CORONAL DIAGNOSTIC SPECTROMETER
SoHO

CDS SOFTWARE NOTE No. 41

Version 2.0 August 10, 1996

Analyzing CDS Data in IDL: An Observers Guide

S. V. H. Haugan
Institute of Theoretical Astrophysics
University of Oslo

s.v.h.haugan@astro.uio.no

1 Introduction

This document is meant as a guide to those who need to analyze data from the Coronal Diagnostic Spectrometer aboard the Solar and Heliospheric Observatory. It is assumed that the reader is somewhat familiar with the design of observation rasters for the CDS. Also, this document is not meant as a substitute for the on-line documentation headers that accompany the routines described here, as well as all other CDS software. Use e.g., `xdoc,'gt_spectrum'` to display the documentation header for the routine `gt_spectrum`. To search for routines that may solve your problem, use e.g., `tftd,'scan'` to find routines with the string "scan" in the routine name or "purpose" line. Learning to use these routines and to read the routine documentation headers will save you a lot of time.

The basic building blocks in CDS studies are rasters. Data from each raster is stored in a separate FITS file. A CDS FITS file can be read into IDL through the function `readcdsfits()`, which returns a structure that contains all the information that is available in the FITS file. The returned structure is called a Quick Look Data Structure (`QLDS`). The spectral information in the file is stored as handle values, and it is *not* recommended to access the data directly. Instead, use the functions described in this (or other) documents as an interface. This will shield your programs from future changes to the data structure, and some time in the future, you might find out your programs will suddenly work with SUMER data as well as CDS data, with little or *no* changes!.

This document describes the routines that are available to pick out (deselect) specific parts of the detector data from the raster into something that is (or should be) possible to analyze for anyone with an interest in doing so.

2 General Overview

Extracting pieces of data from a raster requires the identification of the data to be extracted. In general, a CDS raster covers an area of the sun (or some part of the corona) that is divided into from 1 to 120 pixels in the solar cartesian X direction, and from 1 to about 120 pixels in the solar cartesian Y direction. Generally, the raster contains (parts of) a spectrum for each of these spatial pixels (see Fig. 1). In order to extract the spectrum from a given point in the raster area, it is necessary to specify the position through the use of the X and Y indices, (`XIX` and `YIX`).

Note that if the scan mirror step size is set to 0, but the number of scan mirror steps is more than one, the raster is effectively a time-series (see Fig. 2). The raster area then covers from 1–120 pixels in the solar Y direction, depending on the number of slit positions (for the GIS) or the height in pixels of the extraction windows (for the NIS).

Note that in this case, the *time index* (`TIX`) and the Y index (`YIX`) have to be specified when referring to one specific spectrum. The X index (`XIX`) need not be specified in this case since there is only one allowed value (zero). In those cases where there is only one pixel in the Y direction, the Y index (`YIX`) need not be specified either.

In order to extract parts of the spectrum from the raster, it is necessary to specify *what* part of the spectrum as well. The Normal Incidence Spectrograph (NIS) detector has two wavelength bands, number 1 and 2 (see Fig. 3). The Grazing Incidence Spectrograph (GIS) has four wavelength bands, numbers 1 through 4 (see Fig. 4).

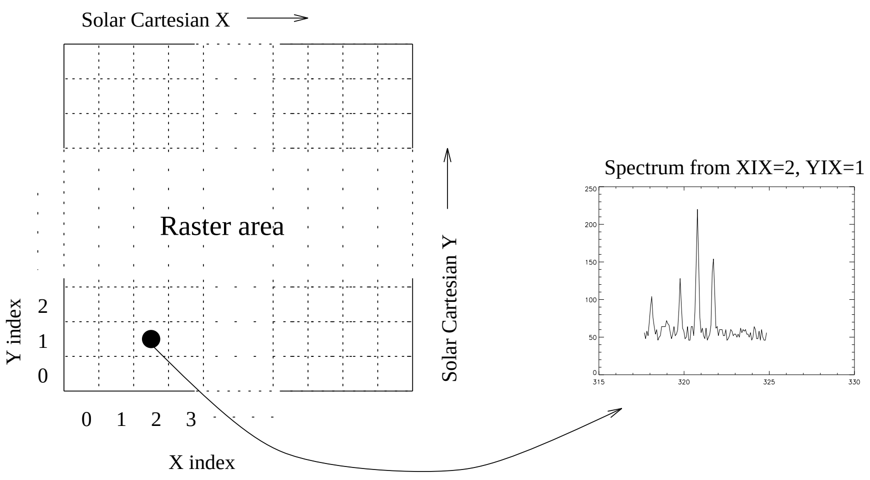

Figure 1: The raster area, with a spectum at each point.

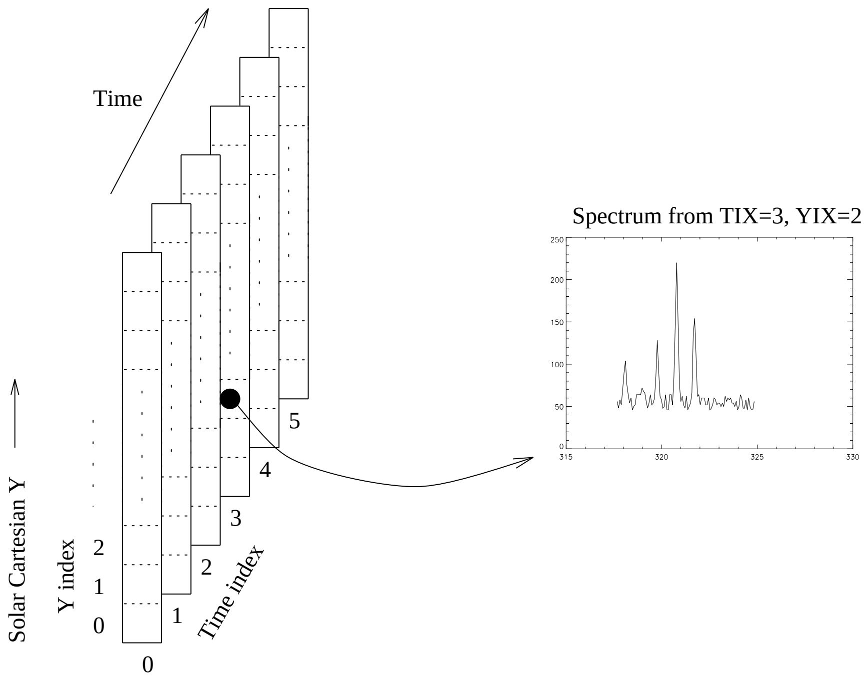

Figure 2: A typical time-series raster

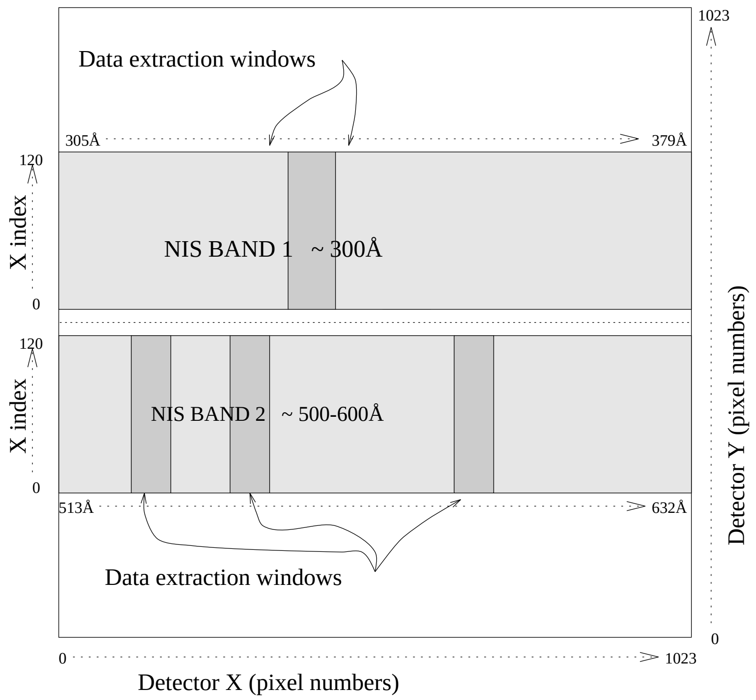

Figure 3: The Normal Incidence Spectrograph (NIS) detector.

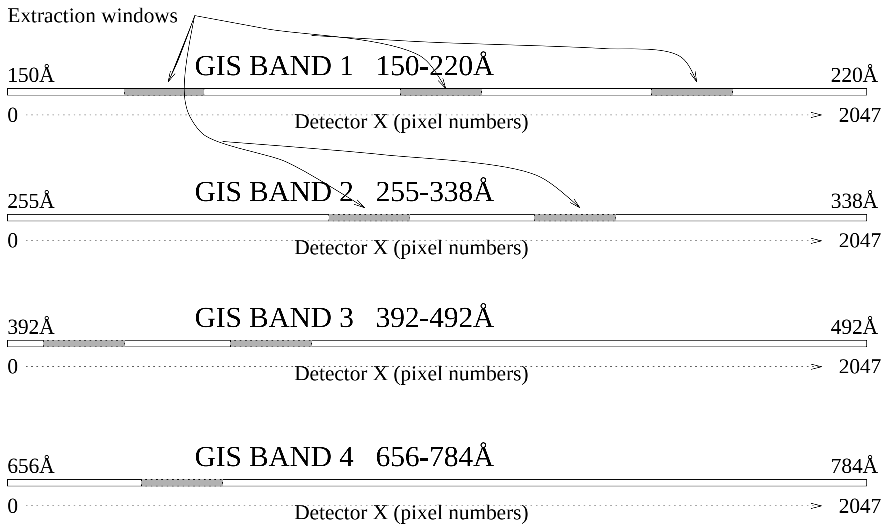

Figure 4: The Grazing Incidence Spectrograph (NIS) detectors.

The simplest way to refer to specific parts of the spectrum is to specify which wavelength band (`BAND`) to extract. The other way is to specify what data extraction window(s) (`WINDOW`) to return. The data extraction windows are defined by `mk_raster` in the design of the raster.

To specify a unique pixel position in the dispersion direction, it is possible to use *either* the wavelength band `BAND` *and* the detector X pixel number (`DETX`), *or* the window name/index (`WINDOW`) *and* an offset (`OFFSET`) within that window (see Fig. 5).

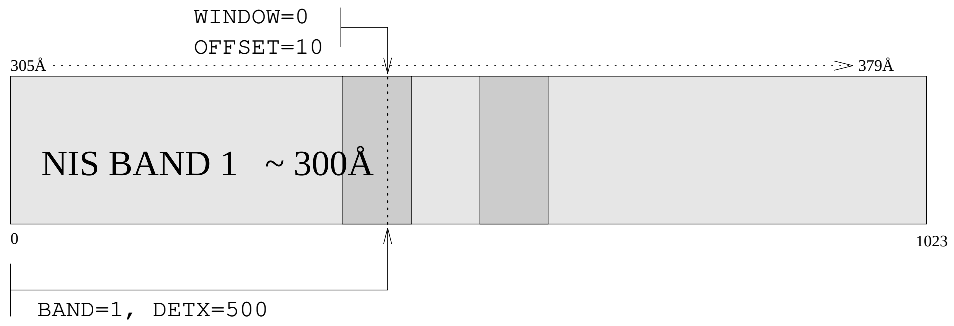

Figure 5: A closeup of the NIS detector, band 1. The combinations (`BAND=1, DETX=500`) and (`WINDOW=0, OFFSET=10`) in this case refer to the same detector pixel number (along the dispersion direction). GIS pixel numbers are specified in the same way

3 Extraction routines (gt_xxx routines)

3.1 Input keywords

The routines use keywords extensively both for input and output parameters, and they have a common nomenclature for all the keywords that they use. The input keywords are:

`XIX` The X index (see Fig. 1)

`YIX` The Y index (see Fig. 1 and 2).

`TIX` The Time index (see Fig. 2).

`BAND` Specifies the wavelength band, (1 or 2 for the NIS, 1–4 for the GIS)

`DETX` Detector X pixel number.

`WINDOW` Specifies the window name(s) or index(-ices).

`OFFSET` Specifies the detector X pixel number relative to the beginning of the specified data extraction window.

3.2 Routine overview

The extraction routines have been designed so that the number of dimensions and the physical interpretation of the dimensions of the return value is always fixed for a given routine. Below is a brief description of each of them. They all take a QLDS as their first argument.

gt_spectrum Returns a one-dimensional array with counts/intensities for a given point (`XIX`, `YIX`, and `TIX` have to be specified, but may be omitted if the only permitted value is zero). `WINDOW` (may be an array) or `BAND` must be specified.

gt_scanx Returns a two-dimensional "slit spectrogram" with spectra along a horizontal line on the sun. The first dimension is the dispersion, and the second is in the solar cartesian X direction. `YIX` (and possibly `TIX`) must be specified when applicable. `WINDOW` or `BAND` must be specified.

gt_scany Returns a two-dimensional "slit spectrogram" with spectra from a vertical line on the sun. First dimension is the dispersion, and the second is along solar cartesian Y. `XIX` and `TIX` must be specified when applicable. `WINDOW` or `BAND` must be specified.

gt_scant Returns a two-dimensional "slit spectrogram", but for one given point on the sun, with the hypothetical slit in the "time direction". First dimension is dispersion, and the second is time. `XIX` and `YIX` must be specified when applicable. `WINDOW` or `BAND` must be specified.

gt_scanp Returns a two-dimensional "slit spectrogram", with the "slit" covering arbitrary points selected by the input keywords. First dimension is dispersion, and the second may be a combination of solar cartesian X and Y, or time. `XIX`, `YIX` and `TIX` should be specified when applicable, and should be arrays such that (`XIX(i),YIX(i),TIX(i)`) refer to the points to be included in the spectrogram. `WINDOW` or `BAND` must be specified.

gt_iimage Returns a two-dimensional image taken at a specific detector X (dispersion) pixel. The pixel must be specified with *either* (`BAND`, `DETX`) *or* (`WINDOW`,`OFFSET`. Return value dimensions are solar cartesian X and Y (in that order).

gt_mimage Returns a two-dimensional image taken at a specific wavelength, supplied as the second parameter. Linear interpolation between neighboring detector pixels is performed. Return value dimensions are solar cartesian X and Y (in that order).

gt_bimage Returns a two-dimensional image, with intensities integrated between two wavelengths (2nd and 3rd parameters). Linear interpolation between pixels is used. Return value dimensions are solar cartesian X and Y (in that order).

gt_windata Returns the complete array of detector counts associated with one detector window. If you are using all of the data associated with a detector window (like averaging the spectra over one or more dimension), or if you would like to "jump around" at different positions in the raster very quickly, this is by far the best routine. You should have a good idea about what the data looks like before you use it, though.

3.3 The /NODESELECT switch

All routines that return spectra or spectrograms will deselect the data into an array of the same length as the detector band whenever the `BAND` keyword is used to specify what part of the spectrum to be returned. This behaviour can be turned off, by setting the keyword `/NODESELECT`. This causes the data to be packed into a contiguous array, without empty gaps inbetween.. `NODESELECT` is the default when using `WINDOWS` to specify the parts to be extracted.

3.4 Output keywords

The `gt_xxx` routines use output keywords to return physical values for solar coordinates, time, and wavelengths for the retrieved data. The values returned in these keywords are always *either* scalars, if the physical entity is constant over the whole returned dataset, *or* of the same dimension as the returned data, in cases where the physical values vary. Note that even though the values may be constant over *one* of the returned dimensions, it is always "blown up" to the same size as the function return value. The output keywords are:

`XSOLAR` The solar cartesian X coordinate (in arcseconds).

`YSOLAR` Solar cartesian Y coordinate (arcseconds).

`TIME` The observation time (TAI format).

`LAMBDA` The wavelength in angstrom.

The routine `gt_windata` does *not* supply any physical information through output keywords.

3.5 Speeding it up

The calculation of the auxiliary data returned through the output keywords consumes more time than the actual extraction of data. If you have no need for these auxiliary data, use the keyword `QUICK` to turn these calculations off. Not *all* routines take this keyword yet, but more will follow.

Also, if you find yourself writing a program that loops over e.g., `XIX`, `YIX`, and/or `TIX` to get at the data, you should probably use `gt_windata` to extract all the data and then vectorize your program.

4 Specific examples

This section will demonstrate the use of some of the mentioned `gt_xxx` functions through practical examples. In the examples, the variable `a` contains a QLDS with data from a NIS raster with 120 pixels in the solar Y direction and 20 pixels in the solar X direction (and only 1 pixel in the "time direction", so `TIX` may always be omitted). The raster has 3 data extraction windows, two in band one, and the last one in band two.

4.1 gt_spectrum

To extract the (band 1) spectrum from the lower left point of the raster, we can use:

```
IDL> s=gt_spectrum(a,xix=0,yix=0,band=1,lambda=lam,xs=x,ys=y)
IDL> help,s,lam
S               FLOAT     = Array(1024)
LAM             FLOAT     = Array(1024)
IDL> plot,lam,s,title='Spectrum from (X,Y) = ('+trim(x)+'",'+trim(y)+'")'
```

Note that it's possible to truncate the keywords as long as it remains unambiguous (`xsolar`⇒`xs` and `ysolar`⇒`ys`). The result of the plot command can be found in Fig. 6 (left). The data from the extraction windows have been placed in an array the same size as the NIS detector. Turning this off with the keyword `/NODESELECT`, we get a slightly different result (Fig. 6, right):

```
IDL> s=gt_spectrum(a,xix=0,yix=0,band=1,lambda=lam,xs=x,ys=y,/nodeselect)
IDL> help,s,lam
S               INT       = Array(200)
LAM             FLOAT     = Array(200)
IDL> plot,lam,s,title='Spectrum from (X,Y) = ('+trim(x)+'",'+trim(y)+'")'
```

If we want just the data from one window, we must identify the window. For this purpose, `gt_wlimits` is a useful procedure:

```
IDL> print,gt_wlimits(a)

       Window:       0             Label: WW_321_29
       Band:         NIS1
       Det-X:              168        to       267
```

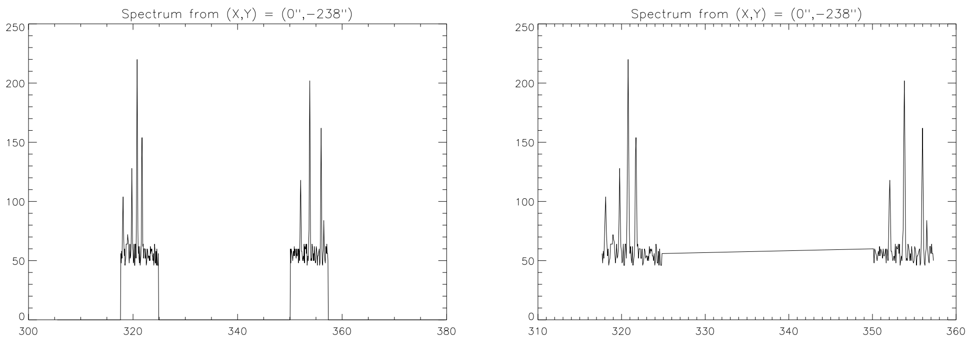

Figure 6: The result of `s=gt_spectrum(a,xix=0,yix=0,band=1,lambda=lam,xs=x,ys=y)` to the left, and `s=gt_spectrum(a,xix=0,yix=0,band=1,lambda=lam,xs=x,ys=y,/nodeselect)` to the right

```
    Det-Y:                522        to       641
    Wavelength:           317.669    to       324.840

    Window:       1              Label: WW_565_97
    Band:         NIS2
    Det-X:                401        to       500
    Det-Y:                382        to       501
    Wavelength:           560.121    to       571.704

    Window:       2              Label: WW_353_74
    Band:         NIS1
    Det-X:                616        to       715
    Det-Y:                522        to       641
    Wavelength:           350.119    to       357.290
  168     267     522     641
  401     500     382     501
  616     715     522     641
```

We want the data from the rightmost data window in Fig. 6, which is window 2 (i.e., the third window, not the second one). The result is plotted in Fig. 7:

```
IDL> s=gt_spectrum(a,xix=0,yix=0,window=2,lam=lam,xs=x,ys=y)
IDL> help,s,lam
S               INT       = Array(100)
LAM             FLOAT     = Array(100)
```

4.2 gt_scanx

If we want to see what the spectrum in band 2 looks like along a horizontal line in the center of the raster, we could do the following (`plot_image` output shown in Fig. 8):

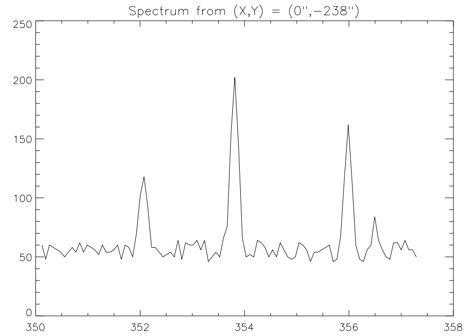

Figure 7: Plot of s=gt_spectrum(a,xix=0,yix=0,window=2,lam=lam,xs=x,ys=y)

```
IDL> sc=gt_scanx(a,yix=60,band=2,/nodeselect,lam=l,xs=x)
IDL> orig=[l(0,0),x(0,0)]
IDL> scale=[l(1,0),x(0,1)]-orig
IDL> plot_image,sc,orig=orig,scale=scale,xtitle='Lambda',ytitle='Solar X',/nosq
```

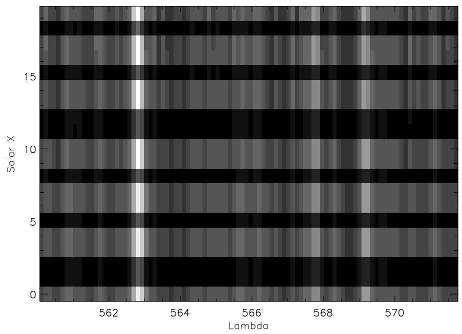

Figure 8: The result of a gt_scanx call. We can see intensity variations along the solar X direction

The gt_scany and gt_scant routines function in a similar way. Specify the applicable indices to indicate where the "slit" should be located, and the routines return data similar to those in Fig. 8.

4.3 gt_scanp

If we want a slit spectrogram along e.g., the diagonal of the area covered by the raster, we would use gt_scanp and specify what points to include (see Fig. 9):

```
IDL> XIX=indgen(120)*19/119 ; Allowable range 0..19
IDL> YIX=indgen(120)        ; Allowable range 0..119 for this raster
IDL> sc=gt_scanp(a,xix=xix,yix=yix,band=2,xs=x,ys=y,/nodeselect)
IDL> plot_image,sc,/nosq,xtitle='Dispersion axis',ytitle='Spatial axis'
```

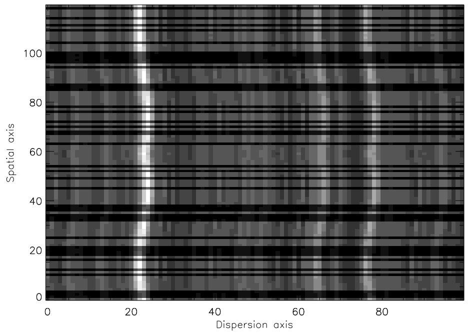

Figure 9: The result of a gt_scanp call, using points along the diagonal. We can see velocity variations along the slit.

Now if we want to know the solar coordinates of the spectrum in the line sc(*,i) in this scan, we simply use what has been output to the x and y arrays: print,x(0,i),y(0,i) (We can use zero as the first index since x doesn't vary as a function of lambda).

If we are interested in the spectrum integrated along the diagonal, we can simply use int_spec=TOTAL(sc,2) which integrates the spectrogram over the 2nd dimension (along the "slit").

4.4 gt_iimage

If the raster at hand contains several pixels in both solar X and Y directions, it is possible to form an image of the raster area with intensities from a given dispersion pixel on the detector.

This is done with the gt_iimage routine as follows, with the plot_image result shown in Fig. 10. We have to specify the detector X pixel number in either of the two ways described at the end of Section 2. See also Fig. 5.

```
IDL> im=gt_iimage(a,window=0,offset=20,xs=x,ys=y)
IDL> orig=[x(0,0),y(0,0)]        ; Origin in arcseconds
IDL> scale=[x(1,0),y(0,1)]-orig ; Scale in arcseconds
IDL> plot_image,im,orig=orig,scale=scale,xtitle='Solar X',ytitle='Solar Y'
```

4.5 gt_mimage

Whilst gt_iimage takes the intensities from a specified dispersion pixel number, gt_mimage attempts to make a monochromatic image at a given *wavelength*. In doing this, it interpolates linearly the intensities from neighbouring pixels. An example follows, with plot_image output in Fig. 11.

```
IDL> im=gt_mimage(a,562.6)
IDL> plot_image,im,xtitle='XIX',ytitle='YIX'
```

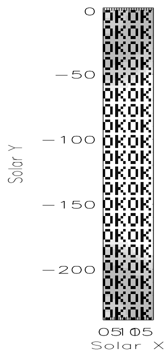

Figure 10: The image from `gt_iimage(a,window=0,offset=20,xs=x,ys=y)`. Note that the plot has been stretched in the X direction for (attempted) clarity.

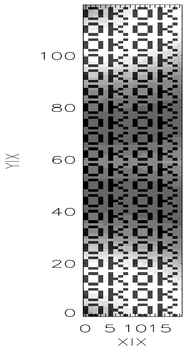

Figure 11: Output from `gt_mimage(a,562.6)`. The plot has been stretched in the X direction

4.6 gt_bimage

`gt_bimage` is similar to `gt_mimage`, except that it attempts to integrate the intensities over a wavelength band specified by the lower and upper limits (2nd and 3rd parameters). There cannot be gaps without detector data inside the integration limits. Example below, with `plot_image` output in Fig. 12.

```
IDL> im=gt_bimage(a,562.0,564.0)
IDL> plot_image,im,xtitle='XIX',ytitle='YIX'
```

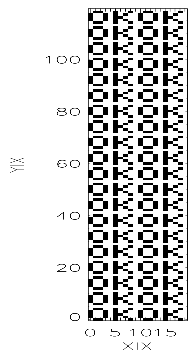

Figure 12: Output from `gt_bimage(a,562.0,564.0)`. The plot has been stretched in the X direction. The background structure seen in Fig. 11 caused by varying velocities has disappeared when integrating over the emission line.

5 Example program

Below is a sample "quick and dirty" program that demonstrates the usefulness of the `gt_xxx` functions. It is a procedure called `quick` that takes a QLDS as a parameter. The QLDS has to contain a raster with more than one pixel in both X and Y directions. It starts by extracting a spectrum from the center of the raster, plotting the spectrum (without wavelengths) as a contiguous array. The user may then click on the spectrum to select a wavelength, and the program displays an image returned by `gt_mimage` at that wavelength. The user may then click on the image to select a new point from which the next spectrum is extracted etc.

Although very simple, this program is easily modified (by having two `cursor` statements) into displaying images integrated over a wavelength band, or even finding *two* images integrated over different wavelength regions, and then displaying their ratios.

```
PRO quick,qlds

  !P.multi = [0,1,1]

  dims = gt_dimension(qlds)     ; Returns a structure with information on
```

```
                              ; the size of the raster. See online doc.

xix = dims.ssolar_x/2         ; We want to start with the middle pixel
yix = dims.ssolar_y/2         ;

nwin = gt_numwin(qlds)        ; Find the number of data extraction windows


winsize = gt_winsize(qlds)       ; We want to know the size (no. of
                              ; dispersion pixels) of the extraction
                              ; windows.
winsize = winsize(0)          ; gt_winsize returns one value for
                              ; each window -- they're identical.

WINDOW = INDGEN(nwin)

WINDOW,0                      ; Get some display windows up.

!quiet = 1

continue = 1
WHILE continue DO BEGIN

   ;; Fetch a spectrum and plot without wavelengths

   s = gt_spectrum(qlds,xix = xix,yix = yix,lambda = l,WINDOW=window)
   plot_io,s > 1,title='Click on plot to get image',$
      subtitle='Click left of y axis to quit'

   ;; Overplot the division between different extraction windows,
   ;; and display the window labels centered on each window

   FOR i = 0,nwin-1 DO BEGIN
      xyouts,(i+0.5)*winsize,MAX(s),gt_wlabel(qlds,i),alignment=0.5
      oplot,i*winsize*[1,1]-0.5,[1,MAX(s)],thick = 3
   ENDFOR

   ;; Wait for a cursor press

   cursor,x,y,/data,/down
   IF x GE 0 AND x LT N_ELEMENTS(l) THEN BEGIN

      ;; Avoid using wavelengths between two windows.

      IF FIX(x) GT 0 AND FIX(x+1) MOD winsize EQ 0 THEN $
         X = FIX(x)

      ;; Lookup the wavelength in the array l

      lam = interpolate(l,x)
      PRINT,"Lambda: "+trim(lam)

      ;; Find the image at this wavelength and display it
```

```
        im = gt_mimage(qlds,lam)

        plot_image,im,title = 'Click on image to select new point'

        cursor,xix,yix,/data,/down
        xix = (FIX(xix) > 0) < (dims.ssolar_x)
        yix = (FIX(yix) > 0) < (dims.ssolar_y)
     END ELSE continue=0
  ENDWHILE

END
```