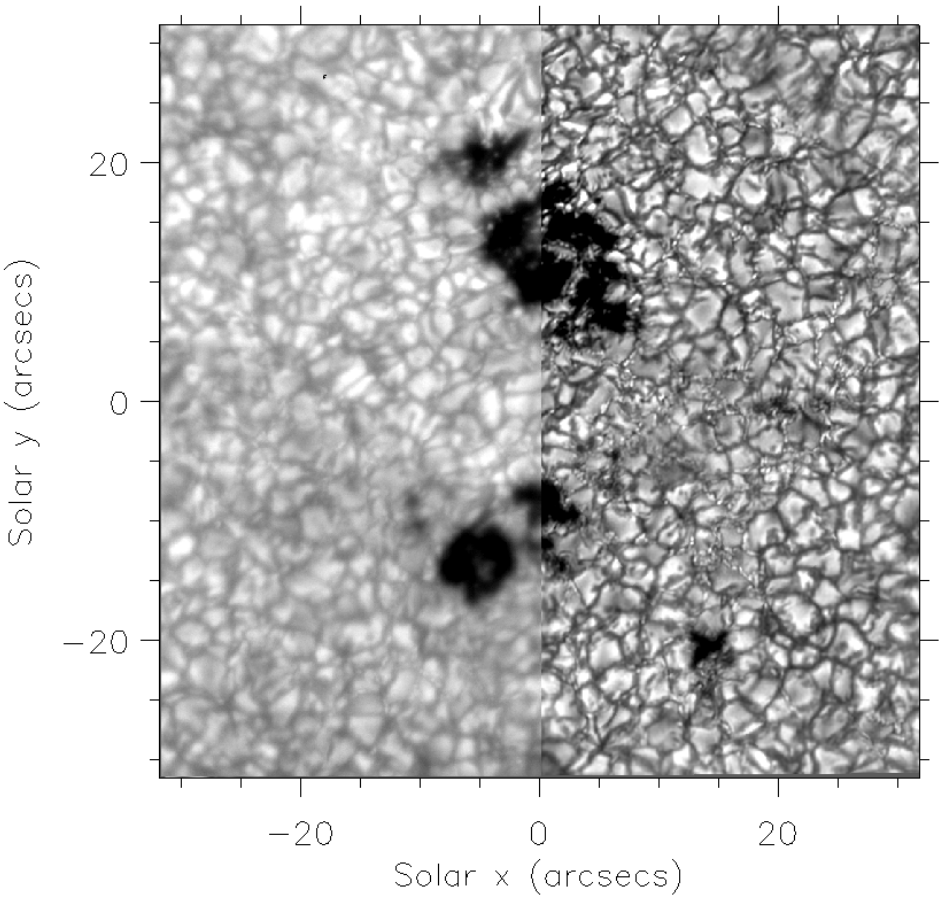
ROSA Reduction Pipeline Manual

Peter H. Keys

Queen's University Belfast



May 15, 2017

ROSA
Rapid Oscillations in the Solar Atmosphere

Queen's University Belfast

Contents

1 Before You Begin

Hopefully, the point of data reduction is evident to you if you are reading this manual and looking to use ROSA data for science. The image on the front cover of this manual is excellent in showing why we process data. The left hand side of the image on the front shows the raw ROSA data taken directly from the instrument, while the right side of the image is the effects of the processing techniques described within this manual. These processes effectively 'clean' up your data and make them science ready. Small-scale features, such as magnetic bright points, should be more clear in the processed side of this image and, in fact, we should be able to resolve features close to the diffraction limit of the telescope. These processes are necessary to fully appreciate and make full use of data from ground-based observatories. Each observatory should have it's own pipeline for each instrument, and these processes may differ significantly based on the chosen/preferred reduction techniques of the institute responsible for the instrument and also due to the instrument setup/operation mechanisms.

Within the SOLARNET project, this pipeline was improved to make it more user friendly and compliant with certain criteria specified by the project. The SOLARNET project was an EU driven project that aimed to provide access to data for EU based researchers, and to prepare for EST by consolidating and streamlining data reduction pipeline, developing best practice guidelines for pipeline development and a list of FITS keywords for complaint data. For ROSA, the processing pipeline can be simplified to:

1. Pre-speckle corrections: Dark and Flat correction. Preparation of specklegrams for reconstruction.
2. KISIP: the Kipenheuer-Institut Speckle Interferometry Package. Speckling the data.
3. Post-speckle corrections: Aligning and destretching the data. Adding in FITS header information for future use.

Before you start in processing your ROSA data I have included some useful information below to help you understand how/why certain things are done within the pipeline. If you have acquired ROSA data yourself at the Dunn Solar Telescope, some of this information should hopefully be second nature and superfluous. However, this may be useful for those of you who have access to data from ROSA through some service mode operations and, therefore, you would not have physically taken the data yourself.

1.1 ROSA

Some of the questions that I hope to shed some light on in this section are:

- *What exactly is ROSA?*
- *How is data acquired?*
- *What data products should I have before leaving?*
- *How do I take good notes (i.e., what notes should I take when I'm there)?*
- *What differences are there in the cameras?*
- *What way should I store data?*

I will not go into the details of how to actually run ROSA or the minutiae of details on the actual setup of ROSA and it's technical details as it is somewaht outside the scope of the manual. I will, however, point you towards the instrument paper (Jess et al. 2010, Sol. Phys., 261: 363 - 373) and the 'quick start' manuals for the different cameras used with ROSA for further reading. These can be found at our webpage (along with the 2010 paper). The relevant links are:
`https://star.pst.qub.ac.uk/~dbj/Publications/Jess_ROSA_instrument.pdf`
`https://star.pst.qub.ac.uk/wiki/lib/exe/fetch.php/public/research_areas/solar_physics/rosa/rosa_user_manual_small.pdf`
`https://star.pst.qub.ac.uk/wiki/lib/exe/fetch.php/public/research_areas/solar_physics/rosa/halpha_manual.pdf`

More information can be taken from the ROSA webpages (e.g. transferring data to HDDs and formatting the HDDs properly).

At its simplest, ROSA is broadband imager which can acquire co-spatial and co-temporal images at high cadence in a number of visible passbands simultaneously. Basically, you have several cameras which are triggered simultaneously from a sync box. The original ROSA computer is used to set exposures, set camera coolers, open/close shutters, define the observation mode and to store the data as it is acquired. A newer set of cameras were added in 2011 which are slightly different in operation as they have their own external windows based machines to set this sort of information through the Andor GUI. This GUI is a little more user friendly, though there are eccentricities to be aware of (more on that when appropriate). The external cameras are triggered with the ROSA cameras to ensure that they are co-temporal.

The available filters with ROSA are:

Table 1: Optical parameters of filters available for use with ROSA. As taken from Jess et al. (2010). References pertaining to where these values are established can be found in Jess et al.

Filter Name	**Central Wavelength (Å)**	**Filter Bandpass (Å)**	**Typical Exp. Time (ms)**	**Photon Count Statistics pixel^{-1} s^{-1}**	**Height of Formation (km)**
Ca II K core	3933.7	1.00	70	68 000	< 1300
Blue Continuum (4170)	4170.0	52.0	10	1	< 250
Blue Continuum (3500)	3501.0	102.00	variable	TBD	< 250
G-band	4305.5	9.20	15	1 189 000	< 250
Magnetograms (Fe I pair)	6302.5	0.21	240	80 000	< 180
UBF	variable	variable (see Fig 1.1)	variable	variable	variable
Hα core (Zeiss Filter)	6562.8	0.25	variable ($\sim$80)	42 000	$< 1\,500$
Hβ (UBF)	4861.0	0.1	200	variable	variable

The Blue continuum filters may be referred to as their central wavelength later on (i.e., '4170' or '3500') so keep that in mind. The variable tags in the table are normally given as it depends on which camera you use (older or newer ones will have different exposures as they have variable QEs etc.) or because it depends on where you tune the UBF (which line and which part of the line, wing or core). The UBF is tunable. Usually on a SOLARNET run we used Hβ through the UBF, though in the past we have also used Na I D_1. The bandpass for the UBF is wavelength dependent and is given by the plot seen in Beckers, Dickson, & Joyce (1975, AFCRL-TR-75-0090; Bedford: AFCRL) and repeated as Figure 1.1 below.

The normal day involves checking the focus of the cameras using the Air Force (AF) targets, then making sure that the cameras are aligned properly by centring the pinholes (put in the pinhole slide then make adjustments to the x and y planes of the camera until the pinhole is in the centre). The telescope technicians will need to do some sun centre alignments and work on the AO each day. Usually there needs to be a little light for these things, but not the light levels you would need when observing.

When the light level has increased enough to observe, you should then check your exposures. Take care with this because you should factor in current light levels and/or targets. If light levels are particularly low you may need to stop observing and then change it later. This will affect your flats and care needs to be taken here. Likewise, if you are say flare watching, you may want to set exposures on the low side so that you don'e over expose in the event of a flare. The original ROSA cameras have a scale of 1004 $\times$ 1002 pixels while the newer cameras have a scale of 512 $\times$ 512 pixels. The older cameras need a count of between 4000 - 6000 (absolute max should be 6500 no more or you'll saturate!) while the newer cameras need 2000 - 3500 or so (maybe it is possible to push it down to about 1800 if you're feeling brave). These are the counts needed to adequately speckle the data.

Having finished taking data it is best to take flats as soon as possible. This process can take about 20 - 30 minutes. You want to take them as close to finishing and as close to the region you observed (though make sure to not go over any AR's or really bright regions) to ensure that the flats counts are as close to the observed counts at that exposure. You need to take them as soon as you finish so that

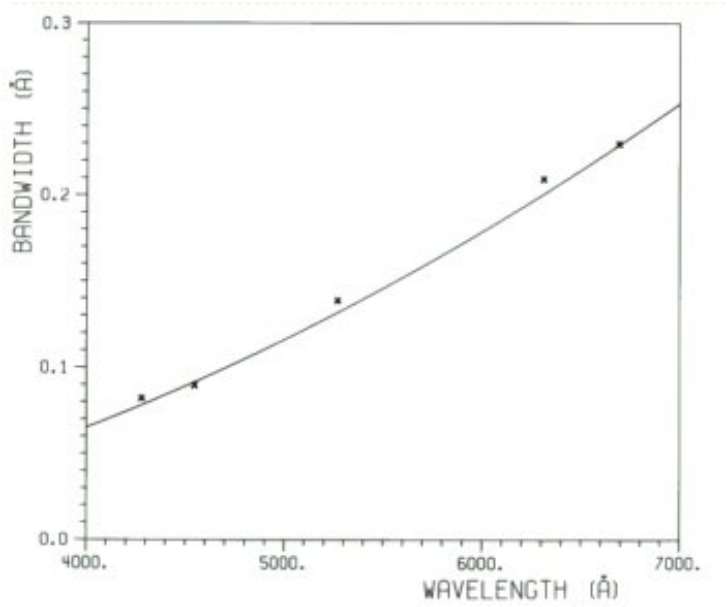


Figure 1: The bandwidth of the Universal Birefringence Filter (UBF) as a function of wavelength. Taken from the instrument paper (Beckers, Dickson, & Joyce (1975)).

light levels are similar (particularly early in the morning) and you need to take it as close to the region to account for limb darkening (i.e., an exposure at the limb may be saturating at DC). The flat helps to see if there are any fringes or gradients in the detector and are a form of calibration data for science ready images.

After the flats you need to take darks. These are taken by closing the shutter on the camera and taking data at the same exposure. This accounts for the base counts in the data as a result of thermal dark currents within the detector. You do not need as many files here than with the darks (about 3000 images on the slowest camera should be enough).

Finally, you will take a series of calibration images (i.e., AF targets, line grids, dot grids, and pinholes) for each camera. This only needs to be done once and takes about 1 minute per calibration file to do. You only have to do this once in a day, as it can be used for each target as the values will be the same. Remember, for darks and flats, you need to take a set of darks and flats for *each* exposure you take, i.e., if you have 2 targets in a day with 2 different exposures, you need to repeat flats/darks for each exposure. If only one camera has changes exposure between pointings you'll only need to take additional flats/darks for this camera (though if it is a 512×512 camera it is often synced from one of the continuum channels, so you need to run it as well to get the flats/darks triggered on the 512×512 camera although you can safely delete these excess flats/darks for the camera that is being used to trigger the newer camera afterwards).

1.2 The ROSA cameras

All cameras are from Andor.

The original ROSA cameras have dimensions 1004 x 1002 (or simply later stated as 1k x 1k) and are operated through the main ROSA machine, which runs Linux RedHat.

The newer ROSA cameras are 512 x 512 and are operated through two external Windows machines using the Andor SOLIS software. They are triggered by the older cameras. One has a high QE in the blue ('DJCcam') while one has a high QE in the red ('HARDcam'). This means that the exposures needed were reduced in comparison to the original cameras so we can push to higher frame rates.

If I can think of anymore useful information on these cameras not already mentioned I will add it later.

1.3 How files are stored

To understand the way that the pipeline works it is essential to understand how the cameras store data and operate. The observing sequence is like this: you hit 'Take signal' on the 512 x 512 cameras which opens a file and waits (so long as you have chosen 'External / External Start' observing modes in the camera menu) for the trigger. The trigger is sent when you hit 'Start Observing' on the main ROSA computer. This opens up the file for the main ROSA cameras (1k x 1k) and starts taking data. The name of this file will be the das machine number with the time stamp when you hit start observe. The name of the 512 x512 camera will be the time when you hit 'Take Signal'. There is likely to be a couple of seconds difference between the time of the files across machines then. This is why you take good notes. The time of the first image across all cameras is the same though and is given by the tie stamp on the 1k x 1k cameras (so long as you set the external cameras up right). Data is appended to these files until a new file is needed to be opened then a trigger opens up the next file. This is why you need to set the external cameras up with the faster running 1k x 1k cameras, so they are triggered fairly quickly. One of the technicians should have set it up in the right way though.

For the 1k x 1k cameras the files are stored as '.fts' which is just a FITS format file. There are a total of 256 images in a complete file. The 512 x 512 cameras save the images in the usual .fits format with naming given as the camera and the time stamp from 'Take Signal'. They store a total of 4095 images in a complete file. This becomes important later on.

When you finish taking data, I would suggest that you make a separate file in the data directory on the machines and move the data there. The file should be the date of observations, then if necessary the target. I would suggest that you add the relevant darks/flats for each target into the right sub-directory. For the 1k x 1k cameras, take 10 dark files (256 x 10 images) and 20 flat files (256 x 20 images). Avoid keeping flats with clouds if possible. In the 512 x 512 cameras 1 full file for darks/flats (4095 images) is best practice.

I would also say that you should rename the targets, grids, dots, pinholes, dark and flat calibrations for the 512 x 512 cameras to make it easier later (and add in the '0' for file 1 – 9 so it reads 'X01', 'X02' etc..) This makes life easier when you get home. The 1k x 1k cameras have options that append the file names with 'dark' and 'flat' form the menu when taking the data. You need to use these. I would also stress to take good notes while you are there. Take the times when you took data and calibration images, are there any clouds in the data, did you have to stop to redo AO (make sure you stop for a new flat if you need to, don'y just keep running as it is a nightmare later on to separate data files form flat files to get a time sequence), what was the pointing, what was the seeing like, what was the target, when did you start/stop a sequence, was IRIS/Hinode pointing at the same thing?

You will thank me later for taking this information. It can be a nightmare if proper notes are not taken and stored. I would also urge you to save them in a word / text file on your computer too. Notes hand written while tired in a dark observing room can be difficult to decipher when you get home.

2 Pre-speckle Processing

ROSA_reduction_pipeline.bat is the main code in the pipeline. This is your go to. Within this there is information on what parts do what and what you need to change etc. This is also where other parts of the code are called from. I will go through each section of the code in turn here based on what each section does. At present I have not set the code up so that you edit it and then run it. That may be in a future iteration (similar to how the IBIS code works). At present you need to go through each section and copy and paste the relevant code into your IDL session. This may seem tedious but ensures that you are not doing anything wrong. I would suggest that you open up an IDL session where the data you are looking to process is stored, and keep it open for the duration of your processing (i.e., don't close it when you go to speckle the data as you will need the data directory information and pointing information re-entered. It is also easier as the code that adds in header information needs the cadence output during the process). If you do close the IDL session before finishing you can get some of the information (e.g. darks/flats) back as they are saved out at the relevant sections in the code.

Also, note that the code is fairly well annotated throughout so that you know what arrays and variables do throughout, and if you need to edit an external file, it will tell you when and where to edit. This manual is probably more detailed, and provides sample outputs from various points in the pipeline

so that you can see what normal operation looks like. The annotations within the code may be useful while you are using the code. I know that this is not considered best practice for 'clean code', but I don't really care. My job was to create a code that was simple for the layman to understand and operate. I hope I have achieved that.

2.1 Setting up file systems

The first section of the code requires you to edit and run a batch file through IDL called *ROSA_directory_generator.bat*. This does the following:

- Split as days, targets & filter e.g. 27Jun2009/AR11023/Gband/
- Have to have a raw folder for pre-processed data
- Have to have a folder for speckled data
- Speckled data split as speckled, mid_processed, processed
- speckled = have been run through KISIP and are speckled (nothing more)
- mid_processed = have been speckled, images aligned, and destretched
- processed = fully processed, i.e., final alignments to other cameras applied to destretched images

However, you should be very aware of the following:

Make sure you have adequate permissions to create the directory structures & enough space to process
Be careful in naming directories so you don't overwrite previous work

It is up to you to edit this file to be suitable for your setup. however, once done it is very easy to manipulate for subsequent data sets. Essentially change the 'raw_directory_path' and the 'reduced_directory_path' initially to suit your system. Then with each subsequent reduction change the date, target and PI at the top of the file. If you have only one target or do not need the PI keyword leave it blank and you will not have these paths created, it will just be stored as DDMmmYYYY/Filter. You can also change the date setup to suit your own preferences. This batch file uses the IDL code 'FILE_MKDIR' to create the directories from the IDL command line.

2.2 Initial information

Before getting into the nitty gritty, you should set up some initial parameters at the start of your processing run. The information that you need to enter are the pointing and the file paths to your data, the filter and the filter dimensions. The filter and dimensions can be commented/uncommented as necessary. I have included the most used ROSA filters here. The names given here are the names given to outputs within the code and are used to search for the relevant folders when saving data so they should be the same as those input into the directory generator batch file. I would suggest **not** changing them, it will only lead to headaches later on.

generally, the continuum channels (4170 & Gband) have camera dimensions 1004×1002, while the Ca K and Hβ are the newer 512×512 models. Be careful with the pointing information. You should have this from your notes or the notes of the telescope operator. It should be inserted as [E/W, N/S] with W and N as the positive values. This is used for information within the header for the reconstructed data. It will update this with the passage of time for your observations given this initial value taken while observing. This is necessary for SOLARNET and makes it easier to compare pointings for a given frame to, say, SDO or Hinode images.

2.3 Darks & Flats

The first stage in processing data. This is fairly straight forward. Some things this section of the code does or you should be aware of:

- You will generate the average dark/flat here
- This will be saved out to the calibration directroy of you data path for future use
- There is a separate procedure for the 1004×1002 and 512×512 cameras due to the way the images are stored in the files.
- Be careful in choosing the right dark/flat for a given exposure time
- Your flat/dark must be separate if you have multiple exposures in one day

The key things to remember here are that you have to make sure you have the right files for the right exposure times for your data when inputting the file search for your darks/flats. Using the wrong exposure will make your reconstructions not as scientifically accurate as you would like and they may be wrong if there is an extreme difference in exposures as can happen with changing from ARs to QS for Ca K. You must also make sure that you have enough images for each. you will not have a good enough dark/flat with 100 images!

When the dark/flat is finished you will get a plot on the screen showing the average counts for each for the filter you are preparing. The values will be different for the 1004×1002 and 512×512 cameras. You should expect:

- Darks: 200 - 350 counts for 1k x 1k camera, about 100 for the 512 x 512 (this camera has a lower noise floor)
- Flats: 4000 - 6000 counts for the 1k x 1k camera, 1800 - 3000 for the 512 x 512

After generating the dark/flat the average is shown as an image on the screen so that you can inspect it for any anomalies (e.g. features that shouldn't be in your flat like ARs or any strange gradients in the darks). There will be some fringes in the flats so don't worry about them. Your average flat and dark will look something like Figures 2.3 and 2.3

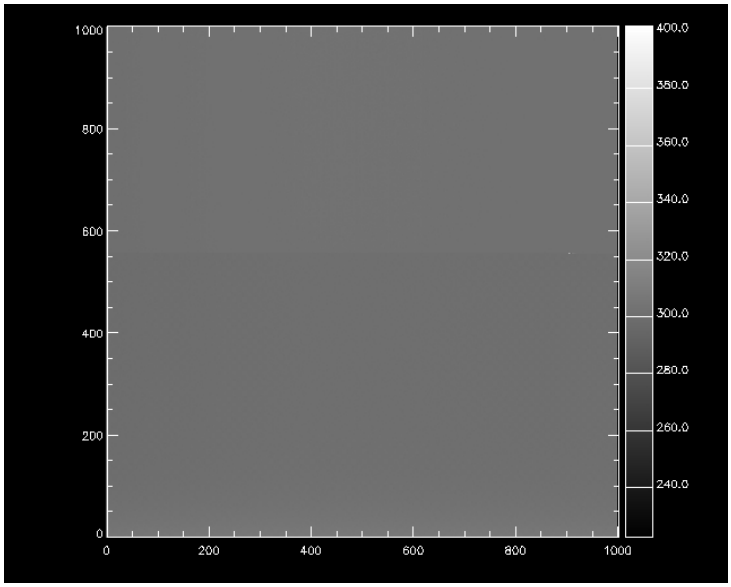


Figure 2: An example average dark image

The code then saves the average dark and flat to the relevant calibration folder for that filter in the reduced data directory in case you need it later to redo the processing or for some calibration needed in you science.

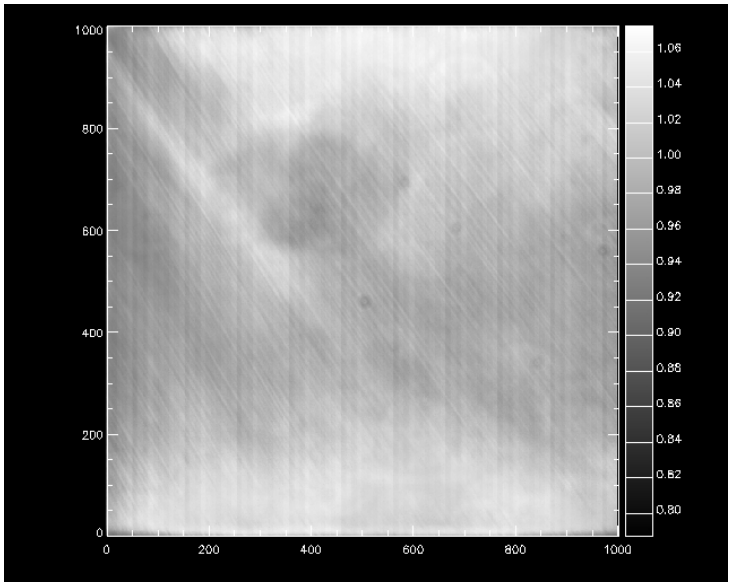


Figure 3: An example average flat-field image

2.4 Creating the Specklegram

The next section creates the average speckle gram and prepares your data for KISIP. This is a key process in determining the subsequent reduced properties of your data (i.e., the cadence!) Again, this is split for the 1004×1002 and 512×512 cameras.

This requires:

- You need to have generated the average dark/flat
- You need to have the 'Raw' directory structures present to save the specklegrams to
- Raw in this sense is the pre-speckled 'Raw' data (i.e. dark/flat subtracted raw data)
- These need to have the 'filter' and 'target' sub-directories as well
- This is where you will select the cadence of your reduced data
- This is decided by the frame rate and the number of images used in a specklegram
- For ease of operation keep as a factor of 256 for 1kx1k cameras (i.e. 32, 64...)
- Can't have too small a value or specklegram/reduced data suffers (16 is pushing it)
- In general, the worse the average seeing the larger the specklegram number needs to be

Note: **Be careful in choosing the right data files as there may be multiple pointings for a given day**

One of the most frequent questions I am asked is on timing information from ROSA data. With ROSA, all cameras are synchronised initially using a trigger box when you start observing. The cameras then run off the frame rate that they have set based on the exposure time of the camera. They are then triggered again when a new file is opened. For the 1k x 1k cameras, the name of the file has the initial time (in UT) when the observer has started to observe. This is in the header of the first raw image. Each subsequent file opened by the ROSA system until observations are stopped have the same time on the filename. Now, with the external windows machines (i.e. the 512 x 512 cameras) a similar process happens. However, the time stamp that it uses is the time when the observer hits the 'take signal' button. The camera then waits for the trigger from the sync box. Therefore, the time stamps on the old ROSA cameras and the newer 512 x 512 cameras are always going to be a few seconds off.

On some occasions, they can be several minutes off if, say, the observer goes off and makes a cup of tea while waiting for the seeing to improve. **Therefore, it is imperative that you use the 1k x 1k continuum channels for your timing information.** Remember that the initial frame across all channels will be the same time. In the past we have always used the cadence and the initial start time to work out accurate times for a specific frame. Within this version of the pipeline, the initial time is input and the cadence established at this stage in the code is used to produce the time stamp in the FITS files of the processed data.

First you have to select the files that you want to process. Remember that you need the right data for the exposure in your darks/flats in the case of multiple pointings with multiple exposure times. Also, you may have some files that are 'test' files in the data set that you may need to ignore. These are pretty obvious as they will be one file with one timestamp. these are normally remnants of exposure tests prior to actually starting an observation sequence.

The next thing you will have to set is the 'burst_number'. This number is the number of images that you want to add together to make an average specklegram. Now, for the 1k x 1k cameras this needs to be a multiple of 256 so that you have a whole number when splitting up the file, so values like: 16, 32, 64 etc. With the 512 x 512 cameras this needs to be a multiple of 4095, so values like 35, 45 etc. There is a version of the code where this is not a requirement but I haven't finalised it yet, so it may arrive in a future version of the code.

When selecting the *burst_number* you need to find a balance between a good average and what the eventual cadence is. What I mean here is that the cadence of the reduced data will be given by:

$$cadence = \frac{burst\_number}{camera\_frame\_rate} \tag{1}$$

If we take an example of the continuum, normally the frame rate of the camera is 30.3 FPS. If we use a *burst_number* of 64 we get a cadence in the reduced data set of 2.1122 s. If we use a *burst_number* of 32 we get a cadence of 1.0561 etc. You may be tempted to push this down as far as possible to maximise your cadence, however, you are likely to effect the quality of your average specklegram. For example, in the scenario above of the continuum data running at 30.3FPS, I would never use a *burst_number* below 16, this would be the absolute limit, and in fact in most cases a value of 64 is probably fine (as a 2 s cadence is more than adequate). The better your data quality, the lower this number can be. However, you should also be aware that the lower the number, the lower the cadence, and the more images you will have after reduction. If space is likely to be an issue, then you need to consider this as well. At this point I will also note that, if you use a slightly different frame rate in your camera, particularly the 1k x 1k cameras, you need to change the frame rate value when calculating the cadence. This value can be multiples of 30.3. Normally it is 30.3, but if for example, you are doing limb observations, you may have had to use a larger exposure with ROSA and may have used a different frame rate (e.g., 15.15FPS).

Note that I have added a print out to screen to show what the cadence and expected complete file numbers to expect given the value you input for the *burst_number*. Tweak if necessary after this step.

Again, the production of the average specklegram is performed individually for the 1k x 1k cameras and the 512 x 512 cameras. This is due to the number of images saved in a file as discussed earlier. You should note that, to make things easier with the 1k x 1k cameras, the final file is discarded as it is usually poor seeing (that's why you stopped observing presumably). This doesn't really make much difference as a full file is only 8 s long, so presumably there is less than 8 s of data there (or 3-4 frames max when reduced). This is important when comparing times between the 1k x 1k cameras and the 512 x 512 cameras.

In comparison this discarding of the last frame for the 512 x 512 cameras is impossible as they contain a maximum of 4095 images. At say 10 FPS that's 409 s of data which is nearly 7 minutes worth. That would be an unacceptable loss. To prevent this from happening, when creating the specklegrams for the 512 x 512 cameras the pipeline looks for a keyword within the data FITS header called 'NUMKIN' within the last file. This value is the number of images contained in the file. For all but the last, this will be 4095. Note that this keyword is not found in the FITS file headers for the 1k x 1k camera, which we would have used to get the last images to get past this issue. When the NUMKIN value is established, the code then establishes how many complete images can be created using the last file. This value is then used to setup a FOR LOOP within the pipeline to deal with these last few images. When running the pipeline, the file number for the processed specklegram is printed to screen (i.e., *Processed image number*

XXXX of YYYY, so you know how long it is going to be. With the 512 x 512 cameras, there will be a break when the new FOR LOOP starts and this information begins printing to the screen again.

With the file number is printed to the screen, you will also get an image showing sample specklegrams during the process as they are created. This will look a little like Figure 3. You will notice that this is not perfect, as it is yet to be speckled. It is only flat/dark subtracted average specklegram. (Also, note that the image on screen is slightly different to this one as the image was created specifically for a talk and is prettier than what is put to the screen). The files will be named *kisip.raw.batchXX.YYY*. This is due to some random reason that KISIP can only accept 3 digit image numbers, so for file number 250 it will be *kisip.raw.batch00.250*, whereas, file number 1056 will be *kisip.raw.batch01.056*. The speckled files will be output with a similar naming structure, but it is rectified when we change back to the FITS format after speckling.

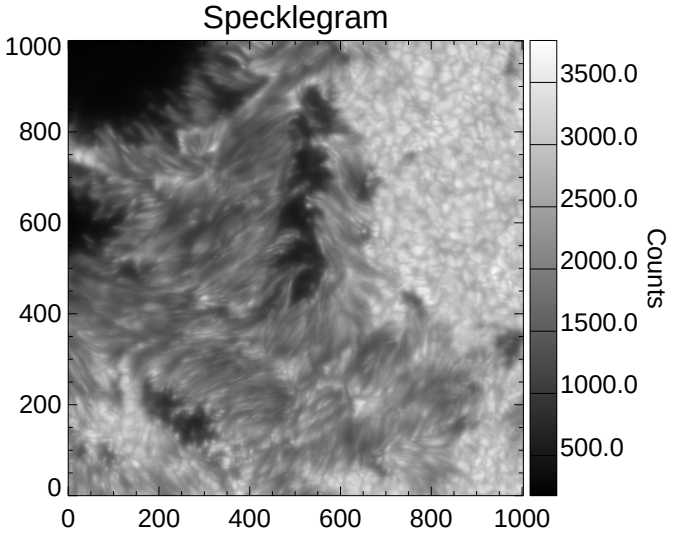


Figure 4: An example specklegram output to screen

I would also like to point out here an issue regarding the calculation of the cadence for the 512 x 512 cameras. For some reason, if you go to low with the CaK camera with the exposure, it will tell you that you are getting 100FPS, but it really defaults to the maximum, which is about 29FPS. Therefore, if you've gone below that value, don't trust the number it gives you. Also, for some reason that I haven't worked out why yet, on occasion the number that it gives you in the GUI for the frame rate is not completely right (maybe off by 0.03 FPS or slightly more). It could be that it is not factoring in readout, or some issue regarding the trigger. However, you should be aware of it. For an absolute value of the cadence I have included a calculation which relies on you knowing the number of files in the continuum ROSA cameras (as a reference) then the number in the 512 x 512 cameras using the file numbers and the NUMKIN value. Using these you can work out what the frame rate is assuming you stopped the cameras all at the one time. This can be used to establish the cadence. This isn't ideal but it seems to give correct values and alleviates that issue.

3 KISIP

There is a separate manual for KISIP, which you should read. It tells you how to set it up and run it. It can be run as a parallel job. You will need to open another window to run it, as it is traditionally run from the command line in a linux session (ANSI C Code). The manual can be found here:

http://www.arm.ac.uk/~cjn/speckle.manual.pdf

When running you need to change some of the values (See Figure **??**). The 2 most important are:

- Phase apodisation (in %): this is the apodisation window that you want during speckle in percent. A value of 25% will take 12.5% from each side of the image. This is needed to ensure that the image returns to zero at the boundaries so you don't have issues when speckling. You can get away with smaller values with better seeing. I would recommend a standard of 25% for the 1k x 1k cameras and 35% for the 512 x 512 cameras. This value reduces the number of usable pixels in the field-of-view.

- Sub-field size (in arcsec): this is the size of the sub-fields that you split the image into for reconstruction. When speckle is run the image is split up and processed then stitched back together in the final image. This value tells the code the size of these squares. The value you choose is a compromise. A lower value gives a better reduction (higher values blur the image a little), however, unless you have really good seeing a lower value may cause a 'tessellation' effect if seeing dips or in low intensity regions (e.g., sunspots). See Section 5 for an example of this effect (and other artefacts introduced in the reduction process). This value is the same for both sets of cameras as it uses the spatial sampling in the Data Properties menu to work out how many pixels this equates to.

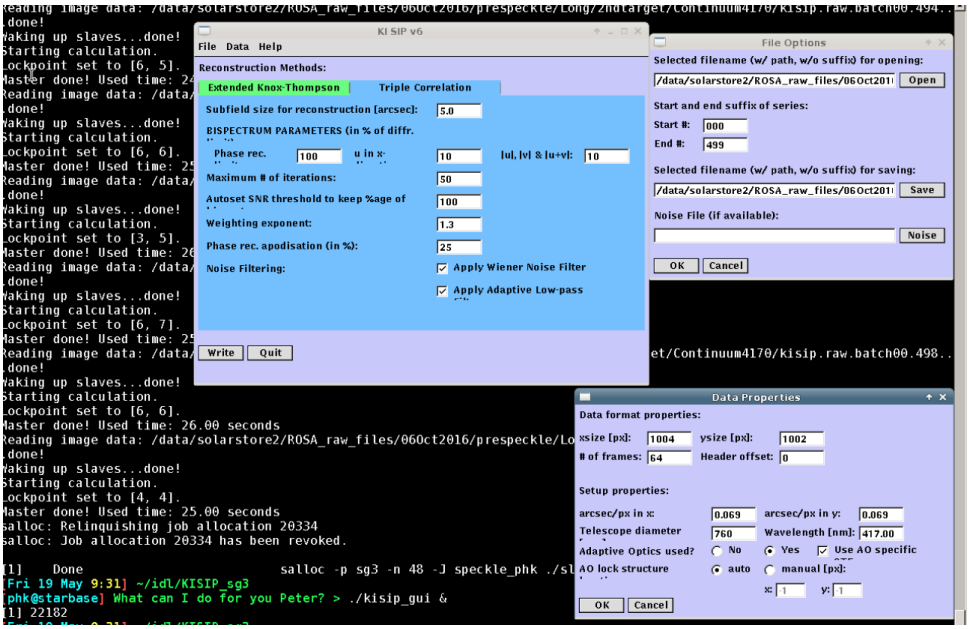


Figure 5: Screen shot of the KISIP code and the various options that you can change. Also, in the background is an example of the output to the screen on running kisip, whereby it tells you where the AO lockpoint is, how long it took to process an image and which image was processed.

You should use the Triple Correlation method. Within 'File Options' you should put in the path for the raw files and the path for where the reduced data will go to. You also have the start and end image numbers for processing. Remember that this should be a 3 digit value. In my experience, you should limit it to 500 images (particularly for the 1k x 1k cameras) as there is an issue with memory leak when trying to process too many images.

Within the 'Data Properties' drop down menu, there are a few things you need to change. These are:

- x/ysize: the pixel size of the images in x and y (1004 x 1002 or 512 x 512)

- number of frames: the burst number you used in creating an average specklgram

- arcsec/pix in x/y: the spatial smapling you used (standard for 1k x 1k is 0.069 and 0.138 for the 512 x 512 cameras)
- Wavelength: the wavelength of the filter you are processing in nm.
- Adaptive optics used: Yes/no. Tick the check box for the AO specific, and use the auto function for lock point (unless you knoe exactly the lock point location then enter it manually. Auto works quite well.)

When finished setting all these options, click 'Write'. Something is printed to screen and then hit 'Quit'. You are now ready to start the speckle process off. What happens here is that there are some files within your KISIP directory which this 'Write/Quit' process edits (i.e., the 'init_*.dat' files) and is used when KISIP runs. Technically, if you know the values you want you can just edit these files and save them with the values you need instead of using the KISIP GUI. If you are new to this though you are best to stick with the GUI for now.

I include here a sample of the values used and how to run KISIP when finished editing these parameters. This is unique to QUB as on our cluster we use queue scheduling software called 'slurm' to ensure that everyone gets fair use of the cluster (you can run NICOLE through slurm as well). Your system will differ. If in doubt talk to your systems administrator.

Normal values for data properties and running KISIP:

; START KISIP GUI
; GOTO /home/phk/idl/KISIP
; TYPE ./kisip_gui &
; FILE: OPTIONS:
; SELECTED FILENAME: /data/rosa3/oldrosa1/Speckle/Data/Raw/13Jul2011/Gband/kisip.raw.batch00
; (note: basically just leave off the final numbering of the filename)
; START: 000
; END: 999
; SELECTED FILENAME: /data/rosa3/oldrosa1/Speckle/Data/Recontructed/13Jul2011/Gband/Gband_kisip_batch00
; (note: basically the save directory minus the final numbering of the filename)
; DATA: PROPERTIES:
; XSIZE: 1004
; YSIZE: 1002
; ARCSEC/PX IN X: 0.069
; ARCSEC/PX IN Y: 0.069
; TELESCOPE DIAMETER: 760 (in mm)
; WAVELENGTH: 430.5 (in nm)
; ADAPTIVE OPTICS USED: yes and tick box
; AO LOCK STRUCTURE: auto (unless poor speckle then manually type in lock point in pixels)
; CLICK ON TRIPLE CORRELATION TAB
; SUBFIELD SIZE: 5.0
; PHASE REC: 100
; U IN X: 10
; U, V U+V: 10
; MAXIMUM ITERATIONS: 60
; AUTOSET THRESHOLD: 95
; WEIGHTING EXPONENT: 1.2
; PHASE APODISATION: 15
; TICK NOISE-FILTERING BOXES
; CLICK ON WRITE
; CLICK ON QUIT
;
; THE OLD QUEUE MANAGER THAT WE USED:
;;;OLD;;; OPEN hostfile
;;;OLD;;; LIST NODES AND CORES USED (eg r22:8)
;;;OLD;;;
;;;OLD;;; SUBMIT PARALLEL JOB USING:

;;;OLD;;; nohup mpirun -machinefile /home/phk/idl/KISIP/hostfile -n 8 /home/phk/idl/KISIP/entry < /dev/null > log.txt &
;;;OLD;;; NOTE: the "8" above is the TOTAL number of cores listed in hostfile
;
;;;OLD;;; SSH INTO wasp
;;;OLD;;; GOTO /home/phk/IDL/KISIP
;
;;;OLD;;; START THE JOB USING:
;;;OLD;;; qsub sg3_kisip.run
;;;OLD;;;
;;;OLD;;; CHECK THE JOB STATUS BY TYPING:
;;;OLD;;; qstat -l
;
;
; THIS INDICATES HOW TO RUN IT USING THE ARC MPI QUEUE MANAGER (SLURM)
; (may be different in other institutes... see KISIP manual)
; Slurm Workload Manager: https://slurm.schedmd.com
;
;
; SSH INTO starbase
; GOTO /home/phk/idl/KISIP_sg3
;
; AFTER RUNNING THE ./kisip_gui PROGRAM:
; START THE JOB USING:
; salloc -p sg3 -n 48 -J speckle_phk ./slurm_kisip_sg3.tcsh &
; OR FOR SG2:
; salloc -p sg2 -n 88 -J speckle_phk ./slurm_kisip_sg2.tcsh &
; OR FOR SG4:
; salloc -p sg4 -n 48 -J speckle_phk ./slurm_kisip_sg4.tcsh &
;
; Note here that sg2, sg3 and sg4 are just names for collected machines on our cluster
; e.g., sg2 = r11 - r21, sg3 = r23 - r26, sg4 = r27 - r29 ;
; TO VIEW THE JOBS LIST TYPE:
; squeue
;
; TO SEE AN INTERACTIVE VIEW TYPE:
; sview

4 Post-speckle Processing

Your data is now speckled, but it is still not quite science ready. There are a few more steps that need to be done to get science ready data. At this point you should return to the IDL session that you had opened up previously to start the next few steps. With KISIP you will get a file named 'kisip.spk.batchXX.log'. I would move this to the 'calib' folder in the reduced data directory at this point. This file stores the values you used to speckle the data. This should be stored in case you wish to reprocess the data later on and need to remember the values you used initially in case you need to change them.

4.1 Returning to FITS format

At this point we want to return to the FITS format with our speckled images. The speckled images will be in a folder '/Filter/speckled/' and will be called something like *kisip.spk.batch00.000.final*. The processed images have the suffix '.final' but you will have other files with the suffixes '.ct', '.subalpha', '.subamp', '.subarec', and '.subrsr'. These additional files can be deleted at this point as they aren't particularly useful now (they are a product of KISIP). **Do not delete the '.final' files yet.**

We use a program called *load_speckle_im.pro* to load in these '.final' files and convert them to FITS format. The process from here on is the same for both the 1k x 1k and 512 x 512 cameras, as now we can take the image dimensions set up in the initial parameters (Section 2.2) part of the pipeline to differentiate the cameras. Note that if you use the *load_speckle_im.pro* program outside the pipeline (e.g., to check the data or something while it is processing) you must have the right image dimensions in the keywords for the function. If not the resultant image will look diagonally stretched and you may think that speckle has failed (when it hasn't). Even using [1000,1000] instead of the correct [1004,1002] will cause this to happen, so keep it in mind.

This process is fairly simple and shouldn't cause problems. It is fairly quick too. The image number currently being processed will be output to the screen as well so yo know how quick it is running. Note that the images will be saved as *Filter_XXXXX.fits* in the 'Filter/speckled/' sub-directory in the reduced data path.

4.2 Rigidly aligning the data

We need to destretch the data to compensate for issues introduced when splitting the image into sub-fields for Speckle. However, before doing this we have to rigidly align the images. This is to ensure that any image drift throughout the observations, brought on by variations in seeing and the success of the AO lock, are accounted for. This can be tricky.

Essentially there are 3 methods (technically more methods but I'll explain in a bit) for aligning the images using cross-correlation. They are:

- Cumulative alignment
- Semi-cumulative alignment
- Frame-to-frame alignment

The success of these methods is largely down to what you observed. For AR's the cumulative alignment method is probably fine, whereas, with quiet Sun studies, you are more likely to use the frame-to-frame alignment as the cross-correlation method has issues when there isn't a long lived structure to align to. That is, the granulation will evolve in the time scale of about 5 minutes so the whole image may look completely different in a few frames. In the case of AR's the spot will always be there so it is less of an issue. In my experience it is best to try all 3 to see what gives the best results.

To do this, comment out the relevant line in the FOR LOOP and run it. You will see a plot similar to Figure 4.2 updated periodically during the process, with a print to the screen telling you which image is currently being processed. This plot gives you the x and y shifts that were calculated in the cross-correlation. You may see a sharp jump (similar to the figure), this does not mean that it has failed. It just means that there may have been a jump in the images due to the AO jumping for whatever reason (e.g. drop in seeing). This is more evident in quiet Sun data as the AO can sometimes struggle, as the AO uses the contrast between the granules and intergranular lanes for a lock. If there is a dip in seeing just for a second or 2, this contrast is lost and the AO jumps. This is why you should test the various methods and why frame-to-frame usually is best for quiet Sun.

Once this is finished I have added a test method for you to see how accurate this process was before proceeding. The method reads in the aligned images. The aligned images are saved as *aligned_XXXXX.fits* in the 'Filter/mid_processed/' sub-directory in the reduced data path. The pipeline reads these files in and makes an array taking every 10^{th} image and plays a video of it using the *xstepper* program in IDL. We use every 10^{th} image as any unacceptable shifts, and a poorly done alignment will be fairly evident. (Add in a picture of 2 frames were this has clearly failed).

Above i have mentioned that there is another way of doing the alignment. This process uses a region you manually select within the field-of-view to align each frame (based on the usual cumulative, semi-cumulative and frame-to-frame alignment processes). This method works best if you have a high contrast source to align to (e.g., a spot or pore). If you choose this method, select the region where that feature is found within the image and input the (x,y) values into x1, x2, y1, and y2. You will need to uncomment some lines in the FOR LOOP to use this approach so that the 'temp' array used for the cross-correlation has the same dimensions as the region you selected. Normally in the other approach the whole field-of-view is used.

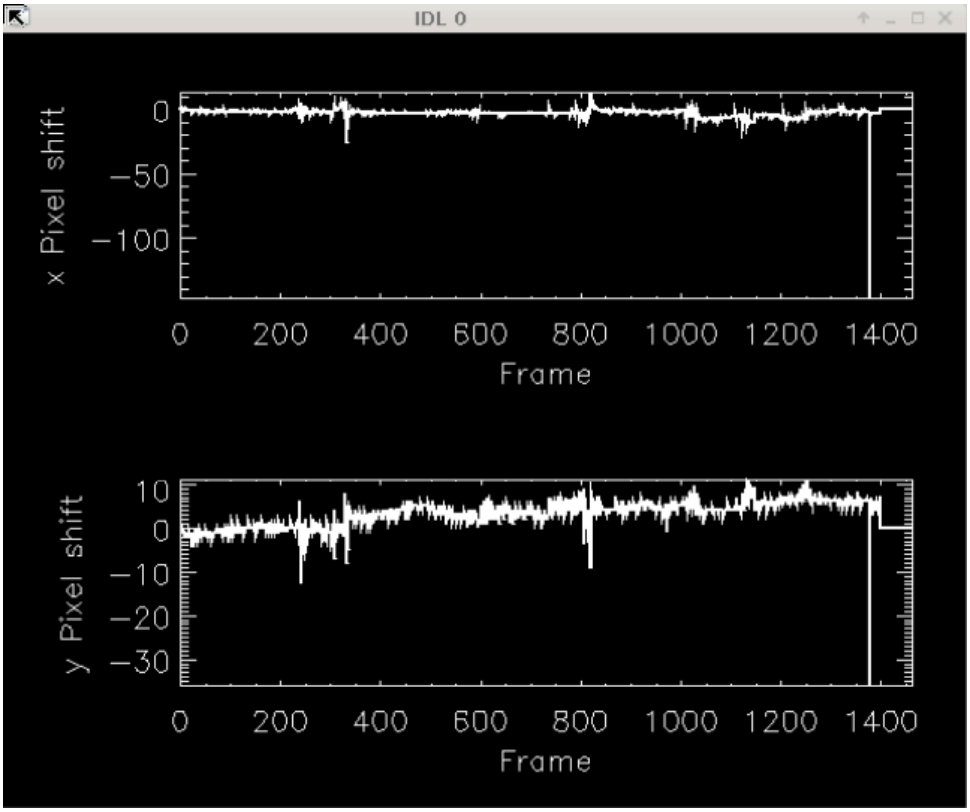


Figure 6: The plot output during rigid alignment. The image constantly updates every coulpe of frames. This is the plot on comlpeteion. The top plot is the x shifts, while the bottom plot is the y shifts calculated by the cross-correlation program.

4.3 Destretching

As mentioned previously, due to the fact that Speckle splits the image up into sub-fields we need to destretch the image. We do this as some of the sub-fields are artificially 'dragged' away from where they should be during speckle and we want to return them to where they should be. The requirements for this part of the pipeline are the following:

- You should have a fairly accurately aligned image sequence
- You need to have some ROSA programs in your IDL path
- These can be found in the 'ROSA_pipeline_init.bat' file
- You should have downloaded these with the pipeline
- At the minute, on loading these, some will fail which is okay
- I haven't got round to deleting these now defunct programs
- Will destretch the data to your 'mid_processed' folder
- You may need to change the kernel sizes depending on the data quality
- These must always be sequentially lower [e.g. boxes of 51 $\rightarrow$ 31 $\rightarrow$ 17..]
- There are some examples included of commonly used values
- After destretching the data then has a slight alignment adjustment again

- At this stage the data is essentially science ready (unless you plan on comparing to other wavelengths - in which case there are more alignments)

You need to make sure that your images are fairly well aligned from the previous Section(4.2), as your destretching will mess up completely if you don't have a properly aligned image sequence.

When you're ready to start you should initialize the batch file 'ROSA_pipeline_init.bat'. This basically dot r's all the necessary programs that you will use in this part of the pipeline. A lot of these are fairly standard IDL routines, some are created just for use with ROSA. You should see that some fail to load as well. Don't worry about that as some of these are now defunct processes that I haven't got round to removing from the batch file. The main codes are the *ROSA_destretch_noRAM4.pro* and the usual *myhanning* codes that are used in numerous pipelines.

The next step loads in your aligned files (from '/Filter/aligned_*.fits'). You will now see an array called '*kernels*'. This is important and requires your input. Effectively this is choosing the kernel sizes that you destretch with. There must be at least 3 values here, and they must decrease sequentially (e.g., $51 \rightarrow 31 \rightarrow 17$). They must also be odd numbers due to the way the box is created (i.e, even number of pixels either side of the centre point). The lower the initial number (and subsequent final number) the finer the destretching process. This can be a bit tricky and you may need to redo the detretching with different kernel values to see what gives the best result. There are some examples of values we tend to use that you can uncomment/comment. I would say use bigger numbers for poorer quality data and quiet Sun data, but check to be sure you are satisfied after.

You must also specify a '*lower_threshold_mask*' value. This is given as a percentage of the width of the image. This value depends on the dimensions of the camera that you are using and the seeing. It should be smaller for larger camera dimensions (i.e., the 1k x 1k camera) and it should be larger for poorer quality data. This value is used in the '*disp_mask*' array for applying a mask while destretching. There is a test of the *disp_mask* array whereby a noisy image is plotted with a contour showing the *disp_mask*. This mask should be within the apodisation window (i.e., the grey bar that you see around the image introduced to the image by speckle).

Next you calculate the destretch vectors which uses *ROSA_destretch_noRAM4.pro*. You will get outputs like Figure 4.3 and 4.3 to the screen. This shows the kernels being plotted and getting successively smaller for each individual frame in the destretching process. The destretched images are saved to '/Filter/mid_processed/' as 'destretched_XXXXX.fits'. After destretching you should save the vectors to the 'calib' directory in the reduced path (using the save line in the pipeline). This ensures that you can quickly redo the detretching in the future.

With the images plotted during destretching, you will also get a print out to the screen showing how much has been done in % and how long it is expected to complete given the current processing speed. This is quite useful as destretching (along with speckling) is the one of the longest processes computationally speaking in the pipeline. For continuum data it can take several hours (maybe a half to a full day) for destretching about 1000 images. It is best to set it off running and return later. With the 512 x 512 cameras it is considerably quicker, as you may expect. Generally, there are less images with these cameras too so it takes maybe an hour or 2 for these images to be destretched.

After the destretch and saving of the destretch vectors is completed, you should do a second rigid alignment to remove any residual shifts in the data. This is the same as the process described in Section 4.2, so I won't go over it again. Again, there are the cumulative, semi-cumulative and frame-to-frame methods as well as an alternative 'quick' method of removing residuals (which I would only recommend if the others don't work). Upon completion, this data will be saved in 'Filter/processed/' as 'destretched_XXXXX.fits'.

4.4 Aligning the cameras

At this stage, you should have fully processed all cameras. What you'll want to do now is align each camera with respect to one another so that comparisons between cameras/filters can be done for science. Unfortunately, this is quite a tedious process.

If you remember, we will have taken some calibration data at the telescope (by putting an acetate sheet in from to the beam). An example of the calibration images we took can be seen in Figure **??**, and they and their uses are:

- AF Targets: for bulk rotations.

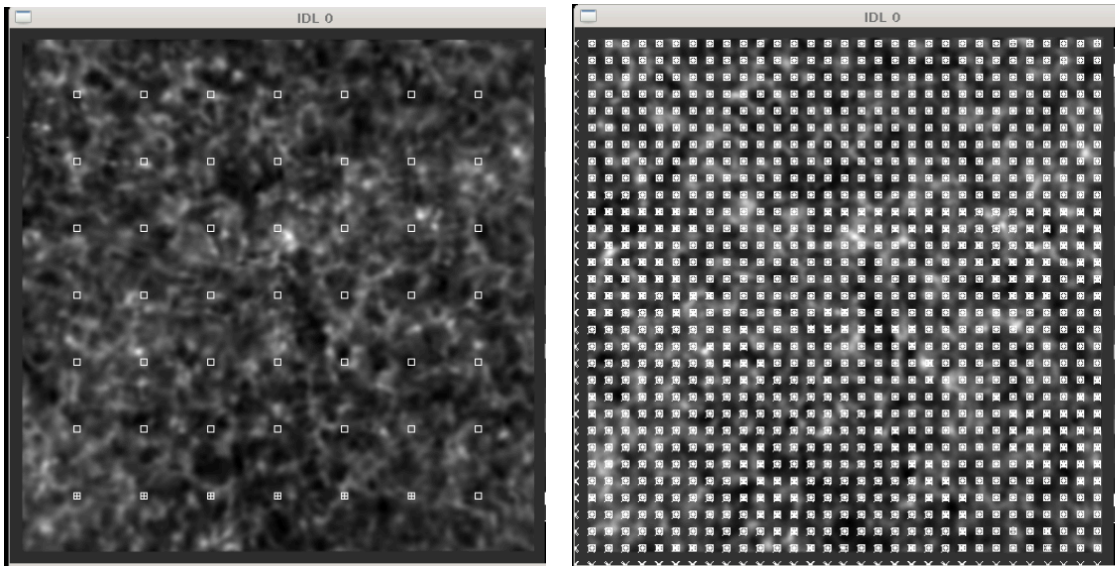


Figure 7: An example of the screen output during destretching. The left image is the second kernel destretch caluclation. Notice also the apodisation windows present as a grey band at the edge of the plot. This is due to speckle reducing the usable field-of-veiw by a few pixels around the image. The right image is the same as the left image, though here we show the third kernel for the same destretch calculation.

- Line grids: for fine scale scaling calculations
- Dot grids: for fine scale rotations
- Pinholes: for establishing offsets in x and y

Initially, you will need to load in and create these images for each camera. This is where good note taking is appreciated, as you will have to have recorded the times for which you took these images when you were observing (they can be taken in any order and you only need one set per day). The necessary images are loaded in and generated using '*ROSA_calibration_file_load.bat*'. All you have to do is change the file paths at the start to ensure that you are loading in the right raw files for each calibration image and each camera, save the batch file and then run it in the IDL command. This will create an average for each calibration image for each camera and save it as a an array called 'Filter_calib'. These are then saved out as a .sav file to your reduced path (not the calib folder as all cameras for one day are saved together). The save file is called 'Alignment_calibration_images.sav'. These need to be generated before the next stage.

The next stage is the tough part, and I would suggest aligning to IBIS at the same time, as it will save some headaches later. You want to align everything to the 4170 continuum, as this is the easiest to align to both the IBIS whitelight channel and the HMI continuum images should you wish to use those data sets at a later stage. First off the pipeline loads in the save file you just created with the arrays of all the calibration images for each camera. You now need to work out the correct rotations visually as the pipeline will plot all of the Targets for each camera. This is a little trial and error. Figure **??** gives an example of how it is done. In the left image we have the G-band and Ca K targets. It is clear that due to the splitting of the light in the optical setup that there is some sort of gross rotation in the images. Normally G-band and the 4170 Continuum cameras are beside each other, so they will have the same orientations (usually), so in this case we can use the G-band as a surrogate for the 4170 in aligning the Ca K. Visual inspection tells us that a rotation of '5' was needed here (using the inbuilt IDL 'ROTATE' function), so all images will need to be rotated by '5' here to have the same orientation as the continuum.

Now that you have this value, comes the tricky part. You now need to calculate the scale, fine-scale rotation and offsets (in x and y) for each camera (aside from the 4170 continuum). In the past we did this by essentially trial and error by looking at the effect of changing the scale and rotation values then, when satisfied, we would calculate the offsets with a cross correlation shift. Now, however, we have

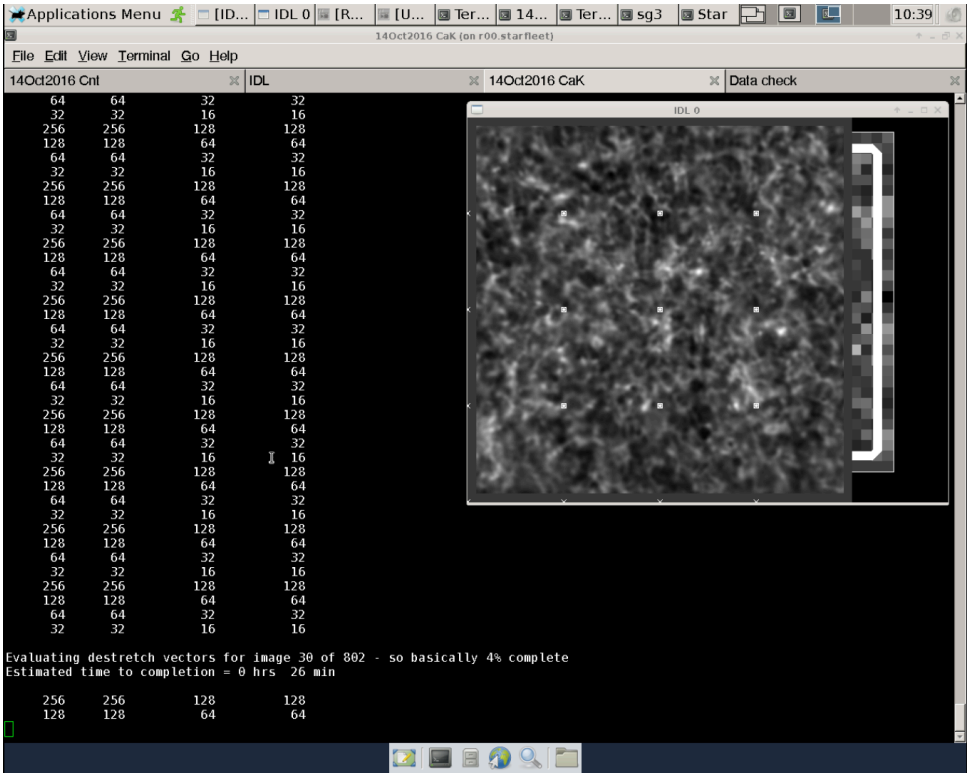


Figure 8: An example of the whole screen during the destretching process. In the plot you see the kernels being used to estimate the destretch vectors (with a residual of the disp_mask from earlier in the right corner of the plot - ignore that part). This is the first kernel destretch caluclation. You can see in the screen the values for the kernels being printed to the screen with an estimate as to how lfar along the destretch is and how long it will take. The vectors printed to the screen are saved to the calib directory after completion of the process. This is Ca K data running on the 512 x 512 cameras, therefore, it is a good deal quicker (26 minutes in this case) than those of the 1k x 1k cameras (continuum channels usually).

developed a slightly more sophisticated approach to establish these values. The new approach removes a lot of the onus on the user to calculate the values, and instead relies on more computational methods to calculate it.

In the new approach we use the dot grid to calculate these values. The pipeline uses the '*dbj_coalign_images.pro*' program to do this. The program uses the fact that the dot grid is fairly uniform to establish the scales, rotations and offsets by using Fourier transforms and cross-correlation techniques to establish the values. **You must have the dot grid image you are trying to align rotated properly (in terms of gross rotation - not fine-scale) prior to running the code.** This code takes a good while to run (maybe 2 days per filter if you are using the really accurate method). The code accepts the keyword 'demagnification=2' to speed up the process, but at the expense of coalignment accuracy, so be aware of that. At the end of the process, the accuracy of the method is printed to the screen and you will also have a difference image of the newly aligned images to see how well the process performed.

At the end of this process, you will get some values printed to the screen to for the scale, fine scale rotation and offsets. You should input these into the appropriate place in the pipeline. The values will then be saved as a set of string arrays into the reduced file path, so that you can easily find the values again if needed later on (if you are re-processing the data or if you are looking to apply the values to another pointing on the same day, when these values will remain valid). This will save some time and the tedium of re-doing this step. You should be able to get a coalignment accuracy of 97% – 99% or thereabouts.

When these values have been calculated you will need to apply them to the reduced data using the

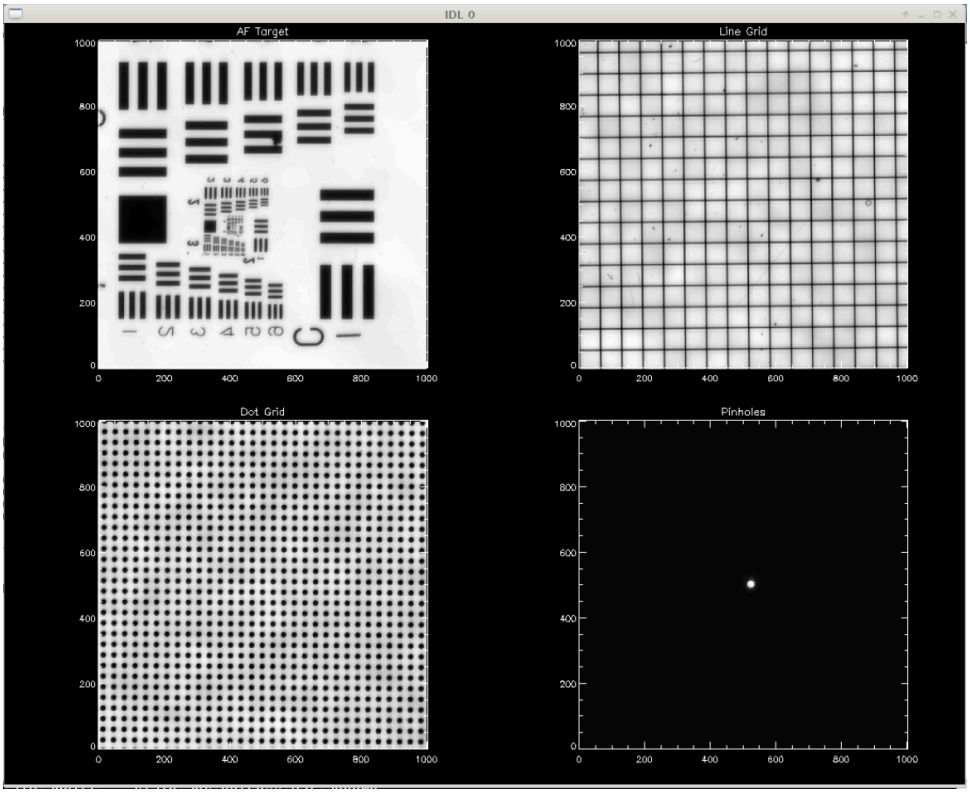


Figure 9: An example of the calibration images for one camera. Clockwise from top left are: Target, Line Grid, Pinholes and Dot grids. These are all for the G-band camera.

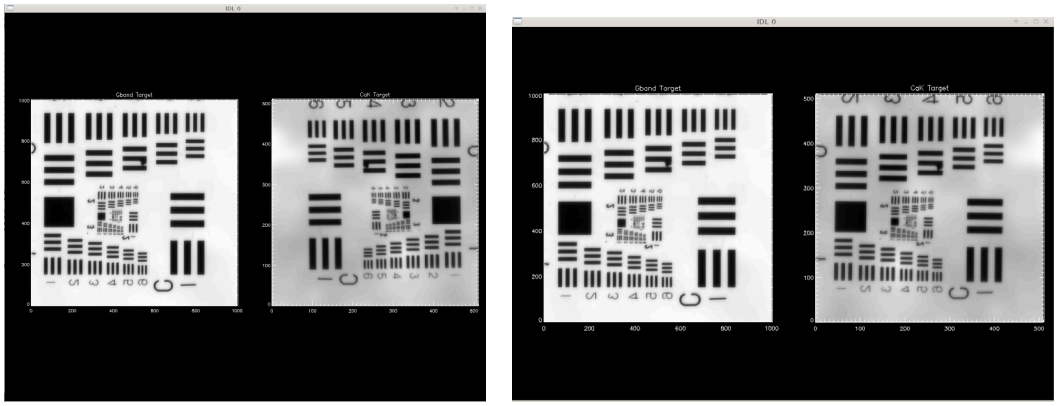


Figure 10: An example of the gross rotation calibration using the targets. The left image is the uncorrected comparison of G-band to Ca K. It is clear there is a need of rotation as the images are mirrored (in this case the rotation value needed was '5'). The right hand side shows the same two images only the Ca K image has been rotated to align with the G-band target.

pipeline. The name that will be output with the files will be the same as the input files and the same path. That means you can leave it completely to the end and you will have the SOLARNET complaint headers (described in the next section) already created or you can do it prior to that and then generate the final SOLARNET compliant FITS files and header information at the very end.

4.5 The FITS header information

This is one of the last key parts of the pipeline, and was a requirement of the SOLARNET project. This last step adds in vital information to the FITS header for each camera, which should hopefully be useful for the observer. This uses some of the initial information that we put in at the very start of the process (e.g. pointing). This process does two things; it changes the filename of the processed data into the format 'ROSA_Filter_YYYYMMDD_hhmmss.sss.fits', which matches the style of other instruments, and it adds information to the header of this processed data. These happen at the one time.

First off you need to edit the batch file '*ROSA_data_properties.bat*'. You need to add in the initial time of the observations (which is taken from the filename of the raw file timestamp in the continuum 1k x 1k raw data sets). You need to add this in the format given in the batch file (making sure to put 'Z' at the end). This is necessary to work out the pointing in each frame. You will also need to edit a few more things. Everything you need to edit is:

- Initial time (LINE 145)
- Comment in /out the relevant filter that you are currently processing (LINES 90:133)
- Put in the target you were observing. This doesn't need to be specific just 'QS' will do. If you have the NOAA AR number put that in. (LINE 185)
- Put in the PI for the data. If you are the PI that's you. If you are processing for someone else, put their name in (LINE 251)
- Exposure time for the raw data (LINE 271)
- Put in the spatial sampling used for this data (LINE 283)

I will at some stage alter this so that you do not have to comment or change much in this file, but at the moment this is how it is. In the future I will probably have something that uses IF statements to work out which filter you used and the associated properties. I will probably move the PI information and start time at some stage too, maybe to the initial setup. Other properties such as cadence are taken from the values you established earlier. If you have exited your IDL session before now you will need to re-enter these values or else you will get a NaN in your FITS header. Note that the start and end times of the data are stored in the header and the time that the file was created. The time stamp for the data is currently in the filename. I will at some stage, more than likely add the time stamp to the FITS header as well to make it easier to search for data from a list of headers.

The purpose of the FITS headers is to add valuable information in the case that someone who did not take or process this data can make use of (e.g. a new Ph.D student). It is also useful to have in case you lose your notes at any stage, at least you will know the data properties quickly. This feature will come useful when the data is injetced into a virtual observatory (this was tested with SOLARNET and ROSA data was successfully input into a virtual observatory which you could search for data based on time stamp, feature and filter used).

At present in the pipeline, after you load in this information with the batch file, the time stamp for each file is calculated using a FOR LOOP. The initial time is changed to the year, month, day etc. using the IDL routine 'TIMESTAMPTOVALUES'. Each part of the loop then adds in the time from the start (using the loop number and the cadence) to add on to this to get the new HHMMSS.sss for the data. I couldn't find a decent inbuilt way of doing this, so I made my own way of doing it using IF statements to add in an extra digit if the mins or seconds went over 60. It is perhaps not the prettiest code but it works. This time information is then used to calculate the new position on the Sun for you pointing in this frame (as rotation tracking is used at the DST normally to track regions). This is done using the code '*find_position.pro*' included in the pipeline. This takes the start time and location and uses the current time for that frame to work out where the new location is, before saving it as a FITS keyword. This value is not likely to change much frame-to-frame but it is still very useful. The value is taken as the centre point of the ROSA field-of-view, which is the centre point of the telescope pointing. The time (current time in UTC) in the loop is established too using 'SYSTIME' so you can remember when you processed the data. A checksum and datasum is added now too.

One last thing that we include is the keyword 'DATARMS'. There was a lot of discussion at SOLARNET meetings on keywords to include about data quality. The consensus was that it is difficult to reach

consensus on this topic due to all the varieties of instrument and what is termed 'good' or 'bad' data. In fact at different wavelengths it may be massively different what is considered 'good/bad'. Data RMS is a sort of place setter here in terms of including a keyword to tell you about the quality of the data. If the seeing drops the RMS is likely to dip as well. However, it doesn't really tell you how close you are to the diffraction limit at a given wavelength. I have adapted a code preferred by Sutterlin, Woger and Reardon for establishing the spatial resolution of a ROSA image by using rings of increasing spatial frequency and then establishing the point at which it returns to noise to establish the spatial resolution. I haven't included it in the pipeline at the minute as I want it to be more automated. It may appear in a future version of this pipeline and I will probably include a keyword which compares this value to the theoretical diffraction limited spatial resolution at that wavelength for a more robust indication of data quality over each frame, but for the moment RMS will have to do.

The complete list of keywords included at the moment is:

SIMPLE = T / Written by IDL: Tue Feb 28 15:07:58 2017
BITPIX = -32 / IEEE single precision floating point
NAXIS = 2 / Number of data axes
NAXIS1 = 1004 /
NAXIS2 = 1002 /
DATE-BEG= '2016-10-11T16:04:30.000Z' /Date of start of observation
DATE-END= '2016-10-11T16:23:55.941Z' /Date of end of observation
LEVEL = 1 /Data level of FITS file
TARGET = 'QS ' /General target of observations
WAVELNTH= 4305.50 /[Angstrom] Characteristic wavelength
WAVEMIN = 4300.90 /[Angstrom] Min wavelength covered by filter
WAVEMAX = 4310.10 /[Angstrom] Max wavelength covered by filter
WCSNAME = 'Heliocentric-Cartesian' /Required for WCS
BTYPE = 'Intensity' /Description of what the data array represents
BUNIT = 'DN ' /Units of data array
CTYPE1 = 'HPLN-TAN' /Type of coordinates along axis 1
CTYPE2 = 'HPLT-TAN' /Type of coordinates along axis 2
CUNIT1 = 'Pixels ' /Units along axis 1
CUNIT2 = 'Pixels ' /Units along axis 2
SOLARNET= 0.500000 /Fully SOLARNET compliant (1), or Partially (0.5)
DSUN-OBS= 149597780 /[km] Distance from instrument to Sun
TELESCOP= 'Dunn Solar Telescope (USA)' /Name of telescope
INSTRUME= 'ROSA ' /Name of instrument
ORIGIN = 'Queens University Belfast' /Creator of FITS file
OBSERVER= 'Peter Keys' /Operator who acquired the data
REQUESTE= 'A. Wenger' /PI for the data
CAMERA = 'Cam 1 ' /Name of camera
DETECTOR= 'Andor iXon DU-885K' /Name of detector
FILTER1 = 'Gband ' /Name of filter
ROT-COMP= 1 /Solar rotation compensation on (1)/off (0)
TEXPOSUR= 16 /[ms] Single exposure time
NSUMEXP = 64.0000 /Number of summed exposures
CADAVG = 2.11221 /[sec] Average actual cadence
SPATSAMP= 0.0690000 /[arcsecs/pix] Spatial sampling employed
HGLT-OBS= 5.00446605285 /Stonyhurst heliographic lattitude
HGLN-OBS= 1.82746376192 /Stonyhurst heliographic longitude
DATARMS = 1.03565 /[DN] Root mean square of data
DATASUM = '3430248068' / data unit checksum updated 2017-02-28T15:07:59
CHECKSUM= '0AT407Q40AQ407Q4' / HDU checksum updated 2017-02-28T15:07:59
FILENAME= 'ROSA_Gband_20161011_160431.056.fits' /
DATE = 'Tue Feb 28 15:07:58 2017' /Date of FITS file creation
END

4.6 Tidying up

At this point you are finished. However, there are a couple of other things that you should do just to check and to tidy things up a bit. You should:

- Check your data to make sure it looks okay, then double check it to be sure
- Delete excess data. There will be 'mid-processed' images that you probably no longer need which will take up a good bit of space. Delete those that aren't needed, but be careful when deleting in case a data set isn't finished properly or you want to relook at something.
- Put the observations log in the same directory as the reduced data. This will help immensely in a years time when someone is looking at the data and wants to check something. If you had co-observations with IRIS or Hinode, I would put the necessary observer identifiers in here too.
- Generate the thumbnails. Within the pipeline, at the very end I have added a bit of code that generates a sample thumbnail of the first images in the sequence. This is useful as in the future it is a quick way of looking at the data to remind yourself of quality and target. The image is saved with a date and initial time stamp for the observation (you may need to add in a '0' at present if your time is something like 1604UT as I had to change it to a HHMM format for output and I haven't bothered to correct the issue were a '0' will be missing for single digit times, which only effects the minute value. NB. I will fix this at some stage but it isn't that important to be fair).
- Enjoy the science.

5 Known Artefacts

I will put in some known image artefacts and explanations into this section when I manage to generate the necessary images. Artefacts will be 'fringing' and that 'tessellation' issue with KISIP sub-fields.

6 Acknowledgements

This pipeline and manual was developed for the SOLARNET project. SOLARNET is a project supported by the EU-FP7 under Grant Agreement 312495. This code uses elements of a previous pipeline developed by Dr. David B. Jess. I improved certain aspects of the pipeline, made it more stream lined and more user friendly. Check the website regularly for updates on the pipeline in the future:
`https://star.pst.qub.ac.uk/wiki/doku.php/public/research_areas/solar_physics/rosa_reduction_pipeline`