The WFSS Overlap Tool

This document describes the tool for evaluating the possible overlap of spectra between sources in the NIRISS WFSS mode. This is useful if one has one or more specific sources in a particular observation for which it is important to avoid spectral overlap as much as possible. The tool allows the user to examine the scene at different orientations and see which orient ranges produce overlap of the spectra and which do not. It needed these limits can then be used as constraints in the APT file for the program.

The tool is written in standard Python and requires a set of data files that describe the point source response of the instrument in normal imaging and in WFSS dispersed images. These files are available separately from the code.

Obtaining the Code and the Reference Files

The code for the tool is stored on github at the link

https://github.com/KevinVolkSTScI/wfss_overlap

which should be publicly available to all users. The reference files needed for the code are stored on Box at

https://stsci.app.box.com/folder/140705149002?s=ih166mw8ded6w0qeiw7znc7xuth7n531

which also should be available to anyone for downloading the files.

The contents of the github repository are:

fits_image_display.py	the code that handles the FITS image display
general_utilities.py	various utility routines
mpfit.py	a non-linear fitting routine
mpfitexpr.py	a wrapper for the mpfit.py
scene_image.py	the code for creation of the scene image
wfss_overlap_tool.py	the main program
wfss_scene.py	the code for making the WFSS scene images

plus the readme file and the license file. The two files mpfit.py and mpfitexpr.py were taken from https://github.com/segasai/astrolibpy/blob/master/mpfit/mpfit.py and were written by Sergey Koposov and Mark Rivers from the IDL code by Craig Markwardt. This code is used because it is more robust than the numpy/scipy equivalents. The contents of the Box folder are

f090w_gr150c_psfimage.fits
f090w_gr150r_psfimage.fits
f115w_gr150c_psfimage.fits
f115w_gr150r_psfimage.fits
f140m_gr150c_psfimage.fits

f140m_gr150r_psfimage.fits
f150w_gr150c_psfimage.fits
f150w_gr150r_psfimage.fits
f158m_gr150c_psfimage.fits
f158m_gr150r_psfimage.fits
f200w_gr150c_psfimage.fits
f200w_gr150r_psfimage.fits
niriss_NIS_x1024_y1024_f090w_predicted_0_0p00_0p00.fits
niriss_NIS_x1024_y1024_f115w_predicted_0_0p00_0p00.fits
niriss_NIS_x1024_y1024_f140m_predicted_0_0p00_0p00.fits
niriss_NIS_x1024_y1024_f150w_predicted_0_0p00_0p00.fits
niriss_NIS_x1024_y1024_f158m_predicted_0_0p00_0p00.fits
niriss_NIS_x1024_y1024_f200w_predicted_0_0p00_0p00.fits
occulting_spots_mask.fits
wfss_overlap.docx
wfss_overlap.pdf
wfss_overlap.pptx

The files in the Box folder are first the WFSS "point spread function" (PSF) FITS files (f090w_gr150c_psfimage.fits, for example), then the imaging PSF fits files (for example, niriss_NIS_x1024_y1024_f090w_predicted_0_0p00_0p00.fits), a mask file for the occulting spots in the NIRISS field of view, and documentation files.

Code Requirements

The code is written in python. It mostly uses standard Python packages (os, math, sys, tkinter, matplotlib, numpy, and scipy) and it uses the astropy package for the FITS file input/output. The astropy package can be installed with the command "pip install astropy" or, if one is using conda, by "conda install astropy". The astropy documentation is found at at https://docs.astropy.org/en/stable/index.html . The other "non-standard" package require is the pysaif package, which is used to map from sky positions to NIRISS detector pixel positions. The pysiaf package is found at https://github.com/spacetelescope/pysiaf . The documentation for the package is found at https://pysiaf.readthedocs.io/en/latest/ . Installation instructions are found in the "Read the Docs" pages. Note that the current verson of pysiaf cannot be installed using "pip", if one uses the command "pip install pysiaf" one gets an obsolete version.

The program is intended to be run from the command line and hence is not arranged as a package. If one is running the code from some directory other than where the source files are located, the directory where the code is stored should be in ones $PYTHONPATH variable.

The code is known to work in Python versions later than 3.7 on either a MacOS system or a Linux system. No testing has been carried out on a Microsoft Windows system.

Code Algorithm

The code carries out a sequence of steps to produce either a simulated direct image or a simulated dispersed NIRISS WFSS image for a given sky pointing and orientation. The general steps used are as follows:

(1) An oversized scene image is created from a star list and an extended source list, or possibly read in from a FITS file. The scene image is made at infinite angular resolution, meaning that each star occupies a single pixel and any extended sources that are defined follow perfect Sersic profiles at the NIRISS pixel resolution. This image is made assuming that the pixels are square and have a uniform angular size projected on the sky of 0.0656″, this being the pre-launch estimated ideal NIRISS pixel size. An optional uniform sky background can be added to the scene image. Details of how the image is generated are given below. The scene image is assumed to be in units of ADU/s equivalent to the JWST pipeline "rate" images. As it is on a uniform pixel grid, the image corresponds to a resampled image from the pipeline. The units of the image are unimportant for the purpose of the code. The scene image is specific to one of the NIRISS WFSS blocking filters.
(2) For any desired sky orientation, the standard scene image is rotated with the scipy.ndimage.rotate function. The rotation angle is measured counter-clockwise from the y axis of the scene image, which is assumed to follow a line of constant right ascension on the sky with the +y direction being north. Note that one could run into issues with this if the scene is right at the north or south poles on the sky. The pixels are assumed to be in a tangent plane projection centred at some specific sky coordinates (α_0, δ_0).
(3) The rotated scene image is then convolved with a PSF image suitable to the configuration requested. The PSF images are normalized to a total signal of 1. The output image is then displayed for inspection. The image display code allows the user to carry out a limited sub-set of the Image Reduction and Analysis Facility (IRAF) display and imexam functions.

The fidelity of the output image depends on several factors. One issue with the code is that the scipy.ndimage.rotate function does not work as well as the IRAF routine. The code appears to be intended for extended object image rotation and not for astronomical image rotation. The rotation call is known to produce low-level artefacts in the output image including negative values where the input image has only positive values. The code masks negative values to zero in the rotated images, but other low-level noise is introduced in the rotation that cannot be removed.

The main limitation of the code is that the dispersed images show exactly the same spectral properties at all positions within the field of view, whereas the ground testing of the NIRISS grisms shows that there is some variation in detail in the curvature of the spectral orders at different positions. This is a result of the convolution, and there is no way to correct for this. The expectation is that changes in the order curvature are small enough that the tool gives a reasonable estimate of the angles to avoid even if the details are not entirely correct.

Finally, the code assumes that the detector area for NIRISS is centred in the pick-off mirror (POM) field of view and that the detector edges are aligned with the POM edges. It is known that the actual situation is somewhat different than this, with the centre of the imaging

area being offset from the centre of the POM and with a small rotation of the imaging area with respect to the POM edges. These simplifications make the calculations much easier to carry out.

Required Inputs for the Code

The code requires either a pre-computed scene image of a particular type or source information that can be used to create the scene image. The normal expectation is that one needs to provide source lists and have the code produce the scene image. The following discusses that situation. The use of a scene image as input is described on page 10 below.

The inputs to the code are assumed to be the same as what is used with the "Multi-Instrument Ramp Generator" code (Mirage) produced at Space Telescope Science Institute. That code can be found at https://github.com/spacetelescope/mirage . The code here is limited in the scene generation compared to the Mirage code (e.g. moving objects cannot be simulated) because the NIRISS WFSS mode does not need a number of the capabilities of the Mirage code. The normal inputs to the code are discussed in the next three sub-sections.

A Stellar Source List

The primary input is assumed to be a stellar source list with sky positions and simulated NIRISS magnitudes. The magnitudes can be input either as AB magnitudes or as the A0V-based magnitudes ("Vega magnitudes" in the Hubble Space Telescope parlance, although there some differences for the JWST magnitude system compare to what is used for HST). The file needs to have the specific structure shown in Figure 1 below otherwise the code cannot read it. First there may be one or more header lines with the '#' symbol at the start of the line. The only use of these lines is that the code looks for the string "abmag" or the string "vegamag". Then there is a column heading line, and after that the data values for the stars.

The columns must be in the format shown. The first column must be an index number. This is not used, but needs to be present. The second and third column must be the stellar position in decimal degrees, right ascension and declination. Following that the NIRISS magnitude values for the imaging filters need to be given. The column heading line must also be in the format shown for the magnitude columns, since the column name is matched to the filter of interest in the calculations.

The file is a Mirage stellar list input file, not the Mirage output point source list file. The latter type of file does not have the correct format for input to the WFSS overlap code.

An Extended Source (Galaxies) List

The extended source input file is similar in structure to the stellar source input file, with the addition of the Sersic profile parameters. The format needs to match the example shown in Figure 2 below. The Sersic profile parameters are the radius in arc-seconds, an ellipticity value between 0 and 1, a position angle in degrees east of north as is normal for astronomical images, and the Sersic profile index. The Sersic parameters must follow the source position and be followed by the magnitude values.

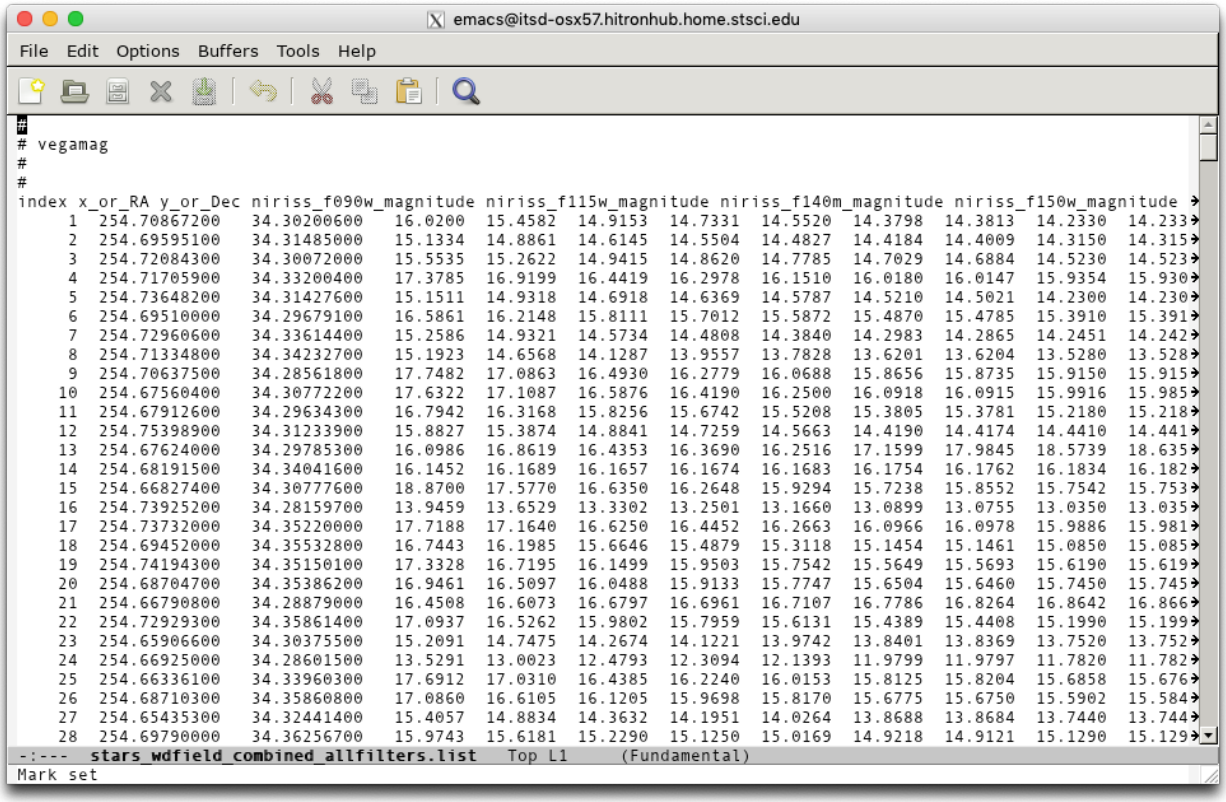

Figure 1: An example of a Mirage star list file in an emacs editor window. The Figure shows the start of the file where the magnitude system is defined by the "vegamag" keyword. This is followed by the column heading line and the data values.

Normal usage of the code is assumed to be having a stellar source list and making the extended source list optional. In the code as long as at least one file is specified it will function. Note that if both files are specified, they must be in the same directory.

Additional Input Values

Two additional parameters are needed for the code. One of these is the point assumed for the telescope in the simulated images. The other is a background count rate per pixel in units of ADU/s.

When a scene image is made the pixel positions associated with each star position are calculated using the pysiaf functions. The NIS_CEN aperture, which is the normal full frame imaging aperture, is used in these calculations. The position specified for the imaging corresponds to the reference point of the NIS_CEN aperture, located at pixel (1024.5, 1024.5). using a 1-based pixel numbering system. This is therefore offset by 1 from the normal Python indexing. The pixel centre is located at an integer number in the pixel position calculations. Thus, the central position is at the corner of pixels (1024, 1024), (1025, 1024), (1024, 1025), and (1025, 1025).

The conversion from sky position to pixels relative to the centre of the image in pysiaf includes a distortion model for the NIRISS detector, the same for all filters. This model currently is derived from ray-tracing simulations. Hence the distortion is allowed for in the

source positions but not in the extended source images used to make the scene image. As the distortion for NIRISS is expected to be small, this should not affect the simulations significantly.

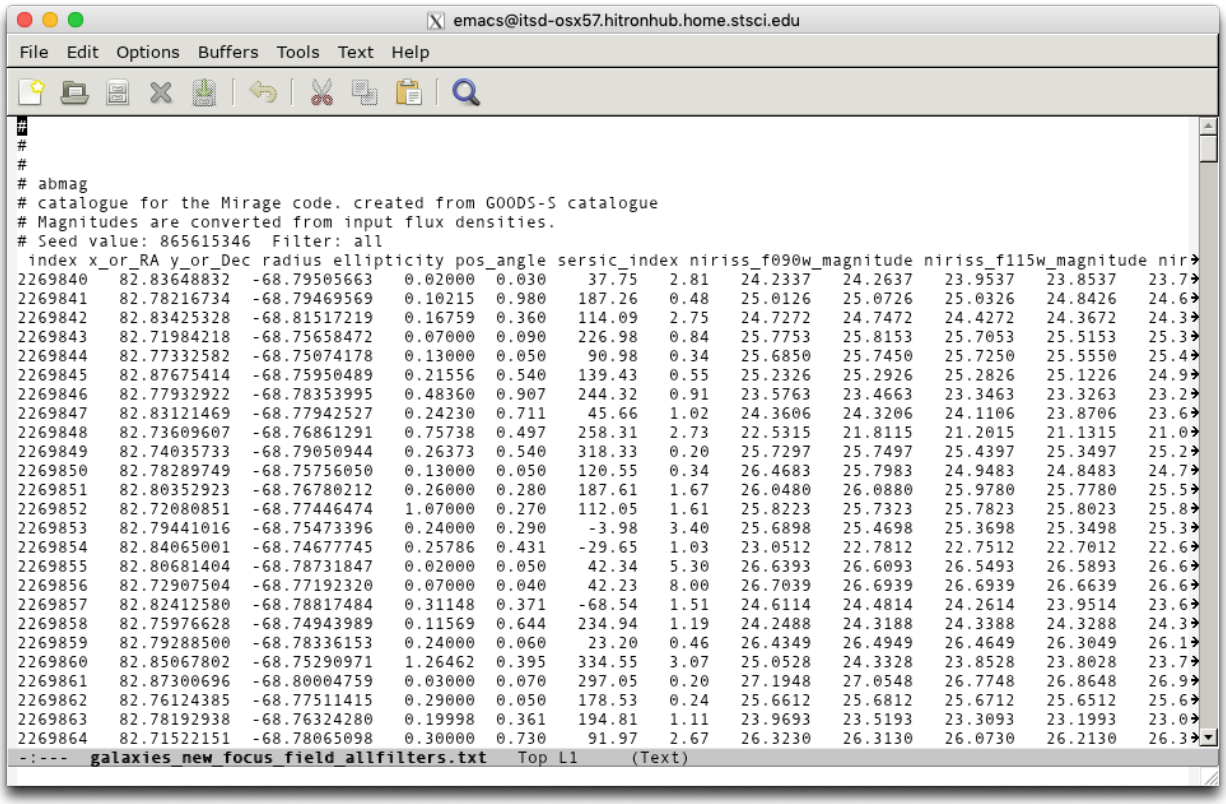

Figure 2: An example of the extended source input file seen in an emacs window. The structure is very similar to that of the stellar input file except that there are additional columns for the Sersic profile parameters. The columns need to be in the order shown above. The Sersic radii are in arc-seconds and the orientation angle is in degrees east of north on the sky.

Creation of the Scene Image

The scene image used in the code is an overside image that can cover the NIRISS pick-off mirror for any arbitrary orientation. The POM area as projected on the sky is assumed to be 2322 pixels square with the ideal pixel size of 0.0656″. The diagonal distance from corner to corner in the POM is then 3283 pixels, and the radius from the POM centre to the corners projected on the sky is 107.68″, The actual scene image size used is about 10% oversized, being 3631×3631 pixels. The nominal size of the scene on the sky is 3.96989′ square. When assembling the source lists for the code to use one should search a radius of at least 1.795′.

To make the scene image the pixel position of each star in the input list is calculated with respect to the defined image centre. If this pixel position is within the image area, the source brightness is calculated from the source magnitude in ADU/s units, and the value is added to the pixel. Each star is placed on a single pixel, whatever one is closest to the calculated position. For the extended sources, if any, the Sersic profile is calculated for a stamp image that is

301×301 pixels with the source position taken to be the centre of the middle pixel (150, 150) in the Python 0-based counting. This is done with the astropy.modelling.models Sersic2D routine. The normalized stamp image is scaled to the total signal calculated from the source magnitude. The edges of the stamp image are then calculated by matching the centre pixel of the stamp image to the pixel position of the source and taking that part of the stamp image that is within the scene image area. Note that if the centre position of the extended source is outside the scene image area then the Sersic profile is not calculated and the source is not included. Hence a large galaxy whose centre is off the scene image would not be included even if it would be large enough to show up on the image. With the padding of the scene image beyond the POM area, a galaxy would have to be larger than about 10″ in (total) radius to be excluded from the image and still be able to affect the POM area.

When making the scene image a uniform background is added to each pixel at the option of the user. The scene image is made in the standard orientation with north up and east at left.

Use of the Tool

The code is invoked by starting the wfss_overlap_tool.py code. This causes the creation of a parameter window as shown in Figure 3. The Figure shows the appearance of the window in a MacOS system; the appearance is slightly different on a Linux system.

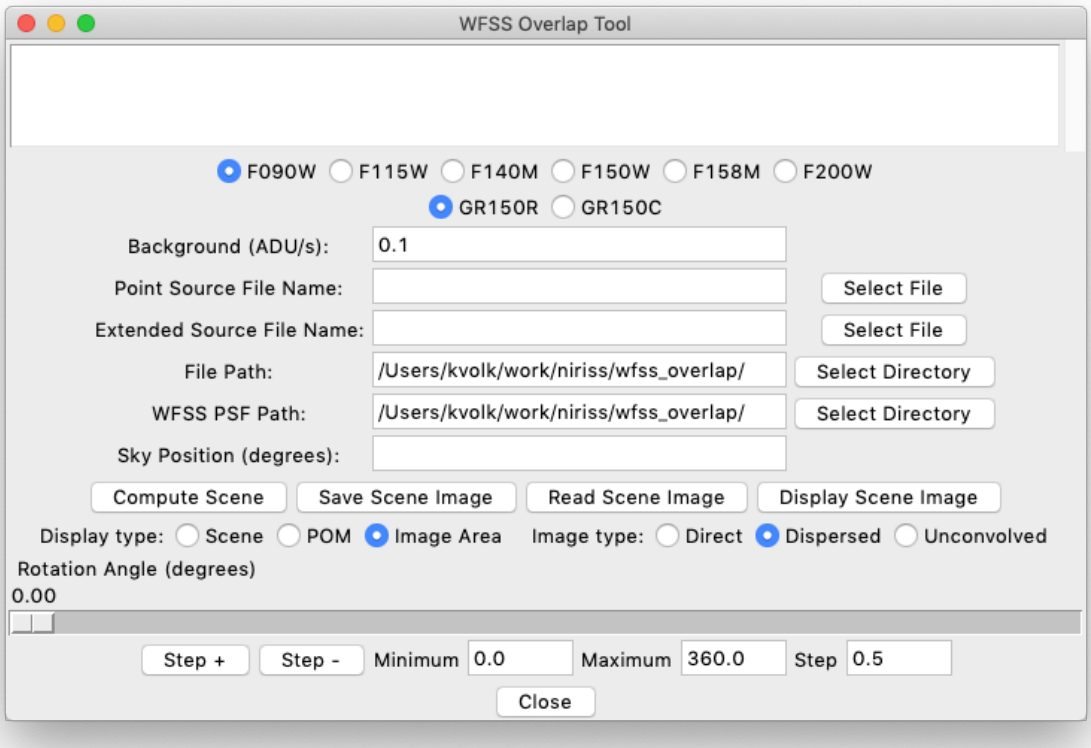

Figure 3: The initial appearance of the WFSS tool main window.

The tool has a message area at the top. Below that is an area for setting the parameters of the simulation. Just below the message area are radio buttons for the WFSS blocking filter and for the grism selection. Below that, entry fields are provided for the required inputs that the code needs to make a scene image. The first of these is the entry field for the sky background value. The next two entry fields are for the names of the stellar source list and the extended source list. These two files are assumed to be in the same directory, which is given in the "File Path" entry field. The code also needs to know the path to the imaging and WFSS PSF FITS images, which needs to be specified in the "WFSS PSF Path" entry field. When the code is started, the directory where the code is run is inserted into both the file path and PSF path fields. Finally, there is an entry field where one can specify the sky position α and δ in decimal degrees.

For selecting the stellar and extended source list files and for setting the path values there are buttons at right. If selected these bring up the normal tkinter file query window from which one can select a file name. When this is done for the "File Path" or "WFSS PSF Path" items the resulting file name is split into the directory path and the file name itself, and the directory path is inserted in the "File Path" field. One can also enter the file name and directory path values into the fields directly. For the two path entry fields the "Select Directory" button also brings up the standard file selection window. If one then selects any file in the target directory the directory path is extracted and put into the entry field. The file name itself is ignored in that case. Figure 4 shows the appearance of the tool with the fields filled in sufficiently to generate a scene image.

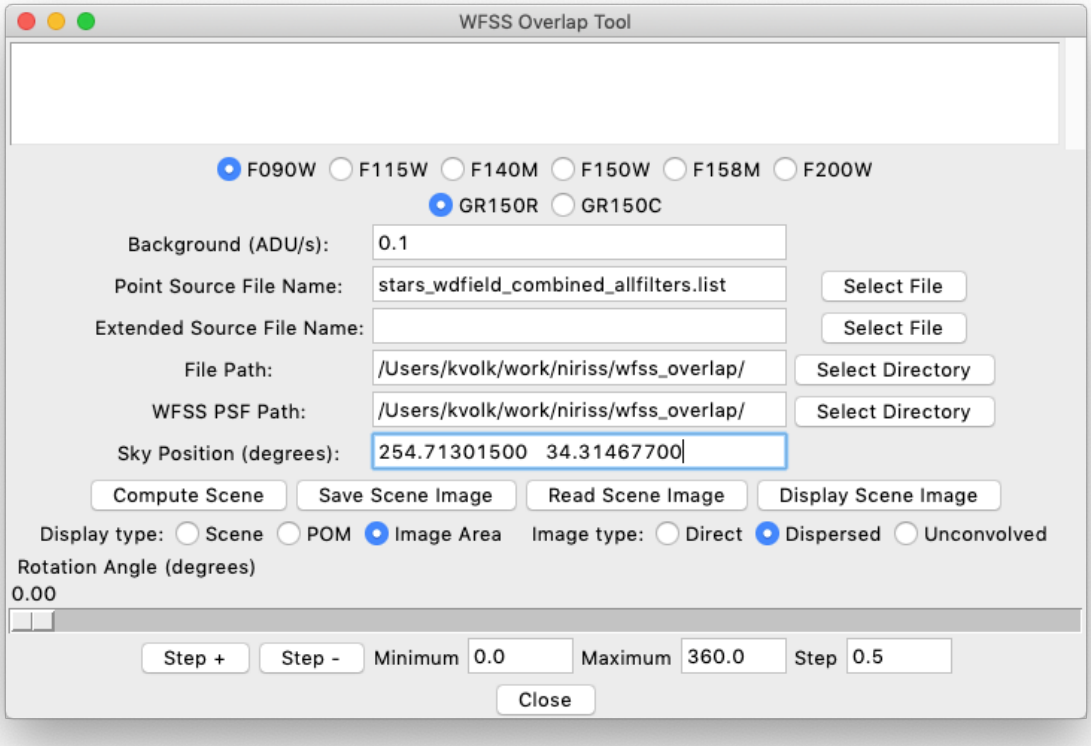

Figure 4: The appearance of the WFSS tool when the parameters for the scene generation have been specified.

The file stars_wdfield_combined_allfilters.list is the same file as was shown in Figure 1. The field is an area surrounding the photometry standard WD 1657+343, using the GAIA DR3 initial release to get the star positions and estimated NIRISS magnitudes. The sky position entered is the position of the white dwarf star in the GAIA catalogue. Thus, when the scene image is made the target star will be at the reference position in the image. One can leave the "Sky Position" entry field blank when making a scene image. If so, then the average position from the stars in the point source list is used as the reference position. Generally, it is better to specify the reference position explicitly before creating a scene image.

The next step that is required is to use the "Compute Scene" button to generate a scene image. In general, one needs one scene image per blocking filter, since the relative brightnesses of the sources in the field will vary over the six NIRISS filters. Figure 5 shows the appearance of the window after the scene image has been generated. There are a couple of lines in the message area noting that the scene image has been made. If there was an error, an error message would be shown in the message area.

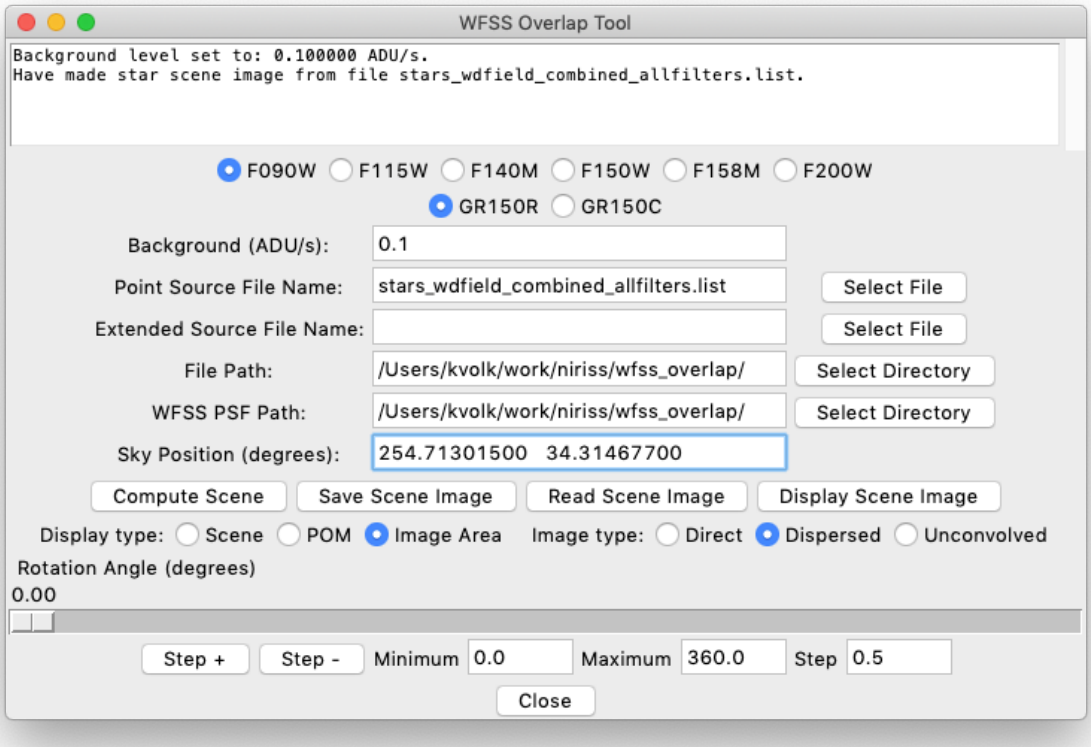

Figure 5: The appearance of the main window after the "Compute Scene" button has been activated.

The scene image may take some time to run if there are many sources, especially extended sources where the calculation time per object is larger. If calculation of a scene image is time consuming, the image can be stored as a FITS file using the "Save Scene Image" button, and such an image can be read back in with the "Read Scene Image" button. In such an instance, the scene image is read in directly and the scene image computation is not needed.

With this capability, one can also read in a scene image created from other observations if desired. However, in such a case the scene image needs to conform to the assumed structure of the scene image. The properties of concern are as follows:

- the image needs to be stored in FITS extension zero;
- the image needs to be real numbers, dimension 3631×3631 pixels with pixel size 0.0656″ square;
- within the scene image, the POM area is 2322×2322 pixels, the Python sub-array indexing being [655:2977, 655:2977];
- within the POM area the "output" image area has 2048×2048 pixels, the Python sub-array indexing being [792:2840, 792:2840];
- no negative numbers should appear in the image;
- the image should be deconvolved with the imaging PSF of the telescope/instrument.

The deconvolution of the image by the PSF is required so that the PSF properties of the instrument (e.g. diffraction spikes) are removed as much as possible from the scene image. If this is not done, such features will be imprinted on the simulations.

In the code the scene image is assumed to be in units of ADU/s in the current NIRISS blocking filter. If a scene from another instrument is used as input, the units can be arbitrary as long as the values are positive. The code generally is expecting numbers of magnitude roughly from 0 to 3000 ADU/s. Values much smaller or larger than this range may cause issues in some of the plots, but would not affect the calculations.

Displaying the Scene Image

Once a scene image has been created, or read in, the tool is ready to show images. One has controls in the lower part of the main window for the scene that is displayed. The filter and the grism are selected by the radio buttons at the top of the window. The rotation angle and the type of image to display are controlled by the options in the lower part of the window. One has three choices for the part of the scene image to display and three choices for the type of image to display. The type of image to be displayed can be either the scene image without convolving with any PSF file, the scene image convolved with the imaging PSF for the selected filter, or the scene image convolved with the WFSS PSF for the selected filter/grism combination. The first of these options, the "Unconvolved" radio button, is intended just to allow the user to verify the scene image if needed. This would not be very useful in the case of a scene with only stars present, but might be useful if there are extended sources in the field.

An example of one of the WFSS PSF files is shown in Figure 6. The image here is for the F090W/GR150C combination. The image is the result of measurements made during ground testing of NIRISS. One is able to identify orders from +7 to –3 in the image, although the higher orders are very faint. There are also a couple of faint optical ghost features seen in the image, which are thought to be due to reflections within the grism itself. These are seen just below the main spectrum, offset somewhat to the left. The entire image is normalized to 1 for the convolution, and the image is arranged so that the original undispersed point source position is in the centre of the image, so it correctly simulates the offsets from the direct image position to the spectral positions.

The convolution of the scene image with the WFSS PSF image means that the relative positions of the spectral orders from the source position is the same for all sources in the field.

What was observed in the ground testing was that the curvature of the line of the spectral orders varies a little over the field, although the sampling in the ground testing was somewhat sparse. This, the real observations will not match the simulations exactly. The hope is that variations in the trace curvature are small so that the results of the tool will be accurate enough for finding overlap.

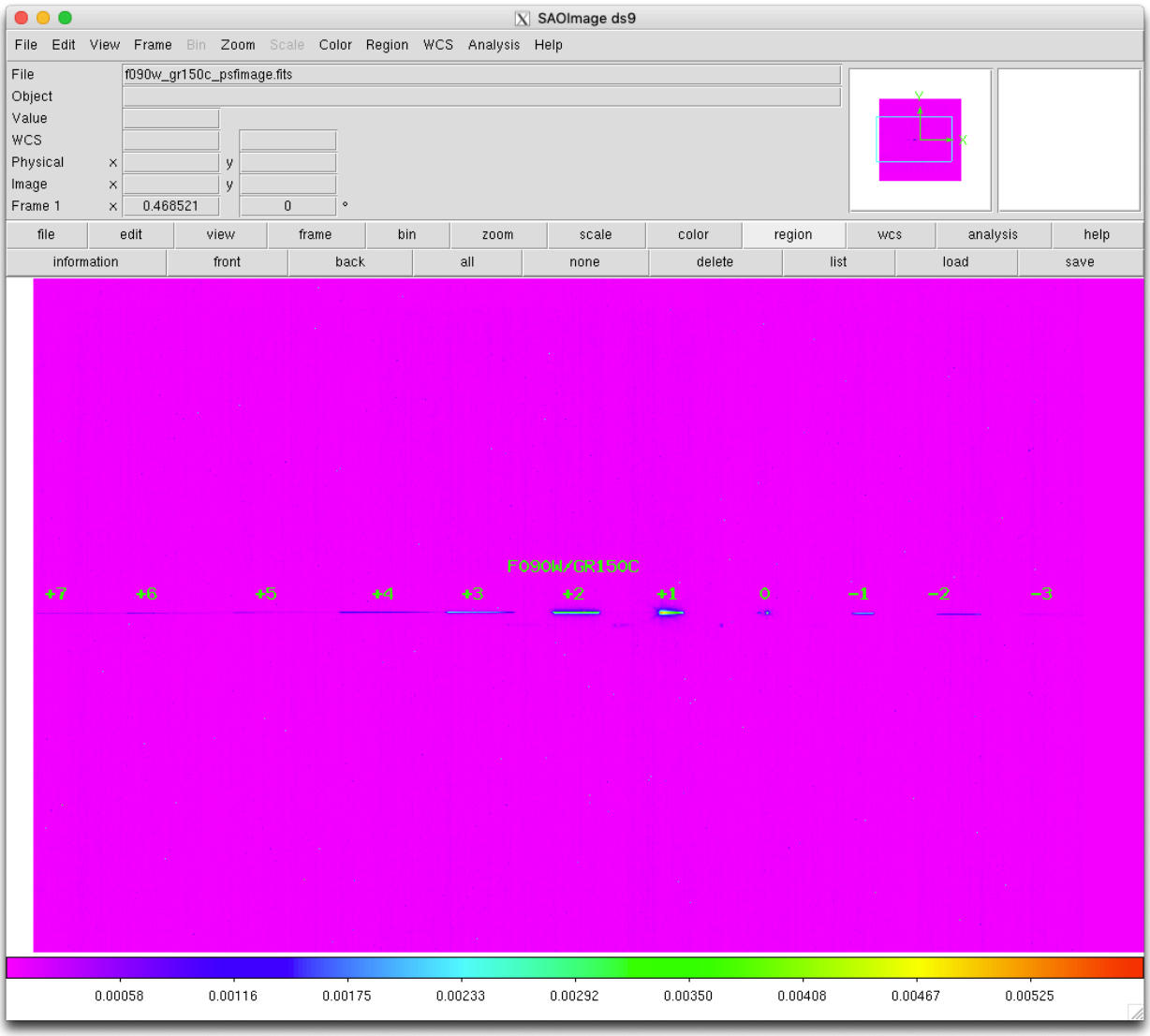

Figure 6: An example of one of the WFSS PSF images, with the spectral orders marked.

Normally one would want to examine the direct image to see the sources at various orientations long with looking at the resulting dispersed images. One has the option of showing only the NIRISS detector area, showing the POM area, or showing the while scene image. For the latter two cases the outline of the image area is drawn with a white dotted square and the outline of the POM area is shown by a white dashed square on the display if the full image area is shown. If the full POM area is shown then the image area is outlined by the dotted square drawn on the image. For the dispersed image case the scene area outside the POM is set to zero

when the full area is displayed, whereas for the direct image case all the sources over the scene area are shown.

When the "Display Scene" button is pressed the first time an image display window is created. All the images are displayed in this window. ***One should not close the image display window.*** If one does this, the code will not make a new window and one will have to restart the tool to get the display window again. The initial appearance of the image display window is shown in Figure 7 below. When an image is initially displayed the full image range is determined and these values are placed in the "Display Minimum" and "Display Maximum" entry fields. Also, an IRAF-style "zscale" function is applied to the image to get the minim and maximum values for a zscale display and these are put into the "Zscale Minimum" and "Zscale Maximum" fields.

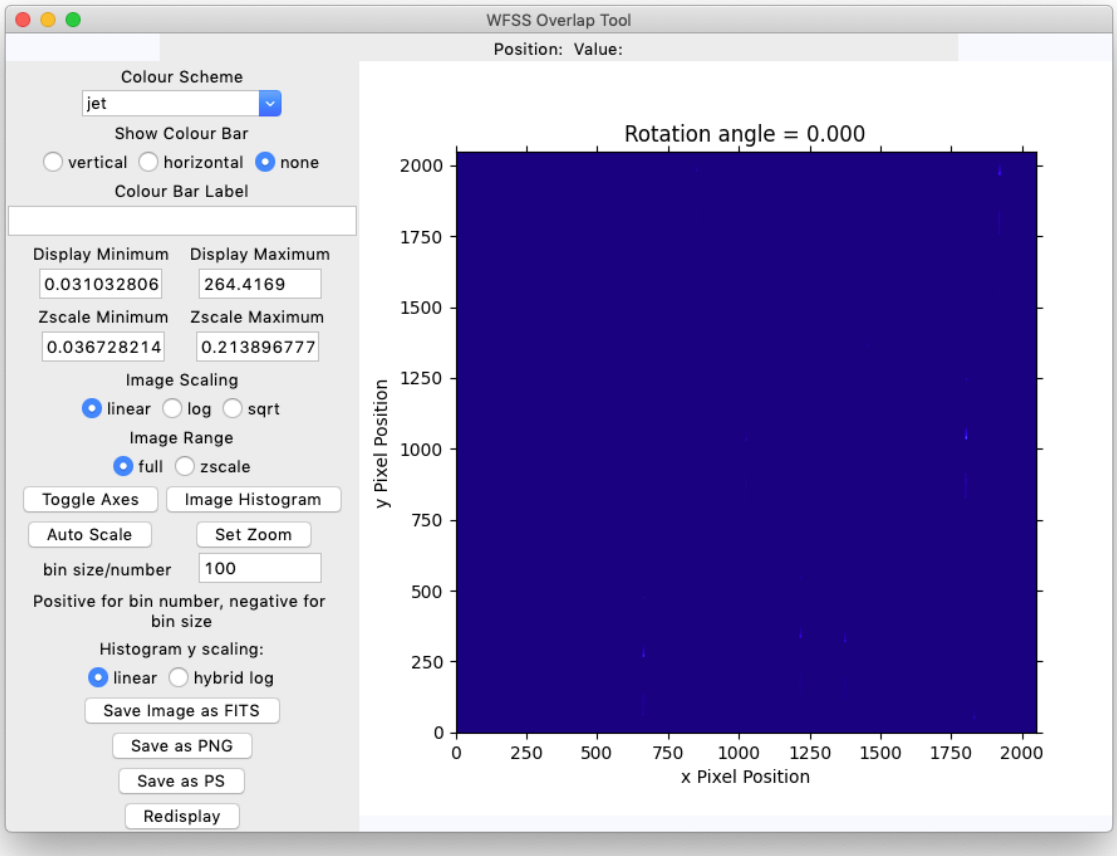

Figure 7: The initial appearance of the image display window, as seen on a MacOS system. The appearance will be slightly different on a Linux machine. The image is displayed at right and the controls for the image display are at right.

As seen in the above Figure, often when the default display parameters are used the image does not show much detail. The default is a linear display over the entire image range. There are a number of controls at left in the window that can be used to see more details in the image. One can use the "log" or "sqrt" radio buttons to change the mapping of the pixels in the

displayed image. The "sqrt" function takes the square-root of the absolute value of the pixels and for the negative pixels multiplies by –1 to make a new image which is then displayed. The "log" radio button directs the code to apply an IRAF-like logarithmic transformation to the image. In this case, the image is normalized to the range from 1 to 1000 linearly from the minimum to the maximum values, then the base 10 logarithm is applied to give values in the range from 0 to 3. This new image is then displayed. Figure 8 shows the appearance of the window with this option selected. This shows the spectra of the objects in the field.

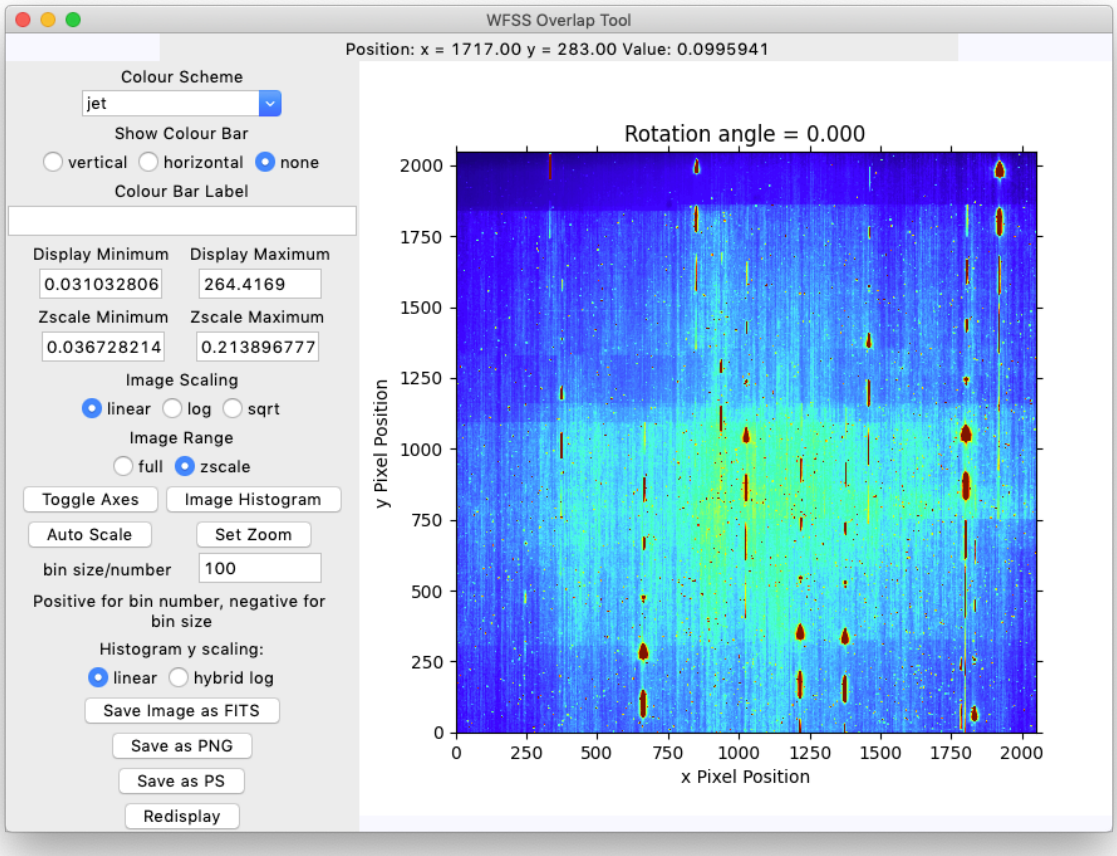

Figure 8: The image displayed using the "zscale" option to show the low level structure.

In the tool one can also change the "Display Maximum" or "Display Minimum" values in the entry fields, and use the "Redisplay" button at the bottom left to redisplay the image using the new parameters. This is only possible if the full image range option is selected, as the code calculates the zscale parameters and uses them directly, so any changes the user may attempt to make the entry values are ignored. If one changes the display range, the original values can be restored using the "Auto Scale" button. Figure 9 and Figure 10 show the display of this same image in the "sqrt" and "log" options, respectively, for the full image range. It is often useful to change the display maximum value to something lower than the full image range and redisplaying the image to allow one to see the low level signals in the image, since these are of the most interest for the evaluation of spectral overlap.

Once an image has been displayed when the cursor is on the image the pixel position and the image value at the cursor are written to the text field at the top of the window. The signal value shown is always the value in the original image and not in any transformed image that might be displayed.

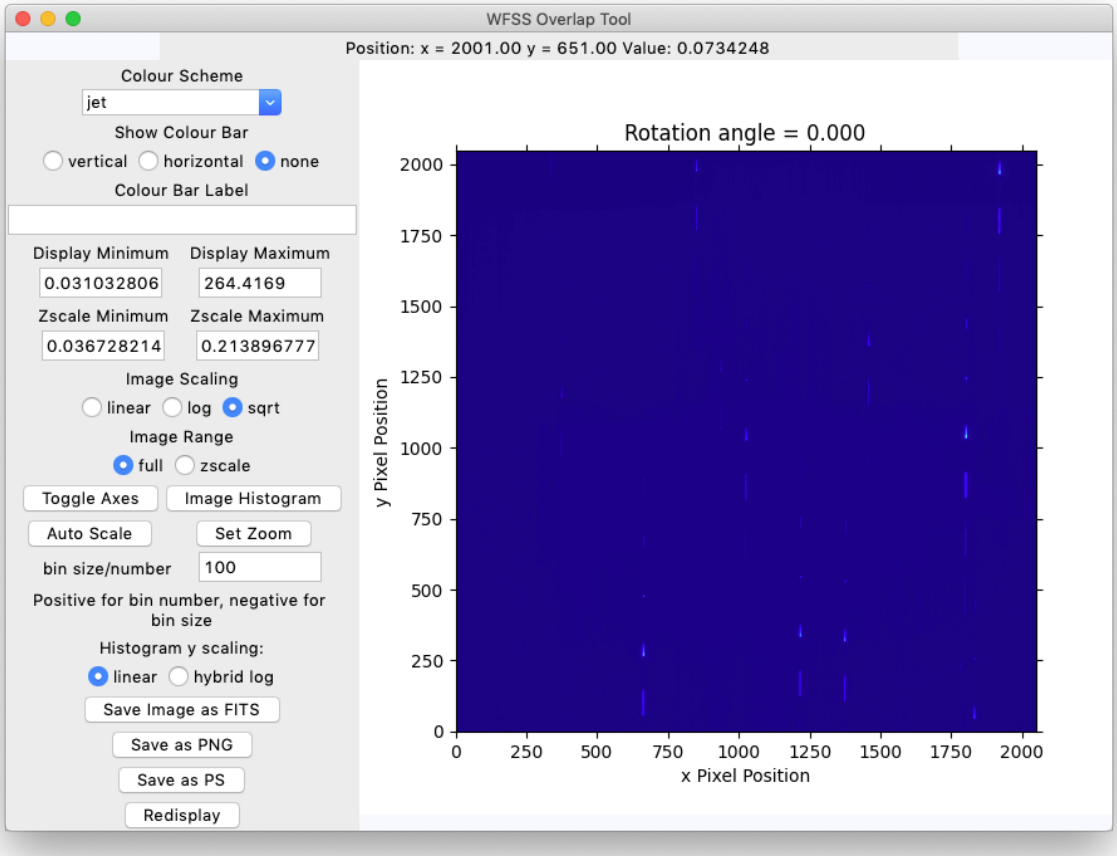

Figure 9: Display of the dispersed image with the "sqrt" option selected.

There is the option of adding a colour bar to the plot area if the "vertical" or "horizontal" radio buttons are selected. When a selection is made any text in the "Colour Bar Label" entry box is used to label the colour bar. If one changes the text in the label window one needs to change the radio button selection to make it take effect.

Another function at left is the "Toggle Axes" button. It toggles the display of the x and y axes in the plot.

The tool has an image histogram option. The display minimum/maximum entry fields set the range of the histogram. The "bin size/number" entry field sets the number of bins in the histogram if the value is positive and larger than 10. The value is truncated to an integer if it is a floating point value. If the value in the entry field is negative the absolute value is taken and this value is then used as the step size in the histogram. A new window is created to show the histogram plot.

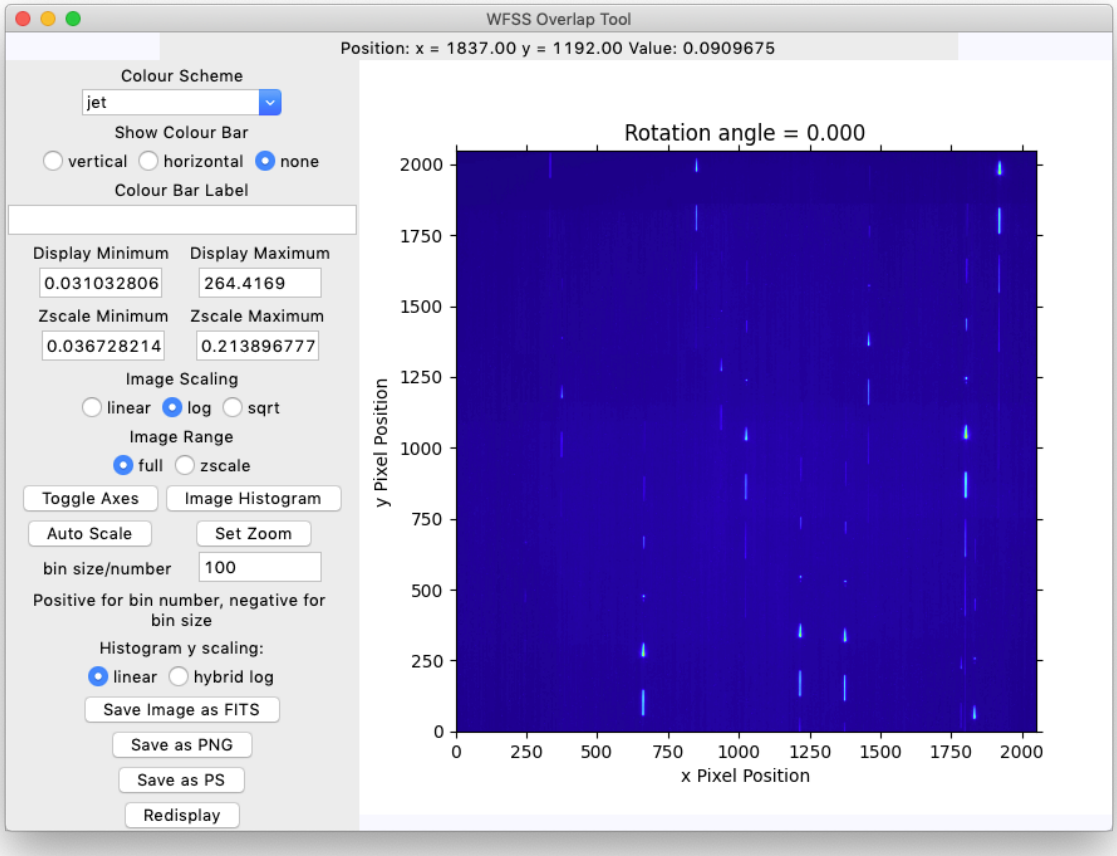

Figure 10: Displaying the dispersed image with the "log" option selected.

Use of the "Zoom" Function

The default in the image display is to show the full image. There is an option to display a sub-image rather than the full image. The "Set Zoom" button brings up an entry window where the user can enter an integer zoom value, 1 or larger. If a zoom value is entered, the size of the input image is divided by the zoom factor (rounded to an even integer) and a sub-image with this dimension is extracted around the image reference position. The reference position is originally placed at the centre of the image, but if one clicks with the mouse anywhere in the image display the pixel position thus selected becomes the reference point for zooming. When a zoom value is entered, the image is redisplayed. The various settings at right including the image range are not affected by extracting a sub-image to display.

The zoom values that are allowed range from 1 to a maximum value that produces a sub-image of size 64 pixels. One needs to set the zoom back to 1 to get the full image display if the value is set.

Each time a new image configuration is displayed by the tool, the zoom value is set to 1.

Key Commands

In the image display window one can use key commands “j” and “k” to get image profiles in the x and y directions respectably. These are similar to the IRAF imexam key commands. The code takes the curser position where the key is pressed, and extracts a pixel area 20 pixels in the x direction and 5 pixels in the y direction for the j key, or 20 pixels in the y direction and 5 pixels in the x direction for the k key. The extracted region is the averaged over the short dimension and plotted as a vector. Thus when one is looking at a dispersed image with the spectra along the y axis one can get a cross-cut profile with the j key, and for the case of spectra along the x axis one can get a cross-cut profile with the k key. Along with the plot of the data values, the code carries out a non-linear least squares fitting of the data with a Gaussian profile plus a constant baseline. The best fit function is overplotted along with the data values, and the parameters are shown on the plot. When such a plot is done, the plot window needs to be closed before any further input can be entered into the main window or in the image display window.

The ”r” and “c” key commands for plotting the row or column at the current cursor position are also provided.

Rotating The Scene Image

The main purpose of the tool is to allow the user to change the position angle and see how this affects the output image. For changing the angle, one has the option of using the slider in the main window to set the rotation angle, measured in degrees east of north in the usual way, to any value from 0° to 360° with a resolution of 0.01°. Alternatively, one can use the “Step+” and “Step–” buttons to change the current rotation angle by whatever step amount is given in the “Step” entry field. The range of allowed angles for the step function is read from the “Minimum” and “Maximum” entry fields. If one just clicks on the slider with the cursor, it moves 0.01° either up or down depending on the side of the slider where the cursor is located.

The image rotation is done using the scipt.ndimage.rotate function as noted previously. The entire image is rotated about the centre position, so if a source is placed on the centre position, then its spectrum or image PSF remains fixed as the rotations are done. Each time a rotation is done the image display is updated. The rotation calculations are not fast enough to allow the tool to have a “movie mode”.