IRIS Technical Note 15
Pipeline Despiking Procedure

Stein V. H. Haugan, John Serafin, and Richard Shine, version 2013 June 17

1. Introduction

Identifying spikes/cosmic ray hits in a CCD exposure seems like such a trivial task - to the eye. While it may take some time to scan the image, you'll know one when you see one. Creating an algorithm that will do the same, however, is daunting. That is, detecting them is not so hard - just flag everything that sticks up above the local surroundings - but distinguishing spikes from real data features is the tricky part!

Optimal spike removal requires careful selection and tuning of the algorithm, based on the instrument's optics, detector characteristics and the nature of the data being observed. In most algorithms, each pixel is compared with its surroundings, and deemed a spike if it is brighter than the surroundings by a given amount (set by some sensitivity parameter(s)). Typically, the "brightness of the surroundings" is calculated either by averaging or by a median based approach. The intensity value for a pixel flagged as a spike is replaced using a median based approach.

2. Algorithms

Two despiking algorithms are under consideration for use in the low level IRIS data processing pipeline. One is based on the Solar Dynamics Observatory (SDO) Atmospheric Imaging Assembly (AIA) pipeline despiker and the other on the algorithm used for data from the Coronal Diagnostic Spectrometer (CDS) on the Solar and Heliospheric Observatory (SOHO).

2.1 TRACE and AIA Heritage Algorithm

The AIA despiking algorithm has its origin in despiker developed for Transition Region and Coronal Explorer (TRACE) images in 1998. One of its requirements was to be fast enough for interactive use on a 1998 era workstation.

Each pixel is compared with the mean of its 8 neighbors and is flagged as a spike if it exceeds the mean by both a fixed DN threshold and a relative threshold. The former is to reduce numerous hits from shot noise in dark and/or zero DN areas and is equal to 4.0 for AIA. The relative threshold is expressed as

(1+frac) where frac is 0.8 for AIA. A spike has to exceed the mean of the perimeter by (1+frac) and only the brightest flare kernels get inappropriately tagged. When a spike is found, it is replaced by the median of the 16 pixels 2 positions away. This perimeter is used rather than the neighboring perimeter used for flagging because some of those pixels have likely been brightened by the spike as well. The algorithm has a parameter to select any value from the sorted perimeter array, but AIA uses the 8th lowest of the 16. The flagging and replace by median process is repeated n times, where n is the number of iteration parameter to the despiker. n is 3 for the AIA EUV channels and 0 for the UV channels. That is, AIA UV images are not despiked. The AIA despiker keeps a list of the original intensities for pixels flagged as spikes, which it provides to the calling routine. This allows reversal of the despiking.

2.2 CDS and EIS Heritage Algorithm

This algorithm's uses a (moving) median-based approach for the detection step. It takes into account that the data from a spectrometer (e.g. CDS, EIS, and IRIS FUV/NUV) is anisotropic: even in the absence of spikes, all emission lines have a "bump" in the wavelength direction, whereas the data may in fact be constant in the direction along the slit.

Adaptation to the anisotropy is achieved by allowing the aspect ratio for the moving median box to be varied. Adaptation to data characteristics such as the spectral and spatial oversampling factors and the inherent "spikiness" of unpolluted data is achieved by adjusting the size of the box and three sensitivity/algorithmic parameters.

A pixel is flagged if:

pix > median * max_factor_hi (for pix >= limit)

or

pix > median + max_var_low (for pix < limit)

The reason for handling values above/below the limit in different manners is that for low-signal regions, the poisson noise (or rather the sum of poisson noise and readout noise) will be on the order of the signal itself. Thus true_signal + noise approaches median*max_factor_hi from above as the signal gets weaker. This gives a large number of false positives. The formula for low-signal regions instead puts a fixed limit on the height of the pixel above the local median.

Also, the nearest neighbors of these pixels are flagged - i.e. an isolated spike will result in a 3x3 crosshair set of pixels being flagged. This is because edges of spikes may be below the detection limits,

yet those pixel values might be tainted. It is possible to adjust this neighboring step by supplying a square kernel array where nonzero entries means that the corresponding neighbor pixel should be flagged. E.g. a 3x3 kernel with all entries set to 1 would cause a square box surrounding the "original" pixel to be flagged. Note that the pattern inside the square kernel need not be symmetric at all. Thus idiosyncracies of the detector (such as bleeding in one direction but not the other) can be taken into account if necessary.

It is also possible to specify multiple iterations of the neighboring step.

Filling in is also done using a (moving) median-based approach, with the same size for the median box as used during detection. The filling step is repeated until all flagged pixels have been filled (large blotches may result in an inability to calculate the median right away).

Detection parameters:

xbox & ybox :	X and Y size of the moving median box
max_var_low :	See flagging formulas above
max_factor_hi :	See flagging formulas above
limit :	Determines which flagging formula applies
neighbor :	No. of times to apply neighboring convolution
kernel :	The convolution kernel for flagging neighbors

Filling parameters:

xbox & ybox :	X/Y sizes of moving median box (same as above)

2.2 Other algorithms or potential improvements (in brief)

Leave out center pixel when calculating moving median (a spiked pixel might shift it).

Using percentiles for median-based detection scheme, i.e. introducing the percentile as another free parameter.

Poisson noise based detection, e.g. max_var_low = Nsigma * sqrt(average or median).

Adjusting parameters as a function of e.g. exposure time, wavelength, etc.

Including the time dimension: Considering subsequent observations of the same location could be used to reduce false positives (repeated spikes in the same location is extremely unlikely) by finding "spikes" that are inherent in the observation. This might again allow an increased detection sensitivity for the single-exposure step, reducing false negatives. Also, the algorithms might be rewritten to take a 3D cube of data as input, applying the algorithms on a 3D box instead of a 2D box. It is difficult to determine the previous and next image during the level 0 to level 1 pipeline processing, so it may become necessary to devise an out of pipeline procedure to obtain a list of spikes from quick look images and feed that list to definitive level 1 processing. It should also be noted that some short lived events can still trigger a few false positives, even with time based filtering.

A voting combination of two or more approaches may be applied. Either require both methods flagging (reduces false positives), or either method flagging (reduces false negatives).

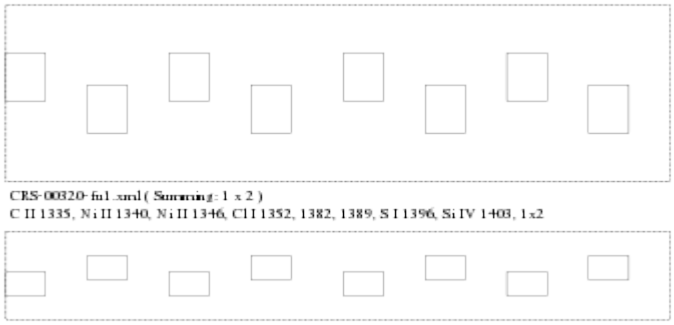
3. Implementation

The IRIS despiking algorithm is implemented by subroutines written in C and tailored to work with IRIS data in the existing JSOC pipeline infrastructure at Stanford. The IRIS level 1 pipline module supplies a level 0 image to be despiked along with some parameters and other arguments to the top level despiking routine sdc_despike(), which despikes the image in place and returns information regarding the results, including thenumber of spiked pixels, their addresses, old pixel values (digital numbers), and new pixel values. The arguments to sdc_despike() are defined as follows:

```
int sdc_despike(
  int *image,          /* To be cleaned */
  char *mask,          /* Ignore these - guaranteed unchanged */
  int nx, int ny,      /* Size (nx,ny) of image */
  int neighbour,       /* Number of neighbor iterations */
  char *kernel,        /* neighbor convolution. Generated if needed */
  int kernel_size,     /* Note: kernel is square */
  int xbox, int ybox, /* Size of median box for detection */
  int max_var_low,     /* Max variation above median in "low" areas */
  float max_factor_hi,/* Max factor above median in "high" areas */
  int limit,           /* Value separating high from low areas */
  int *badblobs,       /* Blobs of "bad pixels", convert to missing */
  int sizeofbads,      /* Number of pre-flagged bad pixels */
  int *nspikes,        /* Number of pixels flagged */
  int **oldvalues,     /* Old values of flagged pixels */
  int **spikelocs,     /* Their one-dimensional index location */
  int **newvalues)     /* And their *new* values */
```

`image` is a pointer to a signed 32 bit array of level 0 image pixel values to be despiked supplied by the

calling module of size `nx` by `ny`. The address of an image pixel is given by `colnum + nx*rownum` where `colnum` increases with wavelength for FUV and NUV images and nominal Solar West for SlitJaw images. `rownum` increases with nominal Solar North for all images. Note that level 0 images have some of these directions reversed for some image types, but that the spectral dimension is always along a “row” and the nominal Solar North-South axis is always along a “column”. The level 0 image will be flipped accordingly during level 0 to level 1 processing as shown in the figure.



`mask` is a pointer to a character array with the same number of elements as `image` and the same pixel address interpretation. Pixels with a `mask` value of 0 at the same address were not read from the CCD and should not be processed. These pixels should have a value `0x80000000` which also indicates that they should not be processed. If `mask` is a `NULL` pointer, it is not used. That is, `mask` is an optional parameter.

`neighbour` is an interger indicating the number of neighbor iterations to be performed, possibly 0.

`kernel` is a pointer to a character array of size `kernel_size` defining the convolution kernel. If it is `NULL`, a default kernel is generated internally.

Integers `xbox` and `ybox` give the size of the median box used to detect spikes.

`badblobs` is a pointer to a signed 32 bit array of permanently bad pixel addresses supplied by the calling routine. These addresses are of the same form `colnum + nx*rownum` as above. The integer argument `sizeofbads` is the number of addresses supplied (may be 0).

The output argument `nspikes` is a pointer to an integer which contains the number of image pixels affected by spiking.

The output arguments `oldvalues`, `spikelocs`, and `newvalues` are all pointers to pointer to integers of dimension `nspikes`. The addresses for detected spikes of the same form `colnum + nx*rownum` are in `spikelocs`, the input spiked values are in `oldvalues`, and the despiked values are in `newvalues`. Saving both the old values and new values allows the despiking to be done in place and provides the flexibility to easily add the option of the final image array having pre despiking or post

despiking values. The memory for all these arrays is allocated by sdc_despike(), it is the responsibility of the calling program to free that memory.

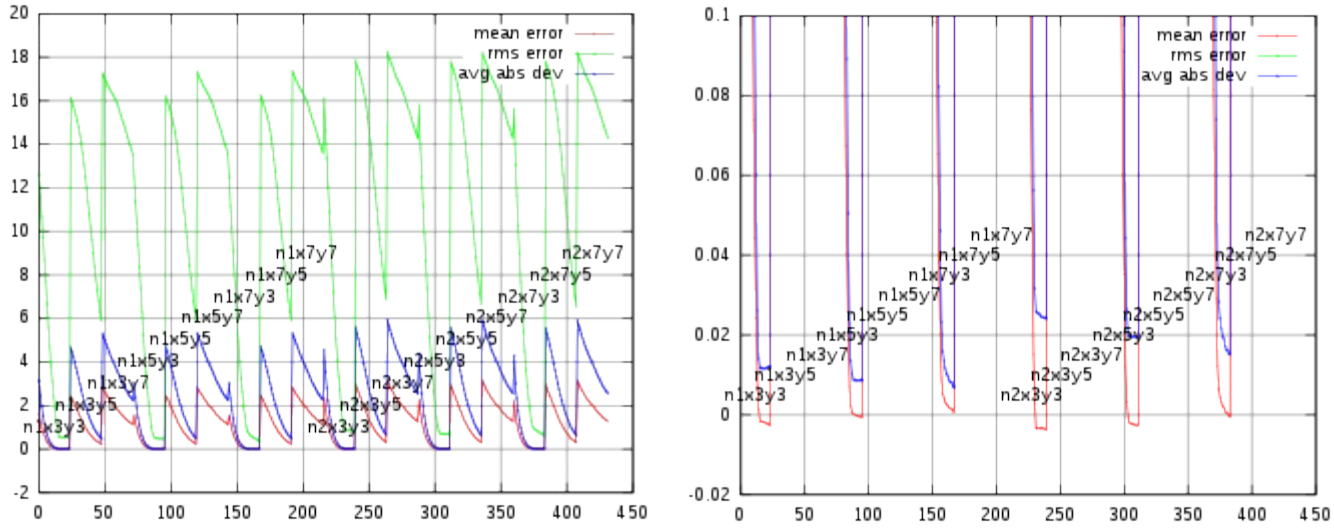
4. Testing/Tuning

A driver has been written to test and tune sdc_despike(), which can read any two dimensional FITS image with 16 bit pixels. Input and output file names and all tuning parameters may all be specified when the program is invoked. The driver/despiker combination was run for an SDO/AIA and a Hinode/SOT FG image with reasonable results, but those images are only comparable to IRIS SlitJaw images, not FUV or NUV images. The data in Hinode EIS images are at least similar to IRIS FUV and NUV images, but level 0 EIS images are more complicated than the test driver can handle. Some utility routines were written to extract the windows from a level 0 EIS file, split each window into 16x256 pixel pieces and assemble all the pieces from all the windows into a single 1024x1024 pixel image. The despiker also worked for this image, but the huge number of spikes precluded quantitative tuning until resources are available create a catalog of spikes for that image or locate an image with all spikes cataloged.

To explore tuning the despiking parameters, Sarah Jaeggli's cr_sim.pro was used to generate 999 cosmic rays saved to a 1096x4144 FITS file. This image of simulated cosmic rays was added to five IRIS deuterium lamp images from June 2012 with FSNs 25677, 25678, 25679, 25680, and 25685. Several scripts were written to step though range of a number of tuning parameters for the despiker, such as number of neighbour iterations, xbox, ybox, limit, max_var_low, and max_factor_hi. These scripts can be browsed at http://www.lmsal.com/~jps/IRIS/Despike/scripts/tuning/

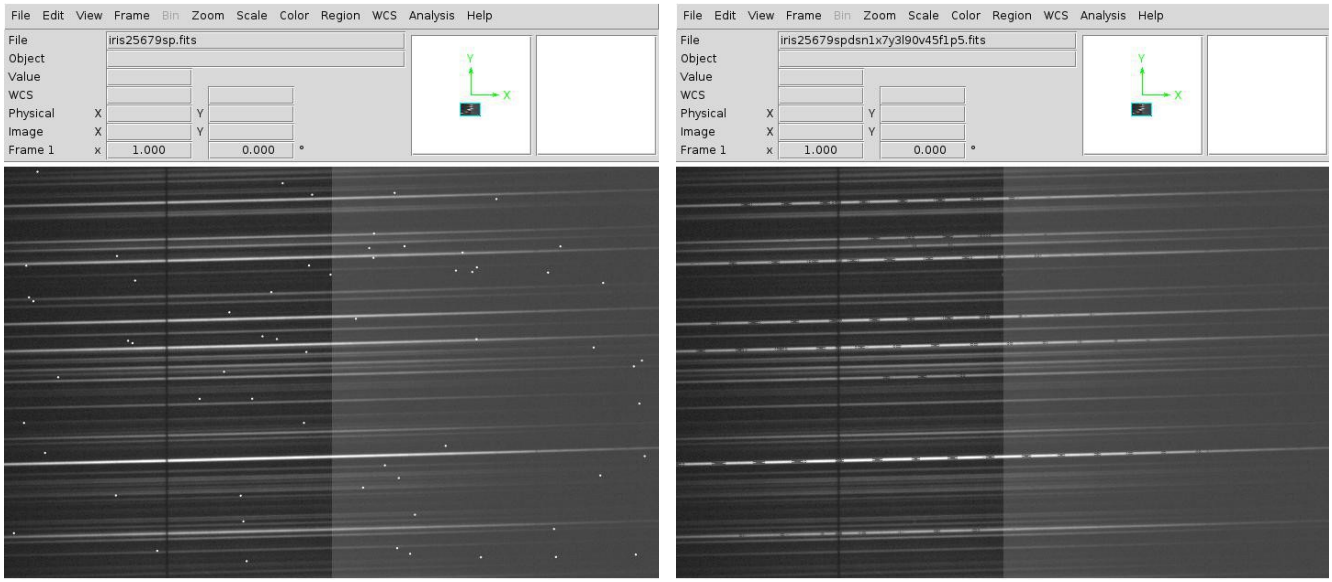
IRIS images generated during ground testing have pedestal (0 signal) values of several hundred. This drives down the ratio of a spiked pixel to that of its neighbors. Test/reference images were prepared by subtracting the minimum pixel value from IRIS FSN 25679, saving that as iris25679.fits, adding the simulated cosmic rays from cr_sim_999.fits, and saving that as iris25679sp.fits, all in http://www.lmsal.com/~jps/IRIS/Despike/iris000025679/. The despiker was tested against iris25679sp.fits for max_factor_hi ranging from 1.05 to 2.2 in steps for 0.05 for xbox 3 to 7, ybox 3 to 7, and neighbour 1 to 2. The despiked image was stored with despiking parameters as part of its file name, e.g. iris25679spdsn1x7y3l90v45f1p9.fits. For each set of parameters, three difference images are computed, one between the input image and the despiked image, another between the despiked image and the original (unspiked) image, and the last between the first difference image and the simulated spikes image. These are the computed spikes, despiking error, and spikes error images respectively. A number of statistics are computed for each error image, mean, population rms, median, minimum, maximum, average absolute deviation, and rms. (These are the statistics returned by the perldl stats

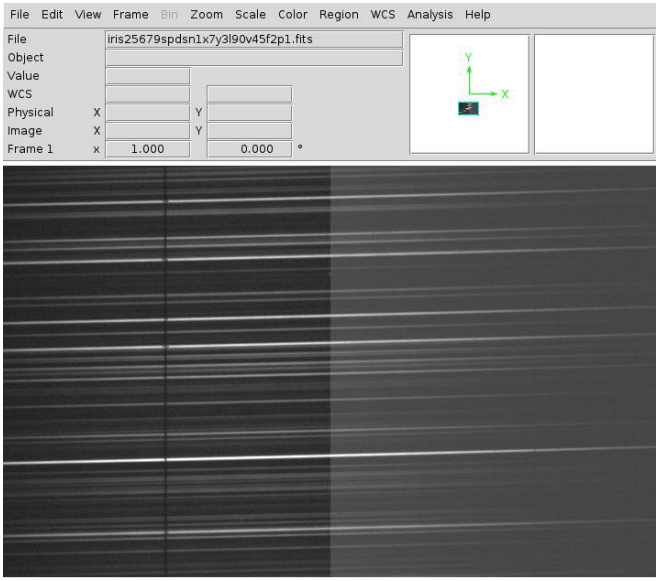
function.) A plot for the mean, rms, and absolute deviation of the computed spikes error image is shown below (full, and vertically zoomed).

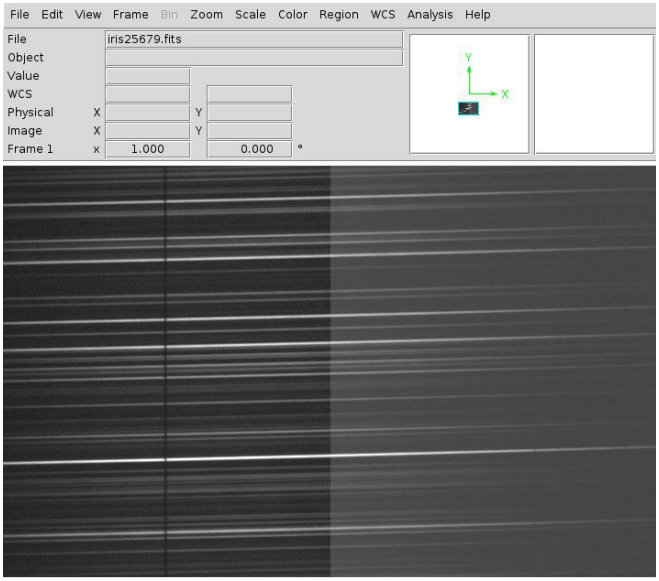


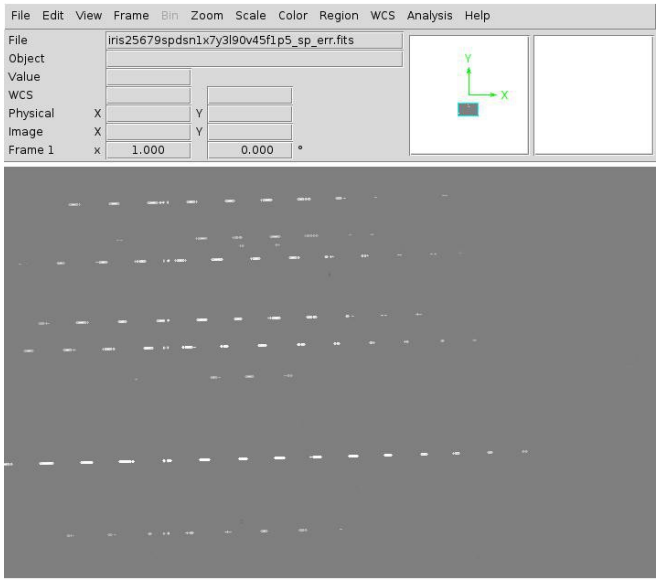
The lowest rms error and lowest average deviation error are for neighbor: 1, xbox: 7, ybox:3, and max_factor_hi: 2.2. It had been expected that the errors would start increasing beyond some value of max_factor_hi as had happened for similar runs where the pedestal had not been subtracted. Presumably a factor higher than 2.2 is required for this to happen.

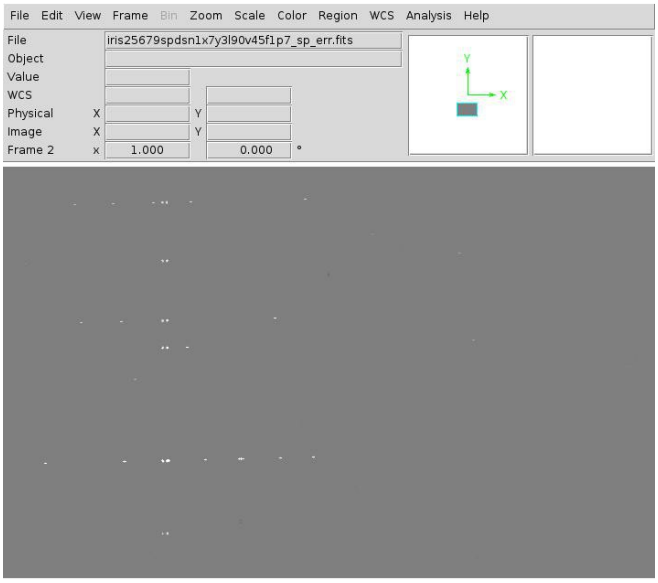
Cutouts from selected images are shown below.

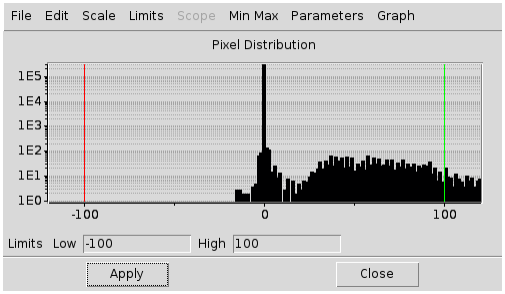


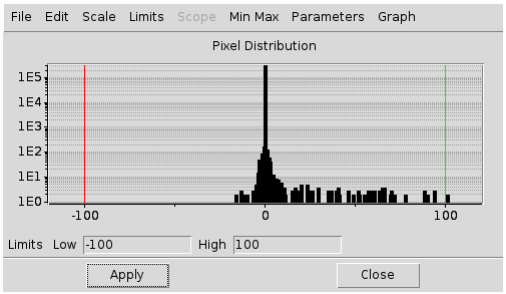

File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File iris25679spdsn1x7y3l90v45f2p1.fits
Object
Value
WCS
Physical X Y
Image X Y
Frame 1 x 1.000 0.000 °

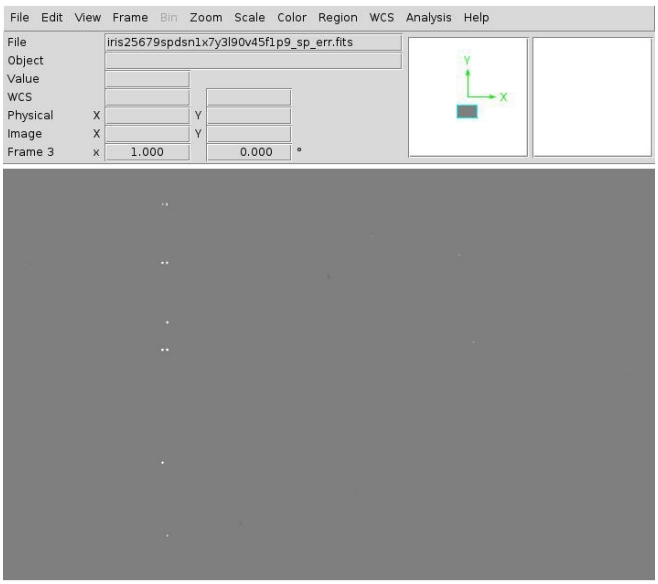

File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File iris25679.fits
Object
Value
WCS
Physical X Y
Image X Y
Frame 1 x 1.000 0.000 °

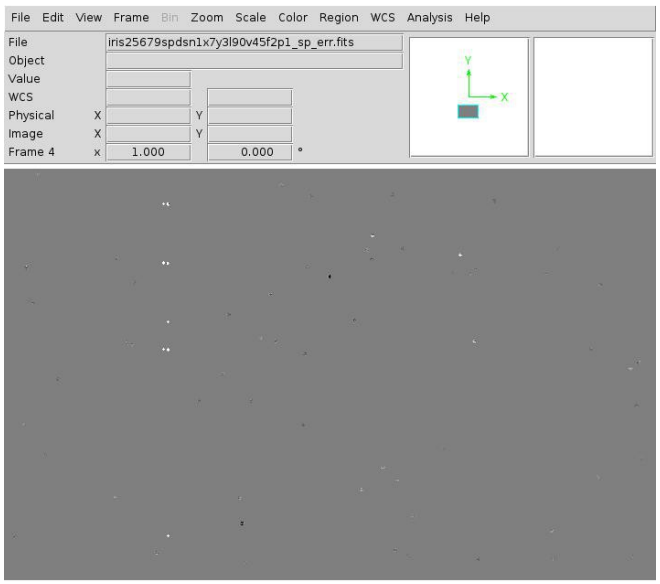

File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File iris25679spdsn1x7y3l90v45f1p5_sp_err.fits
Object
Value
WCS
Physical X Y
Image X Y
Frame 1 x 1.000 0.000 °

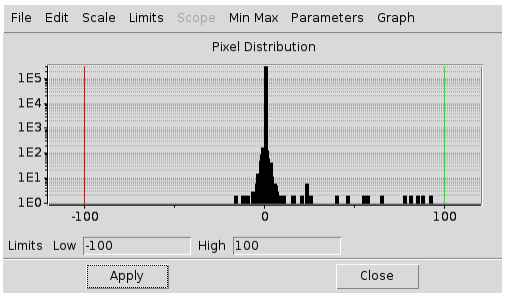

File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File iris25679spdsn1x7y3l90v45f1p7_sp_err.fits
Object
Value
WCS
Physical X Y
Image X Y
Frame 2 x 1.000 0.000 °

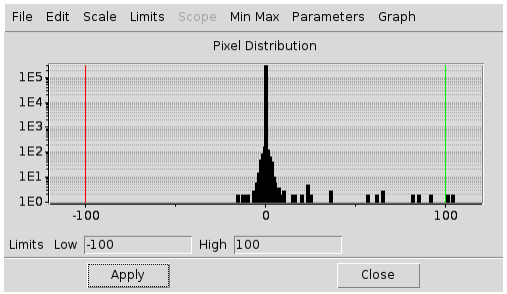

File Edit Scale Limits Scope Min Max Parameters Graph
Pixel Distribution
1E5
1E4
1E3
1E2
1E1
1E0
-100
0
100
Limits Low -100 High 100
Apply
Close

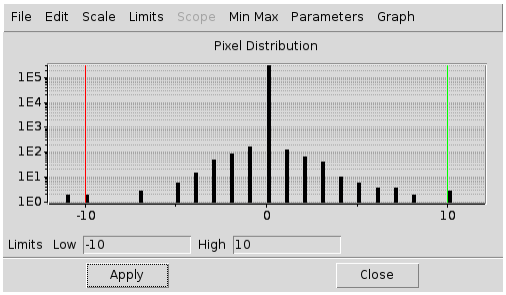

File Edit Scale Limits Scope Min Max Parameters Graph
Pixel Distribution
1E5
1E4
1E3
1E2
1E1
1E0
-100
0
100
Limits Low -100 High 100
Apply
Close











It can be seen that for low values of max_factor_hi, that many spectral lines are flagged as spikes, but that this is greatly reduced as max_factor_hi is increased. However, even at 2.2, a few spectral lines are flagged as spikes where they cross the slot. The region near the slot could be masked against despiking, or perhaps better, temporal filtering could be used to guard against false spikes that persist for more than one image.

5. Further Tuning

The ideal tuning would be to match the algorithms against "detection by eye". A representative collection

of manually processed images should be taken as the truth, and differences between the algorithm outputs should be used as a comparison. Note that there are *two* merit functions that should be considered: False negatives and false positives!

This will be complicated because some spikes and false spikes do not have enough contrast to be noticed, but are missed or falsely flagged as spikes.

No attempt so far has been made to tune the computation of the replacement value for flagged spikes, but the current algorithm seems to be working very well, as shown by the histograms for the spikes error images. Testing with a wider variety of images might reveal some possibilities for improvement.

Despiking has not yet been tested for Slit Jaw images, but the despiker should have an easier time with those, provided a square, and perhaps smaller, kernel is used.

6. Conclusion

Both despiking algorithms are still under consideration. It may be that the TRACE heritage algorithm will be used for Slit Jaw images and the CDS heritage algorithm used for spectral images. As implemented for the IRIS level 1 module, only some the tuning parameters and the convolution kernel are different. The input, output, and mask parameters are the same. So both algorithms may be built into the module and module logic or module parameters would select which algorithm is used for a particular image.