Gwyddion

Open source software for SPM data analysis

Outline

- history, developers and development scheme
- program core and architecture
- modules, tools and plugins
- data processing modules and tools
- advanced statistical functions
- Pygwy scripting interface

Outline

Development started in 2003, formerly as part of unrealized project of NANOMET group joining European metrology institutes working on the field of nanometrology.

Due to lack of software that would be transparent enough, CMI started developement in a small group (Petr Klapetek, David Nečas), that was extended by many other developers in next years.

Gwyddion

Open source software for SPM data analysis

Gwyddion works on GNU/Linux, Microsoft Windows, Mac OS X and FreeBSD operating systems on common architectures, all systems can be used also for developement. Its graphical user interface is based on Gtk+ and port to other systems supported by Gtk+ should be possible.

Gwyddion is Free and Open Source software, covered by GNU General Public License. It aims to provide multiplatform modular program for 2D data analysis that could be easily extended by modules and plug-ins.

Basic structure

Gwyddion relies on the GLib utility library for portability and uses GLib object system GObject for its own objects. Graphical user interface is implemented with the Gtk+ toolkit, with a fair amount of Gwyddion specific extension widgets.

Gwyddion can be divided into four main components:

1. **libraries**, providing basic and advanced data processing routines, graphical user inreface elements and other utility functions and objects,
2. **the application**, quite small and simple, serving primarily as a glue connecting the other components together in a common graphical interface,
3. **modules**, technically run-time loaded libraries, that provide most of the actual functionality and present it to the user, they often extensively use library methods,
4. **plug-ins**, standalone programs that are more independent of Gwyddion than modules, both technically and legally.

Basic structure

The **libgwyddion** library defines some core interfaces, like GwySerializable for data-like objects, GwyContainer, GwySIUnit etc.

The **libprocess** library defines two basic objects: GwyDataField. representing two-dimensional data and GwyDataLine, representing one-dimensional data. There are many process and analysis functions implemented for these objects.

The **libdraw** library provides colour handling and elementary data rendering functions (gradients, selections).

The **libgwydgets** library is a collection of Gwyddion-specific Gtk+ widgets, like GwyDataView, GwyDataWindow, GwyGraph

The **libgwymodule** library deals with module administrative, loading and act as a proxy in their usage.

The **libgwyapp** library contains main application related functions (loading, saving, etc.).

Basic structure

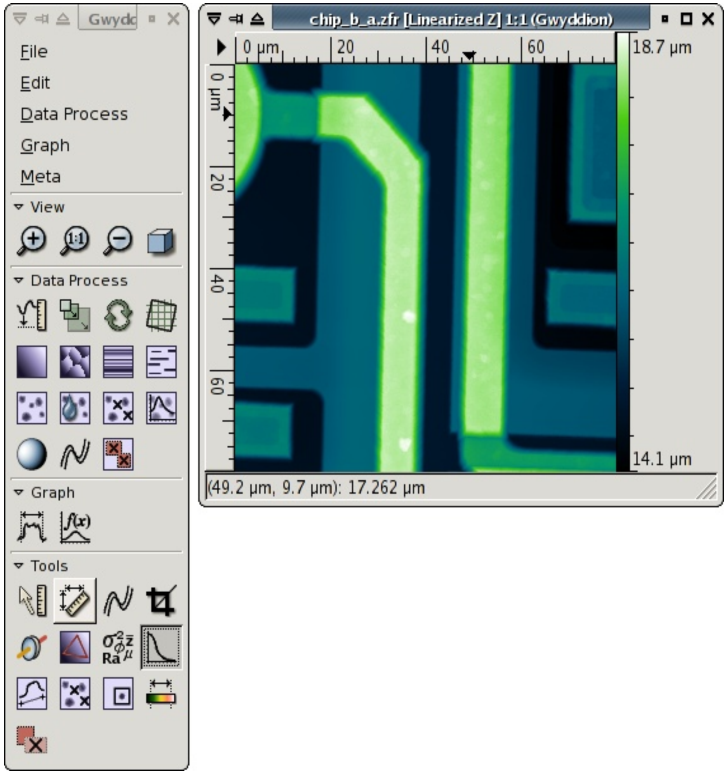
- **data processing modules** provide functions for processing of two-dimensional data arrays (e.g. Fast Fourier Transform module), or changing the graphical presentation of data (e.g. shading module). Data processing modules usually get data (i.e. two-dimensional field of SPM data), possibly ask for processing options and do the requested data processing. More interactive functions are typically better implemented as tool modules.

- **file loading and saving modules** handle import and export of foreign file formats, also the Gwyddion native file format is handled by a module.

- **graph modules** operate on one-dimensional data (graphs), e.g. profiles obtained by Profile selection tool. An example is Function fit module.

- **tool modules** provide tools operating on two-dimensional data directly in application data windows. They have typically more interactive interface than processing modules and allow to select objects on the data with mouse. Examples include Read value or Three-point leveling tools.

User interface



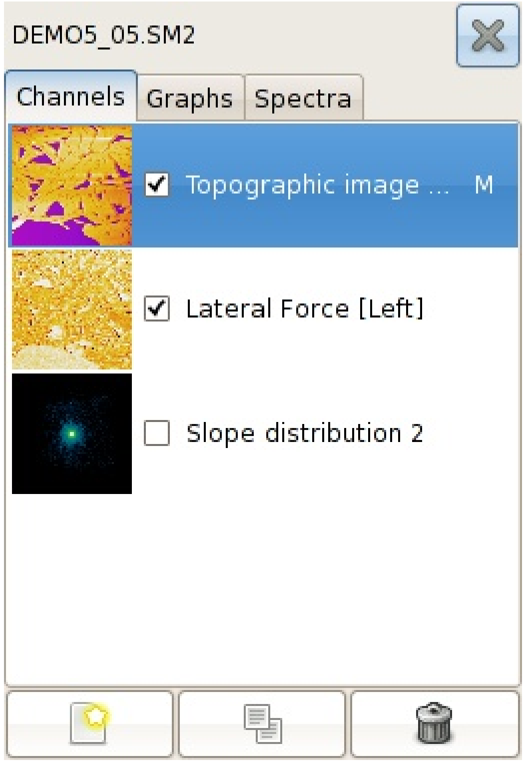
Main window (toolbox)

Icons: selected processing modules (also from Data process), namely for most frequently used operations

Graph modules: fitting, measuring, export

Tools: processing modules using mouse selections (using current DataWindow interactively).

User interface



Data browser
Displays the structure of currently focused file (container).

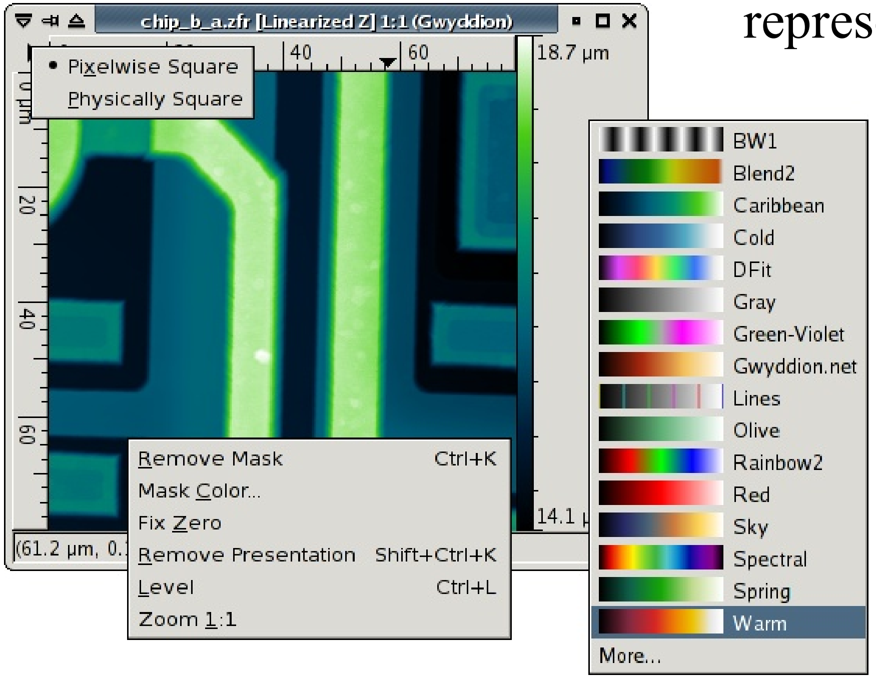
There can be more data in single file, representing more 2D measurements, diferent processing stages, graphs, spectra etc.

Data can be added to container using drag and drop mechanism.

User interface

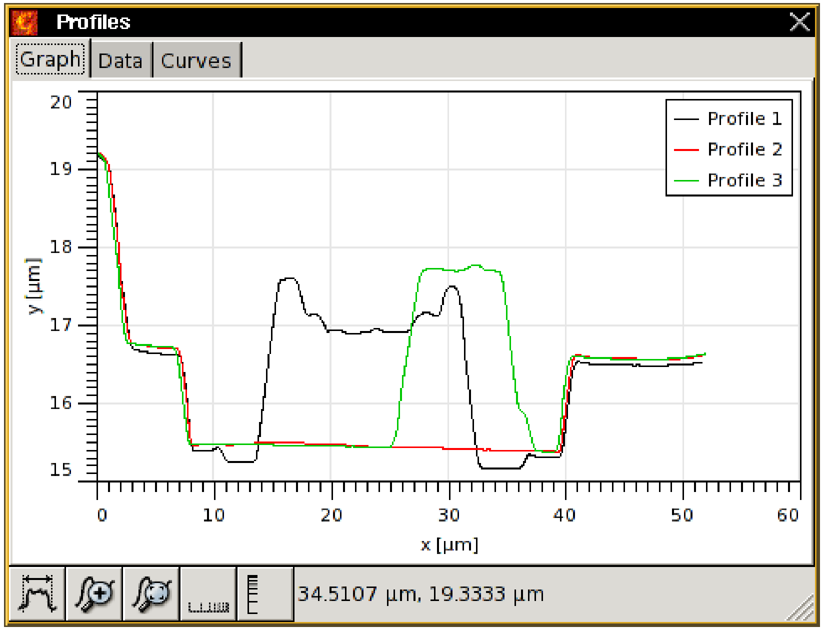
Data window
Key part of Gwyddion – displaying 2D data in false color representation.

Ability to change color scale, pixel representation, make mouse selections etc.



User interface

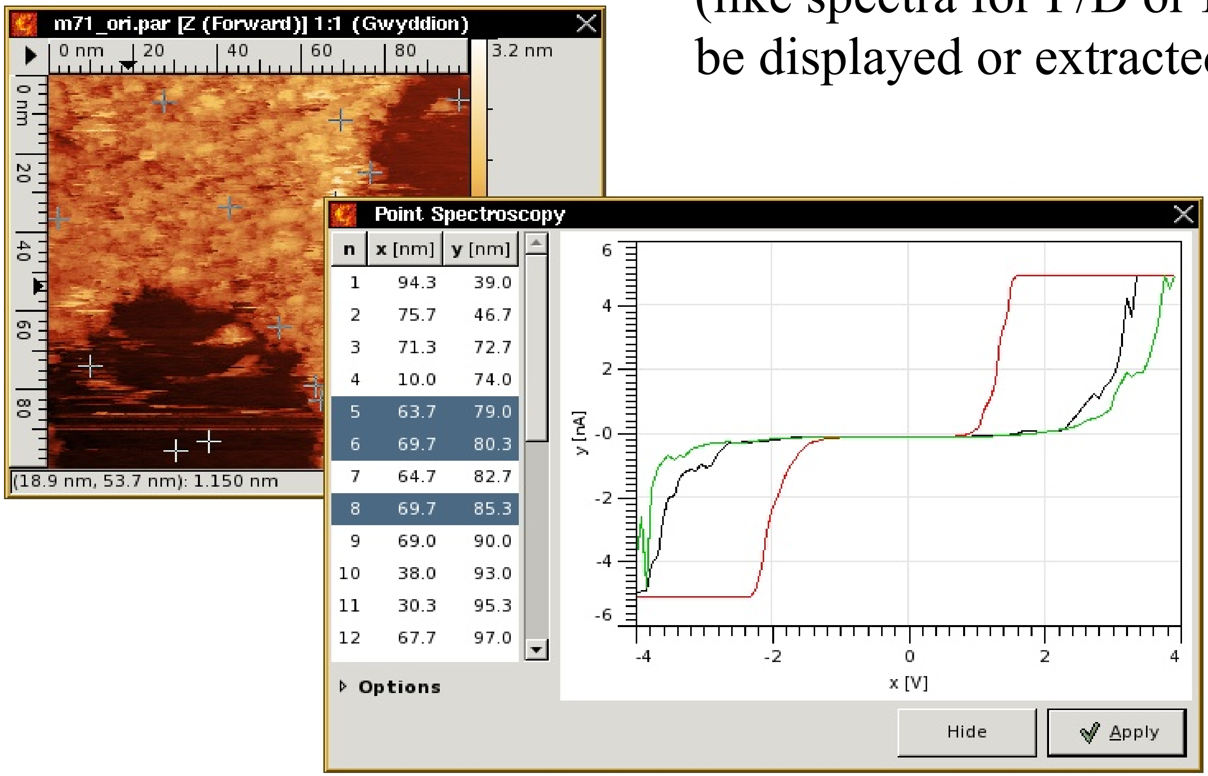
Graph window
Displaying 1D data, graphs, profiles, extracted spectra. Limited processing possibilities, namely for measurement and fitting functions.



User interface

Spectra

Using spectroscopy tool the graphs associated to certain points in 2D data (like spectra for F/D or I/V curves) can be displayed or extracted into graphs.



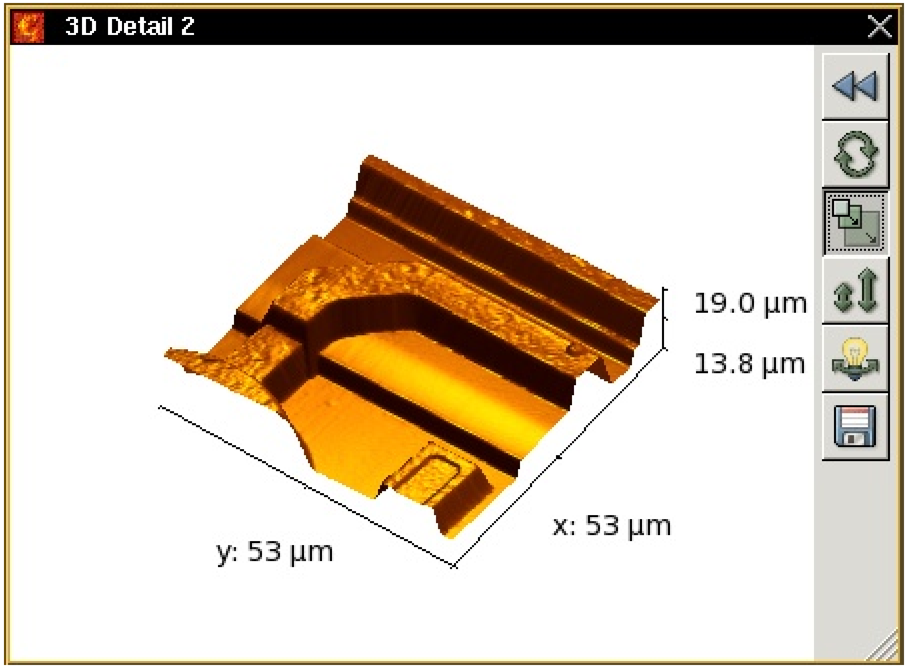
User interface

Mask
Selected area (not necessarily contiguous) used as input or output from data processing modules.

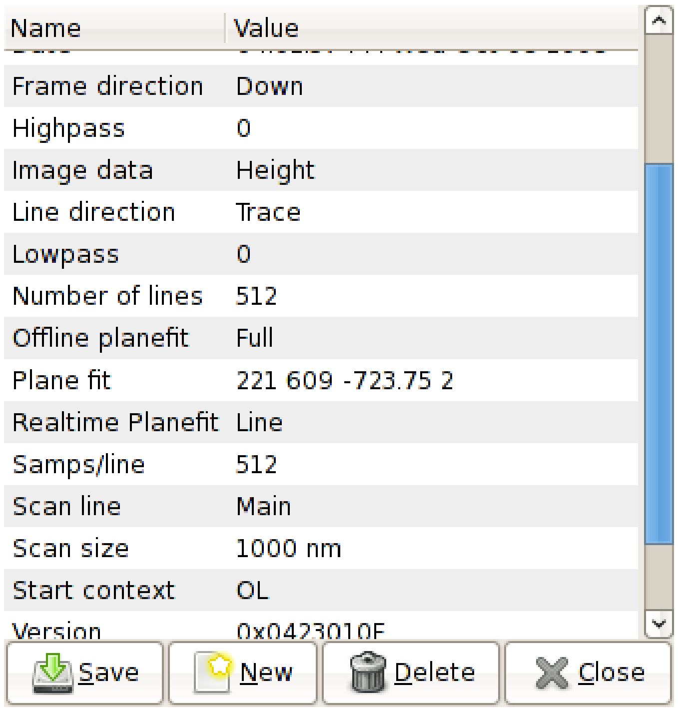
Presentation
Data representation not related directly to z-values (shading, edge detection). Modules still use real data behind.

User interface

3D data display
OpenGL widget showing data in pseudo3D view. Only for export, can be disabled at compile time.



User interface



Metadata

Data related to measurement, if known and understood from file format.

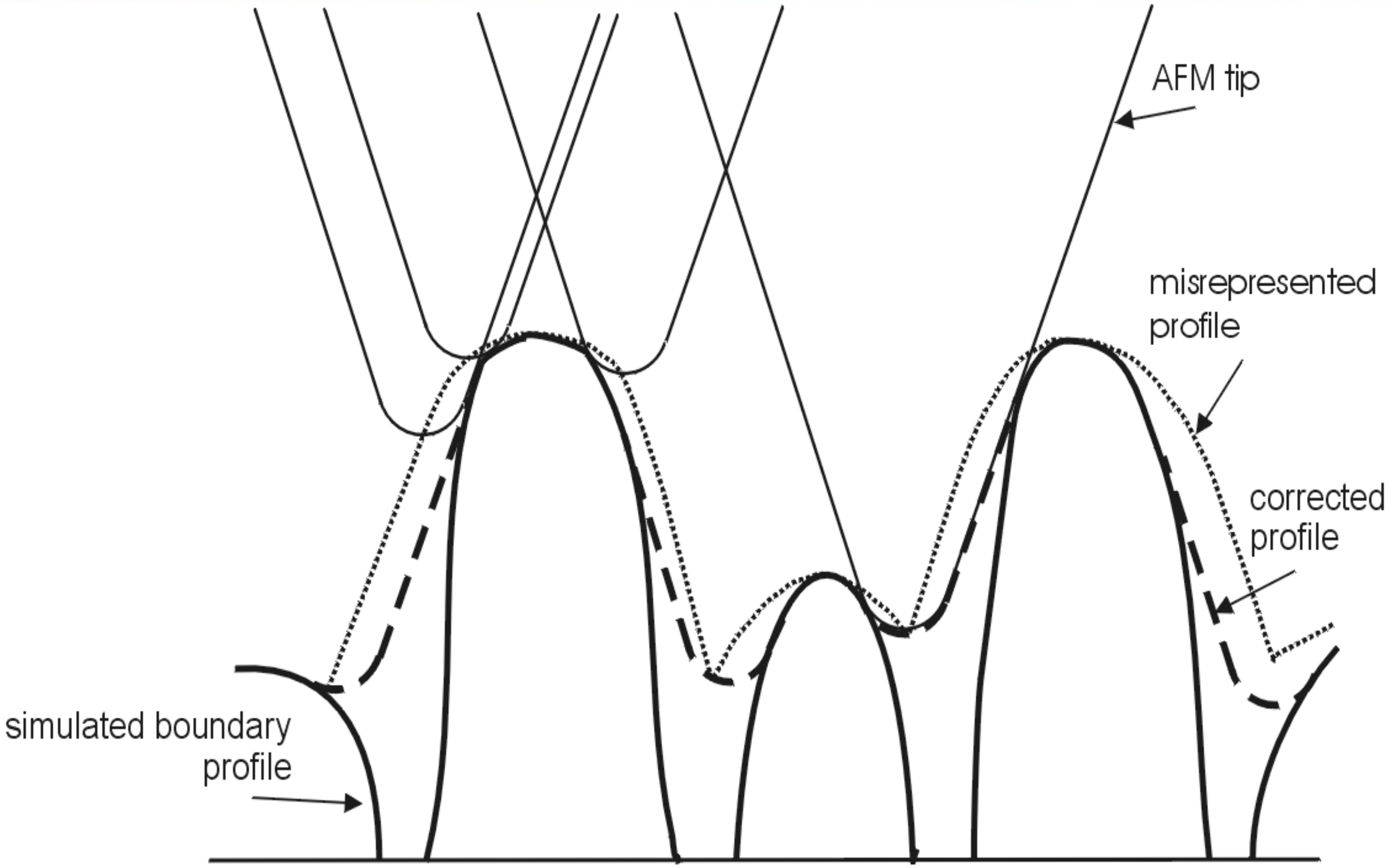
Advanced data processing algorithms

Gwyddion features many different algorithms and is able to perform all the basic tasks in SPM data visualisation, processing, direct or statistical analysis.

Here we discuss more in detail the following sets of data processing tools, that are a bit more advanced:

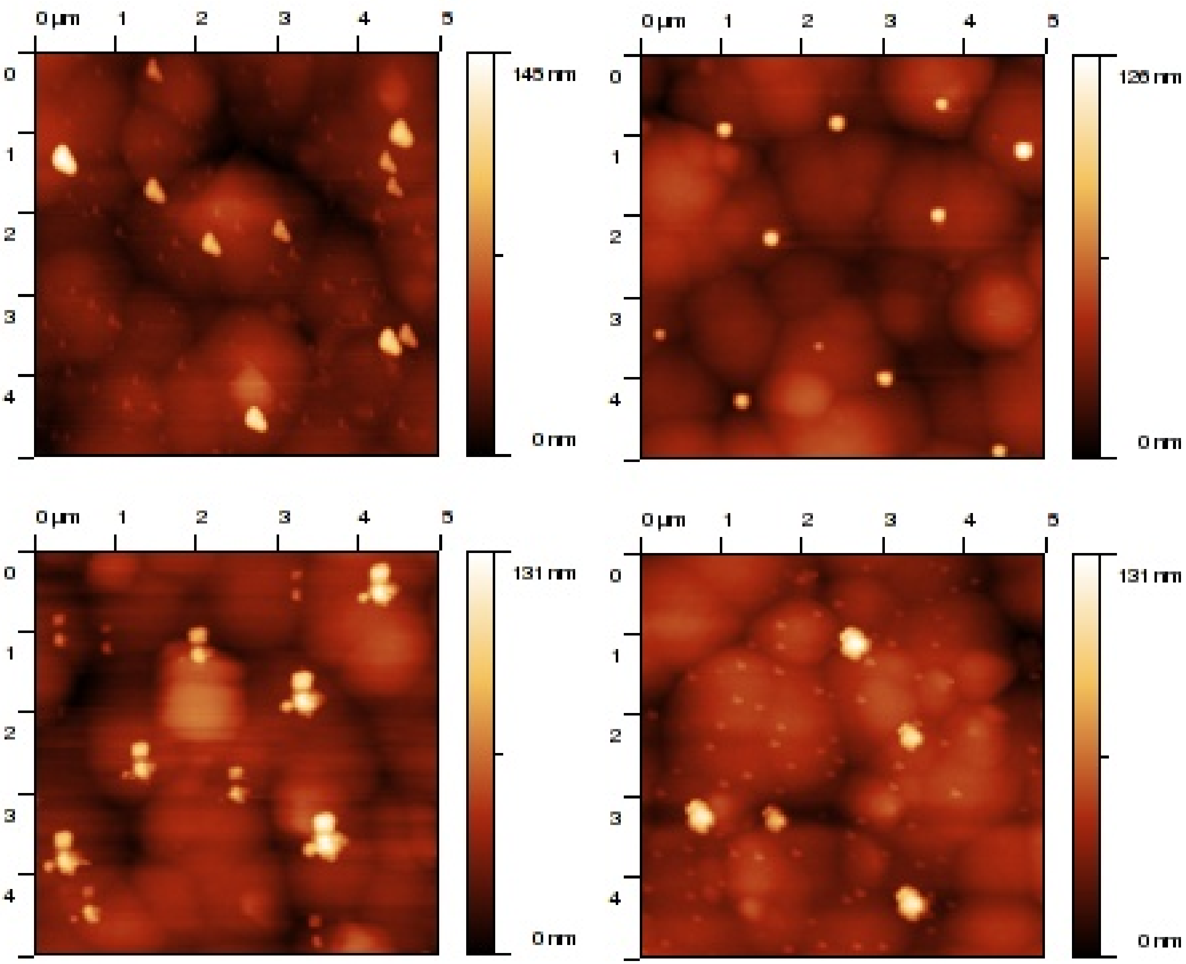
- tip convolution effect related algorithms
- fractal analysis
- grain and particle analysis
- scripting interface

Tip related functions

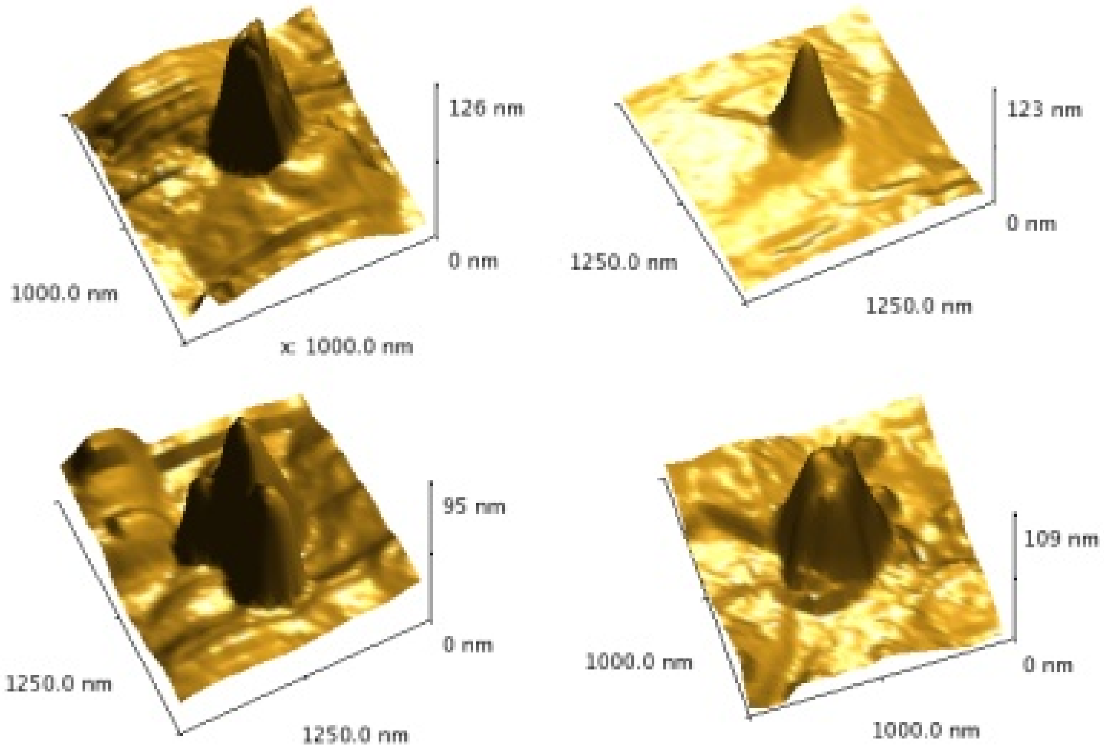


Functions related to AFM tip convolution effect

Tip related functions



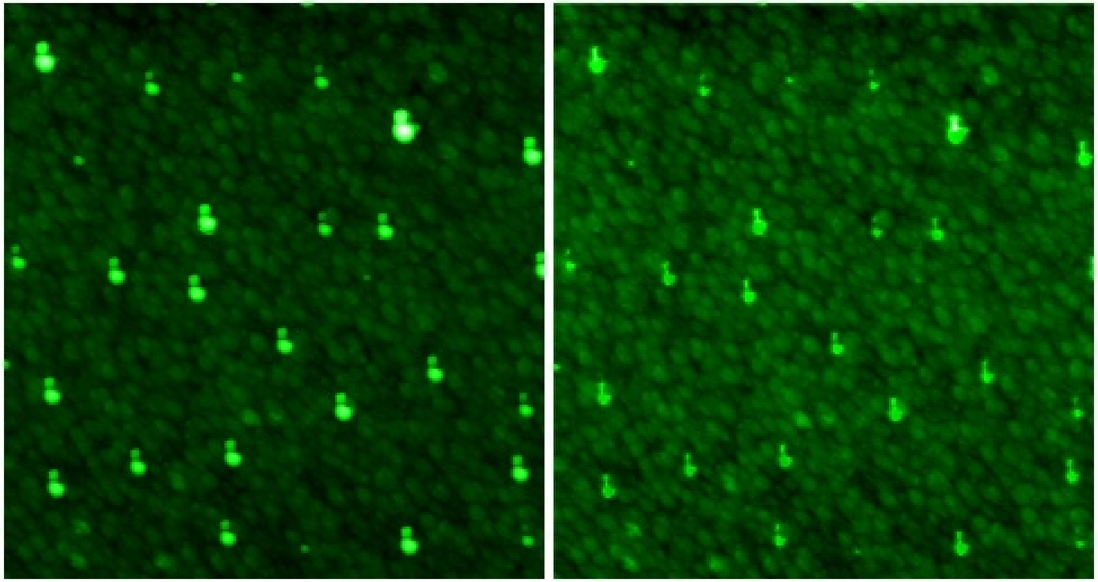
Tip related functions

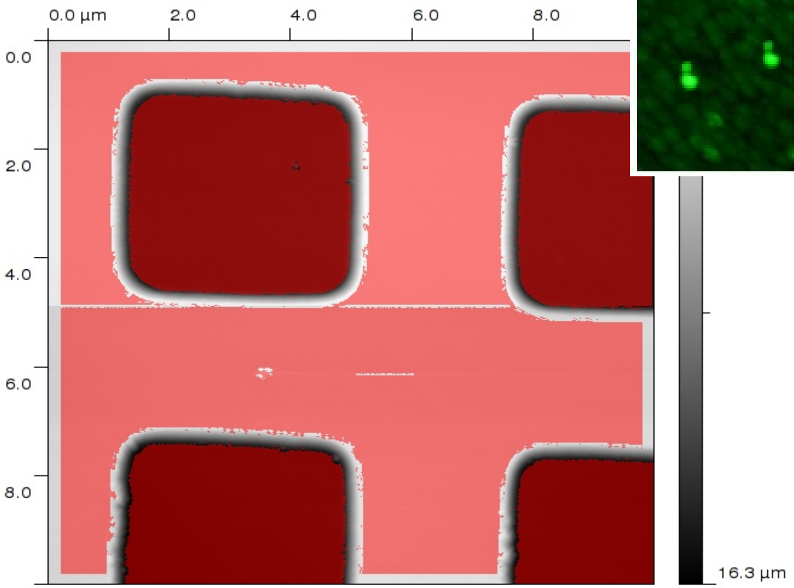


Blind tip estimation algorithm results

Tip related functions

Surface reconstruction

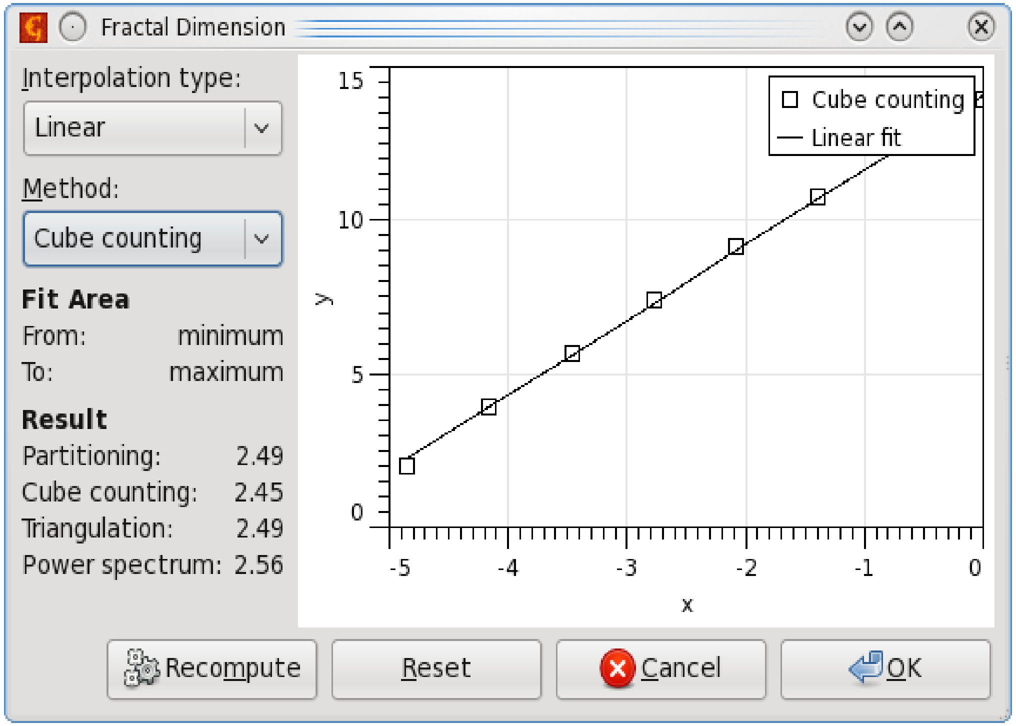




Certainty map

Fractal analysis

Fractal analysis: determining fractal dimension D_f or Hurst exponent H.

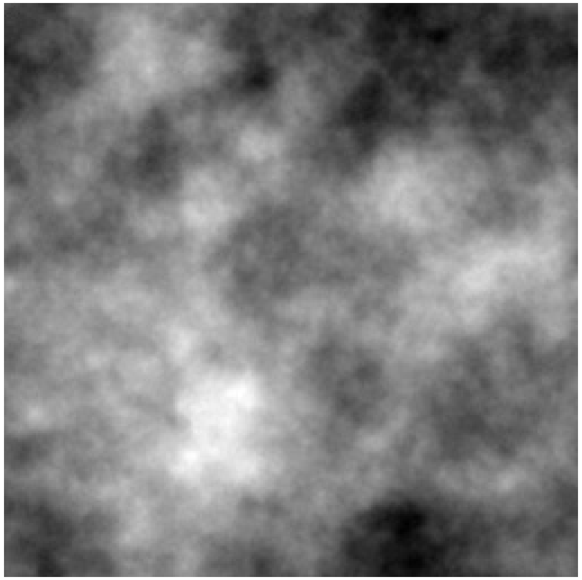


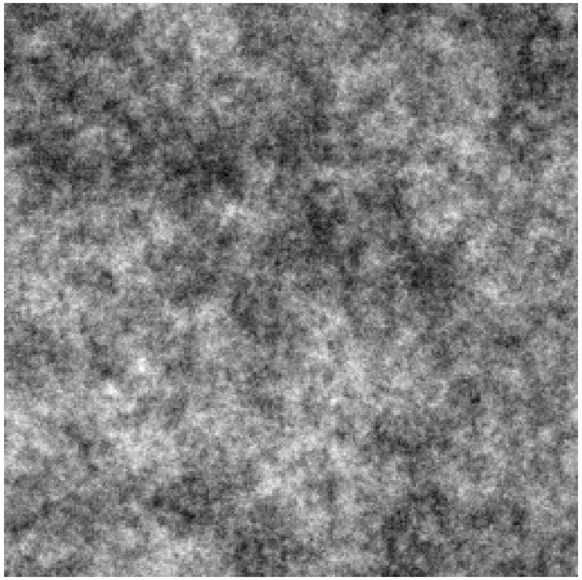
$$D_f = \lim_{l \to 0} \frac{\log N(l)}{\log l^{-1}}$$

where $D_f = 3 - H$

Fractal analysis

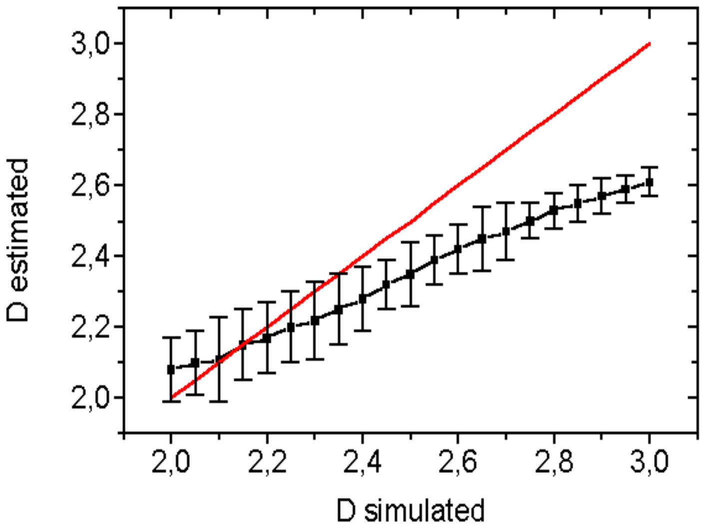
Set of methods for determining the fractal dimension from height fields. Tested on simulated data (using fBm).

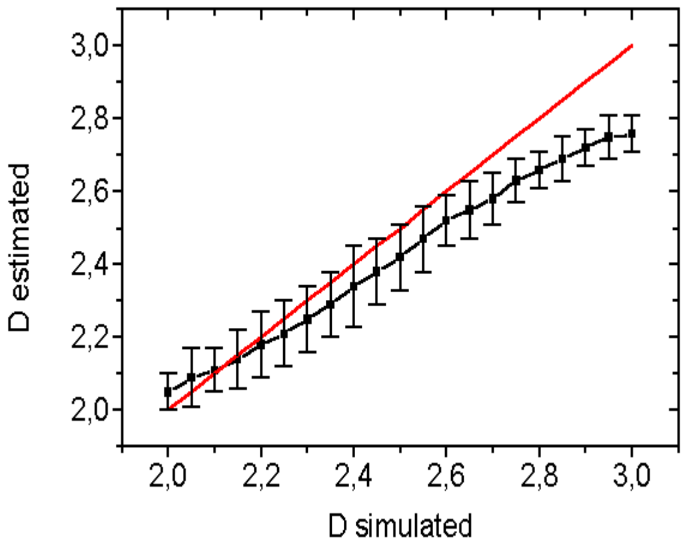




Fractal analysis

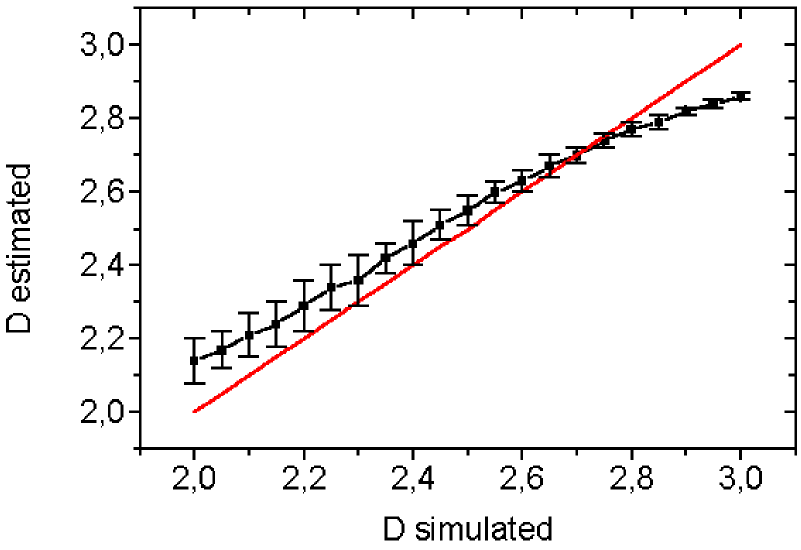
Cube counting and triangulation method efficiency

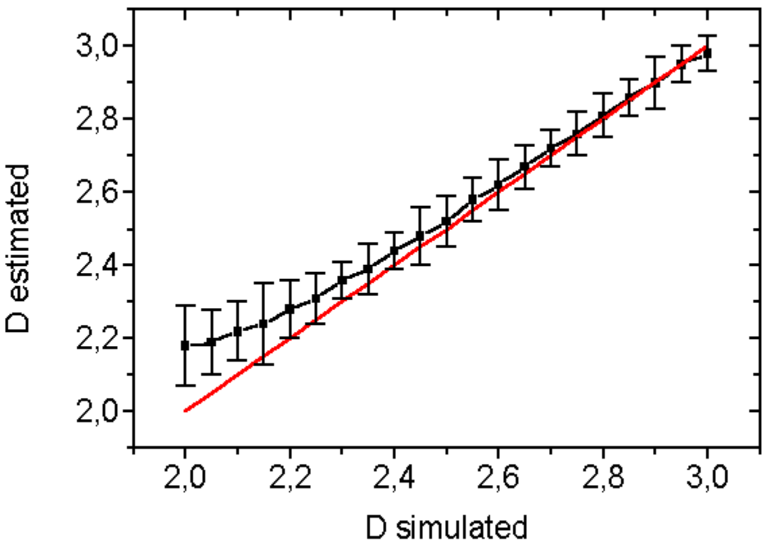




Fractal analysis

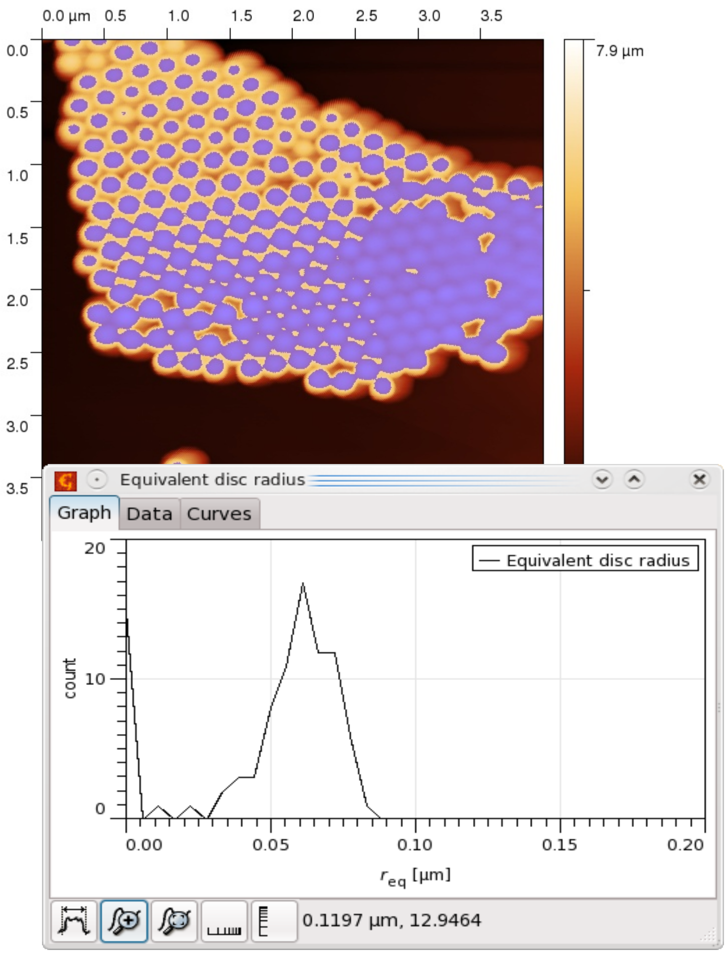
Partitioning and PSDF method efficiency

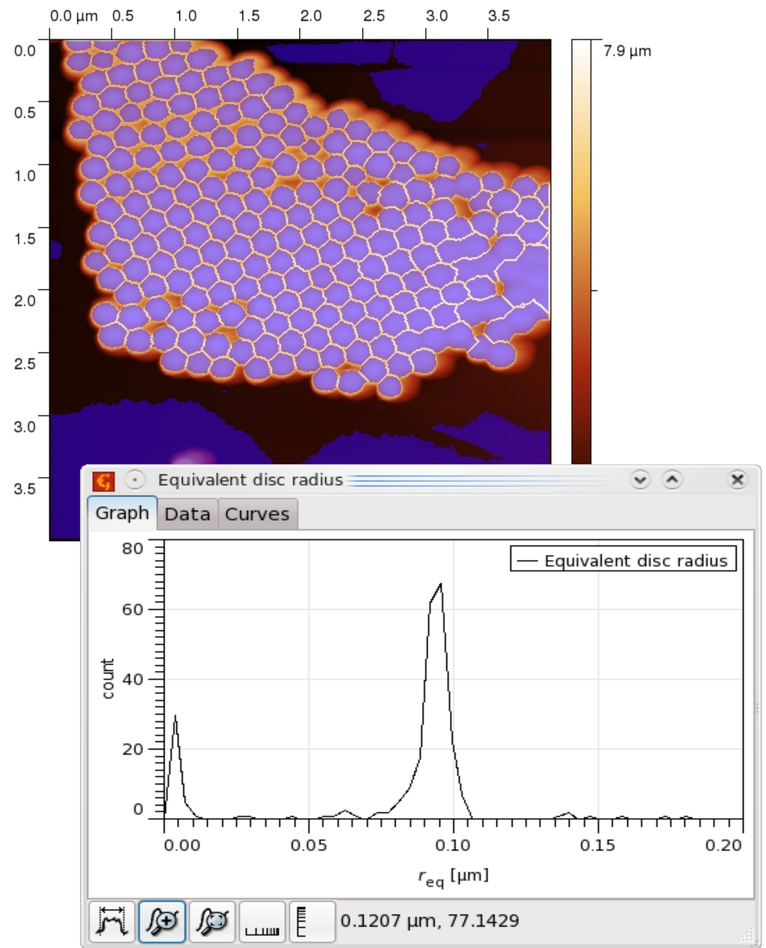




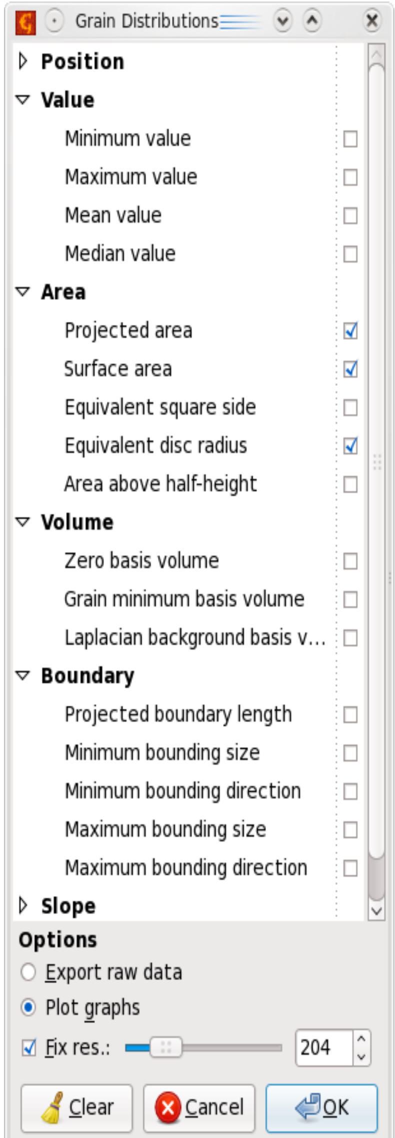
Particle analysis

Image segmentation: thresholding vs watershed algorithm





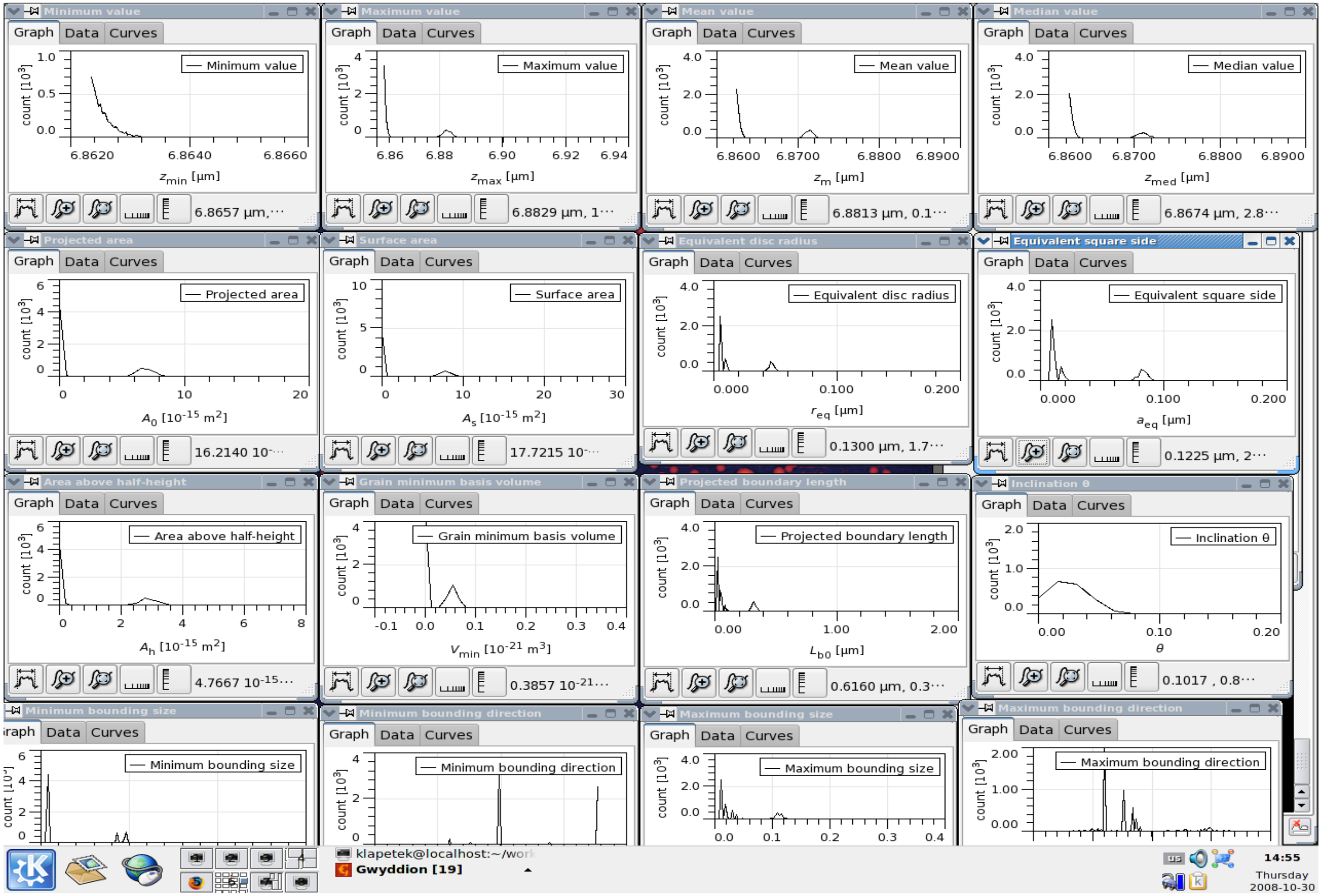
Particle analysis



Particle statistical functions and quantities:

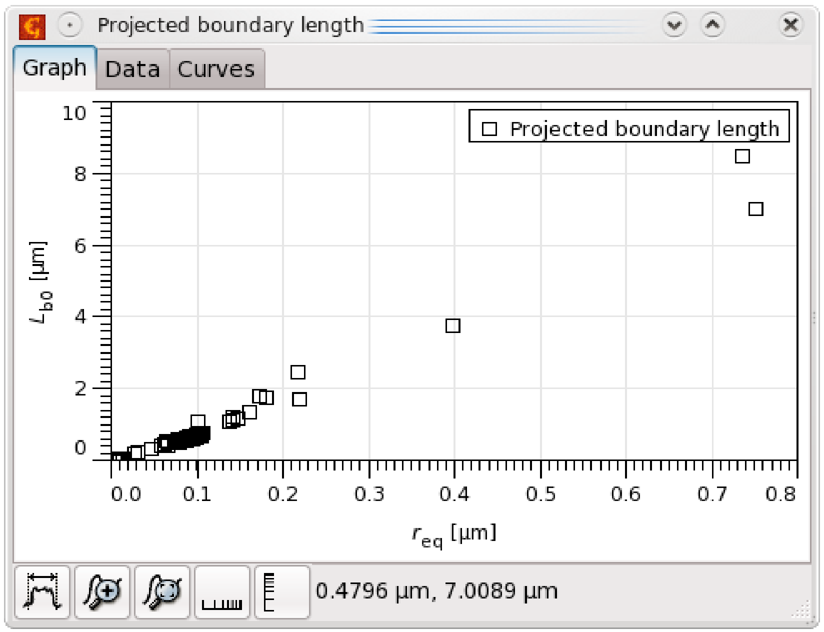
mostly optimized for small aspect ratio particles or spherical particle, however, “boundary” quantities can be used for higher aspect ratio particles as well.

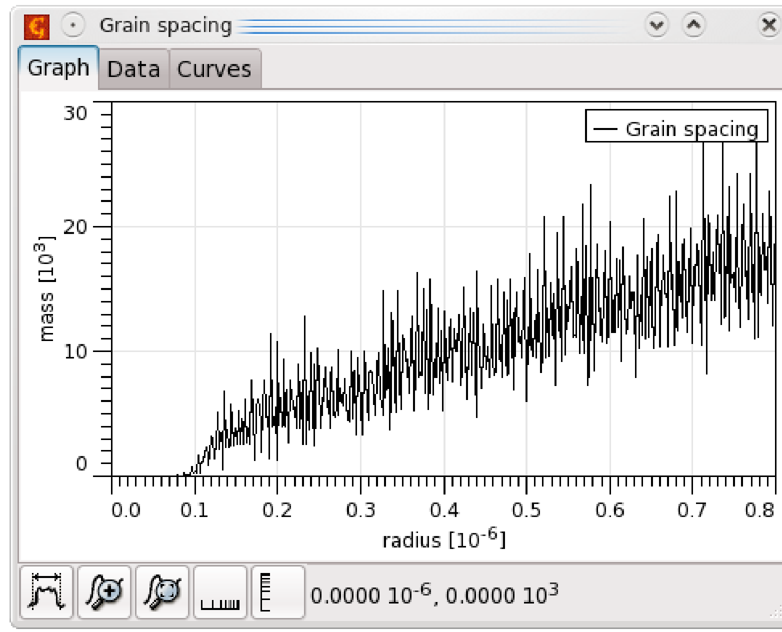
Particle analysis



Particle analysis

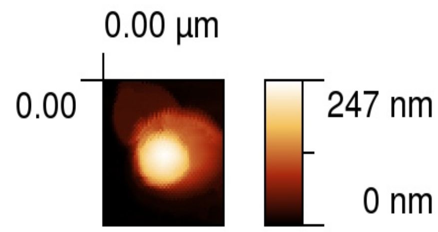
Special statistical functions and quantities can be easily developed both using C and Python libraries.

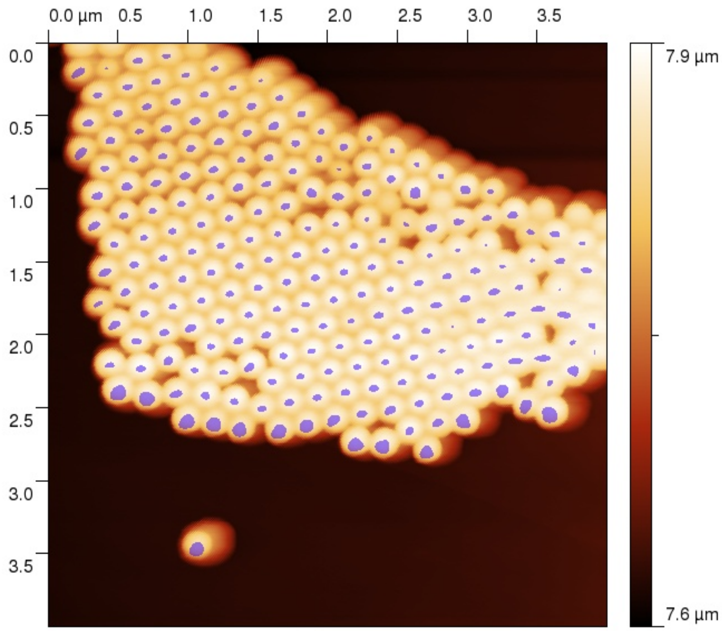




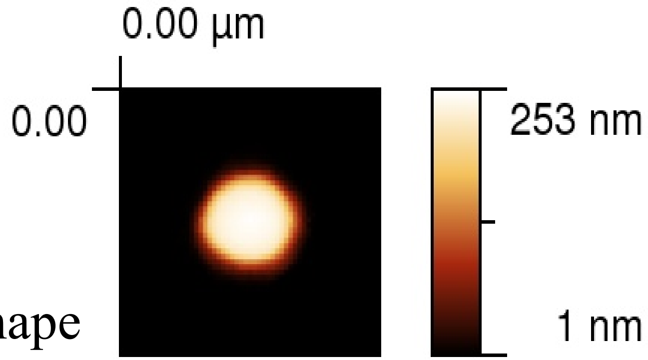
Particle analysis

Tip convolution effects on measured particles



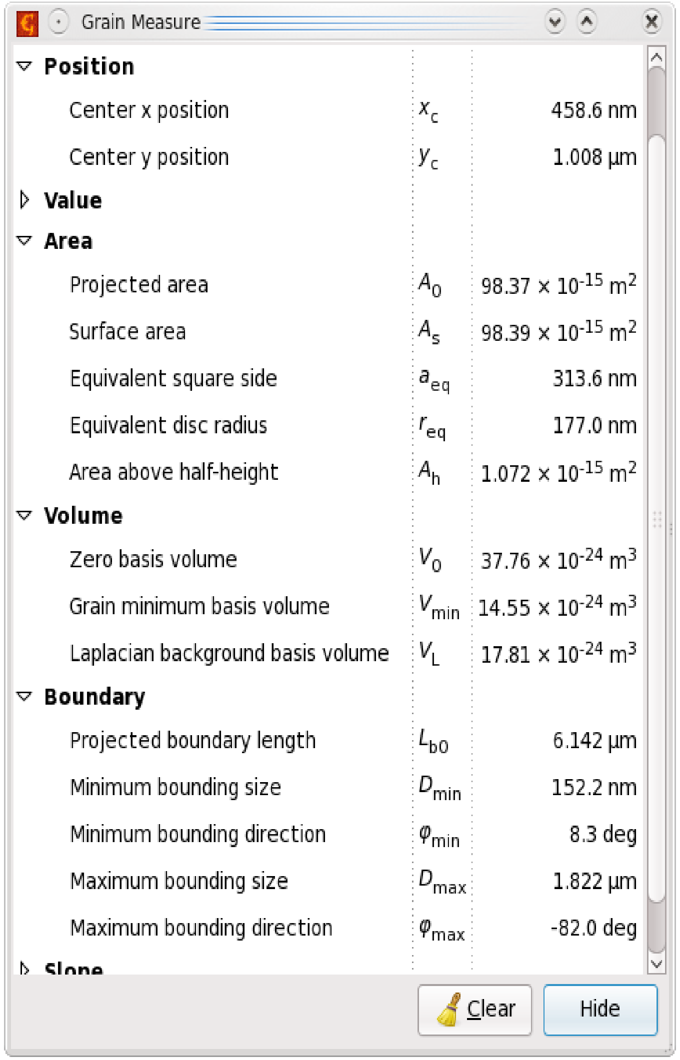


Correlation search for nanoparticles based on its spherical center



Average particle shape

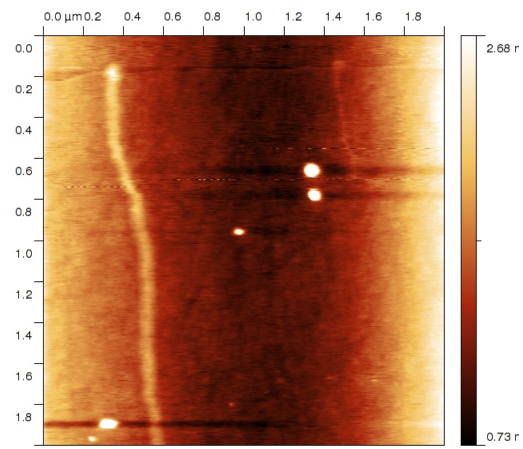
High aspect ratio particles

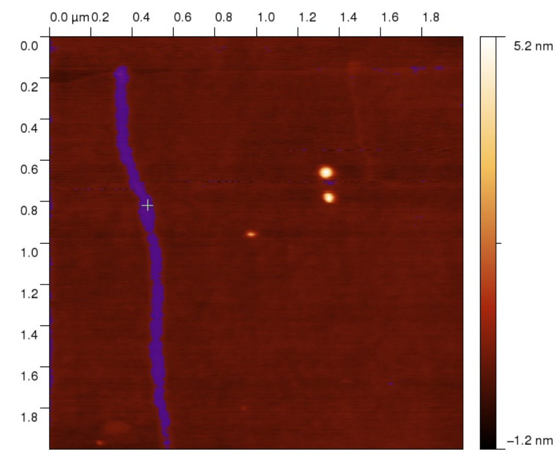


Grain Measure

Position		
Center x position	x_c	458.6 nm
Center y position	y_c	1.008 µm
Value		
Area		
Projected area	A_0	98.37×10^{-15} m^2
Surface area	A_s	98.39×10^{-15} m^2
Equivalent square side	a_{eq}	313.6 nm
Equivalent disc radius	r_{eq}	177.0 nm
Area above half-height	A_h	1.072×10^{-15} m^2
Volume		
Zero basis volume	V_0	37.76×10^{-24} m^3
Grain minimum basis volume	V_{min}	14.55×10^{-24} m^3
Laplacian background basis volume	V_L	17.81×10^{-24} m^3
Boundary		
Projected boundary length	L_{b0}	6.142 µm
Minimum bounding size	D_{min}	152.2 nm
Minimum bounding direction	φ_{min}	8.3 deg
Maximum bounding size	D_{max}	1.822 µm
Maximum bounding direction	φ_{max}	-82.0 deg
Slope		

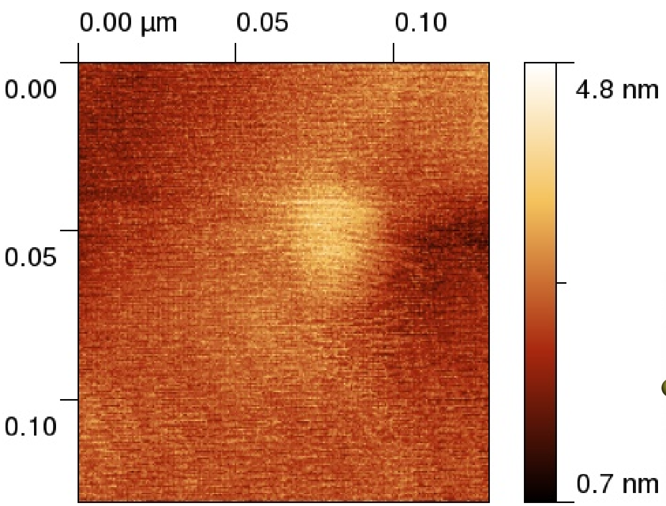
Clear Hide

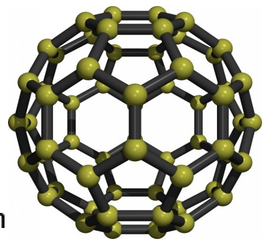




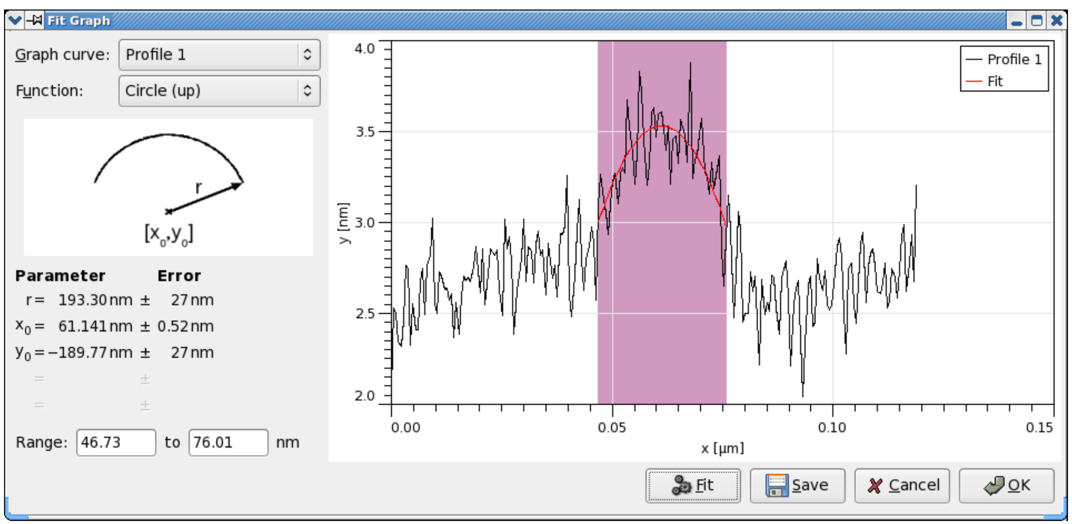
Individual particle properties for carbon nanotube

Nanoparticle measurement uncertainties





Calibration of carbon nanotubes, or fullerenes (here C60), prepared from dispersion



Analysis results:

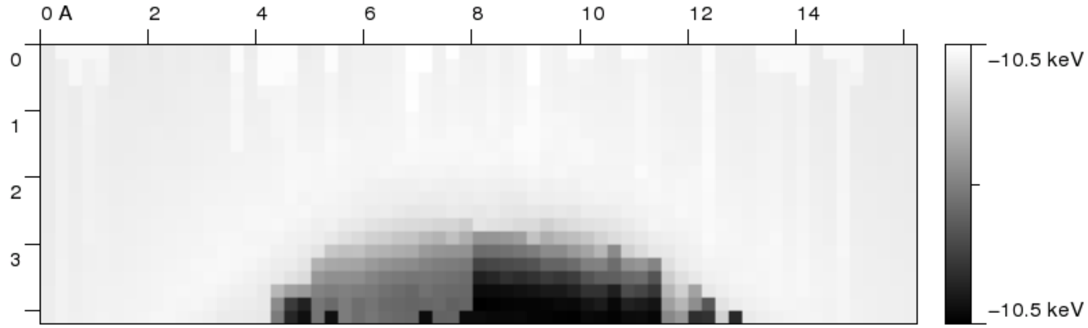
height:

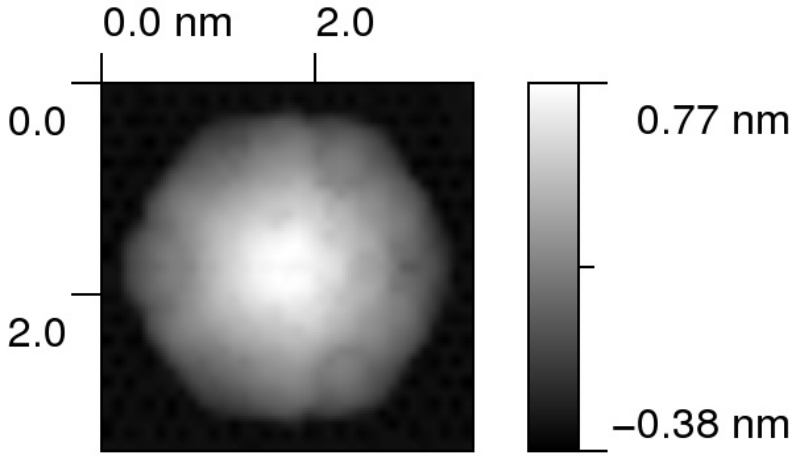
0.8 ± 0.2 nm

width:

32 ± 4 nm

C60 AFM measurement





Constant force (small, repulsive) simulated AFM image with silicon tip.

Large forces again cause big tip structural changes, similarly to DFT calculations.

Height/lateral size values averaged for different forces:

H: 0.97 ± 0.08 nm

W: 1.92 ± 0.12 nm

PyGwy interface

Gwyddion provides a Python binding of nearly all the library functions. Data processing or visualization modules can be therefore written also in Python. This is a recommended method for writing simple modules (if not in C). Former plug-in interface won't be supported in future.

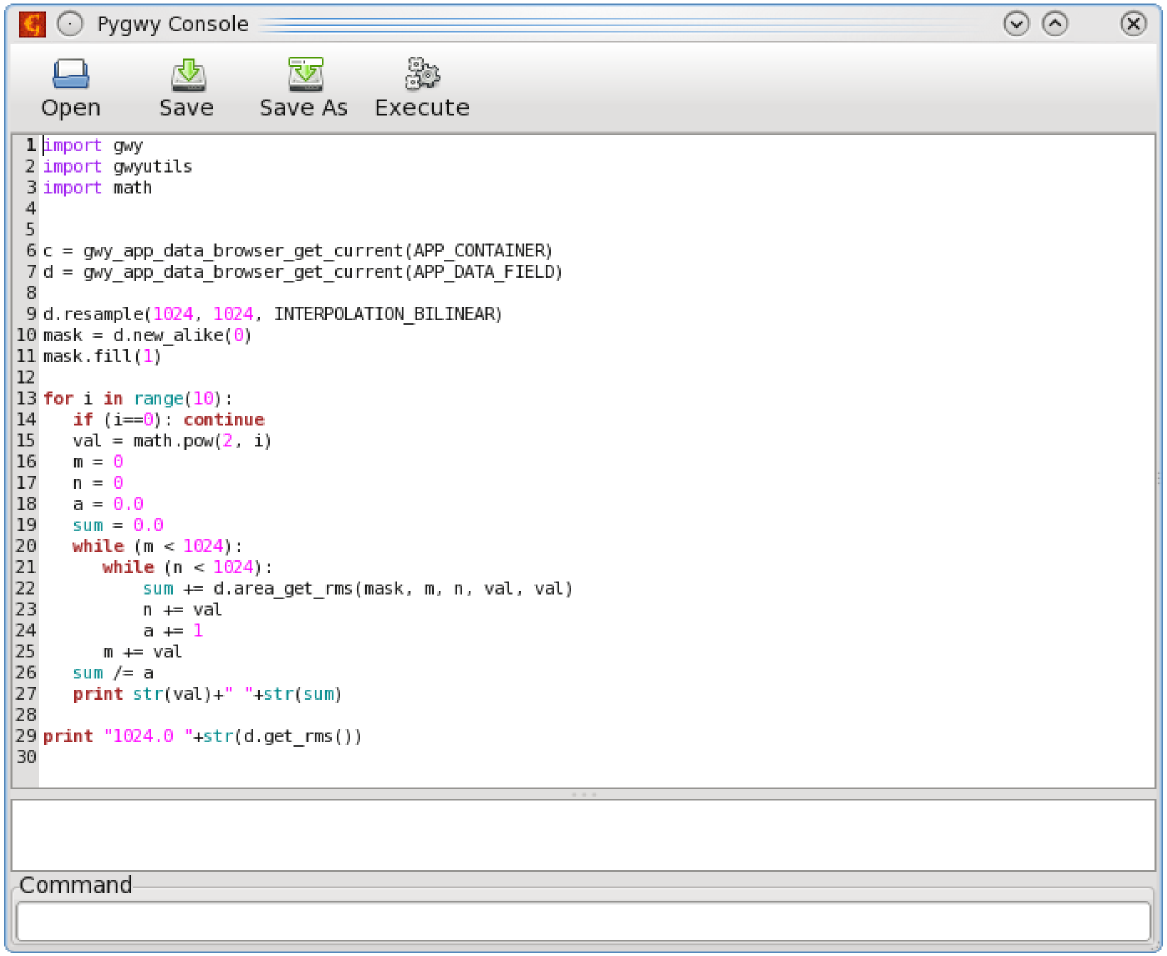
Moreover, there is a batch scripting suport using Python language and Python inferface supported in Gwyddion. For this, a Python console can be used.

PyGwy interface

Example of very simple processing module (invert) using Pygwy

```
import gwy

plugin_menu = "/Correct Data/Invert"
plugin_type = "PROCESS"

def run():

   key = gwy.gwy_app_data_browser_get_current
                              (gwy.APP_DATA_FIELD_KEY)
   gwy.gwy_app_undo_qcheckpointv(gwy.data, key)

   d = gwy.gwy_app_data_browser_get_current(gwy.APP_DATA_FIELD)

   d.invert(0, 0, 1)
   d.data_changed()
```

PyGwy console



Petr Klapetek, Czech Metrology Institute, Okružní 31, 638 00 Brno, Czech Republic

Future directions

- Version 3.0 – simplified and improved.

- 3D calibration, uncertainty propagation and evaluation

- Nonequidistant measurements, general 3D data

- Improved graphs

- More modules dedicated to specific tasks?