MEDIAN FILTERS THEORY AND APPLICATIONS

Milan STORK

e-mail: stork@kae.zcu.cz

University of West Bohemia, Faculty of Electrical Engineering, Department of Applied Electronics, 30614 Plzen, Czech Republic

Keywords: Median filter, recursive median filter, weighted median filter

ABSTRACT
The aim of this paper is to present of standard median and recursive median one-dimensional nonlinear filters. Median filtering is a popular method of noise removal, employed extensively in applications involving speech, signal and image processing. This non-linear technique has proven to be a good alternative to linear filtering as it can effectively suppress impulse noise while preserving edge information. The applications in industry and biomedical signal processing are presented in this article.

I. INTRODUCTION
The standard median (SM) filter is a simple nonlinear smoother that can suppress noise while retaining sharp sustained changes (edges) in signal values. It is particularly effective in reducing impulsive-type noise [1]. The output of SM filter at a point is the median value of the input data inside the window centered at the point. If is the input value before the filter window is moved to the next position. With the same amount of operation, the RM filter usually provides better smoothing capability than the SM filter, at the expense of increased distortion.

II. SOME PROPERTIES OF MEDIAN FILTERS
In RM filter output computation is important point for median filters of window width 2N+1, then

- A constant neighborhood is a region of at last N consecutive identically valued samples.
- An edge is a monotonically rising or falling set of samples surrounded on both sides by constant neighbourhoods of different values.
- An impulse is set of at most N samples whose values are different from the surrounding regions and whose surroundings regions are identically valued constant neighbourhoods.
- An oscillation is any signal structure which is not part of a constant neighbourhood, an edge or an impulse.

Example 1: Consider a length N=2 SM filter. Input vector x = [-1, 5, 8, 11, -2]. After sorting the samples inside filter window are: [11, 8, 5, -1, -2]. The filter output y = 5 (Window is centered at the sample value 5).

For RM filter output computation is important point out that previous outputs {y(k-N),... y(k - 2)} are not necessary to determine the present output y(k), and are therefore redundant.

Property 1: In RM filtering, the output y(k) is represented by

y(k) = med[y(k-1),.., y(k-1), x(k), x(k+1),.., x(k+N)] (3)

Note: y(k-1) is used N times in equation (3)!

Proof: The output y(k) is a locally monotonic sequence of length N+1, that is, y(k-N), y(k) is either nondecreasing or nonincreasing for any k [5]. Consider the nondecreasing case, y(k-N) ≤..≤ y(k-1). Since the output is locally monotonic of length N+1, y(k-1)≤ y(k). This implies that if the number of samples in {x(k),...,x(k+N)}
that are greater than or equal to \(y(k-1)\) is \(q\), then \(q \geq 1\) and clearly \(y(k) = y(k-1)\) if \(1 \leq q \leq N\), and \(y(k) = \min(x(k),...,x(k+N))\) if \(q = N+1\) because, otherwise, the local monotonicity is violated. Thus \(y(k)\) is represented as

\[ y(k) = \min(x(k),...,x(k+N)), \] if \(x(k+i) \geq y(k-1)\)

for all \(i, 0 \leq i \leq N\), \(y(k) = y(k-1)\), otherwise, and

\[ y(k) = \text{med}[y(k-1),...,y(k-1),x(k),x(k+1),...,x(k+N)] \]

This can be proved similarly for nonincreasing case.

Obviously, the implementation of RM filtering will become easier by using this property. The following property is a direct sequel to Property 1 and is particularly useful in implementing 1-D RM filters on a general purpose computer.

**Property 2:** The output \(y(k)\) of the RM filter is given by

\[ y(k) = \text{med}[x_{\min}, x_{\max}, y(k-1)] \tag{4} \]

where \(x_{\min}\) and \(x_{\max}\), respectively, are the minimum and the maximum of \(\{x(k),...,x(k+N)\}\). By incorporating this property fast 1-D RM filters may be obtained.

The output of the RM filter, can be represented as a Boolean expression. For example, under the assumption that the input to this filter is restricted to binary values,

\[ y(k) = y(k-1) \lor [x(k) \land x(k+1) \land \ldots \land x(k+N)] \land [x(k) \lor x(k+1) \lor \ldots \lor x(k+N)] \tag{5} \]

where represent the OR operation by \(\lor\), and the AND operation by \(\land\). Fig. 1 illustrates the logic network realizing the Boolean function in (5). The output of SM filter with multilevel input sequence can be obtained from the output of binary SM filters.

The SM filter for binary signals may by implemented by using counter followed by comparator [8], which is more complicated than the proposed RM filtering algorithm. In [6], in order to avoid the difficulty in implementing SM filters with digital logic, an analog circuit structure was developed. The analog circuit for SM filtering, however, may require development of a customized chip, which is usually expensive, while proposed digital structure admits inexpensive implementation using logic array chips.

RM filters have the threshold decomposition property [9], based on which they can be implemented, as shown in Fig. 2. Here the input is assumed to be an \(M\)-level signal. The implementation based on the threshold decomposition property is parallel and modular, but its hardware complexity grows exponentially with the number of bits in the inputs.

Like in linear filtering, it is common to use cascades of median based filters for, e.g. analysis and filtering purposes. Theoretical analysis of median filter cascades is difficult. Fortunately, cascades of SM filters can be often represented by single weighted median (WM) filter.

**Definition 1:** For the discrete-time continuous-valued input vector \(x = [x_1, x_2, \ldots, x_p]\), the output \(y\) of the WM filter of span \(N\) associated with the integer weights

\[ w = [w_1, w_2, \ldots, w_R] \tag{6} \]

is given by

\[ y = \text{MED}[w_1 \hat{\lor} x_1, w_2 \hat{\lor} x_2, \ldots, w_R \hat{\lor} x_R] \tag{7} \]

where \(R = 2^N + 1\) and \(\text{MED}[\cdot]\) denotes the median operation and \(\hat{\lor}\) denotes duplication, i.e.:

\[ p \hat{\lor} x = x, x, \ldots, x \tag{8} \]

\((x\ is\ used\ p\ times)\)

Example 2: Consider a length 5 WM filter with integer weights \([1, 2, 3, 2, 1]\). Now apply the filter to the following sequence so that the window is centered at the sample value 9.

\[ x = [-2, 4, 9, 12, -3] \]

After sorting and duplication, the samples inside filter window are \([12, 12, 9, 9, 4, 4, -2, -3]\). The WM filter output is \(y = 9\), whereas, the 5-point SM would have produced the result \(y = 4\).

By cascade connection of filters \(F\) and \(G\) we mean that the original input signal \(X\) is filtered by filter \(F\) which produces an intermediate signal \(Y\). This in turn is filtered by filter \(G\) which produces the output signal \(Z\). Cascaded filters \(F\) and \(G\) can be represented as a single filter \(H\) which produces the output directly from the input \(X\). The maximum window length of filter \(H\) is \(RH = RF + RG - 1\), where \(RF\) and \(RG\) are window lengths of filters \(F\) and \(G\), respectively.

Some cascade connection examples of SM filters are shown in Table 1.
\[ [1,1,1]^2 = [1,2,3,2,1] \]
\[ [1,1,1]^3 = [1,2,5,7,5,2,1] \]
\[ [1,1,1]^4 = [1,2,5,12,17,12,5,2,1] \]
\[ [1,1,1]^5 = [1,2,5,12,29,70,99,70,29,12,5,2,1] \]
\[ [1,1,1,1,1] [1,1,1] = [1,2,3,3,3,2,1] \]
\[ [1,1,1,1,1] [1,1,1]2 = [1,3,7,10,11,10,7,3,1] \]
\[ [1,1,1]^2 [1,1,3,1,1]1 = [1,1,4,9,13,9,4,1,1] \]

**Table 1. Some cascade SM filters**

<table>
<thead>
<tr>
<th>Window Size</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>SM filter 2, RM filter 2</td>
</tr>
<tr>
<td>50</td>
<td>SM filter 50, RM filter 50</td>
</tr>
<tr>
<td>4</td>
<td>SM filter 4, RM filter 4</td>
</tr>
</tbody>
</table>

**Figure 3:** Example of filtering sinusoidal (left) and sawtooth (right) signal by means of SM and RM filters. a) input signal, b) SM filter N=2, c) SM filter N=50, d) RM filter N=2, e) RM filter N=8.

**III. SERIAL BIT REALIZATION OF RM FILTER**

Suppose that the input level \( M = 2 \) \((P\)-bit signal). To simplify notation, denote \( \{A_0, A_1... A_{N+1}\} \) the set of data in the window \( \{y(k-1), x(k), ..., x(k+N)\} \) where \( A_0 = y(k-1) \) and \( A_j = x(k+i-1) \), \( 1 \leq i \leq N+1 \). Let the code words (radix-2 binary representation) of \( A_i \) and \( y(k) \) respectively, be \( (a_i^1, a_i^2, ..., a_i^P) \) and \( (b_1, b_2, ..., b_P) \) where \( a_i^1 \) and \( b_i^1 \) are the most significant bits. In the bit-serial realization, the output at each bit is obtained sequentially, starting with most significant bit. At each bit-level, with the exception of the most significant bit, the binary input values at the level are modified before filtering depending on the outputs of more significant bits. To implement the RM filter using the bit-serial approach and the binary RM filter, the input data at each level should be modified as in the following property.

**Property 3:** The output of the binary RM filter at the \( j \)th bit is given by

\[
\begin{align*}
\hat{a}_j^i &= \hat{a}_j^i \land \hat{a}_j^i \lor \hat{a}_j^i \land \hat{a}_j^{N+1} \\
\land \hat{a}_j^i \land \hat{a}_j^{N+1} \\
\end{align*}
\]

where \( \hat{a}_j^i = a_j^i \), \( 0 \leq i \leq N+1 \), and for each \( j \), \( 2 \leq j \leq P \),
\[
\hat{a}_j^i = a_j^i, \\
\text{if } a_j^m = b_j^m \text{ for all } m, \ 1 \leq m \leq j-1, \text{ or} \\
\hat{a}_j^i = a_j^i, \text{if } a_j^m = b_j^m \text{ for all } m, \ 1 \leq m \leq r-1, \text{ and for some } r, \ 1 \leq r \leq j-1, \ a_j^r \neq b_j^r.
\]

This property was proved in [8]. Equation (7) can be rewritten as follows:

\[
\hat{a}_j^i = \hat{a}_j^i \land \hat{a}_j^i \lor \hat{a}_j^i \land \hat{a}_j^{N+1} \\
\land \hat{a}_j^i \land \hat{a}_j^{N+1}
\]

where \( U_j^i = [\hat{a}_j^i \land b_j^i] \lor \hat{a}_j^i \land \hat{a}_j^{N+1} \), and \( U_j^i = 0 \). Exclusive-OR operation is represent by \( \oplus \), and the complement operation by \( \bar{\cdot} \), the OR operation by \( \lor \), and the AND operation by \( \land \). This Boolean expression is useful for VLSI or microcomputer realization.

**Figure 4:** Example of ECG (electrocardiograph) signal filtered by means of SM and RM filters. a) input signal, b) SM filter \( N = 4 \), c) RM filter \( N = 8 \), Sampling frequency = 960 Hz.

**Figure 5:** The signal from accelerometer filtered by SM and RM filters. a) Input signal, accelerometer fixed on arm, b) SM filter \( N = 15 \), c) RM filter \( N = 6 \) d) Input signal, accelerometer fixed on leg, e) SM filter \( N = 10 \), f) RM filter \( N = 8 \), Sampling frequency = 960 Hz., Physical activity - fast walk.

**IV. APPLICATIONS EXAMPLES**

In this part some examples of SM and RM are shown. The 1-D SM and RM filters were implemented on personal computer and tested for different types of data. Fig. 3 illustrates example of sinusoidal (left) and saw-tooth (right), contaminated with impulse noise filtered by SM and RM filters with different window size.

From results, presented in Fig. 3 is clearly visible the different results for SM and RM filters for different window size. RM filter is more sensitive for size than SM
filter. Example of ECG signal filtering is shown in Fig. 4 [10]. One of the possibilities of the physical activity measuring can be the application of an accelerometer fixed on the human body. For medical purposes and "in field" activity testing, the device must be wireless. Therefore wireless data transmission was used. The received digital signal from accelerometer was also filtered by SM and RM filters. Fig. 5 illustrates the signal from accelerometer fixed on arm and leg, during the fast walk [11]. This device was developed for medical purposes and physical activity measuring during sport activity or stress test.

V. CONCLUSION
The nonlinear filters were considered in this paper, they are SM and RM filters. Presented filters pointed out for their usefulness in impulsive environment. In real world there are many signals which are corrupted by impulsive noise or have impulsive character. Signal and noise in this class are more likely to occur spikes or accidentally bursts of outlying observations than one would expect from normally distributed signals. As a result, their density functions decay in the tails less rapidly than Gaussian density function. Underwater acoustic signal, low-frequency atmospheric noise, radar, mobile communications, computer network traffic and many types of man-made noise have all been found to belong to this class. Nonlinear filters offers a flexible, robust approach to the problem of estimating signals in the presence of impulsive noise. Examples presented in this paper, show that in area of biomedical signal processing these filters could be useful too.

ACKNOWLEDGMENT
This paper is based upon work sponsored by the Ministry of Education of the Czech Republic under research and development project LN00B084.

REFERENCES

APPENDIX A
MEDIAN COST FUNCTION
Median filter output is the value \( \theta \) which minimizes the cost function \( F(\theta) \)
\[
F(\theta) = \sum_{k=-N}^{N} | x(k) - \theta |
\]
and after differentiating:
\[
\frac{d}{d\theta} F(\theta) = \frac{d}{d\theta} \sum_{k=-N}^{N} | x(k) - \theta | = \sum_{k=-N}^{N} \text{sign}(x(k) - \theta)
\]

Fig. A1 (left) and A2 (right). A1 - median filter function, A2 - derivative of median filter function.
Median filter example is computed for: \( x(-2) = 6, x(-1) = -2, x(0) = -4, x(1) = 5, x(2) = -1 \), after sorting \([-4, -2, -1, 5, 6]\), median = -1. Graph of median cost function is shown in Fig. A1. Graph of derivative median function \( F(\theta) \) is illustrates in Fig. A2.

**APPENDIX B**

**ONE-DIMENSIONAL MEDIAN FILTER ALGORITHM**

Given a sequence of \( n \) samples within a window:

\[ W = \{x_i\}, \text{ where: } -N \leq i \geq N. \]

Each sample \( x_i \) is represented in \( w \) binary bits, as:

\[ x_i = b_i^w b_i^{w-1} ... b_i^2 b_i^1 \]

where \( b_i^k \) is the \( k \)-th bit of \( x_i \) with a weight of \( 2^{(k-1)} \).

Similarly, denoting the median \( M \), to have the binary representation:

\[ M = m^w m^{w-1} ... m^2 m^1 \]

then the algorithm can be stated as folows:

1. Start with \( j = w \)
2. \( m^j = \text{majority} (b_i^{(N)}, b_i^{(N+1)}, b_i^{(N+2)} ... b_i^N) \)
3. Do, in parallel, for all \( i \), where: \(-N \leq i \geq N\)
   - If \( b_i^j = m^j \), then bits of \( x_i \) do not change
   - If \( b_i^j \neq m^j \) and \( b_i^j = 1 \), then all bits of \( x_i \) are set to ones
   - If \( b_i^j \neq m^j \) and \( b_i^j = 0 \), then all bits of \( x_i \) are set to zeros
4. Decrement \( j \). If \( j \neq 0 \), go to step 2
5. End

Example: Median\{12, 2, 5\}

<table>
<thead>
<tr>
<th>12</th>
<th>2</th>
<th>5 Dec. numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>L</td>
<td>L =&gt; maj_8(H, L, L) = L</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
</tbody>
</table>

in 1. row bit MSB = H and maj = L
=> all bits in 1. coll. are set to H
↓
| H  | L | L |
| H  | L | H |
| H  | L | H |

in 2. row bit = L and maj = H
=> all bits in 2. coll. are set to L
↓
| H  | L | L |
| H  | L | H |
| H  | L | L => maj_2(H, L, L) = L |
| H  | L | H |