3.2

Nonlinear Filtering for Image Analysis and Enhancement

Gonzalo R. Arce,
Jan Bacca,
University of Delaware
José L. Paredes
University of Los Andes, Venezuela

1 Introduction

Digital image enhancement and analysis have played, and will continue to play, an important role in scientific, industrial, and military applications. In addition to these applications, image enhancement and analysis are increasingly being used in consumer electronics. Internet Web users, for instance, not only rely on built-in image processing protocols such as JPEG and interpolation, but also have become image processing users equipped with powerful yet inexpensive software such as Photoshop. Users not only retrieve digital images from the Web but are now able to acquire their own by use of digital cameras or through digitization services of standard 35 mm analog film. The end result is that consumers are beginning to use home computers to enhance and manipulate their own digital pictures. Image enhancement refers to processes seeking to improve the visual appearance of an image. As an example, image enhancement might be used to emphasize the edges within the image. This edge-enhanced image would be more visually pleasing to the naked eye, or perhaps could serve as an input to a machine that would detect the edges and perhaps make measurements of shape and size of the detected edges. Image enhancement is important because of its usefulness in virtually all image processing applications.

Image enhancement tools are often classified into (a) point operations, and (b) spatial operators. Point operations include contrast stretching, noise clipping, histogram modification, and pseudocoloring. Point operations are, in general, simple nonlinear operations that are well known in the image processing literature and are covered elsewhere in this *Handbook*. Spatial operations used in image processing today are, on the other hand, typically linear operations. The reason for this is that spatial linear operations are simple and easily implemented. Although linear image enhancement tools are often adequate in many applications, significant advantages in image enhancement can be attained if nonlinear techniques are applied [1]. Nonlinear methods effectively preserve edges and details of images while methods using linear operators tend to blur and distort them. Additionally, nonlinear image enhancement tools are less susceptible to noise. Noise is always present due to the physical randomness of image acquisition systems. For example, underexposure and low-light conditions in analog photography conditions lead to images with film-grain noise which, together with the image signal itself, are captured during the digitization process.

This article focuses on nonlinear and spatial image enhancement and analysis. The nonlinear tools described in

this article are easily implemented on currently available computers. Rather than using linear combinations of pixel values within a local window, these tools use the local weighted median. In Section 2, the principles of weighted medians (WM) are presented. Weighted medians have striking analogies with traditional linear FIR filters, yet their behavior is often markedly different. In Section 3, we show how WM filters can be easily used for noise removal. In particular, the center WM filter is described as a tunable filter highly effective in impulsive noise. Section 4 focuses on image enlargement, or zooming, using WM filter structures which, unlike standard linear interpolation methods, provide little edge degradation. Section 5 describes image sharpening algorithms based on WM filters. These methods offer significant advantages over traditional linear sharpening tools whenever noise is present in the underlying images. Section 6 goes beyond image enhancement and focuses on the analysis of images. In particular, edge detection methods based on WM filters are described as well as their advantages over traditional edge-detection algorithms.

2 Weighted Median Smoothers and Filters

2.1 Running Median Smoothers

The running median was first suggested as a nonlinear smoother for time series data by Tukey in 1974 [2]. To define the running median smoother, let $\{x(\cdot)\}$ be a discrete time sequence. The running median passes a window over the sequence $\{x(\cdot)\}$ that selects, at each instant n, a set of samples to comprise the observation vector $\mathbf{x}(n)$. The observation window is centered at n, resulting in

$$\mathbf{x}(n) = [x(n - N_L), \ldots, x(n), \ldots, x(n + N_R)]^T \quad (1)$$

where N_L and N_R may range in value over the nonnegative integers and $N = N_L + N_R + 1$ is the window size. The median smoother operating on the input sequence $\{x(\cdot)\}$ produces the output sequence $\{y\}$, where at time index n

$$y(n) = \text{MEDIAN}[x(n - N_L), \ldots, x(n), \ldots, x(n + N_R)] \quad (2)$$

$$= \text{MEDIAN}[x_1(n), \ldots, x_N(n)] \quad (3)$$

where $x_i(n) = x(n - N_L + 1 - i)$ for $i = 1, 2, \ldots, N$. That is, the samples in the observation window are sorted and the middle, or median, value is taken as the output. If $x_{(1)}, x_{(2)}, \ldots, x_{(N)}$ are the sorted samples in the observation window, the median smoother outputs

$$y(n) = \begin{cases} x_{(\frac{N+1}{2})} & \text{if } N \text{ is odd} \\ \frac{x_{(\frac{N}{2})} + x_{(\frac{N}{2}+1)}}{2} & \text{otherwise} \end{cases}. \quad (4)$$

In most cases, the window is symmetric about $x(n)$ and $N_L = N_R$.

The input sequence $\{x(\cdot)\}$ may be either finite or infinite in extent. For the finite case, the samples of $\{x(\cdot)\}$ can be indexed as $x(1), x(2), \ldots, x(L)$, where L is the length of the sequence. Due to the symmetric nature of the observation window, the window extends beyond a finite extent input sequence at both the beginning and end. These end effects are generally accounted for by appending N_L samples at the beginning and N_R samples at the end of $\{x(\cdot)\}$. Although the appended samples can be arbitrarily chosen, typically these are selected so that the points appended at the beginning of the sequence have the same value as the first signal point, and the points appended at the end of the sequence all have the value of the last signal point.

To illustrate the appending of input sequence and the median smoother operation, consider the input signal $\{x(\cdot)\}$ of Fig. 1. In this example, $\{x(\cdot)\}$ consists of 20 observations from a 6-level process, $\{x : x(n) \in \{0, 1, \ldots, 5\}, n = 1, 2, \ldots, 20\}$. The figure shows the input sequence and the resulting output sequence for a window size 5 median smoother. Note that to account for edge effects, two samples have been appended to both the beginning and end of the sequence. The median smoother output at the window location shown in the figure is

$$y(9) = \text{MEDIAN}[x(7), x(8), x(9), x(10), x(11)]$$

$$= \text{MEDIAN}[1, 1, 4, 3, 3] = 3.$$

Running medians can be extended to a recursive mode by replacing the "causal" input samples in the median

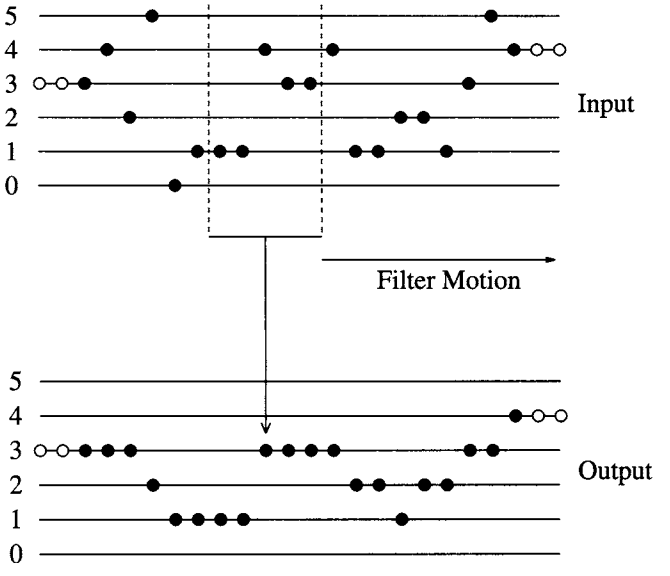

FIGURE 1 The operation of the window width 5 median smoother. ○: appended points.

smoother by previously derived output samples [3]. The output of the recursive median smoother is given by

$$y(n) = \text{MEDIAN}\,[y(n-NL), \ldots, y(n-1), x(n), \ldots, x(n+N_R)]. \quad (5)$$

In recursive median smoothing, the center sample in the observation window is modified before the window is moved to the next position. In this manner, the output at each window location replaces the old input value at the center of the window. With the same amount of operations, recursive median smoothers have better noise attenuation capabilities than their non-recursive counterparts [4, 5]. Alternatively, recursive median smoothers require smaller window lengths than their nonrecursive counterparts in order to attain a desired level of noise attenuation. Consequently, for the same level of noise attenuation, recursive median smoothers often yield less signal distortion. In image processing applications, the running median window spans a local 2 dimensional area. Typically, an $N \times N$ area is included in the observation window. The processing, however, is identical to the 1 dimensional case in the sense that the samples in the observation window are sorted and the middle value is taken as the output.

The running 1D or 2D median, at each instant in time, computes the sample median. The sample median, in many respects, resemble the sample mean. Given N samples $x_1, \ldots, x_N$ the sample mean, $\bar{X}$, and sample median, $\tilde{X}$, minimize the expression

$$G(\beta) = \sum_{i=1}^{N} |x_i - \beta|^p \quad (6)$$

for $p=2$ and $p=1$, respectively. Thus, the median of an odd number of samples emerges as the sample whose sum of absolute distances to all other samples in the set is the smallest. Likewise, the sample mean is given by the value β whose square distance to all samples in the set is the smallest possible. The analogy between the sample mean and median extends into the statistical domain of parameter estimation where it can be shown that the sample median is the Maximum Likelihood (ML) estimator of location of a constant parameter in Laplacian noise. Likewise, the sample mean is the ML estimator of location of a constant parameter in Gaussian noise [6]. This result has profound implications in signal processing, as most tasks where non-Gaussian noise is present will benefit from signal processing structures using medians, particularly when the noise statistics can be characterized by probability densities having heavier than Gaussian tails (which leads to noise with impulsive characteristics) [7–9].

2.2 Weighted Median Smoothers

Although the median is a robust estimator that possesses many optimality properties, the performance of running medians is limited by the fact that it is temporally blind. That is, all observation samples are treated equally regardless of their location within the observation window. Much like weights can be incorporated into the sample mean to form a weighted mean, a weighted median can be defined as the sample which minimizes the weighted cost function

$$G_p(\beta) = \sum_{i=1}^{N} W_i |x_i - \beta|^p \quad (7)$$

for $p=1$. For $p=2$, the cost function (7) is quadratic and the value β minimizing it is the normalized weighted mean

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} W_i (x_i - \beta)^2 = \frac{\sum_{i=1}^{N} W_i \cdot x_i}{\sum_{i=1}^{N} W_i} \quad (8)$$

with $W_i > 0$. For $p=1$, $G_1(\beta)$ is piecewise linear and convex for $W_i \geq 0$. The value β minimizing (7) is thus guaranteed to be one of the samples $x_1, x_2, \ldots, x_N$ and is referred to as the weighted median (WM), originally introduced over a hundred years ago by Edgemore [10]. After some algebraic manipulations, it can be shown that the running weighted median output is computed as

$$y(n) = \text{MEDIAN}[W_1 \diamond x_1(n), W_2 \diamond x_2(n), \ldots, W_N \diamond x_N(n)] \quad (9)$$

where $W_i > 0$ and $\diamond$ is the replication operator defined as $W_i \diamond x_i = \overbrace{x_i, x_i, \ldots, x_i}^{W_i \text{ times}}$. Weighted median smoothers were introduced in the signal processing literature by Brownigg in 1984 and have since received considerable attention [11–13]. The WM smoothing operation can be schematically described as in Fig. 2.

Weighted Median Smoothing Computation. Consider the window size 5 WM smoother defined by the symmetric weight vector $\mathbf{W} = [1, 2, 3, 2, 1]$. For the observation $\mathbf{x}(n) = [12, 6, 4, 1, 9]$, the weighted median smoother output is found as

$$\begin{aligned} y(n) &= \text{MEDIAN}[1 \diamond 12, 2 \diamond 6, 3 \diamond 4, 2 \diamond 1, 1 \diamond 9] \\ &= \text{MEDIAN}[12, 6, 6, 4, 4, 4, 1, 1, 9] \\ &= \text{MEDIAN}[1, 1, 4, 4, \underline{4}, 6, 6, 9, 12] = 4 \end{aligned} \quad (10)$$

where the median value is underlined in equation (10). The large weighting on the center input sample results in this

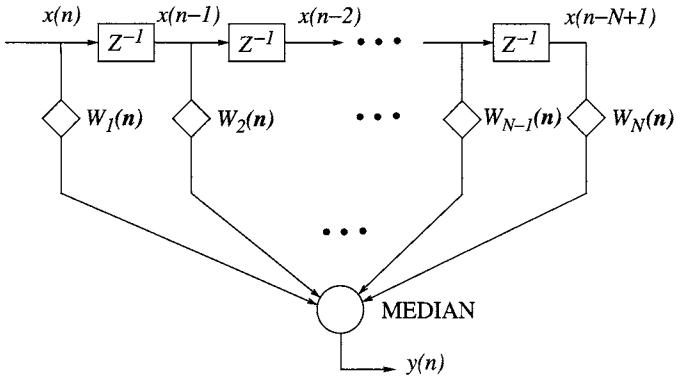

FIGURE 2 The weighted median smoothing operation.

sample being taken as the output. As a comparison, the standard median output for the given input is $y(n) = 6$.

Although the smoother weights in the above example are integer valued, the standard WM smoother definition clearly allows for positive real–valued weights. The WM smoother output for this case is as follows:

1. Calculate the threshold $W_0 = \frac{1}{2}\sum_{i=1}^{N} W_i$;
2. Sort the samples in the observation vector $\mathbf{x}(n)$;
3. Sum the weights corresponding to the sorted samples beginning with the maximum sample and continuing down in order;
4. The output is the sample whose weight causes the sum to become $\geq W_0$.

To illustrate the WM smoother operation for positive real–valued weights, consider the WM smoother defined by $\mathbf{W} = [0.1, 0.1, 0.2, 0.2, 0.1]$. The output for this smoother operating on $\mathbf{x}(n) = [12, 6, 4, 1, 9]$ is found as follows. Summing the weights gives the threshold $W_0 = \frac{1}{2}\sum_{i=1}^{5} W_i = 0.35$. The observation samples, sorted observation samples, their corresponding weight, and the partial sum of weights (from each ordered sample to the maximum) are:

observation samples	12,	6,	4,	1,	9	
corresponding weights	0.1,	0.1,	0.2,	0.2,	0.1	
sorted observation samples	1,	4,	6,	9,	12	(11)
corresponding weights	0.2,	0.2	0.1,	0.1,	0.1	
partial weight sums	0.7,	0.5,	0.3,	0.2,	0.1	

Thus, the output is 4 since when starting from the right (maximum sample) and summing the weights, the threshold $W_0 = 0.35$ is not reached until the weight associated with 4 is added.

An interesting characteristic of WM smoothers is that the nature of a WM smoother is not modified if its weights are multiplied by a positive constant. Thus, the same filter characteristics can be synthesized by different sets of weights. Although the WM smoother admits real–valued positive weights, it turns out that any WM smoother based on real–valued positive weights has an equivalent integer–valued weight representation [14]. Consequently, there are only a finite number of WM smoothers for a given window size. The number of WM smoothers, however, grows rapidly with window size [13].

Weighted median smoothers can also operate on a recursive mode. The output of a recursive WM smoother is given by

$$y(n) = \text{MEDIAN}[W_{-N_1} \diamond y(n - N_1), \ldots, W_{-1} \diamond y(n-1), W_0 \diamond x(n), \ldots, W_{N_1} \diamond x(n + N_1)] \quad (12)$$

where the weights W_i are as before constrained to be positive-valued. Recursive WM smoothers offer advantages over WM smoothers in the same way that recursive medians have advantages over their non-recursive counterparts. In fact, recursive WM smoothers can synthesize non-recursive WM smoothers of much longer window sizes [14].

2.2.1 The Center Weighted Median Smoother

The weighting mechanism of WM smoothers allows for great flexibility in emphasizing or deemphasizing specific input samples. In most applications, not all samples are equally important. Due to the symmetric nature of the observation window, the sample most correlated with the desired estimate is, in general, the center observation sample. This observation leads to the center weighted median (CWM) smoother, which is a relatively simple subset of WM smoother that has proven useful in many applications [12].

The CWM smoother is realized by allowing only the center observation sample to be weighted. Thus, the output of the CWM smoother is given by

$$y(n) = \text{MEDIAN}[x_1, \ldots, x_{c-1}, W_c \diamond x_c, x_{c+1}, \ldots, x_N] \quad (13)$$

where W_c is an odd positive integer and $c = (N+1)/2 = N_1 + 1$ is the index of the center sample. When $W_c = 1$, the operator is a median smoother, and for $W_c \geq N$, the CWM reduces to an identity operation.

The effect of varying the center sample weight is perhaps best seen by way of an example. Consider a segment of recorded speech. The voiced waveform "a" noise is shown at the top of Fig. 3. This speech signal is taken as the input of a CWM smoother of size 9. The outputs of the CWM, as the weight parameter $W_c = 2w + 1$ for $w = 0, \ldots, 3$, are shown in the figure. Clearly, as W_c is increased less smoothing occurs. This response of the CWM smoother is explained by relating the weight W_c and the CWM smoother output to select order statistics (OS).

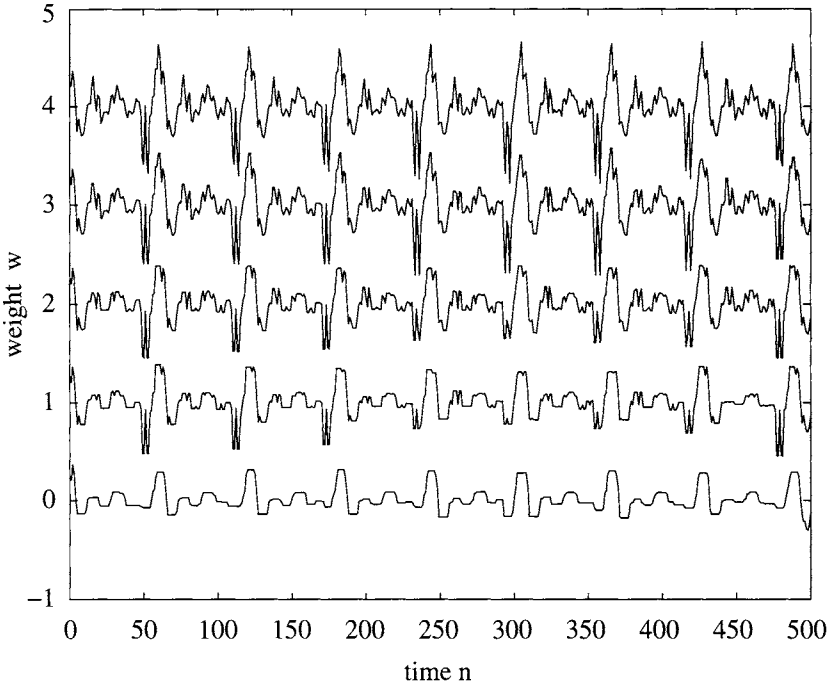

FIGURE 3 Effects of increasing the center weight of a CWM smoother of size $N=9$ operating on the voiced speech "a." The CWM smoother output is shown for $W_c=2w+1$, with $w=0,1,2,3$. Note that for $W_c=1$ the CWM reduces to median smoothing, and for $W_c=9$ it becomes the identity operator.

The CWM smoother has an intuitive interpretation. It turns out that the output of a CWM smoother is equivalent to computing

$$y(n) = \text{MEDIAN}[x_{(k)}, x_c, x_{(N-k+1)}], \tag{14}$$

where $k=(N+2-W_c)/2$ for $1 \le W_c \le N$, and $k=1$ for $W_c > N$. Since $x(n)$ is the center sample in the observation window, i.e., $x_c=x(n)$, the output of the smoother is identical to the input as long as the $x(n)$ lies in the interval $[x_{(k)}, x_{(N+1-k)}]$. If the center input sample is greater than $x_{(N+1-k)}$ the smoothing outputs $x_{(N+1-k)}$, guarding against a high rank order (large) aberrant data point being taken as the output. Similarly, the smoother's output is $x_{(k)}$ if the sample $x(n)$ is smaller than this order statistic. This CWM smoother performance characteristic is illustrated in Figs. 4 and 5. Figure 4 shows how the input sample is left unaltered if it is between the trimming statistics $x_{(k)}$ and $x_{(N+1-k)}$ and mapped to one of these statistics if it is outside this range. Figure 5 shows an example of the CWM smoother operating on a constant-valued sequence in additive Laplacian noise. Along with the input and output, the trimming statistics are shown as an upper and lower bound on the filtered signal. It is easily seen how increasing k will tighten the range in which the input is passed directly to the output.

FIGURE 4 The center weighted median smoothing operation. The center observation sample is mapped to the order statistic $x_{(k)}$ ($x_{(N+1-k)}$) if the center sample is less (greater) than $x_{(k)}$ ($x_{(N+1-k)}$), and left unaltered otherwise.

2.2.2 Permutation Weighted Median Smoothers

The principle behind the CWM smoother lies in the ability to emphasize, or de-emphasize, the center sample of the window by tunning the center weight, while keeping the weight values of all other samples at unity. In essence, the value given to the center weight indicates the "reliability" of the center sample. If the center sample does not contain an impulse (high reliability), it would be desirable to make the center weight large such that no smoothing takes place (identity filter). On the other hand, if an impulse was present in the center of the window (low reliability), no emphasis should be given to the center sample (impulse), and the center weight should be given the smallest possible weight, i.e., $W_c=1$, reducing the CWM smoother structure to a simple median. Notably, this adaptation of the center weight can be easily achieved by considering the center sample's rank among all pixels in the window [15, 16]. More precisely, denoting the rank of the center sample of the window at a given location as $R_c(n)$, then the simplest *permutation* WM smoother is defined

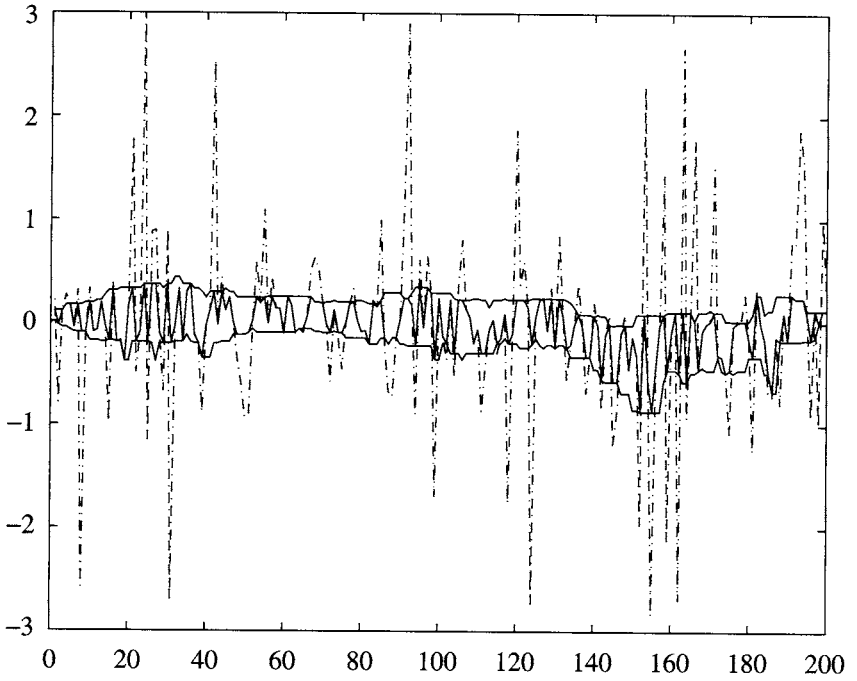

FIGURE 5 An example of the CWM smoother operating on a Laplacian distributed sequence with unit variance. Shown are the input (–·–·–) and output (——) sequences as well as the trimming statistics $x_{(k)}$ and $x_{(N+1-k)}$. The window size is 25 and $k=7$.

by the following modification of the CWM smoothing operation

$$W_c(n) = \begin{cases} N & \text{if } T_L \le R_c(n) \le T_U \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

where N is the window size and $1 \le T_L \le T_U \le N$ are two adjustable threshold parameters that determine the degree of smoothing. Note that the weight in (15) is data adaptive and may change between two values with n. The smaller (larger) the threshold parameter T_L (T_U) is set to, the better the detail-preservation. Generally, T_L and T_U are set symmetrically around the median. If the underlying noise distribution was not symmetric about the origin, a non-symmetric assignment of the thresholds would be appropriate.

The data-adaptive structure of the smoother in (15) can be extended so that the center weight is not only switched between two possible values, but can take on N different values:

$$W_c(n) = \begin{cases} W_{c(j)}(n) & \text{if } R_c(n) = j, \quad j \in \{1, 2, \ldots, N\} \\ 0 & \text{otherwise} \end{cases}. \tag{16}$$

Thus, the weight assigned to x_c is drawn from the center weight set $\{W_{c(1)}, W_{c(2)}, \ldots, W_{c(N)}\}$. With an increased number of weights, the smoother in (16) can perform better although the design of the weights is no longer trivial and optimization algorithms are needed [15, 16]. A further generalization of (16) is feasible where weights are given to all samples in the window, but where the value of each weight is data-dependent and determined by the rank of the corresponding sample. In this case, the output of the permutation WM smoother is found as

$$y(n) = \text{MEDIAN}\,[x_1(n) \diamond W_{1(R_1)}, x_2(n) \diamond W_{1(R_2)} \\ x_1(n) \diamond W_{1(R_1)}] \tag{17}$$

where $W_{i(Ri)}$ is the weight assigned to $x_i(n)$ and selected according to the sample's rank R_i. The weight assigned to x_i is drawn from the weight set $\{W_{i(1)}, W_{i(2)}, \ldots, W_{i(N)}\}$. Having N weights per sample, a total of N^2 samples need to be stored in the computation of (17). In general, optimization algorithms are needed to design the set of weights although in some cases the design is simple, as with the smoother in (15). Permutation WM smoothers can provide significant improvement in performance at the higher cost of memory cells [15].

2.2.3 Threshold Decomposition and Stack Smoothers

An important tool for the analysis and design of weighted median smoothers is the threshold decomposition

property [17]. Given an integer-valued set of samples $x_1, x_2, \ldots, x_N$ forming the vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$, where $x_i \in \{-M, \ldots, -1, 0, \ldots, M\}$. The threshold decomposition of $\mathbf{x}$ amounts to decomposing this vector into $2M$ binary vectors $\mathbf{x}^{-M+1}, \ldots, \mathbf{x}^0, \ldots, \mathbf{x}^M$ where the ith element of $\mathbf{x}^m$ is defined by

$$x_i^m = T^m(x_i) = \begin{cases} 1 & \text{if } x_i \geq m; \\ -1 & \text{if } x_i < m, \end{cases} \tag{18}$$

where $T^m(\cdot)$ is referred to as the thresholding operator. Using the sign function, the above can be written as $x_i^m = \operatorname{sgn}(x_i - m^-)$ where m^- represents a real number approaching the integer m from the left. Although defined for integer-valued signals, the thresholding operation in (18) can be extended to non-integer signals with a finite number of quantization levels. The threshold decomposition of the vector $\mathbf{x} = [0,0,2,-2,1,1,0,-1,-1]^T$ with $M = 2$, for instance, leads to the 4 binary vectors

$$\begin{aligned} \mathbf{x}^2 &= [-1, -1, 1, -1, -1, -1, -1, -1, -1]^T \\ \mathbf{x}^1 &= [-1, -1, 1, -1, 1, 1, -1, -1, -1]^T \\ \mathbf{x}^0 &= [1, 1, 1, -1, 1, 1, 1, -1, -1]^T \\ \mathbf{x}^{-1} &= [1, 1, 1, -1, 1, 1, 1, 1, 1]^T. \end{aligned} \tag{19}$$

Threshold decomposition has several important properties. First, threshold decomposition is reversible. Given a set of thresholded signals, each of the samples in $\mathbf{x}$ can be exactly reconstructed as

$$x_i = \frac{1}{2} \sum_{m=-M+1}^{M} x_i^m. \tag{20}$$

Thus, an integer-valued discrete-time signal has a unique threshold signal representation, and vice versa:

$$x_i \overset{T.D.}{\leftrightarrow} \{x_i^m\},$$

where $\overset{T.D.}{\leftrightarrow}$ denotes the one-to-one mapping provided by the threshold decomposition operation.

The set of threshold decomposed variables obey the following set of partial ordering rules. For all thresholding levels $m > \ell$, it can be shown that $x_i^m \leq x_i^\ell$. In particular, if $x_i^m = 1$ then $x_i^\ell = 1$ for all $\ell < m$. Similarly, if $x_i^\ell = -1$ then $x_i^m = -1$, for all $m > \ell$. The partial order relationships among samples across the various thresholded levels emerge naturally in thresholding and are referred to as the *stacking constraints* [18].

Threshold decomposition is of particular importance in weighted median smoothing since they are commutable operations. That is, applying a weighted median smoother to a $2M+1$ valued signal is equivalent to decomposing the signal to $2M$ binary thresholded signals, processing each binary signal separately with the corresponding WM smoother, and then adding the binary outputs together to obtain the integer-valued output. Thus, the weighted median smoothing of a set of samples $x_1, x_2, \ldots, x_N$ is related to the set of the thresholded weighted median smoothed signals as [14, 17]

$$\begin{aligned} &\text{Weighted MEDIAN}(x_1, \ldots, x_N) \\ &= \frac{1}{2} \sum_{m=-M+1}^{M} \text{Weighted MEDIAN}(x_1^m, \ldots, x_N^m). \end{aligned} \tag{21}$$

Since $x_i \overset{T.D.}{\leftrightarrow} \{x_i^m\}$ and Weighted MEDIAN $\left(x_i|_{i=1}^N\right) \overset{T.D.}{\leftrightarrow} \{\text{Weighted MEDIAN}\left(x_i^m|_{i=1}^N\right)\}$, the relationship in (21) establishes a *weak* superposition property satisfied by the nonlinear median operator which is important from the fact that the effects of median smoothing on binary signals are much easier to analyze than that on multilevel signals. In fact, the weighted median operation on binary samples reduces to a simple Boolean operation. The median of three binary samples x_1, x_2, x_3, for example, is equivalent to: $x_1x_2 + x_2x_3 + x_1x_3$, where the + (OR) and x_ix_j (AND) Boolean operators in the $\{-1, 1\}$ domain are defined as

$$\begin{aligned} x_i + x_j &= \max(x_i, x_j) \\ x_i x_j &= \min(x_i, x_j). \end{aligned} \tag{22}$$

Note that the operations in (22) are also valid for the standard Boolean operations in the $\{0, 1\}$ domain.

The framework of threshold decomposition and Boolean operations has led to the general class of nonlinear smoothers referred here to as stack smoothers [18], whose output is defined by

$$S(x_1, \ldots, x_N) = \frac{1}{2} \sum_{m=-M+1}^{M} f(x_1^m, \ldots, x_N^m) \tag{23}$$

where $f(\cdot)$ is a Boolean operation satisfying (22) and the stacking property. More precisely, if two binary vectors $\mathbf{u} \in \{-1, 1\}^N$ and $\mathbf{v} \in \{-1, 1\}^N$ stack, i.e., $u_i \geq v_i$ for all $i \in \{1, \ldots, N\}$, then their respective outputs stack, $f(\mathbf{u}) \geq f(\mathbf{v})$. A necessary and sufficient condition for a function to posses the stacking property is that it can be expressed as a Boolean function which contains no complements of input variables [19]. Such functions are known as *positive Boolean* functions (PBFs).

Given a positive Boolean function $f(x_1^m, \ldots, x_N^m)$ which characterizes a stack smoother, it is possible to find the equivalent smoother in the integer domain by replacing the

binary AND and OR Boolean functions acting on the x_i's with *max* and *min* operations acting on the multilevel x_i samples. A more intuitive class of smoothers is obtained, however, if the positive Boolean functions are further restricted [14]. When self-duality and separability is imposed, for instance, the equivalent integer domain stack smoothers reduce to the well known class of weighted median smoothers with positive weights. For example, if the Boolean function in the stack smoother representation is selected as $f(x_1, x_2, x_3, x_4) = x_1x_3x_4 + x_2x_4 + x_2x_3 + x_1x_2$, the equivalent WM smoother takes on the positive weights $(W_1, W_2, W_3, W_4) = (1, 2, 1, 1)$. The procedure of how to obtain the weights W_i from the PBF is described in [14].

2.3 Weighted Median Filters

Admitting only positive weights, WM smoothers are severely constrained as they are, in essence, smoothers having "low-pass" type filtering characteristics. A large number of engineering applications require "band-pass" or "high-pass" frequency filtering characteristics. Linear FIR equalizers admitting only positive filter weights, for instance, would lead to completely unacceptable results. Thus, it is not surprising that weighted median smoothers admitting only positive weights lead to unacceptable results in a number of applications.

Much like the sample mean can be generalized to the rich class of linear FIR filters, there is a logical way to generalize the median to an equivalently rich class of weighted median filters that admit both positive and negative weights [20]. It turns out that the extension is not only natural, leading to a significantly richer filter class, but it is simple as well. Perhaps the simplest approach to derive the class of weighted median filters with real-valued weights is by analogy. The sample mean $\bar{\beta} = \text{MEAN}(X_1, X_2, \ldots, X_N)$ can be generalized to the class of linear FIR filters as

$$\beta = \text{MEAN}\,(W_1 \cdot X_1, W_2 \cdot X_2, \ldots, W_N \cdot X_N) \tag{24}$$

where $X_i \in R$. In order to apply the analogy to the median filter structure (24) must be written as

$$\begin{aligned}\bar{\beta} = &\text{MEAN}\,(|W_1| \cdot \text{sgn}(W_1)X_1, |W_2| \cdot \text{sgn}(W_2)X_2, \ldots,\\ &|W_N| \cdot \text{sgn}(W_n)X_N)\end{aligned} \tag{25}$$

where the sign of the weight affects the corresponding input sample and the weighting is constrained to be non-negative. By analogy, the class of weighted median filters admitting real-valued weights emerges as [20]

$$\begin{aligned}\tilde{\beta} = &\text{MEAN}\,(|W_1| \diamond \text{sgn}(W_1)X_1, |W_2| \diamond \text{sgn}(W_2)X_2, \ldots,\\ &|W_N| \diamond \text{sgn}(W_n)X_N)\end{aligned} \tag{26}$$

with $W_i \in R$ for $i = 1, 2, \ldots, N$. Again, the weight signs are uncoupled from the weight magnitude values and are merged with the observation samples. The weight magnitudes play the equivalent role of positive weights in the framework of weighted median smoothers. It is simple to show that the weighted mean (normalized) and the weighted median operations shown in (25) and (26) respectively minimize

$$\begin{aligned}G_2(\beta) &= \sum_{i=1}^{N} |W_i|(\text{sgn}(W_i)X_i - \beta)^2 \quad \text{and}\\ G_1(\beta) &= \sum_{i=1}^{N} |W_i|\left|\text{sgn}(W_i)X_i - \beta\right|.\end{aligned} \tag{27}$$

While $G_2(\beta)$ is a convex continuous function, $G_1(\beta)$ is a convex but piecewise linear function whose minimum point is guaranteed to be one of the "signed" input samples (i.e., $\text{sgn}(W_i)\, X_i$).

Weighted Median Filter Computation. The WM filter output for non-integer weights can be determined as follows [20]:

1. Calculate the threshold $T_0 = \frac{1}{2}\sum_{i=1}^{N} |W_i|$.
2. Sort the "signed" observation samples $\text{sgn}(W_i)X_i$.
3. Sum the magnitude of the weights corresponding to the sorted "signed" samples beginning with the maximum and continuing down in order.
4. The output is the signed sample whose magnitude weight causes the sum to become $\geq T_0$.

The following example illustrates this procedure. Consider the window size 5 WM filter defined by the real valued weights $[W_1, W_2, W_3, W_4, W_5]^T = [0.1, 0.2, 0.3, -0.2, 0.1]^T$. The output for this filter operating on the observation set $[X_1, X_2, X_3, X_4, X_5]^T = [-2, 2, -1, 3, 6]^T$ is found as follows. Summing the absolute weights gives the threshold $T_0 = \frac{1}{2}\sum_{i=1}^{5} |W_i| = 0.45$. The "signed" observation samples, sorted observation samples, their corresponding weight, and the partial sum of weights (from each ordered sample to the maximum) are:

Observation samples	−2,	2,	−1,	3,	6
Corresponding weights	0.1,	0.2,	0.3,	−0.2,	0.1
Sorted signed observation samples	−3,	−2,	−1,	2,	6
Corresponding absolute weights	0.2,	0.1,	0.3,	0.2,	0.1
Partial weight sums	0.9,	0.7,	<u>0.6</u>,	0.3,	0.1

Thus, the output is −1 since when starting from the right (maximum sample) and summing the weights, the threshold $T_0 = 0.45$ is not reached until the weight associated with −1 is added. The underlined sum value above indicates that this is the first sum which meets or exceeds the threshold.

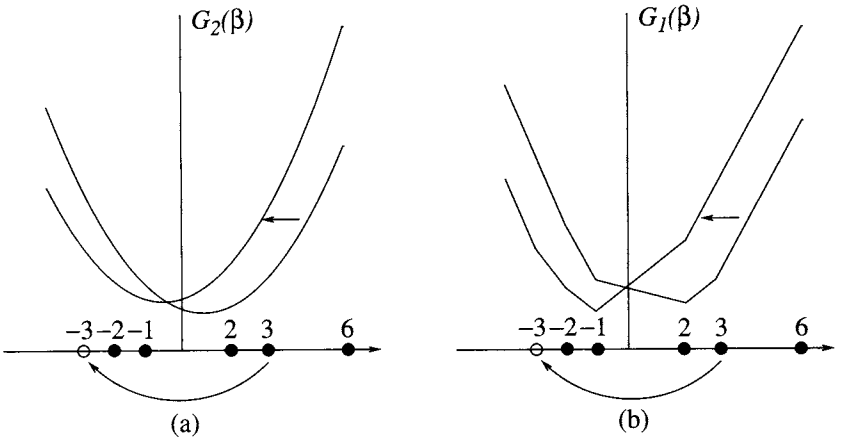

FIGURE 6 Effects of negative weighting on the cost functions $G_2(\beta)$ and $G_1(\beta)$. The input samples are $[X_1, X_2, X_3, X_4, X_5]^T = [-2, 2, -1, 3, 6]^T$ which are filtered by the two set of weights $[0.1, 0.2, 0.3, 0.2, 0.1]^T$ and $[0.1, 0.2, 0.3, -0.2, 0.1]^T$, respectively.

The effect that negative weights have on the weighted median operation is similar to the effect that negative weights have on linear FIR filter outputs. Figure 6 illustrates this concept where $G_2(\beta)$ and $G_1(\beta)$, the cost functions associated with linear FIR and weighted median filters respectively, are plotted as a function of β. Recall that the output of each filter is the value minimizing the cost function. The input samples are again selected as $[X_1, X_2, X_3, X_4, X_5] = [-2, 2, -1, 3, 6]$ and two sets of weights are used. The first set is $[W_1, W_2, W_3, W_4, W_5] = [0.1, 0.2, 0.3, 0.2, 0.1]$ where all the coefficients are positive, and the second set being $[0.1, 0.2, 0.3, -0.2, 0.1]$ where W_4 has been changed, with respect to the first set of weights, from 0.2 to -0.2. Figure 6(a) shows the cost functions $G_2(\beta)$ of the linear FIR filter for the two sets of filter weights. Notice that by changing the sign of W_4, we are effectively moving X_4 to its new location $\text{sgn}(W_4)X_4 = -3$. This, in turn, pulls the minimum of the cost function towards the relocated sample $\text{sgn}(W_4)X_4$. Negatively weighting X_4 on $G_1(\beta)$ has a similar effect as shown in Fig. 6(b). In this case, the minimum is pulled towards the new location of $\text{sgn}(W_4)X_4$. The minimum, however, occurs at one of the samples $\text{sgn}(W_i)X_i$. More details on WM filtering can be found in [20, 21].

2.4 Weighted Median Filters for Color Images

The extension of the weighted median for use with multi-dimensional (multichannel) signals is not straightforward. Sorting multicomponent (vector) values and selecting the middle value is not well defined as in the scalar case, [22–26]. In consequence, the weighted median filtering operation of a multidimensional signal can be achieved in a number of ways [22–27].

Although we concentrate on the filtering of color images, the concepts defined in this section, can also be applied to the filtering of N-component imagery [28]. Color images are represented by three components: red, green, and blue; with combinations of these to produce the entire color spectrum.

2.4.1 Marginal WM Filter

The simplest approach to WM filtering a color image is to process each component (red, green, and blue) independently by a scalar WM filter and then combine them to produce the filtered color image. A drawback associated with this method is that different components can be strongly correlated and, if each component is processed separately, this correlation is not exploited. In addition, since each component is filtered independently, the filter outputs can combine to produce colors not present in the original image. The advantage of marginal processing is the computational simplicity.

2.4.2 Vector WM Filter (WVM)

A more logical extension, yet more computationally expensive approach is found through the minimization of a weighted cost function which takes into account the multicomponent nature of the data. Here, the three components are jointly filtered by a vector WM filter, such that the cross-correlations between components is exploited, leading to a filtered color image. Vector WM filtering requires the extension of the original WM filter definition as follows. Define $\underline{\mathbf{x}}_i = [x_i^1, x_i^2, x_i^3]^T$ as a 3-dimensional vector, where x_i^1, x_i^2, and x_i^3 are respectively the red, green, and blue components of the ith pixel in a color image, and recall that the weighted median of a set of 1-dimensional samples x_i $i = 1, \ldots, N$ is given by

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} |W_i| |\text{sgn}(W_i)x_i - \beta|. \tag{28}$$

Extending this definition to a set of 3-dimensional vectors $\underline{x}_i$ for $i = 1, \ldots, N$ leads to

$$\hat{\underline{\beta}} = \arg\min_{\underline{\beta}} \sum_{i=1}^{N} |W_i| \left\| \underline{s}_i - \underline{\beta} \right\| \tag{29}$$

where $\hat{\underline{\beta}} = [\hat{\beta}^1, \hat{\beta}^2, \hat{\beta}^3]^T$, $\underline{s}_i = \mathrm{sgn}(W_i)\underline{x}_i$ and $\|\cdot\|$ is the L_2 norm defined as

$$\left\| \underline{s}_i - \underline{\beta} \right\| = ((s_i^1 - \beta^1)^2 + (s_i^2 - \beta^2)^2 + (s_i^3 - \beta^3)^2)^{\frac{1}{2}}. \tag{30}$$

The vector weighted median thus requires N scalar weights, with one scalar weight assigned per each input sample. Unlike the 1-dimensional case, $\hat{\underline{\beta}}$ is not generally one of the $\mathbf{s}_i$, indeed, there is no closed form solution for $\hat{\underline{\beta}}$. Moreover, solving (29) involves a minimization problem in a 3 dimensional space that can be computationally expensive. To overcome these shortcomings, a suboptimal solution for (29) is found if $\hat{\underline{\beta}}$ is restricted to be one of the signed samples $\underline{s}_i$. This leads to the following definition:

The vector WM filter output of $\underline{x}_1, \ldots, \underline{x}_N$ is the value of $\hat{\underline{\beta}}$, with $\hat{\underline{\beta}} \in \{\underline{s}_1, \ldots, \underline{s}_N\}$ such that

$$\sum_{i=1}^{N} |W_i| \left\| \hat{\underline{\beta}} - \underline{s}_i \right\| \leq \sum_{i=1}^{N} |W_i| \left\| \underline{s}_j - \underline{s}_i \right\| \quad \text{for all } j = 1, \ldots, N \tag{31}$$

This definition can be implemented as follows.

1. For each signed sample $\underline{s}_j$, compute the distances to all the other signed samples ($||\underline{s}_j - \underline{s}_i||$) for $i = 1, \ldots, N$ using (30).
2. Compute the sum of the weighted distances given by the right side of (31).
3. Choose as filter output the sample $\underline{s}_j$ that produces the minimum sum of the weighted distances.

In a more general case, the same procedure can be employed to calculate the vector weighted median of a set of input samples using other distance measures. The vector median in Astola et al. (1990) [29] uses the L_p norm defined as $||\underline{\mathbf{x}}||_p = (\sum |\mathbf{x}_i|^p)^{\frac{1}{p}}$ as a distance measure, transforming (29) into

$$\hat{\underline{\beta}} = \arg\min_{\underline{\beta} \in \{\underline{s}_i\}} \sum_{i=1}^{N} |W_i| \left\| \underline{s}_i - \underline{\beta} \right\|_p. \tag{32}$$

Several optimization algorithms for the design of the weights in (32) have been developed. One such method, proposed by Shen and Barner (2004) [30] can be summarized as follows:

By definition, the WVM filter is selection type and its output is one of the input samples as it is shown in (32). First it is necessary to find the closest sample to the desired output, say $\underline{s}_{e_{min}}$. The output of the filter is then calculated using the current weights. If the output of the filter is $\underline{s}_{e_{min}}$ the weights are considered optimal. Otherwise, the weights should be modified in order to obtain $\underline{s}_{e_{min}}$ as the output. The weights should be updated according to

$$W_i(n+1) = W_i(n) + \mu \Delta W_i, \; i = 1, 2, \ldots, N \tag{33}$$

where

$$\Delta W_i = \frac{d(\underline{s}_{e_{min}}) - d(\underline{s}_{j0})}{\left\| \underline{s}_{j0} - \underline{s}_i \right\| - \left\| \underline{s}_{e_{min}} - \underline{s}_i \right\|}, \; i = 1, 2, \ldots, N \tag{34}$$

where $d(\underline{s}_j) = \sum_{i=1}^{N} W_i \left\| \underline{s}_j - \underline{s}_i \right\|$ and $\underline{s}_{j0}$ is the current filter output. This algorithm is a greedy approach since it determines the weight changes based on local characteristics. Despite the existence of several optimization algorithms like the one just shown, weighted vector medians have not significantly spread beyond image smoothing applications. A more general vector median filter structure is presented next.

2.4.3 Weighted Multichannel Median Filtering Structures

The multivariate filtering structure is derived from the maximum likelihood estimation of location, this time in a multivariate signal space. Consider a set of independent but not identically distributed vector valued samples obeying a joint Gaussian distribution with identical location parameter $\underline{\mu}$,

$$f(\underline{x}_i) = \frac{1}{(2\pi)^{\frac{M}{2}} |\mathbf{C}_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\underline{x}_i - \underline{\mu})^T \mathbf{C}_i^{-1} (\underline{x}_i - \underline{\mu})} \tag{35}$$

where $\underline{x}_i$ and $\underline{\mu}$ are all M-variate column vectors ($M = 3$ for a RGB color image), and $\mathbf{C}_i$ is the $M \times M$ cross-channel correlation matrix of the sample $\underline{x}_i$. In most multichannel applications, the signals from sub-channels are often correlated. Further, the correlation structure between sub-channels may often be stationary or at least quasi-stationary for a period of time. In these cases, the assumption that the correlation matrices $\mathbf{C}_i$ differ only by a scale factor is valid, that is $\mathbf{C}_i^{-1} = q_i \mathbf{C}^{-1}$. The corresponding MLE is then

$$\underline{\mu} = \left(\sum_{i=1}^{N} q_i \mathbf{C}^{-1} \right)^{-1} \left(\sum_{i=1}^{N} q_i \mathbf{C}^{-1} \underline{x}_i \right) \tag{36}$$

where $\left(\sum_{i=1}^{N} q_i \mathbf{C}^{-1}\right)^{-1}$ is a constant matrix and $\sum_{i=1}^{N} q_i \mathbf{C}^{-1} \underline{\mathbf{x}}_i$ provides the filtering structure. Removing the constant, the filtering structure can be formulated as

$$\underline{\mathbf{Y}} = \sum_{i=1}^{N} V_i \mathbf{W}^T \underline{\mathbf{x}}_i \tag{37}$$

$$= \sum_{i=1}^{N} V_i \begin{bmatrix} W^{11} & \dots & W^{M1} \\ \vdots & \ddots & \vdots \\ W^{1M} & \dots & W^{MM} \end{bmatrix} \begin{bmatrix} x_i^1 \\ \vdots \\ x_i^M \end{bmatrix}, \tag{38}$$

where V_i is the (time/spatial) weight applied to the i-th vector sample in the observation window and W_{ij} is the cross-channel weight exploiting the correlation between the i-th and j-th components of a sample.

Even though it is mathematically intractable to derive a similar result as in (38) from a multivariate Laplacian distribution, it is still possible to define a nonlinear multivariate filter by direct analogy by replacing the summations in (38) with median operators. This filter is referred to as the Weighted Multichannel Median (WMM) and is defined as follows (Li et al. (2004) [32]).

$$\underline{\mathbf{Y}} = \text{MEDIAN}(|V_i| \diamond \text{sgn}(V_i)\underline{\mathbf{Q}}_i|_{i=1}^{N}), \tag{39}$$

where

$$\underline{\mathbf{Q}}_i = \begin{bmatrix} \text{MEDIAN}(|W^{j1}| \diamond \text{sgn}(W^{j1})x_i^j|_{j=1}^{M}) \\ \text{MEDIAN}(|W^{j2}| \diamond \text{sgn}(W^{j2})x_i^j|_{j=1}^{M}) \\ \vdots \\ \text{MEDIAN}(|W^{jM}| \diamond \text{sgn}(W^{jM})x_i^j|_{j=1}^{M}) \end{bmatrix} \tag{40}$$

is an M-variate vector. As it was stated before, there is no unique way of defining even the simplest median over vectors, in consequence, the outer median in (39) can have several different implementations. Due to its simplicity and ease of mathematic analysis, a sub-optimal implementation, where the outer median in (39) is replaced by a vector of marginal medians, can be used. Thus, the marginal weighted multichannel median (marginal WMM) is defined as in Li et al. (2004) [32].

$$\underline{\mathbf{Y}} = \begin{bmatrix} \text{MED}(|V_i| \diamond \text{sgn}(V_i)Q_i^1|_{i=1}^{N}) \\ \text{MED}(|V_i| \diamond \text{sgn}(V_i)Q_i^2|_{i=1}^{N}) \\ \vdots \\ \text{MED}(|V_i| \diamond \text{sgn}(V_i)Q_i^M|_{i=1}^{N}) \end{bmatrix} \tag{41}$$

where $Q_i^l = \text{MED}(|W^{jl}| \diamond \text{sgn}(W^{jl})x_i^j|_{j=1}^{M})$ for $l = 1, \dots, M$.

Filter Optimization. Assume that the observed process $\underline{\mathbf{x}}(n) = [x^1(n)x^2(n) \dots x^M(n)]^T$ is statistically related to a desired process $\underline{\mathbf{D}}(n) = [D^1(n)D^2(n) \dots D^M(n)]^T$ of interest, typically considered a transformed or corrupted version of $\underline{\mathbf{D}}(n)$. The filter input vector at time n is $\mathbf{X}(n) = [\underline{\mathbf{x}}_1(n)\underline{\mathbf{x}}_2(n) \dots \underline{\mathbf{x}}_N(n)]^T$.

Assume that the time/spatial dependent weight vector is $\underline{\mathbf{V}} = [V_1 V_2 \dots V_N(n)]^T$, and the cross-channel weight matrix is

$$\mathbf{W} = \begin{bmatrix} W^{11} & \dots & W^{1M} \\ \vdots & \ddots & \vdots \\ W^{M1} & \dots & W^{MM} \end{bmatrix}.$$

Denote $Q_i^l = \text{MED}(|W^{jl}| \diamond \text{sgn}(W^{jl})x_i^j|_{j=1}^{M})$ for $l = 1, \dots, M$, then the output of the marginal WMM can be defined as $\hat{\underline{\mathbf{D}}} = [\hat{D}^1 \hat{D}^2 \cdots \hat{D}^M]^T$, where $\hat{D}^l = \text{MED}(|V_i| \diamond \text{sgn}(V_i)Q_i^l|_{j=1}^{N})$ for $l = 1, \dots, M$. Under the least mean absolute error (LMA) criterion, the cost function to minimize is

$$J_1(\underline{\mathbf{V}}, \mathbf{W}) = E\left\{ \left\| \underline{\mathbf{D}} - \hat{\underline{\mathbf{D}}} \right\|_1 \right\} \tag{42}$$

Using the instantaneous estimate for the gradient, we obtain the adaptive algorithm for the time dependent weight vector $\underline{\mathbf{V}}$ as follows,

$$V_i(n+1) = V_i(n) + \mu_v \text{sgn}(V_i(n))\underline{\mathbf{e}}^T(n)\underline{\mathbf{G}}_i^{\hat{D}}(n) \tag{43}$$

where $\underline{\mathbf{G}}_i^{\hat{D}} = [G_i^{\hat{D}^1} \dots G_i^{\hat{D}^M}]^T$ and $G_i^{\hat{D}^l} = \text{sgn}(\text{sgn}(V_i)Q_i^l - \hat{D}^l)$ for $l = 1, \dots, M$, and $\underline{\mathbf{e}}(n) = \underline{\mathbf{D}}(n) - \hat{\underline{\mathbf{D}}}(n)$. The adaptive algorithm for the cross-channel weight matrix $\boldsymbol{W}$ is:

$$W^{st}(n+1) = W^{st}(n) + \mu_w \text{sgn}(W^{st}(n))e^t(n)(\underline{\mathbf{V}}^T(n)\underline{\mathbf{A}}^s(n)), \tag{44}$$

where $\underline{\mathbf{A}}^s = [A_1^s A_2^s \dots A_N^s]^T$, and $A_i^s = \delta(\text{sgn}(V_i)Q_i^l - \hat{D}^l)\text{sgn}(\text{sgn}(W^{st})x_i^s - Q_i^t)$ for $i = 1, \dots, N$ where $\delta(x) = 1$ for $x = 0$ and $\delta(x) = 0$ otherwise, s, $t = 1, 2, \dots, M$ and $e^t(n)$ is the t-th component of $e(n)$.

3 Image Noise Cleaning

Median smoothers are widely used in image processing to clean images corrupted by noise. Median filters are particularly effective at removing outliers. Often referred to as "salt and pepper" noise, outliers are often present due to bit errors in transmission, or introduced during the signal acquisition stage. Impulsive noise in images can also occur as a result to damage to analog film. Although a weighted median smoother can be designed to "best" remove the noise, CWM smoothers often provide similar results at a much lower complexity [12]. By simply tunning the center weight, a user can obtain the desired level of smoothing. Of course, as the center weight is decreased to attain the desired level of impulse

suppresion, the output image will suffer increased distortion particularly around the image's fine details. Nonetheless, CWM smoothers can be highly effective in removing "salt and pepper" noise while preserving the fine image details. Figures 7(a) and (b) depict a noise free gray-scale image and the corresponding image with "salt and pepper" noise. Each pixel in the image has a 10% probability of being contaminated with an impulse. The impulses occur randomly and were generated by MATLAB's imnoise funtion. Figures 7(c) and (d) depict the noisy image processed with a 5×5 window CWM smoother with center weights 15 and 5, respectively. The impulse-rejection and detail-preservation tradeoff in CWM smoothing is clearly illustrated in Figs. 7(c) and (d). A color version of the "portrait" image was also corrupted by "salt and pepper" noise and filtered using CWM. Marginal CWM smoothing was performed in Fig. 8. The differences between marginal and vector WM processing will be illustrated shortly.

At the extreme, for $W_c = 1$, the CWM smoother reduces to the median smoother which is effective at removing impulsive noise. It is, however, unable to preserve the image's fine details [33]. Figure 9 shows enlarged sections of the noise-free image (left), and of the noisy image after the median smoother has been applied (center). Severe blurring is introduced by the median smoother and it is readily apparent in Fig. 9. As a reference, the output of a running mean of the same size is also shown in Fig. 9 (right). The image is severly degraded as each impulse is smeared to neighboring pixels by the averaging operation.

Figures 7 and 8 show that CWM smoothers can be effective at removing impulsive noise. If increased detail-preservation is sought and the center weight is increased, CWM smoothers begin to breakdown and impulses appear on the output. One simple way to ameliorate this limitation is to employ a recursive mode of operation. In essence, past inputs are replaced by previous outputs as described in (12) with the only difference that only the center sample is weighted. All the other samples in the window are weighted by one. Figure 10 shows enlarged sections of the non-recursive CWM filter (left) and of the corresponding recursive CWM smoother, both with the same center weight ($W_c = 15$). This figure illustrates the increased noise attenuation provided by recursion without the loss of image resolution.

Both, recursive and non-recursive CWM smoothers, can produce outputs with disturbing artifacts particularly when the center weights are increased in order to improve the detail-preservation characteristics of the smoothers. The artifacts are most apparent around the image's edges and details. Edges at the output appear jagged and impulsive noise can break through next to the image detail features. The distinct response of CWM smoother in different regions of the image is due to the fact that images are non-stationary in nature. Abrupt changes in the image's local mean and texture carry most of the visual information content. CWM smoothers process the entire image with fixed weights and are inherently limited in this sense by their static nature. Although some improvement is attained by introducing recursion or by using more weights in a properly design WM smoother structure, these approaches are also static and do not properly address the non-stationarity nature of images.

Significant improvement in noise attenuation and detail preservation can be attained if permutation WM filter structures are used. Figure (10)(right) shows the output of the permutation CWM filter in (15) when the "salt and pepper" degraded "portrait" image is inputted. The parameters were given the values $T_L = 6$ and $T_U = 20$. The improvement achieved by switching W_c between just two different values is significant. The impulses are deleted without exception, the details are preserved, and the jagged artifacts typical of CWM smoothers are not present in the output.

Figures 8–10 depict the results of marginal component filtering. Figure 11 illustrates the differences between marginal and vector processing. Figure 11(a) shows the original image, 11(b) shows the filtered image using marginal filtering, 11(c) shows the filtered image using weighted vector median filtering, and 11(d) the filtered image using marginal multivariate weighted median filtering. As Fig. 11 shows the marginal processing of color images removes more noise than in the vector approach, however, it can introduce new color artifacts to the image while the multivariate median removes most of the noise without introducing many artifacts.

To further illustrate the differences between the optimal weighted median filtering schemes for color images we proceed as follows. A RGB color image contaminated with 10% correlated salt and pepper noise is processed by the WVM filter, and the marginal WMM filter separately. The observation window is set to 3×3 and 5×5. The optimal weights for the filters are obtained first by running optimization algorithms over a small part of the corrupted image. The resulting weights are then passed to the corresponding filters to denoise the whole image. The filter outputs are depicted in Figs. 12 and 13.

As a measure of the effectiveness of the filters, the mean absolute error of the outputs was calculated for each filter, the results are summarized in Table 1. Peak signal-to-noise ratio (PSNR) was also used to evaluate the fidelity of the filtered images.

TABLE 1 Average MAE and PSNR of the output images

	MAE		PSNR (dB)	
Filter	3×3	5×5	3×3	5×5
Noisy signal	0.1480		14.12	
WVM	0.0705	0.0561	23.38	24.72
Marginal WMM	0.0242	0.0347	32.74	30.67

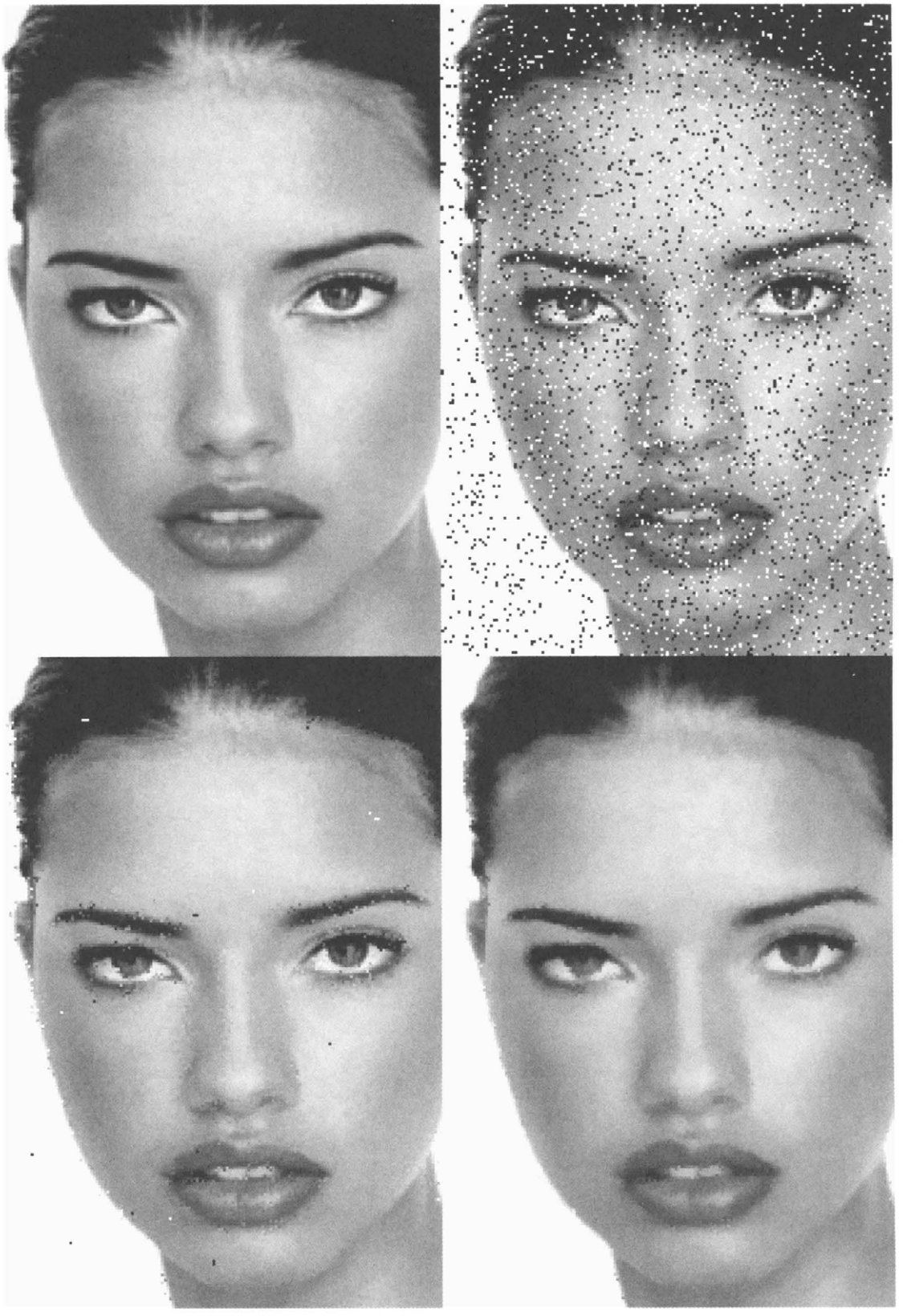

FIGURE 7 Impulse noise cleaning with a 5×5 CWM smoother: (a) original gray-scale "portrait" image, (b) image with "salt and pepper" noise, (c) CWM smoother with $W_c = 15$, (d) CWM smoother with $W_c = 5$.

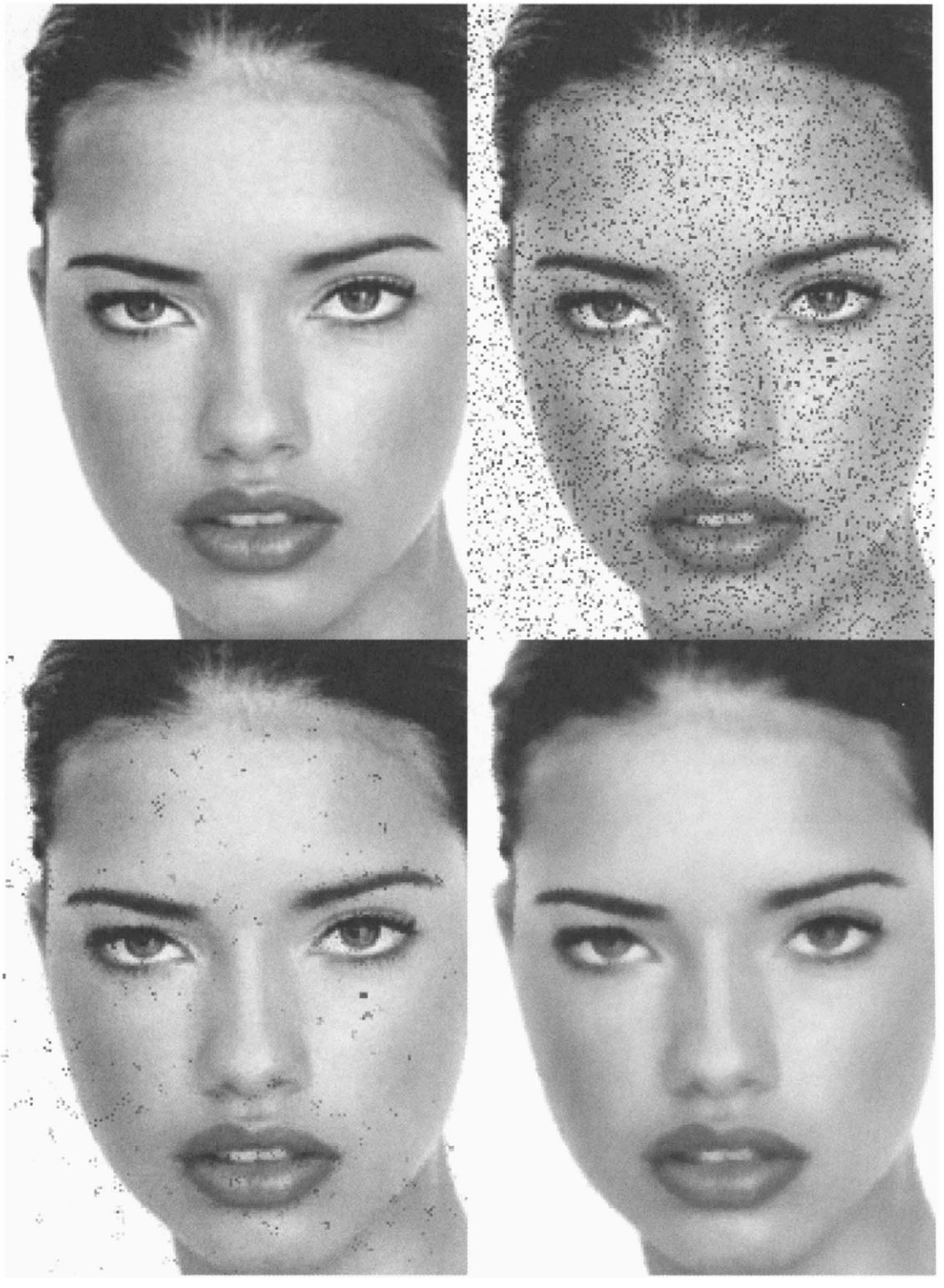

FIGURE 8 Impulse noise cleaning with a 5×5 CWM smoother: (a) original "portrait" image, (b) image with "salt and pepper" noise, (c) CWM smoother with $W_c = 16$, (d) CWM smoother with $W_c = 5$. (See color insert.)

FIGURE 9 (Enlarged) Noise-free image (left), 5×5 median smoother output (center), and 5×5 mean smoother (right). (See color insert.)

The statistics in Table 1 show that the marginal WMM filter outperforms the WVM filter in this color image denoising simulation. Moreover, the output of the marginal WMM filter is almost salt and pepper noise free. As a comparison, the output of the WVM filter is visually less pleasant with many unfiltered outliers. Notice that the output of the marginal WMM filter with the 3×3 observation window preserves more image details than that of the 5×5 realization, and has a better PSNR. Figure 14 shows the optimum weights obtained for all the filters used in this example.

The noise generated for this example was cross-channel correlated. As a result, Figs. 14 (c) and (f), show that the optimum cross-channel weights for the 3×3 and 5×5 window are very similar, since they are based on the same statistics. Figures 14 (b) and (e) show that, spatially, the marginal WMM filter tries to emphasize the center sample of the window. This is an expected result since the noise samples are spatially independent. Finally, Figs. 14 (a) and (d) show a distribution of the spatial weights that is not as smooth as the one shown in Figs. 14 (b) and (e), this shows the negative effects that the cross channel correlation of the noise generates in the WVM filter.

4 Image Zooming

Zooming an image is an important task used in many applications, including the World Wide Web, digital video, DVDs, and scientific imaging. When zooming, pixels are inserted into the image in order to expand the size of the image, and the major task is the interpolation of the new pixels from the surrounding original pixels. Weighted medians have been applied to similar problems requiring interpolation, such as interlace to progressive video conversion for television systems [13]. The advantage of using the weighted median in interpolation over traditional linear methods is better edge-preservation and less of a "blocky" look to edges.

To introduce the idea of interpolation, suppose that a small matrix must be zoomed by a factor of 2, and the median of the closest two (or four) original pixels is used to interpolate each new pixel:

$$\begin{bmatrix} 7 & 8 & 5 \\ 6 & 10 & 9 \end{bmatrix} \xrightarrow{\text{Zero Interlace}} \begin{bmatrix} 7 & 0 & 8 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 10 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\xrightarrow{\text{Median Interpolation}} \begin{bmatrix} 7 & 7.5 & 8 & 6.5 & 5 & 5 \\ 6.5 & 7.5 & 9 & 8.5 & 7 & 7 \\ 6 & 8 & 10 & 9.5 & 9 & 9 \\ 6 & 8 & 10 & 9.5 & 9 & 9 \end{bmatrix}$$

FIGURE 10 (Enlarged) CWM smoother output (left), recursive CWM smoother output (center), and permutation CWM smoother output (right). Window size is 5×5. (See color insert.)

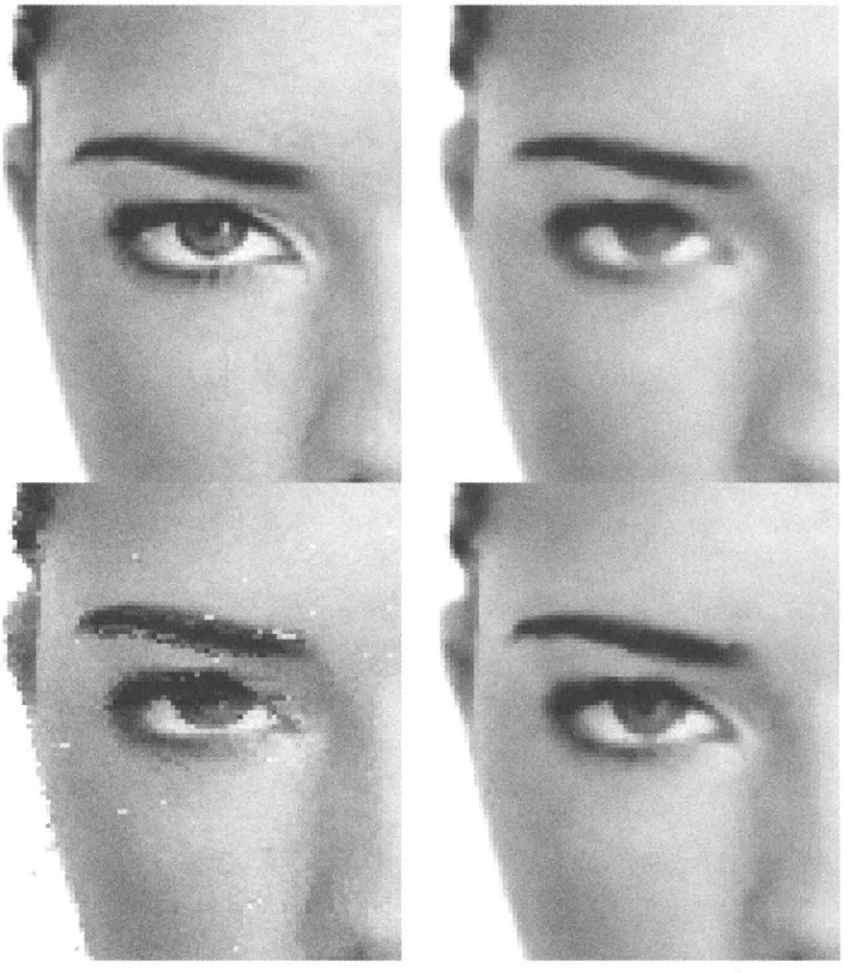

FIGURE 11 (a) Original image, (b) filtered image using marginal WM filter, (c) filtered image using vector WM filter, (d) filtered image using the marginal WMM filter. (See color insert.)

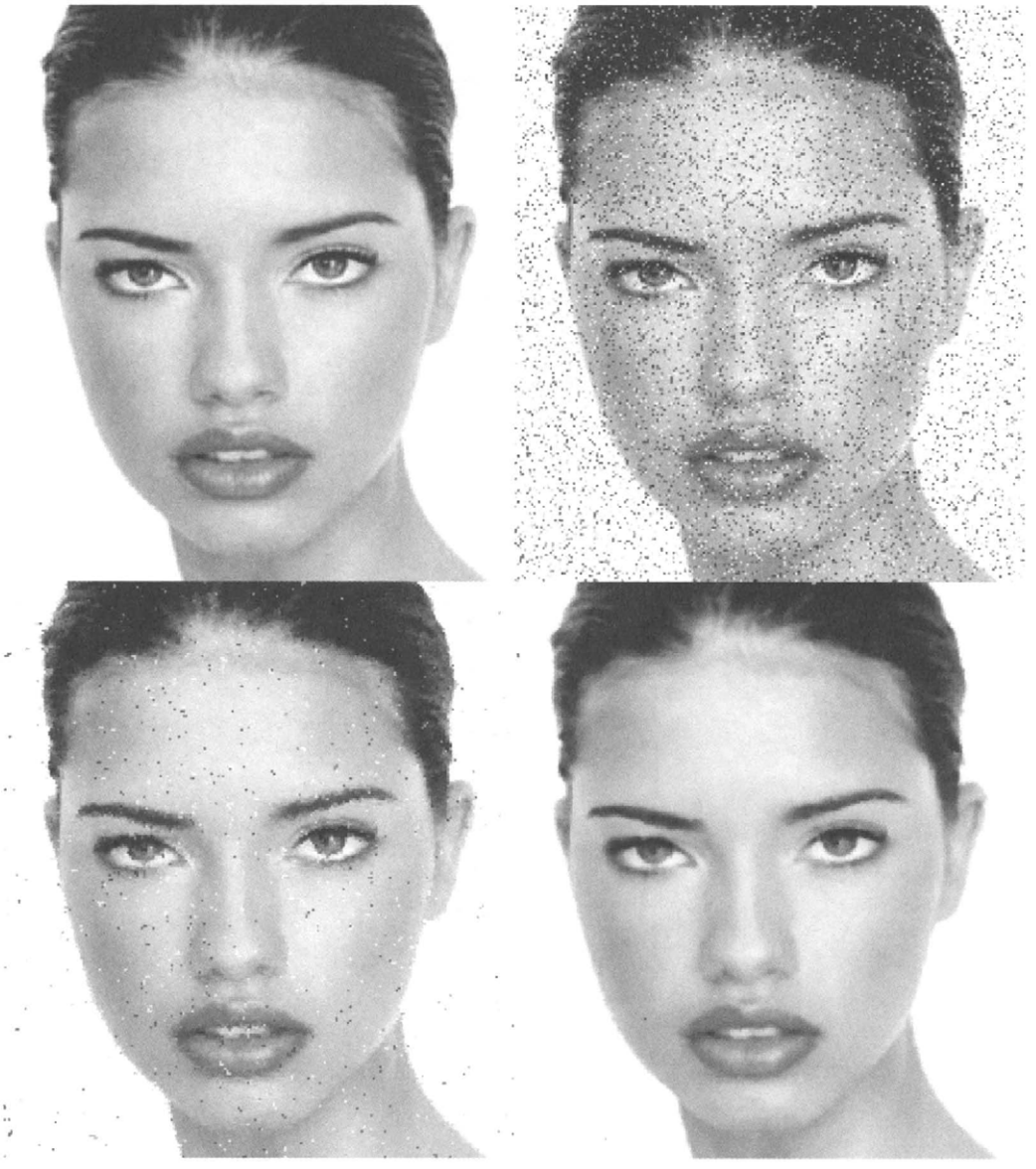

FIGURE 12 Multivariate medians for color images in "salt and pepper" noise. From topleft clockwise: noiseless image, contaminated image, WVM with 3×3 window, marginal WMM with 3×3 window. (See color insert.)

Zooming commonly requires a change in the image dimensions by a non-integer factor, such as a 50% zoom where the dimensions must be 1.5 times the original. Also, a change in the length-to-width ratio might be needed if the horizontal and vertical zoom factors are different. The simplest way to accomplish zooming of arbitrary scale is to double the size of the original as many times as needed to obtain an image larger than the target size in all dimensions, interpolating new pixels on each expansion. Then the desired image can be attained by subsampling the larger image, or taking pixels at regular intervals from the larger image in order to obtain an image with the correct length and width. The subsampling of images and the possible filtering needed are topics well known in traditional image processing, thus, we will focus on the problem of doubling the size of an image.

A digital image is represented by an array of values, each value defining the color of a pixel of the image. Whether the color is constrained to be a shade of gray, in which case only one value is needed to define the brightness of each pixel, or whether three values are needed to define the red, green, and blue components of each pixel does not affect the definition of the technique of weighted median interpolation. The only difference between gray scale and color images is that an ordinary weighted median is used in gray scale images while color requires a vector weighted median.

FIGURE 13 Multivariate medians for color images in salt and pepper noise (continued). WVM with 5×5 window (left), and marginal WMM with 5×5 window (right). (See color insert.)

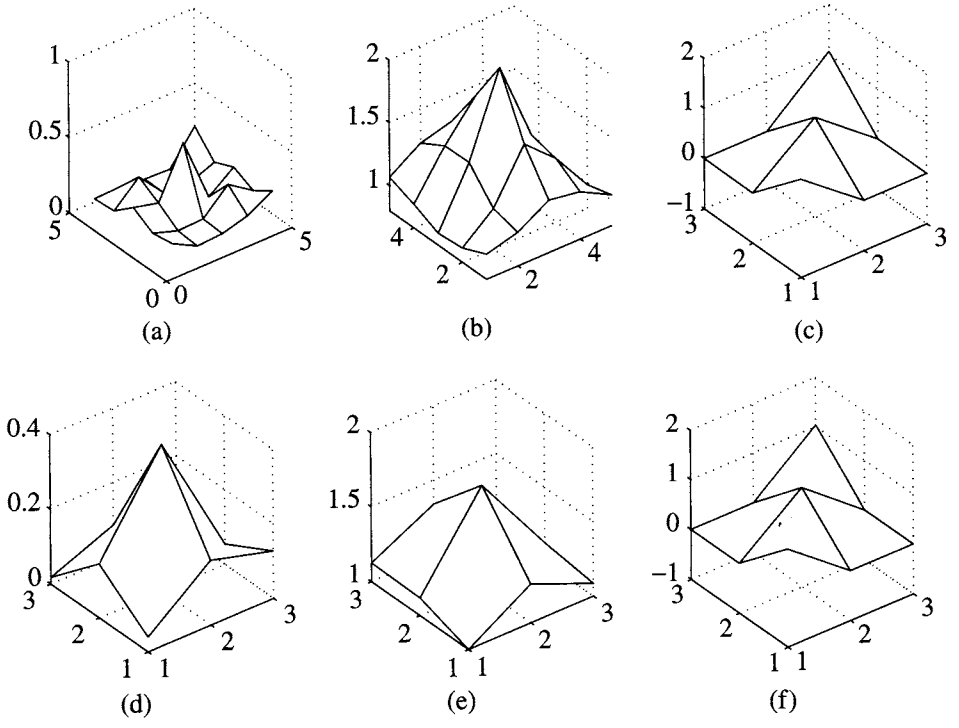

FIGURE 14 Optimum weights for the multivariate medians for color images in "salt and pepper" noise, (a) 5×5 WVM, (b) $\underline{\mathbf{V}}$ in 5×5 marginal WMM I, (c) $\mathbf{W}$ in 5×5 marginal WMM I, (d) 3×3 WVM, (e) $\underline{\mathbf{V}}$ in 3×3 marginal WMM I, (f) $\mathbf{W}$ in 3×3 marginal WMM I.

To double the size of an image, first an empty array is constructed with twice the number of rows and columns as the original (Fig. 15(a)), and the original pixels are placed into alternating rows and columns (the "00" pixels in Fig. 15(a)). To interpolate the remaining pixels, the method known as polyphase interpolation is used. In the method, each new pixel with four original pixels at its four corners (the "11" pixels in Fig. 15(b)) is interpolated first by using the weighted median of the four nearest original pixels as the value for that pixel. Since all original pixels are equally trustworthy and the

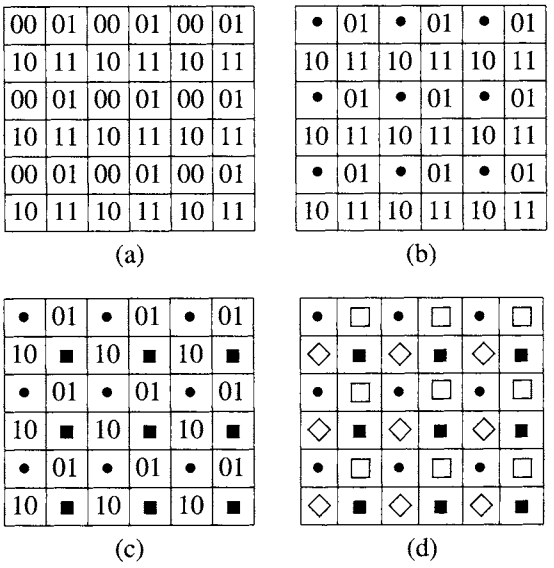

FIGURE 15 The steps of polyphase interpolation.

same distance from the pixel being interpolated, a weight of 1 is used for the four nearest original pixels. The resulting array is shown in Fig. 15(c). The remaining pixels are determined by taking a weighted median of the four closest pixels. Thus each of the "01" pixels in Fig. 15(c) is interpolated using two original pixels to the left and right and two previously interpolated pixels above and below. Similarly, the "10" pixels are interpolated with original pixels above and below and interpolated pixels ("11" pixels) to the right and left.

Since the "11" pixels were interpolated, they are less reliable than the original pixels and should be given lower weights in determining the "01" and "10" pixels. Therefore the "11" pixels are given weights of 0.5 in the median to determine the "01" and "10" pixels, while the "00" original pixels have weights of 1 associated with them. The weight of 0.5 is used because it implies that when both "11" pixels have values that are not between the two "00" pixel values then one of the "00" pixels or their average will be used. Thus "11" pixels differing from the "00" pixels do not greatly affect the result of the weighted median. Only when the "11" pixels lie between the two "00" pixels will they have a direct effect on the interpolation. The choice of 0.5 for the weight is arbitrary, since any weight greater than 0 and less than 1 will produce the same result. When implementing the polyphase method, the "01" and "10" pixels must be treated differently due to the fact that the orientation of the two closest original pixels is different for the two types of pixels. Figure 15(d) shows the final result of doubling the size of the original array.

To illustrate the process, consider an expansion of the grayscale image represented by an array of pixels, the pixel in the ith row and jth column having brightness $a_{i,j}$. The array $a_{i,j}$ will be interpolated into the array $x_{i,j}^{pq}$, with p and q taking values 0 or 1 indicating in the same way as above the type of interpolation required:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \Rightarrow \begin{bmatrix} x_{1,1}^{00} & x_{1,1}^{01} & x_{1,2}^{00} & x_{1,2}^{01} & x_{1,3}^{00} & x_{1,3}^{01} \\ x_{1,1}^{10} & x_{1,1}^{11} & x_{1,2}^{10} & x_{1,2}^{11} & x_{1,3}^{10} & x_{1,3}^{11} \\ x_{2,1}^{00} & x_{2,1}^{01} & x_{2,2}^{00} & x_{2,2}^{01} & x_{2,3}^{00} & x_{2,3}^{01} \\ x_{2,1}^{10} & x_{2,1}^{11} & x_{2,2}^{10} & x_{2,2}^{11} & x_{2,3}^{10} & x_{2,3}^{11} \\ x_{3,1}^{00} & x_{3,1}^{01} & x_{3,2}^{00} & x_{3,2}^{01} & x_{3,3}^{00} & x_{3,3}^{01} \\ x_{3,1}^{10} & x_{3,1}^{11} & x_{3,2}^{10} & x_{3,2}^{11} & x_{3,3}^{10} & x_{3,3}^{11} \end{bmatrix}.$$

The pixels are interpolated as follows:

$$x_{i,j}^{00} = a_{i,j}$$

$$x_{i,j}^{11} = \text{MEDIAN}[a_{i,j},\ a_{i+1,j},\ a_{i,j+1},\ a_{i+1,j+1}]$$

$$x_{i,j}^{01} = \text{MEDIAN}[a_{i,j},\ a_{i,j+1},\ 0.5 \diamond x_{i-1,j}^{11},\ 0.5 \diamond x_{i+1,j}^{11}]$$

$$x_{i,j}^{10} = \text{MEDIAN}[a_{i,j},\ a_{i+1,j},\ 0.5 \diamond x_{i,j-1}^{11},\ 0.5 \diamond x_{i,j+1}^{11}].$$

An example of median interpolation compared with bilinear interpolation is given in Fig. 16. Bilinear interpolation uses the average of the nearest two original pixels to interpolate the "01" and "10" pixels in Fig. 15(b) and the average of the nearest four original pixels for the "11" pixels. The edge-preserving advantage of the weighted median interpolation is readily seen in the figure.

5 Image Sharpening

Human perception is highly sensitive to edges and fine details of an image and since they are composed primarily by high-frequency components, the visual quality of an image can be enormously degraded if the high frequencies are attenuated or completely removed. On the other hand, enhancing the high-frequency components of an image leads to an improvement in the visual quality. *Image sharpening refers to any enhancement technique that highlights edges and fine details in an image.* Image sharpening is widely used in printing and photographic industries for increasing the local contrast and sharpening the images. In principle, image sharpening consists in adding to the original image a signal that is proportional to a high-pass filtered version of the original image. Figure 17 illustrates this procedure often referred to as unsharp masking [34, 35] on a 1 Dimensional signal. As shown in Fig. 17, the original image is first filtered by a high-pass filter which extracts the high frequency components, and then a scaled version of the high-pass filter output is added to the original image producing thus a sharpened image of the original. Note that the homogeneous regions of the

FIGURE 16 Example of zooming. Original is at the top with the area of interest outlined in white. On the lower left is the bilinear interpolation of the area, and on the lower right the weighted median interpolation.

signal, i.e., where the signal is constant, remain unchanged. The sharpening operation can be represented by

$$s_{i,j} = x_{i,j} + \lambda * \mathcal{F}(x_{i,j}) \tag{45}$$

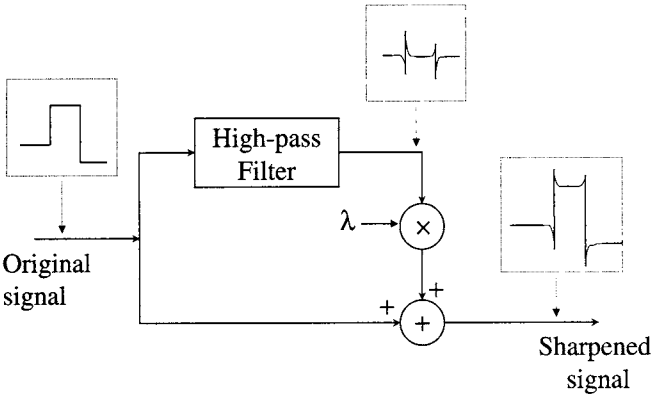

FIGURE 17 Image sharpening by high frequency emphasis.

where $x_{i,j}$ is the original pixel value at the coordinate (i, j), $\mathcal{F}(\cdot)$ is the high-pass filter, λ is a tuning parameter greater than or equal to zero, and $s_{i,j}$ is the sharpened pixel at the coordinate (i, j). The value taken by λ depends on the grade of sharpness desired. Increasing λ yields a more sharpened image.

If color images are used $x_{i,j}$, $s_{i,j}$, and λ are three-component vectors, whereas if gray-scale images are used $x_{i,j}$, $s_{i,j}$, and λ are single-component vectors. Thus the process described here can be applied to either gray-scale or color images with the only difference that vector filters have to be used in sharpening color images whereas single-component filters are used with gray-scale images.

The key point in the effective sharpening process lies in the choice of the high-pass filtering operation. Traditionally, linear filters have been used to implement the high-pass filter, however, linear techniques can lead to unacceptable results if the original image is corrupted with noise. A trade-off between noise attenuation and edge highlighting can be obtained if a weighted median filter with appropriated weights is used. To illustrate this, consider a WM filter

applied to a gray-scale image where the following filter mask is used

$$W = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \quad (46)$$

Due to the weight coefficients in (46), for each position of the moving window, the output is proportional to the difference between the center pixel and the smallest pixel around the center pixel. Thus, the filter output takes relatively large values for prominent edges in an image, and small values in regions that are fairly smooth, being zero only in regions that have constant gray level.

Although this filter can effectively extract the edges contained in a image, the effect that this filtering operation has over negative-slope edges is different from that obtained for positive-slope edges.[1] Since the filter output is proportional to the difference between the center pixel and the smallest pixel around the center, for negative-slope edges, the center pixel takes small values producing small values at the filter output. Moreover, the filter output is zero if the smallest pixel around the center pixel and the center pixel have the same values. This implies that negative-slope edges are not extracted in the same way as positive-slope edges. To overcome this limitation the basic image sharpening structure shown in Fig. 17, must be modified such that positive-slope edges as well as negative-slope edges are highlighted in the same proportion. A simple way to accomplish that is: (a) extract the positive-slope edges by filtering the original image with the filter mask described above; (b) extract the negative-slope edges by first pre-processing the original image such that the negative-slope edges become positive-slope edges, and then filter the pre-processed image with the filter described above; (c) combine appropriately the original image, the filtered version of the original image and the filtered version of the pre-processed image to form the sharpened image.

Thus both positive-slope edges and negative-slope edges are equally highlighted. This procedure is illustrated in Fig. 18, where the top branch extracts the positive-slope edges and the middle branch extracts the negative-slope edges. In order to understand the effects of edge sharpening, a row of a test image is plotted in Fig. 19 together with a row of the sharpened image when only the positive-slope edges are highlighted Fig. 19(a), only the negative-slope edges are highlighted Fig. 19(b), and both positive-slope and negative-slope edges are jointly highlighted Fig. 19(c).

[1]A change from a gray level to a lower gray level is referred to as a negative-slope edge, whereas a change from a gray level to a higher gray level is referred to as a positive-slope edge.

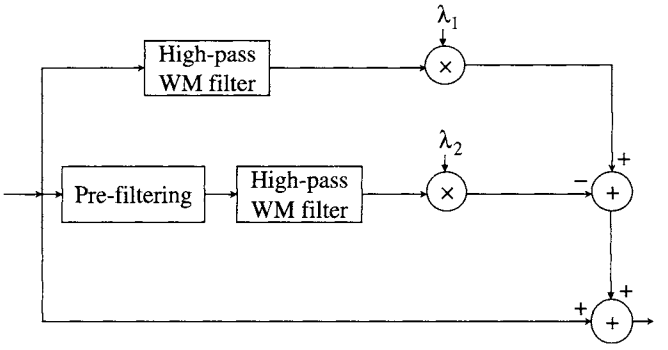

FIGURE 18 Image sharpening based on the weighted median filter.

In Fig. 18, λ_1 and λ_2 are tuning parameters that control the amount of sharpness desired in the positive-slope direction and in the negative-slope direction respectively. The values of λ_1 and λ_2 are generally selected to be equal. The output of the pre-filtering operation is defined as

$$x'_{i,j} = M - x_{i,j} \quad (47)$$

with M equal to the maximum pixel value of the original image. This pre-filtering operation can be thought of as a flipping and a shifting operation of the values of the original image such that the negative-slope edges are converted in positive-slope edges. Since the original image and the pre-filtered image are filtered by the same WM filter, the positive-slope edges and negative-slopes edges are sharpened in the same way.

In Fig. 20, the performance of the WM filter image sharpening is compared with that of traditional image sharpening

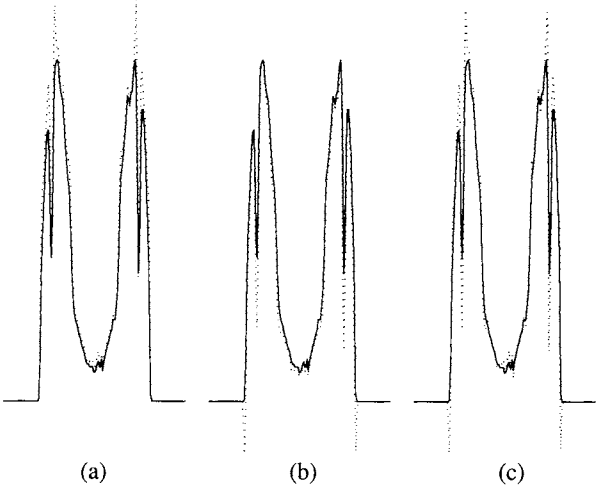

FIGURE 19 Original row of a test image (solid line) and row sharpened (dotted line) with (a) only positive-slope edges, (b) only negative-slope edges, and (c) both positive-slope and negative-slope edges.

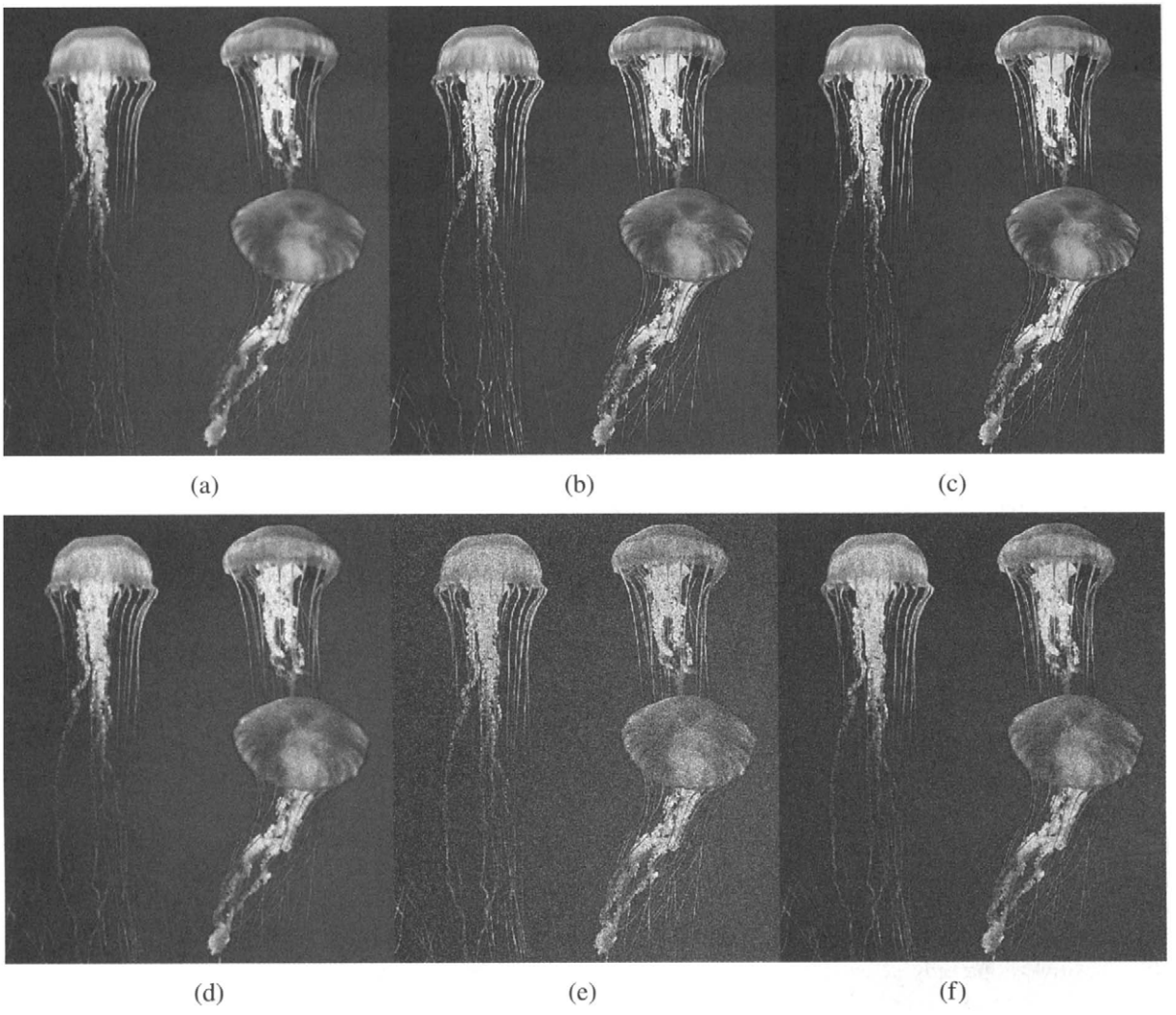

FIGURE 20 (a) Original image sharpened with (b) the FIR-sharpener, and (c) with the WM-sharpener. (d) Image with added Gaussian noise sharpened with (e) the FIR sharpener, and (f) the WM sharpener. (See color insert.)

based on linear FIR filters. For the linear sharpener, the scheme shows in Fig. 17 was used. The parameter λ was set to 1 for the clean image and to 0.75 for the noise image. For the WM sharpener, the scheme of Fig. 18 was used with $\lambda_1 = \lambda_2 = 2$ for the clean image, and $\lambda_1 = \lambda_2 = 1.5$ for the noise image. The filter mask given by (46) was used in both linear and median image sharpening. As before each component of the color image was processed separately.

6 Edge Detection

Edge detection is an important tool in image analysis, and is necessary for applications of computer vision in which objects need to be recognized by their outlines. An edge detection algorithm should show the locations of major edges in the image while ignoring false edges caused by noise. The most common approach used for edge detection is illustrated in Fig. 21. A high-pass filter is applied to the image to obtain the amount of change present in the image at every pixel. The output of the filter is thresholded to determine those pixels which have a high enough rate of change to be considered lying on an edge, i.e., all pixels with filter output greater than some value T are taken as edge pixels. The value of T is a tunable parameter which can be adjusted to give the best visual results. High thresholds lose some of the real edges, while low values result in many false edges, thus a tradeoff needs to be made to get the best results. Other techniques such as edge thinning can be applied to further pinpoint the location of the edges in an image.

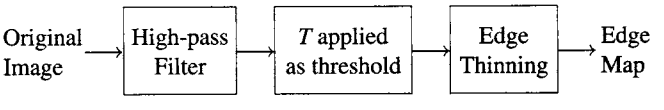

FIGURE 21 The process of edge detection.

The most common linear filter used for the initial high-pass filtering is the Sobel operator, which uses the following 3×3 masks (Sobel masks):

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

These two masks are convolved with the image separately to measure the strength of horizontal edges and vertical edges, respectively, present at each pixel. Thus if the amount to which a horizontal edge is present at the pixel in the i-th row and j-th column is represented as $E^h_{i,j}$, and if the vertical edge indicator is $E^v_{i,j}$, then the values are:

$$\begin{aligned} E^h_{i,j} &= -x_{i-1,j-1} - 2x_{i-1,j} - x_{i-1,j+1} + x_{i+1,j-1} \\ &\quad + 2x_{i+1,j} + x_{i+1,j+1} \\ E^v_{i,j} &= -x_{i-1,j-1} - 2x_{i,j-1} - x_{i+1,j-1} + x_{i-1,j+1} \\ &\quad + 2x_{i,j+1} + x_{i+1,j+1} \end{aligned}$$

The two strengths are combined to find the total amount to which any edge exists at the pixel: $E^{total}_{i,j} = \sqrt{{E^h_{i,j}}^2 + {E^v_{i,j}}^2}$. This value is then compared to the threshold T to determine the existence of an edge.

In place of using linear high-pass filters, weighted median filters can be used. To apply weighted medians to the high-pass filtering, the weights from the Sobel masks can be used. The Sobel linear high-pass filters take a weighted difference between the pixels on either side of $x_{i,j}$. On the other hand, if the same weights are used in a weighted median filter, the value returned is the difference between the lowest-valued pixels on either side of $x_{i,j}$. If the pixel values are then flipped about some middle value, the difference between the *highest* pixels on either side can also be obtained. The flipping can be achieved by finding some maximum pixel value M and using $x'_{i,j} = M - x_{i,j}$ as the "flipped" value of $x_{i,j}$, thus causing the highest values to become the lowest. The lower of the two differences across the pixel can then be used as the indicator of the presence of an edge. If there is a true edge present, then both differences should be high in magnitude, while if noise causes one of the differences to be too high, the other difference is not necessarily affected. Thus the horizontal and vertical edge indicators are:

$$E^h_{i,j} = \min\left(\text{MEDIAN}\begin{bmatrix} -1\diamond x_{i-1,j-1}, & -2\diamond x_{i-1,j}, & -1\diamond x_{i-1,j+1}, \\ 1\diamond x_{i+1,j-1}, & 2\diamond x_{i+1,j}, & 1\diamond x_{i+1,j+1} \end{bmatrix}, \text{MEDIAN}\begin{bmatrix} -1\diamond x'_{i-1,j-1}, & -2\diamond x'_{i-1,j}, & -1\diamond x'_{i-1,j+1}, \\ 1\diamond x'_{i+1,j-1}, & 2\diamond x'_{i+1,j}, & 1\diamond x'_{i+1,j+1} \end{bmatrix}\right)$$

$$E^v_{i,j} = \min\left(\text{MEDIAN}\begin{bmatrix} -1\diamond x_{i-1,j-1}, & 1\diamond x_{i-1,j+1}, \\ -2\diamond x_{i,j-1}, & 2\diamond x_{i,j+1}, \\ -1\diamond x_{i+1,j-1}, & 1\diamond x_{i+1,j+1} \end{bmatrix}, \text{MEDIAN}\begin{bmatrix} -1\diamond x'_{i-1,j-1}, & 1\diamond x'_{i-1,j+1}, \\ -2\diamond x'_{i,j-1}, & 2\diamond x'_{i,j+1}, \\ -1\diamond x'_{i+1,j-1}, & 1\diamond x'_{i+1,j+1} \end{bmatrix}\right)$$

and the strength of horizontal and vertical edges $E^{h,v}_{(i,j)}$ is determined in the same way as the linear case: $E^{h,v}_{(i,j)} = \sqrt{{E^h_{i,j}}^2 + {E^v_{i,j}}^2}$.

Another addition to the weighted median method is necessary in order to detect diagonal edges. Horizontal and vertical indicators are not sufficient to register diagonal edges, so the following two masks must also be used:

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}.$$

These masks can be applied to the image just as the Sobel masks above. Thus the strengths of the two types of diagonal edges are $E^{d1}_{i,j}$ for diagonal edges going from the bottom left of the image to the top right (using the mask on the left above) and $E^{d2}_{i,j}$ for diagonal edges from top left to bottom right (the mask on the right), and the values are given by:

$$E^{d1}_{i,j} = \min\left(\text{MEDIAN}\begin{bmatrix} -2\diamond x_{i-1,j-1}, & -1\diamond x_{i-1,j}, \\ -1\diamond x_{i,j-1}, & 1\diamond x_{i,j+1}, \\ 1\diamond x_{i+1,j}, & 2\diamond x_{i+1,j+1} \end{bmatrix}, \text{MEDIAN}\begin{bmatrix} -2\diamond x'_{i-1,j-1}, & -1\diamond x'_{i-1,j}, \\ -1\diamond x'_{i,j-1}, & 1\diamond x'_{i,j+1}, \\ 1\diamond x'_{i+1,j}, & 2\diamond x'_{i+1,j+1} \end{bmatrix}\right)$$

$$E^{d2}_{i,j} = \min\left(\text{MEDIAN}\begin{bmatrix} 1\diamond x_{i-1,j}, & 2\diamond x_{i-1,j+1}, \\ -1\diamond x_{i,j-1}, & 1\diamond x_{i,j+1}, \\ -2\diamond x_{i+1,j-1}, & -1\diamond x_{i+1,j} \end{bmatrix}, \text{MEDIAN}\begin{bmatrix} 1\diamond x'_{i-1,j}, & 2\diamond x'_{i-1,j+1}, \\ -1\diamond x'_{i,j-1}, & 1\diamond x'_{i,j+1}, \\ -2\diamond x'_{i+1,j-1}, & -1\diamond x'_{i+1,j} \end{bmatrix}\right).$$

(a) (b) (c)

FIGURE 22 (a) Original image, (b) edge detector using linear method, and (c) median method.

A diagonal edge strength is determined in the same way as the horizontal and vertical edge strength above: $E_{i,j}^{d1,d2} = \sqrt{E_{i,j.}^{d1\,2} + E_{i,j.}^{d2\,2}}$. The indicator of all edges in any direction is the maximum of the two strengths $E_{i,j}^{h,v}$ and $E_{i,j}^{d1,d2}$: $E_{i,j}^{total} = \max\left(E_{i,j}^{h,v}, E_{i,j}^{d1,d2}\right)$. As in the linear case, this value is compared to the threshold T to determine whether a pixel lies on an edge. Figure 22 shows the results of calculating $E_{i,j}^{total}$ for an image. The results of the median edge detection are similar to the results of using the Sobel linear operator. Other approaches for edge detector based on median filter can be found in [36–39].

7 Conclusion

The principles behind WM smoothers and WM filters have been presented in this article, as well as some of the applications of these nonlinear signal processing structures in image enhancement. It should be apparent to the reader that many similarities exist between linear and median filters. As illustrated in this article, there are several applications in image enhancement were WM filters provide significant advantages over traditional image enhancement methods using linear filters. The methods presented here, and other image enhancement methods that can be easily developed using WM filters, are computationally simple and provide significant advantages, and consequently can be used in emerging consumer electronic products, PC and internet imaging tools, medical and biomedical imaging systems, and of course in military applications.

Acknowledgment

This work was supported in part by the National Science Foundation under grant MIP-9530923.

References

[1] Y. H. Lee and S. A. Kassam, "Generalized median filtering and related nonlinear filtering techniques," *IEEE Trans. on Acous., Speech, and Signal Processing*, 33, June 1985.

[2] J. W. Tukey, "Nonlinear (nonsuperimposable) methods for smoothing data," in *Conf. Rec.*, (Eascon), 1974.

[3] T. A. Nodes and N. C. Gallagher, Jr., "Median filters: some modifications and their properties," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 30, 739–746, Oct. 1982.

[4] G. R. Arce and N. C. Gallagher, "Stochastic analysis of the recursive median filter process," *IEEE Transactions on Information Theory*, IT-34, July 1988.

[5] G. R. Arce, "Statistical threshold decomposition for recursive and nonrecursive median filters," *IEEE Trans. on Information Theory*, IT-32, 243–253, Mar. 1986.

[6] E. L. Lehmann, *Theory of point estimation*. New York: Wiley, 1983.

[7] A. C. Bovik, T. S. Huang, and J. D. C. Munson, "A generalization of median filtering using linear combinations of order statistics," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31, Dec. 1983.

[8] H. A. David, *Order statistics*. New York: Wiley Interscience, 1981.

[9] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A First Course in Order Statistics*. New York: Wiley, 1992.

[10] F. Y. Edgeworth, "A new method of reducing observations relating to several quantities," *Phil. Mag. (Fifth Series)*, 24, 222–223, 1887.

[11] D. R. K. Brownrigg, "The weighted median filter," *Communications of the ACM*, 27, 807–818, Aug. 1984.

[12] S. -J. Ko and Y. H. Lee, "Center weighted median filters and their applications to image enhancement," *IEEE Transactions on Circuits and Systems*, 38, Sept. 1991.

[13] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: a tutorial," *Trans. on Circuits and Systems II*, 41, May 1996.

[14] O. Yli-Harja, J. Astola, and Y. Neuvo, "Analysis of the properties of median and weighted median filters using

threshold logic and stack filter representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 39, Feb. 1991.

[15] G. R. Arce, T. A. Hall, and K. E. Barner, "Permutation weighted order statistic filters," *IEEE Transactions on Image Processing*, 4, Aug. 1995.

[16] R. C. Hardie and K. E. Barner, "Rank conditioned rank selection filters for signal restoration," *IEEE Transactions on Image Processing*, 3, Mar. 1994.

[17] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Median filtering by threshold decomposition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32, Dec. 1984.

[18] P. D. Wendt, E. J. Coyle, and N. C. Gallagher, Jr., "Stack filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34, Aug. 1986.

[19] E. N. Gilbert, "Lattice-theoretic properties of frontal switching functions," *J. Math. Phys.*, 33, Apr. 1954.

[20] G. R. Arce, "A general weighted median filter structure admitting negative weights," *IEEE Transactions on Signal Processing*, 46, Dec. 1998.

[21] J. L. Paredes and G. R. Arce, "Stack filters, stack smoothers, and mirrored threshold decomposition," *IEEE Transactions on Signal Processing*, 47, 2757–2767, Oct. 1999.

[22] V. Barnett, "The ordering of multivariate data," *Journal of Royal Statistical Society A*, 139, 331–354, 1976.

[23] P. E. Trahanias, D. Karakos, and A. N. Venetsanopoulos, "Directional processing of color images: Theory and experimental results," *IEEE Transactions on Image Processing*, 5, 868–880, June 1996.

[24] R. Hardie and G. R. Arce, "Ranking in r^p and its use in multivariate image estimation," *IEEE Transactions on Video Technology*, 1, June 1991.

[25] I. Pitas and P. Tsakalides, "Multivariate ordering in color image filtering," *IEEE Transactions on Circuits and Systems for Video Technology*, 1, Feb. 1991.

[26] V. Koivunen, "Nonlinear filtering of multivariate images under robust error criterion," *IEEE Transactions on Image Processing*, 5, June 1996.

[27] C. Small, "A survey of multidimensional medians," *Int. Stat. Rev.*, 58, no. 3, 1990.

[28] V. Koivunen, N. Himayat, and S. A. Kassam, "Nonlinear filtering techniques for multivariate images, design and robustness characterization," *Signal Processing*, 57, 81–91, Feb. 1997.

[29] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, 78, 678–689, Apr. 1990.

[30] Y. Shen and K. Barner, "Fast optimization of weighted vector median filters," *IEEE Transactions on Signal Processing*, 2004. Submitted.

[31] E. A. Robinson, *Multichannel Time Series Analysis*. Houston, Texas: Goose Pond Press, 1983.

[32] Y. Li, G. R. Arce, and J. Bacca, "Generalized vector medians for correlated channels," in *Proceedings of the 2004 EUSIPCO European Signal Processing Conference*, (Vienna, Austria), Sept. 2004.

[33] A. C. Bovik, "Streaking in median filtered images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35, Apr. 1987.

[34] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

[35] J. S. Lim, *Two-dimensional Signal and Image Processing*. Englewood Cliffs, New Jersey, Prentice Hall, 1990.

[36] I. Pitas and A. Venetsanopoulos, "Edge detectors based on order statistics," *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8, July 1986.

[37] A. C. Bovik and D. C. Munson, "Edge detection using median comparisons," *Computer Vision, Graphics, and Image Processing*, 33, 1986.

[38] D. Lau and G. R. Arce, "Edge detector using weighted median filter," *Tech. Rep.* 97-05-15, Department of Computer and Electrical Engineering, University of Delaware, 1997.

[39] I. Pitas and A. N. Venetsanopoulos, "Nonlinear order statistic filters for image filtering and edge detection," *Signal Processing*, 10, 1986.

[40] S. Hoyos, Y. Li, J. Bacca, and G. R. Arce, "Weighted median filters admitting complex-valued weights and their optimization," *IEEE Transactions on Signal Processing*, 52, no. 10, Oct 2004, 2776–2787.

[41] S. Hoyos, J. Bacca, and G. R. Arce, "Spectral design of weighted median filters: A general iterative approach," *IEEE Transactions on Signal Processing*, 53, no. 3, March 2005.

[42] G. Arce, "Nonlinear Signal Processing A Statistical Approach," Wiley-Interscience, 2004.

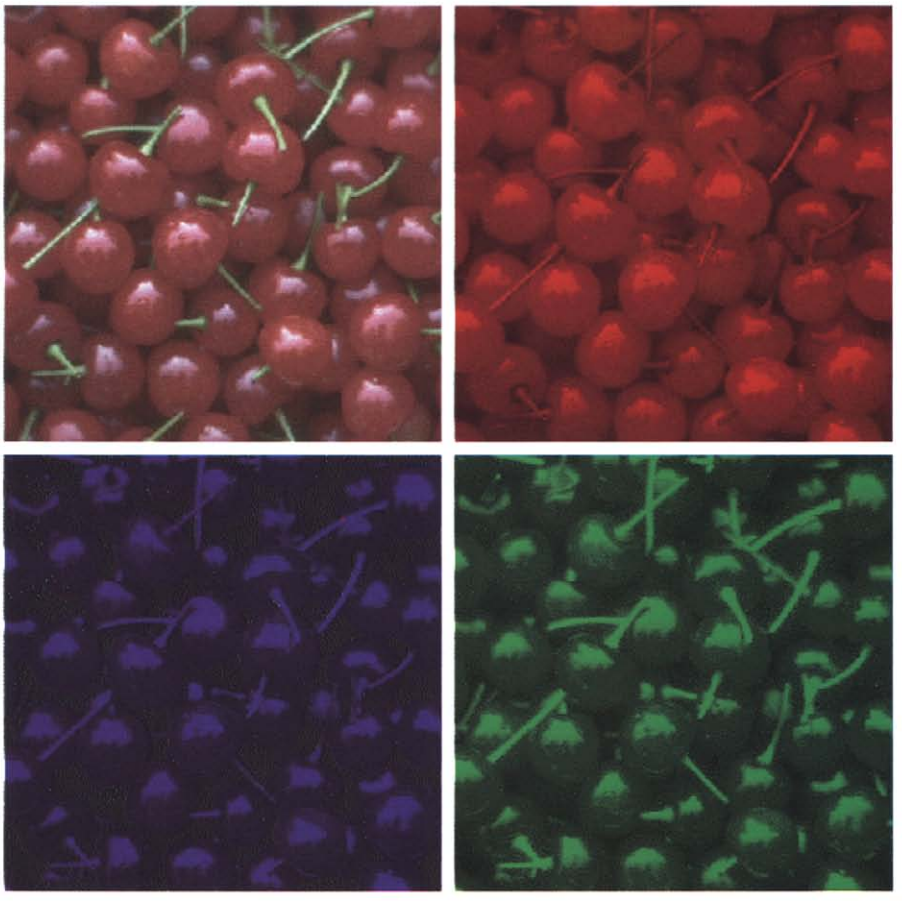

FIGURE 1.1.13 Color image "cherries" (top left) and (clockwise) its red, green, and blue components.

FIGURE 3.2.8 Impulse noise cleaning with a 5×5 CWM smoother: (top left) original "portrait" image, (top right) image with "salt and pepper" noise, (bottom left) CWM smoother with $W_c = 16$, (bottom right) CWM smoother with $W_c = 5$.

FIGURE 3.2.9 (Enlarged) Noise-free image (left), 5×5 median smoother output (center), and 5×5 mean smoother (right).

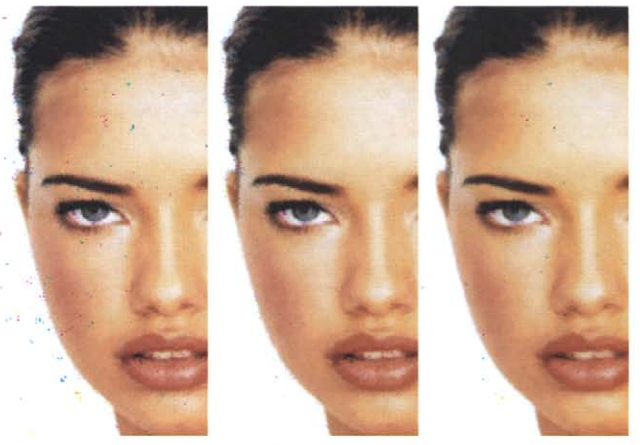

FIGURE 3.2.10 (Enlarged) CWM smoother output (left), Recursive CWM smoother output (center), and Permutation CWM smoother output (right). Window size is 5×5.

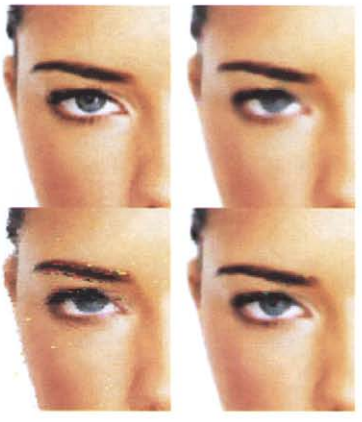

FIGURE 3.2.11 (top left) Original image, (top right) filtered image using Marginal WM filter, (bottom left) filtered image using Vector WM filter, (bottom right) filtered image using the Marginal WMM filter.

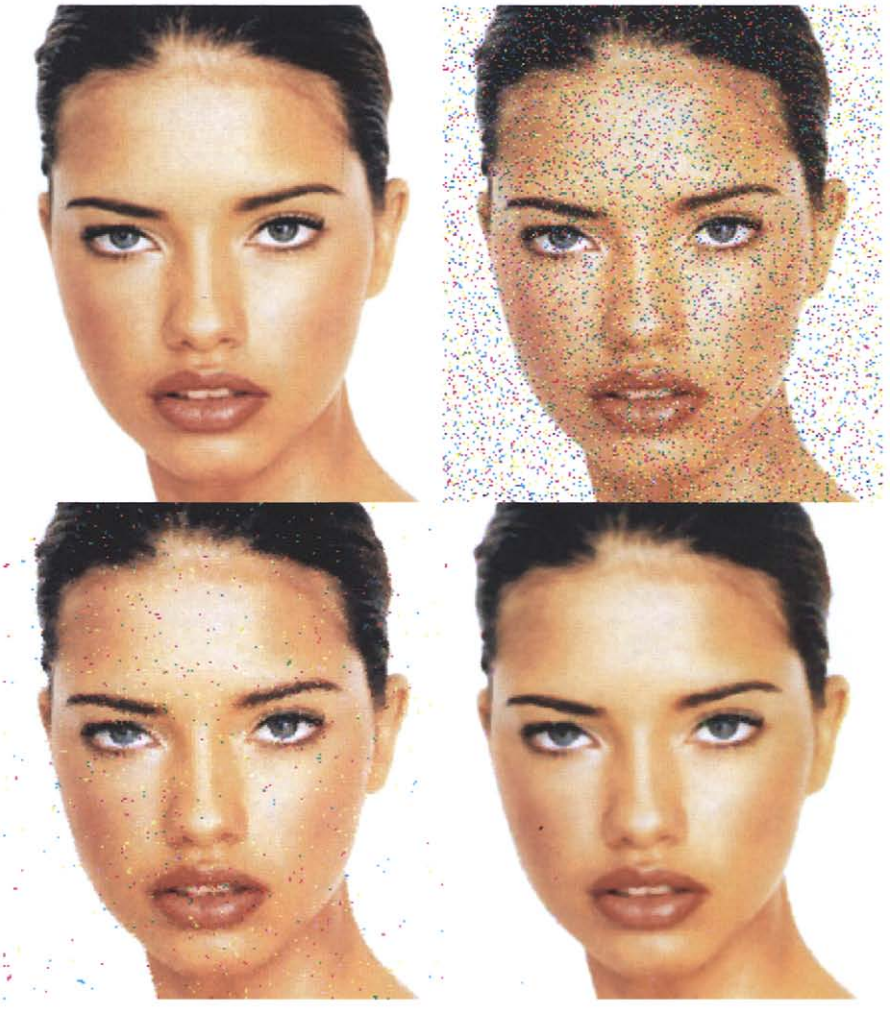

FIGURE 3.2.12 Multivariate medians for color images in "salt and pepper" noise. From left to right and top to bottom: noiseless image, contaminated image, WVM with 3×3 window, marginal WMM with 3×3 window.

FIGURE 3.2.13 Multivariate medians for color images in salt and pepper noise (continued). WVM with 5×5 window (left), and marginal WMM with 5×5 window (right).

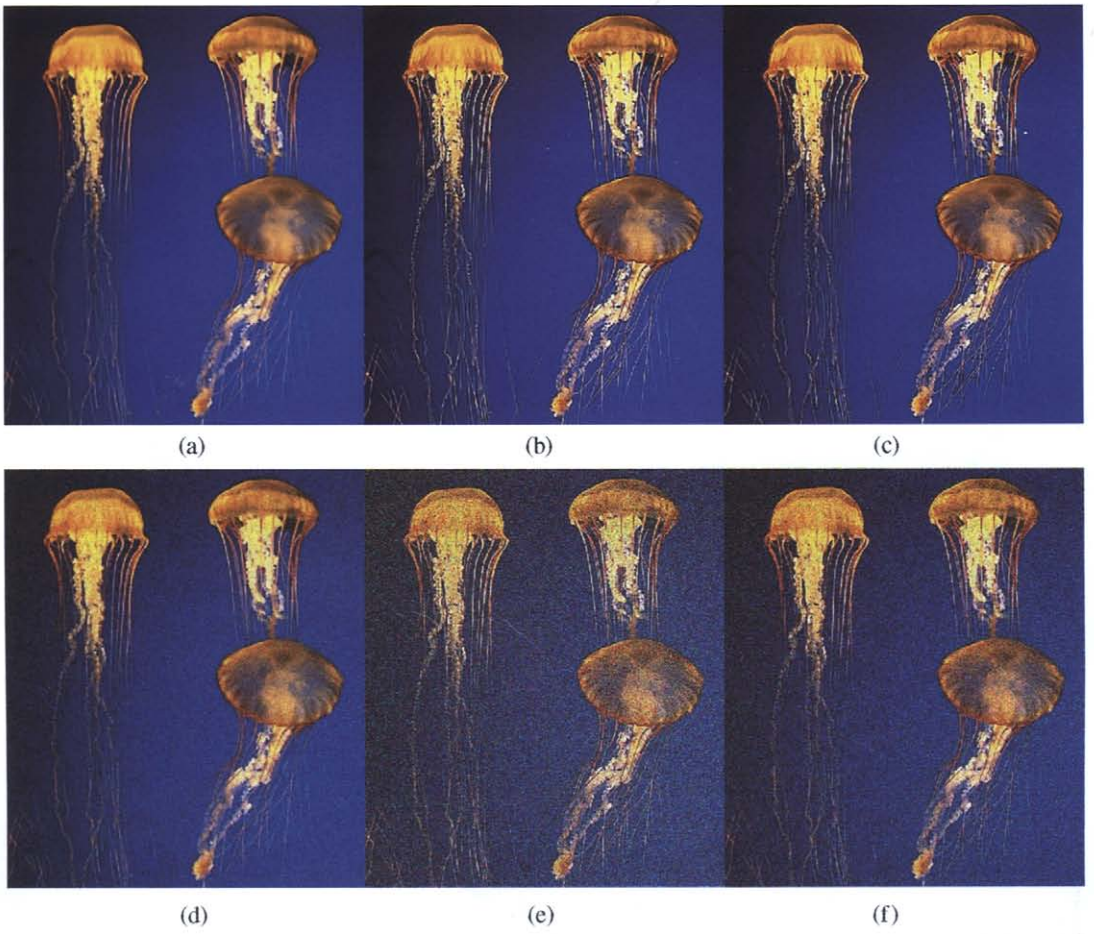

FIGURE 3.2.20 (a) Original image sharpened with (b) the FIR-sharpener, and (c) with the WM-sharpener. (d) Image with added Gaussian noise sharpened with (e) the FIR-sharpener, and (f) the WM-sharpener.