Mosaic Industries

Median Window Smoothing

APPLICATION NOTE
MI-AN-018

Summary

The following software shows how to find the median value of a bunch of data points.

```
\ ********* MEDIAN WINDOW SMOOTHING *****************
\ Copyright 1994, Mosaic Industries Inc.  All rights reserved.
\
\ We have found this median smoothing to be a mathematically robust, noise-
\ immune and well-behaved way to find the median value of a collection of data
\ points. The longer the buffer size you select, the stronger the filtering and
\ the longer the \ delay in response to changing inputs. This algorithm can be
\ implemented in fixed point form for faster
\ computation. In addition, you may decide to skip the secondary filtering
\ associated with the 2ND.BUFFER; this speeds up the algorithm a bit at the
\ expense of some more \ noise.
\ To use: call DIM.MEDIAN.BUFFERS to initialize the buffers, then call SMOOTH
\ when each data point is taken.  SMOOTH returns the most recent median value.

DECIMAL

13 CONSTANT WINDOW.SIZE       \ buffer size; MUST BE ODD!
\ startup delay >= size/2 sample times
MATRIX: MEDIAN.BUFFER         \ row buffer, holds sorted samples

WINDOW.SIZE 2/ CONSTANT MEDIAN.INDEX        \ median is in middle spot in buf

MATRIX: 2ND.BUFFER            \ row buffer for secondary averaging of median
WINDOW.SIZE 2/ CONSTANT 2ND.BUFFER.SIZE
VARIABLE 2ND.BUFFER.POINTER   \ column index

: INIT.MEDIAN.FILTER    ( -- )
      ' MEDIAN.BUFFER ZERO.MATRIX
      ' 2ND.BUFFER ZERO.MATRIX
      2ND.BUFFER.POINTER OFF
      ;

: DIM.MEDIAN.BUFFERS ( -- )
      \ dims and inits all to zeros at startup
      1 WINDOW.SIZE ' MEDIAN.BUFFER DIMMED
      \ Initialize a secondary buffer of size N/2.  This buffer maintains the
      \ last N/2 medians found so that they can be averaged for output.
      1 2ND.BUFFER.SIZE ' 2ND.BUFFER DIMMED
      INIT.MEDIAN.FILTER
      ;
```

```
: FIND.INSERTION.POINT ( r -- n )
\ the input r is the new floating point value to be inserted in the buffer and
\ the output n is the index where it should be inserted.  Finds the location to
\ insert a number, r, into the buffer so that the buffer values are always in
\ increasing order.
      WINDOW.SIZE LOCALS{ &Insertion.Point }    ( r -- )
      WINDOW.SIZE 0
      DO    FDUP                        \ duplicate the number
            0 I MEDIAN.BUFFER F@ F<     \ compare it to each buffer entry in turn
            IF    I TO &Insertion.Point \ and leave the loop
                  LEAVE
            ENDIF
      LOOP
      FDROP
      &Insertion.Point
      ;
: INSERT ( r \ n -- )
\ r is the number to be inserted; n is the Insertion.Point  If n <=
\ WINDOW.SIZE/2 inserts r into the buffer at n after shifting elements at
\ position n and greater up to greater indices. If n > WINDOW.SIZE/2 inserts r
\ into the buffer at n-1 after shifting elements at position n-1 and less down
\ to lesser indices.
      LOCALS{ &Insertion.Point }                              ( r -- )
      &Insertion.Point MEDIAN.INDEX <=
      IF    &Insertion.Point WINDOW.SIZE 2-
            DO    0 I MEDIAN.BUFFER F@          \ shift to the right
                  0 I 1+ MEDIAN.BUFFER F!
            -1 +LOOP
            0 &Insertion.Point MEDIAN.BUFFER F! \ emplace new number
      ELSE
            &Insertion.Point 1- 0
            DO    0 I 1+ MEDIAN.BUFFER F@       \ shift to the left
                  0 I MEDIAN.BUFFER F!
            LOOP
            0 &Insertion.Point 1- MEDIAN.BUFFER F!    \ emplace new number
      ENDIF
      ;
```

```
: SMOOTH ( r1 -- r2 )
\ Inserts the input r1 into the Buffer & fetches the Buffer's middle value,
\ an estimate of the running median, for insertion into the 2nd.Buffer,
\ and averages all values of the 2nd.Buffer to yield the output r2.
            FDUP FIND.INSERTION.POINT          ( r\n -- )
            INSERT                    ( -- )       \ Insert new number into buffer
\ Fetch the center number of the buffer and place it into the second.buffer
            0 MEDIAN.INDEX MEDIAN.BUFFER F@     ( estimated.median -- )
\ we could stop here if pressed for time, but we add another layer of
\ filtering:
            0 2ND.BUFFER.POINTER @ 2ND.BUFFER F!
            2ND.BUFFER.POINTER @ 1+ 2ND.BUFFER.SIZE MOD
            2ND.BUFFER.POINTER !    \ update pointer to circular 2nd.buffer
            ' 2ND.BUFFER MATRIX.SUM
            2ND.BUFFER.SIZE FLOT  F/            ( averaged.median -- )
      ;
```


Mosaic Industries

5437 Central Ave Suite 1, Newark, CA 94560 Telephone: (510) 790-8222 Fax: (510) 790-0925