MEDIAN FILTERS

William T. Hicks, PE
IMET Corporation

ABSTRACT

Most modern digital filtering is done by taking the average (mean) of a signal or some weighted average. Another method is to use feedback, which more closely resembles how analog filters with feedback operate. In the case of low pass filters, all these methods tend to give a trade off in getting the signal to pass while attenuating the higher frequency noise. An alternative is to use a median filter, which selects the mid value of a group of points. While this is not as computationally simple as other filters, it allows for the attenuation of noise while allowing sudden changes in signal level to pass thru unaltered. This paper discusses the characteristics of median filters and methods of implementing them.

KEY WORDS

Median Digital Filter.

INTRODUCTION

Much of today's filtering of analog signals is done in the digital domain. This is normally accomplished by first digitizing the signal with an Analog to Digital (A/D) converter and then using a computer or micro computer to process the data using mathematical functions. Once the data has been converted to digital format, the digital domain allows for filtering the signal with functions that remain stable as time and temperature change.

There are two main methods of digital filtering. The first is Infinite Impulse Response (IIR) filtering and the second is Finite Impulse Response (FIR) filtering. IIR filters use digital feedback, and of the two types, most closely relate to analog filters with feedback. By using feedback, the length of calculations is reduced to get the same order of filtering. One down side of these filters is a potential stability problem. FIR filters require more calculations per point, but do not require calculations for every input if there are fewer outputs (decimation), while IIR filters must calculate every point. Also many modern Digital Signal Processor (DSP) computers have special circuits to optimize FIR calculations.

In FIR calculations a group of the latest samples is taken and each sample multiplied by a coefficient (weight) and all are then added together. These weights are calculated using various windowing functions, such as Hanning, Hamming, Kaiser, Cosine, and others. The simplest

window is the rectangular, where all weights are equal. This results in just summing up the group of input signals [normally multiplied by 1/(number of points)], which is the same as finding the mean of the group of data points. Therefore, this rectangular FIR filter is also referred to as a Mean filter.

The Mean filter is probably the easiest to calculate, and gives filtering of the input to remove noise, while distorting the input and not having a flat pass band or very sharp rolloff. Other FIR filters using the same number of input points (taps), but having non-equal weights, can give flatter pass band, sharper rolloff, less distortion, and other different properties. There is normally some tradeoff in which properties are improved the most.

MEDIAN FILTER

The Median of a group of input points is the amplitude of the point that is in the middle after arraigning all points in ascending or descending order, based on amplitude. The median of a group of points is easy to understand, but not easy to calculate. While IIR and FIR filter have basic mathematical equations, the Median is more of data base problem. Another difference is that the Median is a non-linear process.

So with these disadvantages, why use the Median filter? Currently the Median filter is used almost exclusively for video filtering. More specifically it is use to filter shot noise from video signals. Shot noise is where single samples have a large noise value, and shows up on the video monitor as dots that are darker or lighter then their surroundings. A FIR filter could filter the noise out. However, if the noise amplitudes are very large compared to the ambient signal level, the amount of filtering needed also smoothes out the edges of legitimate signals, giving a blurring look to the picture.

The Median of a group of data with one "outlier" or shot noise spike is still the normal level. The shot noise is completely removed. At the same time, if there is an edge change in the signal, there is no filtering of the signal and the edge still passes through intact. Both of these characteristics are demonstrated in the figures at the end of this paper.

In a video signal, the filtering is done in a two dimensional calculation, but only uses the data point being tested and its 8 adjacent neighbors, or 9 points. A good description of this is in [http://www.cee.hw.ac.uk/hipr/html/median.html], where the method is described and pictures of the results are shown. A review of available papers on Median filters resulted in only papers discussing video signals. Also the papers reviewed only used very short Median filters, that is those with only a few points for the calculation.

There are probably other signals, including one dimensional signals, that would benefit from Median filtering. One possible deterrent to using Median filters is the perceived amount of calculation needed. This would especially be a problem if real time processing is required. Therefore, a computationally simple method to find the median was looked into. One program found [http://quasimodo.versailles.inra.fr/deterrents/tk/awave/median35.txt] is written in C and is a straightforward implementation. This still seemed to be too complex, especially if the

processor being used is some simple micro controller with a limited processing capability. Therefore, an attempt was made to write a shorter program to find the Median.

MEDIAN FILTER PROGRAM

The program code is located in the appendix. The program that was written is dependent on the data format, and is not as generic as the program above. However the data used in many applications is fixed point, fixed length and can take advantage of this program. The sample program assumes 16 bit fixed point data in and out. It also assumes an odd number of data points (it finds the middle value). To change it to an even number of data points, the two middle points could be averaged. The program was written for a small micro controller in a modified C used by the micro controller.

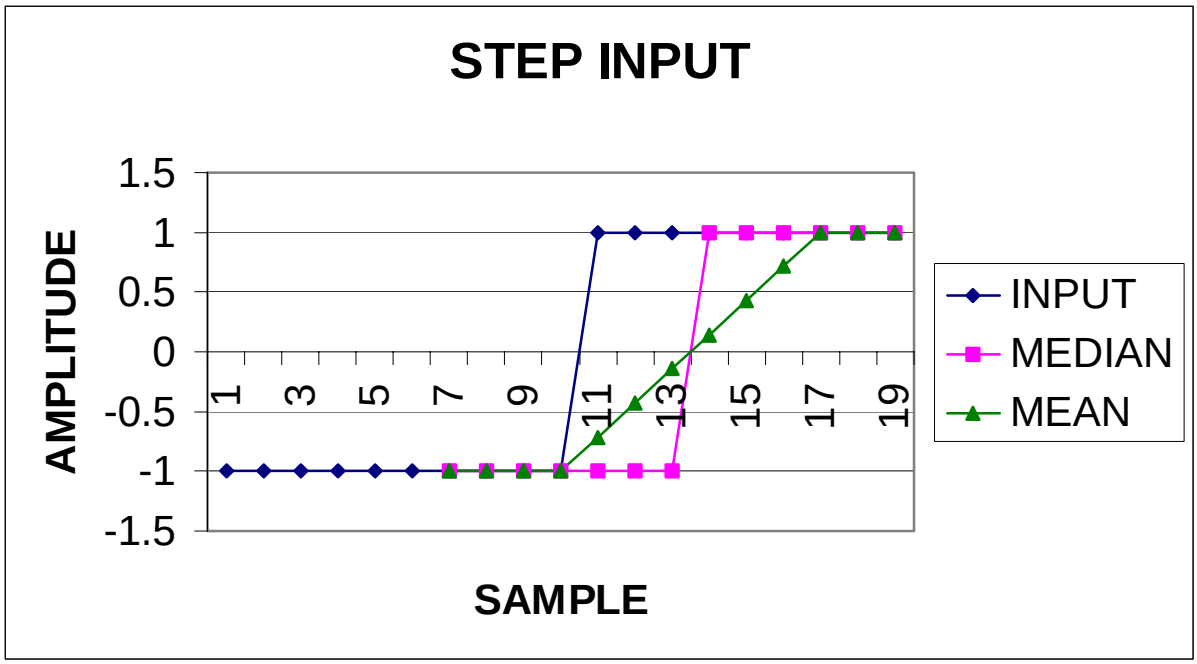
The program has an outer loop that goes through the 16 bits and an inner loop that goes through the number of data points. While this program is 16 (number of bits) longer then the FIR, it does not require any multiplications, which is helpful in small processors.

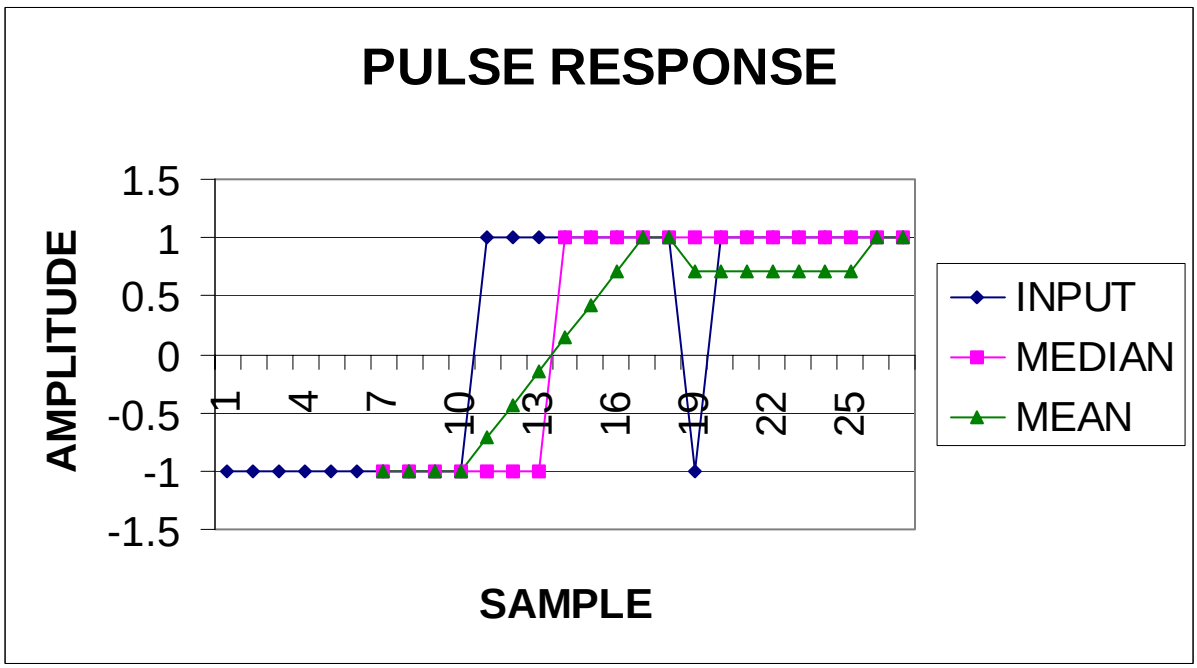
RESULTS

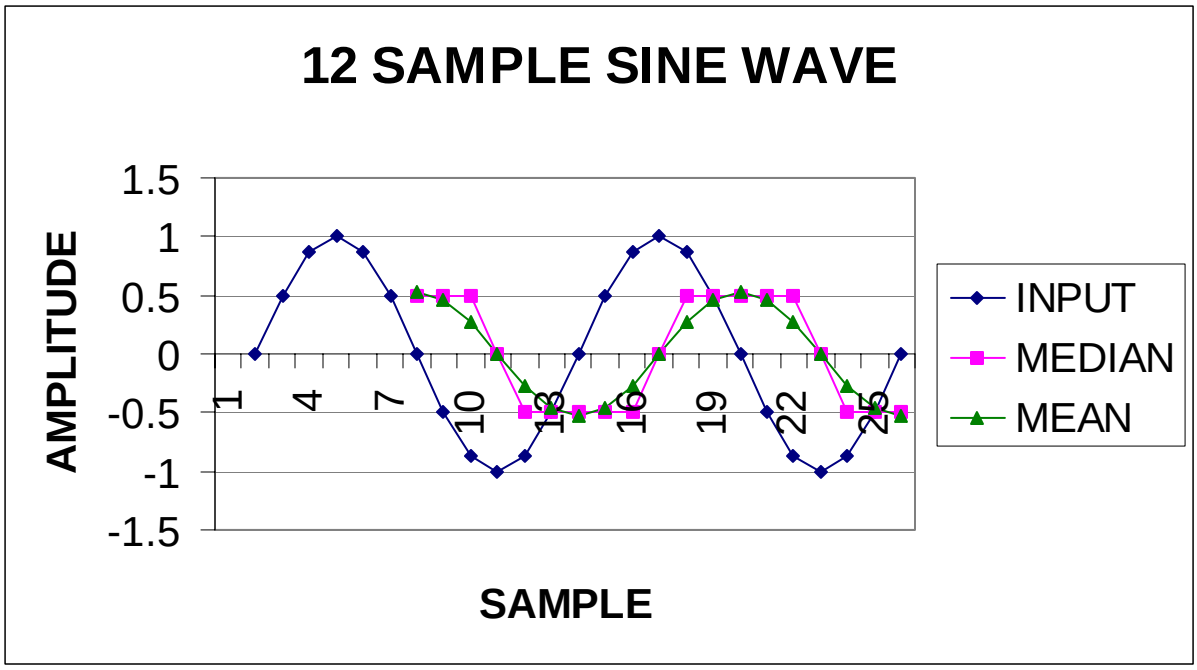
The plots shown below are done using Windows Excel. The filters are 7 samples long. The first plot shows the results of a step input into the Median and the Mean filters. Notice how the Median filter causes only a delay, and no change in the shape of the input. The Mean filter does cause a sloping of the input, resulting in loss of sudden changing information.

The second plot has one full scale impulses on the input. Notice how the Median filter totally removes the noise impulse while the Mean filter reduces, but does not remove it.

The last plot has a sine wave input that is set to one sine cycle per 12 samples. Note that the Median filter is more distorted (flat top/bottom) then the Mean filter. Other data taken showed that the distortion is less as the ratio of sine wave points per wave to the number of filter points increases. This shows that at low sine wave frequencies the Median filter approaches the lower distortion of the Mean filter.


STEP INPUT
AMPLITUDE
1.5
1
0.5
0
-0.5
-1
-1.5
1
3
5
7
9
11
13
15
17
19
SAMPLE
INPUT
MEDIAN
MEAN


PULSE RESPONSE
AMPLITUDE
1.5
1
0.5
0
-0.5
-1
-1.5
1
4
7
10
13
16
19
22
25
SAMPLE
INPUT
MEDIAN
MEAN



CONCLUSIONS

As can be seen from the data presented and the discussion given, the Median filter has characteristics that may be useful in certain filtering situations. It is very good at removing narrow noise spikes while still retaining the sharp edges of sudden changes in signal level. This is to be placed against the increased complexity of the code to generate the filter function and the increased distortion at some frequencies between impulse and pass band.

APPENDIX

Program Code

```
//NumberWords must be 3-255, odd; DataFile is int16
int16 MedianFilter(int8 NumberWords, int16 *DataFile)
{
 //int16 NumberWords;
 //int16 DataFile;
 int16 Adder = 0x8000;
 int16 Match = 0x8000;
 int16 Data;
 int8 Hcount;
 int8 Lcount;
 int8 Bits;
 int8 Words;
 int8 HalfWords;
 HalfWords = NumberWords >> 1;
 for (Bits = 0; Bits < 16; Bits ++)//test each bit
```

```
 {
  Adder = Adder >> 1;
  Hcount = 0;
  Lcount = 0;
  for (Words = 0; Words < NumberWords; Words ++)//test all data
  {
   Data = DataFile[Words];
   if(Data > Match) {Hcount ++;}
   else if(Data < Match) {Lcount ++;}
  }
  //test bit value
  if      (Hcount == Lcount)  {break;}
  else if (Hcount > Lcount)   {Match = Match + Adder;}
  else                        {Match = Match - Adder;}
 }
 if ((match == 1) && (Lcount > HalfWords)) {Match = 0;}
 return Match;
}
```