# pdsspect Documentation

***Release 0.1.0***

**PlanetaryPy**

**Jul 27, 2017**

# Contents:

CHAPTER 1

# pdsspect - A Python PDS Image Region of Interest Selection Tool

**NOTE:** This is Alpha quality software that is being actively developed, use at your own risk. This software is not produced by NASA.

- Free software: BSD license
- Documentation: https://pdsspect.readthedocs.org.

## 1.1 Features

- NASA PDS Image Viewer

**NOTE:** This is alpha quality software. It lacks many features and lacks support for many PDS image types. This software is not produced by NASA.

## 1.2 Install

On OS X you must first install the Qt UI toolkit using Homebrew (http://brew.sh/). After installing Homebrew, issue the following command:

```
brew install qt
```

### 1.2.1 Install Using Pip

Install pdsspect using pip:

```
pip install pdsspect
```

Then install your choice of pyside, pyqt4, or pyqt5

### 1.2.2 Install for Development

Create a new virtual environment, install the *pdsspect* module with git, and setup the PySide environment. You must install either PySide, PyQt5, or PyQt4 as well (recommend PyQt5):

```
Make a clone of ``pdsspect`` and change to main directory. We recommend
making a virtual environment for to install ``pdsspect`` in.

pip install -e .
pip install PyQt5
```

Now you should be able to run the *pdsspect* program.

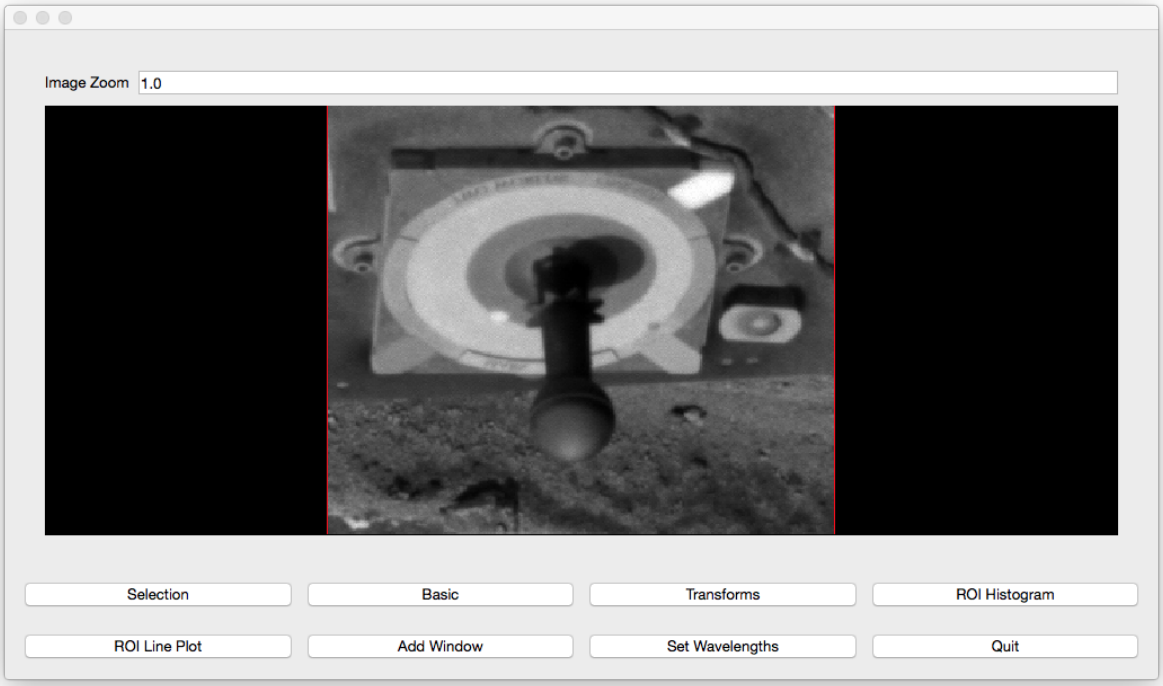
This works on Linux as well (Ubuntu 14.04).

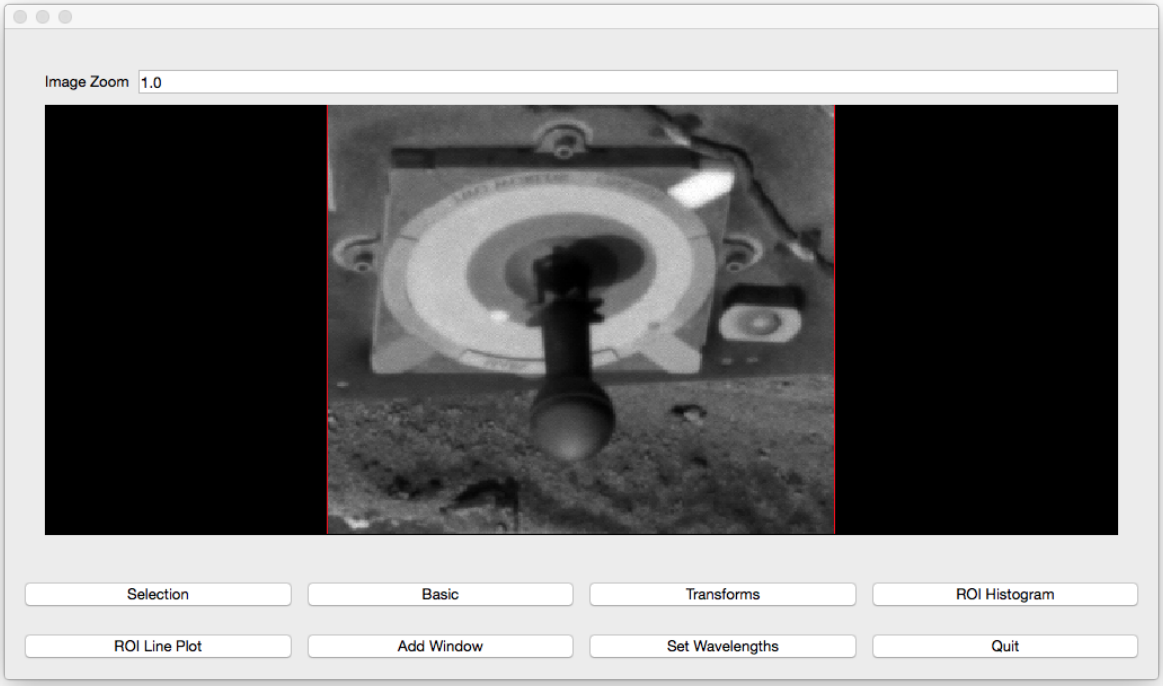
## 1.3 Quick Tutorial

Open an image in the command line:

```
pdsspect tests/mission_data/2m132591087cfd1800p2977m2f1.img
```

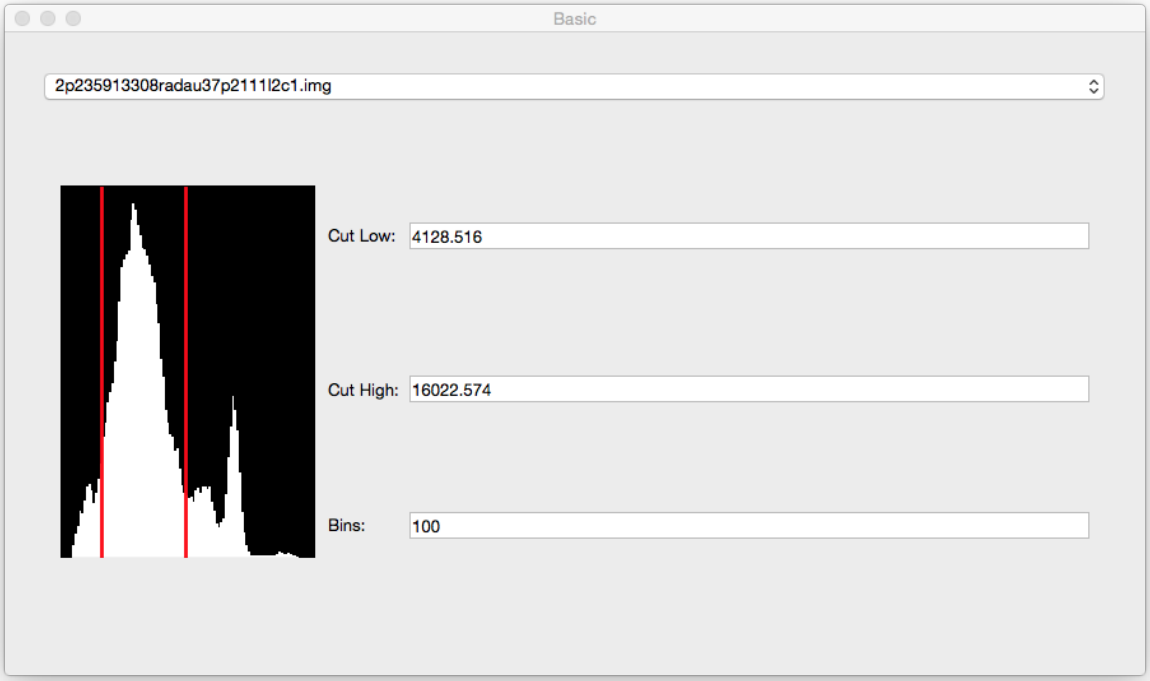
This will open the default window:





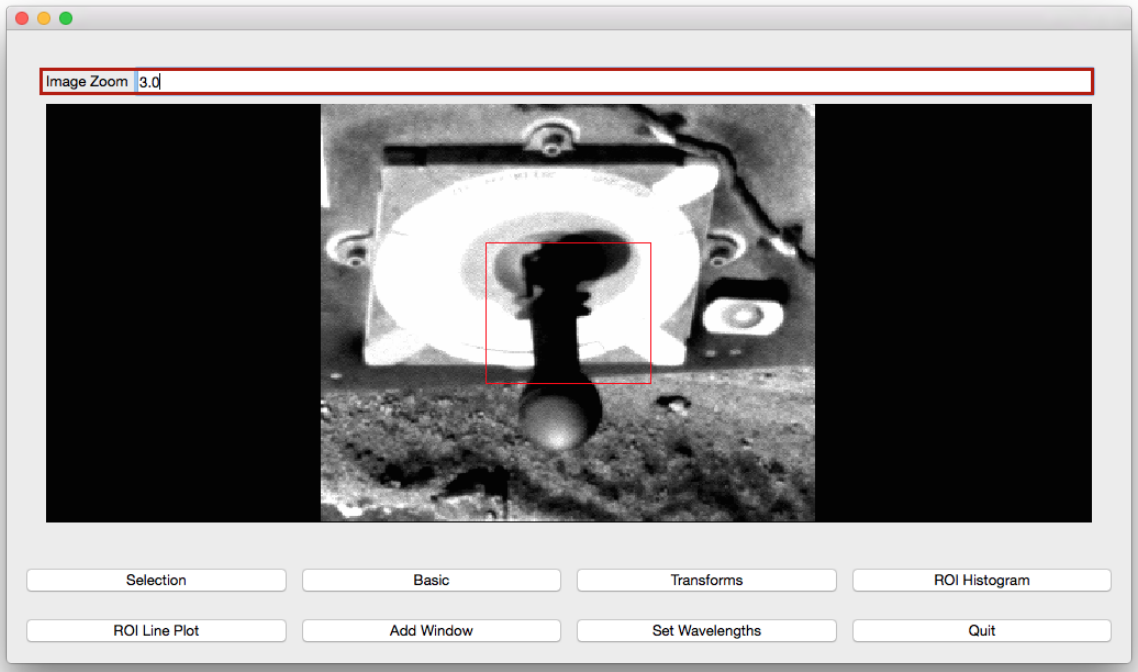


The bottom left window is considered the main window. In this window, the user can adjust the position of the pan and open other windows. The bottom right window is the `basic` window. Pressing the `basic` button will open this window if closed. However, it starts out open. In this window, the user can change the image in the views and adjust the cut levels by either moving the red lines or typing in the numbers in the cut boxes:


The top window is the `pan window` which displays the data in the main window's red box. The main function of this window is to make Region of Interest (ROI) selections.

In the `zoom` box in the main window, the user can change the size of the box and the data in the pan view:

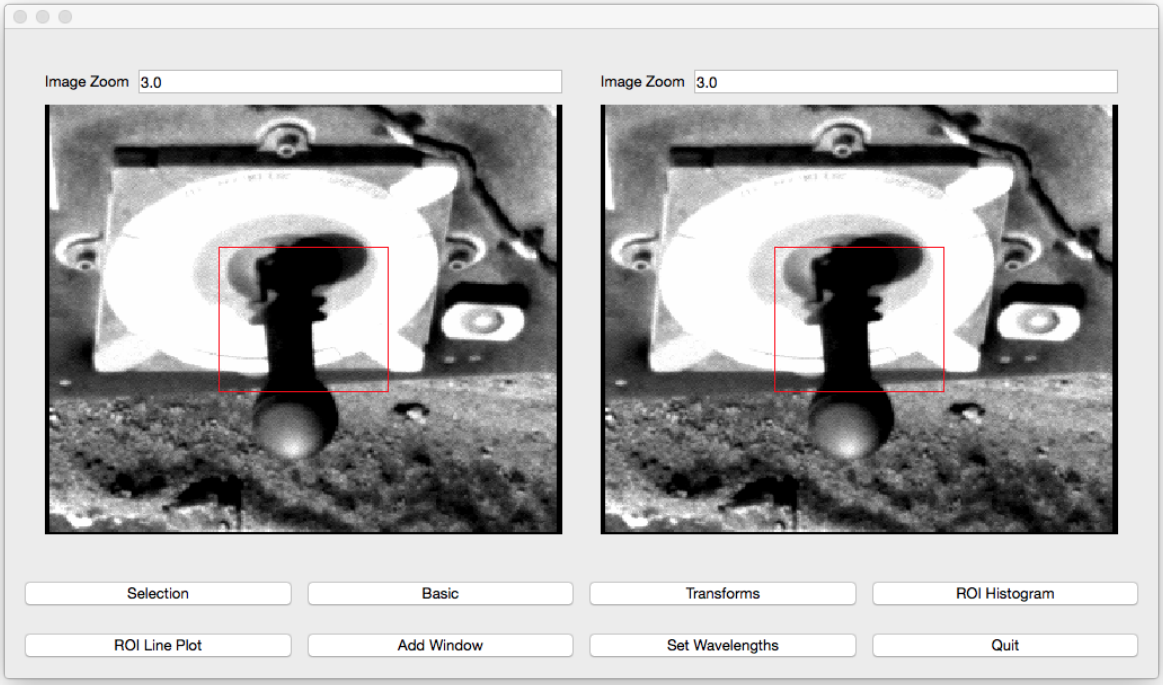




the mouse wheel can also be used to change the zoom. Rolling the wheel foward and backwards will adjust the zoom amount by `+1` or `-1` respectively. The user can adjust the position of the box by clicking in the main window where the center of the pan should be. Using the arrow keys will also adjust the position of the box by `1` in the direction of

the arrow key.

Clicking the `Add Window` button will open another view. This view will have the same image, cut levels, and zoom by default.




If the image's are the same, chaning the cut levels on one image will automatically change the cut levels on another image. However, one can change the zoom on one view without changing the zoom another view.

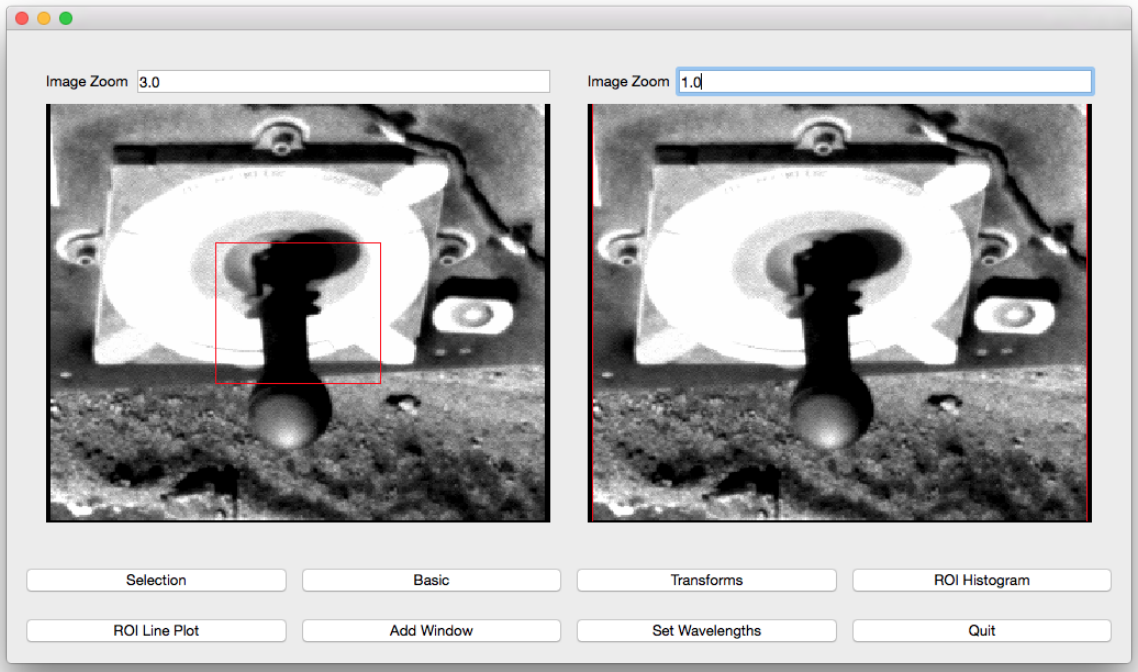
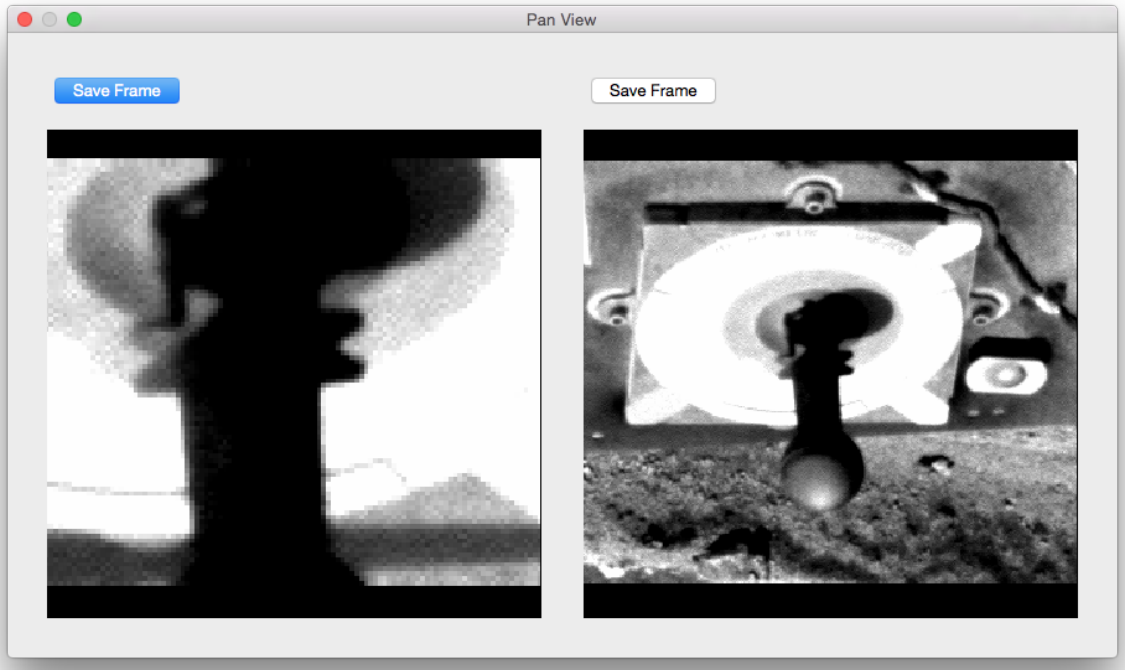
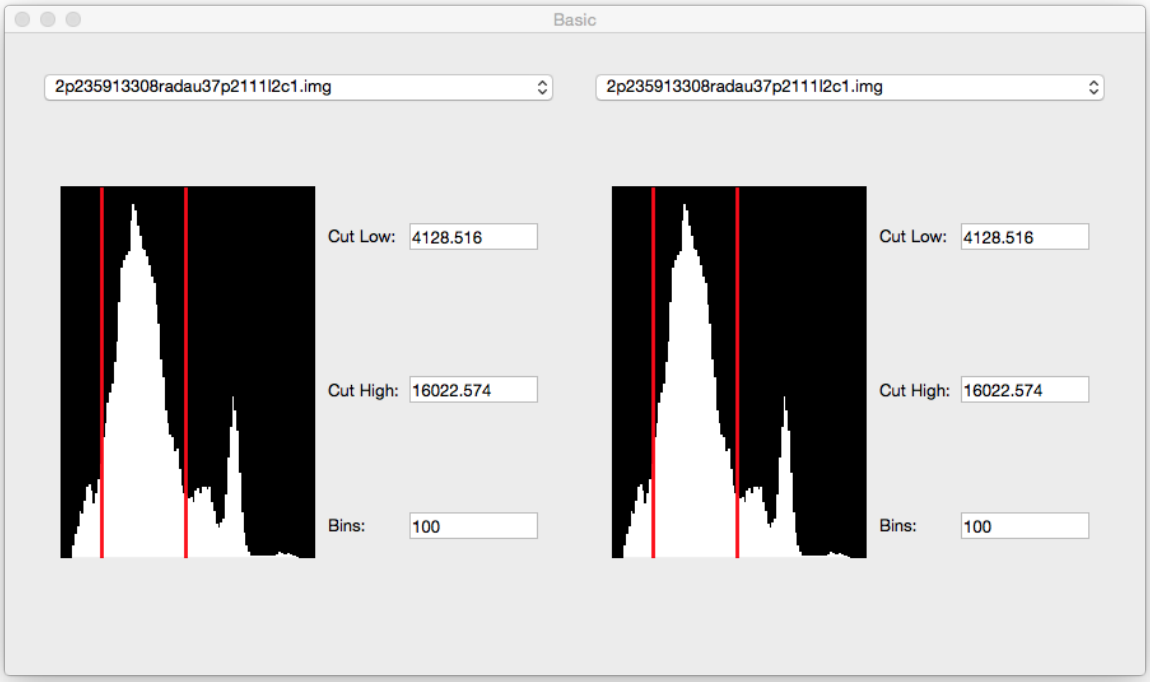
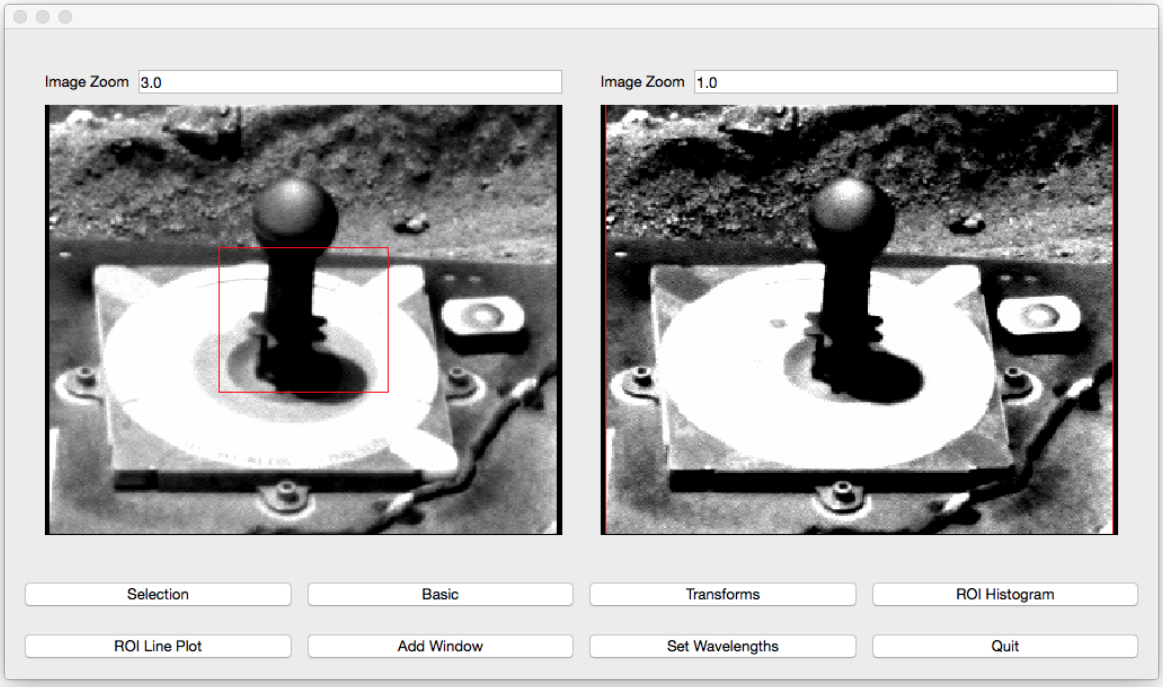


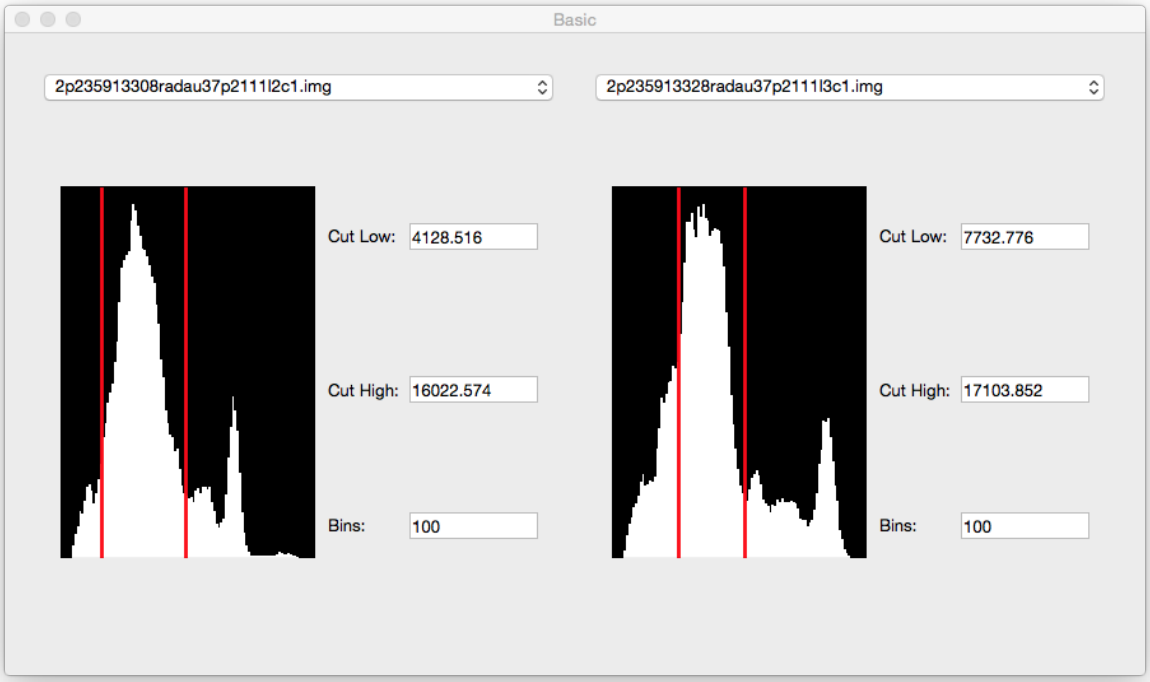
Image Zoom 3.0
Image Zoom 1.0
Selection
Basic
Transforms
ROI Histogram
ROI Line Plot
Add Window
Set Wavelengths
Quit


Pan View
Save Frame
Save Frame

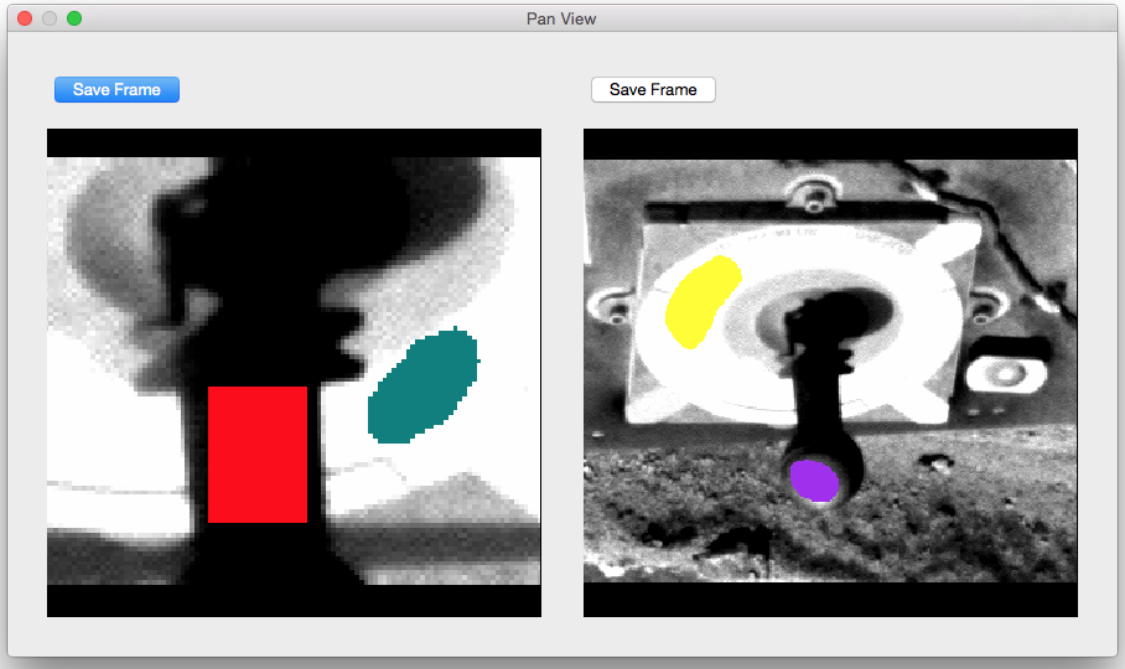



When the images are different, adjusting the cut levels on one image will only change the cut levels on that image:

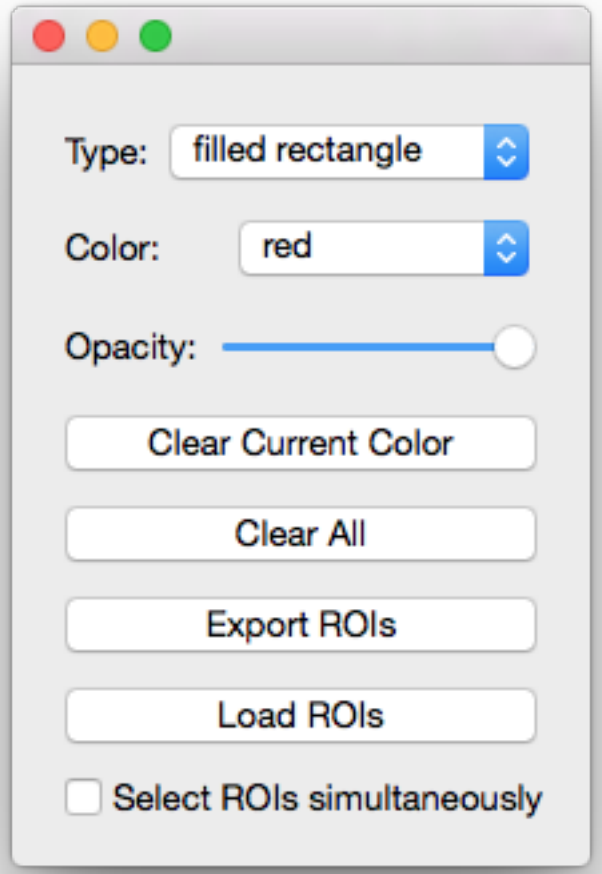





The user can create separate ROIs in each view:




Clicking the `Selection` button will open the `Selections Window`:




In this window, the user can choose the color of the ROI. The possible choices for colors: `red`, `brown`, `lightblue`, `lightcyan`, `darkgreen`, `yellow`, `pink`, `teal`, `goldenrod`, `sienna`, `darkblue`, `crimson`, `maroon`, `purple`, and `eraser (black)`. The selection type can be changed in this window as well. The possible types are `filled rectangle`, `filled polygon`, and `pencil` (single points).

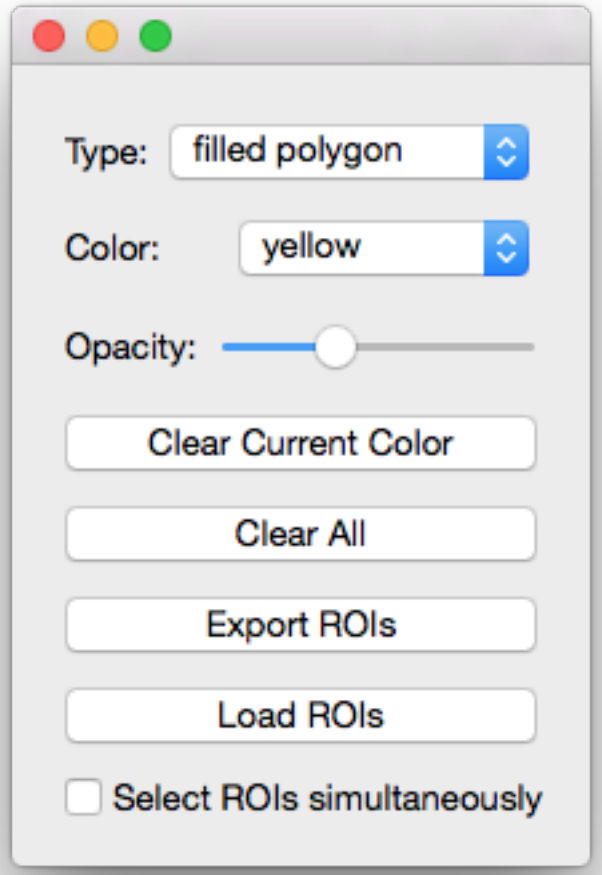
Furthermore, in this window, the user can clear the current color or clear all ROIs. Most importantly, the user can export ROIs to `.npz` files. These files contain boolean masks and of the images and a list of files open at the time of export. The ROIs in the 2nd, 3rd, 4th, etc. views will be labeled as `color#view` while the ROIs in the first view is still labeled as `color`. For example, to see the data in an example file `example.npz`, use numpy load method to view and utilize data.

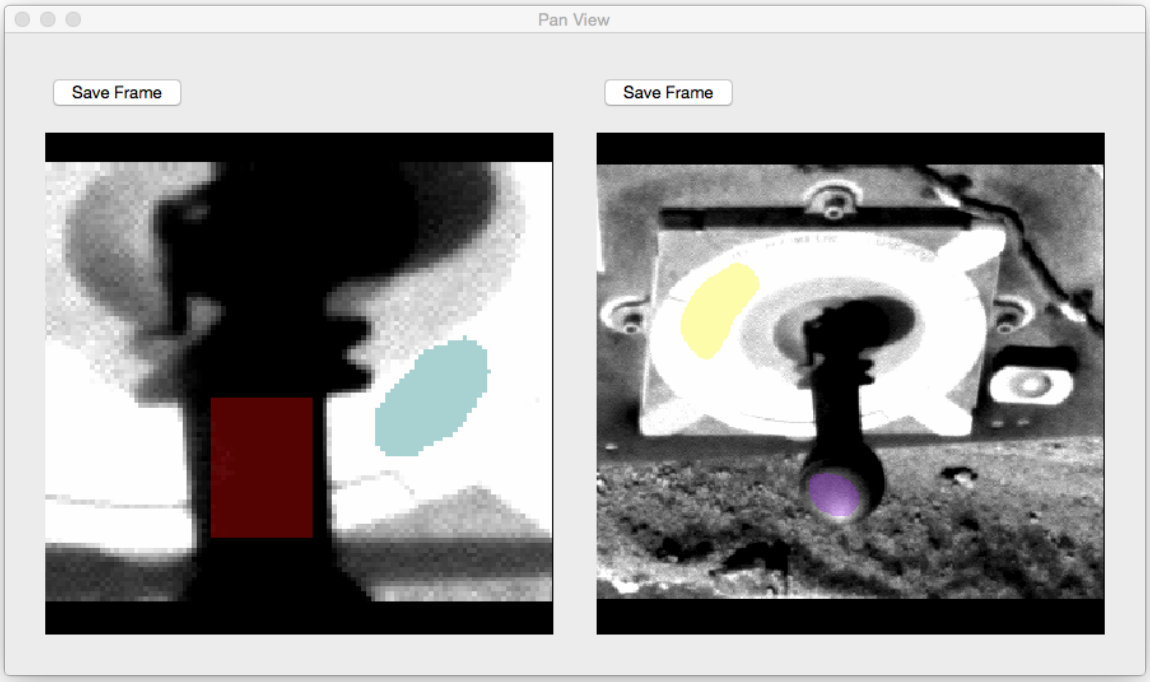
```
>>> import numpy as np
>>> selections = np.load('example.npz')
>>> selections['red'][114:118, 142:146]
array([[ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]], dtype=bool)
```

```
>>> selections['purple2'][48:52, 146:150]
array([[False, False, False, False],
       [False,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]], dtype=bool)
```

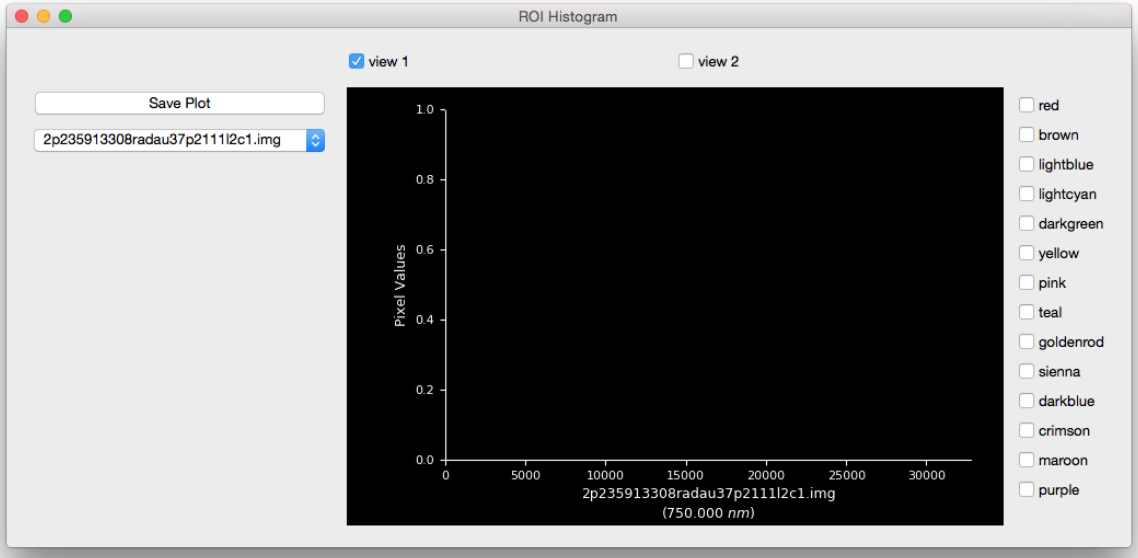
The user can also import ROI selections. However the images that are open must be in the `files` list in the `.npz` file.

Changing the opacity in the Selecitons window will change the opacity on all the ROIs in every view:

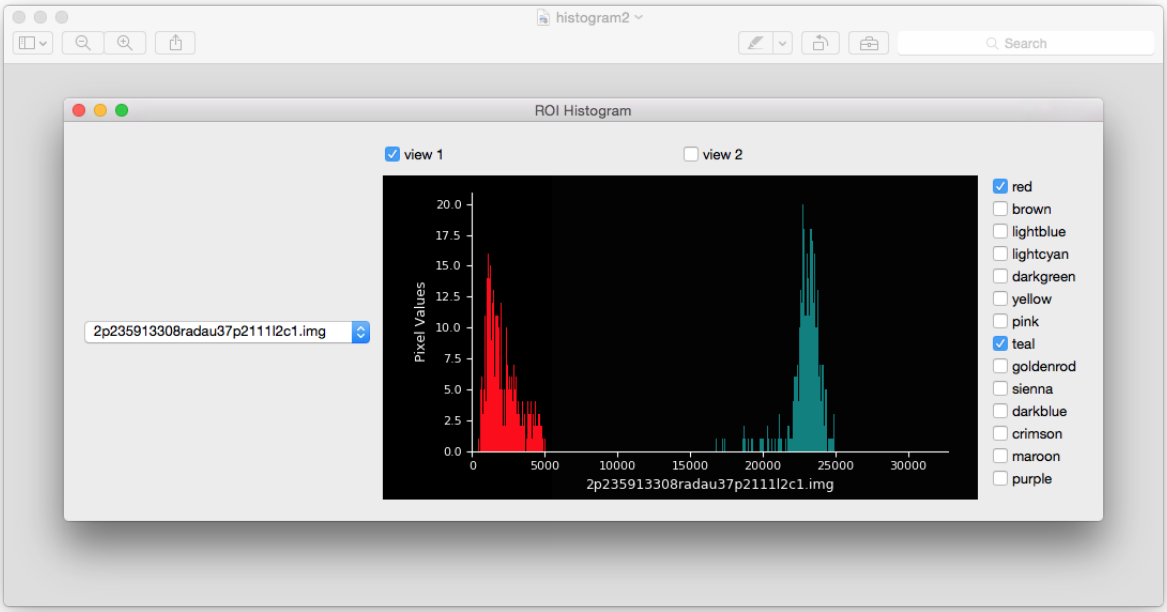




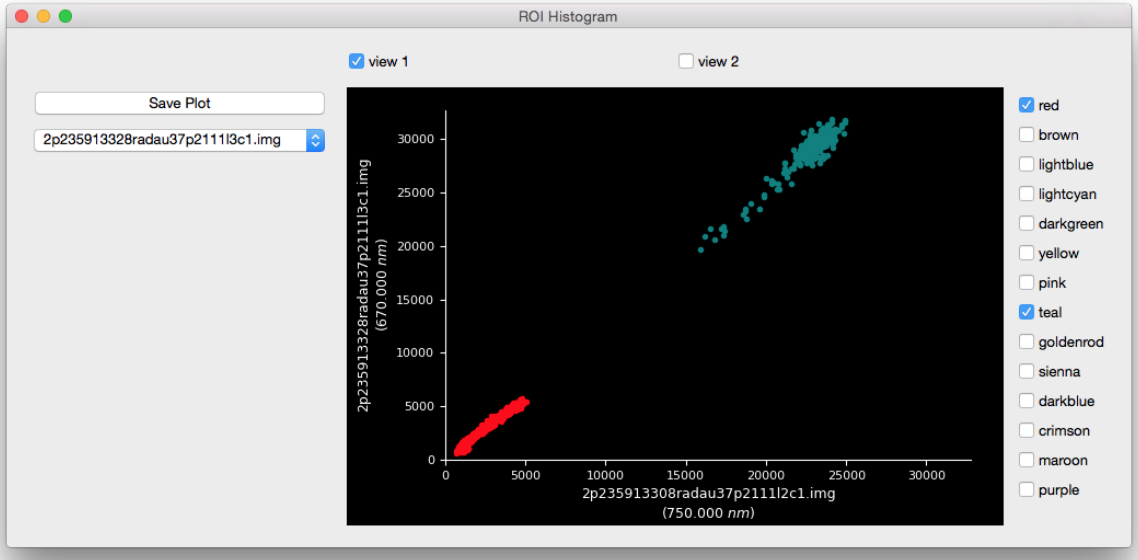

You can view the data within the ROIs with the `ROI Histogram` window. Open the window by pressing the `ROI Histogram` button in the main viewer.



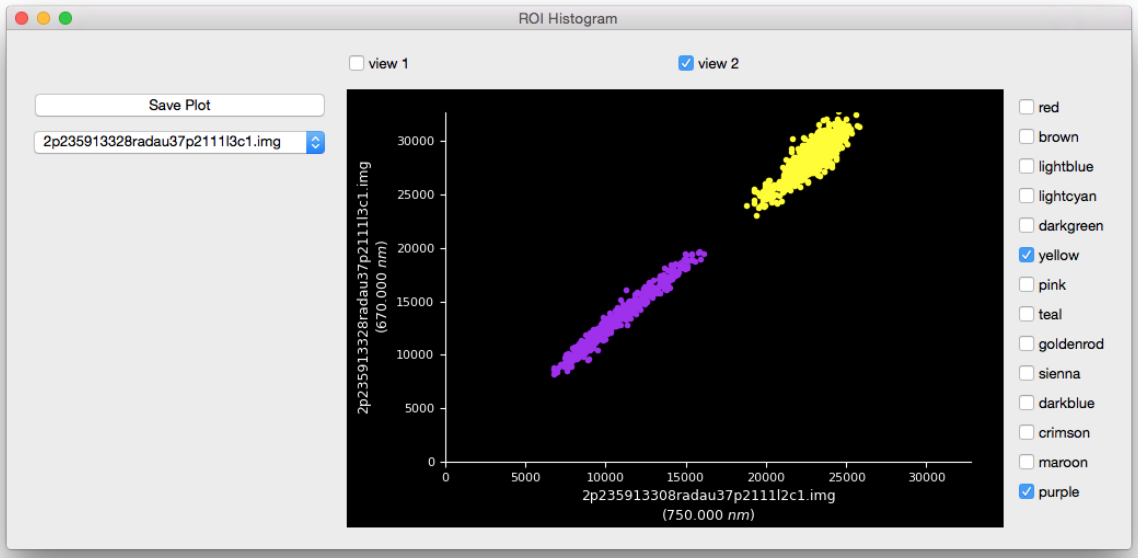

Display the data in the ROI by color by checking the checkbox next to the color. When the image in the menu and the current image in the checked view are the same, the plot will be a histogram:




When the menu and the current image are different, the plot will compare the data:

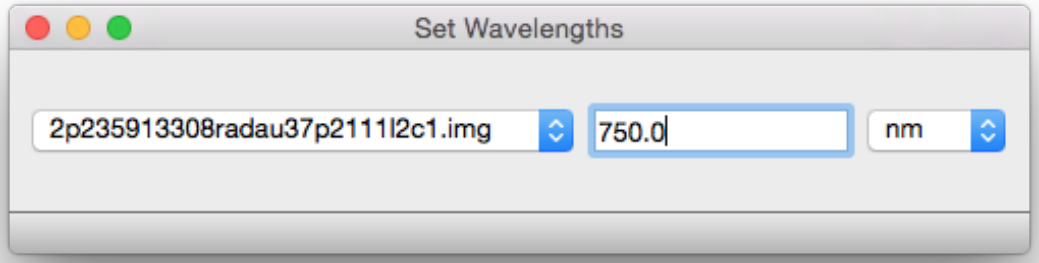



To view the data in the other view, check the view number:

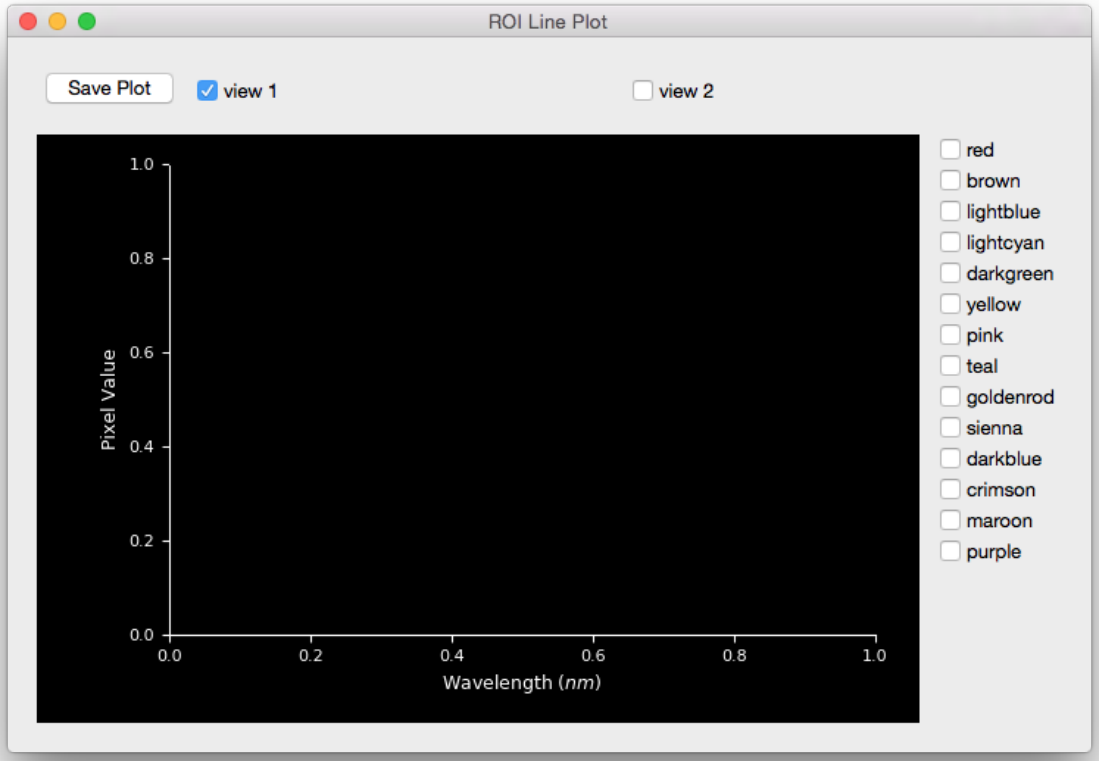


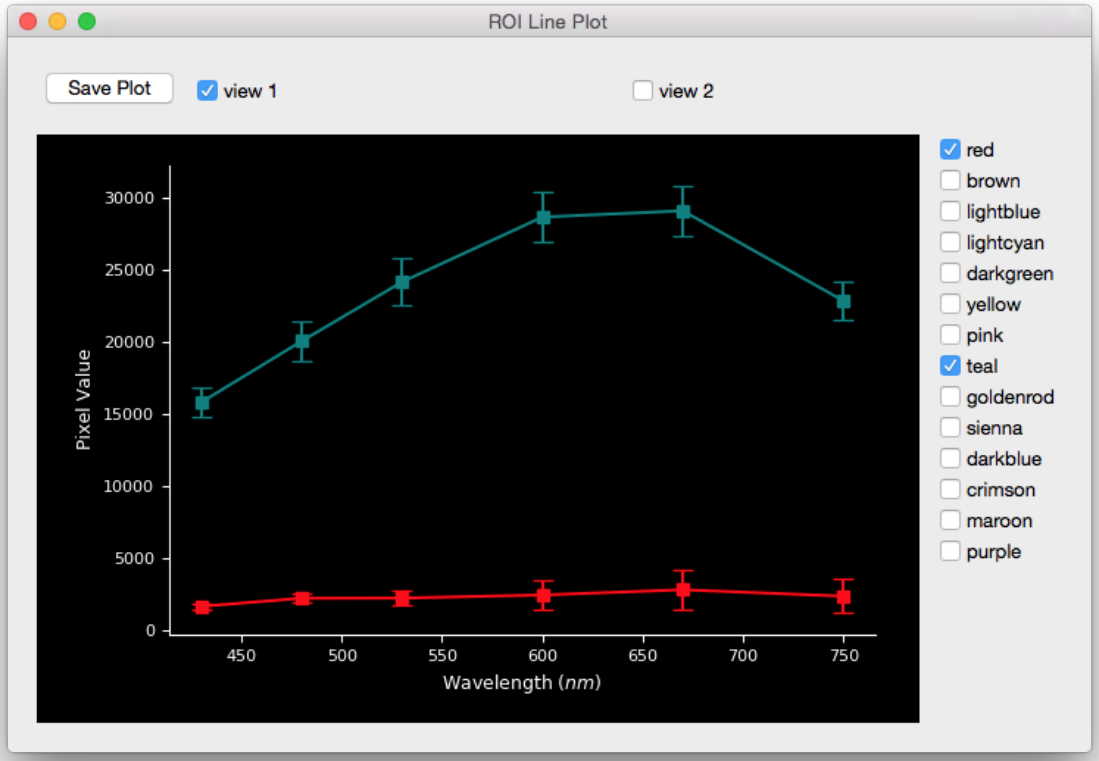

Overlay ROIs by checking other boxes. The order (depth) of the histogram data will be in the order that the user checks the boxes (i.e., checking red and then purple will result in purple overlaying the red).

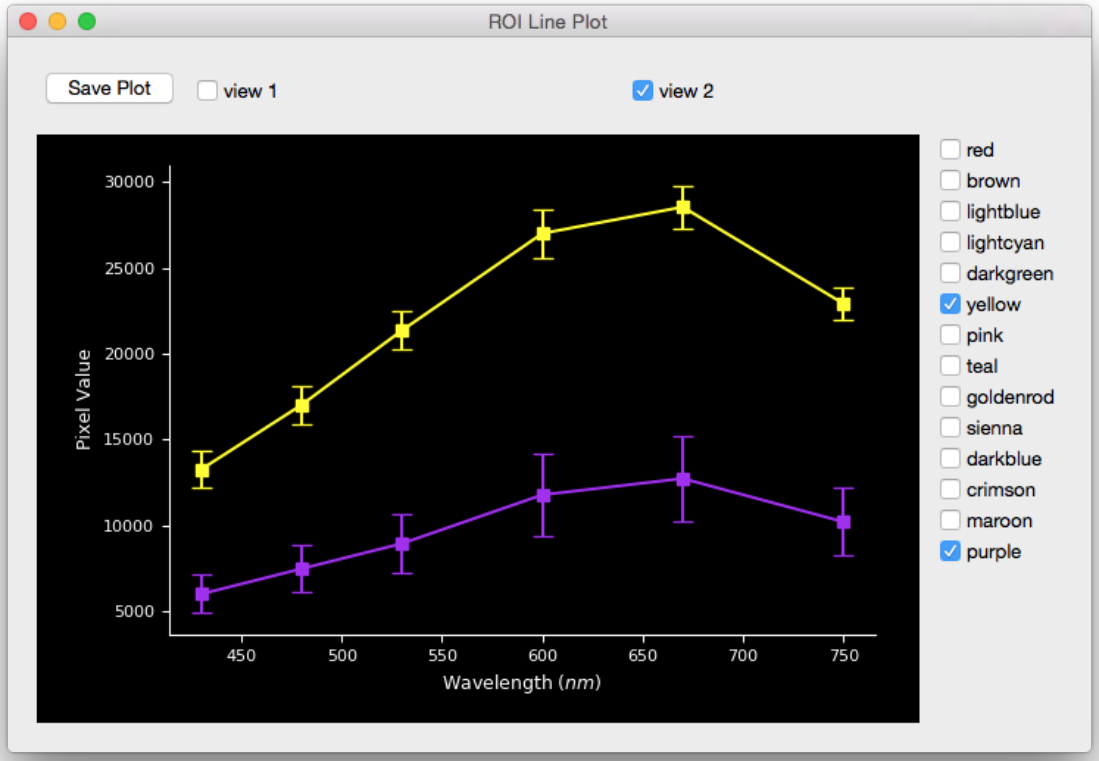
To perform multispectral analysis use `ROI Line Plot`. If analyzing images that are not *fully supported (see here for list of instruments supported by pdsspect)* the user must manually input the image wavelength with `Set Wavelength` widget:



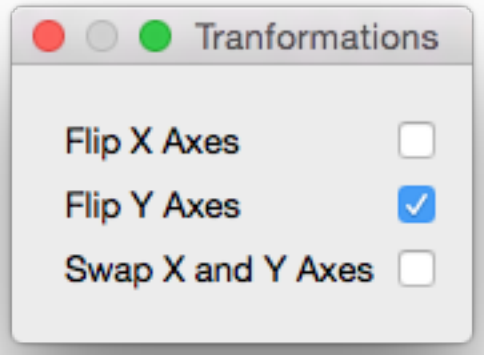

`ROI Line Plot` works similar to that of the histogram plot except it will compare each image with an associated wavelength.

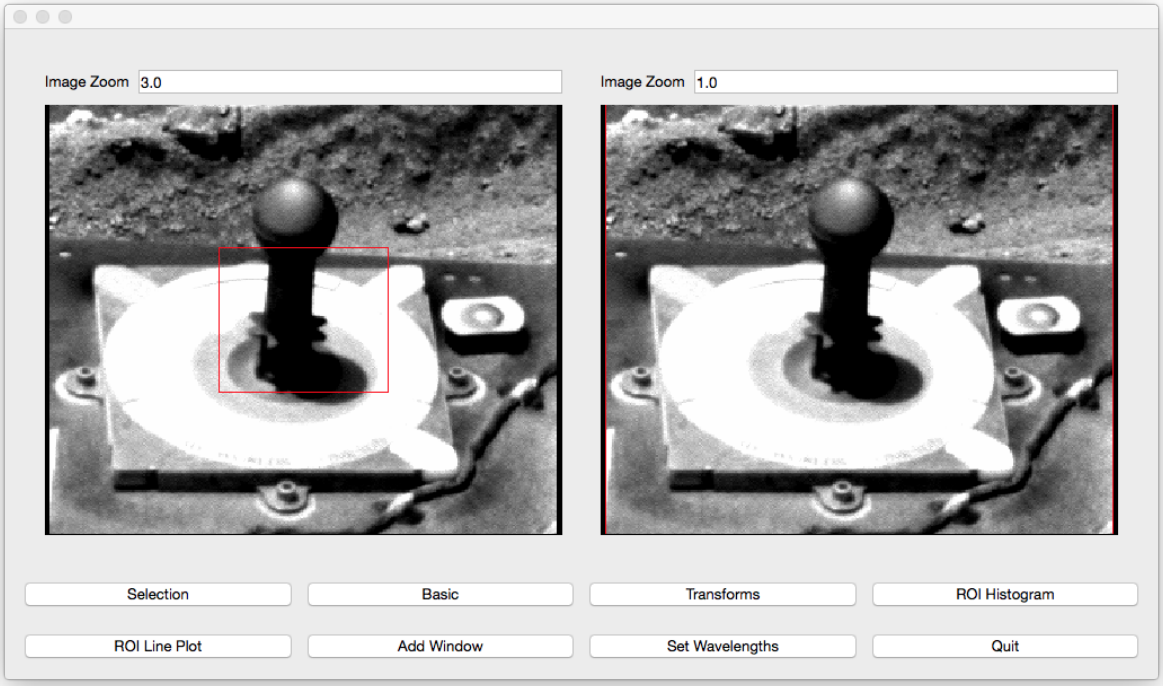

ROI Line Plot
Save Plot
view 1
view 2
red
brown
lightblue
lightcyan
darkgreen
yellow
pink
teal
goldenrod
sienna
darkblue
crimson
maroon
purple
Pixel Value
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Wavelength (nm)

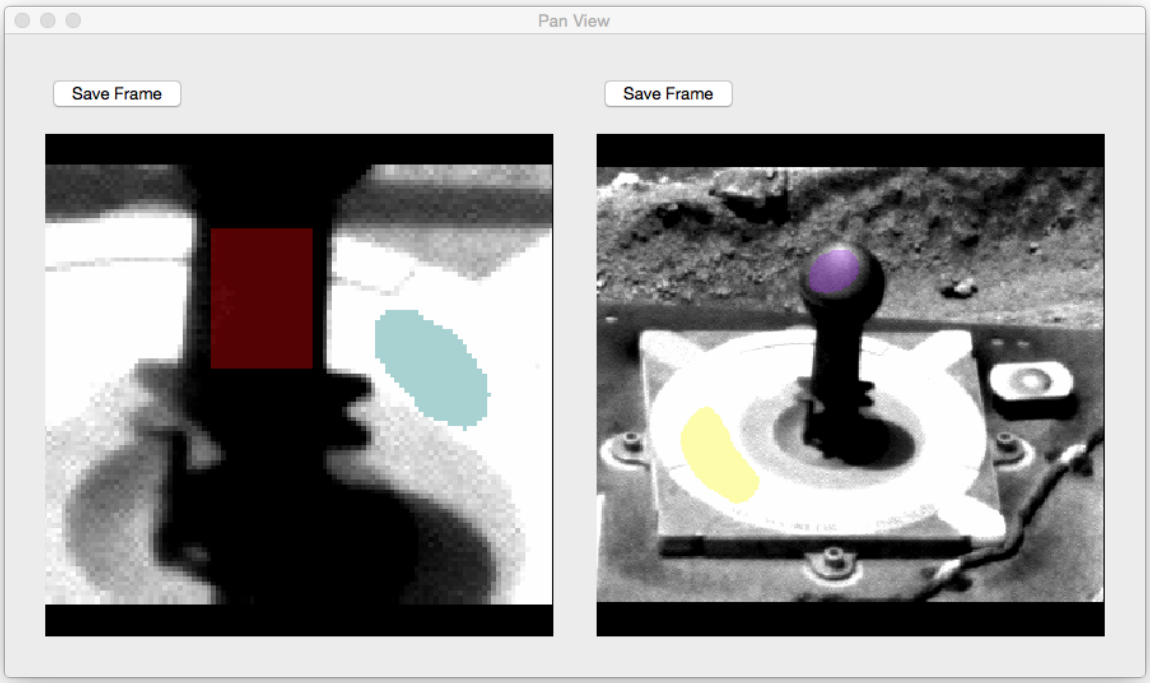

ROI Line Plot
Save Plot
view 1
view 2
red
brown
lightblue
lightcyan
darkgreen
yellow
pink
teal
goldenrod
sienna
darkblue
crimson
maroon
purple
Pixel Value
30000
25000
20000
15000
10000
5000
0
450
500
550
600
650
700
750
Wavelength (nm)




The user can flip the image over different axis with the Transforms window. The transformation will apply to each image in all the views:






Note that when opening multiple images at the same time, it is best practice that they are all the same shape. Otherwise the images will have the smallest common shape and not look as expected (i.e., If when loading two images where one image has a shape of `(63, 36)` and the other image has a shape of `(24, 42)`, the displayed shape will be `(24, 36)`. This will cause the first image to have the right side cut off and the second image to have the top cut off). This

is done so all ROIs created can apply to the entire list of images. To avoid this behavior, either only open images that have the same shape or open images one at a time.

### 1.3.1 Images In Example

- 2p235913308radau37p2111l2c1.img
- 2p235913328radau37p2111l3c1.img
- 2p235913348radau37p2111l4c1.img
- 2p235913368radau37p2111l5c1.img
- 2p235913399radau37p2111l6c1.img
- 2p235913431radau37p2111l7c1.img

## 1.4 Supported Instruments

- **MER**
  - Pancam
- **MSL**
  - Mastcam
- **Cassini**
  - Imaging Science Subsystem (ISS)

### 1.4.1 Adding More Instruments

We welcome anyone to create more models for instruments that are not yet supported. Please follow the Pull Request guide to make sure your model is compatible with the rest of the models. See Pull Request #20 as an example.

#### Pull Request Checklist

Please include the following checklist in your PR so we know you have completed each step:

```
- [ ] Created model as subclass of [InstrumentBase](https://github.com/planetarypy/
↪pdsspect/blob/master/instrument_models/instrument.py#L7)
- [ ] Added model to [get_wavelength](https://github.com/planetarypy/pdsspect/blob/
↪master/instrument_models/get_wavelength.py)
- [ ] Documented Model
- [ ] Tested Model
- [ ] Added model to [test_get_wavelength](https://github.com/planetarypy/pdsspect/
↪blob/master/tests/test_get_wavelength.py) test
- [ ] Added instrument to supported_instruments.rst list
```

#### Style

- Set PR label to `Instrument Model`

- If an issue was created, please include `Fixes #<issue_number>` at the top of the PR to automatically close the issue
- Please include a link to any documents used to find the filter wavelength. Follow the example for Mastcam and/or Pancam
- When documenting your model, use numpy docs.See these examples. Also add to instrument_models.rst following the format of the other models
- For tests, if one of the core mission_data images is not from your instrument, create a minimal label in the tests__init__.py. You must test the model itself and test that it works in test_get_wavelength
- Add the mission and instrument to the `supported_instruments.rst` file following the set format.

# CHAPTER 2

## pdsspect

**class** pdsspect.pdsspect.**PDSSpect**(*image_set*)

Bases: PyQt5.QtWidgets.QMainWindow, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

Main Window of pdsspect

**Parameters** **image_set** (*PDSSpectImageSet*) – pdsspect model

**image_set**
*PDSSpectImageSet* – The model for each view

**pdsspect_view**
*PDSSpectViewWidget* – The main viewer for panning

**pan_view**
*PanView* – The view in which the user makes ROI selections

**selection_btn**
QtWidgets.QPushButton – Button to open the selections window

**selection_window**
*Selection* – The selection window to adjust ROI, import ROIs, and export ROIs

**basic_btn**
QtWidgets.QPushButton – Button to open the basic window

**basic_window**
*BasicWidget* – Window to adjust cut levels and change images

**transforms_btn**
QtWidgets.QPushButton – Open Transforms window

**transforms_window**
*Transforms* – Window to flip x axis, flip y axis, or switch x and y axis

**roi_histogram_btn**
QPushButton – Open ROI Histogram window

**roi_histogram_window**
*ROIHistogramWidget* – The ROI Histogram Window

**roi_line_plot_btn**
QPushButton – Open ROI Line Plot window

**roi_line_plot_window**
*ROILinePlotWidget* – The ROI Line Plot Window

**add_window_btn**
QPushButton – Add another window

**quit_btn**
QtWidgets.QPushButton – Quit

**button_layout1**
QtWidgets.QHBoxLayout – Layout for the buttons. If you want to re-adjust where the buttons go, override this attribute

**button_layout2**
QtWidgets.QHBoxLayout – Layout for the buttons. If you want to re-adjust where the buttons go, override this attribute

**main_layout**
QtWidgets.QVBoxLayout – Place the image viewer over the buttons. Overide this attribute if changing overall layout

**add_window**()
Add another window to make more ROIs

**image_sets**
list – All the image sets, including the current one

**open_basic**()
Open the Basic Window

**open_roi_histogram**()
Open the ROI Histogram Window

**open_roi_line_plot**()
Open the ROI Line Plot Window

**open_selection**()
Open the Selection Window

**open_set_wavelengths**()
Open Set Wavelengths window

**open_transforms**()
Open the Transforms Window

**quit**(**args*)
Quit pdsspect

pdsspect.pdsspect.**pdsspect**(*inlist=None*)
Run pdsspect from python shell or command line with arguments

**Parameters** **inlist** (list) – A list of file names/paths to display in the pdsspect

### Examples

From the command line:

To view all images from current directory

pdsspect

To view all images in a different directory

pdsspect path/to/different/directory/

This is the same as:

pdsspect path/to/different/directory/*

To view a specific image or types of images

pdsspect 1p*img

To view images from multiple directories:

pdsspect * path/to/other/directory/

From the (i)python command line:

```
>>> from pdsspect.pdsspect import pdsspect
>>> pdsspect()
Displays all of the images from current directory
>>> pdsspect('path/to/different/directory')
Displays all of the images in the different directory
>>> pdsspect ('1p*img')
Displays all of the images that follow the glob pattern
>>> pdsspect('a1.img, b*.img, example/path/x*img')
You can display multiple images, globs, and paths in one window by
separating each item by a command
>>> pdsspect(['a1.img, b3.img, c1.img, d*img'])
You can also pass in a list of files/globs
pdsspect returns a dictionary of the ROIs:
>>> rois = pdsspect(['a1.img, b3.img, c1.img, d*img'])
>>> rois['red'][:2, :2]
array(
    [
        [False, False],
        [False, False]
    ],
    dtype=bool
)
```

# CHAPTER 3

## pdsspect_image_set

The main model for all the views in pdsspect

**class** pdsspect.pdsspect_image_set.**ImageStamp**(*filepath*, *metadata=None*, *logger=None*, *wavelength=nan*, *unit='nm'*)

Bases: ginga.BaseImage.BaseImage

BaseImage for the image view canvas

**Parameters**

- **filepath** (str) – The path to the image to be opened
- **metadata** (*None*) – Metadata for *BaseImage*
- **logger** (*None*) – logger for *BaseImage*
- **wavelength** (float [nan]) – Image's filter wavelength. If nan, will try to use *instrument_models.get_wavelength.get_wavelength()* to get the wavelength
- **unit** (str [nm]) – Wavelength unit. Must be one of *accepted_units*

**pds_image**
PDS3Image – Image object that holds data and the image label

**image_name**
str – The basename of the filepath

**seen**
bool – False if the image has not been seen by the viewer. True otherwise Default if False

**cuts**
tuple – The cut levels of the image. Default is two *None* types

**accepted_units**
list – List of accepted units: nm, um, and AA

**data**
numpy.ndarray – Image data

**get_wavelength**()
astropy.units.quantity.Quantity Copy of the wavelength

**unit**
astropy.units.Unit – The *wavelength* unit

Setting the unit will convert the wavelength value as well. The new unit must also be one of the *accepted_units*

**wavelength**
int – The images wavelength

**class** pdsspect.pdsspect_image_set.**PDSSpectImageSet**(*filepaths*)
Bases: object

Model for each view is pdsspect

The images loaded should all have the same shape. Otherwise the images will have the smallest common shape and not look as expected (i.e., If when loading two images where one image has a shape of `(63, 36)` and the other image has a shape of `(24, 42)`, the displayed shape will be `(24, 36)`. This will cause the first image to have the right side cut off and the second image to have the top cut off). This is done so all ROIs created can apply to the entire list of images. To avoid this behavior, either only open images that have the same shape or open images one at a time.

**Parameters** **filepaths** (list) – List of filepaths to images

**colors**
list of str – List of possible color names to make ROIs.

The possible choices for colors: `red`, `brown`, `lightblue`, `lightcyan`, `darkgreen`, `yellow`, `pink`, `teal`, `goldenrod`, `sienna`, `darkblue`, `crimson`, `maroon`, `purple`, and `eraser (black)`

**selection_types**
list of str – Selection types for making ROIs. The possible types are *Filled Rectangle*, *Filled Polygon*, and *Filled Rectangle*, (single points).

**accepted_units**
list – List of accepted units: `nm`, `um`, and `AA`

**images**
list of *ImageStamp* – Images to view and make selections. Must all have the same dimensions

**filepaths**
list – List of filepaths to images

**current_color_index**
int – Index of the current color in *colors* list for ROI creation (Default is 0)

**add_coords_to_roi_data_with_color**(*coordinates*, *color*)
Add coordinates to ROI data in the with the given color

**Parameters**

- **coordinates** (numpy.ndarray or tuple) – Either a `(m x 2)` array or a tuple of two arrays

  If an array, the first column are the x coordinates and the second are the y coordinates. If a tuple of arrays, the first array are x coordinates and the second are the corresponding y coordinates.

- **color** (str) – The name a color in *colors*

**add_subset**(*subset*)
Add a subset to the list of subsets

**Parameters** **subset** (*SubPDSSpectImageSet*) – Subset to add to the list of subsets

**all_rois_coordinates**
tuple of two numpy.ndarray – Coordinates of where there is a pixel selected in a ROI

**alpha**
float – The alpha value between 0 and 1

Setting the alpha value will change the opacity of all the ROIs and then set the data in the views

**alpha255**
float The alpha value normalized between 0 and 255

**center**
tuple of two float – x and y coordinate of the center of the pan.

Setting the center will move the pan to the new center. The center points cannot result in the pan being out of the image. If they are they will be changed so the pan only goes to the edge.

**color**
str – The current color in the *colors* list determined by *current_color_index*

**create_subset**()
Create a subset and add it to the list of subsets

**Returns** **subset** – The newly created subset

**Return type** *SubPDSSpectImageSet*

**current_image**
*ImageStamp* – The current image determined by *current_image_index*

**current_image_index**
int – Index of the current image in *images*

Setting the index will set the image in the views

**delete_all_rois**()
Delete all of the ROIs

**delete_rois_with_color**(*color*)
Delete the ROIs with the given color

**Parameters** **color** (str) – The name a color in *colors*

**edges**
tuple of four float – The left, bottom, right and top edges of the pan

**filenames**
list of str – Basenames of the *filepaths*

**flip_x**
bool – If True, flip the x axis

Setting the flip_x will display the transformation in the views

**flip_y**
bool – If True, flip the y axis

Setting the flip_y will display the transformation in the views

**get_coordinates_of_color**(*color*)
The coordinates of the ROI with the given color

**Parameters** **color** (*str*) – The name a color in *colors*

**Returns** **coordinates** – The first array are the x coordinates and the second are the corresponding y coordinates

**Return type** tuple of two numpy.ndarray

**map_zoom_to_full_view**()
Get the change in x and y values to the center of the image

**Returns**

- **delta_x** (*float*) – The horizontal distance to the center of the full image
- **delta_y** (*float*) – The vertical distance to the center of the full image

**pan_data**
numpy.ndarray – The data within the pan

**pan_height**
float – Height of the pan area

**pan_roi_data**
numpy.ndarray – The ROI data in the pan

**pan_slice**
numpy.s_ – Slice of pan to extract data from an array

**pan_width**
float – Width of the pan area

**register**(*view*)
Register a View with the model

**remove_subset**(*subset*)
Remove a subset to the list of subsets

**Parameters** **subset** (*SubPDSSpectImageSet*) – Subset to remove to the list of subsets

**reset_center**()
Reset the pan to the center of the image

**selection_index**
int – Index of the ROI selection type

**selection_type**
str – The current selection type in *selection_types* determined by *selection_index*

**set_unit**()
Set each image to *unit*

**simultaneous_roi**
bool – If true, new ROIs appear in every view

Setting *simultaneous_roi* will set all windows to have the same ROIs as the first window. Any new ROI created will appear in each window

**subsets**
list of *SubPDSSpectImageSet* – The list of subsets

**swap_xy**
bool – If True, swap the x and y axis

Setting the swap_xy will display the transformation in the views

**transforms**
tuple of bool – the *flip_x*, *flip_y*, and *swap_xy* transformations

**unit**
str – The image set's current wavelength unit

**unregister**(*view*)
Unregister a View with the model

**x_radius**
float – Half the image width

**y_radius**
float – Half the image height

**zoom**
int – Zoom factor for the pan

The zoom factor determines the width and height of the pan area. For example, if zoom=2, then the width would be half the image width and the height would be half the image height. Setting the zoom will adjust the pan size in the views.

**class** pdsspect.pdsspect_image_set.**SubPDSSpectImageSet**(*parent_set*)
Bases: *pdsspect.pdsspect_image_set.PDSSpectImageSet*

A Subset of an *PDSSpectImageSet*

**Parameters** **parent_set** (*PDSSpectImageSet*) – The subset's parent

**parent_set**
*PDSSpectImageSet* – The subset's parent

**add_coords_to_roi_data_with_color**(*coordinates*, *color*)
Add coordinates to ROI data in the with the given color

**Parameters**

- **coordinates** (numpy.ndarray or tuple) – Either a (m x 2) array or a tuple of two arrays

  If an array, the first column are the x coordinates and the second are the y coordinates. If a tuple of arrays, the first array are x coordinates and the second are the corresponding y coordinates.

- **color** (str) – The name a color in colors

**add_subset**(*subset*)
Add a subset to the list of subsets

**Parameters** **subset** (*SubPDSSpectImageSet*) – Subset to add to the list of subsets

**all_rois_coordinates**
tuple of two numpy.ndarray – Coordinates of where there is a pixel selected in a ROI

**alpha**
float – The alpha value between 0 and 1

Setting the alpha value will change the opacity of all the ROIs and then set the data in the views

**alpha255**
float The alpha value normalized between 0 and 255

**center**
tuple of two float – x and y coordinate of the center of the pan.

Setting the center will move the pan to the new center. The center points cannot result in the pan being out of the image. If they are they will be changed so the pan only goes to the edge.

**color**
`str` – The current color in the `colors` list determined by `current_color_index`

**create_subset**()
Create a subset and add it to the list of subsets

**Returns** **subset** – The newly created subset

**Return type** *SubPDSSpectImageSet*

**current_image**
*ImageStamp* – The current image determined by *current_image_index*

**current_image_index**
`int` – Index of the current image in `images`

Setting the index will set the image in the views

**delete_all_rois**()
Delete all of the ROIs

**delete_rois_with_color**(*color*)
Delete the ROIs with the given color

**Parameters** **color** (`str`) – The name a color in `colors`

**edges**
`tuple` of four `float` – The `left`, `bottom`, `right` and `top` edges of the pan

**filenames**
`list` of `str` – Basenames of the `filepaths`

**flip_x**
`bool` – If True, flip the x axis

Setting the `flip_x` will display the transformation in the views

**flip_y**
`bool` – If True, flip the y axis

Setting the `flip_y` will display the transformation in the views

**get_coordinates_of_color**(*color*)
The coordinates of the ROI with the given color

**Parameters** **color** (`str`) – The name a color in `colors`

**Returns** **coordinates** – The first array are the x coordinates and the second are the corresponding y coordinates

**Return type** `tuple` of two `numpy.ndarray`

**map_zoom_to_full_view**()
Get the change in x and y values to the center of the image

**Returns**

- **delta_x** (`float`) – The horizontal distance to the center of the full image
- **delta_y** (`float`) – The vertical distance to the center of the full image

**pan_data**
`numpy.ndarray` – The data within the pan

**pan_height**
float – Height of the pan area

**pan_roi_data**
numpy.ndarray – The ROI data in the pan

**pan_slice**
numpy.s_ – Slice of pan to extract data from an array

**pan_width**
float – Width of the pan area

**register**(*view*)
Register a View with the model

**remove_subset**(*subset*)
Remove a subset to the list of subsets

**Parameters** **subset** (*SubPDSSpectImageSet*) – Subset to remove to the list of subsets

**reset_center**()
Reset the pan to the center of the image

**selection_index**
int – Index of the ROI selection type

**selection_type**
str – The current selection type in selection_types determined by *selection_index*

**set_unit**()
Set each image to *unit*

**simultaneous_roi**
bool – If true, new ROIs appear in every view

Setting *simultaneous_roi* will set all windows to have the same ROIs as the first window. Any new ROI created will appear in each window

**subsets**
list of *SubPDSSpectImageSet* – The list of subsets

**swap_xy**
bool – If True, swap the x and y axis

Setting the swap_xy will display the transformation in the views

**transforms**
tuple of bool – the *flip_x*, *flip_y*, and *swap_xy* transformations

**unit**
str – The image set's current wavelength unit

**unregister**(*view*)
Unregister a View with the model

**x_radius**
float – Half the image width

**y_radius**
float – Half the image height

**zoom**
int – Zoom factor for the pan

The zoom factor determines the width and height of the pan area. For example, if `zoom=2`, then the width would be half the image width and the height would be half the image height. Setting the zoom will adjust the pan size in the views.

# CHAPTER 4

## pdsspect_view

Window to pan the main image and open other dialog windows

**class** pdsspect.pdsspect_view.**PDSSpectViewController**(*model*, *view*)
Bases: object

Controller for the *PDSSpectView*

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view** (PDSpectView) – View to control

**change_pan_center**(*x*, *y*)
Change the center of the pan

**Parameters**

- **x** (float) – The x coordinate of the center of the pan
- **y** (float) – The y coordinate of the center of the pan

**change_pan_size**(*zoom*)
Change the size of the pan by changing the zoom factor

**Parameters** **zoom** (float) – The new zoom factor

**class** pdsspect.pdsspect_view.**PDSSpectView**(*image_set*)
Bases: PyQt5.QtWidgets.QWidget, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

View to pan the main image

**Parameters** **image_set** (*PDSSpectImageSet*) – pdsspect model

**image_set**
*PDSSpectImageSet* – pdsspect model

**controller**
*PDSSpectViewController*

**main_layout**
QtWidgets.QVBoxLayout

**zoom_layout**
QtWidgets.QHBoxLayout – Layout for *zoom_label* and *zoom_text*

**zoom_label**
QtWidgets.QLabel – Label the *zoom_text* text box

**zoom_text**
QtWidgets.QLineEdit – Text box to enter the zoom factor. Zoom will change on `return key`

**view_canvas**
PDSImageViewCanvas – canvas to place the image on

**pan**
ginga.canvas.types.basic.Box – Pan that represents the pan. Data inside the pan is displayed in *PanView*

**pan_view**
*PanView* – View to display data in the *pan*

**adjust_pan_size**()
Change the pan size as deterined by *image_set*

**arrow_key_move_center**(*view_canvas*, *keyname*)
Adjust center with arrow press by a single pixel

**Parameters**

- **view_canvas** (*view_canvas*) – The view canvas
- **keyname** (str) – Name of the key

**change_center**(*view_canvas*, *button*, *data_x*, *data_y*)
Adjust center to mouse position. Arguments supplied by callback

**Parameters**

- **view_canvas** (*view_canvas*) – The view canvas
- **button** (qtpy.QtCore.QMouseButton) – The mouse button pressed
- **data_x** (float) – x coordinate of mouse
- **data_y** (float) – y coordinate of the mouse

**change_zoom**()
Change zoom to what is in the text box

**move_pan**()
Move the pan as determined by the *image_set*

**redraw**()
Redraw the *view_canvas*

**set_image**()
Set image on *view_canvas*

**set_transforms**()
Apply transforms flip_x, flip_y, and switch_xy

**zoom_with_scroll**(*view_canvas*, *zoom_event*)
Change the zoom by 1 with the mouse wheel

**Parameters**

- **view_canvas** (*view_canvas*) – The view canvas
- **zoom_event** (ginga.Bindings.ScrollEvent) – The zoom event

**class** pdsspect.pdsspect_view.**PDSSpectViewWidget**(*image_set*)

Bases: PyQt5.QtWidgets.QWidget

Widget to hold the the differen *PDSSpectView*

**Parameters** **image_set** (*PDSSpectImageSet*) – pdsspect model

**image_set**

*PDSSpectImageSet* – pdsspect model

**create_spect_view**(*image_set*)

Create a *PDSSpectView* and add to the widget

**Parameters** **image_set** (*PDSSpectImageSet*) – pdsspect model

**Returns** **spect_view** – *PDSSpectView* added to the widget

**Return type** *PDSSpectView*

# CHAPTER 5

## pan_view

Display data in pan and make ROI selections

**class** pdsspect.pan_view.**PanViewController**(*image_set*, *view*)
Bases: object

Controller for the *PanView*

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view** (*PanView*) – View to control

**image_set**
*PDSSpectImageSet* – pdsspect model

**view**
*PanView* – View to control

**add_ROI**(*coordinates*)
Add a region of interest

**Parameters** **coordinates** (numpy.ndarray or tuple) – Either a (m x 2) array or a tuple of two arrays

If an array, the first column are the x coordinates and the second are the y coordinates. If a tuple of arrays, the first array are x coordinates and the second are the corresponding y coordinates.

**erase_ROI**(*coordinates*)
Erase any region of interest inside coordinates

**Parameters** **coordinates** (numpy.ndarray or tuple) – Either a (m x 2) array or a tuple of two arrays

If an array, the first column are the x coordinates and the second are the y coordinates. If a tuple of arrays, the first array are x coordinates and the second are the corresponding y coordinates.

**class** pdsspect.pan_view.**PanView**(*image_set*, *parent=None*)

Bases: PyQt5.QtWidgets.QWidget, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

View of the image inside the pan

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **parent** (*None*) – The parent of the view

**image_set**
*PDSSpectImageSet* – pdsspect model

**controller**
*PanViewController* – The view's controller

**parent**
*None* – The view's parent

**main_layout**
QtWidgets.QVBoxLayout – The main layout of the view

**view_canvas**
*PDSImageViewCanvas* – Canvas to view the image

**check_ROI_in_pan**(*func*)
Wrapper to make sure ROI stays inside the current view

**check_roi_in_process**(*func*)
Wrapper to make sure the roi making is in process

**continue_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Continue the ROI making on click

**extend_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Extend the ROI on mouse motion

**is_erasing**
bool – True if current color is eraser false otherwise

**move_pan**()
Set the data when the pan is moved

**redraw**()
Redraw *view_canvas*

**save_frame**()
Save current frame as image

**set_data**()
Set pan data on the canvas

**set_image**()
Set the data

**set_roi_data**()
Set the ROI data on the canvas

**start_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Start the ROI at the mouse location

**stop_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Stop ROI on right click

**class** pdsspect.pan_view.**PanView**(*image_set*, *parent=None*)

Bases: PyQt5.QtWidgets.QWidget, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

View of the image inside the pan

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **parent** (*None*) – The parent of the view

**image_set**
*PDSSpectImageSet* – pdsspect model

**controller**
*PanViewController* – The view's controller

**parent**
*None* – The view's parent

**main_layout**
QtWidgets.QVBoxLayout – The main layout of the view

**view_canvas**
*PDSImageViewCanvas* – Canvas to view the image

**check_ROI_in_pan**(*func*)
Wrapper to make sure ROI stays inside the current view

**check_roi_in_process**(*func*)
Wrapper to make sure the roi making is in process

**continue_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Continue the ROI making on click

**extend_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Extend the ROI on mouse motion

**is_erasing**
bool – True if current color is eraser false otherwise

**move_pan**()
Set the data when the pan is moved

**redraw**()
Redraw *view_canvas*

**save_frame**()
Save current frame as image

**set_data**()
Set pan data on the canvas

**set_image**()
Set the data

**set_roi_data**()
Set the ROI data on the canvas

**start_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Start the ROI at the mouse location

**stop_ROI**(*view_canvas*, *button*, *data_x*, *data_y*)
Stop ROI on right click

# CHAPTER 6

## pds_image_view_canvas

**class** pdsspect.pds_image_view_canvas.**PDSImageViewCanvas**

Bases: ginga.qtw.ImageViewCanvasQt.ImageViewCanvas

ImageViewCanvas for pdsspect views

**add_subview**(*subview*)

Add a ImageViewCanvas as a subview

**Parameters** **subview** (ginga.qtw.ImageViewCanvasQt) – View canvas to add as a subview

**Raises** TypeError – When subview is not an ImageViewCanvas object

**cut_levels**(*cut_low*, *cut_high*)

Adjust the cut levels of the view and all the subviews

**Parameters**

- **cut_low** (float) – The low cut level
- **cut_high** (float) – The high cut level

**transform**(*flip_x*, *flip_y*, *swap_xy*)

Apply transforms to the view and all the subviews

**Parameters**

- **flip_x** (bool) – Flip x axis if True. Otherwise, do not
- **flip_y** (bool) – Flip y axis if True. Otherwise, do not
- **swap_xy** (bool) – Swap the x and y axis if True. Otherwise, do not

# CHAPTER 7

# selection

Window to pick selection type/color, load/export ROIs and clear ROIS

**class** pdsspect.selection.**SelectionController**(*image_set*, *view*)
Bases: object

Controller for *Selection*

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view** (*Selection*) – View to control

**image_set**
*PDSSpectImageSet* – pdsspect model

**view**
*Selection* – View to control

**add_ROI**(*coordinates*, *color*, *image_set=None*)
Add ROI with the given coordinates and color

**Parameters**

- **coordinates** (numpy.ndarray or tuple) – Either a (m x 2) array or a tuple of two arrays

  If an array, the first column are the x coordinates and the second are the y coordinates. If a tuple of arrays, the first array are x coordinates and the second are the corresponding y coordinates.

- **color** (str) – The name a color in *colors*

**change_alpha**(*new_alpha*)
Change the alpha value to a new alpha value

**Parameters** **new_alpha** (float) – Value between 0 and 100

**change_current_color_index**(*index*)
Change the current color index to a new index

**Parameters** **index** (int) – The new color index

**change_selection_index**(*index*)
Change the selection index to a new index

**Parameters** **index** (int) – The new selection index

**clear_all**()
Clear all ROIs

**clear_current_color**()
Clear all the ROIs with the currently selcted color

**class** pdsspect.selection.**Selection**(*image_set*, *parent=None*)
Bases: PyQt5.QtWidgets.QWidget, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

Window to make/clear/load/export ROIs and choose selection mode/color

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **parent** (*None*) – Parent of the view

**image_set**
*PDSSpectImageSet* – pdsspect model

**parent**
*None* – Parent of the view

**controller**
*SelectionController* – View controller

**type_label**
QtWidgets.QLabel – Label for the selection menu

**selection_menu**
QtWidgets.QComboBox – Drop down menu of selection types

**type_layout**
QtWidgets.QHBoxLayout – Horizontal box layout for selection

**color_label**
QtWidgets.QLabel – Label for the *color_menu*

**color_menu**
QtWidgets.QComboBox – Drop down menu for color selection

**color_layout**
QtWidgets.QHBoxLayout – Horizontal box layout for color selection

**opacity_label**
QtWidgets.QLabel – Label for the *opacity_slider*

**opacity_slider**
QtWidgets.QSlider – Slider to determine opacity for ROIs

**opacity_layout**
QtWidgets.QHBoxLayout – Horizontal box layout for opacity slider

**clear_current_color_btn**
QtWidgets.QPushButton – Button to clear all ROIs will the current color

**clear_all_btn**
QtWidgets.QPushButton – Button to clear all ROIs

**export_btn**
QtWidgets.QPushButton – Export ROIs to .npz file

**load_btn**
QtWidgets.QPushButton – Load ROIs from .npz file

**simultaneous_roi_box**
QtWidgets.QPushButton – When checked, new ROIs appear in every window

**main_layout**
QtWidgets.QVBoxLayout – Vertical Box layout for main layout

**change_alpha**(*new_alpha*)
Change alpha value when *opacity_slider* value changes

**change_color**(*index*)
Change the color when color selected in *color_menu*

**change_selection_type**(*index*)
Change selection type when selected in *selection_menu*

**clear_all**()
Clear all ROIs

**clear_current_color**()
Clear all ROIs with current color

**export**(*save_file*)
Export ROIS to the given filename

**Parameters** **save_file** (str) – File with .npz extension to save ROIs

**load_selections**(*selected_files*)
Load ROIs from selected files

**Parameters** **selected_files** (list of str) – Paths to files storing ROIs

**open_save_dialog**()
Open save file dialog and save rois to given filename

**show_open_dialog**()
Open file dialog to select .npz files to load ROIs

# CHAPTER 8

## transforms

Apply simple tranformations to the views

**class** pdsspect.transforms.**TransformsController**(*image_set*, *view*)
Bases: object

Controller for *Transforms*

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view** (*Transforms*) – View to control

**image_set**
*PDSSpectImageSet* – pdsspect model

**view**
*Transforms* – View to control

**set_flip_x**(*flip_x*)
Set *flip_x*

**Parameters** **flip_x** (bool) – True to flip x axis, otherwise, False

**set_flip_y**(*flip_y*)
Set *flip_y*

**Parameters** **flip_y** (bool) – True to flip y axis, otherwise, False

**set_swap_xy**(*swap_xy*)
Set *swap_xy*

**Parameters** **swap_xy** (bool) – True to swap x and y axis, otherwise, False

**class** pdsspect.transforms.**Transforms**(*image_set*, *view_canvas*)
Bases: PyQt5.QtWidgets.QDialog, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

Window to apply simple transformations

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view_canvas** (*pds_image_view_canvas.PDSImageViewCanvas*) – The view canvas to apply transformations to

**image_set**
*PDSSpectImageSet* – pdsspect model

**view_canvas**
*pds_image_view_canvas.PDSImageViewCanvas* – The view canvas to apply transformations to

**controller**
*TransformsController* – The view's controller

**flip_x_label**
QtWidgets.QLabel – Label for *flip_x_box*

**flip_x_box**
QtWidgets.QCheckBox – Flip x axis when checked

**flip_y_label**
QtWidgets.QLabel – Label for *flip_y_box*

**flip_y_box**
QtWidgets.QCheckBox – Flip y axis when checked

**swap_xy_label**
QtWidgets.QLabel – Label for *swap_xy_box*

**swap_xy_box**
QtWidgets.QCheckBox – Swap x and y axis when checked

**layout**
QtWidgets.QGridLayout – Layout for widget

**flip_x_checked**(*state*)
Flip x axis when checked

**Parameters** **state** (int) – The state of the checkbox (this argument is ignored and the state is checked in a more explicit way)

**flip_y_checked**(*state*)
Flip y axis when checked

**Parameters** **state** (int) – The state of the checkbox (this argument is ignored and the state is checked in a more explicit way)

**swap_xy_checked**(*state*)
Swap x and y axis when checked

**Parameters** **state** (int) – The state of the checkbox (this argument is ignored and the state is checked in a more explicit way)

# CHAPTER 9

## roi

Region of interest creation

**class** pdsspect.roi.**ROIBase**(*image_set*, *view_canvas*, *color='red'*, *linewidth=1*, *linestyle='solid'*, *showcap=False*, *fill=True*, *fillcolor=None*, *alpha=1.0*, *drawdims=False*, *font='Sans Serif'*, *fillalpha=1.0*, ***kwargs*)

Bases: ginga.canvas.types.basic.Polygon

Base class for all ROI shapes

**contains_arr**(*x_arr*, *y_arr*)

Determine whether the points in the ROI are in arrays

The arrays must be the same shape. The arrays should be result of `np.mgrid[y1:y2:1, x1:x2:1]`

**Parameters**

- **x_arr** (numpy.ndarray) – Array of x coodinates
- **y_arr** (numpy.ndarray) – Array of y coordinates

**Returns** **result** – Boolean array where coordinates that are in ROI are True

**Return type** numpy.ndarray

**continue_ROI**(*data_x*, *data_y*)

Abstract method to continue the ROI process

**create_ROI**(*points=None*)

Create a Region of interest

**Parameters** **points** (list of tuple of two int) – Points that make up the vertices of the ROI

**Returns** **coordinates** – `m x 2` array of coordinates.

**Return type** numpy.ndarray

**static draw_after**(*func*)

Wrapper to redraw canvas after function

**extend_ROI**(*data_x*, *data_y*)
Abstract method to extend the ROI process

**lock_coords_to_pixel**(*data_x*, *data_y*)
Lock the coordinates to the pixel

The coordinate of the pixel is located at the bottom left corner of the pixel square while the center of the pixel .5 units up and to the right of the corner. So if the given coordinates are (2.3, 3.7), the pixel coordinates will be (2, 3) and the center of the pixel is (2.5, 3.5). This method locks the given coordinates to the pixel's coordinates

**Parameters**

- **data_x** (float) – The given x coordinate
- **data_y** (float) – The given y coordinate

**Returns**

- **point_x** (float) – The corresponding x pixel coordinate
- **point_y** (float) – The corresponding y pixel coordinate

**static lock_coords_to_pixel_wrapper**(*func*)
Wrapper to lock data coordinates to the corresponding pixels

**start_ROI**(*data_x*, *data_y*)
Abstract method to start the ROI process

**stop_ROI**(*data_x*, *data_y*)
Abstract method to stop the ROI process

**class** pdsspect.roi.**Polygon**(*image_set*, *view_canvas*, *color='red'*, *linewidth=1*, *linestyle='solid'*, *showcap=False*, *fill=True*, *fillcolor=None*, *alpha=1.0*, *drawdims=False*, *font='Sans Serif'*, *fillalpha=1.0*, ***kwargs*)
Bases: *pdsspect.roi.ROIBase*

Polygon Region of Interest

**continue_ROI**(*data_x*, *data_y*)
Create new vertex on the polygon on left click

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**extend_ROI**(*data_x*, *data_y*)
Extend the current edge of the polygon on mouse motion

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**start_ROI**(*data_x*, *data_y*)
Start the ROI process

The ROI will be a ginga.canvas.types.basic.Path object

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**stop_ROI**(*data_x*, *data_y*)
Close the polygon on right click

The polygon will close based on last left click and not on the right click. There must be more than 2 points to formulate a polygon

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**class** pdsspect.roi.**Rectangle**(*image_set*, *view_canvas*, *color='red'*, *linewidth=1*, *linestyle='solid'*, *showcap=False*, *fill=True*, *fillcolor=None*, *alpha=1.0*, *drawdims=False*, *font='Sans Serif'*, *fillalpha=1.0*, ***kwargs*)
Bases: *pdsspect.roi.ROIBase*

Rectangle Region of interest

**extend_ROI**(*data_x*, *data_y*)
Exend the rectangle on region of interest on mouse motion

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**start_ROI**(*data_x*, *data_y*)
Start the region of interest on left click

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**stop_ROI**(*data_x*, *data_y*)
Stop the region of interest on right click

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**class** pdsspect.roi.**Pencil**(**args*, ***kwargs*)
Bases: *pdsspect.roi.ROIBase*

Select individual pixels

**continue_ROI**(*data_x*, *data_y*)
Add another pixel on left click

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**move_delta**(*delta_x*, *delta_y*)
Override the move_delta function to move all the points

**Parameters**

- **delta_x** (float) – Change in the x direction

- **delta_y** (float) – Change in the y direction

**start_ROI**(*data_x*, *data_y*)
Start choosing pixels on left click

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**stop_ROI**(*data_x*, *data_y*)
Set all pixels as roi cooridinates on right click

**Parameters**

- **data_x** (float) – The x coordinate
- **data_y** (float) – The y coordinate

**Returns coordinates** – Coordinates of points selected

**Return type** numpy.ndarray

# CHAPTER 10

# basic

**class** pdsspect.basic.**BasicHistogramModel**(**args*, ***kwargs*)

Bases: *pdsspect.histogram.HistogramModel*

Model for the hhistograms in the Basic Widgets

**connected_models**

list – Other *BasicHistogramModel* for other histograms

**bins**

int The number of bins the histogram uses

Setting the bins will notify the views that the bins have changed

**connect_model**(*model*)

Connect another model to this model

**model**

*BasicHistogramModel* – Connect the model to current model

**Raises** ValueError – When *model* is not *BasicHistogramModel*

**cut_high**

float The higher cut level

Setting the high cut value will adjust the cut values in the image view and notify the views that the high cut value changed.

**cut_low**

float The lower cut level

Setting the low cut value will adjust the cut values in the image view and notify the views that the low cut value changed

**cuts**

tuple The lower and higher cut levels. If the lower and higher cut levels are not set, use the image_view cut levels

Setting the cuts will adjust the cut levels in the image viewer and notify the views that the cuts have changed. The low cut must be less than the high cut, otherwise they will be switched to satisfy that condition.

**data**
ndarray The current image data

**disconnect_from_all_models**()
Disconnect all models from this model

**disconnect_model**(*model*)
Disconnect another model from this model

**model**
*BasicHistogramModel* – Disconnect the model from current model

**Raises** ValueError – When *model* is not *BasicHistogramModel*

**image_view**
ImageViewCanvas The image view canvas

Setting the image view will reset the data

**register**(*view*)
Register a view with the model

**Parameters** **view** (QtWidgets.QWidget) – A view that utilizes this model

**restore**()
Restore the cut levels

**set_data**()
Set the data the histogram is to display

**unregister**(*view*)
Unregister a view with the model

**Parameters** **view** (QtWidgets.QWidget) – A view that utilizes this model

**view_cuts**
tuple The image_view cut levels

**warn**(*title*, *message*)
Display a warning box

Each view must define a warn method that returns a boolean value: True when a warning box is displayed and False when a warning box not displayed. Only one display box will be displayed. This is because multiple views should not have different handling for the same errors.

**class** pdsspect.basic.**BasicHistogramController**(*model*, *view*)
Bases: *pdsspect.histogram.HistogramController*

Controller for histogram views

**Parameters**

- **model** (*BasicHistogramModel*) – histogram model
- **view** (object) – View with *BasicHistogramModel* as its model

**model**
*BasicHistogramModel* – histogram model

**view**
object – View with *BasicHistogramModel* as its model

**restore**()
Restore the histogram

**set_cut_high**(*cut_high*)
Set the high cut level to a new value

**Parameters** **cut_high** (float) – New high cut value

**set_cut_low**(*cut_low*)
Set the low cut level to a new value

**Parameters** **cut_low** (float) – New low cut value

**set_cuts**(*cut_low*, *cut_high*)
Set both the low and high cut levels

**Parameters**

- **cut_low** (float) – New low cut value
- **cut_high** (float) – New high cut value

**class** pdsspect.basic.**BasicHistogramWidget**(**args*, ***kwargs*)
Bases: *pdsspect.histogram.HistogramWidget*

*HistogramWidget* in a different layout

**class** pdsspect.basic.**BasicController**(*image_set*, *view*)
Bases: object

Controller for *Basic* window

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view** (*Basic*) – View to control

**image_set**
*PDSSpectImageSet* – pdsspect model

**view**
*Basic* – View to control

**change_current_image_index**(*new_index*)
Change the current image index to a new index

**Parameters** **new_index** (int) – The new index for images to determine the current image

**class** pdsspect.basic.**BasicWidget**(*image_set*, *view_canvas*)
Bases: PyQt5.QtWidgets.QWidget

Widget to hold each basic window

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view_canvas** (*PDSImageViewCanvas*) – view canvas

**image_set**
*PDSSpectImageSet* – pdsspect model

**basics**
list of *Basic* – *Basic* in the widget

**add_basic**(*image_set*, *view_canvas*)
Add a *Basic* to the widget

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view_canvas** (*PDSImageViewCanvas*) – view canvas

**connect_model**(*basic*)
Connect the models of other basic windows to the given window

The models are connected when they have the same current image

**Parameters** **basic** (*Basic*) – Basic window connect/disconnect its histogram model to others

**class** pdsspect.basic.**Basic**(*image_set*, *view_canvas*, *basic_widget*)
Bases: PyQt5.QtWidgets.QWidget, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

Window to apply cut levels and choose the current image

**Parameters**

- **image_set** (*PDSSpectImageSet*) – pdsspect model
- **view_canvas** (*PDSImageViewCanvas*) – Canvas to view the image

**image_set**
*PDSSpectImageSet* – pdsspect model

**view_canvas**
*PDSImageViewCanvas* – Canvas to view the image

**controller**
*BasicController* – Controller for view

**image_menu**
QtWidgets.QComboBox – Drop down menu to pick the current image

**histogram**
*HistogramModel* – Model for the *histogram_widget*

**histogram_widget**
*BasicHistogramWidget* – The histogram widget to adjust the cut levels

**layout**
QtWidgets.QVBoxLayout – The main layout

**change_image**(*new_index*)
Change the image when new image selected in *image_menu*

**Parameters** **new_index** (int) – The new index to determine the current image

**set_image**()
When the image is set, adjust the histogram

# CHAPTER 11

## histogram

**class** pdsspect.histogram.**HistogramModel**(*image_view*, *cut_low=None*, *cut_high=None*, *bins=100*)

Bases: object

Model for a Histogram which can apply cut levels to an image

Any View that utilizes this model must define the following methods: set_data, change_cut_low, change_cut_high, change_cuts, warn, and change_bins. The warn method must return a boolean and if more than one view utilizes this model, you should consider only one actually creating a warning box and return True while the others just return False.

**Parameters**

- **image_view** (ImageViewCanvas) – The image view canvas
- **cut_low** (float) – The lower cut level
- **cut_high** (float) – The higher cut level
- **bins** (int) – The number of bins the histogram uses

**bins**

int The number of bins the histogram uses

Setting the bins will notify the views that the bins have changed

**cut_high**

float The higher cut level

Setting the high cut value will adjust the cut values in the image view and notify the views that the high cut value changed.

**cut_low**

float The lower cut level

Setting the low cut value will adjust the cut values in the image view and notify the views that the low cut value changed

**cuts**
tuple The lower and higher cut levels. If the lower and higher cut levels are not set, use the image_view cut levels

Setting the cuts will adjust the cut levels in the image viewer and notify the views that the cuts have changed. The low cut must be less than the high cut, otherwise they will be switched to satisfy that condition.

**data**
ndarray The current image data

**image_view**
ImageViewCanvas The image view canvas

Setting the image view will reset the data

**register**(*view*)
Register a view with the model

**Parameters** **view** (QtWidgets.QWidget) – A view that utilizes this model

**restore**()
Restore the cut levels

**set_data**()
Set the data the histogram is to display

**unregister**(*view*)
Unregister a view with the model

**Parameters** **view** (QtWidgets.QWidget) – A view that utilizes this model

**view_cuts**
tuple The image_view cut levels

**warn**(*title*, *message*)
Display a warning box

Each view must define a warn method that returns a boolean value: True when a warning box is displayed and False when a warning box not displayed. Only one display box will be displayed. This is because multiple views should not have different handling for the same errors.

**class** pdsspect.histogram.**HistogramController**(*model*, *view*)
Bases: object

Controller for histogram views

**Parameters**

- **model** (*HistogramModel*) – histogram model
- **view** (object) – View with *HistogramModel* as its model

**model**
*HistogramModel* – histogram model

**view**
object – View with *HistogramModel* as its model

**restore**()
Restore the histogram

**set_bins**(*bins*)
Change the number of bins the histogram uses

**Parameters** **bins** (int) – The number number of bins for the histogram

**set_cut_high**(*cut_high*)
Set the high cut level to a new value

**Parameters** **cut_high** (float) – New high cut value

**set_cut_low**(*cut_low*)
Set the low cut level to a new value

**Parameters** **cut_low** (float) – New low cut value

**set_cuts**(*cut_low*, *cut_high*)
Set both the low and high cut levels

**Parameters**

- **cut_low** (float) – New low cut value
- **cut_high** (float) – New high cut value

**class** pdsspect.histogram.**HistogramWidget**(*model*, *parent=None*)
Bases: PyQt5.QtWidgets.QWidget

View to display the histogram with text boxes for cuts and bins

**Parameters** **model** (*HistogramModel*) – The view's model

**model**
*HistogramModel* – The view's model

**controller**
*HistogramController* – The view's controller

**histogram**
*Histogram* – The histogram itself

**change_bins**()
Change the bins box text

**change_cut_high**()
Set the high cut box text

**change_cut_low**()
Set the low cut box text

**change_cuts**()
Set the low and high cut boxes' text

**keyPressEvent**(*event*)
When the enter button is pressed, adjust the cut levels and bins

**warn**(*title*, *message*)
Displayed a timed message box the warning

**class** pdsspect.histogram.**Histogram**(*model*)
Bases: matplotlib.backends.backend_qt5agg.FigureCanvasQTAgg

The Histogram View

**Parameters** **model** (*HistogramModel*) – The view's model

**model**
*HistogramModel* – The view's model

**controller**
*HistogramController* – The view's controller

**change_bins**()
Adjust the number of bins without adjusting the lines

**change_cut_high**(*draw=True*)
Change the position of the right line to the high cut level

**change_cut_low**(*draw=True*)
Change the position of the left line to the low cut level

**change_cuts**()
Change the position of the left & right lines to respective cuts

**set_data**(*reset_vlines=True*)
Set the histogram's data

**Parameters** **reset_vlines** (bool) – Reset the vertical lines to the default cut levels if True, otherwise False. True by default

# CHAPTER 12

## roi_plot

Parent classes for any widget that plots data

**class** pdsspect.roi_plot.**ROIPlotModel**(*image_set*)
Bases: object

Model for ROI Plot and accompanying widget

**Parameters** **image_set** (*PDSSpectImageSet*) – pdsspect model

**selected_colors**
list – Colors to display in the histogram

**latex_units**
list of 3 str – The latex strings of *pdsspect_image_set.PDSSpectImageSet.accepted_units*

**add_selected_color**(*color*)
Select a color and inform views to display new color

**Parameters** **color** (str) – The color to add

**has_multiple_views**
bool – True if there are multiple views, False otherwise

**image_set**
*PDSSpectImageSet* – Image set that corresponds with the current view

**image_sets**
list – All the image sets, including the current one

**register**(*view*)
Register view with the model

**remove_selected_color**(*color*)
Remove a selected color and inform views to not display the color

**Parameters** **color** (str) – The color to remove

**unit**
str – Latex version of *pdsspect_image_set.PDSSpectImageSet.unit*

**unregister**(*view*)
Unregister view with the model

**view_index**
int – The index of the view to display the ROI data

If there are not multiple views, view_index is automatically `-1`.

**class** pdsspect.roi_plot.**ROIPlotController**(*model*, *view*)
Bases: object

Controller for ROI plot and accompanying widget

**Parameters**

- **model** (*ROIPlotModel*) – The model
- **view** (QtWidgets.QWidget) – The view

**model**
*ROIPlotModel* – The model

**view**
QtWidgets.QWidget – The view

**color_state_changed**(*color*)
Select or remove the color when a checkbox color changes

**Parameters** **color** (str) – The name of the checkbox whose state changed

**remove_color**(*color*)
Remove a given color

**Parameters** **color** (str) – The color to remove

**select_color**(*color*)
Selected a given color

**Parameters** **color** (str) – The color to select

**set_view_index**(*index*)
Set the index of the view

**Parameters** **index** (int) – Index of the view

**class** pdsspect.roi_plot.**ROIPlotWidget**(*model*)
Bases: PyQt5.QtWidgets.QWidget, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

Widget to hold the histogram and checkboxes

Checkboxes are created in *create_color_checkbox()* which is why they do not appear in the `__init__()` method.

**Parameters** **model** (*ROIPlotModel*) – The model

**model**
*ROIPlotModel* – The model

**controller**
*ROIPlotController* – The controller

**checkbox_layout**
QtWidgets.QVBoxLayout – Place the checkboxes vertically

**main_layout**
QtWidgets.QGridLayout – Place in grid layout so histogram stretches while boxes are stationary

**roi_plot**
*ROIPlot* – The plot of ROI data

**save_btn**
QtWidgets.QPushButton – Save the plot as an image

**red_checkbox**
*ColorCheckBox* – Red checkbox that displays red ROI data when checked

**brown_checkbox**
*ColorCheckBox* – Brown checkbox that displays brown ROI data when checked

**lightblue_checkbox**
*ColorCheckBox* – Lightblue checkbox that displays lightblue ROI data when checked

**lightcyan_checkbox**
*ColorCheckBox* – Lightcyan checkbox that displays lightcyan ROI data when checked

**darkgreen_checkbox**
*ColorCheckBox* – Darkgreen checkbox that displays darkgreen ROI data when checked

**yellow_checkbox**
*ColorCheckBox* – Yellow checkbox that displays yellow ROI data when checked

**pink_checkbox**
*ColorCheckBox* – Pink checkbox that displays pink ROI data when checked

**teal_checkbox**
*ColorCheckBox* – Teal checkbox that displays teal ROI data when checked

**goldenrod_checkbox**
*ColorCheckBox* – Goldenrod checkbox that displays goldenrod ROI data when checked

**sienna_checkbox**
*ColorCheckBox* – Sienna checkbox that displays sienna ROI data when checked

**darkblue_checkbox**
*ColorCheckBox* – Darkblue checkbox that displays darkblue ROI data when checked

**crimson_checkbox**
*ColorCheckBox* – Crimson checkbox that displays crimson ROI data when checked

**maroon_checkbox**
*ColorCheckBox* – Maroon checkbox that displays maroon ROI data when checked

**purple_checkbox**
*ColorCheckBox* – Purple checkbox that displays purple ROI data when checked

**add_view**(*index=None*)
Add a view box to the widget

**Parameters** **index** (int [Default None]) – The index to add the view to

**check_color**(*checkbox_color*)
Called when the state a checkbox is changed

**Parameters** **checkbox_color** (str) – The color label of the check box

**check_view_checkbox**(*view_checkbox*)
Check the view box at the given index

**Parameters** **view_checkbox** (*ViewCheckBox*) – The view check box whose state changed

**create_color_checkbox**(*color*)
Create a checkbox with the given color

**Parameters** **color** (str) – The color to name the checkbox

**save_plot**()
Save the plot as an image

**class** pdsspect.roi_plot.**ROIPlot**(*model*)
Bases: matplotlib.backends.backend_qt5agg.FigureCanvasQTAgg, pdsspect.pdsspect_image_set.PDSSpectImageSetViewBase

Plot of the data in each ROI color

**Parameters**

- **model** (*ROIPlotModel*) – The model
- **image_set** (*PDSSpectImageSet*) – pdsspect model

**model**
*ROIPlotModel* – The model

**image_set**
*PDSSpectImageSet* – pdsspect model

**set_roi_data**()
Set data when ROI is created/destroyed or checkbox is toggled

**class** pdsspect.roi_plot.**ColorCheckBox**(*color*)
Bases: PyQt5.QtWidgets.QCheckBox

Custom checkbox that emits its color (str) when toggled

**Parameters** **color** (str) – The color to name the checkbox

**color**
str – The color to name the checkbox

**stateChanged**
QtCore.Signal – Signal that emits a string when check box changes its state

Read more about Signals here

**nextCheckState**()
Adjust checkbox's toggle & emit color when checkbox is clicked

**class** pdsspect.roi_plot.**ViewCheckBox**(*index*)
Bases: PyQt5.QtWidgets.QCheckBox

Custom checkbox that emits its index (int) when toggled

**Parameters** **index** (int) – The index of the view

**index**
int – The index of the view

**stateChanged**
QtCore.Signal – Signal that emits the box itself when check box changes its state

Read more about Signals here

**nextCheckState**()
Adjust checkbox's toggle & emit checkbox when checkbox is clicked

# CHAPTER 13

## roi_histogram

**class** pdsspect.roi_histogram.**ROIHistogramModel**(*image_set*)

Bases: *pdsspect.roi_plot.ROIPlotModel*

Model for ROI histogram and accompanying widget

**add_selected_color**(*color*)

Select a color and inform views to display new color

**Parameters** **color** (str) – The color to add

**compare_data**

bool – True if *image_index* is not -1

**has_multiple_views**

bool – True if there are multiple views, False otherwise

**image_index**

int – The index of the image to which to compare data with

When setting *image_index*, it may be changed to -1 if the image is the same as the current image. Furthermore, when setting the *view_index*, the *image_index* may be changed to -1 if the *view_index* and the *current_image_index* are the same.

**image_set**

*PDSSpectImageSet* – Image set that corresponds with the current view

**image_sets**

list – All the image sets, including the current one

**register**(*view*)

Register view with the model

**remove_selected_color**(*color*)

Remove a selected color and inform views to not display the color

**Parameters** **color** (str) – The color to remove

**unit**

str – Latex version of *pdsspect_image_set.PDSSpectImageSet.unit*

**unregister**(*view*)
Unregister view with the model

**view_index**
int – The index of the view to display the ROI data

If there are not multiple views, view_index is automatically -1.

**xdata**(*color*)
Data inside a ROI with the given color for the current image

**Parameters** **color** (str) – The color of the ROI

**Returns** **data** – Data in ROI color for the xaxis

**Return type** numpy.ndarray

**xlim**
list of two float – min max of current image's data

**ydata**(*color*)
Data inside a ROI with the given color for the image in the menu

**Parameters** **color** (str) – The color of the ROI

**Returns** **data** – Data in ROI color for the yaxis

**Return type** numpy.ndarray

**ylim**
list of two float – min max of yaxis image

**class** pdsspect.roi_histogram.**ROIHistogramController**(*model*, *view*)
Bases: *pdsspect.roi_plot.ROIPlotController*

Controller for ROI histogram and accompanying widget

**Parameters**

- **model** (*ROIHistogramModel*) – The model
- **view** (*ROIHistogramWidget* or *ROIHistogram*) – The view

**model**
*ROIHistogramModel* – The model

**view**
*ROIHistogramWidget* or *ROIHistogram* – The view

**color_state_changed**(*color*)
Select or remove the color when a checkbox color changes

**Parameters** **color** (str) – The name of the checkbox whose state changed

**remove_color**(*color*)
Remove a given color

**Parameters** **color** (str) – The color to remove

**select_color**(*color*)
Selected a given color

**Parameters** **color** (str) – The color to select

**set_image_index**(*index*)
Set the index of the image in the menu

**Parameters** **index** (int) – Index of the image menu

**set_view_index**(*index*)
Set the index of the view

**Parameters** **index** (int) – Index of the view

**class** pdsspect.roi_histogram.**ROIHistogramWidget**(*model*)
Bases: *pdsspect.roi_plot.ROIPlotWidget*

Widget to hold the histogram and check boxes

**Parameters** **model** (*ROIHistogramModel*) – The model

**model**
*ROIHistogramModel* – The model

**controller**
*ROIHistogramController* – The controller

**image_menu**
QtWidgets.QComboBox – Menu to select image for y axis

**select_image**(*index*)
Select an image when image is selected in the menu

**Parameters** **index** (int) – The index of the selected image

**class** pdsspect.roi_histogram.**ROIHistogram**(*model*)
Bases: *pdsspect.roi_plot.ROIPlot*

Histogram view of the data in each ROI color

**Parameters** **model** (*ROIHistogramModel*) – The model

**model**
*ROIHistogramModel* – The model

**set_data**()
Set the data of the selected colors on the histogram

**set_image**()
Set data when image is changed

# CHAPTER 14

## roi_line_plot

**class** pdsspect.roi_line_plot.**ROILinePlotModel**(*image_set*)

Bases: *pdsspect.roi_plot.ROIPlotModel*

Model for ROI Line plot and widget

**add_selected_color**(*color*)

Select a color and inform views to display new color

**Parameters** **color** (str) – The color to add

**data_with_color**(*color*)

Get the data inside the ROI color if the image has a wavelength

**Parameters** **color** (str) – The color of the ROI

**Returns** **data** – Sorted list of arrays of data by wavelength

**Return type** list or numpy.ndarray

**has_multiple_views**

bool – True if there are multiple views, False otherwise

**image_set**

*PDSSpectImageSet* – Image set that corresponds with the current view

**image_sets**

list – All the image sets, including the current one

**register**(*view*)

Register view with the model

**remove_selected_color**(*color*)

Remove a selected color and inform views to not display the color

**Parameters** **color** (str) – The color to remove

**unit**

str – Latex version of *pdsspect_image_set.PDSSpectImageSet.unit*

**unregister**(*view*)
Unregister view with the model

**view_index**
int – The index of the view to display the ROI data

If there are not multiple views, view_index is automatically -1.

**wavelengths**
list – Sorted list of wavelengths in the *image_set*

**class** pdsspect.roi_line_plot.**ROILinePlotController**(*model*, *view*)
Bases: *pdsspect.roi_plot.ROIPlotController*

Controller for *ROILinePlotWidget*

**color_state_changed**(*color*)
Select or remove the color when a checkbox color changes

**Parameters** **color** (str) – The name of the checkbox whose state changed

**remove_color**(*color*)
Remove a given color

**Parameters** **color** (str) – The color to remove

**select_color**(*color*)
Selected a given color

**Parameters** **color** (str) – The color to select

**set_view_index**(*index*)
Set the index of the view

**Parameters** **index** (int) – Index of the view

**class** pdsspect.roi_line_plot.**ROILinePlotWidget**(*model*)
Bases: *pdsspect.roi_plot.ROIPlotWidget*

Widget to hold line plot and check boxes

**Parameters** **model** (*ROILinePlotModel*) – The model

**model**
*ROILinePlotModel* – The model

**controller**
*ROILinePlotController* – The controller

**class** pdsspect.roi_line_plot.**ROILinePlot**(*model*)
Bases: *pdsspect.roi_plot.ROIPlot*

Line plot of ROI data

**Parameters** **model** (*ROILinePlotModel*) – The model

**model**
*ROILinePlotModel* – The model

**set_data**()
Set the data of the selected colors on the line plot

# CHAPTER 15

## set_wavelength

**class** pdsspect.set_wavelength.**SetWavelengthModel**(*image_set*)

Bases: object

Model for *SetWavelengthWidget*

**Parameters** **image_set** (*PDSSpectImageSet*) – pdsspect model

**image_set**
*PDSSpectImageSet* – pdsspect model

**accepted_units**
list – List of accepted units – nm, um, and AA

**current_image**
*ImageStamp* – Current image in menu

**current_image_index**
int – Index of current image in menu

**display_current_wavelength**()
Display current wavelength in registered views

**unit**
str – *image_set* unit

Setting the :attr‘unit‘ will set the *image_set* unit

**unit_index**
int Index of – attr‘unit‘ in *accepted_units*

**class** pdsspect.set_wavelength.**SetWavelengthController**(*model*, *view*)

Bases: object

Controller for *SetWavelengthWidget*

**Parameters**

- **model** (*SetWavelengthModel*) – Model for *SetWavelengthWidget*
- **view** (*SetWavelengthWidget*) – The view to control

**model**
*SetWavelengthModel* – Model for *SetWavelengthWidget*

**view**
*SetWavelengthWidget* – The view to control

**change_unit**(*index*)
Set the model's *SetWavelengthModel.unit*

**Parameters** **index** (int) – Index of *SetWavelengthModel.accepted_units* to change the *SetWavelengthModel.unit* to

**set_current_image_index**(*index*)
Set the model's *SetWavelengthModel.current_image_index*

**Parameters** **index** (int) – Index to change *SetWavelengthModel.current_image_index* to

**set_image_wavelength**(*wavelength*)
Set the model's *SetWavelengthModel.current_image* wavelength

**Parameters** **wavelength** (float) – The model's *SetWavelengthModel.current_image* new wavelength

**class** pdsspect.set_wavelength.**SetWavelengthWidget**(*model*)
Bases: PyQt5.QtWidgets.QMainWindow

Widget to set images wavelength

Using a QtWidgets.QMainWindow for the status bar at the bottom.

**Parameters** **model** (*SetWavelengthModel*) – Model for *SetWavelengthWidget*

**model**
*SetWavelengthModel* – Model for *SetWavelengthWidget*

**controller**
*SetWavelengthController* – The widgets controller

**image_menu**
QtWidgets.QComboBox – Menu to choose the image to set the wavelength

**wavelength_text**
QtWidgets.QLineEdit – Text box to enter and display wavelength

**units_menu**
QtWidgets.QComboBox – Menu to choose unit of wavelength

**main_layout**
QtWidgets.QHBoxLayout – Main layout of widget

**change_unit**(*index*)
Change *SetWavelengthModel.unit* to unit in *units_menu*

**Parameters** **index** (int) – Index of *SetWavelengthModel.accepted_units* to change the *SetWavelengthModel.unit* to

**display_current_wavelength**()
Display the *SetWavelengthModel.current_image* wavelength in *wavelength_text*

**select_image**(*index*)
Select current image

**Parameters** **index** (int) – Index to change *SetWavelengthModel.current_image_index* to

**set_wavelength**()
Set the *SetWavelengthModel.current_image* wavelength to value in *wavelength_text*

**show_status_bar_wavelength_set**()
Alert user wavelength is set

CHAPTER 16

# Instrument Models

## 16.1 Supported Instruments

- **MER**
  - Pancam
- **MSL**
  - Mastcam
- **Cassini**
  - Imaging Science Subsystem (ISS)

## 16.2 get_wavelength

Get the wavelength from an image's label

`instrument_models.get_wavelength.`**`get_wavelength`**(*label*, *unit*)

Get the filter wavelength from the label of an image

See *Supported Instruments* for full list of supported missions and instruments. If the instrument is not supported, *`get_wavelength()`* will return `nan`.

**Parameters**

- **`label`** (`pvl.PVLModule`) – Image's label
- **`unit`** (`str` [`nm`]) – The wavelength unit. Best practice for `unit` to exist in *`pdsspect.pdsspect_image_set.ImageStamp.accepted_units`*

**Returns**

**wavelength** – The filter wavelenth from the image rounded to 3 decimal places.

If image does not have a wavelength or the instrument is not *supported*, wavelength with be `nan`

**Return type** float

**See also:**

*instrument_models.mastcam.Mastcam.get_wavelength()* Get Mastcam wavelength

*instrument_models.pancam.Pancam.get_wavelength()* Get Pancam wavelength

*instrument_models.cassini_iss.CassiniISS.get_wavelength()* Get Cassini ISS wavelength

instrument_models.get_wavelength.**is_pancam**(*label*)
Determine if label is for a Pancam image

**Parameters** **label** (pvl.PVLModule) – Image's label

**Returns** **is_pancam** – True if label is from a Pancam image, False otherwise

**Return type** bool

instrument_models.get_wavelength.**is_mastcam**(*label*)
Determine if label is for a Mastcam image

**Parameters** **label** (pvl.PVLModule) – Image's label

**Returns** **is_mastcam** – True if label is from a Mastcam image, False otherwise

**Return type** bool

instrument_models.get_wavelength.**is_instrument**(*func*)
Wrapper for instrument determining functions

Tries the function, if there is a TypeError, then return False. The TypeError will occur when the label's get() method returns None

## 16.3 instrument

Provide base class for all instrument models

**class** instrument_models.instrument.**InstrumentBase**(*label*)
Abstract Base Class for all instrument models

**Parameters** **label** (pvl.PVLModule) – Image's label

**label**
pvl.PVLModule – Image's label

**get_wavelength**(*unit*, **args*, ***kwargs*)
Abstract method to get the image's wavelength

**Parameters** **unit** (str [nm]) – The wavelength unit. Best practice for unit to exist in *pdsspect.pdsspect_image_set.ImageStamp.accepted_units*

**Returns** **wavelength** – The image's filter wavelength

**Return type** float

## 16.4 mastcam

**class** instrument_models.mastcam.**Mastcam**(*label*)
Bases: *instrument_models.instrument.InstrumentBase*

Model to get the filter wavelength of a Mastcam image

See Mastcam Multispectral Imaging on the Mars Science Laboratory Rover: Wavelength Coverage and Imaging Strategies at the Gale Crater Field Site for more details on Mastcam's filter wavelengths

**group**
str – INSTRUMENT_STATE_PARMS

**wavelength_key1**
str – CENTER_FILTER_WAVELENGTH

**wavelength_key2**
str – FILTER_CENTER_WAVELENGTH

**get_wavelength**(*unit='nm'*)
Get the wavelength from mastcam image

**Parameters** **unit** (str [nm]) – The wavelength unit. Best practice for unit to exist in *pdsspect.pdsspect_image_set.ImageStamp.accepted_units*

**Returns** **wavelength** – Filter wavelength of the mastcam image

**Return type** float

## 16.5 pancam

**class** instrument_models.pancam.**Pancam**(*label*)
Bases: *instrument_models.instrument.InstrumentBase*

Model to get the filter wavelength of a Patcam image

See Pancam for more details on Pancam's filter wavelengths.

**pancam_left**
str – PANCAM_LEFT

**pancam_right**
str – PANCAM_RIGHT

**unit**
str – nm

**left_filters**
dict – Key is the filter number and the value is the wavelength for PancamL

**right_filters**
dict – Key is the filter number and the value is the wavelength for PancamR

**camera**
bool – Images camera. Should either be *left_filters* or *right_filters*

**filter_num**
int – The images filter number

**get_wavelength**(*unit='nm'*)
Get the filter wavelength from the image

**Parameters** **unit** (str [nm]) – The wavelength unit. Best practice for unit to exist in *pdsspect.pdsspect_image_set.ImageStamp.accepted_units*

**Returns** **wavelength** – The image's filter wavelength

**Return type** float

**is_left**
bool – True if image is from Pancam Left

**is_right**
bool – True if image is from Pancam Right

## 16.6 cassini_iss

**class** instrument_models.cassini_iss.**CassiniISS**(*label*)
Bases: *instrument_models.instrument.InstrumentBase*

Model to get the filter wavelength from Cassini ISS image

See Cassini Imaging Science Subsystem (ISS) Data User's Guide (Page 149) for table of filter name and corresponding wavelengths. We use the effective wavelength rather than the center wavelength.wavelength

**NA_filters**
dict – Dictionary of the ISS Narrow Angle Camera filter names and wavelengths

Key is the two filternames and the value is the wavelength in nm

**WA_filters**
dict – Dictionary of the ISS Wide Angle Camera filter names and wavelengths

Key is the two filternames and the value is the wavelength in nm

**unit**
str – The default unit is nm

**filter_name**
str – The image's filter names joined by a comma and space

For example, in the label the filtername appears as `("CL1","UV3")` and so *filter_name* returns `'CL1, UV3'`

**get_wavelength**(*unit='nm'*)
Get the image's filter wavelength

**Parameters** **unit** (str [nm]) – The desired wavelength of the unit

**Returns** **wavelength** – The image's filter wavelength rounded to 3 decimal places

**Return type** float

**is_NA**
bool – True if image is from Narrow Angle Camera

**is_WA**
bool – True if image is from Wide Angle Camera

CHAPTER 17

# Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. To understand and better read the code, you should have an understanding of the Model-View-Controller (MVC) architecture.

You can contribute in many ways:

## 17.1 Types of Contributions

### 17.1.1 Report Bugs

Report bugs at https://github.com/planetarypy/pdsspect/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 17.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

### 17.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

### 17.1.4 Write Documentation

pdsspect could always use more documentation, whether as part of the official pdsspect docs, in docstrings, or even on the web in blog posts, articles, and such.

### 17.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/planetarypy/pdsspect/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

## 17.2 Get Started!

Ready to contribute? Here's how to set up *pdsspect* for local development.

1. Fork the *pdsspect* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pdsspect.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pdsspect
$ cd pdsspect/
$ pip install -r requirements.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ make lint
$ make test
$ make test-all
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

## 17.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/planetarypy/pdsspect/pull_requests and make sure that the tests pass for all supported Python versions.

## 17.4 Tips

To run a subset of tests:

```
py.test tests/
```

CHAPTER 18

# Credits

## 18.1 Development Lead

- PlanetaryPy Developers <contact@planetarypy.com>

## 18.2 Contributors

- Perry Vargas <perrybvargas@gmail.com>
- Austin Godber <godber@uberhip.com>

CHAPTER 19

# History

## 19.1 0.1.1 (“2017-08-21”)

- Make compatible to be opened by other programs like pystamps

## 19.2 0.1.0 (“2017-08-20”)

- First release on PyPi

CHAPTER 20

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## i



## p

# Index

## D



## E



## F

## G



## H



## I

## K



## L



## M



## N

## O



## P



## Q



## R

## S

## T

## U



## V



## W

## X



## Y



## Z