A worked example for the SWIFT data reduction software

Ryan Houghton

Contents

1 Introduction 1

2 Preliminaries 2

3 Prepping 3

4 Extracting vertical lines 6

5 Wavelength calibration 11

6 Flat fielding and illumination correction 13

7 Horizontal line correction 14

8 Checking the calibrations 19

9 Correcting for bad pixels and cosmics 23

10 Spatial flexure 26

11 Making science cubes 28

12 Putting it all together 31

13 Further Processing 33

13.1 Sky subtraction 33
13.2 Combining cubes 34
13.3 Correcting for DAR 35
13.4 Extracting 1D spectra 35
13.5 Telluric Correction 36

Chapter 1

Introduction

In this example, we will calibrate and reduce data taken with the Short Wavelength Integral Field spectrograph (SWIFT) in the 235mas scale (seeing mode). The data were all taken on one night (5th May 2012). The science target is an early-type galaxy in the Coma cluster; the science goal is to determine the internal kinematics (bulk motion of the stars) from the doppler shifts of the calcium triplet lines at around 0.85μm.

Data reduction in this case refers to the removal of instrumental effects. Thus the output of this example are 3D data cubes: two spatial dimensions and one spectral. The removal of telluric absorption, the calibration of fluxes and the combination of multiple exposures is not the principal subject of this document, although some useful advice is given that the end in Further Processing (§13)

Chapter 2

Preliminaries

To work through this example, you need a large tar file called **SwiftWorkedExample.tgz** (this should have been made available to you). Starting in a (bash/tcsh) command prompt, untar the contents of this file; it should unpack into a new directory called **WorkedExample**.

```
> tar xvfz SwiftWorkedExample.tgz
```

Then change directory in to WorkedExample and do a directory listing.

```
> cd WorkedExample
> ls
```

You should see the following list of files

```
master003.fits  master008.fits  master013.fits  slave003.fits   slave008.fits   slave013.fits
master004.fits  master009.fits  master014.fits  slave004.fits   slave009.fits   slave014.fits
master005.fits  master010.fits  master074.fits  slave005.fits   slave010.fits   slave074.fits
master006.fits  master011.fits  master077.fits  slave006.fits   slave011.fits   slave077.fits
master007.fits  master012.fits  master078.fits  slave007.fits   slave012.fits   slave078.fits
```

Files from the master chip (and spectrograph) are prefixed *master*, while those from the slave are prefixed slave. The three digit number following this is the *exposure number*. Most of these files are calibrations: exposures 9 to 11 are the (coarse) vertical line frames; exposures 3 to 5 are the arc line frames; exposures 6 to 8 are the halogen lamp frames (note that ideally 5 exposures would be used); exposures 12 to 14 are the horizontal line frames. The three science exposures are 74, 77 and 78, observing with the object-sky-object sequence.

Chapter 3

Prepping

The first stage in the data reduction process involves removing the bias level, trimming the overscan and averaging multiple *calibration* frames (science frames are not averaged). Collectively, this is referred to as prepping.

Start by loading the Swift pipeline in IRAF (installation details can be found in the pipeline manual).

```
ecl> swift
```

The prompt should now read

```
swift>
```

Now *unlearn* the *swiftredMS* task.

```
unlearn swiftredMS
```

and then edit the parameter file for the same task

```
epar swiftredMS
```

Enter the exposure numbers for the science frames, the VLs, the Arcs, the halogen lamps and the HLs. After you've done that, the first few parameters should look like the following:

```
sci_inpu=             074,078
prefix_s=                   -
vl_frame=         009,010,011
al_frame=         003,004,005
(lamp_fr=         006,007,008)
(hl_fram=         012,013,014)
```

Note that exposure numbers are given in 3 digit format and separated by commas.

Confirm the scale of the observations (all 235mas) and that you want to prep the data. The relevant parameters should look like the following:

```
(scale  =                0.235)
(S_PREP =                  yes)
(S_EXVL =                   no)
(S_ARCCA=                   no)
(S_FI   =                   no)
(S_HL   =                   no)
(S_CALCU=                   no)
(S_BPS  =                   no)
(S_SFLEX=                   no)
(S_SCICU=                   no)
(S_MSMER=                   no)
```

Next you should verify that you will prep the science, VL, HL, arc and lamp files. The relevant parameters should look like the following:

```
(p_flat =                  yes)
(p_vtrac=                  yes)
(p_arc  =                  yes)
(p_sci  =                  yes)
(p_htrac=                  yes)
```

Note that if you weren't providing HL or halogen lamp frames (i.e. you didn't want to flat field, illumination correct or correct the warp in the field of view), the **p_htrace** and **p_flat** parameters above could be set to **no**. You should now check that the other parameters relevant to the prepping stage are set correctly. They should be as follows:

```
(do_b   =                  yes)
(do_trim=                  yes)

(master_=                   1.)
(master_=                 0.97)
(slave_a=                   1.)
(slave_a=                 0.97)
(msgainr=                   1.)
(masterb=      swift$mBPM2.dat)
(slavebp=      swift$sBPM2.dat)
```

With that done, you can run the pipeline! Either *epar* in to the param file and use **:go**, or do the following:

```
swift> swiftredMS (mode='h')
```

The pipeline should finish and exit cleanly with the following:

```
ALL PREPPING COMPLETE!
**************************************************
*                Reduction Complete!             *
**************************************************
```

Of course the reduction isn't complete; just the prepping.

Chapter 4

Extracting vertical lines

In order to reconstruct a cube, we need to calibrate the position of the individual slitlets as a function of wavelength. This is done with the vertical line (VL) frames. Switch off the prepping stage and turn on the exvl stage.

```
(S_PREP =                    no)
(S_EXVL =                   yes)
```

Now check that the extraction parameters are correct. The *cut_low* and *cut_high* params are the most important; they define the amount to extract left and right of the central line (refer to the SWIFT reduction manual). If you're planning to reduce frames taken in the 80mas or 16mas scales, you may need to edit these, as well as the *vl_b_sa* and *vl_widt* (refer to the data reduction manual for further details). For now, they should look as follows:

```
(vl_b_sa=                      )
(vl_widt=                     3)
(cut_low=                 -45.1)
(cut_hig=                  45.1)
```

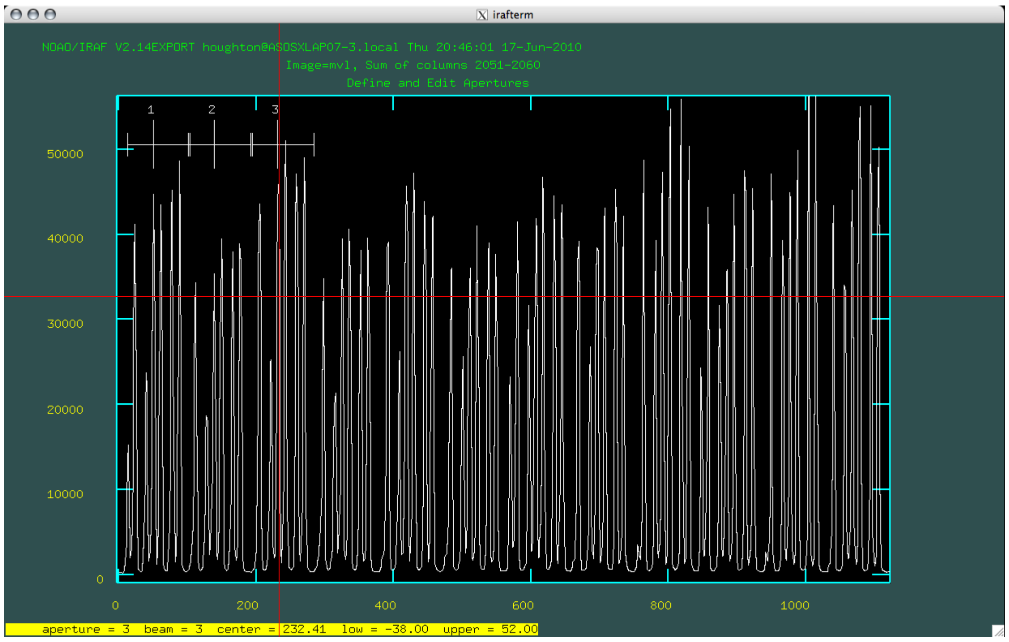
The addition of 0.1 to the *cut_low* and *cut_high* parameters is just a get-around for a bug in a pall, which occasionally rounds 45.0 to 44.
Now run the pipeline again and trace the slitlets in the master and slave chips (refer to the reduction manual for further details).

```
swift> swiftredMS (mode='h')
```

You will be then asked:

```
Edit apertures for [vertical line frame]?  (yes):
```

You should answer **yes** to this. A window will pop up which should look something like Fig. 4.1 which shows a cut through all the vertical lines. For example, in the 235mas scale with the coarse vertical lines, each slitlet should have 7 vertical lines running along



```
NOAO/IRAF V2.14EXPORT houghton@ASOSXLAP07-3.local Thu 20:46:01 17-Jun-2010
Image=mvl, Sum of columns 2051-2060
Define and Edit Apertures
```

```
aperture = 3  beam = 3  center = 232.41  low = -38.00  upper = 52.00
```

Figure 4.1: An example of the coarse vertical line frame in the 235mas spaxel scale being labelled with extraction apertures (centred on the middle (of 7) traces). Note that apertures (or slitlets) are strictly numbered from left to right in chronological order.

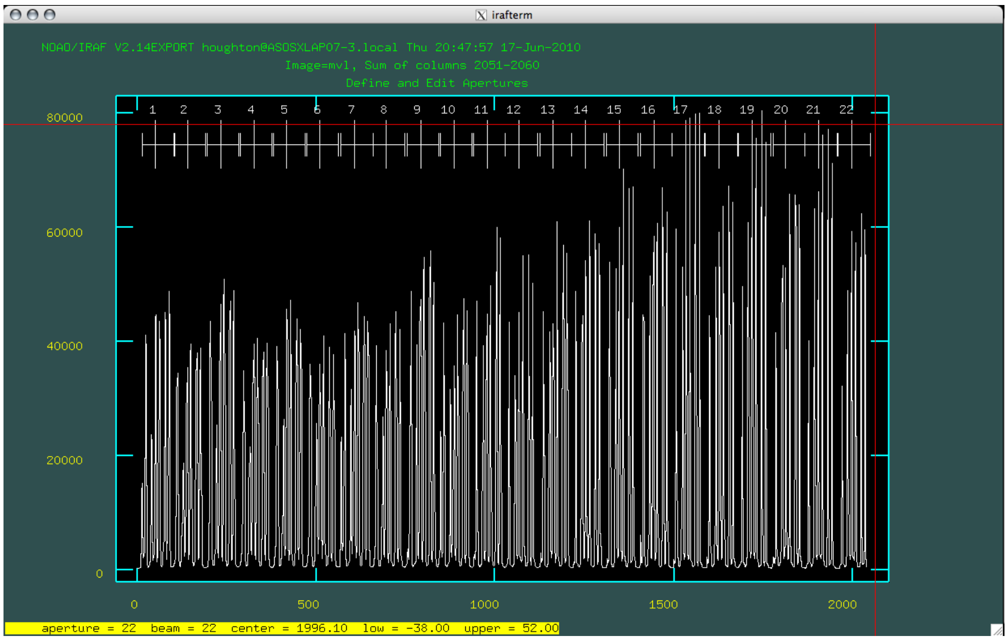
the wavelength axis, grouped in a two, then a three, then another two (the first line may be just off the edge of some slitlets). For other scales, using the coarse or fine vertical lines, this pattern changes. You now need to mark the central trace in each of the slitlets (i.e. in the 235mas scale with the coarse vertical lines, this is the middle line in the grouping of three); at this point you may find it easier to zoom in using the **w** and **e** keys. To do this, move the mouse (and the red cross hairs) over the middle trace in the first (far left) slitlet and press **m** (for **m**ark). A white line should appear above the trace showing the width of the slitlet and the number of the aperture (or in this case, slitlet). You need to repeat this another 22 times, moving from the left to the right in strict order (see Fig. 4.1). Finally you should be left with 22 marked apertures to trace and extract (as in Fig. 4.2). You can delete traces and remark them if you make a mistake, but the ordering is strictly with aperture number ascending from left to right as shown in Fig. 4.2.

Once you have marked the apertures, quit by pressing **q** in the plotting window. After that, you will be asked:

```
Trace apertures for [vertical line frame]? (yes):
```

You must reply **yes** to this question; failure to do so will result in failure to reduce any data. You will then be asked:

```
Fit traced positions for [vertical line frame] interactively? (yes):
```



```
NOAO/IRAF V2.14EXPORT houghton@ASOSXLAP07-3.local Thu 20:47:57 17-Jun-2010
Image=mvl, Sum of columns 2051-2060
Define and Edit Apertures
```

```
aperture = 22  beam = 22  center = 1996.10  low = -38.00  upper = 52.00
```

Figure 4.2: A coarse vertical line frame observed in the 235mas spaxel scale. All the 22 traces have been marked and are shown above the relevant areas.

At this point, it's best to reply **yes** although if you're in a hurry you might risk trusting the automated tracing (it's not bad) and reply **no**; if you choose the latter option, be sure to create the cubes of the calibration data (by setting **S_CALCUBES** to **yes**) and inspect the cube created by the vertical line frame (in something like QFitsView, an external application) to make sure all the vertical lines are aligned and... vertical.

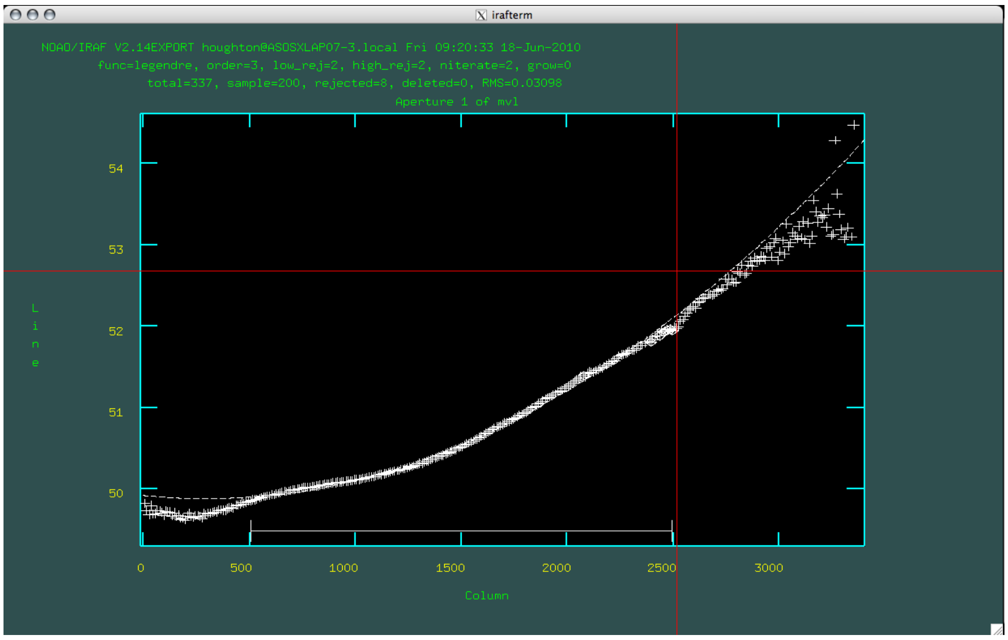
If you choose to trace the lines interactively, you will be asked:

```
Fit curve to aperture 1 of [vertical line frame] interactively? (yes):
```

To which you should of course reply **yes**. You will then see something along the lines of the trace shown in Fig. 4.3. Note that the white band at the bottom shows the **sample** which is being fitted. Press **?** to learn more about how you can change this fit. Try to fit only the regions where the trace was stable: you can clearly see that at short and long wavelengths the scatter increases dramatically (due to a PSF problem at short and long wavelengths - this data was taken before 2010) and these regions should not be fit by the polynomial (as is the case in Fig. 4.3). Furthermore, keep the order of the fit low (around 3-5) so a reliable extrapolation can be made into the poor PSF regions.

Once you're happy with the fit, press **q** to quit: you will then be asked if you want to fit the trace for aperture 2 interactively... if you do, reply **yes**. This process will repeat for all 22 slitlets. At the end you will be asked:

```
Extract apertures for [vertical line frame] ? (yes):
```



```
NOAO/IRAF V2.14EXPORT houghton@ASOSXLAP07-3.local Fri 09:20:33 18-Jun-2010
        func=legendre, order=3, low_rej=2, high_rej=2, niterate=2, grow=0
                total=337, sample=200, rejected=8, deleted=0, RMS=0.03098
                                              Aperture 1 of mvl
```

Figure 4.3: The *swiftredMS* routine follows the maxima of the selected vertical line. At the very short and long wavelengths, it can sometimes have difficulty due to problems with the SWIFT PSF in these regions, particularly if the data was taken before 2010; thus is is best to fit the low scatter regions in the centre (where the PSF is good) and use a low order polynomial to extrapolate into the affected regions.

to which you must reply **yes**. The routine then extracts 22 2D specra from the vertical line frame, following the traces you just fitted and correcting for the curvature (via linear interpolation).

You just calibrated the vertical lines in the master spectrograph! But you now need to calibrate them in the slave spectrograph. So repeat the process, first marking the apertures (slit lets), then tracing and extracting them.

After calibrating the vertical lines, the rest is normally automatic: it bags up the 22 2D slitlets into a single MEF (mutli-extension fits) file with suffix *.strip.fits* for both the master and slave spectrographs.

Again, if all has gone well, you'll end up with

```
**************************************************
*               Reduction Complete!              *
**************************************************
```

Chapter 5

Wavelength calibration

The spectra need to be wavelength calibrated using the arc frames. Turn off the exvl stage and turn on the arccal stage:

```
(S_EXVL =                     no)
(S_ARCCA=                    yes)
```

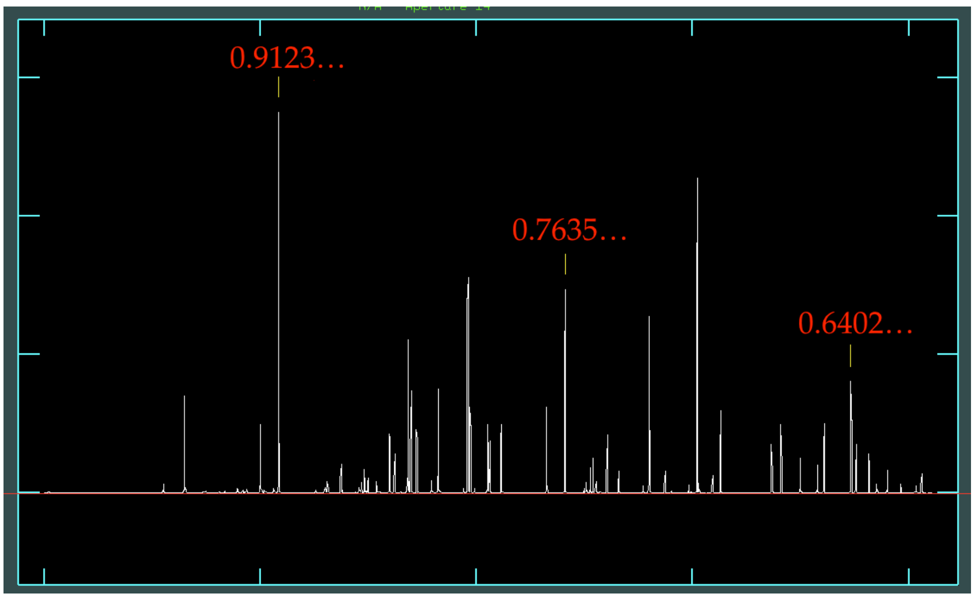


Figure 5.1: The three key SWIFT arc lines to identify manually

You may want to check that other relevant parameters are correctly set.

```
(line_li=       swift$ArNe.dat)
```

```
(xorder =                  5)
(yorder =                  4)

(nlost  =                 10)
(lam0   =               0.63)
(dlam   =             1.0E-4)
(nlam   =               4112)
(loglam =                 no)

(trans_i=             linear)
```

When you're happy, run the pipeline.

```
swift> swiftredMS (mode='h')
```

In this worked example, you will probably need to confirm the automatic solutions for master slitlets 1 & 3, slave slitlet 18 and manually find the solutions for master slitlet 8 and slave slitlets 4, 6 & 13.

There is a trick to manual identifications; you don't need to identify all the lines, just 3 at 0.6402um, 0.7635um and 0.9123um. Then hit **l**, **f**, **q** a few times to let the automatic finding algorithm do the rest. You should then get a good fit (as described in the data reduction manual). Again, when the pipeline is finished, it displays

```
**************************************************
*                Reduction Complete!             *
**************************************************
```

Chapter 6

Flat fielding and illumination correction

Now we need to create a flat field frame from the halogen lamp exposures. We can also make an illumination frame from the same exposures. To do this, turn off the arc calibration stage and turn on *S_FI*.

```
(S_ARCCA=                   no)
(S_FI   =                  yes)
```

Then run the pipeline.

```
swift> swiftredMS (mode='h')
```

Note that one can split the flat field and illumination problem into four parts: 1) pixel-to-pixel variations AKA flat field variations, (e.g. quantum efficiency differences between pixels), 2) detector illumination variations (e.g. dust on the detector surface), 3) field-of-view illumination variations (e.g dust, hairs or cracks on the optical surfaces) and 4) spectral illumination variations (quantum efficiency variation with wavelength as well as illumination of a single spaxel by different points on the sky as a function of wavelength). I have chosen to group 1) & 2) into “flat field correction” and 3) & 4) into “illumination correction”.

After processing the data, you should find the files *masterFLAT.fits* and *slaveFLAT.fits*; these are the flat fields for the two detectors. You can open them in ds9 and inspect them; they should have values around 1. There are also two cubes created: *masterILLUM.fits* and *slaveILLUM.fits*. These are the illumination cubes; you can inspect them in QFitsView and see how the illumination changes across the field of view and with wavelength. Again, when the pipeline is finished, it displays

```
**************************************************
*                Reduction Complete!             *
**************************************************
```

Chapter 7

Horizontal line correction

SWIFT has a slightly warped field of view. Across the majority of the detector, horizontal lines will be observed as horizontal; but at the top and bottom (along the longest dimension), horizontal lines appear slightly tilted: there is some asymmetric shear present. For most science cases, where the object lies near the middle of the detector, this shear is not a problem and is not noticeable. However, if the science target fills the field of view (e.g. a nearby planet observed with the 80mas or 16mas scales) or if you wish to mosaic several SWIFT pointings, you will probably need to correct the data cubes for the shear effect. See Fig. 7.1 for an illustration of the asymmetric shear across the SWIFT field of view (master & slave combined).

Start by turning off the flat field stage, and turning on the horizontal line calibration.

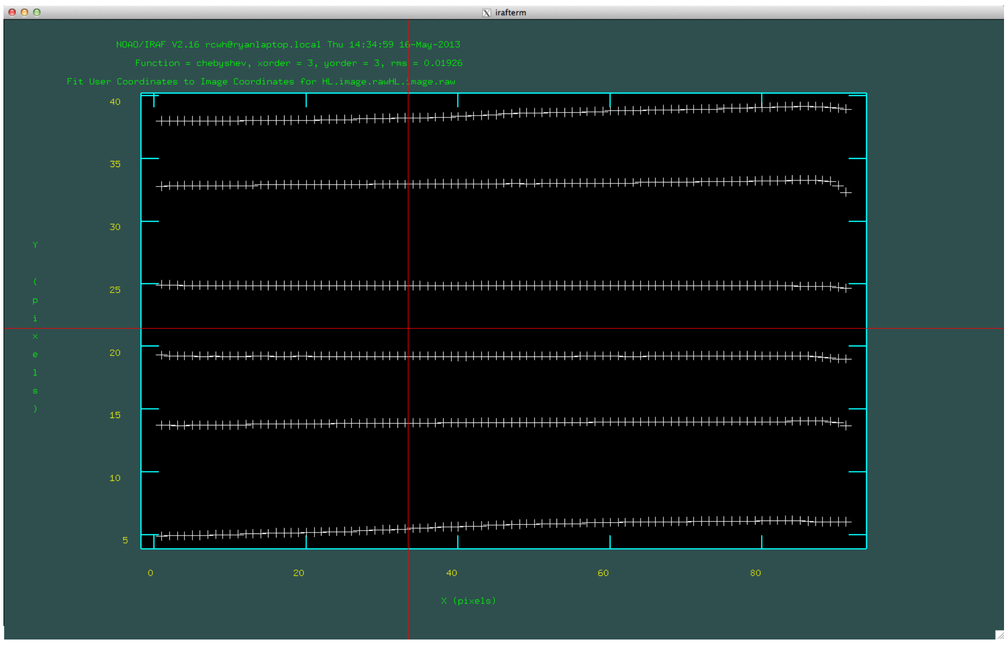
```
(S_FI   =                   no)
(S_HL   =                  yes)
```

Then run the pipeline.

```
swift> swiftredMS (mode='h')
```

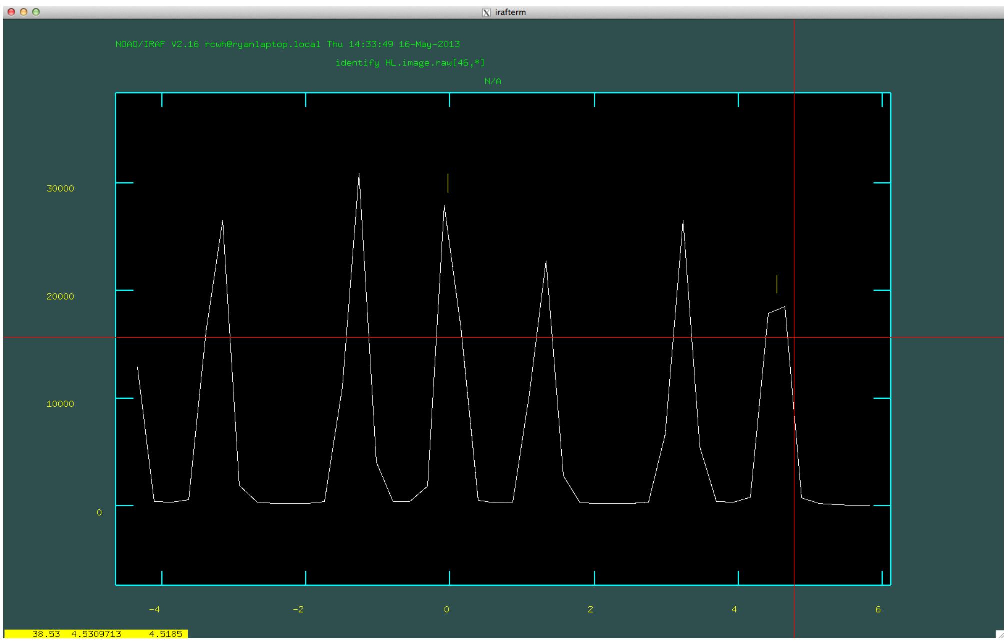
Note that, as for the vertical line stage, if you're using the 80mas or 16mas scales with the coarse horizontal lines, you may need to edit *vl_b_sa* and *vl_widt* (refer to the data reduction manual for further details). The default settings are fine for the 235mas scale and the coarse horizontal lines.

This routine requires some user input. It starts asking you to identify the horizontal lines; this is where the relative positions of the lines (in terms of arc seconds) are defined. Ideally, it would identify the lines automatically, but in practice it requires some help from you. Like the vertical lines, in the 235mas scale, the coarse lines appear in a pair, followed by a triple, followed by a pair. Again, as for the vertical lines, identify the middle line in the triple by pressing **m**; now type in the position of this line, which is *defined* to be **0.0** – the centre of the detector. Next, identify (**m**ark) the furthest line to the right. Depending on which scale you're using, you need to specify the position of this line in arc seconds from the central line. For the 235mas scale and the coarse horizontal lines, this line is **4.5185** arcsec away; if you type **4.5** and hit enter, IRAF will do the



```
NOAO/IRAF V2.16 rcwh@ryanlaptop.local Thu 14:34:59 16-May-2013
    Function = chebyshev, xorder = 3, yorder = 3, rms = 0.01926
Fit User Coordinates to Image Coordinates for HL.image.rawHL.image.raw
```

Figure 7.1: The location of the horizontal lines in the 235mas SWIFT FoV.



```
NOAO/IRAF V2.16 rcwh@ryanlaptop.local Thu 14:33:49 16-May-2013
                         identify HL.image.raw[46,*]
                                                  N/A
```

```
    38.53  4.5309713      4.5185
```

Figure 7.2: An example of the horizontal lines marked and identified in the 235mas scale.

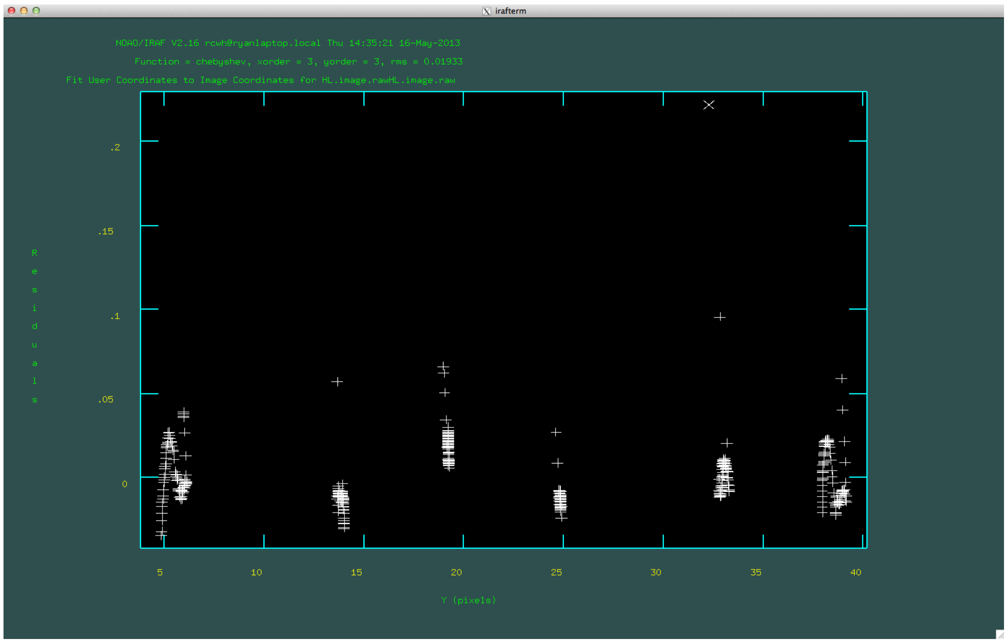

```
NOAO/IRAF V2.16 rcwh@ryanlaptop.local Thu 14:35:21 16-May-2013
   Function = chebyshev, xorder = 3, yorder = 3, rms = 0.01933
Fit User Coordinates to Image Coordinates for HL.image.rawHL.image.raw
```

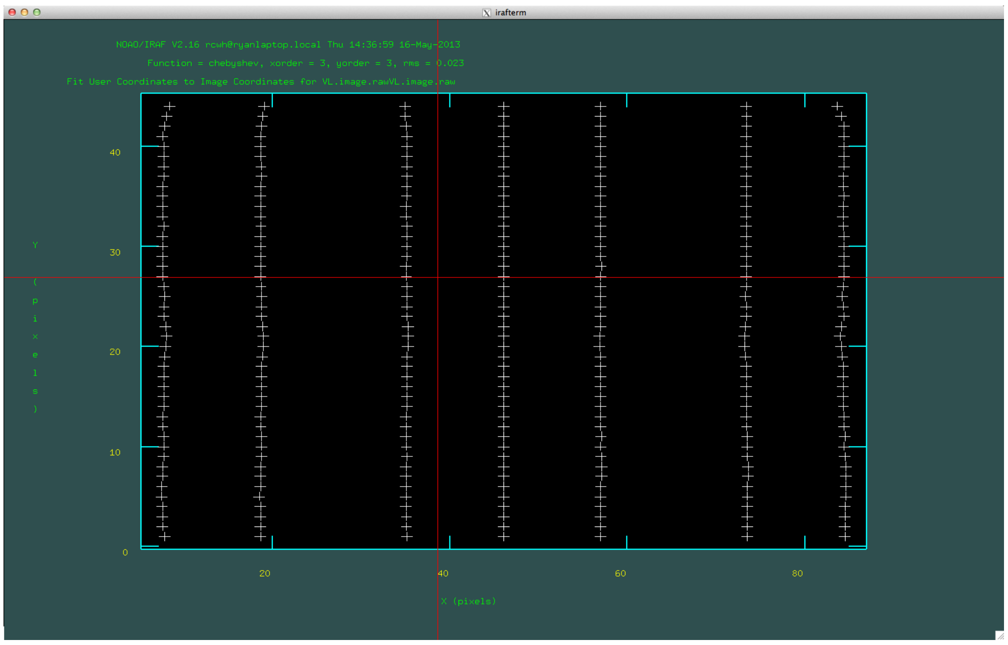
Figure 7.3: Typical residuals from a polynomial fit to the horizontal lines (correcting the asymmetric shear) in the 235mas scale; note one point with a residual of 0.2 pixels has been deleted from the fit.

rest. With two lines identified (as in Fig. 7.2), you can usually automatically identify the rest; press **l**. Hopefully now, all the lines show yellow pointers above them; if not, you should manually identify them (the coordinate file for the 235mas scale is in the SWIFT IRAF package directory, **swift$HL235.dat**). Now press **f** to fit a linear function to these points. Then hit **q** twice to exit. The next stage automatically identifies all those lines left and right of the middle, and then fits the (x, y) pixel locations in the collapsed image to (x', y') arc second locations in the sky; you are asked if you want to do this interactively

```
Fit HL.image.raw interactively (yes):
```

There is no need to do this interactively, unless you want to check that everything is working well. If you do enter **yes**, then you will see something like Fig. 7.3. You can delete points and refit; hit **?** to learn more, or investigate the *fitcoords* IRAF routine. The residuals of the fit should be $\lesssim 0.1$ pixels; you can see in Fig. 7.3 that we deleted a point with a residual of 0.2 pixels, although the effect on the fit was marginal.

You now have to repeat a similar process for the vertical lines; this is because you just used the horizontal lines to define the horizontal plate scale and you now need to do the same using the vertical lines. Note that the vertical lines are not in the same sky locations as the horizontal lines! SWIFT has a roughly 2:1 aspect ratio in the field of view, so the vertical lines are spaced twice as far apart as the horizontal lines. See Figs. 7.4, 7.5 and 7.6 for analogies to Figs. 7.1, 7.2 and 7.3. Note that the vertical

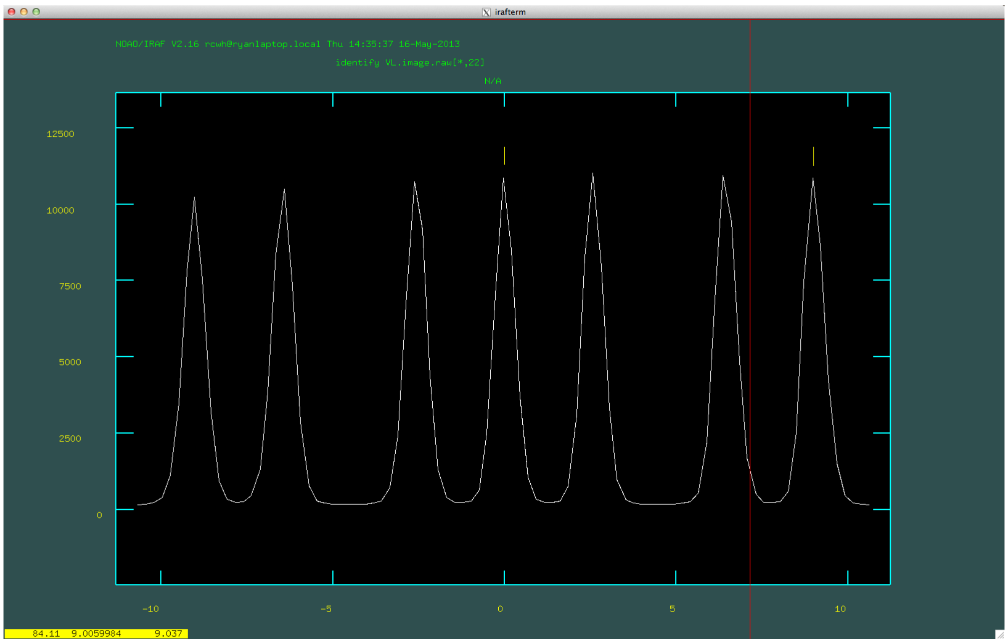


```
NOAO/IRAF V2.16 rcwh@ryanlaptop.local Thu 14:36:59 16-May-2013
Function = chebyshev, xorder = 3, yorder = 3, rms = 0.023
Fit User Coordinates to Image Coordinates for VL.image.rawVL.image.raw
```

Figure 7.4: The location of the vertical lines in the 235mas SWIFT FoV.

line to the right in Fig. 7.4 has position **9.037** (not 4.5185 as it was for the HL) in the 235mas scale (see **swift$VL235.dat**). After running this stage, you should inspect **HL.cube.fits** and **VL.cube.fits** in QFitsView and check that the lines remain fixed across all wavelengths. When complete, the pipeline returns

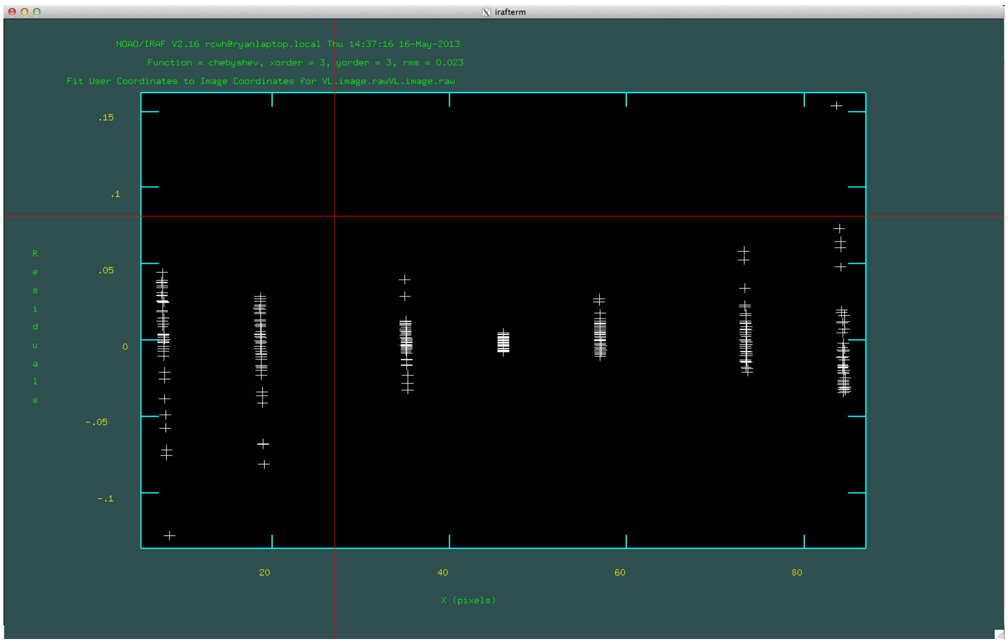
```
**************************************************
*                Reduction Complete!             *
**************************************************
```



```
NOAO/IRAF V2.16 rcwh@ryanlaptop.local Thu 14:35:37 16-May-2013
                        identify VL.image.raw[*,22]
                                        N/A
```

```
84.11  9.0059984      9.037
```

Figure 7.5: An example of the vertical lines marked and identified in the 235mas scale.



```
NOAO/IRAF V2.16 rcwh@ryanlaptop.local Thu 14:37:16 16-May-2013
    Function = chebyshev, xorder = 3, yorder = 3, rms = 0.023
Fit User Coordinates to Image Coordinates for VL.image.rawVL.image.raw
```

Figure 7.6: Typical residuals from a polynomial fit to the vertical lines in the 235mas scale.

Chapter 8

Checking the calibrations

It's always a good idea to check that your calibration frames look like what you expect them to. Although this stage is far from essential, it's strongly recommended; a lot can go wrong with IFU data. So turn off the horizontal line correction and turn on *S_CALCU*.

```
(S_HL   =                    no) Calibrate horizontal trace correction? (yes/no)
(S_CALCU=                   yes) Make cubes of the calib data (arc/trace)? (y/n)
```

Then run the pipeline.

```
swift> swiftredMS (mode='h')
```

Now you'll have to watch the pipeline make cubes of the calibrations; be patient (tea?). Eventually it will show you a reduced and then re-collapsed 2D image of the (master & slave) arc frame and ask

```
Please inspect the ARC frame in buffer #1 of DS9
Is this transform acceptable? (y/n)
```

The object of showing you the reduced ARC frame is to make sure the arc lines are all exactly horizontal (a single row represents a single wavelength). In ds9, zoom in at the top and bottom of the image to check. Note that you may see the PSF degrade slightly; this isn't the same as the wavelength solution being wrong. One example of a "thing to watch out for" is shown in Fig. 8.1; notice how the arc lines warp upwards at the end of some slitlets. In this case, the wavelength calibration is wrong and may need to be repeated with more interaction from the user (individually inspecting the identification of arc lines and polynomial fits across each slitlet; in this case, the parameters *interfi*, *reidans* & *interac* may be useful). If you're happy with how the arc frame looks (i.e. it looks like Fig. 8.2), reply **yes**. The pipeline will then create a cube of the vertical line data and display a wavelength collapsed image in ds9. Again, it will ask if it looks OK. You should check that the lines in the top and bottom halves (slave and master) are aligned; no offset should be visible. If there is, something has gone wrong. Once you're happy (it looks like Fig. 8.3), reply **yes**.

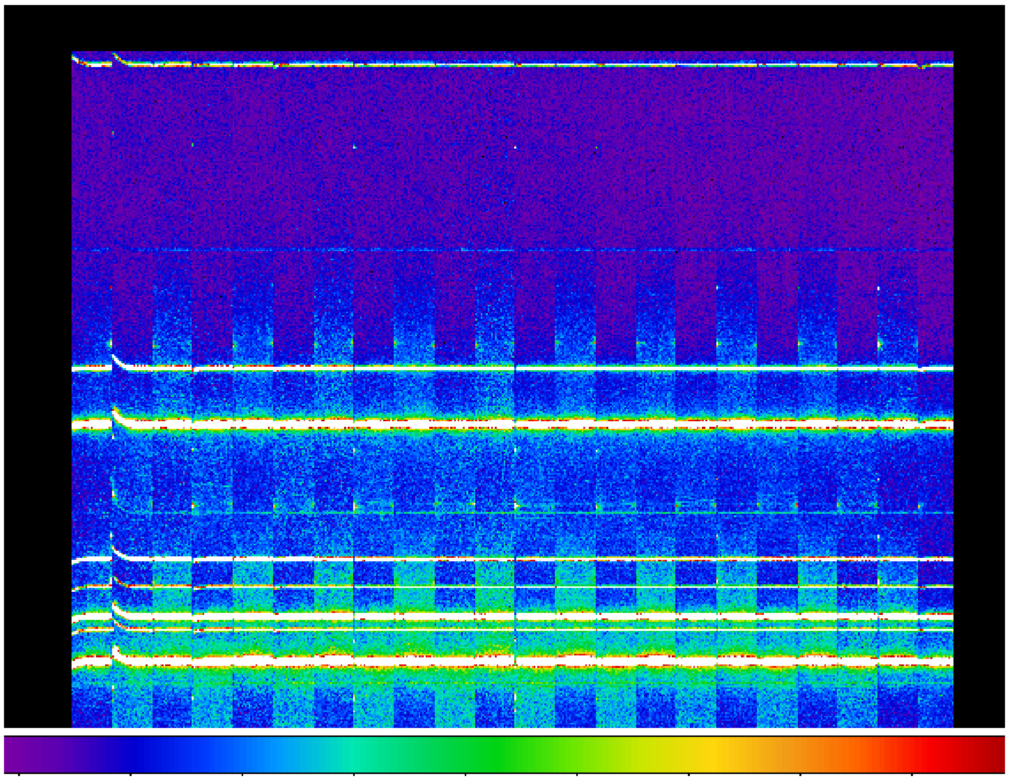


Figure 8.1: An example of what can go wrong with the automatic wavelength calibration: here we see that the first few slitlets aren't quite right; we're going to need to repeat the *S_ARCCAL* process being very careful with the *reidentify* stages. It's always a good idea to inspect the transformed arc frame, even if things appeared to work fine from the written output (as in this case).

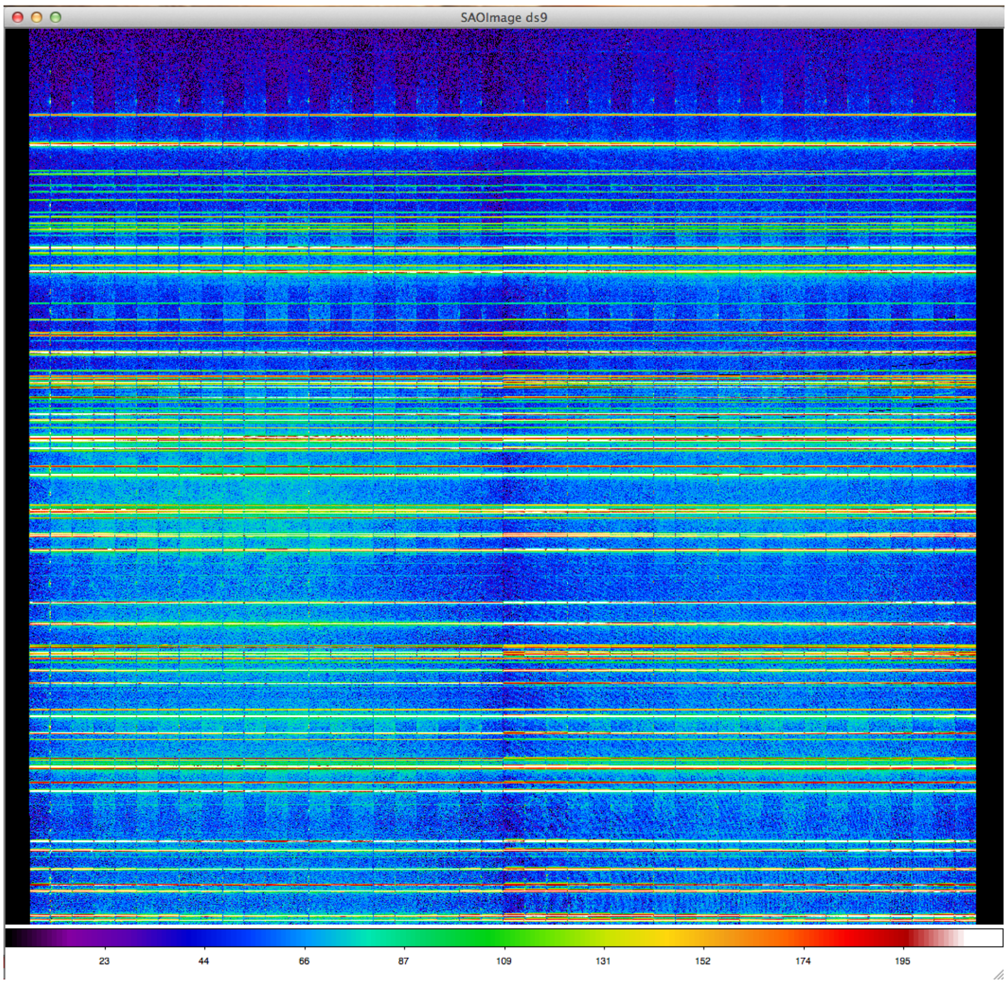


Figure 8.2: An example of what a reduced and then flattened arc frame should look like; no discontinuities and no warps.

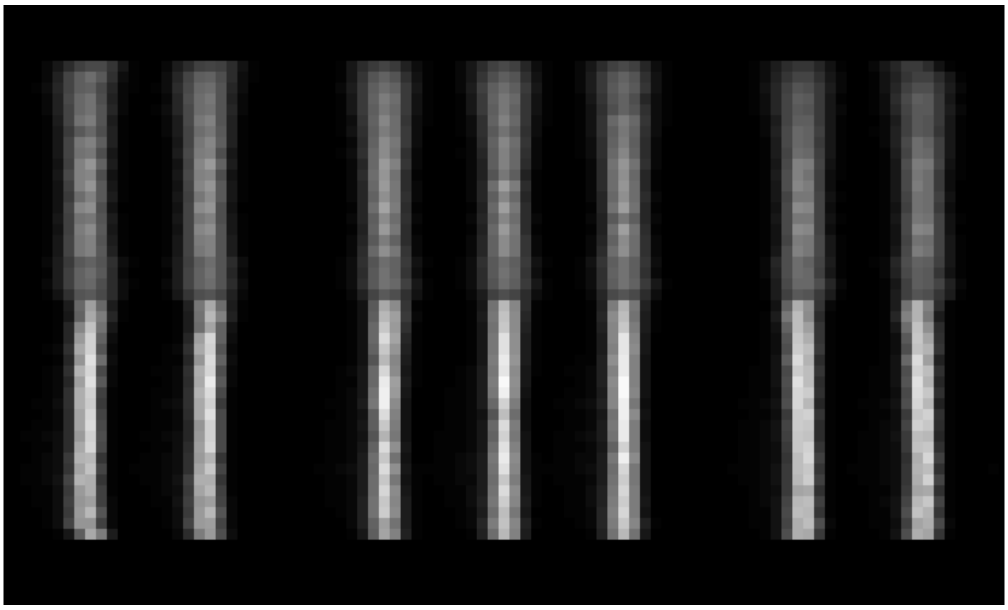


Figure 8.3: An example of what a reduced vertical line frame (collapsed across all wavelengths) should look like; no discontinuities and no warps.

And that's the end of this stage (and the formal instrument calibrations). If you did find something wrong above and replied **no**, the pipeline just stops; it's up to you to seek a solution to the problem.

Chapter 9

Correcting for bad pixels and cosmics

You're now ready to start reducing the science frames. The first step is to (optionally) mark all known bad pixels and find cosmic rays. So turn off *S_calcu* and turn on *S_BPS*. Note that you also need to change the prefix to select prepped science frames (using the $\wedge$**p** option); if you try to reduced unprepped frames, you'll end up with nonsense (or more likely an error). The *swiftredMS* parameters should look like the following

```
prefix_s=                  ^p
(S_CALCU=                  no)
(S_BPS  =                 yes)
```

If you want to find cosmics and treat them as bad pixels, you'll also need to change the *do_laco* parameter to **yes**.

Note that there are a variety of cosmic ray strike types and one type (low energy electrons) leaves a round 'blob' or 'spot' on the detector (see Don Groom's excellent description in 'Cosmic Rays and Other Nonsense in Astronomical CCD Imagers'). The La Cosmic (van Dokkum 2001) routine, which we use here, isn't very good at finding these spots (it's excellent at finding everything else). If your science frames are of faint targets (e.g. 15mins on a Coma cluster galaxy as done here), you can threshold them for bright pixels: these will undoubtably be spots or remaining cosmics. To do this, set the *do_thre* parameter to **yes**. You can change the level at which the pixels are thresholded at, with the *thresho* parameter, but I think the default is pretty good. **WARNING:** if you're observing a bright target (star, planet, nearby galaxy) where the counts are likely to be greater than the threshold value, then using this option will clip the bright parts of your exposure, likely rendering the reduced version useless. There is probably no reason to threshold (or even cosmic ray correct) very short exposures such as telluric or flux standard stars.

For the purpose of this worked example, it's safe to use the threshold, and the parameters should be set as follows

```
(do_laco=                 yes)
```

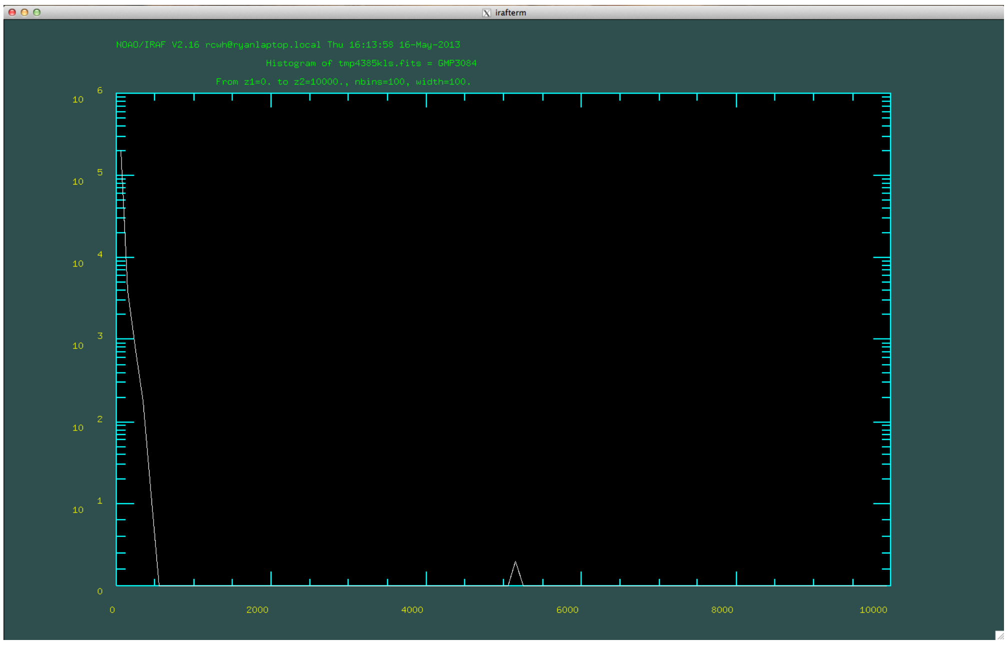


Figure 9.1: A histogram of pixel values for a single slitlet, produced after application of the La Cosmic routine. Note the small number of pixels with values > 1000; these are the 'spot' strikes, likely from low energy electrons, which La Cosmic finds hard to detect but which we can threshold out.

```
(do_thre=                  yes)
(thresho=                1000.)
```

Now you're ready to run the pipeline.

```
swift> swiftredMS (mode='h')
```

Cosmic ray detection takes around 15 mins per exposure. In this case, there are three exposures; best go get some lunch. If you're reducing a whole night's data, you'd best leave the cosmic ray stage running overnight. If you are thresholding the data for muon strikes, the pipeline displays a histogram of the pixel values of each slitlet. This is another check: the main distribution of pixel values should fall short of the threshold value (1000); only a few peaks should be present beyond this, which are the spot strikes you're cleaning (see Fig. 9.1) One thing to note: La Cosmic is very good at *detecting* cosmic rays, but it's not as infallible when it comes to replacing the affected pixels. When you come to combine your data frames, it's best to reject the bad pixels and cosmics.

When you've finished detecting cosmics, you'll find a set of new fits files in the working directory, prefixed with $\wedge$**bp**; these are the **b**ad pixel, **p**repped versions of the original input files. If you use *cat files*, you'll find they have two extentions: a SCIence extension and a Bad Pixel Mask extension (which also includes cosmics).

```
ecl> catfits bp*
```

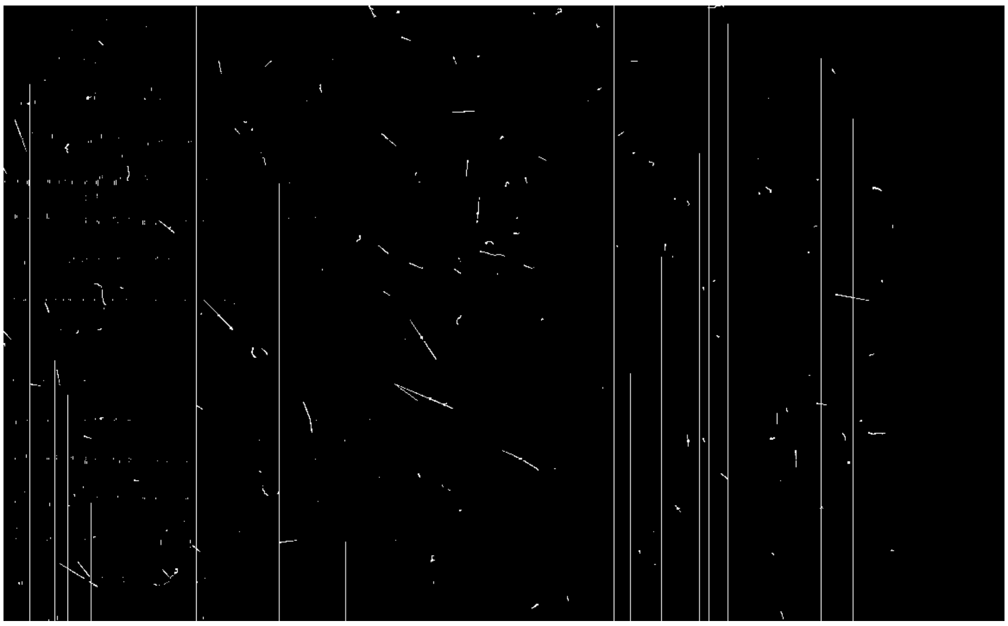


Figure 9.2: A typical bad pixels mask, which also includes cosmic rays.

```
EXT#  FITSNAME      FILENAME              EXTVE DIMENS       BITPI OBJECT

0     bpmaster074.f                                          8
1       IMAGE       SCI                         4112x2040    -32   GMP3084
2       IMAGE       BPM                         4112x2040    -32   Bad pixel ima
0     bpmaster078.f                                          8
1       IMAGE       SCI                         4112x2040    -32   GMP3084
2       IMAGE       BPM                         4112x2040    -32   Bad pixel ima
0     bpslave074.fi                                          8
1       IMAGE       SCI                         4112x2040    -32   GMP3084
2       IMAGE       BPM                         4112x2040    -32   Bad pixel ima
0     bpslave078.fi                                          8
1       IMAGE       SCI                         4112x2040    -32   GMP3084
2       IMAGE       BPM                         4112x2040    -32   Bad pixel ima
```

If you display one of the BPMs, you'll probably find a few cosmics, as in Fig. 9.2.
When complete, the pipeline returns

```
***************************************************
*               Reduction Complete!               *
***************************************************
```

Chapter 10

Spatial flexure

Pre 2010, SWIFT used to show considerable flexure, both spatially and spectrally. However, this has now been significantly reduced and we don't really find any need to correct for the spatial element anymore; its manifests as a shift or wrapping of the cube around the longest dimension. If you really think there is considerable spatial flexure in your data (which may cause a problem in the illumination correction), then keep reading. Else, skip to the next section.

This procedure attempts to remove the spatial component of the flexure in SWIFT. When the telescope moves across the sky, stresses in the instrument create flexure of components and cause the image on the detector to move by a few pixels; this motion can be in the spectral direction or in the spatial direction with the end result that subsequent exposures (lasting > 5 mins) may not align with each other on the detector. This causes problems for sky subtraction, illumination correction and (blind) spatial alignment. The first of these is solved in the next section (wavelength calibrating using sky lines). To combat the last two, here we attempt to cross-correlate a signal from the science exposure (created from sky lines) with one from the arc exposure (created from arc lines which were taken while the telescope is pointing at zenith, like all calibrations). By doing so, we can estimate the shift in the spatial direction and subsequently correct for it with a nearest neighbour interpolation, avoiding smoothing the data (although the user can specify the type of interpolation with *interp*, **nearest** neighbour is recommended for the aforementioned reason).

The first stage in the process is to isolate a signal (emission lines) in alternating

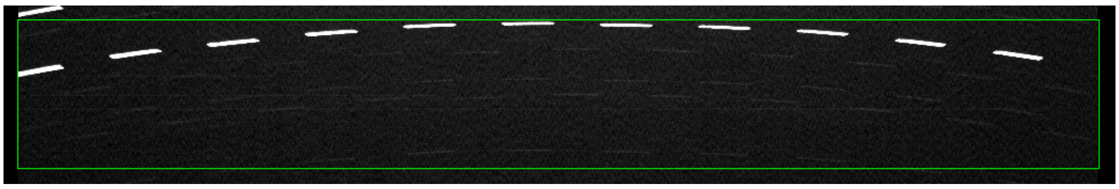


Figure 10.1: An example of a suitable *arc_sec* is shown in green: see how a bright arc emission line has been isolated in alternating (odd) slitlets.

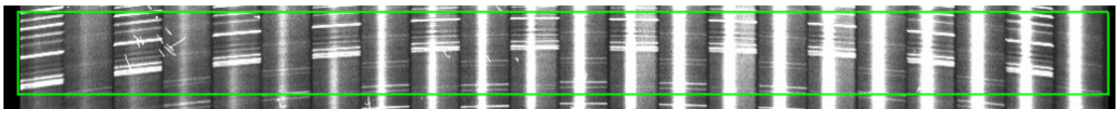


Figure 10.2: Like Fig. 10.1, we highlight a region which isolates emission lines in alternate slitlets; however, this time, the emission lines are sky lines. Note it is difficult to clearlyisolate a single skyline in alternate slitlets, but isolation of a group of bright lines is feasible. The bright vertical line passing through the slitlets is the object spectrum.

slitlets; by then summing over the spectral direction, we get a 1D image of the slitlet edges in both the arc and the science frame (which resemble histograms where alternate bins have alternate high and low levels).

This routine now has the ability to automatically select a box in the raw frames which brackets a single emission line, like in Fig. 10.1 by using *sw_sellinereg*. Set *arc_sec* and *sky_sec* to **auto** if you want to use this ability (recommended, because it adjusts the boxes for every input image using the wavelength calibrations). But make sure you set *arclam* and *skylam* to sensible values, or leave them at the defaults. You also need to give *lam0*, *dlam* and *nlam* and *sflip*.

Prior to summing over the spectral dimension, it may help to fit and remove any object spectrum in the science slitlets: this is achieved with *rmobject* which fits a low order polynomial along each column in the spectral direction and subtracts it; emission lines are not subtracted because those pixels are rejected in the fitting phase.

We then supersample (by a factor of 10) the two 1D slit images and cross-correlate them. Prior to this stage, cosmic ray hits have not been removed and so can cause difficulty in the cross-correlation. For this reason we allow the 1D slit images to be median filtered prior to supersampling; this aids in removing cosmics and forcing the edges of the slitlets to be sharp; the width of the box filter can be changed with *xwin*. The reason for supersampling is that the cross-correlation peak can then be read to subpixel accuracy.

Finally, we look at the cross-correlation peak (there is more than one, but we isolate ourselves to look for shifts smaller than *max_shift*). Rather than fitting a Gaussian, we just locate the maximum value and use the location of this maximum to calculate the relative shift between the frames; the science frame is then shifted to match the arc frame (and therefore all other calibration frames that were taken at zenith).

So if you want to make use of this recipe, turn off the BPS stage and turn on the SFLEX stage.

```
(S_BPS  =                    no)
(S_SFLEX=                   yes)
```

Now you're ready to run the pipeline.

```
swift> swiftredMS (mode='h')
```

After a bit of processing, you'll have some more files, prefixed with **sbp** (**s**hifted, **b**ad pixel corrected, **p**repped frames).

Chapter 11

Making science cubes

This is it; the time you've been waiting for. After this stage, you'll get your first glimpse at the reduced science cubes. Start by making sure the *S_BPS* and *S_SFLEX* stages are turned off, and the *S_SCICU* and *S_MSMER* stages are both turned on.

```
(S_BPS  =                     no)
(S_SFLEX=                     no)
(S_SCICU=                    yes)
(S_MSMER=                    yes)
```

The *S_SCICU* stage makes master and slave cubes from each spectrograph, while the *S_MSMER* combines the two cubes from the master and slave spectrographs, correcting for differences in illumination and gain. There's no need to run them separately, but you can do if you wish.

There are a few other parameters you should check before running the pipeline. Most obviously, you need to think about *prefix_sci*. Which files do you want to make cubes from? Clearly you need to use prepped frames, so at the very least, you should set *prefix_sci* to $\wedge$**p**. If you've bad pixel and cosmic detected & corrected the input frames with *S_BPS*, then you should pass these files by setting *prefix_sci* to $\wedge$**bp**. If you've also corrected for spatial flexure using *S_SFLEX*, you should set *prefix_sci* to $\wedge$**sbp**. Get the idea?

Take a look at the following parameters

```
(do_flat=                    yes)
(do_illu=                    yes)
(do_skyl=                    yes)
(do_hl  =                    yes)
```

These all affect the reduction of science cubes. The first asks if you want to flat field the data; we do here because we calibrated the flats earlier, but it's not essential if you want a quick look at the data. The second asks if you want to illumination correct the data; again, we do because we created the calibrations (from the halogen lamp frames); but it's not necessary for just a quick look at the data. You should now think about spectral flexure. If your exposures show clear sky lines (e.g. >5mins in the 235mas scale), then you can wavelength calibrate on them using the *do_skyl* parameter. This adds an extra interpolation, but if you need to subtract sky lines (using the ABBA or OSO sequence) then this stage is important, otherwise you won't obtain an accurate sky subtraction. Finally, the last parameter asks whether to correct for the asymmetric shear using the horizontal line calibration. This really is optional; it adds an extra interpolation and isn't often required if your science object doesn't extend to the edges of the field of view.

For this worked example, we set the parameters as follows

```
prefix_s=                ^bp
(do_flat=                yes)
(do_illu=                yes)
(do_skyl=                yes)
(do_hl  =                yes)
```

Now you're ready to run the pipeline.

```
swift> swiftredMS (mode='h')
```

Occasionally the automated wavelength calibration on the skylines can fail. When it does so, it asks you to manually identify the sky lines. Although the wavelengths are given in the data reduction manual, if you press **l**, I bet nearly all the lines will be automatically identified, leaving you to delete a few rouge identifications and then iterate the fit and detection with **l**, **f**, **q**. This problem occurs in the worked example, for slitlet 20 of slave074.fits. You're prompted with:

```
AUTOIDENTIFY: NOAO/IRAF V2.16 rcwh@ryanlaptop.local Fri 15:21:06 17-May-2013
  Spectrum                  # Found   Midpoint Dispersion        RMS
  bpslave074.ftstrip_20[*,46]  No solution found
bpslave074.ftstrip_20[*,46]: Examine identifications interactively?  (yes):
```

If you hit enter, you see a sky spectrum with no skylines identified! But don't give up, just press **l**. You should see something like Fig. 11.1. Phew! Then **f** to fit the lines. Quit with **q**, then iterate with **l**, **f**, **q** to make sure no other lines are picked up.

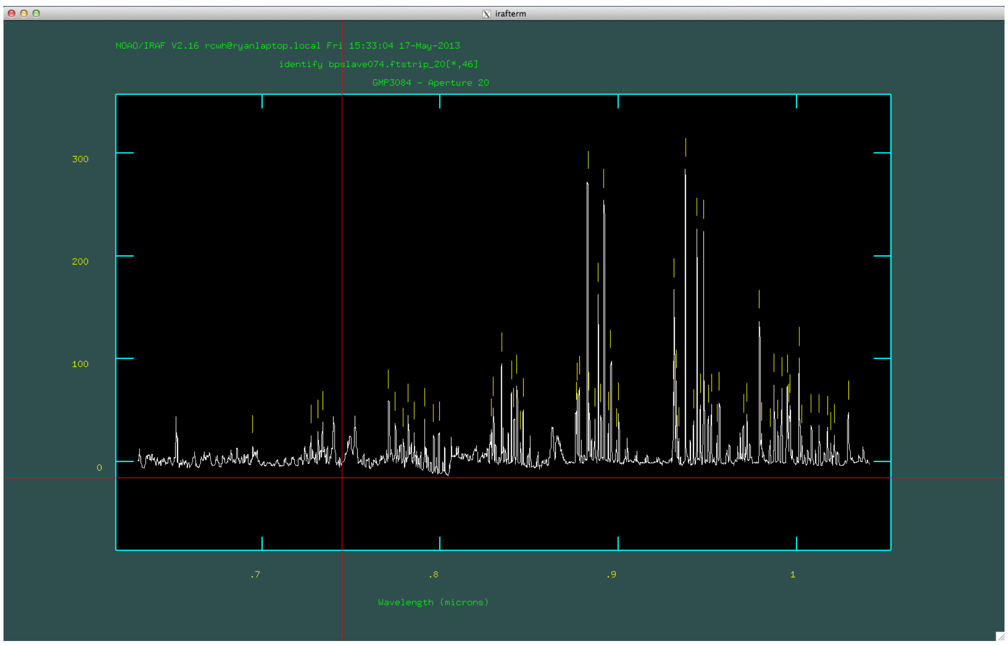

irafterm
NOAO/IRAF V2.16 rcwh@ryanlaptop.local Fri 15:33:04 17-May-2013
identify bpslave074.ftstrip_20[*,46]
GMP3084 - Aperture 20
300
200
100
0
.7
.8
.9
1
Wavelength (microns)

Figure 11.1: An example of a polynomial fit to the skylines.

Chapter 12

Putting it all together

OK, so you've got reduced science cubes. But do you have to go through all those stages if you want to reduce more cubes form the same night or run? No! All you have to do is prep them and reduce them; you can even do that in a single step. Just turn on the prepping stage, make sure the pipeline knows that it's going to get raw frames with *prefix_sci* set to `-`, then turn off prepping of all frames apart from the science frames, then turn on the cube creation and the cube merging stages. You can turn on the bad pixel and cosmic stages too if you like. So you could repeat the entire reduction of the science frames, from scratch, using a single pass of the pipeline, with the following setup.

```
swiftredMS.sci_input = "074,078"
swiftredMS.prefix_sci = "-"
swiftredMS.vl_frame = ""
swiftredMS.al_frame = ""
swiftredMS.lamp_frame = ""
swiftredMS.hl_frame = ""
swiftredMS.vlname = "VL"
swiftredMS.alname = "ARC"
swiftredMS.lampname = "LAMP"
swiftredMS.flatname = "FLAT"
swiftredMS.illumname = "ILLUM"
swiftredMS.hlname = "HL"
swiftredMS.scale = 0.235
swiftredMS.S_PREP = yes
swiftredMS.S_EXVL = no
swiftredMS.S_ARCCAL = no
swiftredMS.S_FI = no
swiftredMS.S_HL = no
swiftredMS.S_CALCUBES = no
swiftredMS.S_BPS = yes
swiftredMS.S_SFLEX = no
swiftredMS.S_SCICUBE = yes
swiftredMS.S_MSMERGE = yes
swiftredMS.p_flat = no
swiftredMS.p_vtrace = no
swiftredMS.p_arc = no
swiftredMS.p_sci = yes
swiftredMS.p_htrace = no
swiftredMS.do_b = yes
swiftredMS.do_trim = yes
swiftredMS.do_flat = yes
swiftredMS.do_illum = yes
swiftredMS.do_lacos = yes
swiftredMS.do_thresh = yes
swiftredMS.do_skylines = yes
swiftredMS.do_hl = yes
```

Chapter 13

Further Processing

Officially, that's the data reduction over. We've removed all of the instrument artefacts (as best as we can). But in practice, there will be other steps to processing the data, which many people still refer to as data reduction. Here's a bit of help with them.

13.1 Sky subtraction

Well you'll probably want to subtract off the sky and combine frames into a single frame. If you observed in ABBA mode, you can just subtract off neighbouring frames. Similarly, for OBJ-SKY-OBJ. But you'll also want to combine the bad pixel masks. What about variance frames? There is an additional routine called *doABBA* which is designed to help with these issues. It will subtract pairs of exposures and propagate the bad pixels. Furthermore, it will create a first order estimate of the variance of each pixel, assuming Poisson statistics (VAR(X)=X), but ignoring the fact that we've corrected for illumination variations. For the science frames in this worked example, here is how we set it up.

```
doABBA.sciA = "074"
doABBA.sciB = "078"
doABBA.OSO = no
doABBA.var = yes
doABBA.skyscale = no
doABBA.inter = no
doABBA.verbose = yes
doABBA.SCIlist = "SCIlist.txt"
doABBA.VARlist = "VARlist.txt"
doABBA.BPMlist = "BPMlist.txt"
```

You can see that we've paired the A and B frames. If we had a sky frame (e.g. frame number 077), we could use the following instead

```
doABBA.sciA = "074,078"
doABBA.sciB = "077,077"
doABBA.OSO = yes
doABBA.var = yes
doABBA.skyscale = no
```

Note that we activated the *OSO* flag above, so that the pipeline doesn't make B-A frames (SKY-OBJ frames in this case) as it would if the observations used the ABBA sequence. Notice also that this routine has parameters for SCIlist, VARlist and BPMlist. We'll get to how these are used next.

13.2 Combining cubes

To combine all the data into a single cube, you're best off using a python script called *cube_combine.py*, initially developed by Azin Khan (summer student at Oxford in 2012). You'll need the offsets of the cubes; to calculate these, measure the position of the galaxy in each collapsed cube. QFitsView does this if you press **g** when hovering the mouse over the galaxy). Next, subtract these coordinates off the midpoint of the SWIFT field of view (45,22). These are then the (pixel) offsets required to shift the galaxy to the centre of the FOV and you should write them in a text file, one pair of offsets per line. For example, from ms074.sci.cube.fits, I measured a position of (19.55, 26.37) and from ms078.sci.cube.fits I measured a position of (63.5, 23.7). Thus the *offsets.txt* file should read

```
 25.45 -4.37
-18.5  -1.7
```

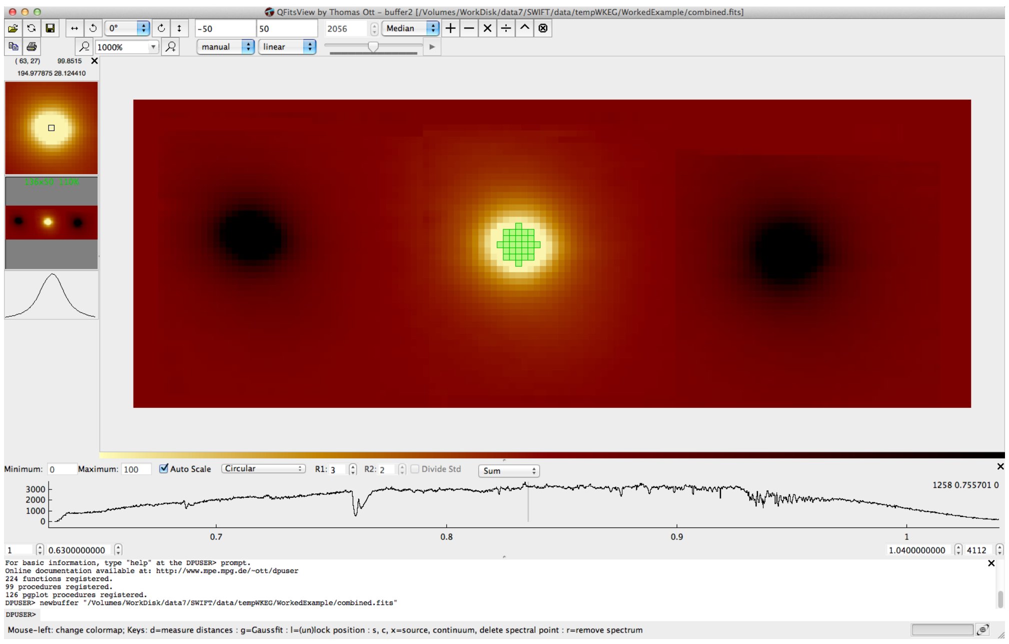
Fortunately, *doABBA* creates the other input files for you: SCIlist.txt, BPMlist.txt and VARlist.txt. these are just text files with one file per line, listing the science (A-B) exposures, the combined bad pixel frames and the first order variance frames. So long as you have Scisoft (or similar) installed (for the required python modules), you can use the following command from the (bash/tcsh) command line to combine the two (A-B, B-A) frames:

```
python [/path/]cube_combine.py SCIlist.txt offsets.txt BPMlist.txt 0.1 combined.fits quality.fits
```

If you want to know more on the input and output options, just call it without commands.

```
python [/path/]cube_combine.py
```

Now if you view the file *combined.fits*, you'll see the final output with one positive in the middle and two (redundant) negatives either side, as is normal when reducing data taken with the ABBA sequence. Obviously, only the central region is useful for science purposes.



```
For basic information, type "help" at the DPUSER> prompt.
Online documentation available at: http://www.mpe.mpg.de/~ott/dpuser
224 functions registered.
99 procedures registered.
126 pgplot procedures registered.
DPUSER> newbuffer "/Volumes/WorkDisk/data7/SWIFT/data/tempWKEG/WorkedExample/combined.fits"
DPUSER>
```

Mouse-left: change colormap; Keys: d=measure distances : g=Gaussfit : l=(un)lock position : s, c, x=source, continuum, delete spectral point : r=remove spectrum

Figure 13.1: An example of the final, combined data cube seen in QFitsView.

13.3 Correcting for DAR

The atmosphere distorts the path of the light depending on it's wavelength. This means that the position of the object appears to move in the IFU FOV with wavelength. You can estimate and correct for this effect knowing the wavelength, airmass and position angle. Fraser Clarke and Kieran O'Brien have developed a python code to do just this. It's designed to work on individual SWIFT exposures. Naturally, it adds an interpolation to the data processing. You might need to *easy_install* or *pip* the *progressbar* module.

```
python [/path/]swift_DAR.py ms074.sci.cube.fits DARms074.sci.cube.fits
```

If you want to know more on the input and output options, just call it without commands.

```
python [/path/]IFS/swift_DAR.py
```

13.4 Extracting 1D spectra

You likely want to extract a spectrum from a data cube. You can do this in QFitsView for various aperture sizes (fixed size, fixed position), or you can use the *sw_exspec* routine (fixed aperture, fitting position with wavelength). The latter uses *apphot* to perform aperture photometry on each wavelength channel *and* uses the telluric absorption lines to tweak the wavelength solution (to combat flexure). Unfortunately, depending on your IRAF version, this task may or may not work for you. With Scisoft IRAF, this task may not work; with STSci IRAF, this task probably will work.

The basic idea is that we collapse the cube into an image, identify the source (if more than one peak is found, the user is asked to identify the one of interest by typing the the number of the identification, checking by x,y coordinates), and then centroid and extract a fixed aperture spectrum at each wavelength. There is an option to rescale the aperture with wavelength; this rescaling follows the scaling of the diffraction limited PSF, not the seeing scaling with wavelength ($\lambda^{-1/5}$). As such, it's designed for AO assisted data; seeing limited data should use a fixed aperture.

You can also view and save the derived centroids as a function of wavelength. This is to check for (and potentially correct) for differential atmospheric refraction.

You can also choose to "fix" all the bad pixels before extracting the spectrum; this is quite useful as LACosmic isn't great at replacing them, so it's better to do that here before extracting the spectrum.

```
input   =        combined.fits  Give input 3D fits cube
(outsci =           spec.fits) Give name for output 2D fits science spectrum
(outerr =         specerr.fits) Give name for output 2D fits error spectrum
(outxcoo=                    ) Give fits file name for aperture x-axis position
(outycoo=                    ) Give fits file name for aperture y-axis position
(dofixpi=                  no) Fix pixels before extracting spectrum (uses 'BPM
(telwcal=                  no) Re-calibrate wavelength solution based on telluric lines
(bpmval =                   0) Value in mask representing bad pixels
(cshow  =                 yes) Show the x and y coordinate traces? (yes/no)
(apertur=                  5.) Give radial aperture size in spaxels
(apresca=                  no) Rescale aperture with wavelength (like diffractiion limi
(z1     =                  0.) Give minimum value to display
(z2     =              65000.) Give maximum value to display
(ctype  =              median) Give type of cube collapse (mean or median)
(verbose=                 yes) Print verbose output? (y/n)
```

13.5 Telluric Correction

This is best accomplished by extracting a 1D spectrum from your science target, extracting a 1D spectrum from your telluric standard, and then using the IRAF *telluric* routine to tweak the wavelength and depth of the telluric features. Then you can use *imstack* and *blkrep* to create a 3D cube of the telluric spectrum and divide your object cube(s) by it. A few regular SWIFT users (myself included) have taken to observing A0V (i.e. Vega) type stars because they're bright, easy to find near your science target, and models are readily available to divide out the underlying stellar spectrum, leaving the just telluric features. This works to around the 1% level, which is good enough in most cases. We have an IDL routine to optimise the fitting (and removal) of the A0V spectrum, email us (rcwh@astro.ox.ac.uk) if you'd like to use this code.