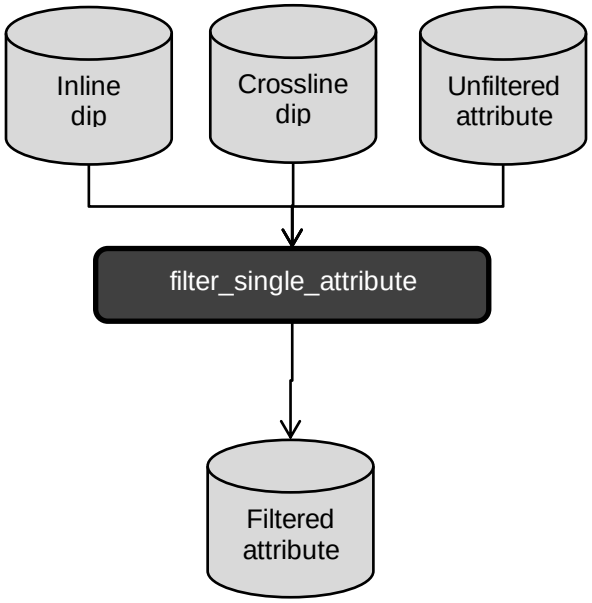
Filtering a single attribute – Program filter_single_attribute

Computation flow chart

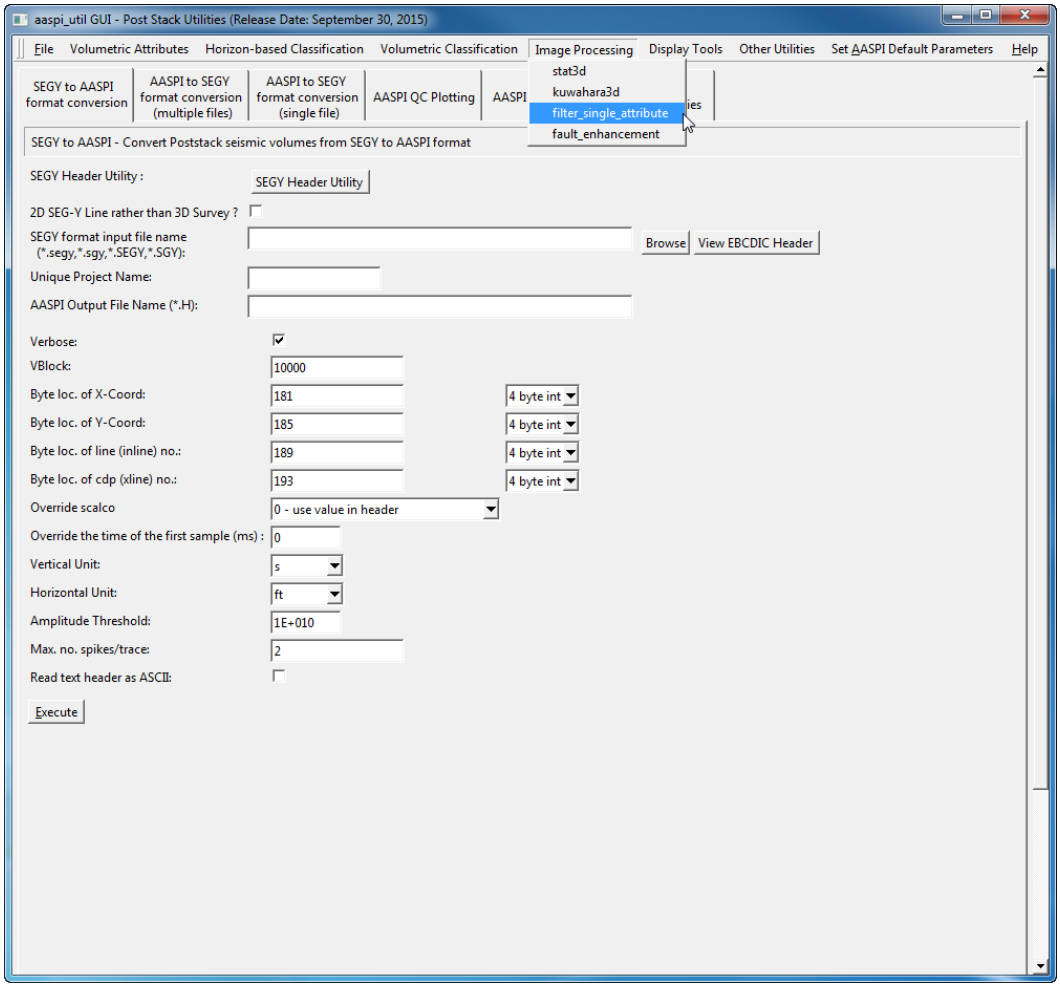
The input to program **filter_single_attribute** includes unfiltered input attribute as well as estimates of the inline and crossline dip components. Program **filter_single_attribute** can be run iteratively, whereby the output can be used as input for the next iteration.



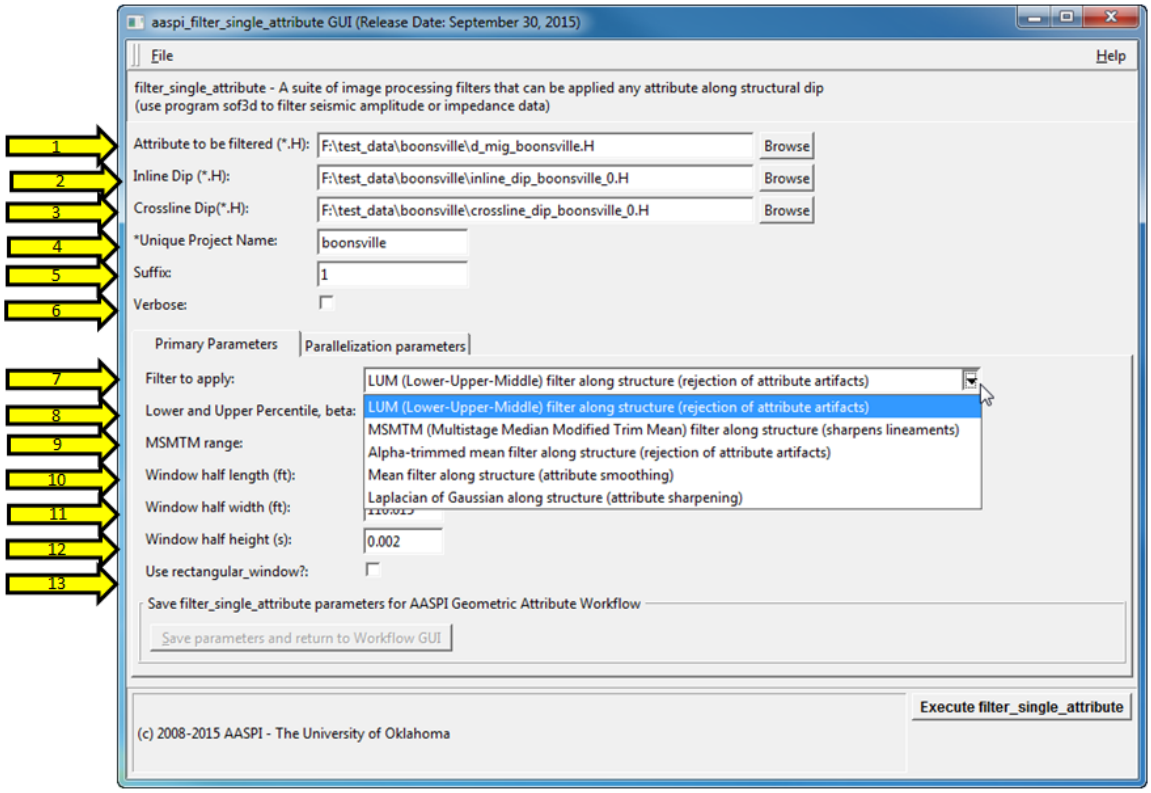
Computing mean, median, and other filtered dip volumes

In the **aaspi_util** menu, find the Image Processing tab andchoose program **filter_single_attribute.**

Image Processing: Filtering a single attribute – Program **filter_single_attribute**



The following window appears:



filter_single_attribute has three input files: (1) The attribute to be filtered (2) inline and (3) crossline components of dip. There is one output file – and the filtered. Like most AASPI programs the algorithm runs in parallel. In this example I've set the *Suffix* to be '1' indicating that this is the first pass of filtering. The possible filters include LUM (lower-upper-middle), MSMTM (multistage median-based modified trimmed mean), *median* and *mean* filters. Al-Dossary and Marfurt (2007) show the applicability of LUM and MSMTM filters.

Among the parameters, (7) is the filter we would like to apply, in this case the LUM filter. The default window size consists of the neighboring traces and samples, in this case +/-110 ft and +/- 0.002 s.

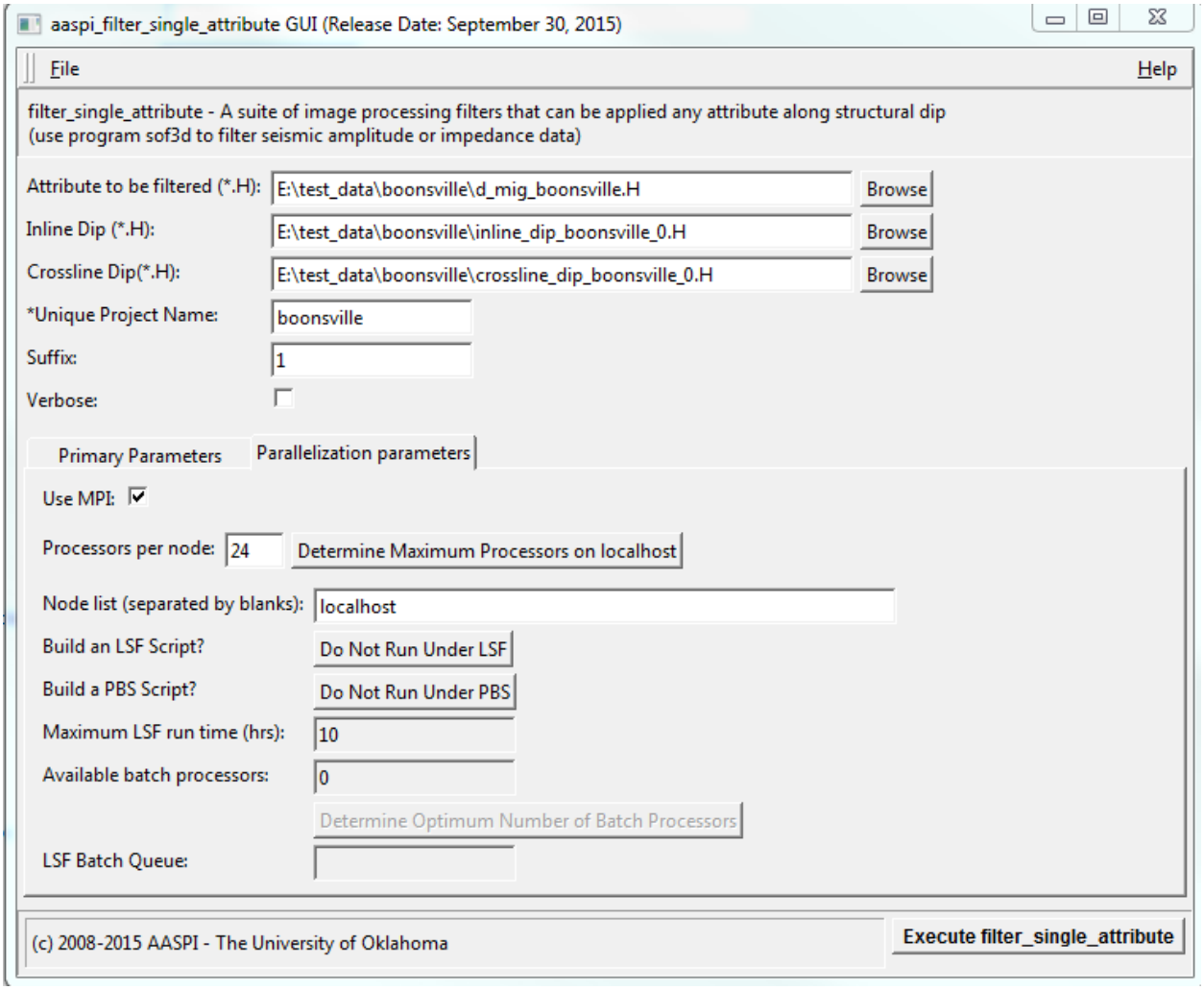
If you have selected the LUM filter, then the (8) *beta* value becomes active. If we set *beta* to be 50%, the result will be the same as using the median filter. In contrast, if we set it to 0%, the result will be as if we had not filtered the data. If we set *beta* to be 20%, then values which fall between 20 – 80

% of the confidence estimate will be kept. Values that fall below 20% of the confidence estimate will be set to the lower threshold 20% confidence value, and values that fall above 80% of the confidence estimate will be set to the upper threshold 80% confidence value. In this manner values that fall below our lower threshold and above our upper threshold will be clipped.

The MSMTM (Multistage median-based modified trimmed mean) filter is able to preserve detail, meaning it acts as a lineament preserving filter that can smooth noise. The MSMTM is a modified trimmed mean (MTM) filter that implements a multistage median filter (MSM). A data sample's value is kept if it lies in the range of [$m - q$, $m + q$], where m is calculated using an MSM filter and q is a user defined range. Larger values of q result in some smearing of lineaments through higher amplitude "noise" areas, while smaller values of q better preserve narrow lineaments. For further discussion, please refer to Al-Dossary and Marfurt (2007).

The *Parallelization parameters* panel only asks for the list of nodes and the number of processors per node:

Image Processing: Filtering a single attribute – Program **filter_single_attribute**



Like all AASPI codes, click *Execute* and intermediate information will be printed in the xterm from which aaspi_util was launched:

Image Processing: Filtering a single attribute – Program **filter_single_attribute**

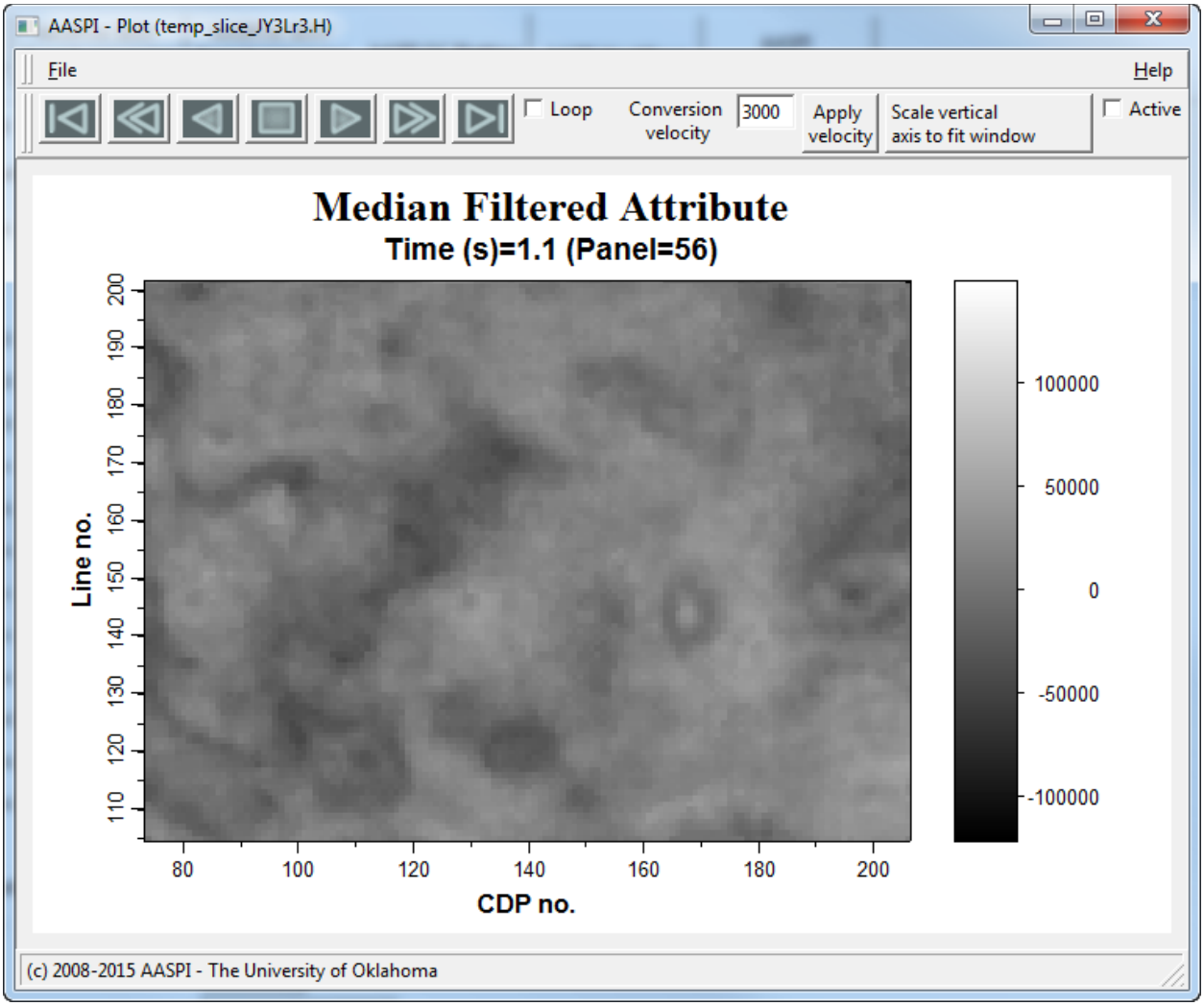
```
process                                    task       time (hr)  time/trace (s
)
 20:                              read  data           0.000           0.000
         0 : memory deallocated residing only on master deallocated
         0 : shared arrays residing on both master and slave deallocated
  1:                              read  data           0.000           0.000
         9 number of traces processed:          582
process                                    task       time (hr)  time/trace (s
)
        10 :end loop over lines
        10 number of traces processed:          582
process                                    task       time (hr)  time/trace (s
)
 12:                              read  data           0.000           0.000
 16:                              read  data           0.000           0.000
 16:                         send data via MPI          0.000           0.000
 16:                      receive data via MPI          0.000           0.000
 16:                      send results via MPI          0.000           0.000
 16:                   receive results via MPI          0.000           0.000
 16:                      calculate attributes          0.000           0.000
 16:                     write results to disk          0.000           0.000
 17:                              read  data           0.000           0.000
 17:                         send data via MPI          0.000           0.000
 17:                      receive data via MPI          0.000           0.000
 17:                      send results via MPI          0.000           0.000
 17:                   receive results via MPI          0.000           0.000
 17:                      calculate attributes          0.000           0.000
 17:                     write results to disk          0.000           0.000
 17:                              total time           0.001           0.004
         17 : memory residing only on slaves deallocated
         17 : shared arrays residing on both master and slave deallocated
 20:                         send data via MPI          0.000           0.000
 20:                      receive data via MPI          0.000           0.000
 20:                      send results via MPI          0.000           0.000
 20:                   receive results via MPI          0.000           0.000
 20:                      calculate attributes          0.000           0.000
 20:                     write results to disk          0.000           0.000
 20:                              total time           0.001           0.005
         20 : memory residing only on slaves deallocated
         20 : shared arrays residing on both master and slave deallocated
 21:                              read  data           0.000           0.000
 21:                         send data via MPI          0.000           0.000
 21:                      receive data via MPI          0.000           0.000
 21:                      send results via MPI          0.000           0.000
 21:                   receive results via MPI          0.000           0.000
 21:                      calculate attributes          0.000           0.000
 21:                     write results to disk          0.000           0.000
 21:                              total time           0.001           0.005
         21 : memory residing only on slaves deallocated
         21 : shared arrays residing on both master and slave deallocated
  1:                         send data via MPI          0.000           0.000
```

Once the job is completed, typing *ls –ltr* at the terminal prompt shows that the following files were created:

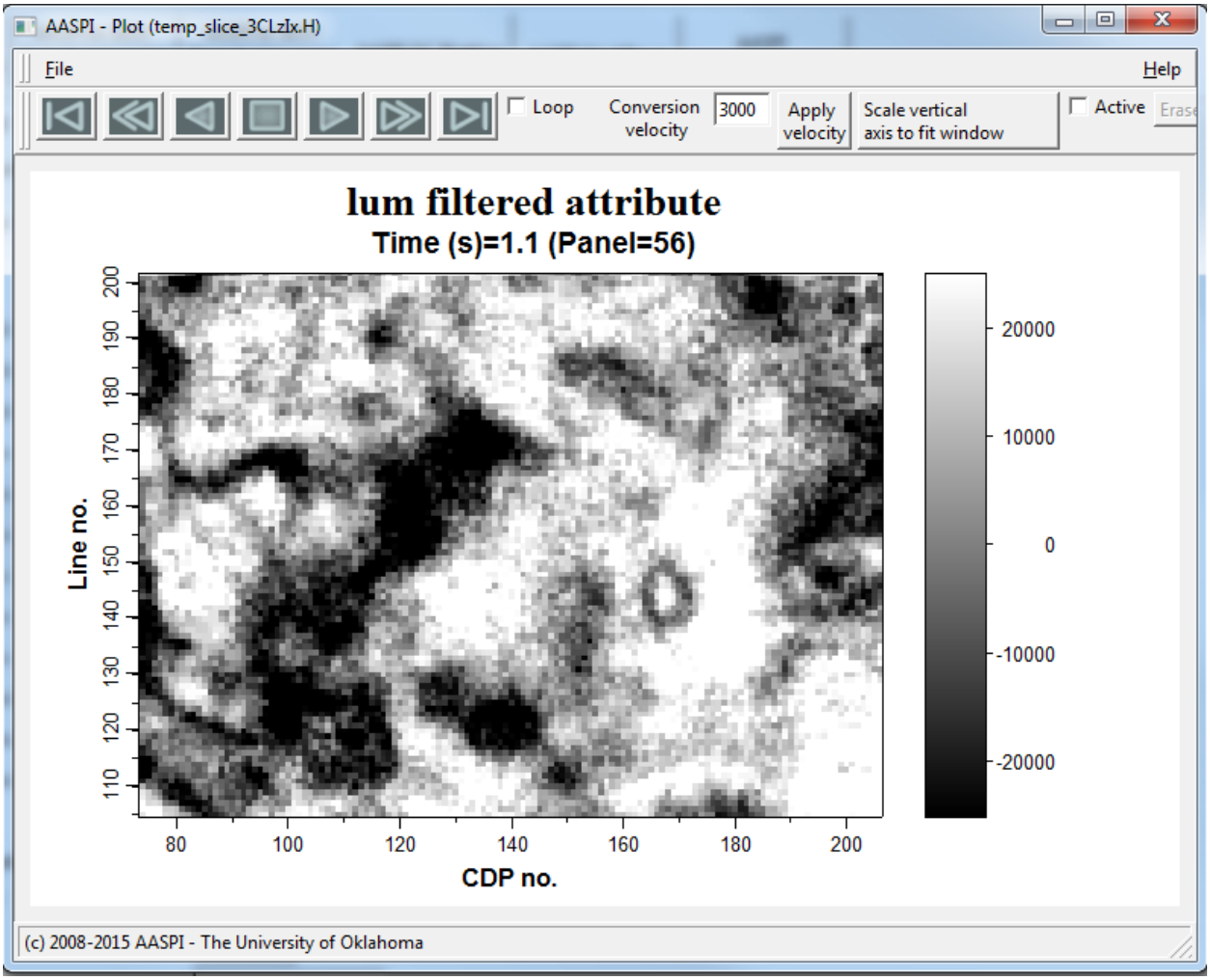
```
-rw-r--r--  1 kmarfurt aaspi       31 Aug  3 16:10 live_processor_list
-rw-r--r--  1 kmarfurt aaspi     1921 Aug  3 16:10 inline_dip_median_filt_boonsville_1.H@@
-rw-r--r--  1 kmarfurt aaspi     2987 Aug  3 16:10 inline_dip_median_filt_boonsville_1.H
-rw-r--r--  1 kmarfurt aaspi     1927 Aug  3 16:10 dip_magnitude_median_filt_boonsville_1.H@@
-rw-r--r--  1 kmarfurt aaspi     3023 Aug  3 16:10 dip_magnitude_median_filt_boonsville_1.H
-rw-r--r--  1 kmarfurt aaspi     1925 Aug  3 16:10 dip_azimuth_median_filt_boonsville_1.H@@
-rw-r--r--  1 kmarfurt aaspi     3040 Aug  3 16:10 dip_azimuth_median_filt_boonsville_1.H
-rw-r--r--  1 kmarfurt aaspi     1927 Aug  3 16:10 crossline_dip_median_filt_boonsville_1.H@@
-rw-r--r--  1 kmarfurt aaspi     3005 Aug  3 16:10 crossline_dip_median_filt_boonsville_1.H
-rw-r--r--  1 kmarfurt aaspi     1909 Aug  3 16:10 conf_median_filt_boonsville_1.H@@
-rw-r--r--  1 kmarfurt aaspi     2776 Aug  3 16:10 conf_median_filt_boonsville_1.H
-rw-r--r--  1 kmarfurt aaspi    22535 Aug  3 16:11 image_filt3d_boonsville_1.out
[kmarfurt@opal boonsville]$
```

Note that we have created filtered versions of the input attribute data. The part of the name *median_filt* denotes the kind of filter that was applied. Had we applied an LUM filter, we would see *lum_filt* instead.

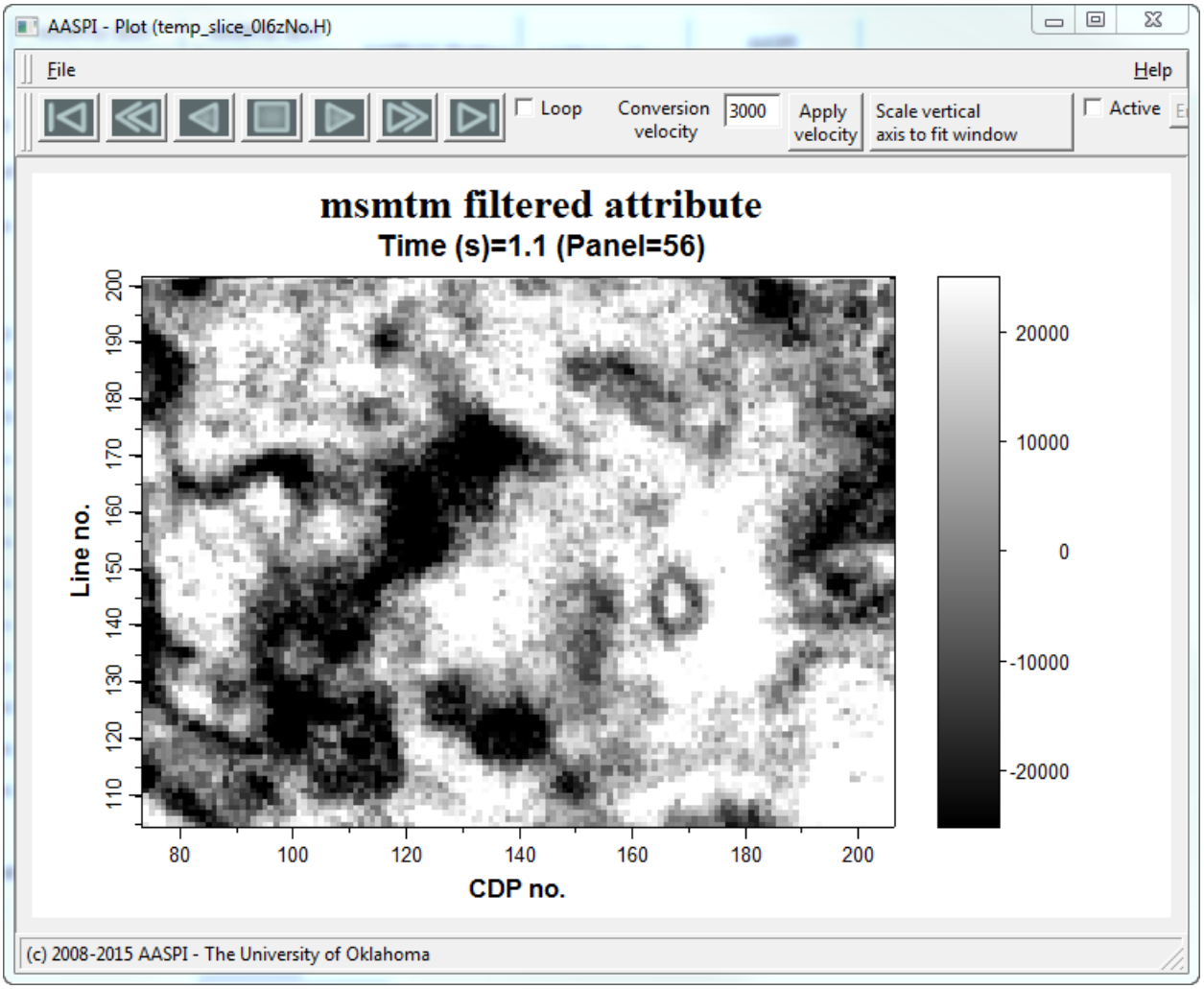
The results of the median filter look like this (time slice, t = 1.1 sec):



Here we see what the result of the LUM filter looks like (time slice, t = 1.1 sec):



And here is the result of the MSMTM filter with q = 4 (time slice, t = 1.1 sec)



We note that the median filtered image is overall is less noisy, smoother, with a little less N-S acquisition footprint. However, it also has lower resolution than the input image shown previously. In comparison to the median filter, the LUM filtered image shows more acquisition footprint, but it has enhanced the collapse features too. The MSMTM filter improves in regards to the footprint and shows better details near the collapse features.

References:

S. al-Dossary, K. J. Marfurt, 2007, Lineament-preserving filtering: Geophysics, **72**, P1-P8.