-j}] + 2^{K-J}, \quad \text{for} \quad J \in \{2, 3, \ldots, K\}. \tag{2.29}$$

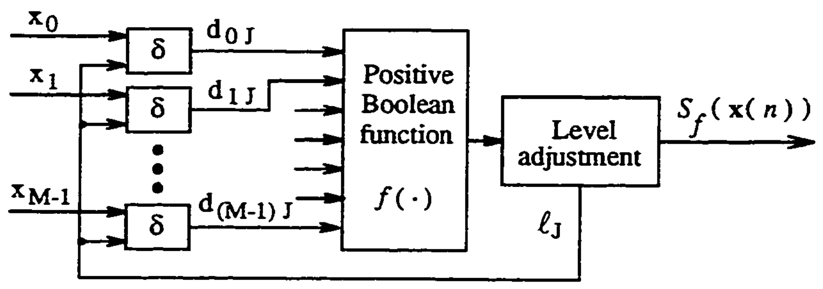


Figure 2.8: Block diagram of a bit-serial BTS architecture for stack filtering.

It has been demonstrated in [13], that the BTS algorithm given by (2.27)-(2.29) is functionally equivalent to the operation of stack filtering defined by (2.4), i.e.,

$$y = S_f(\mathbf{x}(n)) = \sum_{J=1}^{K} f(\delta(x_0 - \ell_J), \ldots, \delta(x_{M-1} - \ell_J)) \cdot 2^{K-J} . \tag{2.30}$$

The BTS algorithm realizes the operation of stack filtering through a recursive binary-level processing at K different levels. With this approach, the number of threshold or PBF operations is reduced to K (as compared to L − 1, which is the number of these operations required in the parallel architecture of [86]). In the first step of the BTS method, all input samples in the window, x_i, $(i = 0, 1, \ldots, M-1)$, are compared to ℓ_1 $(= 2^{K-1})$. By applying the PBF to the outputs of the threshold operators, the most significant bit (MSB) of the output is obtained. Depending on this first output bit, the threshold level ℓ_2 in the second iteration assumes a value of 2^{K-2} or $2^{K-1} + 2^{K-2}$. At level ℓ_2, the PBF f is evaluated to obtain the next MSB of the output. Then, the already known values of the function f at levels ℓ_1 and ℓ_2 are used to evaluate the threshold level ℓ_3. This recursive binary-level processing is repeated K times, and at the K*th* iteration, the least significant bit of the output is obtained. Thus, note that in the actual implementation of the BTS algorithm, it is not necessary to perform the operations of (2.27) independently, as in each successive determination of the threshold level, the evaluation of the function f provides one bit of the output.

In the bit-serial BTS approach for stack filtering, the level adjustment is carried out based on the result of the operation of positive Boolean filtering, and by using increments which are powers of 2. Consequently, the bit-serial BTS algorithm of [13] does not take advantage of the additional ordering information associated with the binary threshold sequences. At each iteration, the threshold sequence can be interpreted as a partitioning of the window samples into elements which are greater and elements which are smaller than the current threshold level. By using this additional information available, it is expected that a significant improvement in terms of the computational time can be achieved. This possibility is investigated in Chapter 4.

2.7 Input Compression-Based Architectures

The time-area complexities of the conventional architectures for stack filtering ($\mathcal{O}(\mathrm{L})$ for the parallel structure and $\mathcal{O}(\lceil\log_2 \mathrm{L}\rceil)$ for the bit-serial configuration) are determined by the global dynamic range ($[0, \mathrm{L}-1]$) of the multilevel input signal. However, more efficient architectures can be developed by taking advantage of the fact that the local dynamic range spanned by the samples appearing in the input window at a time-instant n cannot be larger than the window-size M.

In [1], Adams et al. have introduced more efficient parallel and BTS configurations for stack filtering, by employing an input compression technique which maps the samples appearing in the filter-window to the integers $\{0, 1, \ldots, \mathrm{M}-1\}$. The general structure of the input compression-based architecture for stack filtering of [1], which is illustrated in Figure 2.9, has been designated as a rank order state machine (ROSM). Note that with the architecture of Figure 2.9, the operation of rank order-based (stack) filtering using the level of the compressed input values, R_0, R_1, ..., $\mathrm{R}_{\mathrm{M}-1}$, can be realized by employing either a parallel or a BTS architecture for stack filtering. The compressed values, R_0, R_1, ..., $\mathrm{R}_{\mathrm{M}-1}$, represent the ranks of the samples x_0, x_1, ..., $\mathrm{x}_{\mathrm{M}-1}$, appearing in the input window. A rank of 0 corresponds to the smallest sample within the window, while a rank

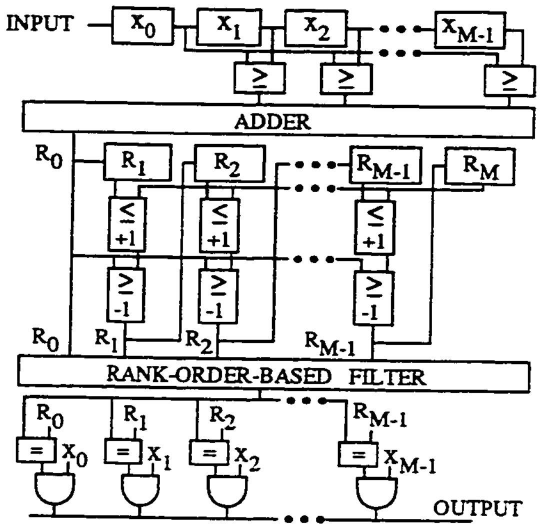


Figure 2.9: The input compression-based architecture for stack filtering.

of $M - 1$ represents the largest one. The input compression algorithm can be described as follows [1].

Algorithm 2.1

1. Initialize the ranks $R_0, R_1, \ldots, R_{M-1}$ to $0, 1, \ldots, M - 1$, respectively. Note that, assuming a causal signal, the initial input window contains all zero values.
2. At each time instant n, let $R_j = R_{j-1}$, $j = 1, 2, \ldots, M - 1$.
3. For the samples appearing in the input window at time instant n, $\mathbf{x}(n) = \{x_0, x_1, \ldots, x_{M-1}\}$, if x_0 is greater than or equal to c elements of $\mathbf{x}(n)$, then set $R_0 = c$. Thus, R_0 is the rank of x_0 among the sample-values in the input-window.
4. The remaining rank values for the $M - 1$ input samples that were members of the previous window, are not yet the correct ranks of the elements of the current window. Adjust them as follows. If $R_j \leq R_M$, where $j \in \{1, 2, \ldots, M - 1\}$, then increase the value of R_j by one. If $R_0 \geq R_j$, then decrease the value of R_j by one. Therefore, each rank-element R_j is either incremented by one, decremented by one, or unchanged.

In those applications where $M \ll L$, the input compression-based parallel and BTS architectures for stack filtering achieve improved computational efficiencies over their parallel and BTS counterparts which do not use the input-compression technique. For instance, if the ROSM uses a parallel architecture for stack filtering, the number of threshold levels is reduced from L to M. This leads to an important reduction in the number of binary filters, at the expense of only a few additional components required for the input compression. When the input compression technique is used in conjunction with the BTS configuration for stack filtering, the number of iterations in the BTS-loop is reduced from K (in the conventional BTS approach) to $\lceil \log_2 M \rceil$.

The ROSM architecture keeps track of the relative ranks of the samples appearing in the input window of the stack filter as the window slides along the input sequence, by mapping each sample value to a different rank, even in the case when several samples assume non-distinct values. This technique is well suited for the case of 1-D stack filtering, where there is only one new sample appearing in the input-window at each time-instant. However, in the case of a 2-D stack filter of size $M = N \times N$, there are N new samples appearing in the input-window at each time-instant. As a result, in the case of 2-D stack filtering, the computational time associated with the ROSM architecture increases very fast with M. Moreover, in the input compression-based BTS approach for stack filtering, the number of iterations per output sample is always equal to $\lceil \log_2 M \rceil$, even for windows with non-distinct elements. Further, in image processing applications where the number of quantization levels of $L = 16$ or $L = 32$ is used, even a filter-size $M = 5 \times 5$ is comparable with L, and as a result, the input compression-based BTS algorithm for stack filtering does not provide any computational improvement over the one using the conventional BTS method. The possibility of developing an alternative input compression algorithm, that would successfully overcome the above-mentioned drawbacks of the ROSM, is investigated in Chapter 5.

2.8 Summary

In this chapter, a review of the stack filter theory and the relevant background material has been presented. The definition of a positive Boolean function as given in [53] has been stated, and the concept of threshold decomposition of multilevel signals has been introduced. The stacking property of Boolean functions, which is essential to the understanding of stack filters, has also been described. The formal definition of the operation of stack filtering has been presented, along with the binary-level implementation of stack filters. In this implementation, the multilevel input signal is first decomposed into a set of binary signals by threshold decomposition, then a positive Boolean function is performed on each of these threshold signals, and finally, all the binary signals are added together yielding the multilevel output. A multilevel max-min implementation of the operation of stack filtering developed by Fitch in [22] has been presented, and the generalization of stack filters to allow different PBF's to operate at different levels of a stack filter configuration has been briefly reviewed.

As shown by Coyle and Lin in [17], the stacking property of PBF's allows the development of a theory for the optimal design of a stack filter based on the MAE criterion. It has been shown that the problem of decomposing the MAE into a sum of mean absolute errors incurred by the Boolean filters at each level of the filter configuration, can be carried out by a simpler and more direct method then the one presented in [17].

The conventional techniques for the efficient implementation of the operation of stack filtering, i.e., the bit-serial BTS and the input compression-based methods, have been also described in this chapter.

Chapter 3

L_p Norm Design of Stack Filters

The problem of signal estimation using optimal stack filters has been formulated by Coyle and Lin in [17], based on employing the MAE criterion. As stated in [16], the reason for the choice of the MAE criterion in the design of stack filters has been primarily its mathematical tractability. H^wever, in some practical signal processing applications, the availability of stack filters designed using a higher order error objective function could be more desirable. For instance, in applications involving restoration of signals corrupted with impulsive noise, the use of the MSE criterion could provide improved results compared to those achieved by using the MAE design. To illustrate this possibility, let us assume that by applying two different stack filters characterized by the PBFs f_1 and f_2 to the same signal, we get, respectively, the following sets of absolute errors: $e_1 = \{0, 1, 1, 1, 0, 1\}$ and $e_2 = \{0, 0, 0, 4, 0, 0\}$. It is easy to see that since $\sum_{j=1}^{6} e_1(j) = \sum_{j=1}^{6} e_2(j)$, the two stack filters are equivalent from the standpoint of an absolute error formulation of the design problem. However, since $\sum_{j=1}^{6} e_1^2(j) = 4$ while $\sum_{j=1}^{6} e_2^2(j) = 16$, with a mean square error design approach, the two stack filters are different. The filter characterized by f_1, with a smaller mean square error, would be preferred. Minimum MSE stack filters could also provide improved results compared to the minimum MAE stack filters in applications involving nonlinear techniques for predictive coding. Specifically, in such applications, the objective in stack filter design should be to minimize the variance of the prediction error (i.e., the MSE), as is done in the design of linear filters for predictive coding [37].

Another practical advantage of using a higher order error approach to stack filter design is the possibility of obtaining a good engineering approximation to the solution of the minimax optimization problem. In image processing, the benefit of the availability of such a solution stems from the visual interpretation of the maximum absolute error between the desired and estimated images. As mentioned in [25], this error gives a measure of the "fidelity" of a processed image to the original one.

In spite of the benefits that could possibly be derived by employing an objective function given as the pth order error between the desired and estimated signals in stack filter design, so far, no attempt has been directed to develop a mathematical framework needed for this design problem. The lack of such an analytical framework could possibly stem from the notion that the problem of an L_p norm-based design would be mathematically intractable. In this chapter, the possibility to formulate the problem of designing the pth order error-optimal stack filter as a linear optimization problem is investigated [70], [76]. As a first step, the problem of designing minimum mean square error (MSE) stack filters is considered [4], [75]. It is shown that in the case of signal estimation using stack filters, the MSE between the desired and estimated signals can be expressed as a linear function of the outputs of a PBF to all possible binary input vectors. An investigation into the reason behind this interesting property of stack filters, suggests a mathematical approach for solving the general design problem of L_p norm optimal stack filters. The possibility of extending the conventional MAE design of an important subclass of stack filters (defined by linearly separable PBFs), the weighted order statistic (WOS) filters, is also investigated in this chapter [79].

3.1 Background and Notations

Let $\mathrm{X}(n)$ denote a process which is received at the input of a stack filter $\mathcal{S}_f$ characterized by a PBF f of window size M (see Figure 2.5). The input process $\mathrm{X}(n)$ is assumed to be a corrupted version of some desired process $\mathrm{S}(n)$. At each instant n, the output of the stack filter is an estimate of $\mathrm{S}(n)$. This estimate is based on the sequence $\mathbf{x}(n)$ that appears in the input window of the stack filter, and it is denoted by $\mathcal{S}_f(\mathbf{x}(n))$. As shown in Section 2.4 [17], in stack filtering, the instantaneous error at the time instant n, $e(n) \triangleq (\mathrm{S}(n) - \mathcal{S}_f(\mathbf{x}(n)))$, can be determined as

$$\mathrm{S}(n) - \mathcal{S}_f(\mathbf{x}(n)) = \sum_{\ell=1}^{\mathrm{L}-1} (\, \mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))\,)\,, \tag{3.1}$$

where L denotes the number of quantization levels of the input and output signals, and $\mathrm{s}_\ell(n)$ and $\boldsymbol{x}_\ell(n)$ designate, respectively, the binary values at the level ℓ in the threshold decompositions of $\mathrm{S}(n)$ and $\mathbf{x}(n)$. To simplify the notations, in this paper, we denote $\mathrm{S}(n)$, $\mathcal{S}_f(\mathbf{x}(n))$, $\mathrm{s}_\ell(n)$, $f(\boldsymbol{x}_\ell(n))$, and $e(n)$ by S, $\mathcal{S}_f$, s_ℓ, f_ℓ, and e, respectively. Thus, in rest of the thesis, we assume the dependency of these quantities on the independent variable n to be implicit.

As an immediate consequence of (3.1), the mean absolute error between the desired signal and the filter output is given by

$$E[\,|\,e\,|\,] = \sum_{\ell=1}^{\mathrm{L}-1} E[\,|\,\mathrm{s}_\ell - f_\ell\,|\,]\,. \tag{3.2}$$

Thus, a minimum MAE stack filter f can be found as the solution of a linear program which minimizes the MAE given by (3.2), subject to the constraint that f be a PBF [17].

3.2 MSE Optimal Stack Filtering

The objective of this chapter is to investigate the possibility of expressing a more general objective function than the MAE given by (3.2), namely, the pth order error between the desired and estimated signals, $E[\,|\,e\,|^p\,]$, as a linear combination of the binary level decision errors. As a first step, we begin this investigation by considering the problem of stack filter design using the mean square error objective function, i.e.,

$$E\left[|e|^2\right] = E\left[|\mathrm{S} - \mathcal{S}_f|^2\right] . \tag{3.3}$$

In order to derive an expression for the multilevel error e as a function of the errors at the binary levels, one may attempt, as it is done for the design of MAE stack filters [17], to simply substitute $\sum_{\ell=1}^{\mathrm{L}-1} \mathrm{s}_\ell$ for S and $\sum_{\ell=1}^{\mathrm{L}-1} f_\ell$ for $\mathcal{S}_f$ in (3.3). However, in the context of the present formulation, it can be easily observed that this substitution leads to a complicated expression for e^2. Specifically, due to the presence of the terms S^2 and $\mathcal{S}_f^2$ on the right side of (3.3), such an expression would contain cross-product terms of the forms $\mathrm{s}_i \cdot \mathrm{s}_j$, $f(\boldsymbol{x}_i) \cdot f(\boldsymbol{x}_j)$, and $\mathrm{s}_i \cdot f(\boldsymbol{x}_j)$. We should rather aim at obtaining an expression for e^2 as a function of f_ℓ $(\ell = 1, 2, \ldots, \mathrm{L}-1)$, so that the minimization process of the MSE given by (3.3) could be carried out much more easily.

As illustrated in Figure 3.1, by multiplying the outputs of the Boolean operators at each level ℓ of the stack filter $\mathcal{S}_f$ by the value of ℓ itself, we obtain a sequence of $\mathcal{S}_f$ consecutive integers. Thus, as a consequence of the stacking property of f, we have

$$\sum_{\ell=1}^{\mathrm{L}-1} \ell f_\ell = \frac{1}{2}\mathcal{S}_f(\mathcal{S}_f + 1) \tag{3.4}$$

and therefore,

$$\mathcal{S}_f^2 = 2\sum_{\ell=1}^{\mathrm{L}-1} \ell f_\ell - \mathcal{S}_f . \tag{3.5}$$

Similarly, as a result of the definition of the threshold decomposition of S, we can also have

$$\mathrm{S}^2 = 2\sum_{\ell=1}^{\mathrm{L}-1} \ell \mathrm{s}_\ell - \mathrm{S} . \tag{3.6}$$

Now, using (3.5) and (3.6), we have

$$\begin{aligned} e^2 &= 2\,\mathrm{S}\,(\mathrm{S} - \mathcal{S}_f) - \left(\mathrm{S}^2 - \mathcal{S}_f^2\right) \\ &= (2\,\mathrm{S} + 1)\,(\mathrm{S} - \mathcal{S}_f) - 2\sum_{\ell=1}^{\mathrm{L}-1} \ell\,(\mathrm{s}_\ell - f_\ell) \\ &= 2\sum_{\ell=1}^{\mathrm{L}-1} \left[\,(\mathrm{S} - \ell + 0.5)\,(\mathrm{s}_\ell - f_\ell)\,\right] , \end{aligned} \tag{3.7}$$

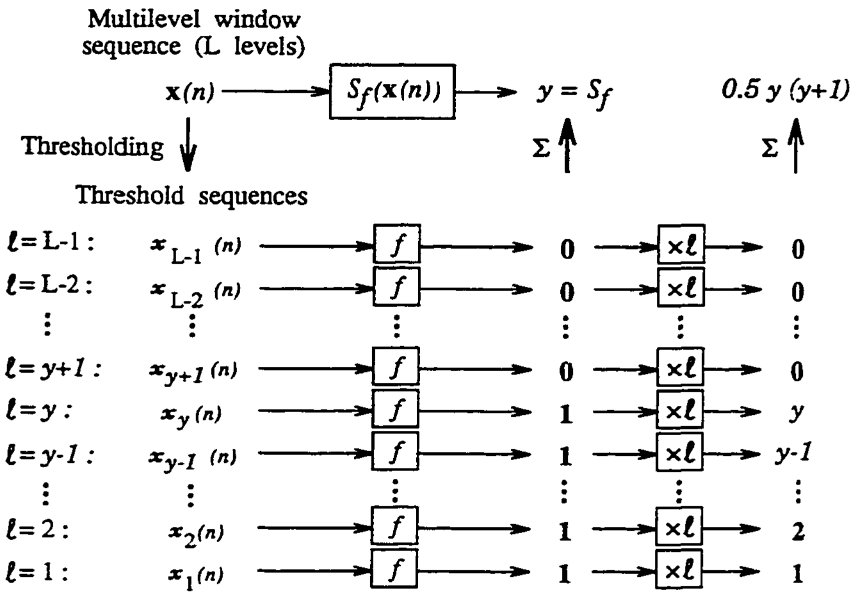


Figure 3.1: An illustration of the implementation of a square-law device at the output of a stack filter.

and, therefore, the MSE can be determined as

$$E\left[\,|e|^2\right] = 2\sum_{\ell=1}^{L-1} E\left[\,(\mathrm{S}-\ell+0.5)\,(\mathrm{s}_\ell - f_\ell)\,\right]\,. \tag{3.8}$$

Note that the term $e'_\ell(n) \triangleq (\mathrm{S}(n)-\ell+0.5)\,(\mathrm{s}_\ell(n)-f(\boldsymbol{x}_\ell(n)))$, is a weighted expression of the binary error $e_\ell(n) \triangleq \mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))$.

Now, to solve the optimization problem associated with the design of stack filters under the MSE criterion, an explicit expression for $E\left[\,|\,e'_\ell(n)\,|\,\right]$, as a function of the variables $f(\boldsymbol{b}_j) \in \{0,1\}$, where $\left\{\boldsymbol{b}_j | j = 1, 2, \ldots, 2^{\mathrm{M}}\right\}$ is the set of all Boolean vectors of size M, should be derived. Let us consider the combined experiment of observing different realizations $\mathrm{s}_\ell(n) \in \{0,1\}$ of the desired threshold process at level ℓ of a stack filter architecture, the corresponding realizations $\boldsymbol{x}_\ell(n) \in \{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{2^{\mathrm{M}}}\}$ of the binary input to the Boolean filter f at level ℓ, and the associated multilevel realizations $\mathrm{S}(n) \in \{0, 1, \ldots, \mathrm{L}-1\}$. The sample space of this experiment is given by

$$\{\,(0,\boldsymbol{b}_1),\ (0,\boldsymbol{b}_2),\ \ldots,\ (0,\boldsymbol{b}_{2^M}),\ (1,\boldsymbol{b}_1),\ (1,\boldsymbol{b}_2),\ \ldots,\ (1,\boldsymbol{b}_{2^M})\,\}$$
$$\times\ \{\,0,\ 1,\ \ldots,\ \mathrm{L}-1\,\}\,,$$

where a pair $(0,\boldsymbol{b}_j)$ denotes the outcome $(\mathrm{s}_\ell(n)=0, \boldsymbol{x}_\ell(n)=\boldsymbol{b}_j)$, a pair $(1,\boldsymbol{b}_j)$ designates the outcome $(\mathrm{s}_\ell(n)=1, \boldsymbol{x}_\ell(n)=\boldsymbol{b}_j)$, and the multiplication operator denotes the cartesian product. Thus, the expected value of $e'_\ell(n)$ is given by

$$E\,[\,e'_\ell(n)\,] \;=\; \sum_{k=0}^{\mathrm{L}-1}\{\sum_{j=1}^{2^{\mathrm{M}}}[\,(k-\ell+0.5)P_\ell(0,\boldsymbol{b}_j|k)(-f(\boldsymbol{b}_j))\,+$$
$$+\;\;(k-\ell+0.5)P_\ell(1,\boldsymbol{b}_j|k)(1-f(\boldsymbol{b}_j))\,]\,\}P(k)\,, \qquad (3.9)$$

where the notations $P_\ell(0,\boldsymbol{b}_j|k)$, $P_\ell(1,\boldsymbol{b}_j|k)$, and $P(k)$ imply, respectively, $P[\mathrm{s}_\ell(n)=0, \boldsymbol{x}_\ell(n) =\boldsymbol{b}_j|\mathrm{S}(n)=k]$, $P[\mathrm{s}_\ell(n)=1, \boldsymbol{x}_\ell(n)=\boldsymbol{b}_j|\mathrm{S}(n)=k]$, and $P[\mathrm{S}(n)=k]$. Consequently, the MSE in signal estimation using stack filters can be expressed as a linear function of the outputs $f(\boldsymbol{b}_j)$ $(j=1,2,\ldots,2^{\mathrm{M}})$ of the Boolean stack filter f, i.e.,

$$\mathrm{MSE}(\mathcal{S}_f) \;=\; \frac{1}{2}\sum_{j=1}^{2^{\mathrm{M}}}[\alpha'_j\cdot f(\boldsymbol{b}_j)+\beta'_j]\,, \qquad (3.10)$$

where

$$\alpha'_j=\sum_{k=0}^{\mathrm{L}-1}\sum_{\ell=1}^{\mathrm{L}-1}[\,(\ell-k-0.5)(P_\ell(0,\boldsymbol{b}_j|k)+P_\ell(1,\boldsymbol{b}_j|k))P(k)\,]\;, \qquad (3.11)$$

and

$$\beta'_j=\sum_{k=0}^{\mathrm{L}-1}\sum_{\ell=1}^{\mathrm{L}-1}[\,(k-\ell+0.5)P_\ell(1,\boldsymbol{b}_j|k)P(k)\,]\;. \qquad (3.12)$$

Now, based on this error formulation, it can be observed that the minimum MSE (MMSE) stack filter can be determined as a solution of a linear program with the same constraint matrix imposing the stacking property on f as the linear program that finds a MMAE stack filter [17].

In order to have a better understanding of the reason as to why, in stack filtering, it is possible to reduce the MSE to a linear combination of the outputs of the Boolean filter, it is very useful to follow the derivation of the expression for $\mathcal{S}_f^2$ given by (3.5), when $\sum_{\ell=1}^{L-1} f_\ell$ was substituted for $\mathcal{S}_f$ directly, instead of using the relation (3.4). We have

$$\begin{aligned}\mathcal{S}_f^2 &= \left(\sum_{\ell=1}^{L-1} f_\ell\right)^2 \\ &= \sum_{\ell=1}^{L-1}\left(\sum_{\lambda=1}^{L-1} f_\ell f_\lambda\right) \\ &= \sum_{\ell=1}^{L-1}\sum_{\lambda=1}^{\ell} f_\ell f_\lambda + \sum_{\ell=1}^{L-2}\sum_{\lambda=\ell+1}^{L-1} f_\ell f_\lambda\,. \end{aligned} \tag{3.13}$$

Now, since the Boolean function f has the stacking property, it follows that for any $\lambda \leq \ell$, we have $f_\ell f_\lambda = f_\ell$. Consequently, the terms containing cross-product terms of the form $f_\ell(\boldsymbol{x}_i) \cdot f_\lambda(\boldsymbol{x}_j)$ are actually linear terms in the design variable $f(\boldsymbol{x}_i)$. Thus,

$$\sum_{\lambda=1}^{\ell} f_\ell f_\lambda = \ell f_\ell\,. \tag{3.14}$$

Let us now evaluate the second double summation appearing in (3.13). Note that $\lambda \in [\ell+1, L-1]$ is greater than ℓ, and therefore, $f_\ell f_\lambda = f_\lambda$. We have

$$\begin{aligned}\sum_{\ell=1}^{L-2}\sum_{\lambda=\ell+1}^{L-1} f_\ell f_\lambda &= f_2 + f_3 + \ldots + f_{L-1} \\ &\quad + f_3 + \ldots + f_{L-1} \\ &\quad + \ldots + \ldots \\ &\quad + f_{L-1} \\ &= \sum_{\ell=1}^{L-1}(\ell-1) f_\ell\,. \end{aligned} \tag{3.15}$$

Now, by substituting from (3.14) and (3.15) in (3.13), we get

$$\mathcal{S}_f^2 = \sum_{\ell=1}^{L-1} \ell f_\ell + \sum_{\ell=1}^{L-1}(\ell-1) f_\ell\,, \tag{3.16}$$

which is the same as (3.5). It can be observed that the expression for S^2 as given by (3.6) can also be obtained by following the same approach as above, based on the structure of the threshold decomposition of S.

Based on the technique of decomposing the MSE in signal estimation using stack filters proposed in this section, we next investigate the possibility of deriving a binary-level expression similar to (3.8) in the more general case of an objective function given by the pth order error between the desired and estimated signals.

3.3 L_p Norm Design of Stack Filters

The design of stack filters is generally carried out under the assumption of first-order ergodicity (i.e., it is assumed that sample averages are equal to time averages), using "training" sequences [47], [81]. In this thesis, by using the same assumption, the terms of pth order error norm and L_p norm are used interchangebly. The pth order error between the desired signal and the filter's output is defined as the expected value of the pth power of the multilevel instantaneous error $e = (\mathrm{S} - \mathcal{S}_f)$, i.e.,

$$E[\,|\,e\,|^p\,] = E[\,|\,\mathrm{S} - \mathcal{S}_f\,|^p\,]\ . \tag{3.17}$$

Now, in the case of training-based design of stack filters, we are interested to evaluate the L_p norm of the error between the desired and estimated signals, which is formally defined as

$$L_p(\mathcal{S}_f) \triangleq \left[\sum_{n=1}^{N} |\,e(n)\,|^p\right]^{\frac{1}{p}}\ , \tag{3.18}$$

where N denotes the length of the training set.

The derivation of a binary-level expression for the L_p norm of the error between the desired and estimated signals is obtained using the results given by the following two theorems.

Theorem 3.1 In stack filtering, the multilevel instantaneous error e raised to the power p, that is, $e^p = (\mathrm{S} - \mathcal{S}_f)^p$, can be expressed as

$$(\mathrm{S} - \mathcal{S}_f)^p = \sum_{\ell=1}^{\mathrm{L}-1} \mathrm{A}_p(\ell)\cdot(\mathrm{s}_\ell - f_\ell)\ , \tag{3.19}$$

where

$$\mathrm{A}_p(\ell) \triangleq (\mathrm{S} - \ell + 1)^p - (\mathrm{S} - \ell)^p . \tag{3.20}$$

Proof. We will prove this theorem by induction. It can be easily seen that the expression (3.19) holds for $p = 1$ and $p = 2$, since for these particular cases, the relation given by (3.19) reduces to those given by (3.1) and (3.7), respectively. We now assume that the result is true for $p - 1$, i.e.,

$$(\mathrm{S} - \mathcal{S}_f)^{p-1} = \sum_{\ell=1}^{\mathrm{L}-1} \mathrm{A}_{p-1}(\ell) \cdot (\mathrm{s}_\ell - f_\ell) , \tag{3.21}$$

where $\mathrm{A}_{p-1}(\ell) = (\mathrm{S} - \ell + 1)^{p-1} - (\mathrm{S} - \ell)^{p-1}$, and prove that it is also true for p. The pth power of the error, $(\mathrm{S} - \mathcal{S}_f)^p$, can be expressed as

$$\begin{aligned}(\mathrm{S} - \mathcal{S}_f)^p &= (\mathrm{S} - \mathcal{S}_f)^{p-1}(\mathrm{S} - \mathcal{S}_f) \\ &= \sum_{\ell=1}^{\mathrm{L}-1} \left\{ \mathrm{A}_{p-1}(\ell) \cdot \sum_{\lambda=1}^{\mathrm{L}-1} [(\mathrm{s}_\ell - f_\ell)(\mathrm{s}_\lambda - f_\lambda)] \right\} .\end{aligned} \tag{3.22}$$

In order to evaluate the inner summation, let it be denoted by T, i.e.,

$$T \triangleq \sum_{\lambda=1}^{\mathrm{L}-1} [(\mathrm{s}_\ell - f_\ell)(\mathrm{s}_\lambda - f_\lambda)] . \tag{3.23}$$

Observe that the general term in the above summation, $t_\lambda \triangleq (\mathrm{s}_\ell - f_\ell)(\mathrm{s}_\lambda - f_\lambda)$, can be written as

$$t_\lambda = (\mathrm{s}_\ell \mathrm{s}_\lambda - f_\ell \mathrm{s}_\lambda) + (\mathrm{s}_\ell \mathrm{s}_\lambda - f_\lambda \mathrm{s}_\ell) - \mathrm{s}_\ell \mathrm{s}_\lambda + f_\ell f_\lambda . \tag{3.24}$$

Now, the summation T becomes

$$T \;=\; \mathrm{S}\,(\mathrm{s}_\ell - f_\ell) \;+\; \mathrm{s}_\ell \sum_{\lambda=1}^{\mathrm{L}-1} (\mathrm{s}_\lambda - f_\lambda) \;-\; \ell\,(\mathrm{s}_\ell - f_\ell) \;-\; \sum_{\lambda=\ell+1}^{\mathrm{L}-1} (\mathrm{s}_\lambda - f_\lambda) . \tag{3.25}$$

Note that the quantities $-\ell(\mathrm{s}_\ell - f_\ell)$ and $-\sum_{\lambda=\ell+1}^{\mathrm{L}-1}(\mathrm{s}_\lambda - f_\lambda)$ in the above expression for T have been obtained from $-\sum_{\lambda=1}^{\mathrm{L}-1} \mathrm{s}_\ell \mathrm{s}_\lambda$ and $\sum_{\lambda=1}^{\mathrm{L}-1} f_\ell f_\lambda$ by using the technique of decomposing the latter summations over the intervals $[1,\ell]$ and $[\ell+1, \mathrm{L}-1]$, as done in (3.13). Now, $(\mathrm{S} - \mathcal{S}_f)^p$ becomes

$$(\mathrm{S} - \mathcal{S}_f)^p = \sum_{\ell=1}^{\mathrm{L}-1} [\mathrm{A}_{p-1}(\ell)(\mathrm{S}-\ell)(\mathrm{s}_\ell - f_\ell)] + U - V \,, \tag{3.26}$$

where U and V denote, respectively,

$$U \triangleq \sum_{\ell=1}^{\mathrm{L}-1} \left[\mathrm{A}_{p-1}(\ell)\ \mathrm{s}_\ell \sum_{\lambda=1}^{\mathrm{L}-1} (\mathrm{s}_\lambda - f_\lambda) \right] , \tag{3.27}$$

and

$$V \triangleq \sum_{\ell=1}^{\mathrm{L}-2} \left[\mathrm{A}_{p-1}(\ell) \sum_{\lambda=\ell+1}^{\mathrm{L}-1} (\mathrm{s}_\lambda - f_\lambda) \right] . \tag{3.28}$$

The expression for U can be simplified as

$$\begin{aligned} U &= \left[\sum_{\ell=1}^{\mathrm{L}-1} \mathrm{A}_{p-1}(\ell)\ \mathrm{s}_\ell \right] \cdot \left[\sum_{\lambda=1}^{\mathrm{L}-1} (\mathrm{s}_\lambda - f_\lambda) \right] \\ &= \left\{ \sum_{\ell=1}^{\mathrm{S}} \left[(\mathrm{S}-\ell+1)^{p-1} - (\mathrm{S}-\ell)^{p-1} \right] \right\} \cdot \left[\sum_{\lambda=1}^{\mathrm{L}-1} (\mathrm{s}_\lambda - f_\lambda) \right] \\ &= \mathrm{S}^{p-1} \sum_{\lambda=1}^{\mathrm{L}-1} (\mathrm{s}_\lambda - f_\lambda) \,. \end{aligned} \tag{3.29}$$

Similarly, V as given by (3.28) can also be expressed as

$$\begin{aligned} V = \quad & \left[\mathrm{S}^{p-1} - (\mathrm{S}-1)^{p-1} \right] \cdot \{ (\mathrm{s}_2 - f_2) + (\mathrm{s}_3 - f_3) + \ \dots \ + (\mathrm{s}_{\mathrm{L}-1} - f_{\mathrm{L}-1}) \} \\ & + \left[(\mathrm{S}-1)^{p-1} - (\mathrm{S}-2)^{p-1} \right] \cdot \qquad \{ (\mathrm{s}_3 - f_3) + \ \dots \ + (\mathrm{s}_{\mathrm{L}-1} - f_{\mathrm{L}-1}) \} \\ & + \dots \\ & + \left[(\mathrm{S}-\mathrm{L}+3)^{p-1} - (\mathrm{S}-\mathrm{L}+2)^{p-1} \right] \cdot \qquad (\mathrm{s}_{\mathrm{L}-1} - f_{\mathrm{L}-1}) \\ = \quad & \sum_{\ell=1}^{\mathrm{L}-1} \left[\mathrm{S}^{p-1} - (\mathrm{S}-\ell+1)^{p-1} \right] \cdot (\mathrm{s}_\ell - f_\ell) \,. \end{aligned} \tag{3.30}$$

By substituting for U and V as given by (3.29) and (3.30), respectively, in (3.26), we obtain

$$(\mathrm{S}-\mathcal{S}_f)^p = \sum_{\ell=1}^{\mathrm{L}-1}\left[(\mathrm{S}-\ell+1)^{p-1}-(\mathrm{S}-\ell)^{p-1}\right](\mathrm{S}-\ell)(\mathrm{s}_\ell - f_\ell) \; +$$

$$+ \;\; \mathrm{S}^{p-1}\sum_{\ell=1}^{\mathrm{L}-1}(\mathrm{s}_\ell - f_\ell) \;\; - \;\; \sum_{\ell=1}^{\mathrm{L}-1}\left[\mathrm{S}^{p-1}-(\mathrm{S}-\ell+1)^{p-1}\right]\cdot(\mathrm{s}_\ell - f_\ell)\,, \qquad (3.31)$$

and it can be easily seen that (3.31) is indeed equivalent to (3.19), as required.

(Q.E.D.)

Now, based on the result of Theorem 3.1, we state another important property of stack filters, concerning the absolute value of the pth power of the multilevel error in stack filtering.

Theorem 3.2 In stack filtering, the absolute value of the pth order error between the desired signal and the estimated one can be determined as a summation of the absolute values of the weighted errors, $e_\mathrm{b}(\ell) \triangleq \mathrm{A}_p(\ell)\cdot(\mathrm{s}_\ell - f_\ell)$, appearing at the binary levels of the filter, i.e.,

$$|\,\mathrm{S}-\mathcal{S}_f\,|^p = \sum_{\ell=1}^{\mathrm{L}-1}|\mathrm{A}_p(\ell)|\cdot|\,\mathrm{s}_\ell - f_\ell\,|\;. \qquad (3.32)$$

Proof. We will prove this theorem by considering the following three cases.

Case (i): $\mathrm{S}=\mathcal{S}_f$. In this case, the relation given by (3.32) is trivially satisfied, since both sides become zero.

Case (ii): $\mathrm{S}<\mathcal{S}_f$. In this case, $\mathrm{s}_\ell = f_\ell = 1$ for $\ell = 1,2,\ldots,\mathrm{S}$, $\mathrm{s}_\ell = f_\ell = 0$ for $\ell=\mathcal{S}_f+1,\mathcal{S}_f+2,\ldots,\mathrm{L}-1$, and $\mathrm{s}_\ell - f_\ell < 0$ for $\ell=\mathrm{S}+1,\mathrm{S}+2,\ldots,\mathcal{S}_f-1,\mathcal{S}_f$. Therefore, we can write

$$\sum_{\ell=1}^{\mathrm{L}-1}\mathrm{A}_p(\ell)\cdot(\mathrm{s}_\ell - f_\ell) = -\sum_{\ell=\mathrm{S}+1}^{\mathcal{S}_f}\mathrm{A}_p(\ell)\cdot|\,\mathrm{s}_\ell - f_\ell\,|\;. \qquad (3.33)$$

Now, since $S < \mathcal{S}_f$, we can also write

$$S - \mathcal{S}_f = (-1) \cdot | S - \mathcal{S}_f | \ , \tag{3.34}$$

and substituting from (3.33) and (3.34) in (3.19) we have that

$$(-1)^p \cdot | S - \mathcal{S}_f |^p = - \sum_{\ell=S+1}^{\mathcal{S}_f} A_p(\ell) \cdot | s_\ell - f_\ell | \ . \tag{3.35}$$

Now, depending on whether p is odd or even, we can have two possible situations for the sign of each coefficient $A_p(\ell)$, with $\ell \in [S+1, \mathcal{S}_f]$. We analyze each case separately.

a) When p is odd, and since $(S - \ell + 1) > (S - \ell)$, we have $(S - \ell + 1)^p > (S - \ell)^p$, and therefore each coefficient $A_p(\ell)$ is strictly positive. Thus, in this case, (3.35) becomes

$$\begin{aligned} - | S - \mathcal{S}_f |^p &= - \sum_{\ell=S+1}^{\mathcal{S}_f} |A_p(\ell)| \cdot | s_\ell - f_\ell | \\ &= - \sum_{\ell=1}^{L-1} |A_p(\ell)| \cdot | s_\ell - f_\ell | \ , \end{aligned} \tag{3.36}$$

and consequently, the relation given by (3.32) is satisfied. Note that in (3.36), it has been possible to extend the summation interval from $[S+1, \mathcal{S}_f]$ to $[1, L-1]$, since $(s_\ell - f_\ell) = 0$ for $\ell \leq S$, and $\ell > \mathcal{S}_f$.

b) When p is even, and since $(S - \ell + 1) > (S - \ell)$ and $(S - \ell)$ is strictly negative, we have $(S - \ell + 1)^p < (S - \ell)^p$. Thus, in this situation, all coefficients $A_p(\ell)$ are strictly negative, and therefore (3.35) can be written as

$$| S - \mathcal{S}_f |^p = \sum_{\ell=1}^{L-1} |A_p(\ell)| \cdot | s_\ell - f_\ell | \ , \tag{3.37}$$

and the relation given by (3.32) is again satisfied.

Case (iii): $S > \mathcal{S}_f$. In this case, $s_\ell = f_\ell = 1$ for $\ell = 1, 2, \ldots, \mathcal{S}_f$, $s_\ell = f_\ell = 0$ for $\ell = S+1, S+2, \ldots, L-1$, and $s_\ell - f_\ell > 0$ for $\ell = \mathcal{S}_f + 1, \mathcal{S}_f + 2, \ldots, S$. Thus, we have that

$$\sum_{\ell=1}^{L-1} A_p(\ell) \cdot (s_\ell - f_\ell) = \sum_{\ell=\mathcal{S}_f+1}^{S} A_p(\ell) \cdot | s_\ell - f_\ell | \ , \tag{3.38}$$

and since $\mathrm{S} > \mathcal{S}_f$, we can also write

$$\mathrm{S} - \mathcal{S}_f = |\,\mathrm{S} - \mathcal{S}_f\,| \; . \tag{3.39}$$

Now, substituting from (3.38) and (3.39) in (3.19), we have

$$\begin{aligned} |\,\mathrm{S} - \mathcal{S}_f\,|^p &= \sum_{\ell=\mathcal{S}_f+1}^{\mathrm{S}} \mathrm{A}_p(\ell) \cdot |\,\mathrm{s}_\ell - f_\ell\,| \\ &= \sum_{\ell=1}^{\mathrm{L}-1} |\mathrm{A}_p(\ell)| \cdot |\,\mathrm{s}_\ell - f_\ell\,| \; , \end{aligned} \tag{3.40}$$

and the relation given by (3.32) is satisfied. (Q.E.D.)

As a consequence of Theorem 3.2, the expected value of the pth order error given by (3.17) and the L_p norm given by (3.18) can be determined, respectively, as

$$E[\,|\,e\,|^p\,] = \sum_{\ell=1}^{\mathrm{L}-1} E[\,|\mathrm{A}_p(\ell)| \cdot |\,\mathrm{s}_\ell - f_\ell\,|\,] \tag{3.41}$$

and

$$L_p(\mathcal{S}_f) = \left\{ \sum_{n=1}^{N} \sum_{\ell=1}^{\mathrm{L}-1} |\mathrm{A}_p(\ell, n)| \cdot |\,\mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))\,| \right\}^{\frac{1}{p}} \; , \tag{3.42}$$

where

$$\mathrm{A}_p(\ell, n) = (\mathrm{S}(n) - \ell + 1)^p - (\mathrm{S}(n) - \ell)^p \; . \tag{3.43}$$

Thus, in stack filtering, both $E[\,|\,e\,|^p\,]$ and $L_p^p(\mathcal{S}_f)$ are linear functions of the decision errors at the binary levels of the filter. Following an approach similar to the one used in the derivation of the expression for the MSE given by (3.10), it can be shown that $E[\,|\mathrm{A}_p(\ell)| \cdot |\,\mathrm{s}_\ell - f_\ell\,|\,]$ can be expressed as

$$\begin{aligned} E[\,|\mathrm{A}_p(\ell)| \cdot |\,\mathrm{s}_\ell - f_\ell\,|\,] = \sum_{k=0}^{\mathrm{L}-1} \{ \sum_{j=1}^{2^{\mathrm{M}}} [\,|\,(k-\ell+1)^p - (k-\ell)^p\,| \cdot P_\ell(0, \boldsymbol{b}_j | k) \cdot f(\boldsymbol{b}_j) \; + \\ + \; |\,(k-\ell+1)^p - (k-\ell)^p\,| \cdot P_\ell(1, \boldsymbol{b}_j | k) \cdot (1 - f(\boldsymbol{b}_j))\,]\,\} \cdot P(k) \; , \end{aligned} \tag{3.44}$$

and therefore, we have

$$E[\,|\,e\,|^p\,] = \sum_{j=1}^{2^M} [\,\alpha_p(j)\cdot f(\boldsymbol{b}_j) + \beta_p(j)\,]\,, \tag{3.45}$$

where

$$\alpha_p(j) = \sum_{k=0}^{L-1}\sum_{\ell=1}^{L-1} [\,|\,(k-\ell+1)^p - (k-\ell)^p\,|\cdot(\,P_\ell(0,\boldsymbol{b}_j,k) - P_\ell(1,\boldsymbol{b}_j,k)\,)\,]\;, \tag{3.46}$$

and

$$\beta_p(j) = \sum_{k=0}^{L-1}\sum_{\ell=1}^{L-1} [\,|\,(k-\ell+1)^p - (k-\ell)^p\,|\cdot P_\ell(1,\boldsymbol{b}_j,k)\,]\;. \tag{3.47}$$

As a result, a stack filter which is optimal for signal estimation in the sense of the L_p norm can be determined as the solution of a linear program with the same constraint matrix imposing the stacking property on f as the linear program that finds a minimum MAE stack filter [17].

It is interesting to emphasize here that by letting $\mathrm{S} = 0$ in the expression for the pth order error given by (3.19), an expression for the pth order moment of the output of a stack filter about the origin can be obtained as

$$E\left[S_f^p\right] = \sum_{\ell=1}^{L-1} E[\,(\,\ell^p - (\ell-1)^p\,)\cdot f_\ell\,]\;. \tag{3.48}$$

Note that in [90], an expression for the moments of the output of a stack filter has been derived, based on a formula for the filter's output distribution. Thus, (3.48) gives an alternative derivation of the output moments of stack filters, based on their responses at the binary levels. This, in turn, suggests that in addition to the immediate application of Theorems 3.1 and 3.2 to the design of stack filters, the results of these theorems could also provide a new insight in the statistical analysis of stack filters.

Based on the expression for the L_p norm as given by (3.42), we next investigate the possibility of formulating the problem of minimax stack filtering.

3.4 Formulation of the Minimax Design Problem

Apart from being more general than the MAE or MSE design techniques, the L_p norm approach to stack filter design also offers a sound mathematical framework to formulate the design problem of minimizing the maximum absolute error between the desired and estimated signals (i.e., the minimax design problem). It should be pointed out that a previous attempt to formulate the minimax problem of stack filter design has used a heuristic technique, in which the original L_∞ problem of minimizing the multilevel maximum absolute error has been replaced by a simpler problem of minimizing the maximum average error between the output of the filter and its input [25], [28].

In this thesis, in order to formulate the L_∞ design problem, we use the general expression for the L_p norm as given by (3.42), and follow the commonly used approach of multiplying and dividing the L_p norm by the maximum absolute error, and then taking the limit as $p \to \infty$ [6]. In the specific case of our problem, it can be observed that the maximum absolute error between the desired and estimated signals can be expressed as

$$e_{\mathcal{M}} \triangleq \max_n \max_\ell \mathrm{A}_{\mathcal{M}}(\ell, n) \ , \tag{3.49}$$

where

$$\mathrm{A}_{\mathcal{M}}(\ell, n) \triangleq e_b(\ell, n) \cdot \max\{|\mathrm{S}(n) - \ell + 1| \, , |\mathrm{S}(n) - \ell|\} \ , \tag{3.50}$$

and

$$e_b(\ell, n) \triangleq |\, \mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))\, | \ . \tag{3.51}$$

By multiplying and dividing the right side of (3.42) by $e_{\mathcal{M}}$, we obtain

$$L_p(\mathcal{S}_f) = e_{\mathcal{M}} \cdot \left\{ \sum_{n=1}^{N} \sum_{\ell=1}^{\mathrm{L}-1} \frac{|\mathrm{A}_p(\ell, n)|}{|e_{\mathcal{M}}|^p} \cdot |\mathrm{s}_\ell(n) - f(\boldsymbol{x}_\ell(n))| \right\}^{\frac{1}{p}} \ . \tag{3.52}$$

Note that, by definition (see (3.49)-(3.51)), $e_{\mathcal{M}}$ is a positive constant, and thus, $|e_{\mathcal{M}}| = e_{\mathcal{M}} > 0$. By taking the limit of both sides of (3.52) as $p \to \infty$, all the terms of the double summation become zero, except those for which $\mathrm{A}_{\mathcal{M}}(\ell, n) = e_{\mathcal{M}}$. Each of the nonzero terms assumes a value of 1, and their summation is a finite integer κ. Thus, since $\lim_{p\to\infty} \kappa^{1/p} = 1$, the minimax stack filter can be determined as the solution to the following optimization problem:

$$\min_{f} \max_{n} \max_{\ell} \mathrm{A}_{\mathcal{M}}(\ell, n) , \tag{3.53}$$

where $\mathrm{A}_{\mathcal{M}}(\ell, n)$ is given by (3.50).

The exact solution to the minimax problem given by (3.52) requires a combinatorial search over all possible PBF filters of size M. However, it is known that the total number of PBFs of size M is greater than $2^{2^{\mathrm{M}/2}}$ [27]. Thus, in practical engineering problems, it is convenient to approximate the minimax solution by chosing a large value for p, and minimizing the cost function L_p^p with L_p given by (3.42).

3.5 Design of Weighted Order Statistic Filters

The complexity of the linear program to find an optimal stack filter increases faster than exponentially with the size of the filter window. This has led to the development of an alternative approach of designing stack filters. Specifically, in [91], it has been proposed to restrict the design problem to those subclasses of stack filters of well-known practical significance in signal processing. One such example is the subclass of stack filters defined by linearly separable PBFs (LSPBFs). These filters perform the operation of WOS filtering at the multilevel domain [2], [34], [71]. The problem of designing a WOS filter which is optimal in the MAE sense has been solved in [91]. In this section, based on the formulation of the pth order error between the desired and estimated signals in stack filtering, as given in Section 3.3, we extend the MAE design of WOS filters of [91] to the L_p norm-based design.

3.5.1 Definition of a WOS Filter

As shown in [34], the output of a WOS filter at an instant n can be obtained by the following procedure:

(a) replicate each input sample, $\mathrm{X}(n-j)$ $(j = 0, 1, \ldots, \mathrm{M}-1)$, appearing in the filter's input window (which is denoted by $\mathbf{x}(n) = \{\mathrm{X}(n), \mathrm{X}(n-1), \ldots, \mathrm{X}(n-\mathrm{M}+1)\}$)[1] at time n, by a given positive integer w_j called the weight;

(b) sort the resulting vector of $\sum_{j=0}^{\mathrm{M}-1} \mathrm{w}_j$ elements;

(c) choose the $\mathrm{w_T}$-th largest value ($\mathrm{w_T}$ denotes a positive integer called the threshold) from the sorted vector.

Therefore, a WOS filter is completely determined by a set of positive integer weights, $\mathrm{w}_0, \mathrm{w}_1, \ldots, \mathrm{w}_{\mathrm{M}-1}$, and $\mathrm{w_T}$, and it is defined by the following input-output relation:

$$\mathrm{WOS}(\mathbf{x}(n)) = \mathrm{w_T}\text{-th largest value in the set}$$

$$\left\{ \overbrace{\mathrm{X}(n), \ldots, \mathrm{X}(n)}^{\mathrm{w}_0 \text{ times}}, \ldots, \overbrace{\mathrm{X}(n-\mathrm{M}+1), \ldots, \mathrm{X}(n-\mathrm{M}+1)}^{\mathrm{w}_{\mathrm{M}-1} \text{ times}} \right\}.$$

A WOS filter is a special type of stack filter, which is characterized by a linearly separable PBF [34], [54], [88]. A PBF $f(\boldsymbol{x})$ is said to be linearly separable if it can be expressed in the form

$$f(x_0, x_1, \ldots, x_{\mathrm{M}-1}) = \begin{cases} 1 & \text{if } \sum_{j=0}^{\mathrm{M}-1} \mathrm{w}_j \cdot x_j \geq \mathrm{w_T} \\ 0 & \text{otherwise}, \end{cases} \tag{3.54}$$

[1] In this thesis, the samples appearing in the filter window $\mathbf{x}(n)$ are interchangebly denoted either by $\{\mathrm{X}(n), \mathrm{X}(n-1), \ldots, \mathrm{X}(n-\mathrm{M}+1)\}$, or by $\{\mathrm{x}_0, \mathrm{x}_1, \ldots, \mathrm{x}_{\mathrm{M}-1}\}$. The latter notation is more compact, since the dependency of the variables designating the window samples on the window-position (time) n is assumed to be implicit.

where all w_j's and w_T are positive real numbers. A WOS filter characterized by the weights $\tilde{w}_0, \tilde{w}_1, \ldots, \tilde{w}_{M-1}$, and $\tilde{w}_T$, is said to be normalized if $\tilde{w}_i = w_i / \left(\sum_{j=0}^{M-1} w_j\right)$, $i = 0, 1, \ldots, M-1$, and $\tilde{w}_T = w_T / \left(\sum_{j=0}^{M-1} w_j\right)$. Thus, for a normalized WOS filter, we have $\sum_{i=0}^{M-1} \tilde{w}_i = 1$. The operation of normalizing the weights $\mathbf{w}$ of a WOS filter is used in the experimental investigation described in the next section to compare the weights of the WOS filters designed using different error criteria.

3.5.2 L_p Norm Design of WOS Filters

The L_p norm design of a WOS filter requires the determination of the weights $\mathbf{w} = [w_0 \, w_1 \, \ldots \, w_{M-1}]$, such that the L_p norm given by (3.42) or the pth order error of (3.41), in estimating a signal $S(n)$ from a noise corrupted observation $X(n)$ of the same, is minimized. In order to derive algorithms for this design, it is noted that $E[\,|e|^p\,]$ given by (3.41) can be equivalently expressed as

$$E[\,|e|^p\,] = \sum_{\ell=1}^{L-1} E\left[\,|A_p(\ell)| \cdot (s_\ell - f_\ell)^2\,\right] . \tag{3.55}$$

Thus, the L_p norm design of WOS filters involves finding a PBF $f(\boldsymbol{x})$ which minimizes (3.55), subject to $f(\boldsymbol{x})$ being linearly separable. In order to overcome the difficulty of imposing the constraint of linear separability, we follow an approach commonly used in WOS filter design [91], [93], in which a linear approximation of $f(\boldsymbol{x})$ is employed. With this approximation, the L_p norm design of WOS filters is carried out by minimizing a cost function given as

$$J_p(\mathbf{w}) = \sum_{\ell=1}^{L-1} E\left[\,|A_p(\ell)| \cdot (s_\ell - \mathbf{w} \cdot \boldsymbol{x}_\ell^T\,)^2\,\right] . \tag{3.56}$$

One may notice that after replacing the function $f(\boldsymbol{x})$ appearing in the expression for $E[\,|e|^p\,]$ given by (3.55) by a linear function, the WOS filter becomes, in fact, a linear FIR filter. That is, the design problem reduces to finding an optimal linear FIR filter with nonnegative weights. However, in contrast to the traditional LMS linear filtering [87], this optimal FIR filter minimizes a weighted sum of squared errors incurred at the levels of the threshold decomposition architecture.

The positive weights $\mathbf{w}^* = \left[\mathrm{w}_0^* \, \mathrm{w}_1^* \ldots \mathrm{w}_{M-1}^*\right]$ that minimize (3.56), determine a LSPBF f^* given by

$$f^*(x_0, x_1, \ldots, x_{M-1}) = \begin{cases} 1 & \text{if} \quad \sum_{j=0}^{M-1} \mathrm{w}_j^* \cdot x_j \geq 0.5 \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.57}$$

Following an approach similar to the one given in [91] for the case of designing an approximately optimal MAE WOS filter, it can be shown that $J_p(\mathbf{w}^*)$ is close to the L_p norm achieved by an L_p-optimal stack filter, in spite of the fact that $J_p(\mathbf{w})$ of (3.56) is not identical to $E[\,|\,e\,|^p\,]$ given by (3.55).

Now, using the cost function $J_p(\mathbf{w})$ given by (3.56), nonadaptive and adaptive algorithms for the L_p norm-based design of WOS filters are derived.

(i) Nonadaptive Design

The gradient vector of the cost function $J_p(\mathbf{w})$ of (3.56) is given by

$$\nabla J_p = -2 \sum_{\ell=1}^{L-1} E\left[|\mathrm{A}_p(\ell)| \cdot \left(\mathrm{s}_\ell(n) - \mathbf{w} \cdot \boldsymbol{x}_\ell^{\mathrm{T}}\right) \boldsymbol{x}_\ell^{\mathrm{T}}\right]. \tag{3.58}$$

Thus, when the positivity constraints are not imposed on $\mathbf{w}$, a WOS filter which is approximatley optimal in the sense of the L_p norm can be obtained as the solution of the following set of linear equations

$$\mathbf{R} \cdot \mathbf{w}^* = \mathbf{c}\,, \tag{3.59}$$

where $\mathbf{R}$ is an autocorrelation matrix given by

$$\mathbf{R} = \sum_{\ell=1}^{L-1} E\left[|\mathrm{A}_p(\ell)| \cdot \boldsymbol{x}_\ell^{\mathrm{T}} \cdot \boldsymbol{x}_\ell\right], \tag{3.60}$$

and $\mathbf{c}$ is a cross-correlation vector given as

$$\mathbf{c} = \sum_{\ell=1}^{L-1} E\left[|\mathrm{A}_p(\ell)| \cdot \mathrm{s}_\ell \cdot \boldsymbol{x}_\ell^{\mathrm{T}}\right]. \tag{3.61}$$

The entries of the autocorrelation matrix can be easily evaluated at the multilevel domain as

$$\mathbf{R}(i,j) = E\left[\sum_{\ell=1}^{\mathbf{L_R}(i,j)} |(\mathrm{S}(n) - \ell + 1)^p - (\mathrm{S}(n) - \ell)^p|\right], \tag{3.62}$$

with

$$\mathbf{L_R}(i,j) = \min\{\mathrm{X}(n-i), \mathrm{X}(n-j)\}, \tag{3.63}$$

and $i, j = 0, 1, \ldots, \mathrm{M}-1$. Similarly, the entries of the cross-correlation vector $\mathbf{c}$ can be determined as

$$\mathbf{c}(i) = E\left[\sum_{\ell=1}^{\mathbf{L_c}(i)} |(\mathrm{S}(n) - \ell + 1)^p - (\mathrm{S}(n) - \ell)^p|\right], \tag{3.64}$$

with

$$\mathbf{L_c}(i) = \min\{\mathrm{X}(n-i), \mathrm{S}(n)\}, \tag{3.65}$$

and $i = 0, 1, \ldots, \mathrm{M}-1$. Note that in the special case of $p = 1$, $\mathbf{R}$ and $\mathbf{c}$ become equal to the morphological correlation matrices appearing in the conventional MAE design of WOS filters developed in [91], i.e.,

$$\mathbf{R}(i,j) = E[\min\{\mathrm{X}(n-i), \mathrm{X}(n-j)\}], \tag{3.66}$$

and

$$\mathbf{c}(i) = E[\min\{\mathrm{X}(n-i), \mathrm{S}(n)\}]. \tag{3.67}$$

With linear inequality constraints of imposing the positivity of the weights, the L_p norm-based optimization problem can be solved by a gradient projection method similar to the one used for the design of MAE WOS filters [91], [93].

(ii) Adaptive Design

When the statistics of the observed and desired signals are not available, the following adaptive algorithm can be used to estimate the weights of the L_p optimal WOS filter:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \sum_{\ell=1}^{\mathrm{L}-1} \left[|\mathrm{A}_p(\ell,n)| \left(\mathrm{s}_\ell(n) - \mathbf{w}(n) \cdot \boldsymbol{x}_\ell^T(n)\right) \boldsymbol{x}_\ell^T(n)\right]. \tag{3.68}$$

An alternative adaptive design algorithm can be derived by using a sigmoidal approximation for the linearly separable PBF $f(\boldsymbol{x})$ appearing in the expression given by (3.55) for the cost function $E[|e|^p]$, instead of the linear approximation which has been used for the derivation of (3.68).

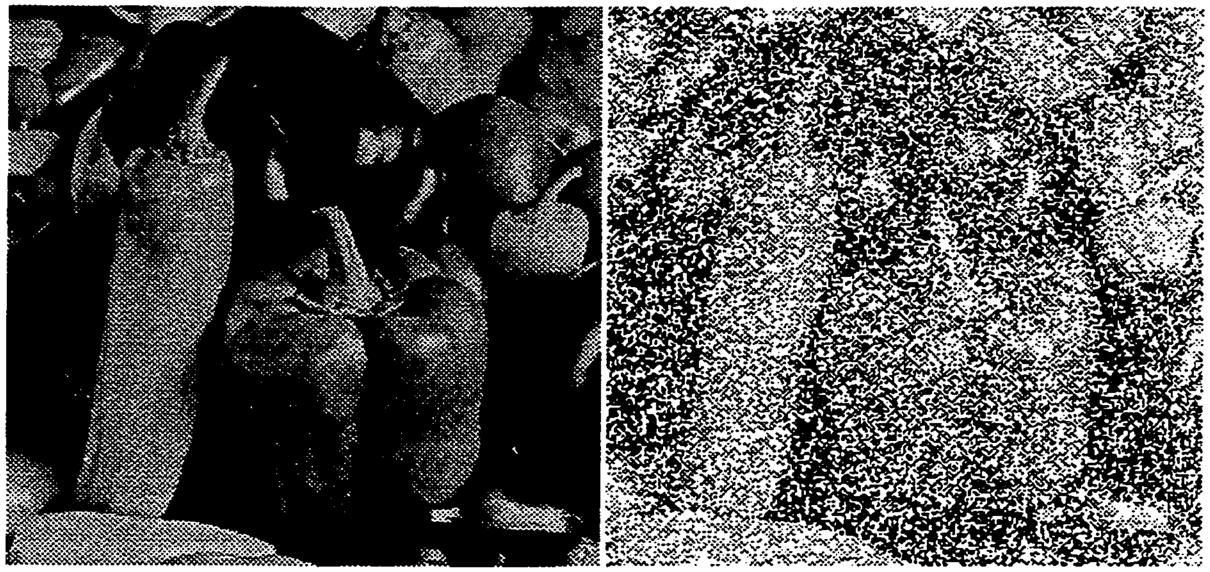
While nonadaptive algorithms are known to yield good results when the observed and desired signals are jointly stationary, an adaptive algorithm can track the time-varying statistics of the signals and it is suitable for low-cost implementations [93].

3.6 Experimental Investigation

The fact that it is possible to design stack filters using a large variety of error criteria is an important analytical result, broadening the theoretical understanding of these filters. In addition, this result is expected to be useful in practical engineering problems as well. For instance, as suggested in the introduction to this chapter, in applications involving restoration of signals corrupted with impulsive noise, the use of the L_p norm design with $p > 1$ could provide improved results compared to those achieved by using the conventional MAE design. To investigate this possibility, a typical application of restoring images corrupted with impulsive noise is next considered. Experiments employing two different test images are carried out. The Matlab programs used in designing the stack filters of these experiments are listed in Appendices A—C.

Experiment I

Figures 3.2(a) and (b) show, respectively, a 256×256 8-bit Peppers image and its noise-corrupted version, which are used in this experiment. The noise-corrupted image of Figure 3.2(b) has been generated by adding positive impulsive noise to the noise-free image of Figure 3.2(a). The impulsive noise has a probability of occurrence of 0.45, and a magnitude of 255. By considering an example where the additive impulsive noise is purely positive and has a high probability of occurrence, we expect that the filter with best per-



(a) (b)

Figure 3.2: A 256 $\times$ 256 8-bit Peppers image. (**a**) The original image. (**b**) Noise-corrupted version of (a).

formance in terms of removing the impulsive noise should be close to a rank order filter selecting the smallest element within the filter window (i.e., the minimum filter).

Figure 3.3 illustrates the processed images obtained by applying the L_p-optimal stack filters of size 3×3 with $p = 1, 2, 3$, and 4, to the degraded image of Figure 3.2(b). The L_4-optimal stack filter is the minimum filter and, as seen from Figure 3.3, it is capable of removing more impulses compared to the conventional MMAE stack filter, as well as compared to the L_2- and L_3-optimal stack filters. By carying out the L_p norm designs with $p = 5, 6, \ldots, 9$, and 10, we have also observed in this experiment that not only the L_4-optimal stack filter is the minimum filter but also the L_p-optimal filters with $p = 5, 6, \ldots, 9$, and 10 are minimum filters. This leads to a conjecture that, actually, in this example, the minimum filter is the L_∞-optimal stack filter, but a value of $p \geq 4$ is large enough for the L_p norm design to yield the minimum stack filter. The values of the MAE and the L_p errors, with $p = 2, 3$, and 4, of the restored images of Figure 3.3 are listed in Table 3.1.

Figure 3.4 illustrates the processed images obtained by applying the L_p WOS filters of size 3×3 with $p = 1, 5, 8$, and 10. The values of the L_p errors, with $p = 1, 5, 8$, and 10, of the restored images of Figure 3.4 are listed in Table 3.2, while the corresponding values of $\sqrt[p]{J_p(\mathbf{w})}$ are given in Table 3.3. The weights and threshold values of the normalized L_p WOS filters with $p = 1, 5, 8$, and 10, are given in Table 3.4. All these WOS filters have been designed by using the nonadaptive approach. As can be seen from Figure 3.4, the L_{10} WOS filter, which is actually the minimum filter (see Table 3.4), provides a better visual performance compared to that provided by the conventional MAE WOS filter, as well as compared to the visual performances provided by the L_5 and L_8 WOS filters. Finally, it should also be observed that, as indicated by the results in Tables 3.2 and 3.3, a MAE WOS filter which is optimal in the sense of the $J_1(\mathbf{w})$ error but only approximately optimal in the MAE sense, can actually achieve MAE-values that are larger than the MAE's provided by the L_p WOS filters with $p \geq 2$.

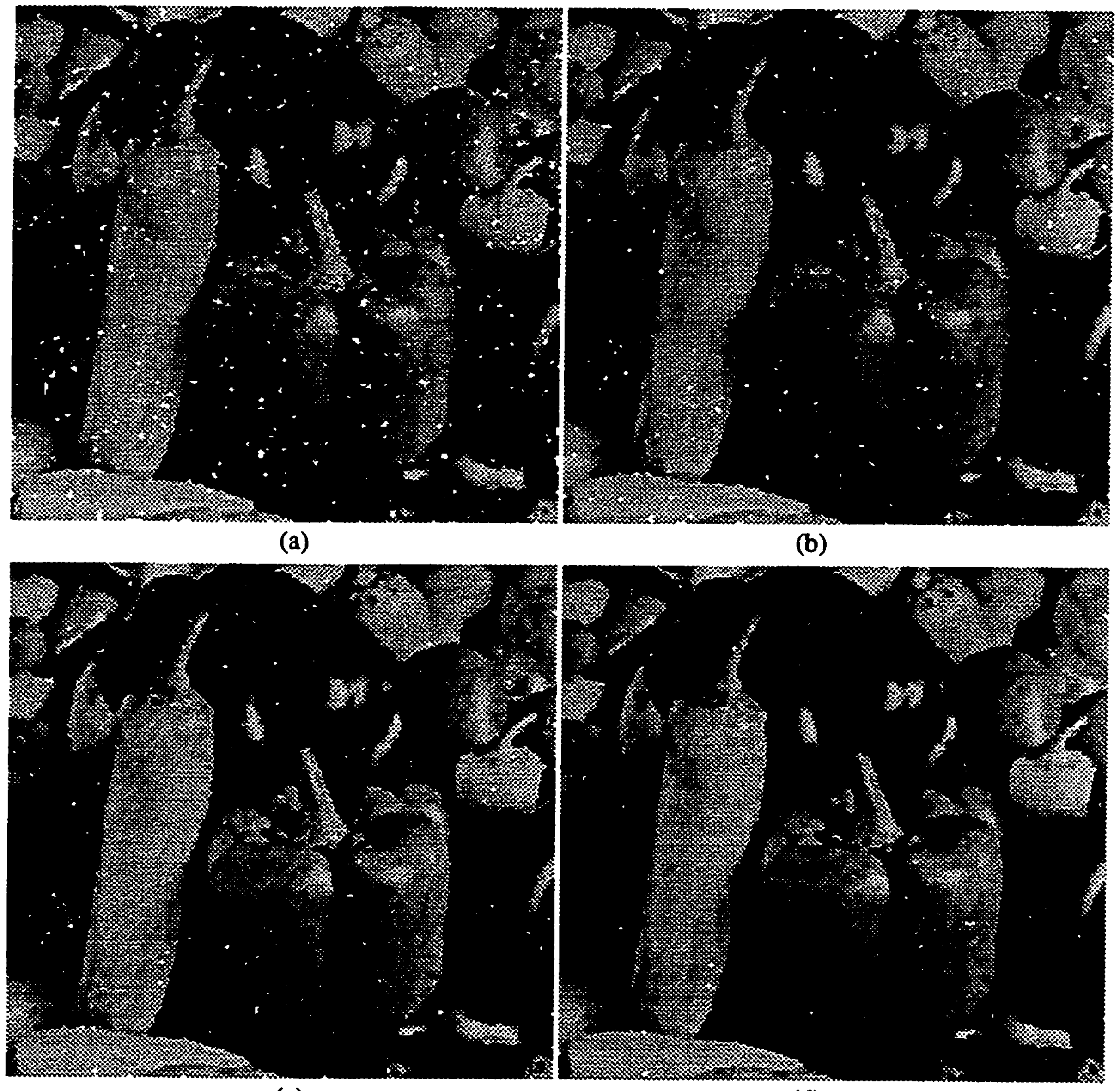
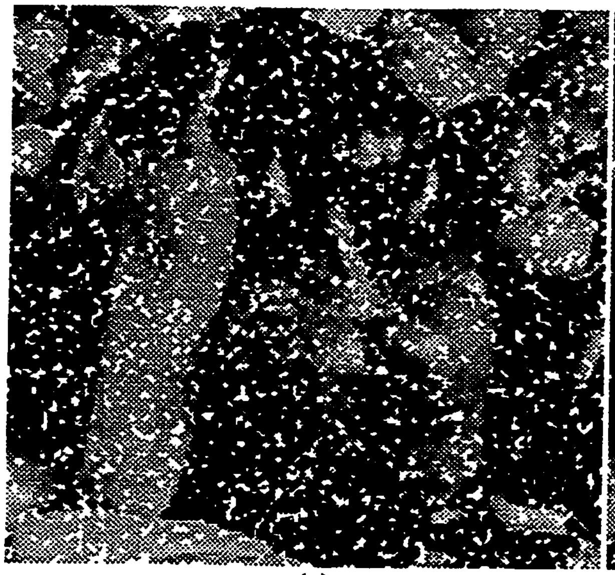
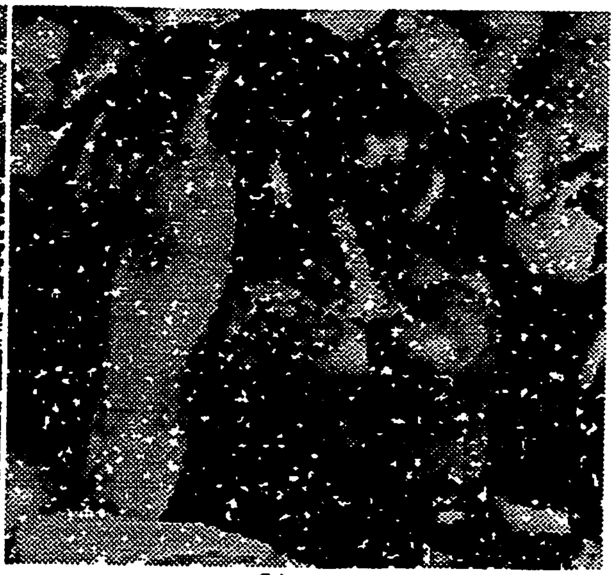
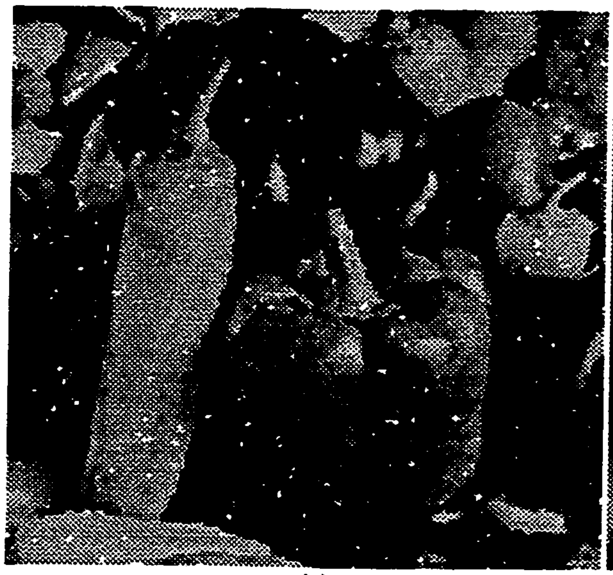

(a) (b) (c) (d)

Figure 3.3: Processed images applying optimal 3×3 stack filters to the noise-degraded image of Figure 3.2(b). The stack filters have been designed by using (a) MAE, (b) MSE, (c) L_3, and (d) L_4 criteria.


(a)


(b)


(c)

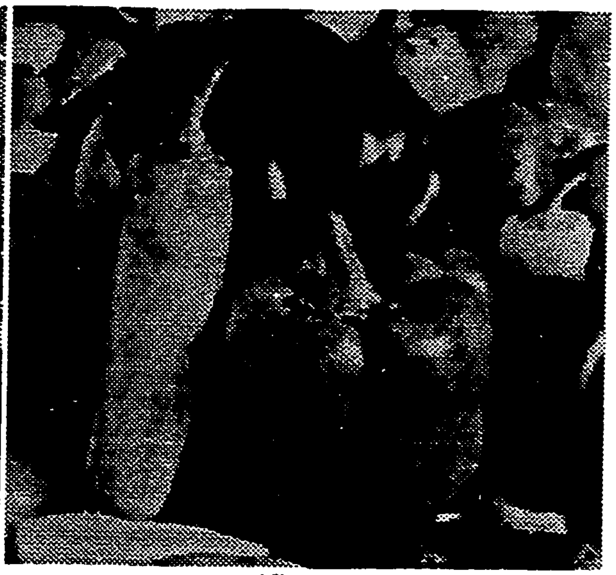

(d)

Figure 3.4: Processed images applying approximately optimal 3×3 WOS filters to the noise-degraded image of Figure 3.2(b). The WOS filters have been designed by using (a) MAE, (b) L_5, (c) L_8, and (d) L_{10} criteria.

Filter	MAE	L_2 Error	L_3 Error	L_4 Error	RERIN[%]
MMAE stack	7.79	23.17	42.71	60.95	—
MMSE stack	8.41	20.62	36.57	52.42	58.38
L_3-optimal stack	10.01	21.33	35.67	50.02	78.80
L_4-optimal stack	12.11	23.56	36.79	49.24	94.44

Table 3.1: Errors in the restored images using the L_p norm stack filters designed in Experiment I.

Filter	MAE	L_5 Error	L_8 Error	L_{10} Error	RERIN[%]
MAE WOS	21.12	111.35	141.47	154.79	—
L_5 WOS	12.17	92.57	125.74	140.72	65.91
L_8 WOS	8.58	68.90	103.10	119.64	92.90
L_{10} WOS	12.11	60.33	86.73	100.48	99.14

Table 3.2: Errors in the restored images using the L_p norm WOS filters designed in Experiment I: MAE, L_5, L_8, and L_{10} errors, and RERIN efficiency.

Filter	$J_1(\mathbf{w})$	$\sqrt[5]{J_5(\mathbf{w})}$	$\sqrt[8]{J_8(\mathbf{w})}$	$\sqrt[10]{J_{10}(\mathbf{w})}$
MAE WOS	26.88	118.20	145.94	158.36
L_5 WOS	27.82	117.45	144.44	156.60
L_8 WOS	31.35	118.53	143.72	155.14
L_{10} WOS	36.26	120.85	144.19	154.75

Table 3.3: Errors in the restored images using the L_p norm WOS filters designed in Experiment I: $J_1(\mathbf{w})$, $\sqrt[5]{J_5(\mathbf{w})}$, $\sqrt[8]{J_8(\mathbf{w})}$, and $\sqrt[10]{J_{10}(\mathbf{w})}$ errors.

MAE WOS Filter	L_5 WOS Filter
$\mathbf{w}: \begin{pmatrix} 0.0902 & 0.1073 & 0.0940 \\ 0.1149 & 0.1906 & 0.1127 \\ 0.0954 & 0.1070 & 0.0881 \end{pmatrix}$ $w_T: 0.6334$	$\mathbf{w}: \begin{pmatrix} 0.1104 & 0.1112 & 0.1101 \\ 0.1140 & 0.1151 & 0.1110 \\ 0.1119 & 0.1101 & 0.1063 \end{pmatrix}$ $w_T: 0.6972$
L_8 WOS Filter	**L_{10} WOS Filter**
$\mathbf{w}: \begin{pmatrix} 0.1119 & 0.1117 & 0.1106 \\ 0.1141 & 0.1143 & 0.1095 \\ 0.1130 & 0.1103 & 0.1047 \end{pmatrix}$ $w_T: 0.8077$	$\mathbf{w}: \begin{pmatrix} 0.1125 & 0.1125 & 0.1105 \\ 0.1144 & 0.1146 & 0.1090 \\ 0.1132 & 0.1105 & 0.1028 \end{pmatrix}$ $w_T: 0.9256$

Table 3.4: Weights of the L_p WOS filters ($p = 1, 5, 8, 10$) designed in Experiment I.

The performance of the L_p stack and the WOS filters with $p \geq 2$ designed in this experimental investigation, has been also evaluated in terms of a heuristic measure of their efficiency to remove impulsive noise, compared to those achieved by the conventional MAE stack and WOS filters. Let us denote the number of residual impulses which are left in a processed image obtained by applying a stack filter $\mathcal{S}_f$ to a given noise-corrupted image by $\mathrm{NRI}(\mathcal{S}_f)$. In the special case of applying the MAE stack filter $\mathrm{MAE}\,\mathcal{S}_f$ to the given noise-corrupted image, the number of residual impulses is denoted by $\mathrm{NRI}(\mathrm{MAE}\,\mathcal{S}_f)$. A measure of the efficiency of an L_p stack filter $\mathcal{S}_f$, relative to that of the filter $\mathrm{MAE}\,\mathcal{S}_f$, in removing the impulses that appear in the given noise-corrupted image, can be defined as

$$\mathrm{RERIN}(\mathcal{S}_f)[\%] \triangleq \frac{\mathrm{NRI}(\mathrm{MAE}\,\mathcal{S}_f) - \mathrm{NRI}(\mathcal{S}_f)}{\mathrm{NRI}(\mathrm{MAE}\,\mathcal{S}_f)} \times 100\,, \tag{3.69}$$

where RERIN stands for "relative efficiency in removing impulsive noise". The RERIN efficiencies of the L_2-, L_3-, and L_4-optimal stack filters designed in this experiment. are given in the last column of Table 3.1. Similarly, the values of the RERIN efficiencies (relative to the efficiency of the MAE WOS filter) of the approximately optimal L_p WOS filters with $p = 5, 8$, and 10, are provided in the last column of Table 3.2.

Experiment II

A 256×256 8-bit Lenna image, shown in Figure 3.5(a), has been used in the second experiment. The noise-corrupted image of Figure 3.5(b) has been generated by adding "salt-and-pepper" impulsive noise to the noise-free image of Figure 3.5(a). The impulsive noise has a probability of ocurrence of 0.2, a magnitude of ± 255, and the positive and negative impulses appear with equal probabilities.

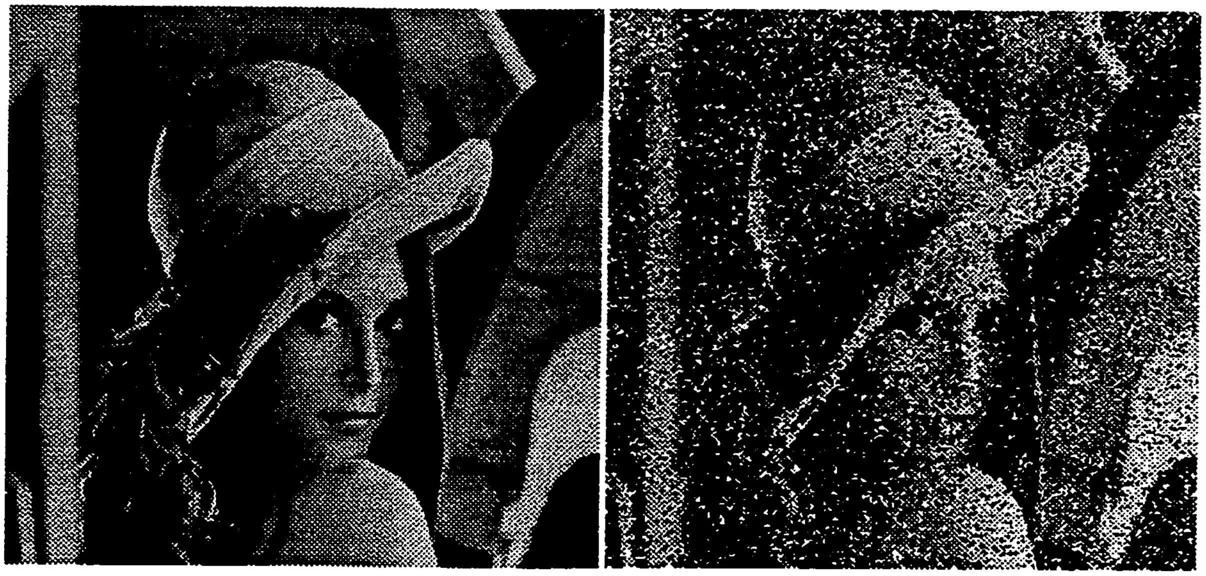
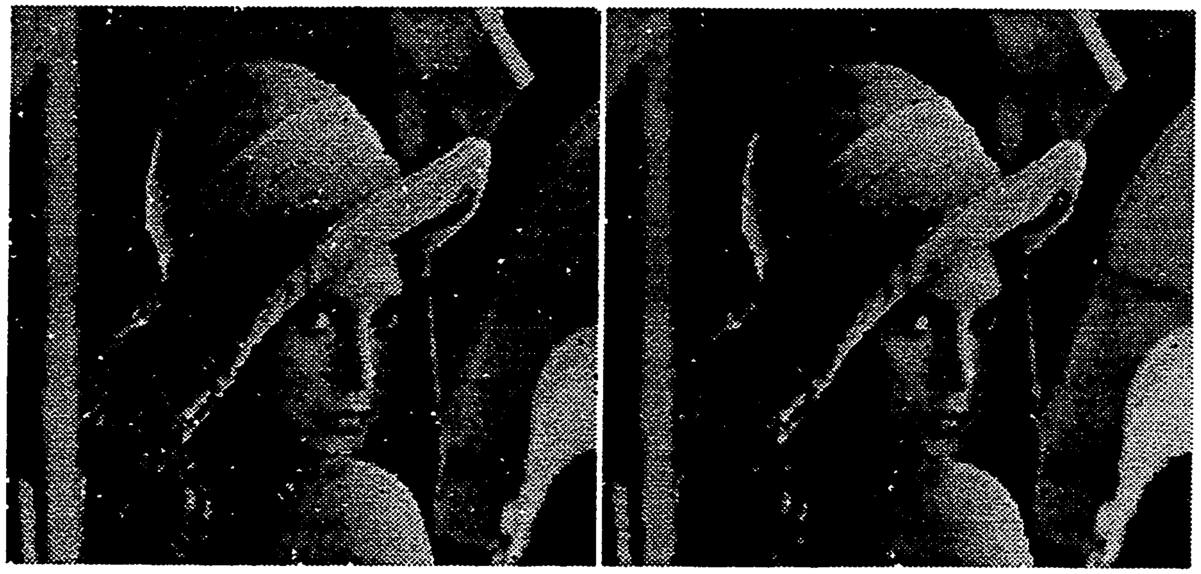


Figure 3.5: A 256 × 256 8-bit Lenna image. (a) The original image. (b) Noise-corrupted version of (a).

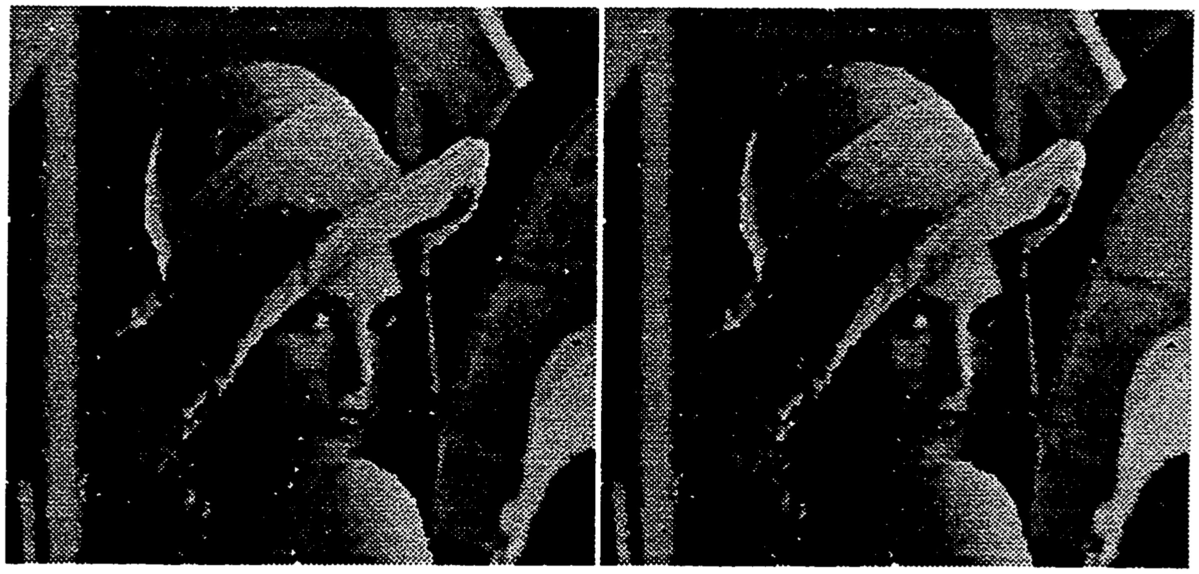
Figures 3.6(a) and (b) show, respectively, the processed images obtained by applying the MMAE 3×3 stack filter and the 3×3 median filter, to the degraded image of Figure 3.5(b). Note here that the median filter has been applied solely on the basis of the heuristical observation that, in this experiment, the impulsive-noise is approximately symmetrical (i.e., the positive and negative impulses have equal probabilities of occurrence). However, as can be observed from Figure 3.6, in this example, the median filter itself is capable of removing the impulsive noise more effectively compared to the MAE-optimal stack filter. By employing higher-order error objective functions, it has been observed that in this experiment, the results of median filtering are actually very close to those of the L_p-optimal stack filtering with $p \geq 4$, both in terms of the visual performance as well as in terms of the L_p-errors. The processed images obtained by applying the L_4- and L_8-optimal stack filters of size 3×3 to the degraded image of Figure 3.5(b) are shown in Figure 3.7. The values of the L_p errors with $p = 1, 4$, and 8, of the restored images of Figures 3.6 and 3.7 are listed in Table 3.5, together with the RERIN efficiencies of the corresponding filters.

Figures 3.8 and 3.9 illustrate the processed images by applying the MAE WOS, median, L_4 WOS, and L_8 WOS filters of size 3×3 to the noise-corrupted image of Figure 3.5(b). The values of the L_p errors with $p = 1, 4$, and 8, of the restored images of Figures 3.8 and 3.9 are listed in Table 3.6, while the corresponding values of $\sqrt[p]{J_p(\mathbf{w})}$ are given in Table 3.7. The weights and threshold values of the normalized median filter and L_p WOS filters with $p = 1, 4$, and 8, are given in Table 3.8. As observed from Figures 3.8 and 3.9, and from the results of Tables 3.6—3.8, the L_4 and L_8 WOS filters of this experiment are very close to the median itself.



(a) (b)

Figure 3.6: Processed images applying (**a**) MAE-optimal stack and (**b**) median filters of size 3 × 3, to the degraded image of Figure 3.5(b).



(a) (b)

Figure 3.7: Processed images applying (**a**) L_4- and (**b**) L_8-optimal stack filters of size 3 × 3, to the degraded image of Figure 3.5(b).

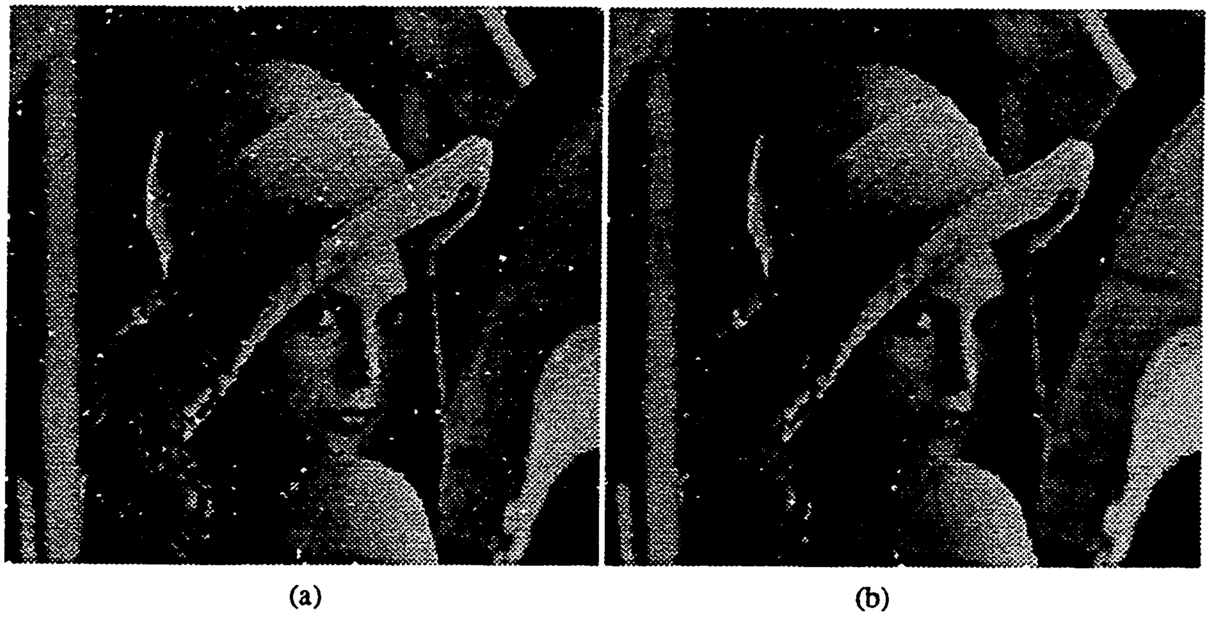


Figure 3.8: Processed images applying (a) MAE WOS and (b) median filters of size 3×3, to the degraded image of Figure 3.5(b).

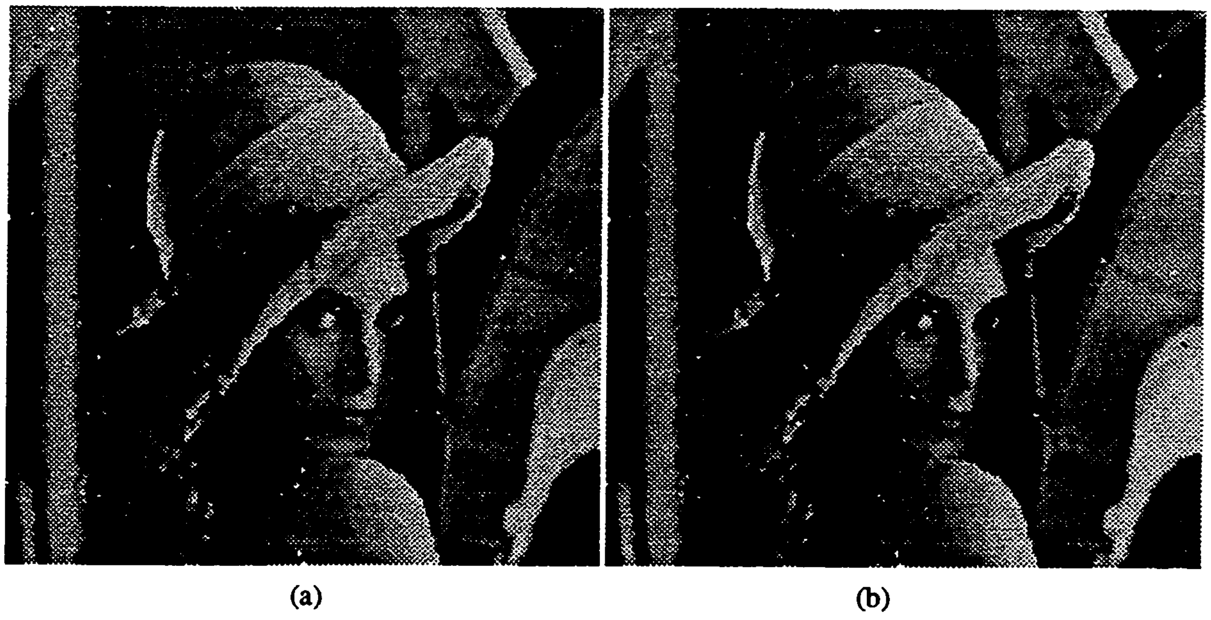


Figure 3.9: Processed images applying the approximately optimal (a) L_4 and (b) L_8 WOS filters of size 3×3, to the degraded image of Figure 3.5(b).

Filter	MAE	L_4 Error	L_8 Error	RERIN[%]
MMAE stack	4.92	43.77	88.92	—
Median	6.13	34.15	76.39	81.84
L_4-optimal stack	6.11	33.37	74.12	80.27
L_8-optimal stack	6.37	33.57	73.07	81.84

Table 3.5: Errors in the restored images using the L_p norm stack filters designed in Experiment II.

Filter	MAE	L_4 Error	L_8 Error	RERIN[%]
MAE WOS	5.31	42.46	87.89	—
Median	6.13	34.15	76.39	77.25
L_4 WOS	6.13	34.15	76.39	77.25
L_8 WOS	6.13	34.15	76.39	77.25

Table 3.6: Errors in the restored images using the L_p norm WOS filters designed in Experiment II: MAE, L_4, and L_8 errors, and RERIN efficiency.

Filter	$J_1(\mathbf{w})$	$\sqrt[4]{J_4(\mathbf{w})}$	$\sqrt[8]{J_8(\mathbf{w})}$
MAE WOS	8.12	66.733	111.258
Median	11.55	73.264	115.654
L_4 WOS	8.62	65.257	109.907
L_8 WOS	8.67	65.275	109.893

Table 3.7: Errors in the restored images using the L_p norm WOS filters designed in Experiment II: $J_1(\mathbf{w})$, $\sqrt[4]{J_4(\mathbf{w})}$, and $\sqrt[8]{J_8(\mathbf{w})}$ errors.

MAE WOS Filter	Median Filter
$\mathbf{w}: \begin{pmatrix} 0.0840 & 0.0909 & 0.0946 \\ 0.1157 & 0.2323 & 0.1154 \\ 0.0941 & 0.0901 & 0.0830 \end{pmatrix}$ $w_T: 0.4677$	$\mathbf{w}: \begin{pmatrix} 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \end{pmatrix}$ $w_T: 0.5555$
L_4 WOS Filter	L_8 WOS Filter
$\mathbf{w}: \begin{pmatrix} 0.1102 & 0.1113 & 0.1118 \\ 0.1125 & 0.1135 & 0.1131 \\ 0.1101 & 0.1096 & 0.1079 \end{pmatrix}$ $w_T: 0.4686$	$\mathbf{w}: \begin{pmatrix} 0.1139 & 0.1137 & 0.1139 \\ 0.1101 & 0.1091 & 0.1113 \\ 0.1093 & 0.1103 & 0.1083 \end{pmatrix}$ $w_T: 0.4713$

Table 3.8: Weights of the median and of the L_p-WOS filters ($p = 1, 4, 8$) designed in Experiment II.

3.7 Summary

In this chapter, the problem of designing optimal stack filters by employing an L_p norm of the error between the desired signal and the estimated one has been solved. It has been shown, for the first time, that in stack filtering, the pth order error between the desired and estimated signals can be expressed as a linear function of the decision errors incurred by the Boolean operators at each level of the filter. Therefore, an L_p-optimal stack filter can be determined as the solution of a linear program. Based on the derived expression of the L_p norm, a rigorous formulation of the problem of minimax design of stack filters has also been developed. The conventional MAE design of WOS filters has been extended to the L_p norm-based design. It should also be pointed out that the proposed L_p norm-based framework of designing stack filters is not restricted to the case of using the same Boolean operator at all the binary levels of the filter. Specifically, the results stated in Theorems 3.1 and 3.2 are valid in the more general case, when the Boolean operators are allowed to change with each threshold level (i.e., for the class of so-called "generalized stack filters", defined in [18] and briefly presented in Section 2.4). Notwithstanding the main contribution of this chapter as a significant analytical result, providing a new insight into the theory of (generalized) stack filter design, the framework of L_p norm design of stack filters, developed in this chapter, can find immediate applications in solving practical engineering problems. Specifically, it has been shown that in applications involving restoration of images corrupted with impulsive noise, the L_p norm stack filters are capable of removing impulsive noise much more effectively, and providing a better visual performance, compared to that provided by the conventional MAE stack filters.

Chapter 4

A Bit-Serial Window-Partitioning Algorithm for Stack Filtering

As discussed in Chapter 2, in the BTS approach for rank-order-based filtering, the level adjustment is carried out based on the result of the operation of Boolean filtering characterized by a positive Boolean function f, and by using increments which are powers of 2 (see (2.28)-(2.29)). Consequently, the bit-serial BTS architecture of [13] (see Figure 2.8) does not take advantage of the additional information associated with the binary threshold sequence d_{0J}, d_{1J}, ..., $d_{(M-1)J}$. Specifically, the binary values d_{iJ}'s can be used to divide the window-sequence $\{x_0, x_1, \ldots, x_{M-1}\}$ into two subsets, namely, $\mathcal{A}_J = \{\, x_i \,|\, x_i \geq \ell_J \,,\ i = 0, 1, \ldots, M-1 \,\}$, and $\mathcal{B}_J = \{\, x_i \,|\, x_i < \ell_J \,,\ i = 0, 1, \ldots, M-1 \,\}$. Thus, the subset $\mathcal{A}_J$ consists of those window samples which are greater than or equal to ℓ_J, while the subset $\mathcal{B}_J$ comprises the window samples which are smaller than ℓ_J. In view of the stacking property of f, a binary output f=1 at level ℓ_J indicates that the multilevel output of the stack filter $\mathcal{S}_f$ is equal to one of the elements of $\mathcal{A}_J$, and therefore, level ℓ_{J+1} should be selected from the elements of $\mathcal{A}_J$. Similarly, a binary output f=0 at level ℓ_J indicates that the multilevel output is equal to one of the elements of $\mathcal{B}_J$, and as such, ℓ_{J+1} should be selected from these elements. Based on this observation, in this chapter, a new and efficient bit-serial hardware-oriented algorithm for stack filtering is developed

[3], [74], [78]. In the proposed algorithm, at each iteration, the new threshold level ℓ_J is obtained by using both the result of the operation of Boolean filtering, as well as the binary threshold sequence corresponding to the previous level, ℓ_{J-1}. An analysis of the computational complexity of the proposed algorithm is carried out, and an architecture suitable for its VLSI implementation is also developed.

4.1 Proposed Algorithm

The proposed technique for carrying out the operation of level adjustment is illustrated by the block diagram of Figure 4.1. Specifically, the threshold level ℓ_J is adjusted by employing both the result of the operation of Boolean filtering and the binary threshold sequence d_{0J}, d_{1J}, ..., $d_{(M-1)J}$, in such a way that each threshold level ℓ_J is equal to one of the samples $x_0, x_1, \ldots, x_{M-1}$.

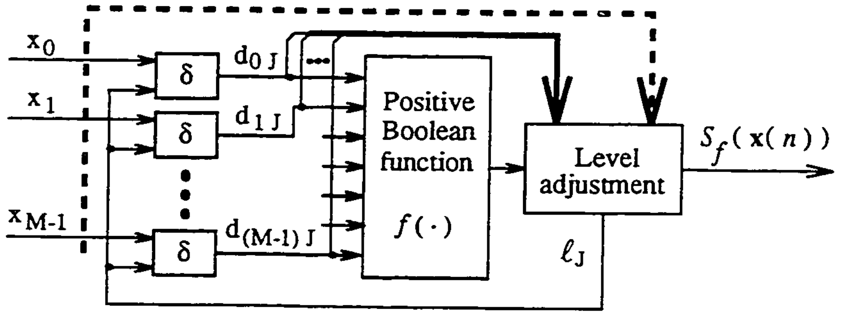


Figure 4.1: Block diagram of the proposed bit-serial technique for stack filtering.

The proposed level-adjustment technique associates a Boolean variable B_i to each window sample x_i, such that $B_i = 1$ as long as, in order to determine the multilevel output of the stack filter S_f, it is still necessary to carry out the operations of thresholding and Boolean filtering at a level equal to x_i. The variable B_i is otherwise set to 0. At the beginning of the proposed procedure, all variables B_i, $i = 0, 1, \ldots, M-1$, are set to 1. Then, at iteration J, a variable B_i is reset to 0 if one of the following two situations occurs:

1) If at level ℓ_J we have

$$f_J \triangleq f(\delta(x_0 - \ell_J), \delta(x_1 - \ell_J), \ldots, \delta(x_{M-1} - \ell_J)) = 0 ,$$

on the basis of the stacking property of f, it follows that $f(\delta(x_0-x_i), \delta(x_1-x_i), \ldots, \delta(x_{M-1} -x_i)) = 0$ for any $x_i \geq \ell_J$. Therefore, any further operation of thresholding and Boolean filtering at a level $x_i \geq \ell_J$ is redundant.

2) If at level ℓ_J we have $f_J = 1$, on the basis of the stacking property of f, it follows that $f(\delta(x_0 - x_i), \delta(x_1 - x_i), \ldots, \delta(x_{M-1} - x_i)) = 1$ for any $x_i \leq \ell_J$. Therefore, again any further operation of thresholding and Boolean filtering at a level $x_i \leq \ell_J$ is redundant.

Thus, at iteration J, if one of the following two situations occurs: a) $x_i \geq \ell_J$ and $f_J = 0$ or b) $x_i \leq \ell_J$ and $f_J = 1$, B_i is reset to 0. On the other hand, if c) $x_i < \ell_J$ and $f_J = 0$ or d) $x_i > \ell_J$ and $f_J = 1$, B_i remains unchanged. The steps to be followed in the adjustment of threshold levels as used in the proposed bit-serial algorithm for stack filtering can be stated as follows.

1. Initialize all B_i's to 1 and set $J = 0$.

2. If $B_J = 1$, evaluate the PBF at the level $\ell_J = x_J$.

  - A result $f_J = 1$ means that either $\mathcal{S}_f > \ell_J$, or $\mathcal{S}_f = \ell_J$. Consequently, any $x_i < \ell_J$ (i.e., any x_i for which $d_{iJ} = 0$) cannot constitute the output of $\mathcal{S}_f$, and therefore set $B_i = 0$ for all such x_i's. Also, in order to avoid any further operation of thresholding and Boolean filtering at level ℓ_J, set $B_i = 0$ for all x_i's that are equal to ℓ_J.
  - A result $f_J = 0$ means that $\mathcal{S}_f < \ell_J$. Consequently, any $x_i \geq \ell_J$ (i.e., any x_i for which $d_{iJ} = 1$) cannot constitute the output of $\mathcal{S}_f$, and therefore set $B_i = 0$ for all such x_i's.

3. Set $J = J + 1$. If $J \leq M - 1$, go to Step 2; otherwise stop.

The principle used in the above procedure is to divide the input vector $\mathbf{x} = (x_0, \ldots, x_{M-1})$ into two parts by employing the threshold sequence $\mathbf{d} \triangleq \{d_{iJ} | i = 0, 1, \ldots, M-1\}$, and then select one part, according to the result of applying the Boolean operator f, as the work-interval for the next iteration of thresholding and Boolean filtering. At each iteration J, the work-interval consists of those window-samples x_i's for which $B_i = 1$.

In order to synthesize the expression of the Boolean variable B_i, we observe that corresponding to each operation of thresholding at a level ℓ_J, each pair (x_i, f_J) can assume one of the four possible states, as shown in Table 4.1. The four possible ordering relations between x_i and ℓ_J (i.e., "<", "≤", ">", and "≥") are indicated by the threshold signals d_{iJ} and d'_{iJ}, where

$$d_{iJ} \triangleq \delta(x_i - \ell_J) = \begin{cases} 1 & \text{if} \quad x_i \geq \ell_J \\ 0 & \text{if} \quad x_i < \ell_J \, , \end{cases} \tag{4.1}$$

and

$$d'_{iJ} \triangleq \delta'(x_i - \ell_J) = \begin{cases} 1 & \text{if} \quad x_i > \ell_J \\ 0 & \text{if} \quad x_i \leq \ell_J \, . \end{cases} \tag{4.2}$$

d_{iJ}, d'_{iJ}	f_J	B_{iJ}
$d_{iJ} = 0$ $(x_i < \ell_J)$	0	$B_{i(J-1)}$
$d_{iJ} = 1$ $(x_i \geq \ell_J)$	0	0
$d'_{iJ} = 0$ $(x_i \leq \ell_J)$	1	0
$d'_{iJ} = 1$ $(x_i > \ell_J)$	1	$B_{i(J-1)}$

Table 4.1: Syntheses of the expression of the binary variable B_i.

Using Table 4.1, it can be easily observed that B_i can be expressed as

$$B_{iJ} = (\bar{f}_J \, \bar{d}_{iJ} + f_J \, d'_{iJ}) \cdot B_{i(J-1)} \, . \tag{4.3}$$

Now, the main algorithm for bit-serial stack filtering based on the above procedure can be formalized using the pseudocode conventions of [14] as given next.

Algorithm 4.1 *Bit-Serial Window-Partitioning (BSWP) Algorithm for Stack Filtering*

```
1. $\mathbf{B} \leftarrow$ ones(M)
2. for J $\leftarrow$ 0 to M-1
3.     if $\mathrm{B_J} = 1$
4.         then $\mathbf{d} \leftarrow$ COMP-GEQ($\mathbf{x}$, $\mathbf{x}_\mathrm{J}$)
5.              $\mathbf{d}' \leftarrow$ COMP-G($\mathbf{x}$, $\mathbf{x}_\mathrm{J}$)
6.              $f \leftarrow$ PBF($\mathbf{d}$)
7.              if $f$=1
8.                  then $\mathcal{S}_f \leftarrow \ell_\mathrm{J}$
9.              for i $\leftarrow$ 0 to M-1
10.                 $\mathbf{B}_i \leftarrow (\bar{f}\,\bar{\mathrm{d}}_i + f\,\mathrm{d}'_i) \cdot \mathbf{B}_i$
11. return $\mathcal{S}_f$
```

A listing of a Matlab program implementing the BSWP algorithm is given in Appendix D.

The proposed algorithm uses two procedures designated as COMP-GEQ and COMP-G to determine the threshold sequences $\mathbf{d} = \{\mathrm{d_{0J}}, \mathrm{d_{1J}}, \ldots, \mathrm{d_{(M-1)J}}\}$ and $\mathbf{d}' = \{\mathrm{d'_{0J}}, \mathrm{d'_{1J}}, \ldots, \mathrm{d'_{(M-1)J}}\}$ corresponding to a level ℓ_J of the stack filter configuration. The procedure called PBF($\mathbf{d}$) evaluates the output of the Boolean filter f corresponding to the threshold sequence $\mathbf{d}$.

At the beginning of an operation of stack filtering, any window sample can be selected as the output $\mathcal{S}_f$. Therefore, at line 1 of Algorithm 4.1, all the entries of the partition vector $\mathbf{B} = \{\mathrm{B}_i \,|\, i = 0, 1, \ldots, \mathrm{M}-1\}$ are set to 1. When the condition at line 3 of the algorithm is satisfied, as it is always the case for $\mathrm{J} = 0$, an operation of Boolean filtering $f = \mathrm{PBF}(\mathbf{d})$ is applied to the threshold sequence $\mathbf{d}$. A binary result of $f = 1$ indicates that either $\mathcal{S}_f > \ell_\mathrm{J}$ (in which case the output of $\mathcal{S}_f$ is found among the elements x_i for which $\mathrm{B}_{i\mathrm{J}} = 1$) or $\mathcal{S}_f = \ell_\mathrm{J}$. Therefore, at line 8 of the algorithm, $\mathcal{S}_f$ is set to ℓ_J. At line 10, the new entries of the partition vector $\mathbf{B}$ are calculated using (4.3). Thus, B_i's are adjusted according to the results of the operations of thresholding and Boolean filtering.

4.2 An Architecture Suitable for Hardware Implementation of the Proposed BSWP Algorithm

An architecture (designated as the BSWP architecture) suitable for hardware implementation of Algorithm 4.1 is illustrated in Figure 4.2. The datapath consists of M identical processing units (PU's), each comprising a register-block REG and a comparator COMP. A register-block REG consists of two register-fields denoted as B and X. A sample value x_i appearing in the input window of a stack filter is loaded into the register-field X of REG_i, while the register B of REG_i is used to represent the partition flag corresponding to x_i.

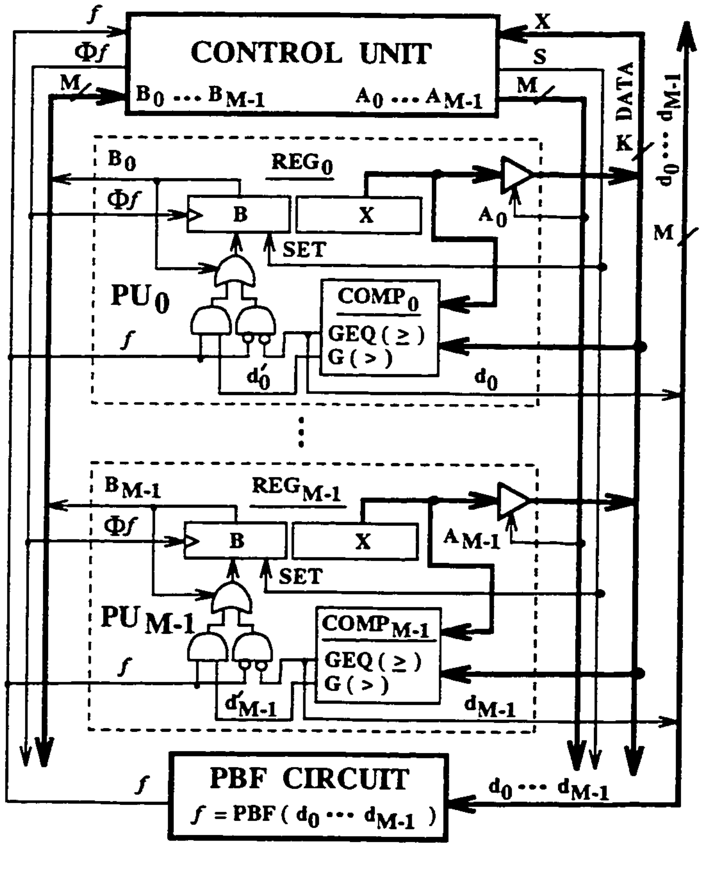


Figure 4.2: An architecture suitable for VLSI implementation of the bit-serial window-partitioning algorithm.

The structure of the control unit is shown in Figure 4.3. It consists of a counter, a decoder, a shift-register, and an output register. When the counter is incremented to a value J, the output A_J of the decoder is set to 1, and all A_j's with $j \neq J$ are set to 0. Accordingly, the output of X(REG_J) is placed on the data-bus, and each comparator $COMP_i$, $i = 0, 1, \ldots, M-1$, compares the content of the register-field X(REG_i) with the value stored in X(REG_J). This operation of parallel comparison realizes both the procedures COMP-GEQ and COMP-G that are used at the line 4 of Algorithm 4.1.

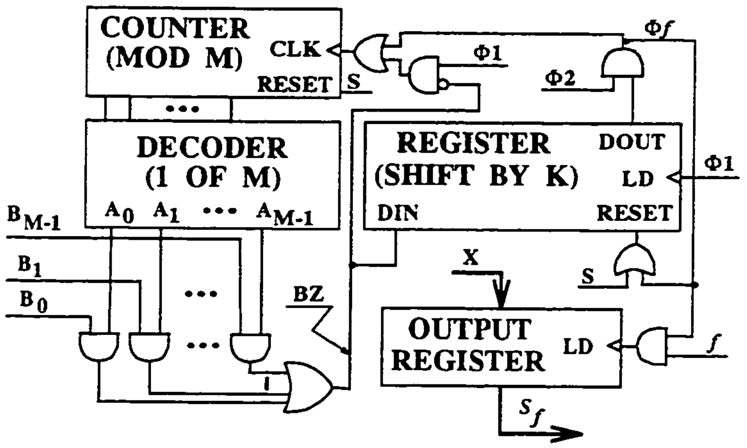


Figure 4.3: Structure of the control unit of the BSWP architecture of Figure 4.2.

In the BSWP algorithm, each new threshold level ℓ_J is equal to one of the window samples x_i's for which $B_i = 1$. In the BSWP architecture given in Figures 4.2 and 4.3, the selection of the threshold level ℓ_J is achieved by employing the variable $BZ = \bigcup_{i=0}^{M-1} A_i B_i$. Specifically, the counter of Figure 4.2 is incremented with each rising edge of the main clock Φ1, as long as the variable BZ asumes a value of 0. On the other hand, when BZ = 1, the content of the counter remains unchanged for the duration (T_{PBF}) required to complete an operation of thresholding and Boolean filtering. This operation when BZ = 1 is accomplished by using the shift-register of Figure 4.3, whose size K is such that $K \cdot T > T_{PBF}$, where T denotes the period of Φ1.

An illustration of the waveforms associated with the operation of the architecture of Figure 4.2 is given in Figure 4.4. In this illustration, it is assumed that the PBF has to be evaluated for J = 0, 1, and 4, and that the size K of the shift-register of Figure 4.3 is equal to 2.

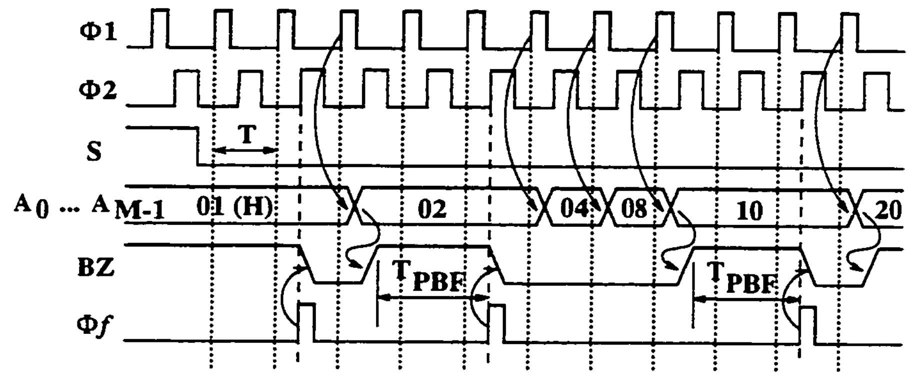


Figure 4.4: Waveforms associated with the operation of the BSWP architecture for stack filtering.

4.3 Complexity Analysis of the BSWP Algorithm

The proposed BSWP algorithm for stack filtering uses a divide-and-conquer strategy. In this strategy, the work interval at the iteration J is partitioned into two subsets $\mathcal{A}_J$ and $\mathcal{B}_J$, such that each element of $\mathcal{A}_J$ is smaller than each element of $\mathcal{B}_J$. Thus, from the point of view of the complexity analysis, this partitioning technique is similar to the one employed by the algorithm Quicksort [14]. As a result, the running-time properties of the proposed BSWP algorithm for stack filtering are similar to those of the Quicksort algorithm. As it is known, in spite of its slow worst-case running time, the Quicksort algorithm is generally the best practical choice for sorting because of its remarkably fast average-case running time. It has been shown in [14], that the partitioning technique used by the Quicksort algorithm leads to an average-case running time which is much closer to the best-case than to the worst-case running time.

The worst-case running time of the BSWP algorithm in terms of the number of iterations per output sample is equal to M. This worst-case scenario is elicited only when the input sequence $\mathbf{x}$ is arranged to some specific order for a specific rank. On the other hand, the best-case scenario corresponds to the case when the partitioning procedure produces two subsets of equal size at each iteration. In this best-case partitioning, the number of iterations per output sample is equal to $\log_2 \mathrm{M}$. In the subsequent, following a procedure similar to the one given in [14], we show that the partitioning technique used by the BSWP algorithm leads to an average-case running time of $\mathcal{O}(\log_2 \mathrm{M} + k)$, where the constant k is quite small.

Let us assume that the elements of $\mathbf{x}$ are in a random order. Also, to simplify the analysis, it is convenient to assume that all input elements are distinct numbers.[1] The probability of selecting the r-th largest element of a work interval with n elements as the threshold level ℓ_{J} is equal to $1/n$. As a result, the partitioning technique used by the BSWP algorithm produces a partition whose low side (i.e., $\mathcal{A}_{\mathrm{J}}$) has r elements with the probability of $1/n$ for any $r \in \{1, 2, \ldots, n-1\}$. Let $T(n)$ represent an upper bound on the expected number of iterations per output sample required by the BSWP algorithm for completing the operation of stack filtering when the work interval has n elements. Thus, $T(n)$ is a monotonically increasing function of n, and it can be determined as the solution of the following recurrence

$$T(n) \leq \frac{1}{n} \sum_{r=1}^{n-1} \max(T(r), T(n-r)) \,. \tag{4.4}$$

In (4.4), it is assumed that regardless of the rank r of ℓ_{J} among the n elements of the work interval at the iteration J, the work interval at the iteration (J + 1) is given by $\mathcal{A}_{\mathrm{J}}$ or $\mathcal{B}_{\mathrm{J}}$, depending on which one has the larger number of elements. In order to solve the recurrence (4.4), we observe that in view of the fact that $T(n)$ is a monotonically increasing function of n,

[1]For the case when $\mathbf{x}$ has non-distinct elements, the estimate of the average-case running time would be smaller.

$$\max(T(r), T(n-r)) = \begin{cases} T(r) & \text{if} \quad r \geq \lceil n/2 \rceil \\ T(n-r) & \text{if} \quad r < \lceil n/2 \rceil \, . \end{cases} \tag{4.5}$$

Thus, (4.4) becomes

$$T(n) \leq \frac{1}{n} \left(\sum_{r=1}^{\lceil n/2-1 \rceil} T(n-r) + \sum_{r=\lceil n/2 \rceil}^{n-1} T(r) \right) . \tag{4.6}$$

Now, if n is odd, each term $T(\lceil n/2 \rceil)$, $T(\lceil n/2+1 \rceil)$, ..., $T(n-1)$ appears twice on the right side of (4.6), and if n is even, each term $T(\lceil n/2 \rceil + 1)$, $T(\lceil n/2+2 \rceil)$, ..., $T(n-1)$ appears twice and the term $T(\lceil n/2 \rceil)$ appears only once. In either case, the right side of (4.6) is bounded from above by $\frac{2}{n}\sum_{r=\lceil n/2 \rceil}^{n-1} T(r)$, i.e.,

$$T(n) \leq \frac{2}{n} \sum_{r=\lceil n/2 \rceil}^{n-1} T(r) \, . \tag{4.7}$$

We solve the recurrence (4.7) by substitution. Assume that $T(n) \leq \log_2 n + k$ for any $n \in \{2, 3, \ldots, n-1\}$, and for some constant k that satisfies the initial condition of the recurrence (i.e., $k \geq T(2) - 1$). Using this inductive hypothesis, we have

$$\begin{aligned} T(n) &\leq \frac{2}{n} \sum_{r=\lceil n/2 \rceil}^{n-1} (\log_2 r + k) \\ &\leq \frac{2}{n} \sum_{r=\lceil n/2 \rceil}^{n-1} (\log_2 n + k) \;=\; \frac{2}{n}(n - \lceil \frac{n}{2} \rceil)(\log_2 n + k) \\ &\leq \frac{2}{n}(n - \frac{n}{2})(\log_2 n + k) \;=\; \log_2 n + k \, . \end{aligned} \tag{4.8}$$

Thus, using the proposed BSWP algorithm for carrying out an operation of stack filtering of window size M, the average number of iterations per output sample is $\leq \log_2 \mathrm{M} + k$.

In order to illustrate that the constant k assumes a small value, an experimental investigation is carried out by applying an operation of median filtering employing the proposed BSWP algorithm to the typical test image of Figure 4.5. In the experiment, the average number of iterations per output sample has been determined for median filters of sizes 3×3, 5×5 and 7×7, considering two different numbers of quantization levels, $\mathrm{L} = 64$ and $\mathrm{L} = 256$. The results are illustrated in Table 4.2, and compared to $\log_2 \mathrm{L}$, the

Figure 4.5: A test-image of size 256 × 256.

L	M	Average value of N/S			Savings
		BSWP		BTS	
		Exp.	$\log_2$ M	($\log_2$ L)	
L = 64	3 × 3	3.43	3.17	6	43%
	5 × 5	4.21	4.64	6	30%
	7 × 7	4.66	5.61	6	22%
L = 256	3 × 3	4.04	3.17	8	50%
	5 × 5	5.38	4.64	8	33%
	7 × 7	6.61	5.61	8	17%

Table 4.2: Experimental comparison of the execution times (in terms of the number of iterations per output sample, N/S) of the proposed BSWP algorithm with that of the BTS algorithm for stack filtering.

number of iterations per output sample required when the conventional BTS approach [13] is used. By comparing the *3rd* and *4th* columns of Table 4.2, it can be observed that a value of k=1 is sufficient when considering filters of sizes up to 7×7. Also, by comparing the *3rd* and *5th* columns of the table, it can be seen that, as expected, important improvements in terms of the computational speed can be obtained by using the proposed BSWP algorithm instead of the conventional BTS procedure for stack filtering.

The average number of iterations per output sample required by the proposed BSWP algorithm and architecture is $\mathcal{O}(\log_2 M)$. This shows that the BSWP approach is generally much faster than the BTS technique, which requires $\mathcal{O}(\log_2 L)$ iterations per output sample, since L is usually much larger than M. Yet, as seen by comparing the block diagrams of Figures 2.8 and 4.1, the chip-area of the BSWP architecture is comparable to that of the conventional BTS architecture[13].

It should be also observed that in applications where the number of quantization levels is much larger than $L = 256$, the proposed BSWP algorithm and architecture provide even more significant computational savings over the conventional BTS approach than those listed in Table 4.2. This is particulary the case in speech processing, where a 16-bit quantization scheme is generally employed.

4.4 Summary

In this chapter, a new algorithm designated as the bit-serial window-partitioning algorithm for stack filtering has been introduced. In this algorithm, at each iteration, the new threshold level ℓ_J is obtained by using both the result of the operation of Boolean filtering, as well as the binary threshold sequence corresponding to the previous level ℓ_{J-1}, in such a way that each level ℓ_J is equal to one of the samples appearing in the input window of the filter. With the BSWP algorithm, the average number of iterations per output sample is $\mathcal{O}(\log_2 M)$. It has been shown that the average-case number of iterations per output sample is much closer to the best-case number of iterations per output sample than to the worst-case.

An architecture suitable for the VLSI implementation of the BSWP algorithm has also been developed. It has been shown that the proposed architecture achieves an increased computational-speed over the BTS configuration for stack filtering, without an additional expense in terms of the chip-area.

Chapter 5

A New Compression-Based Algorithm and Architecture for 2-D Stack Filtering

The time-area complexities of the conventional parallel and bit-serial architectures for stack filtering are determined by the number of quantization levels L of the multilevel input signal. However, as recently observed in [1], more efficient architectures can be developed by employing an input compression technique, which reduces the number of threshold levels from L to M, the size of the filter's window. The input compression technique of [1] keeps track of the relative ranks of the samples appearing in the input window of the stack filter by mapping each sample-value to a different rank, even in the case when several samples assume non-distinct values. In spite of being very efficient for 1-D stack filtering applications, this input compression technique is less suitable for the implementation of 2-D stack filters. In image processing applications, it is desirable to take advantage of the fact that very often many pixels appearing in the filter-window at a certain time-instant assume non-distinct values. Let us denote the number of distinct grey level values of a sequence $\mathbf{x}(n)$ appearing in the input window of a 2-D stack filter by $\tilde{M}$. The fact that $\tilde{M}$ is generally smaller than the filter-size M can be exploited to reduce the computational complexity of stack filtering, if a suitable encoding of the elements of $\mathbf{x}(n)$ into elements of the set $\{0, 1, \ldots, \tilde{M} - 1\}$ can be performed. Such an encoding can be conveniently achieved by employing the window sequence coding (WSC) transformation introduced in this chapter [72]. Using the WSC transformation in conjunction with the BTS technique, a new algorithm for 2-D stack filtering is developed [77]. It is shown that the proposed algorithm is generally more efficient than the conventional input compression-based BTS algorithm [1].

5.1 The WSC Transformation

Let us construct a set $\mathbf{x}^*(n) = \{x_0^*, x_1^*, \ldots, x_{\tilde{M}-1}^*\}$ by taking each element of $\mathbf{x}(n)$ at its first occurrence, when the elements of $\mathbf{x}(n)$ are scanned through from left to right. Thus, $\mathbf{x}^*(n)$ consists of only the distinct elements of $\mathbf{x}(n)$. Now, using $\mathbf{x}^*(n)$, a new set denoted by $\tilde{\mathbf{x}}^*(n)$ is derived as

$$\tilde{\mathbf{x}}^*(n) = \{ \ \tilde{x}_i^* \ | \ \tilde{x}_i^* = [\sum_{j=0}^{\tilde{M}-1} \delta(x_i^* - x_j^*)] - 1 \ ; \ i = 0, 1, \ldots, \tilde{M} - 1 \ \} \ . \qquad (5.1)$$

Therefore, each element $\tilde{x}_i^* \in \{0, 1, \ldots, \tilde{M} - 1\}$ of $\tilde{\mathbf{x}}^*(n)$ is given by the total number of elements in $\mathbf{x}^*(n)$ which are smaller than x_i^*. Since both $\mathbf{x}^*(n)$ and $\tilde{\mathbf{x}}^*(n)$ consist of only distinct elements, a mapping

$$F_n : \mathbf{x}^*(n) \rightarrow \tilde{\mathbf{x}}^*(n) \ , \quad \text{with} \quad F_n[x_i^*] = \tilde{x}_i^* \ , \quad i = 0, 1, \ldots, \tilde{M} - 1 \ . \qquad (5.2)$$

is a one-to-one correspondence, i.e., its inverse exists and it is given by

$$F_n^{-1}[\tilde{x}_i^*] = x_i^* \ , \quad i = 0, 1, \ldots, \tilde{M} - 1 \ . \qquad (5.3)$$

The correspondence F_n, which maps the domain set $\mathbf{x}^*(n)$ onto the range set $\tilde{\mathbf{x}}^*(n)$ given by (5.1) is designated as a window-sequence coding (WSC) transformation. The WSC transformation given by (5.2) enjoys the following property.

Property 5.1 The ordering relations among the elements of the range set $\tilde{\mathbf{x}}^*(n)$ are the same as those among the elements of the domain set $\mathbf{x}^*(n)$, i.e.,

$$\tilde{\mathrm{x}}_i^* > \tilde{\mathrm{x}}_j^* \text{ if and only if } \mathrm{x}_i^* > \mathrm{x}_j^* \text{ for all pairs } (i,j), \text{ with } i,j \in \{0,1,\ldots,\tilde{\mathrm{M}}-1\}\,. \quad (5.4)$$

The proof of this property is straightforward, and it is omitted.

Example 5.1 As an example of constructing the sets $\mathbf{x}^*(n)$ and $\tilde{\mathbf{x}}^*(n)$, we consider the case of an input-window sequence $\mathbf{x}(n) = \{97, 255, 97, 93, 32, 97, 93, 93, 97\}$ with $\mathrm{M} = 9$. Following the procedure described above, we get $\mathbf{x}^*(n) = \{97, 255, 93, 32\}$, and $\tilde{\mathbf{x}}^*(n) = \{2, 3, 1, 0\}$. Therefore, in this case, the WSC transformation is given by

$$\mathrm{F}_n \; : \; \{97, 255, 93, 32\} \rightarrow \{2, 3, 1, 0\}, \quad (5.5)$$

that is, $\mathrm{F}_n[97] = 2$, $\mathrm{F}_n[255] = 3$, $\mathrm{F}_n[93] = 1$, and $\mathrm{F}_n[32] = 0$.

Using the WSC transformation, the elements of $\mathbf{x}^*(n)$ are encoded into the elements of the set $\{0, 1, \ldots, \tilde{\mathrm{M}}-1\}$, and the sequence $\mathbf{x}(n)$ is mapped into a sequence $\tilde{\mathbf{x}}(n)$ given by

$$\tilde{\mathbf{x}}(n) = \{\ \tilde{\mathrm{x}}_i \mid \tilde{\mathrm{x}}_i = \mathrm{F}_n[\mathrm{x}_i]\,;\, i = 0, 1, \ldots, \mathrm{M}-1\ \}\,. \quad (5.6)$$

In view of Property 5.1 and the fact that the element-values in $\mathbf{x}(n)$ are the same as those in $\mathbf{x}^*(n)$, it follows that the ordering relations among the elements of $\tilde{\mathbf{x}}(n)$ are the same as those among the elements of $\mathbf{x}(n)$, i.e.,

$$\tilde{\mathrm{x}}_i > \tilde{\mathrm{x}}_j \text{ if and only if } \mathrm{x}_i > \mathrm{x}_j\,, \quad \text{and} \quad \tilde{\mathrm{x}}_i = \tilde{\mathrm{x}}_j \text{ if and only if } \mathrm{x}_i = \mathrm{x}_j\,, \quad (5.7)$$

for all pairs (i,j) with $i,j \in \{0, 1, \ldots, \mathrm{M}-1\}$.

Example 5.2 In 2-D stack filtering, the 1-D set $\mathbf{x}(n)$ is normally obtained by concatenating the rows of the 2-D window sequence. If for a specific window $\mathbf{x}(n)$ in Figure 5.1, the WSC transformation of (5.5) is applied, the result, as shown, is given by $\tilde{\mathbf{x}}(n)$. The image

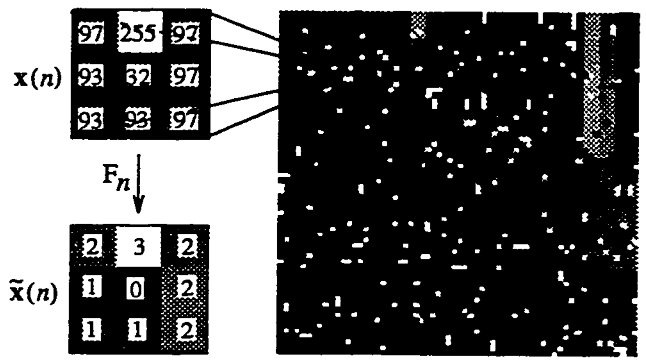


Figure 5.1: An example of applying the WSC transformation F_n to a 2-D sequence of size 3×3.

shown in Figure 5.1 represents a portion of the upper left corner of the noisy test-image Bank of Figure 5.9(b), with 256 grey levels. In this example, the number of iterations required for applying an operation of stack filtering to the encoded sequence $\tilde{\mathbf{x}}(n)$ by using the BTS algorithm is given by $\lceil \log_2 4 \rceil = 2$. If the same stack filtering technique is applied to a sequence obtained from $\mathbf{x}(n)$ by employing the input compression algorithm of [1], a total of $\lceil \log_2 9 \rceil = 4$ iterations are required, since $M = 3 \times 3 = 9$.

Let us now observe that, in practice, the WSC transformation can be applied without the need of explicitly constructing the correspondence F_n. Specifically, the sequence $\tilde{\mathbf{x}}(n)$ can be obtained directly from $\mathbf{x}^*(n)$ (i.e., by using the distinct elements of $\tilde{\mathbf{x}}(n)$), without constructing the set $\tilde{\mathbf{x}}^*(n)$, as

$$\tilde{\mathbf{x}}(n) = \{ \ \tilde{\mathbf{x}}_i \ \mid \ \tilde{\mathbf{x}}_i = [\sum_{j=0}^{\tilde{M}-1} \delta(\mathbf{x}_i - \mathbf{x}_j^*)] - 1 \ ; \ i = 0, 1, \ldots, M-1 \ \} \ . \tag{5.8}$$

This fact is now used for the development of an architecture which is suitable for an efficient hardware implementation of the WSC transformation.

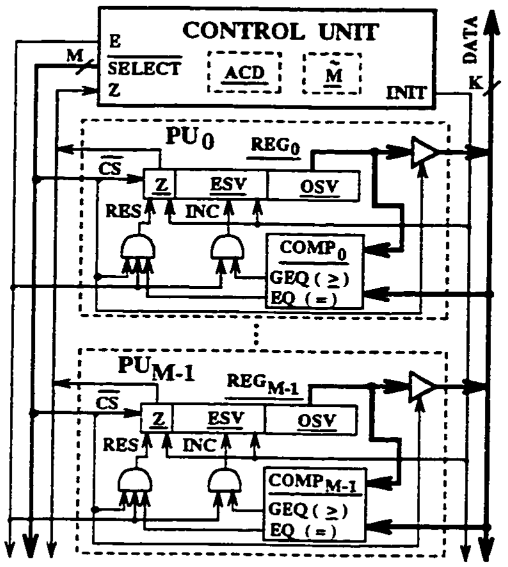
5.1.1 A Hardware Architecture for the WSC Transformation

An architecture suitable for implementing the WSC transformation is illustrated in Figure 5.2(a). The datapath consists of M identical processing units (PUs), each comprising a register-block REG with tri-state bus-drivers at the output and a comparator COMP. A register-block REG_j consists of three register-fields denoted by OSV, ESV and Z. An original sample-value $\mathbf{x}_j$ is loaded into the register-field OSV of REG_j, while the register-field $\text{ESV}(\text{REG}_j)$ is used to accumulate the encoded sample-value $\tilde{\mathbf{x}}_j$. Flag Z is used to provide a mechanism for selecting only the distinct elements of $\mathbf{x}(n)$ (i.e., for implicitly constructing the set $\mathbf{x}^*(n)$). The control unit generates the sequence of operations required for applying the WSC transformation to the sequence $\mathbf{x}(n)$. The control unit comprises an address counter/decoder block ACD and a counter $\tilde{\text{M}}$. The ACD block is used for the selection of a processing unit PU_j that places the content of its OSV register-field on the DATA-bus. The counter denoted by $\tilde{\text{M}}$ is used to accumulate the number of distinct elements in the input sequence $\mathbf{x}(n)$. At the beginning of each new operation of WSC transformation, the control unit performs an initialization of the architecture. During this initialization, all $\overline{\text{SELECT}}$ signals are set to 1, the enable signal E and counter $\tilde{\text{M}}$ are set to 0, and a pulse signal 0-1-0 is sent to all REG_j's over the INIT-line. During the time-frame when E=0 and INIT=1, the Z-flags are set to 1 and all ESV register-fields are set to -1 (i.e., all the bits of each ESV register-field are set to 1).

The complete data-flow corresponding to the process of applying the WSC transformation to a sequence $\mathbf{x}(n)$ is described by the following algorithm.

Algorithm 5.1 *WSC Algorithm*

```
Set $\overline{\text{SELECT}} = 1$; Set $\text{E} = 0$; Set $\tilde{\text{M}} = 0$;
Set $\text{INIT} = 1$;
Set $\text{INIT} = 0$;
```



(a)

REG_k (k)	0	1	2	3	4	5	6	7	8
OSV	97	255	97	93	32	97	93	93	97
Initial state (Z ; ESV)	1 ; -1	1 ; -1	1 ; -1	1 ; -1	1 ; -1	1 ; -1	1 ; -1	1 ; -1	1 ; -1
Step j = 0 (Z ; ESV)	1 ; 0	1 ; 0	0 ; 0	1 ; -1	1 ; -1	0 ; 0	1 ; -1	1 ; -1	0 ; 0
Step j = 1 (Z ; ESV)	1 ; 0	1 ; 1	0 ; 0	1 ; -1	1 ; -1	0 ; 0	1 ; -1	1 ; -1	0 ; 0
Step j = 2	No change : $Z(REG_2) = 0$								
Step j = 3 (Z ; ESV)	1 ; 1	1 ; 2	0 ; 1	1 ; 0	1 ; -1	0 ; 1	0 ; 0	0 ; 0	0 ; 1
Step j = 4 (Z ; ESV)	1 ; 2	1 ; 3	0 ; 2	1 ; 1	1 ; 0	0 ; 2	0 ; 1	0 ; 1	0 ; 2
Step j = 5	No change : $Z(REG_5) = 0$								
Step j = 6	No change : $Z(REG_6) = 0$								
Step j = 7	No change : $Z(REG_7) = 0$								
Step j = 8	No change : $Z(REG_8) = 0$								
Final state (ESV)	2	3	2	1	0	2	1	1	2

(b)

Figure 5.2: (a) An architecture suitable for a hardware implementation of the WSC transformation. (b) An example of applying the WSC transformation using the architecture of (a).

```
For $j = 0, 1, \ldots, M - 1$ do sequentially
    Set $\overline{CS}(REG_j) = 0$;
    If $Z(REG_j) = 1$ then do
        Set $E = 1$; Set $\tilde{M} = \tilde{M} + 1$;
        For $k = 0, 1, \ldots, M - 1$ do in parallel
            If $GEQ(COMP_k) = 1$
                then set $ESV(REG_k) = ESV(REG_k) + 1$;
            If $EQ(COMP_k) = 1$ and $\overline{CS}(REG_k) = 1$
                then set $Z(REG_k) = 0$;
        End
        Set $E = 0$;
    End
    Set $\overline{CS}(REG_j) = 1$;
End
```

Example 5.3 In this example, we illustrate the implementation of the WSC transformation on the architecture of Figure 5.2(a), by taking a specific window sequence, $\mathbf{x}(n) =$

$\{97, 255, 97, 93, 32, 97, 93, 93, 97\}$, that was considered in Example 5.1. The contents of all OSV-fields of register-blocks REG_k ($k = 0, 1, \ldots, 8$) during each step $j = 0, 1, \ldots, 8$ of Algorithm 5.1 are shown in Figure 5.2(b).

As seen from Figure 5.2(b), at iteration j, if $Z(\mathrm{REG}_j) = 0$, no operation of comparison is carried out. Thus, the computational time associated with the WSC architecture is determined by the number of distinct samples of each window-sequence. An analysis of the complexity of the proposed architecture of Figure 5.2(a) is carried out next.

5.1.2 Complexity Analysis of the WSC Hardware Algorithm

In this section, a comparison of the time-area complexity of the proposed WSC architecture with that of the input compression part of the ROSM architecture[1] of [1], in performing an operation of 2-D window-sequence compression, is carried out. For conducting this comparison, it is convenient to adopt the convention that the computational time associated with a K-bit comparator is one unit, and its physical realization occupies one unit of chip area.[2] In terms of this measure, the chip-area and computational-time requirements corresponding to the WSC and 2-D input compression architectures can be expressed as given in Table 5.1. While the area required by the WSC architecture in terms of the number of K-bit comparators is equal to M, the chip-area required by the input compression part of the 2-D ROSM architecture is slightly bigger and it is equal to Q M, with

$$Q = 1 + \frac{1}{K} \lceil \log_2 M \rceil \,. \tag{5.9}$$

[1]Note that two realizations of the ROSM have been proposed in [1] by employing different solutions for the implementation of the rank-update-logic stage. These two types of ROSM's have been designated as: 1) the ROSM with a feedback and 2) the ROSM without a feedback. In the complexity analysis of the algorithms proposed in this chapter, a comparison is carried out with respect to the ROSM with feedback, which achieves better time-area complexity than provided by the second type of ROSM.

[2]Note that the number of registers is the same for both the ROSM and the WSC architectures.

The expression of Q has been calculated by observing that the input compression part of the 2-D ROSM architecture comprises M K-bit comparators and M rank comparators. A rank comparator is a ($\lceil \log_2 M \rceil$)-bit comparator, and therefore it requires a chip-area of $\frac{1}{K} \lceil \log_2 M \rceil$ units.

Architecture	Area	Time
WSC architecture	M	$\tilde{M}_A$
2-D ROSM (compression part)	Q M	4 N

Table 5.1: Chip-area and computational-time requirements of the WSC architecture and of the input compression part of the 2-D ROSM architecture.

The average running time of the WSC architecture is equal to $\tilde{M}_A$, the average number of distinct samples appearing in the input window of a 2-D stack filter. On the other hand, the average running time of the input compression part of the ROSM architecture, carrying out an operation of 2-D stack filtering of size $M = N \times N$, is a constant equal to 4 N. This value has been obtained by observing that in the case of 2-D stack filtering, for a window of size $M = N \times N$, there are N new samples appearing in the input window at each iteration, and for each new sample that appears in the input window, there is a delay of approximately 4 units caused by a K-bit comparator (1 delay unit) and an adder (approx. 3 delay units [1]).

The time-area requirements indicated in Table 5.1 illustrate that in 2-D applications, the complexity of the WSC architecture is lower than (or at least comparable to) that of the ROSM configuration, as long as $4QN \geq \tilde{M}_A$. This result is further analyzed in Table 5.2, which gives the values of $\tilde{M}_A$ corresponding to each of the noisy test-images of Figure 5.9, when filters with sizes of 3×3, 5×5 and 7×7 are to be applied. This table 5.2 illustrates a comparison between the computational-time complexities of the WSC and 2-D ROSM architectures, as reflected by the relation between 4 N and $\tilde{M}_A$, for each filter size M = N $\times$ N, N = 3, 5, 7. In this comparison, it is assumed that the WSC and 2-D ROSM architectures are employed for applying an operation of 2-D stack filtering to the noisy test-images of Figure 5.9. The cases of both L = 64 and L = 256 grey levels are analyzed.

M=NxN (Filter size) / $\tilde{M}_A$, 4N	L	3x3	5x5	7x7	L	3x3	5x5	7x7
Airplane : $\tilde{M}_A$	L = 64	4.7	8.4	11.9	L = 256	6.8	13.7	22.3
Bank : $\tilde{M}_A$		4.8	8.9	12.5		6.9	13.9	22.8
Lenna : $\tilde{M}_A$		4.9	9.3	13.1		7.1	14.9	24.9
Hat : $\tilde{M}_A$		5.4	10.6	15.2		7.4	15.6	27.1
4N		12	20	28		12	20	28

Table 5.2: A comparison between the computational-time complexities of the WSC and 2-D input compression architectures for applying an operation of window-sequence compression to the specific noisy test-images of Figure 5.9.

The results of Table 5.2 show that in 2-D stack filtering, significant computational-time improvements are to be expected, when using the WSC approach instead of the input compression technique of [1]. Moreover, for commonly used filters of sizes 3 $\times$ 3, 5 $\times$ 5, or 7 $\times$ 7, the time-area complexity of the WSC architecture is also much better than that of the ROSM configuration for the four images considered in this experiment. The possibility of developing a WSC-based BTS algorithm which is faster than the input compression-based BTS approach of [1], without requiring an increased chip-area, is investigated in the next section.

5.2 A WSC-Based BTS Algorithm

By using the WSC transformation in conjunction with the conventional BTS procedure, a WSC-based BTS algorithm for stack filtering is now presented. Figure 5.3 is the block

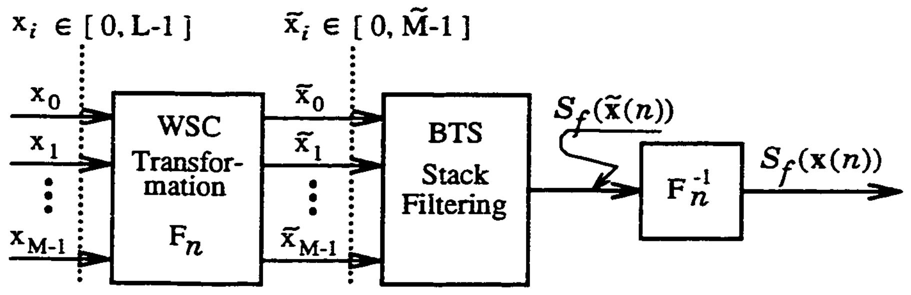


Figure 5.3: Block diagram of a WSC-based BTS technique for stack filtering.

diagram illustrating the sequence of the operations in the proposed technique. The various steps involved with this technique can be described in the form of the following algorithm.

Algorithm 5.2 *WSC-Based BTS Algorithm*

1. Construct the WSC transformation F_n and determine

$$\tilde{\mathbf{x}}(n) = \{ F_n[x_i(n)] \mid i = 0, 1, \ldots, M-1 \} .$$

2. Using the binary-tree search approach and taking $K \doteq K_n = \lceil \log_2 \tilde{M} \rceil$, apply the operation of stack filtering to the sequence $\tilde{\mathbf{x}}(n)$. Let $\mathcal{S}_f(\tilde{\mathbf{x}}(n)) = \tilde{x}_j^*$.

3. Determine the output of the filter at the time instant n as $y_n = F_n^{-1}[\tilde{x}_j^*] = x_j^*$.

Thus, the output of Algorithm 5.2 is given by

$$\begin{aligned} y_n &= F_n^{-1} [\mathcal{S}_f(\tilde{\mathbf{x}}(n))] \\ &= F_n^{-1} [\sum_{J=1}^{K_n} f(\delta(\tilde{x}_0 - \ell_J), \delta(\tilde{x}_1 - \ell_J), \ldots, \delta(\tilde{x}_{M-1} - \ell_J)) \cdot 2^{K_n - J}] , \end{aligned} \tag{5.10}$$

and the threshold level in the Jth step can be obtained iteratively as

$$\ell_1 = 2^{K_n - 1} , \tag{5.11}$$

$$\ell_J = [\sum_{j=1}^{J-1} f(\delta(x_0 - \ell_j), \ldots, \delta(x_{M-1} - \ell_j)) \cdot 2^{K_n - j}] + 2^{K_n - J} , \quad \text{for } J \in \{2, \ldots, K_n\} . \tag{5.12}$$

Note that (5.10) follows readily from (2.30).

In Algorithm 5.2, an output y_n given by (5.10), corresponding to an input $\mathbf{x}(n)$, is obtained by performing an operation of stack filtering $\mathcal{S}_f$ to the sequence $\tilde{\mathbf{x}}(n)$. In order to demonstrate that this algorithm is functionally equivalent to applying the operator $\mathcal{S}_f$ directly to $\mathbf{x}(n)$, we first present the following two lemmas.

Lemma 5.1 If $\mathbf{x}_o(n) = \{\mathrm{x}_{(0)}, \mathrm{x}_{(1)}, \ldots, \mathrm{x}_{(\mathrm{M}-1)}\}$ denotes an ordered set obtained by sorting the elements of $\mathbf{x}(n)$ in the ascending order, then

$$\mathcal{S}_f(\mathbf{x}(n)) = \sum_{\ell=1}^{2^{\mathrm{K}}-1} f(\delta(\mathrm{x}_0 - \ell), \delta(\mathrm{x}_1 - \ell), \ldots, \delta(\mathrm{x}_{\mathrm{M}-1} - \ell))$$

$$= \sum_{i=0}^{\mathrm{M}-1} \{ [\mathrm{x}_{(i)} - \mathrm{x}_{(i-1)}] \cdot f(\delta(\mathrm{x}_0 - \mathrm{x}_{(i)}), \delta(\mathrm{x}_1 - \mathrm{x}_{(i)}), \ldots, \delta(\mathrm{x}_{\mathrm{M}-1} - \mathrm{x}_{(i)})) \}\,, \quad (5.13)$$

where $\mathrm{x}_{(-1)} = 0$.

Proof: The relation (5.13) follows immediately, by expressing the summation $\sum_{\ell=1}^{2^{\mathrm{K}}-1} f(\cdot)$ as

$$\sum_{\ell=1}^{2^{\mathrm{K}}-1} f(\cdot) = \sum_{\ell=1}^{\mathrm{x}_{(0)}} f(\cdot) + \sum_{\ell=\mathrm{x}_{(0)}+1}^{\mathrm{x}_{(1)}} f(\cdot) + \ldots + \sum_{\ell=\mathrm{x}_{(\mathrm{M}-2)}+1}^{\mathrm{x}_{(\mathrm{M}-1)}} f(\cdot)$$

$$= \sum_{i=0}^{\mathrm{M}-1} \left(\sum_{\ell=\mathrm{x}_{(i-1)}+1}^{\mathrm{x}_{(i)}} f(\cdot) \right)\,, \quad (5.14)$$

and observing that for each interval $[\mathrm{x}_{(i-1)} + 1, \mathrm{x}_{(i)}]$, with $i = 0, 1, \ldots, \mathrm{M}-1$, we have

$$f(\delta(\mathrm{x}_0 - \ell), \delta(\mathrm{x}_1 - \ell), \ldots, \delta(\mathrm{x}_{\mathrm{M}-1} - \ell)) = f(\delta(\mathrm{x}_0 - \mathrm{x}_{(i)}), \delta(\mathrm{x}_1 - \mathrm{x}_{(i)}), \ldots, \delta(\mathrm{x}_{\mathrm{M}-1} - \mathrm{x}_{(i)}))\,,$$

for all $\ell \in [\mathrm{x}_{(i-1)} + 1, \mathrm{x}_{(i)}]$. Thus, the inner summation in (5.14) becomes

$$\sum_{\ell=\mathrm{x}_{(i-1)}+1}^{\mathrm{x}_{(i)}} f(\delta(\mathrm{x}_0 - \ell), \delta(\mathrm{x}_1 - \ell), \ldots, \delta(\mathrm{x}_{\mathrm{M}-1} - \ell)) =$$

$$= \left[\mathrm{x}_{(i)} - \mathrm{x}_{(i-1)}\right] \cdot f(\delta(\mathrm{x}_0 - \mathrm{x}_{(i)}), \delta(\mathrm{x}_1 - \mathrm{x}_{(i)}), \ldots, \delta(\mathrm{x}_{\mathrm{M}-1} - \mathrm{x}_{(i)}))\,,$$

and the right side of (5.14) reduces to that of (5.13). □

Lemma 5.2 The following equivalence is true for any stack filter $\mathcal{S}_f$:

$$\mathcal{S}_f(\mathbf{x}(n)) = \mathbf{x}_{(\mathrm{I})} \iff \mathcal{S}_f(\tilde{\mathbf{x}}(n)) = \tilde{\mathbf{x}}_{(\mathrm{I})}\,, \tag{5.15}$$

where I denotes an arbitrary number in the set $\{0, 1, \ldots, \ \mathrm{M}-1\}$.

Proof: In view of Lemma 5.1, and using the notation $\tilde{\mathbf{x}}_o(n) = \{\tilde{\mathbf{x}}_{(0)}, \tilde{\mathbf{x}}_{(1)}, \ldots, \tilde{\mathbf{x}}_{(\mathrm{M}-1)}\}$ to designate the ordered set obtained by sorting the elements of $\tilde{\mathbf{x}}(n)$ in the ascending order, $\mathcal{S}_f(\tilde{\mathbf{x}}(n))$ can be expressed as

$$\mathcal{S}_f(\tilde{\mathbf{x}}(n)) \;=\; \sum_{i=0}^{\mathrm{M}-1} \{\, [\tilde{\mathbf{x}}_{(i)} - \tilde{\mathbf{x}}_{(i-1)}] \cdot f(\delta(\tilde{\mathbf{x}}_0 - \tilde{\mathbf{x}}_{(i)}), \delta(\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_{(i)}), \ldots, \delta(\tilde{\mathbf{x}}_{\mathrm{M}-1} - \tilde{\mathbf{x}}_{(i)}))\,\}\,, \tag{5.16}$$

where $\tilde{\mathbf{x}}_{(-1)} = 0$. In order to prove that the forward implication of (5.15) is true, we assume that $\mathcal{S}_f(\mathbf{x}(n)) = \mathbf{x}_{(\mathrm{I})}$, and therefore, based on Property 5.1 of F_n, we have

$$\begin{aligned} f(\delta(\tilde{\mathbf{x}}_0 - \tilde{\mathbf{x}}_{(i)}), \ldots, \delta(\tilde{\mathbf{x}}_{\mathrm{M}-1} - \tilde{\mathbf{x}}_{(i)})) \;&=\; f(\delta(\mathbf{x}_0 - \mathbf{x}_{(i)}), \ldots, \delta(\mathbf{x}_{\mathrm{M}-1} - \mathbf{x}_{(i)})) \;= \\ &= \begin{cases} 1 & \text{for} \quad i \leq \mathrm{I} \\ 0 & \text{for} \quad i > \mathrm{I}\,. \end{cases} \end{aligned} \tag{5.17}$$

Now, using (5.17), (5.16) becomes

$$\mathcal{S}_f(\tilde{\mathbf{x}}(n)) \;=\; \tilde{\mathbf{x}}_{(0)} + \sum_{i=0}^{\mathrm{I}} [\tilde{\mathbf{x}}_{(i)} - \tilde{\mathbf{x}}_{(i-1)}] \;=\; \tilde{\mathbf{x}}_{(\mathrm{I})}\,. \tag{5.18}$$

The demonstration of the backward implication of (5.15) is omitted, since it follows a similar argument as the proof of the forward implication. □

Using Lemma 5.2, we will now establish the main theorem.

Theorem 5.1 The WSC-based BTS algorithm performs the operation of stack filtering, i.e., y_n as given by (5.10) also yields

$$y_n = \mathcal{S}_f(\mathbf{x}(n))\,. \tag{5.19}$$

Proof: Assuming that $\mathcal{S}_f(\tilde{\mathbf{x}}(n)) = \tilde{\mathbf{x}}_{(\mathrm{I})}$, from (5.10), we have

$$y_n = \mathrm{F}_n^{-1}\left[\mathcal{S}_f(\tilde{\mathbf{x}}(n))\right] = \mathrm{F}_n^{-1}\left[\tilde{\mathbf{x}}_{(\mathrm{I})}\right]. \tag{5.20}$$

Based on the definition and properties of F_n, there should be an element $\mathbf{x}_{(\mathrm{I})}$ in $\mathbf{x}(n)$ such that $\tilde{\mathbf{x}}_{(\mathrm{I})} = \mathrm{F}_n\left[\mathbf{x}_{(\mathrm{I})}\right]$ and thus $\mathrm{F}_n^{-1}\left[\tilde{\mathbf{x}}_{(\mathrm{I})}\right] = \mathbf{x}_{(\mathrm{I})}$. Therefore, (5.20) can be written as

$$y_n = \mathbf{x}_{(\mathrm{I})}. \tag{29}$$

Now, using $\mathcal{S}_f(\tilde{\mathbf{x}}(n)) = \tilde{\mathbf{x}}_{(\mathrm{I})}$, and the backward implication of Lemma 5.2, it follows that $\mathbf{x}_{(\mathrm{I})} = \mathcal{S}_f(\mathbf{x}(n))$, and consequently $y_n = \mathbf{x}_{(\mathrm{I})} = \mathcal{S}_f(\mathbf{x}(n))$. □

5.3 Architecture for the WSC-Based BTS Algorithm

Algorithm 5.2 can be implemented by employing the WSC architecture discussed in Section 5.1.1, and the BTS configuration along with a control circuitry and a circuitry to implement F_n^{-1}. The block diagram of the architecture is shown in Figure 5.4, whereas the associated timing information is provided in Figure 5.5. In the block diagram, the shaded boxes represent the WSC-control unit and the PUs of the WSC architecture of Figure 5.2. It is to be noted that the architecture of Figure 5.4 uses the same registers $\mathrm{REG}_0, \mathrm{REG}_1, \ldots, \mathrm{REG}_{M-1}$, as does the WSC architecture of Figure 5.2. As the interconnections among the blocks of the WSC architecture are clearly illustrated in Figure 5.2, they are not depicted in the diagram of Figure 5.4. Instead, the diagram of Figure 5.4 emphasizes the way in which the blocks of the WSC architecture are incorporated into the WSC-based BTS configuration for stack filtering. While the shaded blocks of the architecture of Figure 5.4 implement Step 1 of Algorithm 5.2 (i.e, the WSC transformation), the BTS block of the architecture realizes Step 2 of this algorithm. As shown in Figure 5.5, the signal denoted by $\overline{\mathrm{WSC}}/\mathrm{BTS}$ is used to control the sequence of WSC and BTS operations. Note that the samples appearing in a window-sequence $\mathbf{x}(n)$ are loaded into the OSV register-fields at the beginning of each WSC-operation, during the time when D_LD = 1.

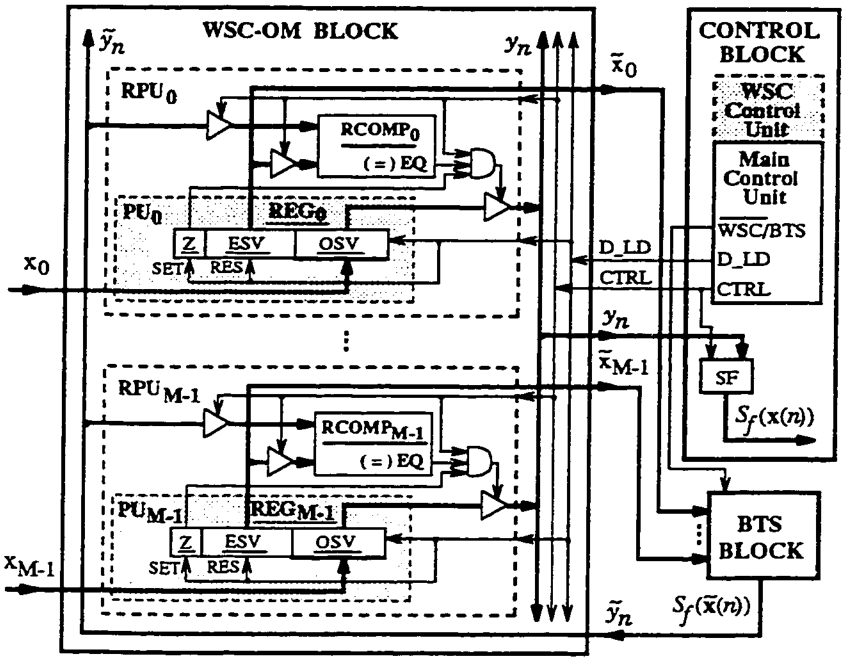


Figure 5.4: Block diagram of a WSC-based BTS architecture for stack filtering.

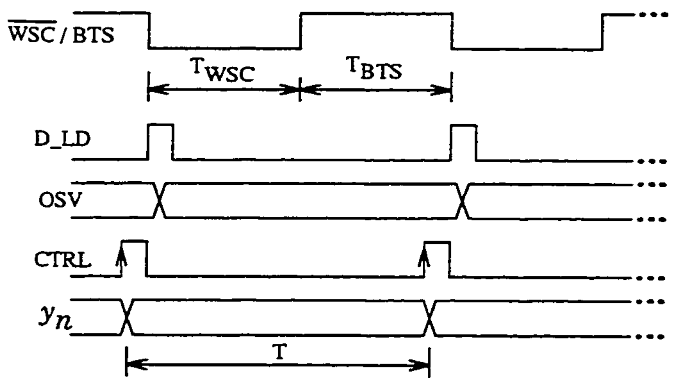


Figure 5.5: Waveforms associated with the architecture of Figure 5.4.

At the same time, all Z-flags are set to 1 and all ESV register-fields are initialized with -1. The BTS block of Figure 5.4 has a configuration similar to that of the binary-tree search structure of Figure 2.8, except that the adjustment of the threshold level is accomplished with a time-varying value of K, i.e., $K \doteq K_n = \lceil \log_2 \tilde{M} \rceil$. Step 3 of Algorithm 5.2 is realized by the output-mapping (OM) unit of the WSC-OM block. In Figure 5.4, the OM unit is implemented by the circuit of the unshaded part of the WSC-OM block, and it consists of

M rank processing units (RPUs). These RPUs are used to determine the output of the filter at the time instant n, $y_n = \mathcal{S}_f(\mathbf{x}(n))$, based on the knowledge of the rank $\tilde{y}_n = \mathcal{S}_f(\tilde{\mathbf{x}}(n))$ of this output among the samples $\mathbf{x}(n)$ appearing at the input of the filter at time n. Each rank processing unit RPU$_j$ is built around the register REG$_j$ of the processing unit PU$_j$ of the WSC architecture. Each block RPU$_j$ comprises a rank comparator RCOMP$_j$, which is used to determine whether or not $\tilde{y}_n$ is equal to ESV(REG$_j$). The OSV-field of each register REG$_j$ is connected to the bus denoted by y_n through a tri-state buffer. This tri-state buffer is normally in the high-impedance state, except for the case when $\tilde{y}_n$ = ESV(REG$_j$), Z(REG$_j$) = 1, and CTRL = 1. In the exception case, the OSV-field of register REG$_j$ contains the result of the filtering operation, and consequently, it is required to place its content on the y_n-bus. The result of applying an operation of 2-D stack filtering to the sequence $\mathbf{x}(n)$ is loaded into the register SF after the rising edge of signal CTRL.

5.4 A Pipelined Architecture for the WSC-Based BTS Algorithm

The computational efficiency of the architecture shown in Figure 5.4 can be significantly improved, with a slight increase in the chip-area, by employing a technique of pipelining. Such a modified architecture is shown in Figure 5.6. This architecture has been obtained from the block diagram of Figure 5.4 by completely separating the WSC and OM units of the WSC-OM block. In terms of the chip-area, this operation amounts to only introducing M additional registers, denoted by RREG$_0$, RREG$_1$, ..., RREG$_{M-1}$. At the time-instant n, when the WSC block performs the encoding of the sequence $\mathbf{x}(n)$, the OM and BTS blocks operate on the previously encoded sequence $\tilde{\mathbf{x}}(n-1)$.

The timing information regarding the waveforms used in the architecture of Figure 5.6 is provided in Figure 5.7. As compared to the timing diagram of Figure 5.5, a new signal, denoted by BTSCLK, is also shown in Figure 5.7. This signal, which is generated

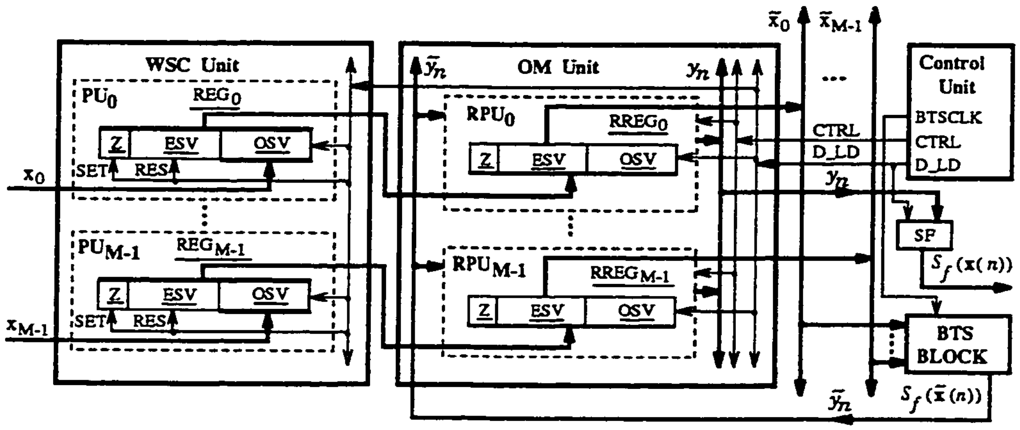


Figure 5.6: Proposed pipelined WSC-BTS architecture for stack filtering.

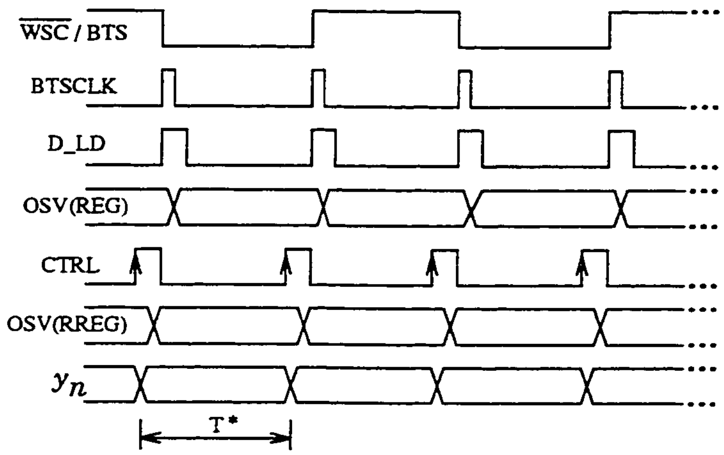


Figure 5.7: Waveforms associated with the architecture of Figure 5.6.

from $\overline{\text{WSC}}$/BTS, is used to control the data transfer between the registers REG$_j$'s and RREG$_j$'s. As expected, the throughput 1/T* of the architecture of Figure 5.6 is two times higher than 1/T, the throughput of the architecture of Figure 5.4.

In the complexity analysis carried out in the next section, it is shown that in some cases the WSC transformation takes longer computational time than the BTS operation itself. In these situations, it is desirable to use a multistage architecture, employing several WSC- and OM-stages.

5.5 Complexity Analysis of the Pipelined WSC-Based BTS Architecture

In Section 5.1.2, a comparison between the time-area complexities of the WSC architecture and the input compression part of the 2-D ROSM architecture was carried out, by adopting the convention that the computational time associated with a K-bit comparator is one unit, and its physical realization occupies one unit of chip area. In this section, we extend the analysis, and compare the time-area complexities of the proposed pipelined WSC-based BTS architecture and the complete ROSM architecture for 2-D stack filtering. To this end, it would be useful to introduce the following notations:

$\hat{T}_{WSC}$	—	estimated duration of the operation of WSC transformation;
$\tilde{T}_{ROSM}$	—	duration of the operation of 2-D ROSM input compression;
$\hat{T}_{BTS}$	—	estimated duration of a BTS operation with the WSC-based approach;
$\tilde{T}_{BTS}$	—	duration of a BTS operation with the ROSM approach;
T_{th}	—	duration of a thresholding operation within the BTS block;
T_f	—	duration of the operation of Boolean filtering f;
T_{OM}	—	duration of the operation of output mapping.

It is readily seen that $\hat{T}_{BTS}$ and $\tilde{T}_{BTS}$ can be evaluated using the following relations

$$\hat{T}_{BTS} = \lceil \log_2(\tilde{M}_A) \rceil (T_{th} + T_f) + T_{OM} , \tag{5.21}$$

$$\tilde{T}_{BTS} = \lceil \log_2 M \rceil (T_{th} + T_f) + T_{OM} . \tag{5.22}$$

It is to be noted that both the proposed and the ROSM-BTS architectures use ($\lceil \log_2 M \rceil$)-bit comparators for carying out the thresholding operations within their respective BTS blocks. As a result, T_{th} is given by

$$T_{th} = \frac{1}{K} \lceil \log_2 M \rceil . \tag{5.23}$$

Also, as shown in [1], the operation of Boolean filtering may take 1, 2, 3 or 4 delay-units[3], i.e., $T_f = 1, 2, 3,$ or 4, while the computational time T_{OM} is approximately equal to 0.5 units.

A comparison between the time-area complexities of the proposed and ROSM-BTS architectures for 2-D stack filtering is illustrated in Table 5.3. This comparison is carried out by considering the case of the noisy test-image Hat, for which, as shown in Table 5.2, the worst-case values of $\tilde{M}_A$ have been obtained. It is to be noted here that, as illustrated by (5.21) and (5.22), the relation between $\hat{T}_{BTS}$ and $\tilde{T}_{BTS}$ is completely determined by that between $\lceil \log_2(\tilde{M}_A) \rceil$ and $\lceil \log_2 M \rceil$. In Table 5.3, the chip-area is expressed by calculating the number of WSC- and ROSM-stages required for implementing an operation of 2-D stack filtering with maximum throughput when using the proposed and the ROSM-BTS architectures, respectively. The maximum throughputs for the proposed and ROSM-BTS configurations are given by $(1/\hat{T}_{BTS})$ and $(1/\tilde{T}_{BTS})$, respectively. In Table 5.3, the number of WSC stages is denoted by NS_{WSC}, while that of ROSM stages is designated as NS_{ROSM}. The values of NS_{WSC} and NS_{ROSM} are calculated as

$$NS_{WSC} = \lceil \hat{T}_{WSC} / \hat{T}_{BTS} \rceil , \tag{5.24}$$

$$NS_{ROSM} = \lceil \tilde{T}_{ROSM} / \tilde{T}_{BTS} \rceil . \tag{5.25}$$

As it can be easily seen by analyzing the results in Table 5.3, important improvements in terms of the throughput are achieved with the proposed architecture over the ROSM-BTS configuration without increasing the chip-area. Specifically, for all filter-sizes and all possible values of T_f, except for the case of 7×7 filters with $T_f = 4$, $NS_{WSC} \leq NS_{ROSM}$. Moreover, as shown in Table 5.1, the chip-area required by a WSC-stage is smaller than the one corresponding to a 2-D ROSM-stage by a factor of $Q = 1 + (1/K)\lceil \log_2 M \rceil$. As a result, even in the case of a filter of size 7×7 with $T_f = 4$,

[3] In [1], the computational-time has been expressed in terms of gate-delay units. As such, a K-bit comparison takes 3 gate-delay units, while the operation of computing the PBF may take at least 2 and at most 12 gate-delay units. As stated in Section 5.1.2, the delay-unit is taken as the computational time required to carry out a K-bit comparison. Using the measure of [1], this computational-time can be equivalently expressed as 3 gate-delay units.

T_f	Filter size					
	3 x 3		5 x 5		7 x 7	
	$\hat{T}_{WSC}$	$\tilde{T}_{ROSM}$	$\hat{T}_{WSC}$	$\tilde{T}_{ROSM}$	$\hat{T}_{WSC}$	$\tilde{T}_{ROSM}$
	7.4	12	15.6	20	27.1	28
	$\hat{T}_{BTS}$	$\tilde{T}_{BTS}$	$\hat{T}_{BTS}$	$\tilde{T}_{BTS}$	$\hat{T}_{BTS}$	$\tilde{T}_{BTS}$
1	5.0	6.5	7.0	8.625	9.25	11.0
2	8.0	10.5	11.0	13.625	14.25	17.0
3	11.0	14.5	15.0	18.625	19.25	23.0
4	14.0	18.5	19.0	23.625	24.25	29.0
	NS_{WSC}	NS_{ROSM}	NS_{WSC}	NS_{ROSM}	NS_{WSC}	NS_{ROSM}
1	1	2	2	2	2	2
2	2	2	3	3	3	3
3	1	1	2	2	2	2
4	1	1	1	1	2	1

Table 5.3: A comparison between the time-area complexities of the WSC-BTS and ROSM-BTS architectures.

for which $NS_{WSC} = 2$ while $NS_{ROSM} = 1$, the chip-area corresponding to the proposed architecture is still comparable with that of the ROSM-BTS configuration. Specifically, in this case the ROSM stage occupies $1.75 \times 49 = 85.75$ units of chip-area, while the two WSC stages take $2 \times 49 = 98$ units of area.

5.6 Experimental Investigation

Four test-images and their noisy versions have been used in this chapter to evaluate and compare the computational efficiencies of the WSC-BTS and ROSM-BTS architectures for 2-D stack filtering. All test images have a size of 256×256 and two versions of grey levels, $L = 64$ and $L = 256$, have been used. The original test images for $L = 256$ are shown in Figure 5.8, and they are designated as Airplane, Bank, Lenna and Hat images. The corresponding noise-corrupted versions of these images are illustrated in Figure 5.9, and they have been obtained from their respective noise-free versions by using additive impulsive noise with a probability of occurrence of 10%. The impulsive noise is uniformly

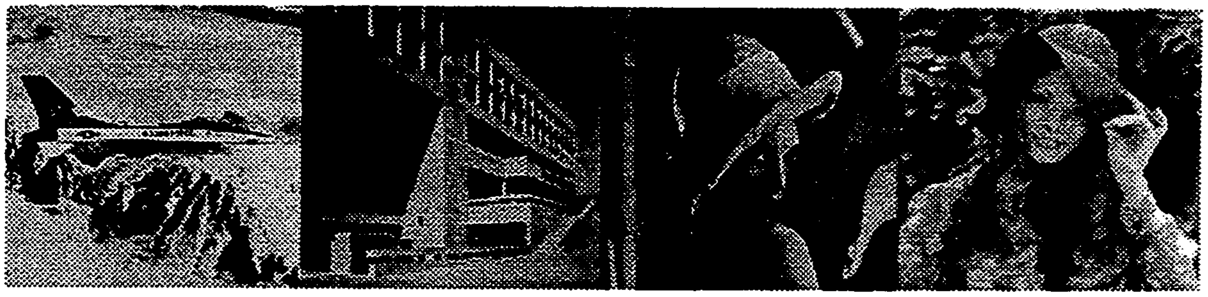

(a) (b) (c) (d)

Figure 5.8: Original test-images used in the experiments: a) Airplane, b) Bank, c) Lenna, and d) Hat. All the images have a size of 256 × 256 and L = 256.

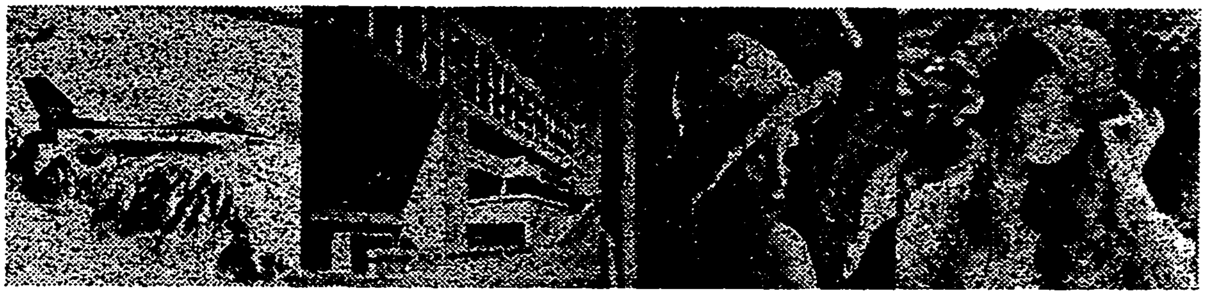

(a) (b) (c) (d)

Figure 5.9: Noisy test-images used in the experiments: a) Airplane, b) Bank, c) Lenna, and d) Hat. All the images have a size of 256 × 256, and are corrupted with impulsive noise with a probability of occurrence of 10%.

distributed over the grey level interval [-L,L]. The saturation levels of 0 and (L-1) are used for the noise-corrupted images, leading to a salt-and-pepper noise pattern.

An illustration of the computational-time efficiency that can be achieved with the proposed WSC-based BTS approach using pipelined architecture over the ROSM-BTS technique for 2-D stack filtering, is provided in Table 5.4. The results of this table have been calculated based on counting the total number of iterations $N_{WSC-BTS}$ required to complete an operation of 2-D stack filtering by using the proposed approach in the case of each of the original and noisy test-images. Note that the total number of iterations $N_{ROSM-BTS}$ required to perform an operation of 2-D stack filtering by employing an ROSM-BTS architecture is completely determined by the size M of the filter, that is,

	Filter size / Image	L	3 x 3	5 x 5	7 x 7	L	3 x 3	5 x 5	7 x 7
Original	Airplane	L = 64	59 %	48 %	48 %	L = 256	33 %	20 %	23 %
	Bank		58 %	46 %	44 %		31 %	19 %	21 %
	Lenna		57 %	44 %	43 %		31 %	16 %	18 %
	Hat		49 %	35 %	34 %		29 %	11 %	13 %
Noisy	Airplane		47 %	35 %	37 %		28 %	15 %	19 %
	Bank		46 %	35 %	36 %		27 %	15 %	19 %
	Lenna		45 %	32 %	34 %		26 %	13 %	17 %
	Hat		39 %	27 %	29 %		25 %	10 %	12 %

Table 5.4: An illustration of the computational-time efficiency η of the WSC-BTS approach over the ROSM-BTS technique for 2-D stack filtering.

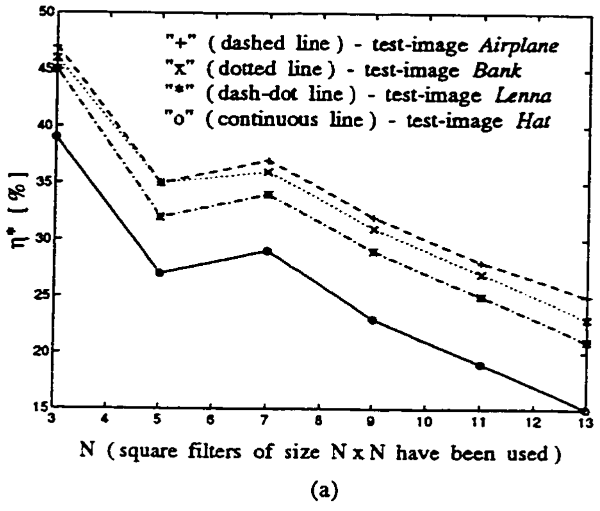
$N_{\text{ROSM-BTS}} = \lceil \log_2 M \rceil \times 256 \times 256$. The computational-time efficiency is determined using a measure given by

$$\eta = \frac{N_{\text{ROSM-BTS}} - N_{\text{WSC-BTS}}}{N_{\text{ROSM-BTS}}} \cdot 100\,[\%]\,. \tag{5.26}$$

As seen from the results of Table 5.4, significant improvements in terms of the computational-speed can be obtained when the operation of 2-D stack filtering is carried out by using the proposed architecture. The computational efficiency of this architecture is further reinforced by the curves shown in Figure 5.10. These curves have been drawn showing the values of the computational efficiency η^* calculated as

$$\eta^* = \begin{cases} \eta & \text{for} \quad L \geq M \\ \tilde{\eta} & \text{for} \quad L < M\,, \end{cases} \tag{5.27}$$

where $\tilde{\eta} = (1/N_{\text{BTS}})(N_{\text{BTS}} - N_{\text{MBTS}}) \cdot 100\,[\%]$ and $N_{\text{BTS}} = \lceil \log_2 L \rceil \times 256 \times 256$, as a function of N $(= \sqrt{M})$. The reason for modifying the measure of the computational-time efficiency from (5.26) to (5.27) is that for L < M, the ROSM-BTS architecture is less efficient than the conventional BTS configuration [13] itself. Therefore, in this case, the



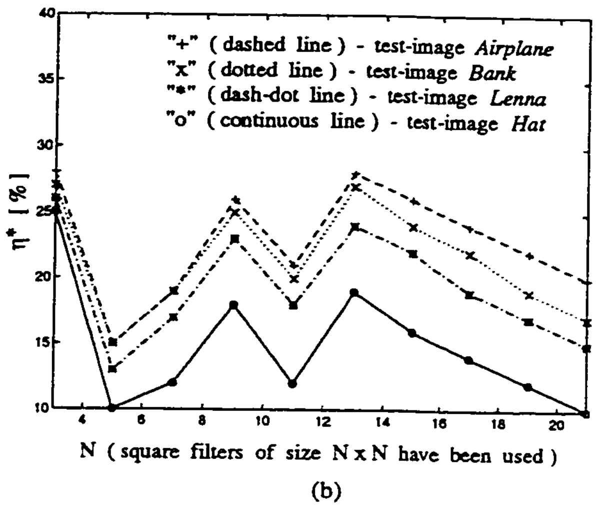


Figure 5.10: An illustration of the computational-time efficiency η^* of the proposed WSC-BTS pipelined architecture over the ROSM-BTS or BTS architectures for 2-D stack filtering. The curves illustrate the values of η^* as a function of N, where N $\times$ N is the size of the filter. The inputs to the filters are the noisy test-images with the number of grey-levels (a) L = 64, and (b) L = 256.

performance of the WSC-BTS approach is compared with that of the BTS technique. The computational-time efficiency η^*, as depicted in Figure 5.10, has been obtained using the noisy test-images mentioned earlier with L = 64 and L = 256.

The curves of Figure 5.10 illustrate that the proposed approach of 2-D stack filtering achieves considerably improved efficiency over the ROSM-BTS technique or the BTS technique itself for a very large range of filter sizes. By analyzing the results of Figure 5.10(b), it can be also observed that the computational-time efficiencies corresponding to 5 $\times$ 5, 7 $\times$ 7, and 11 $\times$ 11 filters are lower than those which can be achieved by using the filters of window-sizes 3 $\times$ 3, 9 $\times$ 9, and 13 $\times$ 13. This type of fluctuation is a result of the relation between η and the distance $D = \left|\mathrm{M} - 2^{\lfloor \log_2 \mathrm{M} \rfloor}\right|$ for different filter-sizes M. Specifically, η assumes larger values when D is smaller, since for a smaller D, the probability that at each window-position there are less than $\lfloor \log_2 \mathrm{M} \rfloor$ distinct samples is larger.

5.7 Further Discussion on the WSC-BTS and BSWP Architectures

The number of iterations per output sample required by the WSC-BTS architecture is smaller than $\lceil \log_2 M \rceil$. By comparison, the BSWP architecture developed in the previous chapter is slower, requiring an average of $\log_2 M + k$ iterations per output sample. For instance, employing a 3×3 filter with $L = 256$, the WSC-BTS technique requires approximately 3 iterations per output sample while, as given in Table 4.2, the BSWP algorithm requires approximately 4 iterations per output sample. However, when the PBF circuit can be implemented using a reasonably small chip area (e.g., in the case of rank order filters), the WSC-BTS architecture occupies significantly increased chip-area compared to that of the BSWP configuration for stack filtering. Thus, in this situation, the BSWP approach provides an improved time-area complexity compared to that of the WSC-BTS technique. On the other hand, when the PBF circuit has to be implemented using ROM memories, the PBF part of the stack filter configuration occupies the bulk of the chip-area. In this case, the comparison between the time-area complexities of the BSWP and WSC-BTS techniques clearly favors the latter. Specifically, in this case, the increase in the chip-area required by the WSC-BTS structure compared to the BSWP configuration is far less important than the computational savings that can be achieved by using the WSC-BTS approach instead of the BSWP technique.

5.8 Summary

In this chapter, a modified binary-tree search algorithm for 2-D stack filtering has been developed. The algorithm uses a new input compression technique designated as the window-sequence coding transformation, which takes advantage of the fact that in most images, many pixels appearing in the input-window of a 2-D stack filter at a certain time-instant assume non-distinct values. It has been demonstrated that the proposed WSC-based BTS algorithm is functionally equivalent to the operation of stack filtering. A hardware architecture implementing the proposed algorithm for stack filtering has been given. In order to improve the computational efficiency, a pipelined version of this architecture has also been developed. A comparison between the computational-efficiencies of the proposed WSC-BTS approach and the ROSM-BTS technique for 2-D stack filtering has been carried out. It has been shown that with the proposed approach, considerable improvements in terms of the computational-time can be achieved without an increase in the chip-area. The new approach is especially attractive in those cases when the window-size M is only slightly larger than an integer that is a power of two (e.g., M=9 — as it is the case for a 3×3 filter — or M=21 — as it is the case for a 5×5 filter in which the corners are chopped off). Significant improvements in computational efficiency are also achieved in those image-processing applications where L and M assume comparable values. For instance, in the case of 3×3 filters, computational-improvements of at least $\eta = 25\%$ can be easily obtained for images with L=256, while for images with L=64, the efficiency can easily be increased to a value of $\eta = 45\%$.

Chapter 6

Design of Locally Optimal Rank Order Filters with Application to Deinterlacing Problems

The computational cost associated with on-line design and implementation of optimal stack filters may not be unreasonably high in those image processing applications in which suitable signal and noise models that do not change frequently, are available. However, in commercial applications such as video coding for digital television systems, signal models do not conform with this constraint and as such this optimal solution could be quite expensive. Since at the multilevel domain a stack filter performs an operation of signal-dependent rank order filtering and in typical images, the details of the scene are locally situated, it can be expected that a filter bank of locally optimal rank order operators may provide an effective alternative solution to the general on-line design and implementation of optimal stack filters.

In this chapter, we investigate the possibility of designing locally optimal rank order filters and using them for spatial interpolation. An efficient algorithm for the design of rank order filters which are locally optimal in the MAE sense is proposed. The algorithm uses a threshold decomposition technique in which the rank of a window sample $\mathbf{x}_j$ is determined by employing the threshold sequence corresponding to the level $\ell = \mathrm{x}_j$. As a result, with the proposed algorithm, the minimum mean absolute error (MMAE) design solution of rank order filters (ROFs) is achieved without carrying out the operation of sorting the samples appearing at the input-window of the filter, as is generally required in the conventional techniques for the MMAE design of ROFs. It is shown that the proposed algorithm can be efficiently implemented in hardware by employing a slightly modified version of the WSC architecture of Chapter 5. An application of the proposed algorithm and architecture to spatial interpolation is also investigated by developing a procedure for intrafield deinterlacing of video signals.

6.1 An Algorithm for the MAE Design of Locally Optimal Rank Order Filters

The problem of determining a rank order filter of size M that is optimal for signal estimation in the mean absolute error sense can be solved by using a conceptually simple procedure. Specifically, an optimal rank order filter can be determined by first evaluating the MAE corresponding to each possible rank r, and then selecting the ROF which achieves the minimum MAE over all possible values of $r \in \{0, 1, \ldots, \mathrm{M}-1\}$. An algorithm suitable for this design can be easily established as given below.

1. Set $\mathrm{MAE}(r) = 0$ for $r = 0, 1, \ldots, \mathrm{M}-1$, and set $n = 0$.
2. Sort the elements of the window sequence $\mathbf{x}(n) = \{\mathrm{x}_0, \mathrm{x}_1, \ldots, \mathrm{x}_{\mathrm{M}-1}\}$ in decreasing order, and let $\mathbf{x}_o(n) = \{\mathrm{x}_{(0)}, \mathrm{x}_{(1)}, \ldots, \mathrm{x}_{(\mathrm{M}-1)}\}$ denote the resulting set.
3. Set $r = 0$.
4. Update the MAE corresponding to the current value of r, i.e., set
$$\mathrm{MAE}(r) = \mathrm{MAE}(r) + \left| \mathrm{S}(n) - \mathrm{x}_{(r)} \right|,$$
where $\mathrm{S}(n)$ denotes the desired output at the window position n.
5. While $r < \mathrm{M}$, set $r = r + 1$ and go to Step 4.

6. While n is smaller than the size of the training set, let $n = n + 1$ and go to Step 2.
7. Set $r = 1$ and ROF $= 0$.
8. If MAE(ROF) $>$ MAE(r), then set ROF $= r$.
9. While $r < \mathrm{M}$, set $r = r + 1$ and go to Step 8. Otherwise stop.

In this section, an alternative algorithm for the MAE design of locally optimal ROFs is developed, by employing a threshold decomposition technique. Specifically, it is observed that the rank r of an element $\mathbf{x}_j$ can be determined by using the threshold sequence $\{\delta(\mathbf{x}_0 - \mathbf{x}_j), \delta(\mathbf{x}_1 - \mathbf{x}_j), \ldots, \delta(\mathbf{x}_{\mathrm{M}-1} - \mathbf{x}_j)\}$, as formalized by the two lemmas given below.

Lemma 6.1 An element $\mathbf{x}_j$ that appears only once in the set $\mathbf{x}(\mathrm{n})$ is the *r-th* largest element of this set if and only if

$$r = \sum_{k=0}^{\mathrm{M}-1} \delta(\mathbf{x}_k - \mathbf{x}_j) . \tag{6.1}$$

Lemma 6.2 If for an element $\mathbf{x}_j$ of the set $\mathbf{x}(\mathrm{n})$, there are Z_j other elements in $\mathbf{x}(\mathrm{n})$ which assume the same value as $\mathbf{x}_j$, then the ranks of these $(\mathrm{Z}_j + 1)$ elements are given by $r, r-1, \ldots, r - \mathrm{Z}_j$, where r is given by (6.1).

Lemmas 6.1 and 6.2 can be easily proved by contradiction. As a consequence of these lemmas, it can be observed that the time-consuming operation of sorting the elements of the sequence $\mathbf{x}(\mathrm{n})$ can be eliminated by employing a threshold decomposition architecture for the MAE design of locally optimal ROFs. The steps of the proposed algorithm are summarized below.

1. Set MAE(r) $= 0$ for $r = 0, 1, \ldots, \mathrm{M} - 1$, and set $n = 0$.
2. Set $j = 0$.
3. Determine the rank r of $\mathbf{x}_j$ using the results of Lemmas 6.1 and 6.2, and compute the local MAE (LMAE) corresponding to selecting $\mathbf{x}_j$ as the output of the ROF at

the window-position n, i.e.,

$$\mathrm{LMAE} = |\,\mathrm{S}(n) - \mathbf{x}_j\,| \ .$$

4. Update the global MAE associated with r, i.e.,

$$\mathrm{MAE}(r) = \mathrm{MAE}(r) + \mathrm{LMAE}\,.$$

5. While $j < \mathrm{M}$, set $j = j + 1$ and go to Step 2.
6. While n is smaller than the size of the training set, let $n = n + 1$ and go to Step 2.
7. Set $r = 1$ and ROF $= 0$.
8. If MAE(ROF) $>$ MAE(r), then set ROF $= r$.
9. While $r < \mathrm{M}$, set $r = r + 1$ and go to Step 8. Otherwise stop.

Using the above procedure, a much more efficient scheduling of the operations of summation involved in updating the MAE is achieved. The possibility of using the WSC architecture of Chapter 5 for the development of an efficient configuration for the MAE design of locally optimal ROFs is investigated in the next section.

6.2 An Architecture for the MAE Design of Locally Optimal ROFs

An architecture suitable for an efficient VLSI implementation of the MAE design of locally optimal ROFs is illustrated in Figure 6.1. The datapath consists of M processing units (PUs), each comprising a register-block REG with a tri-state bus-driver at the output and a comparator COMP. A register-block REG_j consists of three register-fields denoted by X, B, and Z. An original sample-value $\mathbf{x}_j$ is loaded into the register-field X of REG_j, while the latch $\mathrm{B}(\mathrm{REG}_j)$ is used for storing the binary result of the comparison which is caried out in COMP_j. Let us denote by $\mathbf{x} = \{\mathbf{x}_0^*, \mathbf{x}_1^* \ldots, \mathbf{x}_i^*, \ldots, \mathbf{x}_{\tilde{\mathrm{M}}-1}^*\}$ a set consisting of the $\tilde{\mathrm{M}}$ distinct elements of $\mathbf{x}(\mathrm{n})$. The register-field $\mathrm{Z}(\mathrm{REG}_j)$ is used to provide a mechanism to determine the rank of the element $\mathbf{x}_j = \mathbf{x}_i^*$ when there are several elements in $\mathbf{x}(\mathrm{n})$ assuming the same value $\mathbf{x}_i^*$. This mechanism is required in view of the result of Lemma 6.2. Specifically,

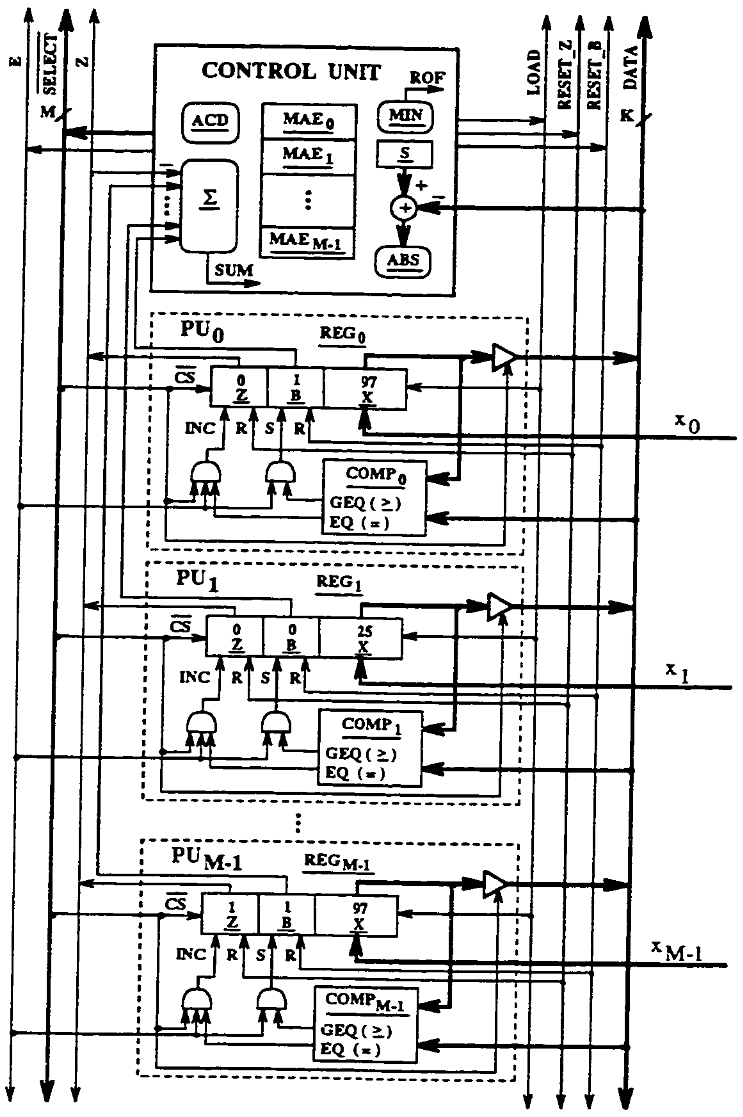


Figure 6.1: Block diagram of the proposed architecture for the MAE design of locally optimal rank order filters.

when the elements of $\mathbf{x}(n)$ are scanned through from left to right, and an element $\mathbf{x}_j = \mathbf{x}_i^*$, for which there are other Z_i elements assuming the same value $\mathbf{x}_i^*$, is encountered, then the register fields Z corresponding to all these $Z_i + 1$ elements of $\mathbf{x}(n)$ are incremented. At each iteration j, $j = 0, 1, \ldots, M-1$, the inputs to a comparator $\mathrm{COMP_k}$ are given by $\mathbf{x}_j$ and $\mathbf{x}_k$. As a result, at iteration j, the binary set

$$\delta(\mathbf{x}(n) - \mathbf{x}_j) = \{\, B(\mathrm{REG}_k) \mid k = 0, 1, \ldots, M-1 \,\} \tag{6.2}$$

represents the threshold sequence corresponding to the element $\mathbf{x}_j$ of $\mathbf{x}(n)$. The rank of $\mathbf{x}_j$ is given by the difference between the total number of 1's in the threshold sequence given by (6.2) and $Z(\mathrm{REG}_j)$.

The control unit generates the sequence of operations required for the MAE design of a locally optimal rank order filter. It comprises an address counter/decoder block ACD, an adder Σ, and a sub-block to determine the MMAE ROF for estimating a signal $S(n)$, given a training sequence $X(n)$. The sub-block consists of a memory-block $\mathrm{MAE_0}$, $\mathrm{MAE_1}$, $\ldots$, $\mathrm{MAE_{M-1}}$, a register S, and the operational blocks MIN, ABS, and "+". The ACD block is used for the selection of a processing unit PU required to place the content of its register-field X on the DATA-bus during a certain iteration j. The rank of an element $\mathbf{x}_j$ is determined by employing the adder Σ whose output is represented as SUM. The rank is then used for decoding the address of the MAE storage-location (i.e., $\mathrm{MAE_{SUM}}$) which is adjusted at iteration j. Specifically, at iteration j, the content of $\mathrm{MAE_{SUM}}$ is increased by the amount $\mathrm{ABS}(S - \mathbf{x}_j)$, where ABS denotes the operation of taking the absolute value. The result of the design procedure (i.e., a locally optimal ROF) is given by the lowest address of an MAE storage-location whose content is minimum over the entire memory-block $\mathrm{MAE_0}$, $\mathrm{MAE_1}$, $\ldots$, $\mathrm{MAE_{M-1}}$, i.e.,

$$\mathrm{ROF} = \min \{\, r \mid \mathrm{MAE}_r = \min \{\mathrm{MAE_0}, \mathrm{MAE_1}, \ldots, \mathrm{MAE_{M-1}}\,\}\,\}\,. \tag{6.3}$$

The dataflow corresponding to the architecture of Figure 6.1 for the MAE design of locally optimal ROFs can be formalized in the form of the following algorithms, using the conventional pseudocode notations.

Algorithm 6.1 *Algorithm for Designing Locally Optimal MAE ROFs*

```
For $k = 0, 1, \ldots, \mathrm{M}-1$ do in parallel Set $\mathrm{MAE}_k = 0$;
Set $\overline{\mathrm{SELECT}} = 1$; Set $\mathrm{E} = 0$; Set $\mathrm{LOAD} = 0$;
Set $\mathrm{RESET\_Z} = 0$; Set $\mathrm{RESET\_B} = 0$;
For $n = 0, 1, \ldots, \mathrm{N}-1$ do sequentially
    Set $\mathrm{RESET\_Z} = 1$; Set $\mathrm{LOAD} = 1$;
    Set $\mathrm{RESET\_Z} = 0$; Set $\mathrm{LOAD} = 0$;
    For $j = 0, 1, \ldots, \mathrm{M}-1$ do sequentially
        Set $\overline{\mathrm{CS}}(\mathrm{REG}_j) = 0$; Set $\mathrm{RESET\_B} = 1$;
        Set $\mathrm{RESET\_B} = 0$; Set $\mathrm{E} = 1$;
        For $k = 0, 1, \ldots, \mathrm{M}-1$ do in parallel
            If $\mathrm{GEQ}(\mathrm{COMP}_k) = 1$ then set $\mathrm{B}(\mathrm{REG}_k) = 1$;
            If $\mathrm{EQ}(\mathrm{COMP}_k) = 1$ and $\overline{\mathrm{CS}}(\mathrm{REG}_k) = 1$
                then set $\mathrm{Z}(\mathrm{REG}_k) = \mathrm{Z}(\mathrm{REG}_k) + 1$;
        End
        Set $\mathrm{MAE}_{\mathrm{SUM}} = \mathrm{MAE}_{\mathrm{SUM}} + \mathrm{ABS}(\mathrm{S} - \mathrm{X}(\mathrm{REG}_j))$;
        Set $\overline{\mathrm{CS}}(\mathrm{REG}_j) = 1$; Set $\mathrm{E} = 0$;
    End
End
Set $\mathrm{ROF} = \min\{ r \mid \mathrm{MAE}_r = \min\{\mathrm{MAE}_0, \mathrm{MAE}_1, \ldots, \mathrm{MAE}_{\mathrm{M}-1}\}\}$ ;
```

As previously stated, with the proposed architecture, the MAE design of ROFs is accomplished without carying out the operation of sorting the samples of the sets $\mathbf{x}(n)$, $n = 0, 1, \ldots, \mathrm{N}-1$. Specifically, at each iteration j, the rank of the element x_j is directly determined by using the results of Lemmas 6.1 and 6.2. As an example, by considering a specific window-sequence $\mathbf{x}(\mathrm{n}) = \{97, 25, 97\}$, the ranks of the samples $\mathrm{x}_0 = 97$, $\mathrm{x}_1 = 25$ and $\mathrm{x}_2 = 97$ are easily determined by employing Algorithm 6.1, as being 2, 3 and 1, respectively. The contents of all register-fields after the step $j = 2$ of the algorithm are indicated in Figure 6.1.

The idea of using locally optimal MAE ROFs for signal estimation can be employed in solving a variety of problems in image and image-sequence processing. Therefore, many 2-D and 3-D signal processing applications can benefit from the availability of the proposed architecture of MMAE ROFs, which is computationally very efficient and yet suitable for low-cost implementation. In the next section, the possibility of using the proposed architecture for the problem of video-signal deinterlacing through spatial interpolation is studied. In this context, the MMAE ROFs can find immediate commercial applications with the emerging advanced digital television systems.

6.3 Application to the Deinterlacing of Video Signals

In the subsequent, a technique that uses a rank-order filter-bank for intrafield deinterlacing of video signals is introduced, and an investigation of its performance is carried out by means of a simulation. In the proposed technique, an image-field (interlaced image) is divided into overlapping blocks of 6×6 pixels, as shown in Figure 6.2(a). Then, an MMAE design procedure in which the samples appearing in a filter window W are used to estimate a sample S, is applied to each of these blocks. As a result, an ROF which is locally optimal for signal estimation over each individual block of 6×6 pixels is determined. By employing this rank-order filter-bank, a deinterlaced image-frame is obtained from the given field by spatial interpolation. Specifically, as shown in Figure 6.2(b), a block of 8×4 pixels of the deinterlaced image-frame is obtained by using the lines of an original block of 4×4 pixels of the given field and by employing the MMAE ROF corresponding to this 4×4 block to interpolate the missing lines in the 8×4 block of the image frame.

An investigation of the performance of the proposed interpolation technique, in terms of the subjective quality of a deinterlaced image-sequence, has been carried out. An illustration of the performance of the proposed deinterlacing technique in the space domain is shown in Figure 6.3. Figure 6.3(a) depicts an interlaced image-frame of 170×160 pixels consisting of two consecutive fields, each of 170×80 pixels. A very annoying artifact can be observed along the vertical lines, due to the horizontal movement of the camera. Figure 6.3(b) shows the result obtained by using one of the conventional deinterlacing

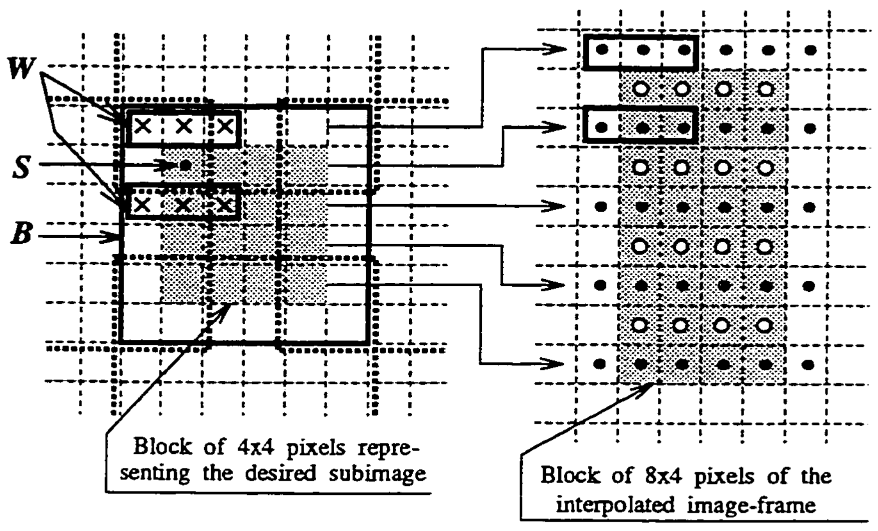

W
S
B
Block of 4x4 pixels representing the desired subimage
Block of 8x4 pixels of the interpolated image-frame
(a)
(b)

Figure 6.2: The proposed deinterlacing scheme. (a) A block B of 6×6 pixels of the given image field (interlaced image), which is used as the training set for the MMAE design of a locally optimal ROF. (b) A block of 8×4 pixels of the deinterlaced image frame.

procedures, namely, the line-doubling [44]. As it can be noticed, this deinterlacing method introduces a disturbing artifact along the horizontal lines. Figure 6.3(c) illustrates the result obtained by using the proposed technique for spatial interpolation using the locally optimal MAE ROFs introduced in this chapter. From this figure, it is observed that the spatial continuity of lines and contours of the image is much better preserved, as compared to the images of Figures 6.3(a) and 6.3(b).

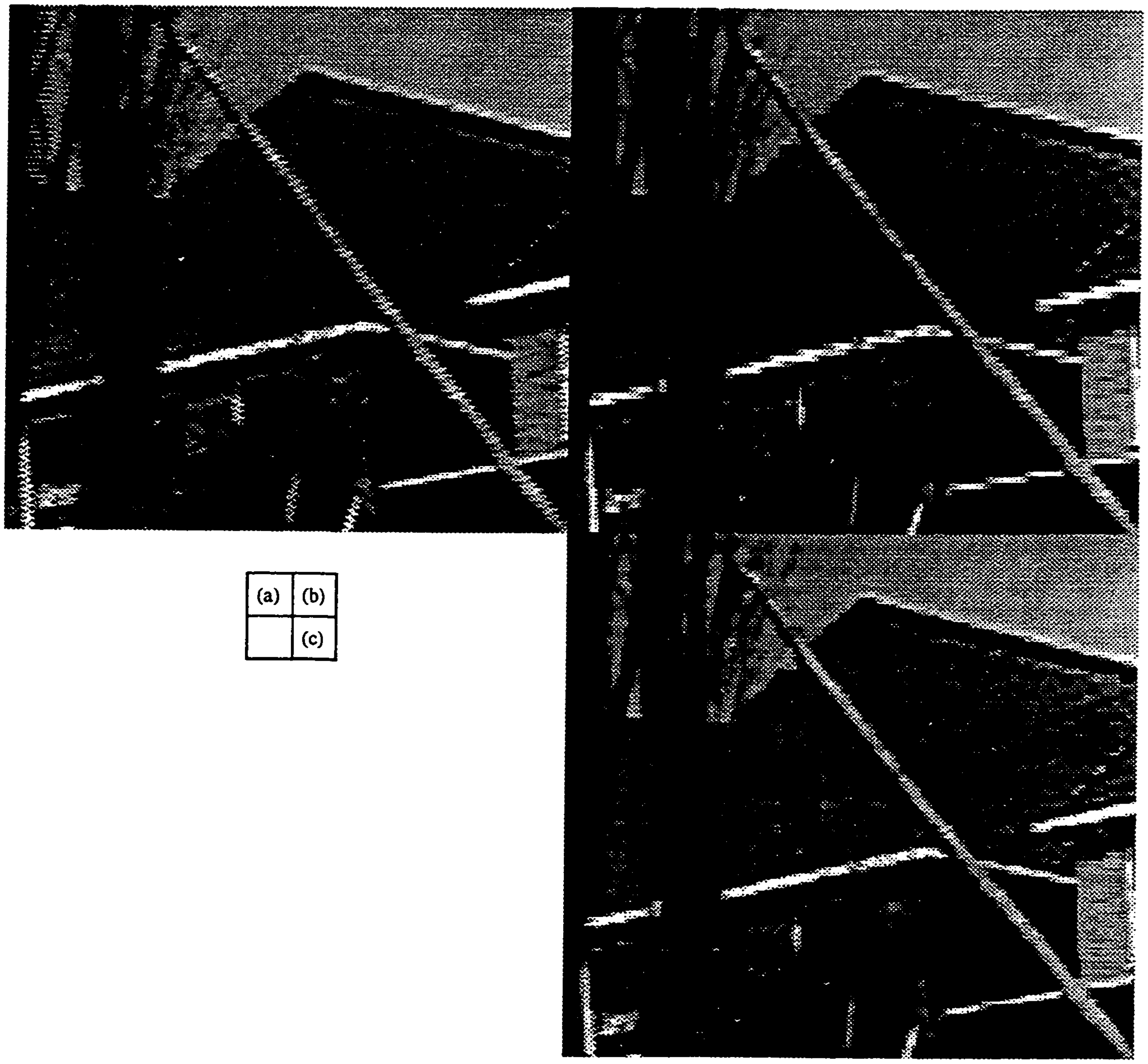


Figure 6.3: An illustration of the performance of the proposed deinterlacing technique. (a) Interlaced image-frame of 170 × 160 pixels consisting of two consecutive fields of 170 × 80 pixels. (b) Deinterlaced image-frame obtained from a field of 170 × 80 pixels by line-doubling. (c) Deinterlaced image-frame obtained by the application of the proposed technique for spatial interpolation using locally optimal MAE ROFs.

6.4 Summary

In this chapter, motivated by the fact that, in typical images, the details of the scene are locally situated, a hardware-oriented design of locally optimal MAE ROFs has been developed. It has been shown that the Boolean rank order operator associated with the ROF does not require sorting. As a result, the proposed design technique offers improved efficiency compared to employing a conventional multilevel configuration of rank order filter for the MMAE design and implementation of ROFs. It has also been shown that the proposed design can be conveniently implemented in hardware by employing a slightly modified version of the WSC architecture of Chapter 3. An application of the proposed design and implementation of locally optimal MAE ROFs to spatial interpolation has been investigated, by developing a procedure for intrafield deinterlacing of video signals. The proposed deinterlacing technique can be successfully used in low-cost commercial applications for digital television systems.

Chapter 7

Conclusion

7.1 Concluding Remarks

Motivated by the recent successes of using stack filters in image processing applications, in this investigation, we have been concerned with the problems of efficient design and implementation of stack filters. The focus has been on exploring the possibility of using the L_p-norm optimality criterion for the design of stack filters and on the development of hardware-oriented algorithms for the implementation of the filters designed using this optimality criterion.

The problem of designing optimal generalized stack filters by employing an L_p norm of the error between the desired signal and the estimated one has been solved. It has been shown that the L_p norm can be expressed as a linear function of the decision errors at the binary levels of the filter. As a result, an L_p-optimal stack filter can be determined as the solution of a linear program. The conventional design of using the mean absolute error (MAE) criterion, therefore, becomes a special case of the general L_p norm-based design developed in this thesis. Other special cases of the proposed approach, the problems of optimal mean square error ($p = 2$) and minimax ($p \to \infty$) stack filtering, that are of particular interest in signal processing, have also been discussed. The conventional MAE design of an important subclass of stack filters, the weighted order statistic filters, has been extended to the L_p norm-based design. By considering a typical application of restoring images corrupted with impulsive noise, several design examples have been presented, to illustrate the performance of the L_p-optimal stack filters with different values of p. Simulation results have shown that the L_p-optimal stack filters with $p \geq 2$ provide a better performance in terms of their capability in removing impulsive noise, compared to that achieved by using the conventional minimum MAE stack filters.

A new algorithm designated as the bit-serial window-partitioning (BSWP) algorithm for stack filtering has been developed. In this algorithm, the threshold adjustment is carried out by using both the result of the operation of Boolean filtering and the sequence of binary threshold values. An analysis of the computational complexity of the BSWP algorithm has shown that the average number of iterations required for its implementation is $\mathcal{O}(\log_2 M)$ per output sample, where M is the size of the filter. It has been shown that in those signal processing applications where M < L (L being the number of grey levels), considerable improvements in terms of the time-area complexity can be achieved by employing the proposed BSWP algorithm instead of the bit-serial binary tree search (BTS) approach of [13]. Through experimental investigation, it has been shown that these improvements are in the order of 20% to 45% for filters of sizes 3×3, 5×5 and 7×7. An architecture suitable for VLSI implementation for the proposed BSWP algorithm has been also developed.

A window-sequence coding-based BTS (WSC-BTS) algorithm for 2-D stack filtering has been developed. The algorithm uses a binary-tree search method for stack filtering in conjunction with a new technique for encoding the samples appearing in the input window of a stack filter. This encoding technique, which has been designated as a window-sequence coding (WSC) transformation, takes advantage of the observation that in typical images, many pixels appearing in the filter-window at a certain time-instant assume non-distinct values. It has been demonstrated that the proposed WSC-based BTS algorithm is functionally equivalent to the operation of stack filtering. An architecture implementing the WSC-BTS algorithm has also been developed. A comparison between the computational efficiency of the rank order state machine (ROSM)-based BTS architecture [1] and that of the proposed WSC-BTS architecture for 2-D stack filtering has shown that with the proposed approach, significant improvements in terms of the computational-time can be achieved, without increasing the chip-area. It has been observed that for typical images, using 3×3 filters, computational-improvements of at least 25% can be easily obtained for L = 256, while the improvements can easily be increased to 45% for L = 64.

Motivated by the fact that in typical images, the details of the scene are locally situated, a hardware-oriented design of locally optimal mean absolute error rank order filters has been proposed. It has been shown that since a Boolean rank order operator does not require sorting, the proposed algorithm achieves an improved computational efficiency over the technique of employing the conventional multilevel configurations of a rank order filter (ROF). The application of the proposed algorithm for the design of locally optimal mean absolute error ROFs to spatial interpolation has been investigated, by developing a procedure for intrafield deinterlacing of video signals. The technique developed has been found to be superior compared to the line-doubling technique in preserving the spatial continuity of lines and contours.

7.2 Scope for Future Investigation

An important contribution of this thesis has been the development of a general, L_p-norm-based theory for signal estimation using stack filters. The results of this theory have been verified in the context of a typical application of restoring images corrupted with impulsive noise. However, it is expected that the L_p-norm based theory for optimal stack filtering could find very interesting applications in solving real-life engineering problems. For instance, the minimum MSE stack filters could find immediate applications in wireless communication systems. Specifically, the use of linear predictors in conjunction with non-linear stack filter-based estimators could provide efficient coding performance along with a robust behavior in the presence of impulsive noise. Another important domain where the L_p norm stack filters could be very helpful is the area of low-level computer vision. It is expected that the L_p-norm optimal stack filters could find interesting applications in feature extraction and in shape representation and description.

In light of the results of Chapters 4 and 5, it is expected that an investigation on the implementation of specific multidimensional rank order operators and morphological filters using the BSWP and WSC-BTS stack filter structures could provide very attractive solutions. It is known that 1-D rank order filters enjoy very fast implementations based on a technique of maintaining the window as a sorted list [67]. However, in the case of multidimensional ROFs, the solution of employing a sorted list requires a different and more complex hardware implementation [33]. On the other hand, a stack filter implementation of ROFs does not depend on the dimension of an input signal (e.g., the same architecture can be used for the implementation of a 1×9 or a 3×3 filter). Nevertheless, the increased computational complexity of the conventional configurations for stack filters have made the solution of implementing ROFs as stack filters appear very inefficient. Thus, in view of our results in improving the computational efficiency of stack filtering, the solution of realizing ROFs using a threshold configuration architecture could provide a versatile and yet an efficient alternative for the implementation of ROFs. The implementation of ROFs could also benefit from the possible new sorting networks using L_p comparators. This would complement the existing solutions that use max-min sorting networks [9] and nonlinear L_p mean comparators [58].

Stack filtering has long been known to be closely related to morphological signal processing [21], [36], [51], [68]. Recently, in [41] and [40], it has been shown that gray scale morphological opening and closing operations can be accomplished by using a bit-serial BTS architecture for stack filtering. In this architecture, a suitable operation of positive Boolean filtering is applied at each level, preceded by an associated bit-modification operation. The operations of morphological opening and closing could benefit from a redesigned version of bit-modification logic so that the BSWP algorithm may be used in these operations.

Finally, it should be pointed out that the theory of stack filters is still in its infancy. Increased research efforts are needed in this area to refine the theory itself and to apply it to more practical problems. To this end, the research efforts of this investigation could be expected to provide a basis for developing nonlinear processing techniques as powerful alternatives to the theory of linear filters.

References

[1] G. B. Adams, E. J. Coyle, L. Lin, L. E. Lucke, and K. K. Parhi. Input Compression and Efficient VLSI Architectures for Rank Order and Stack Filters. *Signal Processing*, 38:441–453, 1994.

[2] M. O. Ahmad, M. N. S. Swamy, Q. S. Gu, and C. E. Savin. Multidimensional Filtering and Applications. *Proceedings of the Micronet Annual Workshop*, pages 3.18–3.35, 1995.

[3] M. O. Ahmad, M. N. S. Swamy, and C. E. Savin. A Hardware-Oriented Algorithm and Architecture for Stack Filtering. *Proceedings of the Micronet Annual Workshop*, pages 63–64, 1996.

[4] M. O. Ahmad, M. N. S. Swamy, and C. E. Savin. Lp Norm Design of Stack Filters and its Application to Image Restoration. *Proceedings of the Micronet Annual Workshop*, pages 61–62, 1997.

[5] D. A. Akopian, O. Vainio, S. S. Agaian, and J. T. Astola. SBNR Processor for Stack Filters. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, 44(3):197–208, March 1997.

[6] A. Antoniou. *Digital Filters: Analysis, Design and Applications.* McGraw-Hill, New York, 1993.

[7] G. R. Arce. Multistage Order Statistic Filters for Image Sequence Processing. *IEEE Transactions on Signal Processing*, 39(5):1146–1163, May 1991.

[8] K. E. Barner and G. R. Arce. Permutation Filters: A Class of Nonlinear Filters Based on Set Permutations. *IEEE Transactions on Signal Processing*, 42(4):782–798, April 1994.

[9] C. G. Boncelet. Recursive Algorithm and VLSI Implementations for Median Filtering. *Proceedings of the International Symposium on Circuits and Systems*, pages 1745–1747, June 1988.

[10] A. C. Bovik. Streaking in Median Filtered Images. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(4):493–503, April 1987.

[11] A. C. Bovik, T. S. Huang, and D. C. Munson, Jr. A Generalization of Median Filtering Using Linear Combinations of Order Statistics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31(6):1342–1350, December 1983.

[12] A. C. Bovik, T. S. Huang, and D. C. Munson, Jr. The Effect of Median Filtering on Edge Estimation and Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):181–194, March 1987.

[13] K. Chen. Bit-Serial Realizations of a Class of Nonlinear Filters Based on Positive Boolean Functions. *IEEE Transactions on Circuits and Systems*, 36(6):785–794, June 1989.

[14] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Mass., 1990.

[15] E. J. Coyle. Rank Order Operators and the Mean Absolute Error Criterion. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36:63–76, January 1988.

[16] E. J. Coyle. Stack Filters in Signal and Image Processing. *ISCAS-94 Tutorials*, pages 22–39, May 1994.

[17] E. J. Coyle and J. H. Lin. Stack Filters and the Mean Absolute Error Criterion. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(8):1244–1254, August 1988.

[18] E. J. Coyle, J. H. Lin, and M. Gabbouj. Optimal Stack Filtering and the Estimation and Structural Approaches to Image Processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:2037–2066, December 1989.

[19] H. A. David. *Order Statistics*. John Wiley & Sons, New York, 1970.

[20] K. K. Delibasis, P. E. Undrill, and G. G. Cameron. Genetic Algorithm Implementation of Stack Filter Design for Image Restoration. *IEE Proceedings - Vision, Image and Signal Processing*, 143(3):177–183, June 1996.

[21] O. Egger, W. Li, and M. Kunt. High Compression Image Coding Using an Adaptive Morphological Subband Decomposition. *Proc. IEEE*, 83:272–287, February 1995.

[22] J. P. Fitch. Software and VLSI Algorithms for Generalized Rank Order Filtering. *IEEE Transactions on Circuits and Systems*, 34(5):553–559, May 1987.

[23] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, Jr. Median Filtering by Threshold Decomposition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-32(6):1183–1188, December 1984.

[24] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, Jr. Threshold Decomposition of Multidimensional Ranked Order Operations. *IEEE Transactions on Circuits and Systems*, CAS-32(5):445–450, May 1985.

[25] M. Gabbouj. *Estimation and Structural-Based Approach for the Design of Optimal Stack Filters*. PhD thesis, Purdue University, West Lafayette, IN, December 1989.

[26] M. Gabbouj and E. J. Coyle. Minimum Mean Absolute Error Stack Filtering with Structural Constraints and Goals. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(6):955–968, June 1990.

[27] M. Gabbouj and E. J. Coyle. On the LP which Finds an MMAE Stack Filter. *IEEE Transactions on Signal Processing*, 39(11):2419–2424, November 1991.

[28] M. Gabbouj and E. J. Coyle. Minimax Stack Filtering in a Parametrized Environment. *Proceedings of the International Symposium on Circuits and Systems*, pages 97–100, May 1992.

[29] M. Gabbouj, E. J. Coyle, and N. C. Gallagher, Jr. An Overview of Median and Stack Filtering. *Circuits, Systems and Signal Process.*, 11(1):7–45, January 1992.

[30] N. C. Gallagher, Jr. and G. L. Wise. A Theoretical Analysis of the Properties of Median Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1136–1141, December 1981.

[31] D. Z. Gevorkian, K. O. Egiazarian, S. S. Agaian, J. T. Astola, and O. Vainio. Parallel Algorithms and VLSI Architectures for Stack Filtering Using Fibonacci p-Codes. *IEEE Transactions on Signal Processing*, 43(1):286–295, January 1995.

[32] E. N. Gilbert. Lattice Theoretic Properties of Frontal Switching Functions. *Journal of Mathematical Physics*, 33:57–67, April 1954.

[33] M. R. Hakami, P. J. Warter, and C. G. Boncelet. A New VLSI Architecture Suitable for Multidimensional Order Statistic Filtering. *IEEE Transactions on Signal Processing*, 42(4):991–993, April 1994.

[34] O. Y. Harja, J. T. Astola, and Y. A. Neuvo. Analysis of the Properties of Median and Weighted Median Filters Using Threshold Logic and Stack Filter Representation. *IEEE Transactions on Signal Processing*, 39(2):395–410, February 1991.

[35] P. J. Huber. *Robust Statistics.* John Wiley & Sons, New York, 1981.

[36] H. Hwang and R. A. Haddad. Multilevel Nonlinear Filters for Edge Detection and Noise Suppression. *IEEE Transactions on Signal Processing*, 42(2):249–258, February 1994.

[37] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[38] N. S. Jayant. Average and Median-Based Smoothing Techniques for Improving Digital Speech Quality in the Presence of Transmission Errors. *IEEE Transactions on Communications*, 24(9):1043–1045, September 1976.

[39] B. I. Justusson. *Median Filtering: Statistical Properties.* In Two-Dimensional Digital Signal Processing II, by T. S. Huang (Editor). Springer-Verlag, 1981.

[40] S. J. Ko, A. Morales, and K. H. Lee. A Fast Implementation Algorithm and a Bit-Serial Realization Method for Grayscale Morphological Opening and Closing. *IEEE Transactions on Signal Processing*, 43(12):3058–3061, December 1995.

[41] S. J. Ko, A. Morales, and K. H. Lee. Block Basis Matrix Implementation of the Morphological Open-Closing and Close-Opening. *IEEE Signal Processing Letters*, 2(1):7–9, January 1995.

[42] J. Lee and V. J. Mathews. A Fast, Recursive Least-Squares Second-Order Volterra Filter and its Performance Analysis. *IEEE Transactions on Signal Processing*, 41:1087–1102, March 1993.

[43] M. D. Levine. *Vision in Man and Machine.* McGraw-Hill, New York, 1985.

[44] J. S. Lim. *Two-Dimensional Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1990.

[45] J. H. Lin and E. J. Coyle. Minimum Mean Absolute Error Estimation Over the Class of Generalized Stack Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(4):663–678, April 1990.

[46] J. H. Lin and Y. T. Kim. Fast Algorithms for Training Stack Filters. *IEEE Transactions on Signal Processing*, 42(4):772–781, April 1994.

[47] J. H. Lin, T. M. Sellke, and E. J. Coyle. Adaptive Stack Filtering Under the Mean Absolute Error Criterion. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(6):938–954, June 1990.

[48] H. G. Longbotham and A. C. Bovik. Theory of Order Statistic Filters and Their Relationship to Linear FIR Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(2):275–287, February 1989.

[49] L. Lucke and K. Parhi. Parallel Processing Architectures for Rank Order and Stack Filters. *IEEE Transactions on Signal Processing*, 42:1178–1189, May 1994.

[50] P. Maragos, R. W. Schafer, and M. A. Butt (Eds.). *Mathematical Morphology and Its Applications to Image and Signal Processing.* Kluwer Academic Publishers, Norwell, Massachusetts, 1996.

[51] P. A. Maragos and R. W. Schafer. Morphological Filters - Part II: Their Relations to Median, Order-Statistic, and Stack Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35:1170–1184, August 1987.

[52] V. J. Mathews. Adaptive Polynomial Filters. *IEEE Signal Processing Magazine*, 8(3):10–26, July 1991.

[53] S. Muroga. *Threshold Logic and Its Applications.* John Wiley & Sons, New York, 1971.

[54] J. Nieweglowski, M. Gabbouj, and Y. Neuvo. Weighted Medians - Positive Boolean Functions Conversion. *Signal Processing*, 34(2), November 1993.

[55] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

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

[57] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, New York, third edition, 1991.

[58] M. Pappas and I. Pitas. Sorting Networks Using Nonlinear L_p Mean Comparators. *Proceedings of the International Symposium on Circuits and Systems*, pages 1–4, 1996.

[59] D. Petrescu, I. Tabus, and M. Gabbouj. Prediction Based on Boolean, FIR-Boolean Hybrid and Stack Filters for Lossless Image Coding. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 2965–2968, April 1997.

[60] I. Pitas and A. N. Venetsanopoulos. *Nonlinear Digital Filters: Principles and Applications.* Kluwer Academic Publishers, Norwell, Massachusetts, 1990.

[61] W. K. Pratt. *Digital Image Processing.* John Wiley & Sons, New York, 1991.

[62] W. V. Quine. Two Theorems About Truth Functions. *Boletin Sociedad Matematica*, Y:64–70, March 1953.

[63] N. Rama Murthy and M. N. S. Swamy. A VLSI Architecture for the Implementation of Real-Time Order Statistic Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 4(2):2096–2099, June 1991.

[64] H. Rantanen, M. Karlsson, P. Pohjala, and S. Kalli. Color Video Signal Processing with Median Filters. *IEEE Transactions on Consumer Electronics*, 38:157–161, August 1992.

[65] J. A. Rea, H. G. Longbotham, and H. N. Kothari. Fuzzy Logic and Mathematical Morphology: Implementation by Stack Filters. *IEEE Transactions on Signal Processing*, 44:142–147, January 1996.

[66] P. A. Regalia. *Special Filter Design.* In Handbook for Digital Signal Processing, by S. K. Mitra and J. F. Kaiser (Editors). John Wiley & Sons, New York, 1993.

[67] D. S. Richards. VLSI Median Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38:145–153, January 1990.

[68] P. Salembier, L. Tores, F. Meyer, and C. Gu. Region-Based Video Coding Using Mathematical Morphology. *Proc. IEEE*, 83:843–857, June 1995.

[69] C. E. Savin. *Linearly Separable Stack-Like Architecture for the Design of Weighted Order Statistic Filters with Application in Image Processing.* M.A.Sc. Thesis, Concordia University, Montréal, Québec, Canada, September 1993.

[70] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. Lp Norm Design of Stack Filters. Submitted to *IEEE Transactions on Image Processing.*

[71] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. Design of Weighted Order Statistic Filters Using Linearly Separable Stack-Like Architecture. *Proceedings of the Midwest Symposium on Circuits and Systems*, pages 753–756, August 1994.

[72] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. A Window-Sequence Coding Transformation Suitable for Computationally-Efficient Bit-Serial Implementation of Stack Filters. *Proceedings of the Midwest Symposium on Circuits and Systems*, pages 704–707, August 1995.

[73] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. On the Derivation of the Linear Program which Finds a Minimum Mean Absolute Error Stack Filter. *Bulletin of the Polytechnic Institute of Iasi*, XLI (XLV):37–45, 1995.

[74] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. Bit-Serial Window Partitioning Algorithm for Stack Filtering. *Electronics Letters*, 32(15):1359–1361, July 1996.

[75] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. Minimum Mean Square Error Design of Stack Filters. *Proceedings of the Midwest Symposium on Circuits and Systems*, pages 644–647, August 1996.

[76] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. A General Framework for the Design of Stack Filters Using the Lp Norm Objective Function. To appear in *Proceedings of the Midwest Symposium on Circuits and Systems*, August 1997.

[77] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. A Modified Binary-Tree Search Architecture for Two-Dimensional Stack Filtering. To appear in *Signal Processing*, 61(1), 1997.

[78] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. Fast VLSI Architecture for Rank Order Based Filtering Using a Bit-Serial Window Partitioning Technique. To appear in *Proceedings of the Midwest Symposium on Circuits and Systems*, August 1997.

[79] C. E. Savin, M. O. Ahmad, and M. N. S. Swamy. Lp Norm Design of Weighted Order Statistic Filters. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 2369–2372, April 1997.

[80] D. L. Snyder. *Random Point Processes.* John Wiley & Sons, New York, 1975.

[81] I. Tabus, D. Petrescu, and M. Gabbouj. A Training Framework for Stack and Boolean Filtering - Fast Optimal Design Procedures and Robustness Case Study. *IEEE Transactions on Image Processing*, 5(6):809–826, June 1996.

[82] J. W. Tukey. Nonlinear (Nonsuperposable) Methods for Smoothing Data. *1974 EASCON (Conference Record)*, page 673 (abstract only), 1974.

[83] J. W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, Reading, Mass., 1977.

[84] S. G. Tyan. *Median Filtering: Deterministic Properties.* In Two-Dimensional Digital Signal Processing II, by T. S. Huang (Editor). Springer-Verlag, 1981.

[85] T. Viero, K. Öistämö, and Y. Neuvo. Three-Dimensional Median-Related Filters for Color Image Sequence Filtering. *IEEE Transactions on Circuits and Systems for Video Technology*, 4:129–142, April 1994.

[86] P. D. Wendt, E. J. Coyle, and N. C. Gallagher. Stack Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):898–911, August 1986.

[87] B. Widrow and S. D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1985.

[88] R. Yang, M. Gabbouj, and P. T. Yu. Parametric Analysis of Weighted Order Statistic Filters. *IEEE Signal Processing Letters*, 1(6):95–98, June 1994.

[89] R. Yang, L. Yin, M. Gabbouj, J. Astola, and Y. Neuvo. Optimal Weighted Median Filters Under Structural Constraints. *IEEE Transactions on Signal Processing*, pages 591–604, March 1995.

[90] L. Yin. Stack Filter Design: a Structural Approach. *IEEE Transactions on Signal Processing*, 43(4):831–840, April 1995.

[91] L. Yin, J. T. Astola, and Y. A. Neuvo. Adaptive Stack Filtering with Application to Image Processing. *IEEE Transactions on Signal Processing*, 41(1):162–184, January 1993.

[92] L. Yin and Y. A. Neuvo. Fast Adaptation and Performance Characteristics of FIR-WOS Hybrid Filters. *IEEE Transactions on Signal Processing*, 42(7):1610–1628, July 1994.

[93] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo. Weighted Median Filters: A Tutorial. *IEEE Transactions on Circuits and Systems*, 43(3):157–192, March 1996.

[94] J. Yoo, C. A. Bouman, E. J. Delp, and E. J. Coyle. The Nonlinear Prefiltering and Difference of Estimates Approach to Edge Detection: Applications of Stack Filters. *CVGIP: Graphical Models and Image Processing*, 55(2):140–159, March 1993.

[95] P. T. Yu. Some Representation Properties of Stack Filters. *IEEE Transactions on Signal Processing*, 40(9):2261–2266, September 1992.

[96] P. T. Yu and R. C. Chen. Fuzzy Stack Filters - Their Definitions, Fundamental Properties, and Application in Image Processing. *IEEE Transactions on Image Processing*, 5(6):838–854, June 1996.

[97] 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 Transactions on Circuits and Systems*, 38(9):1003–1020, September 1991.

Appendix A

Listing of Matlab Program to Determine the L_p Norm

```
function [Alpha,Beta] = lp_norm(p,S,X,Filter_size)
% [Alpha,Beta] = LP_NORM(p,X,S) calculates the coefficients Alpha and
%    Beta appearing in the binary-level expression of the Lp norm of
%    the error between the desired image S and its estimated version
%    using stack filters. The estimated version of S is to be obtained
%    by applying a stack filter characterized by an arbitrary PBF f to
%    the input image X. The image X is assumed to be a noise-corrupted
%    version of S. The value of p itself is expected to be specified
%    as an input to the function LP_NORM. Filter_size = [x_size y_size]
%    is a vector with two elements specifying the size of the filter.
%    For instance, in the case of a 3x3 filter, both x_size and y_size
%    are equal to 3. Similarly, for a 1x3 filter, the number of rows,
%    x_size, is equal to 1, and the number of columns, y_size, is equal
%    to 3. Note that a 1xM filter is simply a 1-D filter of size M.
% NOTE: the filters are expected to be rectangular, and BOTH x_size
%       AND y_size SHOULD BE ODD NUMBERS. At each window position,
%       the output of the filter is to replace the input pixel
%       situated at the center of the window, at its current position.
% FUNCTIONS CALLED: LP_NORM invokes the function BORDER, which adds
%       a suitable frame to the input matrix X: this frame represents
%       the initial conditions for applying the filtering operation at
%       filter positions situated on the margins of the input image X.
%
x_size = Filter_size(1);      % x_size - no. of rows of the filter window
y_size = Filter_size(2);      % y_size - no. of cols. of the filter window
M = x_size * y_size;          % M is the size of the binary vectors
Alpha = zeros(2^M,1);         % Initialize Alpha
Beta = zeros(2^M,1);          % Initialize Beta
% Determine the number of quantization levels.
GL = max( max(max(X)), max(max(S)) );   % There are GL+1 levels, i.e.,
                                        % 0,1,...,GL. However, the operation
                                        % of thresholding is carried out
                                        % only at the levels 1,2,...,GL.
% There is a pair of coefficients (Alpha(b_10+1),Beta(b_10+1)) corresponding
% to each binary vector b. The notation b_10 designates the base-10 value of
% the vector b interpreted as a binary number. Note that the coefficients
% Alpha and Beta corresponding to a vector b=[0 0 ... 0] are Alpha(1) and
% Beta(1), and NOT Alpha(0) and Beta(0) (in MATLAB, the indices start with 1
% and not with 0). To calculate the indices of the coefficients Alpha and
% Beta corresponding to an arbitrary threshold vector b, we use the following
% vector of "Powers-of-two":
Powers_of_2 = (2*ones(1,M)) .^ [M-1:-1:0];   % For instance, if M=4, then
                                             % Powers_of_2 = [8 4 2 1].
% Determine the size of the images S and X.
```

```
if size(S)~=size(X)
  error('The sizes of matrices S and X do not agree !'); end
[Nr Nc] = size(S);            % Nr(Nc) designates the no. of rows(cols.)
% Augment the input image by bordering the matrix X with a frame whose
% dimensions are determined in accordance with the size of the filter.
AX = border(X,Filter_size);
% Main computation
for i=1:Nr
   for j=1:Nc
      % In 2-D stack filtering, a 1-D set x is normally obtained
      % by concatenating the rows of the 2-D window sequence.
      x = reshape( AX(i:i+(x_size-1),j:j+(y_size-1)), 1, M );
      % Append 0 and GL to x, and sort this augmented vector Ax
      % in ascending order.
      Ax = [0 x GL]; Sx = sort(Ax);
      % Calculate the consecutive differences x(2)-x(1), x(3)-x(2),
      % ..., x(M+2)-x(M+1).
      Dx = diff(Sx);
      % Denote the total number of nonzero elements in Dx by nz,
      % and let I denote a vector whose elements are the indices of
      % the nonzero elements in Dx.
      nz = nnz(Dx); I = find(Dx);
      % For reasons of computational efficiency, at each window position
      % (i,j), the operation of thresholding is carried out only at
      % levels which are equal to distinct samples appearing in the
      % window sequence, i.e., at the levels Sx(I(1)+1), Sx(I(2)+1),
      % ..., Sx(I(nz)+1). As it is known, over each interval of threshold
      % levels Tl=[Sx(I(k))+1,Sx(I(k)+1)], with k=1,2,...,nz, the binary
```

```
        % threshold vector b remains unchanged: b=(x>=Sx(I(k)+1)).
        % The vector of binary signals appearing in the threshold decomposition
        % of S(i,j) over each interval of threshold values Tl is denoted by sl.
        for k=1:nz
           Tl = Sx(I(k))+1 : Sx(I(k)+1);
           b = (x>=Sx(I(k)+1));
           b_10 = b*(Powers_of_2)' + 1;
           sl = (S(i,j)>=Tl); sl_polar = 1-2*sl;
           Alpha(b_10) = Alpha(b_10) + ...
              sum( sl_polar .* abs( ((S(i,j)-Tl+1).^p) - ((S(i,j)-Tl).^p) ) );
           Beta(b_10) = Beta(b_10) + ...
              sum( sl .* abs( ((S(i,j)-Tl+1).^p) - ((S(i,j)-Tl).^p) ) );
        end
     end
  end
  Alpha = (1/(Nr*Nc))*Alpha;
  Beta = (1/(Nr*Nc))*Beta;
```

Appendix B

Listing of the Function to Generate Boundary Conditions for Images

```
function AX = border(X,Filter_size)
% AX = BORDER(X,Filter_size) borders the input image X with a frame
%    (margin) whose dimensions are calculated based on the Filter_size.
% Filter_size = [x_size y_size] is a vector with two elements specifying
%    the size of the filter. For instance, in the case of a 3x3 filter, both
%    x_size and y_size are equal to 3. Similarly, for a 1x3 filter, the
%    number of rows, x_size, is equal to 1, and the number of columns,
%    y_size, is equal to 3. Note that a 1xM filter is simply a 1-D filter
%    of size M.
% The border represents the initial conditions for applying a filtering
%    operation at filter positions situated on the margins of the input
%    image X. The border is composed of 8 rectangles designated as:
%    1) Left_Margin, 2) Right_Margin, 3) Top_Margin, 4) Bottom_Margin,
%    5) Upper_Left_Corner, 6) Upper_Right_Corner, 7) Lower_Left_Corner,
%    and 8) Lower_Right_Corner. Each of these margin-rectangles and
%    corner-rectangles are obtained by repeating suitably chosen pixels
%    appearing in the first and last (y_size-1)/2 columns and in the
%    first and last (x_size-1)/2 rows of the input image X.
% NOTE: the filters are expected to be rectangular, and BOTH x_size
%          AND y_size SHOULD BE ODD NUMBERS. At each window position,
%          the output of the filter is to replace the input pixel
%          situated at the center of the window, at its current position.
%
x_size = Filter_size(1);      % x_size - no. of rows of the filter window
y_size = Filter_size(2);      % y_size - no. of cols. of the filter window
[Nr Nc] = size(X);            % Nr(Nc) designates the no. of rows(cols.)
AX = zeros(Nr+x_size-1,Nc+y_size-1);    % Initialize the bordered image
AX( (x_size-1)/2+1:Nr+(x_size-1)/2 , (y_size-1)/2+1:Nc+(y_size-1)/2 ) = X;
% CONSTRUCT THE MARGIN-RECTANGLES
Top_Margin = [];   Bottom_Margin = [];
for i = 1:(x_size-1)/2
   Top_Margin = [ X(i,:) ; Top_Margin ];
   Bottom_Margin = [ Bottom_Margin ; X(Nr-i+1,:) ];
end
Columns = (y_size-1)/2+1:Nc+(y_size-1)/2;
AX( 1:(x_size-1)/2 , Columns ) = Top_Margin;
AX( Nr+(x_size-1)/2+1:Nr+(x_size-1) , Columns ) = Bottom_Margin;
Left_Margin = [];   Right_Margin = [];
for i = 1:(y_size-1)/2
   Left_Margin = [ X(:,i)  Left_Margin ];
   Right_Margin = [ Right_Margin  X(:,Nc-i+1) ];
end
```

```
Rows = (x_size-1)/2+1:Nr+(x_size-1)/2;

AX( Rows , 1:(y_size-1)/2 ) = Left_Margin;

AX( Rows , Nc+(y_size-1)/2+1:Nc+(y_size-1) ) = Right_Margin;

% CONSTRUCT THE CORNER-RECTANGLES

Upper_Left_Corner = X(1,1)*ones((x_size-1)/2,(y_size-1)/2);

AX( 1:(x_size-1)/2 , 1:(y_size-1)/2 ) = Upper_Left_Corner;

Upper_Right_Corner = X(1,Nc)*ones((x_size-1)/2,(y_size-1)/2);

AX( 1:(x_size-1)/2 , Nc+(y_size-1)/2+1:Nc+(y_size-1) ) = Upper_Right_Corner;

Lower_Left_Corner = X(Nr,1)*ones((x_size-1)/2,(y_size-1)/2);

AX( Nr+(x_size-1)/2+1:Nr+(x_size-1) , 1:(y_size-1)/2 ) = Lower_Left_Corner;

LRC = X(Nr,Nc)*ones((x_size-1)/2,(y_size-1)/2);    % LRC = Lower_Right_Corner

AX( Nr+(x_size-1)/2+1:Nr+(x_size-1) , Nc+(y_size-1)/2+1:Nc+(y_size-1) ) = LRC;
```

Appendix C

Listing of the Function to Verify the Correctness of the Program for L_p Norm

```
function [M_NORM,B_NORM] = vf_norm(p,S,r,Filter_size)
% [M_NORM,B_NORM] = VF_NORM(p,S,r,Filter_size) verifies whether or not
%   the operation of the function LP_NORM.M is error-free. The function
%   VF_NORM.M calculates the Lp norm of the error between a given test
%   signal S and its processed version using a rank order filter of
%   size Filter_size=[x_size y_size] and parameter "r". The specific Lp
%   norm which is to be calculated by VF_NORM.M is passed by the value
%   of the argument "p" (e.g., p=2 corresponds to the MSE). VF_NORM.M
%   calculates the Lp norm by using both a multilevel approach, and a
%   binary level technique employing the results of function LP_NORM.M :
%   1) In the multilevel approach, an operation of rank order filtering
%      is directly applied to the test signal (image) S. At each window
%      position, the output of this operation of rank order filtering is
%      determined by selecting the "r"-th largest element appearing in
%       the input window of size (x_size * y_size). Let X denote the pro-
%       cessed image. Then, at the multilevel, the Lp norm can be calcu-
%       lated by using the following formula:
%             M_NORM = (1/(Nr*Nc)) * sum(sum( (abs(S-X)).^p )),
%       where Nr and Nc denote the number of rows and columns of images
%       S and X.
%    2) At the binary level, the Lp norm can be calculated by using the
%       coefficients Alpha and Beta, and the binary level outputs of a
%       rank order filter of parameter "r". In this approach the coeffi-
%       cients Alpha and Beta are obtained by employing the function
%       LP_NORM.M. The value of the Lp norm obtained by this approach is
%       denoted by B_NORM.
%    By comparing the values of M_NORM and B_NORM for different arguments
%    p, S, r, and Filter_size, the user can thoroughly verify the opera-
%    tion of LP_NORM.M. Specifically, for any combination of input argu-
%    ments p, S, r, and Filter_size, one has to obtain the same value for
%    both M_NORM and B_NORM.
% FUNCTIONS CALLED: BORDER.M, LP_NORM.M
x_size = Filter_size(1);      % x_size - no. of rows of the filter window
y_size = Filter_size(2);      % y_size - no. of cols. of the filter window
M = x_size * y_size;          % M is the size of the binary vectors
[Nr Nc] = size(S);            % Nr(Nc) designates the no. of rows(cols.)
%--------------------------------------------------------------------------
% 1) THE  MULTILEVEL  APPROACH
%    Generate an image X by applying a rank order filter of parameter "r"
%    to the input image S. At each window position, the output of this
% ter of parameter "r"
%    to the input image S. At each window position, the output of this
%    filter is given by the r-th largest element in the window.
%---------------------------------------------------------------------------
% Augment the input image by bordering the matrix S with a frame whose
% dimensions are determined in accordance with the size of the filter.
AS = border(S,Filter_size);
X = zeros(Nr,Nc);        % Initialize the matrix X
for i=1:Nr
   for j=1:Nc
      % In 2-D stack filtering, a 1-D set x is normally obtained
      % by concatenating the rows of the 2-D window sequence.
      x = reshape( AS(i:i+(x_size-1),j:j+(y_size-1)), 1, M );
      % Sort the vector x in descending order.
      Sx = fliplr(sort(x));
      % The output of the filter at the position (i,j) is given
      % by the r-th element of Sx.
      X(i,j) = Sx(r);
   end
end
% Now, at the multilevel, the Lp norm can be calculated as follows:
M_NORM = (1/(Nr*Nc))*sum(sum( (abs(S-X)).^p ));
%---------------------------------------------------------------------------
% 2) THE  BINARY  LEVEL  APPROACH
%    Calculate the coefficients Alpha and Beta by employing the function
%    LP_NORM.M, and then calculate the Lp norm using the binary outputs
%    of a rank order filter of parameter "r". Note that the function
%    [Alpha,Beta] = lp_norm(p,S,X,Filter_size) should be invoked with the
%    arguments S and X assuming a common value : S.
%---------------------------------------------------------------------------
```

```
[Alpha,Beta] = lp_norm(p,S,S,Filter_size);
B_NORM = 0;               % Initialize B_NORM
for j=1:2^M
   % Convert (j-1) to binary
   b = zeros(1,M); d = j-1;
   for k=1:M
      b(k) = d-2*floor(d/2);
      d = floor(d/2);
   end
   b = fliplr(b);
   % Add Alpha(j)*f(b)+Beta(j) to the current value of B_NORM
   f = (sum(b)>=r);
   B_NORM = B_NORM + Alpha(j)*f + Beta(j);
end
```

Appendix D

Listing of Matlab Program to Implement the BSWP Algorithm

```
function FX = bswp_filter(PBF,X,Filter_size)
% FX = bswp_filter(PBF,X,Filter_size) applies an operation of stack
%    filtering characterized by the positive Boolean function PBF to
%    the image X, using the BSWP algorithm. PBF is a column vector
%    whose j-th entry represents the output of the characteristic PBF
%    when the binary input vector is determined by the base-2 value of
%    (j-1). Filter_size = [x_size y_size] is a vector with two elements
%    specifying the size of the stack filter.
% NOTE: the filters are expected to be rectangular, and BOTH x_size
%       AND y_size SHOULD BE ODD NUMBERS. At each window position,
%       the output of the filter is to replace the input pixel
%       situated at the center of the window, at its current position.
% FUNCTIONS CALLED: BORDER.M
x_size = Filter_size(1);      % x_size - no. of rows of the filter window
y_size = Filter_size(2);      % y_size - no. of cols. of the filter window
M = x_size * y_size;          % M is the size of the binary vectors
```

```
% To determine the output of the PBF corresponding to an arbitrary input
% given as a binary vector b, we use the following vector of "Powers-of-two":
Powers_of_2 = (2*ones(1,M)) .^ [M-1:-1:0];
% Augment the input image by bordering the matrix X with a frame whose
% dimensions are determined in accordance with the size of the filter.
AX = border(X,Filter_size);
% Determine the size of the input image X.
[Nr Nc] = size(X);              % Nr(Nc) designates the no. of rows(cols.)
FX = zeros(Nr,Nc);  % Initialize the matrix FX, representing the output image
% Main computation
for i=1:Nr
   for j=1:Nc
       xn = X(i:i+(x_size-1),j:j+(y_size-1)); x = reshape(xn,1,M);
       B = ones(1,M);
       for k=1:M
          if B(k)==1
             d = (x>=x(k)); d1 = (x>x(k)); f = PBF(d*Powers_of_2'+1);
             if (f==1) SF = x(k); end
             B = (((~f)*(~d))|(f*d1)).*B;
          end
       end
       FX(i,j) = SF;
   end
end
```