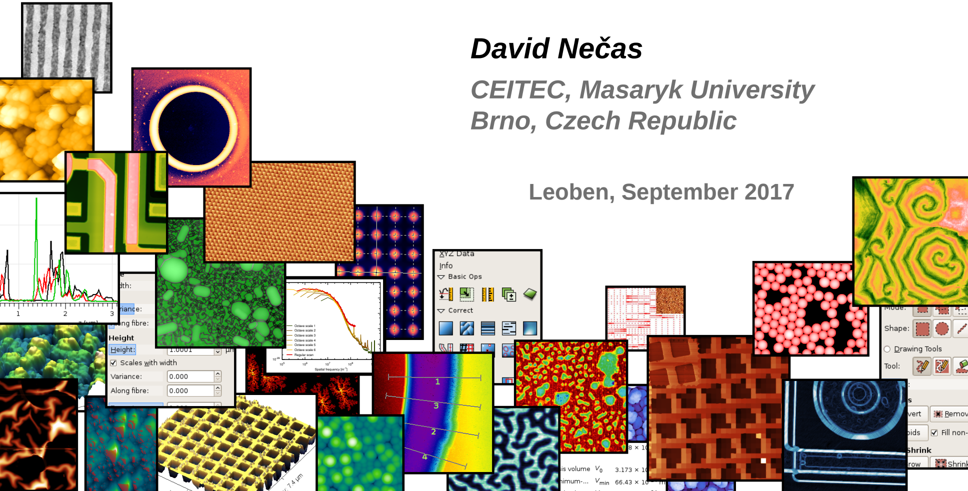
Gwyddion

Open source software for SPM data analsysis

David Nečas

CEITEC, Masaryk University
Brno, Czech Republic

Leoben, September 2017

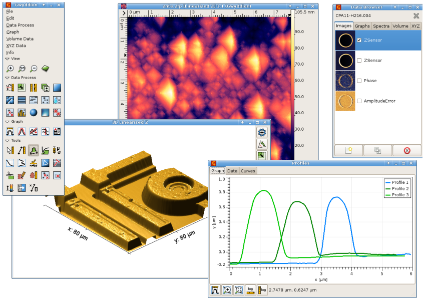


Outline

About Gwyddion

Gwyddion is software for **SPM** (scanning probe microscopy) **data analysis**. A few basic characteristics:

- **Free and Open Source** – it is not only free of charge but the full source code is available, can be studied, modified, redistributed, etc.
- **Multiplatform** – it works on GNU/Linux, Microsoft Windows, Mac OS X and FreeBSD operating systems on common architectures.
- **Modular** – consists mainly of many small pieces that provide individual functions, are rather independent and can be developed independently by third parties.

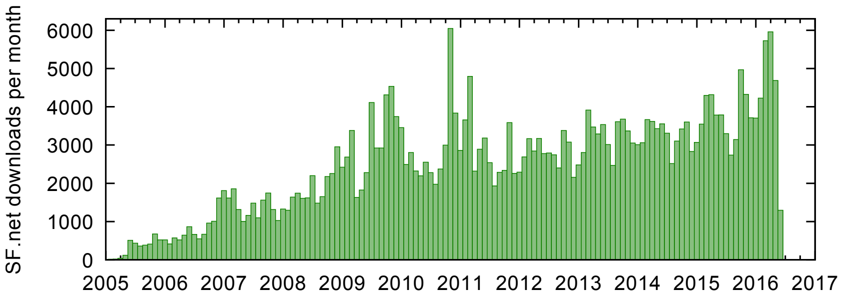


`http://gwyddion.net/`

Gwyddion is mainly focused on **2D raster data** (image) **analysis**. However, it offers also some functions for 1D (curve) and 3D (volume) data.

D. Nečas and P. Klapetek, Gwyddion: An open-source software for SPM data analysis, Cent. Eur. J. Phys. **10** (2012) 181-188, `doi:10.2478/s11534-011-0096-2`

History

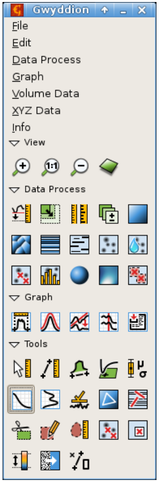


Development started in 2003, originally motivated by the lack of **sufficiently transparent** data processing software for metrology purposes.

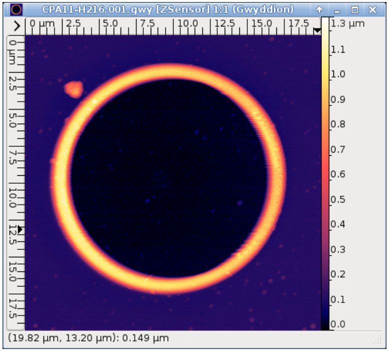
The project was founded by two fresh **Masaryk University** graduates. Nowadays it supported by the **Czech Metrology Institute** and being developed by a diverse group of people from around the world. So far 40+ contributors from 12+ countries have participated in the project.

After a dozen years Gwyddion includes support for more than **120 data formats** and all the typically used processing routines for SPM data processing, as well as many not so typical algorithms.

Overview

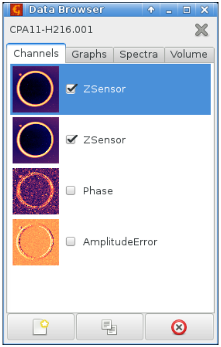


Toolbox


CPA11-H216.001.gwy [ZSensor] 1:1 (Gwyddion)
0 µm 2.5 5.0 7.5 10.0 12.5 15.0 17.5
1.3 µm 1.1 1.0 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.0

(19.82 µm, 13.20 µm): 0.149 µm

Data (image) window



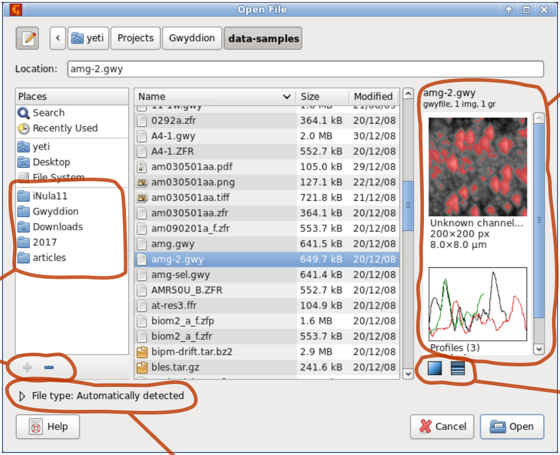
Data browser

What we usually want to do:

1. Open AFM data.
2. Visualise, inspect and explore.
3. Apply levelling and/or corrections.
4. Measure and characterise.
5. Save the results.

Opening AFM files

Opening an AFM file is simple. Just use *File* → *Open* and select the data file(s).



Open File

yeti | Projects | Gwyddion | **data-samples**

Location: amg-2.gwy

Places
Search
Recently Used
yeti
Desktop
File System
iNula11
Gwyddion
Downloads
2017
articles

Name	Size	Modified
0292a.zfr	364.1 kB	20/12/08
A4-1.gwy	2.0 MB	30/12/08
A4-1.ZFR	552.7 kB	20/12/08
am030501aa.pdf	105.0 kB	29/12/08
am030501aa.png	127.1 kB	22/12/08
am030501aa.tiff	721.8 kB	21/12/08
am030501aa.zfr	364.1 kB	20/12/08
am090201a_f.zfr	553.7 kB	20/12/08
amg.gwy	641.5 kB	20/12/08
amg-2.gwy	649.7 kB	20/12/08
amg-sel.gwy	641.4 kB	20/12/08
AMR50U_B.ZFR	552.7 kB	20/12/08
at-res3.ffr	104.9 kB	20/12/08
biom2_a_f.zfp	1.6 MB	20/12/08
biom2_a_f.zfr	553.7 kB	20/12/08
bipm-drift.tar.bz2	2.9 MB	20/12/08
bles.tar.gz	241.6 kB	20/12/08

amg-2.gwy
gwyfile, 1 img, 1 gr

Unknown channel...
200×200 px
8.0×8.0 µm

Profiles (3)

File type: Automatically detected

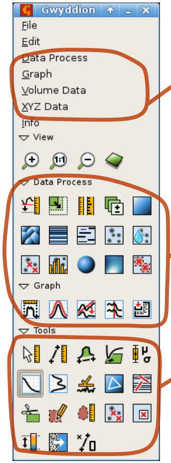
Help | Cancel | Open

Preview of file contents
Quick access to favourite directories
Add/remove here
Line correction and/or plane levelling of image preview

Filters, file type – should stay at *Automatically detected*

Use *File* → *Open Recent* → *Document History* to load files we opened before.

Toolbox



Complete function menu

- ‘set parameters & run’ kind of functions
- organised according to data types
- applied to the active data of corresponding type
- may be insensitive if they require e.g. a mask

Shortcut buttons

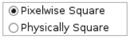
- quicker access to common functions
- subset of functions in the complete menus
- can be customised using *Edit* → *Toolbox*

Tools

- functions you use directly in data windows
- one is active at a time
- each has an associated tool window
- select or measure something interactively

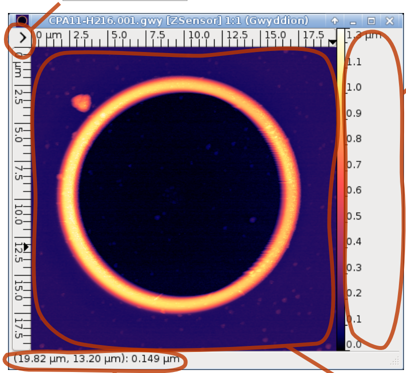
Menus and buttons that are not encircled perform some built-in basic functions: Information, zoom and 3D view, undo & redo, file management, resource settings. Everything else are functions provided by modules – we will see later what it means.

Image window



[✓ Pixelwise Square]
Physically Square

aspect ratio menu

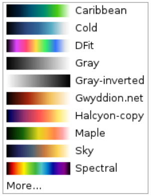


CPA11-H216.001.gwy [ZSensor] 1:1 (Gwyddion)

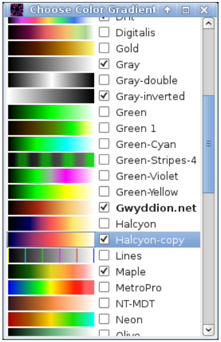
(19.82 µm, 13.20 µm): 0.149 µm

quick position and value information

colour palette menu

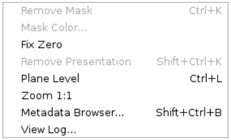


Caribbean
Cold
DFit
Gray
Gray-inverted
Gwyddion.net
Halcyon-copy
Maple
Sky
Spectral
More...



Choose Color Gradient

- [ ] Digitalis
- [ ] Gold
- [x] Gray
- [ ] Gray-double
- [x] Gray-inverted
- [ ] Green
- [ ] Green 1
- [ ] Green-Cyan
- [ ] Green-Stripes-4
- [ ] Green-Violet
- [ ] Green-Yellow
- [x] **Gwyddion.net**
- [ ] Halcyon
- [x] Halcyon-copy
- [ ] Lines
- [x] Maple
- [ ] MetroPro
- [ ] NT-MDT
- [ ] Neon



```
Remove Mask                    Ctrl+K
Mask Color...
Fix Zero
Remove Presentation     Shift+Ctrl+K
Plane Level                    Ctrl+L
Zoom 1:1
Metadata Browser...     Shift+Ctrl+B
View Log...
```

image menu

colour mapping details are controlled by *Colour Range* tool

Right-clicking on the basic data window accesses various common functions and settings.

Data processing

The image functions are in *Data Process* menu.

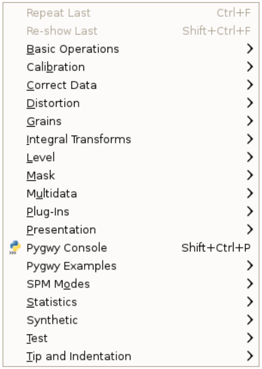
Apply levelling and/or corrections

- mean-plane subtraction *Level* → *Plane Level*
- predominant-plane subtraction *Level* → *Facet-Level*
- three-point plane subtraction *Tools* → *Three-Point Level*
- polynomial levelling *Level* → *Polynomial Background*
- smart base plane flattening *Level* → *Flatten Base*
- row correction *Correct Data* → *Align Rows*

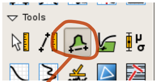
Measure and characterise

- simple measurements *Tools* → *Read Value*, *Distance*
- profile extraction *Tools* → *Profile*
- statistical parameters *Tools* → *Statistical Quantities*
- statistical distributions *Tools* → *Statistical Functions*

Many of these functions are **tools**.

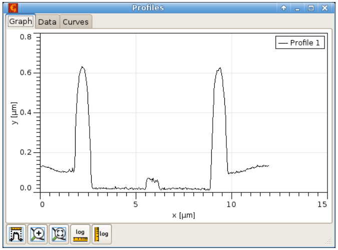


Using tools



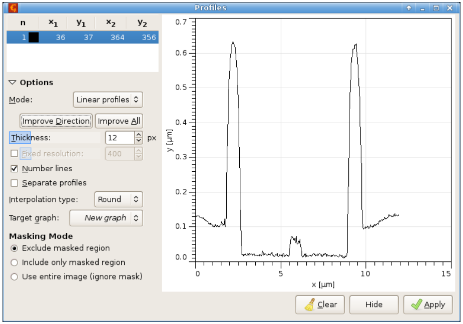
1. Select the Profile tool in the toolbox.

4. pressing Apply creates the resulting profile graph we can use graph functions.



3. Select line(s) in the data window adjust parameters in the tool window.

2. The Profile tool window appears with a preview.



n	x_1	y_1	x_2	y_2
1 ■	36	37	364	356

Options

Mode: Linear profiles

Improve Direction Improve All

Thickness: 12 px

☐ Fixed resolution: 400

☑ Number lines

☐ Separate profiles

Interpolation type: Round

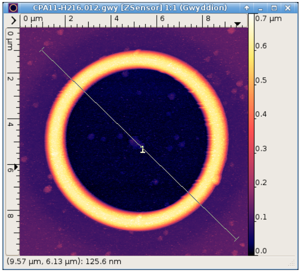
Target graph: New graph

Masking Mode

◉ Exclude masked region

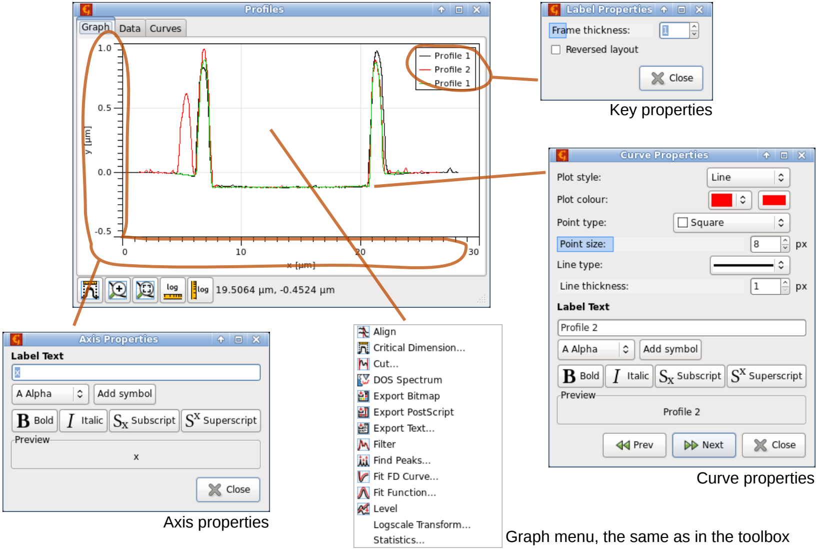
○ Include only masked region

○ Use entire image (ignore mask)



(9.57 µm, 6.13 µm): 125.6 nm

Graph window



19.5064 µm, -0.4524 µm

Frame thickness: 1
Reversed layout

Key properties

Plot style: Line
Plot colour: 🟥
Point type: Square
Point size: 8 px
Line type:
Line thickness: 1 px
Label Text
Profile 2
A Alpha | Add symbol
Bold | Italic | Subscript | Superscript
Preview: Profile 2

Curve properties

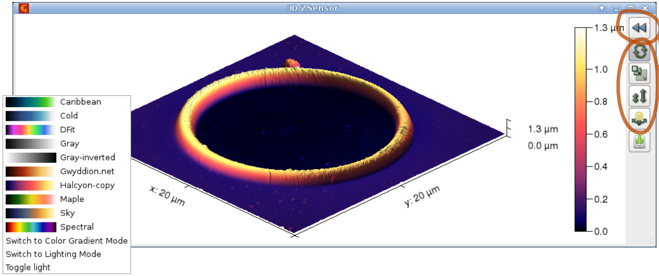
Label Text
x
A Alpha | Add symbol
Bold | Italic | Subscript | Superscript
Preview: x

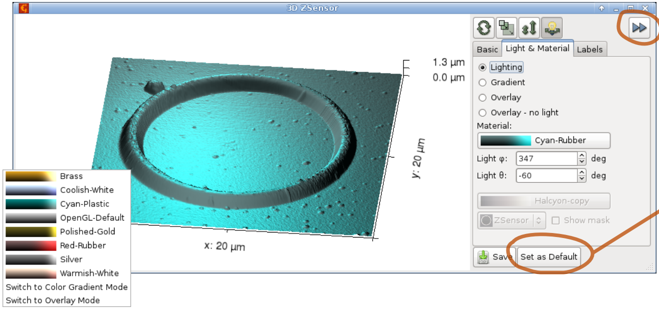
Axis properties

Graph menu, the same as in the toolbox

Curves can be dragged and dropped from *Curves* tab to other graphs.

OpenGL 3D view





● Lighting
○ Gradient
○ Overlay
○ Overlay - no light

Material: Cyan-Rubber

Light φ: 347 deg

Light θ: -60 deg

3D view mouse

- R (rotate)
- S (scale)
- V (value-scale)
- L (light source)

expand/collapse controls

set current view (with all settings) as the default

Saving results

Processed data

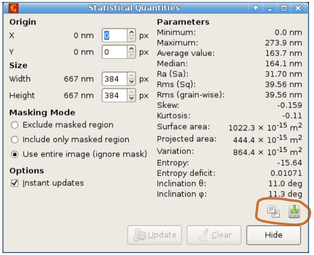
- Always save to **GWY files** (`.gwy`).
  They can contain all data types, selections, . . .

Text export

- Scalar results (statistics or tables) – Copy or Save buttons
- Graph curves – use *Graph* → *Export Text*
- Image data (single) – Save with `.txt` extension

Image export

- Images – Save with an image extension: `.png`, `.pdf`, `.svg`, . . .
- 3D rendering – use the *Save* button
- Ctrl-C copies window contents to the clipboard



Statistical Quantities

Origin

X	0 nm	0	px
Y	0 nm	0	px

Size

Width	667 nm	384	px
Height	667 nm	384	px

Masking Mode

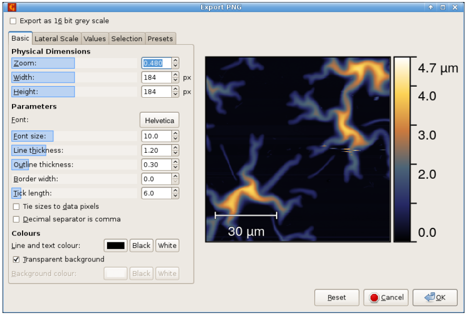
- ○ Exclude masked region
- ○ Include only masked region
- ● Use entire image (ignore mask)

Options

- ☑ Instant updates

Parameters

Minimum:	0.0 nm
Maximum:	273.9 nm
Average value:	163.7 nm
Median:	164.1 nm
Ra (Sa):	31.70 nm
Rms (Sq):	39.56 nm
Rms (grain-wise):	39.56 nm
Skew:	-0.159
Kurtosis:	-0.11
Surface area:	1022.3×10^{-15} m^2
Projected area:	444.4×10^{-15} m^2
Variation:	864.4×10^{-15} m^2
Entropy:	-15.64
Entropy deficit:	0.01071
Inclination θ:	11.0 deg
Inclination φ:	11.3 deg



Export PNG

☐ Export as 16 bit grey scale

Physical Dimensions

Zoom:	0.480	
Width:	184	px
Height:	184	px

Parameters

Font:	Helvetica
Font size:	10.0
Line thickness:	1.20
Outline thickness:	0.30
Border width:	0.0
Tick length:	6.0

- ☐ Tie sizes to data pixels
- ☐ Decimal separator is comma

Colours

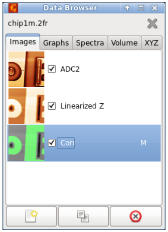
Line and text colour: ■ Black White

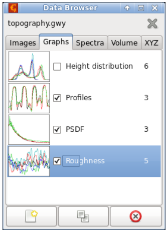
☑ Transparent background

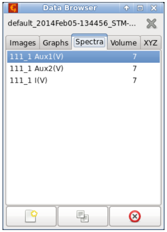
Background colour: Black White

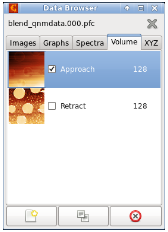
Browsing data

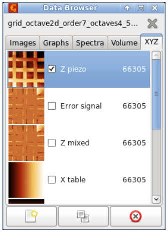
What is the last window shown in the overview? It is called **Data Browser**.






Data Browser
default_2014Feb05-134456_STM-...
Images Graphs Spectra Volume XYZ
111_1 Aux1(V) 7
111_1 Aux2(V) 7
111_1 I(V) 7





What it can do?

- show all the various data in the current file and information about them (M/P/C image flags, number of curves in graphs and spectra, volume data levels)
- show, hide, rename, duplicate and delete individual data
- drag and drop data to another file
- close entire files

If we lose the data browser *Info* → *Show Data Browser* shows it again.

When all data (images, graphs, etc.) of a file are closed the entire file is closed.

When a file is imported from non-GWY format only the first data found in the file is shown.

Data kinds

Images (height fields, height maps, data fields) – the main kind of data in Gwyddion. Most functions work with images. Images can have various other kinds of data associated with them: masks, presentations, selections, . . .

Graphs – usually the result of some data reduction, but can also be loaded directly. A bunch of functions is available in the *Graph* menu.

Single point spectra (SPS) – F-D, I-V or other curves measured at specific coordinates. The cannot exist standalone in Gwyddion; must be merged with the corresponding image files. The *Spectro* tool extracts SPS to graph curves.

Volume data – usually spectra measured in each image point. A few basic operations are available, together with several functions to fit F-D volume data or extract data of lower dimensionality.

XYZ data – data fields measured in irregular point sets. The newest addition so far; only a few functions are available.

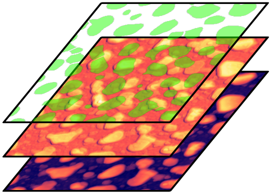
Missing: It seems XYZ spectra, i.e. curves measured in irregular point sets are the future in SPM. However, taming this beast is tough.

Masks, presentations and selections

While correcting or analysing images, we often make use of auxiliary data.

Three important kinds of auxiliary data that can be attached to an image in Gwyddion are masks, presentations and selections.

The first two are images; the last are geometrical shapes ‘drawn’ on the data with mouse.

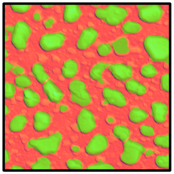


mask of marked grains

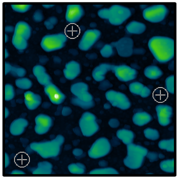
shading presentation

underlying data

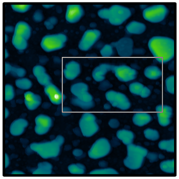
all data processing operations use the data



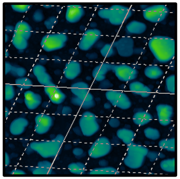
composed data view



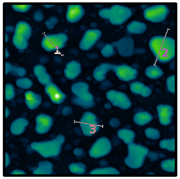
points (with radii)



rectangle



lattice



lines (with widths and numbers)

Metadata and logs

The last auxiliary pieces of information we mention are metadata and logs.

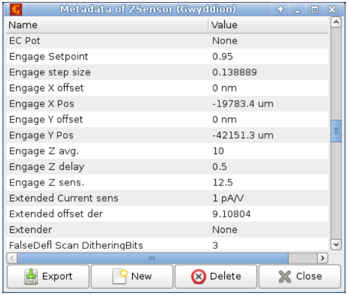
Metadata

- any additional parameters or information we can read from the SPM file
- there can be none at all, or hundreds of structured items
- entirely instrument and file format specific
- editable – we can add our own items (or even change existing)

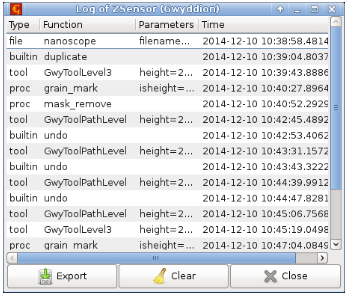
Log

- the record of all performed operations
- contains name of the operation, parameters and time
- can be removed and/or disabled if we do not like it

Both are accessible from the right-click menu of images and volume data (graphs and spectra do not have them).



Name	Value
EC Pot	None
Engage Setpoint	0.95
Engage step size	0.138889
Engage X offset	0 nm
Engage X Pos	-19783.4 um
Engage Y offset	0 nm
Engage Y Pos	-42151.3 um
Engage Z avg.	10
Engage Z delay	0.5
Engage Z sens.	12.5
Extended Current sens	1 pA/V
Extended offset der	9.10804
Extender	None
FalseDefl Scan DitheringBits	3

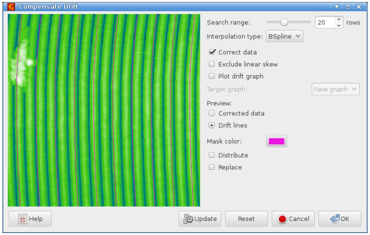


Type	Function	Parameters	Time
file	nanoscope	filename...	2014-12-10 10:38:58.4814
builtin	duplicate		2014-12-10 10:39:04.8037
tool	GwyToolLevel3	height=2...	2014-12-10 10:39:43.8886
proc	grain_mark	isheight=...	2014-12-10 10:40:27.8964
proc	mask_remove		2014-12-10 10:40:52.2929
tool	GwyToolPathLevel	height=2...	2014-12-10 10:42:45.4892
builtin	undo		2014-12-10 10:42:53.4062
tool	GwyToolPathLevel	height=2...	2014-12-10 10:43:31.1572
builtin	undo		2014-12-10 10:43:43.3222
tool	GwyToolPathLevel	height=2...	2014-12-10 10:44:39.9912
builtin	undo		2014-12-10 10:44:47.8281
tool	GwyToolPathLevel	height=2...	2014-12-10 10:45:06.7568
tool	GwyToolLevel3	height=2...	2014-12-10 10:45:19.0498
proc	grain_mark	isheight=...	2014-12-10 10:47:04.0849

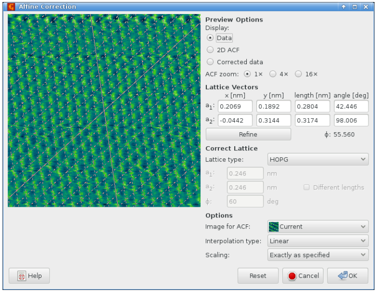
Common features

Standard SPM data processing features:

- 2D and 3D views and graphs
- value and profile reading
- geometrical transforms
- image merging
- line levelling
- plane and polynomial levelling
- three-point levelling
- denoising filters
- statistical characterisation
- roughness measurement
- slope distribution
- grain marking and measurement
- Fourier transform
- wavelet transform
- edge detection
- . . .



Search range: 20 rows
Interpolation type: BSpline
[✓] Correct data
[ ] Exclude linear skew
[ ] Plot drift graph
Target graph: New graph
Preview:
( ) Corrected data
(●) Drift lines
Mask color: 🟪
[ ] Distribute
[ ] Replace



Preview Options
Display:
(●) Data
( ) 2D ACF
( ) Corrected data
ACF zoom: (●) 1× ( ) 4× ( ) 16×

Lattice Vectors

	x [nm]	y [nm]	length [nm]	angle [deg]
a_1:	0.2069	0.1892	0.2804	42.446
a_2:	-0.0442	0.3144	0.3174	98.006

Refine ϕ: 55.560

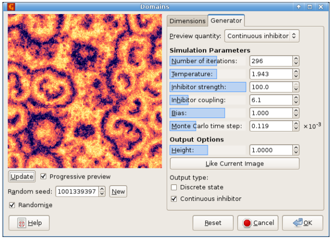
Correct Lattice
Lattice type: HOPG
a_1: 0.246 nm
a_2: 0.246 nm [ ] Different lengths
ϕ: 60 deg

Options
Image for ACF: Current
Interpolation type: Linear
Scaling: Exactly as specified

Uncommon features

Some uncommon and odd features:

- processing of data under arbitrarily shaped masks, including:
  - value reading
  - profile extraction
  - statistical quantities
  - value & slope distributions
  - ACF, HHCF, PSDF, . . .
  - plane & poly levelling
  - filtering
- artificial data generation
  - frequency space synthesis
  - deposition models
  - regular patterns
  - Ising-like models
  - various noise types
- feedback loop simulation
- lateral force simulation
- tip modelling and blind estimation
- tip convolution and erosion



Preview quantity: Continuous inhibitor

Simulation Parameters

Number of iterations:	296
Temperature:	1.943
Inhibitor strength:	100.0
Inhibitor coupling:	6.1
Bias:	1.000
Monte Carlo time step:	0.119 $\times 10^{-3}$

Output Options

Height: 1.0000

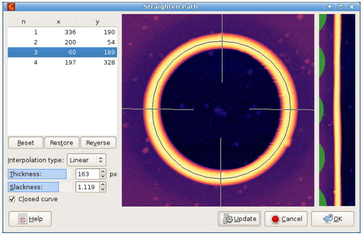
Like Current Image

Output type:
- [ ] Discrete state
- [✓] Continuous inhibitor

Update [✓] Progressive preview

Random seed: 1001339397 New

[✓] Randomise



n	x	y
1	336	190
2	200	54
3	60	189
4	197	328

Interpolation type: Linear

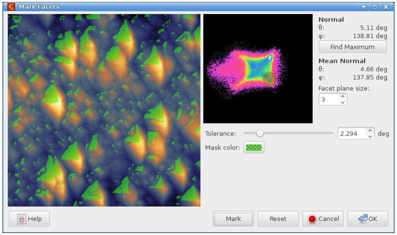
Thickness: 163 px

Slackness: 1.119

[✓] Closed curve

Uncommon features (continued)

- various curious statistics:
  - Minkowski functionals
  - entropy
  - area scale graph
- angularly averaged profiles
- affine distortion correction
- drift correction
- lattice vector measurement
- calibration and uncertainty calculation
- force-distance curve fitting
- volume spectra processing
- XYZ data processing
- facet analysis
- correlation search, cross-correlation
- MFM modelling
- point spread function estimation
- neural network data processing
- . . .



Mark Facets

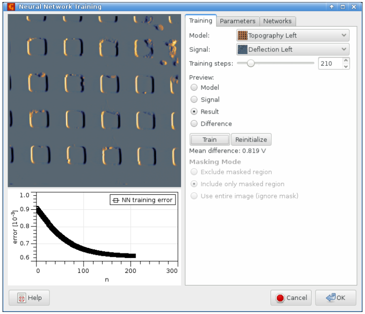
Normal
θ: 5.11 deg
φ: 138.81 deg

Mean Normal
θ: 4.66 deg
φ: 137.85 deg

Facet plane size: 3

Tolerance: 2.294 deg

Mask color: 🟩



Neural Network Training

Model: Topography Left
Signal: Deflection Left
Training steps: 210

Preview:
- ○ Model
- ○ Signal
- ● Result
- ○ Difference

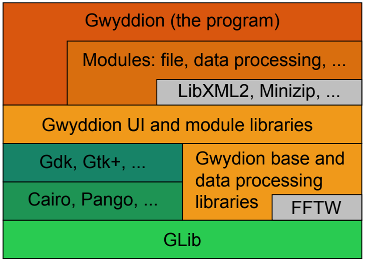
Mean difference: 0.819 V

Masking Mode
- ○ Exclude masked region
- ● Include only masked region
- ○ Use entire image (ignore mask)

Basic structure

Main Gwyddion components:

- **Libraries**, providing basic and advanced data processing routines, graphical user interface elements and other utility functions and objects.
- **The program**, quite small and simple, serving primarily as a glue connecting the other components together in a common graphical interface.
- **Modules**, technically run-time loaded libraries, that provide actual user-facing functionality. They are well-separated and can be developed by third parties. They often extensively use library methods.


Gwyddion (the program)
Modules: file, data processing, ...
LibXML2, Minizip, ...
Gwyddion UI and module libraries
Gdk, Gtk+, ...
Cairo, Pango, ...
Gwydion base and data processing libraries
FFTW
GLib

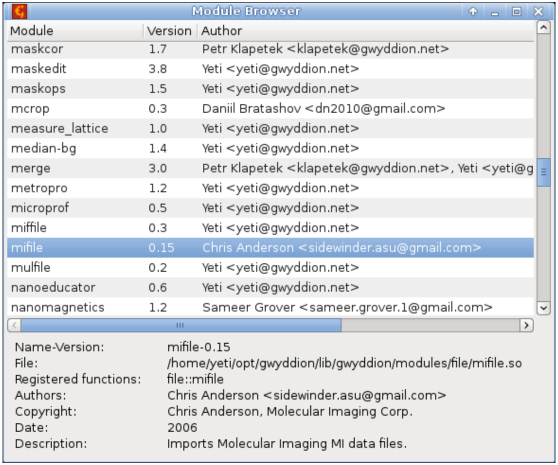
Gwyddion relies heavily on a set of open source libraries from the **GTK+ stack**:

- the GLib system library for portability, GObject object system and various utility functions
- GTK+ toolkit for the graphical user interface (while adding a fair number of new widgets for data visualisation)

Plus FFTW is used for FFT, compression and file format libraries for file handling, . . .

Module information

```
modules
├── pygwy.so
├── file
│   ├── apedaxfile.so
│   ├── gwyfile.so
│   ├── iso28600.so
│   ├── nanoscope.so
│   └── ...
├── process
│   ├── arithmetic.so
│   ├── basicops.so
│   ├── level.so
│   ├── otsu_threshold.so
│   └── ...
├── graph
│   └── ...
├── tool
│   └── ...
└── volume
    └── ...
```



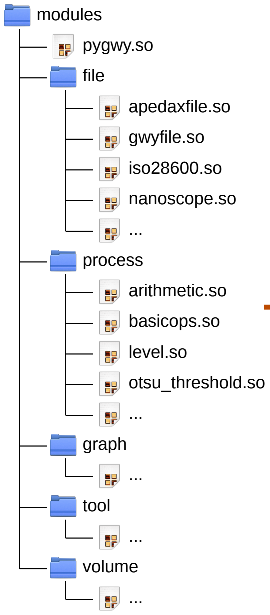
Module	Version	Author
maskcor	1.7	Petr Klapetek <klapetek@gwyddion.net>
maskedit	3.8	Yeti <yeti@gwyddion.net>
maskops	1.5	Yeti <yeti@gwyddion.net>
mcrop	0.3	Daniil Bratashov <dn2010@gmail.com>
measure_lattice	1.0	Yeti <yeti@gwyddion.net>
median-bg	1.4	Yeti <yeti@gwyddion.net>
merge	3.0	Petr Klapetek <klapetek@gwyddion.net>, Yeti <yeti@g
metropro	1.2	Yeti <yeti@gwyddion.net>
microprof	0.5	Yeti <yeti@gwyddion.net>
miffile	0.3	Yeti <yeti@gwyddion.net>
mifile	0.15	Chris Anderson <sidewinder.asu@gmail.com>
mulfile	0.2	Yeti <yeti@gwyddion.net>
nanoeducator	0.6	Yeti <yeti@gwyddion.net>
nanomagnetics	1.2	Sameer Grover <sameer.grover.1@gmail.com>

Name-Version:	mifile-0.15
File:	/home/yeti/opt/gwyddion/lib/gwyddion/modules/file/mifile.so
Registered functions:	file::mifile
Authors:	Chris Anderson <sidewinder.asu@gmail.com>
Copyright:	Chris Anderson, Molecular Imaging Corp.
Date:	2006
Description:	Imports Molecular Imaging MI data files.

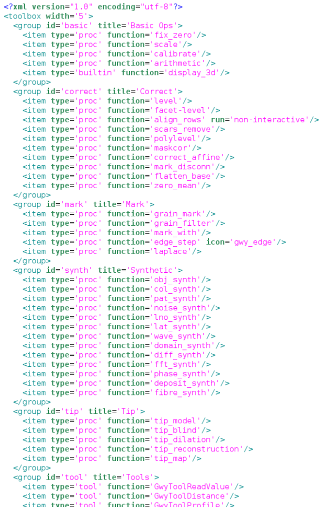
We can see the list of modules in *Info* → *Module Browser*.

There is approximately a bazillion of them because essentially every function or data format is provided by its own module.

Constructing the toolbox



+

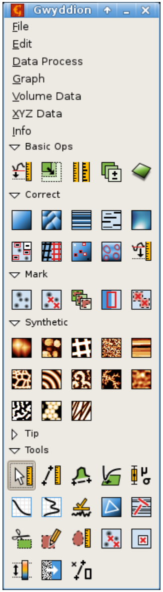


```
<?xml version="1.0" encoding="utf-8"?>
<toolbox width='5'>
  <group id='basic' title='Basic Ops'>
    <item type='proc' function='fix_zero'/>
    <item type='proc' function='scale'/>
    <item type='proc' function='calibrate'/>
    <item type='proc' function='arithmetic'/>
    <item type='builtin' function='display_3d'/>
  </group>
  <group id='correct' title='Correct'>
    <item type='proc' function='level'/>
    <item type='proc' function='facet-level'/>
    <item type='proc' function='align_rows' run='non-interactive'/>
    <item type='proc' function='scars_remove'/>
    <item type='proc' function='polylevel'/>
    <item type='proc' function='maskcor'/>
    <item type='proc' function='correct_affine'/>
    <item type='proc' function='mark_disconn'/>
    <item type='proc' function='flatten_base'/>
    <item type='proc' function='zero_mean'/>
  </group>
  <group id='mark' title='Mark'>
    <item type='proc' function='grain_mark'/>
    <item type='proc' function='grain_filter'/>
    <item type='proc' function='mark_with'/>
    <item type='proc' function='edge_step' icon='gwy_edge'/>
    <item type='proc' function='laplace'/>
  </group>
  <group id='synth' title='Synthetic'>
    <item type='proc' function='obj_synth'/>
    <item type='proc' function='col_synth'/>
    <item type='proc' function='pat_synth'/>
    <item type='proc' function='noise_synth'/>
    <item type='proc' function='lno_synth'/>
    <item type='proc' function='lat_synth'/>
    <item type='proc' function='wave_synth'/>
    <item type='proc' function='domain_synth'/>
    <item type='proc' function='diff_synth'/>
    <item type='proc' function='fft_synth'/>
    <item type='proc' function='phase_synth'/>
    <item type='proc' function='deposit_synth'/>
    <item type='proc' function='fibre_synth'/>
  </group>
  <group id='tip' title='Tip'>
    <item type='proc' function='tip_model'/>
    <item type='proc' function='tip_blind'/>
    <item type='proc' function='tip_dilation'/>
    <item type='proc' function='tip_reconstruction'/>
    <item type='proc' function='tip_map'/>
  </group>
  <group id='tool' title='Tools'>
    <item type='tool' function='GwyToolReadValue'/>
    <item type='tool' function='GwyToolDistance'/>
    <item type='tool' function='GwyToolProfile'/>
```

custom ui/toolbox.xml

(you do not actually need to edit some XML files, just use *Edit* → *Toolbox*)

=



custom toolbox

User directory

Where the `ui/toolbox.xml` file lives?

In the Gwyddion user directory:

- `~/.gwyddion` in Unix systems
- *`Some`*`\`*`Kind`*`\`*`Of`*`\`*`User`*`\`*`Directory`*`\gwyddion` in MS Windows

It contains various things:

- definitions of toolbox and keyboard shortcuts (in `ui`)
- `settings`, i.e. remembered values of all adjustable parameters
- resources, such as user false colour gradients
- presets, i.e. named parameter sets for instance for image export
- ...

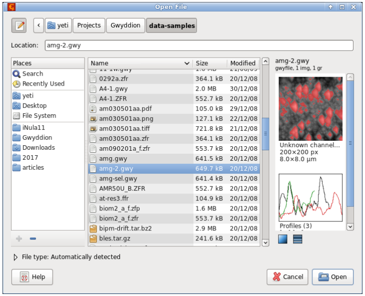
This can be useful when we want to send some of these thing to a colleague or transfer them to a new computer.

Opening a file

One customization possibility is writing new modules. This is an specialised topic. We will just demonstrate how modularity works.

What happens when we open a file?

- The program calls function `gwy_file_open()` for the file.
- All 120+ file modules are asked: *Can you load this file?*
- Each file module examines the file contents and returns a *score*.
- Higher score means the module is more confident it can load this file.
- A winner is chosen.
- This module is finally asked to load the file.
- It loads all the importable data and returns something called *GwyContainer*.
- From this point everything is as if we opened a Gwyddion GWY file.
- In fact, this is how we load GWY files too.

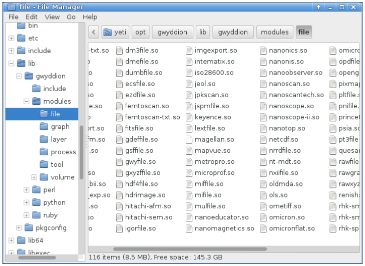


Location: amg-2.gwy

Name	Size	Modified
0292a.zfr	364.1 kB	20/12/08
A4-1.gwy	2.0 MB	30/12/08
A4-1.ZFR	552.7 kB	20/12/08
am030501aa.pdf	105.0 kB	29/12/08
am030501aa.png	127.1 kB	22/12/08
am030501aa.tiff	721.8 kB	21/12/08
am030501aa.zfr	364.1 kB	20/12/08
am090201a_f.zfr	553.7 kB	20/12/08
amg.gwy	641.5 kB	20/12/08
amg-2.gwy	649.7 kB	20/12/08
amg-sel.gwy	641.4 kB	20/12/08
AMR50U_B.ZFR	552.7 kB	20/12/08
at-res3.ffr	104.9 kB	20/12/08
biom2_a_f.zfp	1.6 MB	20/12/08
biom2_a_f.zfr	553.7 kB	20/12/08
bipm-drift.tar.bz2	2.9 MB	20/12/08
bles.tar.gz	241.6 kB	20/12/08

amg-2.gwy
gwyfile, 1 img, 1 gr
Unknown channel...
200×200 px
8.0×8.0 µm
Profiles (3)

File type: Automatically detected



```
-txt.so   dm3file.so         imgexport.so       nanonics.so        omicro
          dmefile.so         intematix.so       nanonis.so         opdfile
o         dumbfile.so        iso28600.so        nanoobserver.so    openg
so        ecsfile.so         jeol.so            nanoscan.so        pixma
so        ezdfile.so         jpkscan.so         nanoscantech.so    pltfile.
e.so      femtoscan.so       jspmfile.so        nanoscope.so       pnifile.
          femtoscan-txt.so   keyence.so         nanoscope-ii.so    prince
rt.so     fitsfile.so        lextfile.so        nanotop.so         psia.s
fm.so     gdeffile.so        magellan.so        netcdf.so          pt3file
.so       gsffile.so         mapvue.so          nrrdfile.so        quesa
          gwyfile.so         metropro.so        nt-mdt.so          rawfile
so        gxyzffile.so       microprof.so       nxiifile.so        rawgra
bii.so    hdf4file.so        miffile.so         oldmda.so          rawxyz
exp.so    hdrimage.so        mifile.so          ols.so             renish
.so       hitachi-afm.so     mulfile.so         ometiff.so         rhk-sm
so        hitachi-sem.so     nanoeducator.so    omicron.so         rhk-sm
o         igorfile.so        nanomagnetics.so   omicronflat.so     rhk-sp
```

116 items (8.5 MB), Free space: 145.3 GB

What is in a GWY file?

The data browser shows the content of a GWY file as some images, spectra, graphs, etc.

We know the file can also contain masks, presentations, selections, metadata, false colour mapping settings, 3D view settings, logs, etc.

Actually, we might not talked about some of these but they are there anyway…

When we save data to a GWY file and open it later, all these things are restored.

Understanding the data organisation is also very useful for scripting.

So what is really inside?

At the top level, a GWY file consists of something called *GwyContainer*.

It contains all the various pieces of data. They are identified by names looking exactly like (Unix) file paths.

GWY files are binary. However, there is a useful program *gwydump* that can print their contents in a human-readable form.

```
gwydump -v -t -d 1 file.gwy
Header GWYP
"" object=GwyContainer
    "/0/data" object=GwyDataField
    "/0/data/log" object=GwyStringList
    "/0/data/title" string="Topography Left"
    "/0/data/visible" boolean=TRUE
    "/0/mask" object=GwyDataField
    "/0/mask/alpha" double=0.499992
    "/0/mask/blue" double=0.0321813
    "/0/mask/green" double=1
    "/0/mask/red" double=0
    "/0/meta" object=GwyContainer
    "/0/select/pointer" object=GwySelectionPoint
    "/1/data" object=GwyDataField
    "/1/data/log" object=GwyStringList
    "/1/data/title" string="Topography Left"
    "/1/data/visible" boolean=TRUE
    "/1/mask" object=GwyDataField
    "/1/mask/alpha" double=0.499992
    "/1/mask/blue" double=0.0321813
    "/1/mask/green" double=1
    "/1/mask/red" double=0
    "/1/meta" object=GwyContainer
    "/1/select/pointer" object=GwySelectionPoint
    "/1/select/rectangle" object=GwySelectionRectangle
```

What is in a GWY file?

The data for the first image is called `"/0/data"`.

The data for the second image is called `"/1/data"`.

The mask for the first image is called `"/0/mask"`.

Some of the naming rules may seem odd. The reason is compatibility. Somewhat odd data names are better than breaking compatibility with older versions.

If we look in more detail, again using *gwydump*, we can see that an image has again some named components. And some of the components have their own components...

Altogether, a GWY file is a tree.

Unlike the coarse view, this level of detail is seldom useful, even in scripting.

If we want to read or write GWY files in a standalone program, there is a library for that – *libgwyfile*.

```
gwydump -v -t file.gwy
Header GWYP
"" object=GwyContainer
    "/0/data" object=GwyDataField
        "xres" int32=256
        "yres" int32=255
        "xreal" double=5e-05
        "yreal" double=4.98047e-05
        "si_unit_xy" object=GwySIUnit
            "unitstr" string="m"
        "si_unit_z" object=GwySIUnit
            "unitstr" string="m"
        "data" double array of length 65280
    "/0/data/log" object=GwyStringList
        "strings" string array of length 8
    "/0/data/title" string="Topography Left"
    "/0/data/visible" boolean=TRUE
    "/0/mask" object=GwyDataField
        "xres" int32=256
        "yres" int32=255
        "xreal" double=5e-05
        "yreal" double=4.98047e-05
        "si_unit_xy" object=GwySIUnit
            "unitstr" string="m"
        "si_unit_z" object=GwySIUnit
            "unitstr" string=""
        "data" double array of length 65280
    "/0/mask/alpha" double=0.499992
    "/0/mask/blue" double=0.0321813
    "/0/mask/green" double=1
    "/0/mask/red" double=0
    "/0/meta" object=GwyContainer
        "AcAmpl" string="2"
        "AcEnable" string="False"
        "AcFreq" string="328000"
        "Angle" string="0"
        "Mode" string="Contact"
        "SampBias" string="5.960465E-07"
        "SetPoint" string="0.8000305"
        ...
```

Pygwy

Python scripting, called **pygwy** for short – it is also the name of the Gwyddion module which adds the Python scripting capability.

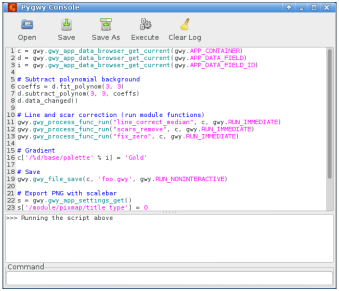
In fact, it can refer to three different things.

Pygwy console – allows running Python scripts inside Gwyddion, executing various program functions, mixing manual data processing and automation.

Standalone scripts – Python programs that run outside Gwyddion, `import gwy` and utilise its functions to process SPM data.

Loading a file can be as simple as:

```
import gwy
data = gwy.gwy_file_load('somefile.afm')
```



```
c = gwy.gwy_app_data_browser_get_current(gwy.APP_CONTAINER)
d = gwy.gwy_app_data_browser_get_current(gwy.APP_DATA_FIELD)
i = gwy.gwy_app_data_browser_get_current(gwy.APP_DATA_FIELD_ID)

# Subtract polynomial background
coeffs = d.fit_polynom(3, 3)
d.subtract_polynom(3, 3, coeffs)
d.data_changed()

# Line and scar correction (run module functions)
gwy.gwy_process_func_run("line_correct_median", c, gwy.RUN_IMMEDIATE)
gwy.gwy_process_func_run("scars_remove", c, gwy.RUN_IMMEDIATE)
gwy.gwy_process_func_run("fix_zero", c, gwy.RUN_IMMEDIATE)

# Gradient
c['/%d/base/palette' % i] = 'Gold'

# Save
gwy.gwy_file_save(c, 'foo.gwy', gwy.RUN_NONINTERACTIVE)

# Export PNG with scalebar
s = gwy.gwy_app_settings_get()
s['/module/pixmap/title type'] = 0
```

```
>>> Running the script above
```

Pygwy console with a script

Of course, we are now running Python, not Gwyddion. So while we still have lots of functions available, some things only make sense only inside Gwyddion itself.

The last option is to write **Gwyddion modules in Python**. This is a bit more involved – nevertheless simpler than in C (which is the norm).

Using Gwyddion functions

When doing any of the above things, we must distinguish two styles (shown as C code for a change but it translates directly to Python):

Using library functions – happily poking in our images or other data, not caring in the least about Gwyddion, the program:

```
GwyDataField *datafield = gwy_data_field_new(400, 400, 3e-6, 3e-6, FALSE);
gwy_data_field_fill(datafield, 3.0);
```

Pretending to be Gwyddion – running functions from modules and generally managing data like Gwyddion does.

This may be the only way to utilise some functionality provided by modules.

However, it is less straightforward:

```
...
gwy_container_set_enum_by_name(settings, "/module/facet-level/mode",
                               GWY_MASK_EXCLUDE);
gwy_app_data_browser_select_data_field(data, id);
gwy_app_run_process_func_in_mode("facet-level", GWY_RUN_IMMEDIATE);
...
```

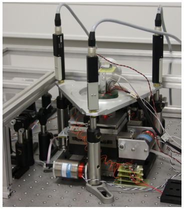
With some care we can mix and match the two styles.

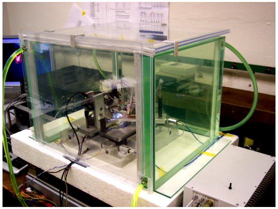
Driving custom built instrumentation

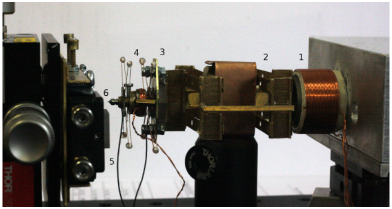
We can employ Gwyddion's data format handling, data processing, visualisation and other utility functions in other tasks.

Instrumentation development at CMI & CEITEC:

- voice coil based large area SPM instruments
- nanoindentation devices
- spectroscopic digital reflectometry
- special metrology SPM system (with ISI ASCR)
- large area low cost system (with ISI ASCR)



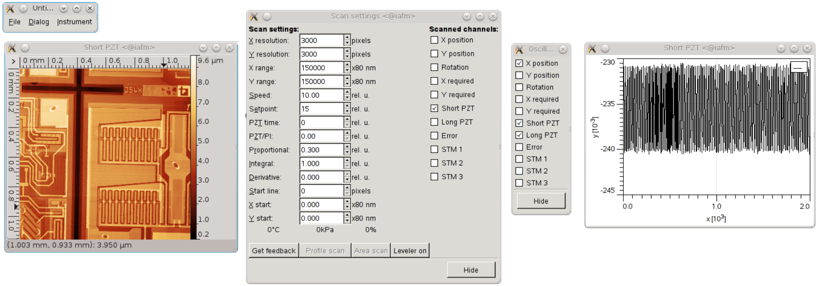




Driving custom built instrumentation

GwyScan is a simple SPM control software based on Gwyddion libraries, compatible with our voice coil, piezoelectric, stick slip and PiezoWalk based SPMs.

All the hardware related functionality is only a single file with less than 1000 lines of code.



Scan settings <@iafm>

Scan settings:

X resolution:	3000	pixels
Y resolution:	3000	pixels
X range:	150000	x80 nm
Y range:	150000	x80 nm
Speed:	10.00	rel. u.
Setpoint:	15	rel. u.
PZT time:	0	rel. u.
PZT/PI:	0.00	rel. u.
Proportional:	0.300	rel. u.
Integral:	1.000	rel. u.
Derivative:	0.000	rel. u.
Start line:	0	pixels
X start:	0.000	x80 nm
Y start:	0.000	x80 nm

0°C 0kPa 0%

Scanned channels:

- ☐ X position
- ☐ Y position
- ☐ Rotation
- ☐ X required
- ☐ Y required
- ☑ Short PZT
- ☐ Long PZT
- ☐ Error
- ☐ STM 1
- ☐ STM 2
- ☐ STM 3

Get feedback | Profile scan | Area scan | Leveler on | Hide

Oscill...

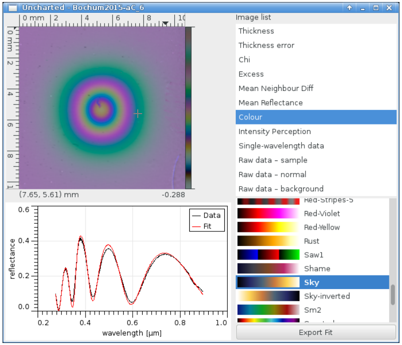
- ☑ X position
- ☐ Y position
- ☐ Rotation
- ☐ X required
- ☐ Y required
- ☑ Short PZT
- ☑ Long PZT
- ☐ Error
- ☐ STM 1
- ☐ STM 2
- ☐ STM 3

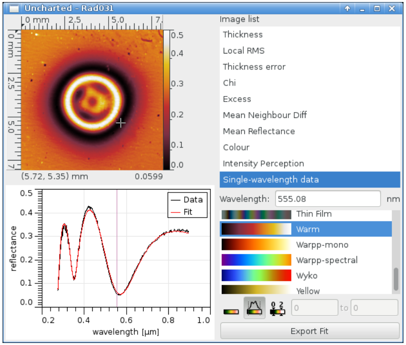
This approach has various

Benefits: Lightweight libraries, most data handling in the application, versatile system applicable virtually to anything, open-source licensing.

Drawbacks: No direct support for data acquisition (like in Labview), no direct support for any SPM instrumentation related tasks (like in GXSM).

Something completely different





Uncharted is a **imaging spectrophotometry** data visualisation software used in our CEITEC group.

It uses Gwyddion widgets for visualisation.

The data are fitted using another Gwyddion-based program (command line, nothing to show here).

Parameter maps are written in the Gwyddion GWY file format.

Conclusion

Conclusion:

- Use Gwyddion for your SPM data analysis.

Thank you!

Ackowledgements
Czech Metrology Institute
Masaryk University

CEITEC
Central European Institute of Technology
BRNO | CZECH REPUBLIC

Gwyddion contributors
Christopher Anderson, Vinicius Barboza, François Bianco, Jindřich Bílek, Даниил Браташов, Christian Bühler, Matthew Caldwell, Anna Campbellova, Owain Davies, Livia Della Seta, Lennart Fricke, Gianfranco Gallizia, Petr Grolich, Sameer Grover, Андрей Груздев, Martin Hasoň, Jan Hořák, Lukáš Chvátal, Jeong Jiseong, Dirk Kähler, Antony Kikaxa, Petr Klapetek, Felix Kling, Александр Ковалев, Philipp Leufke, Niv Levy, Fellype do Nascimento, David Nečas, Sven Neumann, Nenad Ocelic, Andrés Muñiz Piniella, Philipp Rahe, Евгений Рябов, François Riguet, Johannes Römer, Vojtěch Salajka, Jeffrey J. Schwartz, Luke Somers, Martin Šiler, Miroslav Valtr, Jozef Veselý, Rolf Würdemann, Samo Ziberna, Rok Zitko.