Processing the UAF HySpex aerial hyperspectral imagery

A user guide and best practices document

Authors: Chris Waigl, Pete Hickman, Patrick Graham, Marcel Buchhorn, Yuan Tian, Carl Schmitt

Date last revised: 2022-06-12

Background

The purpose of this document is to describe the steps necessary to process aerial imagery from a NEO HySpex dual hyperspectral VNIR/SWIR camera system to atmospherically corrected reflectance, including observer BRDF correction. It originated with the University of Alaska Fairbanks (UAF) HyLab team and relates to the way hyperspectral imagery is handled within the Alaska EPSCoR "Fire and Ice" project. It does not speak for NEO or any other vendor, but describes the experiences of the UAF team at the time of writing.

Every researcher or remote sensing analyst will have their personal preferences, tool sets to automate certain tasks, or constraints and goals that will lead to slightly different choices. This has been the case even within our team at UAF. This document does not aim at describing the one right way click by click, but to provide all stepping stones for generating science-ready data and to clarify choices and trade-offs. A competent user of the software tools should be able to complete each subtask.

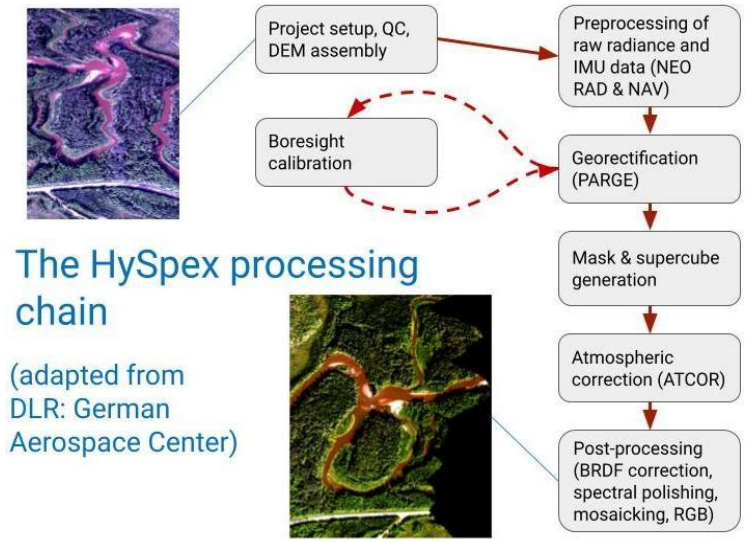


Figure 1: High-level overview of the HySpex processing chain

The Toolset

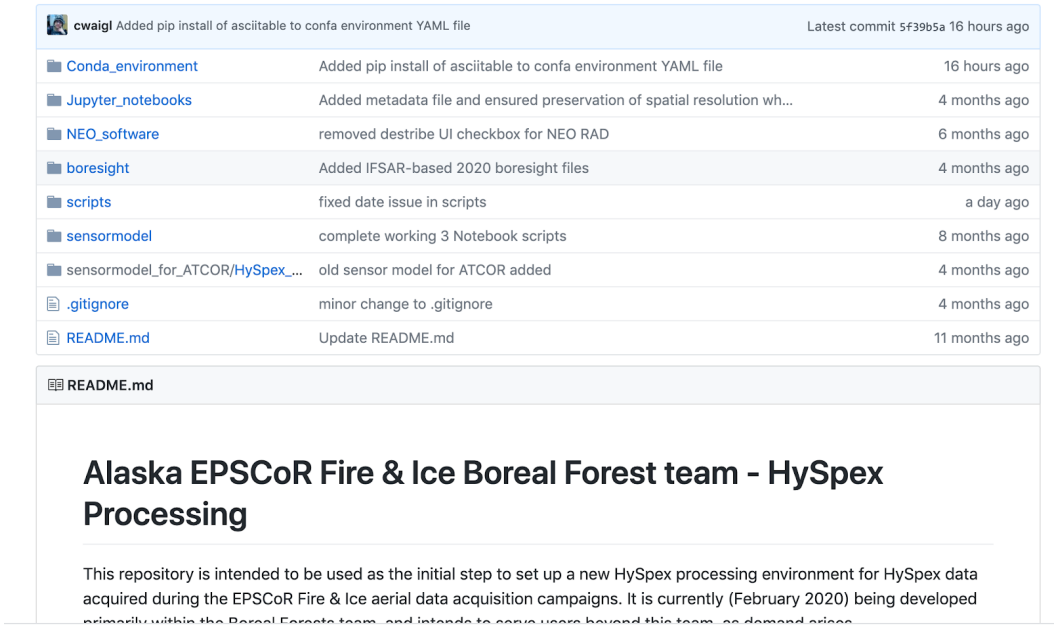
Software list

We presume every user is familiar with the following tools, or has the means to familiarize themselves with them:

- **HySpex NAV** and **RAD** (proprietary software from HySpex vendor NEO)
- **PARGE** (georectification code published by ReSe - we use v. 3.4.2 and 3.5)
- **ATCOR4** (atmospheric correction published by ReSe - we use v. 7.3.0; includes BREFCOR module for observer BRDF correction)
- **ENVI** for interactive inspection, QC, data management tasks.
- A good text editor that allows for efficient find/replace (for editing header files). Recommended: **VSCode**.
- **Python** scripts / Jupyter Notebooks for housekeeping and pre/postprocessing tasks

Script and configuration repository

We are maintaining a GitHub repository that contains a set of scripts and configuration files at https://github.alaska.edu/cwaigl/FireIce_BF_Hyspex .



Contents:

- Helper instructions to set up a Python environment using conda (see below) .
- Jupyter Notebooks for interactive setup and preprocessing tasks
- *[internal only]* A copy of the NEO HySpex NAV and RAD software

- *[internal only]* Boresight calibration files
- Helper scripts in Python
- *[internal only]* Sensor model for the specific sensor (for PARGE)
- *[internal only]* Sensor model for the specific sensor (for ATCOR4)

(The points marked "internal only" will be replaced with placeholders in the public version. We recommend to fork the repository and put your own files in the respective locations.)

Regarding Python

You can install a Python environment (we are currently using 3.7) in any way you like. The GitHub repository contains full instructions about how to get an environment with all relevant libraries using the Anaconda (in fact, miniconda) distribution and packaging tool, including a YAML file that can be used to create the environment in one command.

The Appendix has a list of the Python libraries that are not in the standard library. Together with their dependencies they should cover all you need.

Environment setup and taking stock

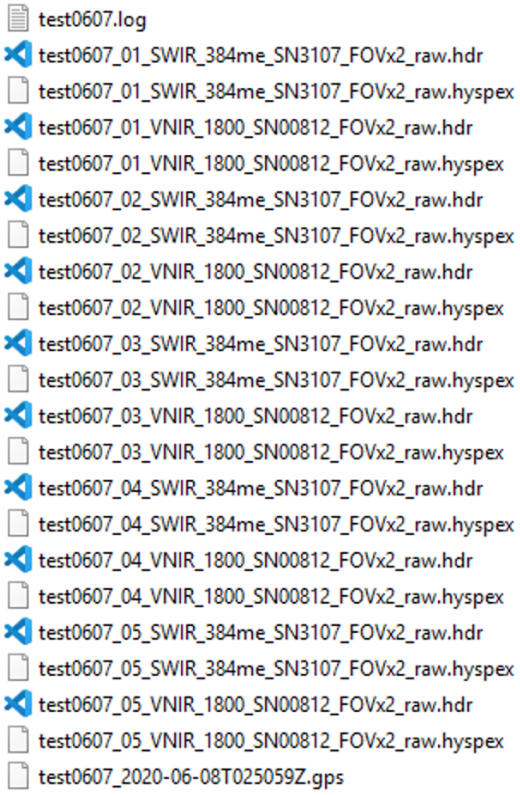
Inventory

For this step and the next ("Preprocessing") we presume you have cloned or downloaded the GitHub repository and have Jupyter Notebooks, HySpex software, boresight and sensor files all at your disposal in the respective locations of your cloned repo. If you don't use the Jupyter Notebooks, you will need to complete each step manually.

We also presume that you have all the files from the image acquisition flight downloaded from the HySpex AIR DAU (Data Acquisition Unit) into your processing computer, and reside all in the same directory. Data files start with a prefix chosen during image acquisition A full set of files is comprised of:

- ONE log file (.log) which catalogs the other files and contains minimal metadata [prefix].log (~1 kB)
- ONE binary file containing IMU and GPS data for the entire acquisition session [prefix].gps (typically several 100 MB). Sometimes, the software generates multiple GPS files, but only one is large enough to contain the whole dataset. (This can be due to issues with the connectivity in the aircraft and should be troubleshot. The data is, however usually useable.)
- For each flightline, two raw data files (.hyspex) and two header files (.hdr), one pair each from each camera (if you run a single camera system, only one pair of files): [prefix]_[flightline]_VNIR_*_raw.hyspex, [prefix]_[flightline]_VNIR_*_raw.hdr,

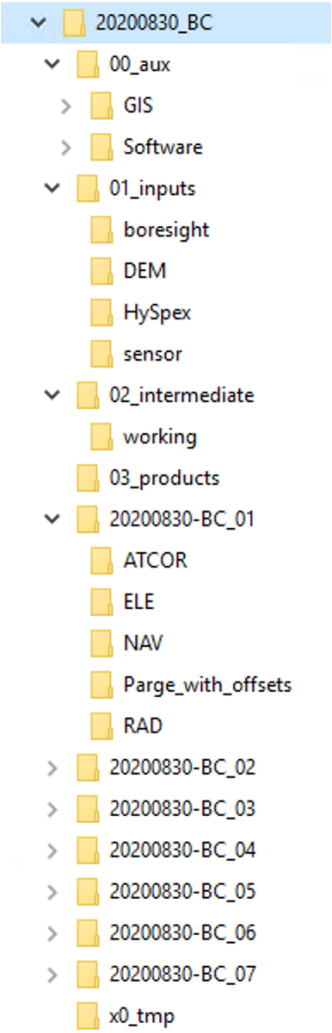
[prefix]_[flightline]_SWIR_*_raw.hyspex, [prefix]_[flightline]_SWIR_*_raw.hdr . In the next screenshot, the prefix was "test0607" and there were 5 flightlines.



Steps to prepare for georectification

We will proceed by the following series of steps:

1. Create a project folder structure to hold the processing files for the flight dataset [**Jupyter Notebook 01**]
2. Preprocess .hyspex files to radiance rasters and .gps files to per-flightline geolocation .txt files. **These two steps can be done in any order.** [**HySpex RAD and HySpex NAV]**
3. Convert GPS .txt files to GIS Shapefiles [**Jupyter Notebook 02**]
4. Create per-flightline folders in the project directory and move all the per-flightline files (output of RAD and NAV) inside them, into suitable subdirectories [**Jupyter Notebook 03**]
5. Generate a DEM that covers the study area covered by the flightlines [**Jupyter Notebook 04**]



Set up project folder structure

Run Jupyter Notebook 01 (or carry out equivalent steps manually). It will

- create a project folder with a (configurable) name like [date]-[descriptive label], similar to the prefix chosen during data acquisition.
- inside, create subdirectories as :
    - 00_aux [for auxiliary files]
    - 01_inputs [for input files]
    - 02_intermediate [for your preliminary outputs]
    - 03_products [for your final products]
    - x0_tmp [scratch space]
- copy RAD and NAV software into 00_aux/Software
- copy sensormodel files into 01_inputs/sensor
- copy boresight files into 01_inputs/boresight
- copy all raw project files (4 per flightline + gps + log) into 01_inputs/HySpex (can take QUITE long).

The reason we keep a copy of RAD and NAV with the data is to be sure we know which software version was used for processing. Also, NAV will write output directly into the directory where it is installed – so installing NAV into a system folder is not ideal.

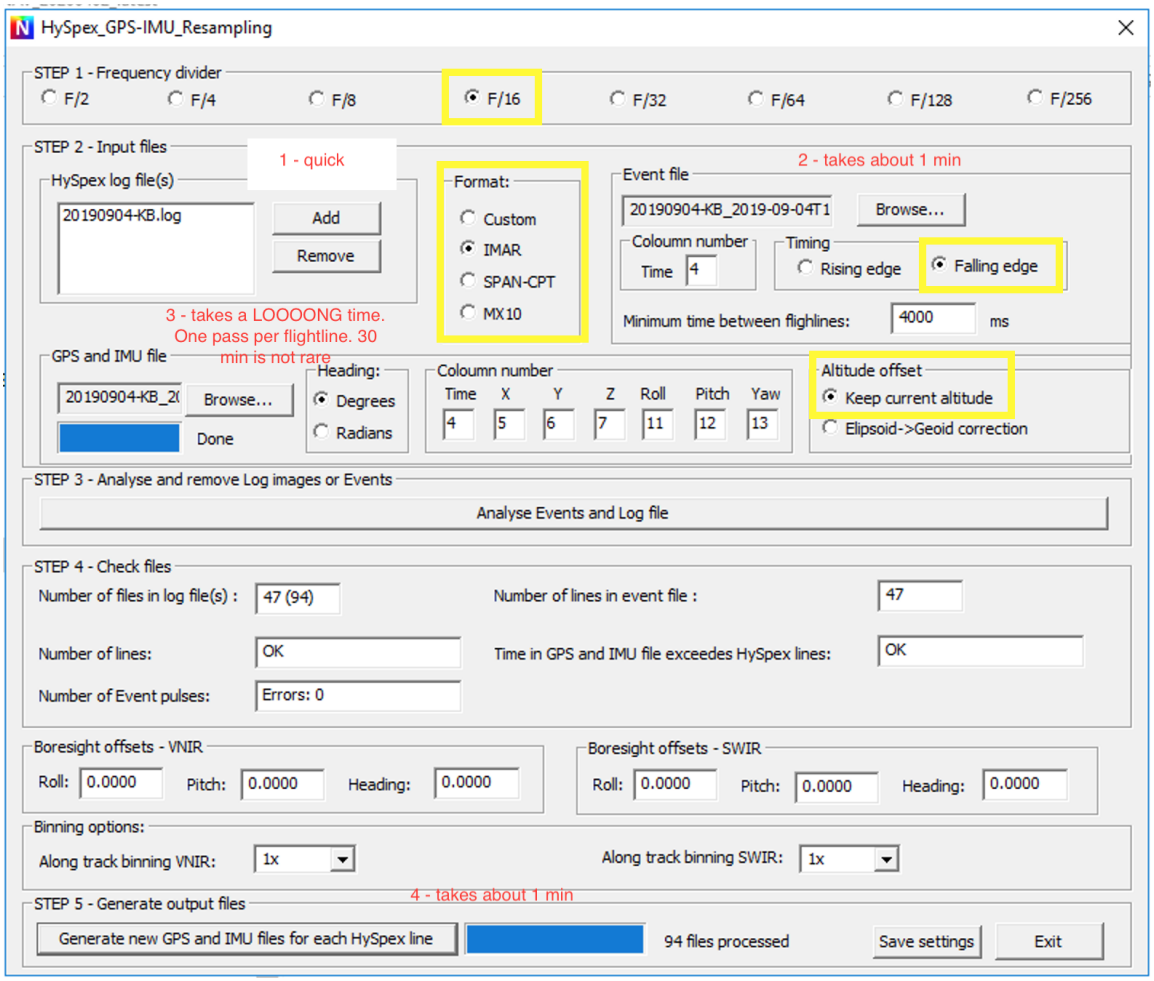
Preprocessing

HySpex NAV

HySpex NAV is a program to resample GPS and IMU data. Start it from 00_aux/Software in the project folder. It outputs two text files per flightline (VNIR and SWIR). It has a single screen GUI.

- **Frequency divider** – this is system specific. 1/16 for the UAF cameras.
- **Log file**: navigate to the file located in 01_inputs/HySpex in the project folder
- **IMU data format** depends on the IMU in use. IMAR in our case.
- The **event file** contains the data that specifies when data acquisition starts and ends (flightline beginning and end). In our setup, GPS, IMU and event data are all in the same GPS file. Navigate to the file located in 01_inputs/HySpex in the project folder
- Be sure to specify *Falling edge* and *Keep current altitude*

- Under **GPS and IMU file** navigate to the same gps file in 01_inputs/HySpex that you just used for the event file. Loading it can take very long as it is parsed immediately. It is not uncommon for this step to take 30 min, while the UI appears frozen. Make a cup of tea.
- As you add files, observe the indicators under STEP 4: There should be OK labels appearing, no errors, and the number of flightlines parsed from the files should correspond to the number of flightlines you have data files for.
- **Do not enter any boresight offsets.** This is a legacy menu point which we do not use. Boresight offsets will be added in PARGE during georeferencing.
- You can select binning to reduce the data volume (and spatial resolution). VNIR spatial resolution is much higher than SWIR, so depending on the flight height (and therefore the natural pixel size) you may want to use 2x binning for VNIR only. We usually leave it at no binning (1x).
- Finally, click *Generate new GPS and IMU files for each HySpex line*
- The output will be in the same directory as the NAV executable

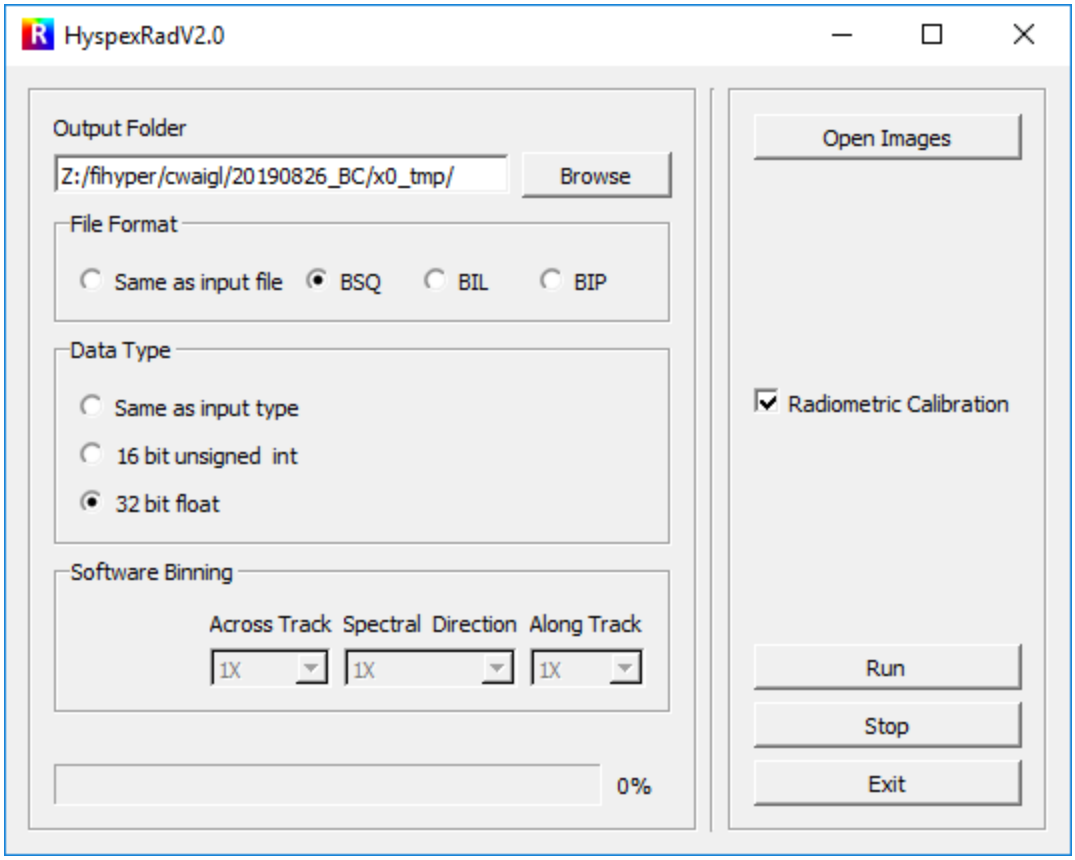


NOTE: In case of NAV erroring out, see Appendix 6 for a known bug, and how to work around it.

HySpex RAD

HySpex RAD is a program to process proprietary .hyspex raw data files to at-sensor radiance rasters. The output is in ENVI format. **We use ENVI band-sequential files with the .bsq extension throughout this project.**

- Navigate to the x0_tmp directory in the project folder for the *Output folder*
- Select *BSQ* and *32 bit float*
- Check *Radiometric Calibration*. If another checkbox is available (eg "Destriping") leave it unchecked
- *Open Images* and navigate to 01_inputs/HySpex in the project folder. Add .hyspex files
- Click *Run*



Generate a flightline Shapefile for GIS

This step is strictly speaking optional, but mapping the location of the flightlines is usually required at some point, eg. for subsetting the DEM. If you *only* want to process the data you can omit the step. But it takes only less than 5 min, so you might as well do it now.

Run Jupyter Notebook 02. The output will be in 00_aux/GIS by default.

Move per-flightline data into per-flightline folders

Run Jupyter Notebook 03. It will

- create a folder [date]-[label]_[flightline] for each flightline (eg. 20200830-BC_01 for the first flightline over the Bonanza Creek area on Aug 30, 202) inside the project directory
- create subfolders that neatly keep track of the files relevant for each processing stage:
    - NAV and RAD for the output of NAV and RAD
    - ELE for per-flightline DEM files
    - Parge_with_offset for the output of PARGE (using boresight offsets)
    - ATCOR for the output of atmospheric correction with ATCOR4
    - BREFCOR for the output of BREFCOR BRDF correction
- move the per-flightline RAD and NAV files into the respective folders. The other subdirectories will for the moment remain empty.

DEM preparation

For georectification and atmospheric correction, a high-quality Digital Elevation Model is required. How you go about generating depends on what is available for your target location. A good DEM has the following attributes:

- DEM should be in ENVI format (.bsq) with header (.hdr). (Theoretically, PARGE can accept GeoTIFFs, but our experience has not been good with the option.)
- DEM should be in UTM [appropriate zone - 6N for Fairbanks area) with a WGS84 datum
- DEM has the same spatial resolution as the target resolution for the data (eg. 1m, 50 cm...). If you generate it from downscaling a coarser DEM, ensure that you apply a low-pass filter to smoothen out artifacts. You may even have to downscale stepwise.
- **Origin coordinate offsets must not be negative.** Check the header file or metadata in ENVI. You have a few options for how to convert to a (0, 0) origin. The ENVI low-pass filter has this beneficial side effect.[1] You can also manually edit the ENVI header. Or use Python or GDAL to process files. If PARGE generates odd error messages, a DEM with a negative coordinate origin can be the cause.

For Alaska, the very nice 5m IFSAR DEM is available. It can be directly resample to 1m. We did not find artifacts, even without filtering. DEM tiles of various sizes are available in GeoTIFF and ArcGrid formats. Choose the DTM (digital terrain model) versions if possible, though DSM works, too.

[1] Why would they be negative? This can happen if you convert from a GeoTIFF file. For GeoTIFF, coordinates usually are specified for the pixel *corners,* while ENVI -- and PARGE internally -- always uses pixel *centers.* If you have a 1m DEM and convert from pixel corner to pixel centers, then your top-left corner might end up at coordinates (-0.5m, -0.5m) measured from the ENVI coordinate origin, the top-left pixel's center.

An alternative for the high latitudes is the 2m Arctic DEM. It is quite suitable. However, the noise level is higher than the IFSAR DEM. It is better described as a Digital Surface Model, and picks up tree stands.

Jupyter Notebook 04 is an automated interactive script that

- reads in the extents of a collection if IFSAR DEM tiles from a directory
- selects the tiles that overlap with the flightlines GIS file (output of Jupyter Notebook 02)
- visualizes the geometries involved (whether the available tiles cover the study area)
- mosaics, reprojects and crops the selected tiles
- saves the output to 01_inputs/DEM in the project folder

Even if it is likely that this Jupyter Notebook needs to be adapted for your specific needs, the effort may be worth it: reproducible, identically produced DEMs eliminate uncertainty in troubleshooting future processing steps.

Georectification (with PARGE)

How and why to run PARGE

PARGE is software for orthorectification and georeferencing of imaging spectroscopy (or other raster) data. It is aware of a variety of sensors and IMU/GPS data file formats and can read DEM data in various formats.

Useful documents are the *PARGE Manual*, *PARGE Quick Reference for Hyspex*, and various PARGE training summaries in written and video form that are available in the documentation repository.

Parge is run separately on VNIR and SWIR files. Data needed to run PARGE:

- Output of HySpex RAD (4 files per flightline - 2x data VNIR & SWIR, 2x ENVI HDR files)
- Output of HySpex NAV (2 TXT files per flightline - VNIR and SWIR)
- DEM mosaicked and cut to area of interest, in ENVI BSQ format with header
- sensor model files (VNIR and SWIR)
- boresight calibration files (VNIR and SWIR) or offsets (APPENDIX 2)

If you don't have boresight calibration files or offsets (to enter manually):

... you run Parge for boresight calibration first. See separate documentation and the Parge manual.

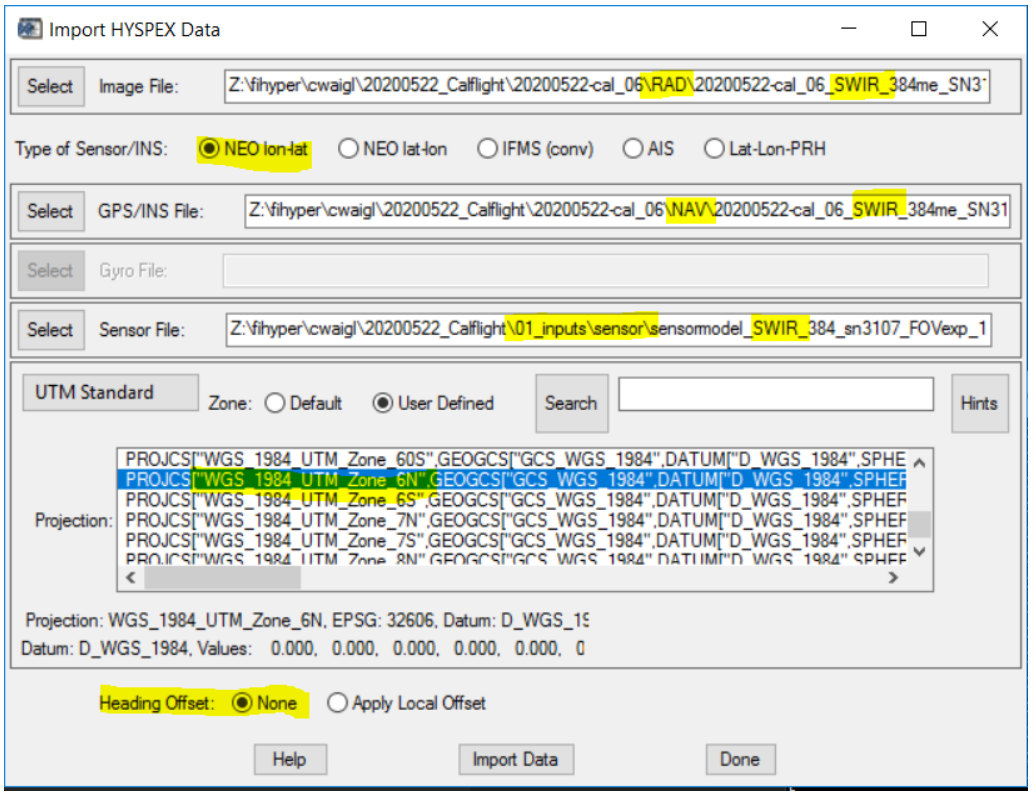
Otherwise, move on to Parge for georectification of aerial imagery

PARGE for georectification of aerial imagery - part 1

Start PARGE. For each flightline, and VNIR and SWIR separately:

1. *File > Reset session* ← PARGE keeps information from its previous run around. Aways start with resetting the session.

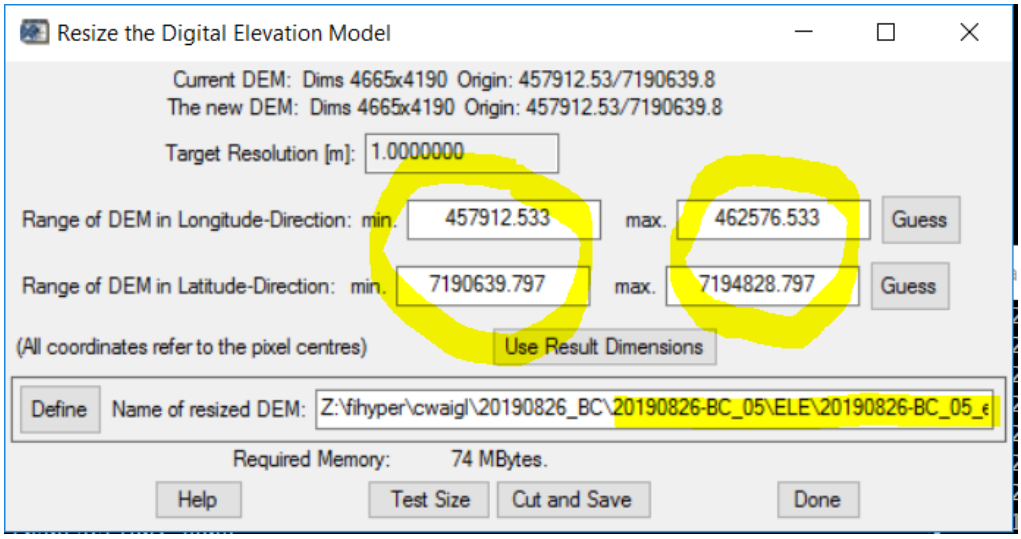
2. *File > Import > HYSPEX*. Select RAD and NAV files for flightline and dataset (VNIR or SWIR) as well as the appropriate sensor model file. Select *NEO lon-lat*, *WGS84/UTM 6N* and *Heading offset: None*. *IMPORT DATA.* Each major tool in PARGE ends with *DONE.*



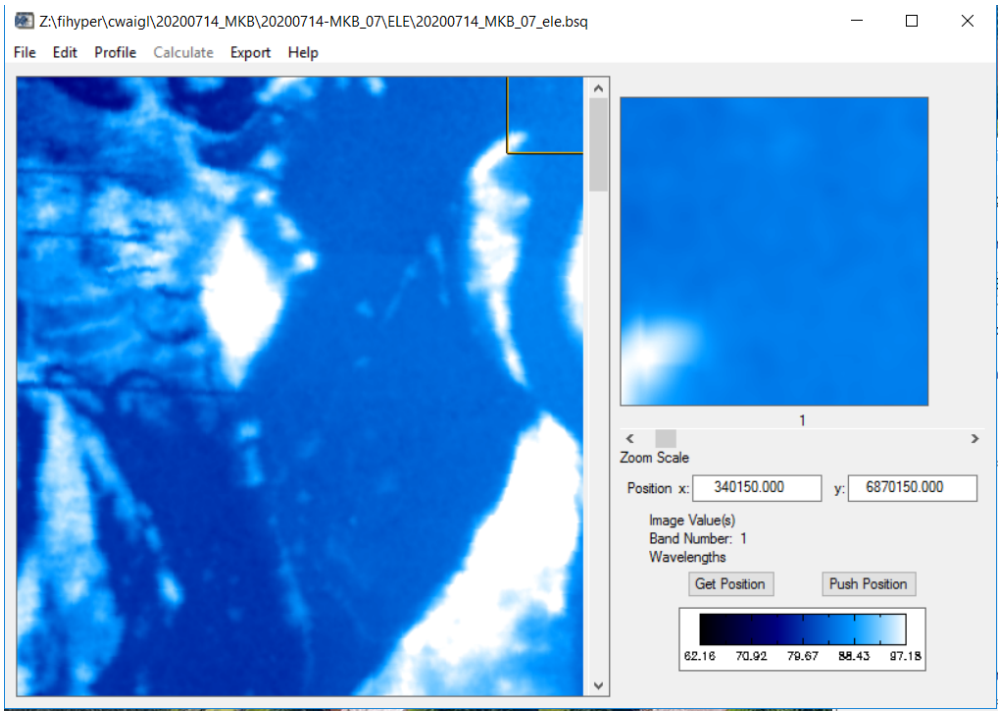
3. *DEM > Import > ENVI*. If this is the first file (VNIR or SWIR) of your current flightline, select and import DEM file that covers the whole dataset, from 01_inputs/DEM and then proceed to Step 4. **If this is the second file (ie. you have already processed the VNIR file and are now processing the SWIR file or vice versa), select and import the *_ele.bsq file from the ELE subdirectory of your flightline folder, then proceed to Step 5.**

4. *DEM > Resize DEM.* Check that the target resolution is your desired resolution (exactly - not 0.997532 m but 1.0000000 m). If it is not, the reprojection of your DEM to UTM may have changed the resolution, and you need to resample it before use.
Guess the range of the longitude and latitude directions, and then round the proposed values by at least 50 m to the nearest suitable 100 m (round the minima down and the

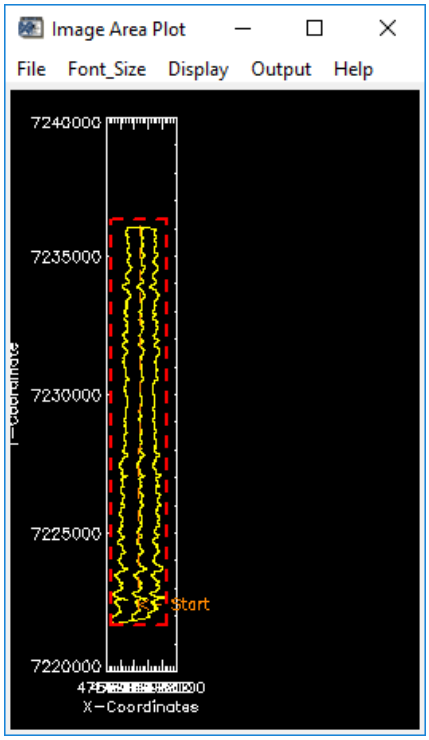
maxima up, increasing the area!). For example in the image below, rounded values would be 457800.000, 462700.000, 7190500.000, 7194900.000. Save the resized DEM to [ProjectDir]\[Date-Prefix_Flightline]**ELE**\[Date-Prefix_Flightline]_ele.bsq. In the example below, this is flightline 5 of the dataset 20190826-BC, so the file name is 20190826-BC_05_ele.bsq . Click on *Cut and Save*. Then *Done*.



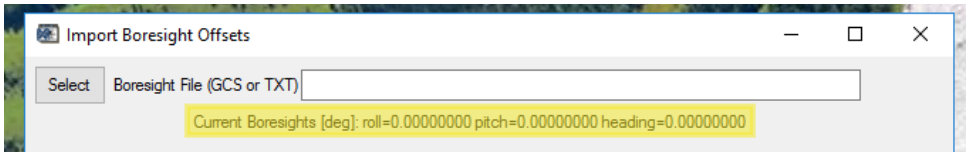
5. *DEM > Display DEM.* This is for quality control purposes. If PARGE has trouble with your DEM file it can output a resized DEM that consists of 0 values only. This would result in incorrect georectification. Check that you can see a topography that corresponds roughly to your expectations, with no artifacts.



6. *Auxiliary > Plot Image Area.* Another verification step. You get a representation of the center line and outer edges of the flightline, already projected into the mapped coordinate system (thus the wavy edge). A colored outline shows your flightline DEM outline (the *_ele.bsq file you just created). **Contrary to other documentation, the outline does not have to be green for the DEM to be ok**. In some versions of PARGE, a red outline indicates that the DEM does not completely contain the flightline, and the outline turns green when it is correct. But in the versions currently in use, the outline is always red (as far as we have found).



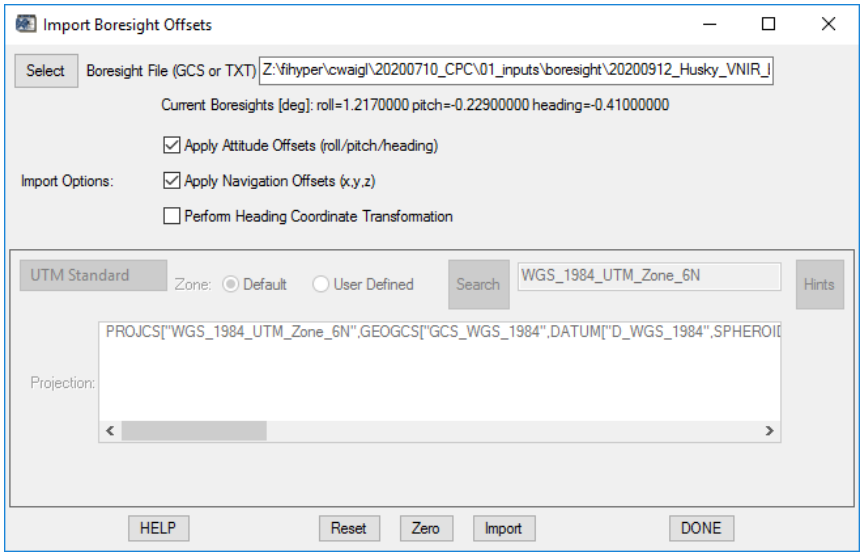
7. *Control > Import Offsets* . Before you load and import boresight offsets, check that all three current boresight offsets are zero[2]:



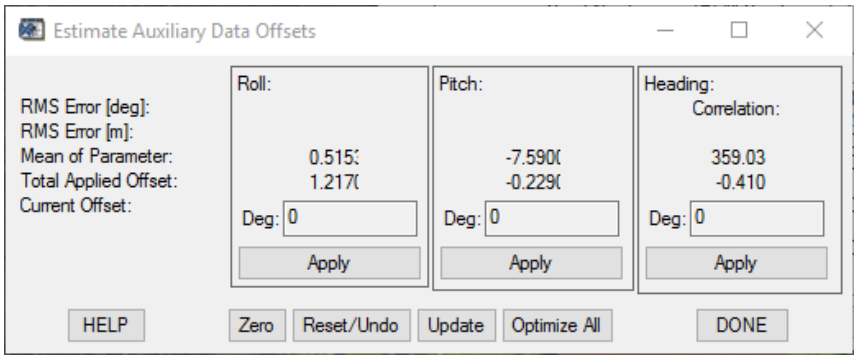
If you see non-zero offsets here for roll or pitch, you should investigate why there would be offsets applied. It is recommended to go back to Step 1. If you understand why there would be offsets, you can zero them out with the *Zero* button and apply new ones. Import boresight offsets from the file VNIR_*_boresight.gcs or SWIR_*_boresight.gcs (in 01_inputs/boresight) depending on whether you're processing a VNIR or a SWIR file.

[2] If you selected "*Apply local offsets*" when importing data, then the initial heading offset will not be zero. The reason is that there is a discrepancy between the north direction in UTM and WGS84. There is a theoretical argument in favor of using this option as well as "*Perform Heading Coordinate Transform*" in the current step, but in practice we have found worse georegistration when we tried this.

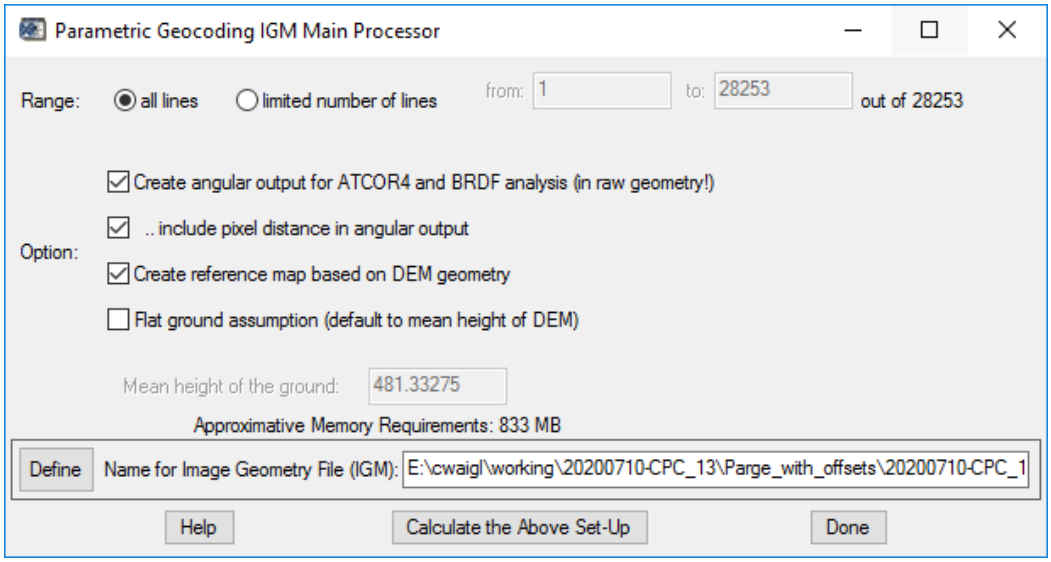
Recommendation is **not** to *Perform Heading Coordinate Transform.* Hit the *Import* button. Then *Done.*
(Alternative to this step is to add the boresight angles one by one in Step 8, by typing them in and hitting *Apply.* But it is safer to import from a file.)



8. *Control > Offset Attitude.* This is to check the boresight offsets before starting processing. Observe the *Total applied offset* values. They should agree with your boresighting tables:



9. *Processor > IGM Main Processor* . This is the first georectification step. Check the first three boxes. In the output file name, replace RAD with Parge_with_offsets (or whatever folder you want to safe the output in, inside the flightline subdirectory.) Note that you do not have to process the whole flightline: for a quick test, or if there is a problem with the flightline, you can just process the first half for example. (The word "line" refers to the rows of the flightline, across flight direction.) Click *Calculate the Above Set-Up.*



PARGE continued - option A - integrated processing (recommended)

NOTE: At this stage, you have two options: either to rectify BOTH VNIR and SWIR

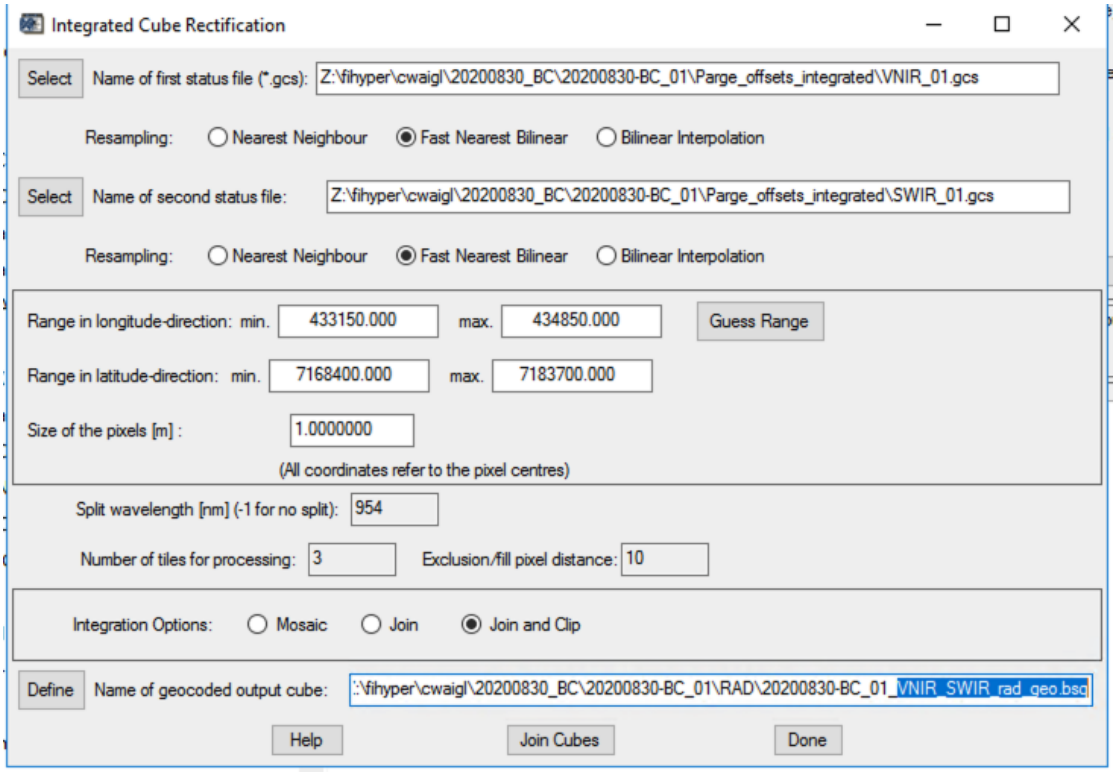
10. *File > Save Status* eg VNIR_06.gcs if you're working on a VNIR file. Then restart at step 1 to process the corresponding SWIR flightline. (Or carry out steps 1-9 for each of the VNIR and SWIR flightlines in the dataset).

11. *File > Reset session* followed by *File > Restore status* using your VNIR_xx.gcs file.

12. *Processor > Integrated Processing:*

    - Select VNIR_xx.gcs and SWIR_xx.gcs.
    - Resampling: fast nearest bilinear
    - Range and pixel size will be populated from DEM file (check if values make sense, don't edit)
    - Split waveliength: 954 nm
    - 3 tiles, pixel fill distance 10 (you can increase this)
    - Join & clip (masks bands to identical extent.
    - Output filename [prefix]_[lineno]_VNIR_SWIR_rad_geo.bsq

    This produces an integrated supercube with 459 bands.



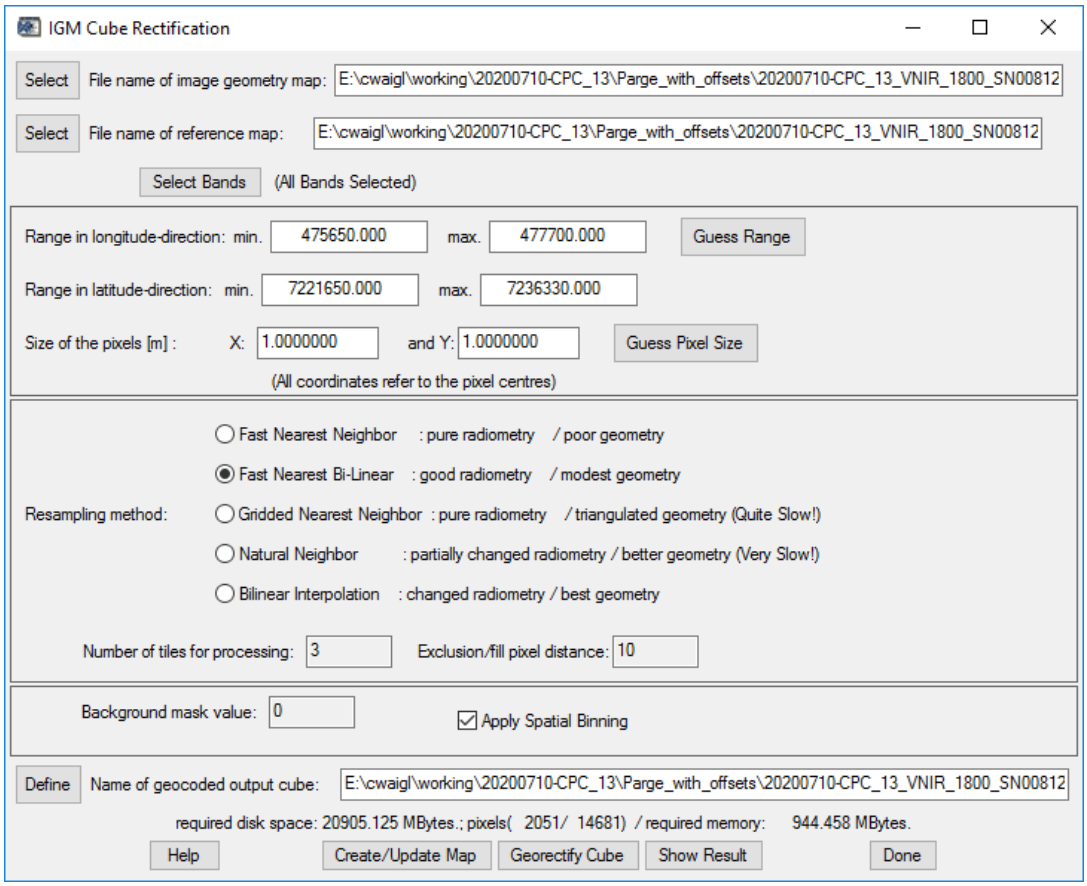
13. *File > Save status:* For example VNIR_SWIR_joint_xx.gcs

PARGE continued - option B - cube rectification separately for VNIR and SWIR (NOT recommended)

This is the old, initially developed method. It is now considered the legacy process. Since it requires more manual intervention and extensive post-processing, we do not recommend it.

10. *Processor > IGM Cube Rectification.* This is the second georectification step, in which the image map created in the previous step is used to warp the flightline raster.
    Do not click *Guess range* again - the values that appear should correspond to the extent of your per-flightline DEM (*_ele.bsq file). This is correct, and you want to keep the values as they are. If you wish to only generate an RGB image, for example, you can select a spectral subset. This makes processing much faster.
    Resampling Method = **Fast Nearest Bi-Linear**

    Number of Tiles = **3.** *Exclude/Fill pixel distance* = **10** (makes it less likely that no-data spots appear if the plane had to bank rapidly and the data is stretched out spatially).
    Make sure the output goes to the Parge_with_offsets subdirectory (or your choice of subdirectory).



11. Before closing the window (*Done)* you can visually check the output with *Show Result.*

12. *File > Save Status* eg VNIR_06.gcs or SWIR_[flightline].gcs, or VNIR_SWIR_[flightline].gcs for integrated processing. It is **very important** to save the processing status in the end: The entire state can be recreated in PARGE by restoring a status file (.gcs). This allows troubleshooting and helps document settings that were chosen.

Quality assurance

One key point to check is that the SWIR and VNIR rasters for each flightline co-register.

An easy way is to open the georectified radiance rasters *_geo.bsq for both the SWIR and VNIR data of the same flightline via *File > Display ENVI File* . The *Get Position* and *Push Position* buttons can be used to move the zoomed-in area and cursor to the same coordinate position. Do they end up in the same location, for example a road intersection or other landmark?

A quantitative method is to open both files in ENVI, and then use the *Map > Registration > Automatic Registration: Image to Image* on bands 171 VNIR and 2 SWIR, both at 954.66 nm. After eliminating the worst automatically generated tie points, RMS co-registration error should be able to be reduced to approximately 0.5 m. The goal is not to warp the images to each other, but it is good practice to save the tie points (GCPs) that ENVI finds.

The most common reason for severely off co-registration is failure to apply the correct boresight angles to the VNIR or the SWIR flightline (or both!). This is where the .gcs files come in very usefully, as it is easy to verify after restoring the status, with *Control > Offset Attitude*.

SWIR and VNIR co-registration is referred to as "relative boresight correction". We can achieve sub-pixel precision here. However absolute boresight correction – that is, co-registration with other georeferenced datasets or external GPS ground control points – requires real-time corrected GPS. In practice, we find errors of 1-3 m in absolute geolocation (not unlike in AVIRIS or WorldView ortho-imagery) without RTK GPS in-flight.

Atmospheric correction (with ATCOR4)

Atmospheric correction with ATCOR4 requires the HySpex sensor model to be installed with the HySpex software in the software install location. This only needs to happen once, but it is good practice to store a copy of the sensor model for ATCOR in the project folder. The sensor model for ATCOR can be found in the GitHub repository.

Preparation for atmospheric correction

Solar parameters, flight geometry & timestamps

Jupyter Notebook 07 is used to generate a navigation metadata file (in JSON format) for each flightline and write it into the NAV directory. The data is based on a) the original RAD metadata HDR file and b) solar angle calculations. The file name is of the format `[prefix]_[lineno]_VNIR_SWIR_rad_geo_flightdata.txt`. Typical control output parameters:

```
Flightline heading: 177.77 °
Flightline elevation: 1195.19 m
Flightline roll: -0.84 °
Flightline roll std: 2.98 °
Flightline pitch: -2.20 °
Flightline pitch std: 1.21 °
Flightline latitude: 65.58 °
Flightline longitude: -145.06 °
NEO start timestamp (UTC): 2021-06-23 20:06:35
Flightline timestamp (UTC): 2021-06-23 20:15:37+00:00
Flightline timestamp (AKDT): 2021-06-23 12:15:37
Solar azimuth: 150.96 °
Solar zenith: 44.41 °
```

Check whether the parameters make sense (eg. the UTC timestamps and location look realistic, and the sun is where it should be - in the SW for late afternoon, say). You can look up the solar

elevation angles for your location as well. (Note: You could also use, or check with, https://www.esrl.noaa.gov/gmd/grad/solcalc/)

Water vapor

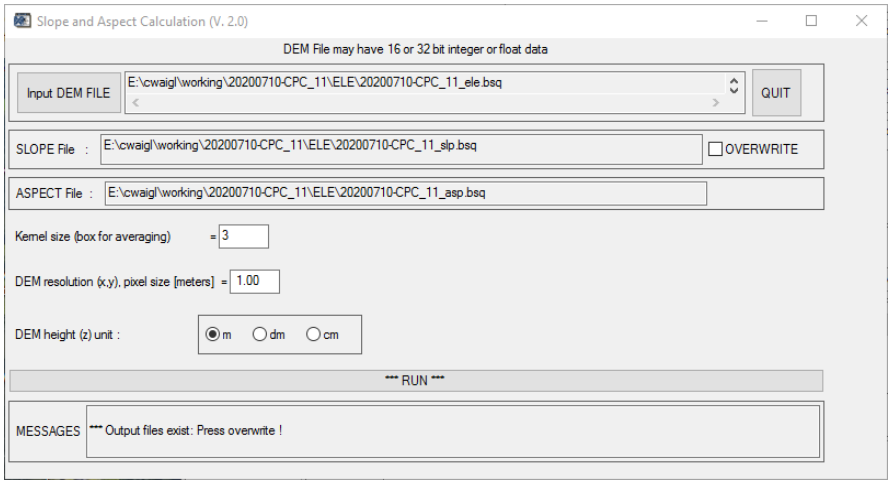
Atmospheric correction requires knowledge of the amount of water vapor in the air column on the day and in the location the data was acquired. There are many options. (Ask your preferred meteorologist for the most appropriate data source.) For the US, the soundings from U Wyoming http://weather.uwyo.edu/upperair/sounding.html are a good start.

Find the data for the day and the nearest location. At the bottom of a sounding, it says "Precipitable water [mm] for entire sounding". Write down that value.

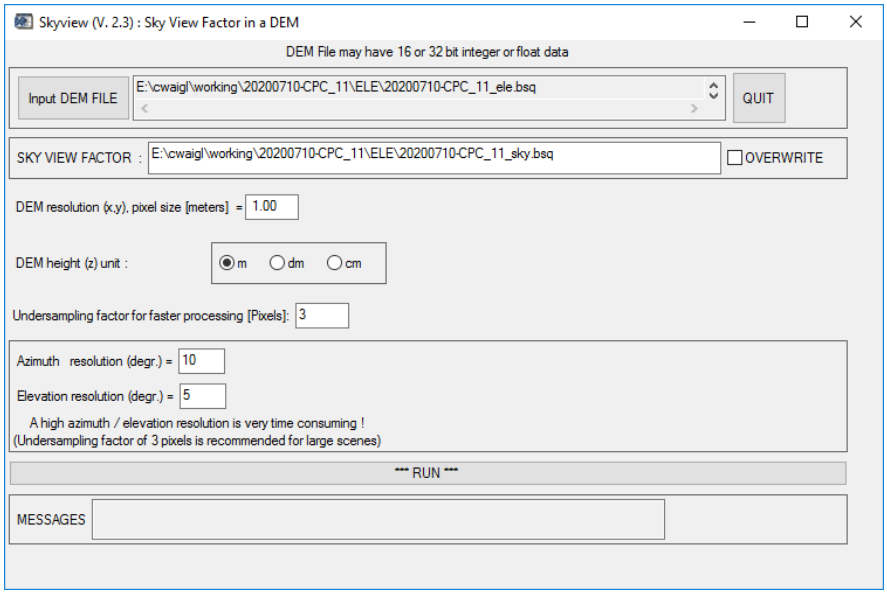
DEM preparation

If you want to use the "rugged terrain" model (default assumption!) ATCOR4 needs slope and aspect maps derived from your flightline DEM as well as (optionally but recommended) skyview factor. Both ATCOR4 and PARGE have tools for that. The ones in ATCOR4 do the job well.

1. In ATCOR4: *Topographic > Slope/Aspect.* Check output paths, resolution and unit. A small averaging kernel is advised.



2. *Topographic > Skyview factor*



*Undersampling factor = **3*** (otherwise processing time is very long)
*Azimuth resolution = **10*** (we want fairly high resolution) or 20 (faster)
*Elevation resolution = **5*** or 10

(NOTE: If you use PARGE, use the stand-alone menu entries *Slope/Aspect* and *Skyview factor.* They process your DEM in its final projection. Do not use *DEM preparation for ATCOR* as it resamples your DEM to raw HySpex geometry - that's not what is wanted here.)

3. *Topographic > Cast Shadow Mask*

   Use parameters from _flightdata.txt file

Supercube generation ***only if integrated processing was not used in PARGE***

The tediousness of this manual step is the main reason integrated processing to a 459 band supercube is recommended. If you produced separate supercubes for VNIR and SWIR, this step is required.

We will run atmospheric correction on a full VNIR/SWIR supercube (for each flightline). To this end we also need to generate a single VNIR/SWIR scan angle file. While the radiance supercube is generated by layer stacking, the scan angle file is achieved by averaging the respective layers (4) of the VNIR and SWIR *_geo_sca.bsq files.

1. Load the following four files into ENVI:
   [date]-[label]_[flightline]*_VNIR_*_geo.bsq
   [date]-[label]_[flightline]*_VNIR_*_geo_sca.bsq
   [date]-[label]_[flightline]*_SWIR_*_geo.bsq
   [date]-[label]_[flightline]*_SWIR_*_geo_sca.bsq

2. Using the *Layer Stack* tool in ENVI on bands 1-170 of VNIR and 2-288 of SWIR radiance files *_geo.bsq . Select *Inclusive* output file range.
   Save as [date]-[prefix]_[flightline]_VNIR_SWIR_supercube_geo.bsq in ATCOR directory belonging to flightline (eg. 20200814-SC_01_VNIR_SWIR_supercube_geo.bsq) [~40 min - regenerating pyramid layers *_geo.bsq.emp takes longer than the stacking itself]

3. (NOTE: **Jupyter Notebook 07a** can facilitate this process). Edit header file [date]-[prefix]_[flightline]_VNIR_SWIR_supercube_geo.hdr (in a text editor, or write a script). Copy over from the *VNIR*_geo.hdr or generate header keys as follows:

   - description {Frameperiod = ... }
   - sensor type = HYSPEX
   - x start = 1
   - y start = 1
   - default bands = {75.000000, 46.000000, 19.000000}
   - pixel size = {1.0000000, 1.0000000}
   - data ignore value = **0**
   - acquisition time = *[update this with the corrected UTC time from the flightdata.txt file generated by the Python script earlier; for example, if the original acquisition time from the *geo.hdr file is 2019-09-04T13:34:52.0Z, it has actually been collected in AKDT, while the *_fligthdata.txt file lists the corrected GPS acquisition time (UTC) . Make sure it is formatted to match the hdr format as, eg, 2019-09-04T21:39:11.0Z]*
   - GPS long-lat-alt = {<*value*>, <*value*>, <*value*>}
   - heading [deg] = <*value*>
   - DEM height [m] = <*value*>

4. Check that the _VNIR_SWIR_supercube_geo.bsq file opens correctly in ENVI.

5. Now to the scan angle files *_geo_sca.bsq. Each of them has four bands with flight geometry data. Unlike the radiance rasters we are not stacking them, but averaging them. Use the *Band Math* tool to generate a new expression: (b1 + b2)/2. Add it to the list. Then highlight the new expression and click *Map Variable to Input File.* Assign *VNIR*_geo_sca.bsq to b1 and *SWIR*_geo_sca.bsq to b2.
   Output the result to [date]-[prefix]_[flightline]_VNIR_SWIR_supercube_geo_sca.bsq in ATCOR directory belonging to the flightline.
6. Edit the *..._supercube_geo_sca.hdr header file (in a text editor, or write a script), copying from the *_VNIR_*_geo_sca.hdr:
   - Description
   - Sensor type (eg “sensor type = ENVI Standard”)
   - X start and y start (i.e. x start = 1 y start = 1)
   - Copy over everything from “default bands = …” to the end of the VNIR geo_sca.hdr file and replace “band names = …” to the end of file in the supercube .hdr.

7. Check that the supercube _geo_sca.bsq file opens in ENVI

At this stage it is possible to apply a mask to the radiance supercube and combined scan angle files. However, masking can also be left until after atmospheric correction. See the subsection in the Postprocessing section.

PARGE 3.5 supposedly includes a tool to stack the VNIR and SWIR data right in Parge. At the time of writing this isn't yet working for us.

ATCOR for rugged terrain (ATCOR4r)

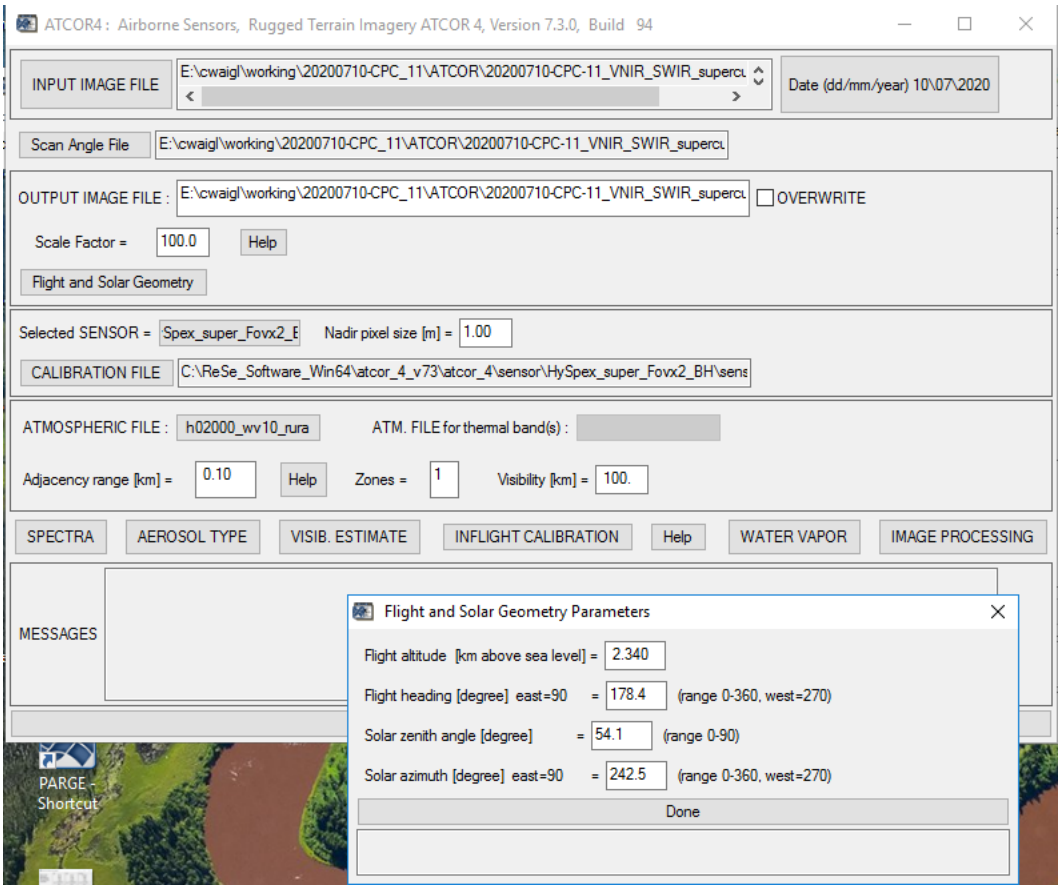
We use the ATCOR for rugged terrain tool in ATCOR4.

1. First, as always, *File > Reset Session.* ATCOR is pretty good about guessing which file goes with the project, but if the session is not reset, spurious parameters can be cached and interfere with the right settings.
2. Then ATCOR > ATCORr
    - The *Input Image* is the *_VNIR_SWIR_supercube_geo.bsq file in the ATCOR directory.
    - *Scan Angle File* (*_VNIR_SWIR_supercube_geo_sca.bsq) and *Output Image File (*_VNIR_SWIR_supercube_geo_atm.bsq) should automatically populate. If not, enter manually.
    - *Scale Factor* = **100.0**
    - *Selected sensor:* If the sensor model is installed with the ATCOR4 software, the HySpex sensor is available here.
    - *Flight and solar geometry* opens a new sub-window. Check the values against the _flightdata.txt file. If they are different by more than a few degrees, they need to be edited. Note that zenith angles can be measured either clockwise from North, or counterclockwise from South.

    - *Nadir pixel size* should be the spatial resolution the data is in.
    - *Atmospheric file* – this requires the water vapor column value previously determined, as well as the average flight height from the *_flightdata.txt file. h02000_wv10_rural for example is the choice of a rural atmosphere model, 10 mm total WV column and a flight height of 2000 m. Select the closest match from dropdown under the button.
    - Adjacency range[km] = **0.10** (this is the influence of neighboring pixels/reflection)
    - Zones = **1**

    - Click on *WATER VAPOR*
        i. Select 1130 nm region, click done - ATCOR4 will calculate water vapor look-up tables (LUTs)
        ii. In dialogue that appears, click “Calculate Water Vapor Map”
        iii. To see preview before calculating water vapor, click “SPECTRA” button

- Visibility[km] -> click VISIB. *ESTIMATE* and transfer value. **As we fly at clear sky, usually 100km is correct.**



3. Once all parameters are set, click on *IMAGE PROCESSING*.
   - Variable visibility = **no** (OR yes, depending on conditions)
   - Variable Water Vapor = **no**
   - Shadow Removal = **no**
   - Value Added Products = **no** (OR yes, depending on desired output)
   - Solar flux @ ground = (depends on user preference)
   - IRC method = **no**
   - Click ‘OK’

   If you selected Value Added Products, in the next dialogue box
   - Select radio button “**2: LAI = … NDVI…**”
   - Click ‘DONE’’

   In the next dialogue box
   - Select region 1130 nm and "Water vapor retrieval with band regression"

   In the next dialogue boxes (topographic BRDF correction). This is necessary to do here, as the BREFCOR module in postprocessing is for observer BRDF correction.
   - recommended: "General purpose method" and "LA + SE (standard)" . According to ATCOR4 documentation and companion report, this works very well, and this

is our experience as well. But an atmospheric scientist may wish to choose a different method.

Once all parameters are set click *Run Atmospheric Correction*

4. Eight outputs:

Atmospherically & geo-corrected image	*_geo_atm.bsq
Settings of the atmospheric correction	*_geo.inn
Log file	*_atm.log
Atmospheric optical thickness	*_atm_aot.bsq
Visibility index	*_geo_atm_visindex.bsq
Illumination file	*_geo_ilu.bsq
Quality indicator (haze, cloud, water)	*_geo_out_hcw.bsq
Water vapor map	*_geo_atm_wv.bsq

Alternative: ATCORf for flat terrain

If you can assume your DEM to be flat (eg, you're only interested in areas over ocean), the *ATCOR > ATCOR4f* may be an option. The steps are the same as in ATCOR4r except that the DEM files are not needed. Alternatively, teams have generated a flat DEM programmatically. (NOTE: **Jupyter Notebook 04a** is supposed to be completedy by CM team.)

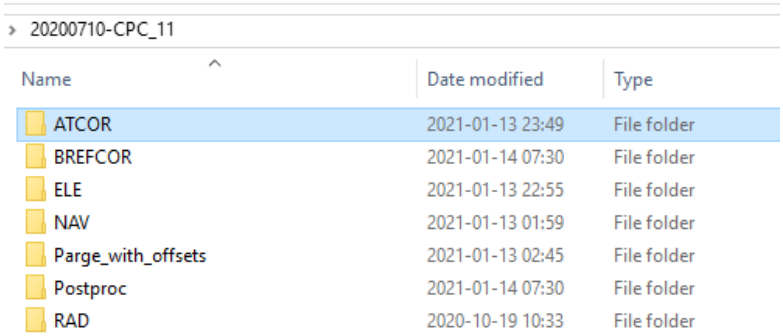
Always start with *File > Reset Session!*

Postprocessing

After the previous steps, the output is an atmospherically corrected reflectance scene. Further processing falls under postprocessessing, and whether or not each step should take place is dictated by the nature and purpose of the imagery.

Except in very flat terrain, observer BRDF correction is likely to be necessary for best quality reflectance data, and standardising metadata in the end is a matter of good housekeeping to assure data are reproducible and can be reused. The other options (spectral polishing and spatial binning, masking, mosaicking) however will not apply for everyone.

Best practice is to create one or several new subdirectories in the flightline file to hold the outputs from these steps. BREFCOR generates multiple output files, so this folder structure may work for you:

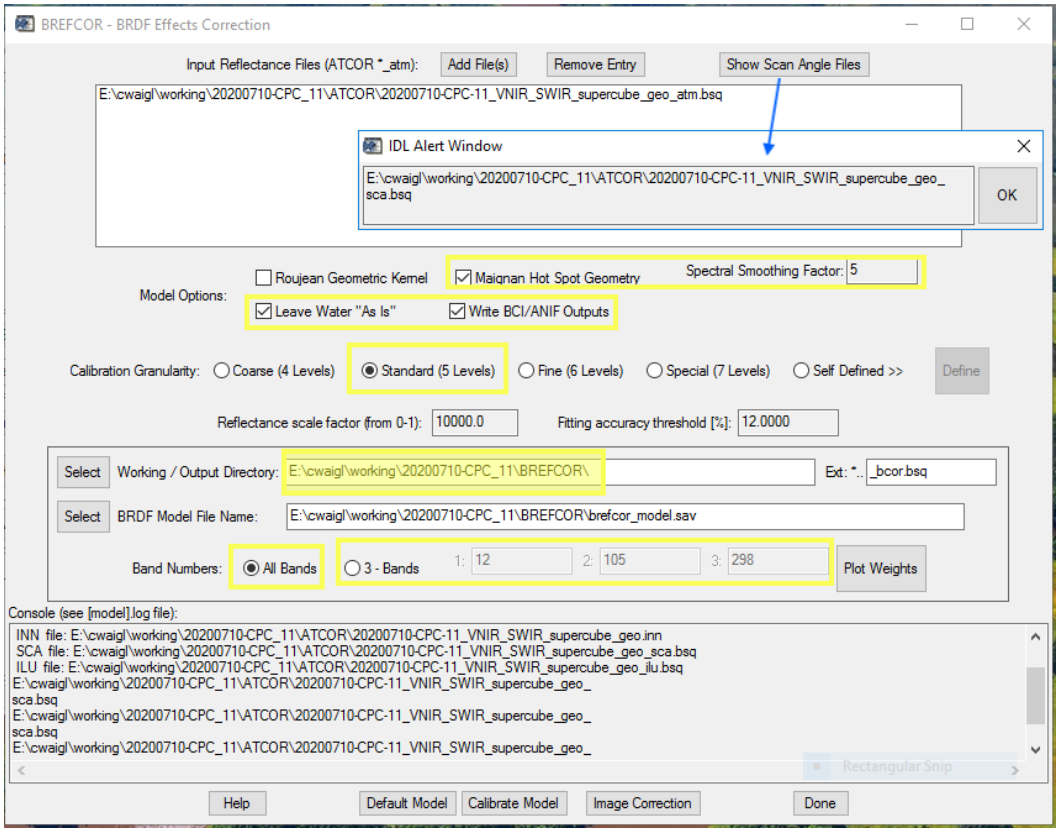


BRDF (observer) correction with BREFCOR

This eliminates the effects of differential illumination and observer angle. Use ATCOR4 *BRDF > BREFCOR Effects Correction*.

The BREFCOR module is able to correct multiple flightlines at once, which is the ideal case. All files have to be in the same directory, all files (the atmospherically corrected reflectance ..._geo_atm.bsq, the corresponding scan angle files ..._geo_sca.bsq and the other ATCOR outputs ...inn.bsq and ...ilu.bsq). The convention is to work in the 02_intermediate/BREFCOR directory. If you want to process multiple flightlines, change the folder structure and move the BREFCOR directory one level higher, above the individual flightline folder. I'll presume that you are running it one flightline at a time.

- Click *Add File(s)* and select *_geo_atm.bsq
- Click *Show Scan Angle Files* and check the correct *_geo_sca.bsq file is picked up.
- Select *Maiganan Hot Spot Geometry, Spectral Smoothing Factor = **5**, Leave Water "As Is", Write BCI/ANIF Outputs*
- Output directory should be your BREFCOR subdirectory
- BRDF Model File Name – the default is brefcor_model.sav. This is fine.



BREFCOR proceeds in two steps: Building (calibrating) the model and correcting the image. To get the first step right, you can run a pre-test by selecting *3 Bands* initially. These should cover VIS, VNIR and SWIR (eg. 12, 105, 298 ← don't select bands for which your particular sensor may be exceptionally noisy).

- Click *Default Model*
- *Calibration Granularity: Standard (5 Levels)*
- Click *Calibrate Model*
- In *Airborne ATCOR > BRDF>BRDF Model Analysis*
- Select the BRDV model (*.sav)
- In *Airborne ATCOR > BRDF > BRDF Model Plot*
- Check if output makes sense. (more than 50% change in output reflectance, likely that something is not right) 0-50% change is ok. Check for each band and each cover class
- If it doesn't make sense, re-run & change calibration granularity to coarser or self-defined

Once you are happy with the 3-band test, calibrate model with *All Bands* selected. After model is calibrated, click *Do Image Correction*

Output: *_geo_atm_bcor.bsq

Spectral binning and polishing

The goal is to reduce noise. Noise can be assessed in PARGE (*QC > SNR Analysis*) or more impressionistically by using the *Z-profile* function in ENVI to sample spectra and inspect them for noise.

Least invasive approach: Spectral binning. Use the *Spectral Binning* tool in ENVI. Two bands per bin (that is, spectral resolution is reduced by 2). Save file as ..._binned.bsq.

More invasive approach – this changes the data: Spectral polishing. ATCOR4 offers several tools for spectral polishing under *Filter > Spectral Polishing*. Please refer to the ATCOR4 manual to make the right choice – using spatial statistics to reduce random noise or spectral filtering to reduce systematic noise. Another option is the *Savitzky-Golar FIlter* tool in ENVI or using scipy.

For the desired scientific application, spectral resampling is often a next step (the preprocessing step for scientific analysis). For example, to enable comparisons, HySpex data may be spectrally resampled to AVIRIS, using both sensors spectral response functions (ENVI has a one-step tool for that). In this case, spectral post-processing of the HySpex data may not be necessary at all.

Masking

If a mask wasn't applied before atmospheric correction, this step is useful to carry out now. We recommend storing masks under 02_intermediary/raster_masks and .../vector_masks

Option 1: Using ENVI

ENVI is a suitable tool, using *Build Mask* followed by *Apply Mask*.

Inputs are the geometrically corrected radiance files (output of PARGE), as well as the corresponding scan angle files. The mask raster itself should be saved with the flightline as [flightline]_VNIR_SWIR_Mask.bsq.

Mask conditions are:

- radiance in VNIR > 0 (use any band)
- radiance in SWIR > 0
- scan angle < 9100 (this is the "nodata" value used by PARGE)

Mask value should be 0, or whatever other nodata value you would like to standardise on.

Masking also often eliminates any shadowed areas along the flightline edge.

Option 2: Using Python and command line gdal

The code in **Jupyter Notebook 06 & 06a** serves as an example of how to define masks in Python. The implementation details depend on what bands should go into the masking, whether masks need to be cropped to deal with sensor shadow or reflection at the swath edge. The workflow is as follows:

1. Use JN 06 to create raster masks
2. Use gdal to convert rasters to vectors
3. Use JN 06 to crop vector masks at the edges or by intersecting with your own ROI polygons
4. Use gdal to apply masks to data.

The notes in Appendix 5, "Masking" are likely useful here.

Mosaicking

ENVI is a suitable tool for mosaicking multiple flightlines into a merged image. Absolute georegistration error grows at the edge of the flightline, so for a perfect result, flightlines may have to be manually co-registered to each other first. (This is prevalent when no real-time corrected GPS is available: boresight correction will be able to correct for the relative offset of SWIR and VNIR to each other, but absolute errors will grow noticeable at the flightline edge.)

As an alternative to ENVI, automated mosaicking with GDAL (gdal_merge) gives quite satisfactory results for applications where exact edge matching is not required.

If Python is used, good results have been obtained using the rasterio library. However, as the file sizes are very large

Mosaicked supercube files can become very large in size. So if mosaicks are only needed for visualization purposes, RGB bands should be selected and extracted *first*.

Metadata creation

Metadata is stored in the .hdr files (if ENVI is the final archival file format.) What metadata to provide is a decision to be made based on the team's and project's needs. If a different file format is chosen (for example, an open format like NetCDF or HDR5, or a stack of GeoTIFF files), researchers should use the appropriate metadata infrastructure.

Suggested metadata entries are:

- Descriptive band names (either VNIR Band 1 ... VNIR Band 170, SWIR Band 1 ... or re-number Band 1 ... Band [last]). Generate numbering in a suitable tool (Excel, Python, text editor that allows for consecutive number generation)
- Default bands to load (eg. 15, 75, 198)
- Corresponding central wavelength for each band, and units

- Bad band list
- FWHM
- Gain and reflectance scale factor
- Acquisition time (take values from ... flightdata.txt file)
- Pixel Size (Check to make sure it is the same as the previous files)
- Date of boresight correction flight, or boresight values
- lat / lon of scene center
- Name of person who processed the flightline
- ID for project
- ID for location, if applicable
- Software versions used (RAD, NAV, PARGE, ATCOR4)
- Method of BRDF correction
- Give thought to applicable metadata keywords (for indexability), for example from https://cfconventions.org/

Even if you haven't used atmospheric correction, the section ***Preparation for atmospheric correction*** above is highly recommended. Preparation for atmospheric correction **Jupyter Notebook 07** generates per-flightline files that contain corrected timestamps, solar and sensor angles etc. **Jupyter Notebook 07** rewrites metadata header files for each flightline and can easily be modified to rewrite HDR files at any stage of the processing chain, if desired.

APPENDIX 1: Python modules

If you wish to install your own Python environment, the following modules and their dependencies should cover everything you need. See instructions at https://github.alaska.edu/cwaigl/FireIce_BF_Hyspex/tree/main/Conda_environment .

- python=3.7
- geopandas
- shapely
- fiona
- rasterio
- descartes
- matplotlib
- jupyter
- notebook
- future
- pip
- asciitable (deprecated, used in old scripts)
- gdal (deprecated, used in some data extraction scripts)

APPENDIX 2: Boresight offsets

[Refer to separate boresight calibration documentation on how to use PARGE for boresight calibration.]

The Boresight values used for the 2019 and 2020 data from the Aviat Husky platform are from the following tables. 2021 and later: New boresight calibration in Cessna 185 was carried out Aug. 2, 2021. **As of this writing, this was the last calibration - any data acquired after the 2021 flight season will require a new calibration.**

May 2020 flight (for data before June 2020):

	Roll		**Pitch**		**Heading**
VNIR	0.489	**1.237** ± 0.045	0.491	**-0.284** ± 0.032	**0.051** ± 0.224
SWIR	0.346	**-0.307** ± 0.026	0.415	**0.191** ± 0.036	**-0.160** ± 0.104

June 2020 flight

(Lesser quality than September 2020 flight, but already processed data was not necessarily reprocessed)

	Roll		**Pitch**		**Heading**
VNIR	0.629	**1.230** ± 0.012	0.644	**-0.341** ± 0.033	**-0.738** ± 0.297
SWIR	0.755	**-0.268** ± 0.009	0.640	**0.118** ± 0.024	**-0.989** ± 0.128

September 2020 flight (preferred for 2020 data June and later)

	Roll		**Pitch**		**Heading**
VNIR	0.489	**1.224** ± 0.032	0.499	**-0.305** ± 0.019	**-0.421** ± 0.102
SWIR	0.513	**-0.308** ± 0.025	0.487	**0.137** ± 0.025	**-0.670** ± 0.060

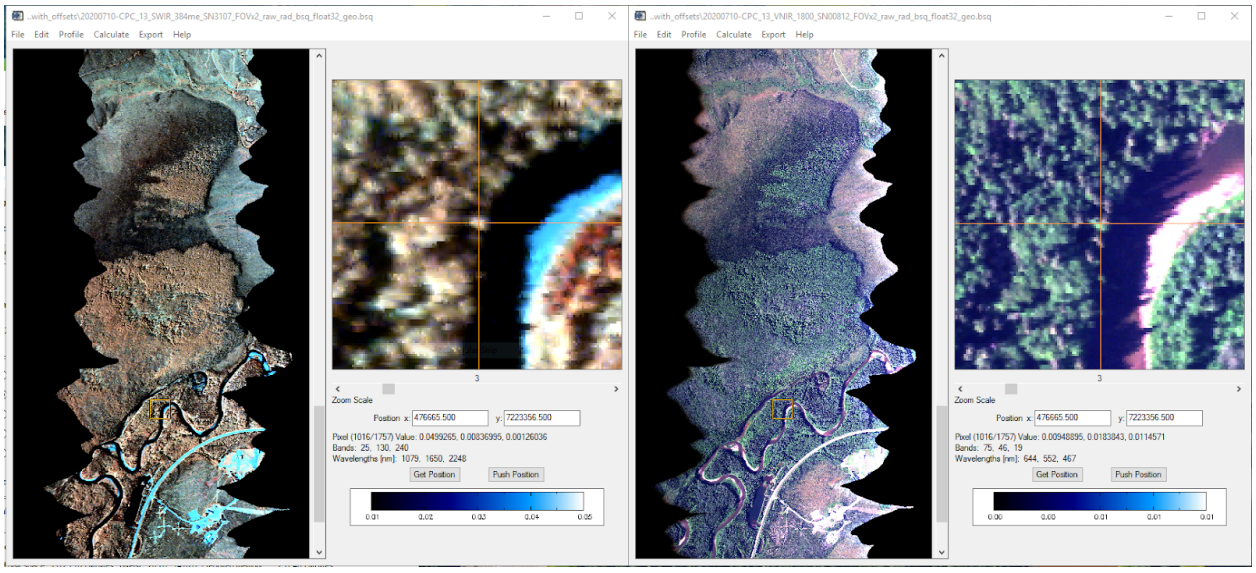
August 2021 flight (for 2021 flying season)

	Roll		**Pitch**		**Heading**
VNIR	0.486	**1.155** ± 0.024	0.496	**-0.275** ± 0.013	**-1.138** ± 0.121
SWIR	0.674	**-0.303** ± 0.026	0.656	**0.165** ± 0.013	**-1.325** ± 0.148

APPENDIX 3: Quick co-registration quality check in Parge

After processing the second file (eg. SWIR) of a flighgtline in PARGE, it is useful to check whether it co-registers with the first (eg. VNIR). A very quick way is to do this once "IGM Cube Rectification" has completed.

- Click on *Show Result.* An image display with zoom window will open.
- Use *File > Display ENVI File* from within this first image display to open the other imagery file of the same flightline. A second image display with zoom window will open.
- Place the cursor on a salient feature in the first window. You can increase the zoom level as desired.
- The click *Push Position* in the first window followed by *Get Position* in the second window. The cursor will jump to the exact same location (see also the coordinates in the mapped coordinate reference system).
- Compare.



APPENDIX 4: Generate a plain text and GIS vector representation of the *entire* flight path

The process described in the Preprocessing section will generate, via HySpex NAV, ASCII (plain text) files for each flightline, as in, a line in which HySpex imagery data is acquired. These files are then transformed into a single linestring Shapefile with Jupyter Notebook 02.

Sometimes however you might want to extract the entire IMU dataset (timestamp, roll, pich, heading, flight elevation, GPS lon & lat) in plain text form for future use. For example, you might want to match the timestamps to external events, such as imagery or measurements acquired with other sensors.

The GPS files that come out of HySpex Air are in a proprietary format that potentially can be read and made available with software that the IMU vendor provides. But HySpex NAV can also be used. For this purpose, you will need to change a setting in the `config.ini` file of HySpex NAV. (Such a file is available, with the software, in our GitHub repository here: https://github.alaska.edu/cwaigl/FireIce_BF_Hyspex/tree/master/NEO_software/HySpex_NAV_20200402_latest) . The setting to change is:

`GPS_to_ASCII = 1` (instead of =0)

This setting will cause HySpex NAV to run somewhat more slowly. But it will result in the output of an additional text file, which contains the entire flight path (see picture on the right), and not only single flightlines (see picture on the left).

APPENDIX 5: GDAL Cheat Sheet

GDAL (the Geospatial Data Abstraction Library) is a Swiss Army Knife for querying, subsetting and combining geospatial raster data. (For vector data the OGR Simple Features library is included). It is the underlying library for many geospatial software tools.

When using Conda environments, GDAL is installed with our standard environment (when installing geopandas). It includes command line tools that we can use from the Windows Miniconda (or Anaconda) console, or a *ix command line.

The library is very powerful and well worth learning

- Introductory reading: https://developers.planet.com/planetschool/getting-started-with-gdal/
- Documentation for command line tools: https://gdal.org/programs/index.html
- More information about raster processing https://geohackweek.github.io/raster/04-workingwithrasters/

Get information about geospatial attributes (as well as raster size and type) of a geospatial raster file

This is particularly useful to verify the file is in the correct projection and format.

```
gdalinfo infile
```

This can generate a lot of information as each band is queried. The command line flag -json produces output in JSON which can then be read into a program. The flag -stats produces statistics.

Make an RGB composite from three bands extracted from a multi-band file

Use gdal_translate: https://gdal.org/programs/gdal_translate.html#gdal-translate .

For example, to combine bands 75, 46, 20 of a HySpex VNIR reflectance or radiance file to generate a pseudo-true color RGB in ENVI format, without further rescaling, use:

```
gdal_translate -b 75 -b 46 -b 20 -of ENVI infile.bsq outfile.bsq
```

To generate a GeoTIFF, we want to rescale the raster values to 8-bit integers, with optimization for output size.

```
gdal_translate -b 75 -b 46 -b 20 -of GTiff -ot Byte -co "COMPRESS=JPEG" -co
"TILED=YES" infile.bsq outfile.tif
```

HOWEVER this process will typically rescale whatever the input bit depth is to 0...255, which can lead to a great loss of dynamic resolution. We typically need to add a scaling factor to stretch each input band range across the 8-bit range. For this it is useful to first extract the three bands without reformatting as indicated in the first example above (we can leave it as an ENVI .bsq file here). In a second step we extract statistics that we use for rescaling. The output may look like this

gdalinfo -stats -nomd -norat 20200830_BC_04_layerstackexample.bsq

```
Driver: ENVI/ENVI .hdr Labelled
[more raster and coordinate system information)
Band 1 Block=1751x1 Type=Float32, ColorInterp=Undefined
  Description = ...
  Min=0.000 Max=0.070
  Minimum=0.000, Maximum=0.070, Mean=0.010, StdDev=0.005
  NoData Value=0
Band 2 Block=1751x1 Type=Float32, ColorInterp=Undefined
  Description = Layer (Band
45:20200830-BC_04_VNIR_1800_SN00812_FOVx2_raw_rad_bsq_float32_geo.bsq)
(549.823000 Nanometers)
  Min=0.000 Max=0.106
  Minimum=0.000, Maximum=0.106, Mean=0.012, StdDev=0.007
  NoData Value=0
Band 3 Block=1751x1 Type=Float32, ColorInterp=Undefined
  Description = Layer (Band
75:20200830-BC_04_VNIR_1800_SN00812_FOVx2_raw_rad_bsq_float32_geo.bsq)
(644.911000 Nanometers)
  Min=0.000 Max=0.117
  Minimum=0.000, Maximum=0.117, Mean=0.009, StdDev=0.007
  NoData Value=0
```

In this case, the data type is Float32, but for example ATCOR output may be Int16... A good approach to rescaling is to rescale the range [0 ... mean + 2 * stdev] to [0...254] . (This leaves 255 available for a nodata value if desired.)

Then in a last step rescale the 3-band file to an 8-bit GeoTIFF. With scale values derived from the output above:

```
gdal_translate -of GTiff -ot Byte -scale_1 0 0.02 0 254 -scale_2 0 0.026 0
254 -scale_3 0 0.023 0 254 -co "COMPRESS=JPEG" -co "TILED=YES"
20200830_BC_04_layerstackexample.bsq 20200830_BC_04_layerstackexample.tif
```

The optional command line option `-a_nodata 0` can be used to assign a nodata value (zero here).

If rescaling multiple flightlines, we want to make sure that the same scale factors are applied to all flightlines, otherwise the illumination will appear different from flightline to flightline. Just get appropriate scaling statistics from one flightline (ideally, one of average brightness) and apply them to all flightlines.

Mosaicking multiple flightlines into a single mosaic

This is easy in GDAL with the combination of `gdalbuildvrt` and `gdal_translate`. With one caveat: We need to make sure that whatever value is written in the flightline edges is assigned to the nodata value of the raster. If you start with BSQ files, this can be done by editing the HDR file and setting the attribute `data ignore value = 0` . If starting with GeoTIFFs, you can use `gdal_translate` to assign a nodata value.

Once you have appropriate RGB composites, move them to a directory, or ensure that they all follow the same naming scheme. For example, my RGB composites might have flightlines like `20200830_BC_*_RGB.bsq` or `20200830_BC_*_RGB.tif.`

Mosaicking is now a two-step process: Build a virtual mosaic of all images with gdalbuildvrt (very fast - doesn't actually write data), then use gdal_translate to reformat the VRT file to a raster in the desired output format (GTiff or ENVI). The `-srcnodata 0` switch takes care of the edges. You may not need it if the nodata value is correctly declared in your infile.

```
gdalbuildvrt -srcnodata 0 mosaic.vrt infilepattern*
gdal_translate mosaic.vrt mosaic.tif
```

Stacking layers

GDAL is not the greatest when it comes to making new layer stacks. In particular if you want to select bands from multiple files to combine into a single file (eg. band 140 SWIR, 120 VNIR and 20 VNIR) there is no easy and straightforward way. You may want to install the `rasterio` library and `rio` command line utilities (https://rasterio.readthedocs.io/en/latest/cli.html) for that, or use ENVI. However, one doable path also uses `gdalbuildvrt` is:

1. Use `gdal_translate` repeatedly to extract single band files using the `-b` switch (for example, `-b 75`)
2. Stack the files using `gdalbuildvrt` and the `-separate` switch https://gis.stackexchange.com/questions/207030/merge-multi-band-geotiff-with-different-datatypes-with-gdal
3. Use `gdal_translate` to create the final file.

Masking

Jupyter Notebook 06 can be used to generate boolean raster masks from ENVI HySpex rasters as they come out of Parge (either from the VNIR and SWIR radiance and scan angle files, or from the stacked VNIR/SWIR raster from integrated processing), including cropping the left or right side of N-S flightlines. For further processing, these raster masks need to be processed to vector masks. The relevant GDAL command is (presuming raster masks in a `rastermasks` directory, and per-flightline vector masks in `vectormasks_flightline)` , for a single file or a list of flightlines, for example:

```
gdal_polygonize.py -8 rastermasks/vnir_swir_05_mask.bsq
vectormasks_flightline/vnir_swir_05_mask.shp
for i in {1..7}; do gdal_polygonize.py -8
rastermasks/vnir_swir_0${i}_mask.bsq
vectormasks_flightline/vnir_swir_0${i}_mask.shp
```

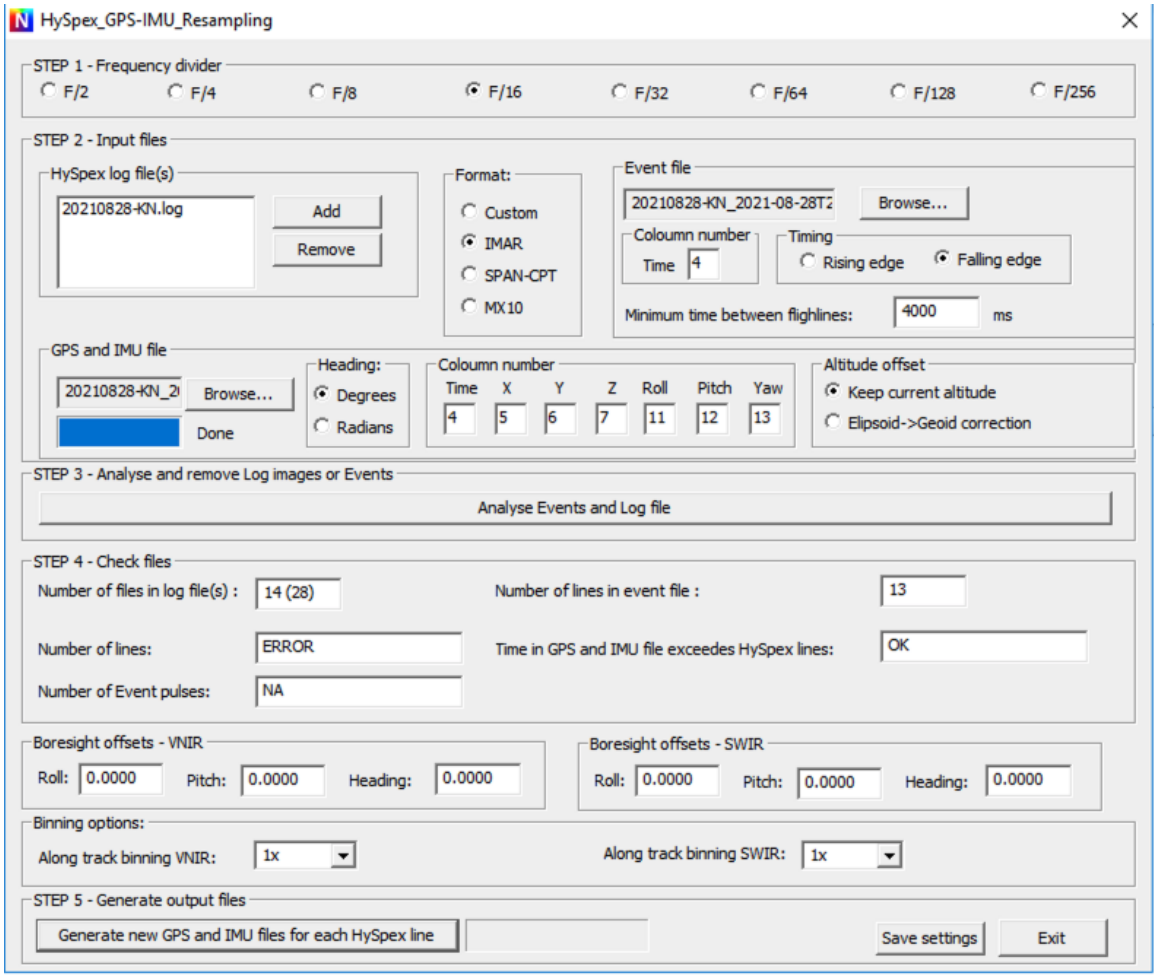
Jupyter Notebook 06a assists with further eroding vector masks (to eliminate undesirable effects at the edge of the valid data) and to apply for example an ROI crop. To apply vector masks to a flightine datafile, GDAL offers these options: `gdal_rasterize` can be used to burn the mask into each band of the ENVI raster. Alternatively, gdalwarp can also crop to a mask and fill the "outside" area with the predefined nodata value (we use 14000), and is easier to use in a loop that processes a whole set of files.

```
for i in {1..459}; do gdal_rasterize -i -burn 14000 -b ${i}
vectormask_cropped/vnir_swir_01_mask.shp
[datadir]/20200830-BC_01_VNIR_SWIR_rad_float32_geo_atm.bsq
```

```
for i in {1..7}; do gdalwarp -srcnodata 0 -dstnodata 14000 -of ENVI
-crop_to_cutline -cutline
vectormasks_cropped/vnir_swir_0${i}_buff5_crop.shp
[indir]/20200830-BC_0${i}_VNIR_SWIR_rad_geo_atm_bcor.bsq
[outdir]/20200830-BC_0${i}_VNIR_SWIR_rad_geo_atm_bcor_crop.bsq; done
```

APPENDIX 6: Known HySpex NAV bug leading to apparent file corruption

Occasionally, NAV returns an error after loading the event file. Typically, the number of flightlines detected in the event file is less than that found in the log file:



In this case, "Generate new GPS and IMU files..." will not work and the GPS file appears to be corrupted. In reality, this is ultimately a known bug in NAV. NEO support will be able to help and generate the per-flightline GPS/IMU files but we can work around it ourselves. One flightline may be lost in the operation.

The root cause of the bug: The GPS file (=event file) stores timestamps as seconds after start-of-the-week, each timestamp corresponding to an "event". The timestamp is therefore a

decimal number between 0 and 604800 (=number of seconds in a week). If there is a data acquisition ongoing during the start of the week, the timestamp will roll over:

```
758235   604799.986250 60.495378 -150.782973 2284.408446 1.488167 -1.736460 258.645156
758236   604799.991250 60.495377 -150.782978 2284.415493 1.494052 -1.751779 258.640801
758237   604799.996250 60.495377 -150.782984 2284.422485 1.494809 -1.773111 258.637956
758238   0.001250 60.495376 -150.782989 2284.429443 1.493060 -1.794794 258.636692
758239   0.006250 60.495375 -150.782994 2284.436421 1.490804 -1.804093 258.636997
758240   0.011250 60.495374 -150.783000 2284.443431 1.491042 -1.799430 258.635720
```

In this case, NAV is unable to handle the GPS file correctly.

How to work around the bug: The workaround requires NAV, a text editor, and possibly a scripting environment for post-processing.

1. Edit config.ini (in the folder that contains the NAV software) setting the configuration option `GPS_to_ASCII = 1`
2. Open NAV and load the LOG and GPS file. This will generate a plain text from the event file with the same filename except that the extension is TXT instead of GPS .
3. Open this TXT file in a text editor and split it in two at the rollover timestamp. Also split the log file. The exact split location (that is, at which flightline) may need some experimentation. In our case, the rollover happened between two flightlines (during the turn), so the full data was recovered. If the rollover happens within a flightline, you may be able to trim it.
4. Close NAV and re-open it, but now use it on the pairs of LOG/TXT files created in 3. For Format select Custom Instead of IMAR and use the column order fields to extract the six data columns. This makes it possible to create per-flightline IMU files in TXT format.
5. Check the output files, and in particular the column order. Parge expects it to be: 'rowid', 'longitude', 'latitude', 'elevation', 'roll', 'pitch', 'heading', 'timestamp'. By default, the TXT representation has latitude (= y) and longitude (= x) reversed. If this happens, you can either go back to 4 and experiment with the column order, or use a scripting language to reverse the respective columns in the per-flightine IMU files.

Implications for data acquisition: The week theoretically(*) begins at 00:00 UTC (midnight) Sunday, which is 16:00 AKDT Saturday afternoon. The week begins at 00:00 UTC (midnight) Sunday If data acquisition starts before this point and ends after it, the bug ***will*** be triggered. As rescuing the data takes considerable extra effort and manual intervention (or a round-trip to NEO), it is preferable to end a session before 16:00 on Saturdays and then start a new one after 16:00.

(*) In the example from which the screenshots were taken, this actually happened at 15:45 AKDT, so 15 minutes before midnight UTC. In any event, Saturday afternoon data acquisition risks being affected by this bug.