HypPy Scripting Manual[1]

command-line interface (CLI)

Wim Bakker

March 11, 2024

Contents

1 Scripting with HypPy **1**

1.1 Example script on Linux 1

1.2 Example script on Windows 2

2 Command Line Usage **5**

2.1 Atmoshpheric Correction 5

2.2 Band Depths 5

2.3 Class statistics 6

2.4 Colorize 6

2.5 Convert EOS HDF 6

2.6 Convert to ENVI format 7

2.7 Convert ENVI decision tree to ASCII decision tree 7

2.8 Convert / Resample Spectral Library 7

2.9 Convex Hull Removal 8

2.10 Convolve 8

2.11 Dark & White Reference Correction 8

2.12 Decision Tree 9

2.13 Destriping and Illumination Correction 9

2.14 Hyperspectral Edge Filtering 10

2.15 Fix Missing Data 10

2.16 Fix 8th pixel striping in SWIR camera 10

2.17 Flip Image in X, Y or Z 11

2.18 Geo-correction 11

2.19 GLT correction, forward and backward 11

2.20 Get Transmittance 12

2.21 Hyperspectral Gradient Filtering 12

2.22 Linear Filter 13

2.23 Log Residuals 13

2.24 Make Legend 13

2.25 Mask Noisy Bands 13

2.26 Merge (Stack) 14

2.27 Hyperspectral Median Filter 14

2.28 Wavelength of Minimum 14

2.29 Normalize Bands 15

2.30 Band Ratio 15

2.31 Band Ratios 15

2.32 Replace Values 16

2.33 Resample Image 16

2.34 Smile Measurement 16
2.35 Solar Correction 17
2.36 Sort Bands by Wavelength 17
2.37 Spatial Binning 17
2.38 Spatial Spectral Binning 17
2.39 Spectral Binning 18
2.40 Spectral Gradient 18
2.41 Spectral Math 18
2.42 Split into Separate Bands 19
2.43 Statistics 19
2.44 Subset Image 19
2.45 Summary Products, Vivian Beck and other indices 20
2.46 Thermal Correction 21
2.47 Convert to PNG, PGW and KML 21
2.48 Get Transmittance 22
2.49 Wavelength Mapping 22
2.50 WMS Get 22
2.51 Zonal Statistics 23

Abstract

This document is meant for users of the command-line applications in the Hyperspectral Python (HypPy)[1] package.

Prerequisites

The programs described in this document were developed using the Python programming language version 2.6. Check the User Manual for references to additional modules that are needed.

The core of these programs forms the envi2 module (see the HypPy Programmers' Guide), which relies heavily on the array processing functionality of the Numerical Python (NumPy) package.

The continuum removal program uses a modified version of the Quick Hull algorithm for finding the convex hull.

Some programs use the Image and ImageTk[2] modules from the Python Imaging Library for reading and writing images, and Matplotlib (a.k.a. PyLab) for making 2D graphs of spectra.

The thermal correction uses the least squares estimator from the Scientific Python (SciPy) package.

[1] In Finland a "hyppy" is a crest in the road that sends a car flying through the air.

[2] On Linux, Image and ImageTk are two separate modules!

Chapter 1

Scripting with HypPy

1.1 Example script on Linux

Example on Linux for processing one OMEGA scene:

```
#!/bin/sh

PATH=$PATH:/home/bakker/Python/HypPy

echo mask_noisy_bands
mask_noisy_bands.py -i ORB0422_4_jdat -t 25.0 -p -v

echo geo_correction
geo_correction.py -f -p -i ORB0422_4_jdat -o ORB0422_4_jdat_Gcor
    -g ORB0422_4_geocube -m omega

echo solar_correction
solar_correction.py -f -i ORB0422_4_jdat_Gcor
    -o ORB0422_4_jdat_Gcor_Scor -s ORB0422_4_specmars.txt

echo thermal_correction
thermal_correction.py -f -i ORB0422_4_jdat_Gcor_Scor
    -o ORB0422_4_jdat_Gcor_Scor_Tcor -t ORB0422_4_jdat_Gcor_Scor_T

echo atmo_correction
atmo_correction.py -f -i ORB0422_4_jdat_Gcor_Scor
    -o ORB0422_4_jdat_Gcor_Scor_Acor
    -t /home/bakker/Python/HypPy/Atmosphere/transmission.dat
    -a ORB0422_4_jdat_Gcor_Scor_alpha

echo logresiduals
logresiduals.py -s -b -f -i ORB0422_4_jdat_Gcor_Scor_Acor
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr
    -a ORB0422_4_jdat_Gcor_Scor_Acor_albedo
    -r ORB0422_4_jdat_Gcor_Scor_Acor_rlub.txt -p
```

```
echo median
median.py -f -i ORB0422_4_jdat_Gcor_Scor_Acor_lr
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med

echo hull
hull.py -f -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_cr -c 3.5

echo summary_products
summary_products.py -f -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_sp
    -c ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_cr

echo tokml
tokml.py -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_sp
    -R 12 -G 13 -B 14

# extra stuff...
echo minwavelength
minwavelength.py -f -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_wav -w 2.0 -W 2.4

echo wavemap
wavemap.py -f -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_wav
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_wav_map -w 2.0 -W 2.4
    -d 0.0 -D 0.15

echo Google Earth
googleearth /data/tmp/ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_sp.kml
```

1.2 Example script on Windows

Example batch file on Windows for processing one OMEGA scene:

```
rem Change the following to your Python executable
set PYTHON="O:\GroupData\ESATools\PortablePython_1.1_py2.6.1\App\python.exe"

rem Change the following to your HypPy directory
set HYPPY="O:\GroupData\ESATools\HypPy"

rem Change the following to your Google Earth executable
set GOOGLEEARTH="C:\Program Files (x86)\Google\Google Earth\client\googleearth.exe"

echo mask_noisy_bands
%PYTHON% %HYPPY%\mask_noisy_bands.py -i ORB0422_4_jdat -t 25.0 -p -v

echo geo_correction
%PYTHON% %HYPPY%\geo_correction.py -f -p -i ORB0422_4_jdat
    -o ORB0422_4_jdat_Gcor -g ORB0422_4_geocube -m omega
```

```
echo solar_correction
%PYTHON% %HYPPY%\solar_correction.py -f -i ORB0422_4_jdat_Gcor
    -o ORB0422_4_jdat_Gcor_Scor -s ORB0422_4_specmars.txt

echo thermal_correction
%PYTHON% %HYPPY%\thermal_correction.py -f -i ORB0422_4_jdat_Gcor_Scor
    -o ORB0422_4_jdat_Gcor_Scor_Tcor
    -t ORB0422_4_jdat_Gcor_Scor_T

echo atmo_correction
%PYTHON% %HYPPY%\atmo_correction.py -f
    -i ORB0422_4_jdat_Gcor_Scor -o ORB0422_4_jdat_Gcor_Scor_Acor
    -t transmission.dat -a ORB0422_4_jdat_Gcor_Scor_alpha

echo logresiduals
%PYTHON% %HYPPY%\logresiduals.py -s -b -f
    -i ORB0422_4_jdat_Gcor_Scor_Acor
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr
    -a ORB0422_4_jdat_Gcor_Scor_Acor_albedo
    -r ORB0422_4_jdat_Gcor_Scor_Acor_rlub.txt -p

echo median
%PYTHON% %HYPPY%\median.py -f -i ORB0422_4_jdat_Gcor_Scor_Acor_lr
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med

echo hull
%PYTHON% %HYPPY%\hull.py -f
    -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_cr -c 3.5

echo summary_products
%PYTHON% %HYPPY%\summary_products.py -f
    -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_sp
    -c ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_cr

echo tokml
%PYTHON% %HYPPY%\tokml.py
    -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_sp -R 12 -G 13 -B 14

rem Extra stuff...
echo minwavelength
%PYTHON% %HYPPY%\minwavelength.py -f
    -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_wav -w 2.0 -W 2.4

echo wavemap
%PYTHON% %HYPPY%\wavemap.py -f
    -i ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_wav
    -o ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_wav_map -w 2.0 -W 2.4
    -d 0.0 -D 0.15

echo Google Earth (untested!)
%GOOGLEEARTH% ORB0422_4_jdat_Gcor_Scor_Acor_lr_med_sp.kml
```

Chapter 2

Command Line Usage

2.1 Atmoshpheric Correction

```
Usage: atmo_correction.py -b -f -i input -t transmission -o output
                          -a alpha

Atmospheric correction.

Options:
  -h, --help  show this help message and exit
  -b          use bad band list from the header
  -f          force overwrite on existing output files
  -i INPUT    input file name
  -t TRANS    input transmission file name
  -o OUTPUT   output file name
  -a ALPHA    output optical depth (alpha) file name
```

2.2 Band Depths

```
Usage: banddepths.py -s -b -f -i input -o output -d delta

Calculate Band Depths.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -d DELTA    delta, shift between bands (default=1)
```

2.3 Class statistics

```
usage: classstats.py [-h] -c CLASSNAME [-s] [-b] [-i INPUT] [-o OUTPUT] [-l SPECLIB]
                     [-r REPORT] [-t TEXREPORT]

Get class statistics.

optional arguments:
  -h, --help    show this help message and exit
  -c CLASSNAME  input ENVI classification file
  -s            sort bands on wavelength
  -b            use bad band list from the header
  -i INPUT      input ENVI image
  -o OUTPUT     output plot file (.png or .pdf)
  -l SPECLIB    output directory for ascii spectral library
  -r REPORT     output report (text file)
  -t TEXREPORT  output report (LaTeX file)
```

2.4 Colorize

```
usage: colorize.py [-h] [-s] [-b] [-f] -i INPUT1 -j INPUT2 -o OUTPUT
                   [-I BAND1] [-J BAND2] -w WSTART -W WEND -d DSTART -D DEND
                   [-c {rainbow,steps}] [-C COLORFILE] [-l]

Make color/intensity map using color table and HSV transform

optional arguments:
  -h, --help          show this help message and exit
  -s                  sort bands on wavelength
  -b                  use bad band list from the header
  -f                  force overwrite on existing output file
  -i INPUT1           input file name for color
  -j INPUT2           input file name for intensity
  -o OUTPUT           output file name
  -I BAND1            input band for color
  -J BAND2            input band for intensity
  -w WSTART           lower stretch value for color
  -W WEND             upper stretch value for color
  -d DSTART           lower stretch value for intensity
  -D DEND             upper stretch value for intensity
  -c {rainbow,steps}  color table: rainbow (default) or steps
  -C COLORFILE        input color table file
  -l                  save legend in a .png file
```

2.5 Convert EOS HDF

```
usage: converthdf.py [-h] [-f] -i INPUT -o OUTPUT

Convert OES HDF image to ENVI format.
```

```
optional arguments:
  -h, --help  show this help message and exit
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
```

2.6 Convert to ENVI format

```
usage: convert.py [-h] [-f] -i INPUT -o OUTPUT

Convert image to ENVI format.

optional arguments:
  -h, --help  show this help message and exit
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
```

2.7 Convert ENVI decision tree to ASCII decision tree

```
usage: converttree.py [-h] [-f] [-d] -i INPUT -o OUTPUT

Convert ENVI or ASCII Decision Tree to ASCII Decision Tree or dot graph

optional arguments:
  -h, --help  show this help message and exit
  -f          force overwrite of output file
  -d          convert tree to dot graph
  -i INPUT    input tree
  -o OUTPUT   ouput tree
```

2.8 Convert / Resample Spectral Library

```
usage: resamplespeclib.py [-h] [-s] [-b] [-r] [-m WMULTIPLIER] -i INPUT
                          [-t TOSPEC] (-o OUTPUT | -e ENVIOUTPUT)

Convert / Resample Spectral Library.

optional arguments:
  -h, --help      show this help message and exit
  -s              sort bands on wavelength
  -b              use bad band list from the header
  -r              recursively scan input directory
  -m WMULTIPLIER  input wavelength multiplier
  -i INPUT        input spectral library
  -t TOSPEC       resample to
  -o OUTPUT       output ASCII directory
  -e ENVIOUTPUT   output ENVI Speclib
```

2.9 Convex Hull Removal

```
Usage: hull.py -s -b -f -i input -o output -m {div|sub} -c cutoff

Convex hull removal.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -m MODE     mode: div (divide, default), sub (subtract)
  -c CUTOFF   cutoff wavelength (or band)
```

2.10 Convolve

```
Usage: convolve.py -f -i input -o output -k kernel
                   --bias bias --offset offset

Linear filter

Options:
  -h, --help       show this help message and exit
  -s               sort bands on wavelength
  -b               use bad band list from the header
  -f               force overwrite on existing output file
  -i INPUT         input file name
  -o OUTPUT        output file name
  -k KERNEL        kernel: smooth, laplace or 9 weights
  --bias=BIAS      Constant for bias (default 1.0)
  --offset=OFFSET  Constant for offset (default 0.0)
```

2.11 Dark & White Reference Correction

```
usage: darkwhiteref.py [-h] [-f] [-m MANIFEST] [-i INPUT] [-d DARKREF]
                       [-w WHITEREF] [-o OUTPUT]
                       [-t {uint8, int16, int32, float32, float64, complex64,
                        complex128, uint16, uint32, int64, uint64}]
                       [-r]

Dark & white reference correction
```

```
optional arguments:
  -h, --help            show this help message and exit
  -f                    force overwrite on existing output file
  -m MANIFEST           manifest xml file name
  -i INPUT              input file name
  -d DARKREF            dark reference file name
  -w WHITEREF           white reference file name
  -o OUTPUT             output file name
  -t {uint8,int16,int32,float32,float64,complex64,complex128,uint16,uint32,int64,uint64}
                        output data type (default float32)
  -r                    robust white reference
```

2.12 Decision Tree

```
usage: decisiontree.py [-h] -t TREE -o OUTPUT [-b1 image band]
                       [-b2 image band] [-b3 image band] [-b4 image band]
                       [-b5 image band] [-b6 image band] [-b7 image band]
                       [-b8 image band] [-b9 image band]

Execute ENVI Decision Tree

optional arguments:
  -h, --help      show this help message and exit
  -t TREE         input ENVI decision tree
  -o OUTPUT       output image file name
  -b1 image band  variable b1
  -b2 image band  variable b2
  -b3 image band  variable b3
  -b4 image band  variable b4
  -b5 image band  variable b5
  -b6 image band  variable b6
  -b7 image band  variable b7
  -b8 image band  variable b8
  -b9 image band  variable b9
```

2.13 Destriping and Illumination Correction

```
Usage: destripe.py -s -b -f -i input -o output -d {hor, ver}
                   -m {sub, div} -p {0, 1, 2, ...} -F

Destriping (p=0) and illumination correction (p>0)

Options:
  -h, --help    show this help message and exit
  -s            sort bands on wavelength
  -b            use bad band list from the header
  -f            force overwrite on existing output file
  -i INPUT      input file name
  -o OUTPUT     output file name
  -d DIRECTION  direction: hor (horizontal, default) or
                ver (vertical)
  -m MODE       correction method: sub (subtract, default) or
                div (divide)
  -p ORDER      polynomial order of correction: destriping
                (p=0, default) or illumination correction (p>0)
  -F            fast version, keeps entire image in memory!
```

2.14 Hyperspectral Edge Filtering

```
Usage: edgy.py -s -b -f -i input -o output -m {SAM|ED|ID|BC|SID}

Hyperspectral edge filter

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -m MODE     mode: SAM (spectral angle, default),
              ED (Euclidean distance), ID (intensity difference),
              BC (Bray-Curtis), SID (spectral
              information divergence)
```

2.15 Fix Missing Data

```
Usage: fixnodata.py -s -b -f -i input -o output -n string

Set nodata values to NaN

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -n NODATA   list of nodata values (=string!)
```

2.16 Fix 8th pixel striping in SWIR camera

```
usage: fixswir.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT

Fix unruly samples of SWIR camera.
```

```
optional arguments:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input image file name
  -o OUTPUT   output image file name
```

2.17 Flip Image in X, Y or Z

```
usage: flip.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT [-m {x,y,z}]

Flip image

optional arguments:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite of existing output file
  -i INPUT    input image file name
  -o OUTPUT   output image file name
  -m {x,y,z}  flip mode: x (flip left and right (default)), y (flip up and
              down), z (flip spectrum)
```

2.18 Geo-correction

```
Usage: geo_correction.py -s -b -f -p -i input -o output
                         -g latlon|geocube -m {latlon|omega}

Geocorrection using latlon or geocube file. OMEGA files should
have 352 bands in the original order.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -p          plot geographic extent
  -g GEOCUBE  input latlon or geocube file name
  -m MODE     mode: latlon (use latitude/longitude file, default),
              omega (use geocube)
```

2.19 GLT correction, forward and backward

```
usage: glt.py [-h] [-s] [-b] [-f] -i INPUT -g GLT -o OUTPUT
              [-m {forward,backward}]
```

```
Forward or backward GLT transform.

optional arguments:
  -h, --help            show this help message and exit
  -s                    sort bands on wavelength
  -b                    use bad band list from the header
  -f                    force overwrite on existing output file
  -i INPUT              input image file name
  -g GLT                input GLT file name
  -o OUTPUT             output image file name
  -m {forward,backward}
                        mode forward (default) or backward
```

2.20 Get Transmittance

```
Usage: transmittance.py -s -b -f -i input -m mola -t transmittance

Calculate Atmospheric Transmittance from OMEGA scene.

Options:
  -h, --help        show this help message and exit
  -s                sort bands on wavelength
  -b                use bad band list from the header
  -f                force overwrite on existing output file
  -i INPUT          input file name
  -m MOLA           MOLA input file name
  -t TRANSMITTANCE  transmittance output file name
```

2.21 Hyperspectral Gradient Filtering

```
Usage: gradient.py -s -b -f -i input -o output
                   -m {SAM|ED|ID|BC|SID} -a
                   X Y XY U D UD XYUD E4 E8 SOBX SOBY SOBEL

Hyperspectral edge detection and gradient filters.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -a          select all gradient filters
  -m MODE     mode: SAM (spectral angle, default),
              ED (Euclidean distance), ID (intensity difference),
              BC (Bray-Curtis), SID (spectral information
              divergence)
```

2.22 Linear Filter

See Convolve.

2.23 Log Residuals

```
Usage: logresiduals.py -s -b -f -k -i input -o output -a albedo
                       -r rlub -p -n stddevs

Normalize image data using Log Residuals or Kwik Residuals

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -k          use kwik residuals instead of log residuals
  -a ALBEDO   albedo output file name
  -r RLUB     rlub output file name (text file)
  -p          plot RLUB
  -n N        number of standard deviations for maximum
             (default=3.0)
```

2.24 Make Legend

```
usage: makelegend.py [-h] -i INPUT [-u]

Create a legend for ENVI Classification image

optional arguments:
  -h, --help  show this help message and exit
  -i INPUT    input file
  -u          Suppress multiple unclassified legend entries
```

2.25 Mask Noisy Bands

```
Usage: mask_noisy_bands.py -v -p -i input -t threshold

Mask noisy bands in bad band list (BBL) of the header.

Options:
  -h, --help    show this help message and exit
  -i INPUT      input file name
  -v            verbose: print useful info
  -p            plot signal to noise ratio and threshold
  -t THRESHOLD  threshold for bad bands
```

2.26 Merge (Stack)

```
usage: merge.py [-h] [-s] [-b] -o OUTPUT image [image ...]

Merge

positional arguments:
  image       input filenames

options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -o OUTPUT   output image file name
```

2.27 Hyperspectral Median Filter

```
Usage: median.py -f -i input -o output -m {med7, med27}

3D-median filter, 7-neighborhood and 27-neighborhood

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -m MODE     mode: med7 (7-neighborhood, default) or
              med27 (27-neighborhood)
```

2.28 Wavelength of Minimum

```
Usage: minwavelength.py -b -f -i input -o output -m {div|sub|none}
                        -w startwav -W endwav

Determine wavelength of minimum plus other parameters

Options:
  -h, --help  show this help message and exit
  -b
  -f
  -i INPUT
  -o OUTPUT
  -w START
  -W END
  -m MODE
```

2.29 Normalize Bands

```
Usage: normalize.py -s -b -f -i input -o output -a stddevs

Normalize bands.

Options:
  -h, --help    show this help message and exit
  -s            sort bands on wavelength
  -b            use bad band list from the header
  -f            force overwrite on existing output file
  -i INPUT      input file name
  -o OUTPUT     output file name
  -a ADD_STDEV  add standard deviations (default=0.0)
```

2.30 Band Ratio

```
Usage: ratio.py -s -b -f -i input -o output -w numerator -W denominator

Calculate Band Ratio.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -w WAV1     wavelength (or band) of numerator
  -W WAV2     wavelength (or band) of denominator
```

2.31 Band Ratios

```
Usage: ratios.py -s -b -f -i input -o output -d delta

Calculate Band Ratios.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -d DELTA    delta, shift between bands (default=1)
```

2.32 Replace Values

```
usage: replace_values.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT -n NODATA
                         [-v NEWDATA]

Replace nodata values to new value

optional arguments:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -n NODATA   list of nodata values (=string!)
  -v NEWDATA  new output value(s), can be NaN or another image file
```

2.33 Resample Image

```
usage: resample.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT [-p STEPSIZE]

Resampling using step size

optional arguments:
  -h, --help   show this help message and exit
  -s           sort bands on wavelength
  -b           use bad band list from the header
  -f           force overwrite on existing output file
  -i INPUT     input file name
  -o OUTPUT    output file name
  -p STEPSIZE  stepsize (default 2)
```

2.34 Smile Measurement

```
usage: smile.py [-h] [-s] [-b] -i INPUT -o OUTPUT

Determine smile

optional arguments:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -i INPUT    input image file name
  -o OUTPUT   output smile file name
```

2.35 Solar Correction

```
Usage: solar_correction.py -b -f -i input -o output -s solarspec

Solar correction using Solar spectrum.

Options:
  -h, --help    show this help message and exit
  -b            use bad band list from the header
  -f            force overwrite on existing output file
  -i INPUT      input file name
  -o OUTPUT     output file name
  -s SOLARSPEC  input Solar spectrum file name
```

2.36 Sort Bands by Wavelength

```
Usage: sortchannels.py -b -f -i input -o output

Sort bands by wavelength.

Options:
  -h, --help  show this help message and exit
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
```

2.37 Spatial Binning

```
usage: spatialbinning.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT [-x BINSIZE]

Spatial binning using bin size

options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -x BINSIZE  bin size (default 3)
```

2.38 Spatial Spectral Binning

```
usage: spatialspectralbinning.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT
                                 [-x XYBINSIZE] [-z ZBINSIZE]

Spatial spectral binning using xy and z bin sizes
```

```
options:
  -h, --help    show this help message and exit
  -s            sort bands on wavelength
  -b            use bad band list from the header
  -f            force overwrite on existing output file
  -i INPUT      input file name
  -o OUTPUT     output file name
  -x XYBINSIZE  xy bin size (default 3)
  -z ZBINSIZE   z bin size (default 3)
```

2.39 Spectral Binning

```
usage: spectralbinning.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT [-z BINSIZE]

Spectral binning using bin size

options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output file
  -i INPUT    input file name
  -o OUTPUT   output file name
  -z BINSIZE  bin size (default 3)
```

2.40 Spectral Gradient

```
usage: spectralgradient.py [-h] [-b] [-s] [-f] -i INPUT -o OUTPUT

Calculate spectral gradient

optional arguments:
  -h, --help  show this help message and exit
  -b          use bad band list
  -s          sort wavelengths
  -f          force overwrite of output file
  -i INPUT    input file
  -o OUTPUT   ouput file
```

2.41 Spectral Math

```
usage: specmath.py [-h] [-s] [-b] -o OUTPUT [-t DATA_TYPE] -e EXPRESSION
                   image [image ...]

Spectral math

positional arguments:
  image          input filenames

optional arguments:
  -h, --help     show this help message and exit
  -s             sort bands on wavelength
  -b             use bad band list from the header
  -o OUTPUT      output image file name
  -t DATA_TYPE   output data type (double, float32, int32, uint16, ...)
  -e EXPRESSION  Python expression
```

2.42 Split into Separate Bands

```
Usage: split.py -s -b -i input -m {ENVI|JPEG|TIFF|PNG}

Split into separate bands.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -i INPUT    input file name
  -m MODE     output format: ENVI, JPEG (default), TIFF, PNG
```

2.43 Statistics

```
usage: statistics.py [-h] [-b] [-s] [-hist] [-stats] -i INPUT [-band BAND]
                     [-w WAVELENGTH] [-bandname BANDNAME] [-n NUMBINS]

Calculate statistics

optional arguments:
  -h, --help          show this help message and exit
  -b                  use bad band list
  -s                  sort wavelengths
  -hist               calculate histogram
  -stats              calculate statistics
  -i INPUT            input file
  -band BAND          select band
  -w WAVELENGTH       select band by wavelength
  -bandname BANDNAME  select band by bandname
  -n NUMBINS          number of bins to use for histogram (default 256)
```

2.44 Subset Image

```
Usage: subset.py -s -b -f -i input -o output -x x0 -X x1
                 -y y0 -Y y1 -B bandlist -m {bip|bil|bsq}

Make subset
```

```
Options:
  -h, --help   show this help message and exit
  -s           sort bands on wavelength
  -b           use bad band list from the header
  -f           force overwrite on existing output file
  -i INPUT     input file name
  -o OUTPUT    output file name
  -x X0        first sample
  -X X1        last sample
  -y Y0        first line
  -Y Y1        last line
  -B BANDLIST  band list (string!),
               e.g. "<2 4 6-8 >10" for 0 1 2 4 6 7 8 10 11 12...
  -m MODE      format of output file: bip, bil or bsq (default)
```

2.45 Summary Products, Vivian Beck and other indices

Pelkey Summary Products

```
Usage: summary_products.py -s -b -f -i input -c continuumremoved
                           -o output -u {mic|nan} -l

Summary Products, mineral and other indices.

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -f          force overwrite on existing output files
  -i INPUT    input file name
  -c CR       input continuum removed file name
  -o OUTPUT   output file name
  -u UNITS    input wavelength units: mic (default, micrometers)
              or nan (nanometers)
  -l          create a logfile
```

Viviano-Beck Summary Products

```
usage: viviano_beck.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT [-u {mic,nan}]
                       [-l]

Summary Products, mineral and other indices.

optional arguments:
  -h, --help    show this help message and exit
  -s            sort bands on wavelength
  -b            use bad band list from the header
  -f            force overwrite on existing output files
  -i INPUT      input file name
  -o OUTPUT     output file name
  -u {mic,nan}  input wavelength units: mic (default, micrometers) or nan
                (nanometers)
  -l            create a logfile
```

Other Indices

```
usage: otherindices.py [-h] [-s] [-b] [-f] -i INPUT -o OUTPUT [-u {mic,nan}]
                       [-l]

Summary Products, mineral and other indices.

optional arguments:
  -h, --help    show this help message and exit
  -s            sort bands on wavelength
  -b            use bad band list from the header
  -f            force overwrite on existing output files
  -i INPUT      input file name
  -o OUTPUT     output file name
  -u {mic,nan}  input wavelength units: mic (default, micrometers) or nan
                (nanometers)
  -l            create a logfile
```

2.46 Thermal Correction

```
Usage: thermal_correction.py -b -f -i input -o output -t thermal

Thermal correction.

Options:
  -h, --help  show this help message and exit
  -b          use bad band list from the header
  -f          force overwrite on existing output files
  -i INPUT    input file name
  -o OUTPUT   output file name
  -t THERMAL  output thermal file name
```

2.47 Convert to PNG, PGW and KML

```
Usage: tokml.py -s -b -e -z -i input -R red -G green -B blue
                -m {NO|MM|1P|SD}

Convert data to .png, .pgw and .kml in one go

Options:
  -h, --help  show this help message and exit
  -s          sort bands on wavelength
  -b          use bad band list from the header
  -i INPUT    input file name
  -e          strip edges
  -z          strip zeros
  -R RED      red band, wavelength or band number
  -G GREEN    green band
  -B BLUE     blue band
  -m MODE     stretch mode: NO (none), MM (min-max),
              1P (1 percent, default), SD (2 standard deviation)
```

2.48 Get Transmittance

See Get Transmittance.

2.49 Wavelength Mapping

```
Usage: wavemap.py -f -i input -o output -w startwav -W endwav
                  -d startdepth -D enddepth
                  -c {rainbow|steps} -l

Make wavelength/depth map using color table and HSV transform

Options:
  -h, --help     show this help message and exit
  -f             force overwrite on existing output file
  -i INPUT       input file name
  -o OUTPUT      output file name
  -w WSTART      lower wavelength
  -W WEND        upper wavelength
  -d DSTART      lower depth value
  -D DEND        upper depth value
  -c COLORTABLE  color table: rainbow (default) or steps
  -l             save legend in a .png file
```

2.50 WMS Get

```
Usage: wmsget.py -u url -l layer -s layerstyle -p reference
                 -x centerx -y centery -r resolution
                 -i imageformat -a width -b height -o output -e

Get image via Web Map Server (WMS).

Options:
  -h, --help     show this help message and exit
  -u URL         URL of the WMS
  -l LAYER       selected layer
  -s STYLE       selected layer style
  -p SRS         selected spatial reference,
                 for instance "EPSG:28992"
  -x CENTERX     selected center x coordinate
  -y CENTERY     selected center y coordinate
  -r RESOLUTION  required resolution
  -i IMFORMAT    output image format, typically "image/jpeg" etc.
  -a WIDTH       output width in pixels
  -b HEIGHT      output height in pixels
  -o OUTPUT      input file name
  -e             flag for also generating ENVI format image
```

2.51 Zonal Statistics

```
usage: zonalstatistics.py [-h] -z ZONES -i INPUT [-b] [-s]
                          [-m {max,mean,median,min,std,sum,m-2s,m+2s}]

Calculate zonal statistics (as tables)

optional arguments:
  -h, --help            show this help message and exit
  -z ZONES              zones file
  -i INPUT              input file
  -b                    use bad band list (on input)
  -s                    sort wavelengths (on input)
  -m {max,mean,median,min,std,sum,m-2s,m+2s}
                        method (default 'mean')
```