

# FITS Viewer and VMI Analysis

## Version 4.0

FITS Viewer and VMI analysis, Version 4.0. This program was written to display and analyse 2-dimensional data, in particular for the analysis of Velocity Mapped Imaging (VMI) data and 2-Dimensional Laser Induced Fluorescence (2D-LIF) data. The program uses the Flexible Image Transport System (FITS) file format for loading, processing and saving of images, although images stored as text may also be imported. The program incorporates many functions to process VMI images. In particular, routines are provided to perform the Inverse Abel Transform on experimental images and generate polar coordinate images of the raw and transformed data. In addition, for raw VMI images that are affected by non-circularity due to stray electric and magnetic fields, routines are provided to determine such distortions and produce a circularised (undeformed) image. This program has been developed and used at Flinders University since 1998 for analysis of VMI and 2D-LIF data. The circularisation algorithms used in this program are described in [J. R. Gascooke, S.T. Gibson and W.D. Lawrance, "A "circularisation" method to repair deformations and determine the centre of velocity map images", *J. Chem. Phys.* **147**, 013924 (2017) (DOI 10.1063/1.4981024)].

When this program is used to analyse data used in publications, it should be referenced as [J.R. Gascooke and W.D. Lawrance, FITS Viewer and VMI Analysis: A Program for Analysing and Circularising VMI Images, DOI 10.4226/86/59278ab872838].

### Disclaimer:

This program was first developed in 1998 and continuously modified/improved over the last 19 years. Although a lot of effort has gone into making the program stable and correct, it hasn't been tested under all conditions and possibilities. Therefore, it cannot be guaranteed that the program will do what you want it to do! Use the program at your own risk. If, through using this software, you become aware of bugs then please contact the author, Jason Gascooke via email: Jason.Gascooke@flinders.edu.au. Suggestions for possible improvements would also be useful, but cannot be guaranteed to be incorporated.

### **FITS Images**

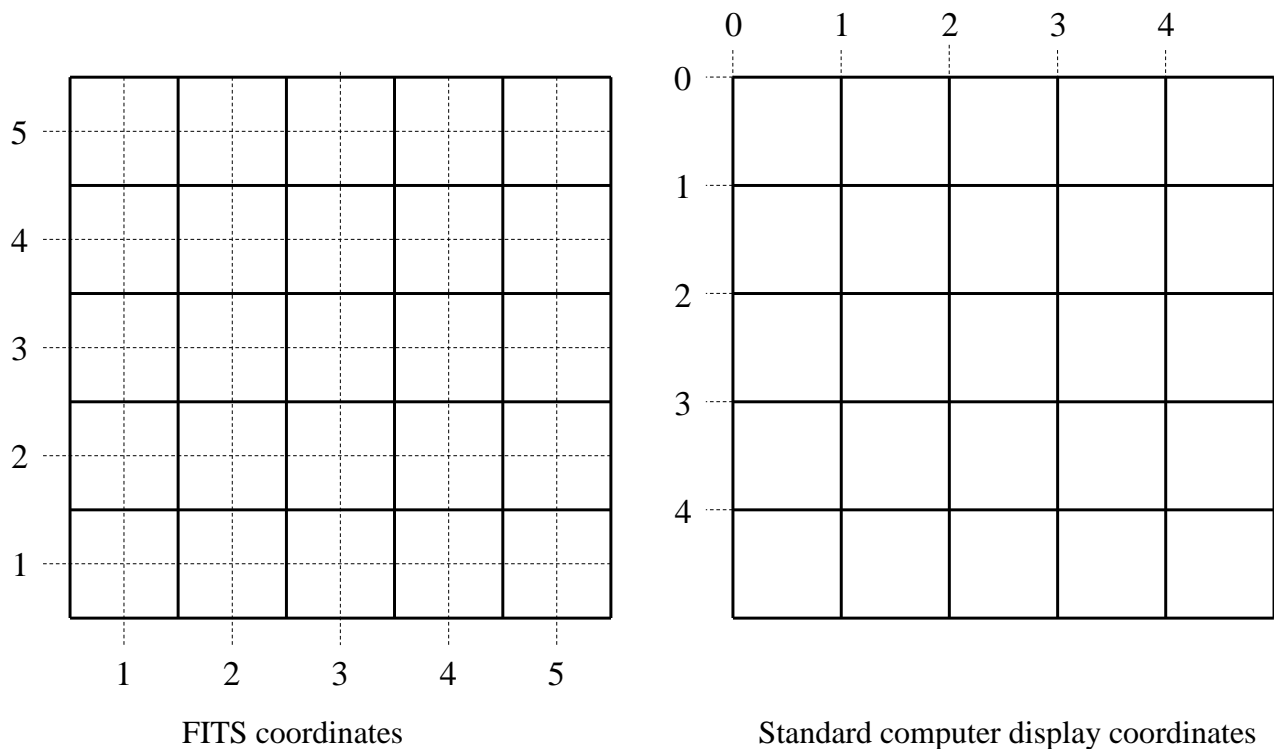
This software uses the "Flexible Image Transport System" (FITS) file format for processing images. This is a file format popularised by the astronomy community and has several advantages over other image file formats. The file structure has a readable header (ASCII) that is used to describe the type, size, calibration of the image and any other desired information about the image. The image format allows up to 999 dimensions of data, and the data can be 8, 16, 32, 64 bit integer data or 32, 64 bit floating point data. This program can read in all these dimensions if contained in the image but will only display the first two dimensions as an image with the remaining dimensions set to their first element. Support for 64 bit integer data is not yet available in this program.

The FITS format is different to other image formats in that there is no information in the file describing how to display the image (i.e there is no colour information). Instead, the purpose of the FITS file format is to contain data, and it is the purpose of a FITS viewer to present the data in the form of an image. (Viewers do have the option of incorporating their own display information in the file header, but this is program dependant).

It is generally accepted that 2-dimensional FITS data should be displayed in the 1st quadrant of a Cartesian plane. That is, the origin located at the lower left corner, and the axis increasing up (y axis) and to the right (x axis). Also, it is accepted that the 1st dimension of data relates to the x

direction and the 2nd dimension is the  $y$  direction. These conventions aren't prescribed in the FITS standard since FITS images contain physical data as pixels (voxels) and not a display image.

Unlike most image formats that have their origin of (0, 0) at the top right corner, the origin for fits coordinate is (1, 1) and, as just mentioned, is located at the lower left corner. This must be taken into account when comparing pixel positions with other programs. Moreover, for a FITS pixel, the axis index is accepted to be the centre at the centre of the pixel, so the (1, 1) pixel extends in the  $x$  and  $y$  directions from 0.5 to 1.5. In computer graphics, the origin pixel (0, 0) ranges from 0 to 1 in the  $x$  and  $y$  directions. This is shown diagrammatically below:



For more information on FITS specifications, see:

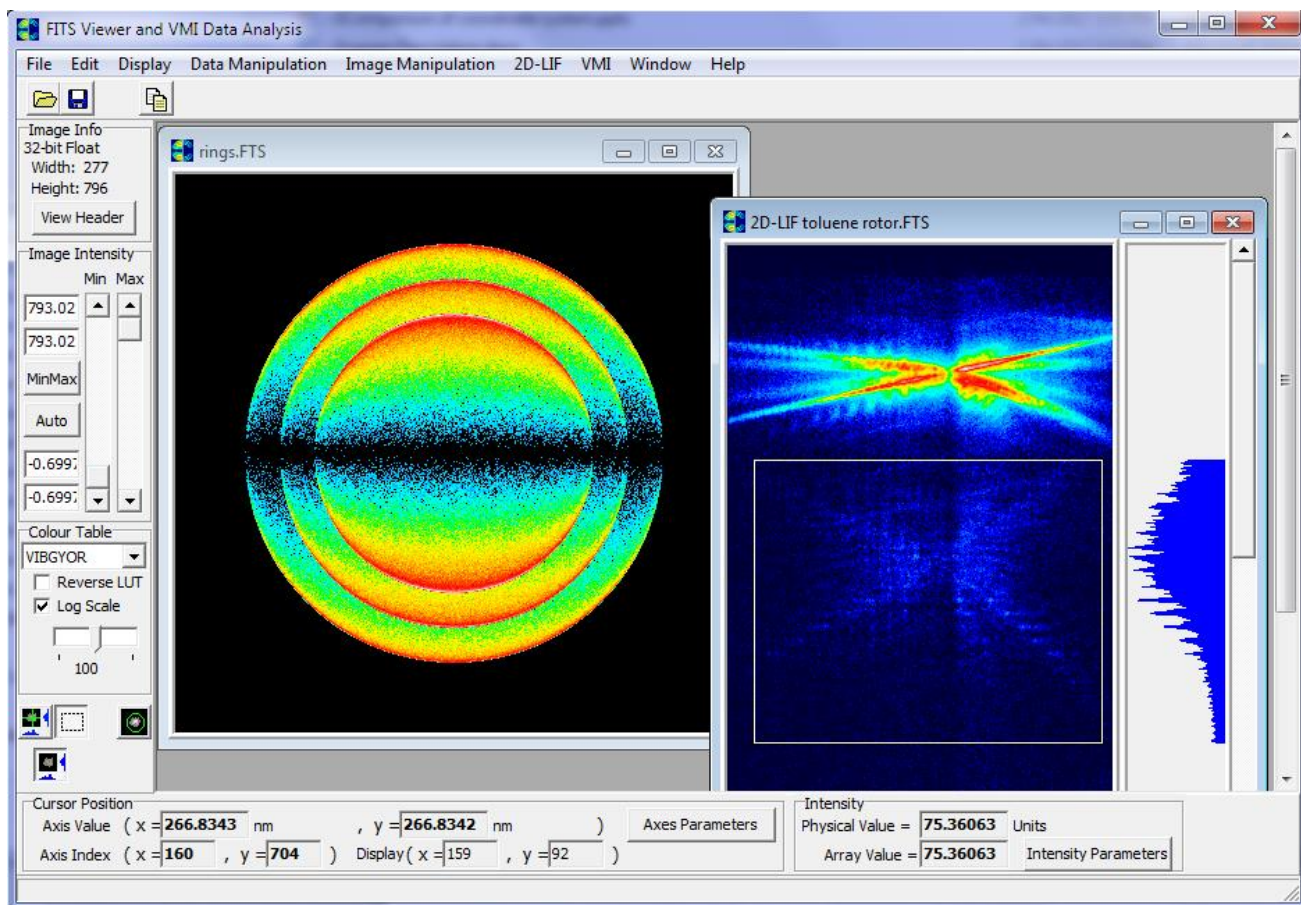
FITS Working Group, "FITS Standard 3.0" (2008) accessible at [http://fits.gsfc.nasa.gov/standard30/fits\\_standard30.pdf](http://fits.gsfc.nasa.gov/standard30/fits_standard30.pdf)  
E.W. Greisen and M.R. Calabretta, "Representations of world coordinates in FITS", A&A, **395**, 1061 (2002). DOI: 10.1051/0004-6361:20021326

## Running the Software

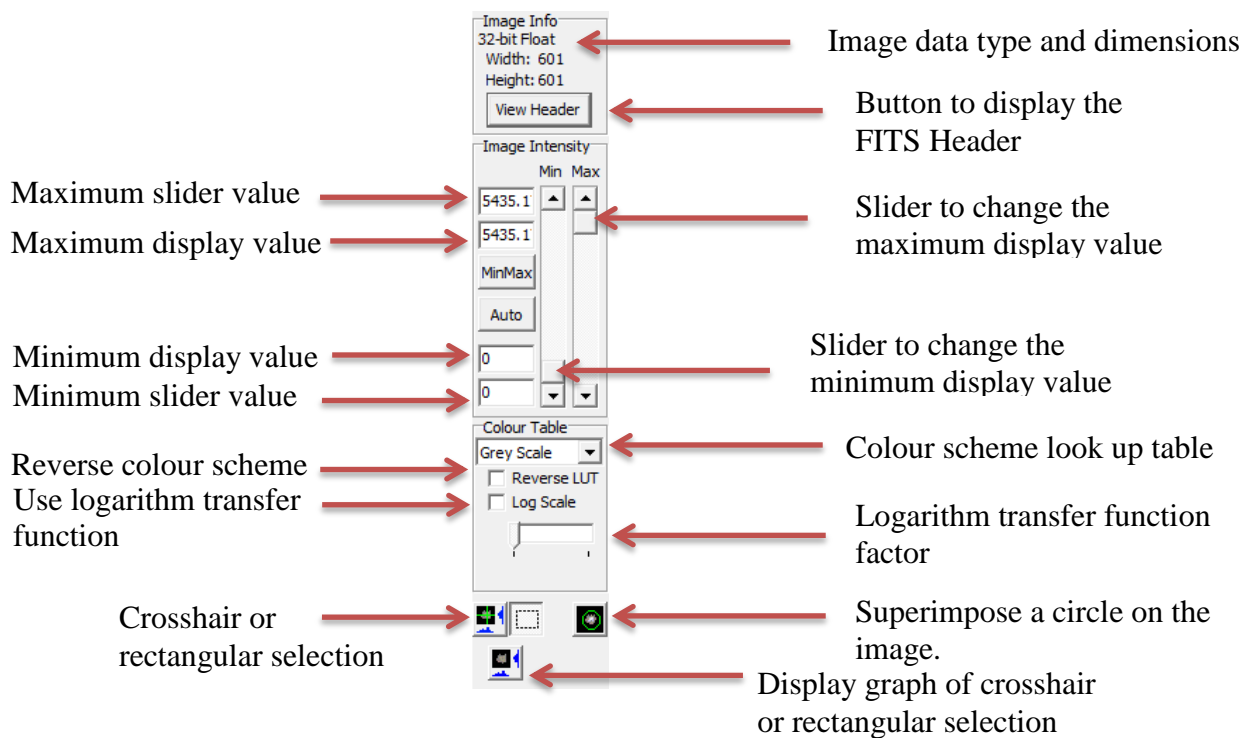
The program is a stand-alone executable that doesn't require additional installations. It should run under all Windows operating systems starting with Windows 98. The program doesn't save any information to the registry nor does it save a configuration file.

The software can be copied to any directory and run from this location (if permissible by local permissions). Just double click on the program from within windows browser, or make a short cut on the desktop or Start menu. It is useful to associate files with extensions ".FITS" and ".FTS" to this program, so you can simply double-click on a FITS image file and load it into the viewer.

When running, the program can display a number of images within the main window. An example screenshot of the program is shown below, where two images have been loaded.



## Side ToolBar



“Image Info” section contains:

- the type of data that is displayed (i.e. number of bits and integer vs float),
- a button to display a window with the current FITS header (see section on FITS headers)

“Image Intensity” section contains:

- controls to modify the appearance of the selected image.

The minimum and maximum display values refer to the array values (intensities) in the image that correspond to the minimum and maximum colours in the colour table. The display values can be changed by sliding the slider bar up and down. Note, that the minimum slider value can’t go above the maximum slider bar value.

Min and max display values can be entered manually followed by pressing the return/enter key.

The min and max limits for the slider bar values can be modified by manually entering the value in the “Maximum/Minimum slider value” followed by the return/enter key.

When an image is initially displayed, the minimum and maximum slider values are set to the minimum and maximum values in the array data of the image.

- buttons to change minimum and maximum display values to predefined values, either to the min and max values in the image via the “MinMax” button, or a best guess using statistical information derived from the image by using the “Auto” button.

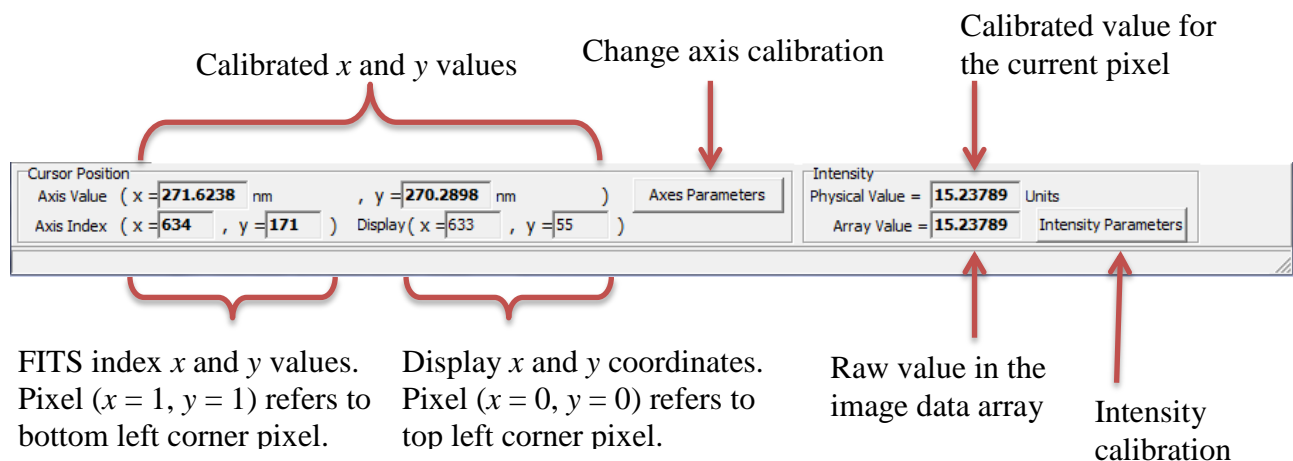
“Colour Table” section contains:

- dropdown box to allow selection of different colour look up tables.
- checkbox to reverse the order of colours in the lookup table.
- checkbox and slider to select a logarithm transfer function, and the a logarithm factor that describes the amount of “logness” of the transfer function (a value of 1.0 represents a linear function and large values cause the colours change much more rapidly for low intensities in the image). Note that after selecting log function, the maximum slider value may need to be increased significantly to obtain a suitable image.

Underneath the above sections are buttons to activate image tools, including:

- crosshair button allowing a cross to be placed on the image. The position of the cross can be moved by left-clicking the mouse, or by using the keyboard. Left, right, up and down arrow keys move the crosshairs by 1 pixel. If the Ctrl key is held down at the same time as pressing left, right, up or down then the crosshairs move by 10 pixels.
- rectangular selection button allows a rectangle to be selected on the image by left clicking the mouse and dragging a rectangle over the image while holding the left mouse button down.
- graph button that displays a graph below and to the right of the image that plots the cross-section if crosshairs are selected, or, when rectangular selection is used the projection of the data in the rectangle is plotted. The graph can be exported by right clicking in the graph window and selecting “Save Plot”.
- button to superimpose a circle on the image, which can be used as a guide to determine approximate centre and radius of circular images. The position of the circle is moved by 1 pixel at a time by pressing left, right, up and down arrow keys. If the Ctrl key is held down at the same time as pressing left, right, up or down then the crosshairs move by 10 pixels. The radius of the circle is changed by holding down the shift key while pressing the left/right keys (and the ctrl key to increase the radius in bigger steps). Values of the circle centre and radius is given in the bottom status bar.

## Lower Information Bar



The current cursor position, or crosshair position on the image, and its associated data value (intensity) is displayed in the lower information bar. Since the FITS file format allows calibration of each axis as well as the values in the image array (intensities) the raw position and array value is shown, along with their calibrated values (including units). Note that FITS images have their origin as (1, 1) and it is customary (but not in the standard) that the origin is located at the lower left pixel. This is different to the usual display coordinates where (0,0) refers to the top left pixel.

Clicking on the "Axis Parameters" button brings up a window that allows the setting of calibration information, such as that displayed below. These allow the changing of FITS header values associated with axis calibration.

The "Axis Parameters" dialog box shows the following settings:

- Y Axis:**
  - Label: Absorption Wavelength
  - Units: nm
  - Scale: 0.0010016 nm per pixel
  - Reference Pixel: 1 pixel
  - Value at Reference Pixel: 270.11948 nm
- X Axis:**
  - Label: Fluorescence Wavelength
  - Units: nm
  - Scale: 0.000905898018766 nm per pixel
  - Reference Pixel: 0 pixel
  - Value at Reference Pixel: 271.049459402843 nm

The dialog also includes a preview window showing a small image with a crosshair and "OK" and "Cancel" buttons.

Program Variable	FITS keyword
Label	CTYPE $n$
Units	CUNIT $n$
Scale	CDEL $Tn$
Reference Pixel	CRPIX $n$
Value at Reference Pixel	CRVAL $n$

In this example the y axis represents Absorption Wavelength in units of nm. Each pixel steps by 0.0010016 nm. Pixel number 1 (the first pixel) is the reference pixel and corresponds to a wavelength of 270.11948 nm. Note that the reference pixel need not be an integer value. In the top

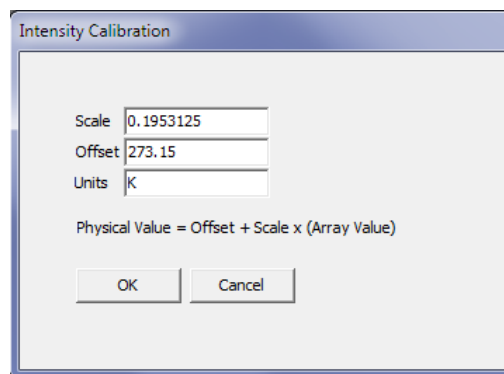
right corner is a reduced size image with the calibrated values of the image extremes displayed underneath and to the left of the image.

The calibrated axis value at a particular index is given by:

$$\text{Calibrated Value} = (\text{index} - \text{CRPIX}_n) * \text{CDELTA}_n + \text{CRVAL}_n$$

Note that the first pixel has an index = 1.

Clicking on the “Intensity Parameters” button produces a window that allows the calibration of the data array (intensity) values. These allow the changing of FITS header values associated with intensity. This is really only of any benefit for integer data.

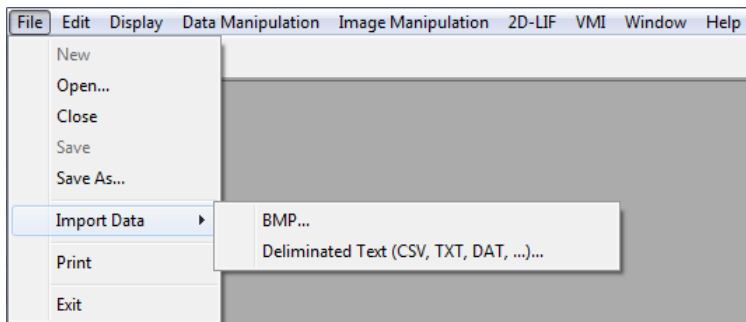
A screenshot of a software dialog box titled "Intensity Calibration". It contains three input fields: "Scale" with the value "0.1953125", "Offset" with the value "273.15", and "Units" with the value "K". Below these fields is the formula "Physical Value = Offset + Scale x (Array Value)". At the bottom are "OK" and "Cancel" buttons.

Program Variable	FITS keyword
Scale	BSCALE
Offset	BZERO
Units	BUNIT

In this example, the data represents temperature, where the array values (intensities) are integers that are converted to values in Kelvin by multiplying by 0.1953125 and adding a 273.15K offset.

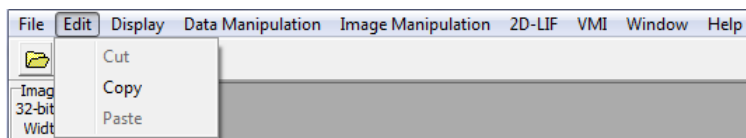
## Menu Items

### *File Menu*



New	Disabled.
Open...	Allows selection of a FITS image to view.
Close	Closes Image.
Save	Disabled.
Save As...	Allows user to save the FITS image.
Import Data	Allows the importation of a greyscale bitmap file (*.BMP) or from an ASCII file containing delimited text. The delimiter may be a space, comma, or a tab. Header lines that don't start with a numerical value are ignored.
Print	Prints the image in a rudimentary way. It is suggested that the copy function is used instead and the image pasted into an application capable of printing images.
Exit	Exits the program

### *Edit Menu*



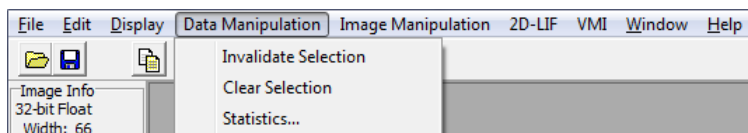
Cut	Disabled.
Copy	Copies the current image bitmap into the clipboard so it can be pasted into other applications.
Paste	Disabled.

### *Display Menu*



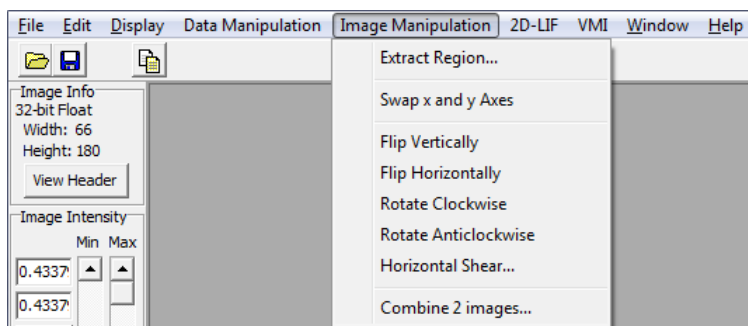
The Display menu contains a menu item to display alternate units in the lower information bar. Currently there is only one option and that is to display nm units in  $\text{cm}^{-1}$ . When selected a check box will appear in the lower information bar to turn on and off unit conversion.

## Data Manipulation



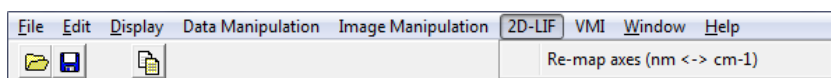
- |                      |   |
|----------------------|---|
| Invalidate Selection | Invalidates the current point (for crosshair selection) or points within a rectangular region (for rectangular selection) by changing the data array value (intensity) to NAN (Not A Number). |
| Clear Selection      | Zeros the current point (for crosshair selection) or points within a rectangular region (for rectangular selection).  |
| Statistics...        | Shows a window giving statistics about the image. If a rectangular region is selected, then the information only pertains to data points within the rectangle.                                |

## Image Manipulation



- |                      |   |
|----------------------|---|
| Extract Region...    | Brings up a window that allows the user to extract a region from the image. If a rectangular selection is active, then the window pre-populates the values with those of the rectangular selection. |
| Swap x and y Axes    | Switches the $x$ and $y$ axis. The pixel at the lower left corner, i.e. (1,1) remains the same. Axis calibrations are preserved.  |
| Flip Vertically      | Flips the image vertically. Axis calibrations are preserved.  |
| Flip Horizontally    | Flips the image horizontally. Axis calibrations are preserved.  |
| Rotate Clockwise     | Rotates the image clockwise. Axis calibrations are preserved.   |
| Rotate Anticlockwise | Rotates the image anticlockwise. Axis calibrations are preserved.   |
| Horizontal Shear...  | Displays a window that allows an image to be sheared (currently only horizontal shearing is available). Shearing is useful in the analysis of 2 dimensional laser induced fluorescence images.      |
| Combine 2 images...  | Opens a window to allows two FITS images to be combined into a single image. Many options exist, including the use of FITS calibration to position 2 images relative to each other.                 |

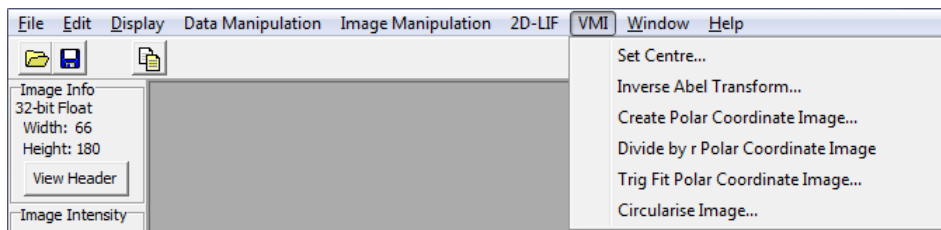
## 2D-LIF (2-dimensional laser induced fluorescence)



- |  |   |
|--|---|
| Re-map axes (nm <-> cm <sup>-1</sup> ) | Converts axes that are in nm to axes in cm <sup>-1</sup> via a nonlinear transformation (and <i>vice versa</i> ). |
|--|---|

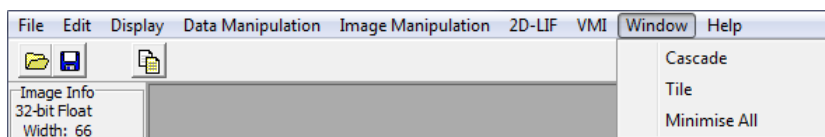


## VMI (velocity mapped imaging)



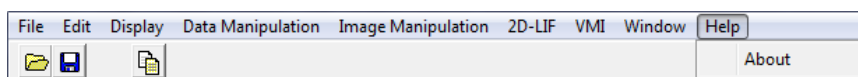
Set Centre...	Allows setting of the centre of the VMI image.
Inverse Abel Transform...	Generates the Inverse Abel transform image.
Create Polar Coordinate Image...	Generates a polar coordinate image.
Divide by r Polar Coordinate Image...	Divides the intensity of each pixel in image by the radial value (assumes the radial coordinate is in the $x$ direction and uses the axis calibration to determine $r$ ).
Trig Fit Polar Coordinate Image...	Fits a single wavy line in a polar coordinate image to a series of sines and cosines.
Circularise Image...	Corrects radial distortions of an image using a series of sine and cosine deformation parameters.

## Window



Cascade	Cascades all open windows within the viewer.
Tile	Tiles all open windows within the viewer.
Minimise All	Minimises all windows.

## Help

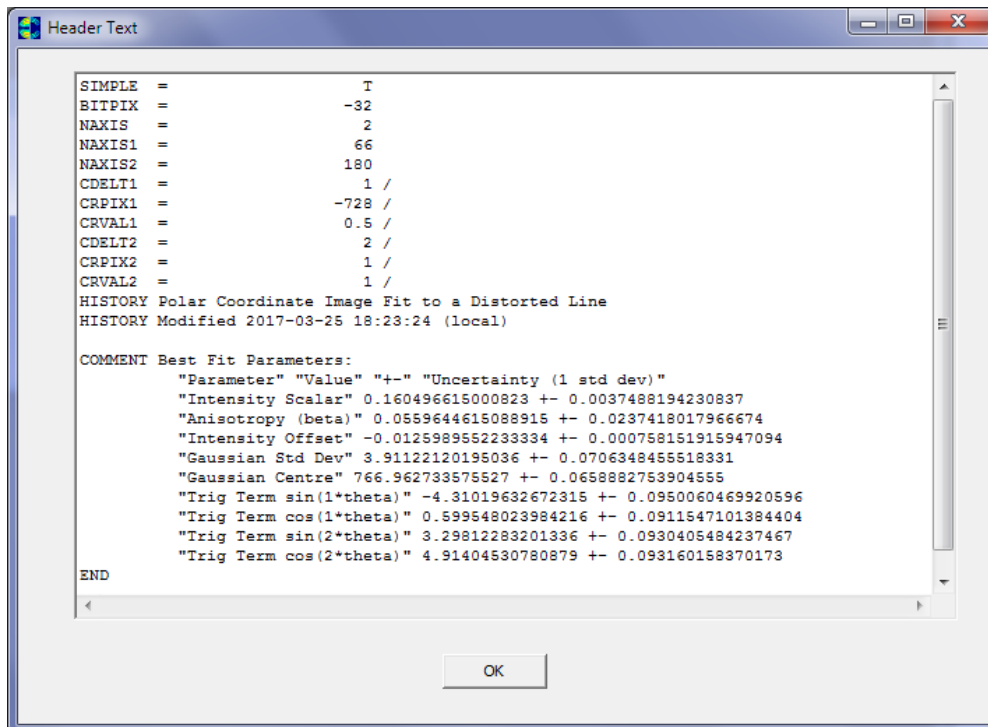


About	Provides more information about the program.
-------	--

## FITS Header Information

The start of a FITS file contains a ASCII readable header. The FITS specification demands that some keywords are mandatory.

A typical example is given below:



```
SIMPLE = T
BITPIX = -32
NAXIS = 2
NAXIS1 = 66
NAXIS2 = 180
CDELT1 = 1 /
CRPIX1 = -728 /
CRVAL1 = 0.5 /
CDELT2 = 2 /
CRPIX2 = 1 /
CRVAL2 = 1 /
HISTORY Polar Coordinate Image Fit to a Distorted Line
HISTORY Modified 2017-03-25 18:23:24 (local)

COMMENT Best Fit Parameters:
      "Parameter" "Value" "+-" "Uncertainty (1 std dev)"
      "Intensity Scalar" 0.160496615000823 +- 0.0037488194230837
      "Anisotropy (beta)" 0.0559644615088915 +- 0.0237418017966674
      "Intensity Offset" -0.0125989552233334 +- 0.000758151915947094
      "Gaussian Std Dev" 3.91122120195036 +- 0.0706348455518331
      "Gaussian Centre" 766.962733575527 +- 0.0658882753904555
      "Trig Term sin(1*theta)" -4.31019632672315 +- 0.0950060469920596
      "Trig Term cos(1*theta)" 0.599548023984216 +- 0.0911547101384404
      "Trig Term sin(2*theta)" 3.29812283201336 +- 0.0930405484237467
      "Trig Term cos(2*theta)" 4.91404530780879 +- 0.093160158370173

END
```

In this example, the SIMPLE keyword is set to the Boolean value TRUE, indicating it is a standard FITS format file. The BITPIX keyword refers to the type of data in the image. Positive numbers contain integer data and negative numbers contain floating point data. The absolute value of this number defines the number of bits in each value. Thus, the above image contains 32 bit floating point data. NAXIS defines the number of axes, which can range from 1 to 999. NAXIS1 and NAXIS2 defined the number of indices in each axis (ie width and height for a 2D image). CDELT<sub>n</sub>, CRPIX<sub>n</sub> and CRVAL<sub>n</sub>, refer to calibration parameters of the axes (see the Axis Parameters section above). Axes can also have labels and units as given by CTYPE<sub>n</sub> and CUNIT<sub>n</sub>. Other keywords in the FITS specification may appear in the header, as well as custom keywords (for example centres of VMI images is given by the VMI\_CX and VMI\_CY keywords in the program).

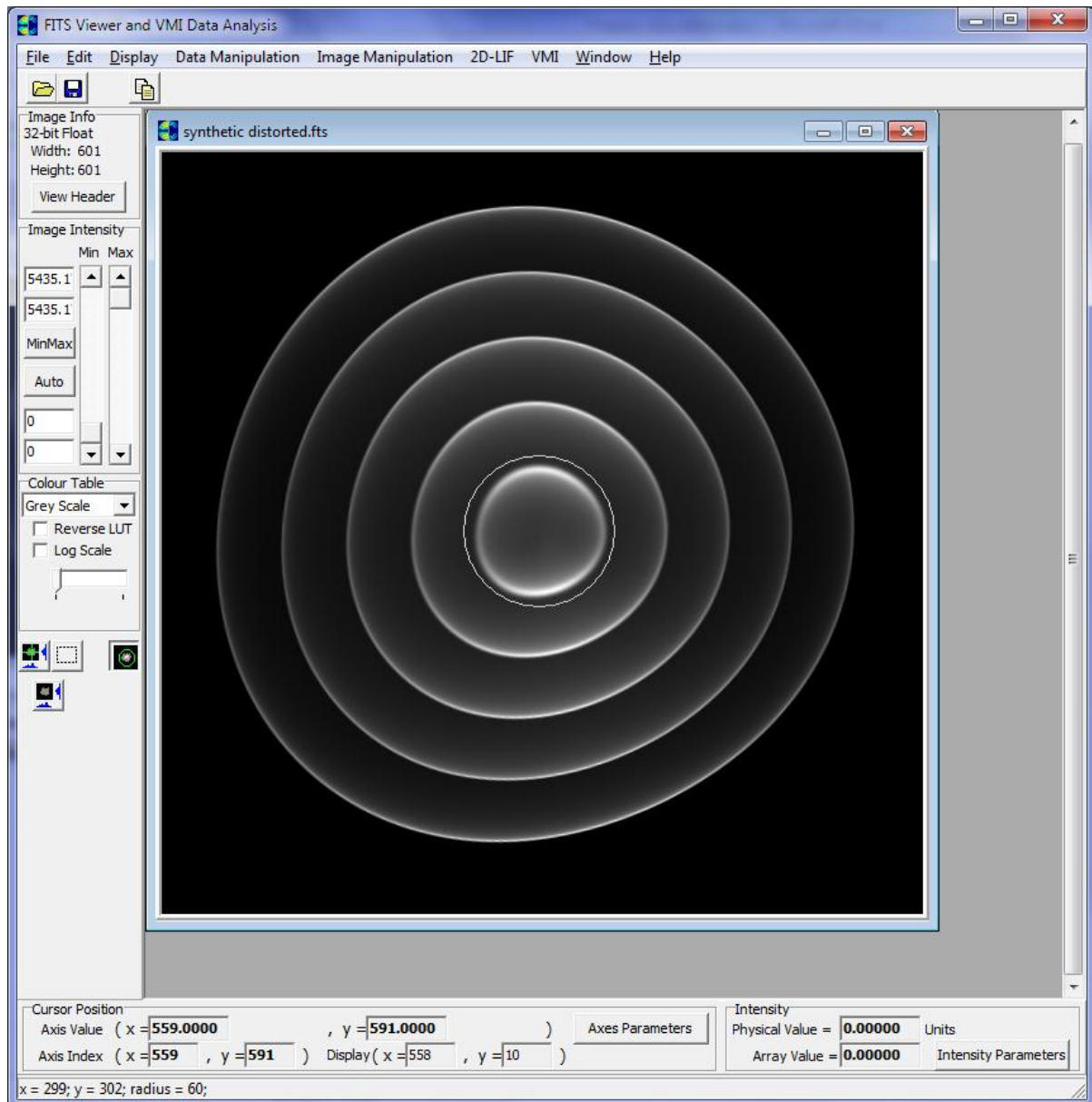
More generic information can be included via HISTORY, COMMENT and blank keywords that allow information about the image, including how it was generated to be saved within the FITS file.

The header must end with the END keyword.

## Circularisation of VMI images

In this section we will show how to apply circularisation to a very distorted synthetic VMI image.

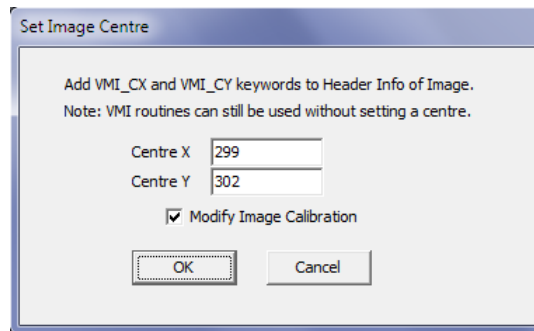
1) Determine a best guess estimate for the image centre. There are numerous methods that could be used, for example one could simply take the average of the ring coordinates at the top, bottom, left and right, or least squares fitting several points chosen around a ring to the equation of a circle. Alternatively a rough centre can be determined using the circle feature in the program and manually moving the circle position and radius such that it looks centred around a ring. This is done in the image below, and the centre of the circle (and radius) is given in the bottom status bar. See discussion of the side tool bar above for more information on the circle feature.



Only an approximate image centre is required to start with, since the circularisation algorithm returns a better estimate of the image centre.

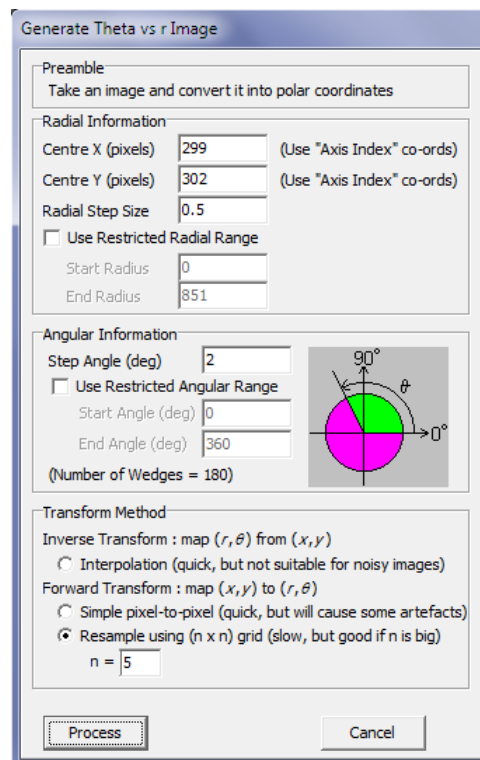
2) Set the centre of the image. This step is optional, but will save you having to constantly re-enter the centre in future processing steps. By setting a centre, two keywords (VMI\_CX and VMI\_CY) are added to the image header, and used in other VMI analysis. From the menu, select

VMI→Set Centre...



The above dialogue box appears and the image centre coordinates can be entered. Note that the coordinates do not have to be integers. A check box is also on the dialogue box that allows a calibration to be put on the image so that the  $x$  and  $y$  cursor coordinates displayed on the lower information bar are relative to the image centre.

3) Generate a polar coordinate plot of the distorted image. Select from the menu VMI→Create Polar Coordinate Image...



If an image centre was set in Step 2, then it is automatically populated in the window, otherwise default values are used and you will need to manually enter the centre. The step sizes (bin sizes) for the radial and angular coordinate can be changed. A radial step size of 0.5 pixels and angular step of 2 degrees generally gives a good image. There are 3 transformation methods to choose from:

Inverse Transform – Interpolation :

For each  $(r, \theta)$  pixel on the polar coordinate image, the corresponding  $(x, y)$  value on the original image is determined by 2-dimensional linear interpolation of the original image. The value is scaled by the Jacobian of the transformation to account for the larger  $drd\theta$  area as  $r$  increases. However, even with this Jacobian applied, interpolation can result in the integrated intensity of the polar coordinate image being different to the original image (in principle, it

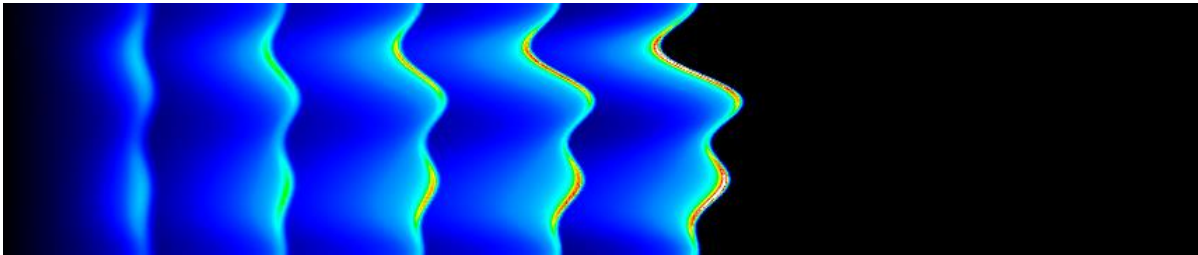
should be the same – you can't lose electrons!). This method is quick, and provides a smooth transform if the original image is smooth. It gives bad results on large noisy images.

#### Forward Transform – Simple pixel-to-pixel :

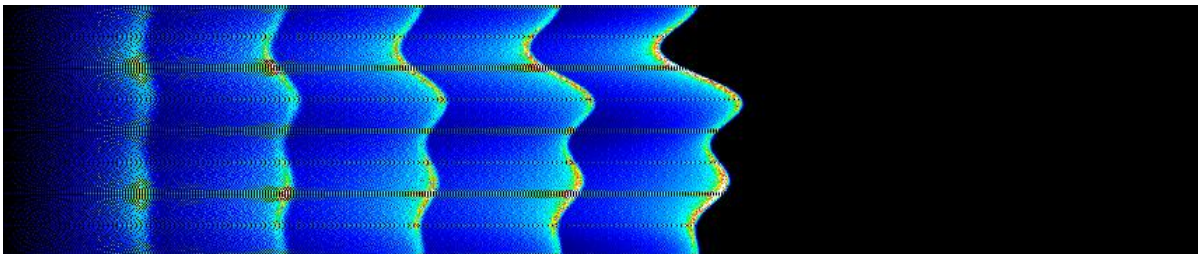
For each  $(x, y)$  pixel on the original image, the corresponding  $(r, \theta)$  polar coordinate is determined and the intensity of the  $(x, y)$  pixel is copied over to the  $(r, \theta)$  pixel in the polar coordinate image. This algorithm is a quick and preserves the integrated intensity of the original image. However, the algorithm will introduce artefacts due to pixellation, where some  $(r, \theta)$  pixels in the polar coordinate image don't have corresponding pixels in the  $(x, y)$  original image.

#### Forward Transform – Resample using $(n \times n)$ grid:

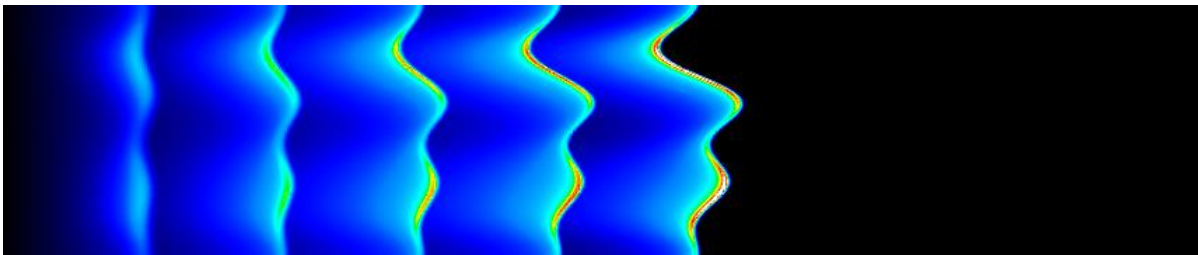
This algorithm is similar to the simple pixel-to-pixel algorithm above except that each  $(x, y)$  pixel is divided into  $n^2$  squares, i.e. a  $(n \times n)$  grid, and each sub-pixel is mapped separately to the polar coordinate image. Each sub-pixel is assumed to have the same intensity, which is  $1/n^2$  times the intensity of the  $(x, y)$  pixel. This algorithm will be  $n^2$  times slower than the previous algorithm, but the artefacts will be reduced. This is the preferred algorithm, but it is up to the user to use an appropriate value of  $n$  as a compromise between speed and reducing artefacts.



Inverse Transform – Interpolation



Forward Transform – Simple pixel-to-pixel



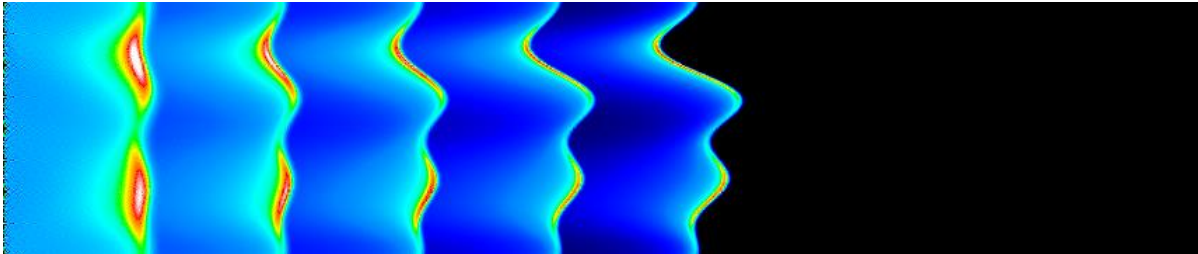
Forward Transform – Resample using  $(10 \times 10)$  grid

The above images are polar coordinate image of the synthetic VMI image using the different transformation methods. The horizontal direction represents the radius, and the vertical direction represents the angle. The images produced are calibrated, so you use the cursor feature to directly read off  $(r, \theta)$  values from the image. Note that the lower left corner is the origin, which in this example corresponds to a  $(r, \theta)$  calibrated value of  $(0.25, 1.0)$ . That is, for this pixel,  $r$  ranges from  $0 \rightarrow 0.5$  and  $\theta$  ranges from  $0 \rightarrow 2$ , as given by the step sizes used in the transformation.

4) Divide intensities in the polar coordinate image by  $r$ . This is necessary since we desire the radial function along a line emanating from the centre of the original distorted image, whereas currently the polar coordinate image displays at each radius the integrated intensity over an annulus sector defined by the radial and angular step size. The Jacobian  $dx dy = r dr d\theta$  means we have to divide the intensities in the polar coordinate image by  $r$ .

From the menu, select VMI→Divide by  $r$  Polar Coordinate Image...

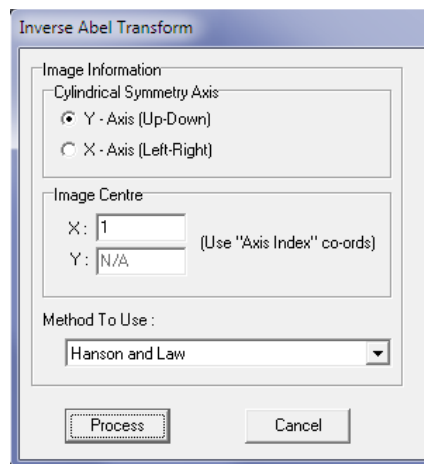
Applying this to the polar coordinate image using the  $(10 \times 10)$  grid we get the scaled version:



Note that this image has the same calibration as the original polar coordinate image.

5) Apply the Inverse Abel Transform. This process will convert the above image into an image that has more Gaussian-like peaks to aid in the fitting process. This transform will only return the correct velocity distribution for two angles only (the two angles perpendicular to the laser polarisation), but is useful for fitting the angular dependence of the distortions.

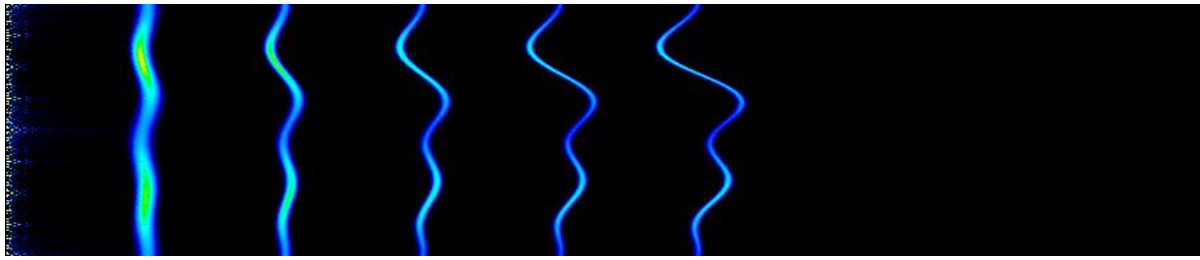
From the menu, select VMI→Inverse Abel Transform...



For this step the cylindrical symmetry axis is the leftmost column of pixels, so Y-Axis (Up-Down) must be selected. Since the window requires the “Image Centre” to be in FITS “Axis Index” coordinates, then X has to be set to 1 (this is the first column of pixels). Note that the calibrated  $r$  value cannot be used. There is currently only one method of performing the Inverse Abel Transform, that being the method described by Hanson and Law.

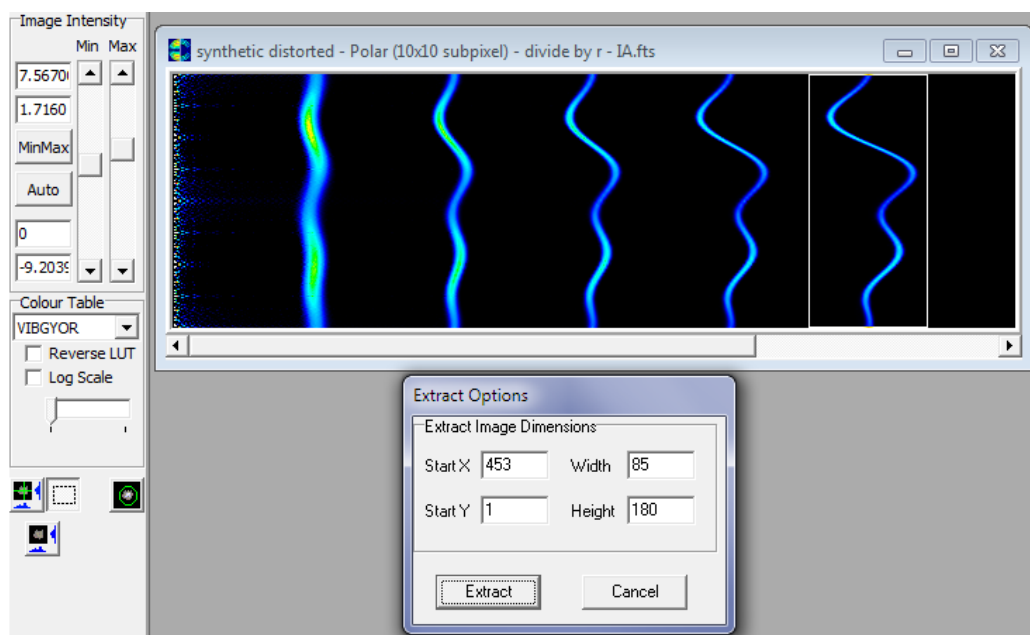
After applying the Inverse Abel Transform to the scaled radial plot image we get:





Note that the polar coordinate calibration remains on the image. Also the algorithm produces a lot of noise near  $r = 0$ , some of which is negative, thus you will need to adjust the lower display intensity to 0.

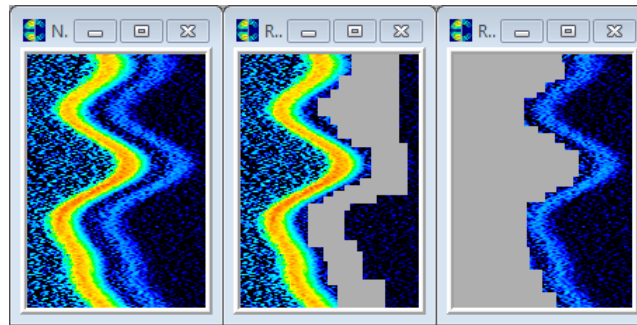
6) Extract each line (wiggly line). The inbuilt fitting algorithm can only fit Gaussian wiggly line at a time. Therefore each line needs to be extracted and analysed individually. The easiest way to do this is to press on the rectangular selection tool and draw rectangle over the lines you wish to extract. You can look at the coordinates in the lower information bar to make sure you start the rectangle at pixel  $y = 1$  and draw a rectangle to the full height. (It doesn't matter if you can't get the cursor positioned exactly right because you can manually enter the coordinates in the Extract Options dialogue box.) Once the rectangle has been drawn, select from the menu Image Manipulation → Extract Region...



The values in the dialogue box correspond to those given by the rectangular selection on the image. To extract the entire angular range, make sure the “Start Y” is set to 1, i.e. the first  $\theta$  value, and the “Height” is set to the height of the image (as given at the top of the left hand tool bar). After pressing “Extract” the selected region becomes a separate image, and, importantly, is still calibrated correctly.



7) Mask unwanted data. It may be impossible to cleanly select a rectangular region without interference from adjacent rings. In these cases, it is necessary to invalidate data that belongs to other rings. Select the rectangular selection tool and draw a rectangle over the data that you wish to invalidate. From the menu select Data Manipulation→Invalidate Selection. The pixels within the selection are converted to Not A Number (NaN) and are displayed as grey on the image. Invalid pixels are ignored during the next fitting step. It will be likely that many small rectangular regions are necessary to invalidate the unwanted feature. As an example, below left is an image that contains a strong and weak peak that cannot be separated using a simple rectangle. By invalidating just the right hand wiggle we get the centre image and by invalidating the left hand wiggle we get the right-hand image. These masked images can now be fitted separately to determine the coefficients of each wiggle.



8) Fit the wiggly lines to a trigonometric series. A wiggly line in a polar coordinate image can be fitted to a Gaussian function where the position of the Gaussian varies with angle. The functional form used in this program is:

$$I(r, \theta) = I(\theta) \exp\left(-\frac{(r - r_g)^2}{2\sigma^2}\right) + c$$

where  $r_g$  is the angle dependant centre radius (position) of the Gaussian given by:

$$r_g(\theta) = r_0 + \sum_{n=1}^N (A_n \sin n\theta + B_n \cos n\theta)$$

where  $r_0$  is the average radial position of the wiggly line,  $N$  is the number of trigonometric terms to include. The angular intensity is given by the standard anisotropy function:

$$I(\theta) = I_0 \left(1 + \frac{1}{2} \beta (3 \cos^2(\theta) - 1)\right)$$

From the menu, select VMI→Trig Fit Polar Coordinate Image...



Fit Theta vs r Image

Fit Information

Num Trigonometric Terms

Fitting Parameters  $r_c = r_0 + \sum (A_n \sin(n\theta) + B_n \cos(n\theta))$

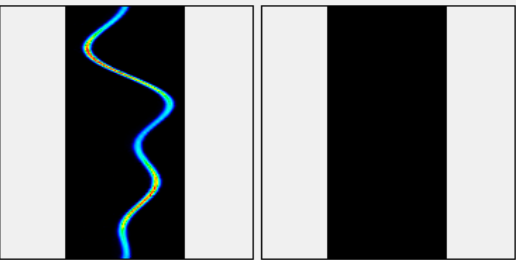
Parameter	Value	Fit	Uncertainty
Intensity	1	<input checked="" type="checkbox"/>	
Anisotropy ( $\beta$ )	0	<input checked="" type="checkbox"/>	
Offset	0	<input checked="" type="checkbox"/>	
Std Dev	5	<input checked="" type="checkbox"/>	
Position ( $r_0$ )	0	<input checked="" type="checkbox"/>	
$\sin(\theta)$	0	<input checked="" type="checkbox"/>	
$\cos(\theta)$	0	<input checked="" type="checkbox"/>	
$\sin(2\theta)$	0	<input checked="" type="checkbox"/>	
$\cos(2\theta)$	0	<input checked="" type="checkbox"/>	

Auto Guess Calculate Copy To Clipboard

☒ Iteratively Fit Trig Terms Save To File

Fit Image

☒ Keep Calc Image Close



The left hand side of the form contains controls to perform the non-linear least squares fitting. The left hand image is the image to be fit and the current best fit (currently blank) is on the right. Change the number of trigonometric terms desired for the fit by entering the number in the top box. Initial starting values can be entered directly, and then pressing “calculate” to update the fit image, or, a more convenient way is to press the “Auto Guess” button. The auto guess provides estimates for the intensity, standard deviation and position of the Gaussian function, while setting all other parameters to zero. (In the current example, the line is so wiggly that the guess is far from perfect, but will be adequate as starting parameters.)

Fit Theta vs r Image

Fit Information

Num Trigonometric Terms

Fitting Parameters  $r_c = r_0 + \sum (A_n \sin(n\theta) + B_n \cos(n\theta))$

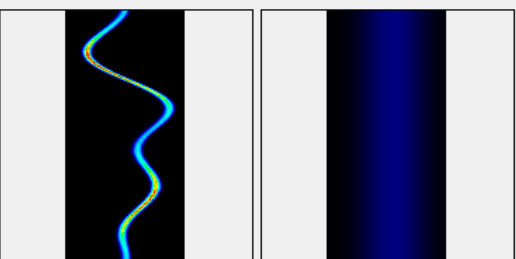
Parameter	Value	Fit	Uncertainty
Intensity	0.0354633156	<input checked="" type="checkbox"/>	
Anisotropy ( $\beta$ )	0	<input checked="" type="checkbox"/>	
Offset	0	<input checked="" type="checkbox"/>	
Std Dev	8.2118740511	<input checked="" type="checkbox"/>	
Position ( $r_0$ )	249.85357386	<input checked="" type="checkbox"/>	
$\sin(\theta)$	0	<input checked="" type="checkbox"/>	
$\cos(\theta)$	0	<input checked="" type="checkbox"/>	
$\sin(2\theta)$	0	<input checked="" type="checkbox"/>	
$\cos(2\theta)$	0	<input checked="" type="checkbox"/>	

Auto Guess Calculate Copy To Clipboard

☒ Iteratively Fit Trig Terms Save To File

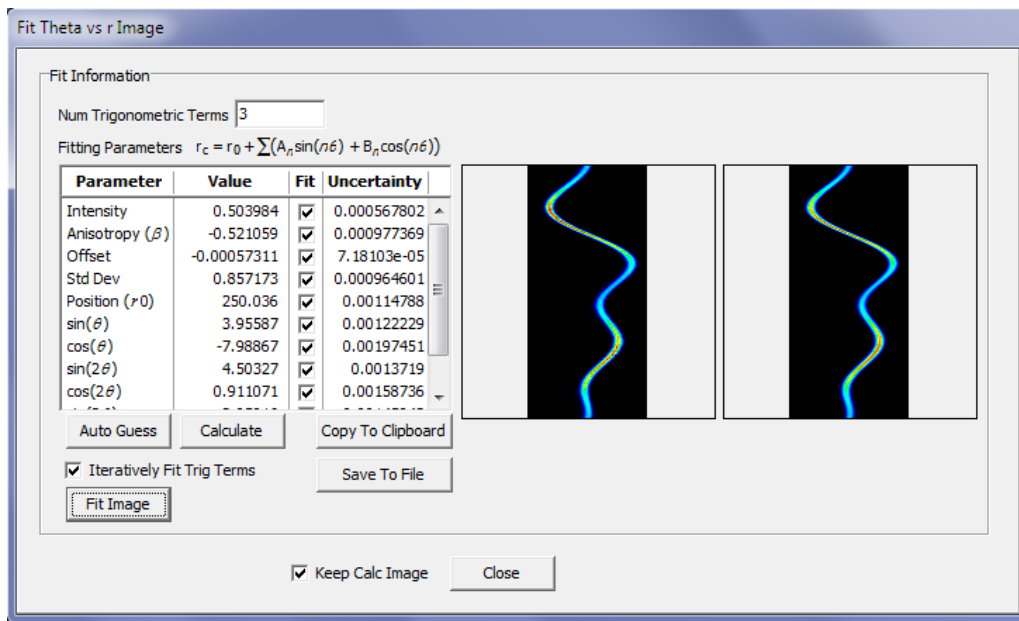
Fit Image

☒ Keep Calc Image Close



To perform the fit, press the “Fit Image” button. When the initial guesses for the trig terms are zero, it is recommended to select the “Iteratively Fit Trig Terms” checkbox. This causes the fitting algorithm to first set all trig terms to zero and then performs a fit with only 1 trig term (i.e.  $\sin(\theta)$  and  $\cos(\theta)$  terms). The result of the fit is used as the initial parameters for a subsequent fit that includes 2 trig terms (i.e.  $\sin(\theta)$ ,  $\cos(\theta)$ ,  $\sin(2\theta)$  and  $\cos(2\theta)$  terms). This process is repeated by iteratively rolling in higher order terms until the max number of desired terms is reached.

If there are values that don't need to be fitted, for example if one wants to force a particular trig term to be zero, then uncheck the box next to the fitted value.

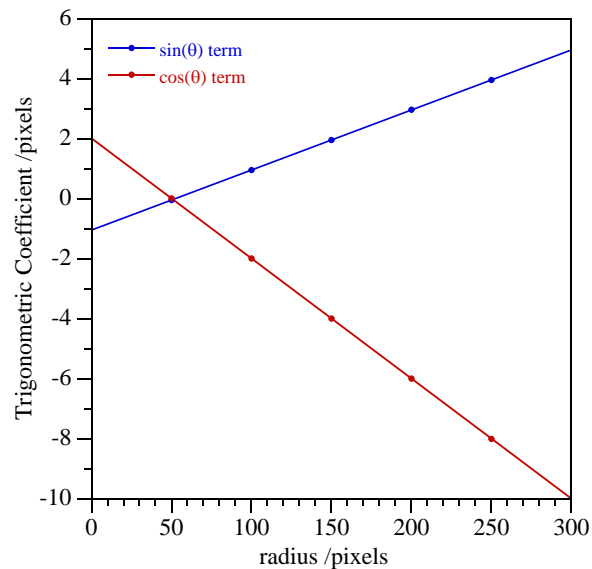


After fitting, the best-fit values and the  $1\sigma$  uncertainty returned from the non-linear least squares algorithm is displayed. This can be copied to the system clipboard (by pressing “Copy to Clipboard”) for pasting into a spreadsheet. Alternatively the current best-fit values can be saved to a file.

To determine the max number of trig terms required, the size of the value should be compared with the uncertainty. A large uncertainty indicates that the term is ill-determined and not required.

If you desire to keep the best fit image, place a tick mark next to “Keep Calc Image” before pressing the “Close” button. This fit image contains in the image header information regarding the best-fit parameters.

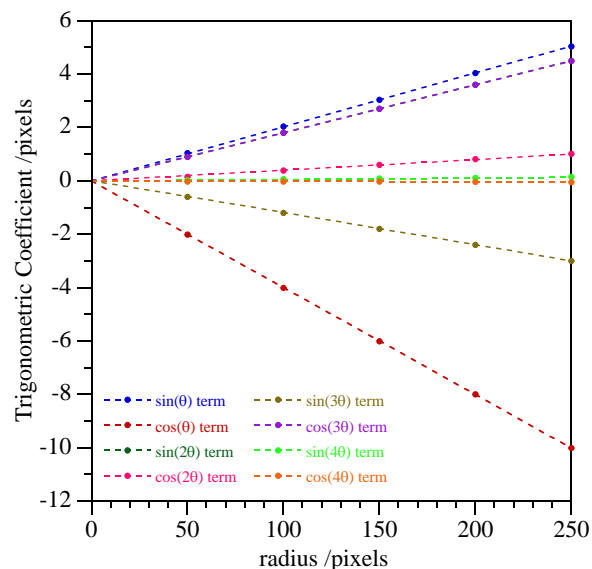
9) Plot the coefficient for the  $\sin(\theta)$  and  $\cos(\theta)$  terms to determine a better image centre. By determining the trigonometric coefficients of several rings at different radii, a plot of the  $\sin(\theta)$  and  $\cos(\theta)$  terms as a function of radius can be used to determine a better centre for the distorted image. Since at  $r = 0$  the angular deformation vanishes, extrapolating these plots to  $r = 0$  gives the trig term values arising from the incorrect choice of centre and hence the correction required.



From the above graph, a linear fit results in an intercept for the  $\sin(\theta)$  term of  $-1.04$  pixels and a  $\cos(\theta)$  intercept of  $2.01$  pixels, meaning that you need to add  $(2.01, -1.04)$  to the current centre. Since the initially chosen centre in Step 1 was  $(299, 302)$ , a more refined determination of the centre is  $(301.01, 300.96)$ . This is very close to the actual centre of  $(301, 301)$ .

10) Repeat steps 2 to 9 using the updated centre location. The above procedure should be repeated with the improved centre and repeated until the plots of  $\sin(\theta)$  and  $\cos(\theta)$  terms versus radius have intercepts of zero within the fitting uncertainty.

11) Plot the coefficients for each of the trigonometric terms as a function of radius. The circularisation procedure implemented in this program assumes that the distortion is linear with radius. The program requires the slope of plot for each trigonometric coefficient versus radius. It is best to fit a linear function that is forced to go through zero, since there can't be any distortion at  $r = 0$ .



12) Circularise the image. After the slopes in the graph above are determined, the distorted image can be circularised. The following functional form is used to characterise the distortion:

$$r_d(\theta) = r_u \left( 1 + \sum_{n=1}^N (a_n \sin n\theta + b_n \cos n\theta) \right)$$

where  $r_d(\theta)$  is the radius of a pixel on the distorted image,  $r_u$  is the corresponding corrected (undistorted) radius and the coefficients  $a_n$  and  $b_n$  are the slopes from the plots obtained in Step 11. The circularisation procedure involves rearranging this equation to determine correct radius,  $r_u$ , for the pixels on the distorted image. Select the distorted image and from the menu select VMI→Circularise Image...

**Circularise Image**

**Circularisation Information**

Centre X (pixels)  (Use "Axis Index" co-ords)

Centre Y (pixels)  (Use "Axis Index" co-ords)

Num Trigonometric Terms

Distortion Parameters:

$$r_{old} = r_{new} \left( 1 + \sum (a_n \sin(n\theta) + b_n \cos(n\theta)) \right)$$

Parameter	Value
$\sin(\theta)$	0.020223853
$\cos(\theta)$	-0.040055312
$\sin(2\theta)$	0.017998415
$\cos(2\theta)$	0.004026888
$\sin(3\theta)$	-0.011989696
$\cos(3\theta)$	0.017981164
$\sin(4\theta)$	0.000613197
$\cos(4\theta)$	-0.000172895

**Transform Method**

Inverse Transform : map new image from distorted image

☐ Interpolation (quick, but not suitable for noisy images)

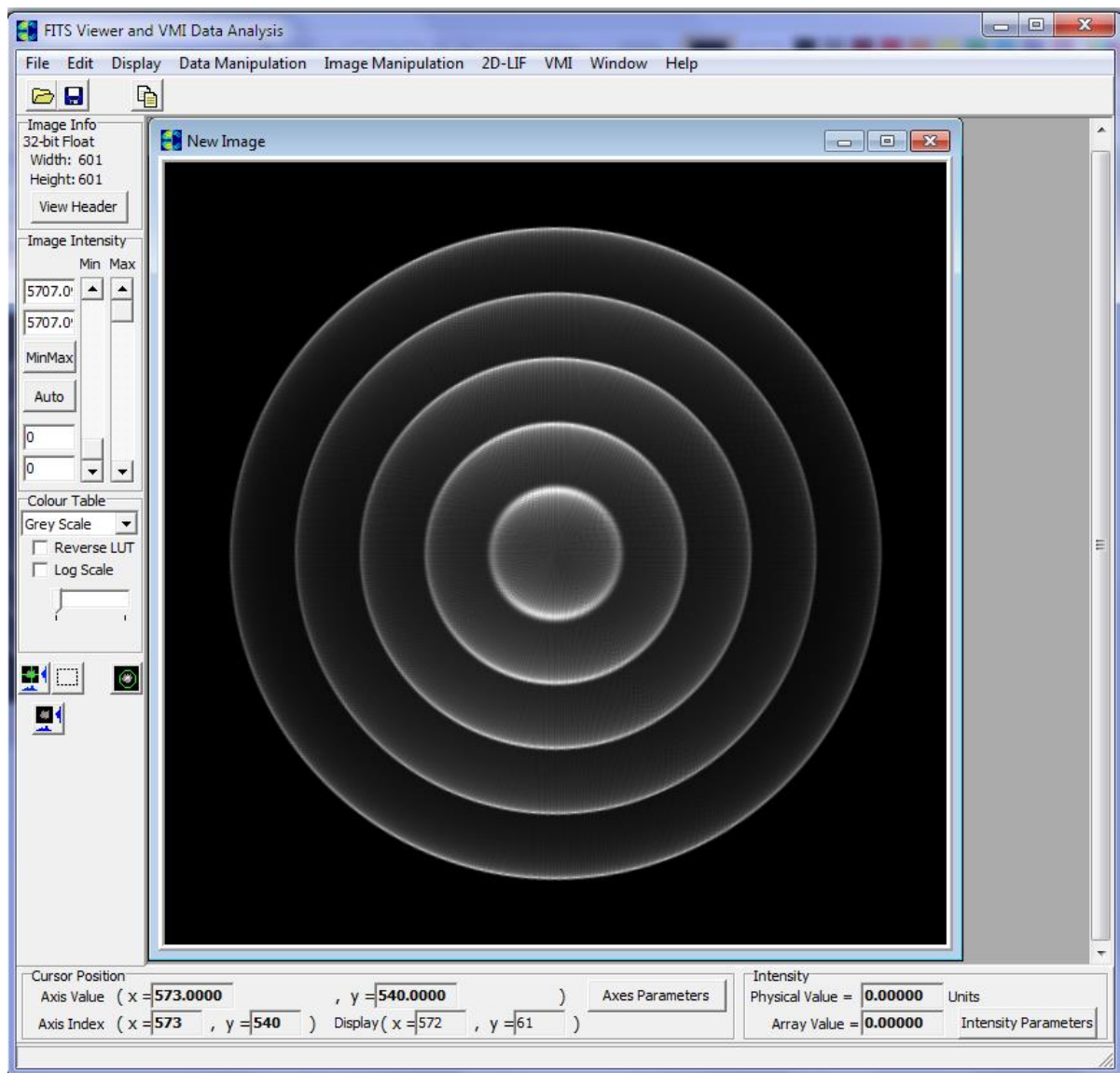
Forward Transform : map distorted image to new image

☐ Simple pixel-to-pixel (quick, but will cause some artefacts)

☒ Resample using (n x n) grid (slow, but good if n is big)

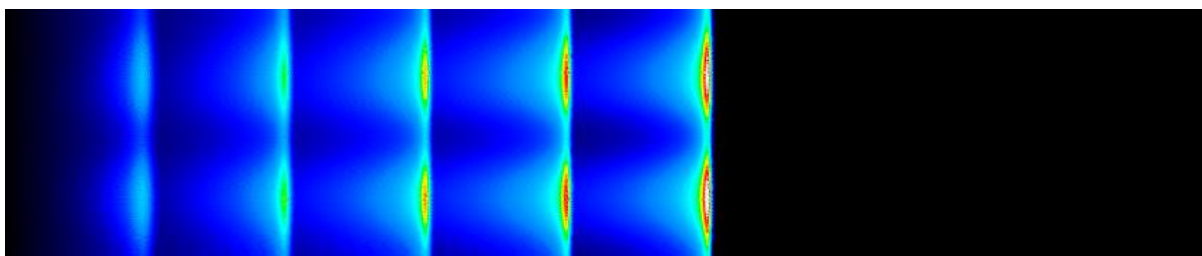
n =

Set the centre  $x$  and  $y$  values, and the number of trigonometric terms you have used in the analysis. Enter all the  $a_n$  and  $b_n$  terms. This can be done manually, or conveniently, by pasting from the system clipboard. Different transformation methods are available and were discussed in Step 3. The circularised test image is shown below.



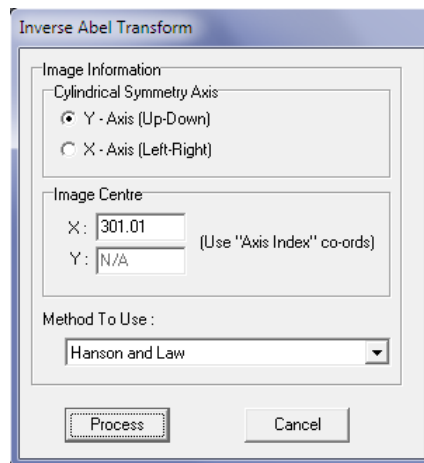
After the image has been generated, you may wish to set the centre of the image using the menu VMI→Set Centre...

13) Confirm the circularity of the image. It is a wise idea that after the image has been circularised, that a polar coordinate plot is generated to ensure there are no wiggly lines present:

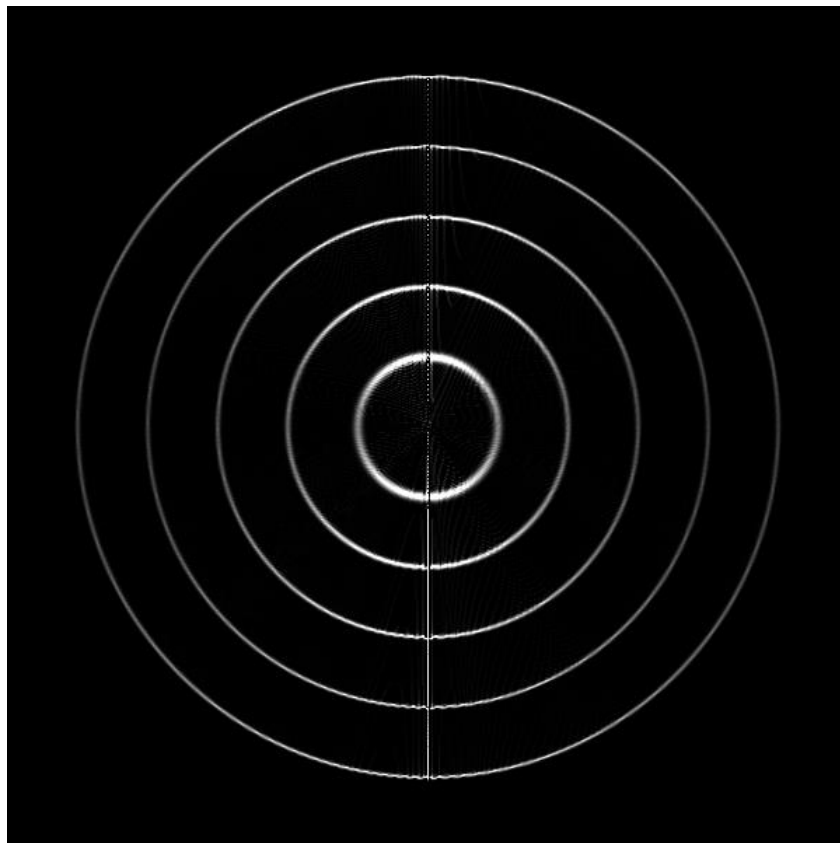


If all the features in a polar coordinate plot are straight vertical lines then the deformed image has been circularised successfully. At this point the corrected image can be saved and used as input for standard VMI analyses that you may already use and are familiar with. Alternatively, you can use the analysis tools in this program as detailed below.

14) Perform Inverse Abel Transform of the corrected image. Select the undistorted image and from the menu select VMI→Inverse Abel Transform...

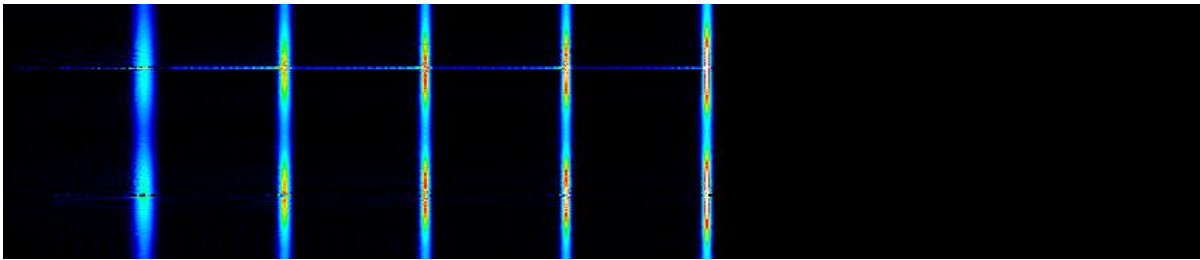


Choose the cylindrical symmetry axis that matches your experimental configuration. If the laser polarisation is vertical in the image, then select Y-Axis and enter the centre  $x$  position of the image (the position may be prefilled if you have set the centre in the image). For a laser polarisation that is horizontal in the image, select X-Axis and enter the centre  $y$  position of the image. There is currently only one method of performing the Inverse Abel Transform, that being the method described by Hanson and Law.

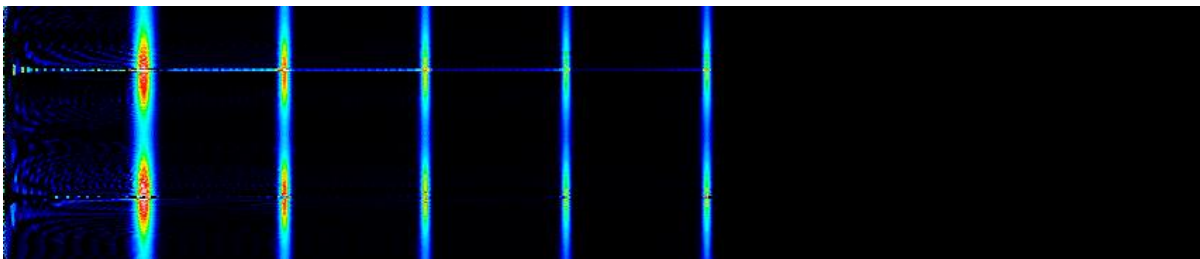


After the image is generated you may (will) need to adjust the display intensities. The centre stripe noise contains large positive and negative values which results in a low contrast image if using min/max values for the intensity display. It is suggested to set the lower intensity to 0 and reduce the max intensity until the image looks acceptable.

15) Generate polar coordinate plot of the Inverse Abel Transform image. Follow the procedure outlined in Step 3 to generate the polar coordinate image.

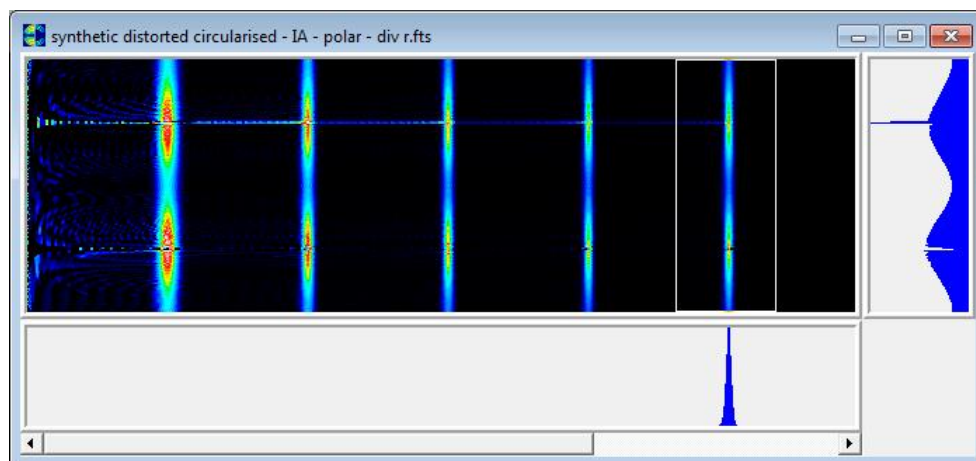


16) Divide the intensities by  $r$ . Since the polar coordinate transformation integrates the original image over an annulus sector defined by the radial and angular step size, the intensities require dividing by  $r$  to obtain the radial distribution function of the Inverse Abel Transformed image. (See Step 4 for further details.) From the menu, select VMI→Divide by  $r$  Polar Coordinate Image...



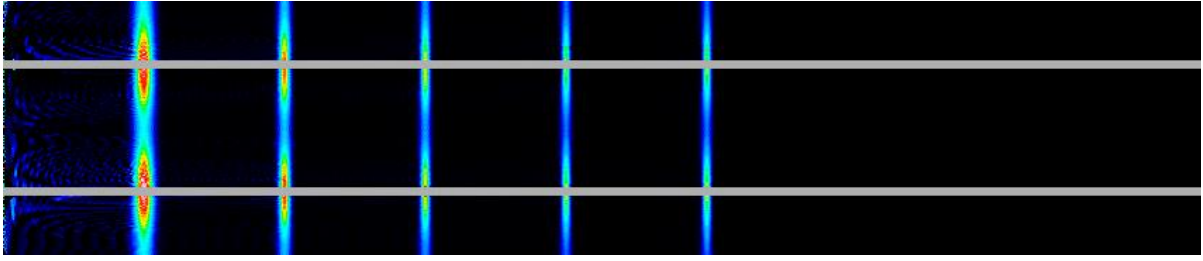
(Note that this divide-by- $r$  step could be done following extraction of the spectrum instead)

The centre noise strip in the Inverse Abel Transform image results in horizontal lines in the polar coordinate image the angle of the laser polarisation ( $90^\circ$  and  $270^\circ$  in the current example). By using the rectangular selection tool and the graph (projection/cross-section) tool the extent of the noise becomes apparent.

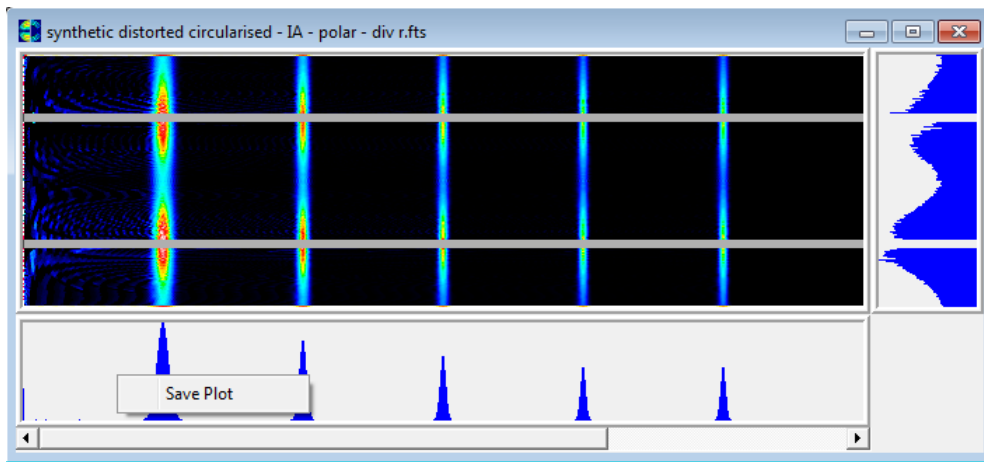


17) Invalidate the horizontal noise pixels. Use the rectangular selection tool to invalidate desired pixels as outlined in Step 7.



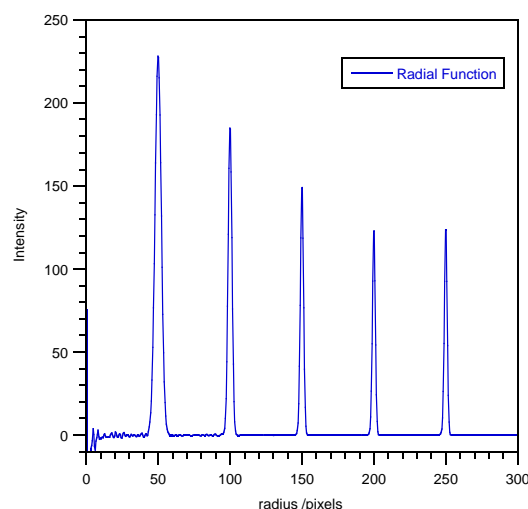


18) Extract the radial summed radial intensity. The radial part of the probability distribution function is generated by vertically integrating the polar coordinate plots, i.e. the projection onto the radial (horizontal) axis. Using the rectangular selection tool and the graph (projection/cross-section) tool, a rectangle can be drawn over the entire image to display the radial spectrum at the bottom. To save this spectrum, right click in the graph window and select “Save Plot”. Use the save dialogue window to save the spectrum. The spectrum is saved as a comma separated variable file where the first column is the FITS index value (1→width), the second column is the calibrated value ( $r$  in units of the original image pixels) and the final column is the intensity value.



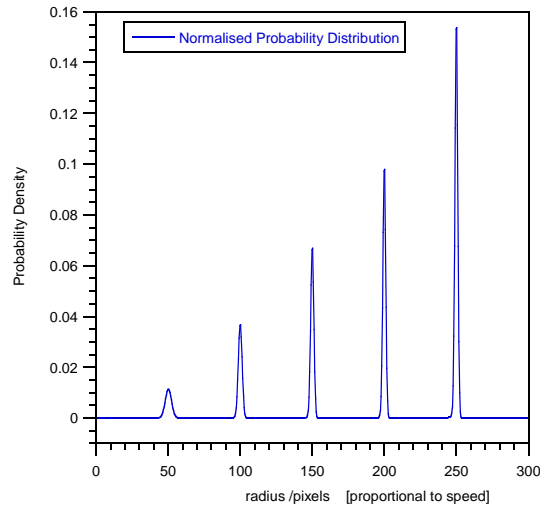
Note, in the above image the graph of the projection onto the  $\theta$  axis (i.e. right hand graph) is contaminated with the large amount of noise approaching  $r = 0$ .

19) Analyse the spectrum. The spectrum generated above is the radial function of the Inverse Abel Transform image:

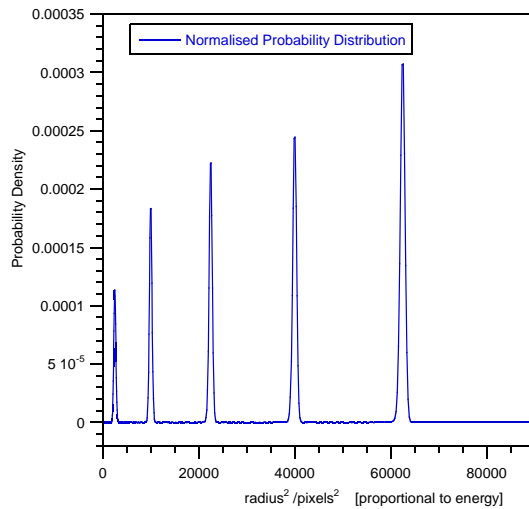




By integrating the radial function over the Newton sphere (ie multiplying the intensities in the spectrum by  $r^2$ ) one generates the total probability distribution as a function of  $r$ , and therefore a speed probability distribution. Since the spectrum is a probability distribution it should be scaled so the integral of the function is equal to 1.



Usually spectra are displayed on an abscissa that has energy units. The energy of the electron/ion is proportional to pixels squared, and since we are plotting a population distribution we need to divide the intensities in the spectrum by the Jacobian for the speed to energy transformation. That is, the intensities need dividing by  $2r$  to ensure the integral of each peak remains unchanged.



The last step in the analysis is to apply the energy calibration to the image (and scaling the intensity appropriately to ensure the integral is 1).