thermoscientific

Phenom Programming Interface

Create your customized solution to fit your workflow

PyPhenom version 2.1

ThermoFisher SCIENTIFIC

Contents

Introduction

The Phenom Programming Interface (PPI) is a tool that allows users to take control of the Phenom microscopes through scripting. PyPhenom is a Python library that has been developed to access the PPI functionalities.

This document describes the installation procedure and the usage of the PyPhenom version 2.1 library. It also provides a reference guide for the most commonly-used functions.

1. Compatibility

PyPhenom 2.1 is compatible with a wide range of Python configurations, from 2.7 to 3.11. Different versions of the PyPhenom library are available for the following Python configurations on Windows:

Table 1 PyPhenom versions for available Python configurations

Python version	32-bit	64-bit
2.7	PyPhenom27-x86.msi	PyPhenom27-x64.msi
3.4	PyPhenom34-x86.msi	PyPhenom34-x64.msi
3.5	PyPhenom35-x86.msi	PyPhenom35-x64.msi
3.6	PyPhenom36-x86.msi	PyPhenom36-x64.msi
3.7	PyPhenom37-x86.msi	PyPhenom37-x64.msi
3.8	PyPhenom38-x86.msi	PyPhenom38-x64.msi
3.9	PyPhenom39-x86.msi	PyPhenom39-x64.msi
3.10	PyPhenom310-x86.msi	PyPhenom310-x64.msi
3.11	PyPhenom311-x86.msi	PyPhenom311-x64.msi
3.12	PyPhenom312-x86.msi	PyPhenom312-x64.msi

2. Installation instructions

3.1 Python installation

Download and install the preferred Python configuration. We advise to install the latest 3.11 64-bit version from https://www.python.org/downloads/.

3.2 PyPhenom installation and activation

Run the PyPhenom##-x##.msi installer from the provided PyPhenom.zip that matches your Python configuration, according to Table 1. The installer will automatically detect the location of your Python installation and install PyPhenom there. If multiple versions of Python are installed, be careful to choose the desired installation directory.

The Windows PyPhenom installer asks for PPI license details during installation, as shown in Figure 1. A PPI license is linked to a specific microscope. If you wish to control different microscopes, you need a license for each and every one of them. If you enter valid PPI license details in the PPI license registration window, the Python scripts can connect to the Phenom without supplying the license details every time. Alternatively, the license installation can be skipped, but must be specified in each PPI script.

PyPhenom is now ready for use.

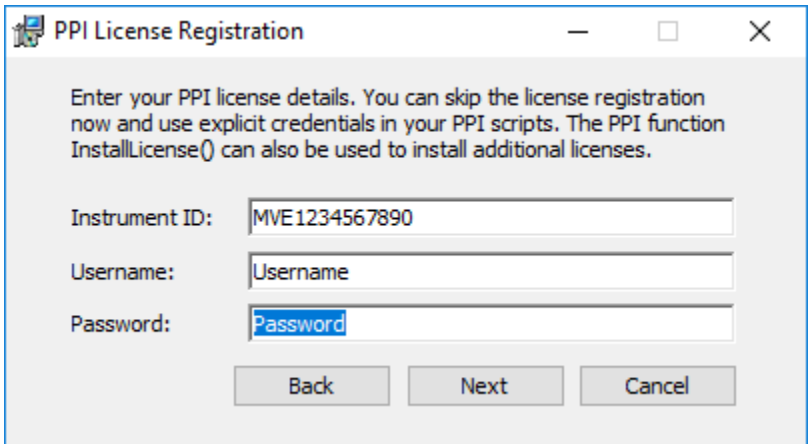


Figure 1 PPI license registration window

3. PPI License

If no PPI License was provided during the installation, or if you want to add a license for another microscope, the PPI license can also be installed later with a Python script:

```
import PyPhenom as ppi

ppi.InstallLicense('MVE012345678', 'Username', 'Password')
```

To check which licenses are installed on your computer:

```
import PyPhenom as ppi

for license in ppi.GetLicenses():
    print('Phenom-ID:', license.instrumentId)
    print('username:', license.username)
    print('password:', license.password)
```

Uninstalling a license can be done by specifying for which instrument ID the license has to be uninstalled:

```
import PyPhenom as ppi

ppi.UninstallLicense('MVE012345678')
```

In the PyPhenom.zip file provided with the PyPhenom installation, a password manager (LicenseManager.pyw[1]) tool is included. The PPI License Manager, shown in Figure 2, allows to view, add and delete PyPhenom licenses on your computer without using code.

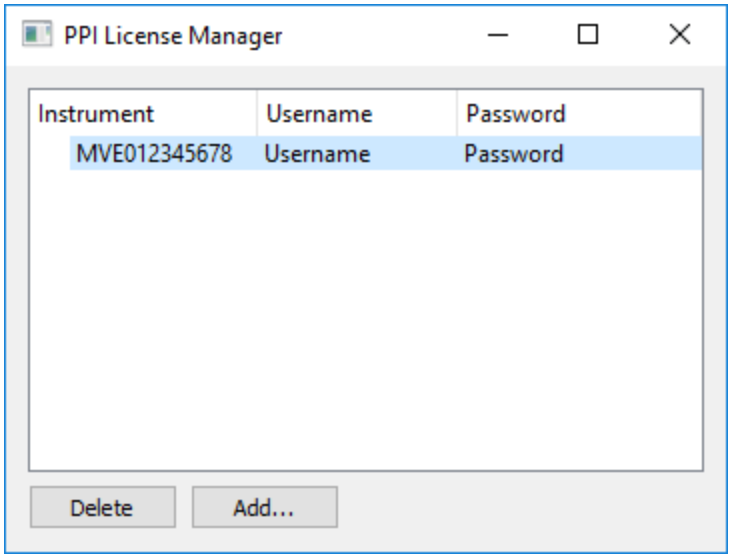


Figure 2 PPI license manager window

4. Getting Started

For your first script, create a new text file with this content and save it as Acquire.py:

```
import PyPhenom as ppi

phenom = ppi.Phenom()
acq = phenom.SemAcquireImage(1920, 1200, 16)
acq.metadata.label = 'PPI Image'
acq = ppi.AddDatabar(acq)
ppi.Save(acq, 'Image.tiff')
```

Now setup your Phenom for live SEM imaging and run this file. If everything is ok, it should create a new image file called Image.tiff with an image from the Phenom. The script consists of a few steps:

- The PyPhenom library is imported.
- Set the connection to the microscope.

[1] LicenseManager.pyw requires PySide. PySide is not supported in Python 3.7 and higher

- Acquisition of an image with certain size (in this case 1920 x 1200 pixels) and with a number of averaged frames (in this case 16).
- In the metadata a label is added with the string: PPI Image.
- A data-bar is added to the image.
- The picture is saved as a tiff file.

These steps will be explained in detail later in this document.

5. Tutorials

This section includes a more detailed description of how PyPhenom can be used. The tutorial examples presented in this section are cut into small snippets and thoroughly explained. In the appendix the complete scripts can be found.

Every PyPhenom script starts with importing PyPhenom in Python:

```
import PyPhenom as ppi
```

This will enable all the PyPhenom functionalities in Python.

The tutorials are written for the Python 3.x. version of PyPhenom. Python 2.7 has some syntax differences with Python 3, therefore the tutorials might not work.

6.1 Find and connect to your Phenom

In this section we explain how all Phenoms connected to the local network can be found and how you can connect to your Phenom. All Phenoms on the local network can be found using the following command:

```
import PyPhenom as ppi

phenoms = ppi.FindPhenoms(1)
```

The identification information of every Phenom that is found can be parsed into a list and be printed:

```
for phenom in phenoms:
    print('Phenom ip: ', phenom.ip)
    print('Phenom id: ', phenom.name)
```

There are different ways to connect to the Phenom based on how the license is set up:

1. The license information is set during the installation of PyPhenom
2. The license information is set with the IP-address of the Phenom
3. The license information is set with the identification ID of the Phenom

1. If the license information is provided during the installation of PyPhenom, to connect to the Phenom only involves one command:

```
phenom = ppi.Phenom()
```

2. With the IP-address of the Phenom, specify the IP-address, the username and password and then connect to the Phenom:

```
address = '000.000.000'
username = 'Username provided by ThermoFisher Scientific'
password = 'Password provided by ThermoFisher Scientific'

phenom = ppi.Phenom(address, username, password)
```

3. The identification ID of the Phenom, specify the PhenomID, the username and password and then connect to the Phenom:

```
PhenomID = 'MVE012345678'
username = 'Username provided by ThermoFisher Scientific'
password = 'Password provided by ThermoFisher Scientific'

phenom = ppi.Phenom(PhenomID, username, password)
```

The input PhenomID, IP-address, password and username are strings. The class ppi.Phenom() creates an object which represents the Phenom. If no Phenom is available, a simulation version of the Phenom can be used within PPI. This object can be obtained by:

```
phenom_simulator = ppi.Phenom('Simulator', '', '')
```

6.2 Basic Phenom commands

This tutorial explains the basic settings of the Phenom:

- Get the instrument mode and the operational mode of the Phenom
- Move the sample from the NavCam to SEM and vice versa
- Move to a position on a sample
- Adjust the beam settings
- Focus
- Viewing modes
- Vacuum settings (Phenom XL-only)

This tutorial assumes that PPI is loaded and that a Phenom object is created as *phenom*.

6.2.1 Get instrument mode and operational mode

To get the status of the Phenom, the instrument mode can be queried:

```
instrument_mode = phenom.GetInstrumentMode()
```

The possible instrument modes are:

Off	In "off" state (but still running)
Operational	Operational
Standby	Standby (takes approx. 5 minutes to become operational)
Hibernate	In hibernation mode (takes approx. 6 minutes to become operational)
Initializing	Initializing
ClosingDown	Shutting down (going to “off” state)
Error	In error condition

To change the instrument mode (e.g. turning on the Phenom or putting it in standby), the instrument mode can be set (as an example the phenom is put into operational mode):

```
phenom.Activate()
```

To get the status of the action of the Phenom, the operational mode can be queried:

```
operational_mode = phenom.GetOperationalMode()
```

The possible operational modes are:

Unavailable	Instrument is unavailable
Loadpos	Sample is in loading position
Unloading	Sample is moving to unload position
SelectingNavCam	Sample is moving to NavCam
SelectingSem	Sample is moving to SEM
LiveNavCam	Live viewing mode under the NavCam
LiveSem	Live viewing mode under the SEM
AcquireNavCamImage	Performing NavCam acquisition
AcquireSemImage	Performing SEM acquisition

6.2.2 Loading and unloading of the sample and the NavCAM functionalities

To unload the sample from the Phenom:

```
phenom.Unload()
```

To load the sample to the Phenom:

```
phenom.Load()
```

To move the sample to the NavCam:

```
phenom.MoveToNavCam()
```

To acquire the NavCam image, the NavCam image parameters (CamParams) must be created first:

```
acqCamParams = ppi.CamParams()
```

The NavCam parameters can be set:

```
acqCamParams.size = ppi.Size(912, 912)
acqCamParams.nFrames = 1
```

The original NavCam delivers only 480 x 480 RGB images; any other size is an invalid parameter. The improved NavCam in Phenom G2 and onward generations provides images with a resolution of 912 x 912 pixels. When requesting a different size, the image is scaled by the Phenom to the requested resolution.

To acquire the NavCam image:

```
acqNavCam = phenom.NavCamAcquireImage(acqCamParams)
```

The NavCam image can be saved to a file:

```
ppi.Save(acqNavCam, 'NavCam.tiff')
```

To move from the NavCam to the SEM:

```
phenom.MoveToSem()
```

Once the sample is loaded into the SEM, the Phenom will perform an auto contrast and brightness action. This lasts a few seconds, so the script can only continue after the auto brightness and contrast has finished. To enforce this wait, a sleep command can be used:

```
time.sleep(7)
```

To use this the time module must be first imported to Python, at the start of the script. The 7 in between the brackets will let the code wait for 7 seconds before it continues.

6.2.3 Sample navigation

The sample position can be moved to an absolute position with the following command:

```
phenom.MoveTo(0.0, 0.0)
```

Where the coordinates x and y are expressed in meters.

The sample can also be moved relative to its current position:

```
phenom.MoveBy(1e-6, 1e-6)
```

Where the distance in x and y is expressed in meters.

6.2.4 Beam settings

The high tension of the electron beam can be set to 5kV, 10kV or 15kV, but also values in between. This can be done by using the following command:

```
phenom.SetSemHighTension(-5000)
```

Where in brackets is the high tension. Note the negative sign in front.

To set the beam spot intensity use the following command:

```
phenom.SetSemSpotSize(3.3)
```

Where 3.3 corresponds to the beam spot intensity. To reproduce the intensities displayed in the Phenom GUI:

- Low = 2.1
- Image = 3.3
- Point = 4.3
- Map = 5.1

6.2.5 Magnification, focus and adjust brightness/contrast

The magnification is defined as the ratio between the size of the display and the imaged field width. To set the magnification essentially translates to setting the field width.

The current horizontal field width (HFW) can be obtained with the command:

```
hfw  = phenom.GetHFW()
```

Where the function returns a value in meters. Or the magnification can be obtained from the current horizontal field width:

```
magnification  = ppi.MagnificationFromFieldWidth(phenom.GetHFW(), 0.5)
```

To correctly calculate the magnification the physical image width needs to be entered. In this case an image width of 0.5 meters is used.

The HFW can be set by (with 10 µm as an example):

```
phenom.SetHFW(10e-6)
```

Where the value in brackets is expressed in meters.

To ease the user, the MagnificationToFieldWidth() function can be used to adjust HFW in terms of a magnification value:

```
phenom.SetHFW(ppi.MagnificationToFieldWidth(5000), 0.5)
```

To correctly calculate the horizontal field width the physical image width needs to be entered. In this case an image width of 0.5 meters is used.

Adjusting the focus essentially translates into setting the working distance.

The current working distance can be queried:

```
wd = phenom.GetSemWD()
```

Where the function returns a value in meters.

The working distance can be changed with this command:

```
phenom.SetSemWD(7e-3)
```

Where the value in brackets is in meters.

To find automatically the optimal working distance:

```
phenom.SemAutoFocus()
```

The current contrast and brightness can be queried:

```
brightness = phenom.GetSemBrightness()
contrast = phenom.GetSemContrast()
```

The contrast and brightness can be set with these commands:

```
phenom.SetSemBrightness(0.4)
phenom.SetSemContrast(0.6)
```

Or an auto contrast brightness can be performed:

```
phenom.SemAutoContrastBrightness()
```

6.2.6 Selection of the detector

The viewing mode that is active in the Phenom user interface can be queried:

```
viewingMode = phenom.GetSemViewingMode()
```

The *SemViewingMode.ScanParams.detector* mode can be used to select a detector modes (e.g. BSD, SED, Topo A, STEM, etc.). The available detector modes are here listed:

All	Use all BSD detector segments
NorthSouth	Subtract bottom two BSD segments from top two (Topo A)
EastWest	Subtract left two BSD segments from right two (Topo B)
A	Select only BSD segment A
B	Select only BSD segment B
C	Select only BSD segment C
D	Select only BSD segment D
Sed	Select SE detector
Stem	Select Stem detector

To change the detector mode, choose one of the *ppi.DetectorMode.xxx* values to set your preference and the phenom.SetSemViewingMode to change the live viewing mode in the Phenom GUI:

```
viewingMode.scanParams.detector = ppi.DetectorMode.NorthSouth
phenom.SetSemViewingMode(viewingMode)
```

The SED detector must be enabled before the detector mode function can be used.

```
phenom.SemEnableSed()
viewingMode = phenom.GetSemViewingMode()
viewingMode.scanParams.detector = ppi.DetectorMode.Sed
phenom.SetSemViewingMode(viewingMode)
```

Enabling the SED detector can be done only if the chamber pressure is below 1 Pa. if the system is in low vacuum mode, the function returns an error.

Disabling the SED detector:

```
phenom.SemDisableSed()
viewingMode = phenom.GetSemViewingMode()
viewingMode.scanParams.detector = ppi.DetectorMode.All
phenom.SetSemViewingMode(viewingMode)
```

To set the STEM detector (only available on Phenom Pharos) as the active viewing mode:

```
viewingMode = phenom.GetSemViewingMode()
viewingMode.scanParams.detector = ppi.DetectorMode.Stem
phenom.SetSemViewingMode(viewingMode)
```

To change the segments of the STEM detector the SetStemSegmentSelection is used. The StemSegmentSelection class is passed to set the active segments. For example, to set the STEM detector to bright field the following code can be used:

```
StemSegmentSelection = ppi.StemSegmentSelection.BF
phenom.SetStemSegmentSelection(StemSegmentSelection)
```

To acquire an image using the STEM detector the following code can be used:

```
viewingMode = phenom.GetSemViewingMode()
viewingMode.scanParams.detector = ppi.DetectorMode.Stem
phenom.SetSemViewingMode(viewingMode)

StemSegmentSelection = ppi.StemSegmentSelection.DF
phenom.SetStemSegmentSelection(StemSegmentSelection)
phenom.SemAutoContrastBrightness()

scanParams = ppi.ScanParams()
scanParams.size = ppi.Size(1920,1200)
scanParams.nFrames = 4
scanParams.detector = ppi.DetectorMode.Stem

acq = phenom.SemAcquireImage(scanParams)
ppi.Save(acq,"STEM.tiff")
```

6.2.7 Vacuum settings

In the Phenom XL the vacuum level can be changed. To query the current vacuum state, use the following command:

```
pressure = phenom.SemGetVacuumChargeReductionState().pressureEstimate
```

The resulting pressure estimate is in pascals.

To change the state of the vacuum (with off as an example):

```
phenom.SemSetTargetVacuumChargeReduction(ppi.VacuumChargeReduction.Off)
```

The following options are available:

- ppi.VacuumChargeReduction.High
- ppi.VacuumChargeReduction.Low
- ppi.VacuumChargeReduction.Off

Where:

- High refers to 60 Pa chamber vacuum
- Low refers to 10 Pa chamber vacuum
- Off refers to 1 Pa chamber vacuum

For P-series, the vacuum level can be checked by getting information on the sample holder.

6.3 Image Acquisition

The acquisition of SEM images is explained in this section. To acquire an image, the scan parameters must be created:

```
acqScanParams = ppi.ScanParams()
```

Next, the scan parameters can be set:

```
acqScanParams.size = ppi.Size(1920,1200)
acqScanParams.detector = ppi.DetectorMode.All
acqScanParams.nFrames = 16
acqScanParams.hdr= False
acqScanParams.scale = 1.0
```

The scan parameters include:

- **size** is the dimensions of the image to scan (max is 2048x2048).
- **detector** is the detector configuration as mentioned in section 6.2.6.
- **nFrames** is the number of frames to average for signal to noise improvement.
- **hdr** is a Boolean to use High Dynamic Range mode; if true it returns a 16-bit raw image without contrast/brightness applied; otherwise an 8-bit image will be acquired with the same contrast and brightness applied as in the live Phenom image.
- **Scale** is the scale of the acquisition within the field of view.

For acquiring an image use the following command:

```
acq = phenom.SemAcquireImage(acqScanParams)
```

An acquisition object contains the image and the metadata which holds the information about the state of the Phenom during the time when it acquired the image. The full acquisition object is structured as follows:

databar	The pixel data of the databar, or the lower part of the image.
fullImage	The pixel data of the full image, including databar.
image	The pixel data of the SEM image only.
magnification	The magnification at which the acquisition has been performed.
metadata	The AcquisitionState object for this acquisition.

When the acquisition object is stored as a tiff, jpeg or png file, this information is stored together with the image. The tiff format is preferred.

It is important to understand that magnification is defined as the width of the displayed image divided by the horizontal field width of the image. By default, the magnification is calculated as if it is displayed, in full screen, on the default 19-inch Phenom monitor. If the image is displayed in another size and the correct magnification needs to be calculated, the exact (physical) width of the displayed image needs to be filled in the metadata using acq.metadata.displayWidth. These dimensions change when you resize a window or print your image.

The label text is defined in the metadata and is set in acq.metadata.dataBarLabel. To add a data-bar to the image the method ppi.AddDatabar() is used.

```
acq.metadata.displayWidth = 0.5
acq.metadata.dataBarLabel = "Label"
acqWithDatabar = ppi.AddDatabar(acq)
```

The acquisitions can be saved to the local hard drive:

```
ppi.Save(acq, 'example.tiff')
ppi.Save(acqWithDatabar, 'exampleWithDatabar.tiff')
```

Earlier acquired Phenom images can also be loaded after they have been saved:

```
acq = ppi.Load('example.tiff')
acqWithDatabar = ppi.Load('exampleWithDatabar.tiff')
```

6.4 Energy-dispersive X-ray spectroscopy (EDS)

6.4.1 EDS licensing

The Energy-dispersive X-ray spectroscopy (EDS) functionality of Element Identification (EID) is available in PyPhenom under an extra license.

Multiple extra licenses are available in PyPhenom for different features (e.g. EDS, EDS mapping and 3DRR). To validate whether a user can use the EDS (spectroscopy) features in PyPhenom the smallest common set of allowed features is checked in the installed licenses. Only the features that are available in all licenses can be used. For example, if two licenses for two different Phenoms are installed on one PC. With one license that has both the spectroscopy and 3DRR features enabled and the other license with only 3DRR enabled. In this case only the 3DRR features can be used on that PC. To be able to use the spectroscopy features the license with only 3DRR has to be deinstalled.

To see if a feature is enabled with the currently installed license(s) the function: ppi.HasLicense() can be used. The function returns a boolean whether the license is available or not. In a simple example:

```
print("Has EDS license: " + str(ppi.HasLicense(ppi.Features.Spectroscopy)))
```

Line and map measurements are available under an additional license. The line and mapping license can be checked with the feature EdsMap:

```
print("Has mapping license: " + str(ppi.HasLicense(ppi.Features.EdsMap)))
```

When a feature is used that is not included in all the installed licenses a runtime exception will be triggered. The error states the missing license. As an example, running a line measurement without a mapping license returns the following error:

```
RuntimeError: Missing license: EdsMap
```

Details on how to install and de-install licenses can be found in chapter 4.

PyPhenom version 2.1 uses the same identification and quantification routines as UI version 1.10

6.4.2 Connecting the EDS detector

To use the EDS functionality in PyPhenom the EDS detector needs to be connected to the computer running the code. This does not have to be the workstation that is supplied with the Phenom. To connect the EDS detector to the computer the USB cable that is plugged in port 8 in figure 3 needs to be connected to the PC running the PyPhenom code.

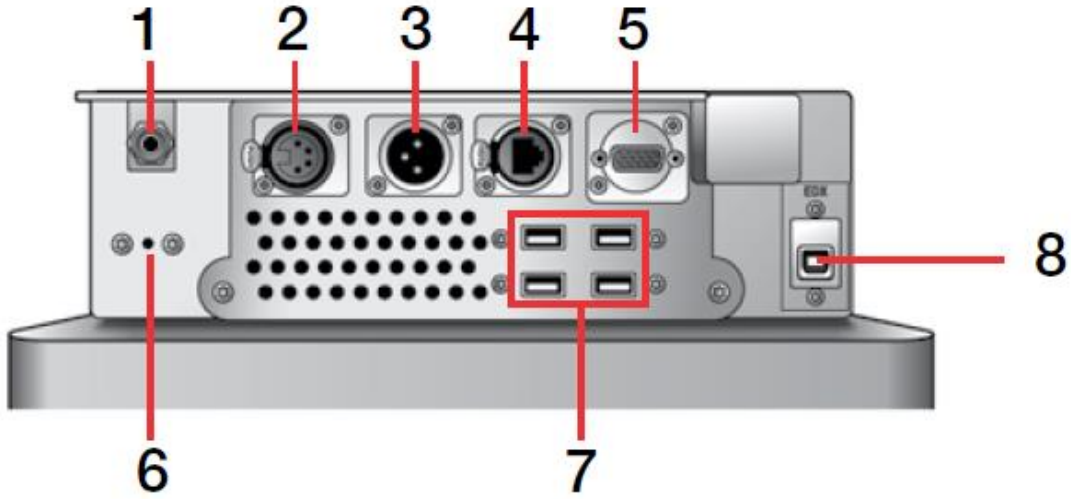


Figure 3 Connector configuration for the Phenom XL

Note: From Phenom server version 6.4 there is unofficial support for connecting the EDS detector over a network connection. To do this the USB-cable running from port 8 needs to be plugged into the USB-ports on the front side of the Phenom (if available), or directly into the ITX-board on the inside of the Phenom. A reboot of the Phenom might be necessary to (re)start the server for the EDS detector.

Only one instance of the spectrometer can run at the same time. For example, an EID acquisition in the Phenom UI cannot be ran simultaneously with an PyPhenom EDS script acquiring EDS data. Therefore, PyPhenom claims exclusive access of the Phenom while acquiring EDS data.

6.4.3 Acquiring EDS data

The acquisition of EDS data will be explained using the following example:

```
import PyPhenom as ppi

phenom = ppi.Phenom('PhenomID', 'Username', 'Password')

analyzer = ppi.Application.ElementIdentification.EdsJobAnalyzer(phenom)

spotData = analyzer.AddSpot(ppi.Position(0, 0), maxTime=10, maxCounts=10000)

analyzer.Wait()

print(ppi.Spectroscopy.Quantify(spotData.spotSpectrum))
ppi.Spectroscopy.WriteMsaFile(spotData.spotSpectrum, 'SpotSpectrum.msa')
```

First PyPhenom is imported. Using PyPhenom the connection with the Phenom is made using ppi.Phenom.

Once the connection is established an analyzer object can be made. The analyzer object controls the acquisitions of EDS data with the Phenom. To create the analyzer ppi.Application.ElementIdentification.EdsJobAnalyzer is used and as an argument the Phenom to which the connection needs to be made is passed.

To the analyzer EDS measurements can be requested. To add, for example, a spot measurement the analyzer.AddSpot is used. In the example the spot is taken at the center of the current image in the Phenom. The acquisition is stopped after 10 seconds or when 10000 counts are reached.

To wait until all requested jobs are finished the analyzer.Wait method is used.

To see the chemical composition of the acquired EDS spectrum ppi.Spectroscopy.Quantify is used. The relevant spectrum is passed into this function.

The acquired spectrum can be saved in an MSA file using ppi.Spectroscopy.WriteMsaFile.

This example is the simplest and most direct way to acquire and analyze the EDS data using PyPhenom. In the next sections each of the elements will be described in more detail.

In PyPhenom the default EDS license there is a minimum interval of 5 seconds in between each EDS acquisition.

6.4.4 Analyzer

EDS spectra are acquired using the EDS job analyzer. Multiple EDS measurements can be requested from the job analyzer. The job analyzer makes sure that the measurements are properly and sequentially acquired. The analyzer can be created like this:

```
analyzer = ppi.Application.ElementIdentification.EdsJobAnalyzer(phenom)
```

When creating the job analyzer, the Phenom which will be used needs to be passed into the analyzer.

To ensure that the beam is always correctly positioned on the sample when doing measurements drift correction is available in the job analyzer. Drift correction works by comparing a reference image with the current image in the Phenom. The beam is realigned using the drift measured between these images. To acquire the reference image for the drift correction:

```
analyzer.AcquireDriftCorrectionReference()
```

Optionally the acquisition size of the drift correction reference image can be passed as an integer. A square image with the passed size (width and height) will be acquired as a reference image. When a reference image is taken the drift correction is activated. It is important that after any stage movement a new reference image is acquired to make sure that no wrong drift corrections are applied.

When acquiring spectra, the analyzer needs to know what presets (i.e. what calibration) needs to be used. To do this the desired preset needs to be selected from the settings object. When the Phenom is calibrated standardly two presets are available: spot and map. Spot is calibrated at the point spot size and map is calibrated at the map spot size. If a custom spot size is used it is advised to use the closest preset or run a custom calibration for this spot size. This spot size can be changed per measurement. The code to request and change the spot size is:

```
settings = phenom.LoadPulseProcessorSettings()
analyzer.preset = settings.spot
```

After the first job has been assigned to the analyzer the analyzer automatically starts acquiring the spectra. The progress of the jobs can be requested with:

```
analyzer.JobProgress()
```

The analyzer will return which job is currently being done.

When one or multiple acquisitions need to be stopped the following possible methods are available and can be used depending on the need:

```
analyzer.Abort()
analyzer.AbortCurrentJob()
analyzer.AbortJob()
analyzer.StopCurrentJob()
analyzer.StopJob()
```

Aborting will immediately stop the acquisition of all data.

Stop will finish the current acquisition of data to have a complete set of information (e.g. a line or map will finish its current pass and then terminate).

The current job or a specific job can be stopped or aborted. To stop the current job the analyzer.StopCurrentJob is used. To stop a specific job the job ID needs to be passed into the analyzer.StopJob. This unique ID is issued when the job is created. For aborting the equivalent methods with abort are used.

A wait method is available for the analyzer to let the code wait until all jobs have finished:

```
analyzer.Wait()
```

While acquiring spectra the analyzer can catch and pass errors. To catch errors the following method can be used:

```
analyzer.EdsAnalyzerFailure.connect(lambda msg:
print('EdsAnalyzerFailure("{}")'.format(msg)))
```

This method will print the error message to the console window.

6.4.5 Positioning

The positioning of the EDS locations in the Phenom is done in relative coordinates. The relative coordinates are calculated relative to the longest axis of the image. The center of the image is defined as (0,0) and to the right and down are the positive directions. If a wide screen image is used the horizontal value ranges from -0.5 to 0.5 and, vertically the range is from $-0.5 \times \frac{height}{width}$ to $0.5 \times \frac{height}{width}$. The default aspect ratio for Phenom images is 16:10, thus the range would be: $\pm 0.5 \frac{10}{16} = \pm 0.3125$. This is visualized in figured 4. To create a position object ppi.Position is used:

```
position = ppi.Position(x, y)
```

Where x and y values are relative positions in floats.

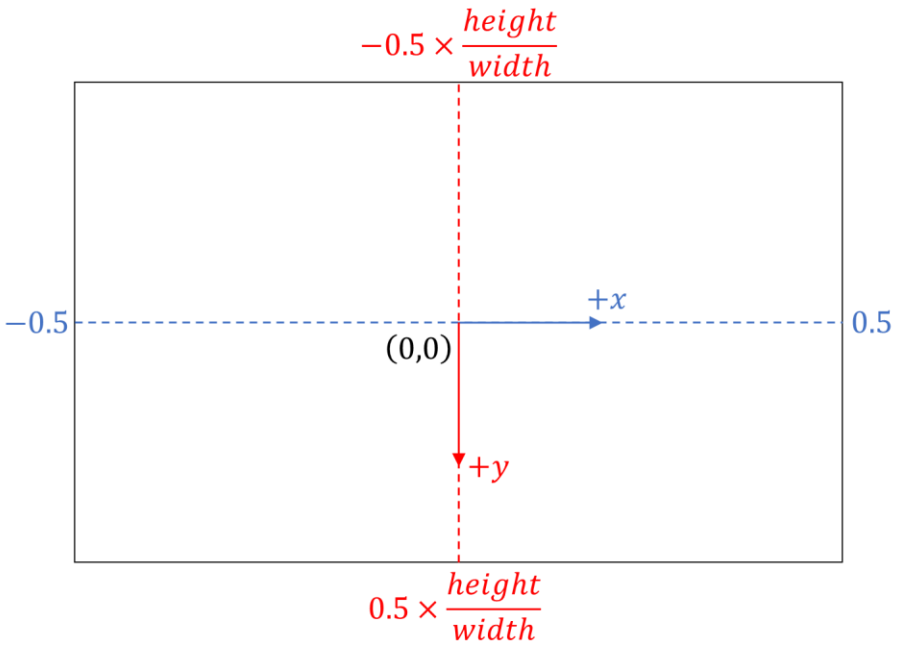


Figure 4 axis system for the EDS position

6.4.6 Transforms

Since converting from one coordinate system to another can be clumsy and error-prone, the PPI libraries offer basic matrix manipulation functionality. There is a type **ppi.Processing.Transformation**,. A Transformation can be used to multiply a point or a rectangle, and they can also be concatenated by multiplying the matrices. You can build your own transformation matrices this way, based on the specific construction helper functions ppi.Processing.Transformation.Scale, ppi.Processing.Transformation.Offset, and ppi.Processing.Transformation.Rotate, as in the example below.

```
# construct a transformation matrix for uniform scaling by factor s
Ts1 = ppi.Processing.Transformation.Scale(s)

# construct a transformation matrix for x and y scaling with different factors
Ts2 = ppi.Processing.Transformation.Scale(sx, sy)

# constuct a "translation matrix"
To = ppi.Processing.Transformation.Offset(dx, dy)

# construct a rotation matrix for counterclockwise rotation by r radians
Tr = ppi.Processing.Transformation.Rotate(r)

# concatenate various transformation matrices
Tc = Tr*Ts2*To
```

There are two helper functions which will construct a transformation matrix to convert image coordinates (with the (0, 0) point at the top left) to the relative scan coordinates as mentioned in the previous section or to so-called "world coordinates" which are physical coordinates (in meters) relative to the center of the sample. The latter needs an Acquisition object because it constructs the transformation matrix based on the physical pixel size and sample position as recorded in the metadata.

Given the transformation matrix, calculating scan positions or physical positions is a matter of multiplying the coordinates by the matrix, as in the example below.

```
# given a 1920x1200 Acquisition acq from a connected Phenom or a file:
Ts = ppi.Application.ToScan(acq.image)
Ts*ppi.Point(0, 0)            # gives (-0.5, -0.3125)
Ts*ppi.Point(1920, 1200)      # gives (0.5, 0.3125)

Tw = ppi.Application.ToWorld(acq)      # note Acquisition, not just the Image
pos = Tw*ppi.Position(1920/2, 1200/2)  # center of the image
# pos = acq.metadata.position, the physical stage position
# of the center of the image

bounds = ppi.RectangleD(0, 0, acq.image.width, acq.image.height)
(Tw*bounds).width()           # gives HFW in meters
```

Using the ToScan() helper function, you can easily find the correct coordinates to position the beam exactly on a given pixel in the image.

6.4.7 Adding jobs to the analyzer

The job analyzer has four types of EDS measurements that can be added:

- Spot, the beam is positioned in a single point and one spectrum is acquired at that location.
- Region, the beam is scanning in an area and one spectrum is acquired of that area.
- Line, the beam is scanning a line. This line is subdivided into points and a spectrum is acquired for each point.
- Map, the beam is scanning in an area. This area is subdivided into points and a spectrum is acquired for each point.

6.4.7.1 Spot

The spot measurement is a point analysis at a specific position in the image. The constructor is:

```
analyzer.AddSpot(position ,maxTime, maxCounts)
```

The position is the ppi position object. Optionally both a maximum time and maximum number of counts can be specified. The add spot method returns a EdsSpotData object with the results of the acquisition.

The EdsSpotData contains the following elements:

- position, the position where the spot measurement was acquired.
- spotSpectrum, the spectrum data of the spot measurement.
- uniqueId, an integer identification number.

6.4.7.2 Region

The region measurement is the same as a spot measurement but instead of on a single point the spectrum is acquired in a rectangular region. The constructor is:

```
analyzer.AddRegion(region ,maxTime, maxCounts)
```

The region is a rectangle that can be constructed like:

```
rect = ppi.RectangleD(left, top, right, bottom)
```

Where the left, top, right and bottom are based on the ratios of the image. These ratios are calculated in the same way as the position. A full screen map of an 16:10 image would be made as follows:

```
rect = ppi.RectangleD(-0.5, -0.3125, 0.5, +0.3125)
```

The maximum time and maximum number of counts are both possible arguments to be passed. The add region method returns a EdsRegionData object with the results of the acquisition.

The EdsRegionData contains the following elements:

- region, a rectangle where the region measurement was acquired.
- sumSpectrum, the spectrum data of the region measurement.
- uniqueId, an integer identification number.

6.4.7.3 Map

The map measurement makes an elemental map of a specified rectangular region. The constructor is:

```
analyzer.AddMap(region, mapSize, timePerPoint)
```

The region is a rectangular area as described in the section of the region measurement. The map size is an integer value with the number of pixels of the elemental map on the longest side. The short side will automatically get the correct number of pixels for that axis. The time per point is the number of seconds that each pixel in the map should in total take (the map is generated by adding up the results of many frames). The add map method returns a EdsMapData object with the results of the acquisition.

The EdsMapData contains the following elements:

- gridSpectrum, the spectrum for each individual pixel in the region.
- region, a rectangle where the region measurement was acquired.
- sumSpectrum, the spectrum data summed over the entire region.
- uniqueId, an integer identification number.

6.4.7.4 Line

The line measurement makes an elemental line of a specified line over the image. The constructor is:

```
analyzer.AddLine(line, numberOfPoints, timePerPoint, numberOfPasses)
```

A line can be constructed by giving the start and end positions of the line:

```
ppi.Line(ppi.Position(xStart, yStart), ppi.Position(xEnd, yEnd))
```

The x and y positions are floats based on the ratio in the image as described in the position section of the manual.

The number of analysis points that needs to be made on the line are passed as an integer. The timePerPoint is the amount of time in seconds that on each pass a spectrum needs to be acquired at each point. The numberOfPasses is how often the line measurement needs to be repeated. The add map method returns a EdsLineData object with the results of the acquisition.

The EdsLineData contains the following elements:

- line, a line where the measurement was acquired.

- lineSpectrum, the spectrum for each individual point in the line.
- sumSpectrum, the spectrum data summed over the entire line.
- uniqueId, an integer identification number.

6.4.8 Identification and quantification

To automatically identify and quantify the elements present in the spectrum the ppi.Spectroscopy.Quantify method is used. In this method the spectrum is passed. The quantify method calculates the mass distribution of the elements present in the spectrum.

```
quantification = ppi.Spectroscopy.Quantify(spectrum)
```

The result of the quantification is the QuantificationResult. In the QuantificationResult the composition is stored. The composition is a list of both the element and the weight concentration fraction.

For example, the weight concentrations of each element in the quantification results can be printed using the following code:

```
for element in quantification.composition.constituents:
    print(f'{element.Z} {element.weightFraction*100:.2f}%')
```

Using quantify without any elements added to it implicitly makes quantify use the automatic element identification algorithm. Alternatively, the elements that need to be identified can also be passed into the quantification algorithm:

```
quantification = ppi.Spectroscopy.Quantify(spectrum, elements)
```

The elements are a list of ppi.Spectroscopy.Element for each element that needs to be quantified.

The automatic element identification algorithm can also be called separately:

```
ID = ppi.Spectroscopy.Identify(spectrum, excludedElements)
```

The excludedElements is an optional list with elements that should be excluded from the identification results. The result of identify is a list with elements that are identified to be part of the composition. If desired, elements can be added or removed from this list. These elements can be passed into the quantify call.

6.4.9 Plotting

The PyPhenom spectroscopy library has built in functionality to plot spectrums and the results of the line and map measurements.

6.4.9.1 Plot spectrum

To make a plot of a spectrum with the identified and quantified peaks the ppi.Spectroscopy.DrawSpectrum method is used:

```
image = ppi.Spectroscopy.DrawSpectrum(spectrum, quantification, energyRange)
```

The energy range is an optional named variable that gives the range of the energy that should be plotted (e.g. from 0 to 10 keV). If a range of 0,0 is given the spectrum auto

sizes, this is the default option. The result is an image object, in figure 5 an example of such an image is shown.

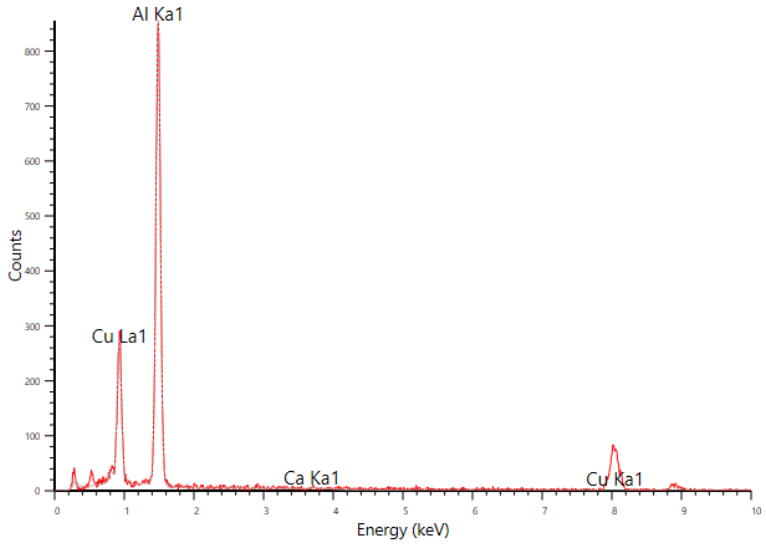


Figure 5 Example of spectrum plot from 0 to 10 KeV

The image can be save using the default ppi.Save method:

```
ppi.Save(image, "Spectrum.bmp")
```

6.4.9.2 Plot line

To plot a line spectrum the ppi.Spectroscopy.DrawLineSpectrum method is used:

```
image = ppi.Spectroscopy.DrawLineSpectrum(acquisition, lineSpectrum, imageSize)
ppi.Save(image, "LineSpectrum.bmp")
```

The acquisition object with the relevant metadata and the line spectrum needs to be supplied. Optionally the size of the created images can be specified using a named variable.

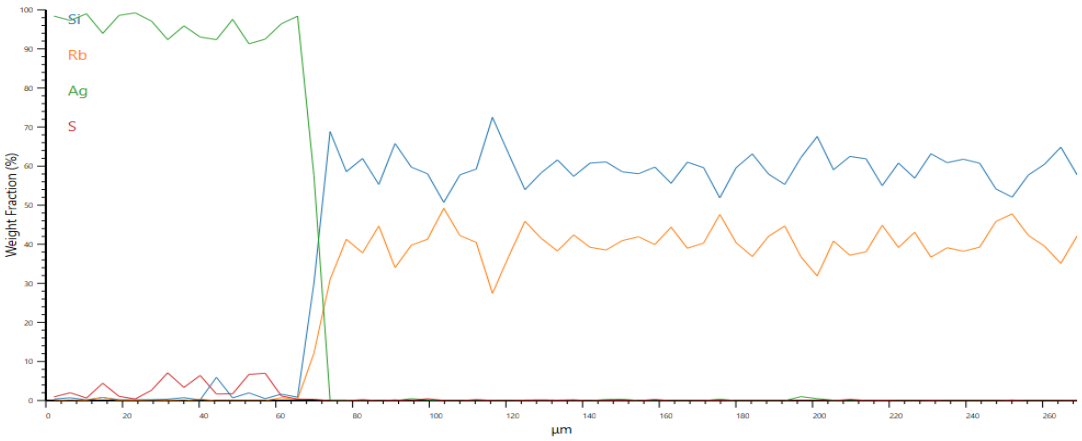


Figure 6 Line scan results of solar cell

6.4.9.3 Plot map

Before drawing the map each pixel in the map needs to be quantified. To do this a QuantMap object needs to be made. To create the quantMap and quantify the gird spectrum data:

```
quantMap = ppi.Spectroscopy.QuantMap(mapData)
quantMap.Quantify()
```

To plot a map overlay of a map the ppi.Spectroscopy.DrawElementMapOverlay is used:

```
map = ppi.Spectroscopy.DrawElementMapOverlay(acquisition, quantMap, region,
colors, alpha)
ppi.Save(map, "ElementMap.bmp")
```

The image of the acquisition object is used to draw the map onto. The quantMap contains the mapping and quantification data. The region is a rectangle corresponding to the location of the map in the image. The colors are a dictionary of the constituents of the identification with the corresponding colors. Optionally the opacity of the overlay can be set using the alpha.

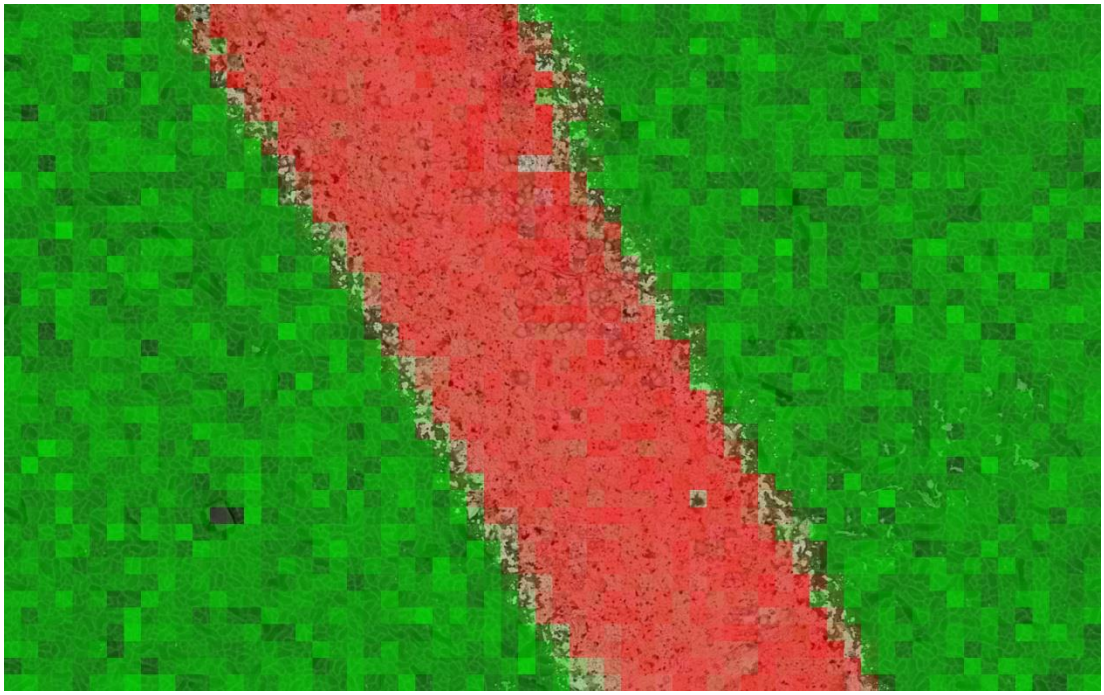


Figure 7 Mapping result of a solar cell. Green is silica and red is silver

6.4.10 Saving and loading

The acquired data can be stored in the EID data structure, which can be loaded into the UI for further analysis. In the EID project structure, measurements are added to an image. Each image can contain multiple measurements. Each measurement contains the measured data and the processing options. The simplified structure of an EID project can look like this:

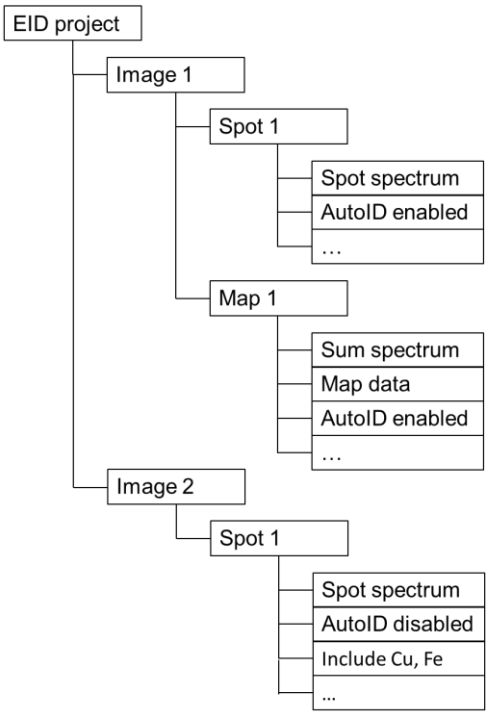


Figure 8 Example of EID project structure

As can be seen the highest layer are the images. In the images are the measurement data and the processing settings that are associated to those measurements.

To create an EID project the ppi.Spectroscopy.EdsProject() is used. In this EID project images can be added and to each image EDS measurements can be attached.

```
project = ppi.Spectroscopy.EdsProject()
```

To add an image to the EID project an image needs to be converted to and EDSImage using ppi.Spectroscopy.EdsImage(). In this method the Phenom image and the image name are passed. To add it to the project the project.AddImage() command is used.

```
semImage = phenom.SemAcquireImage(1920, 1200)
image = ppi.Spectroscopy.EdsImage(semImage, 'Image1')

project = ppi.Spectroscopy.EdsProject()
project.AddImage(image)
```

Using the analyzer, measurements can be made, and the results can be added to the project:

```
spotData = analyzer.AddSpot(ppi.Position(0, 0), maxTime=9.5, maxCounts=10000)
lineData = analyzer.AddLine(ppi.Line(ppi.Position(0, 0), ppi.Position(0, 0.5)),
numberOfPoints=64)
mapData = analyzer.AddMap(ppi.RectangleD(-0.5, -0.3125, 0.5, +0.3125), mapSize=64,
timePerPoint=0.05)

analyzer.Wait()

image.AddAnalysis(spotData)
image.AddAnalysis(lineData)
image.AddAnalysis(mapData)
```

This project can be saved as in the elid format that is compatible with UI:

```
ppi.Spectroscopy.SaveEdsProject(project, example.elid')
```

6.4.11 Release exclusive access and reset viewing mode

While working with EDS it can happen that the code fails while doing a measurement. This can leave the Phenom in a locked and strange state. To unlock the Phenom the exclusive access needs to be release. This can be done in the UI or from code:

```
phenom.ReleaseExclusiveAccess()
```

It can also happen that the viewing mode is still in spot mode. In this case the scale of the scanParams is set to zero to create the smallest possible spot and the image size is set to a small rectangle with a low resolution. The scale and resolution can be set back to reasonable values with the following code:

```
vm = phenom.GetSemViewingMode()
vm.scanParams.scale = 1.0
vm.scanParams.size = ppi.Size(960,600)
phenom.SetSemViewingMode(vm)
```

6.5 Compatibility with NumPy and OpenCV

It is possible to convert a Phenom image to a NumPy array and to convert a NumPy array to a Phenom image. Using these NumPy arrays, image processing operations can be done using external libraries like OpenCV.

NumPy is an open-source package for scientific computing and array manipulation. More information on NumPy and download instructions can be found on:

http://www.numpy.org/

OpenCV is an open-source library that is widely used for image processing. More information on OpenCV and download instructions can be found on:

https://opencv.org/

In these tutorials, the following versions of NumPy and OpenCV are used:

- NumPy: 1.11
- OpenCV: 3.1

To be able to use these packages, they need to be imported into Python:

```
import PyPhenom as ppi
import numpy as np
import cv2
```

First, an image is acquired, in a different but equivalent way to section 6.3:

```
phenom = ppi.Phenom()
acq = phenom.SemAcquireImage(1920, 1200, 16, ppi.DetectorMode.All, False, 1.0)
```

To convert the Phenom acquisition object (*acq* in the example) to a NumPy array, the image is extracted from the acquisition object (*acq.image* in the example) and then the image is converted to an NumPy array using the np.array function:

```
numpyArray = np.array(acq.image)
```

The NumPy array can be used in OpenCV to do image processing. For example, it is possibly to perform a canny edge detection on the image, that was converted into a NumPy array. For that, the cv2.Canny function is used:

```
cannyEdge = cv2.Canny(numpyArray, 32, 176)
```

Where the values passed to the function are the NumPy array and the first and second threshold for the hysteresis procedure in the canny edge detection filter.

The image containing the edge detection can be put back into the acquisition object and this can then be saved including all the metadata that was already available in the acquisition object:

```
acq.image = cannyEdge
ppi.Save(acq, 'cannyEdge.tiff')
```

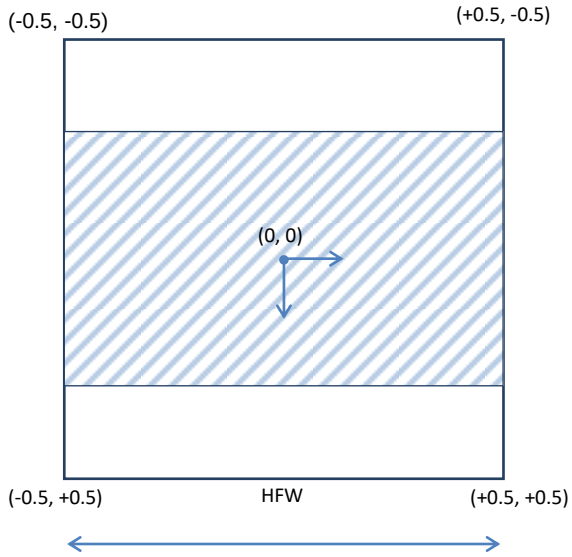
Both NumPy and OpenCV are not developed, maintained or supported by Thermo Fisher Scientific. These examples are merely meant to show that and how the conversion between these packages is possible.

6.6 Patterning

All Phenoms using a ZYNQ-based scan engine (Phenom as of G6, Phenom XL as of G2, and Phenom Pharos as of G2) support *patterning,* and this is available from the Phenom Programming Interface. "Patterning" in this case refers to the scanning of arbitrary shapes defined as lists of (*x, y*) positions at which remains stationary for a given dwell time (also given per position). One could view the "normal" imaging mode as a subset of this patterning mode, where the positions describe a rectangle with a fixed dwell time per pixel (and this is, in fact, exactly how it is implemented). This section describes the PPI interfaces available for patterning, as well as some basic terminology.

6.6.1 Patterning Positions

The patterning API uses a coordinate system relative to the square enclosing the SEM horizontal field width. The Phenom UI shows a rectangular image where the width is defined by the horizontal field width (HFW) value and the height is defined by the ratio image height / image width multiplied by HFW. The picture below shows the square with the full scan area from (-0.5, -0.5) to (+0.5, +0.5) and highlighted rectangle shows how the Phenom GUI image would be positioned inside the scan area.



This is the same coordinate system as used for beam positioning in the EDS use case, as shown in figure 4 in paragraph 6.4.5, and you can use the ToScan() transformation to find the corresponding pixel positions in case you want to construct them based on an existing image.

6.6.2 Pattern Timing

The Phenom scans a pattern as a sequence of *x* and *y* positions where the (*x, y*) scan signals remain stationary for an amount of time defined by the dwell time value. The per-position dwell time on Phenom is programmable in multiples of 25 ns up to 3.35 s. The scan system needs some time to move from one position to the next. For best results, make sure the patterns have about a 1000 or more positions per HFW unit length, and the time to scan from one side of the scan area to the opposite side (number of positions multiplied by dwell time) takes about 0.1 ms or longer. If the pattern tries to move the beam too fast, the resulting beam movement will become distorted and generally scan a smaller area than planned. Good initial values to use for pattern properties are 10^{-3} for pitch and 10^{-6} (1µs) for dwellTime.

6.6.3 Scanning the Pattern

There are several predefined types to define patterns, namely LineScanPattern, PointScanPattern, RectangleScanPattern, BitmapScanPattern, and AggregateScanPattern. This last type builds a new pattern from a sequence of individual scan patterns. For detailed information regarding the individual pattern types, see the reference documentation in Appendix B.55 of this document.

Once the pattern object is defined, the Render() function will create a ScanDefintion object which can be passed to phenom.SetSemScanDefinition() to send it to the Phenom.

Use phenom.SetViewingMode() with a scanMode value of ppi.ScanMode.Pattern to activate the pattern scan.

Now the pattern will be scanned until phenom.SetViewingMode() is called with a scanMode value of ppi.ScanMode.Imaging. This example shows how to scan a 20 degrees rotated rectangle pattern for 5 seconds:

```
import sys
import math
import time
import PyPhenom as ppi

phenom = ppi.Phenom(sys.argv[1])

rectangle = ppi.Patterning.RectangleScanPattern()
rectangle.center = ppi.Position(-0.2, -0.05)
rectangle.size = ppi.SizeD(0.15, 0.05)
rectangle.pitchX = 1e-3
rectangle.pitchY = 1e-3
rectangle.rotation = math.radians(20)
rectangle.dwellTime = 1e-6
rectangle.lineScanStyle = ppi.Patterning.LineScanStyle.Serpentine

phenom.SetSemScanDefinition(rectangle.Render())

vm = phenom.GetSemViewingMode()
vm.scanMode = ppi.ScanMode.Pattern
phenom.SetSemViewingMode(vm)

time.sleep(5)

vm.scanMode = ppi.ScanMode.Imaging
phenom.SetSemViewingMode(vm)
```

6.7 Plotting with Matplotlib

It is often convenient to be able to show images during an automated process or to show the results of image processing. Matplotlib is a widely used library that supports this. More information and download instructions can be found here:

https://matplotlib.org/

To import the plot functionality of Matplotlib and PyPhenom into Python:

```
import PyPhenom as ppi
from matplotlib import pyplot as plt
```

Then, an image is acquired:

```
phenom = ppi.Phenom()
acq = phenom.SemAcquireImage(1024, 1024, 16, ppi.DetectorMode.All, False, 1.0)
```

To create a Matplotlib figure window the function plt.figure() is used:

```
fig = plt.figure()
```

To plot the image, with a gray-scale colormap plt.imshow is used:

```
plot = plt.imshow(acq.image, cmap='gray')
```

To show the image on the screen:

```
plt.show()
```

If this function is called, the script will stop until all images are closed again. Therefore, some caution is advised as to where in the script the show function is called.

Relevant information about the acquisition can be added to the plot. For example, the axes can be set into the physical dimension of the image. To do this the width of the image has to be determined. This is done by extracting the physical pixel size from the metadata of the acquisition object and multiplying that with the pixel count of the image:

```
width = acq.metadata.pixelSize.width * acq.image.width
height = acq.metadata.pixelSize.height * acq.image.height
plot = plt.imshow(acq.image, cmap='gray', extent=[0,width,0,height])
```

Setting the title and axes labels in Matplotlib:

```
plt.title('Phenom acquisition image')
plt.xlabel('x-coordinate')
plt.ylabel('y-coordinate')
```

A color bar can also be added to the plot:

```
fig.colorbar(plot)
```

6.8 Generate PDF documents with PyPhenom

PPI can generate reports using its' own internal PDF-reporting tool. It is based on libHaru (see more: http://libharu.org/). To create a PDF a number of steps must be followed. First a document must be made. In this document a page is created and the content is placed on each page. To create a document the following functions are used:

```
import PyPhenom as ppi
import subprocess
import random

pdf = ppi.Pdf.Document()
pdf.SetPageMode(ppi.Pdf.PageMode.UseOutline)
```

In the document we have to create a page this is done by:

```
page = ppi.Pdf.Document.AddPage(pdf)
page.SetSize(ppi.Pdf.PageSize.A4)
height,width = page.GetHeight(),page.GetWidth()
```

All positioning of text and objects is done in reference with the bottom left corner of the page. The positions are given in points and the default resolution is 72 dpi, thus the default size for A4 is: 595x842 pixels.

A wide variety of content can be placed in a report. As an example, headings, a large section of text, tables, and figures will be explained in this section.

First adding text is explained. In the code snippet below the following types of text are added to the document: a big and bold header, a smaller italic header and a section with a large string that runs over multiple lines.

```
page.BeginText()
page.SetFontAndSize(pdf.GetFont('Times-Bold'), 14)

page.MoveTextPos(50, height - 58)
page.ShowText('1. Example of an PDF-report generated with PPI')


page.MoveTextPos(0,-18)
page.SetFontAndSize(pdf.GetFont('Times-BoldItalic'), 12)
page.ShowText('1.1 Example of a long text')

longText = 'This report shows the powers of PPI and creating reports in Python. In
this first section we show a long text that automatically is shown over multiple
lines. In the second section we show a table filled with random data. The report
is concluded by an image of a simulated image.'

page.SetFontAndSize(pdf.GetFont('Times-Roman'), 12)
page.TextRect(ppi.RectangleD(50,height-84,width-50, 50),
longText,ppi.Pdf.TextAlignment.Justify)
```

To create text the page.BeginText() method is called, after that the font is set with page.SetFontAndSize(), then the text is positioned with respect to the top left of the document with a margin of 50 pixels (about 2 cm in the document) with page.MoveTextPos().

To insert text page.ShowText() is used. To move to the next line we only have to set the relative movement over the page with page.MoveTextPos(). The first time you call page.MoveTextPos() the starting point is the bottom left corner of the document, and the after that it is a relative change to the new position.

The long text is added using the page.TextRect() this method takes care of line breaks and alignments of the text. In this call the following attributes are passed: a PPI rectangle giving the absolute position of the text box, the string with the text that should be printed in the report, and the alignment.

To make a table normal text is used but is displayed in a structured manner. First a header is made using text spaced in a regular horizontal pattern. Below this header we want a single line. But drawing is not allowed between page.BeginText() and page.EndText() parts, so we have to close the text part and draw and then open it again. To draw the line we move the position to the start location and use the page.LineTo() to

define the line. The real drawing is done by page.Stroke(). After that we iterate over the items we want to put in the table and put them all in the right column, with the same spacing. The table ends with a double underlining. The table is filled with random values and formatted with different significance. The code is:

```
page.MoveTextPos(0,-32)
page.SetFontAndSize(pdf.GetFont('Times-BoldItalic'), 12)
page.ShowText('1.2 Example of a table')

page.MoveTextPos(0, -16)
page.SetFontAndSize(pdf.GetFont('Times-Roman'), 12)
page.ShowText('Row #')
page.MoveTextPos(64, 0)
page.ShowText('Value 1')
page.MoveTextPos(64, 0)
page.ShowText('Value 2')
page.MoveTextPos(64, 0)
page.ShowText('Value 3')
page.MoveTextPos(64, 0)
page.ShowText('Value 4')

xTableStart = page.GetCurrentTextPos().x
yTableStart = page.GetCurrentTextPos().y

page.EndText()

page.MoveTo(50,yTableStart-4)
page.LineTo(350,yTableStart-4)
page.Stroke()

page.BeginText()
page.MoveTextPos(xTableStart-32,yTableStart)
for row in range(5):
    page.MoveTextPos(-256, -16)
    page.ShowText(str(row+1))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.3f}'.format(random.random()*1)))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.1f}'.format(random.random()*2)))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.1f}'.format(random.random()*4)))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.0f}'.format(random.random()*8)))

yTableStop = page.GetCurrentTextPos().y

page.EndText()
page.MoveTo(50,yTableStop-4)
page.LineTo(350,yTableStop-4)
page.Stroke()
page.MoveTo(50,yTableStop-6)
page.LineTo(350,yTableStop-6)
page.Stroke()
```

Drawing a figure goes in much the same way as drawing lines. It also has to happen outside the text part and the position has to be filled in absolutely. To demonstrate drawing an image a PPI simulator is used to create a simulation acquisition. To draw the image in the report the page.DrawImage() method is used. The data that goes into this method is the image, the position, height, and width of figure to be draw. In this example a simulator image of the Phenom is loaded:

```
page.BeginText()
page.MoveTextPos(50,yTableStop-32)
page.SetFontAndSize(pdf.GetFont('Times-BoldItalic'), 12)
page.ShowText('1.3 Example of a figure')

yPosFigure = page.GetCurrentTextPos().y

page.MoveTextPos(0,-(width-100)/2-16)
page.SetFontAndSize(pdf.GetFont('Times-Italic'), 12)
page.ShowText('Figure 1. PPI simulator image')

page.EndText()

phenom = ppi.Phenom('','','')
acq = phenom.SemAcquireImage(1024, 1024, 16)
page.DrawImage(acq.image, 50, yPosFigure-(width-100)/2-4, (width-100)/2,(width-
100)/2)
```

To save the pdf pdf.SaveToFile() is used. To open the pdf from a script the subprocess library can be used:

```
pdf.SaveToFile('Example report.pdf')
subprocess.Popen('Example report.pdf', shell=True)
```

The resulting PDF is displayed on the next page.

1. Example of an PDF-report generated with PPI

1.1 Example of a long text

This report shows the powers of PPI and creating reports in Python. In this first section we show a long text that automatically is shown over multiple lines. In the second section we show a table filled with random data. The report is concluded by an image of a simulated image.

1.2 Example of a table

Row #	Value 1	Value 2	Value 3	Value 4
1	0.182	1.2	3.6	3
2	0.169	1.6	1.2	6
3	0.232	0.4	0.1	2
4	0.001	1.2	0.2	5
5	0.048	1.2	0.2	5

1.3 Example of a figure

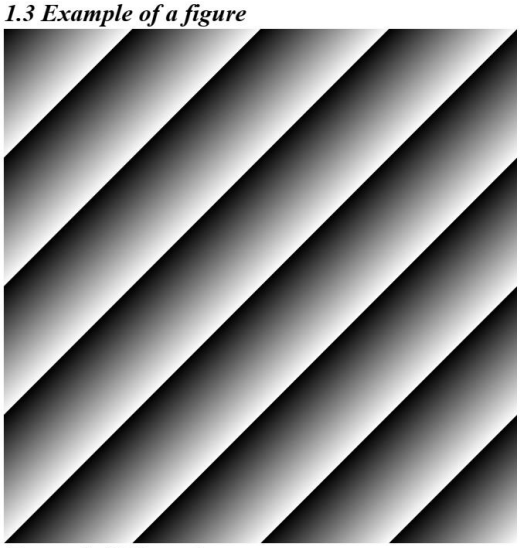


Figure 1. PPI simulator image

6.9 Editing the data bar

The data bar can be edited. The following features can be changed:

- The label
- The font and background color
- The items displayed in the data bar

In the example below is shown how this can be done. First a phenom acquisition object is made. Then the image of the acquisition is transformed to an RGB image to support colored data bars. To edit the data bar the following object is made: ppi.DatabarRenderer(). In this object the data bar can be changed as shown in the example. To add the data bar to the image the following call is made: renderer.AddDatabar(acq). The returning object is an acquisition object with the desired data bar.

```
import PyPhenom as ppi

phenom = ppi.Phenom('','','')
acq = phenom.SemAcquireImage(1920, 1200, 16)

acq.fullImage = ppi.Processing.MakeRGB(acq.image)
acq.metadata.dataBarLabel = 'Label'

renderer = ppi.DatabarRenderer()

renderer.bgColor = ppi.Colors.Black
renderer.textColor = ppi.Colors.White
renderer.fontSize = 12
renderer.showLabel = True
renderer.showScaleBar = True
renderer.showMagnification = True
renderer.magnificationStyle = ppi.MagnificationStyle.Phenom
renderer.showFieldSize = True
renderer.showHighVoltage = True
renderer.showSpotSize= True
renderer.showWorkingDistance = False
renderer.showDetector = False
renderer.showDate = False
renderer.showTime= True
renderer.showSamplePressureEstimate= True

ppi.Save(renderer.AddDatabar(acq), 'Databar.tiff')
```

6.10 Generate documentation with PyDoc

In Python it is possible to generate documentation and to view the online help system of a Python package. This can be done using PyDoc.

To view the documentation of PyPhenom, open the directory where Python is installed using a command prompt (cmd in windows). Typically, the directory is C:\Python3X, where the X is the version number of the python installation. To view the documentation in the command prompt, run the following command in the command prompt:

```
python.exe -m pydoc PyPhenom
```

A HTML webpage containing the documentation can also be generated using PyDoc. This format is more accessible and typically more convenient to use. To generate the webpage run from the same directory this command:

```
python.exe -m pydoc -w PyPhenom
```

In the same directory, a HTML webpage is created. This document contains all the information and the documentation of PyPhenom. It is possible to move this webpage to another directory.

6. Support

For further support or any question about PyPhenom, Thermo Fisher Scientific can be contacted at: application.phenom@thermofisher.com

7. Further reading

To learn more about Python, visit the website: https://www.python.org/

The script in Scripts/PhenomImageDisplay.pyw in PyPhenom.zip uses the PySide module to acquire Phenom images and show them in a GUI application. Visit http://wiki.qt.io/PySide for more information on PySide.

Appendix A. Tutorial code

A.1 Find and connect to your Phenom

```
import PyPhenom as ppi

for phenom in ppi.FindPhenoms(1):
    print('Phenom ip: ', phenom.ip)
    print('Phenom id: ', phenom.name)

PhenomID = 'MVE012345678'
username = 'Username provided by ThermoFisher Scientific'
password = 'Password provided by ThermoFisher Scientific'

phenom = ppi.Phenom(PhenomID, username, password)
```

A.2 Basic Phenom commands & image Acquisition

```
import PyPhenom as ppi
import time

PhenomID = 'MVE012345678'
username = 'Username provided by ThermoFisher Scientific'
password = 'Password provided by ThermoFisher Scientific'

phenom = ppi.Phenom(PhenomID, username, password)

phenom.Load()

phenom.MoveToNavCam()

acqCamParams = ppi.CamParams()
acqCamParams.size = ppi.Size(912, 912)
acqCamParams.nFrames = 1

acqNavCam = phenom.NavCamAcquireImage(acqCamParams)
ppi.Save(acqNavCam, 'NavCam.tiff')

phenom.MoveToSem()
time.sleep(7)

phenom.MoveTo(0, 0)

phenom.SetSemHighTension(-5000)

phenom.SetSemSpotSize(3.3)

phenom.SemAutoFocus()

phenom.SetHFW(ppi.MagnificationToFieldWidth(5000))

phenom.SemAutoContrastBrightness()

viewingMode = phenom.GetSemViewingMode()
viewingMode.scanParams.detector = ppi.DetectorMode.All
phenom.SetSemViewingMode(viewingMode)

acqScanParams = ppi.ScanParams()
acqScanParams.size = ppi.Size(1920, 1200)
acqScanParams.detector = ppi.DetectorMode.All
acqScanParams.nFrames = 16
acqScanParams.hdr = False
acqScanParams.scale = 1.0

acq = phenom.SemAcquireImage(acqScanParams)
acq.metadata.displayWidth = 0.5
acq.metadata.dataBarLabel = "Label"

acqWithDatabar = ppi.AddDatabar(acq)

ppi.Save(acq, 'example.tiff')
ppi.Save(acqWithDatabar, 'exampleWithDatabar.tiff')

phenom.Unload()
```

A.3 EDS measurements and saving

```
import PyPhenom as ppi

PhenomID = 'MVE012345678'
username = 'Username provided by ThermoFisher Scientific'
password = 'Password provided by ThermoFisher Scientific'

phenom = ppi.Phenom(PhenomID, username, password)

print("Has EDX license: " + str(ppi.HasLicense(ppi.Features.Spectroscopy)))
print("Has mapping license: " + str(ppi.HasLicense(ppi.Features.EdsMap)))

analyzer = ppi.Application.ElementIdentification.EdsJobAnalyzer(phenom)
settings = phenom.LoadPulseProcessorSettings()
analyzer.preset = settings.spot
analyzer.AcquireDriftCorrectionReference()

analyzer.EdsAnalyzerFailure.connect(lambda msg:
print('EdsAnalyzerFailure("{}")'.format(msg)))

spotData = analyzer.AddSpot(ppi.Position(0, 0), maxTime=9.5, maxCounts=10000)
lineData = analyzer.AddLine(ppi.Line(ppi.Position(0, 0), ppi.Position(0.5, 0)),
numberOfPoints=64)
mapData = analyzer.AddMap(ppi.RectangleD(-0.5, -0.3125, 0.5, +0.3125), mapSize=64,
timePerPoint=0.05)

analyzer.Wait()

print(ppi.Spectroscopy.Quantify(spotData.spotSpectrum))

ppi.Spectroscopy.WriteMsaFile(spotData.spotSpectrum, 'SpotSpectrum.msa')

semImage = phenom.SemAcquireImage(1920, 1200)
image = ppi.Spectroscopy.EdsImage(semImage, 'Image1')

image.AddAnalysis(spotData)
image.AddAnalysis(lineData)
image.AddAnalysis(mapData)

project = ppi.Spectroscopy.EdsProject()
project.AddImage(image)

ppi.Spectroscopy.SaveEdsProject(project, 'Spectrum.elid')
ppi.Spectroscopy.SaveEdsProject(project, 'Spectrum.phen')
```

A.4 EDS loading and plotting

```
import PyPhenom as ppi

project = ppi.Spectroscopy.LoadEdsProject('Spectrum.elid')

spectrum = project[0][0].spectrum
image = ppi.Spectroscopy.DrawSpectrum(spectrum,
ppi.Spectroscopy.Quantify(spectrum), energyRange=ppi.Range(0, 10e3))
ppi.Save(image, 'Spectrum.bmp')

image = ppi.Spectroscopy.DrawLineSpectrum(project[0].acquisition, project[0][1],
size=ppi.Size(1000, 500))
ppi.Save(image, "LineSpectrum.bmp")

quantMap = ppi.Spectroscopy.QuantMap(project[0][2])
quantMap.Quantify()
ID = ppi.Spectroscopy.Identify(project[0][2].spectrum)
elementMap = ppi.Spectroscopy.DrawElementMapOverlay(project[0].acquisition,
                             quantMap, project[0][2].area,
                            {ID.constituents[0].element: ppi.Rgb24(255, 0, 0),
                             ID.constituents[1].element: ppi.Rgb24(0, 255, 0),
                             ID.constituents[2].element: ppi.Rgb24(0, 0, 255)},
                             alpha = 0.75)

ppi.Save(elementMap, "ElementMap.bmp")
```

This example will load the project saved in section A.3

A.5 Compatibility with NumPy and OpenCV

```
import PyPhenom as ppi
import numpy as np
import cv2

PhenomID = 'MVE012345678'
username = 'Username provided by ThermoFisher Scientific'
password = 'Password provided by ThermoFisher Scientific'

phenom = ppi.Phenom(PhenomID, username, password)

acq = phenom.SemAcquireImage(1920, 1200, 16, ppi.DetectorMode.All, False, 1.0)
acq.image = cv2.Canny(np.array(acq.image), 32, 176)
ppi.Save(acq, 'cannyEdge.tiff')
```

A.6 Plotting with Matplotlib

```
import PyPhenom as ppi
from matplotlib import pyplot as plt

PhenomID = 'MVE012345678'
username = 'Username provided by ThermoFisher Scientific'
password = 'Password provided by ThermoFisher Scientific'

phenom = ppi.Phenom(PhenomID, username, password)

acq = phenom.SemAcquireImage(1920 ,1200, 16, ppi.DetectorMode.All, False, 1.0)

fig = plt.figure()

width = acq.metadata.pixelSize.width * acq.image.width
height = acq.metadata.pixelSize.height * acq.image.height

plot = plt.imshow(acq.image, cmap='gray', extent=[0,width,0,height])

plt.title('Phenom Acquisition image')
plt.xlabel('x')
plt.ylabel('y')

fig.colorbar(plot)

plt.show()
```

A.7 Generate PDF documents with PyPhenom

```
import PyPhenom as ppi
import subprocess
import random

pdf = ppi.Pdf.Document()
pdf.SetPageMode(ppi.Pdf.PageMode.UseOutline)

page = ppi.Pdf.Document.AddPage(pdf)
page.SetSize(ppi.Pdf.PageSize.A4)
height,width = page.GetHeight(),page.GetWidth()

page.BeginText()
page.SetFontAndSize(pdf.GetFont('Times-Bold'), 14)

page.MoveTextPos(50, height - 58)
page.ShowText('1. Example of an PDF-report generated with PPI')


page.MoveTextPos(0,-18)
page.SetFontAndSize(pdf.GetFont('Times-BoldItalic'), 12)
page.ShowText('1.1 Example of a long text')

longText = 'This report shows the powers of PPI and creating reports in Python. In
this first section we show a long text that automatically is shown over multiple
lines. In the second section we show a table filled with random data. The report
is concluded by an image of a simulated image.'

page.SetFontAndSize(pdf.GetFont('Times-Roman'), 12)
page.TextRect(ppi.RectangleD(50,height-84,width-
50,50),longText,ppi.Pdf.TextAlignment.Justify)


page.MoveTextPos(0,-32)
page.SetFontAndSize(pdf.GetFont('Times-BoldItalic'), 12)
page.ShowText('1.2 Example of a table')

page.MoveTextPos(0, -16)
page.SetFontAndSize(pdf.GetFont('Times-Roman'), 12)
page.ShowText('Row #')
page.MoveTextPos(64, 0)
page.ShowText('Value 1')
page.MoveTextPos(64, 0)
page.ShowText('Value 2')
page.MoveTextPos(64, 0)
page.ShowText('Value 3')
page.MoveTextPos(64, 0)
page.ShowText('Value 4')

xTableStart = page.GetCurrentTextPos().x
yTableStart = page.GetCurrentTextPos().y

page.EndText()

page.MoveTo(50,yTableStart-4)
page.LineTo(350,yTableStart-4)
page.Stroke()
```

```
page.BeginText()
page.MoveTextPos(xTableStart-32,yTableStart)
for row in range(5):
    page.MoveTextPos(-256, -16)
    page.ShowText(str(row+1))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.3f}'.format(random.random()*1)))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.1f}'.format(random.random()*2)))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.1f}'.format(random.random()*4)))
    page.MoveTextPos(64, 0)
    page.ShowText(str('{:1.0f}'.format(random.random()*8)))

yTableStop = page.GetCurrentTextPos().y

page.EndText()
page.MoveTo(50,yTableStop-4)
page.LineTo(350,yTableStop-4)
page.Stroke()
page.MoveTo(50,yTableStop-6)
page.LineTo(350,yTableStop-6)
page.Stroke()

page.BeginText()
page.MoveTextPos(50,yTableStop-32)
page.SetFontAndSize(pdf.GetFont('Times-BoldItalic'), 12)
page.ShowText('1.3 Example of a figure')

yPosFigure = page.GetCurrentTextPos().y

page.MoveTextPos(0,-(width-100)/2-16)
page.SetFontAndSize(pdf.GetFont('Times-Italic'), 12)
page.ShowText('Figure 1. PPI simulator image')

page.EndText()

phenom = ppi.Phenom('','','')
acq = phenom.SemAcquireImage(1024, 1024, 16)
page.DrawImage(acq.image, 50, yPosFigure-(width-100)/2-4, (width-100)/2,(width-
100)/2)

pdf.SaveToFile('Example report manual.pdf')
subprocess.Popen('Example report manual.pdf', shell=True)
```

Appendix B Reference guide

B.1 Error Class

Exception type for Phenom specific errors.

message	Error description text.
code	Numeric error identification

B.2 InstrumentMode Enum

The InstrumentMode enum describes the various modes of the instrument.

Value	Description
Off	In off state (but still running)
Operational	Operational
Standby	Standby (takes approx. 5 minutes to become operational)
Hibernate	In hibernation mode (takes approx. 6 minutes to become operational)
Initializing	Initializing
ClosingDown	Going to Off state
Error	The instrument is in error

B.3 OperationalMode Enum

The InstrumentMode enum describes the active mode of operation of the instrument.

Value	Description
Unavailable	The instrument is unavailable
LoadPos	Sample is in loading position
Unloading	Sample is moving to unload position
SelectingNavCam	Sample is moving to NavCam
SelectingSem	Sample is moving to SEM
LiveNavCam	Live viewing mode under the NavCam
LiveSem	Live viewing mode under the SEM
AcquireNavCamImage	Performing NavCam acquisition
AcquireSemImage	Performing SEM acquisition

B.4 ImagingDevice Enum

The ImagingDevice enum describes the active device for image acquisition.

Value	Description
NavCam	The optical microscope “NavCam”
Sem	The scanning electron microscope “SEM”

B.5 NavigationAlgorithm Enum

The NavigationAlgorithm enum describes the algorithm used for stage movements during sample navigation.

The Phenom stage is accurate to about 35 µm (Phenom ProX) in point-to-point movements but can achieve considerably better accuracy if a so-called “backlash correction” is applied. With this correction, the stage is programmed to approach the destination position always from the same direction.

Backlash correction only increases point-to-point accuracy when the previous stage movements also used the correction. Immediately after a single direct movement or a stage jogging movement, the accuracy of the next backlash-corrected stage movement will not be better.

Value	Description
Auto	Let the Phenom decide which algorithm to use
BacklashOnly	Always perform backlash correction
Raw	Use single stage moves only

B.6 StageNavigationMode Enum

The StageNavigationMode enum describes the current status of the stage. The stage will detect whether it is still correctly homed (“calibrated”) and will perform a homing procedure when needed. The sound you hear when starting up the Phenom comes from the homing procedure.

Value	Description
NotAvailable	Stage currently not available
Rest	Stage is not moving
PointToPoint	Stage is performing a point-to-point move
Jog	Stage is performing a jogging move
NotHomed	Stage is not homed

B.7 AcquisitionType Enum

The AcquisitionType enum describes the device used for acquisition.

Value	Description
NavCam	NavCam acquisition
Sem	SEM acquisition

B.8 PixelType Enum

Possible values for image data types.

Value	Description
Unsigned8	Unsigned, 8 bit samples (0 .. 255)
Unsigned16	Unsigned, 16 bit samples (0 .. 65535)
Float32	Floating point, 32 bit samples
RGB	Color (NavCam) samples, 8 bits per channel, as returned by NavCamAcquireImage. Stored in memory as red, green, blue bytes.
BGR	Color (NavCam) samples, 8 bits per channel, as returned by GetLiveImage. Stored in memory as blue, green, red bytes.
RGBA	Color with alpha channel. Stored in memory as red, green, blue, alpha bytes.
BGRA	Color with alpha channel. Stored in memory as blue, green, red, alpha bytes.

B.9 AutoFocusAlgorithm Enum

Value	Description
Auto	Automatically decide to use coarse and fine routines to focus
CoarseFine	Use both the coarse and fine routines to find focus
FineOnly	Only use the fine focus

B.10 ScanMode Enum

The ScanMode enum describes the possible values for the SEM scanning mode.

Value	Description
Blank	Blank the beam (to minimize sample damage)
Imaging	Full frame imaging mode
Spot	Single spot mode
Lines	Line scan mode

B.11 DetectorMode Enum

The DetectorMode enum describes the detector mode for the SEM.

Value	Description
All	Use all BSD detector segments
NorthSouth	Subtract bottom two segments (C,D) from top two (A,B)
EastWest	Subtract left two segments (B,D) from right two (A,C)
A	Select only segment A
B	Select only segment B
C	Select only segment C
D	Select only segment D
SED	Select only SE Detector

B.12 AcquisitionState Class

An AcquisitionState object describes the microscope state at the time of acquisition. AcquisitionState objects are returned from Phenom NavCam and SEM acquisitions. AcquisitionState objects can be stored and loaded together with the image in JPEG and TIFF files.

AcquisitionState properties:

acqHW	Text describing the acquisition hardware.
acquisitionType	Enum, either AcquisitionType.NavCam for NavCam acquisitions or AcquisitionType.Sem for SEM acquisitions.
annotations	annotations
appliedBrightness	as currently applied by the Phenom
appliedContrast	as currently applied by the Phenom
appliedGamma	as currently applied by the Phenom
beamShift	optional [m]
bsdGain	Analog gain applied to the detector signal [dB] (only for SEM images)
bsdMixFactor	mix factor of the BSD signal in the acquisition
bsdOffset	Analog offset applied to the BSD signal [V] (only for SEM images).
dataBarHeight	data bar height in pixels
dwellTime	pixel dwell time [second] (only valid for SEM images)
emissionCurrent	emission current [Ampere]

filamentPower	electron source filament power [Watt]
highVoltage	acceleration voltage [V]
instrumentEdition	Text describing the instrument edition.
instrumentID	Instrument ID text.
instrumentType	Text describing the instrument type.
integrations	Number of frames that where integrated to acquire this image.
interlace	Interlace value (only for SEM images).
multiStage	Multi-axis stage position
pixelSize	Dimensions (width, height) of a pixel in [meters].
position	stage position in [meter]
rotation	scan rotation in [radian]
sampleHeight	Height of the sample in [m]
sampleHolderID	Sample holder ID text, optional.
sampleHolderType	Integer number representing the sample holder type, optional.
samplePressureEstimate	Estimate of sample pressure in [Pa]
sampleRadius	Radius of the sample in [m]
scanCenter	Center of the scan in the available range
scanHW	Text describing the scan hardware.
scanScale	Scale of the scan in the available range
sedGain	Analog gain applied to the SED signal (only for SEM images).
sedMixFactor	Factor of the SED signal in the acquisition
sedOffset	Analog offset applied to the SED signal [V] (only for SEM images).
sedScintillatorVoltage	SED scintillator voltage [V]
segments	detector configuration used for the acquisition
softwareVersion	Text describing the instrument software version.

sourceTilt	x,y sourcetilt values.
sourceTime	total time [hour] the electron source has been "on"
spotPresetName	optional
spotSize	Spotsize (only for SEM images).
stigmator	x,y stigmator values.
time	Time stamp of the acquisition in seconds since 1 Jan 1970. The format is the same as that returned from the time() function in the time module.
workingDistance	distance from sample to lens [meter]

B.13 Sample holder type

Type	Sample holder type
0	Unknown
1	Standard
2	ChargeReduction
3	Metallurgical
4	MetallurgicalChargeReduction
5	MicrotoolTiltRotation
6	InsertXView
7	MicroElectronicsInsert
8	CorePlug
100	XlManualHeight
200	Delphi
513	TemperatureControlled
514	TiltRotation
612	Tensile
868	XlMotorizedHeight
870	XlMotorizedTZXR_11

B.14 Size Class

Integer width, height pair.

width	Width in pixels
height	Height in pixels

B.15 Licensing

LicenseInfo class

instrumentId	*Phenom instrument id, e.g. MVE123456*
username	*PPI license username*
password	*PPI license password*

License Features

Spectroscopy	*EDS Spectrum analysis*
Stitching	*AIM tile stitching*
ParticleMetric	*ParticleMetric*
RoughnessReconstruction	*3DRR RoughnessReconstruction*

CheckLicense(features)

Validate PPI license for specified features, eg: ppi.CheckLicense(ppi.Features.Stitching)

HasLicense(features)

Request if license is valid for specified features.

InstallLicense(instrumentId, username, password)

Install Phenom license details on the system.

UninstallLicense(instrumentId)

Uninstall the PPI license for the given instrument

GetLicenses()

Returns the installed PPI licenses as a list of LicenseInfo objects.

B.16 CamParams Class

The original NavCam delivers only 480 x 480 RGB images; any other size is an invalid parameter. The improved NavCam in Phenom G2 and onwards generations provide images with a resolution of 912 x 912 pixels. When requesting a different size, the image is scaled by the Phenom to the requested resolution.

size	Size object describing the required frame size in pixels.
nFrames	Number of frames to integrate in a single acquisition.

B.17 ScanParams Class

The ScanParams class describes the parameters that define a SEM acquisition.

size	Size object describing the required frame size in pixels.
nFrames	Number of frames to integrate in a single acquisition.
detector	DetectorMode enum that describes the detector for this acquisition.
hdr	Boolean value. Use High Dynamic Range mode: acquire a 16-bit raw image without contrast/brightness applied.
center	Position type. Center of the acquisition within the field of view.
scale	Scale factor of the acquisition within the field of view.

B.18 Image Class

The Image class represents the image data of an acquisition.

size	The size of the image in pixels as a Size object.
width	The width of the image in pixels.
height	The height of the image in pixels.
encoding	The pixel type of the image as a PixelType enum.
strideX	The offset in memory from pixel (x, y) to (x + 1, y)
strideY	The offset in memory from pixel (x, y) to (x, y + 1)

An Image is constructed with one of the following:

Image()

Creates an empty image: size = (0, 0)

Image(size, encoding)

Creates an image with given size and PixelType

Image(width, height, encoding)

Equivalent to Image(Size(width, height), encoding)

Image(object)

Create an Image from a memoryview object. The Image represents a view on the same memory as the memoryview object.

The following methods are available:

IsEmpty()

True if size == (0, 0)

MakeRef(orientation)

Returns a view on the same image in a different orientation:

Neutral: Unchanged

SwapXY:	X and Y directions reversed
InvertX:	Horizontally flipped
InvertY:	Vertically flipped
Rotate90:	Rotated through 90 degrees
Rotate180:	Rotated through 180 degrees
Rotate270:	Rotated through 270 degrees

MakeRef(rectangle)

Returns a sub area view on the same image

MakeRef(left, top, right, bottom**)**

Equivalent to MakeRef(Rectangle(left, top, right, bottom))

Resize(size, encoding)

Reallocate image with new size and PixelType

Resize(width, height, encoding)

Equivalent to Resize(Size(width, height), encoding)

Operators

The Image type supports the following basic operations, where a and b are both image objects and f is a float:

a + b, a + f, f + a:	pixel-wise sum
a - b, a – f, f - a:	pixel-wise subtraction
a * b, a * f, f * a:	pixel-wise multiplication
a / b, a / f, f / a:	pixel-wise division
a += b, a += f:	in place pixel-wise addition
a -= b, a -= f:	in place pixel-wise subtraction
a *= b, a *= f:	in place pixel-wise multiplication
a /= b, a /= f:	in place pixel-wise division
a and b:	pixel-wise logical and
a == b:	size, type and pixel-wise equality
a != b:	size, type or pixel-wise unequal inequality
-a:	pixel-wise negative
~a:	pixel-wise inverse
a[x, y]:	read or assign to pixel value at position x, y

B.19 Acquisition Class

The Acquisition class contains the image and metadata of a single acquisition.

image	The pixel data of the instrument acquisition.

databar	The pixel data of the databar part of the image
fullImage	The pixel data of the full image, including databar.
metadata	The AcquisitionState object for this acquisition.
rawMetadata	String containing the raw metadata
magnification	The magnification of the image

B.20 Phenom Class

Main Phenom Proxy class. This class acts as a proxy to the Phenom. When instantiated, it represents one connection to the Phenom, on which you can make calls. Note that error situations are signaled by throwing exceptions of type Error. The following methods are available:

Phenom()

The default constructor connects to the Phenom for which a license is installed.

Phenom(addressOrInstrumentId)

addressOrInstrumentId: name or IP address of the instrument on the network. A PPI license must be installed for this Phenom.

Phenom(addressOrInstrumentId, username, password)

addressOrInstrumentId: name or IP address of the instrument on the network
username: username used to login on the instrument
password: password used to login on the instrument

GetInstrumentMode()

Query the current InstrumentMode of the Phenom

GetOperationalMode()

Query the current OperationalMode of the Phenom

GetExclusiveAccess()

Request exclusive access for this client

ReleaseExclusiveAccess()

Release exclusive access status for this client

GetExclusiveAccessStatus()

Inquire the current exclusive access status of the Phenom

GetSemSpotSize()

Get the SEM spot size.

SetSemSpotSize(value)

Set the SEM spot size.

Activate()

Make the instrument operational

Load()

Load the sample

MoveToNavCam()

Move the sample to the NavCam position

MoveToSem()

Move the sample under the SEM

Unload()

Unload the sample

StandBy()

Set the instrument in standby

Hibernate()

Set the instrument in hibernate")

PowerOff()

Switch off the instrument

MoveTo(pos, algorithm=NavigationAlgorithm.Auto)

Move to a position specified by absolute coordinates.

pos: Stage position in absolute coordinates (in meters)

algorithm: Select backlash correction for more accurate positioning

MoveBy(deltaX, deltaY, algorithm=NavigationAlgorithm.Auto)

Move to a position specified relative to the current position

deltaX: Stage movement in x-direction, in meters from the current position.

deltaY: Stage movement in y-direction, in meters from the current position.

Algorithm: Select backlash correction for more accurate positioning.

GetStageModeAndPosition()

Get current stage mode and position

GetStageStroke()

Query the current possible stage stroke (i.e., the maximum values for stage positions).

GetHFW()

Query the current field of view size (horizontal field width, “HFW” in meters).

SetHFW(value)

Set the field of view (horizontal field width, “HFW” of the currently active imaging device (i.e., NavCam or SEM) in meters, effectively setting the zoom level.

value: Horizontal Field Width in meters (smaller number is higher magnification factor).

GetHFWRange()

Query the allowed range of HFW values (i.e., the possible magnification range) of the currently active imaging device.

GetNavCamWD()

Query the current working distance of the NavCam (“WD” - the focal distance between the lens and the sample).

SetNavCamWD(value)

Set the current working distance of the NavCam (in meters).

value: Working distance (in meters).

GetNavCamWDRange()

Get the available range of working distance values for the NavCam (in meters)

GetNavCamContrast()

Get the current contrast value of the NavCam (the ratio between light-field (in-line) and dark-field (ambient) lighting of the sample).

SetNavCamContrast(value)

Set the current contrast value of the NavCam (the ratio between light-field (in-line) and dark-field (ambient) lighting of the sample)

value: current ratio (0 .. 1)

GetNavCamBrightness()

Get the current brightness value of the NavCam (total light intensity)

SetNavCamBrightness(value)

Set the current brightness value of the NavCam (total light intensity)

value: Brightness value

NavCamAcquireImage(params)

Acquire a (color) image from the NavCam sensor using the given acquisition parameters.

params: Camera settings for the acquisition

returns: an Acquisition object (combination of image and metadata)

NavCamGetLiveImageCopy(framesDelay=0)

Acquires a (color) image from the NavCam with the settings as currently in effect for live viewing.

framesDelay: number of frames to wait before copying (default value is 0, meaning acquire immediately)

returns: an Acquisition object (combination of image and metadata)

GetSemWD()

Query the current working distance of the SEM (\"WD\" - the focal distance between the lense and the sample)")

SetSemWD(value)

Set the current working distance of the SEM (in meters)

value: Working distance (in meters)

GetSemWDRange()

Get the available range of working distance values for the SEM (in meters)

SemAutoFocus(algorithm=AutoFocusAlgorithm.Auto)

Perform an AutoFocus algorithm and set the current working distance (focus value) to the optimum value. This will take a few seconds, depending on which algorithm is used.

algorithm: Select which AutoFocus algorithm to use

GetSemContrast()

Get the current contrast value of the SEM

SetSemContrast(value)

Set the current contrast value for the SEM

GetSemBrightness()

Get the current brightness value of the SEM

SetSemBrightness(value)

Set the current brightness value for the SEM

SemAutoContrastBrightness()

Perform an automatic contrast/brightness optimization routine on the SEM. This will take a few seconds and set the optimum contrast and brightness values for the current sample

SemAutoSourceTilt()

Perform an automatic source tilt alignment. This may take up to several minutes if the current settings are very far from the optimal values. In this case, the Phenom will perform a full scan of all possible source tilt settings to determine the optimal values. Note that on the Phenom itself, this algorithm is only available in “expert” mode, in the source tilt alignment page. The values found with the algorithm are persistent (i.e., they will stay in effect even if the Phenom is brought to standby, hibernate, or even switched off and back on again). Forcing an auto source tilt alignment is not recommended, since the optimal position is only valid when the electron source has been “on” for a longtime (at minimum half an hour); changing the alignment may result in deteriorated image quality once the source reaches its optimum working conditions.

SemAbortAutoSourceTilt()

Abort an auto source tilt algorithm and restore the source tilt values which were in effect prior to starting the alignment procedure.

GetSemViewingMode()

Query the current viewing mode of the SEM

SetSemViewingMode(value)

Set the current viewing mode of the SEM

GetSemSpotSize()

Query the current SEM spot size (in Amps / $Volt^{½}$)

SetSemSpotSize(value)

Set the SEM spot size (in Amps / $Volt^{½}$)

GetSemSpotSizeRange()

Get the available range of spot sizes for the SEM (in Amps / $Volt^{½}$)

GetSemHighTension()

Query the current SEM high tension (in Volt)

SetSemHighTension(value)

Set the SEM high tension (in Volt)

GetSemHighTensionRange()

Get the available high tension range for the SEM (in Volt)

GetSemRotation()

Get the current SEM scan rotation (in radians)

SetSemRotation(value)

Set the SEM scan rotation (in radians)

SemAcquireImage(params)

Acquire an image from the SEM

params: Scan parameters and detector settings

returns: an Acquisition object (combination of image and metadata).

SemGetLiveImageCopy(framesDelay=0)

Acquire a copy of the currently visible SEM image. Acquires an image from the SEM with the settings as currently in effect for live viewing."

framesDelay: number of frames to wait before copying (default value is 0, meaning acquire immediately)

returns: an Acquisition object (combination of image and metadata)

GetStemSegmentSelection()

Get the current of selected (activated) STEM segments.

returns: An StemSegmentSelection object with the selected STEM segments.

SetStemSegmentSelection(StemSegmentSelection)

Set the selected (activated) STEM segments.

StemSegmentSelection: Class of selected STEM segments

B.21 Phenom Notifications

The Phenom class provides event properties which provide notifications about Phenom state changes. Use the event connect() function to connect your notification handling code. Here is an event example:

```
import PyPhenom as ppi

phenom = ppi.Phenom()
phenom.SemHighTensionChanged.connect(lambda value: print('HV changed:', value))
phenom.SetSemHighTension(-5.3e3)
phenom.SetSemHighTension(-10e3)
phenom.SetSemHighTension(-15e3)
```

The Phenom class provides the following events:

ConnectionLost

Connection to the Phenom is lost

FieldWidthChanged

Field width changed event (float)

FieldWidthRangeChanged

Field width range changed event (Range)

InstrumentModeChanged

Instrument mode changed event (InstrumentMode)

NavCamBrightnessChanged

NavCam brightness changed event (float)

NavCamContrastChanged

NavCam contrast changed event (float)

NavCamWorkingDistanceChanged

NavCam working distance changed event (float)

NavCamWorkingDistanceRangeChanged

NavCam working distance range changed event (Range)

NavCamOverviewImageUpdated

NavCam overview image update event

OperationalModeChanged

Operational mode changed event (OperationalMode)

ProgressAreaSelectionChanged

Progress area selection changed event (LoadingWorkArea, Progress)

SampleHolderEvent

Sample holder event (int, string)

SampleHolderStatusChanged

Sample holder status changed event (SampleHolderStatus)

SemAllowedVacuumChargeReductionChanged

SEM allowed vacuum charge reduction changed event

SemBrightnessChanged

SEM brightness changed event (float)

SemBsdGainChanged

BSD gain changed event (float)

SemBsdOffsetChanged

BSD offset changed event (float)

SemContrastChanged

SEM contrast changed event (float)

SemDeviceModeChanged

SEM device mode changed event (SemMode)

SemHighTensionChanged

SEM high tension changed event (float)

SemImageShiftChanged

SEM image shift changed event (Position)

SemProgressDeviceModeChanged

SemProgressDeviceMode changed event (SemMode, Progress)

SemRemaingTimeToEnableSedChanged

SEM remaing time to enable SED changed event (float)

SemRotationChanged

SEM rotation changed event (float)

SemSedFailureOccured

SED failure occured event (SedFailure)

SemSedStateChanged

SED state changed event (SedState)

SemSpotSizeChanged

SEM spot size changed event (float)

SemSpotSizeRangeChanged

SEM spot size range changed event (Range)

SemStigmateChanged

SEM stigmation changed event (Position)

SemTargetVacuumChargeReductionChanged

SEM target vacuum charge reduction changed event

SemWorkingDistanceChanged

SEM working distance changed event (float)

SemWorkingDistanceRangeChanged

SEM working distance range changed event (Range)

StageModeAndPositionChanged

Stage mode and position changed event (StageModeAndPosition)

B.22 Phenom functions

FindPhenoms function

FindPhenom(timeout)

timeout: time to wait in seconds for a Phenom to respond to a network query.

returns: the first Phenom detected on your local area network.

FindPhenoms(timeout)

timeout: time to wait in seconds for a Phenom to respond to a network query.

returns: list of phenom name and address pairs.

GetImageInfo function

GetImageInfo(filename)

filename: path to a Phenom image file.

returns: AcquisitionState object that contains all acquisition information for the image.

Load function

Load(filename)

Load an Acquisition object from a file. TIFF, JPEG and BMP files are supported. Only TIFF and JPEG files can contain acquisition metadata.

Save function

Save(acquisition, filename)

Save an Acquisition object to a file. TIFF, JPEG and BMP files are supported. Only TIFF and JPEG files can contain acquisition metadata.

GetSampleHolderMap(holderType)

Returns a dictionary object with sample holder labels and their positions.

MagnificationFromFieldWidth(hfw)

Returns the image magnification for the given HFW value as displayed on the Phenom.

MagnificationFromFieldWidth(hfw, displaySize)

Returns the image magnification for the given HFW relative to the given display size.

MagnificationToFieldWidth(magnification)

Returns the HFW value for an image magnification value as displayed on the Phenom.

MagnificationToFieldWidth(magnification, displaySize)

Returns the HFW value for an image magnification value relative to the given display size.

AddDatabar(acquisition)

Returns a new Acquisition object with a databar image attached to the image.

RemoveDatabar(acquisition)

Returns a new Acquisition object with a the databar part of the image removed.

B.23 MovieWriter class

The MovieWriter class provides the functionality to create a movie file form a sequence of images.

MovieWriter(size, path, fps, compressionRatio=0.1)

size: Size of the movie in pixels, e.g. ppi.Size(1920, 1080)

path: Path name of the movie file, e.g. 'C:/Movies/Movie.mp4'

fps: The framerate at which the movie should be played, e.g 20

compressionRatio: The target compression ratio for the movie file.

MovieWriter(width, height, path, fps, compressionRatio=0.1)

width: Width of the movie in pixels, e.g. 1920

height: Height of the movie in pixels, e.g. 1080

path: Path name of the movie file, e.g. 'C:/Movies/Movie.mp4'

fps: The framerate at which the movie should be played, e.g 20

compressionRatio: The target compression ratio for the movie file.

AddFrame(image)

Add *image* as the next frame in the movie.

Close()

Complete the movie file with all images as added up till now.

B.24 Colors

In PyPhenom colors can be made using:

```
Color = ppi.Colors.color
```

Where color is one of the following colors:

AliceBlue = RGB(240,248,255)

AntiqueWhite = RGB(250,235,215)

Aqua = RGB(0,255,255)

Aquamarine = RGB(127,255,212)

Azure = RGB(240,255,255)

Beige = RGB(245,245,220)

Bisque = RGB(255,228,196)

Black = RGB(0,0,0)

BlanchedAlmond = RGB(255,235,205)

Blue = RGB(0,0,255)

BlueViolet = RGB(138,43,226)

Brown = RGB(165,42,42)
BurlyWood = RGB(222,184,135)
CadetBlue = RGB(95,158,160)
Chartreuse = RGB(127,255,0)
Chocolate = RGB(210,105,30)
Coral = RGB(255,127,80)
CornflowerBlue = RGB(100,149,237)
Cornsilk = RGB(255,248,220)
Crimson = RGB(220,20,60)
Cyan = RGB(0,255,255)
DarkBlue = RGB(0,0,139)
DarkCyan = RGB(0,139,139)
DarkGoldenrod = RGB(184,134,11)
DarkGray = RGB(169,169,169)
DarkGreen = RGB(0,100,0)
DarkKhaki = RGB(189,183,107)
DarkMagenta = RGB(139,0,139)
DarkOliveGreen = RGB(85,107,47)
DarkOrange = RGB(255,140,0)
DarkOrchid = RGB(153,50,204)
DarkRed = RGB(139,0,0)
DarkSalmon = RGB(233,150,122)
DarkSeaGreen = RGB(143,188,143)
DarkSlateBlue = RGB(72,61,139)
DarkSlateGray = RGB(47,79,79)
DarkTurquoise = RGB(0,206,209)
DarkViolet = RGB(148,0,211)
DeepPink = RGB(255,20,147)
DeepSkyBlue = RGB(0,191,255)
DimGray = RGB(105,105,105)
DodgerBlue = RGB(30,144,255)
Firebrick = RGB(178,34,34)
FloralWhite = RGB(255,250,240)
ForestGreen = RGB(34,139,34)
Fuchsia = RGB(255,0,255)
Gainsboro = RGB(220,220,220)
GhostWhite = RGB(248,248,255)
Gold = RGB(255,215,0)

Goldenrod = RGB(218,165,32)
Gray = RGB(128,128,128)
Green = RGB(0,128,0)
GreenYellow = RGB(173,255,47)
Honeydew = RGB(240,255,240)
HotPink = RGB(255,105,180)
IndianRed = RGB(205,92,92)
Indigo = RGB(75,0,130)
Ivory = RGB(255,255,240)
Khaki = RGB(240,230,140)
Lavender = RGB(230,230,250)
LavenderBlush = RGB(255,240,245)
LawnGreen = RGB(124,252,0)
LemonChiffon = RGB(255,250,205)
LightBlue = RGB(173,216,230)
LightCoral = RGB(240,128,128)
LightCyan = RGB(224,255,255)
LightGoldenrodYellow = RGB(250,250,210)
LightGray = RGB(211,211,211)
LightGreen = RGB(144,238,144)
LightPink = RGB(255,182,193)
LightSalmon = RGB(255,160,122)
LightSeaGreen = RGB(32,178,170)
LightSkyBlue = RGB(135,206,250)
LightSlateGray = RGB(119,136,153)
LightSteelBlue = RGB(176,196,222)
LightYellow = RGB(255,255,224)
Lime = RGB(0,255,0)
LimeGreen = RGB(50,205,50)
Linen = RGB(250,240,230)
Magenta = RGB(255,0,255)
Maroon = RGB(128,0,0)
MediumAquamarine = RGB(102,205,170)
MediumBlue = RGB(0,0,205)
MediumOrchid = RGB(186,85,211)
MediumPurple = RGB(147,112,219)
MediumSeaGreen = RGB(60,179,113)
MediumSlateBlue = RGB(123,104,238)

MediumSpringGreen = RGB(0,250,154)
MediumTurquoise = RGB(72,209,204)
MediumVioletRed = RGB(199,21,133)
MidnightBlue = RGB(25,25,112)
MintCream = RGB(245,255,250)
MistyRose = RGB(255,228,225)
Moccasin = RGB(255,228,181)
NavajoWhite = RGB(255,222,173)
Navy = RGB(0,0,128)
OldLace = RGB(253,245,230)
Olive = RGB(128,128,0)
OliveDrab = RGB(107,142,35)
Orange = RGB(255,165,0)
OrangeRed = RGB(255,69,0)
Orchid = RGB(218,112,214)
PaleGoldenrod = RGB(238,232,170)
PaleGreen = RGB(152,251,152)
PaleTurquoise = RGB(175,238,238)
PaleVioletRed = RGB(219,112,147)
PapayaWhip = RGB(255,239,213)
PeachPuff = RGB(255,218,185)
Peru = RGB(205,133,63)
Pink = RGB(255,192,203)
Plum = RGB(221,160,221)
PowderBlue = RGB(176,224,230)
Purple = RGB(128,0,128)
Red = RGB(255,0,0)
RosyBrown = RGB(188,143,143)
RoyalBlue = RGB(65,105,225)
SaddleBrown = RGB(139,69,19)
Salmon = RGB(250,128,114)
SandyBrown = RGB(244,164,96)
SeaGreen = RGB(46,139,87)
SeaShell = RGB(255,245,238)
Sienna = RGB(160,82,45)
Silver = RGB(192,192,192)
SkyBlue = RGB(135,206,235)
SlateBlue = RGB(106,90,205)

SlateGray = RGB(112,128,144)

Snow = RGB(255,250,250)

SpringGreen = RGB(0,255,127)

SteelBlue = RGB(70,130,180)

Tan = RGB(210,180,140)

Teal = RGB(0,128,128)

Thistle = RGB(216,191,216)

Tomato = RGB(255,99,71)

Transparent = RGBA(255,255,255,0)

Turquoise = RGB(64,224,208)

Violet = RGB(238,130,238)

Wheat = RGB(245,222,179)

White = RGB(255,255,255)

WhiteSmoke = RGB(245,245,245)

Yellow = RGB(255,255,0)

YellowGreen = RGB(154,205,50)

B.25 PDF reporting document class

AddPage()

Adds a page object to the document.

AddPageLabel(page_num, style, first_page, prefix)

Adds a page number to the document.

CreateOutline(parent, title, encoder)

Creates an outline on the document.

FreeDoc()

keeps and recycles loaded resources (such as fonts and encodings) when new document requires these resources.

GetCurrentEncoder()

Gets the handle of the current encoder of the document object.

GetCurrentPage()

Returns the current page

GetEncoder(encoding_name)

Gets the handle of an encoder object by specified encoding name.

GetFont(font_name)

Gets the current font

GetInfoAttr(type)

gets an attribute value from info dictionary.

GetPageLayout()

Returns the current setting for page layout.

GetPageMode()

Returns the current setting for page mode.

GetStreamSize()

Gets the size of the temporary stream of the document.

HasDoc()

If the specified document handle is valid, it returns True. Otherwise, it returns error-code and error-handler is called.

InsertPage((Page)page)

Creates a new page and inserts it just before the specified page.

LoadJpegImage((str)filename)

Loads an external JPEG image file.

LoadTTFontFromFile((str)file_name, (bool)embedding)

Loads a TrueType font from an external file and register it to a document object.

LoadTTFontFromFile2((str)file_name, (int)index, (bool)embedding)

Loads a TrueType font from an external file and register it to a document object.

LoadType1FontFromFile((str)afm_file_name, (str)data_file_name)

Loads a Type1 font from an external file and registers it in the document object.

NewDoc()

Creates a new document.

ReadFromStream((object)arg2, (int)data)

Copies the data from the temporary stream of the document into buffer

ResetError()

Once an error code is set, IO processing functions cannot be invoked. In the case of executing a function after the cause of the error is fixed, an application have to invoke ResetError() to clear error-code before executing functions.

ResetStream()

Rewinds the temporary stream of the document.

SaveToFile((str)file_name)

Saves the current document to file.

SaveToStream()

Saves the current document to a temporary stream of a document object.

SetCompressionMode((int)mode)

Set the mode of compression.

SetCurrentEncoder((str)encoding_name)

Sets the current encoder for the document.

SetEncryptionMode((EncryptMode)mode, (int)key_len)

Set the encryption mode. As the side effect, ups the version of PDF to 1.4

SetInfoAttr((InfoType)type, (str)value)

Sets the text of an info dictionary attribute, using current encoding of the document.

SetInfoDateAttr((InfoType)type, (object)value)

Sets a datetime attribute in the info dictionary.

SetOpenAction((Destination)value)

Set the first page to appear when a document is opened.

SetPageLayout((PageLayout)value)

Sets how the page should be displayed. If this attribute is not set, the setting of the viewer application is used.

SetPageMode((PageMode)value)

Sets how the document should be displayed.

SetPagesConfiguration((int)page_per_pages)

In the default setting, a doc object has one "Pages" object as root of pages. All "Page" objects are created as a kid of the "Pages" object. Since a "Pages" object can own only 8191 kids objects, the maximum number of pages are 8191 page. Additionally, the state that there are a lot of "Page" object under one "Pages" object is not good, because it causes performance degradation of a viewer application.

An application can change the setting of a pages tree by invoking SetPagesConfiguration (). If page_per_pages parameter is set to more than zero, a two-tier pages tree is created. A root "Pages" object can own 8191 "Pages" object, and each lower "Pages" object can own page_per_pages "Page" objects. As a result, the maximum number of pages becomes 8191 * page_per_pages page. An application cannot invoke SetPagesConfiguration () after a page is added to document.

SetPassword((str)owner_password, (str)user_password)

Sets a password for the document. If the password is set, document contents are encrypted.

SetPermission((int)permission)

Set the permission flags for the document.

UseCNSEncodings()

Enables simplified Chinese encodings. After UseCNSEncodings() is involed, an application can use the following simplified Chinese encodings.

- GB-EUC-H
- GB-EUC-V
- GBK-EUC-H
- GBK-EUC-V

UseCNSFonts()

Snables simplified Chinese fonts. After UseCNSFonts() is involed, an application can use the following simplified Chinese fonts.

- SimSun
- SimSun,Bold
- SimSun,Italic
- SimSun,BoldItalic
- SimHei
- SimHei,Bold
- SimHei,Italic
- SimHei,BoldItalic

UseCNTEncodings()

Enables traditional Chinese encodings. After UseCNTEncodings() is involed, an application can use the following traditional Chinese encodings.

- GB-EUC-H
- GB-EUC-V
- GBK-EUC-H
- GBK-EUC-V

UseCNTFonts()

Enables traditional Chinese fonts. After UseCNTFonts() is involed, an application can use the following traditional Chinese fonts.

- MingLiU
- MingLiU,Bold
- MingLiU,Italic
- MingLiU,BoldItalic

UseJPEncodings()

Enables Japanese encodings. After UseJPEncodings() is invoked, an application can use the following Japanese encodings.

- 90ms-RKSJ-H
- 90ms-RKSJ-V
- 90msp-RKSJ-H
- EUC-H
- EUC-V

UseJPFonts()

Enables Japanese fonts. After UseJPFonts() is involed, an application can use the following Japanese fonts.

- MS-Mincyo
- MS-Mincyo,Bold
- MS-Mincyo,Italic
- MS-Mincyo,BoldItalic
- MS-Gothic
- MS-Gothic,Bold
- MS-Gothic,Italic
- MS-Gothic,BoldItalic
- MS-PMincyo
- MS-PMincyo,Bold
- MS-PMincyo,Italic
- MS-PMincyo,BoldItalic
- MS-PGothic
- MS-PGothic,Bold
- MS-PGothic,Italic

UseKREncodings()

Enables Korean encodings. After UseKREncodings() is involed, an application can use the following Korean encodings.

- KSC-EUC-H
- KSC-EUC-V
- KSCms-UHC-H
- KSCms-UHC-HW-H
- KSCms-UHC-HW-V

UseKRFonts()

Enables Korean fonts. After UseKRFonts() is involed, an application can use the following Korean fonts.

- DotumChe
- DotumChe,Bold
- DotumChe,Italic
- DotumChe,BoldItalic
- Dotum
- Dotum,Bold
- Dotum,Italic
- Dotum,BoldItalic
- BatangChe
- BatangChe,Bold

- BatangChe,Italic
- BatangChe,BoldItalic
- Batang
- Batang,Bold
- Batang,Italic
- Batang,BoldItalic

UseUTFEncodings()

Enables UTF-8 encodings. After UseUTFEncodings() is invoked, an application can include UTF-8 encoded Unicode text (up to 3-byte UTF-8 sequences only). An application can use the following Unicode encodings (but only with TrueType fonts):

- UTF-8

B.26 PDF reporting page class

Arc((object)position, (float)ray, (float)angle1, (float)angle2)

BeginText()

Circle((object)position, (float)ray)

Clip()

ClosePath()

ClosePathEofillStroke()

ClosePathFillStroke()

ClosePathStroke()

Concat((Transformation)transformation)

CreateDestination()

Creates a new destination object for the page.

CreateLinkAnnot((object)rectangle, (Destination)destination)

Creates a new link annotation object for the page.

CreateTextAnnot((object)rectangle, (str)text, (Encoder)encoder)

Creates a new text annotation object for the page.

CreateURILinkAnnot((object)rectangle, (str)text)

Creates a new web link annotation object for the page.

CurveTo((object)p1, (object)p2, (object)p3)

CurveTo2((object)p2, (object)p3)

CurveTo3((object)p1, (object)p3)

DrawImage((Image)image, (float)x, (float)y, (float)width, (float)height)

Ellipse((object)position, (float)xray, (float)yray)

EndPath()

EndText()

Eoclip()

Eofill()

EofillStroke()

Fill()

FillStroke()

GRestore()

GSave()

GetCharSpace()

Gets the current value of the page's character spacing.

GetCurrentFont()

Gets the handle of the page's current font.

GetCurrentFontSize()

Gets the size of the page's current font.

GetCurrentPos()

Gets the current position for path painting. An application can invoke GetCurrentPos() only when graphics mode is gmode is path.

GetCurrentTextPos()

Gets the current position for text showing. An application can invoke Page_GetCurrentTextPos() only when graphics mode is gmode is text.

GetDash()

Gets the current pattern of the page.

GetFillingColorSpace()

Returns the current value of the page's stroking color space.

GetFlat()

Gets the current value of the page's flatness.

GetGMode()

Gets the current graphics mode.

GetGStateDepth()

Returns the number of the page's graphics state stack.

GetGrayFill()

Returns the current value of the page's filling color.

GetGrayStroke()

Returns the current value of the page's stroking color.

GetHeight()

Gets the height of the page.

GetHorizontalScalling()

Returns the current value of the page's horizontal scalling for text showing.

GetLineCap()

Gets the current line cap style of the page.

GetLineJoin()

Gets the current line join style of the page.

GetLineWidth()

Gets the current line width of the page.

GetMiterLimit()

Gets the current value of the page's miter limit.

GetRGBFill()

Returns the current value of the page's filling color

GetRGBStroke()

Returns the current value of the page's stroking color.

GetStrokingColorSpace()

Returns the current value of the page's stroking color space.

GetTextLeading()

Returns the current value of the page's line spacing.

GetTextMatrix()

Gets the current text transformation matrix of the page.

GetTextRenderingMode()

Returns the current value of the page's text rendering mode.

GetTextRise()

Returns the current value of the page's text rising.

GetTransMatrix()

Gets the current transformation matrix of the page.

GetWidth()

Gets the width of the page.

GetWordSpace()

Returns the current value of the page's word spacing.

LineTo((float)x, (float)y)

MeasureText((str)text, (float)width, (bool)word_wrap)

Calculates the byte length which can be included within the specified width.

MoveTextPos((float)x, (float)y)

MoveTextPos2((float)x, (float)y)

MoveTo((float)x, (float)y)

MoveToNextLine()

Rectangle((float)x, (float)y, (float)width, (float)height)

SetCMYKFill((float)c, (float)m, (float)y, (float)k)

SetCMYKStroke((float)c, (float)m, (float)y, (float)k)

SetCharSpace((float)value)

SetDash((DashMode)value)

SetFontAndSize((Font)font, (float)size)

SetGrayFill((float)value)

SetGrayStroke((float)value)

SetHeight((float)value)

Changes the height of a page

SetHorizontalScalling((float)value)

SetLineCap((LineCap)value)

SetLineJoin((LineJoin)value)

SetLineWidth((float)line_width)

SetMiterLimit((float)line_width)

SetRGBFill((Rgb24)value)

SetRGBStroke((Rgb24)value)

SetRotate((int)angle)

Sets rotation angle of the page.

SetSize((PageSize)size [,
(PageDirection)direction=PyPhenom.Pdf.PageDirection.Portrait])

Changes the size and direction of a page to a predefined size.

SetSlideShow((TransitionStyle)style, (float)disp_time, (float)trans_time)

Configures the setting for slide transition of the page.

The transition style. The following values are available.

BarnDoorsHorizontalIn = PyPhenom.Pdf.TransitionStyle.BarnDoorsHorizontalIn

BarnDoorsHorizontalOut = PyPhenom.Pdf.TransitionStyle.BarnDoorsHorizontalOut

BarnDoorsVerticalIn = PyPhenom.Pdf.TransitionStyle.BarnDoorsVerticalIn

BarnDoorsVerticalOut = PyPhenom.Pdf.TransitionStyle.BarnDoorsVerticalOut

BlindsHorizontal = PyPhenom.Pdf.TransitionStyle.BlindsHorizontal

BlindsVertical = PyPhenom.Pdf.TransitionStyle.BlindsVertical

BoxIn = PyPhenom.Pdf.TransitionStyle.BoxIn

BoxOut = PyPhenom.Pdf.TransitionStyle.BoxOut

Dissolve = PyPhenom.Pdf.TransitionStyle.Dissolve

GlitterDown = PyPhenom.Pdf.TransitionStyle.GlitterDown

GlitterRight = PyPhenom.Pdf.TransitionStyle.GlitterRight

GlitterTopLeftToBottomRight = PyPhenom.Pdf.TransitionStyle.GlitterTopLeftToBottomRight

Replace = PyPhenom.Pdf.TransitionStyle.Replace

WipeDown = PyPhenom.Pdf.TransitionStyle.WipeDown

WipeLeft = PyPhenom.Pdf.TransitionStyle.WipeLeft

WipeRight = PyPhenom.Pdf.TransitionStyle.WipeRight

WipeUp = PyPhenom.Pdf.TransitionStyle.WipeUp

SetTextLeading((float)value)

SetTextMatrix((Transformation)transformation)

SetTextRenderingMode((TextRenderingMode)value)

SetTextRise((float)value)

SetWidth((float)value)

Changes the width of a page.

SetWordSpace((float)value)

ShowText((str)text)

ShowTextNextLine((str)text)

Stroke()

TextOut((float)xpos, (float)ypos, (str)text)

TextRect((RectangleD)rect, (str)text, (TextAlignment)alignment)

TextWidth((str)text)

Gets the width of the text in current fontsize, character spacing and word spacing.

B.27 EdsJobAnalyzer class

The following methods are available:

Abort()

AbortCurrentJob()

AbortJob(jobID)

jobID: Integer of job ID to be canceled.

AcquireDriftCorrectionReference(size)

Size: Optional integer to set size of reference area. Default = 512.

ActiveJobId()

Returns: Integer of current job ID.

AddLine(line, numberOfPoints, timePerPoint, numberOfPasses)

line: Line object on which the line is acquired in relative coordinates on the acquisition image.

numberOfPoints: Optionally integer number of points on the line. Default = 100.

timePerPoint: Optionally float time per point on the line in seconds. Default = 0.1.

numberOfPasses: Optionally integer number of iteration over the line. Default = 1.

Returns: EdsLineData object

AddMap(region , mapSize, timePerPoint)

region: recatangleD object where the map needs to be acquired in relative coordinates on the acquisition image.

mapSize: Optionally integer number of pixels on longest axis in the map. Default = 64.

timePerPoint: Optionally float time per pixel in seconds. Default = 0.01.

Returns: EdsMapData object

AddRegion(region, maxTime=0, maxCounts=0)

region: recatangleD object where the region needs to be acquired in relative coordinates on the acquisition image.

maxTime: Optionally float maximum time before stopping if maxCounts is exceeded first it stops based on maxCounts. Default = 0.

maxCounts: Optionally integer maximum counts before stopping if maxTime is exceeded first it stops based on maxTime. Default = 0.

Returns: EdsRegionData object

AddSpot(position, maxTime, maxCounts)

position: position object where the spot needs to be acquired in relative coordinates on the acquisition image.

maxTime: Optionally float maximum time before stopping if maxCounts is exceeded first it stops based on maxCounts. Default = 0.

maxCounts: Optionally integer maximum counts before stopping if maxTime is exceeded first it stops based on maxTime. Default = 0.

Returns: EdsSpotData object

EdsCountPerSeconds()

Returns: Float with number of counts

GetTemperature()

Returns: Float with current detector temperature

IsAnalyzing()

Returns: Boolean whether the detector is still acquiring data

IsDetectorReady()

IsJobIdInUse(jobId)

jobId: integer

Returns: Boolean whether the jobId is in use

IsMaxTemperatureExceeded()

Returns: Boolean whether the maximum detector temperature is exceeded.

JobProgress()

Returns: Float with current progress

StopCurrentJob()

StopJob(jobId)

jobId: integer of jobId to be stopped

Wait()

B.28 EdsLineData Class

The EdsLineData class contains the following parameters:

line

lineSpectrum

sumSpectrum

uniqueId

B.29 EdsLineJob Class

The EdsLineJob class contains the following parameters:

acquisitionTimePerPoint

driftCorrectionInitialAcquisition

imageLine

imageSize

numberOfPasses

preset

uniqueId

B.30 EdsMapData Class

The EdsMapData class contains the following parameters:

gridSpectrum

region

sumSpectrum

uniqueId

B.31 EdsMapJob Class

The EdsMapJob class contains the following parameters:

acquisitionTimePerGridPosition

driftCorrectionInitialAcquisition

imageRect

imageSize

numberOfPasses

preset

uniqueId

B.32 EdsSpotData Class

The EdsSpotData class contains the following parameters:

position

spotSpectrum

uniqueId

B.33 EdsSpotJob Class

The EdsSpotJob class contains the following parameters:

driftCorrectionInitialAcquisition

imagePos

imageSize

maxCounts

maxTime

preset

uniqueId

B.34 Element Enum

All elements are available as an enum:

Ac = PyPhenom.Spectroscopy.Element.Ac
Ag = PyPhenom.Spectroscopy.Element.Ag
Al = PyPhenom.Spectroscopy.Element.Al
Am = PyPhenom.Spectroscopy.Element.Am
Ar = PyPhenom.Spectroscopy.Element.Ar
As = PyPhenom.Spectroscopy.Element.As
At = PyPhenom.Spectroscopy.Element.At
Au = PyPhenom.Spectroscopy.Element.Au
B = PyPhenom.Spectroscopy.Element.B
Ba = PyPhenom.Spectroscopy.Element.Ba
Be = PyPhenom.Spectroscopy.Element.Be
Bi = PyPhenom.Spectroscopy.Element.Bi
Bk = PyPhenom.Spectroscopy.Element.Bk
Br = PyPhenom.Spectroscopy.Element.Br
C = PyPhenom.Spectroscopy.Element.C
Ca = PyPhenom.Spectroscopy.Element.Ca
Cd = PyPhenom.Spectroscopy.Element.Cd
Ce = PyPhenom.Spectroscopy.Element.Ce
Cf = PyPhenom.Spectroscopy.Element.Cf
Cl = PyPhenom.Spectroscopy.Element.Cl
Cm = PyPhenom.Spectroscopy.Element.Cm
Co = PyPhenom.Spectroscopy.Element.Co
Cr = PyPhenom.Spectroscopy.Element.Cr
Cs = PyPhenom.Spectroscopy.Element.Cs
Cu = PyPhenom.Spectroscopy.Element.Cu
Dy = PyPhenom.Spectroscopy.Element.Dy
Er = PyPhenom.Spectroscopy.Element.Er
Es = PyPhenom.Spectroscopy.Element.Es
Eu = PyPhenom.Spectroscopy.Element.Eu
F = PyPhenom.Spectroscopy.Element.F
Fe = PyPhenom.Spectroscopy.Element.Fe
Fr = PyPhenom.Spectroscopy.Element.Fr
Ga = PyPhenom.Spectroscopy.Element.Ga
Gd = PyPhenom.Spectroscopy.Element.Gd
Ge = PyPhenom.Spectroscopy.Element.Ge
H = PyPhenom.Spectroscopy.Element.H
He = PyPhenom.Spectroscopy.Element.He
Hf = PyPhenom.Spectroscopy.Element.Hf
Hg = PyPhenom.Spectroscopy.Element.Hg
Ho = PyPhenom.Spectroscopy.Element.Ho
I = PyPhenom.Spectroscopy.Element.I
In = PyPhenom.Spectroscopy.Element.In
Ir = PyPhenom.Spectroscopy.Element.Ir
K = PyPhenom.Spectroscopy.Element.K
Kr = PyPhenom.Spectroscopy.Element.Kr
La = PyPhenom.Spectroscopy.Element.La
Li = PyPhenom.Spectroscopy.Element.Li
Lu = PyPhenom.Spectroscopy.Element.Lu
Mg = PyPhenom.Spectroscopy.Element.Mg
Mn = PyPhenom.Spectroscopy.Element.Mn
Mo = PyPhenom.Spectroscopy.Element.Mo
N = PyPhenom.Spectroscopy.Element.N
Na = PyPhenom.Spectroscopy.Element.Na

Nb = PyPhenom.Spectroscopy.Element.Nb
Nd = PyPhenom.Spectroscopy.Element.Nd
Ne = PyPhenom.Spectroscopy.Element.Ne
Ni = PyPhenom.Spectroscopy.Element.Ni
None = PyPhenom.Spectroscopy.Element.None
Np = PyPhenom.Spectroscopy.Element.Np
O = PyPhenom.Spectroscopy.Element.O
Os = PyPhenom.Spectroscopy.Element.Os
P = PyPhenom.Spectroscopy.Element.P
Pa = PyPhenom.Spectroscopy.Element.Pa
Pb = PyPhenom.Spectroscopy.Element.Pb
Pd = PyPhenom.Spectroscopy.Element.Pd
Pm = PyPhenom.Spectroscopy.Element.Pm
Po = PyPhenom.Spectroscopy.Element.Po
Pr = PyPhenom.Spectroscopy.Element.Pr
Pt = PyPhenom.Spectroscopy.Element.Pt
Pu = PyPhenom.Spectroscopy.Element.Pu
Ra = PyPhenom.Spectroscopy.Element.Ra
Rb = PyPhenom.Spectroscopy.Element.Rb
Re = PyPhenom.Spectroscopy.Element.Re
Rh = PyPhenom.Spectroscopy.Element.Rh
Rn = PyPhenom.Spectroscopy.Element.Rn
Ru = PyPhenom.Spectroscopy.Element.Ru
S = PyPhenom.Spectroscopy.Element.S
Sb = PyPhenom.Spectroscopy.Element.Sb
Sc = PyPhenom.Spectroscopy.Element.Sc
Se = PyPhenom.Spectroscopy.Element.Se
Si = PyPhenom.Spectroscopy.Element.Si
Sm = PyPhenom.Spectroscopy.Element.Sm
Sn = PyPhenom.Spectroscopy.Element.Sn
Sr = PyPhenom.Spectroscopy.Element.Sr
Ta = PyPhenom.Spectroscopy.Element.Ta
Tb = PyPhenom.Spectroscopy.Element.Tb
Tc = PyPhenom.Spectroscopy.Element.Tc
Te = PyPhenom.Spectroscopy.Element.Te
Th = PyPhenom.Spectroscopy.Element.Th
Ti = PyPhenom.Spectroscopy.Element.Ti
Tl = PyPhenom.Spectroscopy.Element.Tl
Tm = PyPhenom.Spectroscopy.Element.Tm
U = PyPhenom.Spectroscopy.Element.U
V = PyPhenom.Spectroscopy.Element.V
W = PyPhenom.Spectroscopy.Element.W
Xe = PyPhenom.Spectroscopy.Element.Xe
Y = PyPhenom.Spectroscopy.Element.Y
Yb = PyPhenom.Spectroscopy.Element.Yb
Zn = PyPhenom.Spectroscopy.Element.Zn
Zr = PyPhenom.Spectroscopy.Element.Zr

B.35 Oxide Class

The following methods are available:

ToElementConcentration(oxideConcentration)
oxideConcentration: float
Returns: float element concentration

ToOxideConcentration(elementConcentration)
elementConcentration: float
Returns: float oxide concentration

The oxide class has the following parameters:

element

elementCount

oxygenCount

B.36 EdsAnalysis Class

The EdsAnalysis class is the parent class of:

- EdsDifferenceAnalysis
- EdsLineScanAnalysis
- EdsMapAnalysis
- EdsMsaAnalysis
- EdsRegionAnalysis
- EdsSpotAnalysis

The following methods are available:

ClearActiveOxideForElement(Element)

ClearFamilyOverride(Element)

GetActiveOxideForElement(Element)

Returns: Oxide

SetActiveOxideForElement(Oxide)

The EdsAnalysis class has the following parameters:

analysisOrderNr

autoIdEnabled

backgroundFitPoints

elementsWithActiveOxides

elementsWithFamilyOverrides

excludedElements

familyOverride

ignoredElements

imageCutout

includedElements

spectrum

B.37 EdsDifferenceAnalysis Class

The EdsDifferenceAnalysis class inherits all parameters and functions of the EdsAnalysis class and adds the following data:

minuendAnalysisOrderNr

subtrahendAnalysisOrderNr

B.38 EdsLineScanAnalysis Class

The EdsLineScanAnalysis class inherits all parameters and functions of the EdsAnalysis class and adds the following data:

line

spectra

B.39 EdsMapAnalysis Class

The EdsMapAnalysis class inherits all parameters and functions of the EdsAnalysis class and adds the following data:

area

mapColorIntensities

spectra

ImageToMap(acquisition) Helper function to construct a transformation matrix (see section 6.4.6) so you can easily map pixel coordinates from the image object in the EdsMapAnalysis to (x,y) positions in the corresponding EdsGridSpectrum (see below).

B.40 EdsMsaAnalysis Class

The EdsMsaAnalysis class inherits all parameters and functions of the EdsAnalysis.

B.41 EdsRegionAnalysis Class

The EdsRegionAnalysis class inherits all parameters and functions of the EdsAnalysis class and adds the following data:

area

B.42 EdsSpotAnalysis Class

The EdsSpotAnalysis class inherits all parameters and functions of the EdsAnalysis class and adds the following data:

location

B.43 EdsProject Class

The following methods are available:

AddImage(EdsImage)

__getitem__(imageNo)
imageNo: integer

Returns: EdsImage

B.44 EdsImage Class

The following methods are available:

AddAnalysis(EdsAnalysis)

AddAnalysis(EdsRegionData)

AddAnalysis(EdsSpotData)

AddAnalysis(EdsLineData)

AddAnalysis(EdsMapData)

__getitem__(n) (usage in Python: **EdsImage[n])**

n: integer

Returns: EdsImage

Data descriptors defined here:

acquisition

name

B.45 Spectrum Class

The following methods are available:

__add__(Spectrum) (usage in Python: **Spectrum + Spectrum)**

Input: Spectrum to be added

Returns: Spectrum

__sub__(Spectrum) (usage in Python: **Spectrum – Spectrum)**

Input: Spectrum to be subtracted

Returns: Spectrum

energies()

Returns: list

intensities()

Returns: list

Data descriptors defined here:

Data

copy spectrum data as Python list

dimensions

dispersion

offset

B.46 EdsLineSpectrum Class

The following methods are available:

Extract(index)

Input: extract spectrum at integer position index

Returns: Spectrum

ExtractAcquisition(index)

Input: extract eds acquisition at integer position index

Returns: EdsAcquisition

GetLiveTime((int)index)

Input: get live time of spectrum at integer position index

Returns: float

GetRealTime((int)index)

Input: get real time of spectrum at integer position index

Returns: float

Data descriptors defined here:

binSize

dispersion

empty

hasVariableLiveTime

hasVariableRealTime

metadata

offset

size

B.47 EdsGridSpectrum Class

The following methods are available:

Accumulate(Rectangle)

Input: rectangle

Returns: EdsAcquisition; summation of all the spectra in the given rectangle

Accumulate(Position a, Position b, width, SampleType, nPoints)

Returns: EdsLineSpectrum with *nPoints* entries, accumulated along a line from *a* to *b* in the grid spectrum, where each spectrum is summed over *width* pixels perpendicular to the line. When SampleType is ppi.Spectroscopy.GridSpectrumSampleType.Center, each position in the EdsLineSpectrum corresponds to the accumulated *width* spectra crossing through the center of each position; when it is GridSpectrumSampleType.Edge, it corresponds to the edges of the positions. The latter has the requirement that there are at least two sample points (i.e., *nPoints* >= 2), and guarantees that the points *a* and *b* are present in the results.

Extract(x, y)

Input: x, y pixel coordinates

Returns: Spectrum

ExtractAcquisition(x, y)
Input: x, y pixel coordinates
Returns: EdsAcquisition

GetLiveTime(x, y)
Input: x, y pixel coordinates
Returns: float

GetRealTime(x, y)
Input: x, y pixel coordinates
Returns: float

Data descriptors defined here:
binSize
dispersion
empty
hasVariableLiveTime
hasVariableRealTime
height
metadata
offset
size
width

B.48 IdentificationResult Class

Data descriptors defined here:
constituents
fit

B.49 QuantificationResult Class

Data descriptors defined here:
MnKaFWHM
approxBackground
approxComposition
background
composition

cutOff

debug

dimensions

fit

percentagePrecision

sumPeaks

sumPeaksScale

virtualTilt

B.50 AtomicConstituent Class

Data descriptors defined here:

Z

atomicFraction

B.51 WeightConstituent Class

Data descriptors defined here:

Z

weightFraction

B.52 EdsMetadata Class

Data descriptors defined here:

Ccorrection

azimuthAngle

detectorToOpticalAxis

detectorType

detectorWdOffset

elevationAngle

fastPeakingTime

highTension

holderTiltAngle

instrumentId

liveTime

realTime

resolution

sddSurfaceArea

slowPeakingTime

spectrumCorrection

workingDistance

B.53 Spectroscopy Functions

DrawElementMapOverlay(Acquisition, QuantMap, RectangleD, colors, alpha)

colors: dict

optional: alpha float default = 0.5

Returns: Image

DrawLineSpectrum(Acquisition, EdsLineData, FractionType, Size)

optional: FractionType default = weight

optional: Size (of output image) default = 640, 480

Returns: Image

DrawSpectrum (Spectrum, Range, Size)

optional: Range of x-axis default = 0,0 (autosizing)

optional: Size (of output image) default = 640, 480

Returns: Image

EdsAcquisitionFromMsa(MsaData)

Returns: EdsAcquisition

EdsAcquisitionToMsa(EdsAcquisition)

Returns: MsaData

FromAtomicNumber(Number)

Number: int

Returns: Element

FromSymbol(symbol)

symbol: string

Returns: Element

Identify(MsaData)

Returns: IdentificationResult

Identify(MsaData, exclusions)

exclusions: list

Returns: IdentificationResult

IntensityMap(EdsGridSpectrum , minEnergy, maxEnergy)

minEnergy: optional float

maxEnergy: optional float

Returns: Image

LineEnergy(Element, Line)

Returns: float

LineName(Line)

Returns: string

LinesForElement(Element)

Returns: list

LinesForEnergy(energy, maxDelta)

energy: float

maxDelta: float

Returns: list

LoadEdsProject(fileName)

fileName: string

Returns: EdsProject

Quantify(MsaData, elements)

elements: list

Returns: QuantificationResult

Quantify(MsaData, IdentificationResult)

Returns: QuantificationResult

Quantify(EdsAcquisition, elements)

elements: list

Returns: QuantificationResult

Quantify(MsaData)

Returns: QuantificationResult

Quantify(EdsAcquisition)
Returns: QuantificationResult

ReadMsaFile(filename)
filename: string
Returns: MsaData

saveEdsProject(EdsProject, filename)
filename: string

WriteMsaFile()
WriteMsaFile(MsaData, filename)
filename: string

WriteMsaFile(EdsAcquisition, filename)
filename: string

B.54 QuantMap Class

The following methods are available:
DrawElementMapOverlay(Acquisition, colors , alpha)
colors: dict
alpha: optional float
Returns: Image

ForElement(Element)
Returns: Image

Quantify()

Quantify(elements)
elements: list

Render(element_colors)
element_colors: dict
Returns: Image

Data descriptors defined here:
AbsoluteDotThreshold

ConcentrationApproximation

DarkRenderFraction

DotFactorThreshold

LightRenderFraction

MinimumRenderLevel

QuantThreshold

RawValueThresholdHigh

RawValueThresholdLow

RenderLevels

RenderingStrategy

Scaling

ScalingThreshold

B.55 Patterning Classes

LineScanPattern

The LineScanPattern defines a sequence of positions on a line inside the scan area and supports the following properties:

dwellTime	Time in seconds during which the scan will remain at each point on the line.
begin	Begin position of the line, relative to the scan area.
end	End position of the line, relative to the scan area.
pitch	Distance between points on the line, relative to HFW.

PointScanPattern

The PointScanPattern defines a single beam position and supports this property:

position	Beam position, relative to the scan area.

RectangleScanPattern

The RectangleScanPattern defines a sequence of positions defining a rectangle inside the scan area and supports the following properties:

dwellTime	Time in seconds during which the scan will remain at each point in the rectangle.
center	Center position of the rectangle, relative to the scan area.
size	(width, height) of the rectangle, relative to the scan area.
rotation	Optional rotation of the rectangle in radians.
pitchX	Horizontal distance between points in the rectangle, relative to HFW.
pitchY	Vertical distance between points in the rectangle, relative to HFW.
lineScanStyle	Line scanning style: Uniform: each line of positions in the rectangle is scanned from left to right.

	Serpentine: even lines are scanned from left to right, odd lines from right to left.

BitmapScanPattern

The BitmapScanPattern uses an image to define the number of positions and dwell times for each position inside a rectangle. Every pixel in the image defines a single position in the pattern. Use images of about 500x500 pixels or larger for good results. When using too small images, the individual pixel positions will show as individual spots.

This pattern supports the following function and properties:

SetImage(image)	Set a ppi.Image object which defines the pattern.
center	Center position of the pattern, relative to the scan area.
size	(width, height) of the image rectangle, relative to the scan area.
rotation	Optional rotation of the image in radians.
dwellTimeRange	Range of dwell times used to scan the pattern. The per position dwell time will be scaled linearly in this range based on the intensity value of each pixel in the image.
intensity	Defines how pixel intensities are mapped to dwell time: MinimumBlack: black pixels map to dwellTimeRange.begin, white pixels map to dwellTimeRange.end MinimumWhite: white pixels map to dwellTimeRange.begin, black pixels map to dwellTimeRange.end The dwell times for greyscale pixel values is scaled linearly to dwellTimeRange.
maskColor	Optional color value used to mask the image. Any pixel value matching this color will be skipped in the pattern.
lineScanStyle	Line scanning style: Uniform: each line of positions in the rectangle is scanned from left to right. Serpentine: even lines are scanned from left to right, odd lines from right to left.

In images with alpha channels, any pixel with alpha=0 will be skipped and the alpha channel value is used to scale the dwell time.

AggregateScanPattern

The AggregateScanPattern builds a new pattern from a sequence of individual scan patterns. It supports the following functions:

Clear()	Clear the sequence of patterns.
Add()	Add a pattern to the sequence.

B.56 StemSegmentSelection Class

The following selection of segments can be selected:

All

BF

DF

HA

Off

bf1

bf2

bf3

bf4

df1

ha1

ha2

ha3

ha4

ha5

ha6

Appendix C License information

PyPhenom uses several 3rd party libraries and components, for which the license terms are copied below.

Boost

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Ceres Solver

Ceres Solver - A fast non-linear least squares minimizer

Copyright 2015 Google Inc. All rights reserved.

http://ceres-solver.org/

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Eigen

Eigen is partly MPL, partly BSD, and partly LGPL (and partly GPL but we do not use these parts). These licenses are copied verbatim below.

Mozilla Public License Version 2.0

==========

1. Definitions

1.1. "Contributor"
means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

1.2. "Contributor Version"
means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor's Contribution.

1.3. "Contribution"
means Covered Software of a particular Contributor.

1.4. "Covered Software"
means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

1.5. "Incompatible With Secondary Licenses"
means

(a) that the initial Contributor has attached the notice described in Exhibit B to the Covered Software; or

(b) that the Covered Software was made available under the terms of version 1.1 or earlier of the License, but not also under the terms of a Secondary License.

1.6. "Executable Form"
means any form of the work other than Source Code Form.

1.7. "Larger Work"
means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

1.8. "License"
means this document.

1.9. "Licensable"
means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. "Modifications"
means any of the following:

(a) any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or

(b) any new file in Source Code Form that contains any Covered Software.

1.11. "Patent Claims" of a Contributor

means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. "Secondary License"

means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. "Source Code Form"

means the form of the work preferred for making modifications.

1.14. "You" (or "Your")

means an individual or a legal entity exercising rights under this License. For legal entities, "You" includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by such Contributor to use, reproduce, make available,

modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and

(b) under Patent Claims of such Contributor to make, use, sell, offer for sale, have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

(a) for any code that a Contributor has removed from Covered Software; or

(b) for infringements caused by: (i) Your and any other third party's modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or

(c) under Patent Claims infringed by Covered Software in the absence of its Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

(a) such Covered Software must also be made available in Source Code Form, as described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and

(b) You may distribute such Executable Form under the terms of this License, or sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1. The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2. If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3. In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

```
************************************************************************
*                                                                      *
*  6. Disclaimer of Warranty                                           *
*  -------------------------                                           *
*                                                                      *
*  Covered Software is provided under this License on an "as is"       *
*  basis, without warranty of any kind, either expressed, implied, or  *
*  statutory, including, without limitation, warranties that the       *
*  Covered Software is free of defects, merchantable, fit for a        *
*  particular purpose or non-infringing. The entire risk as to the     *
*  quality and performance of the Covered Software is with You.        *
*  Should any Covered Software prove defective in any respect, You     *
*  (not any Contributor) assume the cost of any necessary servicing,   *
*  repair, or correction. This disclaimer of warranty constitutes an   *
*  essential part of this License. No use of any Covered Software is   *
```

* authorized under this License except under this disclaimer. *
* *
****

****
* *
* 7. Limitation of Liability *
* ---------- *
* *
* Under no circumstances and under no legal theory, whether tort *
* (including negligence), contract, or otherwise, shall any *
* Contributor, or anyone who distributes Covered Software as *
* permitted above, be liable to You for any direct, indirect, *
* special, incidental, or consequential damages of any character *
* including, without limitation, damages for lost profits, loss of *
* goodwill, work stoppage, computer failure or malfunction, or any *
* and all other commercial damages or losses, even if such party *
* shall have been informed of the possibility of such damages. This *
* limitation of liability shall not apply to liability for death or *
* personal injury resulting from such party's negligence to the *
* extent applicable law prohibits such limitation. Some *
* jurisdictions do not allow the exclusion or limitation of *
* incidental or consequential damages, so this exclusion and *
* limitation may not apply to You. *
* *
****

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter

shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

> This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain one at http://mozilla.org/MPL/2.0/.

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

You may add additional accurate notices of copyright ownership.

Exhibit B - "Incompatible With Secondary Licenses" Notice

This Source Code Form is "Incompatible With Secondary Licenses", as
defined by the Mozilla Public License, v. 2.0.

BSD license for Eigen:

Copyright (c) 2011, Intel Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LGPL license terms for Eigen:

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those

sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one

of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally

accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library

specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that

everyone can redistribute and change. You can do so by permitting
redistribution under these terms (or, alternatively, under the terms of the
ordinary General Public License).

  To apply these terms, attach the following notices to the library. It is
safest to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

    <one line to give the library's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library `Frob' (a library for tweaking knobs) written by James Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!

Eigen also includes an implementation of MINPACK, for which the license text is:

Minpack Copyright Notice (1999) University of Chicago. All rights reserved

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:
   "This product includes software developed by the University of Chicago, as Operator of Argonne National Laboratory."
   Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. WARRANTY DISCLAIMER. THE SOFTWARE IS SUPPLIED "AS IS" WITHOUT WARRANTY OF ANY KIND. THE COPYRIGHT HOLDER, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF ENERGY, AND THEIR EMPLOYEES: (1) DISCLAIM ANY WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, (2) DO NOT ASSUME ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE ACCURACY, COMPLETENESS, OR USEFULNESS OF THE SOFTWARE, (3) DO NOT REPRESENT THAT USE OF THE SOFTWARE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS, (4) DO NOT WARRANT THAT THE SOFTWARE WILL FUNCTION UNINTERRUPTED, THAT IT IS ERROR-FREE OR THAT ANY ERRORS WILL BE CORRECTED.
5. LIMITATION OF LIABILITY. IN NO EVENT WILL THE COPYRIGHT HOLDER, THE UNITED STATES, THE UNITED STATES DEPARTMENT OF ENERGY, OR THEIR EMPLOYEES: BE LIABLE FOR ANY INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL OR PUNITIVE DAMAGES OF ANY KIND OR NATURE, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS OR LOSS OF DATA, FOR ANY REASON WHATSOEVER, WHETHER SUCH LIABILITY IS ASSERTED ON THE BASIS OF CONTRACT, TORT (INCLUDING NEGLIGENCE OR STRICT LIABILITY), OR OTHERWISE, EVEN IF ANY OF SAID PARTIES HAS BEEN WARNED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGES.

KissFFT

Copyright (c) 2003-2010 Mark Borgerding

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the author nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LibHaru

Copyright (C) 1999-2006 Takeshi Kanno

Copyright (C) 2007-2009 Antony Dovgal

This software is provided 'as-is', without any express or implied warranty.

In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Protobuf

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner of the input file used when generating it. This code is not standalone and requires a support library to be linked with it. This support library is itself covered by the above license.

xraylib

This is xraylib, a library for X-ray matter interactions cross sections for X-ray fluorescence applications: core C library

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE

zlib

Copyright notice:

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.