Practical Session - Working with airborne imaging spectrometer data

Will Jay & Dan Clewley – NEODAAS, Plymouth Marine Laboratory

1. Task

To process a typical NEODAAS Airborne hyperspectral flightline from Level 1 (radiometrically corrected, unmapped) to a Level 2 reflectance product then map to Level 3.

After there are some additional exercises using GDAL and calculating the Normalised Difference Vegetation Index (NDVI).

2. Dataset

For this practical data from the Specim ASIAFenix instrument collected by NEODAAS Airborne (formally NERC-ARF) will be used. These data were collected from a flight over Whitsand Bay, Cornwall, UK on 26th June 2018 as part of a study identifying ways to detect coastal plastic. The line of data used for the practical is shown in Figure 1.

The Fenix instrument covers visible and near infrared (VNIR) to short wave infrared (SWIR) portion of the electromagnetic spectrum, wavelengths (380 – 2500 nm) over a maximum of 622 bands.

For this practical line 2 will be used. This line was flown at an altitude of 747 m from north west to south east at 10:40 (GMT). Both VNIR and SWIR bands were spectrally binned by a factor of 2, providing a total of 622 bands.

Only a subset of a single line has been provided for this practical (zip 547 MB), the entire dataset can be downloaded from the Centre for Environmental Data Analysis (CEDA):

http://data.ceda.ac.uk/neodc/arsf/2018/GB18_186/GB18_186-2018_177a_Whitsand_Bay

Figure 1.Hyperspectral line used for practical session over Whitsand Bay. Subset of line that will be used throughout this practical is in colour.

3. Files to be used

Level 1 data files:

- f177a021b.bil – level 1 file containing calibrated radiances
- f177a021b.hdr - header file for above
- f177a021b_mask.bil - file specifying pixels deemed to be unreliable
- f177a021b_mask.hdr – header file for above

Navigation files:

- f177a021b_nav_post_processed.bil - file containing navigation information from the aircraft
- f177a021b_nav_post_processed.hdr - header file for above

View vector files

- fenix_fov_fullccd_vectors.bil - file to translate from the CCD array to the lens centre
- fenix_fov_fullccd_vectors.hdr - header file for above

DEM (Digital Elevation Model)

- GB18_186-2018_177a-ASTER.dem - digital elevation model produced from ASTER data
- GB18_186-2018_177a-ASTER.hdr - header file for above

Spectra

- 20180515_field_spec_targets.csv - reflectance spectra of white, grey and black targets measured in the field.
- f177a021b_targets_spectra.csv - average at-sensor radiance spectra of white, grey and black targets extracted from the hyperspectral line.

With the exception of the spectra, all files are ENVI Band Interleaved by Line (BIL) format files, which are binary files with a text file containing header information.

More information on these files will be given in the relevant sections.

Note that field spectra measurements need to be taken at the same time as the FENIX sensor records data. However due to issues with these data, this exercise uses spectra of the same targets in the same setting and similar conditions collected on 15th May 2018.

4. Software

The software needed for this practical are listed below and should already be installed in an Anaconda environment named ‘airborne’ as described in the setup instructions.

a. The Airborne Processing Library (APL) Suite

A set of hyperspectral processing tools developed at PML, described in the following paper:

M. A. Warren, B. H. Taylor, M. G. Grant, J. D. Shutler, Data processing of remotely sensed airborne hyperspectral data using the Airborne Processing Library (APL): Geocorrection algorithm descriptions and spatial accuracy assessment, Computers & Geosciences, Volume 64, March 2014, Pages 24-34, ISSN 0098-3004, http://dx.doi.org/10.1016/j.cageo.2013.11.006.

The following tools will be used for this practical:

- aplmask - Used to mask bad pixels from the final products
- aplcorr - Used to calculate the ground position of each pixel
- apltran - Used to convert between different projections
- aplmap - Used to create a georectified image using a level 1 file and positions calculated by

You should have installed APL as part of the workshop set up. Switch to the anaconda environment you installed to for this session. The source is available from: https://github.com/arsf/apl

b. NEODAAS Airborne Scripts

A set of scripts developed by NEODAAS (as part of NERC-ARF) are available from: https://github.com/pmlrsg/arsf_tools

- get_info_from_header.py - Used to get wavelengths and other information from header file.
- apply_elc_to_bil.py - Used to apply an empirical line calibration to a BIL fie.

c. GDAL

The Geospatial Data Abstraction Library (GDAL). Some utilities included with GDAL are used:

- gdal_translate: http://www.gdal.org/gdal_translate.html
- gdallocationinfo: http://www.gdal.org/gdallocationinfo.html
- gdal_calc.py: https://gdal.org/programs/gdal_calc.html

d. Tuiview

An open source raster viewer based on GDAL and written in Python (http://tuiview.org/)

5. Data Delivery Structure

The files to be used are provided in the same structure as you will receive in your own delivery. For clarity, files not used in this practical have been removed.

This structure is detailed below:

- dem - Digital Elevation Model
- doc - Documents such as the data quality report
- flightlines
    - level1b - Level 1 data and mask files
    - line_information - Metadata for each flightline
    - mapped - Mapped level 3 files
    - navigation - Post-processed plane navigation data
- logsheet - Logsheet for the flight detailing all flightlines
- project_information - Metadata for the project
- screenshots - JPG images, giving a quicklook for each mapped flightline
- sensor_FOV_vectors - CCD pixel view vectors

The only relevant directories for this task are dem, flightlines\level1b, flightlines\navigation and sensor_FOV_vectors.

Further information on directory structures will be provided in the Readme for your project. An up-to-date layout description is also located at:

https://nerc-arf-dan.pml.ac.uk/trac/wiki/Processing/FilenameConventions#Delivereddata

6. Getting Started

The tutorials use the Command Line Interface (CLI). The CLI requires typing commands into a terminal window rather than using the mouse to select options. Although the CLI is a more 'old school' way of using a computer it can be very powerful and allow you to automate tasks a lot easier. The commands you need to type use a fixed width font it is recommended you try and type them out rather than just copying and pasting. You can use the TAB key to complete file paths, type the first couple of letters of a file or path and press tab to complete. If there are multiple files with the same start you will need to type the additional characters and press tab again, once you get the hang of this using the command line will start to get more efficient.

The command treats a new line as an instruction to execute the command (the same as pressing the 'enter' button. Therefore, to enter commands spanning multiple lines you must break new lines with ^ on Windows (as shown in the examples) or \ on Linux or macOS.

Note, that when copying the commands from the word document sometime extra blank lines are added. It is recommended to copy commands out of the word document and into a text editor (Notepad on windows will work). You might want to save this text file for future reference.

Activate the `airborne' Anaconda environment, created as part of the course set up.

Navigate to the data directory by typing (for example):

```
cd C:\workshop\hyperspectral_practical\
```

Changing the path to where the data are stored.

Within the practical directory create a folder to store outputs called 'outputs'. You can do this from within a command prompt using the following command:

```
mkdir outputs
```

The rest of the examples in this workshop are for Windows, to use them on Linux or macOS replace the line continuation (^) with the Linux equivalent (\).

7. Convert at-sensor radiance to surface reflectance

Before starting processing, we will view the level1 data using TuiView. As TuiView was designed to view georeferenced data then we need to let it know we also want to use it for viewing level 1 (unmapped) data. We do this by typing the following in the terminal:

```
set TUIVIEW_ALLOW_NOGEO=YES
```

Note, if you are using Linux / macOS then you need to use:

```
export TUIVIEW_ALLOW_NOGEO=YES
```

To open TuiView with the level1b data, type the following into a terminal window.

```
tuiview flightlines\level1b\f177a021b.bil
```

To visualise in true colour, the following bands need to be set to their RGB channels:

- Red = 638.99 nm (Band 153)
- Green = 537.73 nm (Band 94)
- Blue = 467.83 nm (Band 53)

To do this, go to "Edit -> Stretch" and select these wavelengths in drop down menu in the “Bands” row. If the image is too dark or difficult to see you can change the value of the standard deviation to 1.0. Click the blue down arrow in the stretch toolbar to apply to the selected layer or three blue arrows to apply to all layers. Experiment with other values to adjust the display.

You can also specify the bands to load when starting tuiview. Close Tuiview and try opening again using:

```
tuiview --rgb --stddev --bands 153,94,53 flightlines\level1b\f177a021b.bil
```

In Tuiview, select the Query button (looks like a “+”) click on a pixel and select the plot tab. Click on points of the image to view the spectral profile for that pixel. Note that this only works on the top layer.

Figure 2 shows an example of the Query tool.

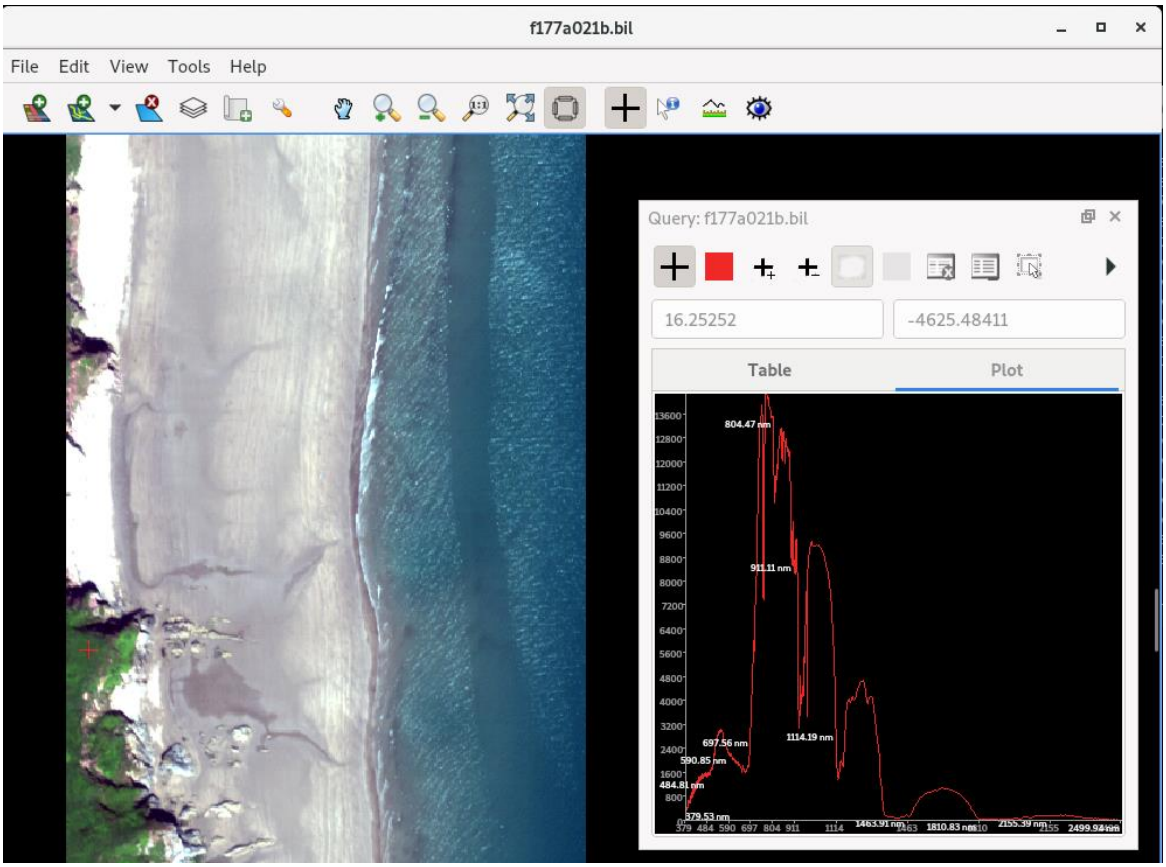

Figure 2. Example of using the query tool to view the radiance of a vegetation pixel.

8. Perform an Empirical Line Calibration

What is measured by the hyperspectral sensor on the plane and a spectrometer in the field differ due to absorption and scattering within the atmosphere. To normalise airborne data across different dates and allow comparison to field data it is often desirable to convert the hyperspectral data to units of surface reflectance. There are many ways of doing this, one method is an Empirical Line Calibration (ELC) which involves creating a relationship between image radiance and surface reflectance for each band. This is often accomplished by laying out a white, grey and black targets during the time of the overpass and taking measurements from them.

Targets were deployed for this flight and can be seen on the beach in the middle of the line, as shown in Figure 3. The three targets this exercise will use to conduct Empirical Line Correction are annotated.

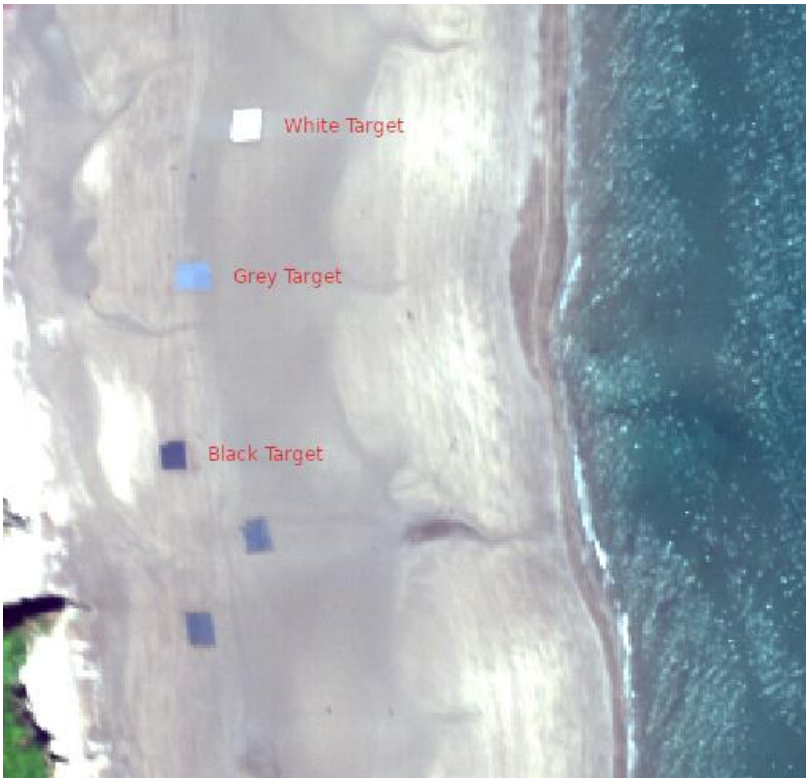

Figure 3. Location of the white, grey and black ground targets that field spectra measurements are associated with.

With the image still open in TuiView find the targets and look at the spectra from the targets. The mean spectra of pixels within each of the three targets mentioned have already been extracted to f177a021b_targets_spectra.csv. The spectra of these targets are shown in Figure 4.

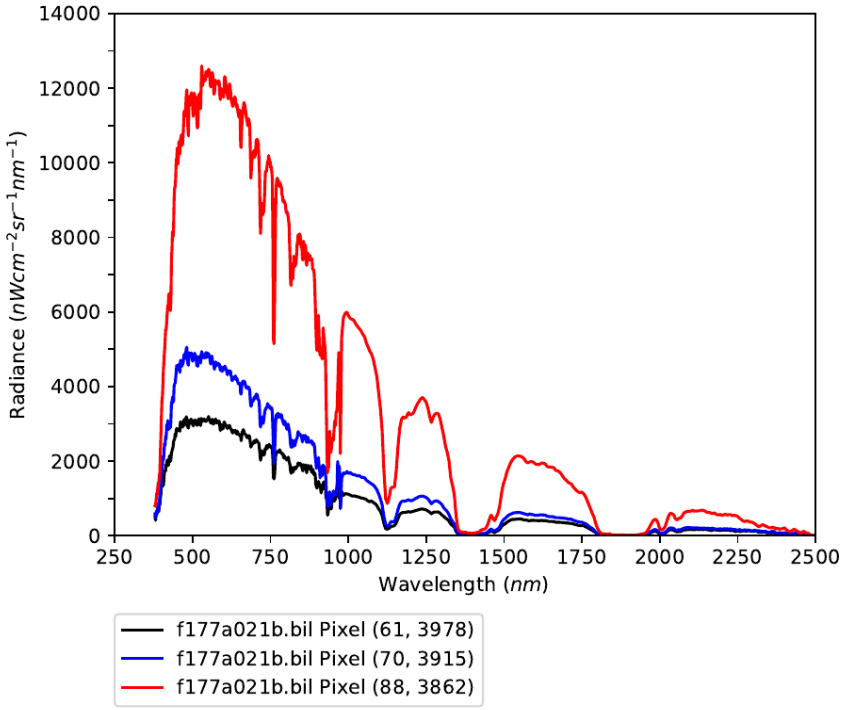

Figure 4. At-sensor radiance values of level 1b pixels within the White (red line), Grey (blue line) and Black (black line) targets. Note that these pixel coordinates are likely to be different in the subset data used in this exercise.

Although setting out the targets and taking measurements from them involves a lot of coordination and work, once collected it is relatively straightforward to apply the correction to the image. Various software packages contain an implementation of an Empirical Line Calibration (e.g., ENVI). For this practical a script written by NEODAAS will be used as it doesn't require a licence.

The average spectra from each of the targets have already been extracted and saved to spectra\ f177a021b_targets_spectra.csv. Multiple field spectra measurements were also taken from each of the targets, the average has been saved to spectra\20180515_field_spec_targets.csv. A plot of the three field spectra is shown in Figure 5. You may wish to open the files in Excel / LibreOffice Calc to view them.

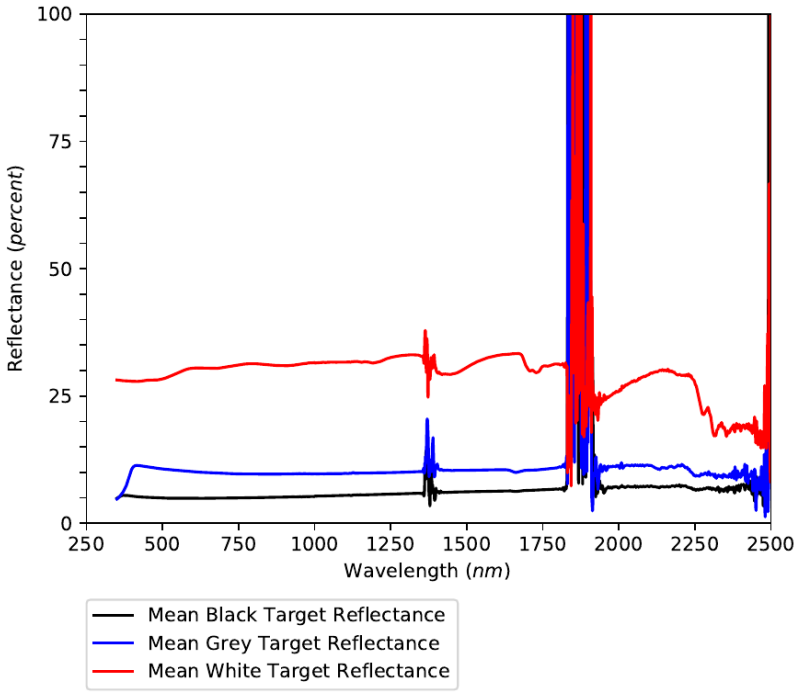

Figure 5. Mean reflectance of targets collected on 15th May 2018. Values have been scaled to 0 – 100 %.

Note that the field spectra in Figure 5 shows significant noise in the regions of 1350 – 1450 nm, 1800 – 1950 nm and wavelengths greater than 2750 nm. Think about why this could be.

To apply the Empirical Line Calibration, type the following command into a terminal window:

```
apply_elc_to_bil --image_spectra spectra\f177a021b_targets_spectra.csv ^
--field_spectra spectra\20180515_field_spec_targets.csv ^
flightlines\level1b\f177a021b.bil outputs\f177a022_elc.bil
```

This will take a couple of minutes to run through, as it is running it will print the line it is currently on. Note if this command isn't found try adding .py (`apply_elc_to_bil.py`).

Once the script has finished open the output file with the original file into separate TuiView windows with the following command:

```
tuiview --separate flightlines\level1b\f177a021b.bil outputs\f177a022_elc.bil
```

Try modifying the command above to load specific bands to RGB, as previously.

If you get the "Only north-up images allowed" error, then run the following command again:

```
export set TUIVIEW_ALLOW_NOGEO=YES
```

View the spectral profile for both BIL files by using the query button again. The reflectance values have been scaled by 10,000 so a value of 5,000 corresponds to a reflectance of 0.5 (50 %). This is done so images can be saved as integers rather than floating point (which requires less storage space).

After you have looked at the spectra from different cover types close the TuiView windows. You should notice that the areas of noise and error (with values exceeding 100 % reflectance) in the

regions previously mentioned (and shown on Figure 5) have been inherited by the reflectance values.

Accuracy Assessment

Additional field spectra data are required to evaluate the accuracy of the reflectance data produced. Figure 6 shows the field spectra of sand and two additional black targets (not used for the Empirical Line Correction). The majority of the spectra do not deviate notably from the field spectra, apart from the regions of noise and error previously identified.

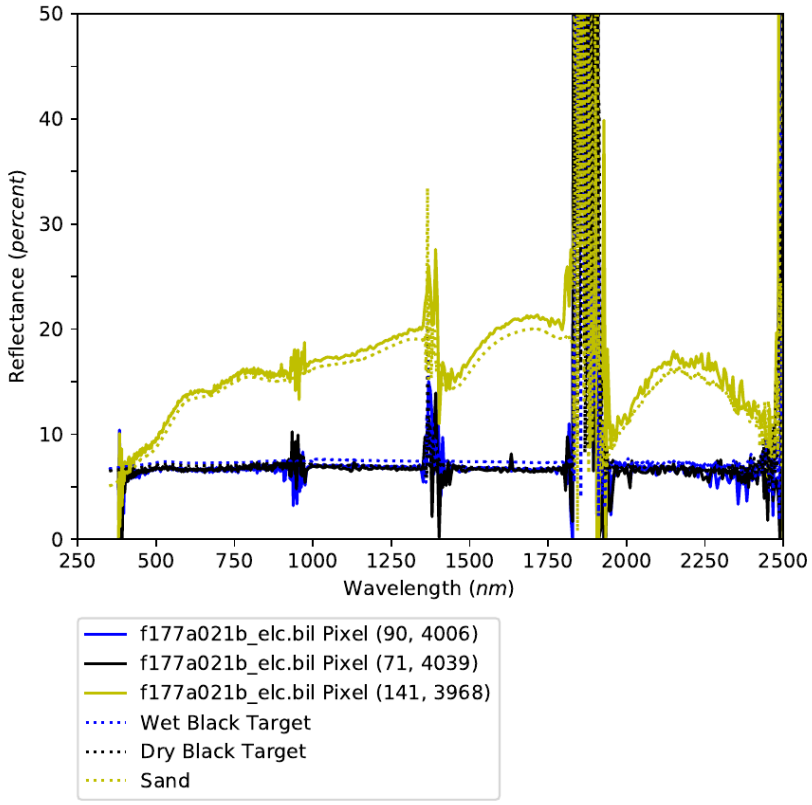

Figure 6. Comparison of field spectra not used for Empirical Line Correction (dotted line) and associated hyperspectral data pixels. Note that reflectance values have been scaled to 0 – 100 % and these pixel coordinates are likely to be different in the subset data used in this exercise.

Try editing the field spectra in Excel to remove the noisy bands, so the correction will interpolate these gaps using the closest surrounding wavelengths and running the process again. What difference does this make?

9. Data Processing using APL

The following steps will take the unmapped file and map to the UTM coordinate system so it can be used with other datasets. Note that these commands need to be run using Linux or Windows. If you are using macOS you will need to run APL using a Docker container (outlined in the course set up document). If there are problems with your installation, read and make sure you understand the commands and download the 'APL Output Data' from the Training Course website and use these to continue with the GDAL commands later.

a. Masking bad pixels with aplmask

The first step of the APL processing is to apply the mask of bad pixels to empirical line corrected data, creating a new file with these pixels set to 0. Pixels can be 'bad' for many reasons (see mask section below).

The mask file

This is a BIL file of the same dimensions as the raw file. For every single pixel in the image (for all bands), a value is given which is the summation of the below flags:

- 0 = Good data.
- 1 = Underflows.
- 2 = Overflows.
- 4 = Bad CCD pixels.
- 8 = Pixel affected by uncorrected smear.
- 16 = Dropped scans.
- 32 = Corrupt raw data.
- 64 = Quality control failures.

A single pixel can have multiple flags. This file was created at an earlier stage in the processing chain (namely by aplcal) when various algorithms are used to identify pixels as one of the above.

Header files

Although not always explicitly mentioned in this tutorial, every time you create a new .bil file, a matching .bil.hdr file will be created. This is required as it tells programs how to interpret the BIL file. You can examine this file in any text editor and it will give you useful information such as the number of bands and the width and length of the flightline.

Command

The command line instructions are all typed into a terminal window / command prompt. You can use the same window as the one used to start APL, you might need to press enter to get to a new line.

The first command to run is:

```
aplmask -lev1 outputs\f177a022_elc.bil ^
-mask flightlines\level1b\f177a021b_mask.bil ^
-output outputs\f177a022_elc_masked.bil
```

Command break down:

- -lev1: The input level 1 file
- -mask: The mask file to be used
- -output: The output level 1 file, with the mask applied

Go to a new line (by pressing Enter) to run the command.

Compare the masked files to the original by viewing in TuiView. Type:

```
tuiview --separate outputs\f177a022_elc.bil outputs\f177a022_elc_masked.bil
```

Use the query button on both TuiView windows to compare spectra in the original and masked files.

b. Generate an IGM file with aplcorr

The next step uses the navigation file, the view vector file and the digital elevation file (DEM) to work out the real world ground position for each pixel in the level 1 file.

The IGM file

The Input Geometry file is a two-dimensional binary image designed to hold the geographical position of each pixel in the level 1 file.

The navigation file

This is a binary BIL file which contains position and attitude information for the plane for the duration of the flightline. The file's "bands" in this case are 7 variables: time, latitude, longitude, height, pitch, roll and heading. There is a separate file for each instrument, due to the instruments differing position and orientation within the aircraft. This file is supplied with airborne data and is generated by NEODAAS using another tool in the APL suite, aplnav.

The view vector/field of view file

This file should not change within the lifetime of the sensor and is simply a way to translate from the CCD array to the principal point of the lens and hence to the correct position on the ground. Beyond remembering to include it, you do not need to worry about this file.

The DEM file

The Digital Elevation Model is a binary file for which each pixel has a value corresponding to its height above a given reference. Our DEM is measured against the WGS84 ellipsoid and has a pixel size of 30m. It has been produced from data from the satellite based ASTER sensor, through a collaboration between Japan and NASA (http://www.jspacesystems.or.jp/ersdac/GDEM/E/).

It is perfectly possible to process your data without a DEM, however where the terrain varies significantly, this will result in positioning errors.

Command

Generate the IGM file from the command line typing the following into a terminal window:

```
aplcorr -vvfile sensor_FOV_vectors\fenix_fov_fullccd_vectors.bil ^
-igmfile outputs\f177a021b.igm -lev1file outputs\f177a022_elc_masked.bil ^
-dem dem\GB18_186-2018_177a-ASTER.dem ^
-navfile flightlines\navigation\f177a021b_nav_post_processed.bil
```

Command break down:

- -lev1file: The input level 1 file (can be the masked or original, doesn't matter for this stage)
- -igmfile: The output IGM file
- -vvfile: The sensor's view vector /field of view file
- -navfile: The navigation file for the flightline
- -dem: The Digital Elevation Model

c. Change the projection of the IGM with apltran

The IGM output by aplcorr is always in latitude-longitude, so if you want to change the projection, for instance to match another dataset, then you should use apltran. We will be changing the projection from latitude-longitude to UTM (Universal Transverse Mercator).

Command

Enter the following into the terminal window:

```
apltran -output outputs\f177a021b_UTM30N.igm -igm outputs\f177a021b.igm ^
-outproj utm_wgs84N 30
```

Command break down:

- -igm: Your input IGM file created by aplcorr, in lat-long
- -output: Your output IGM file (which will be in UTM zone 30 Northern Hemisphere)
- -outproj: The output projection – if UTM be sure to give the correct UTM zone (30 for this flight)

d. Creating a mapped image with aplmap

The final stage is to combine the level 1 files containing the radiances with the IGM in order to create a level 3 image for which every pixel is mapped to the correct location in space, for a given co-ordinate system.

We are going to do this stage twice, creating two different mapped images.

The level 3 file

This is a binary BIL file just like the others. However, the rectangular image contained within has pixels positioned within a given co-ordinate system. The header file gives the nature of this system and the position of the image within, allowing these files to be opened in geographical software and displayed in the correct place, overlaying with other datasets if desired. As the image has to be rectangular, there will be a varying amount of empty/no data pixels in the image, resulting in a larger size compared to level 1 files.

Pixel Size

The desired output pixel size varies according to height above ground level (AGL). See this webpage for how to calculate the approximate size to use:

https://nerc-arf-dan.pml.ac.uk/trac/wiki/Processing/PixelSize

Command

Firstly, we will map the whole flightline again using three selected bands.

Command:

```
aplmap -lev1 outputs\f177a022_elc_masked.bil ^
-mapname outputs\f177a023_elc_utm_wgs84N30_mapped_3band.bil ^
-bandlist 153 94 53 -igm outputs\f177a021b_UTM30N.igm -pixelsize 2 2 ^
-outputdatatype uint16
```

Command Break Down:

- -igm: The transformed IGM created in the above step
- -lev1: The masked BIL file created by aplmask in step 1
- -mapname: The output georectified BIL image
- -pixelsize: Output pixel size, in metres, in the x and y direction
- -bandlist: A space separated list of bands to be placed in the output image
- -outputdatatype: Data type for output image, ‘uint16’ is a 16 bit unsigned integer, the same as the unmapped file.

Then we'll run a spatial subset, using all bands, with the IGM projected to Easting-Northings (UTM30N). This is the same projection that Sentinel 2 data uses, so this data can be viewed along with the outputs from the Sentinel 2 practical.

```
aplmap -lev1 outputs\f177a022_elc_masked.bil ^

-mapname outputs\f177a023_elc_utm_wgs84N30_mapped_subset.bil ^

-bandlist ALL -igm outputs\f177a021b_UTM30N.igm -pixelsize 2 2 ^

-outputdatatype uint16 ^

-area 408900 409247 5578700 5579000
```

Command Break Down:

- -area: A rectangular region specifying the area to process in the order min X, max X, min Y, max Y. We are using UTM coordinates so these are in metres, the X value measured from the edge of the UTM zone and the Y value from the equator.

10. Additional Analysis using GDAL

Unlike the subset area or band images you have created above, the level 3 mapped images supplied in a NEODAAS delivery are of the whole flightline and for all bands. Once unzipped, these files can run into 10s of gigabytes (so for space reasons, an example file has not been included in the practical).

Change format

```
gdal_translate -of GTiff -co COMPRESS=LZW ^

outputs\f177a023_elc_utm_wgs84N30_mapped_3band.bil ^

outputs\f177a023_elc_utm_wgs84N30_mapped_3band_tif.tif
```

Command break down:

- -of: Output Format - we are using GeoTiff file. For all available options see http://www.gdal.org/formats_list.html
- -co: Creation Option – specifies the creation options, these are specific to the format. For GeoTiff we use lossless compression to reduce the file size. For ENVI we can use the creation options to specify the Interleave (BSQ or BIL).
- final two arguments: input and output filenames

Open Windows explorer and note the difference in size between the uncompressed .bil file and the compressed .tif. For mapped files there are lots of pixels which don't contain any data, by using a compressed format we reduce the data size.

Extract pixel values for a given location using GDAL

Previously you have viewed spectral profiles in TuiView. To extract data to plot in another package for further analysis the gdallocatininfo command can be used:

```
gdallocationinfo –geoloc ^
outputs\f177a023_elc_utm_wgs84N30_mapped_subset.bil 408968 5578870
```

Command break down:

- -geoloc: Interpret x and y values as coordinates rather than pixel and line
- first argument: file name
- x y: Easting and Northing of pixel to extract statistics from

This will print the values to the screen. To save as a text file we can redirect the output using '>' we also use the '-valonly' flag to only print the values to the screen.

```
gdallocationinfo -valonly –geoloc ^
outputs\f177a023_elc_utm_wgs84N30_mapped_subset.bil ^
408968 5578870 > outputs\extracted_values.txt
```

This text file can be opened in Excel / LibreOffice for plotting and further analysis.

Calculating Band Indices (e.g., NDVI)

The Normalised Difference Vegetation Index (NDVI) is a commonly used index of vegetation productivity and is a ratio reflectance (ρ) in the of NIR and Red bands.

$$NDVI = \frac{(\rho_{NIR} - \rho_{red})}{(\rho_{NIR} + \rho_{red})}$$

For this example, we will use 800 nm for the NIR channel and 640 nm for the Red channel.

Get wavelengths

The first step is to extract the wavelengths from the header file to a .csv file, which can be opened in Excel / LibreOffice. For this the 'get_info_from_header.py' script from NEODAAS is used:

```
get_info_from_header -o outputs\f177a022_elc_masked_wavelengths.csv ^
outputs\f177a022_elc_masked.bil.hdr
```

Again if you are not using Windows you will need to add .py (`get_info_from_header.py`).

Click to open this file in Excel / LibreOffice. Find the band numbers corresponding to approximately 800 nm and 640 nm.

Calculate NDVI

The next step is to use the bandmath.py command to calculate NDVI for each pixel within a hyperspectral file, for this the gdal_calc.py program will be used.

Note, Windows users might have problems running gdal_calc.py as Windows doesn't recognise it should be run as a Python script. If you run

```
where gdal_calc.py
```

This will print the path to the script, if you run now type 'python' and past the path to this script, for example:

```
python C:\Users\dac\AppData\Local\Miniconda3\Scripts\gdal_calc.py
```

This should then work.

```
python PATH\to\script\gdal_calc.py ^
--calc="((A.astype(numpy.float32) - B.astype(numpy.float32))/ ^
(A.astype(numpy.float32) + B.astype(numpy.float32)))" ^
--outfile=outputs\f177a022_elc_masked_NDVI.bil ^
-A outputs\f177a022_elc_masked.bil --A_band=800nm_band ^
-B outputs\f177a022_elc_masked.bil --B_band=640nm_band ^
--format=ENVI --co="INTERLEAVE=BIL" --type=Float32
```

Replace **800nm_band** and **640nm_band** are the band numbers you have just found.

Command break down:

- --calc: equation to calculate
- -A, -B: files to use for A and B expressions in equation. Don't be put off by the "astype(numpy.float32)" part, we just need to define the value's data types with this data we're using. The equation otherwise would simply be "(A - B)/(A + B)".
- --A_band, --B_band, bands to use for the A and B terms in the equation.
- --outfile: output file

View using:

```
tuiview outputs\f177a022_elc_masked_NDVI.bil
```

Values closer to 1 will be pixels containing more productive vegetation. Note, there might be some gaps due to bad pixels being masked out.

You can visualise this with colour by adding a colourmap. In the 'Layers' window, right click on the layer to visualise and click 'Edit Stretch'. In the window that appears, select 'PseudoColor' in the 'Mode' dropdown box and "RdYlGn (Diverging)" as the colourmap (which is good for NDVI). An example of NDVI data with this colourmap and the stretch setting used is present in Figure 7.

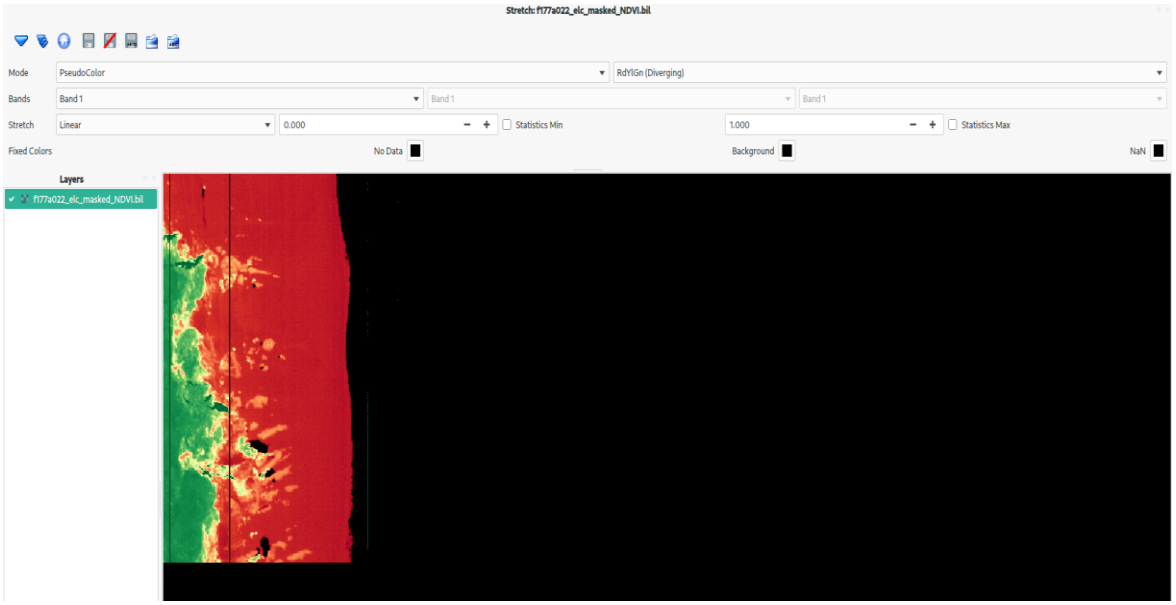

Figure 7. Unmapped FENIX NDVI data displayed with a Red-Yellow-Green colour map. Stretch has had a manually range of 0 to 1 applied.

Map NDVI

The outputs from bandmath.py are ENVI BIL format, as used by APL so the file created can be mapped using:

```
aplmap -lev1 outputs\f177a022_elc_masked_NDVI.bil ^
-mapname outputs\f177a021b_NDVI_utm_wgs84N30_mapped.bil ^
-bandlist 1 -igm outputs\f177a021b_UTM30N.igm ^
-pixelsize 2 2 -outputdatatype float32
```

Note as the values are floating point the outputdatatype is set to float32 so information isn't lost when mapping.

View the mapped NDVI and the 3 band Fenix line using:

```
tuiview --separate outputs\f177a023_elc_utm_wgs84N30_mapped_3band.bil ^
outputs\f177a021b_NDVI_utm_wgs84N30_mapped.bil
```

You may wish to use this method to calculate other band metrics such as Normalised Difference Water Index (NDWI).

This practical has provided an overview of using APL to map hyperspectral data delivered by NEODAAS and some tools you can use to create additional products which you can apply to your own data.

Installation Problem

If running `apltran –help` in the command line result with an error like below:

```
apltran: error while loading shared libraries: libproj.so.12: cannot open shared object file: No such file or directory
```

This might be resolved by installing the following package in the airborne conda environment:

```
conda install –c conda-forge/label/old_name proj.4
```