Department of Electrical and Computer Engineering Technical Reports

Department of Electrical and Computer Engineering

7-1-1984

Analysis and Implementation of Median Type Filters

J. P. Fitch
Purdue University

E. J. Coyle
Purdue University

N. C. Gallagher
Purdue University

Analysis and Implementation of Median Type Filters

J. P. Fitch
E. J. Coyle
N. C. Gallagher

TR-EE 84-20
July 1984

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ANALYSIS AND IMPLEMENTATION
OF
MEDIAN TYPE FILTERS

J.P. Fitch
E.J. Coyle
N.C. Gallagher

School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

TR-EE 84-20
July 1984

ACKNOWLEDGMENTS

The author would like to recognize the support, advice, direction, and friendship of both Neal Gallagher and Ed Coyle, without whom this work would have never begun or progressed. In addition, thanks to Professors Kak and Gleser for their technical discussions and for serving on my graduate committee.

Finally, the financial support provided by Purdue through University Fellowships and the National Science Foundation under grant ECS83-06235 is greatly appreciated.

TABLE OF CONTENTS

Page

APPENDICES

ABSTRACT

Median filters are a special class of ranked order filters used for smoothing signals. These filters have achieved success in speech processing, image processing, and other impulsive noise environments where linear filters have proven inadequate. Although the implementation of a median filter requires only a simple digital operation, its properties are not easily analyzed. Even so, a number of properties have been exhibited in the literature.

In this thesis, a new tool, known as threshold decomposition, is introduced for the analysis and implementation of median type filters. This decomposition of multi-level signals into sets of binary signals has led to significant theoretical and practical breakthroughs in the area of median filters. A preliminary discussion on using the threshold decomposition as an algorithm for a fast and parallel VLSI circuit implementation of ranked filters is also presented.

In addition, the theory is developed both for determining the number of signals which are invariant to arbitrary window width median filters when any number of quantization levels are allowed and for counting or estimating the number of passes required to produce a root-*i.e.* invariant signal, for binary signals. Finally, the analog median filter is defined and proposed for analysis of the standard discrete median filter in cases with a large sample size or when the associated statistics would be simpler in the continuum.

CHAPTER 1
INTRODUCTION

The purpose of most communication systems is the successful transmission of information. Even when the information is perfectly received some filtering may be necessary to prepare the signal for additional processing or to enhance the signal for human interpretation. Examples of applications where enhancement and noise free processing can occur include image processing and pattern recognition. It is more often the case, however, that some random element known as noise influences the correct reception of the signal. In order to design a "better" communication system, it is necessary to define an appropriate measure of error and a statistical model for the noisy system. This mathematical model is then used to determine the optimal technique for filtering the received signal. Frequently, it is too difficult or impossible to optimize over all possible techniques and so attention is restricted to finding the optimal technique within a class of filters. The filters in this more restrictive class should have some common mathematical properties which simplify the analysis necessary to design optimal filters within the class.

The class of linear functions has many properties which simplify the analysis necessary to design filters. This has allowed a rich theory for the design and implementation of optimal linear filters to be developed. An operator $\mathcal{L}(\cdot)$ is said to be linear if $\mathcal{L}(aX+bY) = a\cdot\mathcal{L}(X) + b\cdot\mathcal{L}(Y)$ for any real numbers a and b and inputs X and Y. This is known as the superposition property. From this we see that if $X = \sum_{i=1}^{k} a_i X_i$, then $\mathcal{L}(X)$ can be calculated by finding $a_i \mathcal{L}(X_i)$ and summing over all i. At first this may seem an incredible increase in computation, however, by proper selection of the decomposition of X into $\sum_{i=1}^{k} a_i X_i$ a reduction in total computations can sometimes be achieved. This is true whenever the $\mathcal{L}(X_i)'$ s are very easy to compute relative to $\mathcal{L}(X)$.

Exploitation of the superposition property has led to the development of many mathematical tools which simplify the description and design of linear filters. For instance, a linear system can be represented as the convolution of the input signal with the impulse response of the system. This representation can also be transformed to other domains. Fourier transform techniques are effective for designing filters when the "true" signal and the noise are spectrally separate. In short, the design and characterization of linear filters are well developed areas.

As expected, however, the best linear filter which can be designed for the mathematical model is not always the optimal filter for the model or for the application. Reasons for this lack of optimality include incorrect noise models, changing noise environments, and applications where the linear filter assumption is not appropriate. The underlying problem is that when most models are made simple enough to calculate optimal filtering procedures, they become inadequate approximations to the situation being modeled. In addition, there is the possibility of changing noise characteristics. When this occurs, even an optimal linear filter may be unable to adapt to the different noise when implemented. In short, many applications are not well modeled by linear systems or require some type of robustness-that is, the ability to perform well under various noise conditions.

Demonstration of additional weaknesses of linear filters requires a more precise definition of "filtering" and the object to be filtered. For the purpose of this thesis, a filter is defined by sliding a window across an input signal. At each position of the window the filter output is determined by some mathematical function operating exclusively on the values in the window. An example where the function is linear and is applied to a discrete time signal is provided in Figure 1.1. For this example, a constant window width of five values and the averaging operation, which is linear, define the filter.

Any other function could have been chosen but this simple averaging filter shows some of the shortcomings of linear filters. In image processing, linear filters tend to lengthen ramps, change step functions to ramps, and be sensitive to impulsive type noise [1]. This results in filtered images which appear blurred and unsharp. The "desired" output shown in Figure 1.1 is based on what the human eye prefers to see in images and the intuitive notion that statistical outliers should be discarded as bad information. This shows that other filters are needed. An important question is how to find them, since removal of the linear constraint causes great difficulty.

Perhaps the easiest approach to developing a filter with the intuitive properties described in the preceding paragraph is to start with a simple filter, analyze its weaknesses, and then propose modifications which will give the desired properties. For simplicity we select the averaging filter as the building

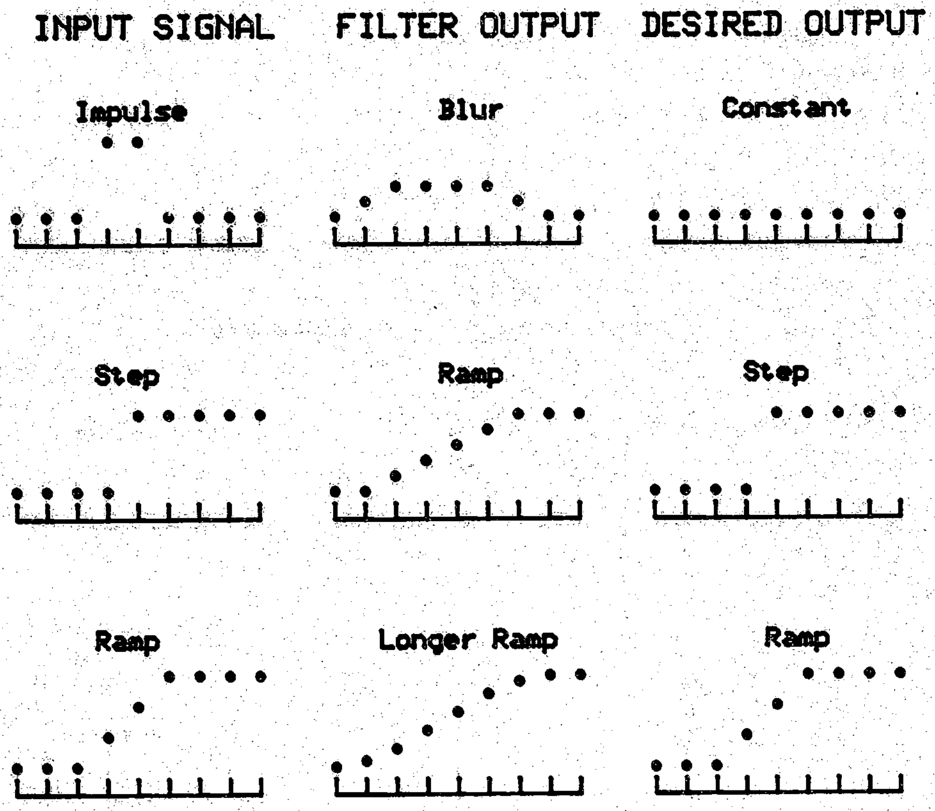

Figure 1.1
Averaging filter applied to three simple signals.

block. As was noted before, the averaging operation is an ineffective filter for impulsive type noise and for signals with sharp edges. The underlying mathematical reason for these sensitivities is that the arithmetic mean of N points can be completely controlled by just one of the values in the sample. Let $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$ denote the average of $X_1,\ldots,X_N$, then solving for X_1, we have

$$X_1 = N\bar{X} - \sum_{i=2}^{N} X_i$$

Therefore if $X_2, \ldots, X_N$ are given we can find an X_1 using the preceding equation such that $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$ for any desired $\bar{X}$. One method used to stabilize the arithmetic average is to eliminate from the summation the extreme values-that is, the values which are very large or small compared to the rest of the sample. To accomplish this it is necessary to rank the X_i' s.

A specific algorithm for stabilizing the estimate of the mean is the alpha trimmed mean [2], denoted by X_α and defined by

$$X_\alpha = \frac{1}{N-2\cdot[\alpha N]} \sum_{i=[\alpha N]+1}^{N-[\alpha N]} X_{(i)}$$

where $[\cdot]$ denotes the greatest integer function, $0 \leq \alpha \leq 0.5$, and $X_{(1)}, \ldots, X_{(N)}$ are the ordered values of the sample $X_1, \ldots, X_N$. That is, the sample is reordered from smallest to largest and labeled $X_{(1)}$ through $X_{(N)}$. As an example, consider the case where N is five and the sample is {-1,3,28,2,8}, then $X_{(1)} = -1$, $X_{(2)} = 2$, $X_{(3)} = 3$, $X_{(4)} = 8$, and $X_{(5)} = 28$. Filtering is done by performing this ranking operation at each position of the window and then taking the average of the middle most values as determined by alpha. We note that alpha equal to zero is simply an averaging filter. If alpha is a half and N is odd, say $N=2n+1$, then we have the other extreme case where the averaging operation is eliminated and the output of the trimmed estimator $X_{0.5}$ equals $X_{(n+1)}$ which is known as the median of the sample.

The median is well known as the minimum absolute error estimate and is even the maximum likelihood estimate of the mean for the two sided exponential process [3]. It is also a robust nonparametric estimate of the median for distributions which are symmetrically distributed about their medians [4]. An example of this is the problem of estimating the median of a Cauchy distribution from a set of independent, identically distributed (i.i.d.) samples. In this example, the mean of the underlying distribution is undefined and the arithmetic average as an estimate is, as expected, totally unstable.

The sample median is an alternative estimate for the center of the distribution. Motivated by these statistical properties of the median operator, Tukey [5,6] proposed the median as a robust sliding window filter which would reduce the effects of statistical outliers while smoothing signals. Despite the known properties of the median as an estimator, the mathematics necessary to analyze the effects of median filters on realistic signals are not simple extensions to the existing theory [1-11]. Output distributions of median filters can be nontrivial to derive and difficult to comprehend [7].

Even though the underlying theory is not well developed, the median filter's use as a practical filter has increased because it is easy to implement and appears to work well in many applications. As an introduction to the properties of the median operator, we note that a window width five median filter gives the desired results for an image processing filter as shown in Figure 1.1. Specifically, the median filter with an appropriate window width, for this example window width five, removes impulses but allows edges and ramps to pass unchanged. Signals which do not change when median filtered are known as root signals of the filter. Because of these properties, an effective use of median filters has been the reduction of high frequency and impulsive noise in digital images without the extensive blurring and edge destruction associated with linear filters [1]. Figure 1.2 demonstrates the effectiveness of the median filter in a particular real image where linear filters failed. Other applications include the smoothing of noisy pitch contours in speech signals and data compression using the root signal properties combined with a block truncation coding (BTC) technique [8-10]. Several fast algorithms for implementing the median filter [9,11,12], make it an even more appealing tool for signal processing applications.

The implementation of a standard median filter requires a simple nonlinear digital operation. To begin, take a sampled signal of length L; across this signal slide a window that spans $2n+1$ points. The filter output at each window position is given the same position as the sample at the center of the window and is set equal to the median value of the $2n+1$ signal samples in the window. Start up and end effects are accounted for by appending n samples to both the beginning and the end of the sequence. The front appended samples are given the value of the first signal sample; similarly, the rear appended samples receive the value of the last sample of the signal. In Figure 1.3 we present an example where a median filter of window width five, $n=2$, is applied to a signal with three levels. The output is given for each pass of the median filter until a root signal is reached. It has been shown that any signal can be filtered to a root in a finite number of passes [13]. Summarizing, the basic idea

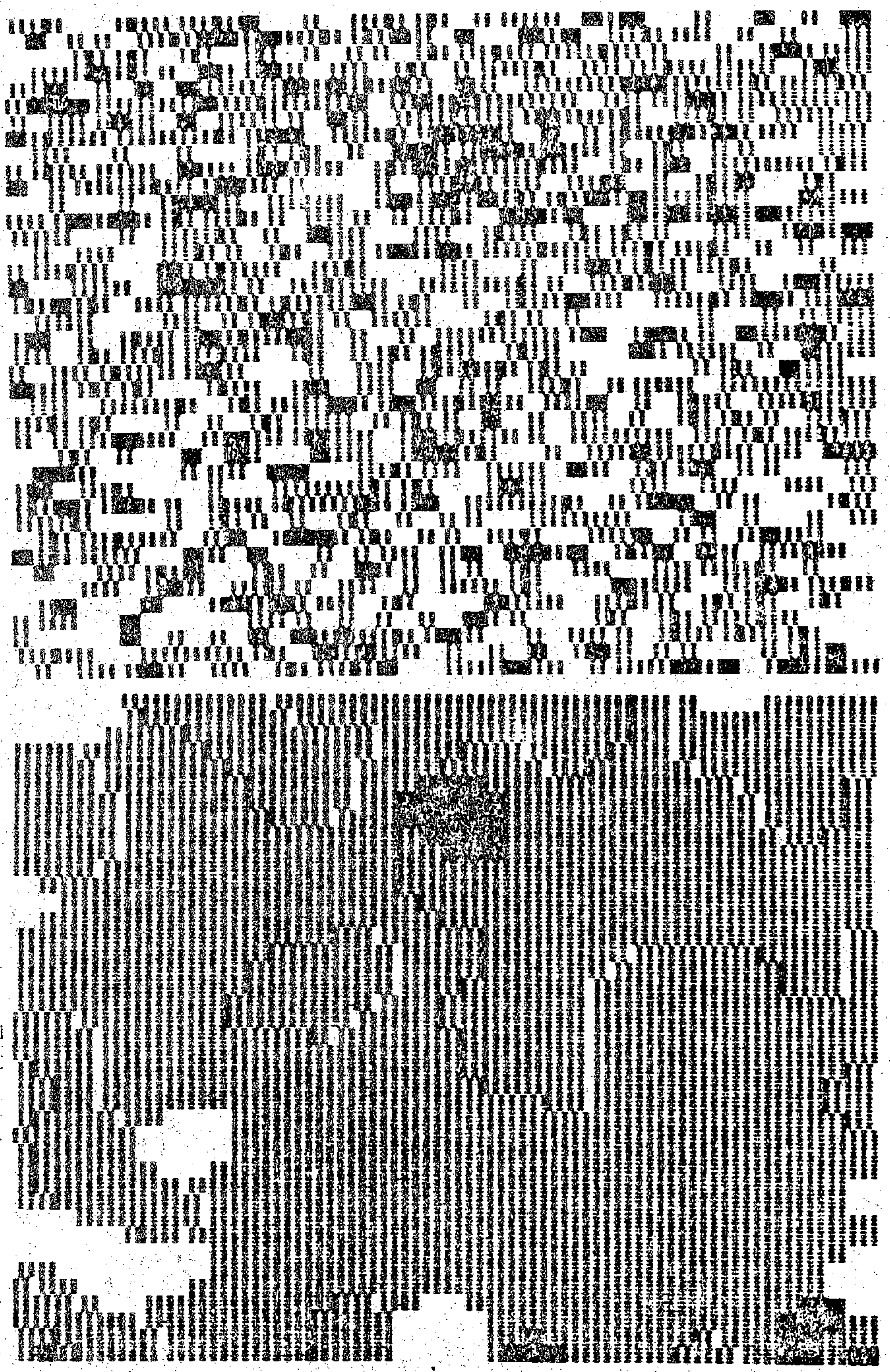

Figure 1.2
Median filtered output of a laser imaging system.

is to rank the samples in the window and select the median value as the filter output.

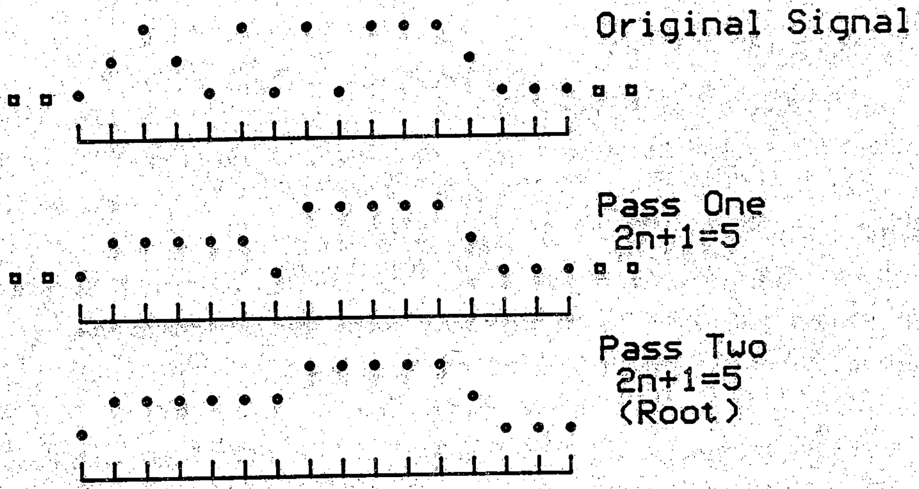

Figure 1.3
Convergence of a median filter to a root signal.

Although the algorithm is simple, the analysis and design of median filters is difficult because they are nonlinear. As was noted before, the median of the set of values {-1,3,28,2,8} is 3. Rewriting this set as {-1+0,3+0,22+6,-1+3,6+2} we note that median{-1,3,22,-1,6} equals 3 and median{0,0,6,3,2} equals 2 but their sum is 5 not 3. This means, of course, that the superposition property and all the results implied by this property do not hold. The elegant techniques for calculating the response of a linear filter by decomposing input signals into sets of simpler signals, analyzing the response of the filter to these simple signals, and then adding these individual responses together to obtain the total response of the system cannot be used. To insure that this major point was not overlooked, Tukey [5], even referred to the median filter as the "nonsuperposable filter". The complicated nature of ranked statistics and the lack of a superposition property have made this filter difficult to analyze and to design. Even though there is no general superposition property, any technique for decomposing input signals into sets of simpler signals which can be separately median filtered and then recombined would be significant in simplifying the analysis of median filters.

In this thesis, a special type of superposition is shown to hold for median filters. Using a particular decomposition of the input signal known as the threshold decomposition, the output of the filter can be determined by separately filtering each of the decomposed signals and then adding their resulting outputs. This powerful new tool allows problems in the analysis and the implementation of median filters for arbitrary level signals to be reduced to the equivalent problems for binary signals. Since the effects of median filters on binary signals are well understood, this technique has theoretical, practical, and intuitive benefits.

The theoretical analysis of the threshold decomposition technique is presented in Chapter 2. It is also shown that this decomposition-superposition scheme is also valid for a general class of multidimensional ranked order operators as well as for any linear filter. Because some of the proofs are similar, several specific proofs are relegated to the Appendices. The mathematics of the decomposition also lends itself to an extremely parallel, versatile, and fast algorithm for the VLSI implementation of the median filter. Chapter 3 is devoted to describing this important result of the decomposition.

In addition to the threshold decomposition for median type filters, several results on the root properties and convergence rates of median filters are also given. These can be found in Chapters 4 and 5. In Chapter 6, the analog median filter is defined and proposed for analysis of the standard discrete median filter in cases with a large sample size or when the associated statistics would be simpler in the continuum.

CHAPTER 2
THE THRESHOLD DECOMPOSITION

2.1 Definition of the Decomposition

We assume that the input signal is a discrete sequence of length L which takes on the value a(m) at position m, $1 \leq m \leq L$. For each m, a(m) is quantized to one of the k values 0,1,...,k-1. Because every countable set of real numbers can be mapped in an order preserving manner onto the integers, there is no loss of generality. In addition, this representation is appropriate because most signal processing tasks are now performed by digital computers.

The threshold decomposition is a technique for reducing a signal with k possible levels to a set of k-1 binary sequences without loss of information. We will show that this decomposition allows a superposition property for the nonlinear class of ranked order filters. Define the level i threshold decomposition of the original signal at point m to be

$$t_0^i(m) = \begin{cases} 1 & \text{,if } a(m) \geq i \\ 0 & \text{,if } a(m) < i \end{cases} \tag{2.1}$$

with $1 \leq m \leq L$ and $1 \leq i \leq k-1$. An example of this decomposition is given in Figure 2.1. Note that because $a(m) = \sum_{i=1}^{k} t_0^i(m)$, the threshold decomposition is a valid technique for breaking inputs to linear filters into binary signals for separate filtering. The output of the linear filter applied directly to the input process $\{a(m)\}$ can be obtained by summing the filter output from all the $\{t_0^i(m)\}$. This is simply the superposition property of linear filters. In the next section, this decomposition is proven to induce a superposition property for median filters as well.

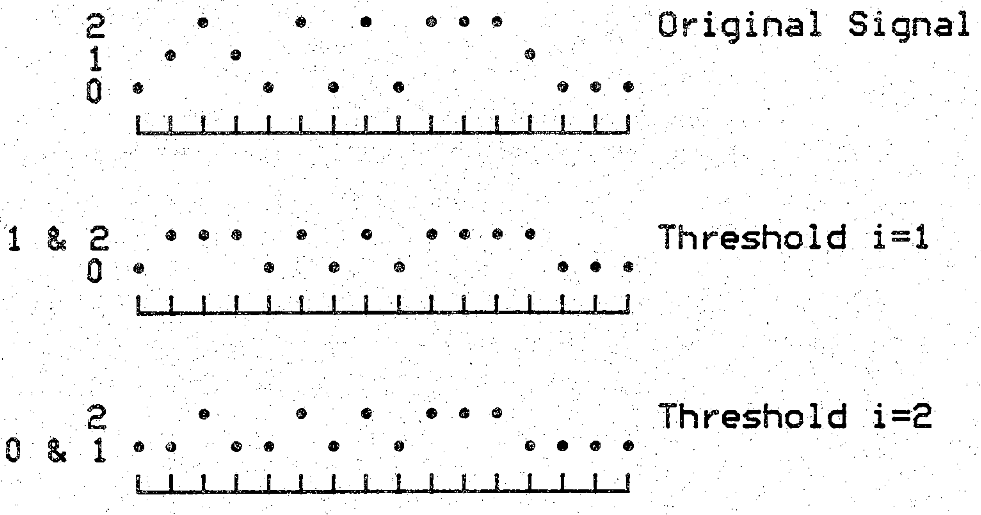

Figure 2.1
The threshold decomposition technique.

2.2 Median Filtering by Threshold Decomposition

Let $y_s(m)$ be the output at position m of a standard median filter with window width $2n+1$ applied to the input sequence $a(m)$. Applying the standard median filter to the thresholded values $\{t_0^i(m)\}$ gives another set of binary sequences

$$x_s^i(m) = \text{median}\left\{ t_0^i(m-n), \ldots, t_0^i(m), \ldots, t_0^i(m+n) \right\} \tag{2.2}$$

where start up and end effects are accounted for by repeating the first or last value of the signal as described in the Introduction. The relation of the binary valued, median filtered threshold sequence to the output $y_s(m)$ of the median filter applied to the original k-level signal is provided by

Lemma 2.2.1: There exists a mapping $f(\cdot)$ from the set of binary median filtered sequences $x_s^i(m)$, $1 \leq i \leq k-1$ to the signal space of k-leveled signals such that $y_s(m) = f(x_s^i(m), 1 \leq i \leq k-1)$.

Proof by construction: Define I(A) to be the indicator function of the event A, that is

$$I(A) = \begin{cases} 1 & \text{,if A is true} \\ 0 & \text{,if A is false} \end{cases}$$

Examine the output of one filter pass on the binary level i threshold decomposition sequence at any position m, $1 \leq m \leq L$

$$x_s^i(m) = \text{median}\left\{ t_0^i(m-n), \ldots, t_0^i(m), \ldots, t_0^i(m+n) \right\} \tag{2.3}$$

$$= \begin{cases} 1 & \text{,if } \sum_{j=-n}^{n} t_0^i(m+j) \geq n+1 \\ 0 & \text{,if } \sum_{j=-n}^{n} t_0^i(m+j) \leq n \end{cases}$$

$$= I\left\{ \sum_{j=-n}^{n} t_0^i(m+j) \geq n+1 \right\} \tag{2.4}$$

$$= I\left\{ \text{At least } n+1 \text{ elements in } t_0^i(m-n), \ldots, t_0^i(m+n) \text{ equal } 1 \right\}$$

$$= I\left\{ \text{At least } n+1 \text{ elements in } a(m-n), \ldots, a(m+n) \text{ are } \geq i \right\} \tag{2.5}$$

By Equation (2.5) we see immediately that if $x_s^p(m) = 1$, then for at least $n+1$ positions in the window the signal value $a(j)$, $m-n \leq j \leq m+n$, is greater than or equal to p. But this implies that at least $n+1$ $a(j)$'s in the window are greater than or equal to q for all $q \leq p$. That is, $x_s^q(m) = 1$, $1 \leq q \leq p$. This gives us the first property of the threshold decomposition.

Property 2.2: If $x_s^p(m) = 1$ then $x_s^q(m) = 1$ for $1 \leq q \leq p$.

We can now describe a mapping to the y's from the x's. For any m such that $1 \leq m \leq L$, we have

$$\begin{aligned} y_s(m) &= \text{median}\{a(m-n), \ldots, a(m), \ldots, a(m+n)\} \\ &= \max\{0, i: \text{At least } n+1 \text{ } a'\text{s in window are } \geq i\} \\ &= \max\{0, i: I(\text{At least } n+1 \text{ } a'\text{s in window are } \geq i)=1\} \end{aligned} \tag{2.6}$$

The combination of Equations (2.5) and (2.6) yields

$$y_s(m) = \max\left\{ 0, i: x_s^i(m) = 1 \right\}$$

and by Property 2.2 of the decomposition

$$y_s(m) = \sum_{i=1}^{k-1} x_s^i(m) \tag{2.7}$$

$$= f\left\{ x_s^i(m), 1 \leq i \leq k-1 \right\}$$

□

In loose terms, the function $f(\cdot)$ stacks the binary signals $x_s^i(m)$ on top of one another starting with $i=1$. The value of the output at position m is then the highest level at position m in the stack at which a one appears.

The function f(•) constructed above is shown to be the inverse of the threshold decomposition by

Lemma 2.2.2: The binary sequences $t_s^i(m)$, $1 \leq m \leq L$, and $1 \leq i \leq k-1$, obtained by thresholding $y_s(m)$, $1 \leq m \leq L$, are identical to the binary sequences $x_s^i(m)$, $1 \leq m \leq L$, and $1 \leq i \leq k-1$, from Lemma 2.2.1.

Proof:

$$t_s^i(m) = \begin{cases} 1 & ,\text{if } y_s(m) \geq i \\ 0 & ,\text{else} \end{cases}$$

Using Equation (2.7) from the proof of Lemma 2.2.1, we obtain

$$t_s^i(m) = \begin{cases} 1 & ,\text{if } \sum_{q=1}^{k-1} x_s^q(m) \geq i \\ 0 & ,\text{else} \end{cases}$$

Property 2.2 allows a reduction to the following

$$t_s^i(m) = \begin{cases} 1 & ,\text{if } x_s^i(m) = 1 \\ 0 & ,\text{else} \end{cases} \tag{2.8}$$

$$= x_s^i(m)$$

□

The above results show how one pass of a median filter over the input signal is equivalent to first thresholding the signal then filtering each threshold sequence and finally reconstructing the output using the function f(•). This operation can clearly be repeated for each pass of the median filter until a root signal is obtained. A simple inductive argument shows that because the decomposition and superposition are inverses, the intermediate reconstructions can be omitted.

Theorem 2.2. The root signal associated with a window width $2n+1$ median filter can be obtained by thresholding the original signal, filtering the resulting binary signals to roots, and then mapping these binary roots back to the k-level root signal using the function f(•) constructed in Lemma 2.2.1.

An example of this technique applied to a three level signal with a window width five median filter is given in Figure 2.2. The same results are obtained as when the filter was applied directly to the signal-see Figure 1.3.

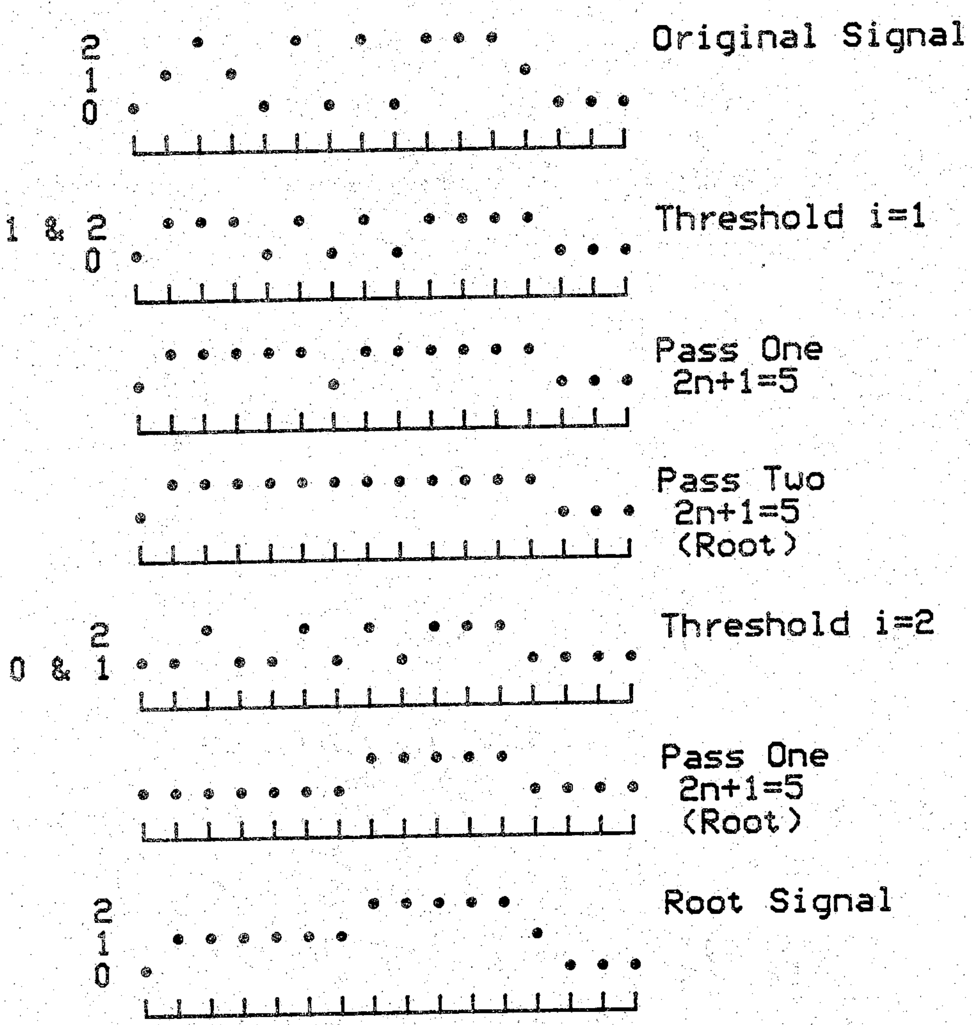

Figure 2.2
Median filtering by threshold decomposition.

This result provides a new tool for both the implementation and the analysis of the median filtering operation applied to arbitrary level signals. An important and surprising consequence of this Theorem is that the reconstruction function f(•) is a linear operator-the arithmetic sum. This linear superposition function seems out of place with the nonlinear median operator. Perhaps even more surprising is that this same technique is valid for a general class of ranked order operators. We begin the description of this more general class by showing that the decomposition is valid for recursive median filters.

2.3 Recursive Filtering by Threshold Decomposition

Recursive median filtering [14] is a modification of the standard process in which the center sample is replaced by the computed median before the window is shifted to the next point. Therefore, the n left most samples in a window are computed medians. If $\{a(m)\}$ is the input sequence to the filter, $1 \leq m \leq L$, then letting $y_r(m)$ be the output of the window width $2n+1$ recursive median filter, we have

$$y_r(m) = \text{median}\{y_r(m-n),\ldots,y_r(m-1),a(m),\ldots,y_r(m+n)\} \tag{2.9}$$

Figure 2.3 illustrates the results of applying a window width five recursive median filter to the same signal used for the standard filter example of Figure 1.3.

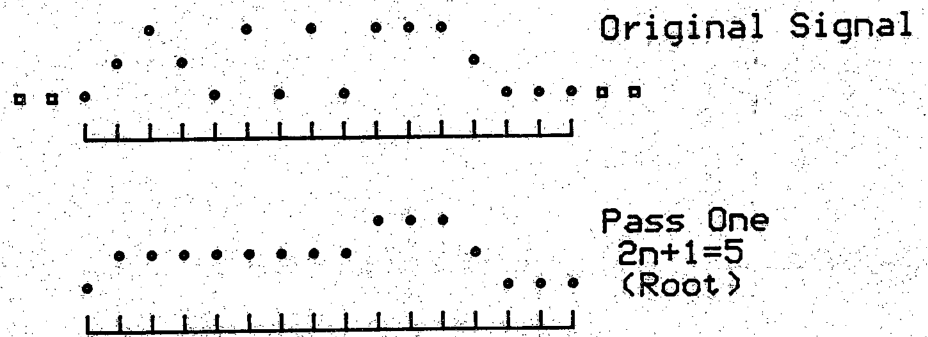

Figure 2.3
Convergence of a recursive median filter to a root.

Note that only one pass was required to reach a root with this recursive median filter. Also, for this signal the root of the recursive filter is different from the root of the standard filter.

Recursive median filters are known to have different properties than standard median filters [14]. For instance, the output after one pass of a recursive median filter is always invariant to additional passes of the same filter-- *i.e.*, it is a root. As was shown by the examples, this root may not equal the standard median filter root for the same signal. Furthermore, the output of a recursive median filter is affected not only by the window size but also by the direction the window slides across the signal. If a decomposition and superposition algorithm similar to Theorem 2.2 can be found for recursive median filters we could compare this modified filter with the standard filter by analyzing the simpler binary signal case. This is important when evaluating whether the recursive or the standard median filter is appropriate for a specific application. In addition, the validity of the threshold decomposition would reduce many problems in the analysis of recursive median filters to their corresponding binary problem. In Figure 2.4 the threshold decomposition technique is used to filter our test signal with a window width five recursive median filter. The output is identical to the results obtained when filtering the signal directly-see Figure 2.3. In the following Theorem a property analogous to the result of the preceding section is given for recursive median filters.

Theorem 2.3: The root signal associated with a window width $2n+1$ recursive median filter can be obtained by thresholding the original signal, applying the recursive median filter to the resulting binary signals, and then mapping these binary roots back to the k-level root signal using the function $f(\cdot)$ constructed in Lemma 2.2.1 of the preceding section.

The proof of this theorem is lengthy. It is developed as a series of properties for recursive median filters. To avoid complicated details it is found in Appendix A. The inductive parts of this proof can be eliminated by showing that the threshold decomposition is valid for a more general class of filters which contains the recursive filter as a special case.

Another reason for relegating the proof to the Appendices is that it is also a simple consequence of a much more general theorem. This theorem demonstrates that the decomposition and superposition properties of thresholding are valid for general multidimensional ranked order operators. Its proof is found in the next section. Because the threshold decomposition and the superposition are inverses, we can represent recursive filters as cascades of simpler filters. This is done for the recursive median case as an example in section 2.5.

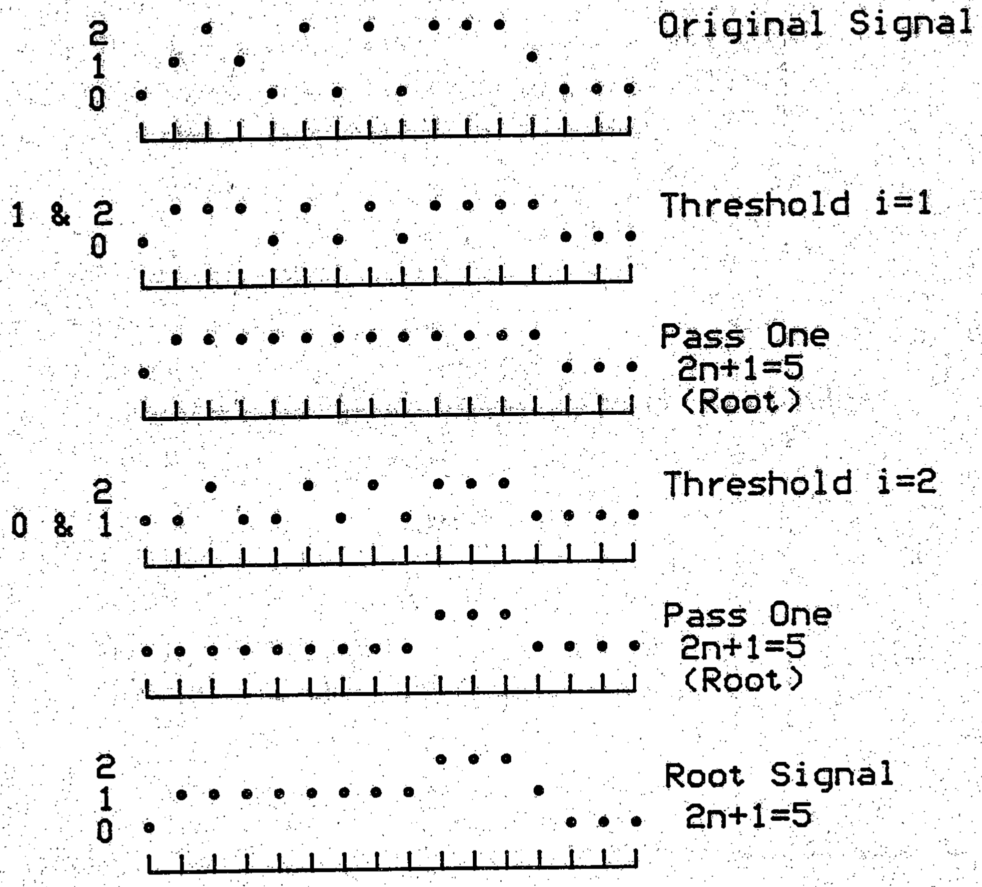

Figure 2.4
Recursive median filtering by threshold decomposition.

2.4 Multidimensional Ranked Order Operations

Modifications to the standard type ranked filter described above have been proposed in the literature [2,14-16]. In this paper we are particularly interested in filters where the output rank parameter r can change as the window slides across the signal. We say that these filters have an adaptive rank parameter; when the window is centered on the m'th position in the input signal, the output rank is r(m). In other words, the rank parameter r is a sequence r(m), $1 \leq m \leq L$.

So that our results will hold for multidimensional input signals we also allow the window shape to change with position. When the filter output position is m, $1 \leq m \leq L$, we let w(m) denote the set of positions which are to be considered in the window. This technique for representing higher dimensions with pointers of a single dimension is demonstrated by the examples in the next section.

For convenience let N(m) be the window size of the filter when it is at position m; that is, the number of positions included in the window w(m). Let the m'th output of this position dependent window shape and ranked order filter applied to an input signal a(m), $1 \leq m \leq L$, be denoted by $\phi_w^r(a,m)$. Then by definition

$$\phi_w^r(a,m) = \text{the r(m)th largest element of} \left\{ a(p) \colon p \in w(m) \right\} \qquad (2.10)$$

To provide a concrete example of the use of the parameters N(m), r(m) and w(m) in the definition of a filtering operation, consider the standard median filter. A standard median filter [13] has a constant window size of N(m) equal to $2n+1$, where n is any fixed integer. Its output rank parameter does not vary with position, so that $r(m) = n+1$. The definition of the window sequence w(m) would be straightforward -- $2n+1$ consecutive points centered about the current window position m -- if it were not for the start up and end effects associated with finite length signals. To account for the start up effects, the first point is repeated n times at the beginning of the signal so that a symmetric window is defined for each of the first n points in the signal [13]. A similar procedure is followed at the end of the signal, the last point is repeated n times. In order to account for these end effects we can define w(m) as

$$w(m) = \begin{cases} (1,\ldots,1,2,\ldots,m,\ldots,m+n) & ,\text{if } 1 \leq m \leq n \\ (m-n,\ldots,m,\ldots,m+n) & ,\text{if } n < m < L-n \\ (m-n,\ldots,m,\ldots,L-1,L,\ldots,L) & ,\text{if } L-n \leq m \leq L \end{cases} \qquad (2.11)$$

Any standard median filter can be described using the window shape sequence in Equation (2.11). We can generalize to standard ranked filters by changing the output rank parameter r(m) to any constant r, $1 \leq r \leq 2n+1$.

This notation defines a very general class of ranked order filters. For instance, multidimensional ranked filters with any shape window are in this class. This includes the square and cross window median filters used for image processing [1,7]. By using a combination of two filters applied in series, separable ranked filters can also be described. In the next section we show how these and other filters can be implemented and analyzed with the threshold decomposition. In Section 5, we show how recursive filters in any dimension can be described with this notation.

As before, assume that the input signal is a discrete sequence of length L which takes on the value a(m) at position m, $1 \leq m \leq L$, and that for each m, a(m) is quantized to one of the k integer values 0,1,...,k−1. Let $\phi_w^r(a,m)$ be the output at position m of a rank r(m) and window shape w(m) filter applied to the input sequence a(m). The window shape is allowed to change with position. The results of the following analysis are therefore valid in multidimensional applications. For convenience, let N(m) denote the number of positions in the window w(m). The filter at position m can be written as

$$\phi_w^r(a,m) = \text{the } r(m)\text{th largest element of} \left\{ a(p) \colon p \in w(m) \right\} \qquad (2.12)$$

Define the threshold decomposition of the original signal at position m to be the set of binary sequences

$$t_0^i(m) = \begin{cases} 1 & ,\text{if } a(m) \geq i \\ 0 & ,\text{if } a(m) < i \end{cases} \qquad (2.13)$$

with $1 \leq m \leq L$ and $1 \leq i \leq k-1$. Applying the ranked filter to these thresholded values gives another set of binary sequences

$$\phi_w^r(t_0^i,m) = \text{the } r(m)\text{th largest element of} \left\{ t_0^i(p) \colon p \in w(m) \right\} \qquad (2.14)$$

Appendix B is devoted to the development of three properties of filters with position dependent window shapes and ranks. These three properties lead directly to a superposition principal for generalized rank filters. Because the derivations are similar to the standard median filter proofs, they are left in the Appendices.

The relation of the binary valued, ranked order filtered threshold sequences to the output of the filter applied to original k-level signal is provided by

Property B.2: There exists a mapping f(•) from the set of filtered threshold decomposed sequences $\phi_w^r(t_0^i,m)$, $1 \leq i \leq k-1$ to the signal space of k-leveled signals such that

$$\phi_w^r(a,m) = f(\phi_w^r(t_0^i,m),\ 1 \leq i \leq k-1)$$

$$= \sum_{i=1}^{k} \phi_w^r(t_0^i,m)$$

The above results show how one pass of a generalized rank filter over the input signal is equivalent to first thresholding the signal then filtering each threshold sequence and finally reconstructing the output using the addition function f(•). This operation can clearly be repeated for a series of generalized ranked filters. A simple inductive argument based on the fact that f(•) and the decomposition are inverse operations between the binary and k level signals shows that the intermediate reconstructions can be omitted. These results are summarized in

Theorem 2.4: The output of a series of ranked order filters with position dependent window shapes applied to a k-level input signal is identical to the superposition, using the function f(•), of the filtered threshold decomposition of the input signal.

This result provides a new tool for both the implementation and the analysis of the ranked order filtering operations applied to arbitrary level signals. Two important applications of this result are the simple representation of recursive ranked operators and separable image processing algorithms as cascades of ϕ filters. A separable image processing algorithm is one in which processing is done row by row and then column by column. In the next section this technique is demonstrated for two popular filters: the recursive median and the separable median filters. Comparison of this technique with the complexity of the direct proof, found in Appendix A, shows that even though the proof for the ϕ filters was direct, the results can be used to show that the decomposition is valid for much more involved filtering schemes.

2.5 Recursive and Separable Median Filters

In this section we demonstrate how several different filters can be put into the form of ranked order operators with position dependent window shapes and ranks. Two popular filters were selected: recursive median filters and separable median filters for images. By Theorem 2.4, any filter which has a general ranked order filter representation can be implemented or analyzed using the threshold decomposition technique.

It is shown directly, in Appendix A, that the threshold decomposition technique is valid for recursive median filters. However, using the results of the previous section, we present an alternate proof by showing that recursive median filters are included in the class of ranked order operators with position dependent window shapes. To do this we need a bank of filters. Each filter in the bank is a median filter. That is, it calculates the median of the values that are specified to be in the window. The window, however, changes shapes with position and is also different for each filter. Let $w^t(m)$ be the set of positions in the window at position m of the signal for filter t in the bank of filters, $1 \le m \le L$ and $1 \le t \le L$. These filters are applied sequentially to the entire input signal with $t=1$ the first and $t=L$ the final filter in the bank. The following definition of these windows results in a bank of filters whose combined effect is equivalent to a recursive median filter of window width $2n+1$.

$$w^t(m) = \begin{cases} (1,\ldots,1,2,\ldots,m,\ldots,m+n) & ,\text{if } 1 \le m = t \le n \\ (m-n,\ldots,m,\ldots,m+n) & ,\text{if } n < m = t < L-n \\ (m-n,\ldots,m,\ldots,L-1,L,\ldots,L) & ,\text{if } L-n \le m = t \le L \\ (m) & ,\text{if } m \ne t \end{cases}$$

Essentially, the signal is passed unless m=t. For the case m=t, the median of the nearest $2n+1$ points is calculated. Because of the preceding filters in the bank, the n points with indices less than m are computed medians. This bank of filters shows that recursive median filters have the threshold decomposition property. By Property B.3 of the Appendices or the results of Section 2.2, we know that the intermediate reconstructions can be eliminated and the recursive filter can be implemented directly on the binary threshold sequences.

Another popular median-type filter is the separable filter used for image processing [7]. This filter is actually a combination of several filters. Suppose we have an image which has M rows and L columns, and is represented by the sequence $a(i,j)$, $1 \le i \le L$ and $1 \le j \le M$. With a separable filter, each row is median filtered as a separate signal then the output is filtered column by column in the same manner. We can represent this as a sequence of median filters with constant window width 2n+1 but position dependent window shapes $w^t(m)$, $1 \leq t \leq 2L$, where $w^t(m)$ equals

$$\begin{array}{ll}
[(t,1),\ldots,(t,1),(t,2),\ldots,(t,m),\ldots,(t,m+n)] & \text{if } t \leq L;\ 1 \leq m \leq n \\
[(t,m-n),\ldots,(t,m),\ldots,(t,m+n)] & \text{if } t \leq L;\ n < m \leq L-n \\
[(t,m-n),\ldots,(t,m),\ldots,(t,L-1),(t,L),\ldots,(t,L)] & \text{if } t \leq L;\ L-n \leq m \leq L \\
[(1,t-L),\ldots,(1,t-L),(2,t-L),\ldots,(m+n,t-L)] & \text{if } t > L;\ 1 \leq m \leq n \\
[(m-n,t-L),\ldots,(m,t-L),\ldots,(m+n,t-L)] & \text{if } t > L;\ n < m \leq L-n \\
[(m-n,t-L),\ldots,(L-1,t-L),(L,t-L),\ldots,(L,t-L)] & \text{if } t > L;\ L-n \leq m \leq L
\end{array}$$

Just as two dimensions were used in the above example, the position dependent window shape allows for the implementation of ranked order filters in arbitrary dimensions.

The threshold decomposition has now been shown to hold for arbitrary dimensioned ranked order operators with position dependent window shapes and adaptive ranks. One ranked type filter not included in this class is the alpha trimmed filter which was discussed in the Introduction. In the next section, the special superposition property is shown to hold for weighted rank filters. The alpha trimmed filter is used as an example for this class of filters.

2.6 Weighted Rank Filters

For ranked order filtering the output is a value in the window as determined by its associated rank among the other values. A generalization of this concept is to average a weighted combination of the ranked values in the window. Associate with each ranked value in the window w(m) a weight. We define the filter output y(m) by

$$y(m) = \sum_{j=1}^{N(m)} h_j \cdot \phi_w^j(a,m) \tag{2.15}$$

$$= \sum_{j=1}^{N(m)} h_j \cdot \text{the } j'\text{th largest element of } \{ a(p), p \in w(m) \}$$

The only restriction placed on the weights h_j is that they be finite. Obviously the ranked order or ϕ filter is a special case where h_j is 1 for j equal the rank r(m) and 0 otherwise. The following reconstruction principle is for weighted rank filters is shown directly

Theorem 2.6: One pass of a weighted rank filter with position dependent window shape is equivalent to the f(•) reconstruction of the filtered threshold decomposed sequences.

Proof: As in the previous section, let a(m), $1 \leq m \leq L$, be the input signal and let $t_0^i(m)$ be the threshold decomposition of the input signal. Begin by applying the reconstruction function to the filtered threshold decomposed sequences. For any fixed m,

$$f\left\{ \sum_{j=1}^{N(m)} h_j \cdot \phi_w^j(t_0^i,m) \right\} = \sum_{i=1}^{k-1} \left\{ \sum_{j=1}^{N(m)} h_j \cdot \phi_w^j(t_0^i,m) \right\} \tag{2.16}$$

$$= \sum_{j=1}^{N(m)} h_j \left\{ \sum_{i=1}^{k-1} \phi_w^j(t_0^i,m) \right\}$$

by Property B.2, see either Appendix B or Section 2.4, we have

$$f\left\{ \sum_{j=1}^{N(m)} h_j \cdot \phi_w^j(t_0^i,m) \right\} = \sum_{j=1}^{N(m)} h_j \; \phi_w^j(a,m) \tag{2.17}$$

$$= y(m) \tag{2.18}$$

□

Using the threshold decomposition for cascaded weighted filters may not be efficient. This is because the advantage of the decomposition is that it leaves binary signals for the majority of the operations. With weighted filters operating on binary filters we are not guaranteed a binary output. Because of the increasing number of possible levels there is no equivalence between the filtered threshold decomposed signals and the binary threshold decomposed filtered signals. That is, the function f(•) and the decomposition are not inverses whenever the filter creates a level which is not one of the original levels chosen for thresholding.

The result contained in Theorem 2.6 is still of interest, though, since it allows the threshold decomposition to be used on filters which are cascades of rank filters followed by a single weighted rank filter. This greatly simplifies the implementation of these filters, which are essentially nonlinear edge preserving, impulse suppressing filters followed by a type of linear smoother.

The final type of filter that we exhibit here is the alpha trimmed mean filter. Trimmed filters are an implementation of the intuitive notion that statistical outliers should not be used in computations. In each window position, the samples are ranked and a specified number of the smallest and largest values are eliminated from consideration before a statistic is calculated. Here the statistic being estimated is the mean. The relationship between alpha trimmed filters and median filters has been discussed in the literature [10]. By demonstrating that alpha trimmed mean filters are members of the class of weighted ranked order operations discussed in the previous section, the threshold decomposition technique can be used for their analysis and implementation.

Let $y_\alpha(m)$ be the output of an alpha trimmed mean filter at position m, $1 \leq m \leq L$. Then by definition,

$$y_\alpha(m) = \sum_{j=1}^{N(m)} h_j \cdot \phi_w^j(a,m)$$

where

$$h_j = \begin{cases} \dfrac{1}{(N(m)-2\alpha)} & ,\text{if } \alpha < j < N(m)-\alpha \\ 0 & ,\text{if } j \leq \alpha \text{ or } j \geq N(m)-\alpha \end{cases}$$

Because the number of small samples and large samples which were removed from the calculation are equal this is known as symmetric alpha trimmed. This technique is purported to give a more stable estimate of the mean than a direct average.

2.7 Summary of Results

The threshold decomposition and the set of binary signals perform the same function for ranked order filters that superposition and sinusoids perform for linear filters -- they allow complex problems to be decomposed into simpler problems. This has very fortunate practical and theoretical consequences.

On the practical side, the decomposition has an important impact on the implementation of ranked order filters. It shows that a ranked order filter for a multi-level signal is simply a parallel connection of filters for binary signals. Furthermore, since the output of the ranked filter for a binary signal is found by counting the number of ones in the window and comparing the result to a threshold, these filters are trivial to implement -- complicated ranking is no longer needed. The possibility of VLSI implementation is apparent and will be elaborated on in the next chapter.

On the theoretical side, the decomposition shows that the analysis of the ranked order filter's effects on multi-level signals is reduced to the much simpler analysis of binary signals. It is now clear that any of the properties which were limited to binary signals can now be extended in a straightforward fashion to multi-level signals. Also, the difficult task of comparing different ranked order operators is now reduced to the binary signal domain.

These theoretical results should improve our understanding of the behavior of these filters. The practical results should lead to the use of these filters in many new real time signal processing applications, particularly real time image processing.

CHAPTER 3
VLSI IMPLEMENTATION

The threshold decomposition technique presented in Chapter 2 allows any ranked order filter to be implemented as a parallel connection of binary ranked order filters. Although this technique is not an effective software algorithm for a multi-purpose computer, the parallelism of the algorithm and the simplicity of each of the parallel sections is amenable to VLSI implementation [17]. Because the actual filtering is done at the binary signal level this design is extremely fast and versatile. In addition, the actual layout can be modified in a straight forward manner to allow hardware implementation of a broad class of ranked filters.

The purpose of this chapter is to show how the threshold decomposition algorithm can be used to design a VLSI circuit for median filtering which is versatile, fast, and extremely parallel. The exact nature of the implementation, including the algorithms for decomposition and superposition, are not included. The chip is described as an information system not in terms of its gate or device level implementation. In Section 3.1 previous algorithms for fast median filtering are described and referenced. This is followed by the threshold decomposition algorithm. Then in Section 3.3 a novel technique is presented for the interconnection of median filter chips to increase the number of bits in the input sequence. These techniques are presented for median filters but can easily be extended to general ranked filters.

3.1 Previous Algorithms

Several fast algorithms exist for the implementation of median filters on multi-purpose digital computers [9,11,12]. Using our standard notation, an input sequence $\{a(m)\}$ is mapped to an output sequence $\{y(m)\}$ by a window width $2n+1$ median filter when

$$y(m) = \text{median}\{a(m-n),\ldots,a(m),\ldots,a(m+n)\}$$

The brute force implementation is to rank the values at each position of the window and select the median value as the filter output. Because the window sequentially slides across the input sequence, a significant reduction in computation is obtained by inserting the next input value into the already ranked set of values from the previous window and dropping the value which has shifted to a position outside the window. This means that, after initialization of the pointers for the first window position, only one value need be ordered at each of the window positions. Whenever the input signal takes on some set of values which is known before processing, say the k integers 0 to k-1, the algorithm can be made even faster by using bin sorts [12]. Because these techniques maintain a set of pointers to the ranked values they are known as histogram algorithms.

An algorithm purported to be faster than the histogram technique was presented by Ataman, *et. al.* [11]. Speed is obtained in this technique by doing as much of the ranking as possible on individual bits of the input words. This algorithm is, in fact, a special application of the threshold decomposition technique. The signal is partitioned only at the levels between bits instead of at all levels. The results are glued together by recombining these bit by bit operations into a complete word. A similar recombination technique can be found in Section 3.3 for interconnecting filter chips.

As can be seen in the preceding discussion, the threshold decomposition is useful in analyzing fast ranked filtering techniques. Because of the large number of binary sequences necessary to represent the input signal, the threshold decomposition algorithm is not recommended for software implementation on a multi-purpose computer. For example, eight bits would require 255 binary median filter sequences. The benefits of having this large number of binary filters operating simultaneously is realized in hardware, however, where this translates to a very parallel and fast design. Past hardware ranking implementations were based on compare and swap algorithms [18]. This new technique eliminates the ranking operation from the implementation of the filter. The resulting design is more versatile and allows a significant savings in time when modifications to the filtering operation are needed. This, of course, translates into an enormous reduction in the cost of fabricating other chips for ranked order operations.

3.2 Implementation of the Threshold Decomposition

Let the input signal be $\{a(m)\}$, taking on the values of the k integers from 0 to k-1. The output of a window width $2n+1$ median filter applied to $\{a(m)\}$ is denoted by $\{y(m)\}$. Finally, the threshold decomposition of the input signal into binary sequences is given by

$$t^i(m) = \begin{cases} 1, & \text{if } a(m) \geq i \\ 0, & \text{else} \end{cases}$$

The flow graph of this filtering system can be seen in Figure 3.1. The basic idea is to decompose an input sequence into a set of binary sequences, filter the binary sequences individually, and then combine the binary outputs for the many level filter output.

The decomposition and superposition operations are easier to implement than they appear. For instance, the superposition need not be a full adder. This reduction in complexity is accomplished by correctly arranging the bit slices so that all the slices necessary to calculate specific output bits are grouped together [19]. Discussion of these characteristics of the implementation is delayed until after chip construction.

The remaining part of the algorithm is the binary median filter. Let

$$x(m) = \text{median}\{t^i(m-n),\ldots,t^i(m+n)\} \tag{3.1}$$

be the binary output of a window width $2n+1$ median filter applied to the level i threshold sequence. Because these are binary values, the median can be rewritten using the indicator function $I(\cdot)$.

$$\begin{aligned} x(m) &= I\left[\text{median}\{t^i(m-n),\ldots,t^i(m+n)\} = 1\right] \\ &= I\left[\text{At least } n+1 \text{ of } t^{i\prime}\text{s equal } 1\right] \\ &= I\left[\sum_{p=-n}^{n} a(m+p) \geq n+1\right] \end{aligned}$$

This means the binary median filter can be implemented with an adder and a comparator-no ranking algorithm is necessary. A system diagram of this adding circuit can be seen in Figure 3.2. This design is easily modified for different window widths, that is, different n. A more appropriate VLSI circuit configuration for the binary median filter which uses a shift register and majority logic decoder can be seen in Figure 3.3.

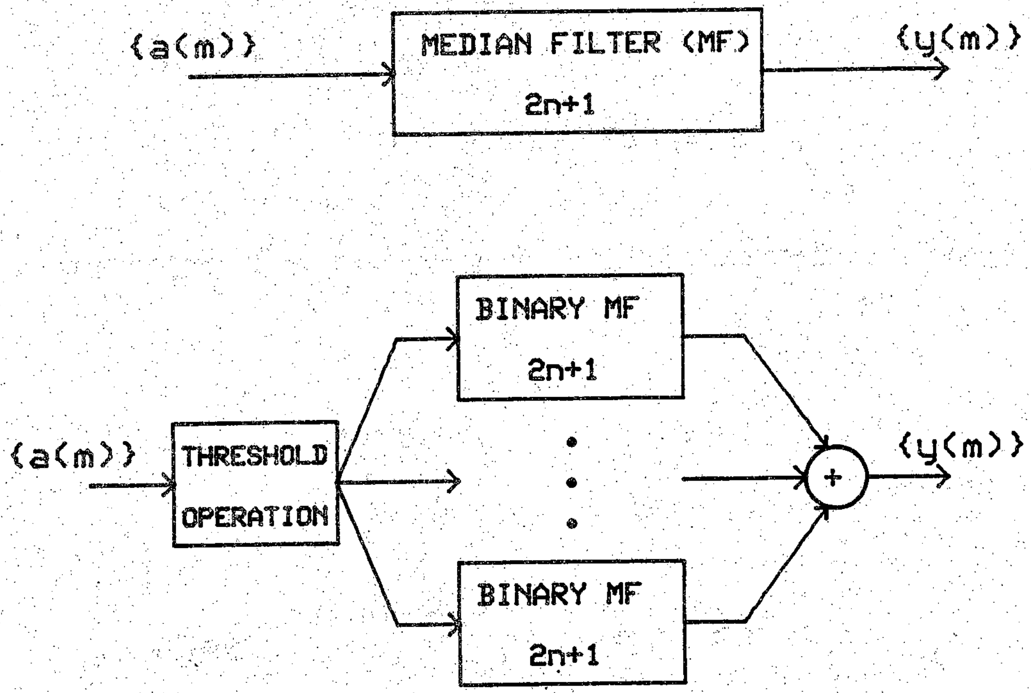

Figure 3.1
Threshold implementation of a median filter.

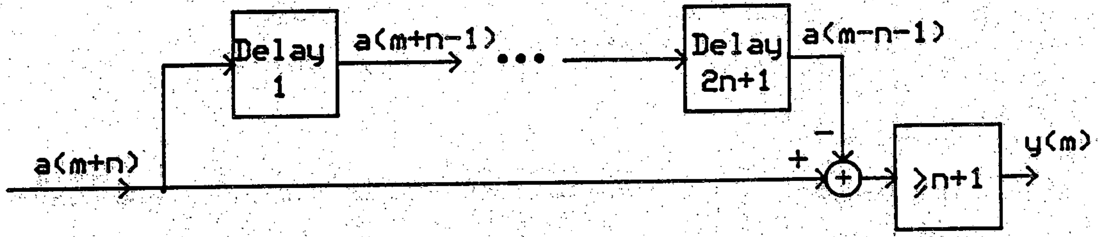

Figure 3.2
Binary median filtering scheme.

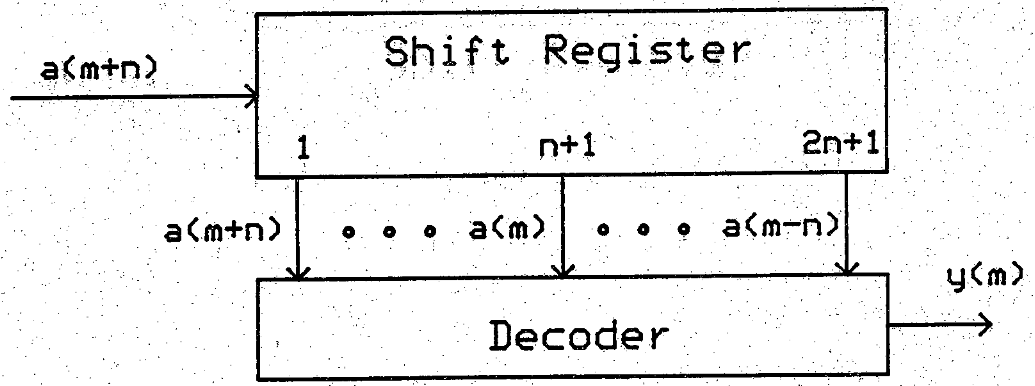

Figure 3.3
Binary median filter algorithm in hardware.

Armed with algorithms for the decomposition, the binary median filter, and the superposition operation, we are ready to explain why this is such a versatile design. A simple modification to a 2n+1 median filter will enable it to do 2n-1, or smaller, median filtering as well. Let z(m) be the output of a window width 2n-1 binary median filter, then

$$z(m) = I\left\{ \sum_{p=-(n-1)}^{(n-1)} t^i(m+p) \geq n \right\} \tag{3.2}$$

$$= I\left\{ \sum_{p=-(n-1)}^{(n-1)} t^i(m+p) + 0 + 1 \geq n+1 \right\}$$

which is a window width 2n+1 median filter with one value in the window fixed at 1 and another value fixed at 0. This can be realized in hardware using the circuit of Figure 3.4 and setting pins A and B to 0 and 1, respectively. This, of course, can be extended to do any window width median filter which is smaller than 2n+1.

With a slight modification, setting the values A and B asymmetrically to 0 or 1, certain ranked filters which are not medians can also be obtained using the same median decoder. In hardware, this can be accomplished by using pin inputs and having the window width control pins turn latches to these inputs on and off.

The same results can be obtained by externally modifying the input sequence to the median filter chip. For example, the sequence

$$t^i(1), 0, 1, t^i(2), \ldots, t^i(2n), 0, 1, t^i(2n+1), \ldots \tag{3.3}$$

as an input to a window width 2n+1 median filter would produce a coded version of the window width 2n-1 median filter. Using our previous notation, the output sequence would be

$$z(1), G, G, z(2), \ldots, z(2n), G, G, z(2n+1), \ldots$$

where G denotes the garbage bits to be thrown away. When multiple passes are performed, the garbage bits must be replaced after each pass with the 0,1 configuration of Equation (3.3). With multi-level signals, the 0's and 1's would become the minimum and maximum allowed values, respectively. The hardware implementation is obviously better because it requires no external modification to the input sequence. In fact the user cannot tell that the same decoder is being used for both filters.

Many applications of median filters require repeated filter passes. Frequently the signal is filtered until it is invariant to additional passes of the

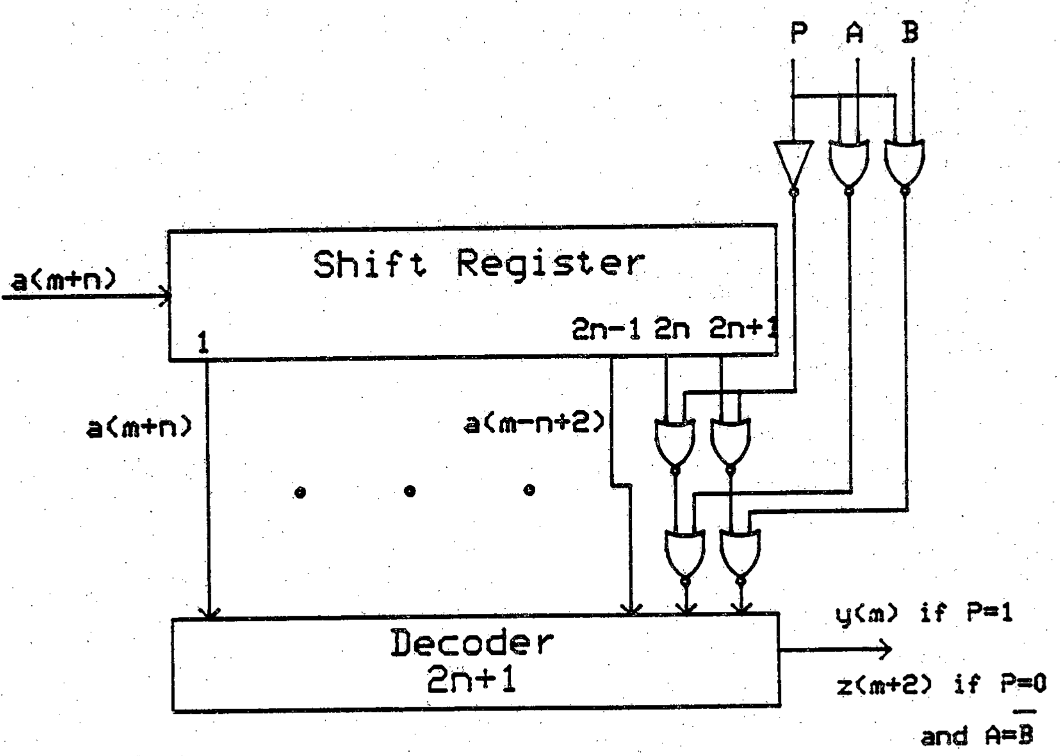

Figure 3.4
Multiple window width implementation.

filter- *i.e.* a root. From the previous chapter, we know that the threshold decomposition and its associated reconstruction function are inverses-that is, we can do repeated filter passes at the binary level. The system modifications necessary to implement this multi-pass strategy can be seen in Figure 3.5. In terms of the design of the chip, all that is required is repeated median filter stages at the binary level. It remains an extremely parallel and fast design.

In the past, recursive median filters were used because inputs are guaranteed to converge to roots after only one pass of the filter [14]. It is worth noting that the recursive filter, because the threshold property holds, can be implemented using a design technique similar to the one just described for the standard median filter. Figure 3.6 demonstrates the necessary modifications to the binary filtering levels when implementing the recursive filter. As before, a systems type diagram and a shift register with a decoder are presented to explain the function of the hardware.

The description of this new design technique for implementing median filters in VLSI is now complete. With a simple binary median decoder, many window width median filters and several different ranked filters can be pin programmed. The design, because of its parallelism, is easily modified. The final advantage is an increase in speed accomplished by doing repeated filter passes at the binary level before reconstruction. Future ideas for hardware implementation include combining the threshold decomposition with an analog to digital (A/D) converter. This provides an increase in speed because many A/D's use a type of threshold counter with a comparator to do the conversion to digital.

3.3 Interconnection of Median Filter Chips

A final problem in the design of a chip is the number of bits to be used for the input and output data words. Whatever is finally decided, it is certain that there will be some application which requires additional accuracy and consequently larger size data words. In this section, an algorithm for increasing the word size by connecting median filter chips of smaller word size together is presented. This is done for an increase of one bit, extensions to additional bits are straight forward. In addition, because this technique is another application of the threshold decomposition, it is presented with only a brief explanation.

Suppose a four bit median filter chip with the desired window width is available, but the application necessitates a five bit processor. The solution is accomplished using the threshold technique seen in Figure 3.7. The input

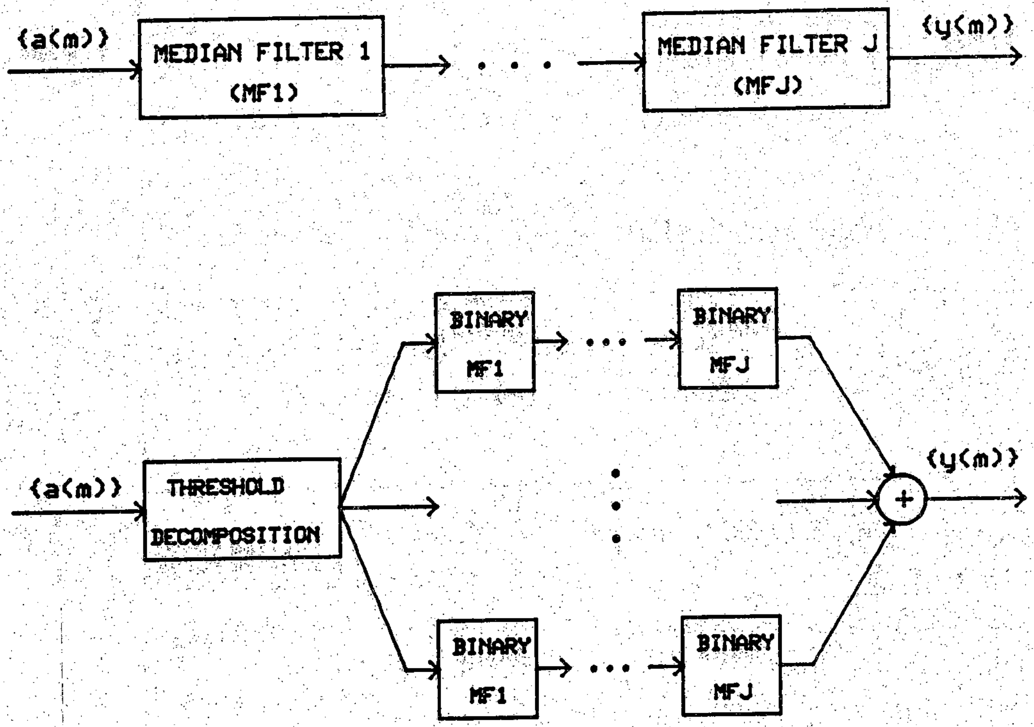

Figure 3.5
Threshold implementation of multiple median filters.

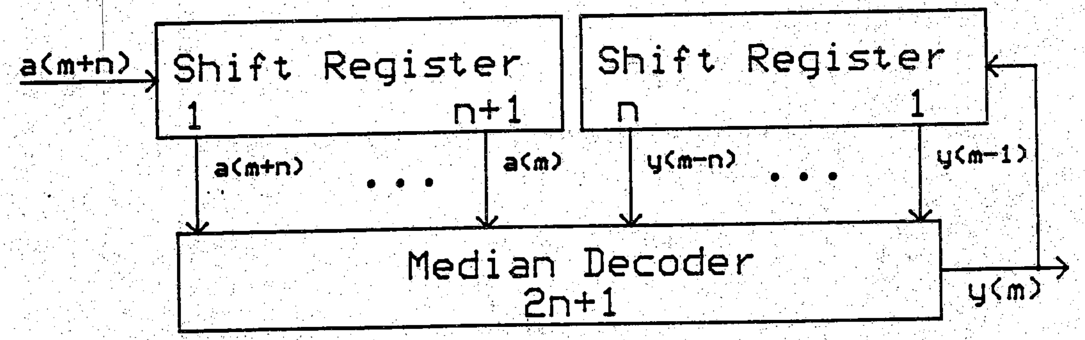

Figure 3.6
Binary recursive median filter in hardware.

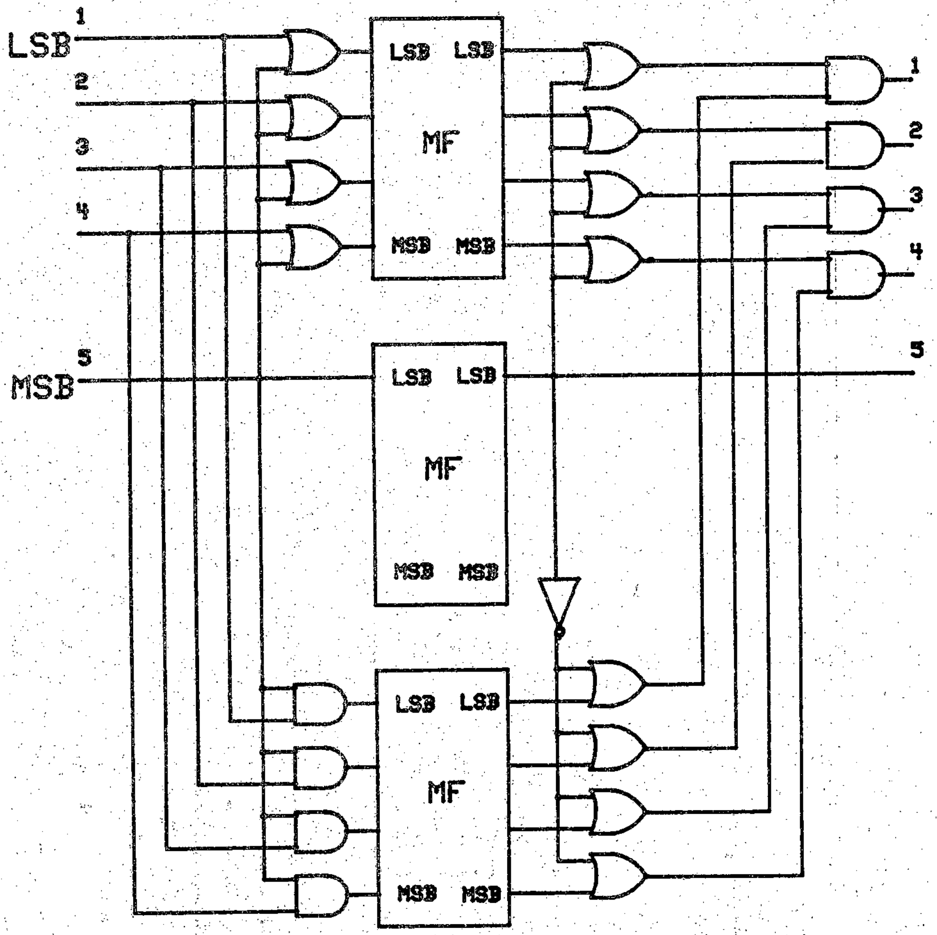

Figure 3.7
Interconnection of chips for increased word size.

signal is thresholded, using external logic, at the most significant bit (MSB). The first chip is used to filter the lower four bits when the signal is less than 2^5 and sees 2^5-1, all 1's, when the input is not less than 2^5. The second chip is used to process the four lower bits when the signal is greater or equal to 2^5 and sees all zeroes when the input is smaller than this level. Another chip is needed to filter the MSB of the input signal. This system maintains the proper order and rank of the input sequence as the values move through the median filter chips. The five bit output is determined by latching onto the appropriate four lower bits as selected by the output of the MSB filter. When the MSB output is a one the higher threshold chip carries the correct lower four bits and the other processor has all ones as its output. When the MSB output is a zero the lower threshold chip carries the correct lower four bits and the other processor has all zeroes for output.

3.4 Summary of Results

As in Chapter 2, the threshold decomposition is shown to be a powerful new tool. It can be applied in the analysis of ranked filters or, as shown in this chapter, it can be used to design a fast VLSI circuit for ranked filter implementation. Other techniques presented in this chapter included methods for modifying the window width and rank of the filter without extreme design modifications. The interconnection technique for increasing word size is also an important practical result.

CHAPTER 4
ROOT PROPERTIES OF MEDIAN FILTERS

4.1 State Description for the Root Signal Set

One of the most significant properties of median filters is that a signal invariant to further median filtering is obtained after only a finite number of passes of the filter. This is true for any input signal. The discovery of this property [13] has led to a much greater understanding of why the median filter has been a successful signal processing tool.

For instance, the edge preserving and impulse removing behavior of the window width 2n+1 median filter can be given precise definition.

1)A *constant neighborhood* is at least n+1 consecutive identically valued points.

2)An *edge* is a monotonic region between two constant neighborhoods of different value. The connecting monotonic region cannot contain any constant neighborhood.

3)A *root signal* is a sequence which is invariant to the median filter.

Theorem 4.1: Given a length L, k valued sequence to be median filtered with a 2n+1 window, a necessary and sufficient condition for the signal to be invariant under median filtering is that the appended signal consist only of constant neighborhoods and edges.

Theorem 4.2: Upon successive median filter window passes, any nonroot signal will become a root after a maximum of (L−2)/2 for L even and (L−1)/2 for L odd, successive filterings. This bound is exact for window width three filters. For larger window widths the bound is known to be of order L/n [20]. Also, any nonroot signal cannot repeat, and the first point to change value on any pass of the filter window will remain constant upon successive window passes.

The existence of a set of root signals leads to many natural questions. The first is the size of the set of roots for any window width filter - this is answered in this chapter. The second is the average number of passes needed to reach a root, which is addressed in the following chapter.

A state model describing the tree structure of the root signal set was proposed by Arce and Gallagher [9]. The use of trees provides an intuitively pleasing approach to defining the states of the model. However, when the signal takes on several quantization levels it is not feasible to draw trees for even relatively short signals. In the next section, we develop a general state description for the root signal set without using trees. Our model is unrestricted in terms of filter window widths and number of signal quantization levels. The result is a complete and exact system of equations for finding the number of root signals associated with any median filter.

We begin our state model with a signal of length one having k possible levels. At each iteration a new term is appended to the sequence. If this longer signal is a root we remain in the root state system, otherwise we leave to the absorbing state of nonroots. That is, once a signal contains a nonroot section, adding additional values to the end cannot make the signal a root. The value of a state at iteration L-1 is the number of root signals of length L which currently satisfy that particular state's conditions. Summing over all states at some iteration gives the number of roots associated with the filter for the specified window width, number of quantization levels, and signal length. In this manner a system of linear equations is developed to describe the number of root signals from a sequence with a fixed, but arbitrary, number of levels. We first derive results for filters with window width three and then make extensions to accommodate larger window sizes.

4.2 Window Width Three Filters

Suppose our signals take on k values, which we choose to be the integers from zero to k-1. When noninteger, discrete values are involved we map them into our model by using the rank minus one of each value. For window width three filters, the states of our system are defined in terms of the last two bits of the signal. If the values of the last two terms of the signal are identical we say that we are in a static state. When the signal value is i, we label the static state as $S_L(i+1)$, $0 \leq i \leq k-1$, where L is the current length of the signal. Similarly, when there is a change of level from j to i, we say that we are in a transitional state. For $j<i$ the transition is upward and we call the state $U_L(i)$, $1 \leq i \leq k-1$. And finally, $j>i$ is a downward transition and is labeled $D_L(k-1-i)$, $1 \leq k-1-i \leq k-1$. We now have all our states specified.

By finding a relationship between the states associated with signals of length L+1 and those of length L, we can recursively generate the state values for arbitrary lengths given the appropriate initial condition. From this we sum over all states associated with the desired length to get the number of possible root signals. Recall that this particular approach is restricted to window width three median filters. Larger windows will be handled in a similar manner, but require more memory in the states than the two bits of this model.

The advantage of these state definitions is that the recursive relationships can be written down by inspection. Each equation in this system specifies the number of root signals which are in the particular states and the number of ways a new bit can be added while remaining in the root space. We have the static states

$$S_{L+1}(i) = \begin{cases} S_L(1) + D_L(k-1) & ,\text{if } i=1 \\ S_L(i) + D_L(k-i) + U_L(i-1) & ,\text{if } 2 \leq i \leq k-1 \\ S_L(k) + U_L(k-1) & ,\text{if } i=k \end{cases} \tag{4.1}$$

and the transitional states

$$U_{L+1}(i) = \begin{cases} S_L(1) & ,\text{if } i=1 \\ \sum_{r=1}^{i} S_L(r) + \sum_{r=1}^{i-1} U_L(r) & ,\text{if } 2 \leq i \leq k-1 \end{cases} \tag{4.2}$$

with

$$D_{L+1}(i) = \begin{cases} S_L(k) & ,\text{if } i=1 \\ \sum_{r=k-i+1}^{k} S_L(r) + \sum_{r=1}^{i-1} D_L(r) & ,\text{if } 2 \leq i \leq k-1 \end{cases} \tag{4.3}$$

The initial conditions of the system are specified by $S_1(i) = 1$, for $i=1,...,k$ and $U_1(i) = D_1(i) = 0$, for $i=1,...,k\text{-}1$. Defining the number of roots to a k-level signal of length L to be $R^k(L)$, we have

$$R^k(L) = \sum_{i=1}^{k} S_L(i) + \sum_{i=1}^{k-1} (U_L(i) + D_L(i)) \tag{4.4}$$

As an example, consider the case of binary signals. Each state is associated with the last two root signal bits by

$$\begin{array}{ll} S(1) \infty 00 & S(2) \infty 11 \\ U(1) \infty 01 & D(1) \infty 10 \end{array} \tag{4.5}$$

Because the number of levels is specified and small we can solve explicitly for $R^k(L+1)$. By using the identities listed above, we arrive at

$$\begin{aligned} R^2(L+1) &= S_{L+1}(1) + S_{L+1}(2) + U_{L+1}(1) + D_{L+1}(1) \\ &= 2 \cdot S_L(1) + 2 \cdot S_L(2) + U_L(1) + D_L(1) \\ &= R^2(L) + S_L(1) + S_L(2) \qquad (4.6) \\ &= R^2(L) + S_{L-1}(1) + S_{L-1}(2) + U_{L-1}(1) + D_{L-1}(1) \\ &= R^2(L) + R^2(L-1) \end{aligned}$$

This resulting recursive equation checks with the derivation for arbitrary window width median filters applied to binary signals found in Arce and Gallagher [9].

As we increase the number of levels in the signal, the system of equations becomes more complex. Therefore, solving the system analytically, although straight forward as in the binary case above, would be tedious. Due to the method of labeling the states, this model lends itself to computer solution. It is worth noting that a reduction in the number of states can be achieved by exploiting the symmetry of the system. By the inherent tree structure of the signals, monotonic increasing edges have a symmetric monotonic decreasing edge. The number of roots in transitional states $U_L(i)$ and $D_L(i)$ are therefore equal for any i. The static states can be reduced in a similar manner with $S_L(i) = S_L(k+1-i)$ for $i=1,...,m$. Where m is k/2 for k even and (k-1)/2 for k odd. In the special case of k odd we have one asymmetric state corresponding to $S_L((k+1)/2)$. Perhaps these symmetries are better displayed by the following associations,

$$\begin{array}{lcl} U(1) \infty 01 & \longleftrightarrow & D(1) \infty (k-1)(k-2) \\ U(2) \infty 02,\ 12 & \longleftrightarrow & D(2) \infty (k-1)(k-3),\ (k-2)(k-3) \\ U(3) \infty 03,\ 13,\ 23 & \longleftrightarrow & D(3) \infty (k-1)(k-4),...,(k-3)(k-4) \quad (4.7) \\ \vdots & & \vdots \\ U(k-1) \infty 0(k-1),...,(k-2)(k-1) & \longleftrightarrow & D(k-1) \infty (k-1)0,...,10 \end{array}$$

for static states

$$\begin{array}{lcl} S(1) \infty 00 & \longleftrightarrow & S(k) \infty (k-1)(k-1) \\ S(2) \infty 11 & \longleftrightarrow & S(k-1) \infty (k-2)(k-2) \\ \vdots & & \vdots \qquad\qquad (4.8) \\ S(m) \infty (m-1)(m-1) & \longleftrightarrow & S(k-m+1) \infty (k-m)(k-m) \end{array}$$

and when k is odd we have the asymmetric state $S(m+1) = A(1) \infty (m)(m)$, where m would be (k-1)/2. The computer programs used in generating Table 4.1 utilize these symmetries.

The identities for the reduced system can be obtained by restricting states to the set of S(i), i=1,...,m and U(i), i=1,...,k-1 and A(1) for k odd. Occurrences of the other states in the controlling equations is removed by replacement with its symmetric state from the set of allowable states. The final system of equations for window width three filters applied to k-level sequences is for k even and m=k/2:

$$U_{L+1}(i) = \begin{cases} S_L(1) & \text{,if } i=1 \\ \sum_{r=1}^{i} S_L(r) + \sum_{r=1}^{i-1} U_L(r) & \text{,if } 2 \le i \le m \\ \sum_{r=1}^{m} S_L(r) + \sum_{r=k-i+1}^{m} S_L(r) + \sum_{r=1}^{i-1} U_L(r) & \text{,if } m+1 \le i \le k-1 \end{cases} \tag{4.9}$$

$$S_{L+1}(i) = \begin{cases} S_L(1) + U_L(k-1) & \text{,if } i=1 \\ S_L(i) + U_L(k-i) + U_L(i-1) & \text{,if } 2 \le i \le m \end{cases} \tag{4.10}$$

$$R(L) = 2 \cdot \sum_{i=1}^{k-1} U_L(i) + 2 \cdot \sum_{i=1}^{m} S_L(i) \tag{4.11}$$

and for k odd with m=(k-1)/2:

$$U_{L+1}(i) = \begin{cases} S_L(1) & \text{,if } i=1 \\ \sum_{r=1}^{i} S_L(r) + \sum_{r=1}^{i-1} U_L(r) & \text{,if } 2 \le i \le m \\ \sum_{r=1}^{m} S_L(r) + A_L(1) + \sum_{r=1}^{m} U_L(r) & \text{,if } i=m+1 \\ \sum_{r=1}^{m} S_L(r) + A_L(1) + \sum_{r=k-i+1}^{m} S_L(r) + \sum_{r=1}^{i-1} U_L(r) & \text{,if } m+2 \le i \le k-1 \end{cases} \tag{4.12}$$

$$S_{L+1}(i) = \begin{cases} S_L(1) + U_L(k-1) & \text{,if } i=1 \\ S_L(i) + U_L(k-i) + U_L(i-1) & \text{,if } 2 \le i \le m \end{cases} \tag{4.13}$$

$$A_{L+1}(1) = A_L(1) + 2 \cdot U_L(m) \tag{4.14}$$

$$R(L) = 2 \cdot \sum_{i=1}^{k-1} U_L(i) + 2 \cdot \sum_{i=1}^{m} S_L(i) + A_L(1) \tag{4.15}$$

It is also possible to express many of the above relationships in recursive form. Consider, for example, the case where k is odd,

$$U_{L+1}(k-1) = U_{L+1}(k-2) + U_L(k-2) + S_L(2) \tag{4.16}$$

The recursive nature of these equations and the previously described reductions by symmetry were used to generate the root counting results presented in Table 4.1.

Table 4.1
Number of window width three roots.

Length	Number of Levels						
n	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8
2	4	9	16	25	36	49	64
3	6	17	36	65	106	161	232
4	10	37	94	195	356	595	932
5	16	77	236	567	1168	2163	3704
6	26	163	602	1673	3886	7973	14932
7	42	343	1528	4917	12890	29325	60112
8	68	723	3882	14455	42744	107777	241718
9	110	1523	9858	42479	141688	395929	971416
10	178	3209	25038	124851	469726	1454643	3904290

We conclude that this system provides a solution to counting the number of roots of a window width three median filter for signals with an arbitrary number of levels. Due to their definitions, the states have a physical meaning in terms of the bits they represent. As we increase the window width, the state definitions will continue to rely on the value of the final bits of the root sequence.

4.3 Arbitrary Window Width Filters

Extending the results of the previous section to larger window widths involves increasing the memory of the state system. Suppose the median filter uses a window of $2n+1$ points. Then by definition, a constant neighborhood is at least $n+1$ consecutive identically valued points. For filters of window width three $n+1$ was two, our state model definitions were based on the last two values of the signal, and the state transition equations were a result of Theorem 4.1. The general window size state model requires information about the final $n+1$ values of the signal and Theorem 4.1 can again be invoked for the transition equations.

Begin with a signal which can take on the values of the k integers from zero to k-1. In order to stay in a root region the last $n+1$ bits of the signal must either be monotone or constant. The static states are associated with constant neighborhoods and the transitional states with edges. If the last $n+1$ bits are monotone or constant with values $a_1, a_2, \ldots, a_{n+1}$, then we define our states as follows

Static State: We say the L-length signal is in static state $S_L(i+1)$, $0 \leq i \leq k-1$, if and only if $a_1 = a_{n+1} = i$.

Transitional States: An L-length signal is in an upward transitional state $U_L(j,i)$, $1 \leq j \leq n$, $1 \leq i \leq k-1$ if and only if $a_{n+1-j} < a_{n+2-j} = a_{n+1} = i$ That is, a_{n+1-j} is less than i and the j bits following a_{n+1-j} have value i. Similarly for monotone decreasing $a_1, a_2, \ldots, a_{n+1}$, we say the signal is in downward transitional state $D_L(j,i)$, $1 \leq j \leq n$, $1 \leq i \leq k-1$ if and only if $a_{n+1-j} > a_{n+2-j} = a_{n+1} = k-1-i$.

In the transitional states $U_L(j,i)$ and $D_L(j,i)$, we call j the delay parameter. For window width three filters, $n=1$ and j takes only one value, reducing our state definitions to those found in the previous section.

As an example consider the three level signal with a filter of window width seven, *i.e.*, $k=3$ and $n=3$. Suppose the last four bits of some root sequence of length L-1 are 0111. The signal is therefore currently in state $U_{L-1}(3,1)$. Note that the delay parameter specifies the length of the constant region at the end of the length $n+1$ sequence. Examining the results of adding different values to make this a length L sequence we see that a 2 appended to the end puts the signal in state $U_L(1,2)$ while a 1 as the new bit moves the signal into static state $S_L(2)$. Note that when a 0 is added to the signal of length L-1 it creates a nonroot region, 01110, which moves the signal into an absorbing state outside the root space.

Using their definitions, the recursive relationships among the states can be written down by inspection. We have the static states

$$S_{L+1}(i) = \begin{cases} S_L(1) + D_L(n,k-1) & ,\text{if } i=1 \\ S_L(i) + U_L(n,i-1) + D_L(n,k-i) & ,\text{if } 2 \leq i \leq k-1 \\ S_L(k) + U_L(n,k-1) & ,\text{if } i=k \end{cases} \qquad (4.17)$$

and the transitional states

$$U_{L+1}(j,i) = \begin{cases} S_L(1) & ,\text{if } j=1, i=1 \\ \sum_{r=1}^{i} S_L(r) + \sum_{r=1}^{i-1} \sum_{q=1}^{n} U_L(q,r) & ,\text{if } j=1, 2 \leq i \leq k-1 \\ U_L(j-1,i) & ,\text{if } 2 \leq j \leq n, 1 \leq i \leq k-1 \end{cases} \tag{4.18}$$

with

$$D_{L+1}(j,i) = \begin{cases} S_L(k) & ,\text{if } j=1, i=1 \\ \sum_{r=1}^{i} S_L(k+1-r) + \sum_{r=1}^{i-1} \sum_{q=1}^{n} D_L(q,r) & ,\text{if } j=1, 2 \leq i \leq k-1 \\ D_L(j-1,i) & ,\text{if } 2 \leq j \leq n, 1 \leq i \leq k-1 \end{cases} \tag{4.19}$$

The initial conditions of the system are specified by $S_1(i) = 1$, for $i=1,...,k$ and $U_1(j,i) = D_1(j,i) = 0$, for all i and j. As before the number of roots to a k-level signal of length L is defined to be $R^k(L)$, and we have

$$R^k(L) = \sum_{i=1}^{k} S_L(i) + \sum_{i=1}^{k-1} \sum_{j=1}^{n} (U_L(j,i) + D_L(j,i)) \tag{4.20}$$

This specifies the system of state equations necessary to compute the number of roots for signals of arbitrary length and number of levels. Again, closed form analytic solutions can be obtained by specifying the values of k and n. However these computations are not only tedious but also must be repeated whenever the number of levels or the filter window width is changed. Computer solution is easily accomplished and the results are provided in Table 4.2. We again note that both symmetries between upward and downward transition states and tree structure symmetries among the static states can be exploited to reduce the amount of computation.

Table 4.2
Number of arbitrary window width roots.

Number of Levels: 2

Window		Number of Roots for L-length Signals								
Width	L=1	2	3	4	5	6	7	8	9	10
3	2	4	6	10	16	26	42	68	110	178
5	2	4	6	8	12	18	26	38	56	82
7	2	4	6	8	10	14	20	28	38	52
9	2	4	6	8	10	12	16	22	30	40
11	2	4	6	8	10	12	14	18	24	32
13	2	4	6	8	10	12	14	16	20	26
15	2	4	6	8	10	12	14	16	18	22
17	2	4	6	8	10	12	14	16	18	20

Number of Levels: 3

Window		Number of Roots for L-length Signals								
Width	L=1	2	3	4	5	6	7	8	9	10
3	3	9	17	37	77	163	343	723	1523	3209
5	3	9	17	27	49	91	163	293	531	959
7	3	9	17	27	39	63	107	181	297	485
9	3	9	17	27	39	53	79	125	201	319
11	3	9	17	27	39	53	69	97	145	223
13	3	9	17	27	39	53	69	87	117	167
15	3	9	17	27	39	53	69	87	107	139
17	3	9	17	27	39	53	69	87	107	129

Number of Levels: 4

Window		Number of Roots for L-length Signals								
Width	L=1	2	3	4	5	6	7	8	9	10
3	4	16	36	94	236	602	1528	3882	9858	****
5	4	16	36	66	136	292	612	1280	2690	5644
7	4	16	36	66	108	192	364	702	1328	2484
9	4	16	36	66	108	164	264	454	812	1460
11	4	16	36	66	108	164	236	354	564	944
13	4	16	36	66	108	164	236	326	464	696
15	4	16	36	66	108	164	236	326	436	596
17	4	16	36	66	108	164	236	326	436	568

CHAPTER 5
NUMBER OF FILTER PASSES TO A ROOT

5.1 The Model

The state model presented in the previous chapter is an exact and complete solution to counting the number of roots for arbitrary level signals and median filters of any window size. That is, the last chapter tells how many roots; in this chapter we investigate the number of passes needed to reach a root. Theorem 4.2, states that the maximum number of filter passes for a fixed length signal is (L−2)/2 for L even and (L−1)/2 for L odd. By decomposing signals into root and nonroot regions, the number of filter passes necessary to reach a root is determined by the longest nonroot region. This result follows from the invariance of the root regions ,as noted by Arce and Gallagher [9], and by applying Theorem 4.2. Suppose the nonroot regions have lengths $k_1, k_2, \ldots, k_n$ where k_m is the maximum, $1 \leq m \leq n$. By Theorem 4.2, we know that each of these sections will require either $(k_i-1)/2$ or $(k_i-2)/2$ passes to become a root region. Therefore, the largest number of passes is required by the section with length k_m. We would like to model binary signals in terms of these different root and nonroot regions.

To keep the description of root and nonroot regions relatively simple to model, we restrict our attention to filters of window width three and binary sequences. Figure 5.1 demonstrates the technique for decomposing a signal into root and nonroot regions. Recall that root regions, as described by Theorem 1 in the introduction, are areas of the signal that are invariant to passes of the median filter. Note that in our example the longest nonroot region, containing 6 bits of the signal, determines the number of filter passes necessary to reach a root. Each nonroot section is a binary oscillation of some finite length. In our example we say that the longest binary oscillation was of length 8. Where we include one point for each of the constant neighborhoods because the signal is still "oscillating" until it is in the adjacent root regions. The number of passes to a root was (8-2)/2 or 3.

An appropriate model must generate binary signals such that the length of the longest oscillation is known. Our approach is to use the familiar "success

```
 000011    0000  11      0000  111 (Root Section)
+      0101    10  010101    10    (Non Root Section)
 ---------------------------------
 000011010100001011010101000010111 (Complete Signal)

 000011101000000111101010000001111 (Pass One)
 000011110000000111110100000001111 (Pass Two)
 000011110000000111111000000001111 (Pass Three, Root)
```

Figure 5.1
Binary signal converging to a root.

runs" model from the theory of stochastic processes. At each iteration of a success runs model a Bernouilli experiment with probability p of success is performed. If there have been i consecutive successes the stochastic process is said to have value i. We relate the model to our signals by noting that there is a one to one correspondence between the length of the longest binary oscillation in the signal and the maximal success run of the model. Specifically, by letting p be the probability that the next bit of the signal continues an oscillation we can use the distribution of maximal runs for our computations. Here a trial of the system corresponds to the addition of a bit to the signal. If a success occurs the new bit is the complement of the current bit and if a failure occurs the new bit has the same value as the current bit. Exact expressions are not easily derived for the distribution of maximal runs. There are, however, two cases where we can arrive at some meaningful expressions. The first case, $p=1/2$, is where all sequences are equally likely. We can treat this as a deterministic problem and count the number of signals requiring k passes to a root. In the second case, we approximate the probability that no runs of length k occur and use this to generate identical results as in case one as well as distributions and expected numbers of filter passes to a root for $0<p<1$. We should note that there is a one to one correspondence between binary oscillations in the signal space and "runs" in our Markov model.

5.2 Deterministic Case

This is a combinatorial analysis of the number of signals of length L requiring k median filter passes to a root. In terms of our probabilistic model we would say that each bit assumes either value with equal probability and is independent of the other bits. Equations are derived for the special case of

$k \geq L/2$. An identity, recursive on both L and k, is developed allowing a table of results to be computed. We begin with a few definitions.

1) L is the length of the signal which is arbitrary, but fixed.

2) $n_L(k)$ is the number of signals of length L with longest binary oscillation (or run) of length k.

3) $\pi_L(i)$ is the number of L-length signals requiring i passes to a root

Recall that for a binary oscillation of length k, the number of filter passes necessary to reach a root is (k-1)/2 for k odd and (k-2)/2 for k even. Therefore if k is odd, the $n_L(k+1)$ signals with longest binary oscillation of length $(k+1)$ require the same number of passes to a root as the $n_L(k)$ signals with longest oscillation of length k. This implies that

for L even:

$$\pi_L(k)=n_L(2k+2) + n_L(2k+1) \quad \text{,if } 0 \leq k \leq (L-2)/2 \tag{5.1}$$

and for L odd:

$$\pi_L(k) = \begin{cases} n_L(L) & \text{,if } k=(L-1)/2 \\ n_L(2k+2) + n_L(2k+1) & \text{,if } 0 \leq k \leq (L-3)/2 \end{cases} \tag{5.2}$$

Note that $\pi_L(0)$ is the number of root signals of length L for this window width three median filter.

In order to count the number of signals of equal length requiring k filter passes to a root, we want to solve for the $n_L(k)$ as defined previously. First note that all oscillations can be divided into two classes:

1) *Butted Oscillation-* the first and/or last bit of the signal is a point in the oscillation.

2) *Nested Oscillation-* the oscillation is terminated on both sides by root regions in the signal - *i.e.*, it is not butted.

If k is greater than L/2, then these two classes partition the group of all L-length signals with maximum binary oscillation of length k into two mutually exclusive sets. Let $B_L(k)$ be defined as the number of signals with maximum binary oscillation of length k and of the butted type. Similarly, let $N_L(k)$ denote the number of signals with maximum oscillation length k that are nested. For k less than L/2, there can be more than one oscillation of length k and consequently, the signal may not be uniquely defined as nested or butted. So we define $b_L(k)$ to be the number of signals with a maximum oscillation length of k where at least one of these oscillations of k bits includes the last bit of the signal. Then for $L/2 < k < L$, $B_L(k)$ is twice $b_L(k)$. Obviously the k-length oscillation in $b_L(k)$ can terminate on either value of a bit. Also, because

$L-(k+1) < k$ we know that first $L-(k+1)$ bits of the signal can take on all permutations and the total sequence will still be included in $b_L(k)$. Then, including the special case of $k=L$ we have

$$b_L(k) = \begin{cases} 2 & ,\text{if } k=L \\ 2^{L-k} & ,\text{if } L/2 \leq k < L \end{cases} \tag{5.3}$$

and

$$B_L(k) = \begin{cases} 2 & ,\text{if } k=L \\ 2^{L-k+1} & ,\text{if } L/2 < k < L \end{cases} \tag{5.4}$$

For nested oscillations greater than half the signal length, $k+2$ bits are fixed (oscillation must terminate within the signal), and this oscillation may assume any position among the remaining $L-(k+2)$ bits. Requiring the oscillation to begin with a zero will decrease the number of possible signals by a factor of two. The number of ways to place our specified $k+2$ bits among the remaining $L-(k+2)$ bits is $L-(k+2)+1$ choose 1. Because the oscillation length is greater than half the signal length, all permutations of the $L-(k+2)$ are acceptable in keeping the maximum oscillation length at k. So for $L/2 < k < L-1$,

$$N_L(k) = 2 \cdot \left\{ \begin{matrix} L-(k+2)+1 \\ 1 \end{matrix} \right\} \cdot 2^{L-(k+2)} \tag{5.5}$$

$$= (L-(k+2)+1) \cdot 2^{L-k-1}$$

$$= (L-k-1) \cdot 2^{L-k-1}$$

because there can be no nested oscillation of lengths L or L-1, $N_L(k)$ is zero for k equal to L or L-1. Conclusion, for $L/2 < k \leq L$,

$$n_L(k) = B_L(k) + N_L(k) \tag{5.6}$$

$$= \begin{cases} 2 & ,\text{if } k=L \\ 4 & ,\text{if } k=L-1 \\ 2^{L-k+1} + (L-k-1) \cdot 2^{L-k-1} & ,\text{if } L/2 < k < L-1 \end{cases}$$

The final special case to be considered before we construct our recursive formula occurs when k is exactly half the signal length. Because $b_L(L/2)$ includes both the signals containing a pair of length $L/2$ oscillations, taking twice $b_L(L/2)$ for $B_L(L/2)$ would count these signals twice. Therefore,

$$\begin{aligned} B_L\left(\frac{L}{2}\right) &= 2\cdot b_L\left(\frac{L}{2}\right)-2 \\ &= 2^{\frac{L}{2}+1}-2 \end{aligned} \tag{5.7}$$

and

$$\begin{aligned} N_L\left(\frac{L}{2}\right) &= \left(L-\frac{L}{2}-1\right)\cdot 2^{L-\frac{L}{2}-1} \\ &= \left(\frac{L}{2}-1\right)\cdot 2^{\frac{L}{2}-1} \end{aligned} \tag{5.8}$$

So for L even,

$$n_L\left(\frac{L}{2}\right) = 2^{\frac{L}{2}+1}+\left(\frac{L}{2}-1\right)\cdot 2^{\frac{L}{2}-1}-2 \tag{5.9}$$

This leaves the derivation of $n_L(k)$ for $k < L/2$. Begin by assuming the value of $n_{L-1}(k)$ is known. There are exactly two ways to append a bit to the right end of a signal of length L-1. Therefore, $n_L(k)$ is approximately equal to twice $n_{L-1}(k)$. If we can subtract off the cases where the appended bit creates an oscillation of length $k+1$ and add on the cases where a binary oscillation of length k is created by the appended bit, we will have $n_L(k)$ specified. Recalling that $b_{L-1}(k)$ is the number of length L-1 signals with maximum binary oscillation of length k butted to the right end of the signal, then we have by definition

$$n_L(k) = 2\cdot n_{L-1}(k) - b_{L-1}(k) + b_{L-1}(k-1) \tag{5.10}$$

We previously specified $b_L(k)$ for k greater than or equal to L/2, leaving the case of $k<L/2$. Our approach is to begin with the number of possible arrangements assuming the last $k+1$ bits are fixed and then subtract off the sequences which have oscillations of length greater than k. A factor of two is necessary because there are two ways to fix the $k+1$ bits. This is also why $b_{L-(k+1)}(k)$ has a coefficient of 1/2

$$b_L(k) = 2\cdot\left\{2^{L-(k+1)}-\sum_{j=k+1}^{L-(k+1)} n_{L-(k+1)}(j) - 1/2\cdot b_{L-(k+1)}(k)\right\} \tag{5.11}$$

$$= 2^{L-k} - 2 \cdot \sum_{j=k+1}^{L-k-1} n_{L-k-1}(k) - b_{L-k-1}(k)$$

Note that whenever k is greater than L in $n_L(k)$, we assume the value of the function is zero. Summarizing our equations:

$$b_L(k) = \begin{cases} 2 & ,\text{if } k=L \\ 2^{L-k} & ,\text{if } L/2 \leq k < L \\ 2^{L-k} - b_{L-k-1}(k) - 2 \cdot \sum_{j=k+1}^{L-k-1} n_{L-k-1}(j) & ,\text{if } 1 < k < L/2 \\ 2 & ,\text{if } k=1 \end{cases} \tag{5.12}$$

and

$$n_L(k) = \begin{cases} 2 & ,\text{if } k=L \\ 4 & ,\text{if } k=L-1 \\ 2^{L-k+1} + (L-k-1) \cdot 2^{L-k-1} & ,\text{if } L/2 < k < L-1 \\ 2^{\frac{L}{2}+1} + (\frac{L}{2} - 1) \cdot 2^{\frac{L}{2}-1} - 2 & ,\text{if } k=L/2 \\ 2 \cdot n_{L-1}(k) - b_{L-1}(k) + b_{L-1}(k-1) & ,\text{if } 1 < k < L/2 \\ 2 & ,\text{if } k=1 \end{cases} \tag{5.13}$$

Using the relationships between $\pi_L(k)$ and $n_L(j)$, as given in Equations (5.1) and (5.2), we can recursively solve for the number of filter passes to a root, as seen in Table 5.1. This technique, though exact for p=1/2, is not as versatile as the approximation described in the next section.

Table 5.1
Distribution of signals using combinatorial analysis.

n(L, k)

L	k=1	2	3	4	5	6	7	8
1	2	0	0	0	0	0	0	0
2	2	2	0	0	0	0	0	0
3	2	4	2	0	0	0	0	0
4	2	8	4	2	0	0	0	0
5	2	14	10	4	2	0	0	0
6	2	24	22	10	4	2	0	0
7	2	40	46	24	10	4	2	0
8	2	66	94	54	24	10	4	2

b(L, k)

L	k=1	2	3	4	5	6	7	8
1	2	0	0	0	0	0	0	0
2	2	2	0	0	0	0	0	0
3	2	2	2	0	0	0	0	0
4	2	4	2	2	0	0	0	0
5	2	6	4	2	2	0	0	0
6	2	10	8	4	2	2	0	0
7	2	16	14	8	4	2	2	0
8	2	26	26	16	8	4	2	2

			Number of Passes to a Root					
L	k=0	1	2	3	4	5	6	7
1	2	0	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0
3	6	2	0	0	0	0	0	0
4	10	6	0	0	0	0	0	0
5	16	14	2	0	0	0	0	0
6	26	32	6	0	0	0	0	0
7	42	70	14	2	0	0	0	0
8	68	148	34	6	0	0	0	0
9	110	306	80	14	2	0	0	0
10	178	624	182	34	6	0	0	0
11	288	1258	406	80	14	2	0	0
12	466	2514	892	184	34	6	0	0
13	754	4990	1936	416	80	14	2	0
14	1220	9852	4162	926	184	34	6	0
15	1974	19368	8876	2038	416	80	14	2
16	3194	37944	18802	4444	928	184	34	6

5.3 Success Runs Model

Returning to the success runs model we find that an approximation to the probability of no success runs of length r in n trials, which we will call $Q_n(r)$, is developed in Feller [21, pp. 322-328]. To use this approximation we need the definition of "run" used in the derivation. We say that a sequence of n letters S and F contains as many S-runs of length r as there are non-overlapping uninterrupted blocks containing exactly r letters S each. For example, SSSSFSSSSSSF contains three S runs of length 3 and also has five S runs of length 2. Now suppose we generate our binary signals using the following procedure.

1) Randomly select a starting bit

2) Let the transition probabilities be Markov with

$$p = \Pr(B_{n+1}=1 | B_n=0) = \Pr(B_{n+1}=0 | B_n=1) \tag{5.14}$$

$$q = \Pr(B_{n+1}=1 | B_n=1) = \Pr(B_{n+1}=0 | B_n=0)$$

$$= 1-p$$

where B_j is the value of the signal at position j, $1 \leq j \leq L-1$.

3) After every bit perform a Bernouilli experiment with probability p of success. Where success means an oscillation begins or continues and failure means a constant neighborhood begins or continues. Here a run of successes is an oscillation of the bits, which demonstrates the equivalence of "run" and "oscillation". For a length L signal we need L-1 Bernouilli trials.

Because we are now dealing with random sequences, we want the probability of an L-length signal requiring less than k passes of a median filter of window width three to reach a root. Call this probability $F_L(k)$. For $p=q=1/2$, all sequences are equally probable and the same results are expected as in the combinatorial analysis of the preceding section. In order to use Feller's result we require a relationship between $F_L(k)$ and $Q_{L-1}(j)$. Assume that L is odd then the maximum number of filter passes to a root is (L-1)/2. Obviously $F_L((L-1)/2)$ equals the probability that no run of length L-1 occurs in the L-1 Bernouilli trials; *i.e.,* $Q_{L-1}(L-1)$. Similarly,

$$F_L\left(\frac{L-1}{2}-j\right) = \Pr(\text{No success run of length} \geq L-1-2j) \tag{5.15}$$

$$= \begin{cases} 1 & ,\text{if } j<0 \\ Q_{L-1}(L-1-2j) & ,\text{if } j=0,1,2,\ldots,\left(\frac{L-1}{2}-1\right) \\ 0 & ,\text{if } j>\frac{L-1}{2}-1 \end{cases}$$

and for L even

$$F_L\left(\frac{L-2}{2}-j\right) = \begin{cases} 1 & ,\text{if } j<0 \\ Q_{L-1}(L-2-2j) & ,\text{if } j=0,1,2,\ldots,\left(\frac{L-2}{2}-1\right) \\ 0 & ,\text{if } j>\frac{L-2}{2}-1 \end{cases} \tag{5.16}$$

We now have the probability that less than k filter passes are sufficient to reach a root in terms of the $Q_{L-1}(j)$. The $F_L(k)$ may be used to calculate any desired statistics for the number of filter passes necessary to reach a root.

The approximation for the $Q_{L-1}(j)$ is easily done by computer. We use the following approach:

1) Let $x_0 = 1$

2)Iterate with $x_{m+1} = 1 + q \cdot p^j \cdot (x_m)^{j+1}$

3)This tends to some value x

4)$Q_L(j) \simeq \frac{1 - p \cdot x}{(j + 1 - j \cdot x) \cdot q} \cdot \frac{1}{x^{L+1}}$

5) The absolute error is less than $\frac{2 \cdot (j-1) \cdot p}{j \cdot q \cdot (1 + p)}$

The results, as seen in Tables 5.2,3, and 4, match those previously derived by an exact combinatorial approach. But we now have additional information for signals where each bit is correlated with the preceding bit. As expected, for p tending to zero the probability of constant neighborhoods tends to one and the expected number of filter passes to a root goes to zero. Similarly, for p near one we expect that on the average the signal will require the maximum number of passes to a root as specified by Theorem 4.2.

This model for the signal space is used to estimate the the number of filter passes to a root for a chosen confidence level. If i is the smallest integer such that the probability of reaching a root signal in i or fewer passes is less than or equal to α then, we say that i passes are necessary to be 100α percent

Table 5.2
Distribution of signals with p=0.5.

n	g(1)	g(2)	g(3)	g(4)	g(5)	g(6)	g(7)	g(8)	g(9)	g(10)
2	.147	.766								
3	.077	.620	.885							
4	.041	.502	.814	.942						
5	.022	.406	.748	.908	.971					
6	.011	.328	.688	.875	.954	.985				
7	.006	.266	.633	.843	.938	.977	.992			
8	.003	.215	.582	.813	.922	.969	.988	.996		
9	.002	.174	.535	.783	.906	.961	.984	.994	.998	
10	.001	.141	.492	.755	.891	.953	.981	.992	.997	.999

Signal		Probability of requiring i passes							
Length	E	0	1	2	3	4	5	6	7
3	.234	.766	.234						
4	.380	.620	.380						
5	.557	.502	.440	.058					
6	.686	.406	.502	.092					
7	.812	.328	.547	.110	.015				
8	.914	.266	.578	.134	.023				
9	1.007	.215	.598	.156	.027	.004			
10	1.088	.174	.609	.178	.033	.006			
11	1.160	.141	.614	.198	.039	.007	.001		
12	1.225	.114	.614	.218	.045	.008	.001		
13	1.283	.092	.609	.236	.051	.010	.002	.000	
14	1.336	.074	.601	.254	.057	.011	.002	.000	
15	1.385	.060	.591	.271	.062	.013	.002	.000	.000
16	1.431	.049	.579	.287	.068	.014	.003	.001	.000

Signal		Number of signals requiring i passes						
Length	E	0	1	2	3	4	5	6
3	.234	6	2					
4	.380	10	6					
5	.557	16	14	2				
6	.686	26	32	6				
7	.812	42	70	14	2			
8	.914	68	148	34	6			
9	1.007	110	306	80	14	2		
10	1.088	178	624	182	34	6		
11	1.160	288	1258	406	80	14	2	
12	1.225	466	2514	892	184	34	6	
13	1.283	754	4990	1936	416	80	14	2
14	1.336	1220	9852	4162	926	184	34	6

g(k)=Pr{No success run of length k in n trials}
E is the expected # of filter passes to a root

Table 5.3
Distribution of signals with p=0.3.

n	g(1)	g(2)	g(3)	g(4)	g(5)	g(6)	g(7)	g(8)	g(9)	g(10)
2	.490	.913								
3	.343	.846	.974							
4	.240	.784	.954	.992						
5	.168	.727	.935	.986	.998					
6	.118	.673	.916	.981	.996	.999				
7	.082	.624	.898	.975	.994	.999	1.00			
8	.058	.578	.880	.969	.992	.998	1.00	1.00		
9	.040	.536	.862	.964	.991	.998	.999	1.00	1.00	
10	.028	.496	.845	.958	.989	.997	.999	1.00	1.00	1.00

Signal		Probability of requiring i passes							
Length	E	0	1	2	3	4	5	6	7
3	.087	.913	.087						
4	.154	.846	.154						
5	.224	.784	.208	.008					
6	.287	.727	.260	.014					
7	.347	.673	.307	.019	.001				
8	.402	.624	.351	.024	.001				
9	.454	.578	.391	.029	.002	.000			
10	.503	.536	.428	.034	.002	.000			
11	.549	.496	.462	.039	.003	.000	.000		
12	.591	.460	.492	.044	.003	.000	.000		
13	.631	.426	.521	.049	.004	.000	.000	.000	
14	.668	.395	.546	.054	.004	.000	.000	.000	
15	.703	.366	.570	.059	.004	.000	.000	.000	.000
16	.736	.339	.591	.064	.005	.000	.000	.000	.000

g(k)=Pr{No success run of length k in n trials}
E is the expected # of filter passes to a root

Table 5.4
Distribution of signals with p=0.7.

n	g(1)	g(2)	g(3)	g(4)	g(5)	g(6)	g(7)	g(8)	g(9)	g(10)
2	.000	.019								
3	.000	.014	.698							
4	.000	.010	.556	.792						
5	.000	.007	.442	.697	.854					
6	.000	.005	.352	.613	.791	.897				
7	.000	.003	.280	.539	.732	.855	.927			
8	.000	.002	.223	.474	.678	.815	.899	.948		
9	.000	.002	.178	.417	.628	.776	.871	.929	.963	
10	.000	.001	.141	.367	.581	.740	.844	.910	.950	.974

Signal		Probability of requiring i passes							
Length	E	0	1	2	3	4	5	6	7
3	.981	.019	.981						
4	.986	.014	.986						
5	1.199	.010	.782	.208					
6	1.297	.007	.690	.303					
7	1.485	.005	.608	.284	.103				
8	1.603	.003	.536	.316	.145				
9	1.761	.002	.472	.340	.134	.052			
10	1.876	.002	.416	.359	.153	.071			
11	2.008	.001	.366	.372	.171	.064	.026		
12	2.115	.001	.322	.382	.187	073	.035		
13	2.228	.001	.284	.387	.202	.083	.031	.013	
14	2.325	.000	.250	.390	.216	.092	.036	.017	
15	2.422	.000	.220	.390	.228	.100	.040	.015	.007
16	2.510	.000	.193	.387	.240	.109	.045	.017	.009

g(k)=Pr{No success run of length k in n trials}
E is the expected # of filter passes to a root

confident of having a root. In Figures 5.2 and 3, plots for several values of α are given for signals generated using two different correlation values of p. There is a reduction from the maximum number of passes, (512-2)/2, to 32 passes with 95 percent confidence of a root when $p=0.9$ is the probability of nonroot regions continuing. For the case of equally probable signals, $p=0.5$, we see that only six filter passes are required for 95 percent root confidence when the signal length is less than or equal to 512. This is a reduction of 249 passes from the maximum, which would have a confidence of 100 percent. These examples indicate the possibility of reducing the number of filter passes from the maximum while maintaining a high confidence of reaching a root.

These results were derived using binary signals and window width three median filters. Because of the threshold decomposition technique presented in Chapter 2, these results hold for arbitrary level signals and window width three median filters immediately. In addition, using the fact that window width three filters are slower to converge than median filters of larger window widths [20], we know that the results derived here are the "worst case" for any window width filter applied to signals with multiple levels. This gives an upper bound on the number of passes to a root which guarantees, at least, some confidence level.

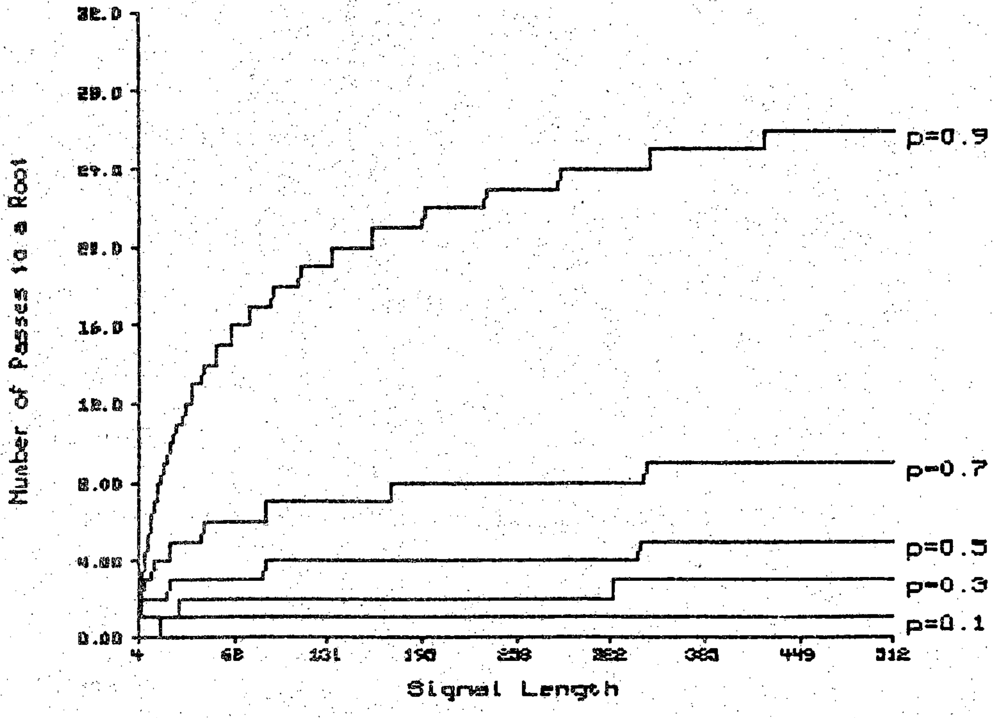

Figure 5.2
Root confidence of 85%.

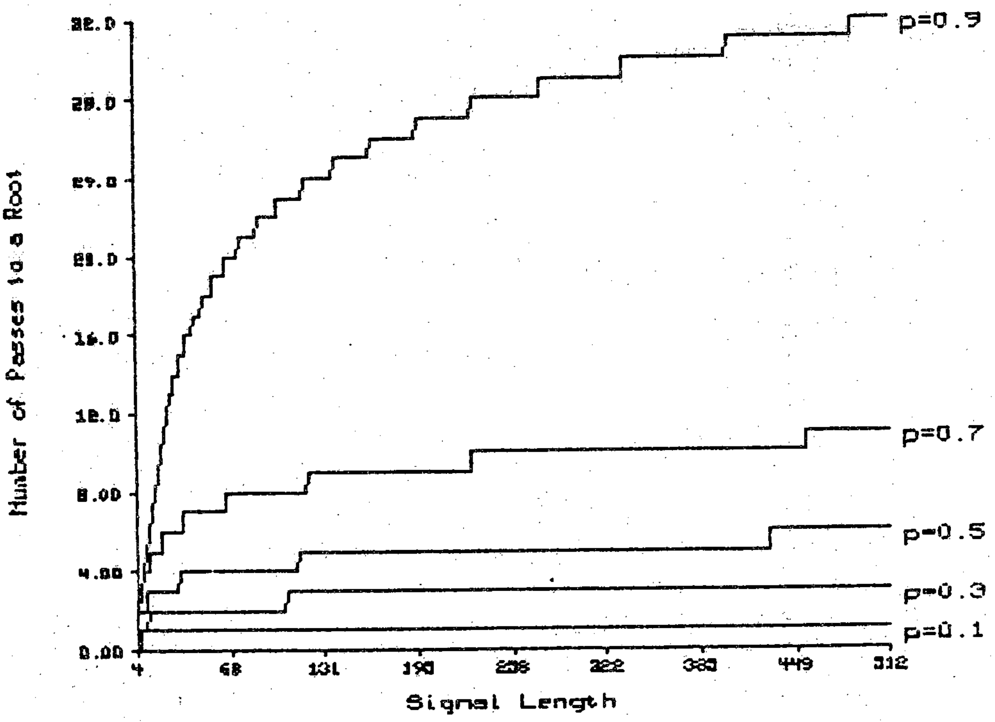

Figure 5.3
Root confidence of 95%.

CHAPTER 6
THE ANALOG MEDIAN FILTER

When random signals are used as inputs, the nonlinear nature of the median filter makes the output distributions difficult to calculate and to comprehend [7]. The threshold decomposition technique for reducing multi-level signals to binary signals does not reduce the statistical computation involved. We need a simpler means of calculating or approximating the output distributions of median filters. The basic idea behind the threshold decomposition--that of sliding a threshold through the various levels of a signal, lends itself to extensions in the domain of many leveled signals. In fact we might approach this problem by letting the number of allowable levels and the support of the input functions go to the continuum. We call the filter associated with the median operator, where these input functions are allowed, the analog median filter. In many cases this extension will make the analysis simpler than for the standard median filter. Because the definition is based on the threshold decomposition technique, however, the same underlying structure is maintained in both filters.

In this chapter, we define and present a preliminary analysis of the analog median filter. Specifically, in Section 6.2 we demonstrate that the discrete median filter is equivalent to an analog median filter operating on a restricted class of signals. In the next section, the analog median filter is shown to be the limiting case of the standard discrete median filter discussed in Section 6.1. A random variable interpretation of the analog filter is given in Section 6.4. Examples and Conclusions follow in the final sections of the chapter.

6.1 Analog Median Filter Definition

We assume throughout this section that the input to the discrete median filter is a sequence of length L which takes on the value a(m) at position m, $1 \leq m \leq L$. Let b(m) be the output at position m of a standard median filter with window width $2n+1$ applied to the input sequence a(m). Then by definition

$$b(m) = \text{median}\{\, a(m-n),\ldots,a(m),\ldots,a(m+n) \,\} \tag{6.1}$$

$$= \max\{\, i\text{: At least } n+1 \text{ of } a(m-n),\ldots,a(m+n) \text{ are } \geq i \,\}$$

$$= \max\{\, i\text{: More than half of } a(m-n),\ldots,a(m+n) \text{ are } \geq i \,\} \tag{6.2}$$

In a similar manner we might have shown

$$b(m) = \min\{\, i\text{: More than half of } a(m-n),\ldots,a(m+n) \text{ are } \leq i \,\} \tag{6.3}$$

Equations (6.2) and (6.3) are equivalent because there is an odd and finite number of values in the ranking window. Later we will see that these equations are equivalent only when the median is unique.

In the definition of the analog median filter, and in other definitions, discussions, and proofs that follow, it is convenient to use the indicator function of the event A which we define as

$$I(A) = \begin{cases} 1 & \text{,if A is true} \\ 0 & \text{,else} \end{cases} \tag{6.4}$$

Also, the notation sup(∘) and inf(∘) will be used to denote the supremum and infimum operations, respectively.

For any appropriate definition, the analog filter that we propose should be a limiting case to the discrete filter. Later in the chapter we will discuss the type of limit which applies. For now, we rely on our intuition for the following definition.

Definition: We say that y(•) is the output of a *analog median filter* of window width w applied to an input signal s(•) on the interval [c,d) if for almost every t∈[c,d)

$$y(t) = \sup\{\, r\text{: More than half of } s(\tau), \tau \in [t-\frac{w}{2}, t+\frac{w}{2}) \text{ are } \geq r \,\}$$

$$= \sup\left\{ r\text{: } \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[s(\tau) \geq r]\, d\tau \geq \frac{w}{2} \right\} \tag{6.5}$$

As in the discrete filter case, end effects are accounted for by repeating the endpoint of the signal over an interval large enough to fill the window [13]. We delay discussing the conditions on the function s(•) which guarantee the existence of the integral in the definition of Equation (6.5) until the stochastic section of the chapter. We should note that another possible definition of the analog median filter which would be analogous to Equation (6.3) of the discrete filter is

$$y_l(t) = \inf\left\{ r\colon \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[s(\tau)\leq r]\, d\tau \geq \frac{w}{2} \right\} \tag{6.6}$$

Unlike the discrete median filter case, see Equations (6.2) and (6.3), in the analog filter case $y_l(t)$ is not necessarily equal to $y(t)$ for all t. An appropriate compromise might be to take the average of $y_l(t)$ and $y(t)$ to obtain the output of the analog median filter. This additional computation seems cumbersome and usually unnecessary; we therefore resolve to define the output of the analog median filter by Equation (6.5). Hopefully this definition will allow us to go back and forth between discrete and analog signals so that results which are easier to obtain in one domain can be transferred to the other domain.

There are other equivalent definitions for the analog median filter. As an example of an alternative definition, assume that m(•) is a measure, such as the Lebesgue measure, defined for sets of real numbers. If the measure of an interval is its length and the measure of the sum of disjoint sets is the sum of the measures of each set we could use the following as the definition of a window width w analog median filter applied to the input signal s(•).

$$y(t) = \sup\left\{ r\colon m\left[\tau\colon s(\tau) \geq r \text{ and } t-\frac{w}{2} \leq \tau < t+\frac{w}{2} \right] \geq \frac{w}{2} \right\} \tag{6.7}$$

End effects are accounted for as in the previous definition. This measure theoretic definition can be easier to manipulate than the integral definition of Equation (6.5). The proofs in the stochastic section of the chapter demonstrate the utility of this definition.

6.2 Analog Filter Representation of a Discrete Filter

Now that we have defined what is meant by an analog median filter, several properties of these filters warrant discussion. The two following Properties result in a Theorem which shows that analog median filters when restricted to operate on a certain class of step functions are equivalent to discrete median filters.

Property 6.1: There exists a function f(•) mapping an interval on the real line to a set of values such that the sampled output of some analog median filter applied to f(•) equals the output of a window width 2n+1 discrete median filter applied to the sequence of samples from f(•).

Proof by construction: Assume we are trying to show the equivalence of the sampled analog median filter output with the sequence b(m). As before, b(m) is the output of a window width 2n+1 discrete median filter applied to the input sequence a(m), $1 \leq m \leq L$. Define the function f(•) on the half open interval $[\frac{1}{2}, L+\frac{1}{2})$ by the combination of step functions

$$f(t) = \sum_{m=1}^{L} a(m)\, I[\, m-\frac{1}{2} \leq t < m+\frac{1}{2}\,] \tag{6.8}$$

The window width of an analog median filter whose sampled output will equal the sequence b(m) is needed to complete the proof. Denote the output of an analog median filter with window width w equal to 2n+1 by y(t). Then

$$y(t) = \sup \left\{ r:\ \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[\, f(\tau) \geq r\,]\, d\tau \geq \frac{w}{2} \right\} \tag{6.9}$$

Assume the samples are taken at the integers 1,2,...,L. The sampled output of the filter is then given by y(m) with $1 \leq m \leq L$, that is

$$y(m) = \sup \left\{ r:\ \int_{m-\frac{w}{2}}^{m+\frac{w}{2}} I[\, f(\tau) \geq r\,]\, d\tau \geq \frac{w}{2} \right\} \tag{6.10}$$

$$= \sup \left\{ r:\ \int_{m-n-\frac{1}{2}}^{m+n+\frac{1}{2}} I[\, f(\tau) \geq r\,]\, d\tau \geq n+\frac{1}{2} \right\}$$

$$= \sup \left\{ r:\ \sum_{p=m-n}^{m+n} I[\, a(p) \geq r\,] \geq n+\frac{1}{2} \right\}$$

$$= \sup \{ \text{More than half of } a(m-n),\ldots,a(m+n) \text{ are } \geq r \}$$

$$= b(m)$$

Which shows that by sampling at the integers the two filters are equivalent.

□

In fact, because the constant regions are of minimum length one, this sampling pattern can be shifted left or right by any arbitrary ϵ, $0<\epsilon<1/2$ without changing the output.

Property 6.2: Using the notation of the previous Property, for any real ϵ and integer m such that $0<\epsilon<1/2$ and $1\leq m\leq L$, we have equality among $y(m\pm\epsilon)$, $y(m)$, and $b(m)$.

Proof: By Property 6.1 $y(m)$ equals $b(m)$ for m any integer $1\leq m\leq L$. For $y(m+\epsilon)$ we have

$$y(m+\epsilon) = \sup\left\{ r: \int_{m+\epsilon-n-\frac{1}{2}}^{m+\epsilon+n+\frac{1}{2}} I[\, f(\tau)\geq r \,]\, d\tau \geq n+\frac{1}{2} \right\}$$

$$y(m+\epsilon) = \sup\left\{ r: \sum_{p=m-n+1}^{m+n} I[a(p)\geq r] + \epsilon\cdot I[a(m+n+1)\geq r] \right.$$

$$\left. + (1-\epsilon)\cdot I[a(m-n)\geq r] \geq n+\frac{1}{2} \right\} \quad (6.11)$$

Let S denote the sum in Equation (6.11). If $S\geq n+1$ or $S\leq n-1$ then the values of the last two terms do not matter in the calculation of the supremum. However, if $S = n$ then the last two terms enter into the calculation and Equation (6.11) reduces to

$$y(m+\epsilon) = \sup\left\{ r: \epsilon\cdot I[a(m+n+1)\geq r] + (1-\epsilon)\cdot I[a(m-n)\geq r] \geq \frac{1}{2} \right\} \quad (6.12)$$

We note that because $0<\epsilon<1/2$ the second term in Equation (6.12) determines if the condition on the supremum is met. The first term can therefore be ignored and we reduce Equation (6.11) to

$$y(m+\epsilon) = \sup\left\{ r: \sum_{p=m-n+1}^{m+n} I[a(p)\geq r] + (1-\epsilon)\cdot I[a(m-n)\geq r] \geq n+\frac{1}{2} \right\} \qquad (6.13)$$

$$= \sup\left\{ r: \sum_{p=m-n+1}^{m+n} I[a(p)\geq r] + I[a(m-n)\geq r] \geq n+1 \right\}$$

$$= \sup\left\{ r: \sum_{p=m-n}^{m+n} I[a(p)\geq r] \geq n+1 \right\}$$

$$= y(m)$$

Shifting the sampling grid in the other direction we have the case for $y(m-\epsilon)$ which follows in a similar manner.

□

Summarizing the results of this section we have

Theorem 6.1: There exists a class of analog median filter input functions $\mathcal{F}$ such that any input sequence $\{a(m)\}$ to a discrete median filter has a representative function f in $\mathcal{F}$ and the output of the discrete filter applied to $\{a(m)\}$ has a representation in $\mathcal{F}$ which is the result of an analog median filter applied to f.

Proof: By Property 6.1 the output, sampled at the integers, of the analog median filter applied to the constructed function is equal to the representative discrete median filter output. By Property 6.2 we know that the output of the analog median filter for this class of constructed input functions is itself a member of the class. Therefore multiple passes are allowed and we have our representative class of input functions $\mathcal{F}$ by construction.

□

The preceding Theorem extends any result for analog median filters to discrete median filters. This is a new analysis tool which allows calculations which are more simply done in the continuum to be extended for the discrete case.

6.3 Analog Filter as Limit of Discrete Filter

In this section consider a window width w analog median filter applied to the signal s(·) on the interval [c,d). By representing the signal s(·) as a sequence of step functions, we can discuss sufficient conditions on the class of signals where the analog median filter can be considered as the limiting case of the discrete median filter.

Begin the approximation of the signal by sampling s(·) at c and then every ϵ units along the axis until the interval [c,d) is covered. Note that ϵ can be any positive real number. The following initial approximation to the signal s(·) can be made

$$s^1(t) = \sum_{m=0}^{L} s(c+m\cdot\epsilon)\cdot I[c+m\cdot\epsilon \leq t < c+(m+1)\cdot\epsilon]$$

where L is the smallest integer such that $c+(L+1)\cdot\epsilon$ is greater than d.

Decreasing the value of ϵ increases the accuracy of the approximation. By cutting the constant regions in half, the following approximation is obtained in k iterations

$$s^{k+1}(t) = \sum_{m=0}^{L\cdot 2^k} s(c+m\cdot\epsilon\cdot 2^{-k})\cdot I[c+m\cdot\epsilon\cdot 2^{-k} \leq t < c+(m+1)\cdot\epsilon\cdot 2^{-k}] \quad (6.14)$$

As k increases we can represent many different functions to within an arbitrary degree of accuracy. So the class of signals we are examining is restricted to functions which can be represented as a converging sequence of step functions. That is, for almost every t in [c,d) the representation

$$s(t) = \lim_{k\to\infty} s^k(t) \quad (6.15)$$

is valid. Most texts on real functions, see Royden [23] for example, show that the class of functions which can be approximated by step functions is not restrictive--at least for functions of engineering interest. Any measurable or continuous function which is unbounded for at most a countable number of points can be represented arbitrarily closely by step functions.

As in the previous section we want to relate the analog and the discrete filters. To do this we approximate the window width w by w^k where w^k equals $w\cdot(1+2^{-k})$. Note that the sequence w^k converges to w. If ϵ is set to w/2 then a window width w^k filter applied to the approximation $s^k(\cdot)$ is equivalent to a window width $1+2^k$ discrete median filter applied to the sequence of points $a^k(m)$, where

$$a^k(m) = s(c+m\cdot w\cdot 2^{-(k+1)}) \quad ,0 \leq m \leq L\cdot 2^k \quad (6.16)$$

This follows directly from the results of the previous section. The approximation $s^k(\cdot)$ is a combination of step functions with constant regions of length $w \cdot 2^{-k}$ and the window width approximation w^k is an integer multiple of this step width.

This gives us the description of analog median filters as the limiting case of discrete filters. Letting k go to infinity, we have an infinite window width for the discrete filter but the signal length has also become infinite. This demonstrates that, for the restricted class of signals which can be represented arbitrarily closely by step functions, many results from discrete median filters apply to analog median filters. The technique of exploiting this equivalence is demonstrated in the following two examples.

By observation we might have noted that signals which are composed of monotonic regions connected by constant regions of minimum length $w/2$ are invariant to passes of an analog median filter of window width w. We call these signals *roots* of the filter. This descriptive property of root signals can be shown directly in a manner analogous to previous discrete median filter proofs, see [13], or it can be considered as an immediate extension by using the technique presented in this section. That is, for any k we know that $a^k(m)$ must be composed of monotonic regions connected by constant regions of minimum length $2^{k-1}+1$. In terms of the corresponding analog filter this minimum length is $w/2$.

As another example, consider the problem of calculating the maximum number of filter passes necessary to reach a root for an analog median filter. This problem is extremely difficult to solve in the analog case; it is not obvious that the filter output even converges to a root in a finite number of passes. However it is known that a discrete median filter of window width $2n+1$ applied to a signal of length L is reduced to a root in no more than order L/n passes of the filter [20]. In our discrete filter approximation, the signal length is $L \cdot 2^k$ and the window width is $1+2^k$. For large k, the ratio tends to the constant L, which is finite. In fact L is approximately $2(d\text{-}c)/w$. Extending these notions to the continuum we conclude

Property 6.3: For the restricted class of signals which can be represented arbitrarily closely by step functions, the output of an analog median filter of window width w applied to a function with support no larger than $[c,d)$ converges to a root in a finite number of passes which is bounded by $2(d\text{-}c)/w$.

Other properties of the analog filter can be shown by extending known results for discrete filters in a similar manner. This technique is important in developing the intuition necessary to use the analog filter for the derivation of practical results.

6.4 Stochastic Interpretation of the Filter

The purpose of this section is to describe the various random and stochastic interpretations of the analog median filter. We begin by showing that an analog median filter applied to a deterministic function f(•) at point t is equivalent to calculating the median of the distribution of a random variable $Z(t)=f(X(t))$, where X(t) is uniformly distributed on $[t-\frac{w}{2},t+\frac{w}{2})$. To obtain the entire output of the filter we allow t to range over the support of f(•). The final random interpretation which we present is for the case of a stochastic process as the input to the analog median filter. Necessary conditions for the existence of filter output distributions are discussed.

Theorem 6.2: The output y(t) of an analog median filter applied to a measurable function f(•) equals the median of the distribution of the random variable $Z(t)=f(X(t))$, where X(t) is uniformly distributed on $[t-\frac{w}{2},t+\frac{w}{2})$.

Proof: First note that the median of a distribution is not always uniquely defined. We avoid discrepancies with our filter definition by always using the largest value α such that

$$P(\,Z(t) \leq \alpha\,) = P(\,Z(t) \geq \alpha\,) \geq \frac{1}{2} \tag{6.17}$$

as the median of the distribution which we denote as $\alpha_0(t)$. Note that P(•) is the associated probability measure. So we can write

$$\alpha_0(t) = \sup\{\,\alpha\colon P(\,Z(t) \geq \alpha\,) \geq \frac{1}{2}\,\} \tag{6.18}$$

Expanding $P(Z(t) \geq \alpha\,)$ we have

$$\begin{aligned} P(\,Z(t) \geq \alpha\,) &= P(\,f(\,X(t)\,) \geq \alpha\,) \\ &= P(\,I[\,f(\,X(t)\,) \geq \alpha\,] = 1\,) \\ &= E_{X(t)}(\,I[\,f(\,X(t)\,) \geq \alpha\,]) \end{aligned} \tag{6.19}$$

where $E_{X(t)}$ denotes the expected value with respect to the random variable X(t). Because X(t) is uniformly distributed over an interval of length w, the expectation reduces to

$$P(\,Z(t) \geq \alpha\,) = \frac{1}{w} \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[\,f(x) \geq \alpha\,]\;dx \tag{6.20}$$

Combining the results of Equations (6.18) and (6.20) we have

$$\alpha_0(t) = \sup\{\,\alpha:\ \frac{1}{w}\int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[f(x) \geq \alpha]\,dx \geq \frac{1}{2}\,\} \tag{6.21}$$

$$= \sup\{\,\alpha:\ \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[f(x) \geq \alpha]\,dx \geq \frac{w}{2}\,\}$$

$$= y(t)$$

where y(t) is the output at position t of an analog median filter applied to a deterministic function f(•).

□

Theorem 6.2 gives a more intuitive interpretation to the nature of the analog median transformation operating on deterministic functions. The usefulness of this interpretation depends on one's familiarity with random variables. Assuming a familiarity, the next step would be to allow stochastic inputs to the filter. In order to proceed in our analysis of random inputs we must first define what is meant by a stochastic process and then determine what class of processes are allowable as inputs to the filter. The following definitions and theorems on stochastic processes are from Doob [24].

We define a stochastic process as any family of random variables $\{x(t,\omega), t\in T\}$. For any fixed $t\in T$, $x(t,\omega)$ is a random variable and is therefore measurable with respect to ω. For a fixed ω, we call the resulting function of t a sample function of the process. Applying a window width w analog median filter to the stochastic input $x(t,\omega)$ results in $y(t,\omega)$ where

$$y(t,\omega) = \sup\left\{ r:\ \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[x(\tau,\omega) \geq r]\,d\tau \geq \frac{w}{2} \right\}$$

$$= \sup\left\{ r:\ m_\ell\{\,\tau:\ x(\tau,\omega) \geq r,\ \tau\in[t-\frac{w}{2}, t+\frac{w}{2})\,\} \geq \frac{w}{2} \right\} \tag{6.22}$$

Where the subscript ℓ on m_ℓ denotes Lebesgue measure and is used to distinguish it from the measure m(•) associated with the probability space. In the definition of a stochastic process no restrictions had been placed on the "t" properties of the function. Our first restriction is to set T equal to some finite interval. From Equation (6.22) it is obvious that in order for the analog median filter to make sense the process must be measurable with respect to the set of all half open intervals of length w with midpoint in the interval T. A process satisfying this condition is called measurable. Specifically, the stochastic process $\{x(t,\omega),\ t\in T\}$ is called *measurable* if the parameter set T is Lebesgue measurable and if $x(t,\omega)$ defines a function measurable in the pair of variables (t,ω). The importance of the measurability of the input stochastic process is demonstrated by the following Property.

Property 6.4: The output distribution of an analog median filter applied to stochastic process $x(t,\omega)$ exists if and only if $x(t,\omega)$ is a measurable process.

Proof: Let $y(t,\omega)$ denote the output of the window width w filter. Then

$$y(t,\omega) = \sup\left\{ r\colon m_\ell \left\{ \tau : x(\tau,\omega) \geq r,\ \tau\in [t-\frac{w}{2}, t+\frac{w}{2}) \right\} \geq \frac{w}{2} \right\} \tag{6.23}$$

For convenience let $A(r,t,\omega)$ be the set of all τ in the interval $[t-\frac{w}{2},t+\frac{w}{2})$ such that $x(t,\omega)$ is greater than or equal to r. That is,

$$A(r,t,\omega) = \{ \tau : x(\tau,\omega) \geq r,\ \tau\in [t-\frac{w}{2}, t+\frac{w}{2}) \} \tag{6.24}$$

We should note that $A(r,t,\omega)$ is a Lebesgue measurable set for any t or r if and only if $x(t,\omega)$ is a measurable process. Substituting this into Equation (6.23)

$$y(t,\omega) = \sup \{ r\colon m_\ell\, [A(r,t,\omega)] \geq \frac{w}{2} \} \tag{6.25}$$

The proof is easily done by investigating the function $G(\cdot)$ where $G(\gamma)$ equals the probability that $y(t,\omega)$ is greater than or equal to γ. If $G(\gamma)$ exists we have

$$\begin{aligned} G(\gamma) &= P\,(y(t,\omega) \geq \gamma) \\ &= P\,(\sup\,[r\colon m_\ell\, \{ A(r,t,\omega) \} \geq \frac{w}{2}] \geq \gamma\,) \end{aligned} \tag{6.26}$$

By noting that $A(r,t,\omega)$ is a subset of $A(\gamma,t,\omega)$ for any $r \geq \gamma$ and using the order of the inequalities, we can eliminate the supremum from Equation (6.26).

$$\begin{aligned} G(\gamma) &= P\,(\, m_\ell\, \{ A(\gamma,t,\omega) \} \geq \frac{w}{2}\,) \\ &= m\,(\, m_\ell\, \{ A(\gamma,t,\omega) \} \geq \frac{w}{2}\,) \end{aligned}$$

$$= m\,(\,m_\ell\,\{\,\tau\colon x(\tau,\omega) \geq \gamma,\, \tau \in [t-\frac{w}{2}, t+\frac{w}{2})\,\} \geq \frac{w}{2}\,) \qquad (6.27)$$

This set of equalities imply that $G(\gamma)$ exists if and only if $x(t,\omega)$ is measurable with respect to the Lebesgue measure $m_\ell(\cdot)$ and the probability measure $m(\cdot)$. This result for $G(\cdot)$ extends directly for the distribution function of $y(t,\omega)$.

□

This Property establishes the necessary and sufficient conditions on the input process which guarantee the existence of the output distribution. We would like to remain in the class of measurable processes even for repeated passes of the continuous filter. By the previous Property, this would guarantee output distributions for any countable number of filter passes. To do this we first show that

Property 6.5: The stochastic output of an analog median filter is measurable with respect to its parameter space if and only if the input process is measurable.

Proof: Let $y(t,\omega)$ denote the output of a window width w analog median filter applied to a stochastic process $x(t,\omega)$. To show that the output is a measurable function in t, it is sufficient to show that $\Gamma = \{t\colon y(t,\omega) \geq \gamma\}$ is a measurable set for any real γ. Expanding Γ we have

$$\Gamma = \{t\colon y(t,\omega) \geq \gamma\}$$

$$= \left\{ t\colon \sup\,[\,r\colon m_\ell\{\,\tau\colon x(\tau,\omega) \geq r,\, \tau \in [t-\frac{w}{2}, t+\frac{w}{2})\,\} \geq \frac{w}{2}\,] \geq \gamma \right\}$$

Because of the order of the inequalities we can eliminate the supremum in a manner similar to the proof of Property 6.4.

$$\Gamma = \left\{ t\colon m_\ell\{\,\tau\colon x(\tau,\omega) \geq \gamma,\, \tau \in [t-\frac{w}{2}, t+\frac{w}{2})\,\} \geq \frac{w}{2} \right\}$$

$$= \{t\colon B(t) \geq \frac{w}{2}\} \qquad (6.28)$$

where

$$B(t) = m_\ell\{\,\tau\colon x(\tau,\omega) \geq \gamma,\, \tau \in [t-\frac{w}{2}, t+\frac{w}{2})\,\}$$

Equation (6.28) has reduced the problem to showing that B(t) is a measurable function. It is in fact easier to show the stricter condition that B(t) is continuous in t. For any $\epsilon > 0$, we have

$$B(t+\epsilon) = m_\ell\{\tau: x(\tau,\omega) \geq \gamma, \tau \in [t+\epsilon-\frac{w}{2}, t+\epsilon+\frac{w}{2})\}$$

$$= m_\ell\{\tau: x(\tau,\omega) \geq \gamma, \tau \in [t+\frac{w}{2}, t+\epsilon+\frac{w}{2}) \cup [t-\frac{w}{2}, t+\frac{w}{2}) - [t-\frac{w}{2}, t+\epsilon-\frac{w}{2})\}$$

By the properties of the Lebesgue measure this equals

$$B(t+\epsilon) = m_\ell\{\tau: x(\tau,\omega) \geq \gamma, \tau \in [t+\frac{w}{2}, t+\epsilon+\frac{w}{2})\}$$

$$+ B(t) - m_\ell\{\tau: x(\tau,\omega) \geq \gamma, \tau \in [t-\frac{w}{2}, t+\epsilon-\frac{w}{2})\}$$

Because the Lebesgue measure of an interval is its length we have the following set of inequalities.

$$B(t+\epsilon) \geq B(t) - \epsilon$$
$$B(t+\epsilon) \leq B(t) + \epsilon$$

Which implies that $| B(t+\epsilon) - B(t) | \leq \epsilon$ for any $\epsilon > 0$. Therefore B(t) is continuous and necessarily a measurable function.

□

Combining Properties 6.4 and 6.5 we have the following

Theorem 6.3: The output of an analog median filter is a measurable stochastic process if and only if the input process is measurable.

Property 6.4 implies that in order to have the existence of output distributions of the analog filter we must have a measurable input process. By the above Theorem, if we start with a measurable input process, we will always obtain measurable processes as output. This allows multiple filter passes to be performed without destroying the measurability of the process.

6.5 Examples and Discussion

The properties of the Dirac delta function make it a popular test input for linear systems. In fact, linear systems are often characterized by their response to this impulsive input. One property of the analog median filter is that its impulse response does not completely characterize its performance. We do the analysis for the purpose of comparison with the responses of the more common class of linear filters. Let $\delta(t)$ denote the Dirac delta function which is defined by the two properties

$$\int_{-\infty}^{\infty} \delta(t)\,dt = 1 \text{ and } \delta(t) = \begin{cases} 0 & ,t\neq 0 \\ \infty & ,t=0 \end{cases} \tag{6.29}$$

In anticipation of the result, we might note that the discrete median filter performs well in the elimination of additive impulsive noise. The notion that the analog median filter should have no response to a delta function is stated and proven in the following

Property 6.6: The response of a window width w analog median filter to a Dirac delta input is identically zero.

Proof: Using the theory of generalised functions [25], we represent the delta function as the limit of a series of functions which integrate to one and converge to have both desired properties. For this proof the simplest of these functions is

$$\chi_n(t) = \begin{cases} n & ,\text{if } \frac{-1}{2n} < t < \frac{1}{2n} \\ 0 & ,\text{else} \end{cases} \tag{6.30}$$

Obviously the two defining properties of the Dirac delta hold because for any n

$$\int_{-\infty}^{\infty} \chi_n(t)\,dt = 1 \text{ and } \delta(t) = \lim_{n\to\infty} \chi_n(t) = \begin{cases} 0 & ,t\neq 0 \\ \infty & ,t=0 \end{cases}$$

Applying a continuous median filter with window width w to $\chi_n(t)$ we obtain $y_n(t)$ where

$$y_n(t) = \sup\left\{ r: \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[\chi_n(t)(\tau) \geq r]\,d\tau \geq \frac{w}{2} \right\}$$

If $w \leq 2/n$ then

$$y_n(t) = \begin{cases} n & ,\text{if } \frac{-1}{2n} < t < \frac{1}{2n} \\ 0 & ,\text{else} \end{cases}$$

and if $w > 2/n$ then $y_n(t)$ equals zero for all t. Obviously for any $w>0$ there exists an $N>0$ such that for all $n\geq N$ we have $w > 2/n$. This implies that $y_n(t)$ is zero for all $n\geq N$ and for this representation of the delta function we have an identically zero output of the analog median filter.

□

It is not surprising that the analog median filter does not respond to impulsive inputs. The main purpose of demonstrating the preceding property is to show that this linear technique is ineffective in the total characterization of the analog median filter.

As another example of how classic linear techniques fail to completely characterize the median filter, we examine the response of the filter to sinusoidal inputs. Begin by fixing w, the window width, and call the output of the analog filter y(t). Let s(·) be the sinusoidal input of the form $s(t) = \sin(2\pi t/\lambda)$, where λ denotes the wavelength of the test signal.

For linear systems, the superposition property allows the filtering of signals which are comprised of sums of simple signals to be done by filtering the individual simple signals and then recombining using addition to obtain the output. So for this example, we would calculate the effect of filtering a sinusoid of wavelength λ and then let λ range over the class of wavelengths necessary to represent the input signal using Fourier techniques. As has been noted before, linear techniques cannot be used to obtain the set of simple signals allowing an additive superposition operator. Now we use the analog median filter to demonstrate why frequency representations of the response of the median filter are difficult to interpret. We begin with three restrictions on λ and t which make the calculations less tedious.

Case 1: $t = \lambda k/2$, $k = 0, \pm 1, \pm 2, \ldots$ By the symmetry of s(·) we know that for this restricted set of time values we have $s(t+\tau) = -s(t-\tau)$. It is easy to show that the output of the analog median filter y(t) will be zero at these points. This set of points is precisely the set where s(t) is also zero. We conclude that for sinusoidal inputs, the zero crossings are invariant to analog median filtering.

Case 2: $w = k\lambda$, $k=1,2,\ldots$ Whenever the window width w of the filter is an integer multiple of the wavelength of the sinusoidal input, a complete cycle of the input will always be in the window of the filter for any time position. That is, because $s(t) = \sin(2\pi tk/w)$ and by the periodicity of the sine function, we have for all t

$$y(t) = \sup\left\{ r: \int_{t-\frac{w}{2}}^{t+\frac{w}{2}} I[\sin(2\pi k\tau/w) \geq r]\, d\tau \geq \frac{w}{2} \right\} = 0 \tag{6.31}$$

We conclude that the analog median filter has no response to sinusoids with $\lambda = w/k$. In terms of the frequency of the sinusoid we have $\nu = 1/\lambda = k/w$ as the class of frequencies with an identically zero response.

Case 3: $\lambda \geq w > 0$. This case is simple because we obtain a root after only one pass of the filter. In addition, the root is simply a "clipped" version of the input. This can be seen in Figure 6.1. Analytically we have

$$y(t) = \begin{cases} \sin\left[\pi\left(\frac{2k+1}{2} - \frac{w}{\lambda}\right)\right] & ,\left(\frac{2k+1}{4}\right)\lambda - \frac{w}{2} \leq t < \left(\frac{2k+1}{4}\right)\lambda + \frac{w}{2} \\ \sin[2\pi t/\lambda] & ,\text{else} \end{cases} \qquad (6.32)$$

where $k = 0, \pm 1, \pm 2, \ldots$ From this we see that the filter is clipping the tops of the sinusoid to constant regions of width w/2. Therefore y(t) is a root to the analog median filter of window width w. This is a well behaved response in that the nonconstant monotonic regions are identical to the input sinusoid.

If we consider the transformation of the filter as being from a single frequency signal to some output signal y(t). Then we immediately note that, because of the constant regions, the transformation spreads the output over the entire spectrum. This means that the sinusoidal response curve, because the normalized output energy is plotted in terms of the input wavelength, may be a misleading description of the filtering phenomenon. As an example, in Figures 6.2 and 6.3 we plot the response of the analog median filter to these sinusoidal signals in terms of the output energy normalized by the input energy. It is important to remember that the output energy is spread over the spectrum even though it is the result of filtering an input which contains only one frequency.

This analysis has allowed us to calculate analytically the sinusoidal response of the filter for $\lambda \geq w > 0$. For smaller wavelengths the structure of the input sinusoid is destroyed making the analysis significantly more complicated and therefore it will not be presented here. Instead refer to the sinusoidal response plots in the Figures. These plots were calculated using the approximation techniques presented earlier in this chapter. The zeros and other predicted behavior from the previous three cases can be verified. Note that, as expected, as λ becomes much greater than w the energy ratio tends to one.

6.6 Summary of Results

The foundation of the theory necessary to use the analog median filter to model discrete problems and to attack the difficult statistics involved with ranked operations is presented. This is a new concept for the analysis of median filters. Even though this is an introduction to the filter, its use as a tool for analytic proofs is demonstrated along with the foundation for statistical interpretations. The necessary and sufficient conditions for the existence of output distributions of multiple analog filter passes is an important result. The practical uses are demonstrated by the intuitive results which can be derived by treating the filter as an entity in itself and then mapping to the discrete domain for comparison with the already popular discrete median filter.

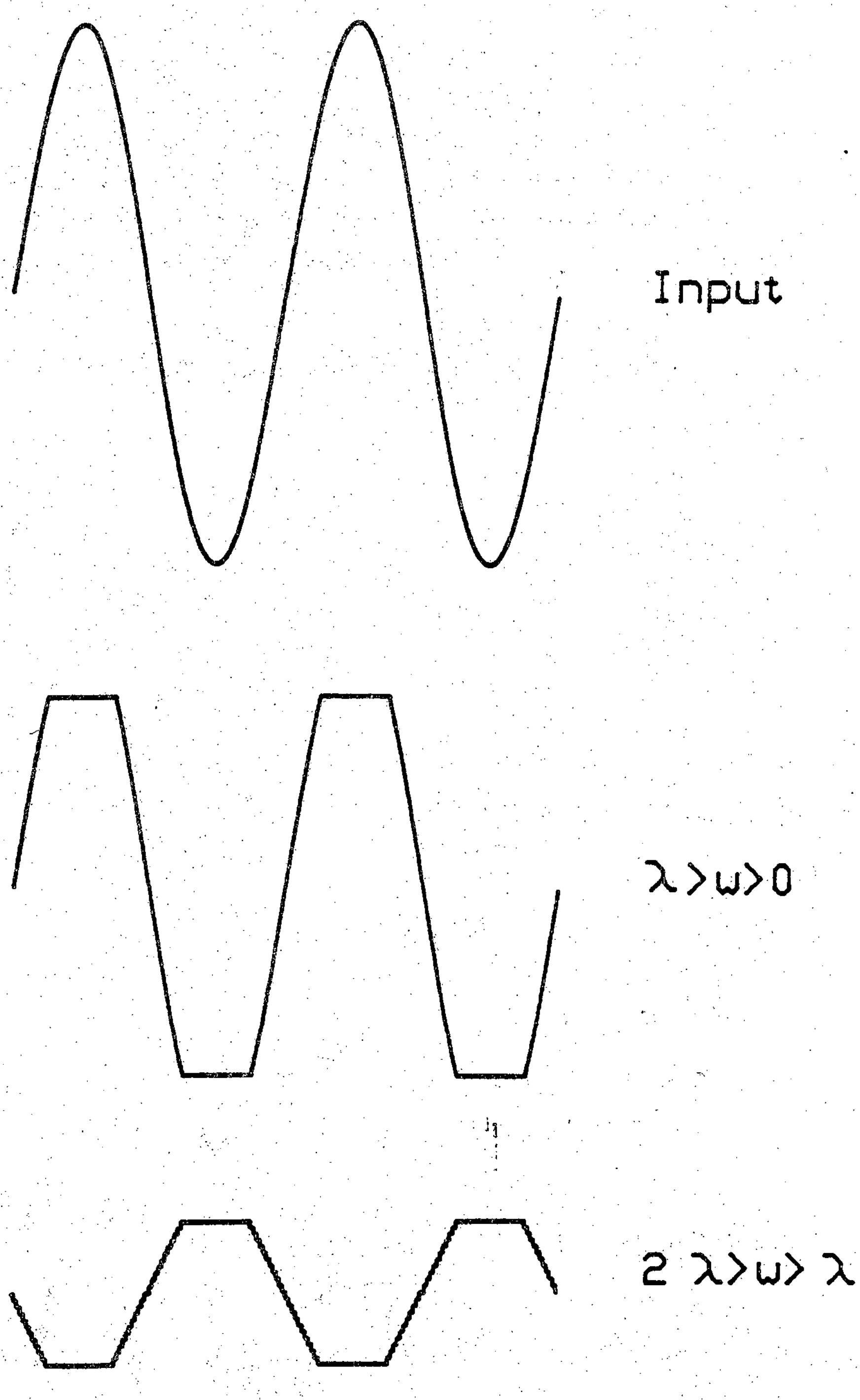

Figure 6.1
Clipping effect of analog median filter.

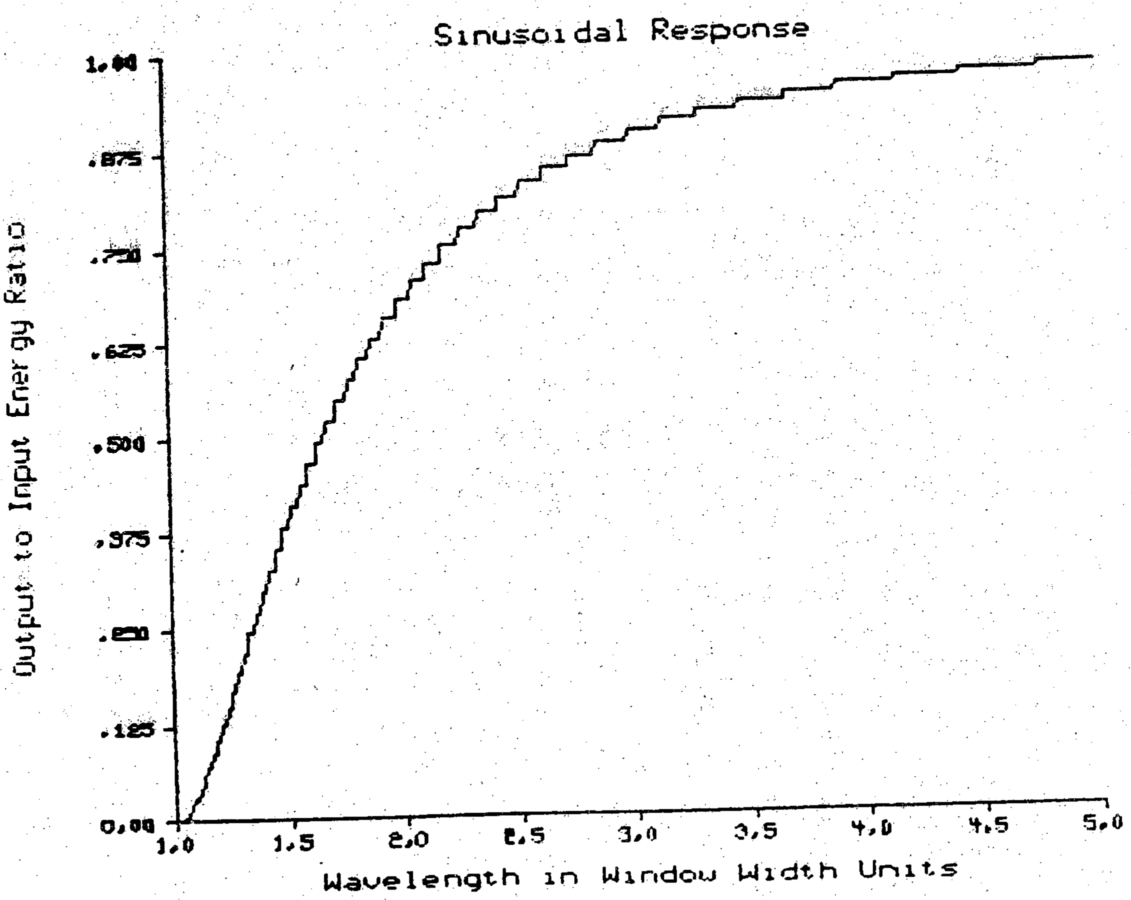

Figure 6.2
Sinusoidal response with $w \leq \lambda \leq 5w$.

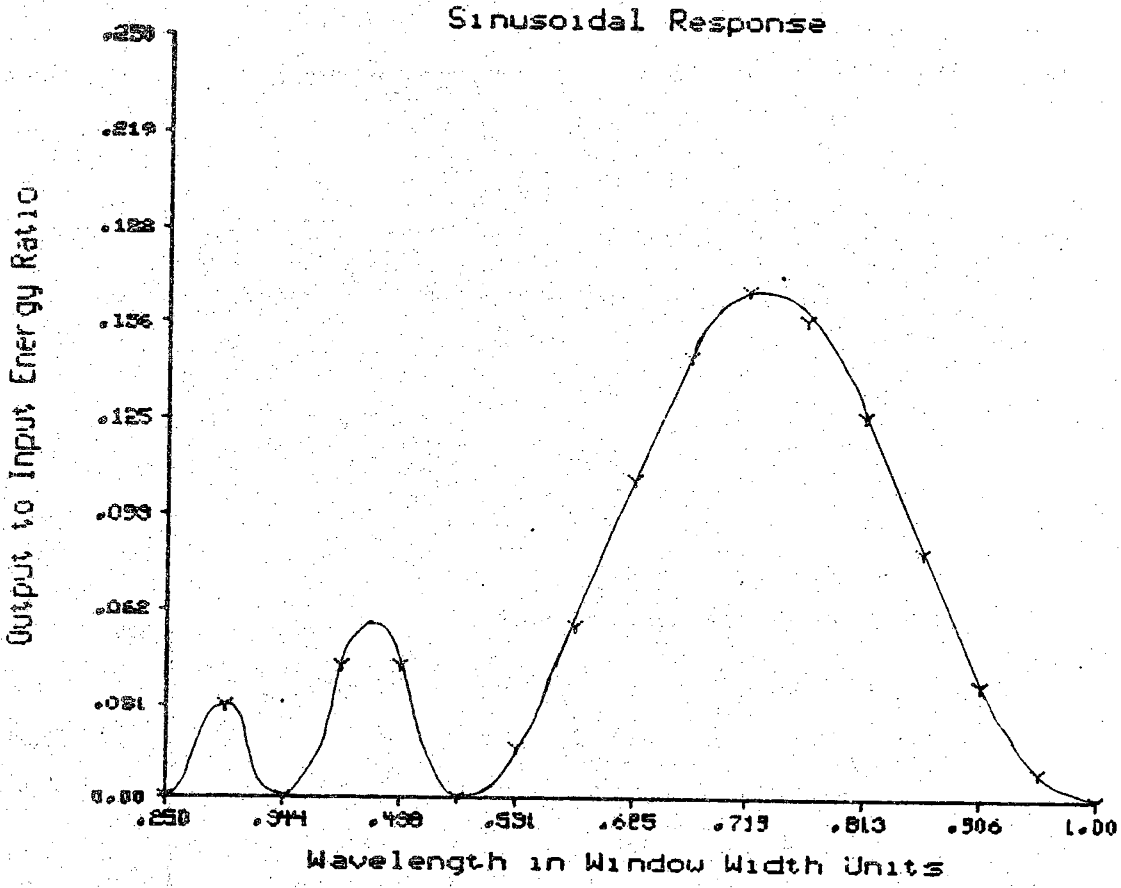

Figure 6.3
Sinusoidal response with $\frac{w}{4} \leq \lambda \leq w$.

CHAPTER 7
CONCLUSIONS

The threshold decomposition and the set of binary signals perform the same function for ranked order filters that superposition and sinusoids perform for linear filters -- they allow complex problems to be decomposed into simpler problems. This has very fortunate practical and theoretical consequences.

On the practical side, the decomposition has an important impact on the implementation of ranked order filters. It shows that a ranked order filter for a multi-level signal is simply a parallel connection of filters for binary signals. Furthermore, since the output of the ranked filter for a binary signal is found by counting the number of ones in the window and comparing the result to a threshold, these filters are trivial to implement -- complicated ranking is no longer needed. The possibility of VLSI implementation is apparent.

On the theoretical side, the decomposition shows that the analysis of the ranked order filter's effects on multi-level signals is reduced to the much simpler analysis of binary signals. It is now clear that many of the properties which were limited to binary signals can now be extended in a straightforward fashion to multi-level signals. Also, the difficult task of comparing different ranked order operators is now reduced to the binary signal domain.

The convergence of binary signals to roots is analyzed for deterministic and random sequences. For arbitrary signal levels, a direct technique for explicit solution to the number of roots of a median filter with arbitrary window width is developed. Using the symmetric tree structure of the signal space we introduce a faster technique for analytic and computer solution.

The analog median filter is defined and proposed for analysis of the standard discrete median filter in cases with a large sample size or when the associated statistics would be simpler in the continuum. Discrete filters are shown to be a subclass of analog filters. Also, an equivalence among analog filters and limits of discrete filters is established. Finally, several stochastic interpretations of the analog median filter are presented including necessary and sufficient conditions on input processes which guarantee the existence of output distributions for multiple passes of the analog median filter.

These theoretical results should improve our understanding of the behavior of these filters. The practical results should lead to the use of these filters in many new real time signal processing applications, particularly real time image processing.

LIST OF REFERENCES

LIST OF REFERENCES

[1] T.S. Huang and G.J. Yang, "Median filters and their applications to image processing," School of Electrical Engineering, Purdue University, West Lafayette, IN; TR. EE 80-1 Jan. 1980.

[2] J.B. Bednar and T.L. Watt, "Alpha-trimmed means and their relationship to median filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-32, pp. 145-153.

[3] P.J. Bickel and K.A. Doksum, *Mathematical Statistics.* San Francisco: Holden-Day, 1977.

[4] R.H. Randles and D.A. Wolfe, *Introduction to The Theory of Nonparametric Statistics.* New York: Wiley, 1979.

[5] J.W. Tukey, "Nonlinear (nonsuperposable) methods for smoothing data," in *Cong. Rec. 1974 EASCON,* pp. 673.

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

[7] T.A. Nodes and N.C. Gallagher, "Theoretical results on the properties of median type filters," School of Electrical Engineering, Purdue University, West Lafayette, IN; TR-EE 82-32, Sept. 1982.

[8] L.R. Rabiner, M.R. Sambur, and C.E. Schmidt, "Applications of a nonlinear smoothing algorithm to speech processing," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-23, pp. 552-557, Dec. 1975.

[9] G.A. Arce and N.C. Gallagher, Jr., "State description for the root-signal set of median filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-30, pp. 894-902, Dec. 1982.

[10] G.R. Arce and N.C. Gallagher, "BTC image coding using median filter roots," *IEEE Trans. Commun.,* vol. COM-31, pp. 784-793, June 1983.

[11] E. Ataman, V.K. Aatre, and K.M. Wong, "A fast method for real-time median filtering," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-28, pp. 415-420, Aug. 1980.

[12] T.S. Huang, G.T. Yang, and G.Y. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-27, pp. 13-18, Feb. 1979.

[13] N.C. Gallagher and G.L. Wise, "A theoretical analysis of the properties of median filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol.

ASSP-29, pp. 1136-1141, Dec. 1981.

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

[15] Y.H. Lee and S.A. Kassam, "Some generalizations of median filters," *Proceedings of the International Conference on Acoust., Speech and Signal Processing,* pp. 411-414, 1983.

[16] A.C. Bovik, T.S. Huang, and D.C. Munson, Jr., "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-31, pp. 1342-1349, Dec. 1983.

[17] C. Mead and L. Conway, *Introduction to VLSI Systems.* California: Addison-Wesley, 1980.

[18] K. Oflazer, "Design and implementation of a single chip 1-D median filter," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-31, pp. 1164-1168, Oct. 1983.

[19] E.J. Coyle, N.C. Gallagher, and S. Bass, "A VLSI implementation of a median filter," NSF proposal for additional funds, 1984.

[20] P.D. Wendt, E.J. Coyle and N.C. Gallagher, Jr., "A limit on the number of passes to a root for median filters," *Proceedings of the Conference on Information Sciences and Systems,* March 1984.

[21] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume I.* New York: Wiley, 1968.

[22] A.C. Bovik, T.S. Huang, and D.C. Munson, Jr., "Image restoration using order-constrained least-squares methods," *Proceedings of the International Conference on Acoust., Speech and Signal Processing,* pp. 828-831, 1983.

[23] H.L. Royden, *Real Analysis.* New York: MacMillan, 1968.

[24] J.L. Doob, *Stochastic Processes.* New York: John Wiley & Sons, 1953.

[25] M.J. Lighthill, *Introduction to Fourier Analysis and Generalised Functions.* Cambridge: University Press, 1958.

APPENDICES

Appendix A
Recursive Median Filtering

In this Appendix, the following Theorem, a property analogous to Theorem 2.2, is proven for recursive median filters.

Theorem A: The root signal associated with a window width 2n+1 recursive median filter can be obtained by thresholding the original signal, applying the recursive median filter to the resulting binary signals, and then mapping these binary roots back to the k-level root signal using the addition function f(•).

The proof of this theorem is best presented as a series of properties for recursive median filters. Throughout this section of the paper let $y_r(m)$ denote the output at position m of a recursive median filter with window width 2n+1 moving left to right across the input sequence a(m). Define the level i threshold decomposition of the original signal at position m to be

$$t_0^i(m) = \begin{cases} 1 & ,\text{if } a(m) \geq i \\ 0 & ,\text{if } a(m) < i \end{cases} \tag{A.1}$$

with $1 \leq m \leq L$ and $1 \leq i \leq k-1$. Recursive median filtering the thresholded values gives another binary sequence

$$x_r^i(m) = \text{median}\left\{ x_r^i(m-n),\ldots,x_r^i(m-1),t_0^i(m), \ldots, t_0^i(m+n) \right\} \tag{A.2}$$

Property A.1: $y_r(1) = f(\ x_r^i(1),\ 1 \leq i \leq k-1\ ) = \sum_{i=1}^{k-1} x_r^i(1)$

That is, the reconstruction function, f(•), works for the first point in the input signal. Note that a similar property holds at the last point of the signal when a recursive median filter moving from right to left across the signal is used.

Proof: To start the filter at position one, N points of value a(1) are appended to the beginning of the signal. The output of the recursive filter at position one is

$$y_r(1) = \text{median}\,(a(1), \ldots, a(1), a(2), \ldots, a(n))$$

$$= a(1)$$

$$= \sum_{i=1}^{k-1} t_0^i(1) \qquad (A.3)$$

The level i threshold decomposition sequences recursively median filtered at position m are given by $x_r^i(m)$. So at position one, after appending n points, we have

$$x_r^i(1) = \text{median}\left\{ t_0^i(1),\ldots, t_0^i(1), t_0^i(2),\ldots, t_0^i(n) \right\}$$

$$= t_0^i(1) \qquad (A.4)$$

Substituting Equation (A.4) into Equation (A.3), we obtain

$$y_r(1) = \sum_{i=1}^{k-1} x_r^i(1)$$

$$= f\left\{ x_r^i(1),\ 1 \leq i \leq k-1 \right\} \qquad (A.5)$$

Which shows that the function f(•) for the standard median filter will also reconstruct the recursive median filtered threshold values at position one of the signal.

□

Property A.2: If $x_r^p(m) = 1$ then $x_r^q(m) = 1$, $1 \leq q \leq p$, $1 \leq m \leq L$.

If the recursively filtered binary sequences were stacked according to threshold value, then the interpretation of this property is that a one occurring at some level implies all the binary sequences of smaller threshold levels are also one.

Proof by induction: We first note that by Property A.1 this claim is true at position one of the signal. If assuming that the property holds at positions one through m-1 implies that it is valid at position m, we then know by induction that it holds for any position of the signal. So assume that m is fixed and $x_r^p(n) = 1$ implies $x_r^q(n) = 1$ for all q such that $1 \leq q \leq p$ and $1 \leq n \leq m-1$. If for some n between one and m-1 we know that $x_r^p(n) = 0$ then we cannot say whether $x_r^q(n)$ is 0 or 1 for $1 \leq q < p$. That is, $x_r^p(n) = 0$ implies that $x_r^q(n) \geq x_r^p(n) = 0$ for any $1 \leq q \leq p$. In other words the number of ones in the sequence $x_r^p(m-n),\ldots,x_r^p(m-1)$ can only stay the same or increase as the parameter p is decreased to q, $1 \leq q \leq p$. This allows us to conclude that if

$$1 = x_r^p(m)$$

$$= \text{median}\left\{ x_r^p(m-n), \ldots, x_r^p(m-1), t_0^p(m), \ldots, t_0^p(m+n) \right\}$$

then

$$1 = \text{median}\left\{ x_r^q(m-n), \ldots, x_r^q(m-1), t_0^p(m), \ldots, t_0^p(m+n) \right\} \qquad \text{(A.6)}$$

Similarly by the definition of the threshold decomposition, we know that $t_0^p(n) = 1$ implies that $t_0^q(n) = 1$ for $1 \leq q \leq p$ and $1 \leq n \leq L$. Therefore, the number of ones in the window cannot decrease when we replace $t_0^p(n)$ with $t_0^q(n)$ in Equation (A.6). This leads to

$$1 = \text{median}\left\{ x_r^q(m-n), \ldots, x_r^q(m-1), t_0^q(m), \ldots, t_0^q(m+n) \right\}$$

$$= x_r^q(m)$$

for $1 \leq q \leq p$. Summarizing, Property A.2 holds at position m whenever it holds for positions one through m-1. But, as was noted at the beginning of the proof, Property A.2 always holds at position one of the signal. By induction on m, the proof of Property A.2 is complete.

□

Property A.3: If the first m-1 positions of the signal can be successfully decomposed and reconstructed for a recursive median filter, then $x_r^i(n) = 1$ if and only if $y_r(n) \geq i$, $1 \leq n \leq m-1$. That is, if

$$y_r(n) = f\left\{ x_r^i(n),\ 1 \leq i \leq k-1 \right\}$$

$$= \sum_{i=1}^{k-1} x_r^i(n) \qquad \text{(A.7)}$$

for all n such that $1 \leq n \leq m-1$ where m is any fixed integer from two to the signal length L, then $x_r^i(n) = 1$ if and only if $y_r(n) \geq i$, $1 \leq n \leq m-1$.

Proof: Begin the proof by thresholding the recursively filtered values $y_r(n)$ for $1 \leq n \leq m-1$. This gives for each i a sequence $t_r^i(n)$, where

$$t_r^i(n) = \begin{cases} 1 & ,\text{if } y_r(n) \geq i \\ 0 & ,\text{else} \end{cases} \tag{A.8}$$

Using the assumptions of this property given in Equation (A.7), we obtain

$$t_r^i(n) = \begin{cases} 1 & ,\text{if } \sum_{q=1}^{k-1} x_r^q(n) \geq i \\ 0 & ,\text{else} \end{cases}$$

Now invoking Property A.2, the previous equation reduces to

$$t_r^i(n) = \begin{cases} 1 & ,\text{if } x_r^i(n) = 1 \\ 0 & ,\text{else} \end{cases} \tag{A.9}$$

Combining Equations (A.8) and (A.9), we have $x_r^i(n) = 1$ if and only if $y_r(n) \geq i$, $1 \leq n \leq m-1$.

□

This result is similar to Lemma 2 for standard median filters. However, it is important to note that we assumed the decomposition worked for the first m-1 positions.

Property A.4: If the first m-1 positions of the signal can be successfully decomposed and reconstructed for a recursive median filter then so can the m'th position.

Proof: First note that because the same hypothesis is used, any results from the proof of Property A.3 can be used in this proof. Examine the recursively filtered binary sequences at position m

$$x_r^i(m) = \text{median}\left\{ x_r^i(m-n), \ldots, x_r^i(m-1), t_0^i(m), \ldots, t_0^i(m+n) \right\} \tag{A.10}$$

$$= \begin{cases} 1 & ,\text{if } \sum_{n=1}^{n} x_r^i(m-n) + \sum_{n=0}^{n} t_0^i(m+n) \geq n+1 \\ 0 & ,\text{if } \sum_{n=1}^{n} x_r^i(m-n) + \sum_{n=0}^{n} t_0^i(m+n) < n+1 \end{cases}$$

$$= I\left\{ \sum_{n=1}^{n} x_r^i(m-n) + \sum_{n=0}^{n} t_0^i(m+n) \geq n+1 \right\}$$

$$= I\left\{ \sum_{n=1}^{n} x_r^i(m-n) \geq n+1-\sum_{n=0}^{n} t_0^i(m+n) \right\}$$

$$= I\left\{ \text{At least } n+1-\sum_{n=0}^{n} t_0^i(m+n) \text{ elements in } \right.$$

$$\left. x_r^i(m-n), \ldots, x_r^i(m-1) \text{ equal one} \right\}$$

Now using Property A.3 we can change the filtered threshold values in the argument of the indicator function to the filtered values of the original signal denoted by the y_r's.

$$x_r^i(m) = I\left\{ \text{At least } n+1-\sum_{n=0}^{n} t_0^i(m+n) \text{ elements in } \right.$$

$$\left. y_r(m-n), \ldots, y_r(m-1) \text{ are } \geq i \right\} \qquad \text{(A.11)}$$

The analysis of the filtered version of the original signal follows

$$y_r(m) = \text{median}\left\{ y_r(m-n), \ldots, y_r(m-1), a(m), \ldots, a(m+n) \right\} \qquad \text{(A.12)}$$

$$= \max\left\{ 0,\ i: \text{At least } n+1 \text{ of } y_r(m-n), \ldots, a(m+n) \text{ are } \geq i \right\}$$

Using the threshold decomposition, we know that the number of $a(m), \ldots, a(m+n)$ which are greater than or equal to i is given by $\sum_{n=0}^{n} t_0^i(m+n)$. Therefore,

$$y_r(m) = \max\left\{ 0,\ i: \text{At least } n+1-\sum_{n=0}^{n} t_0^i(m+n) \text{ elements in} \right.$$

$$\left. y_r(m-n), \ldots, y_r(m-1) \text{ are } \geq i \right\}$$

$$y_r(m) = \max\left\{ 0,\ i: I(\text{ At least } n+1-\sum_{n=0}^{n} t_0^i(m+n) \text{ elements in} \right.$$

$$\left. y_r(m-n), \ldots, y_r(m-1) \text{ are } \geq i\ ) = 1 \right\} \qquad \text{(A.13)}$$

and by Equation (A.11), we have

$$y_r(m) = \max\left\{ 0,\ i: x_r^i(m) = 1 \right\}$$

Using Property A.2 we convert this to a sum

$$y_r(m) = \sum_{i=1}^{k-1} x_r^i(m)$$

$$= f\left\{ x_r^i(m),\ 1 \leq i \leq k-1 \right\}$$

□

Property A.4 states that if the threshold decomposition and reconstruction works for the first m-1 positions of the signal then it works for the m'th position. Using Property A.1 we know the function f(•) always works for recursive filters at position one of the signal. Therefore the combination of these two properties provides an inductive proof that the threshold decomposition technique using recursive filters works at any position of the signal. The proof of Theorem A is complete.

Since recursive median filters converge in one pass, the algorithm introduced in the preceding theorem has many practical advantages over the algorithm using the standard median filter. A simple parallel architecture with one binary recursive median filter preceded by a threshold device may be used for each level.

We note here that a simple modification of the proof of Theorem A shows that the same results hold if we substitute any n'th order operation for the median operation. Thus, fast implementations and analytically useful decompositions exist for these filters as well.

Appendix B
Multidimensional Ranked Order Operations

Assume, as always, that the input signal is a discrete sequence of length L which takes on the value a(m) at position m, $1 \leq m \leq L$, and that for each m, a(m) is quantized to one of the k integer values $0,1,\ldots,k-1$. Let $\phi_w^r(a,m)$ be the output at position m of a rank r(m) and window shape w(m) filter applied to the input sequence a(m). The window shape is allowed to change with position. The results of the following analysis are therefore valid in multidimensional applications. For convenience, let N(m) denote the number of positions in the window w(m). The filter at position m can be written as

$$\phi_w^r(a,m) = \text{the r(m)th largest element of } \left\{ a(p)\text{: } p \in w(m) \right\} \qquad \text{(B.1)}$$

Define the threshold decomposition of the original signal at position m to be the set of binary sequences

$$t_0^i(m) = \begin{cases} 1 & \text{,if } a(m) \geq i \\ 0 & \text{,if } a(m) < i \end{cases} \qquad \text{(B.2)}$$

with $1 \leq m \leq L$ and $1 \leq i \leq k-1$. Applying the ranked filter to these thresholded values gives another set of binary sequences

$$\phi_w^r(t_0^i,m) = \text{the r(m)th largest element of } \left\{ t_0^i(p)\text{: } p \in w(m) \right\} \qquad \text{(B.3)}$$

The remainder of this Appendix is devoted to the development of three properties of filters with position dependent window shapes and ranks. These three properties lead directly to a superposition principal for generalized rank filters.

In the following proofs it is often convenient to use the indicator function of the event A, which is given by

$$I(A) = \begin{cases} 1 & \text{,if A is true} \\ 0 & \text{,if A is false} \end{cases}$$

The following property, although easily shown, allows the filtered binary sequences to be specified at all threshold levels by knowing the highest threshold level at which a one occurs.

Property B.1: If $\phi_w^r(t_0^i,m) = 1$ then $\phi_w^r(t_0^q,m) = 1,\ 1 \le q \le i$.

That is, if the filtered threshold value is one at level i, then it is one for all levels less than i.

Proof: First examine the output of the filter applied to the threshold decomposition of the input signal. For any fixed i, $1 \le i \le k-1$,

$$\phi_w^r(t_0^i,m) = \text{the r(m)th largest element of } \left\{ t_0^i(p) \colon p \in w(m) \right\}$$

$$= \begin{cases} 1 & \text{,if at least } N(m)+1-r(m) \text{ of } (t_0^i(p)\colon p \in w(m)) \text{ equal one} \\ 0 & \text{,if at least } r(m) \text{ of } (t_0^i(p)\colon p \in w(m)) \text{ equal zero} \end{cases}$$

$$= I\left\{ \text{at least } N(m)+1-r(m) \text{ of } (t_0^i(p)\colon p \in w(m)) \text{ equal one} \right\} \qquad \text{(B.4)}$$

Which by the definition of threshold decomposition, see Equation (B.2), becomes

$$= I\left\{ \text{at least } N(m)+1-r(m) \text{ of } (a(p)\colon p \in w(m)) \text{ are } \ge i \right\} \qquad \text{(B.5)}$$

Using Equation (B.5) we know that if

$$1 = \phi_w^r(t_0^i,m)$$

then

$$1 = I\left\{ \text{at least } N(m)+1-r(m) \text{ of } (a(p)\colon p \in w(m)) \text{ are } \ge i \right\}$$

which implies for any q less than or equal to i

$$1 = I\left\{ \text{at least } N(m)+1-r(m) \text{ of } (a(p)\colon p \in w(m)) \text{ are } \ge q \right\}$$

$$= \phi^r_w(t^q_0,m)$$

□

The relation of the binary valued, ranked order filtered threshold sequences to the output of the filter applied to original k-level signal is provided by

Property B.2: There exists a mapping f(•) from the set of filtered threshold decomposed sequences $\phi^r_w(t^i_0,m)$, $1 \leq i \leq k-1$ to the signal space of k-leveled signals such that $\phi^r_w(a,m) = f(\phi^r_w(t^i_0,m),\ 1 \leq i \leq k-1)$.

Proof by construction:

$$\phi^r_w(a,m) = \text{the r(m)th largest element of} \left\{ a(p)\text{: } p \in w(m) \right\}$$

$$= \max \left\{ 0,i\text{: At least } N(m)+1-r(m) \text{ of } (a(p)\text{: } p \in w(m)) \text{ are } \geq i \right\}$$

$$= \max \left\{ 0,i\text{:}I(\text{At least } N(m)+1-r(m) \text{ of } (a(p)\text{: } p \in w(m)) \text{ are } \geq i)=1 \right\} \quad \text{(B.6)}$$

Combining Equations (B.5) and (B.6)

$$\phi^r_w(a,m) = \max \left\{ 0,i\text{: } \phi^r_w(t^i_0,m) = 1 \right\}$$

which by Property B.1 implies

$$\phi^r_w(a,m) = \sum_{i=1}^{k-1} \phi^r_w(t^i_0,m) \quad \text{(B.7)}$$

$$= f(\phi^r_w(t^i_0,m),\ 1 \leq i \leq k-1)$$

□

Any rank value can be obtained by summing the corresponding threshold decomposed rank values over all possible threshold levels. In loose terms, the function f(•) stacks the binary signals $\phi^r_w(t^i_0,m)$ on top of one another starting with i=1. The value of the output at position m is then the highest level at position m in the stack at which a one appears.

The function f(•) constructed above is shown to be the inverse of the threshold decomposition by

Property B.3: The binary sequences $t_1^i(m)$, $1 \leq m \leq L$, and $1 \leq i \leq k-1$, obtained by thresholding $\phi_w^r(a,m)$, $1 \leq m \leq L$, are identical to the filtered threshold values of the original signal $\phi_w^r(t_0^i,m)$, $1 \leq m \leq L$, and $1 \leq i \leq k-1$, (see Property B.2).

Proof:

$$t_1^i(m) = \begin{cases} 1 & \text{,if } \phi_w^r(a,m) \geq i \\ 0 & \text{,if } \phi_w^r(a,m) < i \end{cases}$$

Using Equation (B.7) from the proof of Property B.2, we obtain

$$t_1^i(m) = \begin{cases} 1 & \text{,if } \sum_{q=1}^{k-1} \phi_w^r(t_0^q,m) \geq i \\ 0 & \text{,else} \end{cases}$$

by Property B.1

$$t_1^i(m) = \begin{cases} 1 & \text{,if } \phi_w^r(t_0^i,m) = 1 \\ 0 & \text{,else} \end{cases}$$

$$= \phi_w^r(t_0^i,m)$$

□

The above results show how one pass of a generalized rank filter over the input signal is equivalent to first thresholding the signal then filtering each threshold sequence and finally reconstructing the output using the function f(•). This operation can clearly be repeated for a series of generalized ranked filters. A simple inductive argument based on the fact that f(•) and the decomposition are inverse operations between the binary and k level signals shows that the intermediate reconstructions can be omitted. These results are summarized in

Theorem B: The output of a series of ranked order filters with position dependent window shapes applied to a k-level input signal is identical to the superposition, using the function f(•), of the filtered threshold decomposition of the input signal.

This result provides a new tool for both the implementation and the analysis of the ranked oreder filtering operations applied to arbitrary level signals. Examples of this technique applied to recursive median, separable median, and weighted rank filters are given in Chapter 2 of this Thesis.