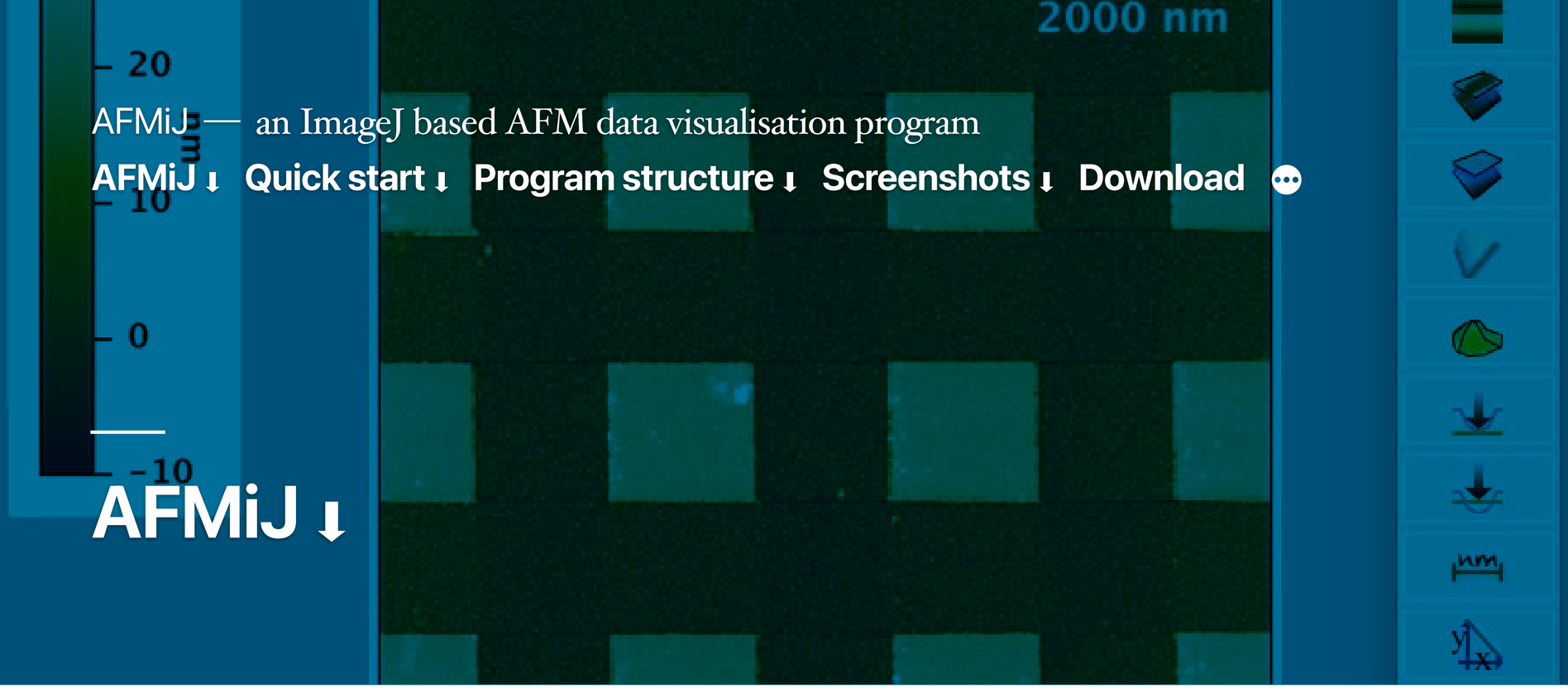


AFMiJ ⬇

NOTE: this document mainly refers to AFMiJ version 0.1.4a.

Updated sections (AFMiJ version 0.1.9a) are shown in green.

– AFMiJ is a collection of Java plugins and ImageJ macros, which allows the reading and the basic manipulation of atomic force microscopy images in ImageJ.

– AFMiJ is based on ImageJ1 site link)

and uses:

- Action Bar (site link) from Mutterer, Jerome et al.
- A forked version of Interactive 3D Surface plot of Kai Barthel (here the original version)
- The non linear background subtraction tool is based on code of Michael Schmid

Limitations:

- loads only AFM images (no force curves or force volume data)
- images with non-square pixels are rendered with wrong proportions (but images can be corrected)
- only few AFM data formats are supported

Motivation:

- To use the image analysis capabilities of ImageJ with AFM data. Many ImageJ plugins are available, extending the program capabilities. A list can be found here.
- Cross-platform.

Further documentation:

Link to Introduction to ImageJ

Quick start ⬇

– AFMiJ requires a working Java >=1.8 installation. On Linux, **openjdk** is required.
There are two main ways to launch the program:

1) OS specific launchers, which are in the distribution main directory (double click on the launcher suitable for your operating system):

- *AFMiJ.OSX.app*
- *AFMiJ.WindowsPS.exe*
- *AFMiJ.linux*

MacOS IMPORTANT NOTE:

on MacOS downloading and unzipping the folder will result in a quarantined (and then blocked) app. To avoid this, use one of the following strategies:

- after unzipping, open a terminal window, cd to the folder containing the app (AFMiJ.OSX.app), then issue the following command:
  xattr -dr com.apple.quarantine AFMiJ.OSX.app
- or unzip with 7zip
- or download and unzip on another operating system, then copy the folder to the MacOS system

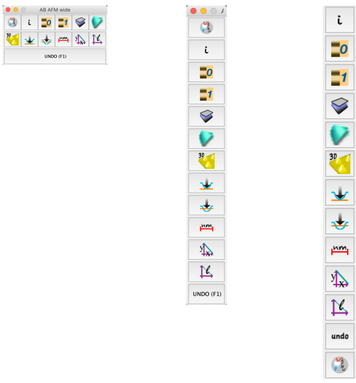
2) Start the program with a terminal command: first cd to the installation main directory and then issue the command *(pay attention to the backquotes)*:

```
java -cp ij.jar `cat plugins/AFMiJ/storage/envvars/JavaOptions` ij.ImageJ
```

– AFMiJ uses two GUIs:

1. the standard ImageJ GUI, with some additional AFMiJ icons

2. a further GUI based on the ActionBar plugin. Three styles can be selected: **wide**, **normal** or **sticky**. A Sticky bar stays attached to the active image. To select which style is used at program startup use the AFMiJ specific preferences). To switch at run-time, use the menu command '*Plugins/ActionBar*'.



Left: Wide ActionBar GUI, center: Normal ActionBar GUI, right: Sticky ActionBar GUI.

Some hints:

- **Important:** save images in TIFF format. In this way several auxiliary data are preserved (image scale, overlay layers (including the scale bar), ROIs, instrumental acquisition info, comments).
- Image profiles and histogram plots can be visualised in real-time activating the *Live* button in the corresponding window
- Selection (lines, polygons, ovals) can be converted to overlays, pressing the ctrl-B key combination (cmd-B on OSX).
- There are several ways to load AFM data: using the *File/Open File...* menu, dropping files in the Status Bar of ImageJ, by using the *Plugin/Readers* submenu, and using the right triangle icon in the ImageJ GUI.
- If there is a selection, or a ROI, the 3D viewer plugin renders only the corresponding area.
- The 3D viewer plugin only works with 8, 16 and 32 bits images.
- The action of the AFMiJ plugins can be restricted to AFM data images, selecting an option in the AFMiJ specific preferences. May be useful to avoid unintended modifications of e.g. plot windows.
- Read the ImageJ documentation for help and more hints : Introduction to ImageJ.

Edit

AFMiJ GUI

opens several files at once, with name filters and options (levelling, color LUT).

Note: files can be also loaded using the standard ImageJ "File/Open..." menu, or by drag and drop in the ImageJ GUI.

Hint: when saving files, use the TIFF format (File/Save As/Tiff...menu), or ctrl+S (cmd + S on OSX). In this way, image scale and other information (e.g. overlay layers) are saved.

left click: add an offset, setting to 0 the lowest point in the image (or the average value, depending on the preferences).

right click: select options. Choose if left-click set to 0 the lowest point in the image or the average height of the image.

left click: apply a 0th order line-by-line leveling to the image.

right click: apply a 0th order line-by-line leveling to the image, using the data marked in red with the Threshold window.

left click: apply a 1st order line-by-line leveling to the image.

right click: apply a 1st order line-by-line leveling to the image, using the data marked in red with the Threshold window.

left click: subtract a plane from the image (or an higher order polynomial, depending on the preferences), using the data marked in red in the Threshold window.

right click: select between plane or polynomial subtraction.

If the image is based on non square pixels, the data are modified in order to obtain an image with the right proportions. Beware, the image dimensions are changed.

Introduce a tilt in the image.

left click: a submenu is opened, where the four possible tilting directions can be selected.

right click: a requester for the amount of introduced tilt is opened.

left click: a submenu is opened, with some AFM dedicated commands (*3D Rendering*, *Select AFM lut*, *Read AFM data*, *Export to .bcrf*, *Export to .gsf*, *3D Spider Export*, *Show Color Scale*, *Convert Palette*, *Generate OSL Script, Show Zoom*) and AFMiJ specific preferences. Note: *Convert Palette* and *Generate OSL Script* read a color scale gradient in Gwyddion format, which is converted to an ImageJ palette or to an Open Shading Language script to be used in Blender rendering.

ActionBar GUI buttons

Show information about AFMiJ.

Show information about the selected image.

Apply a 0th order line-by-line levelling to the image. If the threshold window is used to define an interval, only data marked in red are considered. *(Important: do not press 'apply' in the threshold window, otherwise data will be modified.)*

Apply a 1st order line-by-line levelling to the image. If the threshold window is used to define an interval, only data marked in red are considered. *(Important: do not press 'apply' in the threshold window, otherwise data will be modified.)*

Subtract a plane from the image.

Subtract a polynomial function from the image. If the threshold window is used to define an interval, only data marked in red are considered. *(Important: do not press 'apply' in the threshold window, otherwise data will be modified.)*

Opens a 3D rendering window.

Subtracts an offset, setting the lowest height of the image to 0.
(see Notes below)

Subtracts an offset, setting the average height of the image to 0.
(see Notes below)

Adds a scale bar to the image. Hint: tick the overlay option, otherwise the image data will be changed. Overlays can be added, modified and deleted without affecting the image data.

Prints the length of a line ROI (as overlay). Works in images and in graphs.

Undo the last AFMiJ operation. This undo is separated from the standard ImageJ undo.

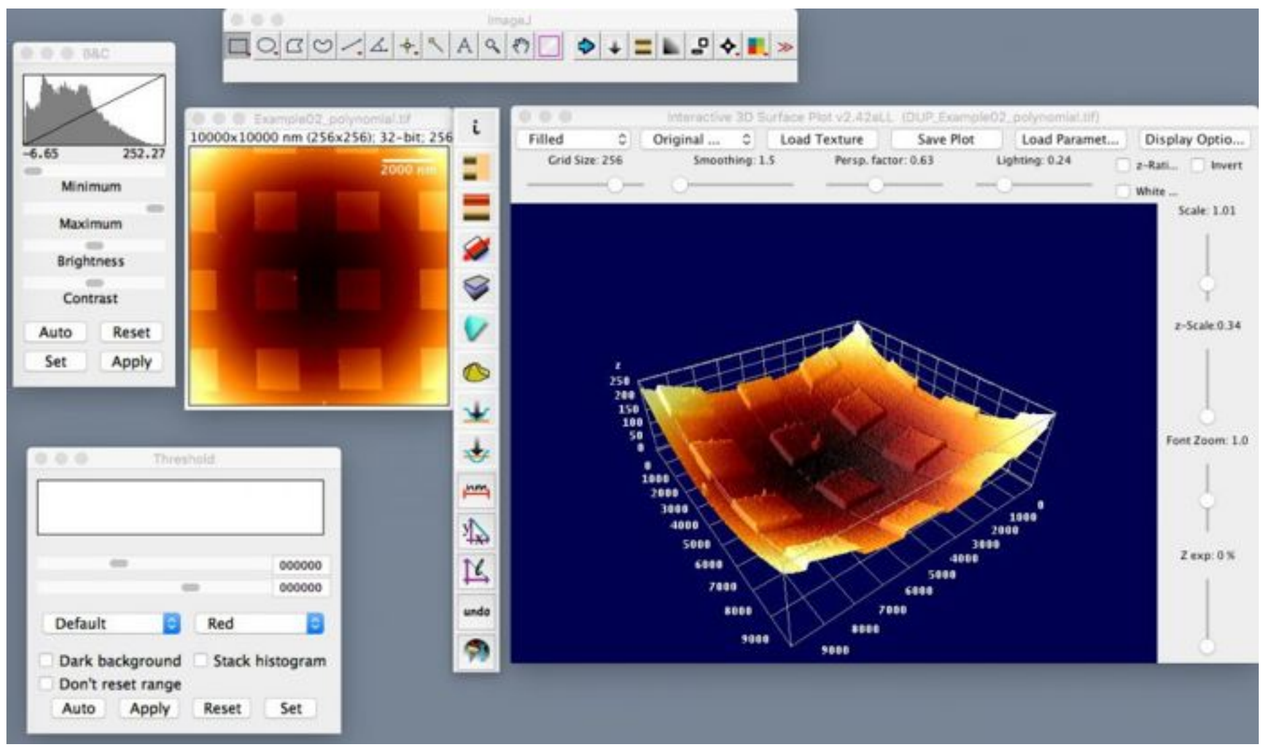
---

*Note 1): if a **line** ROI is present in the image, the offset is evaluated on the line, not on the whole image, but the offset is added to the whole image.*

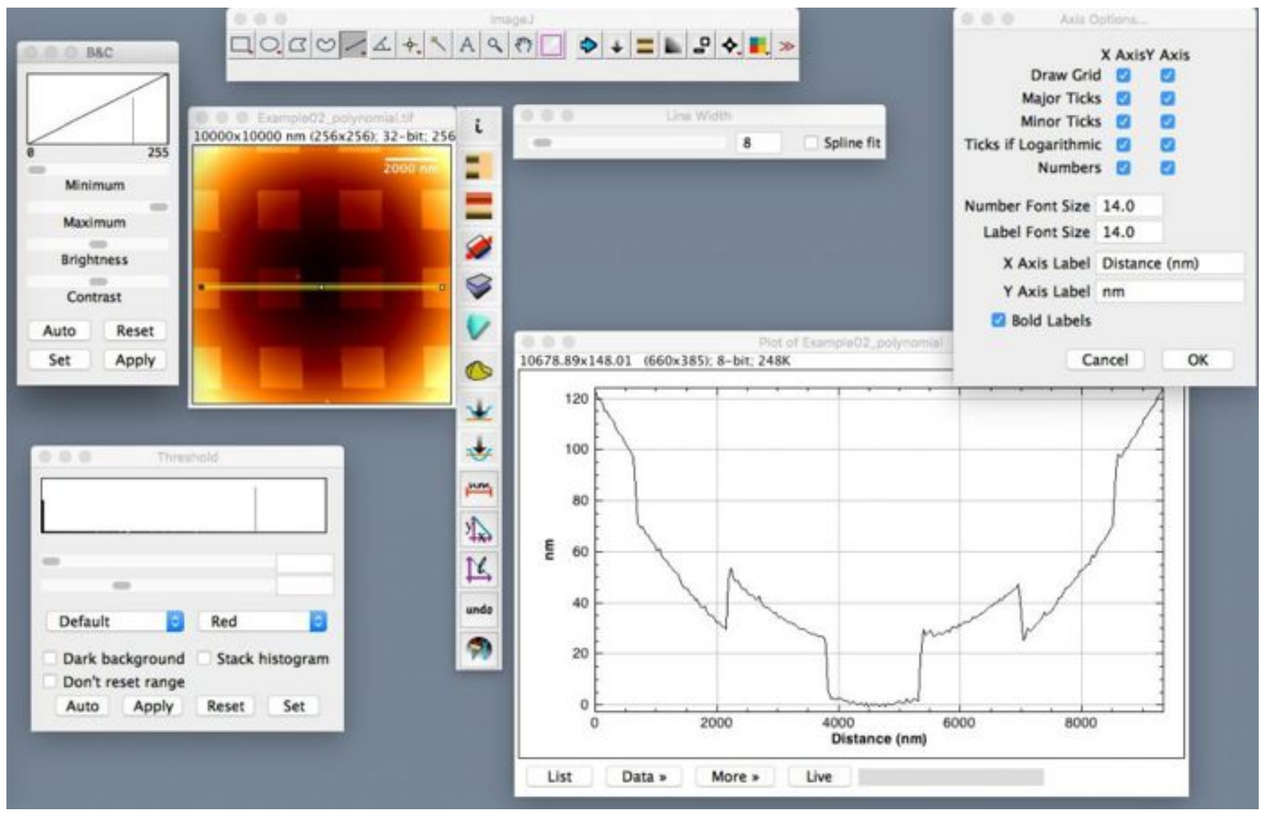
Note 2): *hint: trace a line ROI, then press ctrl+k (cmd+k on OSX) to open a height profile graph window, and press the LIVE button in the graph window to see in real time the effect of the AFMiJ offset buttons (remember to click in the AFM image before pressing the AFMiJ button to make it the active window again).*

*Note 3): if another type of ROI is present (**square**, **circle**, etc.), the offset is evaluated on the ROI, and the ROI area only is affected by the offset, not the whole image.*

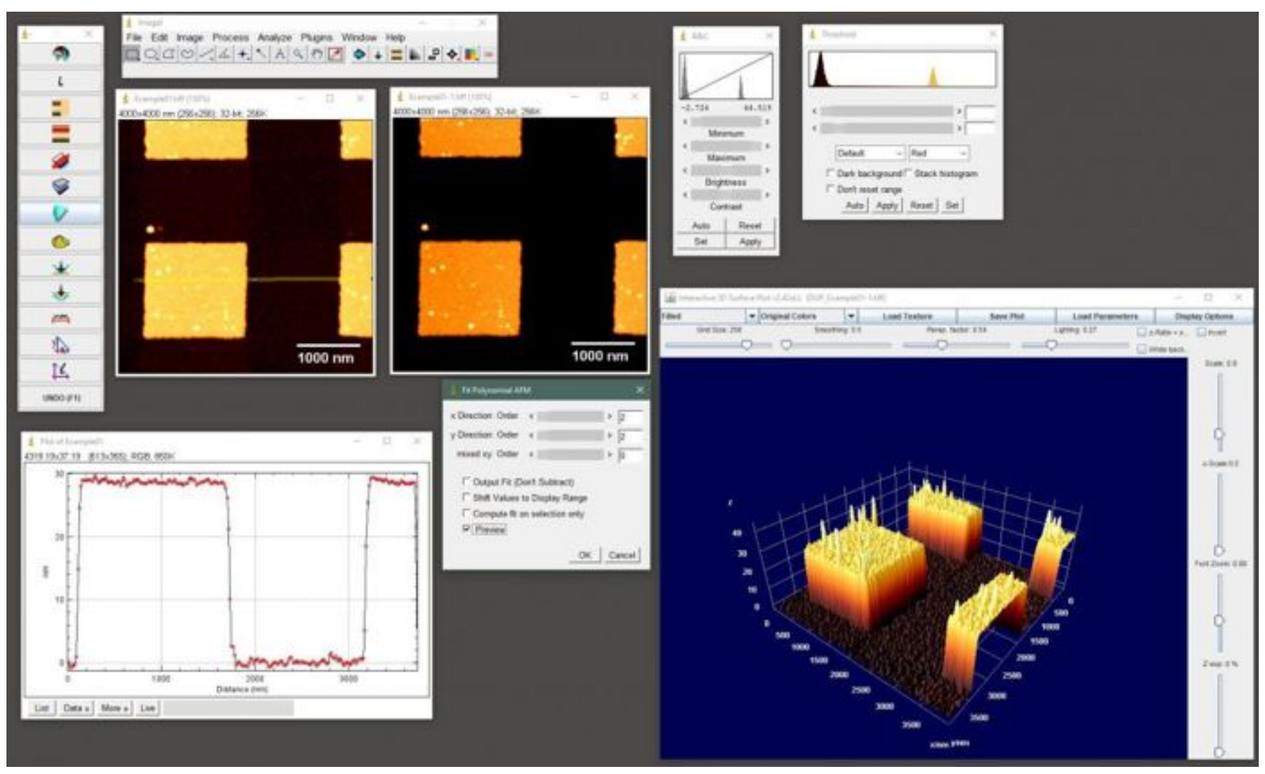
Screenshots ⬇



AFMiJ on OSX (sticky ActionBar). With a 3D viewer window.



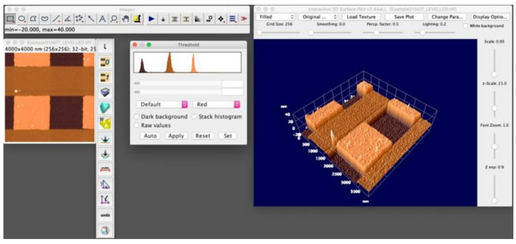
AFMiJ on OSX (sticky ActionBar). With a profile window.



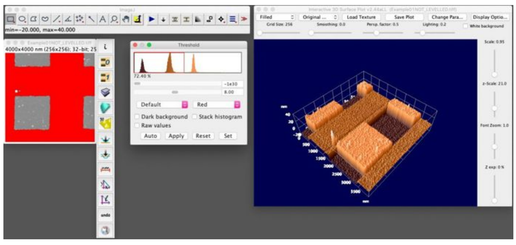
AFMiJ on Windows. Normal ActionBar.

How to use the Threshold window in AFMiJ

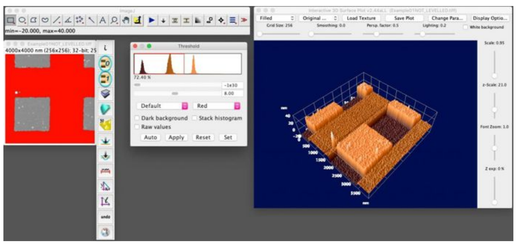
Load the *Example01NOT_LEVELLED.tiff* image from the *Examples* folder.



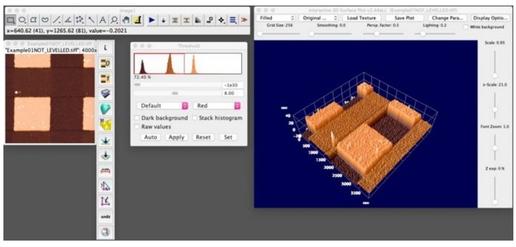
1) The 2D image and 3D image before levelling. In the middle there is the threshold window with the image histogram data.



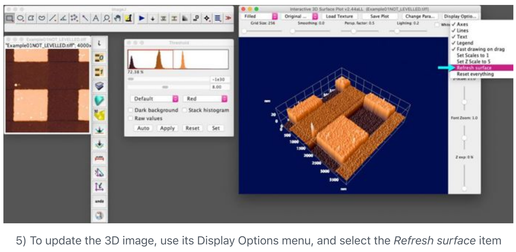
2) Use the threshold window cursors to select the data to be used in the levelling (marked in red in the 2D image).



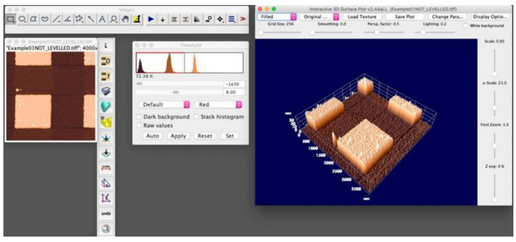
3) Click one of the circled levelling buttons (see the figure) to level the image using the selected data interval. ***Important: do not press the 'Apply' button in the threshold window: data will be changed.***



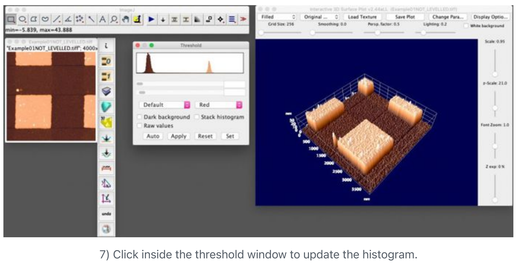
4) The data and the 2D image are now levelled. The 3D image requires a manual refresh.



5) To update the 3D image, use its Display Options menu, and select the *Refresh surface* item (arrow).



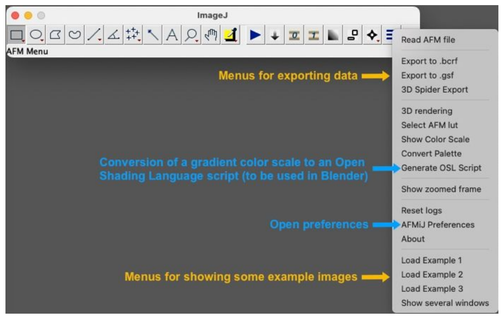
6) The updated 3D image.

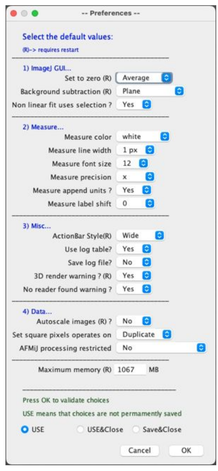


7) Click inside the threshold window to update the histogram.

AFMiJ preferences

Left click this icon to access the AFMiJ specific preferences window



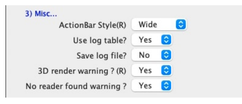


- The fields that appear in the preferences window are controlled by a text file, 'prefvars', in the 'storage/envars' folder.
- The format of that file is described in the 'FormatDescription.rtf' file, found in the same folder.
- Choices become effective after pressing the "OK" button
- IMPORTANT: use this option to change the available memory, the standard ImageJ pref does not work.

Below a description of the main settings:

Settings of image levelling used with the standard ImageJ GUI.

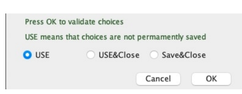
Settings of the *measure* (shortcut 'm') and *length* (shortcut 'l') commands.



- Behavior of ActionBar style (normal, wide or sticky)
- Setting of logging of actions
- Setting of startup warnings

- The first option affects how are rendered windows with multiple data channels (ImageJ stacks). If **yes** is selected, when passing from one channel to the next, the color scale range changes automatically, otherwise a fixed color scale range is used for all channels. This last option makes easier to save data with a user selected scale (only using the tiff format). This option can be temporarily changed using the image contextual menu (select an image, then right-click).
- Second option: when images with rectangular pixels are converted to images with square pixels, duplicate the original image or overwrite it.
- Third option: some typical AFM processing can be restricted to AFM images (actually, images with the *AFMiJ* property set to *yes*. These are images loaded through the AFMiJ plugins, even when saved using the TIFF format)

Amount of memory (requires program restart).

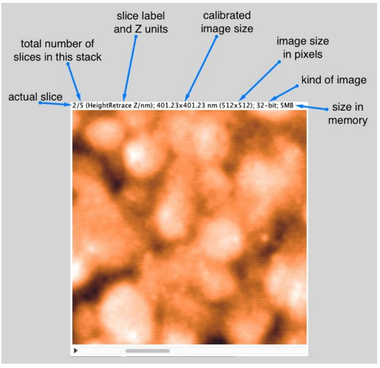


Selection of Preferences window behaviour: (PRESS OK TO ACCEPT CHOICES)

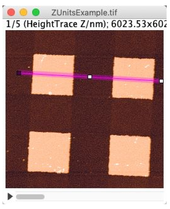
- USE: Preferences window remains open, choices have temporary (session wide) effect (options requiring restart have no effect)
- USE&Close: as above, but window is closed
- Save&Close: window is closed, choices are saved and used in new sessions also.

Program structure

AFM images are stored as *ImageJ stacks* (set of images, containing several *slices*). All the slices share the same bit depth, pixel height, pixel width, lateral calibration, and also the same Z unit. To overcome this problem, and store e.g. height and phase data in the same stack, a label containing the correct unit is set for every slice. The label appears in the line at the top of the image, see below for details.



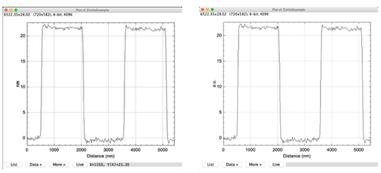
Plotting profiles



AFM data are plotted along the magenta line. See images below.

This slice dependent Z unit information is not stored using a standard ImageJ format. To use it when plotting a Z profile, the label embedded information is passed to the ImageJ plotting routine using a custom macro function, which can be called using a shorcut (small ***k***), or through the menu *Plugins/Macros/plot Profile* (see image below, left panel).

Note: when using the standard ImageJ plot profile command (***ctrl-k***, ***command-k*** on OSX), or the menu command *Analyze/Plot Profile*, the ImageJ plot routine is not aware of the Z unit of the current slice, and the image of the right panel is obtained.



Left: plot obtained with the custom macro command (y units are nanometers). Right: the same plot obtained using the standard ImageJ plot profile command (y units are arbitrary units).

The program subdirectory tree is shown below, hover over the names for a brief description (online only):

AFMiJ_0.1.4a

- ▶ luts
- ▶ macros
- ▶ plugins
  - ActionBar ▶
  - ▶ AFMiJ
    - luts
    - storage
      - app_icons
      - envars
      - images
  - ▶ AFMiJReaders
  - Input-Output

The main directory also contains the three launchers.

AFM Readers

The AFM data readers are stored in the plugins/AFMiJReaders directory (as java compiled programs – sources can be found in the same folder).

The format of the plugin reader names is *AFMREADER**specificname**.class*, where ***specificname*** identifies each plugin. No underscores are allowed in the name. Two methods are implemented: "AFMiJread", with the actual reading code, and "Getversion", which returns the version string. Images are implemented as 32 bit objects. The "AFMiJread" method returns a value, depending on the success in opening the image. Result <0 means failure of the reader, >0 reader success. Result=0 means that the file type is identified but no images are found (e.g. only curves). Readers added to this directory are called one by one, until one able to recognize the data is found. This directory also contains optional ImageJ macros (with an underscore in the name, e.g. *_specificname.ijm*) to be able to call some specific readers from the ImageJ menu.

Note: the Nanoscope reader is in alpha state, to read nanoscope (and molecular imaging) files you may also try the ImageJ plugins available here (https://imagej.nih.gov/ij/plugins/afm.html, even if they seem a bit outdated). In this case, to be used within AFMiJ, the loaded images need to be converted to 32 bits format with the menu command: *Image/Type/32-bit*.

AFMiJ Download

Current version is available on Github:
https://github.com/AFMiJ/AFMiJ

AFMiJ specific code is written using ImageJ macro language and Java. Java source code is available as separate java files or contained in jar files, which are all contained in the distributed zip file.