

User Manual for Spectronon Version 3.5.8

Resonon Inc.

Mar 12, 2025

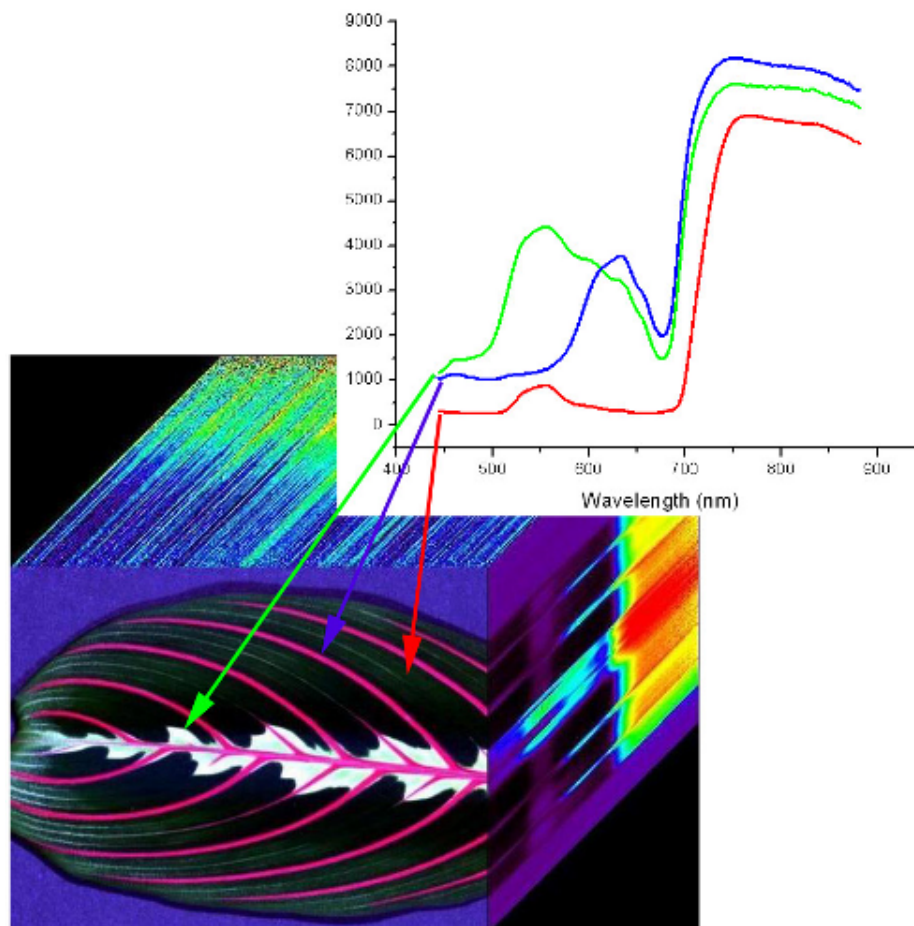
CONTENTS

1	Introduction to Hyperspectral Imaging	1
2	Software Installation	4
3	Basic Data Acquisition	6
3.1	Data Modes and Sources	6
3.2	Start The System	7
3.3	Verifying Imager Calibration	9
3.4	Imager Controls	10
3.5	Stage Controls	11
3.6	Focusing Objective Lens	13
3.7	Reflectance Measurement Calibration	15
3.8	Scanning and Saving Datacubes	17
4	Advanced Data Acquisition	20
4.1	Imager Controls	21
4.2	Stage Controls	27
4.3	Additional Controls	29
5	Basic Data Analysis	34
5.1	Spectronon Tools	34
5.2	Zoom, Pan, Flip, and Rotate Tool	35
5.3	The Inspector Tool – Spectral Plots	36
5.4	Region Of Interest (ROI) Tools	37
5.5	Image Saturation	42
5.6	Image Visualization	42
5.7	Plot Panel	47
5.8	Saving Spectra, Plots, and Images	48
6	Advanced Data Analysis 1: General	50
6.1	File Menu	50
6.2	Datacube Menu	51
6.3	Image Menu	59
6.4	Spectrum Menu	59
6.5	Selection Menu	61
6.6	Plots Menu	62
6.7	Datacubes and Header Files	63
7	Advanced Data Analysis 2: Hyperspectral Classification	65
7.1	General Approach	65
7.2	Spectral Angle Mapper (SAM) Classification	70

8	Advanced Data Analysis 3: Hyperspectral Vegetation Indices	75
8.1	Introduction	75
8.2	Generating HVI maps in Spectronon	78
9	Batch Processing	82
10	Focusing & Calibration Sheets	85
10.1	Small Focusing Sheet	85
10.2	Large Focusing Sheet	87
10.3	Aspect Ratio Calibration Sheet	88
11	Appendix: Built-in Plugins	89
11.1	Cube Plugins	89
11.2	Render Plugins	129
11.3	Filter Plugins	132
11.4	Select Plugins	135
12	Glossary of Hyperspectral Imaging Terminology	139
13	Recalibration Services	143
14	Product Support	144
15	Copyright Notice	145

INTRODUCTION TO HYPERSPECTRAL IMAGING

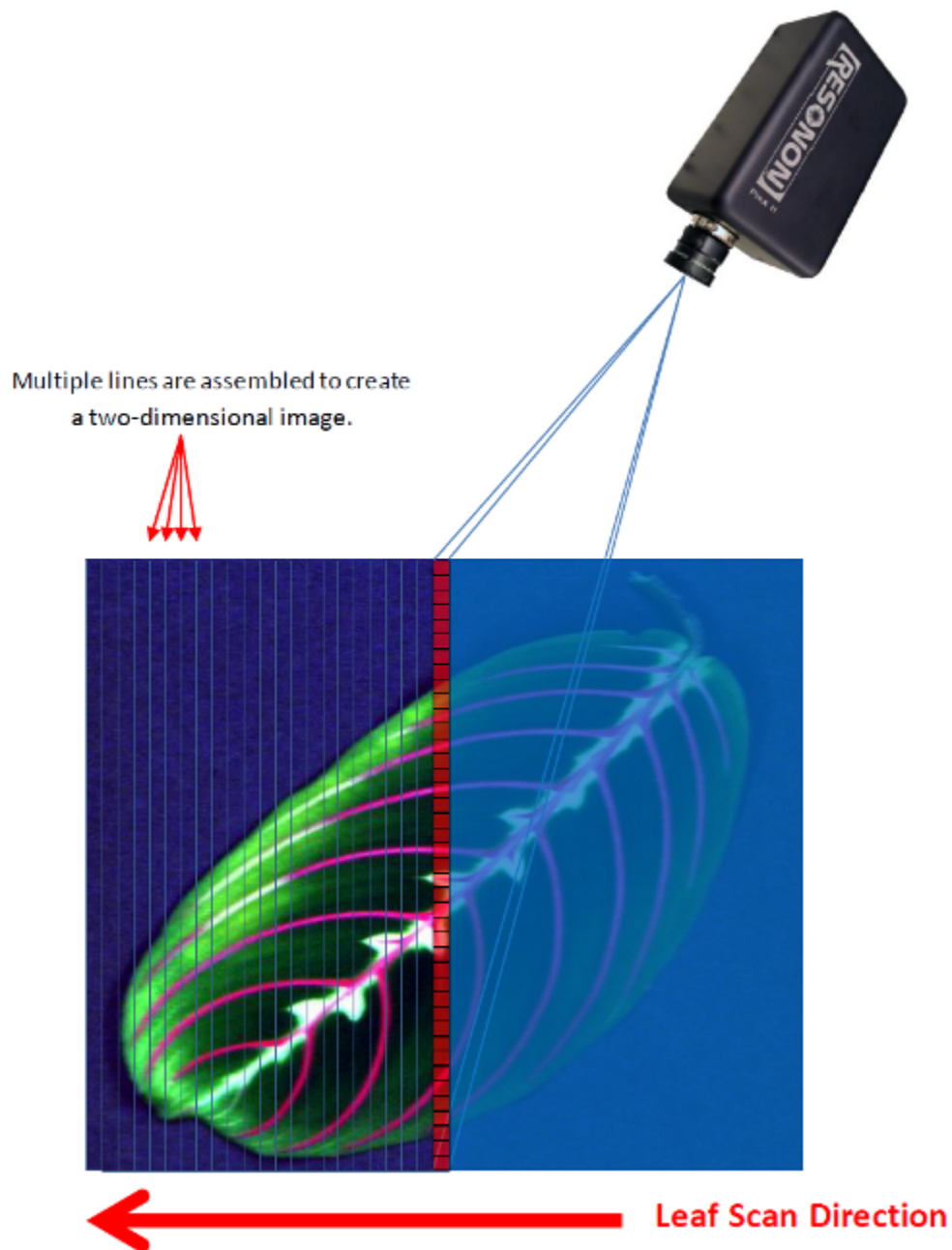
Hyperspectral Imaging, or imaging spectroscopy, refers to the creation of a digital image containing very high spectral (color) resolution. Each spatial point (pixel) in a hyperspectral image represents a continuous curve of incoming light intensity versus wavelength. For example, red, green, and blue arrows below show the spectra for three pixels of an image of a leaf in the image below. The data can also be interpreted as a stack of images, with each layer in the stack representing the scene at a different wavelength – this “stack” of two-dimensional images is referred to as a “datacube.”



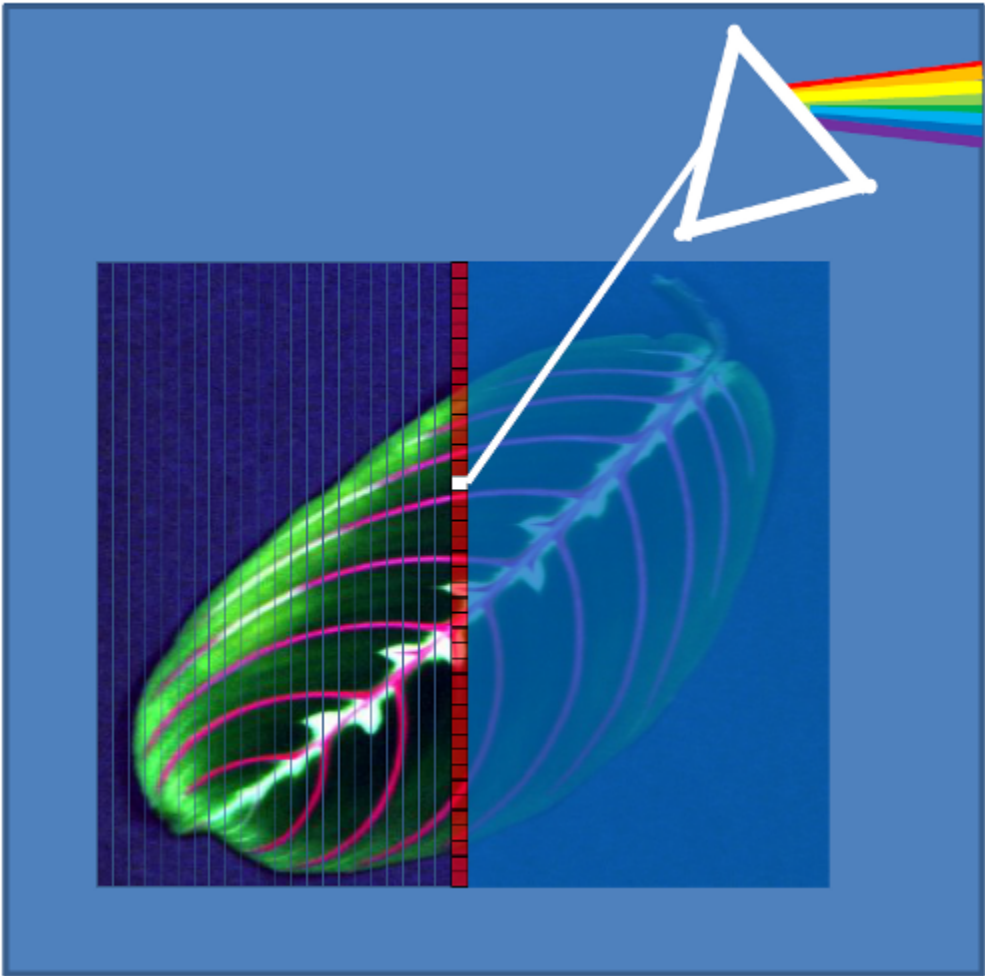
The benefit of the additional information provided by hyperspectral imaging is that it generally allows you to tell the difference between materials better than you can with traditional color images. This capability can be applied to a wide range of current and emerging applications in remote sensing, sorting, quality control, research and development, and more.

Resonon imaging spectrometers are **line-scan** imagers, which means they collect data one line at a time. To assemble

a complete two-dimensional image, multiple lines are imaged as the object (or imager) is translated. The multiple line-images are then assembled line-by-line to form a complete image, as indicated below. A scanning system is often needed to use a Resonon imaging spectrometer.



To obtain hyperspectral data, the signal from each pixel is dispersed (or diffracted) into its spectral components, much like passing the light from each pixel through a prism. This process occurs for every pixel in the line (red squares in the image below). The dispersed signals from all the pixels are imaged onto a focal plane array. One of the benefits of this approach is that all the spectral (color) information is collected at the same time for each pixel. The result is a detailed spectral curve for every pixel in the image.



SOFTWARE INSTALLATION

Spectronon is easily installed using the provided installer. It can be installed even if a previous version is installed, in which case the previous version will be uninstalled before the current version is installed. If you are installing Spectronon for the first time, or upgrading your current installation, the default options should be used. *Ensure that any existing versions of Spectronon (or SpectrononPro, if you are using an older, paid only version of the software) are closed before installing a new version.*

Note: The last version of SpectrononPro is 3.4.11. All subsequent versions of Spectronon, starting with v3.5.0, bundle the ability to collect hyperspectral data using Resonon hyperspectral cameras and systems into a single, free version of the software.

The latest version of the software can also be found at our [download site](#), under *Spectronon*. Resonon provides Spectronon for free, along with sample hyperspectral data, to help develop the hyperspectral imaging community. We believe that open access to hyperspectral analysis tools will contribute to developing hyperspectral imaging as an essential tool across science and engineering disciplines.

The Spectronon installer can also be used to repair broken driver configurations. Run the installer as normal and when you get to the *Choose Components* window, select only the drivers you want to install. You will have to restart your computer if you select the *Pika IR/IRL Driver* option.

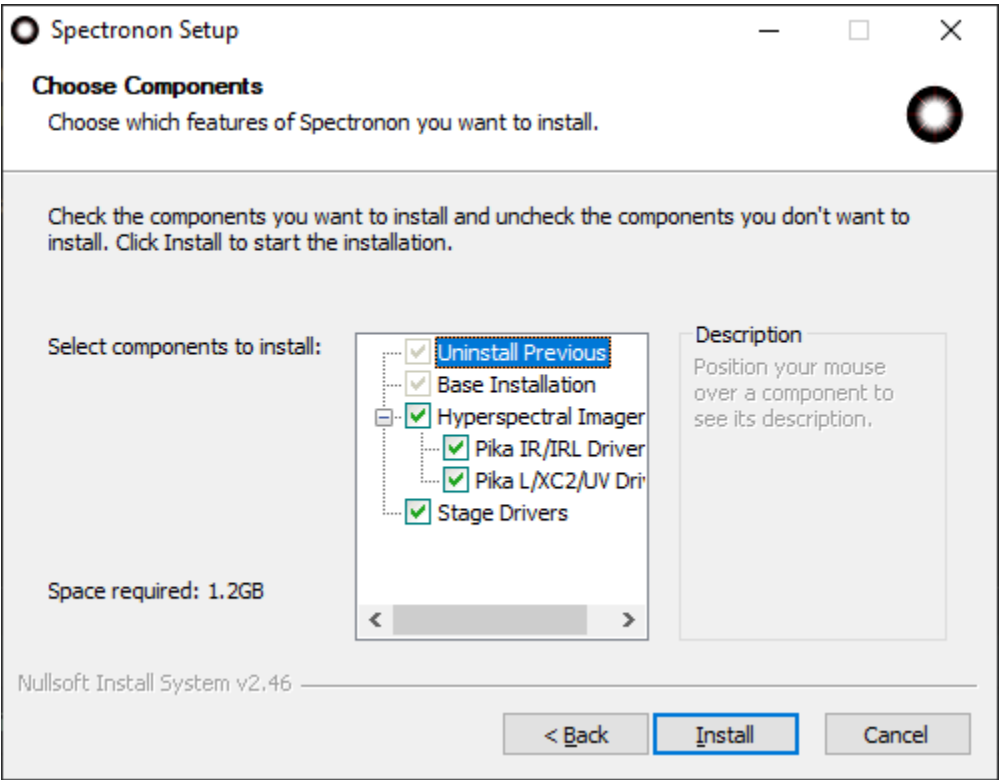


Fig. 1: Choose Components window of the Spectronon installer

BASIC DATA ACQUISITION

3.1 Data Modes and Sources

It is important to consider the **data mode** of the various hyperspectral imaging systems that produce hyperspectral **datacubes**. Hyperspectral data from Resonon imaging systems can be utilized in three forms, as summarized below. Which modes are used is application dependent, and affects subsequent datacube analyses.

3.1.1 Raw Data

These data are spectrally calibrated but are not corrected for the instrument-sensor-response or illumination functions, so the result cannot be directly interpreted as, for example, reflectivity. The units are digital number (DN) from the imager camera's sensor array. If a dark frame has been recorded, these data may have the camera's average dark current subtracted, but no other correction are applied. This is the least useful data form, as the spectral curves do not have physical units. **Spectronon** can collect datacubes in this mode with either a Benchtop System or Outdoor Field System.

3.1.2 Radiance

Raw data can be post-processed to give radiance data, where the DN unit is converted to physical units of microflicks (1 microflick = 1 microwatt per steradian per square centimeter of surface per micrometer of span in wavelength). This mode removes the imager's instrument-sensor-response function from the data. This function is corrected for by using the Imager Calibration Pack (ICP file) with the *Radiance From Raw Data* plugin. The resulting data are the product of illumination and sample-reflectivity functions. In addition to the standard spectral calibration, radiance measurements require Resonon to perform a radiometric calibration (rad cal) on the imager with the desired objective-lens aperture. Radiance conversions are typically performed with **Spectronon** on datacubes collected under stable, uniform lighting conditions, such as with Resonon's Outdoor Field System or Airborne Remote Sensing System.

3.1.3 Reflectivity/Reflectance

In reflectance mode, both the instrument-sensor-response and illumination functions are removed. This leaves the data in absolute reflectance. Resonon's Benchtop System commonly performs reflectance measurements on a sample by applying a response correction that employs a separate scan of a reflectance standard that corrects for the instrument-sensor-response and illumination functions. The reflectance standard, or calibration tile (cal tile), fills the entire field of view (FOV) of the imager. Spectralon® and Fluorilon® are the highest quality reflection standards, but Teflon is acceptable for many applications. (Teflon needs to be sanded with 100 grit sandpaper on an orbital sander to eliminate any specular properties).

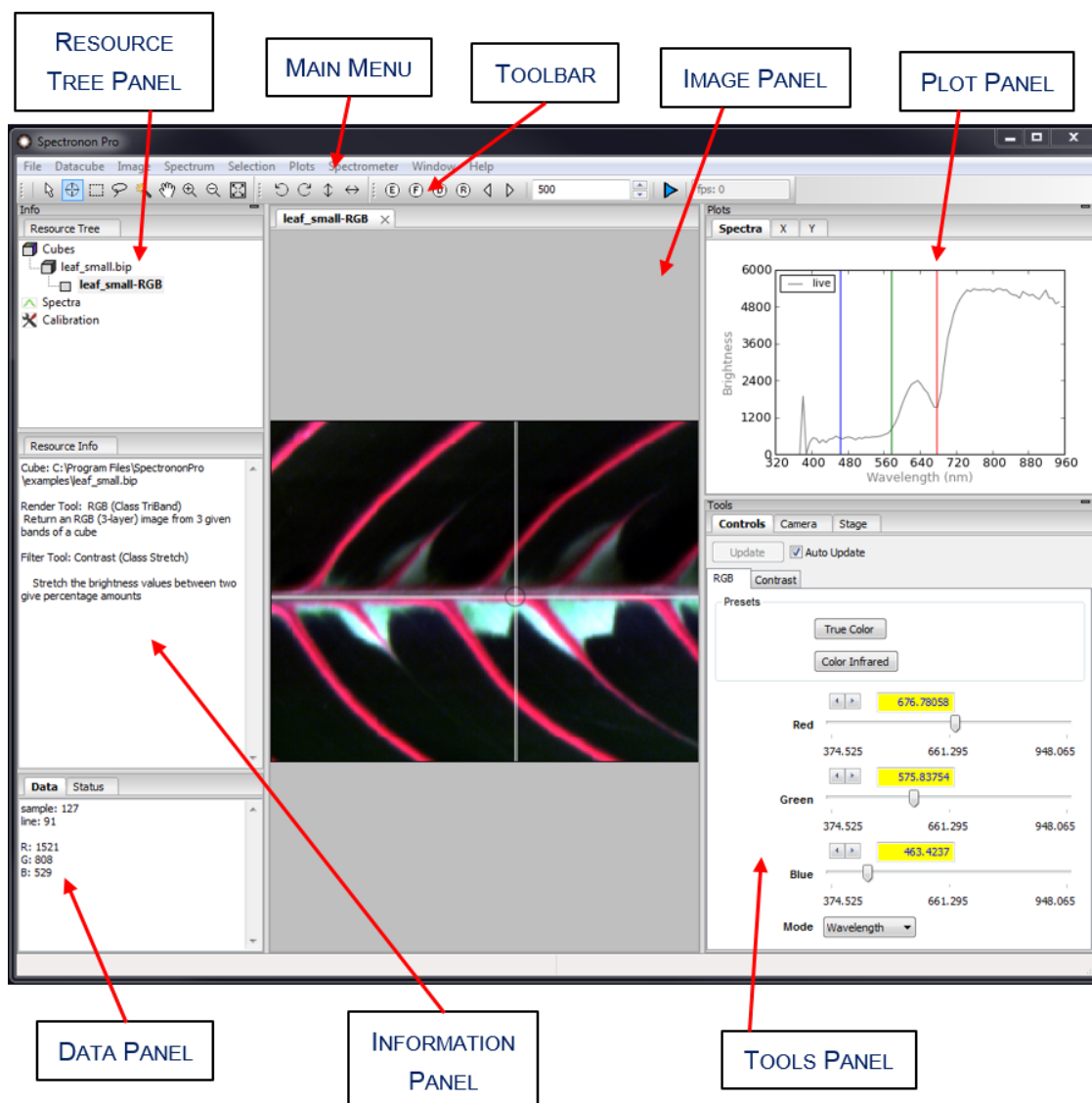
In other system setups, such as with Resonon's Outdoor Field System or Airborne Remote Sensing System, data can be converted to reflectance in one of four ways described below.

1. **White reference:** Data can be processed to reflectance with a calibration measurement against a reflectance standard, such as used with the Benchtop System. This calibration is done with the *Record Correction Cube* feature, as described in [Section 3.7.2: Response Correction](#). It is important to note that reflectance values are only accurate if the solar illumination (cloud, sun angle, etc.) does not change between the collection of the response-correction datacube and the sample datacube. Data can also be converted to reflectance using the *Reflectance from Raw Data and Spectrally Flat Reference Cube* plugin.
2. **Known spectral reference in scene:** After the data have been converted to radiance, the known spectrum of a reference object in the scene can be used to correct the rest of the cube to reflectance. The reference spectrum must be known and in a tab- or space-delimited file. Use the *Reflectance from Radiance Data and Spectrally Flat Reference Spectrum* plugin.
3. **Downwelling irradiance sensor:** An alternative method for converting data to reflectance is to use a downwelling irradiance sensor. This sensor records the solar spectrum during data acquisition. This data is used, along with the Imager Calibration Pack (ICP) file(s) supplied by Resonon for both the spectral imager and the downwelling sensor. This method uses the *Reflectance from Radiance Data and Downwelling Irradiance Spectrum* plugin.
4. **Atmospheric Correction:** Data can be converted to reflectance with the use of atmospheric correction algorithms such as FLAASH (Fast Line of Sight Atmospheric Analysis of Spectral Hypercubes). Please contact Resonon for more information.

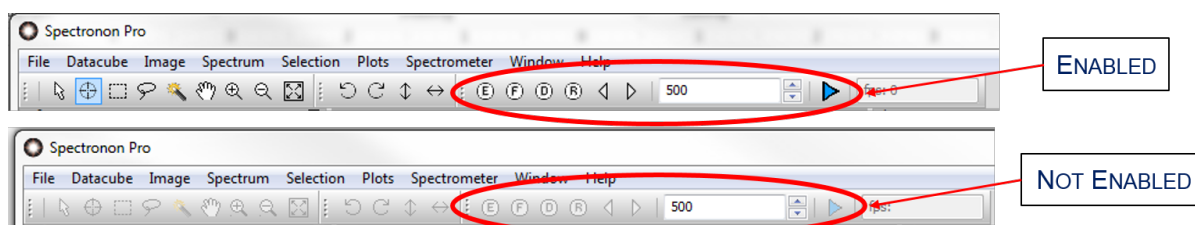
3.2 Start The System

If you have an illumination system, turn it on and let it warm up sufficiently. Some lights may require 15 to 20 minutes to fully stabilize.

With the imager and (optional) stage connected to your computer, launch **Spectronon** by double-clicking on the **Spectronon** icon or starting **Spectronon** from your Start menu. The **Spectronon** icon and user interface are shown below with the various windows labeled.



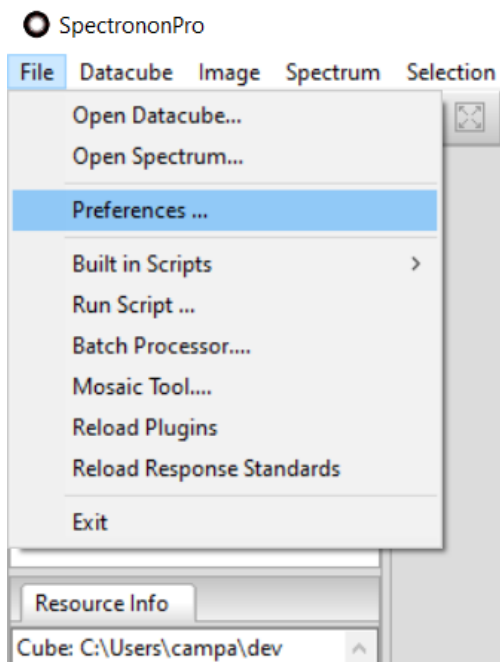
Once the software has started, make sure that the imager and stage controls (if used) are enabled, which indicates that they are properly connected. The imager and stage tools will be greyed out if not enabled, as shown below.



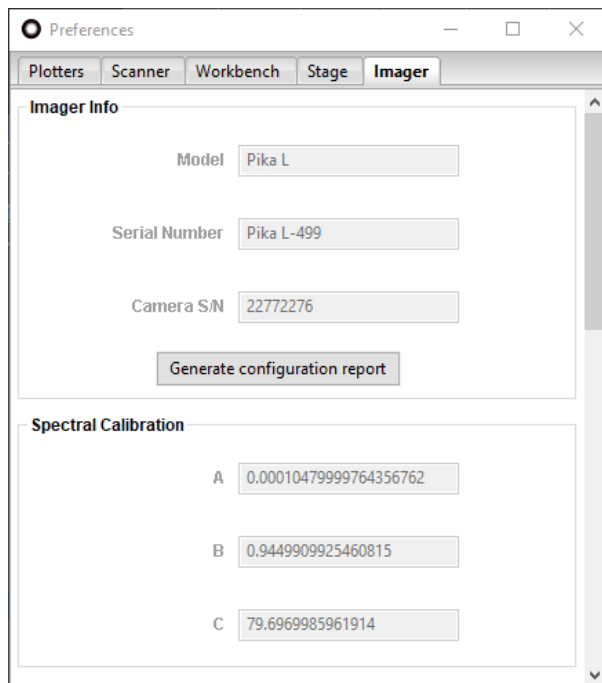
You can get the latest version of **Spectronon** by clicking on *Help* → *Check For Updates*. This won't download the latest version, but will give an alert if there is a newer version, as well as a hyperlink to it.

3.3 Verifying Imager Calibration

From the Main menu, select *File* → *Preferences...*



This will reveal the *Preferences* window. This window has several tabbed panes that provide detailed information about the connected *Imager* and *Stage*, as well as settings for the *Scanner*, managing and analyzing datacubes on the *Workbench*, and visualizing data with *Plotters*.



In the *Preferences* window, select the *Imager* tab to reveal detailed information about your connected hyperspectral imager. Your imager was calibrated at the factory. The spectral calibration numbers are provided on a sheet that comes with your imager and should be verified prior to use. Check the *A*, *B*, and *C* values from the sheet provided against the *Spectral Calibration* values appearing in **Spectronon**. If you cannot locate your calibration sheet or these numbers differ from those reported in the software, then please contact Resonon at: ProductSupport@resonon.com.

3.4 Imager Controls

Camera Settings for the imager can be controlled by clicking on the *Imager* tab in the *Tools* panel in the Main window.

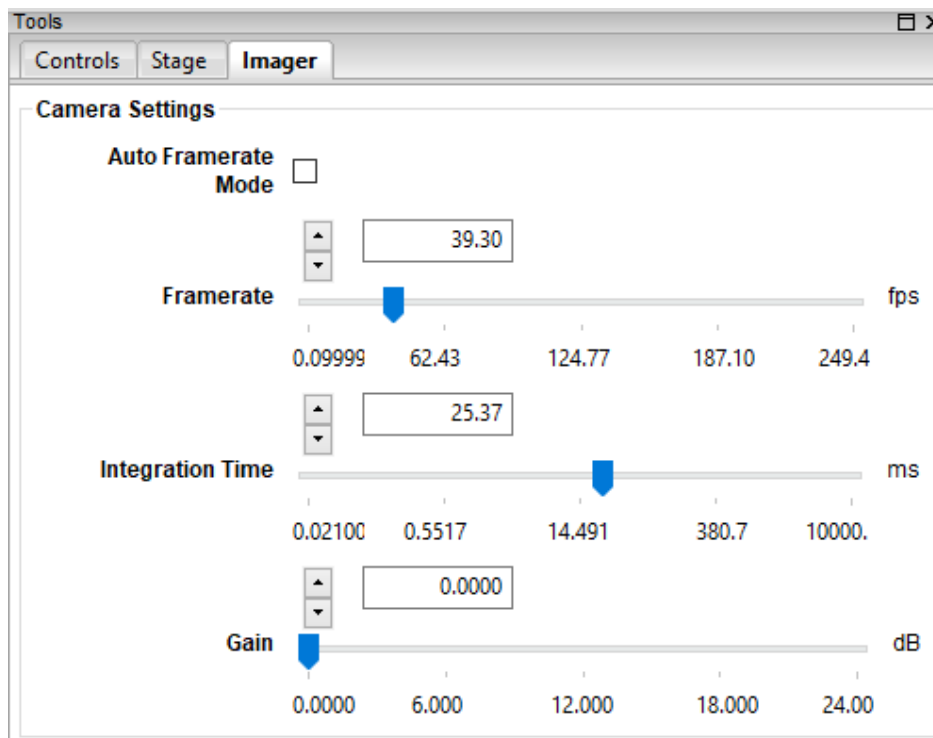


Fig. 1: Imager Camera Settings with Auto Framerate Mode off.

Framerate is equal to the number of datacube lines acquired each second, or frames per second (fps). The *Framerate* limits the maximum *Integration Time*, where typically $Integration\ Time \leq 1 / Framerate$.

Integration Time is the duration of data acquisition for each individual line (also known as exposure time), displayed in milliseconds (ms). Larger integration times require smaller frame rates, and smaller integration times allow faster frame rates. Integration times that are too short have poor signal-to-noise ratio. Integration times that are too long saturate pixels in the detector.

Gain is a factor that increases the signal, but at the expense of signal-to-noise ratio. Keep the gain as low as possible, preferably zero. A gain of 6 dB roughly doubles the signal. (The IR imagers do not have this setting.)

Some of Resonon's imagers support *Auto Framerate Mode*. When enabled, the *Framerate* cannot be set directly. (However, the frame rate can still be viewed.) The user need only select an appropriate *Integration Time* (and *Gain*, possibly) based on the illumination and sample brightness, and **Spectronon** automatically chooses the fastest *Framerate* for that *Integration Time*. This helps produce the fastest possible scans for a given setup.

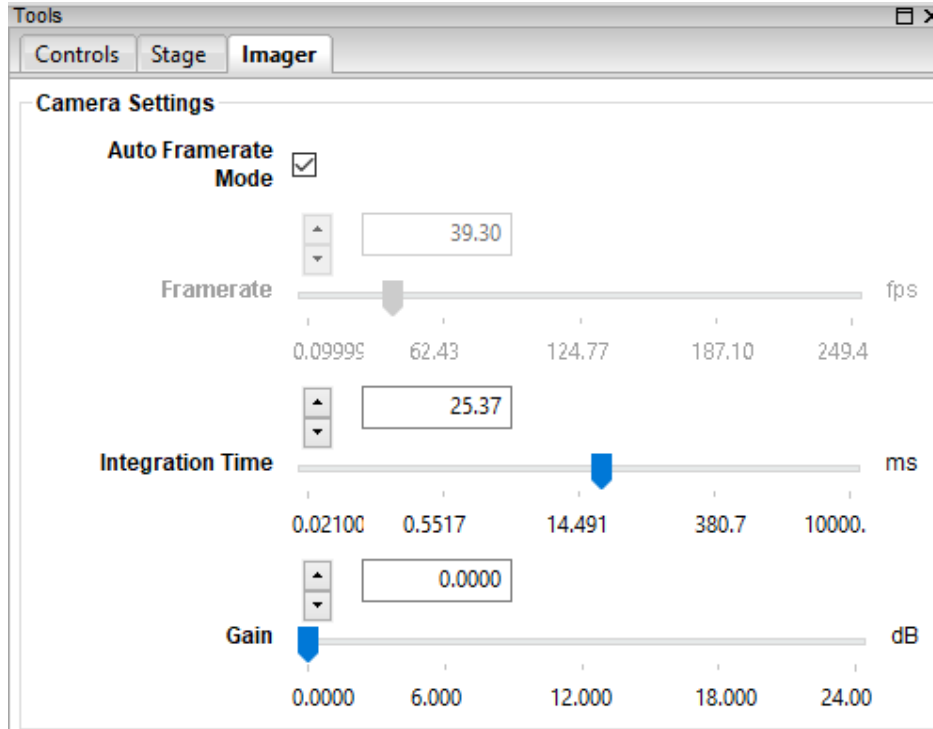
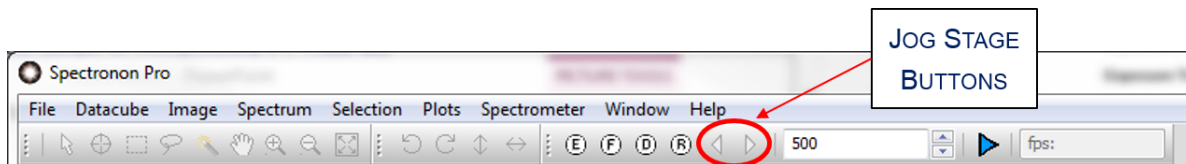


Fig. 2: Imager Camera Settings with Auto Framerate Mode on.

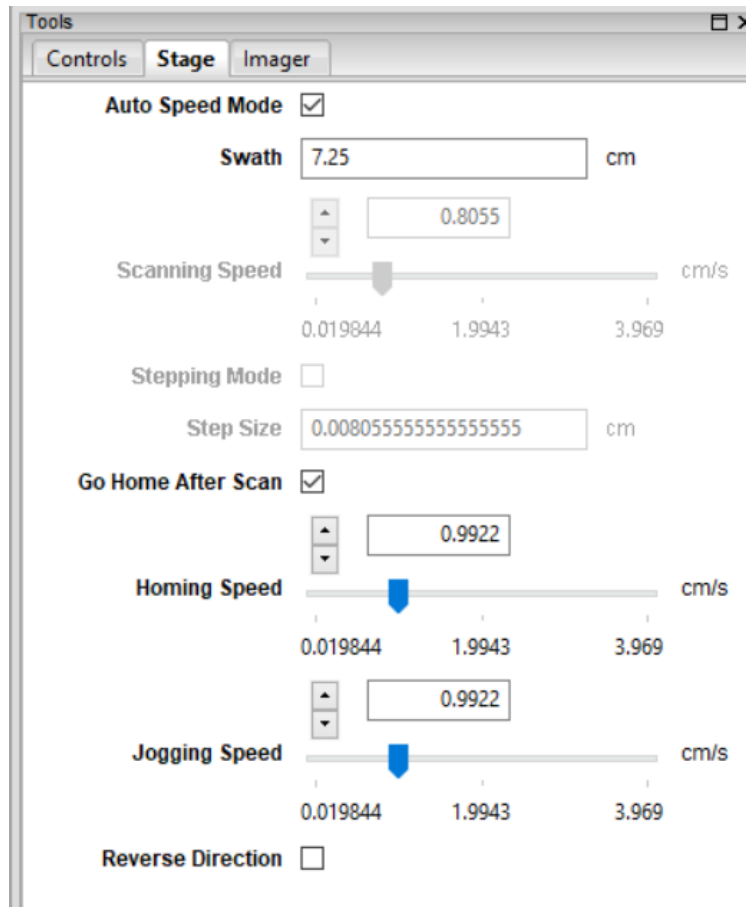
Additional *Imager* settings are available in *Preferences*→*Imager*, described in [Section 4: Advanced Data Acquisition](#).

3.5 Stage Controls

The stage is configured at the factory for either the Benchtop System (cm units) or the Outdoor Field System (deg units). You can move the stage manually by clicking on the *Jog Stage* buttons, located on the toolbar. The buttons move the stage incrementally in either direction. Use the buttons to center the stage underneath the Pika imaging spectrometer.



The stage can be further controlled by clicking on the *Stage* tab in the *Tools* panel. The default stage setting in **Spectronon** is *Auto Speed Mode*. This means that the *Framerate* of the imager's camera is used to automatically calculate the stage's *Scanning Speed*. Because the imager is a line-scan imager, the stage must traverse at a speed proportional to the camera's framerate in order to capture the scene with a unit aspect ratio. If the stage is too slow with respect to the *Framerate*, then the image is elongated in the direction of stage travel. If the stage is too fast with respect to the *Framerate*, then the image is shortened.




For the Benchtop System, the *Swath* value is key to achieving unit aspect ratio scans when using *Auto Speed Mode*. The swath is the width of the stage that is viewed by the imager, and is affected by the choice of **objective lens** and the **working distance** from the (in-focus) object on the stage to the imager. A procedure for determining a value for *Swath* is provided later in [Section 3.8: Scanning and Saving Datacubes](#). In the Outdoor Field System, this *Swath* is replaced by the angular *FOV* (field of view), which is typically the far-field value specified for the given objective lens.

With *Auto Speed Mode* selected, several controls are disabled and serve only as value indicators. The *Scanning Speed* indicates the continuous scanning speed, and is set automatically. If the stage cannot move sufficiently slowly in a continuous manner, then *Stepping Mode* will be enabled. In *Stepping mode* the datacube is acquired line-by-line with the stage coming to a stop for each line, i.e., stepping. (Stepping occurs for a sufficiently long *Integration Time* corresponding to a sufficiently low *Framerate*.) The *Step Size* parameter indicates how much each line in the scan covers (continuous or stepping).

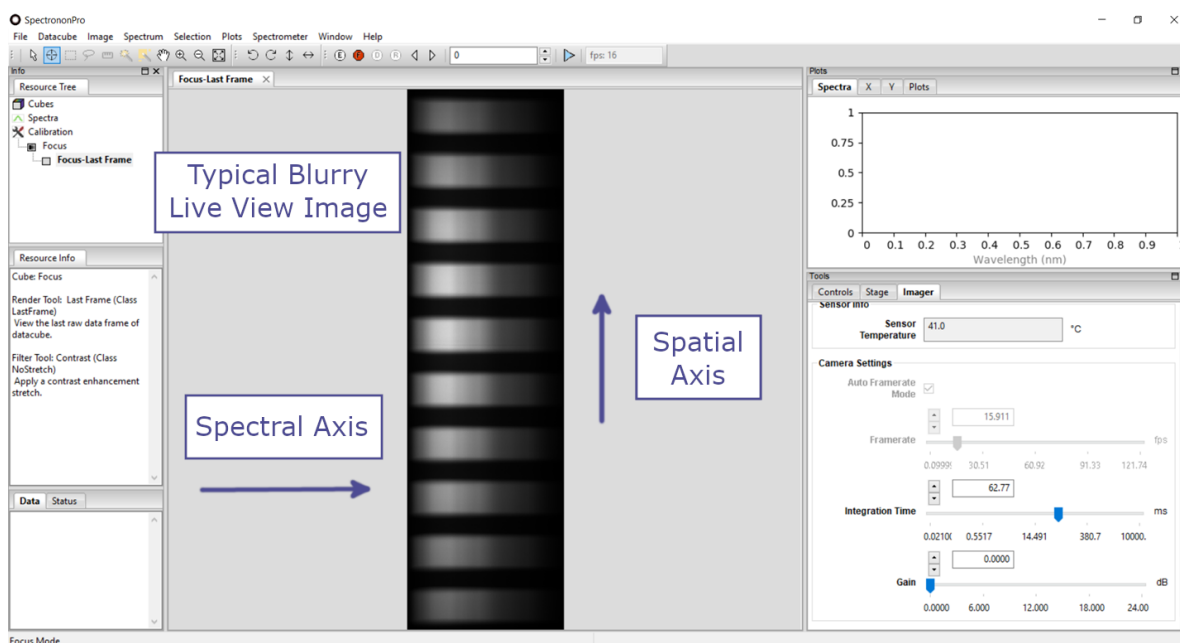
If *Go Home After Scan* is selected, then the stage will return to its starting position after each scan. The speed at which the stage returns to its original position is the *Homing Speed*. (The *Homing Speed* setting is also used to set the between-steps *Scanning Speed* when the stage is automatically put into *Stepping Mode*.) *Jogging Speed* is used for the *Jog Stage* buttons, described above. *Reverse Direction* reverses the scan direction of the stage, including the jog directions.

Additional *Stage* settings are available in *Preferences*→*Stage*, described in [Section 4: Advanced Data Acquisition](#).

3.6 Focusing Objective Lens

You are now ready to **focus the objective lens** of your Pika imaging spectrometer. At first, this process can be somewhat challenging, but with a little practice it becomes straightforward. Begin by clicking on the *Focus* button  located on the **Spectronon** tool bar. This will reveal a live image of individual frames from the imager's camera. (Wave your hand in the **field of view** (FOV) of the imager to confirm that the image is a live view.) One axis of this image represents the spatial (position) axis of your object, and the other is the spectral (wavelength) axis. (To understand this view better, move colored objects within the FOV of your imager after you have focused the objective lens.)

Place an object with multiple light and dark regions within your Pika imaging spectrometer's FOV. For the Benchtop System, make sure that the illumination is adequate and sufficiently uniform, and place a sheet of paper with dark lines on the stage, as provided in [Section 10: Focusing & Calibration Sheets](#). For the Outdoor Field System, if you are in the field and are observing objects at a distance, then direct the imager towards an object with multiple features, such as a tree with many branches. Unless your lens is already focused, you will see a series of blurry or barely discernible lines in the live-view image in the center panel of **Spectronon**.

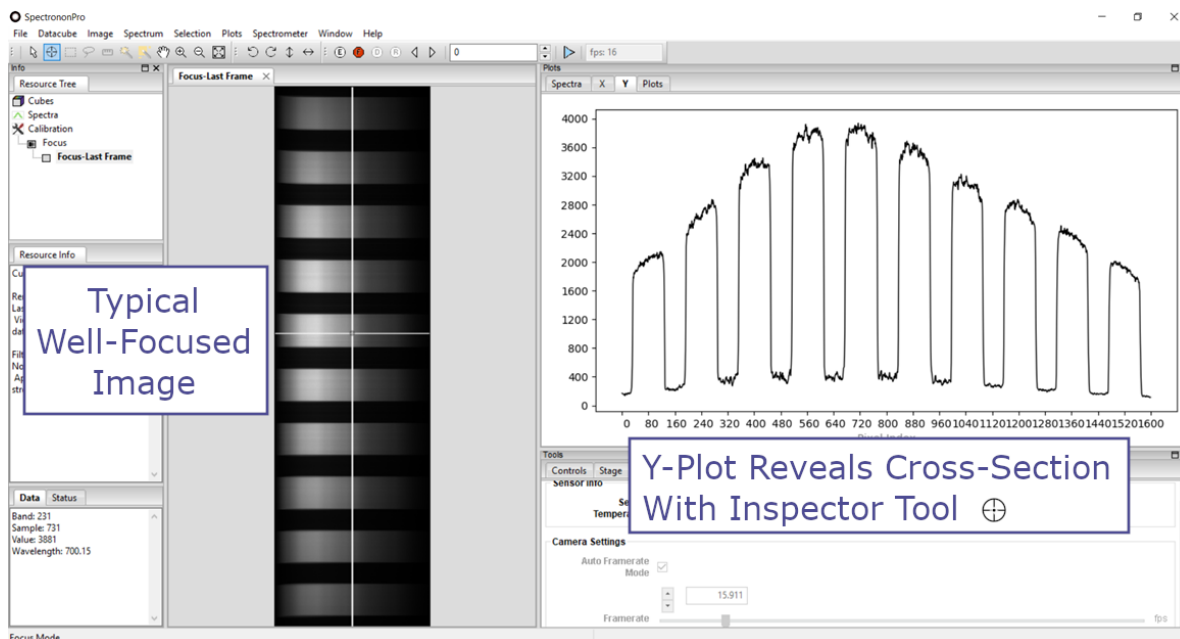


To adjust the focus, first unlock the focus adjustment. With Schneider lenses, this is done by loosening the locking metal collar on your objective lens using an Allen wrench, size 5/64 inch.



LOOSEN LOCKING COLLAR TO ENABLE FOCUSING


Then rotate the objective lens until you see dark lines from your object come into focus, as shown. Maximize the sharpness of the lines.



Hint: Clicking the *Inspector Tool*  on the live view image, and then selecting *X* tab in the *Plots* window will reveal a cross-section plot of your image. Viewing this plot allows you to graphically see the sharpness of your focusing. You can zoom in on the X-axis by clicking the *Zoom Tool*  and then clicking on the X-axis.

Note: In the Benchtop System, the **working distance** is often changed to fill the imager's FOV with the sample being scanned on the stage. If you change the working distance, then you will need to refocus the imager and update the *Swath* value under the *Stage* tab in the *Tools* panel. A good first estimate of the value for *Swath* can usually be made from the ruled lines of the in-focus focusing sheet.




Focusing the Outdoor Field System can be a little more challenging than the benchtop system. If you are focusing on objects that are further than 40 feet start the process with the objective lens screwed in close to the collar. A method that has proven useful is to start the focusing process by increasing the number of lines scanned to 1000. This gives a large scan area to begin the process. You will need to be out of live focus mode when performing this focusing procedure ('F' button should not be red). When the scan begins make a quarter turn with the lens. Continue to make quarter turns until you are confident your scene is in focus. When making the quarter turn, intentionally place your hand in front of the lens. This will create a thin black line in the scan separating one focus length from its neighbor, allowing for easier comparison. This process can take some time, so be patient and remember that it gets easier with practice.

Once you have completed focusing, re-tighten the lock to the focus adjustment. Then click on the *Focus*  tool again to toggle the camera live view off.


Note: See our YouTube video on focusing the benchtop system at <https://www.youtube.com/@resonon/videos>.

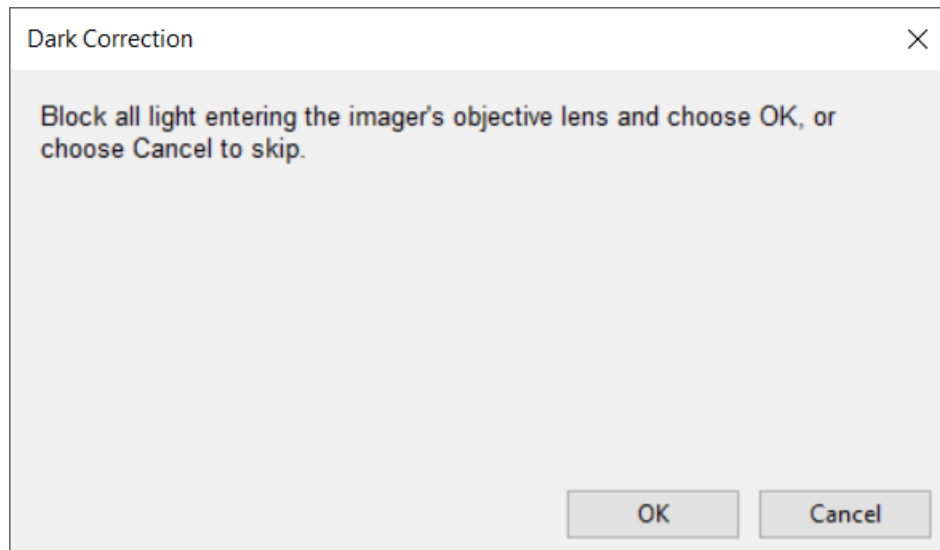
3.7 Reflectance Measurement Calibration


The following discussion describes how to set up your system to scan for **reflectance** scaled to a reference object. If you wish to collect raw data and convert it to radiance do not perform the following correction process.

Note: The **Dark Current** button  and the *Response Correction Cube* button  are disabled in live camera view mode. If needed, then click on the *Focus*  tool to toggle the camera live view off.


3.7.1 Dark Correction

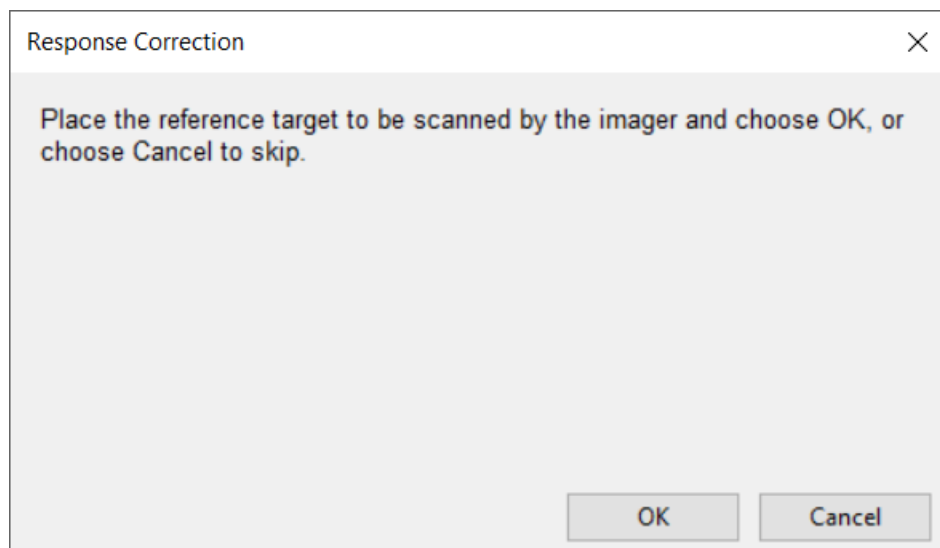
Spectronon can remove the imager camera's average dark current from scans. Begin by clicking on the *Dark Correction* button  on the **Spectronon** toolbar. You will be instructed to block all light entering the imager's objective lens.




After you have the objective lens blocked, click *OK* as instructed. **Spectronon** will then collect multiple dark frames and use these measurements to subtract the average dark current from your measurements. The *Dark Correction* button on the toolbar will appear with a red check through it as soon as the dark frames have been collected . Once you see the red check, unblock the objective lens.

3.7.2 Response Correction

Measuring **absolute reflectance** of an object requires correction to account for instrument-sensor-response and spatially nonuniform illumination effects. To do this, click on the *Response Correction* button  on the **Spectronon** toolbar. A message will appear telling you to place a reference material within your Pika imaging spectrometer's FOV. The reference material should be uniform across the imager's FOV. Examples of reference materials include Spectralon®, Fluorilon®, or sheets of white Teflon®.



Once the reference material is in place, click on *OK*. This will trigger a short scan of the reference material. Once complete, the *Response Correction* button will appear with a red check mark , indicating that the data you collect will be scaled in reflectance with respect to the reference material, including flat-fielding to compensate for spatial variations in your lighting. Any dark correction is subtracted from the response correction.


Note: For additional help with this process see our Calibration video at <https://www.youtube.com/@resonon/videos>.

3.7.3 Invalidating Corrections

Once the imager is calibrated for both dark current and reflectance reference, the imager will remain calibrated until the corrections are removed by the user, or **Spectronon** is restarted. To manually remove the corrections, click *Spectrometer* → *Remove Dark Correction Cube* and *Spectrometer* → *Remove Response Correction Cube*.

Dark current typically changes with the imager camera's integration time and gain settings, the degree of which depends on the imager camera's particular sensor. Furthermore, the response correction is likewise invalidated by changes in the imager camera's integration time. By default, **Spectronon** will warn you before taking a scan if such changes have likely invalidated one/both of the corrections. **Spectronon** does **not** detect changes such as illumination and working-distance adjustments, which also invalidate the response correction.

Hint: Before collecting the dark and response correction cubes, you may first want to set the imager camera's *Framerate* and *Integration Time* while in live view and viewing the reflectance reference. Increase the *Integration Time* until the signal is near, but sufficiently below, saturation over the entire frame (spatially and spectrally). Next, maximize the scan speed by choosing the fastest *Framerate* for that *Integration Time*. (If available, then *Auto Framerate Mode* can be turned on to have **Spectronon** automatically set the fastest frame rate.) Also consider the auto expose feature

accessible as the *Auto Expose* button in the toolbar  in the *Scan* toolbar and described further in [Section 4.1: Imager Controls](#).

3.8 Scanning and Saving Datacubes


3.8.1 Scanning with Unity Aspect Ratio

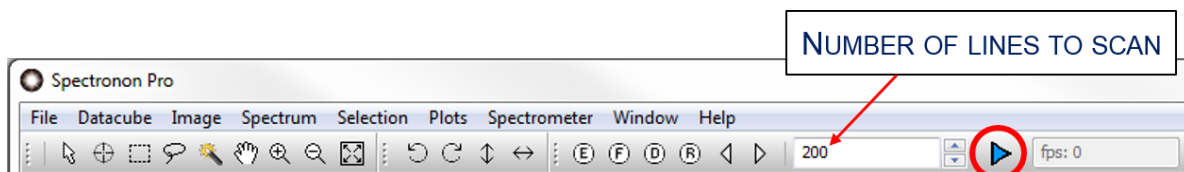
A distortion-free hyperspectral datacube with a unit-aspect-ratio image is achieved when the stage/object advances the distance of the projected image of one spatial pixel per each line acquired at the imager's given frame rate. Thus, changing the *Framerate* setting requires updating the stage's *Scanning Speed* to maintain unity aspect ratio, while changing only the *Integration Time* does not. Similarly, changing the working distance of the imager to the stage changes the object magnification (and FOV or swath width), and thus changes the aspect ratio for any particular *Framerate* and *Scanning Speed* combination.

Hint: Typically, you want the fastest frame rate possible that supports the integration time required for the illumination and sample brightness. This is because faster frame rates produce faster scans.

By default, **Spectronon** has *Auto Speed Mode* enabled, which automates required *Scanning Speed* adjustments each time the *Framerate* changes. This depends on a single calibration parameter, the *Swath* setting (in cm) for the Benchtop System, or the *FOV* setting (in deg) for the Outdoor Field System. Once this parameter is properly “tuned” for a given system setup, you can freely adjust the *Framerate* without having to manually update the *Scanning Speed*, which is updated automatically. For very low frame rates (long integration times), **Spectronon** automatically compensates by turning on *Stepping Mode* and setting the proper *Step Size*.

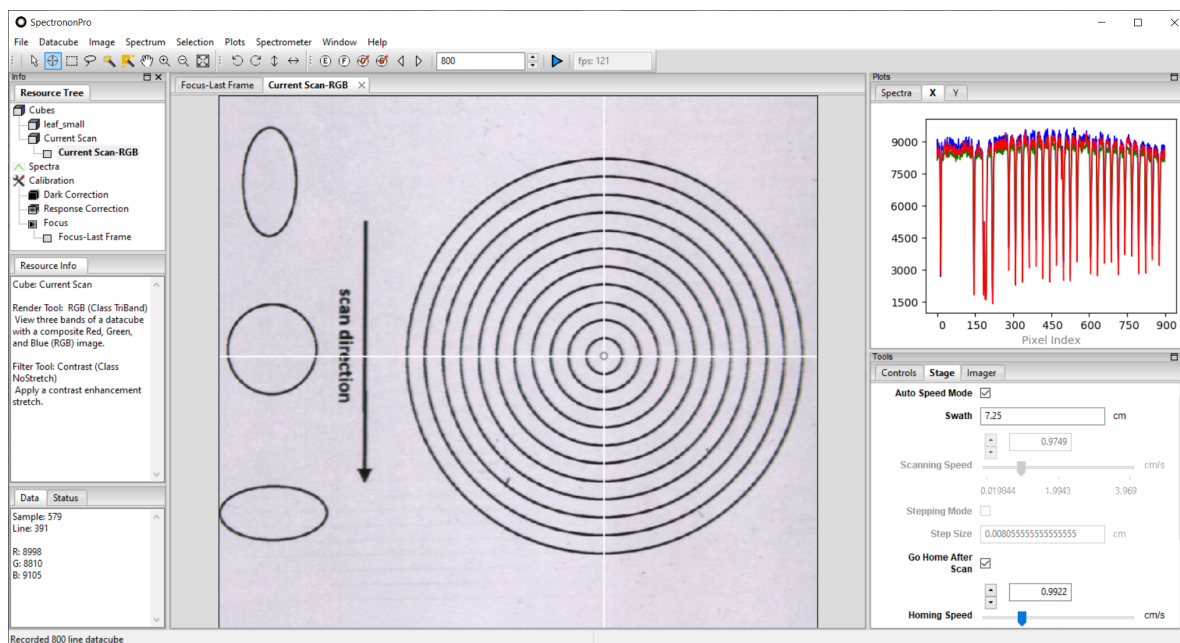
Benchtop System

To determine the *Swath* setting for the Benchtop System, it is useful to first scan an object whose distortion is easy to observe, such as a circle. Print out the **Pixel Aspect Ratio Calibration Sheet** provided in [Section 10: Focusing & Calibration Sheets](#) for this purpose. After placing an object with circles within the FOV of your imager, record a scan with enough lines that you can see the complete circle. (500-800 lines are usually sufficient.) Record the scan by clicking on the *Scan Button* , and a waterfall RGB image should appear in the Image Panel of **Spectronon**. You may need to record several trial images to determine how many lines to scan and where to best position the object.



Hint: Stop a scan by re-clicking on the *Scan Button* , which changes to the *Stop Button*  during the scan.

If the aspect ratio of the image is distorted, then you will need to change the *Swath* setting on the *Stage* tab on the *Tools* panel. If your image is elongated along the scan direction, then the stage was moving too slowly in relation to the frame rate (over-sampling), and you should increase the *Swath* setting, which automatically increases the *Scanning Speed*. Conversely, if your image is compacted along the scan direction, then the stage was moving too quickly in relation to the frame rate (under-sampling), and you should decrease the *Swath* setting, which automatically decreases the *Scanning Speed*. Repeat the above process until your image is no longer distorted, as pictured below.






It is also possible to measure the swath directly by imaging the vertical lines of the provided [Section 10: Focusing & Calibration Sheets](#). Count the number of lines, and use the width between each line to estimate the total width, which is the swath width. This can even be done at the end of the imager focusing described in [Section 3.6: Focusing Objective Lens](#).

Outdoor Field System

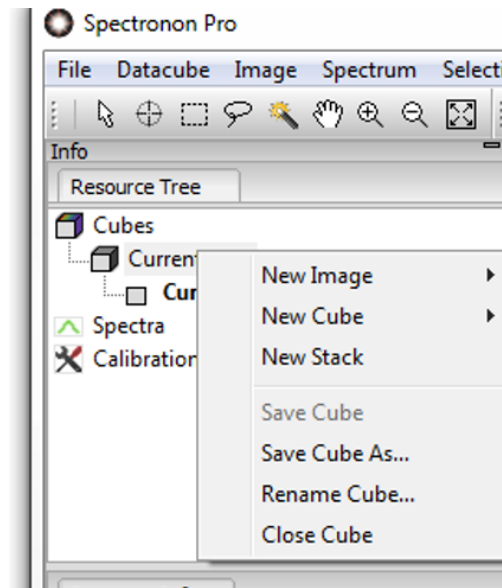
To determine the *FOV* setting for the Outdoor Field System, simply use the (far-field) FOV value specified for the imager's objective lens, which assumes the objective lens is focused at infinity.

Note: For help setting the aspect ratio without using *Auto Speed Mode* see our Setting Aspect Ratio video at <https://www.youtube.com/@resonon/videos>.

3.8.2 Scanning Objects

To scan an image of an object after calibrating the system, place the object on the stage within the imager's FOV and type in the number of lines you would like to scan in the window just to the left of the *Scan Button* . Record a hyperspectral datacube (the image) by pressing the *Scan Button* . (Click  to terminate the scan early, if needed.) Adjust the stage position and increase/decrease the number of lines as needed to cover the object's feature(s) of interest.

A waterfall image of your datacube will appear in the Image Panel of Spectronon, and a new entry labeled *Current Scan* will appear in the *Resource Tree*. To save the scanned datacube, use your mouse to select *Current Scan* and then either right-click or select *Datacube* → *Save Cube*. This will open a dialog window that allows you to name the datacube and save it in a folder of your choosing. If you do not save your datacube, the current scan will be overwritten when you record another datacube, and a warning should appear.

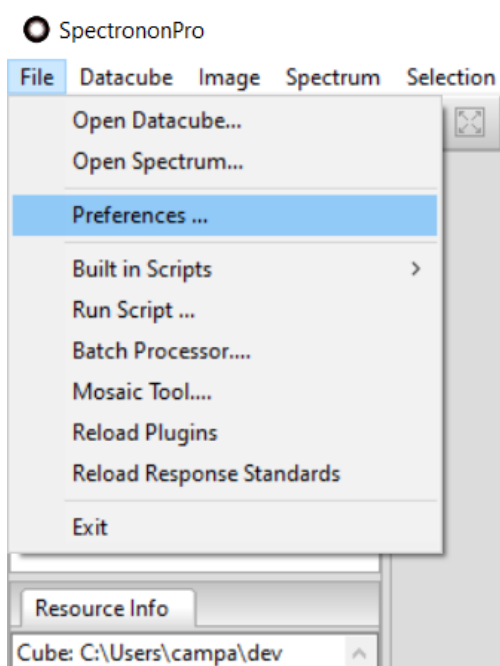


Hint: For additional help scanning and saving datacubes see our Scanning and Saving video at <https://www.youtube.com/@resonon/videos>.

Once a datacube is scanned, you can use all the visualization and analysis tools of **Spectronon[Pro]** on your datacube image, as introduced in [Section 5: Basic Data Analysis](#).

ADVANCED DATA ACQUISITION

Many advanced data acquisition settings in **Spectronon** are available by opening the *Preferences* window from the Main menu by selecting *File* → *Preferences...*



4.1 Imager Controls

Your Pika imaging spectrometer can be controlled from two locations: the *Imager* tab in the *Tools* panel of the main window (frequently used settings only), and the *Preferences* window (full settings).

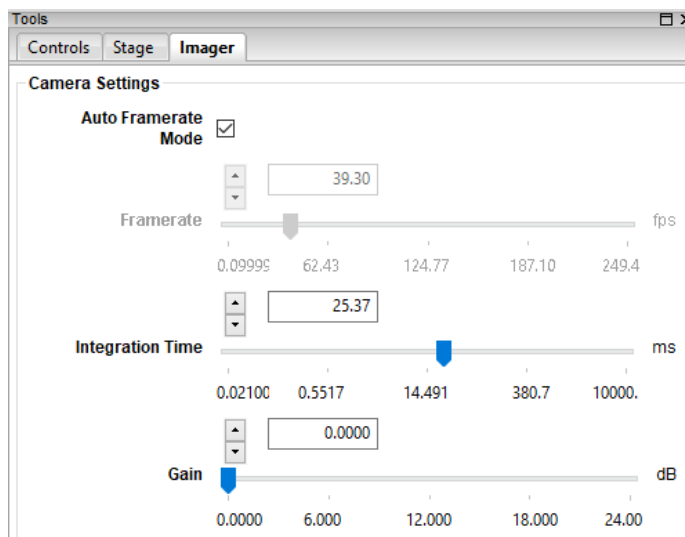


Fig. 1: Imager tab in Tools panel of Main window

Open the *Preferences* window and select the *Imager* tab. The options for different imager models vary (Pika UV, Pika L, Pika XC2, Pika IR, Pika IR+, Pika IR-L, Pika IR-L+).

Preferences

Plotters Scanner Workbench Stage **Imager**

Imager Info

Model

Serial Number

Camera S/N

Spectral Calibration

A

B

C

Binning

Type

Spatial Pixels

Spectral Pixels

Camera Settings

Framerate fps

Integration Time ms

Gain dB

Auto Expose

Target Brightness DN

Target Pixel Histogram Percentile

Fig. 2: Imager tab for a Pika L in Preferences window

4.1.1 Imager Info

This group provides basic information about your Resonon imager, such as model and serial number(s). The *Generate configuration report* button generates text-formatted snapshot of your imager's configuration for future reference, support, or communications with Resonon.

4.1.2 Spectral Calibration

This section lists coefficients used to calibrate your imager. For most Resonon imagers, calibration coefficients are stored in non-volatile memory on the imager itself, and do not need to be modified after they are set by Resonon. See [Section 3.3: Verifying Imager Calibration](#), for additional details. The values are additionally provided on a paper calibration sheet with your imager. Contact Resonon support if you have lost your spectral calibration values.

4.1.3 Binning

The **binning** (adjacent channel summing) of spectral data can be useful for reducing data size, increasing signal-to-noise ratios, or increasing scanning speed. All Resonon hyperspectral imagers support software binning, and some support hardware binning as well.

The binning *Type* control selects the binning type, *software* or *hardware* (when available). The advantages of each of these binning methods is explained below.

If **software binning** is utilized, full frames are read off the imager and then adjacent spectral and/or spatial channels are summed in software, producing a frame of reduced spectral/spatial size inversely proportional to the binning factors. If enough light is available to nearly saturate each unbinned pixel (filling the well of the pixel), this technique has the advantage of increasing the effective well depth by the binning factor. This improves the signal-to-noise ratio and dynamic range of the data.

Hardware binning occurs within the camera electronics before a frame is read. It does not improve the signal to noise like software binning does, but it is useful to increase the maximum frame rate of an imager for higher speed scans, such as in airborne applications, where spatial resolution is directly dependent on frame rate.

The *Spatial Pixels* option indicates how many adjacent spatial pixels to bin. This option is typically useful when imaging an object known to be largely homogeneous, so that the spatial resolution is higher than needed. This also reduces the size of the datacubes.

The *Spectral pixels* option indicates how many spectral channels to bin. When native spectral resolution is higher than needed, this is your best choice to reduce the size of the datacubes.

Note: The binning options listed above sum the binned pixels, so binned and un-binned data will have different scales.

Available settings vary between imager models and are not available for some imager types.

Table 1: Maximum achievable framerate vs. binning type

Imager	Pika L	Pika XC2	Pika UV	Pika IR	Pika IR+	Pika IR-L	Pika IR-L+
Max Frame Rate, software binning (fps)	249	121	111	520	249	364	176
Max Frame Rate, hardware binning (fps)	249	165	142	N/A	N/A	N/A	N/A

How to choose: The first step is to determine whether to use binning in any form. In the spectral dimension, this will be likely determined by the smallest spectral features of the object wished to be resolved compared to the spectral


resolution of the imager. If binning is determined to be allowable, it has the advantage of reducing data size, increasing the signal-to-noise of the data, and/or increasing scanning speed, and should be used.

Spatial binning can also be used but typically spatial resolution is too important to compromise. For some imagers, spatial binning does not impact maximum frame rate the same way as spectral binning does.

If scanning speed is unimportant or there is an excess amount of light present, software binning has the advantage of increasing the signal to noise ratios of the data. If scanning speed is important, hardware binning should be explored to increase imager frame rates. It should also be noted that data can always be binned in post-processing to obtain the SNR benefits of software binning.

4.1.4 Camera Settings

These settings are also available in the *Imager* tab in the *Tools* panel in the Main window and were described in [Section 3.4: Imager Controls](#).

Hint: When adjusting the camera settings, you may observe the effects of your adjustments in live view. To do this, click on the *Focus*  button.

Note: If you observe broken or torn images when recording data at high framerates, you may be able to eliminate this problem by reducing the framerate. These errors may occur due to limitations in the hardware of the computer, including disk-write speed, chipset, motherboard bus speed, CPU speed, and available RAM.

GigE Connection Setup

Resonon's imagers with GigE connections require that your network adapter be configured to use jumbo packets. If you purchased your computer from Resonon, these steps may have already been completed for you.

To access the network adapter configuration in Windows, click the start menu and type "Device Manager" Click on *Device Manager*. In the left column, click on *Network Adapters* in the middle column, right click your wired network adapter, and choose *Properties*. In the *Advanced* tab, click *Jumbo Packet* and set the Value to ~9014 Bytes. Click *OK* and close the *Device Manger* window.

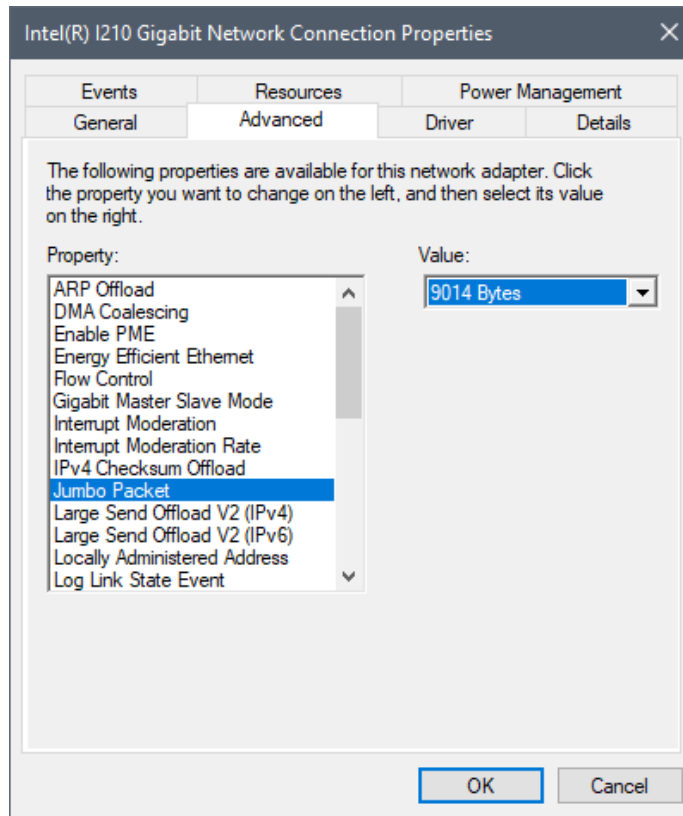


Fig. 3: Device Manager - Network Adaptor - Jumbo Packets

Resonon's Basler GigE imager requires further settings to be configured. In the *Advanced* tab of *Properties* window, you also need to find:

- *Interrupt Moderation*: set to Disabled
- *Interrupt Moderation Rate*: set to Off
- *Receive Buffers*: 2048

Lastly, under the *Power Management* tab, uncheck the option that reads "Allow the computer to turn off this device to save power."

Note: If these options are unavailable, the Basler GigE will most likely not function properly. Please contact Resonon support (support@resonon.com) for further assistance.

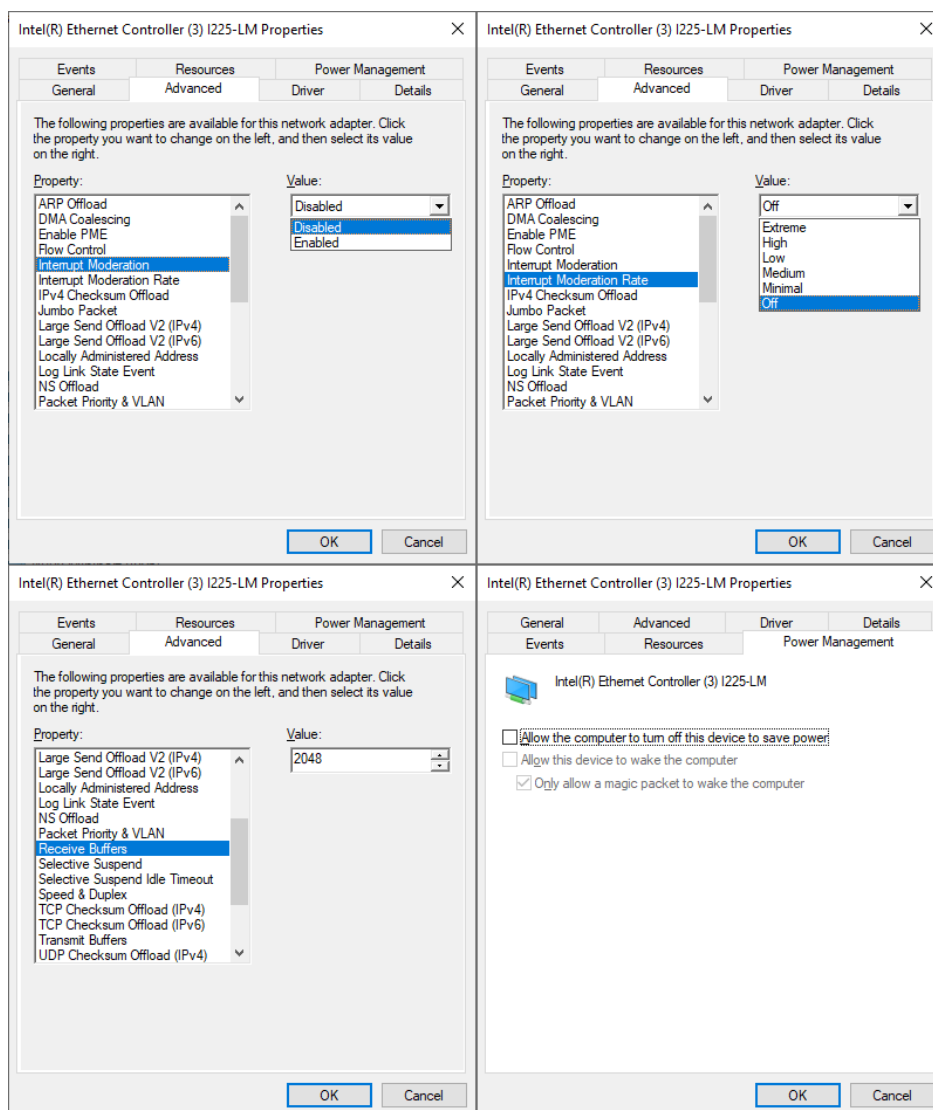


Fig. 4: Device Manager - Network Adaptor - Basler GigE settings

4.1.5 Auto Exposure Parameters

Spectronon features a user-configurable auto exposure algorithm that can be controlled in the *Imager* settings tab. Two parameters are available: *Target Brightness* and *Target Pixel*. During the auto exposure routine, **Spectronon** calculates the histogram of pixel brightness of each frame. For a user-specified (fixed) *Framerate*, it then iteratively adjusts the *Integration Time* setting such that the histogram percentile represented by the *Target Pixel* parameter is at least as bright as the *Target Brightness*, but is not saturated. If the imager's camera supports *Auto Framerate Mode*, then this feature can be turned on to have **Spectronon** automatically select the fastest *Framerate* as the auto exposure routine searches for an appropriate *Integration Time* from the camera's full range.

For example, if the *Target Pixel* is equal to 100% and the *Target Brightness* is equal to 3890 (the default settings for 12 bit cameras), the algorithm will attempt to find an exposure time such that the brightest pixel has a brightness of at least 3890 but less than 4095 (saturation). Similarly, if the *Target Pixel* = 95%, the pixel that represents the 95th percentile of total brightness will be targeted in each frame, instead of the brightest pixel. Selecting a *Target Pixel* less than 100% may cause the brightest pixels to be saturated, but may be useful for achieving desired exposure when features of interest are not the brightest in the field of view. For cameras whose detectors do not typically have "hot

(saturated) pixels”, *Target Pixel* is set to 100% and *Target Brightness* is set to 95% of the camera’s bit depth by default.

4.2 Stage Controls

The stage can be controlled from two locations: the *Stage* tab in the *Tools* panel of the main window (most frequently used settings), and the *Preferences* window (full settings).

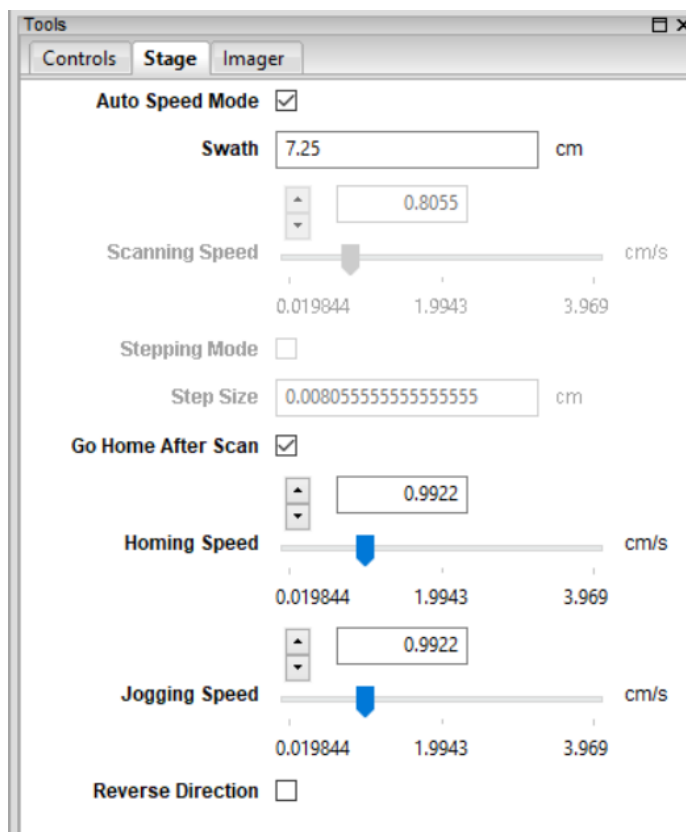


Fig. 5: Stage tab in Tools panel of Main window

Open the *Preferences* window and select the *Stage* tab.

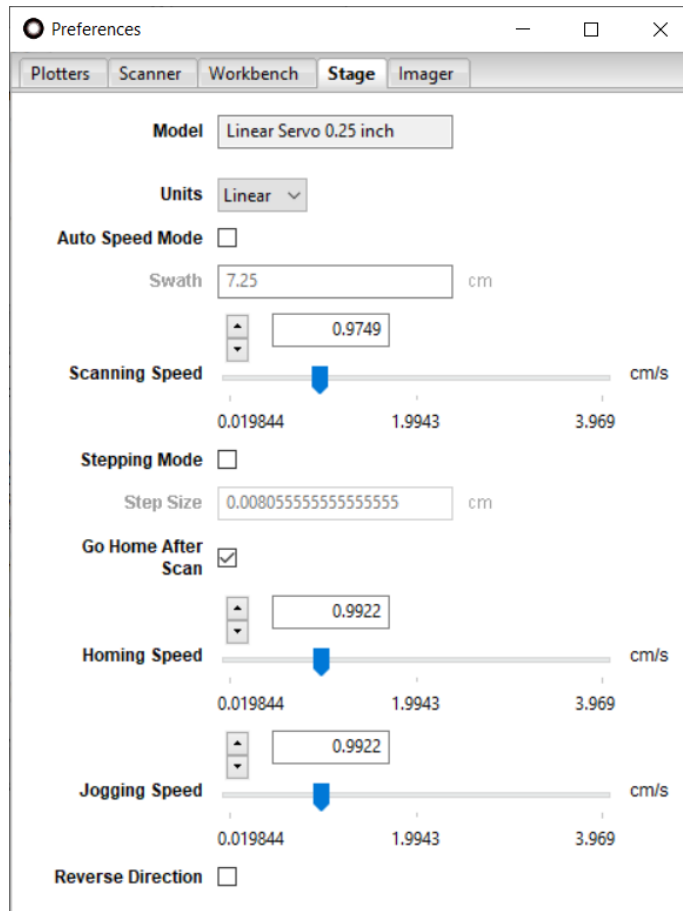


Fig. 6: Stage tab in Preferences window

The stage is typically operated with *Auto Speed Mode* active, which means that the *Scanning Speed* updates automatically when the imager camera's *Framerate* changes. When not in this mode, you have the freedom to customize all of the aforementioned settings.

The following advanced stage settings are available in addition to those described in [Section 3.5: Stage Controls](#).

4.2.1 Model

This indicator provides information about the specific model of stage, including the stage's screw pitch in mm (or inches) per revolution.

4.2.2 Units

For the Benchtop System, the stage settings use linear (cm) units. For the Outdoor Field System, the stage settings use rotation (deg) units. **Spectronon** determines the options automatically from the connected stage.

4.3 Additional Controls

Additional settings are available in other tabs in the *Preferences* window, as well as in the menus of the Main window.

4.3.1 Scanner Preference Tab

The *Scanner* tab in the *Preferences* window provides additional options for scanning datacubes.

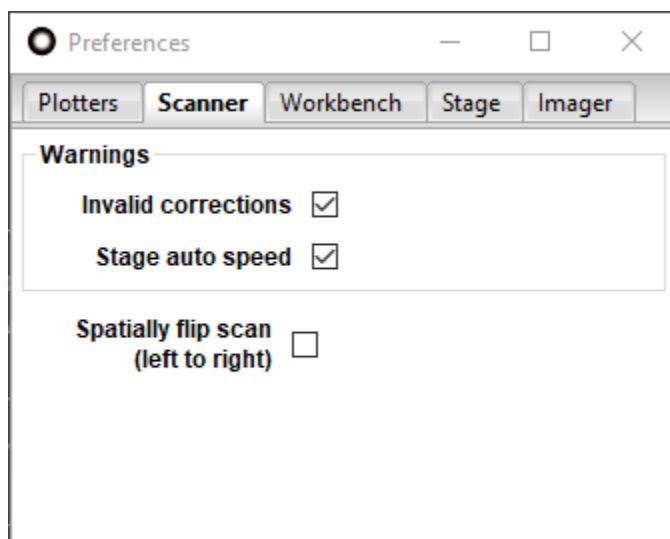


Fig. 7: Scanner tab in the Preferences window

The *Warnings* options allow you to suppress the reporting of certain checks that **Spectronon** performs before taking each scan. These warnings may be alternatively disabled by clicking “don’t show this again” when they are displayed.

The *Spatially flip scan (left to right)* option flips the data orthogonally to the scan direction.

4.3.2 Workbench Preference Tab

The *Workbench* tab in the *Preferences* window provides additional options for analyzing, displaying, scaling, and saving datacubes.

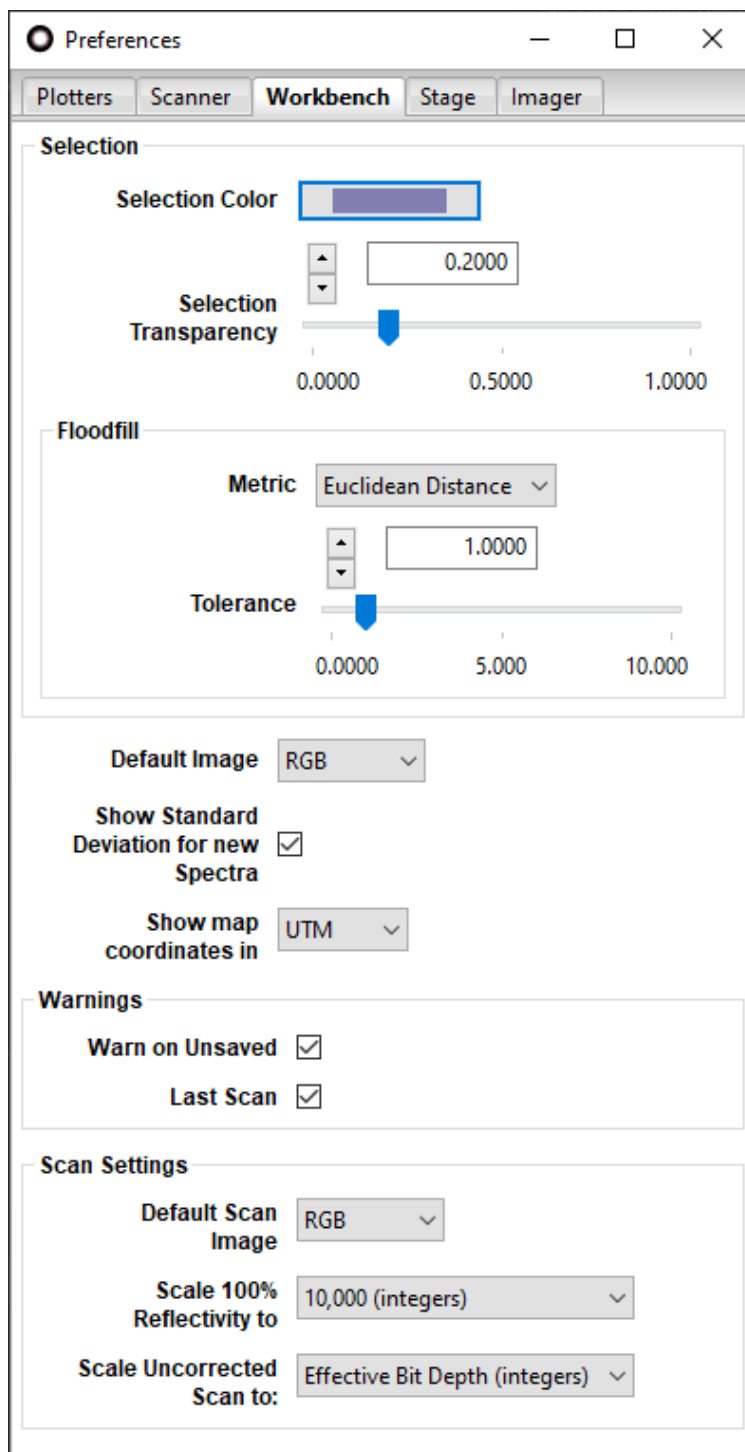



Fig. 8: Workbench tab in the Preferences window

The *Selection* options relate to visualizing selected regions of datacubes.

The *Floodfill Metric* options allow you to specify whether to use *Euclidean Distance* or *Spectral Angle* when using the *flood fill* (wand)  tool to select contiguous regions of spectrally similar pixels. The *Tolerance* parameter determines the sensitivity of these metrics.

Default Image determines whether to visualize your existing datacubes as a Red Green Blue (RGB) composite or single band greyscale image upon opening.

Show Standard Deviation for new Spectra toggles the display of +/-1 standard deviation shaded bands for mean spectra computed from multi-pixel selections.



For datacubes with map coordinates, *Show map coordinates in* selects the units.

The *Warnings* options determine which warning dialogs are triggered for unsaved datacubes. When checked, the *Warn on Unsaved* option shows a warning message if you have not saved your last scan before closing the software. Similarly, the *Last Scan* option warns you if you have not saved your last scan before it is overwritten by a new scan.

The *Scan Settings* options affect how new datacube scans are processed. The *Default Scan Image* option determines whether to visualize your new scans as Red-Green-Blue (RGB) composites or single band greyscale images. This setting does NOT impact the hyperspectral data itself; it only changes how the image of the data is presented in the image panel.

The *Scale 100% Reflectivity to* option allows you to record reflectance data scaled to *1.0 (floats)*, *10,000 (integers)*, or *Effective Bit Depth (integers)*. This saves your data as a standard reflectance value IF you use a reflectance reference with a reflectivity of 1. (Spectralon® and Fluorilon® are good materials for this, and Teflon® is a less expensive and reasonable substitute reference material, although less accurate.) Two-byte unsigned integer data is ensured by selecting *10,000 (integers)*, which also provides reasonable dynamic range. If you use the *1.0 (floats)* option, then each channel of data will require 4 bytes instead of 2 bytes. Thus, the size of your datacube will double with this option as compared to *10,000 (integers)*. The *Effective Bit Depth (integers)* scales the datacube to the *effective* bit depth, which is the hardware bit depth of the imager's camera (e.g., 12-bits, or 4095, for the Pika L, XC2, and UV and 14-bits, or 16,383, for the Pika IR, IR+, IR-L, and IR-L+), combined with any increase from software binning of pixels in the spatial and/or spectral dimension(s), which may require a 4-byte unsigned integer to be properly accommodated.

The *Scale Uncorrected Scan to* option allows you to scale the raw data taken with your imager to *Effective Bit Depth (integers)* or *1.0 (floats)*. We recommend using the *Effective Bit Depth (integers)* unless you require the other option.

Note: The use of “uncorrected” above refers to the user not using the *Dark Current*  or *Response Correction*  tools prior to collecting a datacube. You would do this if you wanted to collect raw data and convert it to Radiance using radiometric calibration data provided by Resonon. Prior to converting, raw data is in digital number (DN).

4.3.3 Spectrometer Menu

The *Spectrometer* menu provides alternative approaches to useful actions, as well as some time saving options.

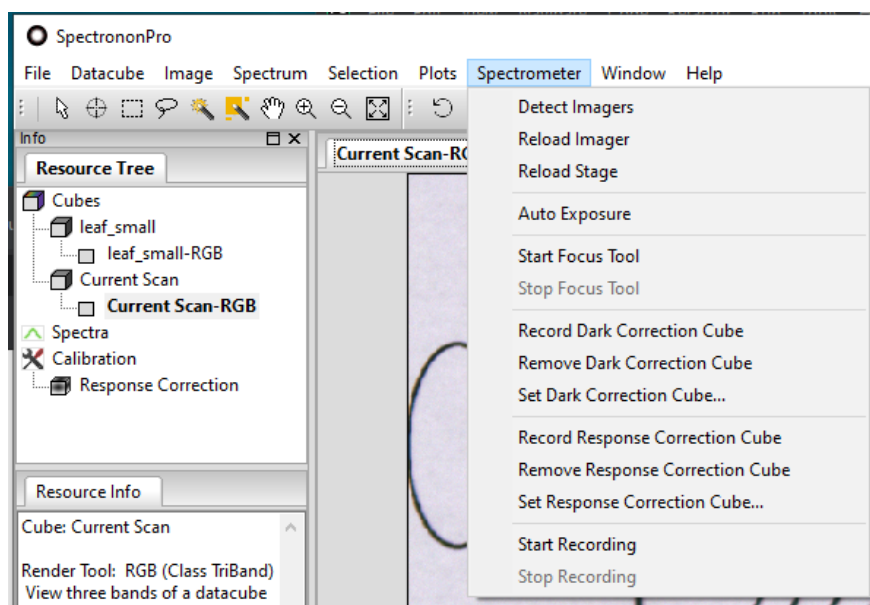



Fig. 9: Spectrometer menu on main window


Spectrometer → *Detect Imagers* initiates a search for connected imagers, allowing you to detect an imager that was not plugged in when **Spectronon** was initially opened.


Spectrometer → *Reload Imager* attempts to reload the Pika hyperspectral imager without restarting **Spectronon**. This option will save you time if you have, for example, accidentally disconnected your imager.



Spectrometer → *Reload Stage* reloads your scanning stage without restarting **Spectronon**. This option saves you time if you have, for example, accidentally disconnected your stage.


Spectrometer → *Auto Exposure* automatically sets the integration time for your lighting conditions according to the settings discussed in [Section 4.1.5: Auto Exposure Parameters](#). You can also start autoexpose with the *Exposure* button

 from the toolbar.



Spectrometer → *Start Focus Tool* provides a live view from your imager in the *Image* panel in the Main window. This tool is useful for adjusting the focus of your objective lens, checking the illumination of your system, or estimating the swath width along the stage. You can also start the focus tool by clicking on the *Focus* button  on the **Spectronon** toolbar.


Spectrometer → *Stop Focus Tool* turns off the live view, which you need to do before you record a scan. The last live view will remain in a tab of the *Image* panel in the Main window. Another way to turn off the live view is to click the *Focus* button , which toggles live view off and on.

Spectrometer → *Record Dark Correction Cube* records a dark current cube. The mean frame of this cube is subtracted from subsequent scans. The presence of a red check on the *Dark Correction* button  indicates whether a dark correction cube is active. When you select this option you will be instructed to block all light from entering the imager's objective lens and then click *OK*. Another way to do this is with the *Dark Correction* button  on the **Spectronon** toolbar.


Spectrometer → *Remove Dark Correction Cube* removes the dark current correction cube. The presence of a red check on the *Dark Correction* button  indicates whether a dark correction cube is active.

Spectrometer → *Set Dark Correction Cube...* sets the dark current correction cube from a file.

Spectrometer → *Record Response Correction Cube* records a datacube of a reference material against which all subsequent measurements will be scaled. Typically this is done with a reference material whose reflectivity is approximately equal to 100%. When you select this option, you will be instructed to place a reference material filling the imager's field of view, and then click on *OK*. A small response correction datacube will then be recorded and its average used for scaling the data as described above. The presence of a red check on the *Response Correction Cube* button  indicates whether a white reference cube is active. Another way to perform this function is to use the *Response Correction Cube* button  on the **Spectronon** toolbar.

Spectrometer → *Remove Response Correction Cube* removes the existing response correction cube. The presence of a red check on the *ResponseCorrection Cube* button  indicates whether a white reference cube is active.

Spectrometer → *Set Response Correction Cube...* sets the white reference cube from a file. This option is rarely utilized, as one should generally record a correction cube regularly and with the current lighting to obtain accurate results.

Spectrometer → *Start Recording* and *Spectrometer* → *Stop Recording* start and stop, scans. This can also be done using the *Scan* button  on the **Spectronon** toolbar. The Scan Button toggles between Start and Stop functions depending on the scanning state.

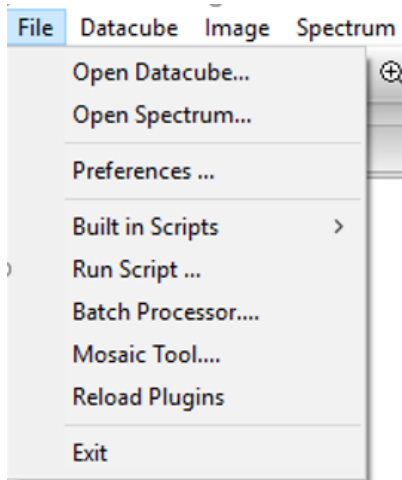
BASIC DATA ANALYSIS

Spectronon provides visualization and manipulation capabilities for hyperspectral images. Spectronon software has all the features of Spectronon, but also enables data acquisition from Resonon's family of imagers. Spectronon software can be downloaded for free on Resonon's website <http://www.resonon.com/>. Spectronon comes bundled with any of Resonon's imaging spectrometers.

This section begins with basic operation of Spectronon, such as opening a hyperspectral datacube and viewing the data. A complete description of visualization tools is provided in sections 6, 7, and 8 on **Advanced Data Analysis**. The Spectronon custom plugin manual discusses how to implement user-written algorithms into Spectronon, enabling custom data analyses.

5.1 Spectronon Tools

To open a datacube, select *File* → *Open Datacube*.

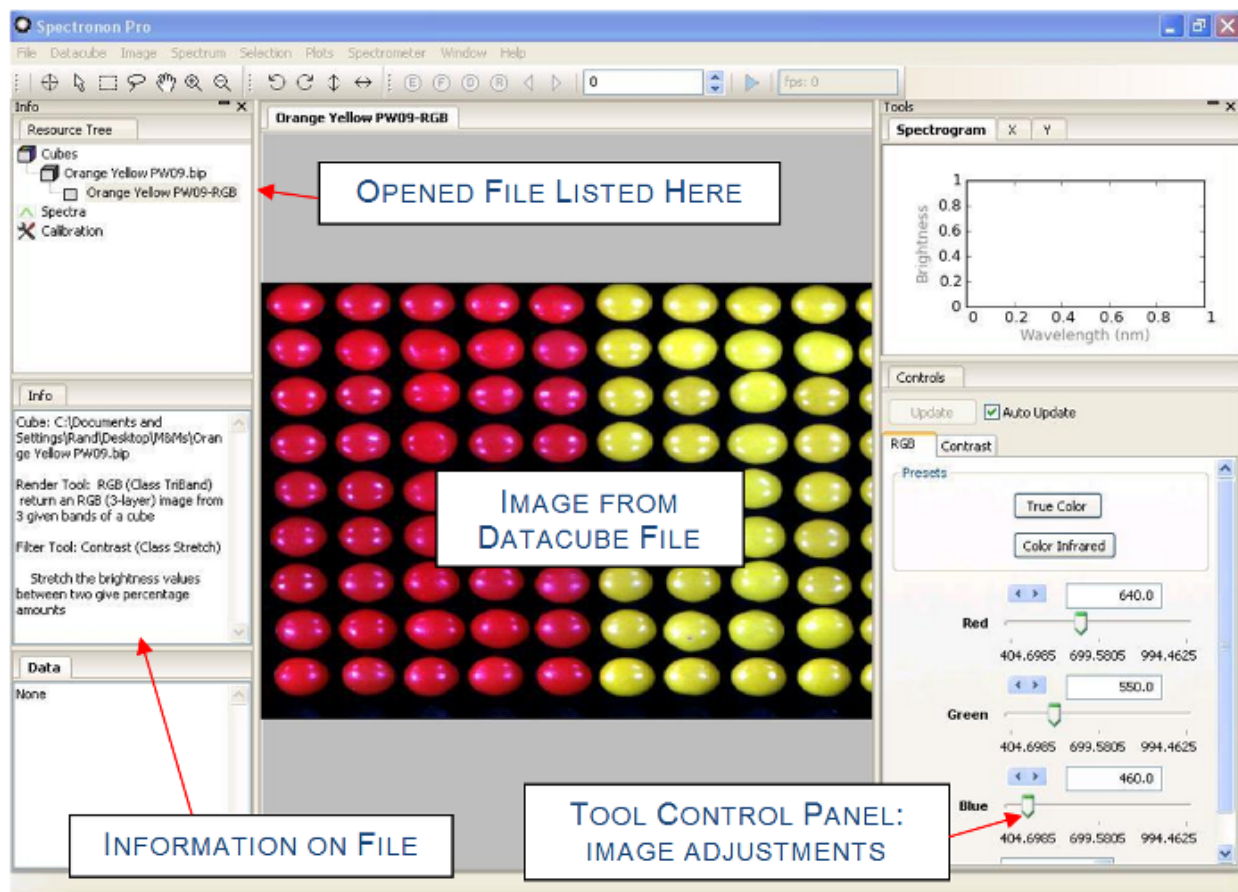


This will open a dialog that allows you to browse to find your datacube. Select your datacube and click on *Open* to load your datacube. This will result in the following:

- An image of your datacube will appear in the image panel
- A listing of the open datacube will appear in the Resource Tree
- Tabs will appear in the Parameters window that allow you to change the image (more on this later)
- Header information on your datacube will appear in the information panel

Note: Spectronon can open any datacube with an ENVI© formatted header. This includes .bip, bil, and .bsq formats.

This chapter employs an example datacube of M&M® and Reese's® Pieces candies. (This datacube can be downloaded from Resonon's website at <http://downloads.resonon.com/>.) By default, the datacube is opened with a true color image of the data, which approximates the appearance of the object under normal lighting conditions by combining red, green, and blue wavelengths from the datacube.



With a few minutes of practice using the available tools, you will be able to manipulate and visualize hyperspectral data quickly and efficiently.

5.2 Zoom, Pan, Flip, and Rotate Tool



To zoom to a specific area of the image, select the *magnify* tool in the toolbar and the cursor will change. Click the *magnify* tool in the image, and the view will zoom in. It is also possible to click and drag a selection within the image to zoom into the selected area.



To zoom out, select the *demagnify* tool and click anywhere in the image.



To zoom all the out and recover the original image, select the *original size* tool and click anywhere in the image.

The user may also zoom in and out using the mouse scroll wheel, if available.



To pan the image while zoomed in, select the pan tool. Click and drag inside of the Image to pan.



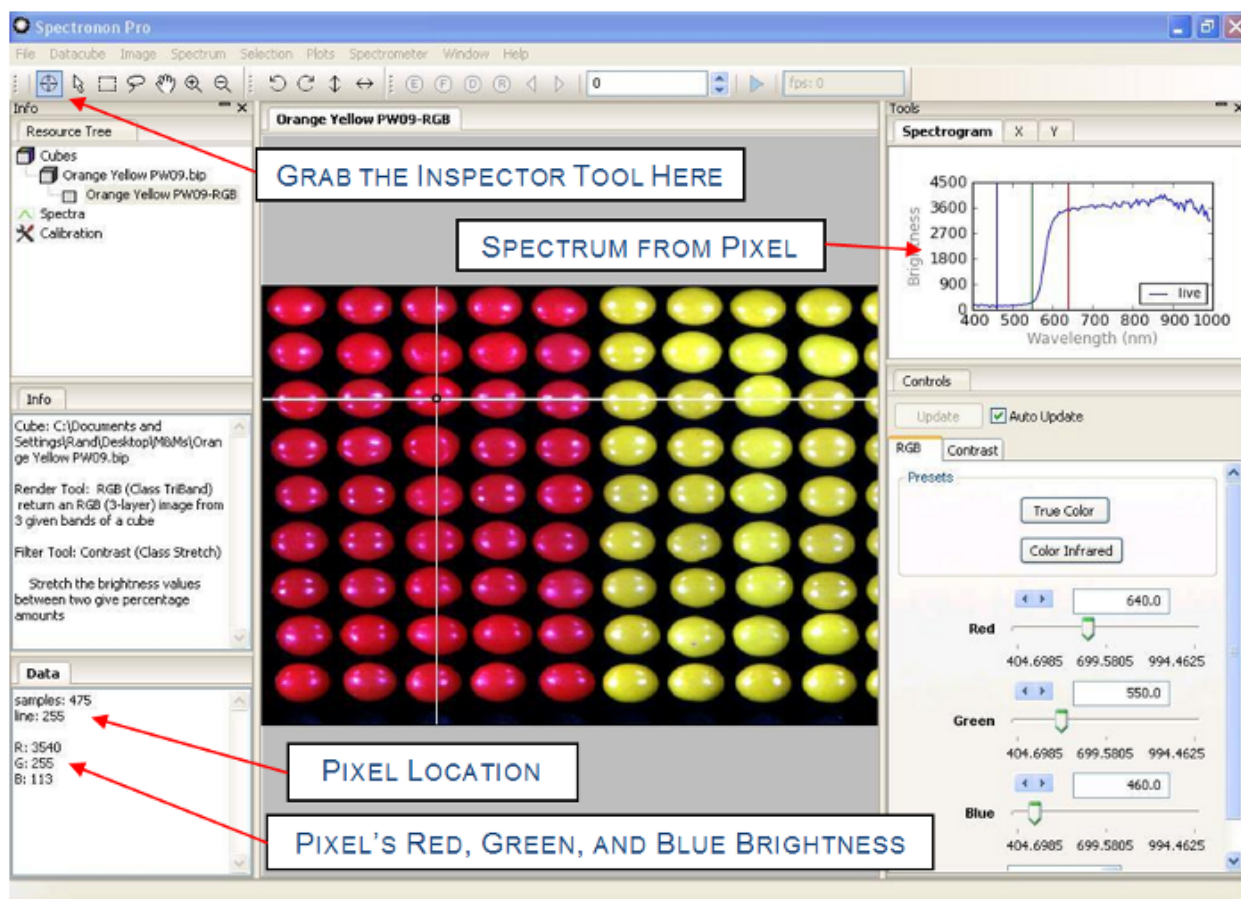
Click these tool to rotate left, rotate right, flip vertically, or flip horizontally the image.

5.3 The Inspector Tool – Spectral Plots



The *inspector* tool allows you to see the spectrum associated with a pixel. Choose the *inspector* from the toolbar, and then click a point inside the image. This will:

- Plot the spectrum for the pixel in the *spectrum plot panel*
- List the pixel location (sample and line number) in the *data panel*
- List the red (R), green (G), and blue (B) brightness values in the *data panel*

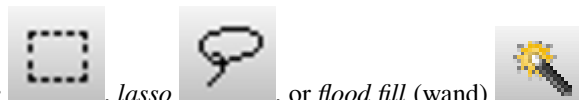


Click on other pixels to see the spectra from other pixels, click and hold while dragging the *inspector* tool to update the *plot panel* continuously.

The red, green, and blue vertical lines in the *spectral plot* indicate the hyperspectral wavelength bands used to generate the current image.

5.4 Region Of Interest (ROI) Tools

It is often useful to consider a group of pixels within the image. The ROI tools enable this capability and provide a number of options. As will be seen later, the ROI tool is often used during one of the first steps in classifying different objects within a hyperspectral image.



To select a Region of Interest (ROI), select either the *marquee* tool from the menu bar. Click and drag a rectangle of interest with the *marquee* tool, or click and drag any closed shape with the *lasso*. The floodfill tool can be used to select a contiguous region of spectrally similar pixels. After selecting an area, right-click to reveal a pop-up menu with several options. Holding control while selecting ROIs allows you to append to the existing selection.

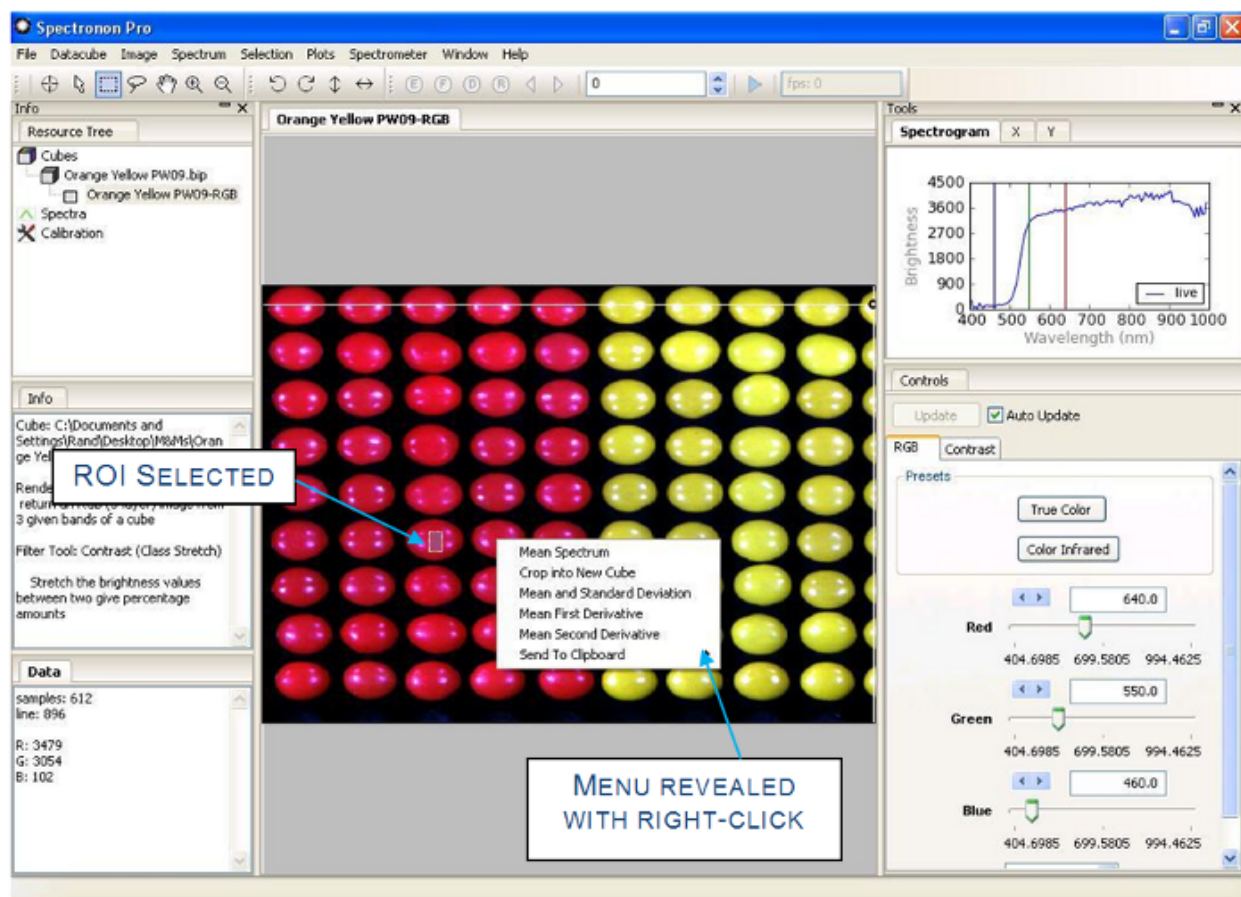
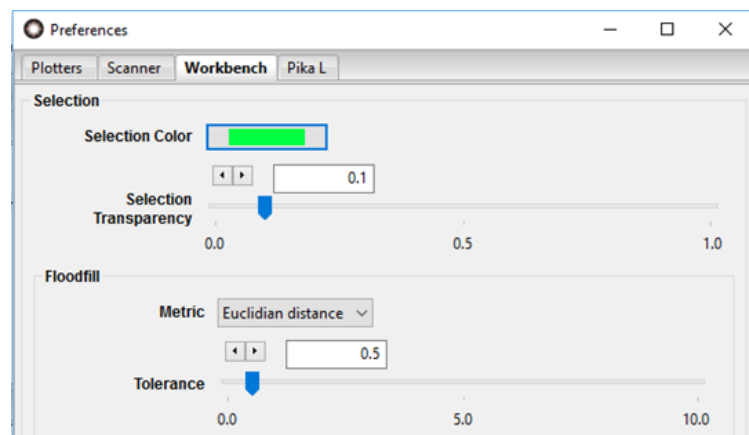


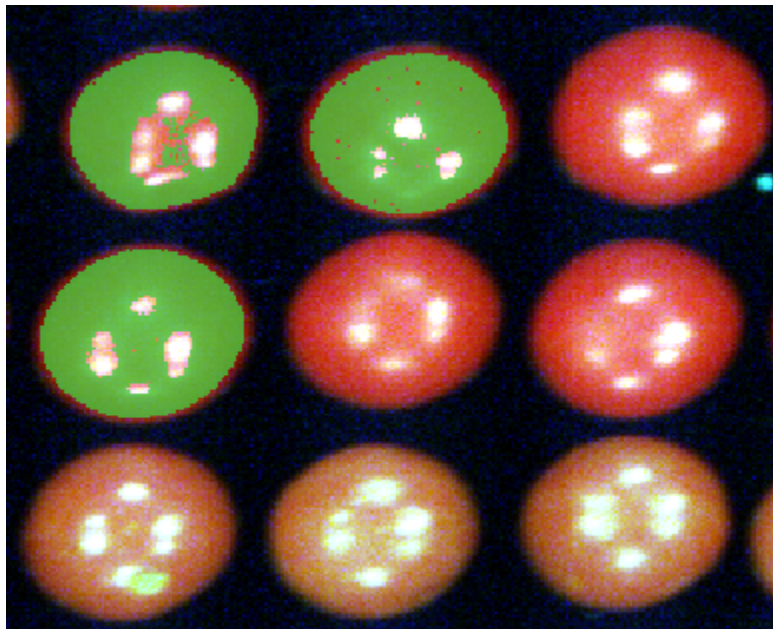
Fig. 1: A small ROI on one of the red candies has been selected and a right-click has revealed the popup selection menu.

Hint: The *selection* menu is also available in the main menu.

The *floodfill* tool shows pixels that are spectrally similar to the chosen pixel. Spectral similarity is assessed with either Euclidean distance or Spectral Angle Mapper (SAM), along with a tolerance value. The user can set these options by accessing the *Workbench* tab from *File* → *Preferences* menu.

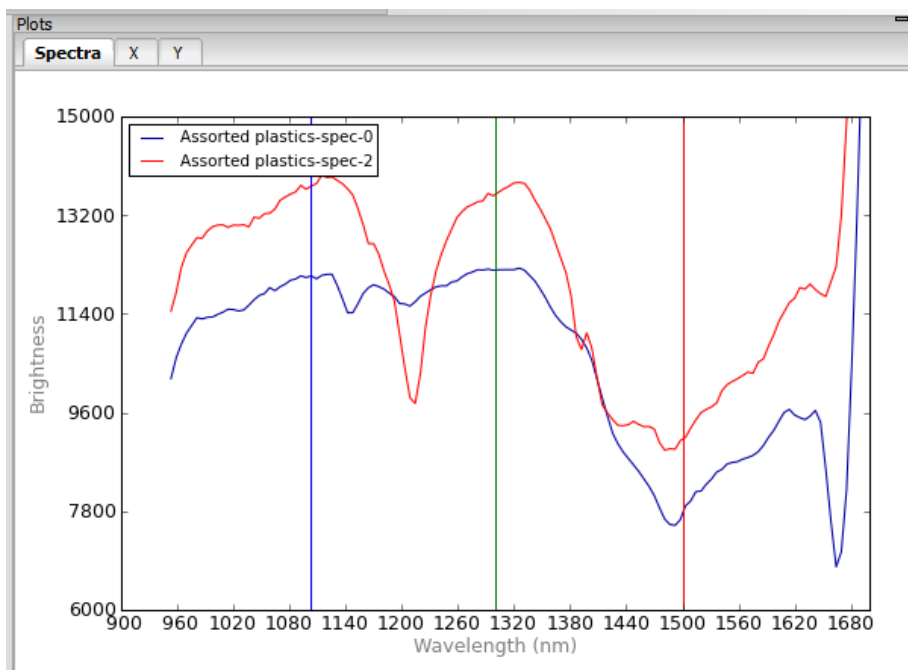


The use of the *floodfill* tool depends on an adjustable tolerance parameter. Floodfill operates on a representation of the datacube scaled from zero to one in each band. It calculates the Euclidean distance or SAM angle in spectral space between the clicked pixel and all contiguous pixels and expands the selection until the selected area contains all of the contiguous pixels for which the spectral distance to the clicked pixel is less than the selected tolerance. Increasing the tolerance will result in a larger selected region with greater spectral variability within that region (i.e. it allows pixels that are less similar to the clicked pixel to be included in the selection). Decreasing the tolerance will result in smaller selected regions with greater spectral similarity. As with the other selection tools, holding control while using the floodfill tool will allow a selection to be built up through multiple clicks of the tool.



An ROI consisting of multiple parts of the image has been selected by using the floodfill tool several times.

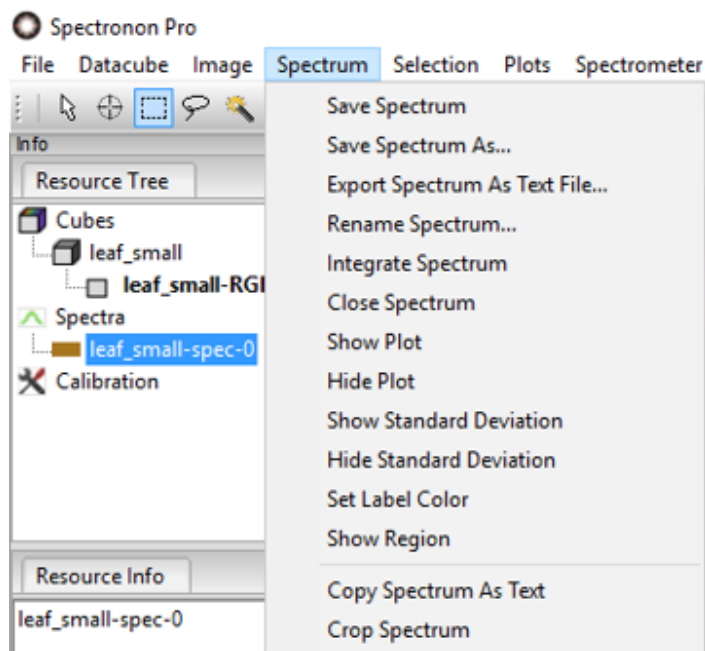
One of the most useful selection options is *mean spectrum*. (Descriptions for the other ROI options can be found in [Basic Data Acquisition](#).) Selecting the mean spectrum option calculates the mean spectrum of all the pixels within the ROI area you selected and plots the result in the spectral plotter. Standard deviation can be shown as an envelope around the spectrum plot. Show or hide the standard deviation envelope with the [Show/Hide Standard Deviation](#) menu items.



Individual spectra can be selected by clicking on the graph of the spectra. You can crop an individual spectrum by selecting it, selecting a rectangular region in the *plot window*, right clicking to bring up the menu, and selecting *crop spectrum*. You can also set the range of the plot by selecting *Plots* → *Spectral Plotter* → *Set Range*.

Hint: To examine the plots in more detail, you may resize the plot panel boundary by dragging the edges. Alternatively, click on the *magnify tool*, then click or drag in the *spectral plotter* to zoom in. The *pan* tool will allow you to pan within the *spectral plotter* as well.

Selecting *Mean Spectrum* creates a new entry in the *resource tree* under a new heading, *spectra*. Right-clicking on the spectrum in the *resource tree* or selecting *Spectrum* from the menu bar will reveal a menu of options. Some of the most used options are listed below.



Save Spectrum

Change the name and save the spectrum as a file.

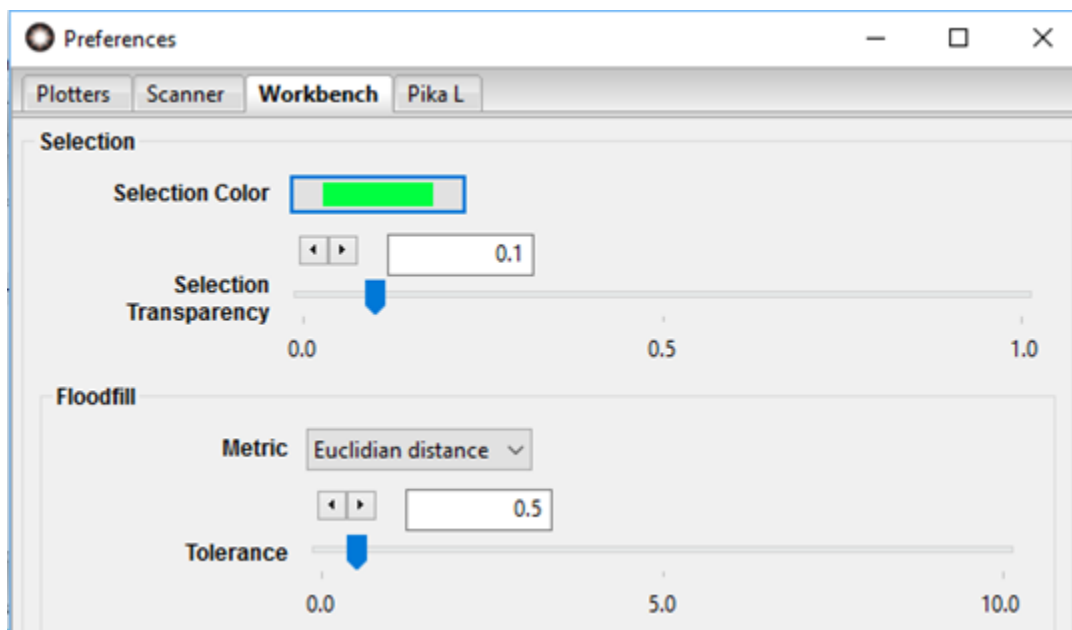
Set Label Color

Open a color picker dialog to change the label color of the spectral plot, and as shown later, classification areas based on this spectrum.

Show Region



Show the originally selected area for the ROI in the current image. This option is useful after you have selected several ROIs.

The color and transparency of the selection can be modified by selecting *File* → *Preferences*. The selection options are located in the *workbench* tab of the preferences window. A selection transparency of 1 represents a completely transparent selection (the selection will not be visible), while a selection transparency of 0 represents a completely opaque selection (the underlying render of the datacube will not be visible through the selection).



5.5 Image Saturation

If any render of a datacube is saturated, and that render is currently selected, Spectronon will alert the user by changing

the *saturation icon*  in the toolbar to red .

The icon will NOT change during a scan and will update if necessary after the scan has been completed. In contrast, the icon will change to red during a live view if the frame becomes saturated.

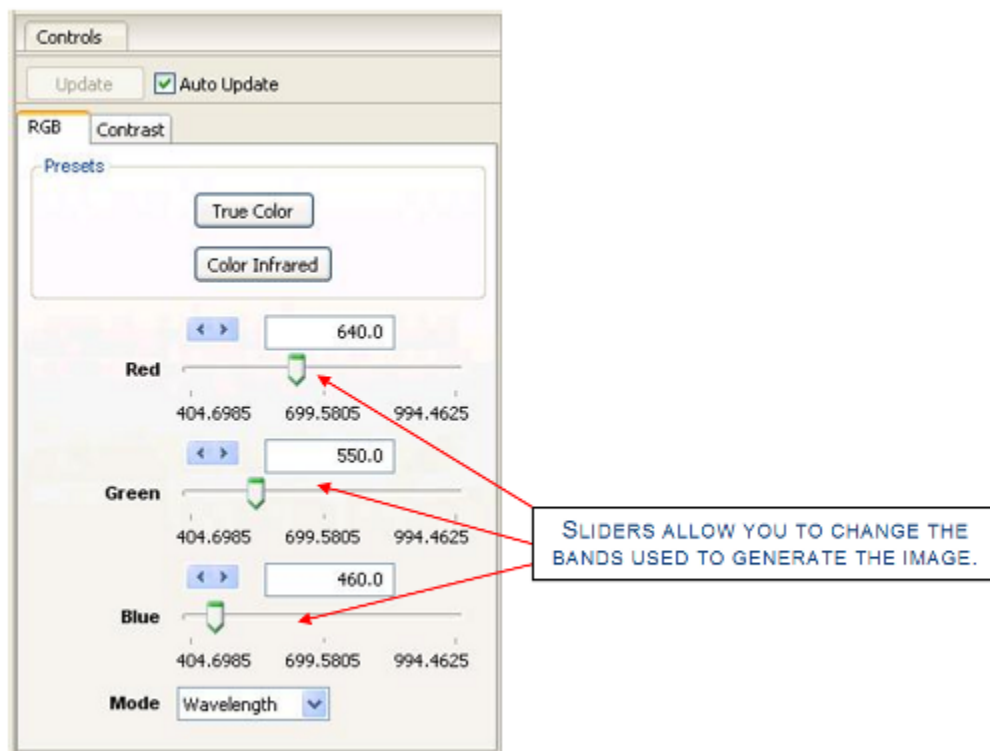
Note that only datacubes with the 'ceiling' header information will trigger the saturation icon to change. The ceiling for raw datacubes is based on the bit depth of the imager which collected the datacube and any spatial or spectral binning. The ceiling for other datacubes may be different depending on how the data is scaled (e.g. if the datacube was converted to radiance or reflectance and the ceiling was adjusted accordingly).

5.6 Image Visualization

Hyperspectral data can be visualized in far more ways than conventional color images. Image controls are provided in the tool control panel.

By default, the image is displayed in *True Color*, which means three representative bands are used to generate a red-green-blue (RGB) image, approximating how it appears to a human eye. The *Color Infrared* preset option provides a false-color RGB image with the red band set to an infrared wavelength. This option is useful for live vegetation datacubes.

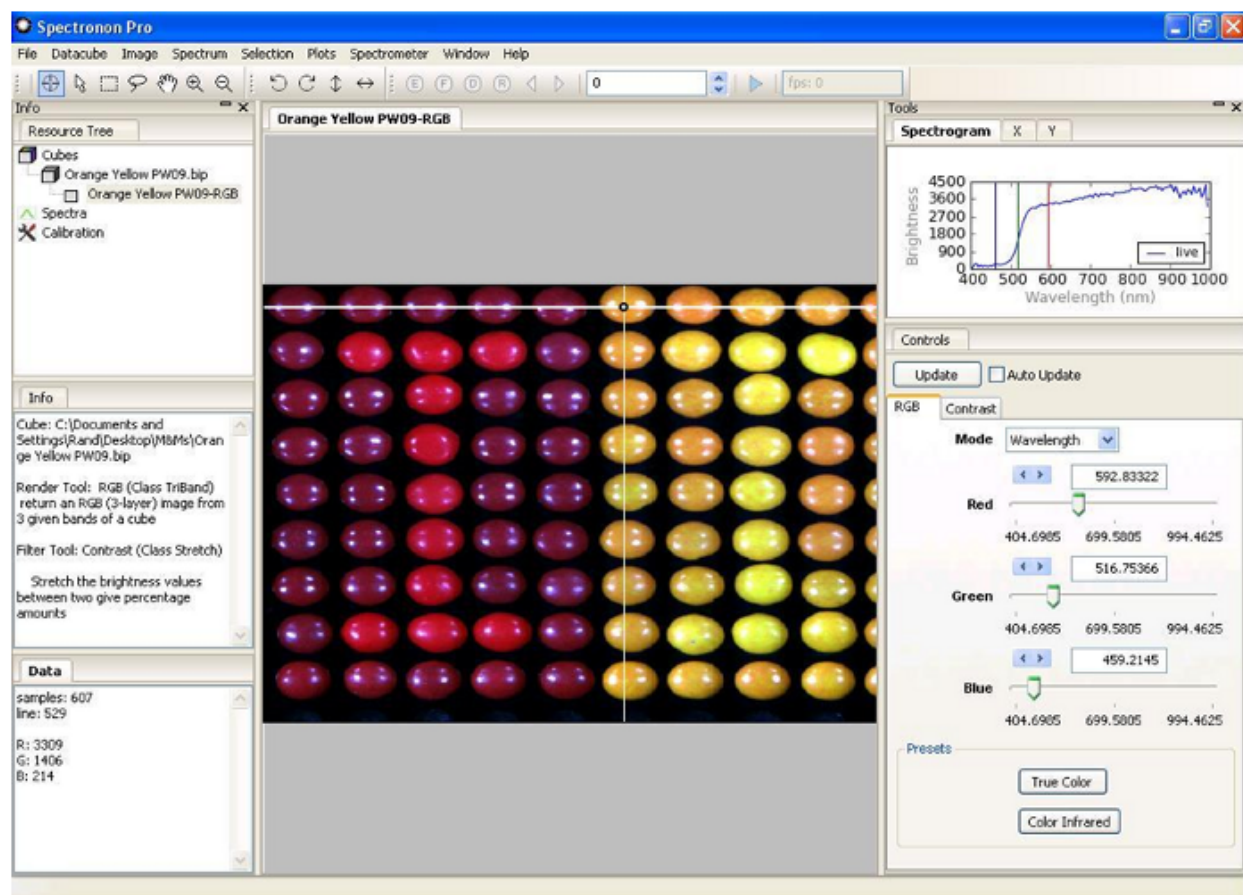
Any time you wish to restore the image to true color, simply click on the *True Color* button under *Presets*.



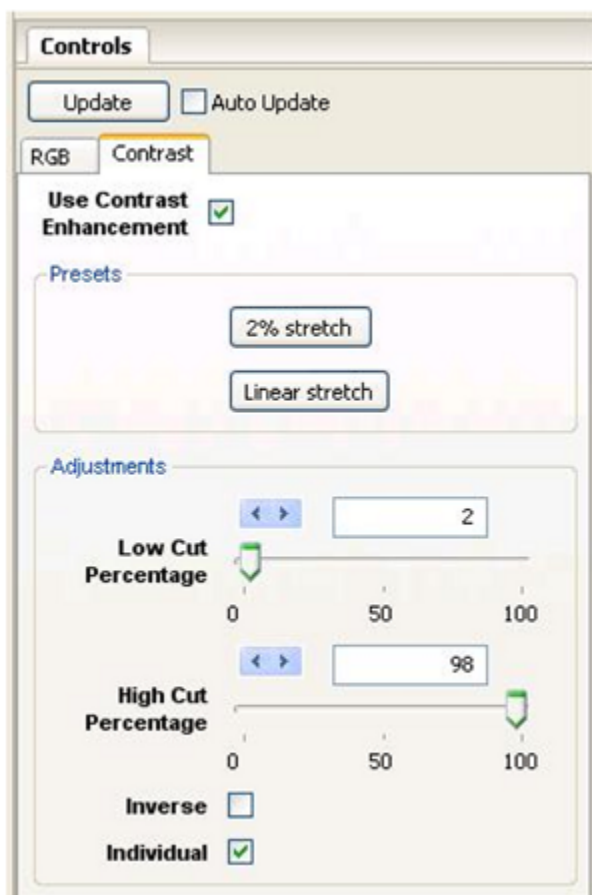
To generate false color images, use the sliders or arrows to change the wavelength bands used to create the RGB Image. This tool is often useful when trying to visualize specific spectral features associated with an object in your image. If *Auto Update* is not selected, click *Update* to generate the new image.

The *Mode* menu allows you to identify the band by wavelength (typically the most useful), or by band number.

As an example, of how false-color images can reveal interesting features, move the red slider to approximately 593 nm, and the green slider to approximately 516 nm, then click *Update*. This generates a new false-colored image, shown below, that reveals there are actually two kinds of red candy, and suggests there are two kinds of yellow candy – each candy type is positioned in the shape of an “T”. In [Section 7](#) a classification technique shows this more clearly.



Note: The Red, Green, and Blue vertical lines in the Spectral Plot show the location of the bands chosen to create the false-color RGB image.



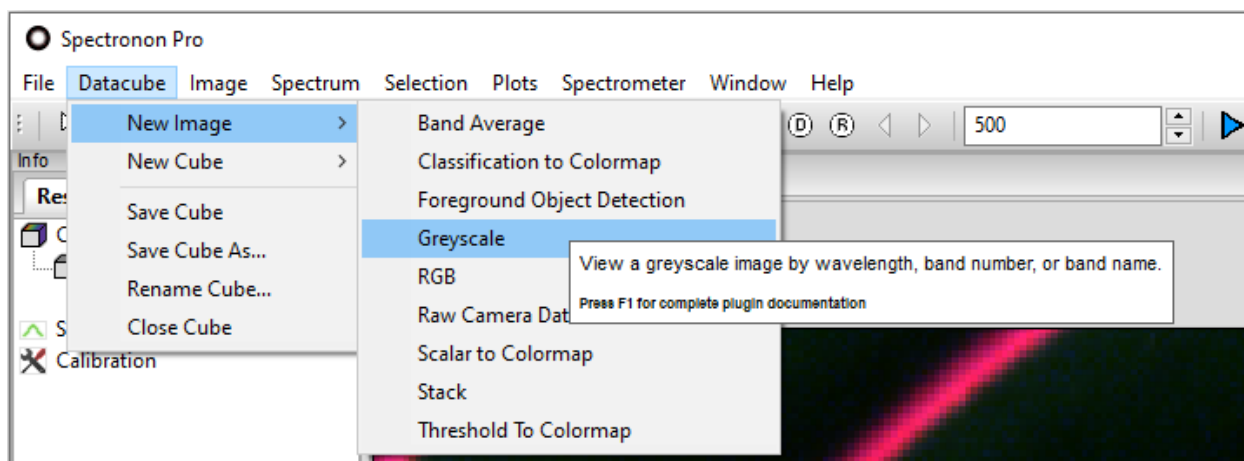
The *Contrast* tab in the tool control panel allows you to adjust the image contrast. If *Use Contrast Enhancement* is not checked, no image enhancement will be done and the tools in the Contrast tab will be not be active.

Note: Contrast enhancement does NOT change the hyperspectral data. It only changes the way the image appears.

Generally, contrast enhancement is beneficial. The *2% stretch* is the default, and it sets the darkest 2% of the pixels in the image to a value of 0, and the brightest 2% of the pixels in the image to maximum brightness (255). This choice minimizes the impact of glare. You can customize the percentage of the dark pixels set to 0 and the percentage of the bright pixels set to 255 with the sliders. The *Linear stretch* option sets these percentages to zero.

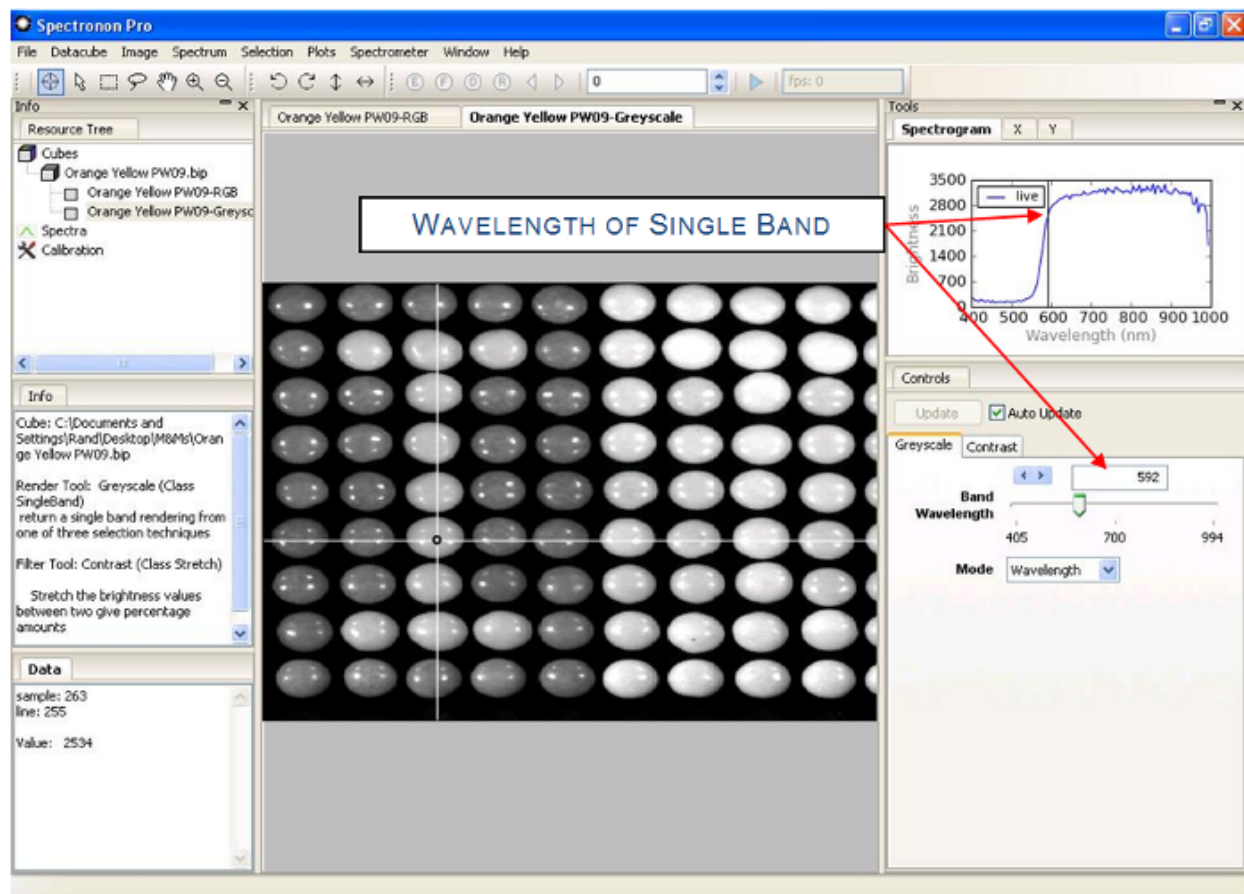
The *Inverse* checkbox is useful if you wish to highlight dark pixels.

The *Individual Bands* checkbox controls whether the brightness levels of the three image layers are considered all together or as individual layers.



It is often useful to view a single band in a standard grayscale (black-and-white) image to visualize the impact of a single spectral feature. To do this, go to the main menu and select *Datacube* → *New Image* → *Greyscale*.

The controls are similar to the RGB controls. If *Auto Update* is not checked, be sure to click *Update* after moving the slider to see the grayscale image for a new band.



As with the RGB images, a vertical line in the Spectral Plot shows the band you have chosen. Note that even though the image is from a single band, the Inspector and ROI Tools will continue to plot and operate on all wavelengths.

Hint: With *Auto Update* selected in the tool control panel, you can quickly scroll through single band images.

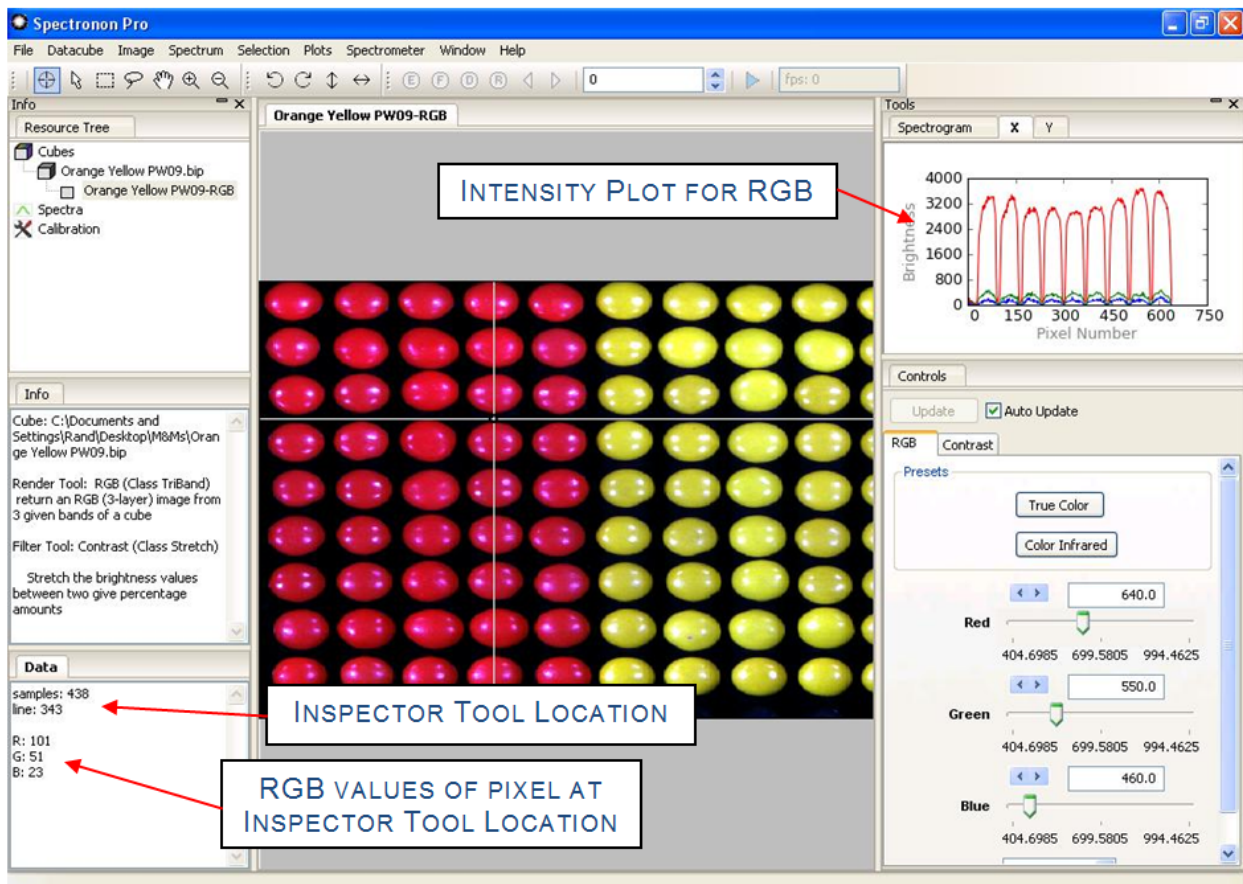
Warning: For large datacubes or slow computers, the *Auto Update* refresh rate may be slow.

5.7 Plot Panel

The plot panel allows you to visualize hyperspectral data graphically. This has already been seen with the use of the Inspector and ROI tools, but here we explore the plot panel in more detail.

The plot panel has three tabs: *Spectra*, *X*, and *Y*. These three tabs provide you with plots along the three axes of a

datacube using the Inspector Tool , as shown below.



Clicking on the *X* and *Y* tabs in the plot panel accesses the corresponding cross-sectional plots. The plot will show the intensity versus position value for the RGB bands used to create the image or the Grayscale band if used with a grayscale image.

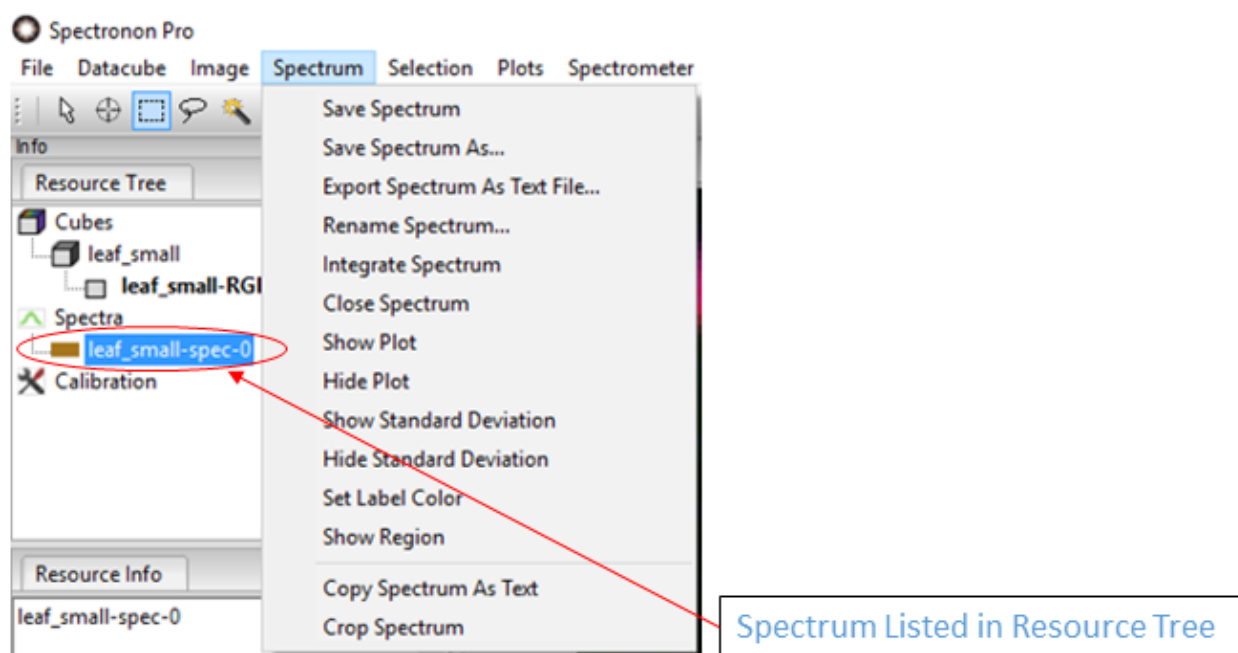
Note: The direction of *X* and *Y* depends on the orientation of your cube. Moving the *Inspector Tool* should reveal which axis you are plotting.

Hint: Use the *magnify tool* , the *demagnify tool* , and the *pan tool*  in the spectral plotter to navigate and examine features.

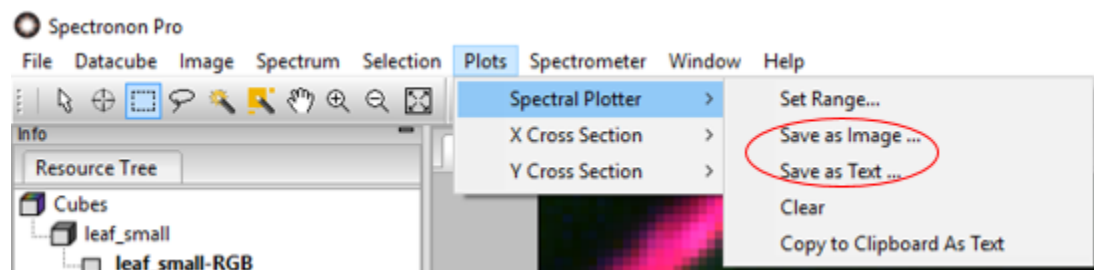
5.8 Saving Spectra, Plots, and Images

Spectronon makes it easy to save the results of your work for further investigations or for making presentations.

To save a spectrum, click the spectrum you wish to save in the *Resource Tree*, and then you may either (1) use the *Spectrum* menu in the main menu, or (2) right-click on the spectrum in the *Resource Tree* to reveal the menu shown below. From this menu, select either *Save Spectrum* or *Save Spectrum As...*. This will open a save dialog. Once saved, the new name will appear when the file is plotted in the spectral plotter, and the file can be re-opened for use in later sessions.

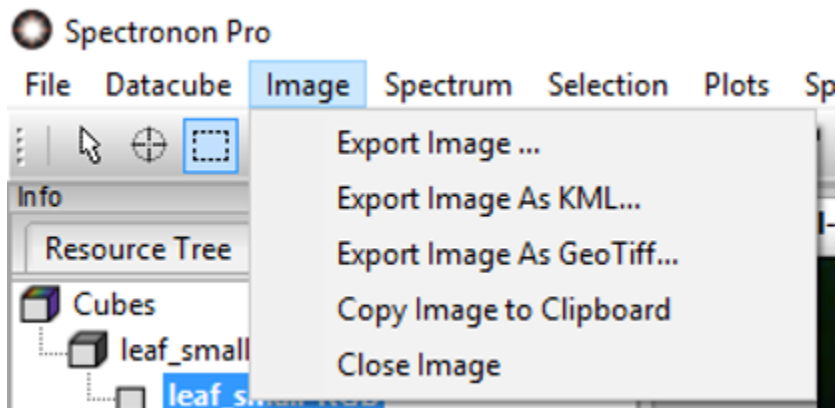


Select the menu option *Copy Spectrum As Text* to copy the data onto your clipboard, from which you can paste it into other applications such as Notepad and Excel.



To save a plot use the *Plots* menu as shown above. Select which plot you wish to save (*Spectral Plotter*, *X Cross Section*, or *Y Cross Section*), and then select *Save as Image* to save as an image or *Save as Text* to save the plotted data as tables in text file. Both options will pop up a save Dialog.

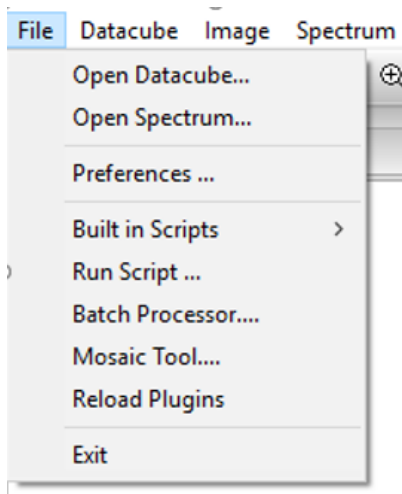
To Save an Image, select *Image* from the main menu, and then *Export Image...* This will pop up a save dialog.



ADVANCED DATA ANALYSIS 1: GENERAL

Spectronon's data visualization and analysis tools are presented in this section, following the structure of the main menu.

6.1 File Menu



File → Open Datacube...

Open an ENVI® compatible datacube file. The datacube is added to the *resource tree* and an image is generated in the image panel.

Hint: Dragging a .bil, .bip or .bsq datacube file into Spectronon will open it.

File → Open Spectrum...

Opens a saved spectrum file.

Hint: Dragging a .spec spectrum file into Spectronon will open it.

File → Preferences...

Opens a window that allows you to set a variety of preference options. Options associated with data visualization can be found under the *Plotters* and *workbench* tabs at the top of the window.

The *Plotters* tab opens a window that allows you to control the presentation of the plots presented in the spectral plotter. Additionally, this window also allows you to set the parameters for exporting plot data so it can be manipulated or plotted using other software tools.

The *workbench* tab opens a window that allows you to set the default image preference. One of the most useful settings is *RGB*, which generates a Red-Green-Blue image based on the values of three chosen hyperspectral bands, which you can choose in the tool control panel. Selecting the *True Color* button in the tool control panel produces an image that approximates the colors you would see looking at the object. Adjusting the sliders allows you to generate false-color images. Be sure to click *Update* after adjusting the sliders. *Greyscale* is another useful option, which presents a greyscale (black and white) image based on a single hyperspectral band. Again, you can adjust this band using sliders in the tool control panel. Be sure to click *Update* after moving a slider.

Hint: Check the *Auto Update* box in the tool control panel so you do not have to continually click *Update*.

File → Built in Scripts

Spectronon enables you to run scripts to best suit your application. A variety of scripts are built in to Spectronon.

File → Run Script...

You may also run Python scripts you write. *File → Run Script* opens a window that allows you to browse to your script.

File → Batch Processor...

For information on Batch Processing your data go to [Section 9: Batch Processing](#).

File → Mosaic Tool...

The mosaic tool is designed for an airborne application and requires georegistered data to operate. This allows the user to create one image by merging several individual images from a raster dataset.

File → Reload Plugins

When making changes to user-written plugins, clicking this will make Spectronon aware of the changes.

File → Exit

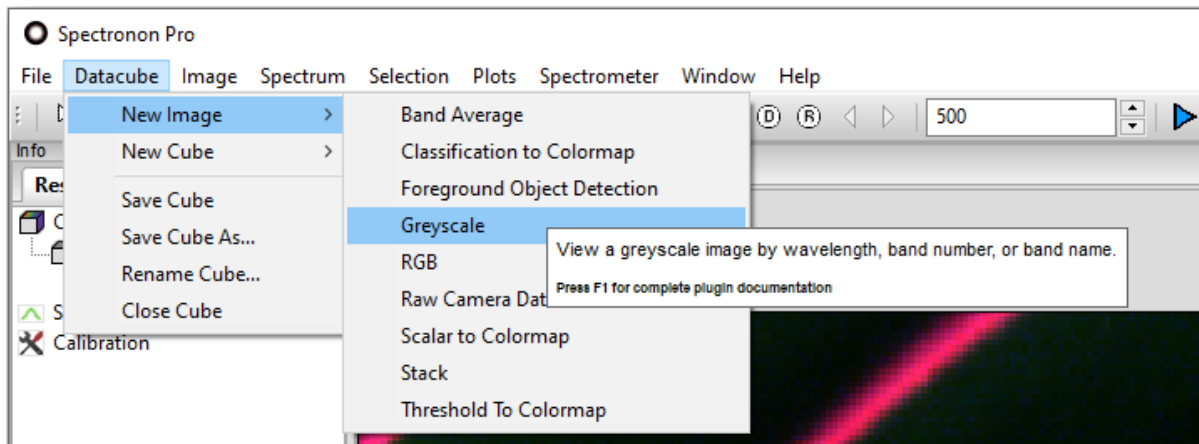
Closes Spectronon.

6.2 Datacube Menu

6.2.1 New Image

Allows you to create a new image in the Image panel. The items in this submenu are implemented as plugins. See [Section 11.2: Render Plugins](#) for full documentation.

The most generally useful options are RGB, Greyscale and Stack.



Hint: If the *New Image* menu is greyed-out, make sure a datacube (not an image) is selected in the *resource tree*.

Datacube → New Image → RGB

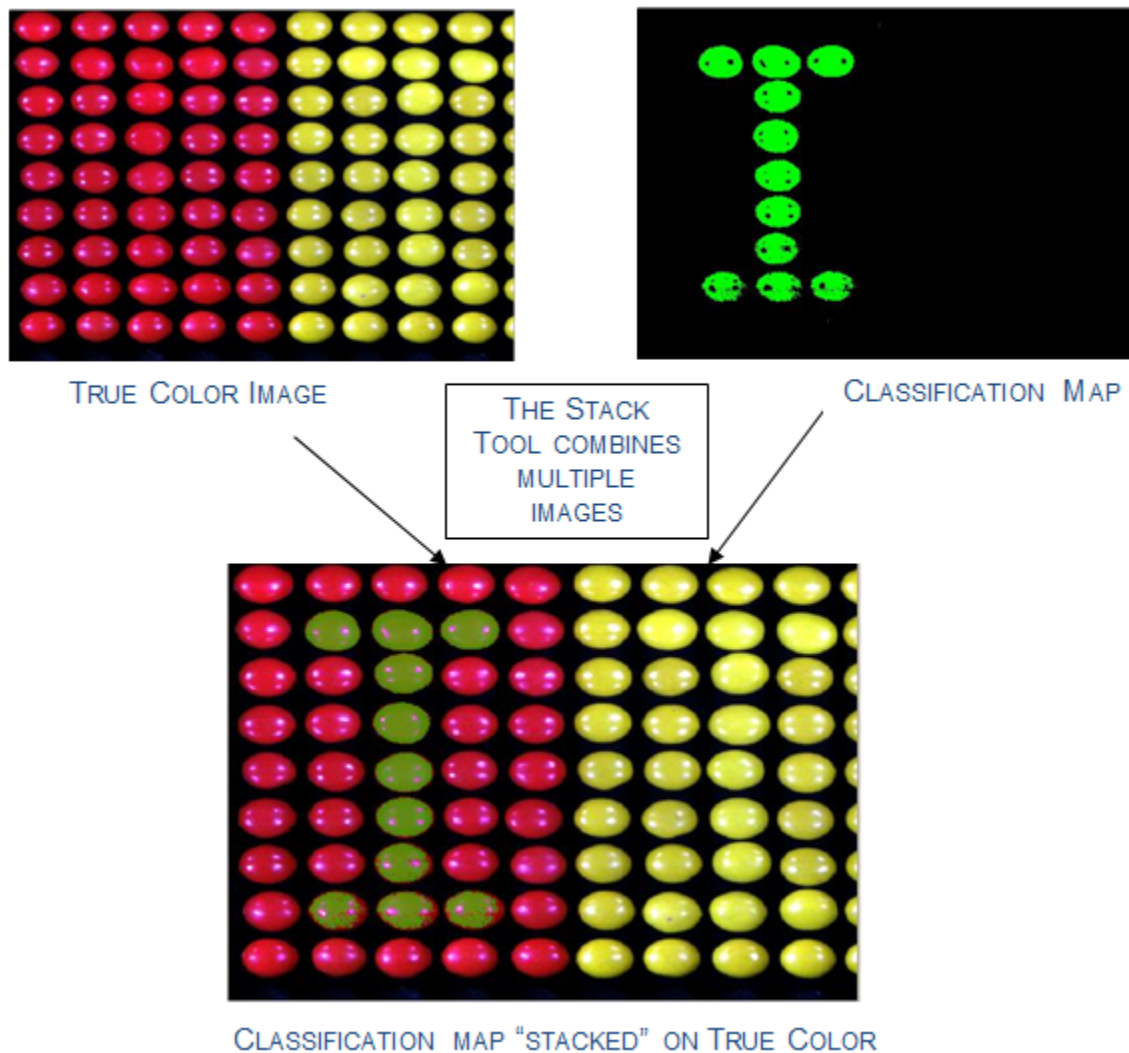
Generates a color image based on three bands of your hyperspectral data. These bands can be chosen and adjusted using sliders in the tool control panel. See [Section 11.2.7: RGB](#).

Datacube → New Image → Greyscale

This produces a single-band image from a single hyperspectral band that can be chosen using a slider in the tool control panel. See [Section 11.2.4: Greyscale](#).

Datacube → New Image → Stack

This tool enables you to overlay multiple images. This tool is particularly useful for presenting classification results. For example, we may have a True Color RGB image of an object, such as shown below, as well as a classification map of one of the candy types. Using the Stack tool, we can combine these images to show the classification map on top of the RGB True Color image.



The stack tool opens a *Stack* tab in the tool control panel. Use the slider to select how many images you wish to combine (*Stack height*). Then click *Update*. Pull-down menus appear that allow you to select the images you wish to combine. A slider (*Alpha*) is available for each image that allows you to set its transparency in the combined image. Once set, click *Update* and your combined image will appear.

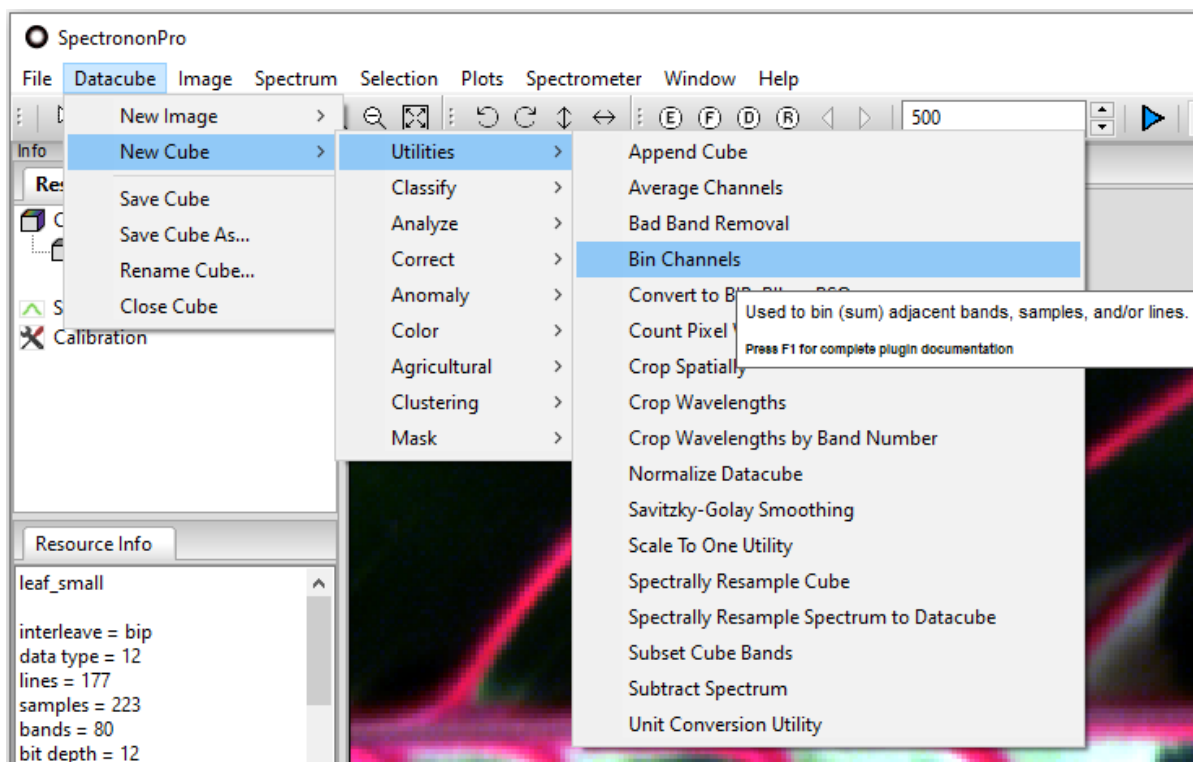
See Section 11.2.10: *Stack*.

6.2.2 New Cube

Datacube -> *New Cube* allows the user to create a new datacube from an existing datacube. The items in this submenu are implemented as plugins. See *Built-in Plugins: Cube Plugins* for full documentation.

Utilities

Allows you to generate a new, modified datacube from the currently open datacube. A few of the more commonly used utilities are explained below. See [Section 11.1.1: Utilities](#) for full documentation.



Datacube → *New Cube* → *Utilities* → *Bin Channels*

Generates a new cube by binning spectral and/or spatial channels, which is often useful to either reduce the size of the datacube or improve the signal-to-noise ratio. This choice creates a new image in the Images Window and also generates a Bin Cube tab in the tool control panel with 3 sliders. The *Sample Bin* and *Line Bin* allow you to bin pixels along a spatial (x,y) axis. The Sample axis refers to the cross-track axis of the imaging spectrometer, and the Line axis refers to the along-track axis of the imaging spectrometer. The *Spectral Bin* slider allows you to bin spectral channels, and is likely the most useful of the binning options. After clicking *Update*, a new binned datacube is generated.

Note: This is a summation, not an average. The datatype of the returned cube may be promoted as required. If you choose *Float Mode* floating-point data will be returned.

See [Section 11.1.1: Bin Channels](#).

Datacube → *New Cube* → *Utilities* → *Crop Wavelengths*

Generates an image in the Image panel and a new tab with sliders in the tool control panel that allow you to crop wavelength bands by choosing a new minimum and maximum wavelength within the cube. Once chosen, click the *Update* button.

Hint: You may want to click on the RGB tab in the tool control panel to reset the bands used to create the image after cropping wavelengths.

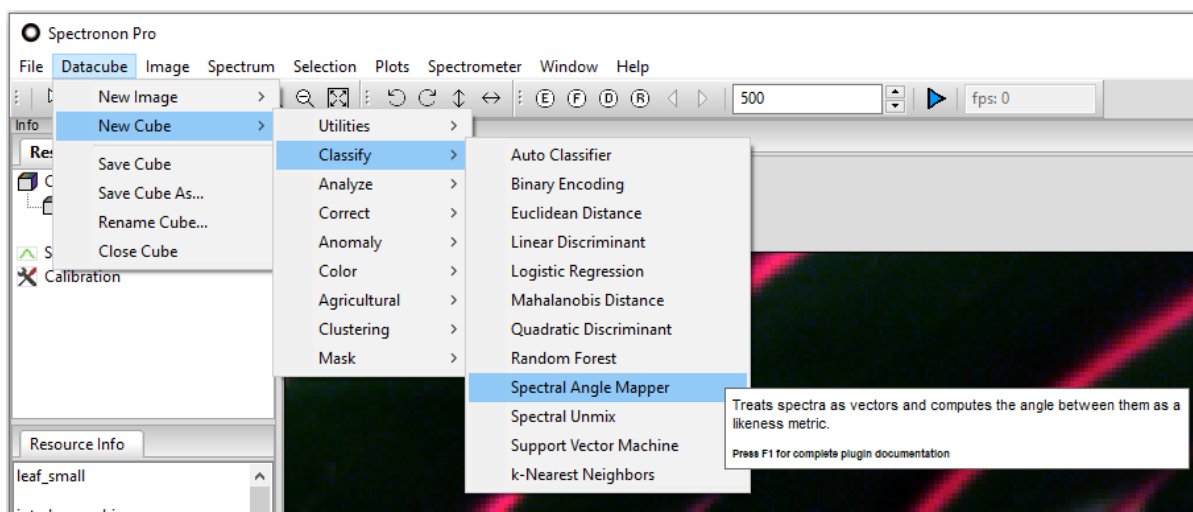
See [Section 11.1.1: Crop Wavelengths](#).

Datacube → New Cube → Utilities → Subtract Spectrum

Generates a new cube by subtracting a background spectrum from all pixels in the datacube. This option is useful, for example, if you are monitoring fluorescent dyes and wish to subtract the background fluorescence of the substrate. See [Section 11.1.1: Subtract Spectrum](#).

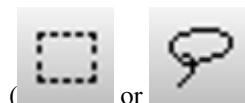
Classify



Allows you to generate classification maps of different objects within your hyperspectral data. More information on hyperspectral classification can be found in [Section 7: Advanced Data Analysis 2: Hyperspectral Classification](#). Also see [Section 11.1.2: Classify](#) for full classification plugin documentation.



Some of the classification algorithms (SAM, Euclidean Distance, Spectral Unmix) use Spectrum objects as inputs, while the statistics-based classifiers (Logistical Regression, Quadratic Discriminant, Support Vector) require datacubes of the classification target. These are made by selecting the ROI with the *lasso* or *marquee* tool, then selecting *Create Cube from Selection*.

When you select one of the classification options, (e.g. SAM), a new cube will be generated in the datatree panel. In many plugins, you will need to specify how many Layers you wish to classify (i.e., how many materials you wish to classify) – you will then need to select a Spectrum or Cube for each of these Layers in a pull-down menu. The Spectra and Cubes available in the pull-down menu can be generated either by using one of the ROI tools, *marquee* or *lasso*



( or ) or by loading a previously saved spectrum or cube.

Note: a detailed discussion of hyperspectral data classification is beyond the scope of this document. Spectronon provides some of the more commonly used algorithms. For more advanced algorithm capability, please see other packages such as ENVI®. Additionally, custom algorithms can be utilized with Spectronon using the custom plugin capability described in the Spectronon custom plugin manual.

Datacube → New Cube → Classify → Binary Encoding Classification

The binary encoding classification technique encodes the data and endmember spectra into zeros and ones, based on whether a band falls below or above the spectrum mean, respectively. See [Section 11.1.2: Binary Encoding](#).

Datacube → New Cube → Classify → Euclidian Distance

This is a commonly used classification algorithm for hyperspectral data that is more sensitive to pixel brightness than SAM. See [Section 11.1.2: Euclidean Distance](#).

Datacube → New Cube → Classify → Spectral Angle Mapper (SAM)

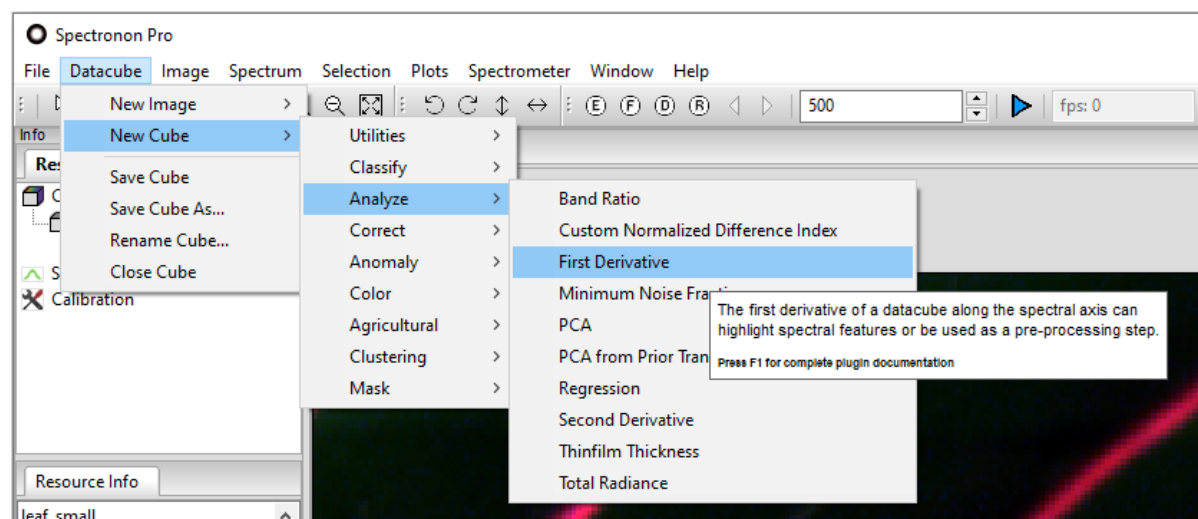
The SAM classification routine is described in detail in [Section 7.2: Spectral Angle Mapper \(SAM\) Classification](#). Also see [Section 11.1.2: Spectral Angle Mapper](#).

Datacube → New Cube → Classify → Spectral Unmix


Spectral unmixing deconvolves the signal in each pixel into a linear combination of known spectra. See [Section 11.1.2: Spectral Unmix](#).



Analyze

Datacube → New Cube → Analyze allows you to perform useful operations on hyperspectral data that will generate new datacubes based on applying analytical functions to your datacube. A few of these are documented here. See [Section 11.1.3: Analyze](#) for full documentation.



Datacube → New Cube → Analyze → First Derivative

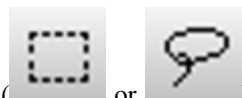
This tool generates a new datacube with the first derivative of the spectral curve for each pixel. A new image of the first derivative datacube is presented in the Image panel. To see the spectral derivatives, utilize the *inspector tool* ,

or use one of the ROI tools *marquee* or *lasso* ( or ). The first derivative curves appear in the spectral plotter. See [Section 11.1.3: First Derivative](#).

Datacube → New Cube → Analyze → Principal Component Analysis (PCA)

A detailed discussion of PCA is beyond the scope of this document (see, for example, Wikipedia for a discussion). This tool generates a new datacube and image with the principal component values for each pixel. Additionally, a *PCA* tab appears in the tool control panel with a slider that allows you to select the number of Bands, or PCA components.

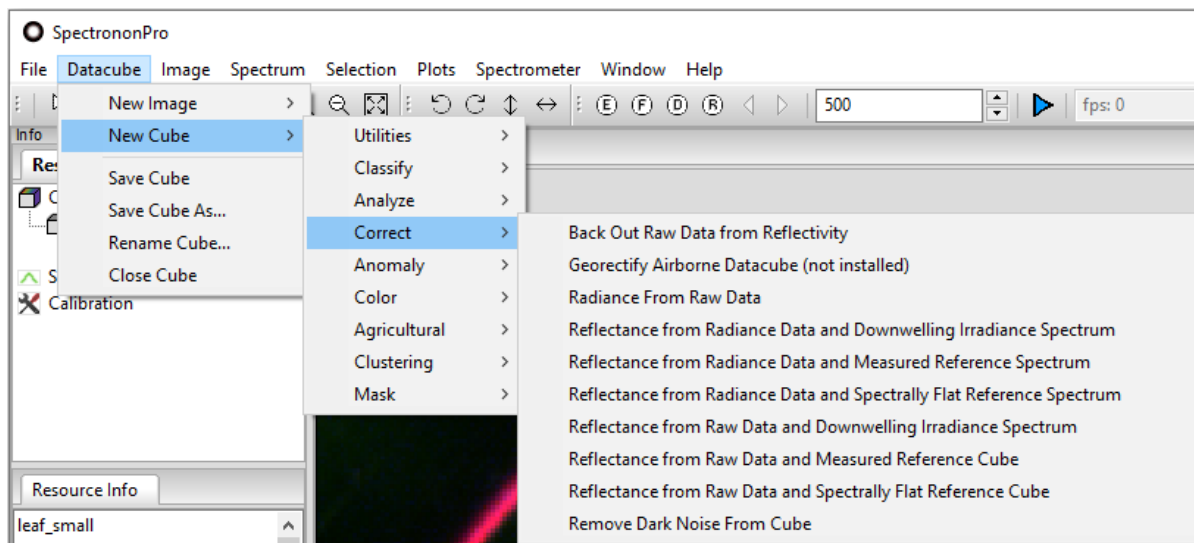
To see the PCA component magnitudes, utilize the *inspector tool* , or use one of the ROI tools *marquee* or *lasso*.



(or). The PCA magnitude curves appear in the *spectral plotter*. As with standard hyperspectral data, classification algorithms can be performed on the new PCA datacube. See [Section 11.1.3: PCA](#).

Correct

Allows you to correct / calibrate a datacube from a measured reference. This tool also allows you to convert your data to either radiance or reflectance values, as described below. See [Section 11.1.4: Correct](#) for full documentation.



Datacube → New Cube → Correct → Radiance From Raw Data

Converts raw data to units of radiance (microflicks) based on the instrument's radiometric calibration file (.icp). See [Section 11.1.4: Radiance From Raw Data](#).

Datacube → New Cube → Correct → Reflectance from Radiance Data and Downwelling Irradiance Spectrum

Converts radiance data to reflectivity based on a downwelling irradiance spectrum collected with a single point spectrometer and cosine corrector. See [Section 11.1.4: Reflectance from Radiance Data and Downwelling Irradiance Spectrum](#).

Datacube → New Cube → Correct → Reflectance from Radiance Data and Measured Reference Spectrum

This approach to converting data to reflectance assumes the input datacube has been corrected spatially, typically by converting to radiance but other approaches would be acceptable. A spectrum of measured reference material is used to correct the datacube spectrally. See [Section 11.1.4: Reflectance from Radiance Data and Measured Reference Spectrum](#).

Datacube → New Cube → Correct → Reflectance from Radiance Data and Spectrally Flat Reference Spectrum

This approach assumes the input datacube has been corrected spatially, typically by converting to radiance but other approaches would be acceptable. A spectrum of spectrally flat reference material (typically Spectralon™ or Fluorilon™) is used to correct the datacube spectrally. See [Section 11.1.4: Reflectance from Radiance Data and Spectrally Flat Reference Spectrum](#).

Datacube → New Cube → Correct → Reflectance from Raw Data and Downwelling Irradiance Spectrum

Converts raw data to reflectivity based on a downwelling irradiance spectrum collected with a single point spectrometer and cosine corrector. See [Section 11.1.4: Reflectance from Raw Data and Downwelling Irradiance Spectrum](#).

Datacube → New Cube → Correct → Reflectance from Raw Data and Measured Reference Cube

Convert raw data to reflectance based on a input datacube of measured reference material. See [Section 11.1.4: Reflectance from Raw Data and Measured Reference Cube](#).

Datacube → New Cube → Correct → Reflectance from Raw Data and Spectrally Flat Reference Cube

Converts raw data to reflectivity based on another datacube of a spectrally flat reference material. See [Section 11.1.4: Reflectance from Raw Data and Spectrally Flat Reference Cube](#).

Anomaly

Anomaly detection tries to determine anomalous outliers in a datacube that do not conform to the expected spectra. This is usually based on a datacube of ‘clutter’, which contains the expected spectra.

See [Section 11.1.5: Anomaly](#) for full documentation.

Color

Allows you to transform your hyperspectral data into CIE colorspace, providing XYZ, xyY, and LAB values for each pixel. Additionally, you can determine the ΔE values for each pixel as compared to a standard set by the user.

See [Section 11.1.6: Color](#) for full documentation.

Agricultural

Create maps of Hyperspectral Vegetation Indices (HVI's). HVI's in Spectronon are discussed in greater depth in [Section 8](#). Also see [Section 11.1.7: Agricultural](#).

Clustering

Clustering tools are unsupervised classification routines that do not require any a priori information of a spectral image. They group the scene into clusters based on spectral similarity. Supervised classification routines can almost always produce better results, if scene knowledge available. See [Built-in Plugins: Cube Plugins - Clustering](#) for full documentation.

6.2.3 Datacube Menu Misc.

Datacube → Save Cube

Saves the selected datacube.

Datacube → Save Cube As

Saves the currently open datacube under a new name or location.

Datacube → Rename Cube

Renames the selected datacube.

Datacube → Close Cube

Removes the selected datacube from the *Resource Tree*.

6.3 Image Menu

Tools for saving or closing images generated from a datacube.

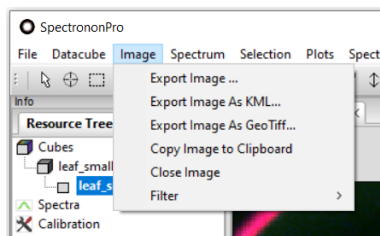


Image → Export Image

Exports an image as TIFF, PNG, BMP, JPG, or GIF.

Export Image as KML

Exports a georectified image in Keyhole Markup Language (KML) format.

Export Image as GeoTiff

Exports a georectified image as a TIFF file.

Copy Image to Clipboard

Copies an image to the Windows clipboard.

Close Image

Closes the selected image.

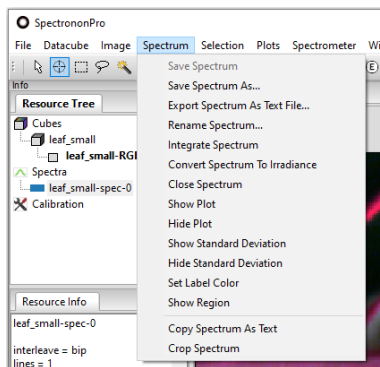
6.3.1 Filter

Changes the filter applied to a generated image. Filters alter the image presentation in some way. Spectronon usually chooses the most appropriate filter automatically. Most commonly this is the *contrast* filter.

All filter menu items are implemented as plugins. See [Section 11.3: Filter Plugins](#) for full documentation.

6.4 Spectrum Menu

The Spectrum menu provides tools for manipulating individual spectra.



Hint: If the Spectrum menu is greyed out, verify that a spectrum is selected in the resource tree.

Spectrum → Save Spectrum

Saves the spectrum selected in the *Resource Tree*.

Spectrum → Save Spectrum As

Saves a spectrum in a file name of your choice.

Spectrum → Export Spectrum as Text File

Saves a spectrum as a text file.

Spectrum → Rename Spectrum

Renames a spectrum.

Spectrum → Integrate Spectrum

Computes the area under the spectral plot for a given spectrum.

Spectrum → Convert Spectrum To Irradiance

Converts a spectrum to irradiance. A downwelling calibration pack is required.

Spectrum → Close Spectrum

Closes the spectrum selected in the *Resource Tree* and removes the associated plot from the spectral plotter.

Spectrum → Show Plot

Plots the spectral curve of the selected spectrum in the *spectral plotter*.

Spectrum → Hide Plot

Hides the spectral curve plot in the *spectral plotter*.

Spectrum → Show Standard Deviation

Shows the standard deviation envelope in the *spectral plotter*.

Spectrum → Hide Standard Deviation

Hides the standard deviation envelope in the *spectral plotter*.

Spectrum → Set Label Color

Changes the color of the selected spectrum's spectral curve and generated classification maps.

Spectrum → Show Region

Highlights the datacube region used to generate the selected spectrum.

Spectrum → Copy Spectrum as Text

Copies a spectrum's spectral data to the Windows clipboard.

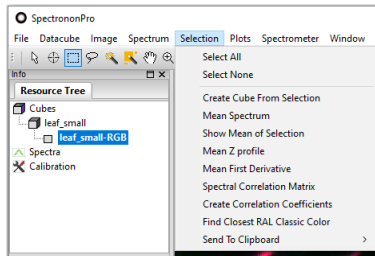
Spectrum → Crop Spectrum

Select a spectrum, then select a region in the spectra window, and then click *Spectrum → Crop Spectrum*. The spectral plot will be cropped to the chosen range.

6.5 Selection Menu



The *Selection* menu items provide a number of options for use with the selection tools (, , or). For all of the following options, you must first select a **Region of Interest (ROI)**.



Note: The *Selection* menu can be accessed from the main menu or by right-clicking in the image after creating an ROI.

Note: Some of these menu items are implemented as selection plugins. See [Section 11.4: Select Plugins](#) for references.

Selection → *Select All*

Selects the entire dataset.

Selection → *Select None*

Clears the current selection.

Selection → *Create Cube From Selection*

Creates a new datacube from the current selection.

Selection → *Mean Spectrum*

This tool will calculate and plot the mean spectrum for the pixels selected in your ROI. See [Section 11.4.3: Mean Spectrum](#).

Selection → *Show Mean of Selection*

Show a dialog box with the mean of each band in the selected region. See [Section 11.4.4: Show Mean of Selection](#).

Selection → *Mean Z profile*

Create and plot the mean profile of the selected region in the Z (band) dimension. If wavelength metadata is present this is equivalent to the Mean Spectrum tool. See [Section 11.4.5: Mean Z profile](#).

Selection → *Mean First Derivative*

Plot the mean first derivative (discrete difference) of the selected area. See [Section 11.4.6: Mean First Derivative](#).

Selection → *Spectral Correlation Matrix*

Creates a heatmap showing band-to-band correlations using the Pearson coefficient. See [Section 11.4.7: Spectral Correlation Matrix](#).

Selection → *Create Correlation Coefficients*

A correlation coefficient aligns reflectance data created with downwelling irradiance data to a ground truth target such as a tarp or reflectance standard. Once created, these coefficients are used in the Reflectance Conversion with Downwelling Irradiance plugins to improve the accuracy of the result. See [Section 11.4.8: Create Correlation Coefficients](#).

Selection → Find Closest RAL Classic Color

Uses spectral information to determine the closest RAL Classic color to selected region. Wavelength and reflectance scale factor metadata must be present. Useful for color matching. See [Section 11.4.9: Find Closest RAL Classic Color](#).

Selection → Median Spectrum

Plot the median spectrum within selected area. See [Section 11.4.10: Median Spectrum](#).

Selection → Mean Nth Derivative

Compute and plot the mean Nth derivative (discrete difference) of the selected region. See [Section 11.4.11: Mean Nth Derivative](#).

Selection → Save random spectra to text file

Randomly selects pixels within the region of interest and exports their spectra (or bands) to a text file. See [Section 11.4.12: Save random spectra to text file](#).

6.5.1 Send to Clipboard

This tool allows you to transfer the data within your selected ROI. The first option is *Copy Mean as Text*, which allows you to paste the mean spectrum values into other applications such as Notepad or Excel. The second option, *All Spectra as Text* copies the spectral curves of all the pixels within your selected ROI. This tool is useful for those who want to perform specific statistical analysis on the data from small regions.

Selection → Send to Clipboard → Copy Mean as Text

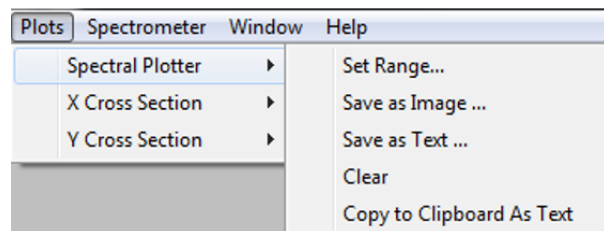
Copies mean spectrum as text to clipboard for pasting into other applications. See [Section 11.4.13: Copy Mean as Text](#).

Selection → Send to Clipboard → All Spectra as Text

Copies all spectra in region as text to clipboard for pasting into other applications. See [Section 11.4.13: All Spectra as Text](#).

6.6 Plots Menu

The *Plots* menu items provide tools for the data shown in the *plot panel*.



Plots → [plot type] → Set Range

Set X-axis and the Y-axis ranges.

Plots → [plot type] → Save as Image

Save a plot to an image file.

Plots → [plot type] → Save as Text

Save plot data values as a .txt file.

Plots → Spectral Plotter → Clear

Clears the spectral plot.

Plots → [plot type] → Copy to Clipboard as Text

Copy plot data to the Windows clipboard as text.

6.7 Datacubes and Header Files

Datacubes collected with Resonon imaging spectrometers require two files, one that contains the hyperspectral data, and also a header file that contains important information about the data. Hyperspectral data can be accessed with many tools, for example, Matlab and ENVI, and is compatible with a variety of languages including: Python, C++, Fortran.

6.7.1 Hyperspectral data files

ENVI-compatible hyperspectral data can be stored in the following three formats:

Band-Sequential In Band-Sequential (BSQ) format each line of the data is followed by the next line in the same spectral band. Hyperspectral data stored in BSQ format will end with .bsq.

Band-Interleaved-by-Line Data saved in Band-Interleaved-by-Line (BIL) format have the first line of the first band followed by the first line of the second band, followed by the first line of the third band, and so forth. This pattern is repeated for subsequent lines. Hyperspectral data stored in BIL format will end with .bil.

Band-Interleaved-by-Pixel Band-Interleaved-by-Pixel (BIP) data have the first pixel for all bands in sequential order, followed by the second pixel for all bands, and so forth for all pixels. Hyperspectral data stored in BIP format will end with .bip.

Resonon imaging spectrometers will save hyperspectral data in BSQ, BIL, or BIP formats, and Spectronon software will open hyperspectral data in all of these formats.

6.7.2 Hyperspectral data header files

Header files for hyperspectral data have the same name as the BSQ, BIL, or BIP files followed by .hdr. The following discussion provides an overview of header files.

Header files can be opened and edited using WordPad. The following list describes some, but not all, of header file commands. Bold indicates the command, quantities in (brackets) are acceptable arguments for the commands, and statements in [square brackets] are additional explanation. Note that the order of these commands can be different from file to file.

ENVI [All ENVI-compatible header files begin with this]

interleave = (bil, bip, bsq) [Tells the software the hyperspectral data format]

data type = (4, 12) [4 is 4-byte floating point; 12 is 2-byte unsigned ; other data types are also supported, but these are common]

lines = (###) [The number of lines in the hyperspectral data where ### is a number]

samples = (###) [The number of samples in the hyperspectral data where ### is a number. This is typically the number of cross-track spatial pixels of the hyperspectral imager.]

bands = (###) [The number of bands in the hyperspectral data where ### is a number]

bit depth = (12, 14) [The bit depth of the hyperspectral data]

shutter = (###) [The shutter time in units of msec where ### is a number. This is mostly for reference, but it may be used by certain plugins]

gain = (#.#) [The camera gain in dB where #.# a number. This is mostly for reference, but it may be used by certain plugins]

framerate = (####) [The framerate at which the data were recorded in frames/sec where ### is a number. This is mostly for reference, but it may be used by certain plugins]

reflectance scale factor = (4095, 1023, 1) [If you divide the data by this factor, the data will scale from 0 to 1, typically for reflectance data]

byte order = (0) [byte order is the ordering of byte significance (<http://en.wikipedia.org/wiki/Endianness>) Data for Spectronon are LSF or Little Endian which means the least significant bytes come first. Spectronon will not properly open BSF or Big Endian files]

header offset = (0) [This is the number of bytes at the beginning of the binary file that should be ignored].

wavelengths = (###, ###, ###, ...) [List of the spectral band centers in units of nm]

rotation = ((#,#), (#,#), (#,#), (#,#)) [This is a default orientation of the view of a datacube in Spectronon. This is a Spectronon only header option].

label = (name) [The label is used by Spectronon to give a human readable alternative to the file name of a file. If omitted, Spectronon will use the file name]

timestamp = (day & time of datacube) [If available, this is when the datacube was recorded. The timestamp is often found on airborne datacubes, and datacubes where a script entered this information explicitly into the header]

The following are for spec.hdr files only, headers for spectrum. Spectrum files are ENVI compatible because they are 1x1 datacubes, but Spectronon adds in these data for its own use.

original cube file = (name of original cube) [This is a Spectronon-only header item for recording the history of a generated spectrum. In the case of a derived datacube, you will see a “history” header value that shows the way this cube was generated]

pointlist = (long list of values) [This is the set of points from the original cube that were averaged together to make this spectrum if the spectrum is a mean]

label color = (#FF00FF) [This is the color used to plot the spectrum]

ADVANCED DATA ANALYSIS 2: HYPERSPECTRAL CLASSIFICATION

7.1 General Approach

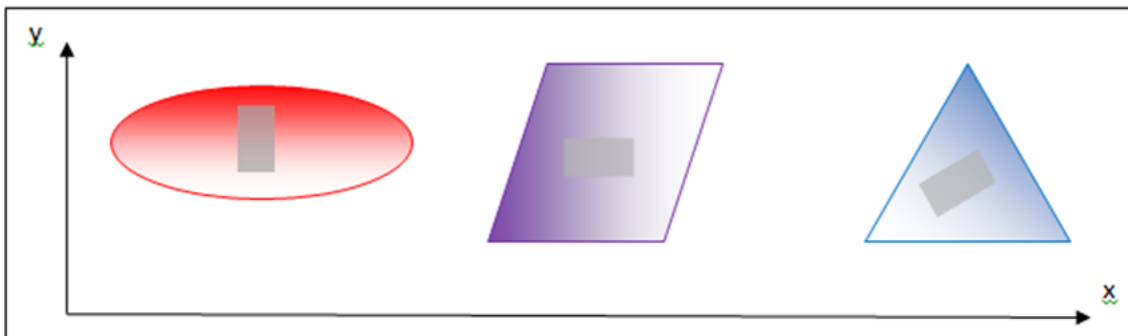
The detailed spectral information in hyperspectral data enables one to distinguish between very similar objects. Additionally, machine vision algorithms enable rapid, accurate, and repeatable classification of objects. With these capabilities, hyperspectral imaging has a broad range of current and potential applications, including sorting (food, raw materials, recycled materials), quality control (pharmaceuticals, food, printed goods), remote sensing (defense, search & rescue, mineral exploration, agriculture), to name just a few.

Critical to all of these applications is implementation of algorithms that classify the pixels within an image based on their spectral profiles. This section provides an introduction to hyperspectral data classification. There are many approaches to classifying objects. However, the general approach to most of these algorithms can be understood by considering the following simplified hypothetical examples.

In general, the algorithms used to classify objects scale with the number of spectral channels. Understanding how the general approaches work with a simple 2-color system makes it relatively easy to understand hyperspectral classification with tens or hundreds of channels. Therefore, consider a 2-color camera that provides digital numbers for how much red and how much blue is in each pixel. (Thus, this example is even easier than a conventional digital color camera that provides digital values for the three colors: red, green, and blue.)

Much like humans need to learn to distinguish between objects, classification algorithms generally also must be trained. Typically this is done by imaging samples of interest, and then using results from this “training set” to learn how to distinguish between objects in general.

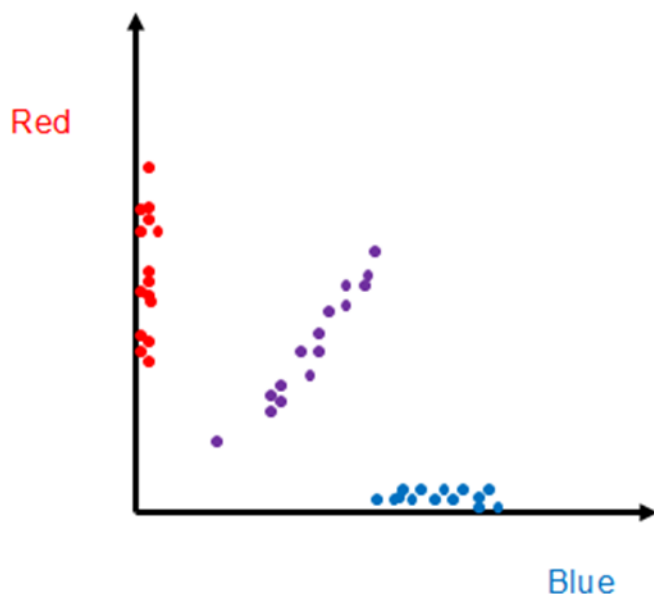
As an example, assume we wish to distinguish between the three objects shown below, a red ellipse, a blue triangle, and a purple (mixture of red and blue) parallelogram. (Ignore the grey rectangles for now.) An x and y axis is drawn to provide location coordinates for each pixel.



Note that each object has a distribution of light and dark pixels, although each object is approximately the same “color”. If we image these objects with our hypothetical 2-color imager that senses only “red” and “blue” channels, there will be two spectral channels per pixel. To train our imaging system, we first select a representative set of training pixels

from the image of each of the objects of interest. This might be done, for example, by selecting the pixels within each grey rectangle indicated in the image above.

A useful way to visualize the color information in these training pixels is to plot the red and blue brightness values for each selected pixel in “color space” along blue and red axes, as shown below. (Note: This color plot uses only the training pixels selected within the three grey rectangles shown in the image above.)

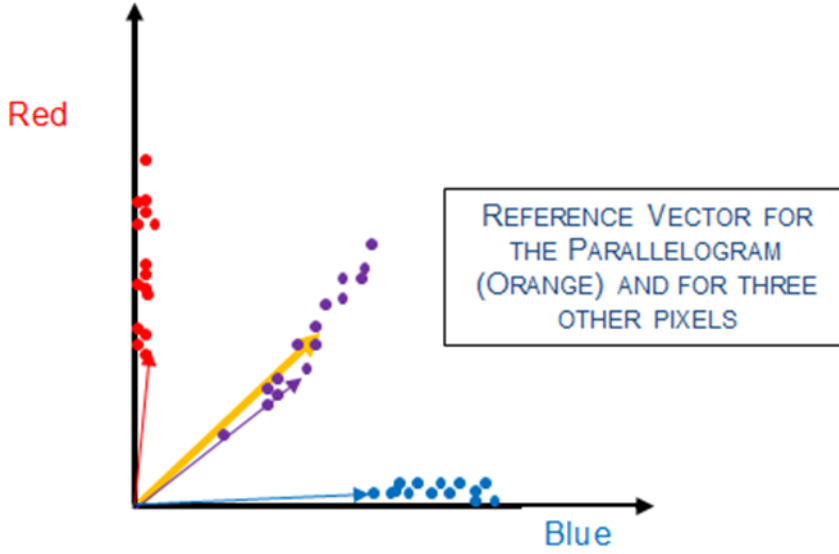


We can immediately recognize the training pixels that align along the vertical (red) axis are associated with the red ellipse, as those pixels clearly have large red brightness values, but small blue brightness. Similarly, the blue triangle pixels lie primarily along the horizontal (blue) axis. The purple pixels, however, lie in between the two axes because the color purple has significant red and blue brightness (i.e., it is a mixture of red and blue). Because some of the pixels are dark and others are light, the distribution of training pixels from each object is spread out from small values to large values in a near-linear manner.

One can see that the pixels from the spectrally distinct objects are separated in this “color space”. Although this example has a 2-dimensional color space, one could create a 3-dimensional color space for RGB cameras, or 100 dimensions for 100-band hyperspectral imagers. In general, additional dimensions provide additional “opportunities” for points from different objects to be distinct in the color space, and thus they are easier to classify in practice, but otherwise the number of dimensions need not concern us now. Admittedly it is difficult to visualize a 100-dimension space, but the mathematical transition is often straight-forward. Consequently, one can invent techniques that work in two dimensions, and they can generally be applied to 100 dimensions.

Consider the following example of the well-known hyperspectral classification algorithm known as Spectral Angle Mapper (SAM). Note that the groups of training pixels from the three different objects in our 2-dimensional color plot are located at different angles relative to the horizontal axis. The SAM technique utilizes this property to classify ALL pixels.

Consider a reference vector at the center of the group of purple training pixels. This vector is shown as a large orange vector below. Similarly, we can envision vectors from the origin to each pixel in the plot (only three are shown to reduce clutter).



One can see that the angle between the reference orange vector is small for the purple pixels, and relatively large for the red and blue pixels. Thus, if we calculate the angle between the reference orange vector and all pixels in the image, we recognize that those pixels with a small angle are from the purple parallelogram, and those pixels with a large angle from the orange reference vector are not purple pixels.

Fortunately there is an easy way to calculate the angle between two vectors by utilizing the vector dot product. To do this, write the orange reference vector in component form as

$$\vec{R} = (R_r, R_b)$$

where the subscript r indicates the red brightness value component and b indicates the blue brightness value component. Similarly, for all pixels in the image, write

$$\vec{P}(x, y) = (P_r(x, y), P_b(x, y))$$

where x and y indicate the location of the pixel in the original image with an ellipse, triangle, and parallelogram. The vector dot product of \vec{R} and any pixel $\vec{P}(x, y)$ is

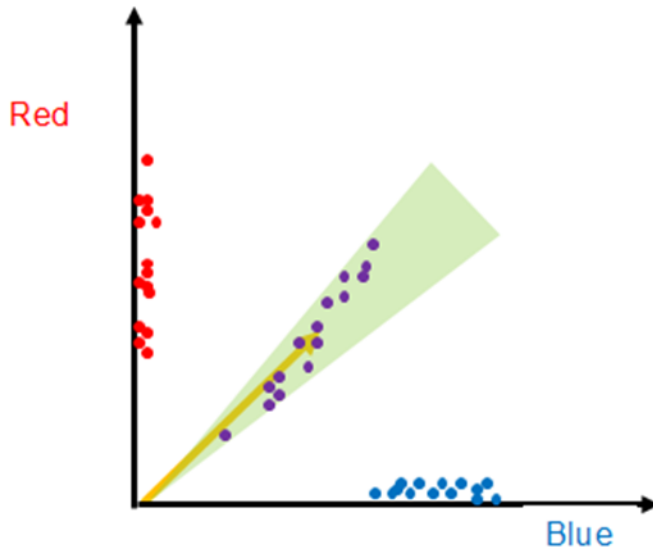
$$\vec{R} \bullet \vec{P}(x, y) = R_r P_r(x, y) + R_b P_b(x, y) = |\vec{R}| |\vec{P}(x, y)| \cos(\theta(x, y))$$

where $|\vec{R}|$ and $|\vec{P}(x, y)|$ are the magnitudes of \vec{R} and $\vec{P}(x, y)$. E.g., $|\vec{R}| = [R_r^2 + R_b^2]^{1/2}$ and $\theta(x, y)$ is the angle between the reference vector \vec{R} and the pixel vector $\vec{P}(x, y)$.

Solving for $\theta(x, y)$ yields

$$\theta(x, y) = \cos^{-1} \left(\frac{\vec{R} \bullet \vec{P}(x, y)}{|\vec{R}| |\vec{P}(x, y)|} \right)$$

By choosing only those pixels that have an angle $\theta(x, y)$ less than some small threshold value, one can effectively identify all the purple pixels. Graphically, this is equivalent to choosing only those pixels within a narrow green cone, as shown below.



By finding the orange reference vector and establishing the threshold of acceptable angles from this vector, we have “trained” our imaging system. To identify objects with the same color as the purple parallelogram, one images the objects and then calculates the angle as described above for each pixel. Pixels with an angle smaller than the threshold are identified to be the same material (color) as the purple parallelogram.

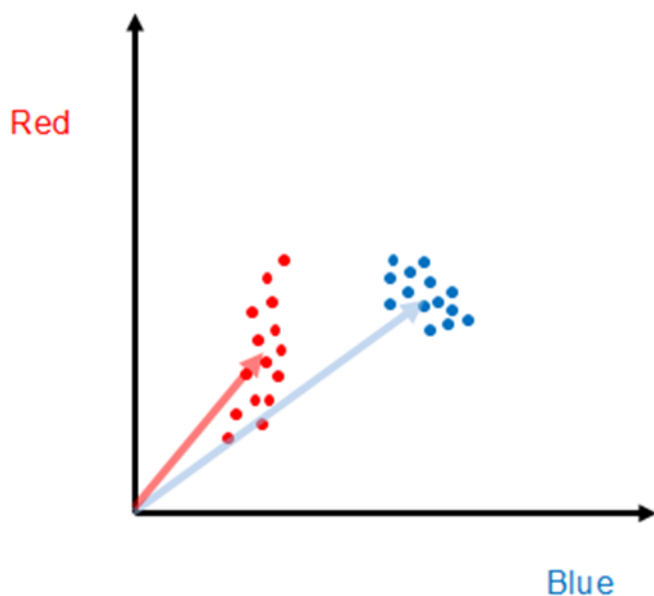
One could also train the system to find pixels the same color as the red ellipse or blue triangle by finding the appropriate reference vector for these objects, thereby enabling one to identify multiple objects within one image by calculating the angle between multiple object reference vectors.

To extend this approach to hyperspectral data with multiple spectral channels, one utilizes the more general definition of a vector dot product

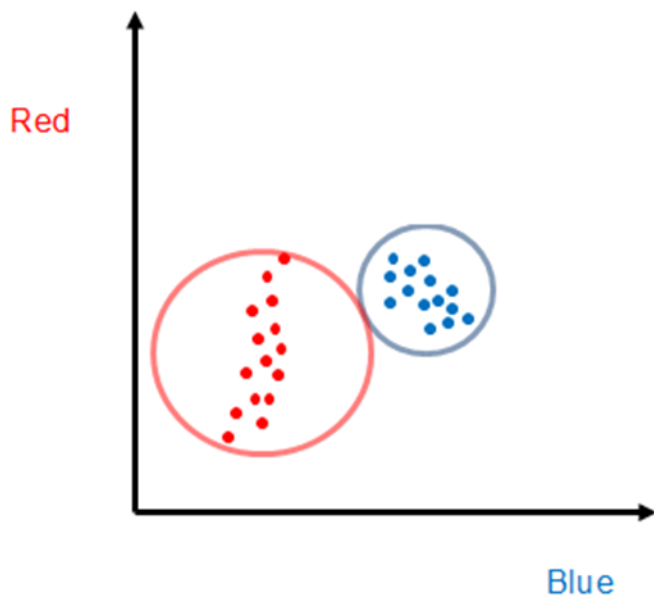
$$\vec{R} \bullet \vec{P}(x, y) = \sum_i R_i P_i(x, y) = |\vec{R}| |\vec{P}(x, y)| \cos(\theta(x, y))$$

where the sum is over all spectral bands (dimensions) indexed by i . The angle $\theta(x, y)$ is still the angle between the two vectors, \vec{R} and $\vec{P}(x, y)$, although now in a complex multi-dimensional color space that is more difficult to visualize – the mathematics and approach are identical.

Of course SAM is not always the best algorithm to use, as one can see with the example below, where pixels from two hypothetical objects we wish to distinguish between are shown as red and blue points. In this case, if one choose representative vectors in the middle of each cluster to perform SAM classification, it is easy to see there would be substantial misclassification. (E.g., two red points lie nearly along the blue representative vector.)



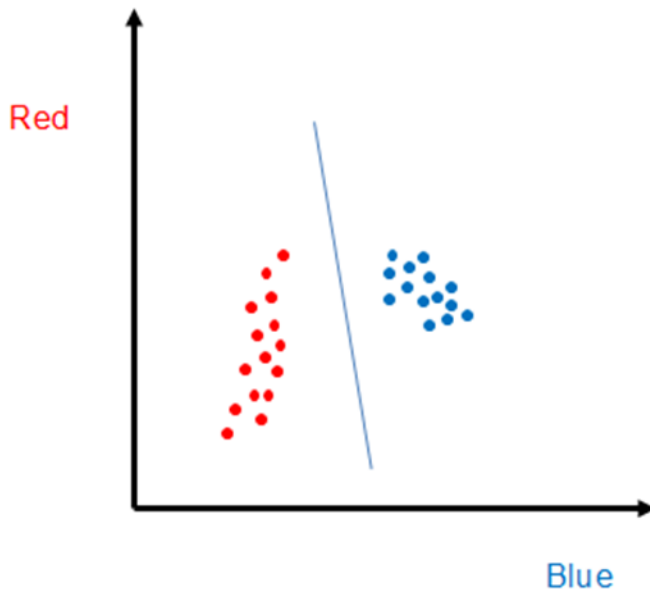
For the case shown above, a different algorithm is needed. One approach that would clearly be superior to SAM, at least for these data, is to find the center points of each set of training points, and then all points within a certain radius would be classified as that object. This is also a well-known hyperspectral classification algorithm known as Euclidean distance. Again, this approach is readily extended to hyperspectral data because the concept of distance between points is well established for multi-dimensional space.



The Euclidean distance approach can readily be improved upon by recognizing that a circle is not the best “enclosure”, and an ellipse whose axes were scaled to the point distribution width would be far better. The hyperspectral classification algorithm that utilizes this information on the distribution of training points is called the Mahalanobis distance approach. As one would expect, the cost of additional algorithm sophistication is often the need for additional processing.

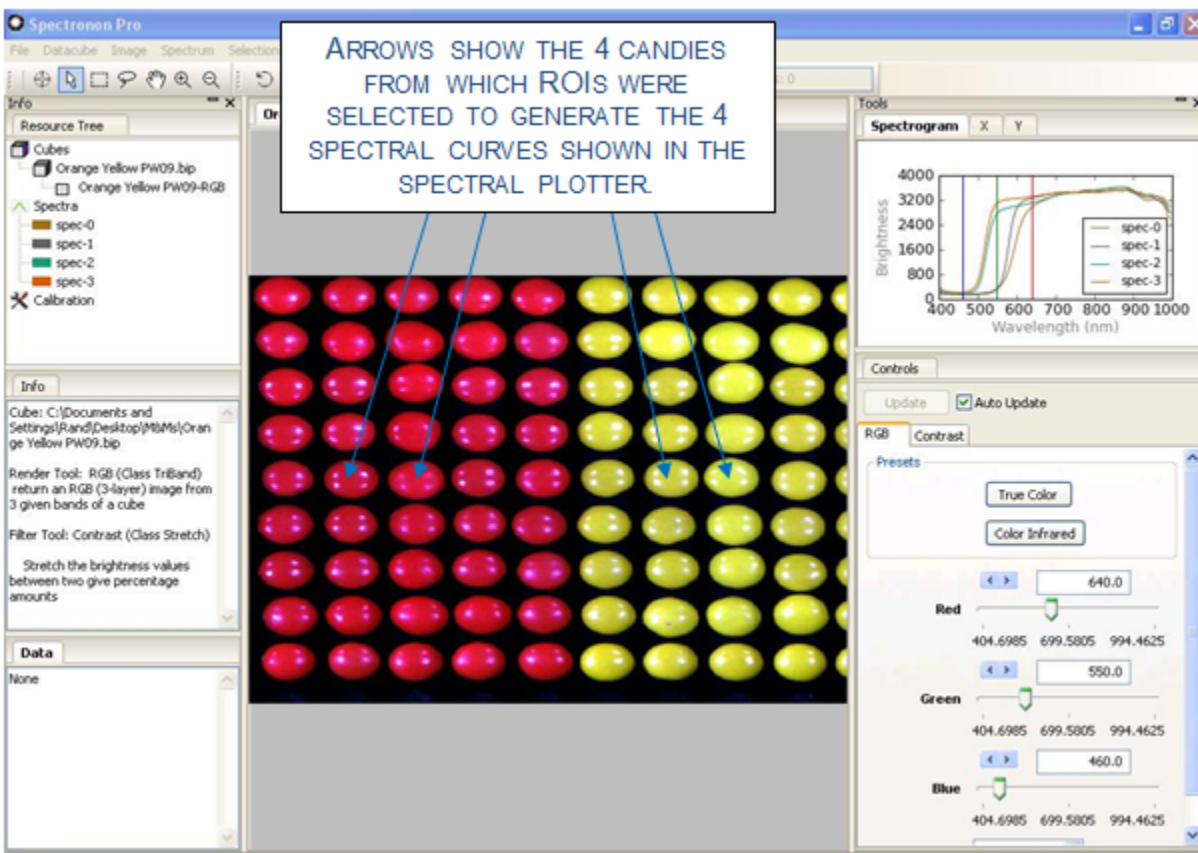
There are, of course, many other approaches to classifying objects using hyperspectral data, such as the one indicated below where one draws a line between the two sets of training points. The system is trained by noting that all pixels

that map to the left of the line are associated with one object, whereas the pixels that map to the right of the line are associated with the other. Extrapolating this approach to higher dimensions is more difficult, as the line becomes a plane in three dimensions, and a hyper-plane in color spaces with dimensions larger than three.

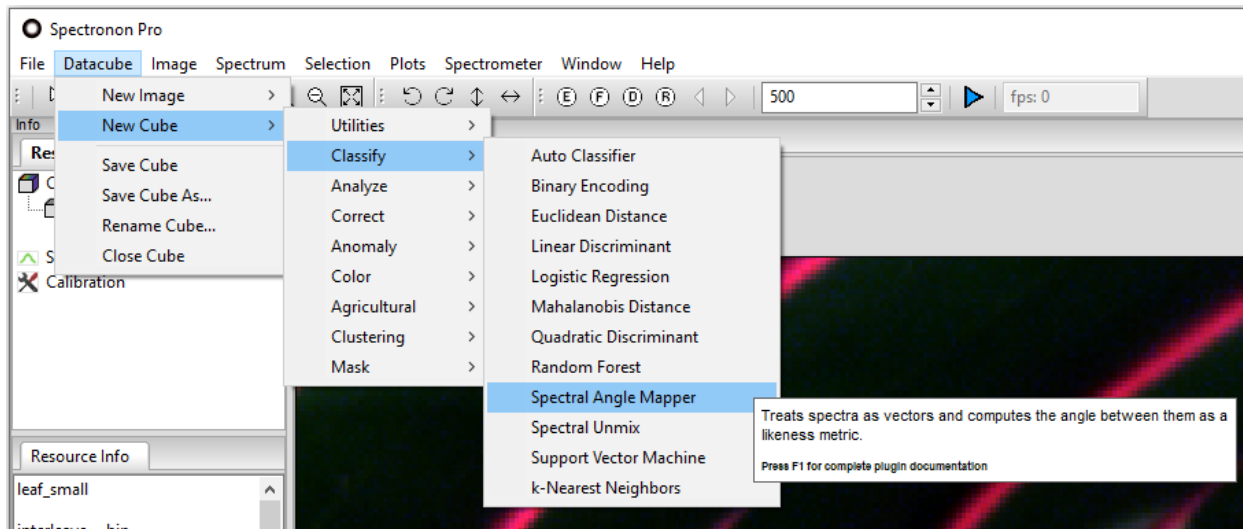


7.2 Spectral Angle Mapper (SAM) Classification

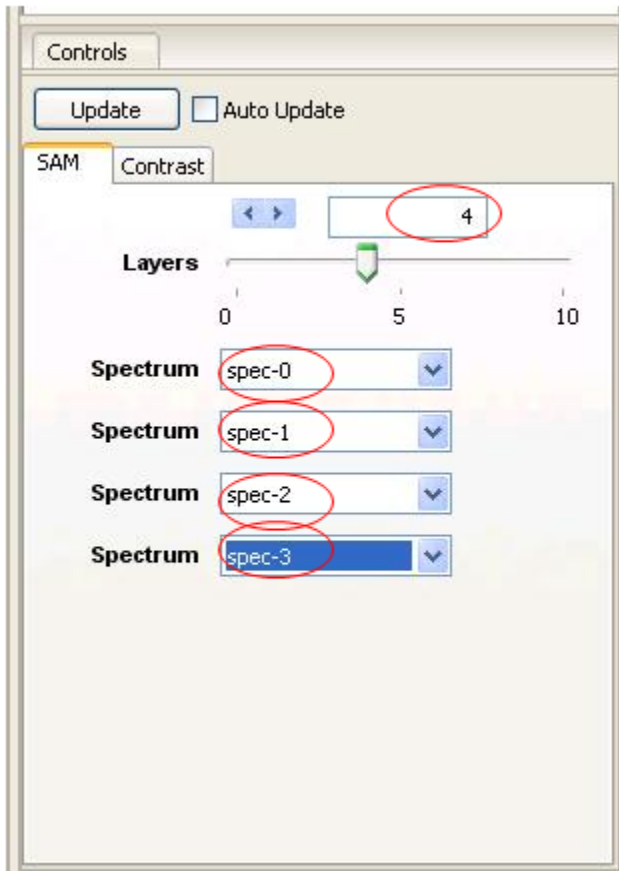
In this example, different kinds of M&M® and Reese's® Pieces candies are classified using the same hyperspectral datacube shown in [Section 5](#). (This datacube can be downloaded from Resonon's website.) To perform SAM, reference spectra must be collected for the objects of interest. In this case, based on prior knowledge, we know that the four candies indicated below (see arrows in image) are all different. Using the *marquee* tool or *lasso*, select small ROIs on each of the four candies (avoid the glare spots) indicated in the figure below. After selecting the ROI, right-click, and then select *Mean Spectrum*. This will generate four spectral curves in the spectral plotter, and also list the four spectral curves in the *Resource Tree*, as shown below.



The four spectral curves shown in the spectral plotter will be used as the Reference Spectra to perform a SAM classification. To perform the classification, click on *Datacube* → *New Cube* → *Classify* → *Spectral Angle Mapper (SAM)*.



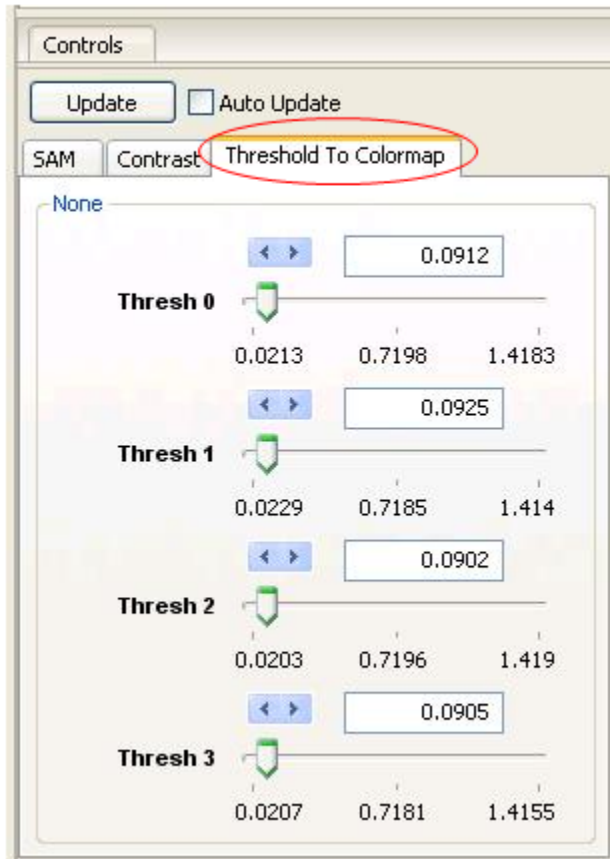
This will open a new window as shown. We wish to classify 4 objects, so use the slider or arrow keys to select a Spectrum for each box.



This will bring up 4 Spectrum pull-down menus. Click into each one of these and select one of the 4 spectra created with the ROI tool.

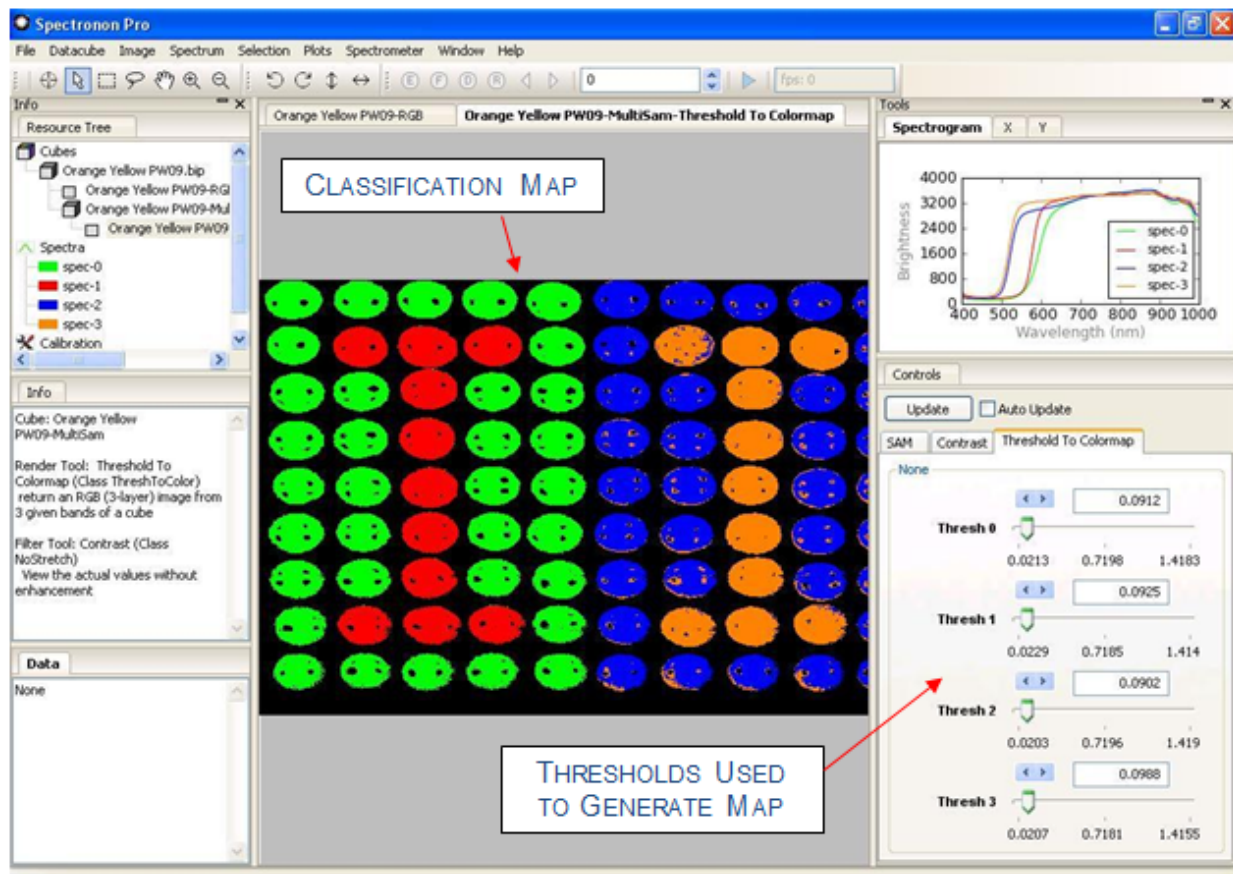
Then click *OK*, and Spectronon calculates the spectral angle (as described above) for each pixel for all 4 Reference spectra.

Typically, several seconds are required for the calculation, which generates a new classification map in the image panel using a default set of Threshold values.



Adjust the Threshold values to obtain a more accurate classification by clicking on the *Threshold to Colormap* tab in the tool control panel.

To adjust the Thresholds, move the sliders, select the arrow keys, or enter values by hand. Each Reference spectrum has its own threshold. After adjusting a threshold, click *Update* and a new classification map will be generated in the Image panel. With a few tries, a classification rendering similar to the one shown below can be generated. Only those pixels within the threshold are colored (the classification colors match those shown in the spectral plotter and *Resource Tree* for the Reference spectrum). If a pixel's Spectral Angle is within the threshold for more than one Reference spectrum, the quantity $(\text{Spectral Angle})/(\text{Threshold Value})$ is calculated for each Threshold and classified for the class that minimizes this quantity. This weighting scheme allows you to emphasize or deemphasize each class to fine-tune your classification map.



SAM is only one of many, many possible classification algorithms. Other classification algorithms are accessed in a similar manner, as described in [Section 11.1.2](#). These algorithms can be found by clicking *Datacube* → *New Cube* → *Classify*. Additionally, user-defined scripts can be written and used with Spectronon for custom classifications algorithms.

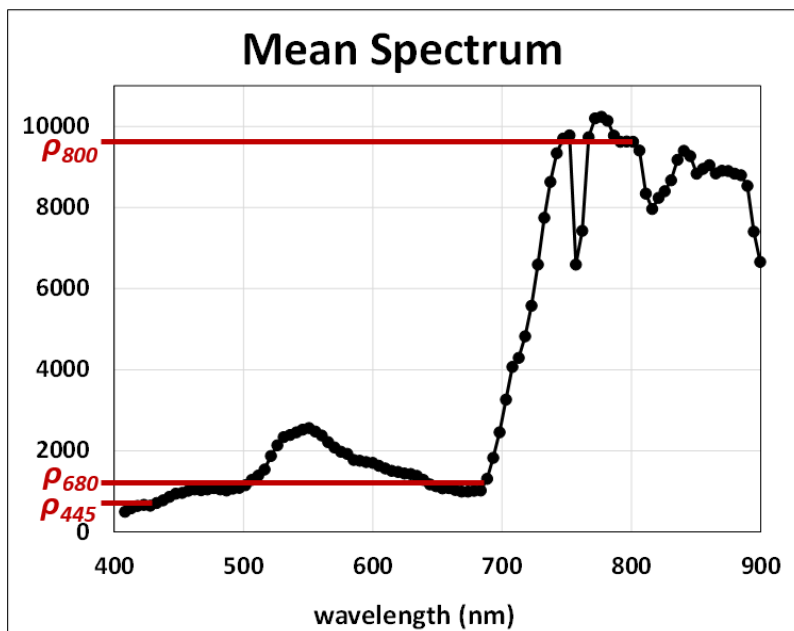
ADVANCED DATA ANALYSIS 3: HYPERSPECTRAL VEGETATION INDICES

8.1 Introduction

Hyperspectral Vegetation Indices (HVI's) are a common tool for analyzing hyperspectral data of agricultural crops. They are typically simple arithmetic combinations of sums and differences of spectral reflectance values at particular wavelengths. For example, the **Structure Insensitive Pigment Index** (SIPI) is defined by the following expression:

$$SIPI = \frac{\rho_{800} - \rho_{445}}{\rho_{800} + \rho_{680}}$$

where the values ρ_{445} , ρ_{680} , and ρ_{800} are shown graphically in the figure below:



HVI's have found many applications in research using airborne remote sensing data, and are currently subjects of intense research efforts.

8.1.1 Useful References

The developers at **ENVI** have written an excellent description of HVI's, which is available at their website. We at Resonon consider ENVI (owned by Harris Corporation) to be excellent hyperspectral data analysis software. Here is a link to their HVI website:

<http://www.harrisgeospatial.com/docs/VegetationAnalysis.html>

Another very good reference discussing HVI's is this book:

Thenkabail, Lyon, and Hute, *Remote Sensing of Vegetation*, CRC Press (2011).

8.1.2 HVI's in Spectronon

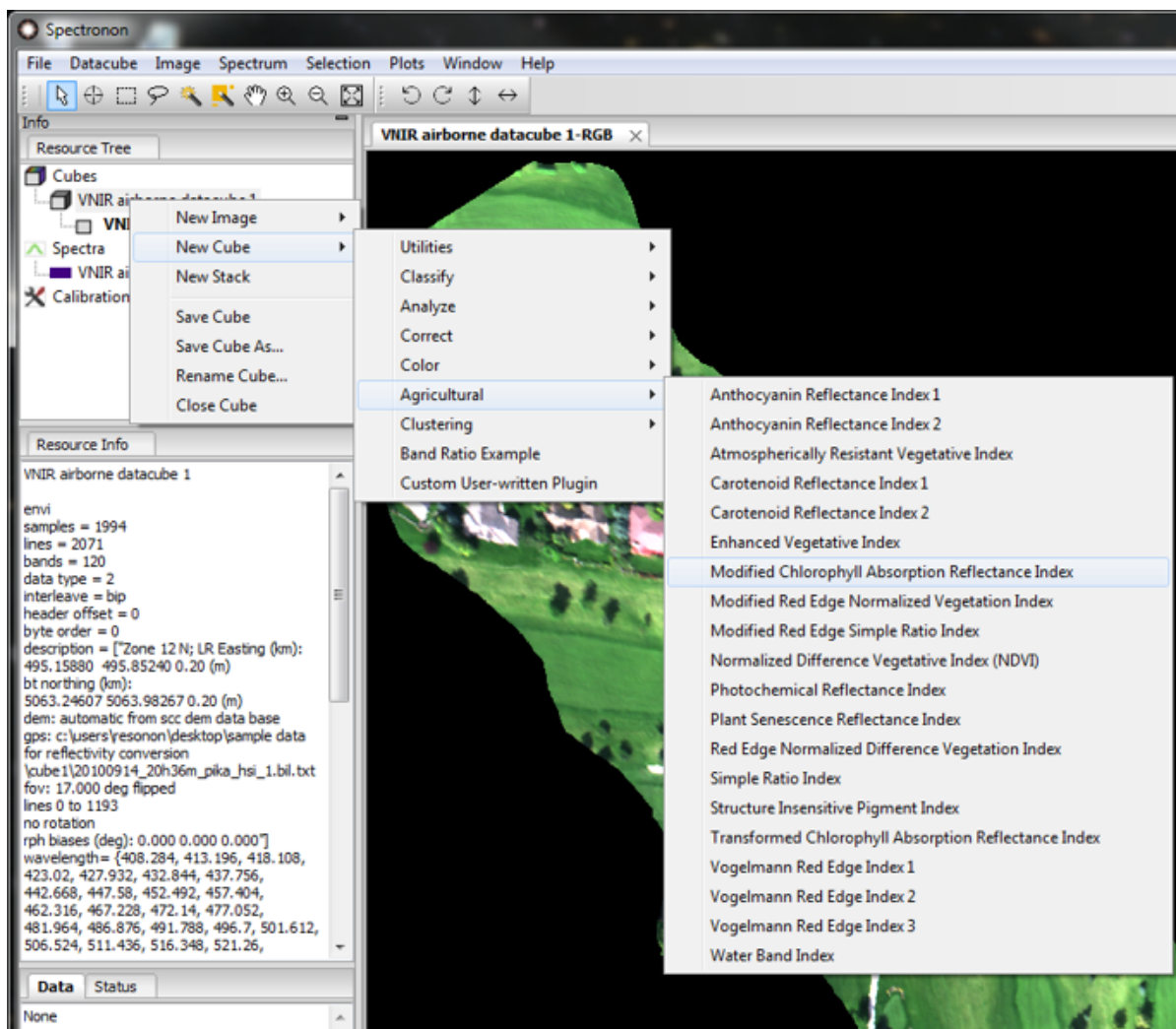
HVI's calculated by Spectronon are listed in the following table. Also listed are the equations that Spectronon uses for each HVI.

Hyperspectral Vegetation Index	Equation in Spectronon
Anthocyanin Reflectance Index 1	$ARI1 = \frac{1}{\rho_{550}} - \frac{1}{\rho_{700}}$
Anthocyanin Reflectance Index 2	$ARI2 = \rho_{800} \left(\frac{1}{\rho_{550}} - \frac{1}{\rho_{700}} \right)$
Atmospherically Resistant Vegetative Index	$ARVI = \frac{NIR - (Red - \gamma(Blue - Red))}{NIR + (Red - \gamma(Blue - Red))}$
Carotenoid Reflectance Index 1	$CRI1 = \frac{1}{\rho_{510}} - \frac{1}{\rho_{550}}$
Carotenoid Reflectance Index 2	$CRI2 = \frac{1}{\rho_{510}} - \frac{1}{\rho_{700}}$
Enhanced Vegetative Index	$EVI = \frac{NIR - Red}{NIR + 6.0 \cdot Red - 7.5 \cdot Blue + 1}$
Modified Chlorophyll Absorption Reflectance Index	$MCARI = \rho_{700} - \rho_{670} - 0.2(\rho_{700} - \rho_{550}) \left(\frac{\rho_{700}}{\rho_{670}} \right)$
Modified Red Edge Normalized Vegetation Index	$MRENDVI = \frac{\rho_{750} - \rho_{705}}{\rho_{750} + \rho_{705} - 2 \cdot \rho_{445}}$
Modified Red Edge Simple Ratio Index	$MRESR = \frac{\rho_{750} - \rho_{445}}{\rho_{705} - \rho_{445}}$
Normalized Difference Vegetative Index	$NDVI = \frac{NIR - Red}{NIR + Red}$
Photochemical Reflectance Index	$PRI = \frac{\rho_{531} - \rho_{570}}{\rho_{531} + \rho_{570}}$
Plant Senescence Reflectance Index	$PSRI = \frac{\rho_{680} - \rho_{500}}{\rho_{750}}$
Red Edge Normalized Difference Vegetation Index	$RENDVI = \frac{\rho_{750} - \rho_{705}}{\rho_{750} + \rho_{705}}$
Simple Ratio Index	$SR = \frac{NIR}{Red}$
Structure Insensitive Pigment Index	$SPI = \frac{\rho_{800} - \rho_{445}}{\rho_{800} + \rho_{680}}$
Transformed Chlorophyll Absorption Reflectance Index	$TCARI = 3 \left[\rho_{700} - \rho_{670} - 0.2(\rho_{700} - \rho_{550}) \left(\frac{\rho_{700}}{\rho_{670}} \right) \right]$
Vogelmann Red Edge Index 1	$VREI1 = \frac{\rho_{740}}{\rho_{720}}$
Vogelmann Red Edge Index 2	$VREI2 = \frac{\rho_{734} - \rho_{747}}{\rho_{715} - \rho_{726}}$
Vogelmann Red Edge Index 3	$VREI3 = \frac{\rho_{734} - \rho_{747}}{\rho_{715} + \rho_{720}}$
Water Band Index	$WBI = \frac{\rho_{970}}{\rho_{900}}$

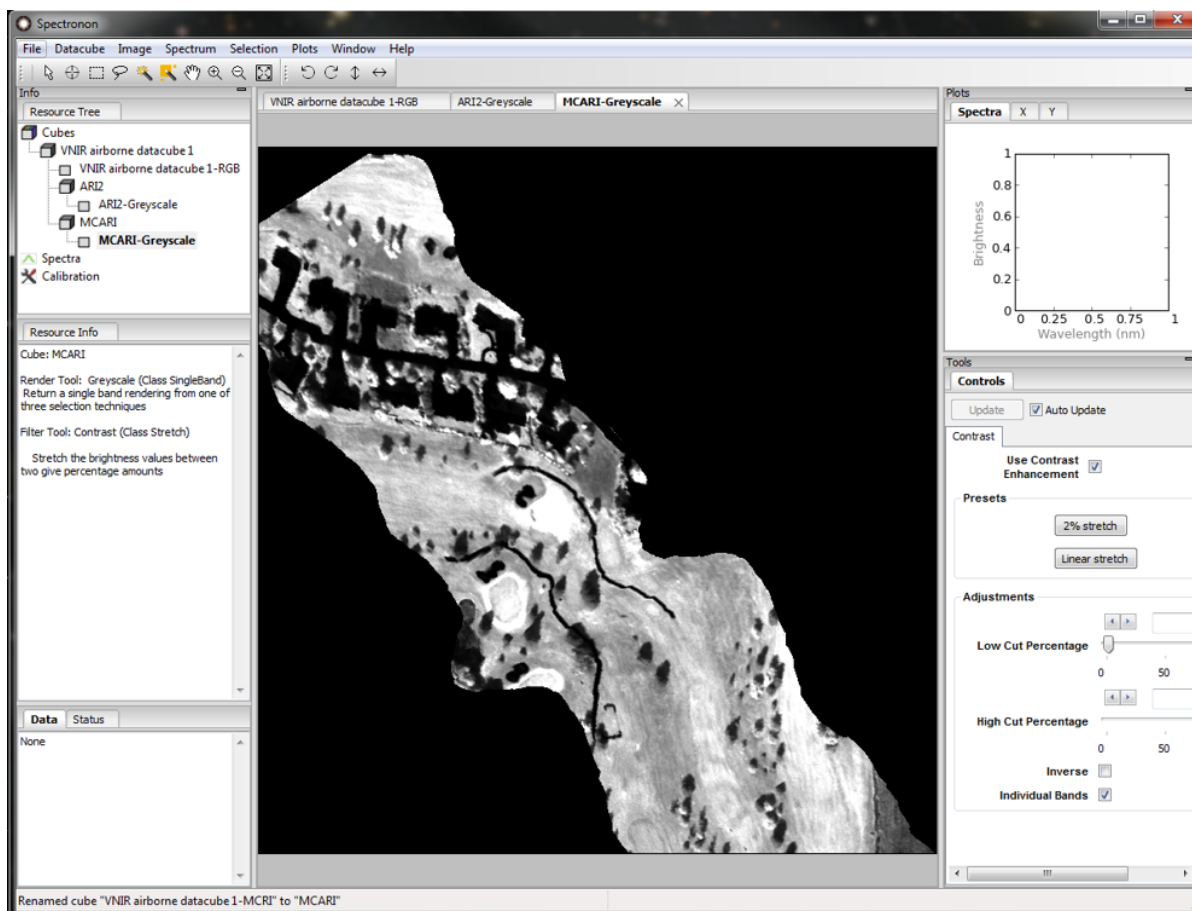
8.2 Generating HVI maps in Spectron


The datacube in the following example was acquired using a Resonon airborne hyperspectral imaging system. This datacube is available for free download from <http://downloads.resonon.com/>.

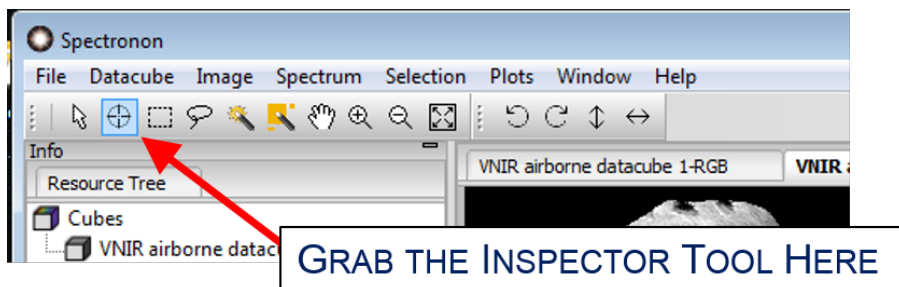
HVI's in Spectron can be found at the path: *Datacube* → *New Cube* → *Agriculture*:



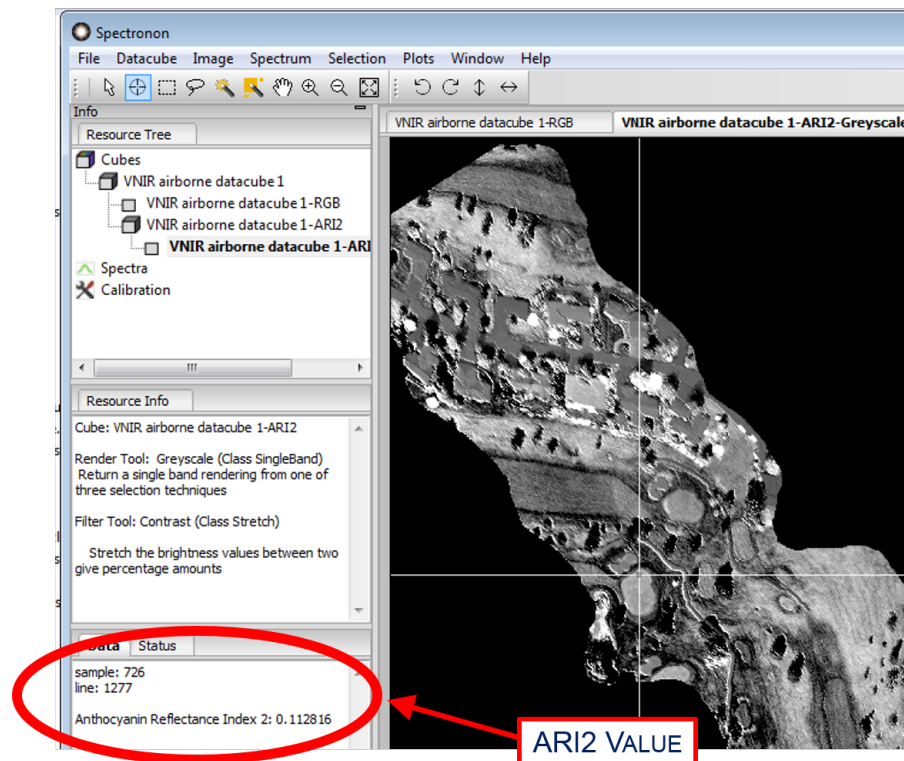
Choosing an HVI from this menu will generate a grey-scale image:



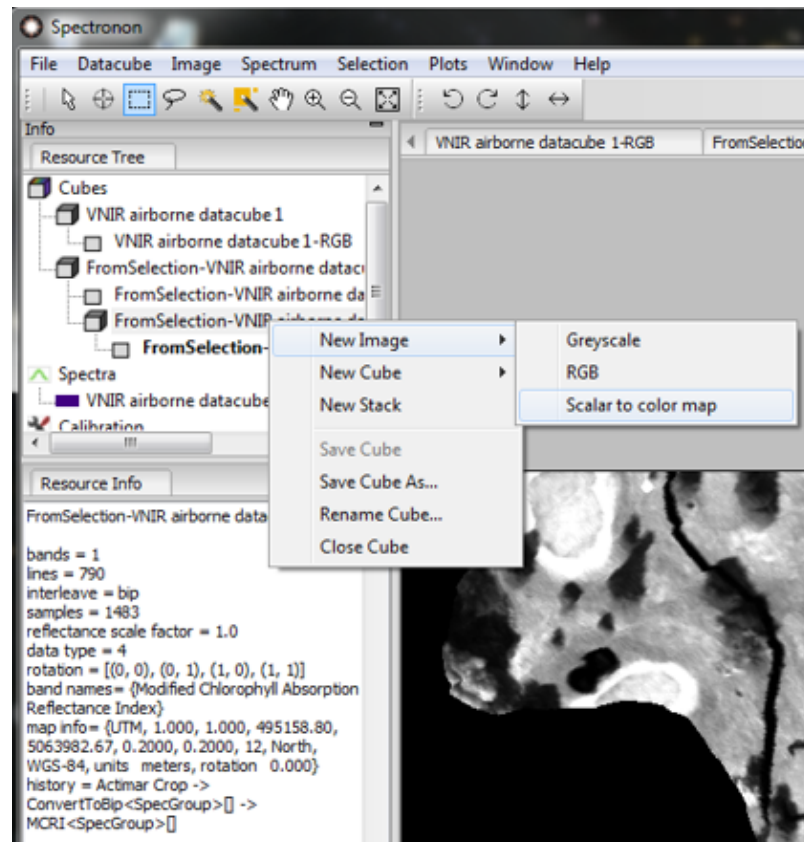
The value of the HVI at any pixel can be obtained with the Inspector Tool :



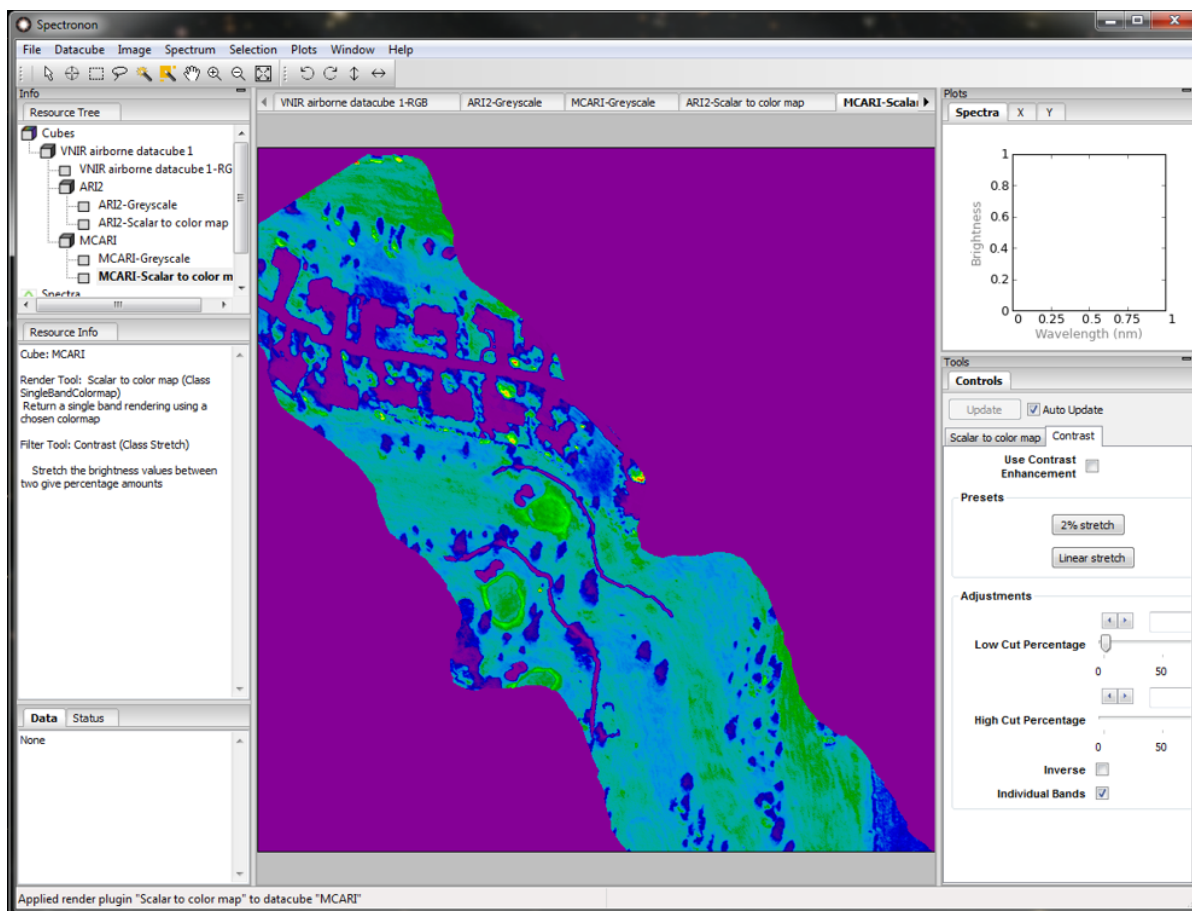
The value of the HVI, along with the coordinates of the inspector tool, can be found in the lower left corner of the screen:



Another tool for visualizing HVI maps is to transform the grey-scale image into a colormap. This can be accomplished with *Datacube* → *New Image* → *Scalar to Colormap*:

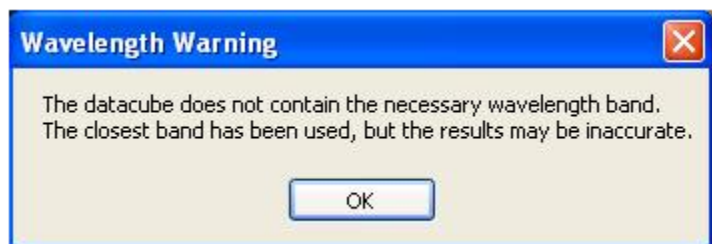


Below is a “spectral” colormap of the MCARI index for the example datacube:



After generating a colormap, the user can change the color scheme from the *Colormap* menu on the *Controls* tab. It may be helpful to toggle the *Contrast Enhancement* feature. Experiment with different colormaps, normalization, and contrast settings, all available on the *Controls* tab.

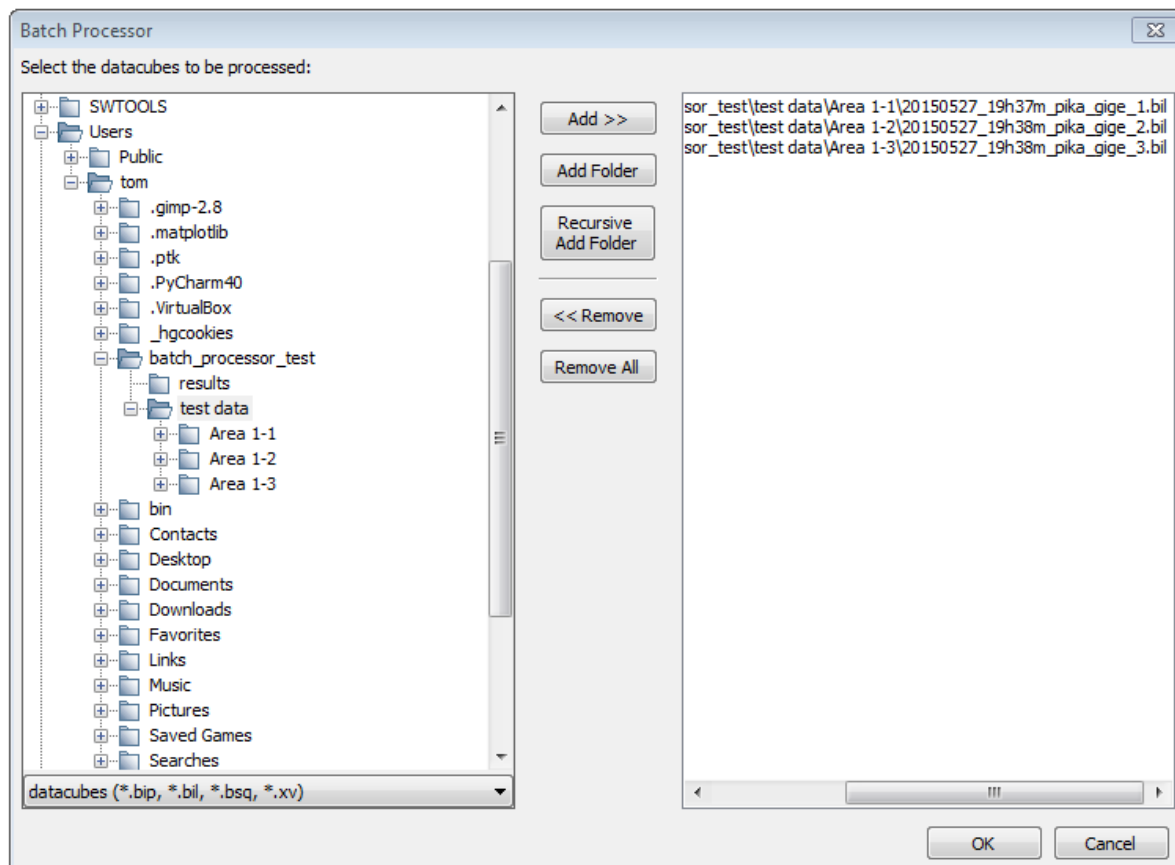
Note: If an HVI is selected that requires spectral bands that do not exist in your datacube, the warning shown below will appear. As described in the warning, clicking “OK” will generate an image using the closest bands available, but the resulting image will often be a poor approximation to a true index mapping.



BATCH PROCESSING

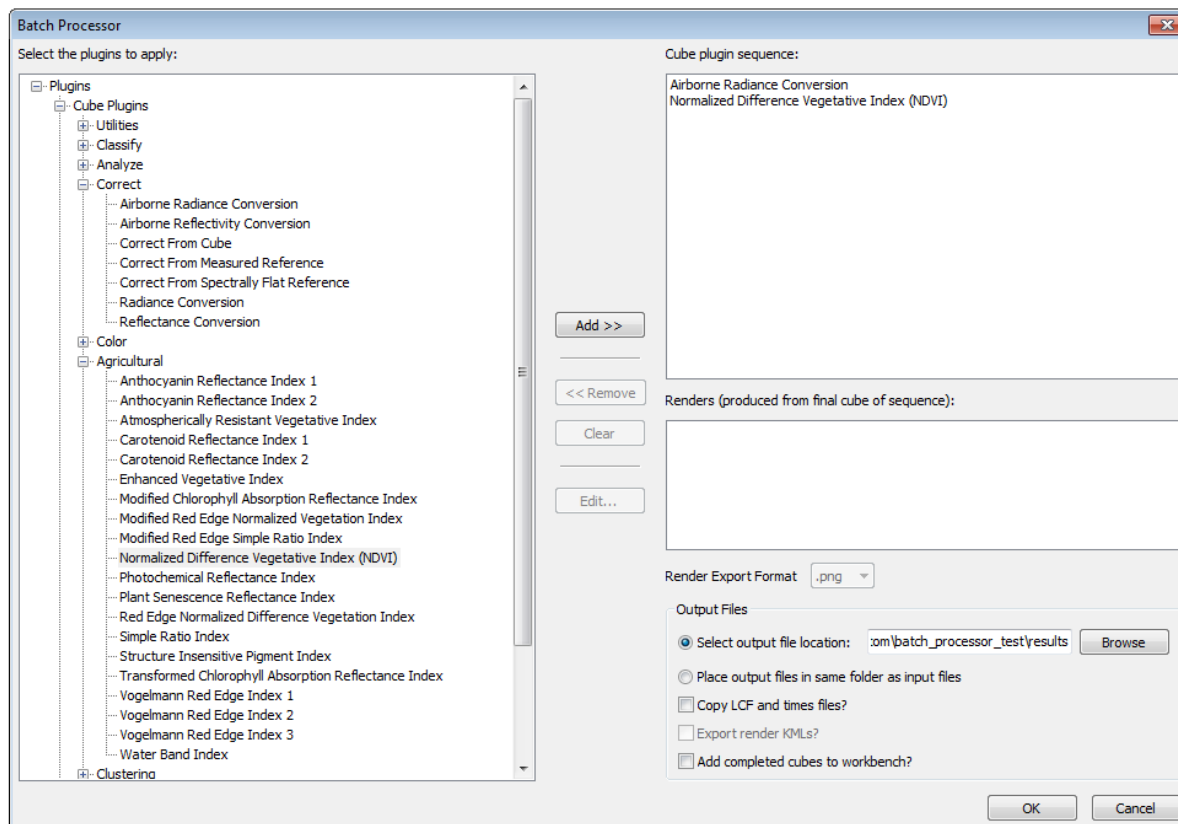
Manually running the same processing sequence on a series of datacubes can be a tedious process. Spectronon's batch processor allows the user to choose a number of datacubes at once and set up a sequence of processing steps (plugins) to execute on the chosen files. This may be of particular interest to users of our Airborne product, but applies to any customer who needs process multiple datacubes at once.

To start the batch processor, select *File* → *Batch Processor*. The batch processing window appears.



Expand the directory tree on the left side of the window to find the datacubes you wish to process. Use the buttons in the middle of the screen to select the desired datacubes. Datacubes that will be processed appear on the right side of the window. In the figure above 3 datacubes are selected for processing.

When you are happy with your selection, click *Ok*. A new window appears to define the plugins you wish to run.

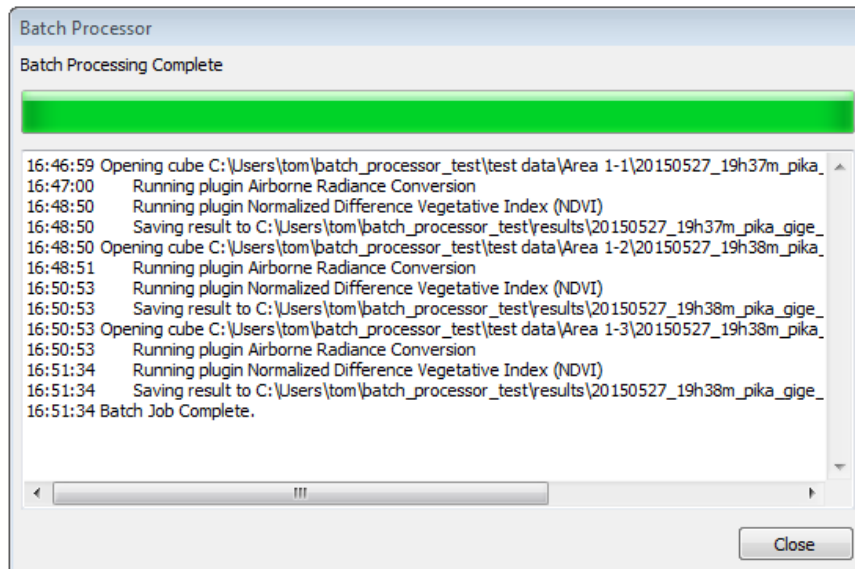


Use the directory tree to select the cube plugins you wish to run. Cube plugins generate new datacubes and will run in the order selected, as shown in the *Cube plugin sequence* panel. Each plugin operates on the output of the plugin before it. Each plugin's setup window appears as you add it to the processing queue. In the figure above the "Airborne Radiance Conversion" plugin is set to run, followed by the NDVI plugin.

Select any render plugins you wish to run. Render plugins generate images from the final result of the cube plugin sequence, and appear in the *Renders* panel. You can choose to save renders as .png, .tiff or .jpg with *Render Export Format*. In the figure above no render plugins are selected, so no images will be generated.

Use the controls at the lower right to select the location to save results. *Copy LCF and times files?* and *Export render KMLs?* are useful for our airborne customers. If desired, the resultant datacubes can be added to the workbench to show in Spectronon when processing is finished.

Press the *OK* button to begin processing. A progress dialog appears.



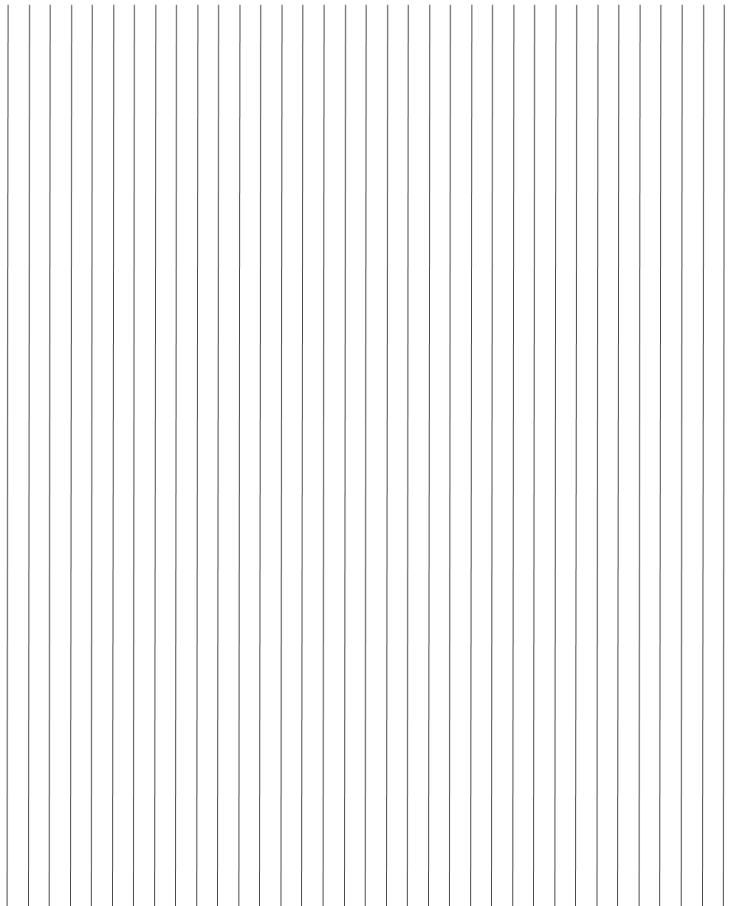
Press the *Close* button to dismiss the dialog after processing is complete. The results are saved to the selected location.

FOCUSING & CALIBRATION SHEETS

Use the focusing sheets to focus the objective lens, and use the aspect ratio calibration sheet to set the stage speed and imager framerate. See the [Basic Data Acquisition](#) for details.

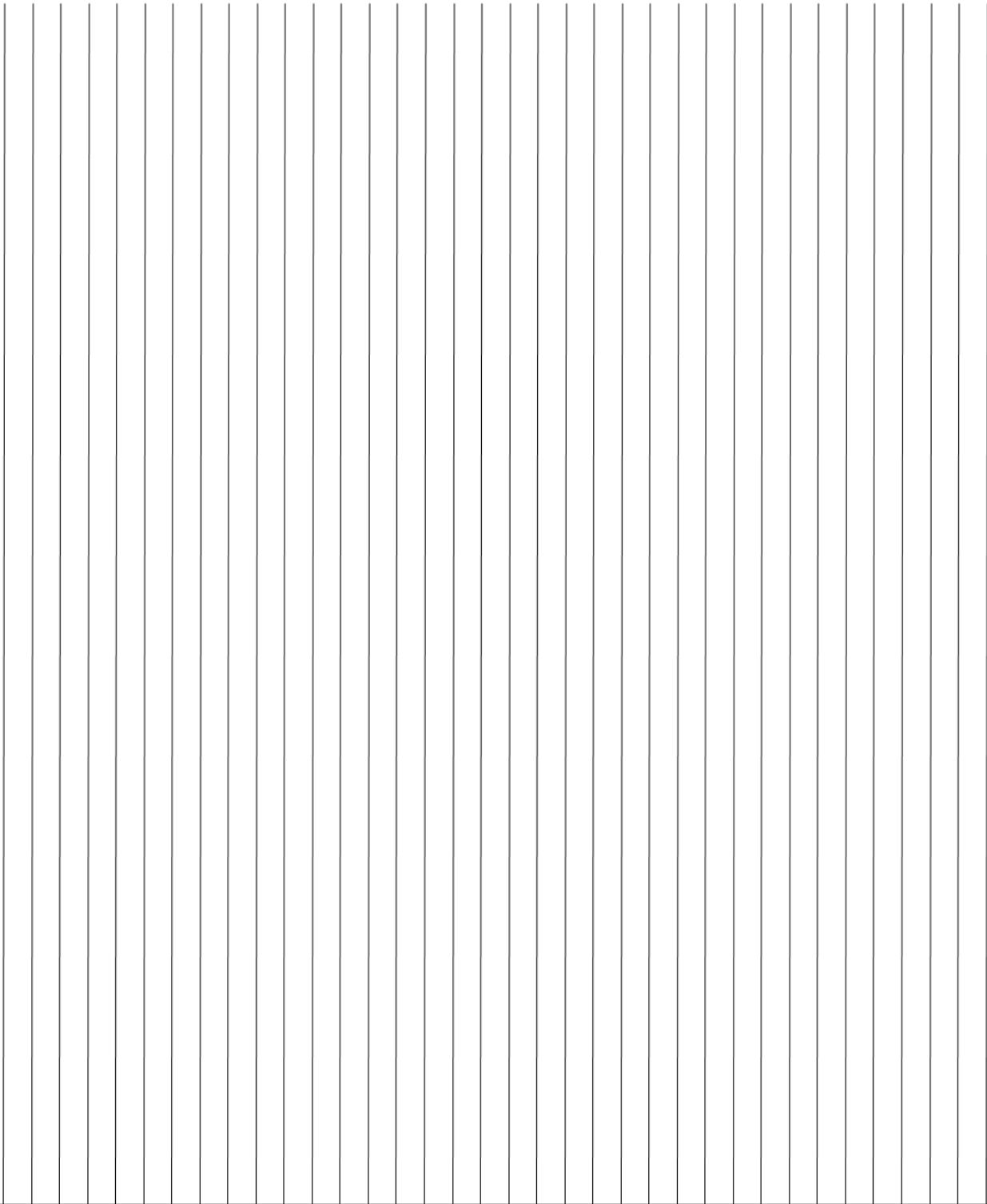
10.1 Small Focusing Sheet

Focusing Sheet



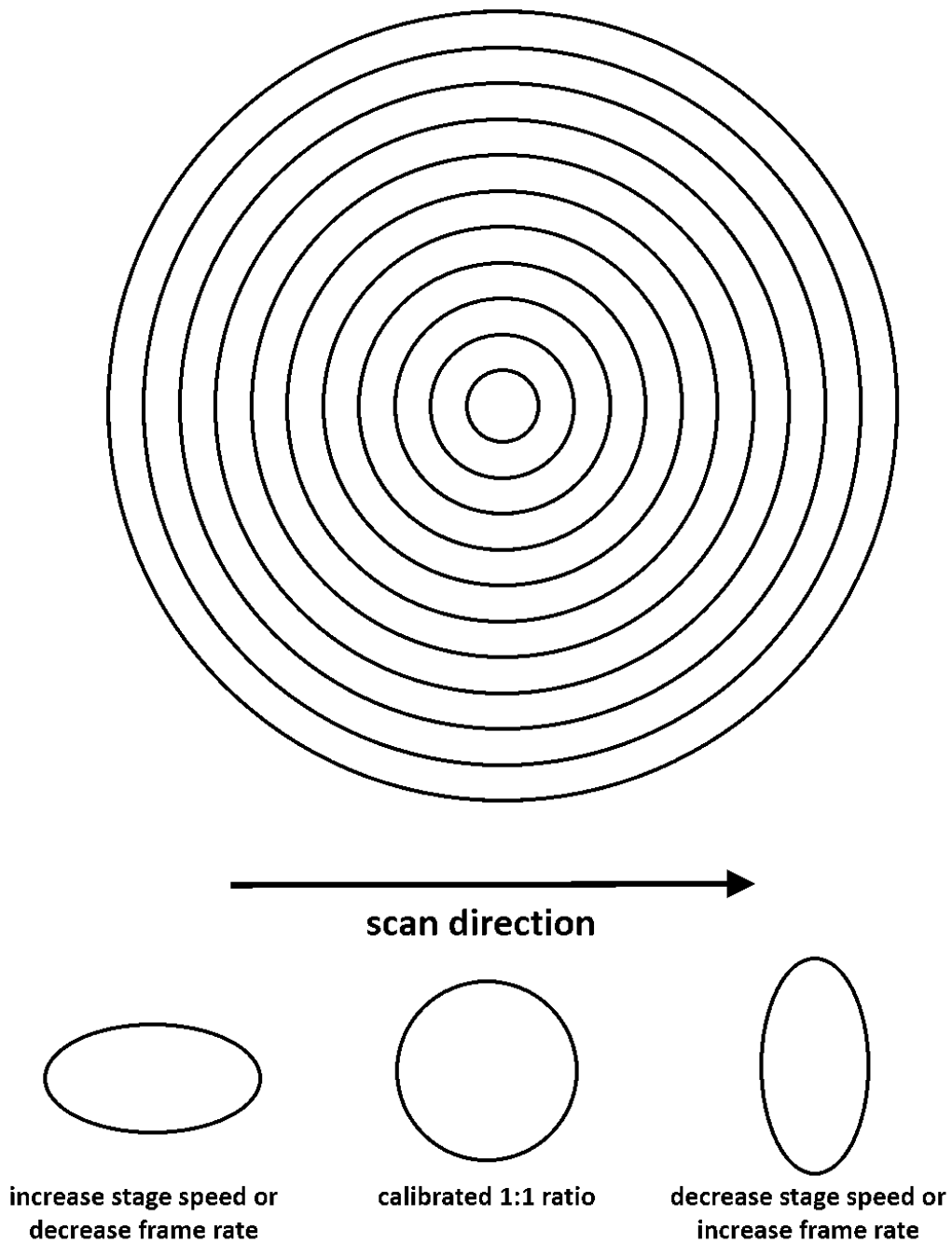
10.2 Large Focusing Sheet

Focusing Sheet



10.3 Aspect Ratio Calibration Sheet

Aspect Ratio Calibration Sheet



APPENDIX: BUILT-IN PLUGINS

11.1 Cube Plugins

11.1.1 Utilities

Append Cube

Description:

Append one cube to another by either lines, bands, samples, or any spatial combination of lines and samples. To use bands, the datacubes must be the same size spatially (lines and samples). To use the lines or samples option, the datacubes must match the number of lines and samples respectively. The option of *any spatial* will scramble the line/sample representation in order to make a spatially rectangular datacube. For this option, the number of bands between cubes must match.

Usage:

New Name: New name of output cube.

Append Dir: Direction to append cube (lines, bands, samples, or any spatial).

Append Cube: Cube to append.

Outputs:

- Appended datacube.

Average Channels

Description:

Averages adjacent bands, samples, and/or lines.

Usage:

Spectral, Sample, and Line Averaging: Select the number of adjacent bands (spectral), sample and/or lines to average together.

Return floating points: Option to return floating point values. Otherwise, the data type of input cube is returned.

Outputs:

- Datacube with adjacent bands, samples and/or lines averaged.

Bad Band Removal

Description:

Remove selected bands from a datacube and optionally interpolate replacements.

Usage:

Start band: First band to remove.

End band: Last band to remove.

Interpolate: If checked, the removed bands will be linearly interpolated from the adjacent bands.

Outputs:

- Datacube with the selected bands removed or interpolated.

Band Summary Statistics

Description:

Calculates the following summary statistics for a band of data:

- minimum value
- maximum value
- 25th, 50th (median), and 75th percentile values
- mean
- standard deviation
- variance
- skew
- kurtosis

Usage:

Mode: Choose to view by wavelength, band number, or band name, if available.

Wavelength/Band/Band Name selector: Choose wavelength/band to view.

Ignore Zeros: Check this box to exclude any pixel with the value 0 from calculation.

Output Decimal Places: Select the number of decimal places to display for calculated statistics.

Generate Histogram Image: Click this button to plot the distribution of values for the currently selected band.

Outputs:

A single band datacube containing the band summarized.

Bin Channels

Description:

Bins (sums) adjacent bands, samples, and/or lines.

Usage:

Spectral, Sample, and Line Binning: Select the number of adjacent bands (spectral), sample and/or lines to bin together. This is a summation, not an average.

Return floating points: Option to return floating point values. Otherwise, the data type of input cube is returned.

Outputs:

- Datacube with adjacent bands, samples and/or lines binned.

Convert to BIP, BIL or BSQ

Description:

Converts one datacube interleave to another.

Usage:

Input format: Shows current interleave.

Output format: Select desired output interleave.

Outputs:

- Datacube rotated to the desired interleave.

Count Pixel Values

Description:

Counts unique class labels, maximum pixel probabilities, or minimum spectral angles in a datacube and displays the counts.

Usage:

Apply to a classified cube class labels, classification probabilities, or spectral angles. The tool will compute the number of class label counts, maximum probability counts, or minimum spectral angle counts.

Input Cube Type: Select Probability, Class Label, or Spectral Angle depending on the input datacube type.

Band Number: For Class Label cubes, select the band the pixel counts are desired for.

Outputs:

- Launches a dialog box containing the counts.

Crop Spatially

Description:

Crop a datacube spatially by starting and ending samples and lines.

Usage:

Start Sample: First sample to keep in resulting datacube.

End Sample: Last sample to keep in resulting datacube.

Start Line: First line to keep in resulting datacube.

End Line: Last line to keep in resulting datacube.

Outputs:

- Cropped datacube.

Crop Wavelengths

Description:

Crops beginning and/or ending bands from datacube by wavelength.

Usage:

Min Wavelength: Starting wavelength of new cube.

Max Wavelength: Ending wavelength of new cube.

Outputs:

- Datacube with beginning and/or ending bands cropped.

Crop Wavelengths by Band Number

Description:

Crops beginning and/or ending bands from datacube, with bands specified by band number.

Usage:

Min Band: Starting band of new cube.

Max Band: Ending band of new cube.

Outputs:

- Datacube with beginning and/or ending bands cropped.

Normalize Datacube

Description:

Normalize each pixel (point spectrum) in a datacube.

Usage:

Method:

- *Unit:* Divide each brightness value within a pixel by the sum of all bands in that pixel, such that the sum of brightness across all bands in the pixel after normalization is 1.
- *Root Mean Square:* Divide each brightness value within a pixel by that pixel's root mean square brightness.
- *Maximum:* Divide each brightness value within a pixel by the maximum that occurs in that pixel, such that the band that contained the maximum value now contains 1 and all other bands contain a number between 0 and 1.
- *Min-Max:* Scales each brightness value within a pixel by subtracting the minimum value that occurs in the pixel and dividing by the maximum, such that the new minimum and maximum values are 0 and 1.

Outputs:

- Normalized datacube.

Savitzky-Golay Smoothing

Description:

The Savitzky-Golay filter fits data with successive low-degree polynomials using linear least squares, resulting in smoothed data while preserving much of the original signal's structure. This implementation will optionally return the n-th derivative of the smoothed signal.

Usage:

Number of points: Size of sliding window for polynomial fitting.

Polynomial Degree: Degree of polynomial used to fit.

Differential order: Order of derivative to return.

Outputs:

- Smoothed datacube or n-th derivative of smoothed datacube.

References:

1. https://en.wikipedia.org/wiki/Savitzky-Golay_filter

Scale To One Utility

Description:

Rescales a datacube to 1 by using either the cube's Reflectance Scale Factor, Ceiling, Bit Depth, or user entered factor.

Usage:

Reflectance Scale Factor: The scaling factor to divide data by. It suggests the value of the cube's Reflectance Scale Factor, Ceiling, and Bit Depth metavalues, if present and in that order.

Outputs:

- Datacube scaled to 1.

Spectrally Interpolate Linear Cube

Description:

Spectrally interpolates a datacube to the bands that would have been produced by a linearly calibrated imager with a different slope and intercept. This may be useful for using data from different spectral imagers in a single model or other classification routine.

Usage:

Slope: Enter new slope.

Intercept: Enter new intercept.

Slope multiplier: Set this to 2 for Pika III imagers using cameras in format 7 mode 2 or other values to appropriately compensate for on-camera binning.

New band count: Number of output bands desired.

Extrapolate linearly outside original cube waves: Extrapolate linearly if desired cube falls outside of existing data. An error will occur if this condition is true but the option left unchecked.

Outputs:

- Datacube interpolated to new wavelengths.

Spectrally Resample Cube

Description:

Resamples (interpolates) a datacube to match the wavelengths of a selected text or header file.

Usage:

This tool will prompt the user to select a text file (txt or csv) or a header file (hdr) containing the desired wavelengths. If txt or csv is chosen, the wavelength values should be in the first column in units of nanometers.

Outputs:

- Resampled datacube.

Spectrally Resample Spectrum to Datacube

Description:

Resamples (interpolates) a Spectrum to match the wavelengths of source datacube.

Usage:

Spectrum: Select the Spectrum to resample.

Outputs:

- Resampled Spectrum.

Subset Cube Bands

Description:

Build a cube consisting of a subset of the bands of the source cube.

Usage:

3 Band RGB: Preset to return a 3 band cube using default Red, Green, and Blue bands.

3 Band Color IR: Preset to return a 3 band cube using default Infrared, Red, and Green bands.

Bands: If not using the above presets, select the number and wavelengths of each band to return.

Outputs:

- Datacube containing the selected subset of bands.

Subtract Cube

Description

Subtract one datacube from another, pixel-by-pixel and band-by-band. The datacubes must have matching shape (line, band, and sample count).

Usage:

Second Cube: The cube to be subtracted.

Outputs:

- A new datacube in which each pixel represents the difference between the source datacube and *Second Cube*.

Subtract Spectrum

Description:

Subtracts a background spectrum from a datacube. This could be a noise or background fluorescence spectrum, for example.

Usage:

Subtract by: Select the Spectrum to subtract.

Return floating points: Option to return a floating point cube. Otherwise, the original cube's data type is used.

Outputs:

- Datacube with spectrum subtracted.

Unit Conversion Utility

Description:

Convert a cube's intensity values by multiplying each element by a constant scale factor.

Usage:

Scale Factor: Number to multiply datacube by.

Return Floating Point: If selected, the cube will be converted to a 32 bit float. Otherwise, the original datatype will be maintained.

11.1.2 Classify

Auto Classifier

Description:

This tool incorporates many proven processes of hyperspectral image classification into a easy-to-use pipeline, while minimizing decisions that must be made by the user. It includes: 1. Feature engineering by concatenation of 1st derivatives to Savitsky-Golay smoothed spectra. 2. Dimensionality reduction with Principal Component Analysis 3. Random Forest classifier

Usage:

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class datacube selectors below.

Class Number: For each Class number, select the datacube associated with the class.

Outputs:

- **A floating point datacube, with each band containing the probability (0-1) of a pixel belonging that Class, ordered by Class Number.**

References:

1. https://en.wikipedia.org/wiki/Savitzky-Golay_filter
2. https://en.wikipedia.org/wiki/Principal_component_analysis
3. <https://scikit-learn.org/1.3/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Binary Encoding

Description:

Binary encoding is a simple classification method with a small computational load. Each band within an input spectrum is encoded into zeros and ones depending on whether the band is above or below the spectral mean. An additional encoding is created, using a zero or one to represent whether the first derivative of that band is negative or positive. This encoding is concatenated with the first. The datacube is encoded in the same way. An exclusive or is then used to compare the datacube's encoding with the input spectrum's encoding, and the result summed. This provides a single number that provides a measure of similarity between the input spectrum and individual pixels of a datacube.

Usage:

Layers: Select the number of input spectra to classify the datacube against.

Spectrum: Select the input spectrum used for classification.

Outputs:

- **A datacube with each band containing the distance of a pixel from a Spectrum,** ordered by input position.

References:

1. https://www.ipi.uni-hannover.de/fileadmin/ipi/publications/Xie_istanbul.pdf

Euclidean Distance

Description:

Computes the ordinary straight-line distance in Euclidean space.

$$\|u - v\|_2 = \left(\sum |u_i - v_i|^2 \right)^{1/2}$$

where u and v are input vectors (spectra).

Usage:

Layers: Select the number of input spectra to classify the datacube against.

Spectrum: Select the input spectrum used for classification.

Outputs:

- **A datacube with each band containing the Euclidean distance of a pixel from a Spectrum,** ordered by input position.

References:

1. https://en.wikipedia.org/wiki/Euclidean_distance

Linear Discriminant

Description:

A generalization of Fisher's linear discriminant that finds a linear combination of features that separate two or more classes, as defined by input training datacubes. It assumes that the independent variables (bands) are normally distributed. It is closely related to logistic regression, which does not make this assumption.

Usage:

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings at the bottom of the dialog box. For each Class grouping, select the datacube in the Class field and the associated dependent value in the Group Number field.

Solver: One of either Singular Value Decomposition (svd), Least Squares (lsqr), or Eigenvalue decomposition (eigen). See references below for more information.

N components: Number of components to keep after dimensionality reduction. Must be less or equal to the smaller value between the number of classes minus one, and the number of spectral bands.

Probability: Check to return a probability of class membership, uncheck to return binary predictions.

Tolerance: Threshold for determining whether a singular value is significant. Only used in Singular Value Decomposition.

Outputs:

- Datacube with each band containing a probability of a pixel being a member of that class, or a single band with integer encoding that represents class membership by input position order.

References:

1. https://en.wikipedia.org/wiki/Linear_discriminant_analysis
2. https://scikit-learn.org/1.3/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.htm

Logistic Regression

Description:

Logistic regression is a statistical model that uses a logistic function to model a binary dependent variable, in this case membership of a class. Despite the name, it is a linear classification algorithm, not a true regression. It is generally fast and powerful.

Usage:

Reg. Method: Regularization method or penalty. See references for more information.

Probability: Check to return a probability of class membership, uncheck to return integer values indicating the predicted class.

Auto Weight?: Check to automatically adjust training weights to be inversely proportional to class frequencies in the input data. If this option is not checked, the classifier may be more likely to assign pixels to classes that occur more frequently in the training data.

C: Inverse of regularization strength; smaller values specify stronger regularization.

Save coefficients to file: Check to save model coefficients to file for use with other software packages.

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings at the bottom of the dialog box. For each Class grouping, select the datacube in the Class field and the associated dependent value in the Group Number field.

Outputs:

- Datacube with each band containing a probability of a pixel being a member of that class, or a single band with integer encoding that represents class membership by input position order.

References:

1. https://en.wikipedia.org/wiki/Logistic_regression
2. https://scikit-learn.org/1.3/modules/linear_model.html#logistic-regression

Mahalanobis Distance

Description:

Mahalanobis distance quantifies the multivariate similarity between each pixel in a datacube and a distribution defined by all of the pixels in a training datacube (or cubes). The distance used is normalized by the inverse of the covariance matrix computed from the training cubes to adjust for correlations between bands and differences in the magnitude of variance across bands.

$$d_M(p, q) = \sqrt{(q - p)^T C^{-1} (q - p)}$$

Where p is the spectrum at the point being evaluated, q is the mean spectrum computed from the input training cube, and C is the covariance matrix computed from the input training cube. For uncorrelated variables with distributions of equal variance, this reduces to the Euclidean distance.

By applying a threshold to the computed Mahalanobis distance, pixels can be classified as members or outliers of a class.

Usage:

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class entries at the bottom of the dialog box. Select Class datacubes for each class.

Outputs:

- Datacube with each band containing the Mahalanobis distance between a pixel and the distribution defined by an input class, as ordered by Class datacube input position. Small numbers indicate greater similarity.

References:

1. https://en.wikipedia.org/wiki/Mahalanobis_distance

PLS-DA

Description:

Partial least squares-discriminant analysis is a statistical model that first encodes categorical response variables to a one-hot binary representation, then uses a partial-least squares(PLS) regression to predict the probability of each class membership. PLS regression is a powerful technique for hyperspectral analysis because it uses a cross-decomposition to reduce dimensionality and transform both the response variable (the class membership probabilities) and the input data (the spectral response) to best model the fundamental relationship between the two.

Usage:

Number of Components To Keep: Used by Partial Least Squares as the number of components to use after features have been transformed. With too few components, predictive power may be lost. With too many, the model may be prone to overfitting.

Return Scores?: Check to return a score proportional to the predicted likelihood of each class membership, with one band per class. Higher values indicate higher likelihood of class membership. Note that these are not probabilities, and may be negative. Uncheck to return integer values indicating the predicted class.

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings at the bottom of the dialog box. For each Class grouping, select the datacube in the Class field and the associated dependent value in the Group Number field.

Quadratic Discriminant

Description:

Quadratic discriminant analysis is closely related to linear discriminant analysis, but it does not assume the covariance of each of the classes is identical. As a result, it tends to fit data better but has more parameters to estimate.

Usage:

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings at the bottom of the dialog box. For each Class grouping, select the datacube in the Class field and the associated dependent value in the Group Number field.

Probability: Check to return a probability of class membership, uncheck to return binary predictions.

Regularization: Parameter to regularize the covariance estimate.

Outputs:

- Datacube with each band containing a probability of a pixel being a member of that class, or a single band with integer encoding that represents class membership by input position order.

References:

1. https://en.wikipedia.org/wiki/Quadratic_classifier
2. <https://scikit-learn.org/1.3/modules/generated/sklearn.qda.QDA.html>

Random Forest

Description:

Random Forest classification is an ensemble method constructed of a multitude of decision trees, each constructed from a random subset of training data and input features. The final output class is the mode of the classes of the individual trees, which helps correct for individual decision trees' tendency to overfit the training set.

Usage:

Probability: Check to return a probability of class membership (one band for each class), uncheck to return integer class membership predictions as a single band.

Estimators: The number of trees in the forest. Higher numbers tend to produce better predictive models, but require more memory and computational time.

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings at the bottom of the dialog box. For each Class grouping, select the datacube in the Class field and the associated dependent value in the Group Number field.

Outputs:

- Datacube with each band containing a probability of a pixel being a member of that class, or a single band with integer encoding that represents class membership by input position order.

References:

1. https://en.wikipedia.org/wiki/Random_forest
2. <https://scikit-learn.org/1.3/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Spectral Angle Mapper

Description:

Compares spectra in a datacube to reference (input) spectra by computing the *spectral angle* between them. By treating the spectra as vectors, the angle between them is simply the inverse cosine of the normalized dot product of the two.

$$\theta(p, q) = \arccos \left(\frac{\sum_{i=1}^n p_i q_i}{(\sum_{i=1}^n p_i^2)^{1/2} * (\sum_{i=1}^n q_i^2)^{1/2}} \right)$$

Thus, two parallel vectors have a spectral angle of zero, while orthogonal vectors have a spectral angle of $\pi/2$. These angles are typically thresholded to determine class membership.

Usage:

Layers: Select the number of input spectra to classify the datacube against.

Spectrum: Select the input spectrum used for classification.

Outputs:

- Datacube with each band containing the spectral angle between input pixels and reference spectra in radians, ordered by input position of spectrum. Smaller numbers indicate greater similarity.

References:

1. https://pdfs.semanticscholar.org/c192/da3f6560c0305926149e7f6324dab441201b.pdf?_ga=2.90400281.2089631239.1589487588-904881142.1587062196

Spectral Unmix

Description:

Assuming that a signal at a given pixel is a linear combination of *pure* endmember spectra, this tool will return the relative abundances of the input endmember, constrained to a defined range. Useful for both fluorescence and remote sensing data.

Usage:

Constrain To: Constrain the output to the defined range.

Endmembers: Number of input endmember spectra. Select individual endmember spectra in the dropdown boxes.

Outputs:

- A datacube with each band containing the relative abundance of the endmember, ordered by input position order.

References:

1. https://www.sfpt.fr/hyperspectral/wp-content/uploads/2013/01/cours_Licciardi.pdf

Support Vector Machine

Description:

A SVM model is a representation of the training examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Usage:

Method: One of Linear, Poly, RBF, or LinearSVC, plus the option of returning probability (“Proba”) instead of discrete class predictions. Linear, Poly, or RBF specify the kernel function to use in SVC, while LinearSVC is very similar to Linear but scales better to large number of samples.

Auto Gamma?: Kernel coefficient for RBF and Polynomial functions.

Auto Weight?: Check to automatically adjust weight inversely proportional to class frequencies in the input data.

C: Inverse of regularization strength; smaller values specify stronger regularization.

Poly Degree: Degree of polynomial kernel function. Not used in other methods.

Classes: Classes are the discrete training data cubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings at the bottom of the dialog box. For each Class grouping, select the data cube in the Class field and the associated dependent value in the Group Number field.

Outputs:

- Datacube with each band containing a probability of a pixel being a member of that class, or a single band with integer encoding that represents class membership by input position order.

References:

1. https://en.wikipedia.org/wiki/Support_vector_machine
2. <https://scikit-learn.org/1.3/modules/svm.html>
3. <https://scikit-learn.org/1.3/modules/svm.html#svm-kernels>

k-Nearest Neighbors

Description:

Neighbors-based classification is computed from a simple majority vote of the nearest neighbors of each point based on a distance metric. To avoid prohibitively long computation times, it is recommended to reduce dimensionality before using k-NN.

Usage:

N Neighbors: Number of neighbors to use in vote.

Weights: Select *uniform* to have all points in neighborhood weighted equally. Select *distance* to weight by inverse distance.

Distance Metric: Distance metric used to compute similarity of neighbors.

- **euclidean** - ordinary straight line distance in n-dimensional space

$$d_{u,v} = \sqrt{\sum_{i=1}^{nbands} (u_i - v_i)^2}$$

- **manhattan** - the sum of the absolute differences in each dimension in n-dimensional space

$$d_{u,v} = \sum_{i=1}^{nbands} |u_i - v_i|$$

- **mahalanobis** - a statistically weighted distance that accounts for correlations and scale differences between bands.

$$d_{u,v} = \sqrt{(u - v)^T C^{-1} (u - v)}$$

where u and v are the point vectors being compared and C is the covariance matrix computed from the complete dataset.

- **minkowski** - a generalization of euclidean and manhattan distance defined as

$$d_{u,v} = \left(\sum_{i=1}^{nbands} |u_i - v_i|^p \right)^{1/p}$$

Setting P to 1 is equivalent to Manhattan distance, P = 2 is equivalent to Euclidean.

- **chebyshev** - the maximum distance in a single dimension between two point vectors

$$d_{u,v} = \max_{i=1}^{nbands} (|u_i - v_i|)$$

See references for more information.

Power (P): Power to be used in the Minkowski distance calculation. Setting P to 1 is equivalent to Manhattan distance, P = 2 is equivalent to Euclidean.

Probability: Check to return a probability of class membership, uncheck to return discrete predictions.

Classes: Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings below. For each Class grouping, select the datacube in the Class field and the associated dependent value in the Value field.

Outputs:

- Datacube with each band containing a probability of a pixel being a member of that class, or a single band with integer encoding that represents class membership by input position order.

References:

1. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
2. <https://scikit-learn.org/1.3/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
3. <https://scikit-learn.org/1.3/modules/generated/sklearn.neighbors.DistanceMetric.html>

11.1.3 Analyze

Band Ratio

Description:

Divide each pixel in one band by those in another band.

$$BR = \frac{Numerator}{Denominator}$$

Usage:

Numerator Wavelength: The wavelength of the band in the numerator of the ratio.

Denominator Wavelength: The wavelength of the band in the denominator of the ratio.

Outputs:

- A new single band floating point datacube from the ratio.

Custom Normalized Difference Index

Description:

Calculate a custom normalized difference index for each pixel, defined as:

$$NDI = \frac{B_1 - B_2}{B_1 + B_2}$$

If the denominator is zero for any pixels, the return value of those pixels will be set to zero.

Usage:

First Band: The wavelength of the first band.

Second Band: The wavelength of the second band.

Outputs:

- A floating point cube of the normalized difference.

First Derivative

Description:

Compute the first derivative along the spectral axis.

Usage:

Smoothed implementation: If true, the derivative is calculated using a Savitzky-Golay filter (recommended). If False, a simple band-to band difference is used (rise / run). This tends to produce noisy results. For fine grained control over the parameters of smoothing used, use the *Savitzky-Golay Smoothing* plugin instead of this version.

Outputs:

- A floating point datacube of the first derivative.

Matched Filter

Description: The matched filter is designed to detect the presence of and contribution of a known signal (target spectrum) within each pixel of a datacube.

Usage:

Target spectrum: The signal to detect.

Background cube: a cube that represents the typical background environment of the target area. By default, the plugin uses the currently loaded datacube as the background.

Outputs: A scalar indicating the strength of the match between the target and each pixel in the datacube. Higher values indicate greater contribution of the target spectrum.

Minimum Noise Fraction

Description:

The Minimum Noise Fraction is a linear transformation of two separate principal component analysis (PCA) rotations. The first rotation utilizes the principal components of the noise covariance matrix to ensure the noise has unit variance and are spectrally uncorrelated. The second PCA rotation is applied to the original data after it has been noise whitened by the first.

Usage:

Bands To Return: The number of rotated bands to retain after the transformation.

Standardize: If this is checked, data is normalized band-wise prior to the transformation by subtracting the mean and dividing by the standard deviation. Unless the input datacube has already been standardized, this should usually be set to True because PCA is sensitive to scale variations across bands.

Est. Noise from Dark Field: If checked, the user will be prompted for a dark noise cube to estimate noise from. If unchecked, the noise will be estimated from the input data.

Outputs:

- A new floating point cube containing the MNF scores of the transformed data.
- The scree plot of eigenvalues is plotted in the User Plot window.

References:

1. Green, A.A., M. Berman, p. Switzer and M.D. Craig (1988) A transformation for ordering multispectral data in terms of image quality with implications for noise removal. IEEE Transactions on Geoscience and Remote Sensing, 26(1):65-74.

PCA

Description:

Principal Component Analysis is an orthogonal linear transformation of data such that the new coordinate system orders bands by decreasing magnitude of explained total variance. The transformation is determined by the eigen decomposition of the covariance matrix. Typically, a relatively small subset of the resulting bands are retained after the coordinate rotation to reduce the dimensionality of the dataset.

Usage:

Bands To Return: The number of rotated bands to retain after the transformation.

Standardize: If this is checked, data is normalized bandwise prior to the transformation by subtracting the mean and dividing by the standard deviation. Unless the input datacube has already been standardized, this should usually be set to True because PCA is sensitive to scale variations across bands.

Save Transformation Matrix: If this is checked, a transformation matrix is saved to disk, which can be used to apply the same transformation to other cubes using the *PCA from Prior Transform* plugin.

Outputs:

- A new floating point cube containing the PCA scores of the transformed data.
- The scree plot of eigenvalues is plotted in the User Plot window.

References:

1. https://en.wikipedia.org/wiki/Principal_component_analysis

PCA from Prior Transform

Description:

Uses a previously saved PCA transformation (from the PCA plugin) to transform other data.

Usage:

Bands To Return: The number of rotated bands to retain after the transformation.

Standardize: If this is checked, data is normalized bandwise prior to the transformation by subtracting the mean and dividing by the standard deviation of the data used to originally fit the transform. Unless the input datacube has already been standardized, this should usually be set to True because PCA is sensitive to scale variations across bands.

PCA Transformation Matrix: Once OK has been pressed, the software will prompt for the previously saved transform.

Outputs:

- A new floating point cube containing the PCA scores of the transformed data.
- The scree plot of eigenvalues is plotted in the User Plot window.

References:

1. https://en.wikipedia.org/wiki/Principal_component_analysis

Regression

Description:

A collection of different statistical methods for predicting a dependent variable (or outcome variable), from one or more independent variables, typically bands in the context of hyperspectral imaging.

Usage:

Method: Can be one of Support Vector (Linear), Support Vector (Radial Basis Function), Ordinary Least Squares, Partial Least Squares, Lasso, Ridge, Ridge (with Cross Validation). Partial Least Squares is a good place to start. See references for more information on these methods.

Alpha: This parameter is used in Ridge and Lasso as the regularization strength parameter. Larger values increase regularization and reduce variance of the estimate.

Auto Gamma: Uncheck this box to specify gamma manually.

Gamma: This parameter is used in SVR RBF to set how far the influence of a single training example reaches, low values meaning greater influence. It can be thought of as the inverse of the radius of influence of model-selected support vectors.

Penalty: This parameter is the regularization parameter, C, used by SVR. The strength of the regularization is inversely proportional to C.

Fit Intercept: Used in OLS, Lasso, Ridge, and RidgeCV to determine whether to calculate the intercept. If data has already been centered, fitting an intercept is not necessary.

Number of Components To Keep: Used by Partial Least Squares as the number of components to use after features have been transformed. With too few components, predictive power may be lost. With too many, the model may be prone to overfitting.

Classes:

Classes are the discrete training datacubes used to build the predictive model. Enter the number of Classes to be used for training, and that number will be reflected in the number of Class groupings below. For each Class grouping, select the datacube in the Class field and the associated dependent value in the Value field.

Outputs:

- A floating point datacube containing the predicted dependent variable.

References:

1. https://en.wikipedia.org/wiki/Regression_analysis
2. https://scikit-learn.org/1.3/modules/generated/sklearn.linear_model.LinearRegression.html
3. https://scikit-learn.org/1.3/modules/generated/sklearn.cross_decomposition.PLSRegression.html

Second Derivative

Description:

Compute the second derivative along the spectral axis.

Usage:

Smoothed implementation: If true, the derivative is calculated using a Savitzky-Golay filter (recommended). If False, a simple band-to band difference is used (rise / run). This tends to produce noisy results. For fine grained control over the parameters of smoothing used, use the *Savitzky-Golay Smoothing* plugin instead of this version.

Outputs:

- A floating point datacube of the second derivative.

Thinfil Thickness

Description:

This algorithm finds the distance between peaks of interference spectra and uses this distance, along with the known Index of Refraction and angle of incidence, to compute film thickness in microns. The sample must be clean and have a mirror finish.

Usage:

Angle of Incidence: Angle between incident light and normal. Imager should be setup at the same angle on the other side of normal.

Index of Refraction: Measured or previously determined IOR of film.

Show Advanced Settings? Show smoothing and peaking finding settings.

SavGol Window Length: Length of Savistky-Golay smoothing window.

SavGol Order: Polynomial order of Savistky-Golay smoothing.

Peak Prominence: Threshold of peak prominence of peak. Lower numbers are more inclusive.

Show Example Fit? Check to see example of peak finding solution and the below line, sample.

Example Line Number: Line number of example peak finding plot.

Example Sample Number: Sample number of example peak finding plot.

Outputs:

- A floating point datacube of the film thickness in microns.

References:

1. https://en.wikipedia.org/wiki/Thin-film_interference
2. <http://www.shimadzu.com/an/uv/support/uv/ap/film.html>
3. https://en.wikipedia.org/wiki/Savitzky-Golay_filter

Total Radiance

Description:

Sums all bands of a datacube in units of radiance to produce total radiance. May also be used to sum any band data into a single band datacube.

Outputs:

- A single band floating point datacube.

11.1.4 Correct

Back Out Raw Data from Reflectivity

Description:

Convert reflectance data back to raw data from a datacube containing the original white reference and optional dark current datacube.

Usage:

The input datacube must be in units of reflectance.

100% Reflectivity Scaled To: The scaling of the input reflectance datacube.

Correction Cube: The white reference datacube used to originally convert the data to reflectance.

Dark Noise Cube: The optional dark noise datacube used to originally covert data to reflectance.

Outputs:

- A datacube of uncorrected (raw) data.

Correct Optical Distortion

Description:

A Brown-Conrady radial distortion model in the across track dimension to correct for distortions. No tangential distortion coefficients used in this model.

Usage:

K1 through K4: These are the radial distortion coefficients. Contact the objective lens manufacturer or Resonon to obtain these coefficients.

Outputs:

- A corrected datacube.

References:

1. [http://en.wikipedia.org/wiki/Distortion_\(optics\)#Software_correction](http://en.wikipedia.org/wiki/Distortion_(optics)#Software_correction)

Georectify Airborne Datacube (not installed)

Description:

Creates georectified output products based on GPS/IMU data.

Usage:

Use Flat Earth Elevation: Check to use flat earth elevation below, uncheck to use DEM file.

Flat Earth Elev. (m): The elevation of the ground in the area the datacube was collected.

Field of View (deg): The field of view of the imager/lens combination used.

Calculated Resolution (m): An estimate of the cross and along track resolutions using the FOV and average above ground height for cross track, and distance flown and frame rate for along track. The recommended resolution is twice the largest estimate, to account for platform instabilities.

Map Resolution (m): Desired resolution of output products, in meters.

Override heading with course? Overrides the heading with GPS course. This may be useful for poor heading accuracy datasets if the system is well aligned with the direction of aircraft flight.

Save Products in Source Folder?: Check to save all products (beside Full Datacube) in original folder. Uncheck to specify destination.

Generate Full Datacube?: Creates a georectified datacube and adds it to the Resource Tree in Spectronon. It is not saved automatically like the other output products.

Generate KML of Image?: Check to generate a KML file of the selected Image.

Generate GeoTiff of Image?: Check to generate a GeoTiff of the selected Image.

Generate Swath Outline?: Check to generate an outline of the FOV down the flight path, useful for debugging.

Interpolate Image?: If checked, the missing pixels of the GeoTiff/KML images will be interpolated.

Interpolate Datacube?: If checked, the missing pixels of the datacube will be interpolated.

Average Overlapping Pixels?: If checked, pixels with multiple projections will contain the average of all projected spectra. Otherwise the pixel will contain the spectrum of the last projection. Does not apply to images, only full datacubes.

Image To Use in KML/GeoTiff: Select which Image to use in KML/GeoTiff (order follows Resource Tree)

Stretch: Apply the specified stretch to GeoTiff/KML image. Absolute is recommended.

Low/High: Absolute stretch values.

Sync Offset (s): Time delay between image data and GPS/IMU.

Imager Roll Offset (deg): Angular offset in the roll direction between imager nadir and IMU nadir.

Imager Pitch Offset (deg): Angular offset in the pitch direction between imager nadir and IMU nadir.

Imager Heading Offset (deg): Angular offset between imager and IMU in the heading direction.

Crop Cube to Start/End Lines?: Used to crop beginning and or ending of datacube in the geocorrection process.

Outputs:

All outputs are optional.

- KML and GeoTiff of Image.
- A georectified datacube.
- Outline KML of imager FOV along flight path.
- Mask of interpolated pixels
- Interpolated LCF file

References:

1. <https://pdfs.semanticscholar.org/c434/80e5c049c0f162646166c893ab74bb3960d8.pdf>
2. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.590.6365&rep=rep1&type=pdf>
3. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.466.3776&rep=rep1&type=pdf>

Radiance From Raw Data

Description:

Converts raw data to units of radiance (microflicks) based on the instrument's radiometric calibration file (.icp). This process is outlined below:

1. The .icp file contains multiple calibration files, one *gain* cube that contains the pixel-by-pixel transfer from digital number to calibrated radiance and typically many *offset* cubes, which are dark noise cubes taken at different integration times and gains. This plugin automatically finds the best suited offset file to use (Auto Remove Dark) unless the user decides to provide one.
2. The gain and offset files are scaled to match the binning factor used in the datacube. Note that because the gain file represents the inverse instrument response, it gets scaled inversely.
3. The gain and offset data are averaged both spatially and spectrally to match the frame size of the datacube.
4. The gain and offset data are flipped spatially depending on the *flip radiometric calibration* header item in the datacube. (This header item is used to track left-right edges in airborne data.)
5. The gain data is scaled by the integration and gain ratios between it and the datacube. Note that gain in Resonon headers is 20 log, not 10 log.
6. The offset data is subtracted from the datacube. The result is multiplied by the gain file.

The resulting data is in units of microflicks.

$$\text{microflicks} = \frac{(\mu W)}{\text{sr} * \text{cm}^2 * \mu m}$$

Usage:

The datacube must be collected with a Resonon instrument with a supplied Imager Calibration Pack (.icp).

Auto Remove Dark Noise: Select to use the most appropriate dark noise file in the ICP file. Uncheck for user supplied dark noise datacube.

Return floating point: By default, the result is returned as a two byte integer because solar illuminated scenes in units of radiance will never exceed 2^{16} uFlicks. For very bright light sources or very dimly illuminated scenes, floating point numbers are necessary.

Set Saturated Pixels to Zero?: This option will set saturated pixels to zero after the radiance conversion process as it is difficult to detect saturation after radiance conversion otherwise.

Saturation Value: Level at which saturation occurs. Defaults to the ceiling value in the header or the bit depth value if ceiling is not available. User can override.

Outputs:

- A datacube containing calibrated radiance units of microflicks.

References:

1. <https://en.wikipedia.org/wiki/Radiance>
2. [https://en.wikipedia.org/wiki/Flick_\(physics\)](https://en.wikipedia.org/wiki/Flick_(physics))

Raw Data From Radiance

Description:

Converts radiance (microflicks) to units of raw data (DN) based on the instrument's radiometric calibration file (.icp). It is designed as the inverse of the Radiance From Raw Data plugin. This process is outlined below:

1. The .icp file contains multiple calibration files, one *gain* cube that contains the pixel-by-pixel transfer from digital number to calibrated radiance and typically many *offset* cubes, which are dark noise cubes taken at different integration times and gains. This plugin automatically finds the best suited offset file to use (Auto Remove Dark) unless the user decides to provide one.
2. The gain and offset files are scaled to match the binning factor used in the datacube. Note that because the gain file represents the inverse instrument response, it gets scaled inversely.
3. The gain and offset data are averaged both spatially and spectrally to match the frame size of the datacube.
4. The gain and offset data are flipped spatially depending on the *flip radiometric calibration* header item in the datacube. (This header item is used to track left-right edges in airborne data.)
5. The gain data is scaled by the integration and gain ratios between it and the datacube. Note that gain in Resonon headers is 20 log, not 10 log.
6. The datacube is divided by the gain file. The offset is added to the result.
7. The datacube is then clipped (np.clip) at 0 and the saturation value to correct for any errors in the inversion process.

The resulting data is in units of digital numbers.

$$\text{microflicks} = \frac{(\mu W)}{\text{sr} * \text{cm}^2 * \mu m}$$

Usage:

The datacube must be collected with a Resonon instrument with a supplied Imager Calibration Pack (.icp).

Undo Auto Remove Dark Noise: Select to use the most appropriate dark noise file in the ICP file. Uncheck for user supplied dark noise datacube. If the image was calibrated to radiance using a supplied dark noise datacube, that same datacube should be used to convert back to raw data.

Outputs:

- A datacube containing raw data in digital numbers.

References:

1. <https://en.wikipedia.org/wiki/Radiance>
2. [https://en.wikipedia.org/wiki/Flick_\(physics\)](https://en.wikipedia.org/wiki/Flick_(physics))

Reflectance from Radiance Data and Downwelling Irradiance Spectrum

Description:

Converts radiance data to reflectivity based on a downwelling irradiance spectrum collected with a single point spectrometer and cosine corrector. This approach assumes the surfaced imaged is Lambertian. Reflectance is then found by:

$$R = \pi * \frac{L}{E}$$

where L is the at-sensor radiance from the imager and E is the downwelling irradiance.

Usage:

Downwelling Type: Choose either Resonon supplied or other vendor calibrated unit.

Use Downwelling Spectrum in Source Folder: Check to automatically use spectrum in same folder as the source code. Uncheck for user supplied downwelling spectrum (open in Spectronon)

Downwelling Calibration: Resonon supplied downwelling calibration file (.dcp)

Auto Remove Downwelling Dark Noise: Check to use previously measured dark noise (inside DCP file). Uncheck for user supplied dark noise spectrum.

Scale 100% Reflectivity To: Scaling factor for output datacube.

Use Correlation Coefficients: Use previously computed Correlation Coefficients to align downwelling data to ground truth.

Outputs:

- Datacube containing reflectivity values.

References:

1. http://www.oceanopticsbook.info/view/surfaces/lambertian_brdfs

Reflectance from Radiance Data and Measured Reference Spectrum

Description:

This approach to converting data to reflectance assumes the input datacube has been corrected spatially, typically by converting to radiance but other approaches would be acceptable. A spectrum of measured reference material is used to correct the datacube spectrally. This spectrum should be collected with the same instrument under the same conditions as the datacube to be collected. The measured reflectance should be a tab, space, or comma delimited file with the first column containing wavelength in nanometers and the second column containing reflectance, scaled to either 0-1 or 0-100%.

Usage:

The input datacube should be in units of radiance (or spatially corrected by other means). The input spectrum is typically created with a region-of-interest tool of reference material in a datacube collected with the same imager and under the same conditions.

Measured Reflectivity: Click this button to select the tab, space, or comma delimited file containing the measured reflectance spectrum of the reference material.

Measured Reflectivity in Percentage?: Check this box if measured reflectivity is scaled to 0-100. If unchecked, a scale of 0-1 is assumed.

ROI Spec: Spectrum of measured reference material.

Scale 100% Reflectivity To: Scaling factor for output datacube.

Outputs:

- A floating point datacube of reflectance data, scaled to the selected factor.

Reflectance from Radiance Data and Spectrally Flat Reference Spectrum

Description:

This approach assumes the input datacube has been corrected spatially, typically by converting to radiance but other approaches would be acceptable. A spectrum of spectrally flat reference material (typically SpectralonTM or FluorilonTM) is used to correct the datacube spectrally. This spectrum should be collected with the same instrument under the same conditions as the datacube to be corrected.

Usage:

The input datacube should be in units of radiance (or spatially corrected by other means). The input spectrum is typically created with a region-of-interest tool of spectrally flat material in a datacube collected with the same imager and under the same conditions.

ROI Spec: Spectrum of spectrally flat reference material.

Flat Reflectivity (0-1): Reflectivity of reference material on a scale of 0-1 (0-100%).

Scale 100% Reflectivity To: Scaling factor for output datacube.

Outputs:

- A floating point datacube of reflectance data, scaled to the user's choice.

Reflectance from Raw Data and Downwelling Irradiance Spectrum

Description:

Converts raw data to reflectivity based on a downwelling irradiance spectrum collected with a single point spectrometer and cosine corrector. This approach assumes the surfaced imaged is Lambertian. Reflectance is then found by:

$$R = \pi * \frac{L}{E}$$

where L is the at-sensor radiance from the imager and E is the downwelling irradiance.

Usage:

Imager Calibration: Resonon supplied imager calibration (.icp) file

Auto Remove Dark Noise: Check to use previously measured dark noise (inside ICP file). Uncheck for user supplied dark noise cube.

Downwelling Type: Choose either Resonon supplied or other vendor calibrated unit.

Use Downwelling Spectrum in Source Folder: Check to automatically use spectrum in same folder as the source code. Uncheck for user supplied downwelling spectrum (open in Spectronon)

Downwelling Calibration: Resonon supplied downwelling calibration file (.dcp)

Auto Remove Downwelling Dark Noise: Check to use previously measured dark noise (inside DCP file). Uncheck for user supplied dark noise spectrum.

Scale Downwelling Dark Noise to Signal Tail: This feature estimates the dark noise of the downwelling sensor at bands with no signal and scales the dark correction to match.

Scale 100% Reflectivity To: Scaling factor for output datacube.

Use Correlation Coefficients: Use previously computed Correlation Coefficients to align downwelling data to ground truth.

Set Saturated Pixels to Zero?: This option will set saturated pixels to zero after the radiance conversion process as it is difficult to detect saturation after radiance conversion otherwise.

Saturation Value: Level at which saturation occurs. Defaults to the ceiling value in the header or the bit depth value if ceiling is not available. User can override.

Outputs:

- Datacube containing reflectivity values.

References:

1. http://www.oceanopticsbook.info/view/surfaces/lambertian_brdfs

Reflectance from Raw Data and Measured Reference Cube

Description:

Convert raw data to reflectance based on a input datacube of measured reference material. This datacube needs to have the reference material fill the entire FOV of the instrument, and be collected with the same instrument under the same conditions as the datacube to be corrected. The measured reflectance should a tab, space, or comma delimited file with the first column containing wavelength in nanometers and the second column containing reflectance, scaled to either 0-1 or 0-100%.

Usage:

The input datacube should be in units of radiance (or spatially corrected by other means). The input spectrum is typically created with a region-of-interest tool of reference material in a datacube collected with the same imager and under the same conditions.

Measured Reflectivity: Click this button to select the tab, space, or comma delimited file containing the measured reflectance spectrum of the reference material.

Measured Reflectivity in Percentage?: Check this box if measured reflectivity is scaled to 0-100. If unchecked, a scale of 0-1 is assumed.

ROI Spec: Spectrum of measured reference material.

Scale 100% Reflectivity To: Scaling factor for output datacube.

Outputs:

- A floating point datacube of reflectance data, scaled to the user's choice.

Reflectance from Raw Data and Spectrally Flat Reference Cube

Description:

Converts raw data to reflectivity based on another datacube of a spectrally flat reference material. This datacube needs to have the reference material fill the entire FOV of the instrument, and be collected with the same instrument under the same conditions as the datacube to be corrected.

Usage:

Correction Cube: Datacube of spectrally flat reference material.

Flat Reflectivity (0-1): Reflectivity of reference material on a scale of 0-1 (0-100%).

Scale 100% Reflectivity To: Scaling factor for output datacube.

Outputs:

- A floating point datacube of reflectance data, scaled to the user's choice.

Remove Dark Noise From Cube

Description:

Subtracts a frame of dark noise from a datacube. This frame can be a single frame of data, or a datacube of dark noise. If a datacube is used, its lines are averaged to create a single frame.

Usage:

Dark Noise Cube: Datacube of recorded dark noise.

Outputs:

- A datacube with dark noise removed.

11.1.5 Anomaly

Elliptic Envelope Anomaly

Description: Anomalies are detected by estimating a robust covariance of the dataset, fitting an ellipse to the central data points, and ignoring those outside. Mahalanobis distances are used to determine a measure of a point's likelihood of being an outlier.

Usage:

The method assumes a Gaussian distributed dataset. For other distributions, other algorithms may perform better, such as SVM Anomaly with the RBF kernel.

Outlier detection from covariance estimation may break or not perform well in high-dimensional settings. In particular, one will always take care to work with $n_samples > n_features ** 2$. Feature reduction should be performed prior in order to accelerate run times.

Clutter: Datacube containing only the statistical normal background, without any outliers.

Outputs: Floating point datacube containing +1 for inliers and -1 for outliers.

References:

1. <https://scikit-learn.org/1.3/modules/modules/generated/sklearn.svm.OneClassSVM.html>
2. https://scikit-learn.org/1.3/modules/outlier_detection.html

Least Squares Anomaly

Description: Probabilistic method for the detection of anomalous outliers based on a squared-loss function.

Usage:

Clutter: Datacube containing only the statistical normal background, without any outliers.

Probability: If checked, the probability of a pixel being anomalous is returned, scaled 0-1. If unchecked, a classification is returned, with 0 being inliers and 1 for outliers.

Outputs: If Probability is checked, a floating point datacube containing a 0-1 probability is returned. If not, a floating point datacube of 0 for inliers and 1 for outliers is returned.

References:

1. J.A. Quinn, M. Sugiyama. A least-squares approach to anomaly detection in static and sequential data. Pattern Recognition Letters 40:36-40, 2014.

RX Anomaly Detector

Description:

The Reed-Xiaoli (RX) anomaly detector detects pixels determined to be spectrally anomalous relative to a user-defined background or neighboring region. It assumes that the background can be modeled by a multivariate Gaussian distribution. The Mahalanobis distance between a given pixel and the background model is calculated and compared to a user-defined threshold, determining whether the pixel belongs to the background or is an anomaly.

$$\delta_{RX}(r) = (r - \mu)^T K_{LxL}^{-1} (r - \mu)$$

Usage:

Global: Compares each pixel to the entire datacube.

Local (Lines): Compares each pixel to the image area within a user-defined number of lines of the pixel under test.

Local (Squares): Compares each pixel to a square image area of the user-defined size adjacent to the pixel under test.

User Selected: Compares each pixel to a user supplied datacube that defines the expected background (also known as clutter).

Outputs:

A floating point datacube containing the Mahalanobis distance to the selected background area. The default Image for this cube is the Threshold To Colormap, which allows the user to threshold the results to better highlight anomalies.

References:

1. Reed I., and X. Yu, "Adaptive Multiple-Band CFAR Detection of an Optical Pattern with Unknown Spectral Distribution." IEEE Transactions on Acoustics, Speech and Signal Processing 38 (1990): 1760-1770.

SVM Anomaly

Description: Anomalies are detected using a One Class Support Vector Machine with a user-supplied background (or clutter) datacube. This cube should contain only statistical *normal* pixels (without anomalies).

Usage:

As with other Support Vector Machines, feature reduction should be performed prior in order to accelerate run times.

Method: Can be one of Linear, Radial Basis Function (RBF), or Polynomial. See references for more information on these methods.

Clutter: Datacube containing only the statistical normal background, without any outliers.

Nu: Upper bound on the fraction of training errors and a lower bound of the fraction of support vectors.

Auto Gamma: If selected, the gamma kernel coefficient is set to 1/(number of features).

Gamma: Kernel coefficient for RBF and polynomial functions.

Degree: Degree of the polynomial kernel function.

Outputs: Floating point datacube containing +1 for inliers and -1 for outliers.

References:

1. <https://scikit-learn.org/1.3/modules/generated/sklearn.svm.OneClassSVM.html>
2. https://scikit-learn.org/1.3/modules/outlier_detection.html

11.1.6 Color

CIE Colorspace Conversion

Description:

The CIE colorspace is a color coordinate system based on the human eye's response to light and color. It is a mathematical generalization of human color vision, allowing one to define and reproduce colors in a objective way.

Usage:

The input datacube must be in units of reflectance with a *reflectance scale factor* header item, ideally with a spectral range of 390-700 nm.

Illuminant: The theoretical source of light in order to compare colors across different lighting.

Standard Observer: The observer function is the model of human vision produced from color matching experiments. The response of the eye is a function of the field of view, standard values being 10 and 2 degrees.

XYZ: The CIE XYZ color space is the extrapolation of RGB to avoid negative numbers. The Y parameter is a measure of the luminance, with X and Z specifying the chromaticity, with Z somewhat equal to blue and X is a mix of cone response chosen to be orthogonal to luminance.

xyY: In CIE xyY, Y is the luminance and x and y represents the chrominance values derived from the tristimulus values X, Y and Z in the CIE XYZ color space.

*L*a*b*:* L*a*b* is a color model and space in which L is brightness and a and b are chrominance components. It contains color values far more than the human gamut, but designed to be device independent.

Outputs:

- A floating point datacube containing XYZ, xyY, and/or L*a*b* values.

References:

1. https://en.wikipedia.org/wiki/CIE_1931_color_space
2. https://en.wikipedia.org/wiki/Standard_illuminant

Delta E*

Description:

The distance metric of Delta E* is measure of color difference in L*a*b* space. It is meant that a Delta E* of one is a *just noticeable difference* across the entire gamut. In practice, a Delta E* of about 2.3 is just noticeable. It is defined as:

$$\Delta E_{ab}^* = \sqrt{(L_2^* - L_1^*)^2 + (a_2^* - a_1^*)^2 + (b_2^* - b_1^*)^2}$$

Usage:

The input datacube must have L*a*b* bands named L* (L*a*b*), a* (L*a*b*), b* (L*a*b*).

Member Count: The number of L*a*b* values to compute distances to.

L*, a*, b*: Values to find Delta E* distances to.

Outputs:

- A floating point datacube containing Delta E* distances to each input L*a*b* value.

References:

1. https://en.wikipedia.org/wiki/Color_difference

XYZ to RGB Colorspace Conversion**Description:**

Convert CIE XYZ colorspace to a RGB colorspace.

Usage:

The input datacube must have bands containing XYZ data, named *X (XYZ)*, *Y (XYZ)*, and *Z (XYZ)*.

RGB Working Space: Desired RGB space to output.

Companding: The desired non-linear transformation of intensity values.

Result datatype: The desired output datatype and bit-depth.

Outputs:

- A floating point datacube containing RGB values.

References:

1. https://en.wikipedia.org/wiki/CIE_1931_color_space
2. https://en.wikipedia.org/wiki/Color_spaces_with_RGB_primaries
3. <https://en.wikipedia.org/wiki/SRGB>

11.1.7 Agricultural**Anthocyanin Reflectance Index 1****Description:**

Index sensitive to relative concentrations of anthocyanin versus chlorophyll pigments. As plants are stressed the leaves contain higher concentrations of anthocyanin pigments. New leaves also have high anthocyanin concentrations. Data should be in units of reflectance.

$$AR1 = \frac{1}{Green} - \frac{1}{Red}$$

The wavelengths of Green and Red are 550 and 700 nm respectively.

Outputs:

- A floating point datacube of CRI data.

References:

1. Gitelson, A., M. Merzlyak, and O. Chivkunova. "Optical Properties and Nondestructive Estimation of Anthocyanin Content in Plant Leaves." *Photochemistry and Photobiology* 71 (2001): 38-45

Anthocyanin Reflectance Index 2

Description:

Index sensitive to relative concentrations of anthocyanin versus chlorophyll pigments. As plants are stressed the leaves contain higher concentrations of anthocyanin pigments. New leaves also have high anthocyanin concentrations. Data should be in units of reflectance.

$$ARI2 = NIR * \left(\frac{1}{Green} - \frac{1}{Red} \right)$$

The wavelengths of Green, Red, and NIR are 550, 700, and 800 nm respectively.

Outputs:

- A floating point datacube of ARI2 data.

References:

1. Gitelson, A., M. Merzlyak, and O. Chivkunova. "Optical Properties and Nondestructive Estimation of Anthocyanin Content in Plant Leaves." *Photochemistry and Photobiology* 71 (2001): 38-45

Atmospherically Resistant Vegetation Index

Description:

Atmospherically Resistant Vegetation Index robust to atmospheric aerosols and topographic effects.

$$ARVI = \frac{NIR - 2 * Red + Blue}{NIR + 2 * Red + Blue}$$

The wavelengths of the NIR, Red, and Blue bands are 800, 680, and 450 nm respectively.

Outputs:

- A floating point datacube of ARVI data.

References:

1. <https://ntrs.nasa.gov/search.jsp?R=19920059657>

Carotenoid Reflectance Index 1

Description:

Index sensitive to carotenoid versus chlorophyll pigments. As plants are stressed the leaves contain higher concentrations of carotenoid pigments. Values for green vegetation range from 1 to 12. Data should be in units of reflectance.

$$CRI = \frac{1}{Green1} - \frac{1}{Green2}$$

The wavelengths of Green1 and Green2 are 510 and 550 nm respectively.

Outputs:

- A floating point datacube of CRI data.

References:

1. Gitelson, A., M. Merzlyak, and O. Chivkunova. "Optical Properties and Nondestructive Estimation of Anthocyanin Content in Plant Leaves." *Photochemistry and Photobiology* 71 (2001): 38-45

Carotenoid Reflectance Index 2

Description:

Index sensitive to carotenoid versus chlorophyll pigments. As plants are stressed the leaves contain higher concentrations of carotenoid pigments. Values for green vegetation range from 1 to 11. Data should be in units of reflectance.

$$CRI2 = \frac{1}{Green} - \frac{1}{Red}$$

The wavelengths of Green and NIR are 510 and 700 nm respectively.

Outputs:

- A floating point datacube of CRI data.

References:

1. Gitelson, A., M. Merzlyak, and O. Chivkunova. "Optical Properties and Nondestructive Estimation of Anthocyanin Content in Plant Leaves." *Photochemistry and Photobiology* 71 (2001): 38-45

Enhanced Vegetation Index

Description:

Enhanced Vegetation Index is optimized for vegetation signal and reduced atmospheric influence.

$$EVI = 2.5 * \frac{NIR - Red}{NIR + 6 * Red - 7.5 * Blue + 1}$$

The wavelengths of the NIR, Red, and Blue bands are 800, 680, and 450 nm respectively.

Outputs:

- A floating point datacube of EVI data.

References:

1. https://en.wikipedia.org/wiki/Enhanced_vegetation_index

Modified Chlorophyll Absorption Ratio Index Improved (MCARI2)

Description:

Index sensitive to the relative abundance of chlorophyll. An improvement of MCARI for better prediction of Leaf Area Index.

$$MCARI2 = \frac{1.5 * (2.5 * (NIR - Red) - 1.3 * (NIR - Green))}{\sqrt{(2 * NIR + 1)^2 - (6 * NIR - 5 * \sqrt{Red})}} - .5$$

The wavelengths of Green, Red, and NIR are 550, 670 and 800 nm respectively.

Outputs:

- A floating point datacube of MCARI2 data.

References:

1. Haboudane, D., et al. "Hyperspectral Vegetation Indices and Novel Algorithms for Predicting Green LAI of Crop Canopies: Modeling and Validation in the Context of Precision Agriculture." *Remote Sensing of Environment* 90 (2004): 337-352.

Modified Chlorophyll Absorption Reflectance Index

Description:

Index sensitive to the relative abundance of chlorophyll.

$$MCARI = ((NIR - Red) - .2 * (NIR - Green)) * \left(\frac{NIR}{Red} \right)$$

The wavelengths of Green, Red, and NIR are 550, 670 and 700 nm respectively.

Outputs:

- A floating point datacube of MCARI data.

References:

1. Daughtry, C., et al. "Estimating Corn Leaf Chlorophyll Concentration from Leaf and Canopy Reflectance." Remote Sensing Environment 74 (2000): 229-239.

Modified Red Edge Normalized Vegetation Index

Description:

Narrowband vegetation index robust to leaf specular reflections.

$$MRENDVI = \frac{NIR - Red}{NIR + Red - 2 * Blue}$$

The wavelengths of the NIR, Red, and Blue bands are 750, 705, and 445 nm respectively. Vegetation has values typically between .2 and .7.

Outputs:

- A floating point datacube of MRESRI data.

References:

1. https://www.researchgate.net/publication/323723081_Vegetation_Indices_Combining_the_Red_and_Red-Edge_Spectral_Information_for_Leaf_Area_Index_Retrieval

Modified Red Edge Simple Ratio Index

Description:

Narrowband vegetation index robust to leaf specular reflections.

$$MRESRI = \frac{NIR - Blue}{NIR + Red - Blue}$$

The wavelengths of the NIR, Red, and Blue bands are 750, 705, and 445 nm respectively. Vegetation has values typically between 2 and 8.

Outputs:

- A floating point datacube of MRESRI data.

References:

1. https://www.researchgate.net/publication/323723081_Vegetation_Indices_Combining_the_Red_and_Red-Edge_Spectral_Information_for_Leaf_Area_Index_Retrieval

Normalized Difference Vegetation Index (NDVI)

Description:

NDVI is an indicator of live green vegetation, primarily qualitatively but also useful as a quantitative tool. It serves as an estimate of the contrast of the red-edge feature of chlorophyll. It is defined as:

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

The wavelengths of the NIR and Red bands are 800 and 680 nm respectively. NDVI returns a value between -1 and 1. Green vegetation is typically between .2 and .8.

Outputs:

- A floating point datacube of NDVI data ranging between -1 and 1.

References: 1. https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index

Photochemical Reflectance Index

Description:

Vegetation index sensitive to carotenoid pigments in live foliage.

$$PRI = \frac{Green - Yellow}{Green + Yellow}$$

The wavelengths of the Green and Yellow are 570 and 531 nm respectively.

Outputs:

- A floating point datacube of PRI data.

References:

1. https://en.wikipedia.org/wiki/Photochemical_Reflectance_Index

Plant Senescence Reflectance Index

Description:

Plant Senescence index sensitive to ratio of carotenoid pigments to chlorophyll.

$$PSRI = \frac{Red - Green}{NIR}$$

The wavelengths of the NIR, Green and Red are 750, 500 and 680 nm respectively.

Outputs:

- A floating point datacube of PSRI data.

References:

1. <https://www.indexdatabase.de/db/i-single.php?id=69>

Red Edge Normalized Difference Vegetation Index

Description:

Similar to NDVI but using the middle of the Red Edge for the red band.

$$RENDVI = \frac{NIR - Red}{NIR + Red}$$

The wavelengths of the NIR and Red bands are 750 and 705 nm respectively. NDVI returns a value between -1 and 1. Green vegetation is typically between .2 and .9.

Outputs:

- A floating point datacube of RENDVI data.

References:

1. https://en.wikipedia.org/wiki/Normalized_Difference_Red_Edge_Index

Simple Ratio Index

Description:

Simple Ratio is a quick way to distinguish green plants from a background. It is defined as:

$$SR = \frac{NIR}{Red}$$

The wavelengths of the NIR and Red bands are 850 and 675 nm respectively. Green vegetation has a resulting ratio much greater than one and most other objects close to one.

Outputs:

- A floating point datacube of SR data.

References:

1. <https://www.hiphen-plant.com/blog/vegetation-indices/>

Structure Insensitive Pigment Index

Description:

Vegetation index sensitive to carotenoid pigments while minimizing impact of canopy structure.

$$SIPI = \frac{NIR - Blue}{NIR + Red}$$

The wavelengths of the NIR, Blue and Red are 800, 445 and 680 nm respectively.

Outputs:

- A floating point datacube of SIPI data.

References:

1. <https://www.sentinel-hub.com/eopproducts/sipi-structure-insensitive-pigment-index>

Transformed Chlorophyll Absorption Reflectance Index

Description:

Index sensitive to the relative abundance of chlorophyll, impacted by signal from soil if the leaf area index is low.

$$TCARI = 3 * ((NIR - Red) - .2 * (NIR - Green)) * \left(\frac{NIR}{Red} \right)$$

The wavelengths of Green, Red, and NIR are 550, 670 and 700 nm respectively.

Outputs:

- A floating point datacube of TCARI data.

References:

1. Haboudane, D., et al. "Hyperspectral Vegetation Indices and Novel Algorithms for Predicting Green LAI of Crop Canopies: Modeling and Validation in the Context of Precision Agriculture." Remote Sensing of Environment 90 (2004): 337-352.

Vogelmann Red Edge Index 1

Description:

Narrowband vegetation index sensitive to chlorophyll concentration, leaf area, and water content.

$$VREI1 = \frac{NIR}{Red}$$

The wavelengths of the NIR and Red are 740 and 720 nm respectively. Vegetation has values typically between 4 and 8.

Outputs:

- A floating point datacube of VREI1 data.

References:

1. http://www.cs.cmu.edu/~casc/specialty_crops_workshop_2012/07a-StressDetection.pdf

Vogelmann Red Edge Index 2

Description:

Narrowband vegetation index sensitive to chlorophyll concentration, leaf area, and water content.

$$VREI2 = \frac{NIR2 - NIR1}{NIR4 + NIR3}$$

The wavelengths of the NIR1, NIR2, NIR3, and NIR5 are 747, 734, 726, and 715 nm respectively. Vegetation has values typically between 4 and 8.

Outputs:

- A floating point datacube of VREI2 data.

References:

1. http://www.cs.cmu.edu/~casc/specialty_crops_workshop_2012/07a-StressDetection.pdf

Vogelmann Red Edge Index 3

Description:

Narrowband vegetation index sensitive to chlorophyll concentration, leaf area, and water content.

$$VREI3 = \frac{NIR2 - NIR1}{NIR4 + NIR3}$$

The wavelengths of the NIR1, NIR2, NIR3, and NIR5 are 747, 734, 720, and 715 nm respectively. Vegetation has values typically between 4 and 8.

Outputs:

- A floating point datacube of VREI3 data.

References:

1. http://www.cs.cmu.edu/~casc/specialty_crops_workshop_2012/07a-StressDetection.pdf

Water Band Index

Description:

Index sensitive to water concentration in vegetation canopies. Smaller numbers mean higher water content.

$$WBI = \frac{NIR2}{NIR1}$$

The wavelengths of NIR1 and NIR2 are 900 and 970 nm respectively.

Outputs:

- A floating point datacube of WBI data.

References:

1. Penuelas, J., et al. "The Reflectance at the 950-970 Region as an Indicator of Plant Water Status." International Journal of Remote Sensing 14 (1993): 1887-1905.

11.1.8 Clustering

HDBSCAN Clustering

Description:

HDBSCAN is a clustering tool that allows varying density clusters and does not force outliers into clusters. This is an important feature when working with noisy data. It is a multi-step algorithm summarized in 5 steps below:

1. Transform the space according to the density estimate.
2. Build the minimum spanning tree of the distance weighted graph via Prim's algorithm.
3. Construct a cluster hierarchy of connected components.
4. Condense the cluster hierarchy based on minimum cluster size.
5. Extract the stable clusters from the condensed tree.

Usage:

Note: This algorithm can require long run times. Dimensionality of features should be reduced first.

Minimum Cluster Size: Sets smallest size grouping to be considered a cluster.

Minimum Samples: Sets a measure of how conservative the clustering should be. The larger the number, the more points will be declared noise.

Outputs:

- A floating point datacube containing integer cluster numbers with -1 representing noise.

References:

1. https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html
2. https://en.wikipedia.org/wiki/Prim%27s_algorithm

K-Means Clustering

Description:

Partitioning algorithm that groups each sample to the cluster with the nearest mean. Iterations then successively recompute cluster means and distances until convergence or the maximum number of iterations has been met.”

Usage:

Note: As with most clustering algorithms, dimensionality of features should be reduced first if possible.

Clusters: Number of desired clusters.

Max Iterations: Maximum iterations to be used. The higher the number the slower the run time but more accurate the results. If convergence has been met the algorithm will stop before the maximum number of iterations has occurred.

Outputs:

- A floating point datacube containing integer cluster numbers.

References:

1. https://en.wikipedia.org/wiki/K-means_clustering

11.1.9 Mask

Apply Mask

Description:

Overlays the supplied mask on the datacube. Pixels with mask values of one get passed through where mask pixels with zeros are set to a user supplied number or cropped. If cropped, spatial relationships of the datacube will likely be lost. This is useful for creating training datacubes for other classification tools.

Masks must be the same spatial dimensions as the datacube they operate on.

Usage:

New Name: New name for resulting datacube.

Technique: For Set Unmasked to Value, masked pixels will be set to the Mask Value. If set to Crop Masked, the pixels will be cropped from the datacube, likely losing the spatial relationship of the pixels as a result. This can be used to build training datacubes for use with classification tools.

Mask Value: Value to set unmasked pixels to.

Mask Cube: Mask datacube.

Outputs: Datacube with mask applied.

Build Mask

Description:

Builds a mask to apply with the Apply Mask tool. This tool uses a threshold to partition pixels into ones or zeros. In the Apply Mask tool, the mask is overlayed on a datacube and pixels with ones get passed through where pixels with zeros are set to a user supplied number or cropped. Typically, the Build Mask tool is applied to a previously classified datacube which can be partitioned with a threshold.

Usage:

Threshold: Threshold to partition data with.

Invert: If checked, pixels above the threshold are set to zero and pixels below the threshold are set to one. If unchecked, the opposite is true.

Outputs:

- A mask datacube containing ones and zeros.

Build Mask from Saturated Spectra

Description:

Builds a mask from saturated spectra to apply with the Apply Mask tool. In the Apply Mask tool, the mask is overlayed on a datacube and pixels with ones get passed through where pixels with zeros are set to a user supplied number or cropped.

Usage:

Saturation Value: Level at which saturation occurs. Defaults to the ceiling value in the header or the bit depth value if ceiling is not available. User can override.

Invert: If checked, saturated pixels are set to one and pixels below the threshold are set to zero. If unchecked, the opposite is true (normal usage).

Outputs:

- A mask datacube containing ones and zeros.

11.2 Render Plugins

11.2.1 Band Average

Description:

Averages all of the bands and displays the greyscale result.

11.2.2 Classification to Colormap

Description:

Display a unique color for each class, where pixel membership is given by the band number in which the maximum or minimum value occurs.

Usage:

Evaluate by: Choose whether class membership is evaluated by minimum value or maximum.

Colors: Click on class color button to choose color.

11.2.3 Foreground Object Detection

Description:

This powerful tool is useful for extracting objects from their backgrounds for creating training sets, making masks, or for visualization purposes.

The tool has two distinct steps. First, input the Background cube or cubes and the known options below. It will return a preview of the results, which can be used to fine tune the output. (Use the Auto Update checkbox to streamline the fine tuning process.) Once satisfied, press the Run Now button in the Export Results box.

Usage: *Number of Background Cubes:* The number of different background cubes. If a spectrum in the target datacube is similar to any of the background cubes, it will be considered background.

Use per-cube thresholds: If checked, each background cube will have a unique threshold slider. If unchecked, a single threshold will be applied to all background cubes.

Threshold: Threshold to fine tune the separation of foreground from background for all cubes (only visible if *Use per-cube thresholds* is unchecked).

Background Cube -> Cube: Select a datacube representative of the background to remove.

Background Cube -> Threshold: Threshold to fine tune the separation of foreground from background for this cube only (only visible if *Use per-cube thresholds* is checked).

Minimum Object Size: Smallest size in pixels for foreground object to extract. Can be used to filter out spatial noise.

Rectangular Cube: Check to preserve the spatial representation of the image and return a square cube. The background pixels needed to make a square image will be padded with zeros, which will contaminate training cubes. If Unchecked, spatial positioning will be scrambled but padding pixels are not necessary. Use this option for creating training cubes. The outline showing the foreground datacubes will disappear if this option is unchecked.

Combine Objects: Check to combine individual objects into a single datacube or mean spectrum. Uncheck to return all individuals.

Outline color: Press to select a different color to outline the results.

Morphological Opening: Erodes then dilates an image, useful for removing small objects.

Morphological Closing: Dilates then erodes an image, useful for filling small holes.

Border Pixel Erosion: Shrinks the foreground by expanding the background, removing unwanted edge pixels.

Structuring Element Size: Size of the structuring element to user in Opening, Closing, and Erosion. The larger the element, the greater the effect.

Save To Disk: Check cube and/or spectra to save them to disk. Specify directory below. Spectra are the mean spectra of the foreground objects, or mean of the combined foreground objects if chosen.

Spectrum Save Format: Select to save spectra as .spec, .txt, or .csv.

Save Directory: Directory to save results to.

Add To Workbench: Check to add datacubes, spectra, and/or mask cubes to the Spectronon Resource Tree (Workbench).

Run Now: Press button to execute.

Outputs:

- Datacubes and/or spectra saved to disk (optionally)
- Datacubes, spectra, and or mask cubes added to Workbench (optionally)

11.2.4 Greyscale

Description:

View a greyscale image by wavelength, band number, or band name.

Mode: Choose to view by wavelength, band number, or band name, if available.

Wavelength/Band/Band Name selector: Choose wavelength/band to view.

11.2.5 Greyscale to Single Color

Description:

Assigns a color to a single band datacube.

Usage:

Color: Select Red, Green, or Blue.

Invert: Check to make lower pixel values the brightest.

Clip Negatives?: Check to clip negative values to zero for display purposes.

11.2.6 Last Frame

Description:

View the last raw data frame of the datacube. A frame is a 2D image with a spatial dimension on one axis and spectral on the other.

11.2.7 RGB

Description:

View three bands of a datacube with a composite Red, Green, and Blue (RGB) image.

Usage:

Mode: Specify band selection by wavelength or band number.

Adjustments (Red, Green, and Blue): Select wavelength or band number for each color.

Presets: Depending on the wavelength range of the datacube, the Preset buttons will specify the above wavelengths for Red, Green, and Blue to standard wavelengths. In the VNIR, the options are True Color (normal color image) or Color Infrared (uses Green, Red, and Infrared wavelengths).

11.2.8 Raw Camera Data

Description:

View raw data frame from a datacube. A frame is a 2D image with a spatial dimension on one axis and spectral on the other.

Usage:

Frame Number: Select the frame number (line number) to view. -1 is the last frame, 0 is the first.

11.2.9 Scalar to Colormap

Description:

Assigns a unique color to each scalar value in a chosen band.

Usage:

Colormap: Choose colormap to use.

Normalize: Normalize band before assigning color.

Mode: Choose band selection by number or name.

Band: Band to visualize.

References:

1. https://matplotlib.org/3.1.1/gallery/color/colormap_reference.html

11.2.10 Stack

Description:

Stack multiple images on top of one another with transparency. This can be useful to show classification results on top of the original RGB image, or show multiple classification results simultaneously.

Usage:

Stack Height: Number of images to stack.

Image: Select image to show.

Alpha: Sets transparency of the image. 1 is no transparency.

Make zero values transparent: Useful for classified or mask datacubes.

11.2.11 Threshold To Colormap

Description: Assigns colors to pixels of a datacube that are above or below a user-specified threshold. Useful for datacubes containing a continuous range of values (such as spectral angles or probability results) that must be thresholded for final classification. One user-specified threshold is available for each band of the datacube.

If multiple bands of a pixel meet the threshold requirement, the color corresponding to the largest band values that meet the threshold (in *probability* or *above threshold* mode) or smallest band values (in *distance* or *below threshold* mode).

Pixels that do not meet the requirements of any threshold are colored black.

Usage:

Classification type: Determines whether colors are assigned corresponding to pixel values above the threshold or below the threshold. In *probability* or *above threshold* mode, colors are assigned corresponding to pixel values that are greater than the threshold. In *distance* or *below threshold* mode, colors are assigned to pixel values that are less than the threshold. *Probability* mode sets the range of possible thresholds between 0 and 1. *Distance* and *below threshold* modes are equivalent.

Value: Threshold value.

Color: Color to assign.

11.3 Filter Plugins

11.3.1 Colormap Stretch...

Description:

Assigns a color to individual scalars in a single band image. Similar to the Scalar to Colormap Image tool.

Usage:

Use Contrast Enhancement: Check to stretch image by below method (Linear Percentage, Linear Absolute, or Histogram).

Colormap: Choose colormap to use.

Method: Choose how to stretch the data before applying colormap.

Parameters: Depending on Stretch method chosen, different parameters are available:

Low/High Cut: Choose high and low cut values to stretch the remainder through. Use Preset buttons go to default presets.

Stretch Bands Individually: Not used in this tool.

Inverse: Invert the high/low colormap scheme.

Bins: NUmber of histogram bins to use.

References:

1. https://matplotlib.org/3.1.1/gallery/color/colormap_reference.html

11.3.2 Contrast

Description:

Apply a linear percentile, linear absolute, or histogram stretch to the image.

Usage:

Use Contrast Enhancement:

Check to stretch image by below method (Linear Percentage, Linear Absolute, or Histogram).

Colormap: Choose colormap to use.

Method: Choose how to stretch the data before applying colormap.

Parameters: Depending on Stretch method chosen, different parameters are available:

Low/High Cut: Choose high and low cut values to stretch the remainder through. Use Preset buttons go to default presets.

Stretch Bands Individually: Check to stretch each band by statistics of each band, uncheck to stretch all bands by statistics of all.

Inverse: Invert the high/low colormap scheme.

Bins: NUmber of histogram bins to use with the histogram stretch.

11.3.3 Gaussian Blur

Description:

Apply a Gaussian blur transformation to each pixel in the image using a kernel which can be given an independent height and width

Usage:

Use Gaussian Blur: Turns the filter on or off.

Kernal height: Height of the Gaussian blur (range 1-15, default 3), in pixels

Kernal width: Width of the Gaussian blur (range 1-15, default 3), in pixels

Border type: One of the OpenCV border types, defaults to 'BORDER_DEFAULT'

11.3.4 Local Modal Filter

Description:

Assign each pixel to the value most frequently occurring in a local spatial window around the pixel.

Usage:

Use Modal Filter: Turns the filter on or off.

Window Size: Size of the window to evaluate most frequent value around given pixel, in pixels.

Window Shape: Shape of the window.

11.3.5 No Filter

Description:

No filter is used. Depending on the datacube values, the resulting image might appear completely dark, white, or have very little contrast on the screen.

11.3.6 Threshold

Description:

Apply a threshold to the image. Options include a simple binary threshold based on a single number, thresholding below minimum and above maximum values, and thresholding values between the minimum and maximum. Inversion of the results is also an option.

Usage:

Min Threshold:

Values below or equal to this threshold are set to white, unless thresholds are locked. If True Between is checked, values below this threshold are set to black.

Max Threshold:

Values above or equal to this threshold are set to white. If True Between is checked, values above this are set to black.

Lock Min-Max:

Locks Min and Max together. Values above or equal to this value are set to white, values below are set to black.

Inverse:

Inverts black and white.

True Between:

Sets values below Min Threshold to black, values above Max Threshold to black, and values between (including equal to) to white.

11.4 Select Plugins

11.4.1 Create Cube From Selection

Description:

Create a new cube from selection. If the selected area is rectangular, the dimensions of the selected area will be preserved. If the selection is not rectangular, the spatial relationship will be scrambled in order to make a rectangular datacube.

11.4.2 Create Mask Cube From Selection

Description:

Create a single band cube from the selection where selected areas are 1 and unselected areas are 0.

11.4.3 Mean Spectrum

Description:

Create and plot a spectrum from the mean of the selected region. Wavelength metadata must be present for this tool to be available.

The standard deviation will also be plotted, which can be turned on or off by right clicking on the Spectrum object in the Resource Tree and selecting Show/Hide Standard Deviation.

11.4.4 Show Mean of Selection

Description:

Show a dialog box with the mean of each band in the selected region. Useful for datacubes with few bands, bands that don't contain spectral data, and/or datacubes containing disparate data.

11.4.5 Mean Z profile

Description:

Create and plot the mean profile of the selected region in the Z (band) dimension. If wavelength metadata is present this is equivalent to the Mean Spectrum tool.

The standard deviation will also be plotted, which can be turned on or off by right clicking on the Spectrum object in the Resource Tree and selecting Show/Hide Standard Deviation.

11.4.6 Mean First Derivative

Description:

Plot the mean first derivative (discrete difference) of the selected area.

11.4.7 Spectral Correlation Matrix

Description:

Creates a heatmap showing band-to-band correlations using the Pearson coefficient. The diagonal is always 1 (perfect correlation as bands are identical), while uncorrelated bands have values of zero. This heatmap may be useful for exploring data qualitatively.

Usage:

Save Correlation to Disk:

Check to save.

Colormap:

Colormap to use.

References:

1. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

11.4.8 Create Correlation Coefficients

Description:

A correlation coefficient aligns reflectance data created with downwelling irradiance data to a ground truth target such as a tarp or reflectance standard. Once created, these coefficients are used in the Reflectance Conversion with Downwelling Irradiance plugins to improve the accuracy of the result.

Usage:

Start by creating a region of interest of the ground truth target.

Use Measured Reflectance:

Check to use a previously measured reflectance spectra for the reference object the region of interest contains. Uncheck to assume a flat reflectivity.

Measured Reflectance File:

Select the previously measured reflectance spectrum. The first column should be wavelength in nanometers, the second column is reflectance from 0-1 or 0-100%. File should be tab, comma, or space delimited.

Flat Reflectivity:

Select flat reflectivity of ground truth target if measured spectrum is not available or desired.

Measured Reflectivity in Percentage:

Check if measured reflectance spectrum is on the scale 0-100. Uncheck if 0-1.

Outputs:

- Spectrum of correlation coefficients.

11.4.9 Find Closest RAL Classic Color

Description:

Uses spectral information to determine the closest RAL Classic color to selected region. Wavelength and reflectance scale factor metadata must be present. Useful for color matching.

References:

1. https://en.wikipedia.org/wiki/RAL_colour_standard

11.4.10 Median Spectrum

Description:

Plot the median spectrum within selected area.

11.4.11 Mean Nth Derivative

Description:

Compute and plot the mean Nth derivative (discrete difference) of the selected region.

11.4.12 Save random spectra to text file

Description:

Randomly selects pixels within the region of interest and exports their spectra (or bands) to a text file.

Usage:

Number of spectra to export:

Number of randomly selected spectra to export.

Add Wavelengths Col:

Option to add the wavelength metadata as the first column of exported data.

11.4.13 Send To Clipboard

Copy Mean as Text**Description:**

Copies mean spectrum as text to clipboard for pasting into other applications.

Copy All as Transposed Text

Description:

Copies mean spectrum as transposed text to clipboard for pasting into other applications.

All Spectra as Text

Description:

Copies all spectra in region as text to clipboard for pasting into other applications.

Copy as Image

Description:

Copies image of region to clipboard for pasting into other applications.

GLOSSARY OF HYPERSPECTRAL IMAGING TERMINOLOGY

Average RMS Spot Radius:

Measure of the imager resolution, averaged across its spectral range.

Binning:

Process by which multiple pixels are lumped together in a single spatial or spectral channel to increase signal, and signal-to-noise ratio. The tradeoff is a loss in spatial or spectral resolution.

Bit Depth:

The resolution of the data recorded for each pixel. For example, a Bit Depth of 12 means that each signal acquisition is stored as one of $2^{12} = 4096$ discrete values.

Datacube:

The complete data associated with a hyperspectral scan including the light intensity at each wavelength, at each pixel in a scan. Similar to RGB images, datacubes have two spatial dimensions and one spectral dimension. However, the spectral dimension contains light intensity for each of hundreds of color channels (as opposed to only three colors in RGB images). Datacubes are accompanied by metadata that store additional information about the datacube (e.g., the size of each dimension and the spectral wavelengths).

Dispersion per Pixel:

Range of wavelengths captured by each pixel.

f/#:

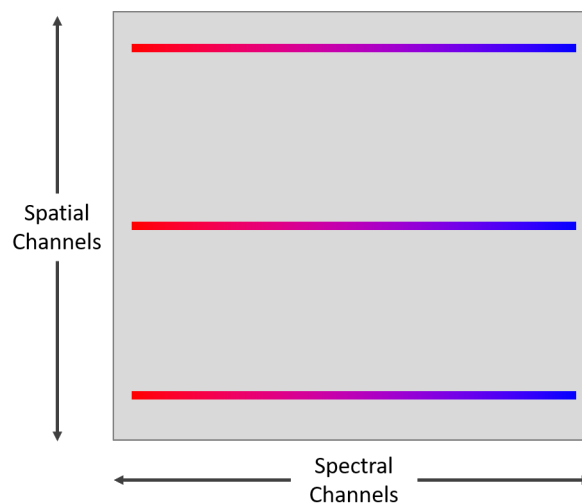
f/# is a measure of the lens speed, and is a quantity that is needed to determine the optical system's radiometric performance. f/# for a hyperspectral imager means the same as it does for a conventional camera.

Image:

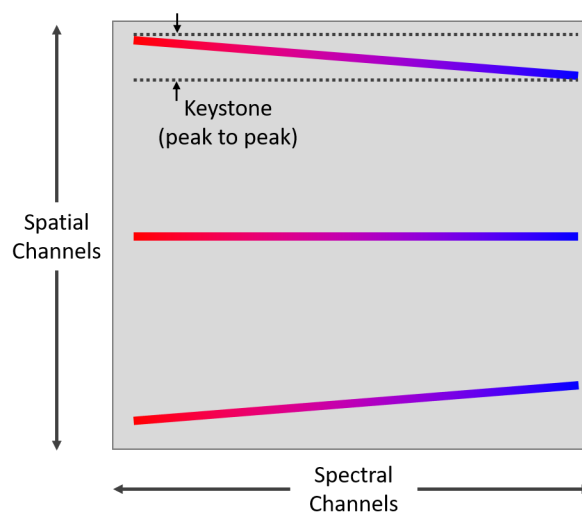
Typically, an RGB rendering of a datacube using three distinct spectral channels (among hundreds) to create a visually meaningful image for the user. An image can also be a gray-scale rendering of a datacube utilizing a single wavelength. Finally, the term image can apply to any pixel map created from the datacube, or the result of an analysis of that datacube, such as a heat map or a color-coded pixel classification.

Keystone Distortion (peak-to-peak):

Keystone is an optical distortion associated with hyperspectral imagers that manifests when light from a single spatial position falls across different spatial rows of the focal plane array for different spectral channels. The figures show how a "perfect" hyperspectral imager would map the signal from three spatial channels, one at the top of the field of view, one in the middle, and one at the bottom, along with an exaggeration of the signal that a "real-life" imager maps the same signal. The peak-to-peak Keystone is the distance between the dashed lines, measured in pixels. All real hyperspectral imagers have some Keystone, which causes the signal to be "mixed" between spatial channels.



"PERFECT" IMAGER



"REAL-LIFE" IMAGER (exaggerated)

Max. Frame Rate:

The maximum line-scan rate of the hyperspectral imager, assuming adequate lighting. It is the rate at which the imager can acquire lines, (not complete scans). The total scan time depends on the number of lines recorded in a complete two-dimensional image.

Peak SNR:

Peak Signal to Noise Ratio.

Pixel Size:

The center-to-center physical distance between pixels on the sensor array.

Pixel Well Depth:

The maximum number of electrons a pixel can store before saturating.

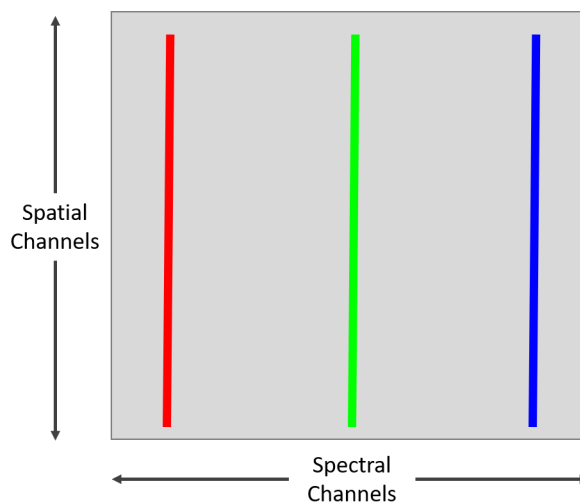
Slit Width:

The width of the slit through which light passes as it enters the spectrometer.

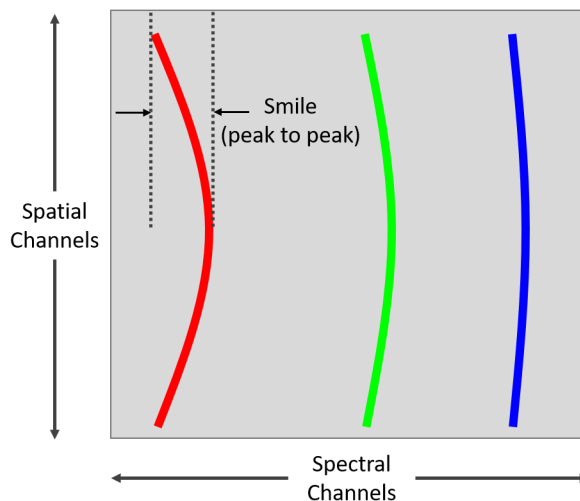
Smile Distortion (peak-to-peak):

Smile is an optical distortion associated with hyperspectral imagers that manifests when light from a single wave-

length falls across different spectral columns of the focal plane array for different spatial channels. The figures show the signal that a “perfect” hyperspectral imager would obtain for a uniformly lit sample with narrow-band red, green, and blue signal, along with an exaggeration of the signal that a “real-life” hyperspectral imager obtains. Peak-to-peak Smile for the red signal is the distance between the black vertical lines. All real hyperspectral imagers have some Smile, usually measured in pixels. Smile leads to spectral signatures that change slightly across different spatial channels.



“PERFECT” IMAGER



“REAL-LIFE” IMAGER (exaggerated)

Spatial Pixels:

Resonon’s hyperspectral imagers are line-scan or push-broom imagers. The Spatial Pixels are the number of pixels along this line.

Spectral Bandwidth:

The width of each spectral data point. Equal to the spectral range divided by the number of spectral channels.

Spectral Channels:

The number of wavelength bands that the hyperspectral imager measures across the Spectral Range.

Spectral Pixels:

The number of sensor pixels across which the spectral range is recorded. This is not equal to the number of spectral channels, as several pixels may be used together to record the light intensity at a single spectral channel.

Spectral Range:

The range of electromagnetic wavelengths (e.g. light) over which the hyperspectral imager collects signal. For reference, visible wavelengths span from approximately 400 to 700 nanometers (nm).

Spectral Resolution (FWHM):

The full-width/half-max of the spectral intensity distribution of a single wavelength input. Spectral Resolution (FWHM) is a measure of the spectral focusing ability of the instrument.

Spectral Sampling:

The spectral range divided by the (number of) spectral pixels. It is often narrower than the spectral resolution.

Spectrometer Magnification:

Ratio between the slit image width on the sensor array and the physical slit width.

Spectrum:

A Two-dimensional graphical representation of the light intensity versus wavelength recorded for a single pixel.

RECALIBRATION SERVICES

Resonon recommends a yearly schedule for wavelength and radiometric calibration for best performance, or immediately after rough handling. Please contact us below for a quote and instructions.

PRODUCT SUPPORT

Please contact Resonon by phone, email, or via the form on our website:

Phone: 1.406.586.3356

Website: <https://resonon.com/support>

Email: support@resonon.com

COPYRIGHT NOTICE

Copyright © 2016 - 2023 Resonon Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means (electronic or mechanical, including photocopying) for any purpose without written permission from Resonon. Resonon will not be responsible or liable for any accidental or inevitable damage that may result from unauthorized access or modifications.

This document may contain errors or inaccuracies, and it may be revised without advance notice. This manual is updated frequently.

Please see our software copyright and EULA at
<https://resonon.com/content/files/Spectronon-End-User-License-Agreement.pdf>

Resonon welcomes any recommendations regarding this manual. Customer feedback is always welcome as it helps us to continuously improve upon the quality of our products.

Please contact us at Resonon with any questions or comments. We look forward to hearing from you.