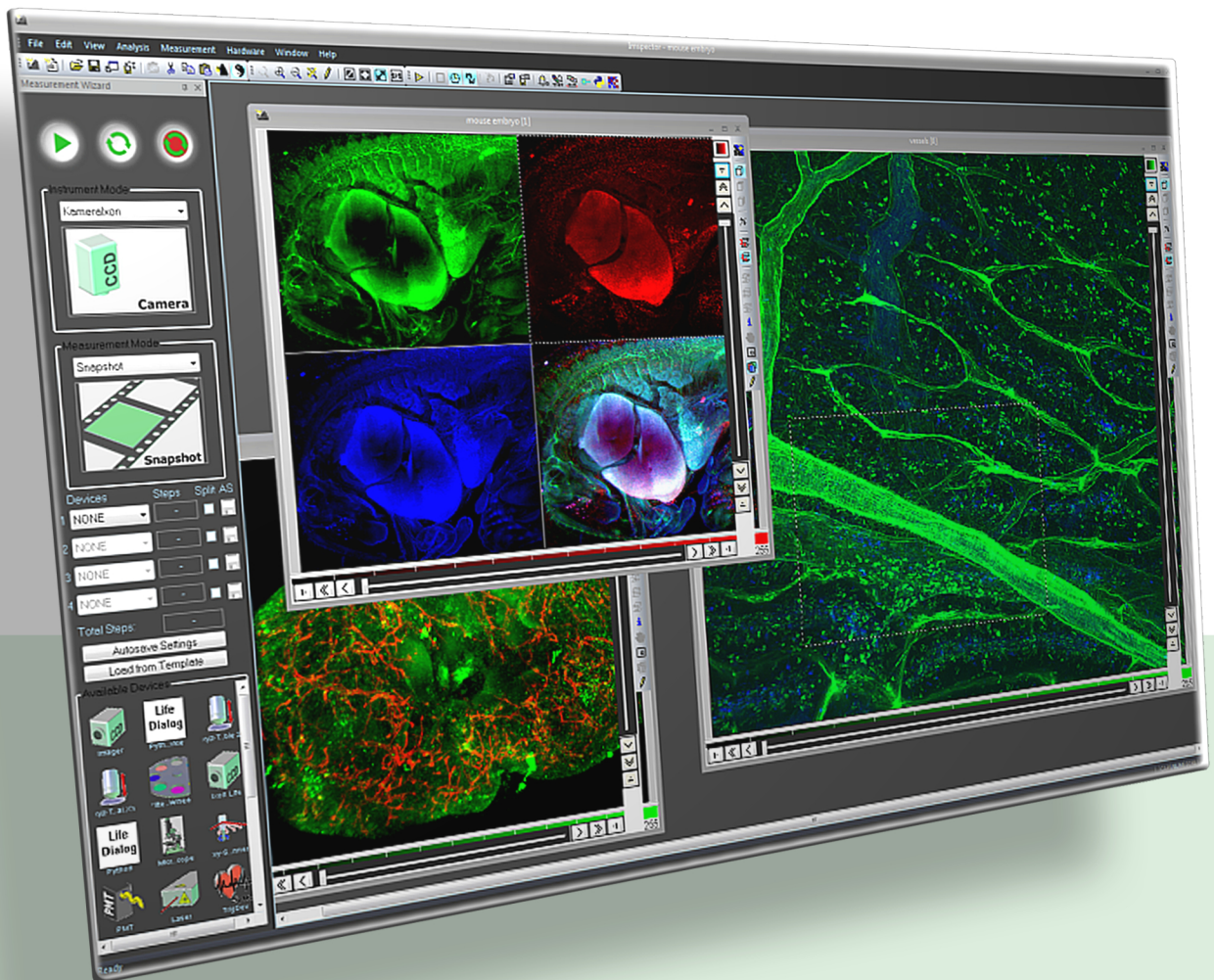


## ImSpector *Software Manual*





# Contents

<b>1</b>	<b>Introduction to ImSpector</b>	<b>1</b>
<b>2</b>	<b>Getting Started</b>	<b>2</b>
2.1	Installation	2
2.2	Launching ImSpector	2
2.3	User Interface	2
<b>3</b>	<b>Hardware Implementation</b>	<b>3</b>
3.1	Register Devices	3
3.2	Configure Devices	4
3.3	Save Hardware Settings	5
<b>4</b>	<b>Measurement Wizard</b>	<b>6</b>
4.1	Features	6
4.2	Scan Axis Selection	6
4.3	Start / Stop / Grab	6
4.4	Measurement Step Display	6
4.5	Recording Configuration	6
4.5.1	RECONFIG.INI	6
4.5.2	AXISCONFIG.INI	7
<b>5</b>	<b>Viewing Acquired Data</b>	<b>9</b>
5.1	The Measurement Window	9
5.1.1	Image and Frame Size	9
5.1.2	Zoom Functions	9
5.1.3	The Toolbar	9
5.1.4	Change View	11
5.2	Viewing Multidimensional Data Stacks	11
5.2.1	Choosing the Display Mode	11
5.2.2	Setting the Hidden Axis	11
5.3	Region of Interest (ROI)	12
5.3.1	Defining and Handling a ROI	12
5.3.2	Processing the ROI Content	12
5.4	Data Presentation	13
5.4.1	Choosing a colormap	13
5.4.2	Choosing the ideal colormap Range	13
5.4.3	Keep Colormap Scaling	13
5.5	Handling Several Data Stacks in a Single Window	13
5.5.1	Gallery Mode and superimposition	13
<b>6</b>	<b>Data Evaluation</b>	<b>18</b>
6.1	Profile Types	18
6.1.1	2D-profiles	18
6.1.2	3D-profiles	18
6.2	Profile Generation	19
6.3	The Profile Window	19
6.3.1	Setting Data Axes Ranges	19
6.3.2	Choosing the Profile Style	19
6.3.3	Zoom Function	19
6.3.4	Using Limits	19
6.4	Graph Evaluation	20

6.5	Copy and Paste Profiles	20
<b>7</b>	<b>Handling Data</b>	<b>21</b>
7.1	Type of Document	21
7.2	Generate New Documents	21
7.3	Autosave	21
7.3.1	Activating Autosave	21
7.3.2	Standard Autosave Settings	22
7.3.3	Advanced Autosave Settings	22
7.3.4	Using Autosave	23
7.3.5	autosave examples	24
7.4	Load and Save Data	24
7.4.1	Save Data	24
7.4.2	Export to File	24
7.4.3	Loading or Importing Data	25
7.4.4	Export to AVI	25
7.4.5	Export to MPEG	25
7.5	Copy Data in New Windows	25
7.6	Export To File	26
7.7	Tiff Export Dialog	26
7.8	Exporting to Ome Tiff	26
7.9	Measurement Information	26
7.10	Batch Conversion	28
7.11	Export All	28
7.12	Display Shadow Cursors	29
<b>8</b>	<b>Handling Windows</b>	<b>29</b>
8.1	Editing a Windows Size and Position	29
8.2	Organisation of Several Windows	29
<b>9</b>	<b>Description of Devices</b>	<b>30</b>
9.1	Scanner	30
9.1.1	Introduction	30
9.1.2	XY-Scanner Dialog Window	30
9.1.3	Single Beam Scanning with EOM	39
9.1.4	The EOM Dialog Window	41
9.2	Cloud Scanner	44
9.2.1	Life Dialog	44
9.2.2	Hardware Dialog	45
9.3	Horizontal Two Photon microscopy	46
9.3.1	Life Dialog	46
9.3.2	SLOT mode	46
9.3.3	Dual-Hemisphere mode	47
9.3.4	Full 360° mode	47
9.3.5	Advanced Settings pull-down menu	48
9.3.6	Adjust	48
9.3.7	Setup	49
9.3.8	Hardware Dialog	50
9.4	Motorized XY Table	51
9.4.1	Visual Scan Life Dialog	51
9.4.2	The Stitching Dialog	58
9.4.3	XY Position Scan Life Dialog [old dialog]	58
9.4.4	XY Mosaic Scan Life Dialog [old dialog]	59

9.4.5	Hardware Dialog	60
9.5	Motorized Z Stepper	61
9.5.1	Life Dialog	61
9.5.2	Hardware Dialog	61
9.5.3	Z Power Adaptation	62
9.6	Pifoc Z Stepper	63
9.6.1	Life Dialog	63
9.6.2	Hardware Dialog	64
9.7	FilterWheel	64
9.7.1	Life Dialog	64
9.7.2	Hardware Dialog	65
9.8	Time Driver	65
9.8.1	Details of the dialog	65
9.9	Imager QE	66
9.9.1	Life Dialog	66
9.10	Ixon	68
9.10.1	Life Dialog	68
9.11	Pixelfly	70
9.11.1	Life Dialog	70
9.11.2	Hardware Dialog	70
9.12	Microscope Device	71
9.12.1	Life dialog	72
9.12.2	Hardware dialog	73
9.12.3	ini file	74
9.13	Autofocus	74
9.13.1	General Settings	74
9.13.2	Creating reference profile	75
9.13.3	Calculation modes	77
9.13.4	Additional Settings	78
9.14	Optical Delay	78
9.14.1	Hardware Dialog	79
9.14.2	Life Dialog	80
9.15	TCSPC	82
9.15.1	Life Dialog	82
9.15.2	Hardware Dialog	83
9.15.3	Troubleshooting	85
9.16	Trigger Sequencer	87
9.16.1	Life Dialog	87
9.16.2	Hardware Dialog	89
9.16.3	Examples	90
9.17	Andor Mosaic	95
9.17.1	Life Dialog	95
9.17.2	Hardware Dialog	97
9.17.3	Trigger Sequencer settings	98
9.17.4	Python	99
9.18	LED	100
9.18.1	Life Dialog	100
9.18.2	Hardware Dialog	101
9.18.3	Python interface	101
9.19	Adaptive Optics	103
9.19.1	Hardware Dialog	104
9.19.2	Life Dialog	105

9.19.3 Python Interface . . . . .	109
9.20 Python Virtual Device . . . . .	112
9.20.1 The measurement loop . . . . .	113
9.20.2 Hardware Dialog . . . . .	114
9.20.3 Life Dialog . . . . .	114
9.20.4 Methods Parameters . . . . .	116
9.20.5 Inspector Methods For Python Device . . . . .	117
9.20.6 Examples: Generating Image Data, Using Own Serial Device . . . . .	120
9.20.7 Error handling . . . . .	124
9.20.8 Add Instrument Mode & Measurement Mode . . . . .	125
<b>10 Description of Plugins . . . . .</b>	<b>126</b>
10.1 Arithmetics . . . . .	126
10.1.1 Arithmetic Operations on a Single Data Stack . . . . .	126
10.1.2 Arithmetic Operations for Combining Two Data Stacks . . . . .	127
10.2 Delete Negative . . . . .	127
10.3 Subtract Background . . . . .	127
10.4 Interpolation . . . . .	127
10.5 Smoothing . . . . .	128
10.5.1 How it works . . . . .	128
10.6 Exponential Fit . . . . .	128
10.7 Filter Box . . . . .	131
10.8 The mean filter in detail . . . . .	132
10.9 The median filter in detail . . . . .	132
10.10 Fit Session . . . . .	133
10.10.1 Example . . . . .	133
10.11 Profile Fit . . . . .	135
10.12 Profile Multiply . . . . .	135
10.13 Spectral Unmixing . . . . .	135
10.14 Image Series Viewer . . . . .	139
10.14.1 Starting the plugin . . . . .	139
10.14.2 Opening a series . . . . .	139
10.14.3 The control panel . . . . .	141
10.14.4 Importing a series - the Import tab . . . . .	141
10.14.5 Encoding a video from a series - the Video tab . . . . .	142
10.14.6 Exporting a (Ome-)Tiff from a series - the Convert tab . . . . .	142
10.14.7 Stitching a series - the Import tab again . . . . .	143
10.14.8 The Slider Control(s) . . . . .	143
10.15 Averaging . . . . .	145
10.15.1 Using the plugin . . . . .	145
10.16 Phasorplot . . . . .	146
10.16.1 Profile basics . . . . .	146
10.16.2 Start . . . . .	147
10.16.3 Plot properties . . . . .	147
10.16.4 Phasor correction . . . . .	149
10.16.5 Data . . . . .	152
10.16.6 Create Stack . . . . .	153
10.17 Laser Adjustment . . . . .	155
10.17.1 Thorlabs . . . . .	155
10.17.2 Lavisision Biotec . . . . .	156
<b>11 List of Shortcuts . . . . .</b>	<b>158</b>

<b>12 Python Script Editor</b>	<b>160</b>
12.1 Introduction	160
12.2 The Script Editor	160
12.3 Inspector lvbt Python Module	163
12.3.1 Overview	164
12.3.2 Global Functions	164
12.3.3 Classes	165
12.4 Script Examples	174
12.4.1 Script Example - Two Measurements	174
12.5 The PySerial extension	175
12.5.1 Using the PySerial extension	175
12.5.2 Example: Create a trigger with PySerial and ImSpector	178
<b>13 Appendix</b>	<b>179</b>
13.1 How To Stitch Inspector Mosaic Image Data With Fiji	179
13.1.1 ImSpector Set Up	179
13.1.2 Stitching Data With Fiji	179
13.2 Python	181
13.2.1 How To Install Numpy	181
13.2.2 Inspector IS Module	182
13.3 Inspector Python Commands	184
13.3.1 IS.RunMeasurement(measurement)	184
13.3.2 IS.SaveMeasurementData(measurement, path, [filename])	184
13.3.3 IS.SetProperty( measurement, property, integer/float/string value)	184
13.3.4 IS.GetProperty( measurement, property, string value)	185
13.3.5 IS.Say(message)	185
13.3.6 IS.ExportToTiff(sourcePath, destinationPath)	185
13.3.7 IS.GetFile()	186
13.3.8 IS.GetDocFileName()	186
13.3.9 IS.GetData([name])	186
13.3.10 IS.GetDataOfWindow(window)	187
13.3.11 IS.GetGraph([name])	188
13.3.12 IS.GetGraphs([name])	188
13.3.13 IS.GetStackDims([name])	188
13.3.14 IS.CreateStack(name,size,type,[window])	189
13.3.15 IS.CreateGraph(name,size,[window])	189
13.3.16 IS.CreateProfile(window,stack,mode,rect)	189
13.3.17 IS.SetLength(length,[name])	190
13.3.18 IS.SetOffset(offset [name])	190
13.3.19 IS.GetStackNames()	191
13.3.20 IS.GetStackNamesOfWindow(window)	191
13.3.21 IS.AnnounceChange(name)	191
13.3.22 IS.Ask(message)	191
13.3.23 IS.GetInt()	192
13.3.24 IS.GetFloat()	192
13.3.25 IS.GetString()	192
13.3.26 IS.MoveZ(device,position)	192
13.3.27 IS.GetTablePosition('device')	193
13.3.28 IS.SetTablePosition('device',x,y,max_movement)	193
13.3.29 IS.ExtractROIs(name,source window,source stack,mode, [destination window])	193
13.3.30 IS.GetROIPosition(window,stack)	194

13.3.31 IS.GetROIPositions(window,stack) . . . . .	194
13.3.32 IS.GetLineScanPositions([stack]) . . . . .	194
13.3.33 IS.ClearOutput() . . . . .	194
13.3.34 IS.SetComment(comment) . . . . .	195
13.3.35 IS.Info() . . . . .	195
13.3.36 Screenshots . . . . .	196
13.4 ImSpector Multi User Setup . . . . .	197
13.5 axisconfig.ini templates . . . . .	198
13.5.1 Burst . . . . .	198
13.5.2 Pifoc . . . . .	199
13.5.3 Snapshot . . . . .	202
13.5.4 Stitching . . . . .	204





# 1 Introduction to ImSpector

ImSpector is designed for data acquisition and evaluation. It allows the implementation and control of various hardware devices like CCD cameras, spectrometers, motorized stages, etc. A special feature is the acquisition and presentation of up to 4-dimensional data sets. A large number of fully automated measurements can be defined. ImSpector offers various evaluation features including profile generation, simple arithmetics, complex functions and fitting routines. Another special feature is that the evaluation can be executed during running measurements.

## 2 Getting Started

### 2.1 Installation

ImSpector is installed and configured on the PC delivered with the measurement system. There is a setup program on the ImSpector CD, if you need to carry out a reinstall or an update. Just unpack the zip file to a directory on your hard disk, start the setup.exe and follow the instructions. Reinstall and update will not change any of your config files but will update drivers, plugins and the main program files.

### 2.2 Launching ImSpector

To launch the software double-click with the left mouse button on the ImSpector icon on the desktop or select ImSpector from the menu Start→Programs→ImSpector. If there is no icon present ImSpector can also be started by double-clicking the file ImSpector.exe in the folder \ImSpector on your hard disk. If you start ImSpector for the first time, you will be asked for a registration key. Please copy the key from the RegKey.txt file of your ImSpector CD into the corresponding field and press OK.

### 2.3 User Interface

The screenshot in figure 1 shows the user interface of ImSpector.

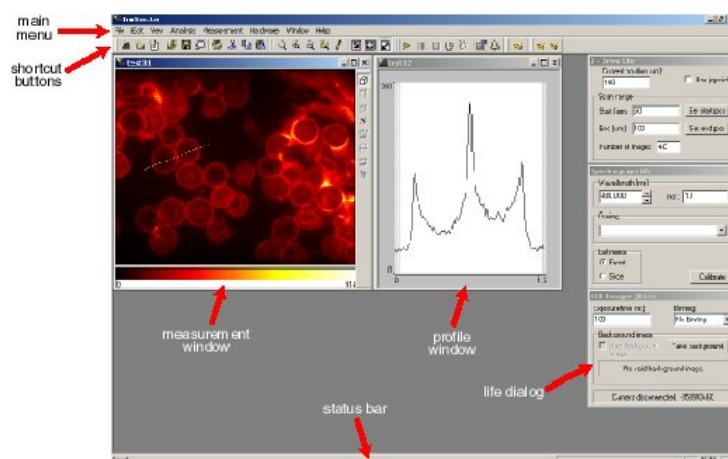


Figure 1: Overview of the ImSpector workspace.

The top horizontal row shows the main menu where most features of ImSpector can be selected. Beneath is the toolbar with shortcut buttons for a fast selection of commonly used features. The row at the bottom of the user interface is the status bar. Located in the upper left corner of the workspace is a measurement window which displays acquired data. A profile window is attached to the right side of it showing an intensity profile along the dashed line visible in the measurement window. At the right side of the screen the life-dialogs are positioned. They display the status and the position of the hardware devices used for measurements.

### 3 Hardware Implementation

This chapters explains how to implement and configure hardware devices that should be controlled by ImSpector.

#### 3.1 Register Devices

Each device must be registered in ImSpector before it can be used in experiments. This has already been done for the devices delivered with the system. To add (or remove) a device select the dialog Register Devices from the popup menu Hardware. The upper box contains a list of the device drivers that are currently loaded on start-up, the lower box shows the currently registered devices. To add a device, mark the corresponding driver, press the button New Device... and enter a name for the new device. To remove a device select it from the 'Registered Devices' list and press Del Device... .



Figure 2: Register available drivers.

The order of the registered devices can be changed by clicking the arrow buttons on the right side. Changing this order also changes the order of the dialogs shown in the ImSpector main window. To run the software correctly please note that not all orders are allowed. The following device order should be used:

Position	Device
1	SLOT
2	TrimScope
3	XY-Scanner
4	ResonantPMT
5	PMT
6	Cameras
7	XYZ-Table
8	other devices here...
...	...
n-1	adaptive optics
n	time

Adding a device to the list of available devices can be done with the 'Edit Driver Config' from the menu Edit.

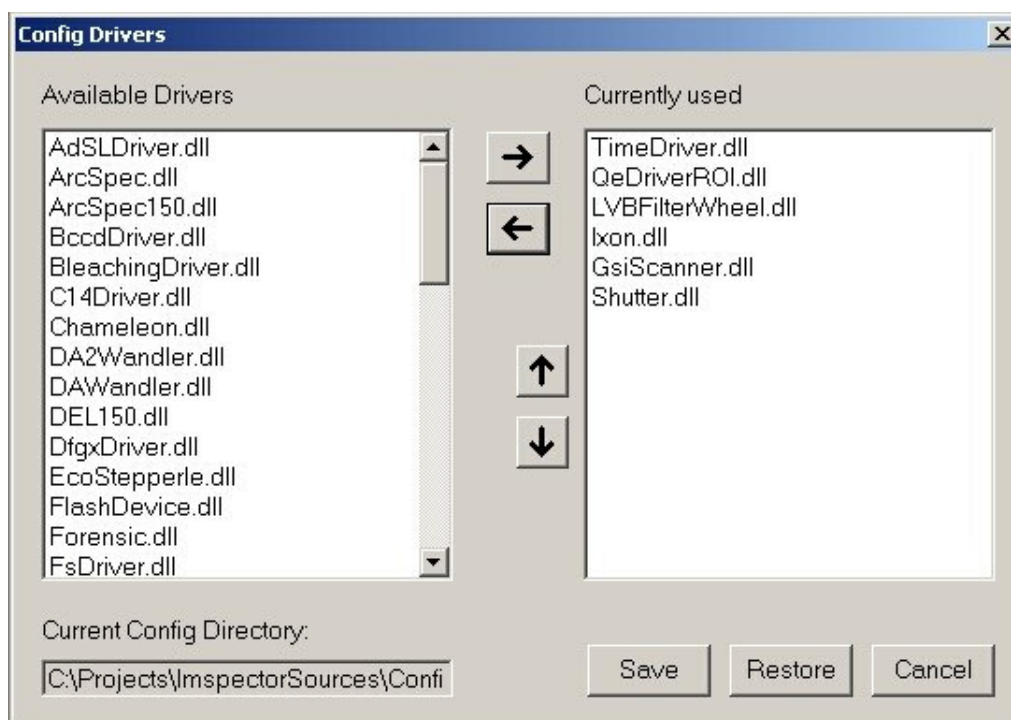


Figure 3: Driver Selection Dialog.

You can control which device drivers are loaded on startup by changing the file plugins.ini in the config directory. To do this select the dialog Edit Driver Config from the menu Edit. The list to the left shows all available drivers, the list to the right the drivers that are currently loaded on startup. To add a driver double click on an entry in the left list or mark it and press the arrow right button. To remove a driver double click on an entry in the right list or use the arrow left button. When you have finished press Save. The first time you change the plugins.ini file, a backup is made (plugins.bak) so that you can always return to the first configuration (by pressing Restore). You have to restart the program for the changes to take effect.

### 3.2 Configure Devices

In order to operate properly each registered device has to be configured (for system customers this is already done once during installation). To change the settings of a device

open the dialog Hardware Properties (Menu: Hardware→Configure...) see figure 4. The dialog contains one tab page for each registered device, where all necessary settings can be made. To test the actual configuration press Initialize. An error message will appear, if the initialization failed.

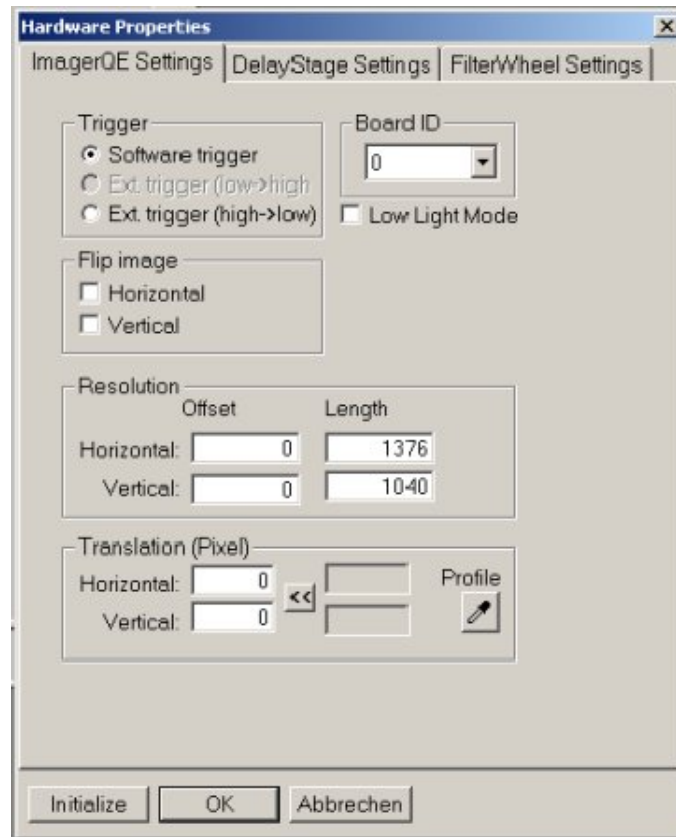


Figure 4: The Hardware Properties Dialog.

### 3.3 Save Hardware Settings

Once all hardware devices have been registered and configured properly the current status should be saved (Menu: Hardware→Save Settings). Inspector will then recall the settings the next time it is started.

## 4 Measurement Wizard

### 4.1 Features

The Measurement Wizard (see figure 5) offers some efficient methods to support using the 4D-measurement concept. It is accessible through Imspectors main menu (Measurement→Wizard):

- Quickly switching detection configuration (e.g. Single Beam, CCD, Eye, ...)
- Grabbing without loosing axis configuration
- Easy access to Axis LifeDialogs
- Measurement step display during measurement

### 4.2 Scan Axis Selection

As in the document settings dialog all available scan axis are selectable in 4 combo boxes. Corresponding LifeDialogs can easily be opened with the tool buttons at the bottom. A second click on the tool button iconifies the LifeDialog. During measurements the axis selection controls are disabled.

### 4.3 Start / Stop / Grab

Measurement Control now includes the Grabbing function. This temporarily deactivates all defined scan axis and starts the measurement. As soon as the measurement terminates (stop button or imaging complete) the axis configuration is recovered for further use. Note that the repeat button in the Inspector Toolbar may be useful for true grabbing.

### 4.4 Measurement Step Display

For better keeping track on the measurement progress the Wizard displays per axis the total steps and the steps already done. Also total acquisition steps are displayed (see figure 5).

### 4.5 Recording Configuration

The central point with the Wizard is selecting a recording configuration. That means keeping axis settings as they are, one can select a different set of data acquisition devices. For example switching between single beam and camera detection is one click. Each entry inside the Data Acquisition control refers to a set of devices to be activated and a set to be deactivated. Furthermore for each predefined configuration certain properties can be set. That could be the number of beams set to "1" for the single beam set and "64" for the camera set. See the next section (reconfig.ini) for details.

#### 4.5.1 RECONFIG.INI

All instrument configurations are defined through a text based ini file. Place the reconfig.ini file into Your Inspector Config directory. To find/set this, select from the Inspector main menu "Edit → Set Config Dir". Systems delivered by LaVision BioTec GmbH are shipped with proper ini-files.

```

1 [REC00]
2 LABEL=Sensicam QE (low light)
3 SWITCHON=XY-Scanner , TriMScope , Imager QE
4 SWITCHOFF=PMTs
5 DLGOFF=xyz-Table XY
6 DLGON=PMTs
7 ICONFILE=singleccd.bmp
8 PROPSET_00=Imager Exposure Time , float , 200.0
9 PROPSET_01=Imager Y Bin , int , 1
10 PROPSET_02=Imager low light mode , int , 1
11 COMMENT=Activates <b>XY-Scanner , TriMScope
</b> and <b>Imager QE</b> for imaging. CCD set
to internal trigger mode<p><b>Important:</b><ul><li>Set
beam number to 64 or 32</li><li>Chose CCD scan mode
</li><li>Adjust CCD integration time</li></ul></b>
12
13 [REC01] ...

```

As in the example there are various options to set for a single configuration

LABEL is the name of the configuration. You will find this name selectable in the dialog.

SWITCHON defines a coma separated list of devices to activate for that configuration. Note that device names have to be case sensitive and before/after commas musn't be additional white spaces.

SWITCHOFF is the corresponding list of devices to be deactivated.

DLGOFF allows to hide specific life dialogs independet of the device named by it's title.

DLGON activates the dialogs of deactive devices.

ICONFILE points to a bitmap file with 172x86 pixels. Path is relativ to Your Imspector config directory.

PROPSET\_XY sets a int, float, double or CString property (identified by its label) to a certain value. Again avoid additional white spaces before/after commas. Entries here must be consecutive (PROPSET\_03 will only be executed if PROPSET\_00, PROPSET\_01 and PROPSET\_02 exist). The properties name is followed by the property type int, float, double, CString and a value.

PROPRESET\_XY is applied when leaving the configuration to set a (safe) default value. Same parameters as with PROPSET.

COMMENT contains a html formatted text to be displayed as a description in the lower area of the wizard dialog.

#### 4.5.2 AXISCONFIG.INI

The measurement configurations are defined in the axisconfig.ini file which is located in the Config directory. The axis which are running during the measurement are specified here. This is an example:

```

1 [AC00]
2 LABEL=Color Timelaps
3 AXIS_FIRST=Filter Wheel F
4 AXIS_SECOND=Time Lapse Time
5 AXIS_THIRD=
6 AXIS_FOURTH=

```



Figure 5: Measurement Wizard.



```
7 ICONFILE=Filter_150.bmp
8 COMMENT=Multi color imaging with time laps
9 RECCONFIGS=REC00
10 PROPSET_00=xy-Scanner Modes,CString," "
11
12 [AC01] ...
```

## 5 Viewing Acquired Data

### 5.1 The Measurement Window

Acquired data is presented in the measurement window.

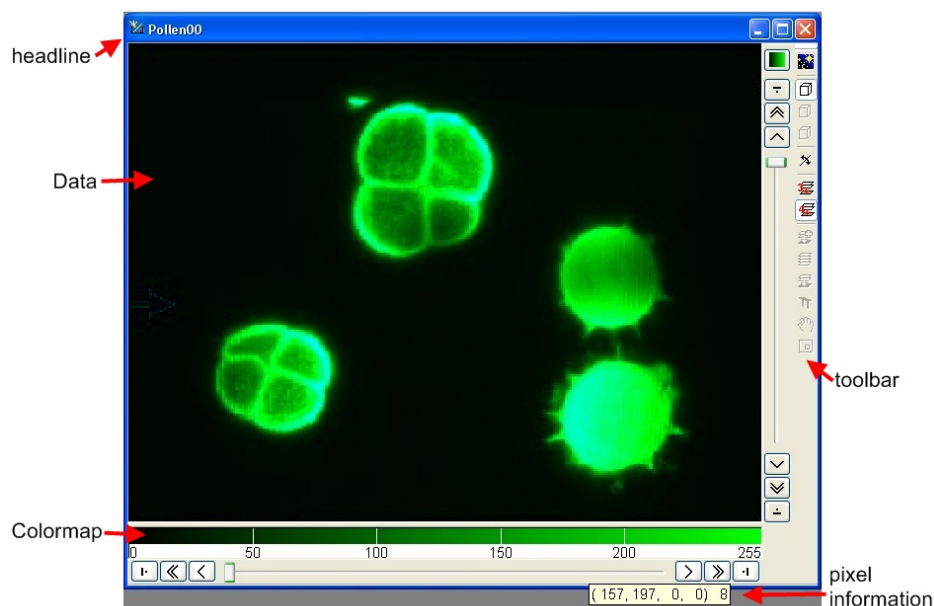


Figure 6: Measurement Window

The row beneath the data shows the colormap which is used to display different pixel values in different colors. The toolbar located at the right side of the measurement window contains buttons for frequently used data handling functions. The stack name appears in the headline and the status bar at the bottom of the user interface. The information for each pixel of the acquired data can be obtained by positioning the mouse pointer on it. It appears in the right corner below the measurement window and states the position in the data stack and the pixel value. If the mouse is rested at a pixel the user gets more detailed information on the different data axes. A right click on the colormap invokes the *colormap menu*. A right click on the acquired data calls up the *data menu*.

#### 5.1.1 Image and Frame Size

To adjust the size of the data in the measurement window and the surrounding frame go to the index card window. The function fit image to frame maximises the size of the image within a frame of constant size. The function fit frame to image keeps the image size constant and fits the frame to the image. These functions keep the ratio of the image axes constant. If the button don't preserve aspect ratio is selected the frame-size can be varied by dragging it with the mouse while the image always fits in the frame.

#### 5.1.2 Zoom Functions

To reveal more details in the acquired data the user can zoom in or zoom out step by step by selecting the appropriate command in the data window menu. In addition the user can zoom in to a previous selected ROI.

#### 5.1.3 The Toolbar

The toolbar of a document (see figure 7) allows some changes of the view on the data.

1. Fit min and max of the colormap. Same result as the shortcuts F9 and F10 one after another.
2. This button allows to get the familiar view at a data stack. The first (x) and the second (y) axis build one image and the third (z) axis reflects a stack of images. For further information see section 5.1.4.
3. This button changes the familiar view of a data stack. The y and z axis are swapped. For further information see section 5.1.4. This button is only usable if the z-axis is bigger than 1.
4. This button changes the familiar view of a data stack. For further information see section 5.1.4. This button is only usable if the z-axis is bigger than 1.
5. This button rotates the current image through 90°.
6. The current third axis becomes the forth axis.
7. The current forth axis becomes the third axis.
8. Adds up the images along the forth axis. Use button 6) or 7) if necessary.
9. Disables the add up or the projection mode of button 8) and 9).
10. Make a projection along the forth axis. The result image contains the highest intensity of every pixel along the forth axis.
11. Shows comment and meta data panel
12. Grab the image to "walk" over it. This button can only be used if the image is smaller than the windowframe.
13. Remember scan range. Same as shortcut ctrl+shift+r or the menu entry *Right mouse click on image* → *Selection* → *Set as New ROI*. Only usable if a rectangle is available.
14. Show Merged Channel: Displays all channels added up next to the single channels.



Figure 7:  
Toolbar.

## 5.1.4 Change View

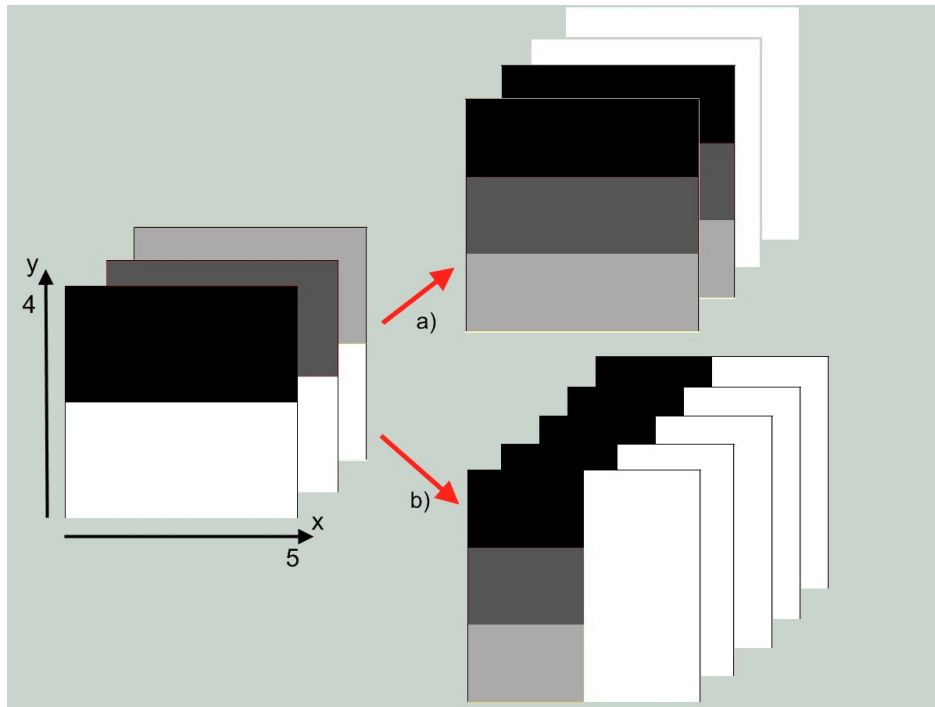


Figure 8: Change View.

Figure 8 shows an example of changing the view with the buttons on the document toolbar (see figure 7, button 2-4).

On the left side of the figure stands a data stack with the axis sizes:  $x = 5$ ;  $y = 4$ ;  $z = 3$ ;  $t = 1$ .

By using the third button at the document toolbar the result will be the data stack pointed by the arrow marked by a). The  $y$  and  $z$  axis swapped their position. Because of the size of the  $y$ -axis we get four images. The axis sizes are now:  $x = 5$ ;  $y = 3$ ;  $z = 4$ ;  $t = 1$ . Looking at the left data stack on figure 8 we now look at the top of the data stack. "Walking" to it's bottom we get four images, one for every pixel in  $y$ -direction.

After pressing the fourth button of the document toolbar the axis sizes changed to:  $x = 4$ ;  $y = 3$ ;  $z = 5$ ;  $t = 1$ . The result is the data stack pointed by the arrow marked by b). Looking at the origin data stack we now look at the right side of it. For this example we get the same image five times.

## 5.2 Viewing Multidimensional Data Stacks

### 5.2.1 Choosing the Display Mode

ImInspector allows the acquisition of up to 4 dimensional data stacks. Any two data axes can be displayed in the measurement window ( $xy$ ,  $yz$ ...). The mode is changed by selecting display from the data menu and activating the desired mode. The active selection is indicated by a tick.

### 5.2.2 Setting the Hidden Axis

Two of the four data axes are displayed in the measurement window while the other two are not visible. The user can choose one of these as the hidden axes which has different display modes. The function parse through allows the user to view successive planes defined by the two visible axes along the hidden axis. Add up means that the values of corresponding pixels in all data planes along the hidden axis are added and presented in a single image.

The function maximum intensity projection searches the highest value along the hidden axis for each pixel. These values are projected into a single picture.

### 5.3 Region of Interest (ROI)

ImSpector allows the selection of a part of the data presented in the measurement window. This part can be copied to a new measurement window to view, process and evaluate it independently from the rest of the data. This might be useful if the user is only interested in a small part of the data or if the size of the data stack is very big.

#### 5.3.1 Defining and Handling a ROI

In a datastack window might exist many kinds of ROIs. Several devices can create ROIs for special purposes. Every ROI is represented by a button at the toolbar right to the data window. Toogling these buttons will show or hide the corresponding ROI, leaving the mouse on it will popup a hint, which kind of ROI it is. All ROIs are shown in the datastack as yellow dotted line, the current one has a white dotted line. Sometimes it is usefull to deactivate all the ROI-buttons except the one which should be edited, to make sure the right ROI is handled.

An ROI can have different shapes. Inspector offers rectangles, ellipses, polygons and freeforms.

**Rectangles** A part of the data is selected by generating a rectangle with the mouse. To do this click at one pixel in the measurement window, keep the mouse button pressed and drag the mouse so that a rectangle appears. Release the mouse button to confirm the selection. To move the active selection to another area click on the field of the rectangle, keep the mouse button pressed and drag it to the desired zone. Release the mouse button to confirm the selection. The size of the rectangle can be varied by aiming at an edge, doing a mouse left click at this point, keeping the mouse button pressed and pulling the rectangle to get the desired size. The coordinates at the upper left and lower right edge show the position of the rectangle in the visible data plane. ImSpector remembers the size and the position of all previous defined rectangles. The menu selection from the measurement window menu contains all functions for rectangle handling. A rectangle can be copied from one measurement window to another by selecting copy rectangle, activating the destination window and choosing paste rectangle. All previous defined rectangles can be revoked in any measurement window by selecting previous or next from the selection menu.

**Other shapes** To switch to an different shape right click one of the drag handles of the ROI or right click on the ROI buttons in the toolbar. A menu will appear in which it is possible to change the shape. In that menu it is also possible to change the editing mode to resizing, rotating or to treat the ROI regulary. For most types of ROIs it is also possible to add or delete other regions.

#### 5.3.2 Processing the ROI Content

To copy the content of a rectangle to a new measurement window select copy, activate the destination measurement window and select paste. Both functions are located in the popup menu edit.

If the rectangle is selected in a 3- or 4-dimensional data set the corresponding part on the two invisible data axes is included in the selection. Copy and paste also transfer these dimensions.

## 5.4 Data Presentation

ImSpector uses colormaps to distinguish between different pixel values. The two values below the colormap show its range. The colormap context menu is activated by a click with the right mouse button on the colormap.

### 5.4.1 Choosing a colormap

The user can choose among several standard colormaps (fire, grayscale...). In addition it is possible to design custom colormaps. Custom colormaps can be loaded and stored. Red, green or blue colormaps can be assigned to images in order to generate RGB images by superimposing three different images (RGB channels). In addition the user can choose between linear and logarithmic colormap scaling. The brightness of RGB and grayscale colormaps can be adjusted with the function gamma correction.

### 5.4.2 Choosing the ideal colormap Range

By editing the numbers below the colormap the range can be customized. To do this double-click on a number, rest the mouse pointer on it and enter the new value. Further movement of the mouse confirms the new selection. In addition to customising the colormap range settings ImSpector offers several functions in order to optimise the range with regard to the displayed data. Fit maximum means that the upper boundary is fitted to the highest pixel value contained in the whole data stack. Fit minimum sets the lower boundary to the lowest pixel value. Fit selection maximum and fit selection minimum do the same but restricted to a previously selected ROI. Fit Min/Max and Fit Selection Min/Max fit both boundaries.

The range and color mode of the colormap can also be adjusted by using the Colormap Dialog from the context menu. The dialog shows a frequency distribution of all pixel intensities for the currently active stack (or several stacks, if in Gallery mode, see figure 11). Upper and lower boundary values for the active stack can be changed by moving the upper and lower limit bars in the graph view. To change the active stack, use the dropdown box or press the Space or Tab key. The colormap for the active stack can be set by pressing on one of the colormap images displayed to the right of the graph view.

### 5.4.3 Keep Colormap Scaling

If the user watches the progress of a running measurement it is sensible to automatically adjust the range of the colormap to the continuously changing data. The function keep colormap scaling effects that every time a new image is taken the colormap range is automatically adjusted to the highest and lowest pixel values.

## 5.5 Handling Several Data Stacks in a Single Window

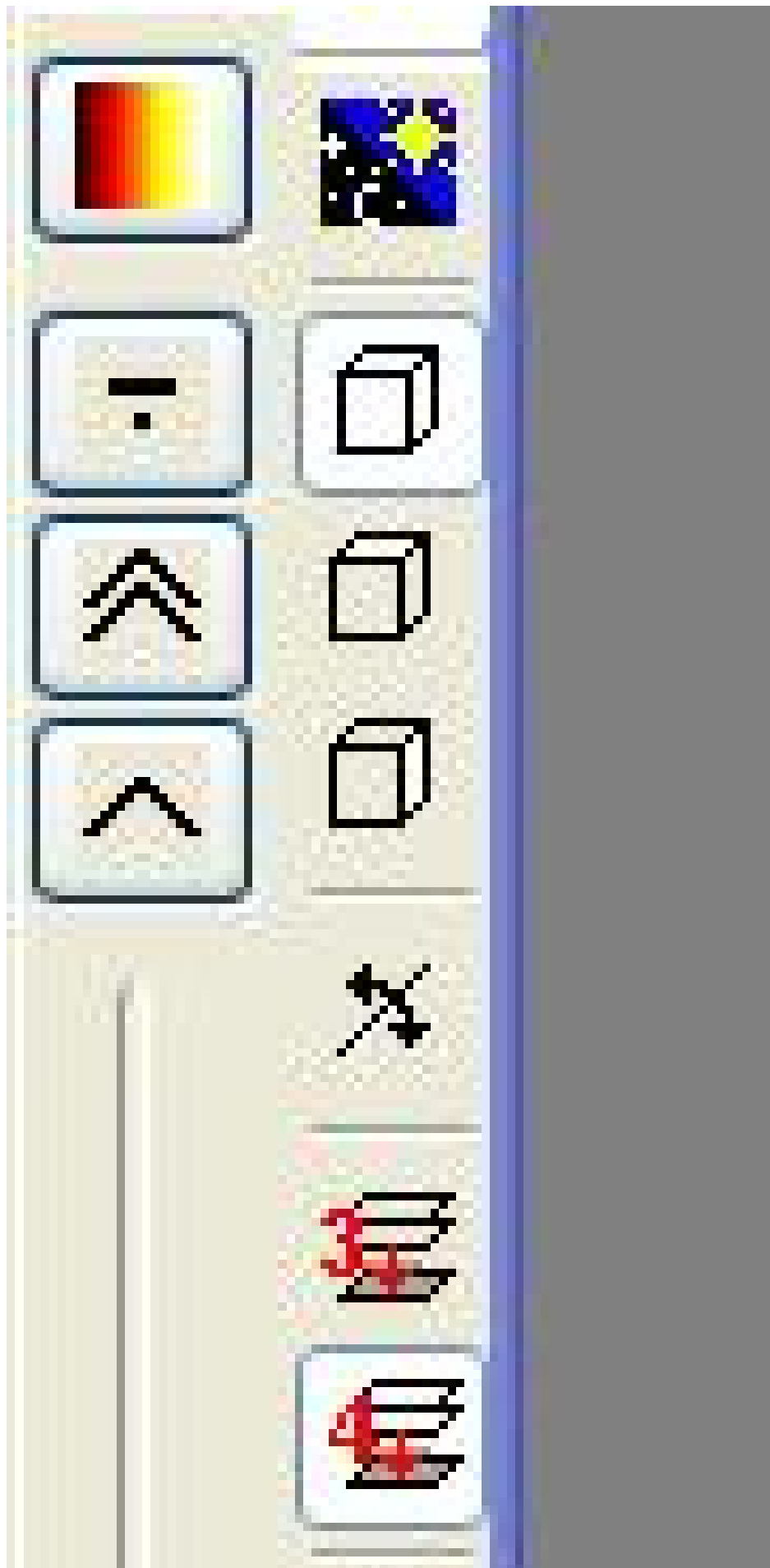
Several data stacks can be handled in a single data window. To generate a data window with different stacks in it simply copy and paste data stacks (or parts of data stacks) in the same data window. Gallery Mode and superimposition

### 5.5.1 Gallery Mode and superimposition

The active stack is indicated by a dashed frame. A stack can be activated by a left click with the mouse. The name of the stack appears in the status bar. The menu manage channels contains all functions for the organisation of different data stacks in a single window. Gallery mode is selected to view all data stacks side by side. Set gallery columns determines the arrangement of the stacks in the window. If one is selected the different stacks are arranged

in a vertical line. The function superimposition overlays data stacks so that images on the same position on the hidden axis are superimposed.

Equalize colormap assigns all data stacks contained in the window the colormap settings of the active stack. The function merge converts all one layered stacks contained in the data window into a single multi-layered stack. Split stack divides the stack along the hidden axis into single layered data stacks and displays them in a new data window in gallery mode.





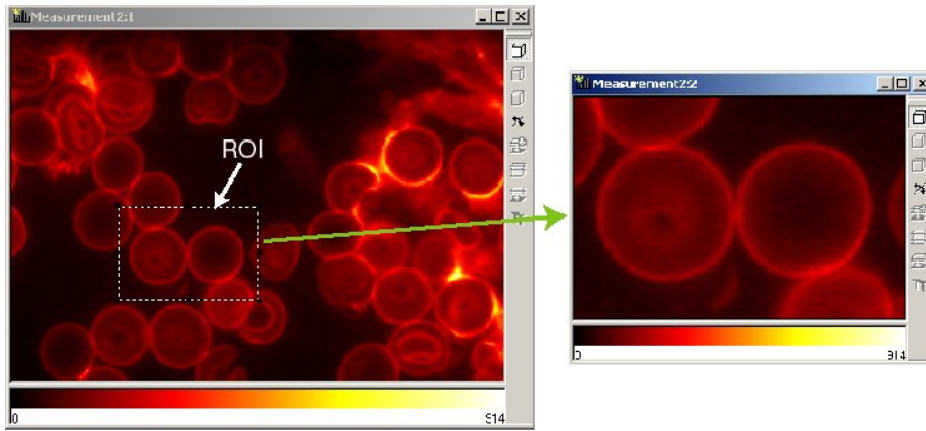


Figure 10: Defining a ROI

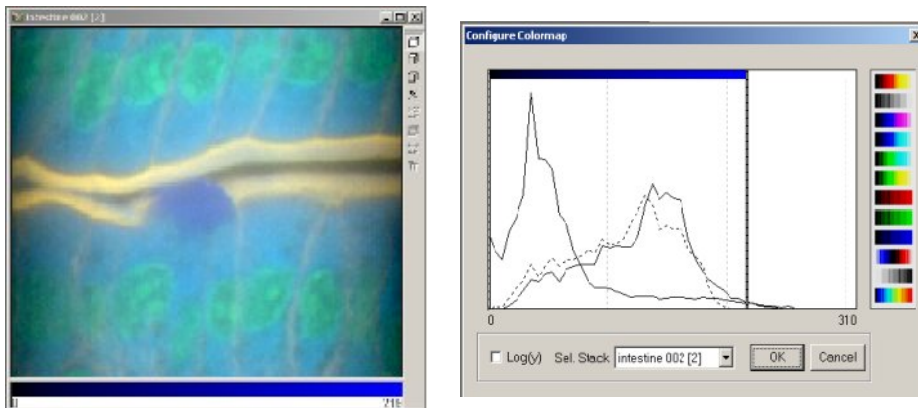


Figure 11: Configure the colormap.

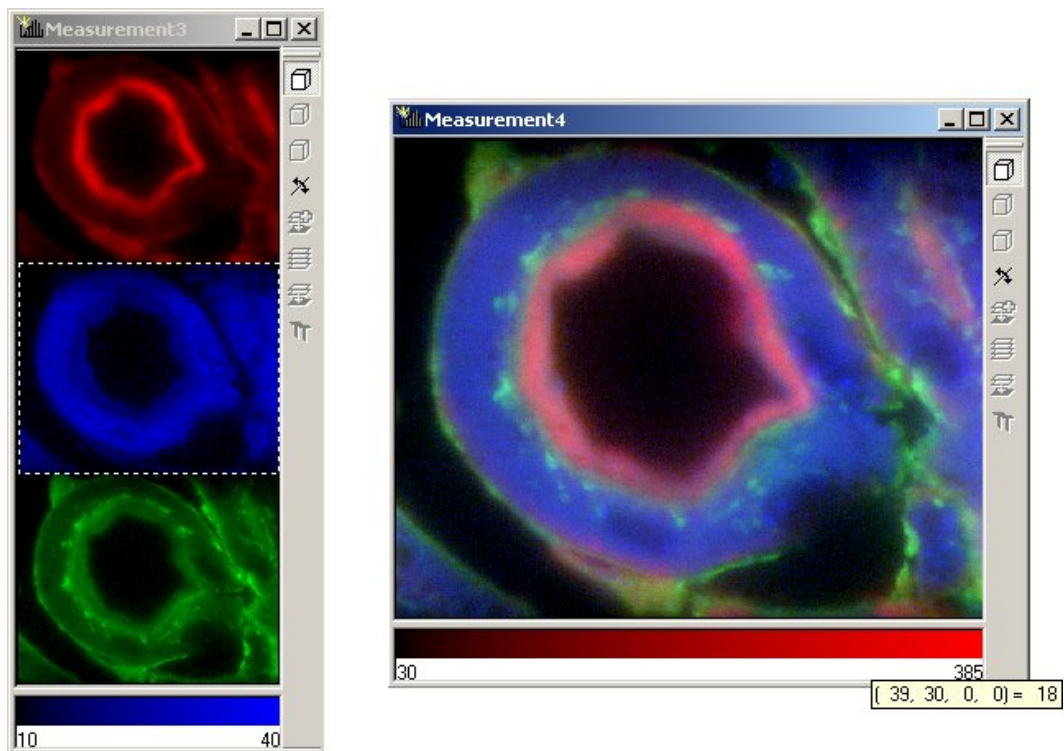


Figure 12: Measurement3 shows images of the same object taken with different filters in gallery mode. Measurement4 shows the same stack as a superimposition of the different stacks.

## 6 Data Evaluation

Profiles are used to view intensity distributions. All profile settings are located in the measurement window menu.

### 6.1 Profile Types

The user can choose among profiles operating on 2D- or 3D-data. A profile is generated by selecting a line or a rectangle in the measurement window. The associated profile window pops up at the right side of the measurement window.

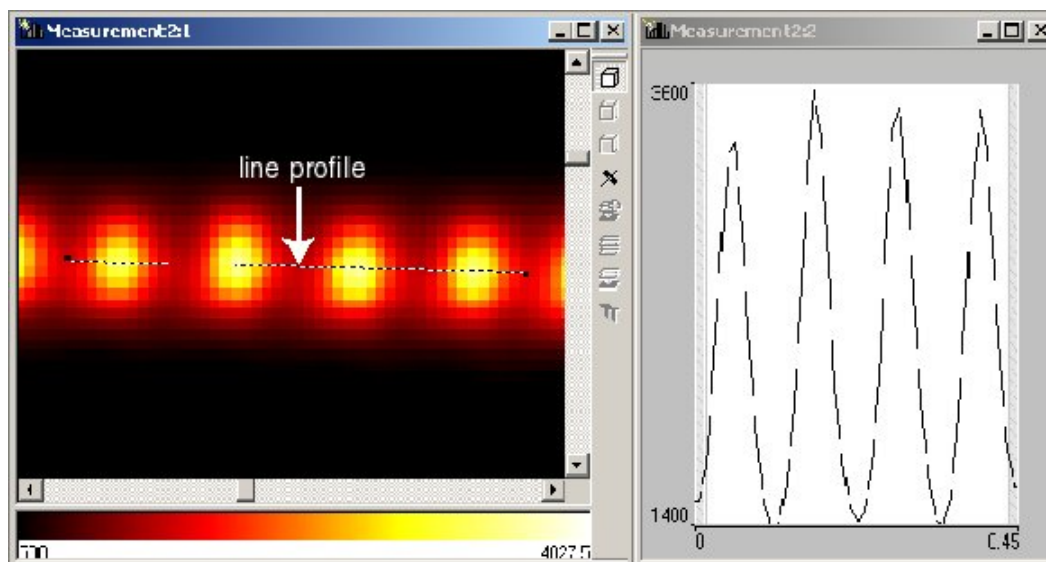


Figure 13: Drawing a profile.

In profiles based on rectangles the user has to choose among the options add pixel values or average pixel values.

#### 6.1.1 2D-profiles

2D-profiles operate on the plane defined by the two data axes visual in the measurement window. By selecting the appropriate display mode the profile can be used on any two data axes (xy, yz...). A line profile displays the pixel values along the selected line in the image. The function add x (add y) sums up all line functions in an interval x (y) and therefore requires the definition of a rectangle in the measurement window. All pixel values in x(y)-direction are added and assigned to the corresponding y(x)-value. The histogram function also requires the generation of a rectangle. It shows the occurrence of the different pixel values in the rectangle. Thereby the x-axis represents the pixel value and the y-axis how often the pixel-values are present. The evaluation of the histogram (average, median, variation, ratio, sum) is shown in a small box in the profile window.

#### 6.1.2 3D-profiles

3D-profiles require the generation of a rectangle. They operate on the two visual data axes and in addition on the hidden data axis. In each plane defined by the two visual data axes all pixel values in the selected rectangle are added (or averaged) and one value is assigned to each plane. The graph displays the summed (or averaged) pixel values per layer on the y-axis and the position on the hidden axis on the x-axis. The z-profile refers to the third data axis and the t-profile to the fourth data axis.

## 6.2 Profile Generation

A line profile is selected by choosing the desired start and end point. To define the position of the start point move the mouse pointer to a pixel in the image and press the left button. Then move the pointer the desired end point while the mouse button must be held. At the end point release the mouse button. The coordinates of the start and the end point of the line appear in the image. Once generated the line can be moved across the image by grabbing it with a mouse left click, holding the button and dragging the mouse. To drop the line release the button. The size and the direction of the line can be changed by clicking the start or the end point of the line and dragging the mouse pointer to the desired point. The selection of a rectangular profile is equivalent to the selection of a ROI.

## 6.3 The Profile Window

The profile window pops up at the right side of the data window directly after a profile is defined.

### 6.3.1 Setting Data Axes Ranges

To adjust the range of a data axis position the mouse pointer on the appropriate axis and press the right button. This invokes the associated data axis menu. The user can choose among linear or logarithmic axis scaling. The dialog enter display range allows the customised definition of the maximum and minimum value that will be displayed. Fit maximum and fit minimum effects that the maximum (minimum) data axis value is fitted to the highest (lowest) intensity contained in the profile. The function fit best always adjusts the axis scaling to fit the graph best (max (min) data axis value = max (min) graph value). The function grow lets the axis interval grow to accommodate all graphs.

### 6.3.2 Choosing the Profile Style

A profile must be activated before its properties can be edited. To activate a profile move the mouse pointer on it and perform a double click with the left button. The profile turns from solid into dashed and a cross-hair appears. A right click on the profile window invokes the profile menu. The dialogue display style allows the user to choose the color and the style (dashed, dotted...) of the graph. This is useful when working with different graphs in a single window. By choosing rename the user can edit the name of the profile.

The dialogue change graph size allows the variation of the axis length, the offset, the amount of pixel and the paste position when inserting a new graph in the profile window.

### 6.3.3 Zoom Function

To have a detailed view on a part of the profile choose select rectangle from the profile menu and enframe the area of interest. The command zoom to selection confines the view to the selected part. To zoom out press the button reset zoom from the shortcut bar.

### 6.3.4 Using Limits

The user can define vertical limits in order to select only a part of the profile. When inactive the limits are visible as thin vertical bars at the left and right side of the profile window. They can be activated by choosing limits→toggle on/off from the profile menu. Once activated they turn their color from grey to red. To adjust the position move the mouse pointer on one of the bars and grab it with a click on the left mouse button. Drag the bar to the desired position while holding the button pressed and release the button. By choosing zoom to selection the view is restricted to the horizontal area between the two boundaries.

## 6.4 Graph Evaluation

A profile has to be activated to get further details about it. The coordinates of the crosshair which indicates the position on the graph appear at the right bottom of the graph window (distance in pixel, pixel value). Tools for further analysis of the profile can be found in the menu analysis. The function label extrema marks all maxima and minima contained in the selected part of the graph. The values of the maxima can be shown as absolute values or as percentage of the highest value contained in the graph. The settings in prominence define how high a peaks must be to be regarded. The noise filter determines the distance of neighbouring peaks. To compare different maxima the user can toggle on a grid (show grid) in the profile window consisting of horizontal lines. In grid settings the number of grid lines can be entered. Normalise all graphs normalises all graphs contained in the profile window to one. Make stack generates a 2D-image out of the profile.

## 6.5 Copy and Paste Profiles

It is often useful to keep a profile or view different profiles in the same profile window. To do this generate a new profile window by selecting file→new→profile window. Then activate a graph that should be kept or compared to other graphs. To copy it into another profile window choose copy from the menu edit. Then activate the new profile window (left click on the frame) and choose paste. This procedure allows to accumulate several graphs in the same profile window.

## 7 Handling Data

The popup menu file contains all commands to load, save or print acquired data or profiles.

### 7.1 Type of Document

Depending on the type of data ImInspector uses three different kinds of windows. A measurement window can be used for all kind of actions like measuring, viewing data or evaluation. A data window can handle the same kind of data as a measurement window and possesses all viewing or evaluation features. The difference in comparison to a measurement window is that a data window can not be used for measurements. A graph window is used to display profiles that were generated by evaluation functions.

### 7.2 Generate New Documents

New documents can be generated by selecting new and choosing the desired kind of document. The user can create independent documents or documents which are bound to already existing data or measurement windows. Independent documents are generated by choosing new measurement or new data file. If another window should be added to an existing file choose new stack window for a window that displays images or new graph window for a window that displays profiles. This kind of window is bound to the window that was activated before generation. The window type and the name are shown in the headline of the window. The name has the structure (file name : number of window)

### 7.3 Autosave

Some measurements can exceed the main memory of your computer, shown by the message *“Not enough memory”*, when starting a measurement. To still run the measurement you can activate the *“autosave”* function of ImInspector. With *autosave* the measurement results are not stored in the main memory of the computer, but are saved directly to your hard disc as a series of tiff files.

With *autosave* there's no limitation in measurement sizes.

#### 7.3.1 Activating Autosave

You can activate *autosave* by clicking *“Autosave Setting”* in the *Measurement Wizard*.

There are three general settings:

- *No Autosave*: Deactivates all *autosave* functions.
- *Open Save Routine after measurements*: After a measurement the standard file saving routine opens automatically.
- *Save Every Step*: Activates the *autosave* function

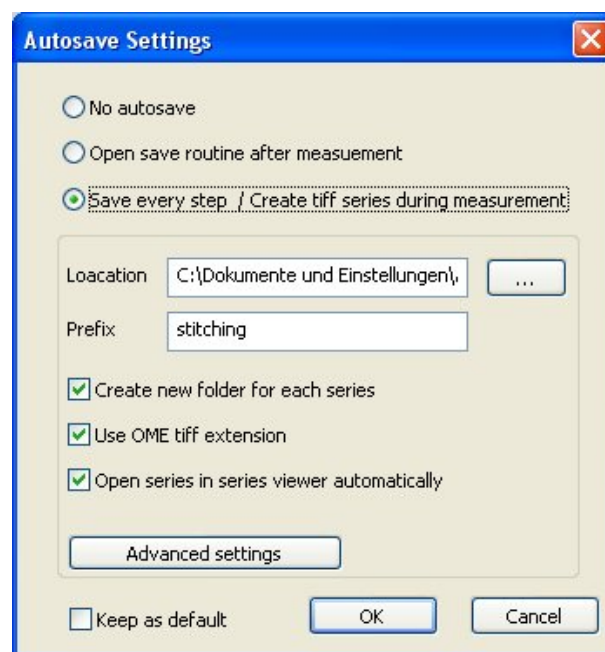


Figure 14: Autosave Settings

### 7.3.2 Standard Autosave Settings

After selecting *Save Every Step* you can specify your *autosave* settings:

- *Location*: shows the directory where the tiffs should be stored.
- *Prefix*: is added to the file names of the tiff series.
- *Create new folder for each series*: for each new measurement the results are stored in a new directory.
- *Use OME tiff extension*: Additional **meta data** information is stored within the tiffs.
- *Open series in series viewer automatically*: During the measurement the series viewer opens. You can browse backwards through your measurement, while it is still running.

### 7.3.3 Advanced Autosave Settings

By clicking the button “Advanced settings”, a new dialog appears. Here you can specify some more *autosave* settings:

**output files** You can select the format of the output files (standard or short).

*standard file names:*

PREFIX\_TIMESTAMP\_AcquisitionDevice\_CHANNELPOSITION\_DEVICE1\_DEVICE2\_DEVICE3.ome.tif  
(3DTimeColor\_14-26-00\_Imager\_C0\_xyz-Table Z0\_Time Time0\_FilterWheel F0.ome.tif)

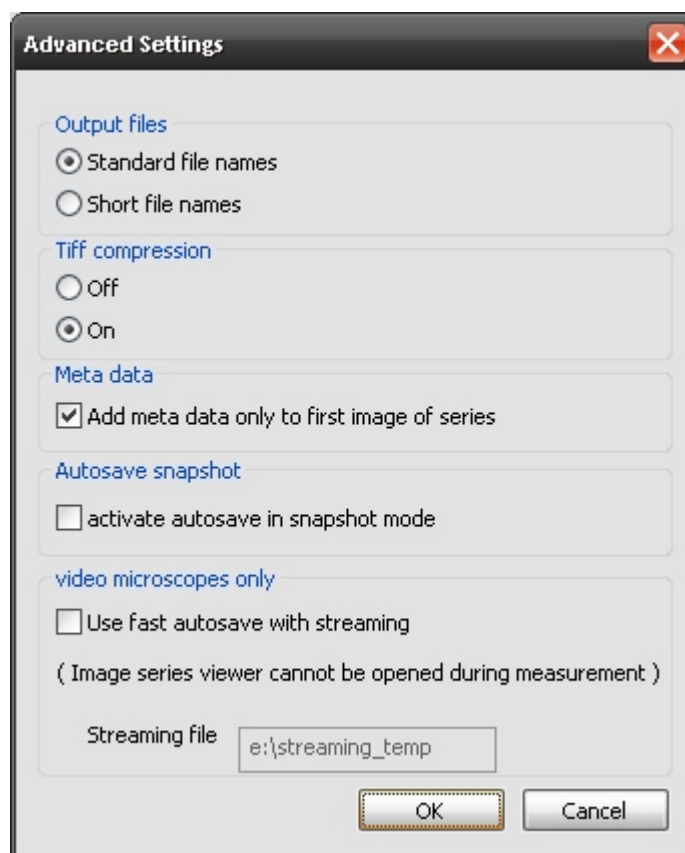


Figure 15: Advanced autosave settings.

short file names:

PREFIX\_CHANNELPOSITIONxx\_Txx\_Zxx.ome.tif  
(3DTimeColor\_C00\_Z0000\_T0000\_F0000.ome.tif)

**Tiff compression** Tiffs can be stored with or without data compression. A lossless data compression is used.

**Meta data** The OME meta data is added to the first file of the series only.

**Autosave snapshot** Autosave is active in snapshot mode, too: every snapshot measurement is stored automatically as a single tiff file.

**Streaming** The measurement data is temporarily buffed into a streaming file. After the measurements ends, the data of this file is converted into the tiff series. This increases the speed of the measurement.

After clicking *OK* the *autosave* function is activated.

If you want to activate *autosave* for every new document, check *keep as default* before clicking *OK*. Then *autosave* is also activated after restarting Inspector.

### 7.3.4 Using Autosave

After activating *autosave*, new buttons appear in the *Measurement Wizard*. The buttons are shown next to the list of devices you are using for your measurement. You can select for each device, if *autosave* should be used. If the *autosave* button is set to *On*, the data of this device is stored as a series of tiffs, it is not shown as data axis in your result stack any more.

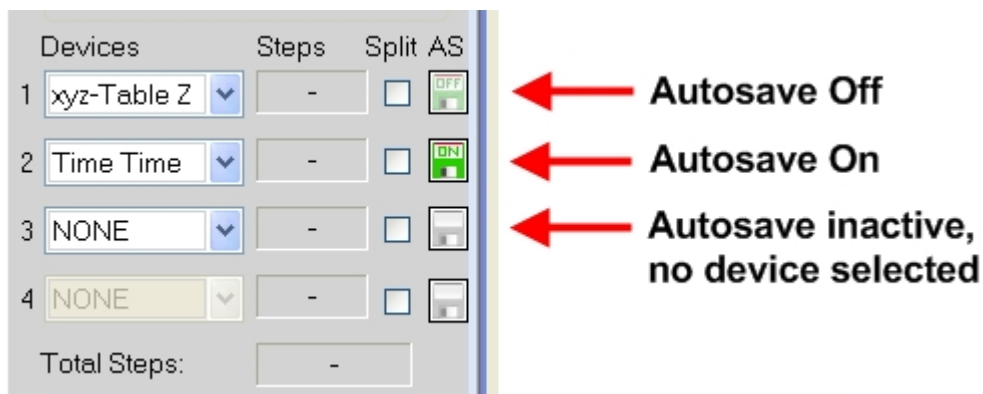


Figure 16: Autosave Buttons.

If you set *autosave* to *On* for one device, for all devices below this device *autosave* is also set to *On*.

If *autosave* is set to *Off*, the data of this device is shown in the result stack as data axis.

If you activate *autosave and split (and use OME)*, the data of this device is stored as different channels.

Tiff series created with *autosave* can be opened with the *Image Series Viewer* of Inspector. You can also import these tiffs with other software. If you use the *OME tiff extension* other software - that supports OME - can read all physical sizes and other meta data that is stored in these tiffs. Further information on OME see <http://www.openmicroscopy.org>.



### 7.3.5 autosave examples

With the settings shown in figure 14 (*Autosave Setting*) for each run of a measurement a new directory is created; it contains the current date, the specified prefix and a time stamp:

```
110321_3DTimeLapse_14-26-00,  
110321_3DTimeLapse_15-46-35, etc.
```

*Autosave* for 3D-TimeLapse:

This mode uses two devices: *XYZ-Table Z* and *Time Time*. Assumed for both devices *autosave* is set to *On* a series of tiff files is created (standard output format):

```
3D_Time_15-44-43_Imager_CO_XYZ-Table Z0_Time Time1.ome.tif,  
3D_Time_15-44-43_Imager_CO_XYZ-Table Z0_Time Time2.ome.tif, [...],  
3D_Time_15-44-43_Imager_CO_XYZ-Table Z100_Time Time100.ome.tif
```

short output format:

```
3D_Time_C00_Z0000_T0001.ome.tif,  
3D_Time_C00_Z0000_T0002.ome.tif, [...],  
3D_Time_C00_Z0100_T0100.ome.tif
```

if you use *autosave* for other devices, like *filterwheel* or *laser* the filenames include these information, too:

```
..._FilterWheel F0.ome.tif,  
..._FilterWheel F1.ome.tif, ...
```

## 7.4 Load and Save Data

### 7.4.1 Save Data

Acquired data can be saved or exported to other file formats. To save your work choose save or save as. ImSpector saves data files as ImSpector Data Files (\*.spv) and measurements as ImSpector Measurements (\*.msr). Thereby not only the content of a measurement or data window is saved. (\*.spv)- or (\*.msr)-files contain all windows linked with each other and the kind of linking (profile, copied ROI...). In addition the positions and the sizes of all windows are regarded.

### 7.4.2 Export to File

The content of data, measurement or graph windows can be exported to different file formats. To export data activate the desired window and choose export to file.

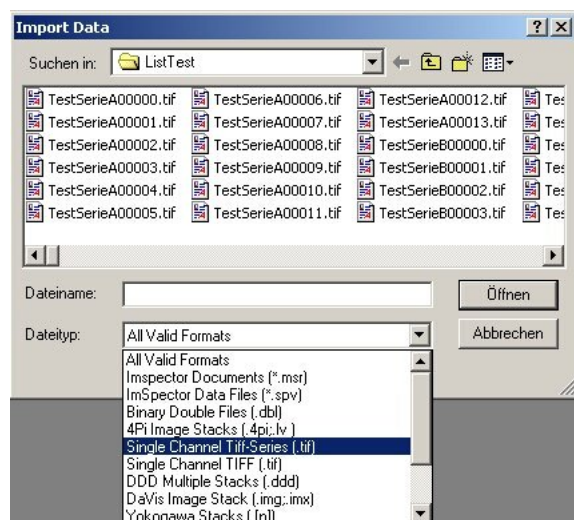


Figure 17: Import Data.

### 7.4.3 Loading or Importing Data

Beside the ImSpector data types several common file formats can be imported. To open or import files choose open and select the data and its format that should be loaded. With the standard File-Open dialog it is also possible to load single channel tif-series into inspector.

Chose "Single Channel Tiff-Series" as data type and select the last image of the series. Next, click on open and the series will be loaded into inspector. The images series will be displayed in a new stack window.

### 7.4.4 Export to AVI

Data stacks can be exported as AVI movies. This can be done by choosing Export to AVI from the file menu. A dialog will open where you can select path and name of the avi file. You can select the desired resolution of the video by activating one of the corresponding options. The option "Keep Resolution" will let you create avi video files that have exactly the same x and y dimensions that your data stack has. Depending on your data stack, this will create a huge video file that can possibly not be displayed fluently. But since it is totally uncompressed and unmodified, it is very well suited to be further processed by any other video editing tools. By selecting one of the other options, the result video can be reduced in its dimensions. This results in a smaller file that can easily be displayed in any video viewer, but that is not that practical for further processing.

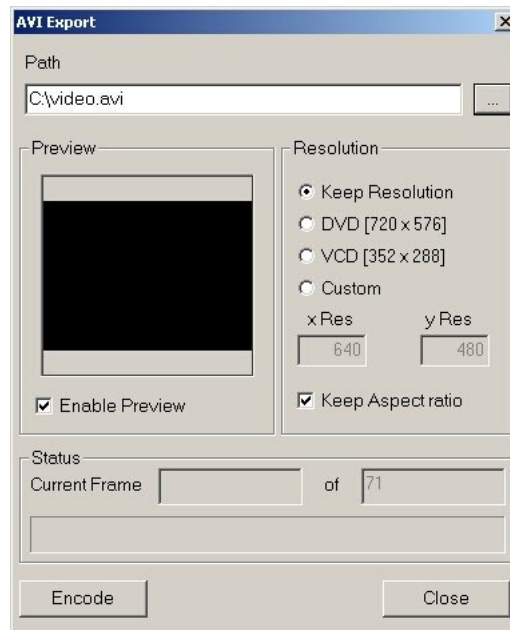


Figure 18: Export to Avi.

### 7.4.5 Export to MPEG

The MPEG export dialog looks exactly the same as the AVI export dialog in illustration 11. Exporting to mpeg reduces the size of the file by a great amount. But it also leads to a loss in quality. Thus the resulting video file is not suitable for post processing. Depending on the selected resolution of the result video, the file size can be very small. This makes a mpeg video very practical for presentation inside another documents, like Powerpoint documents or similar presentation programs.

## 7.5 Copy Data in New Windows

All functions for moving data from one window to another are located in the popup menu edit. The menu item copy is used to put the selected part or the whole data stack to the clipboard. Choosing paste inserts the content of the clipboard into the active window. Selecting purge clipboard deletes all clipboard contents. These commands work for measurement, data and profile windows. The content of a data window can be copied to a measurement window and vice versa but not to a graph window. The content of a graph window can only be copied to another graph window.

## 7.6 Export To File

*Inspector .msr documents* can be exported to different formats. Click on *Edit->Export To File* to open the export dialog. In this dialog the output name must be specified and the output format can be selected. The following formats can be used:

- Tiff / OME Tiff (standard Tiff or Tiff with OME meta data extension)
- RGB Tiff (Colored tiff as it is seen in the stack window - without OME option)
- Binary (image data without any other file information)
- Ascii (image data displayed as text - should be used when exporting profiles)

## 7.7 Tiff Export Dialog

When selectig *Tiff / OME Tiff* in the [Export To File](#) dialog a new dialog opens. In this dialog several settings can be specified:

1. Colour depth: 8 or 16 bit
2. Scaling data to avoid truncation errors.
3. OME meta data: Additional meta data is stored into the tiff header. When exporting multichannel tiff, OME must be activated (See: [Exporting to Ome Tiff](#)) When multichannel or tiled imaging is used each channel and each table position is stored into a single tiff file. The 3rd and 4th dimension of the data stack is stored inside these tiffs. The Ome meta data describes how to interpret the data.
4. Split 3rd/4th axis to single files: if no OME is used, the third and/or fourth data axis can splitted into single tiff files. It can be specified if the actual axis value (e.g. time position or stack position of a 3D stack) is directly stored in the file names.

## 7.8 Exporting to Ome Tiff

*Inspector .msr documents* can be exported to tiffs with the *Open Microscopy Eviroment File Format* extension. OME tiffs support data up to 5 dimensions and store several meta data information. For creating OME tiffs you can use the normal export dialog, the batch conversion [7.10] or the autosave function [7.3] of *inspector*. Batch conversion and export can be found in the menu "Edit" (*further information about OME see: <http://www.openmicroscopy.org>*).

## 7.9 Measurement Information

You can display and change important measurement information and other meta data of your measurement. Open the measurement information window by clicking the "i" at the right side of the stack window.

*Inspector documents* can handle different meta data information:

- Device information, like scanner, laser or camera settings
- Measurement information, like instrument mode or creation date
- Stack information, like pixel number or physical sizes
- Comment or description of the measurement
- User specific meta data, like name or institution

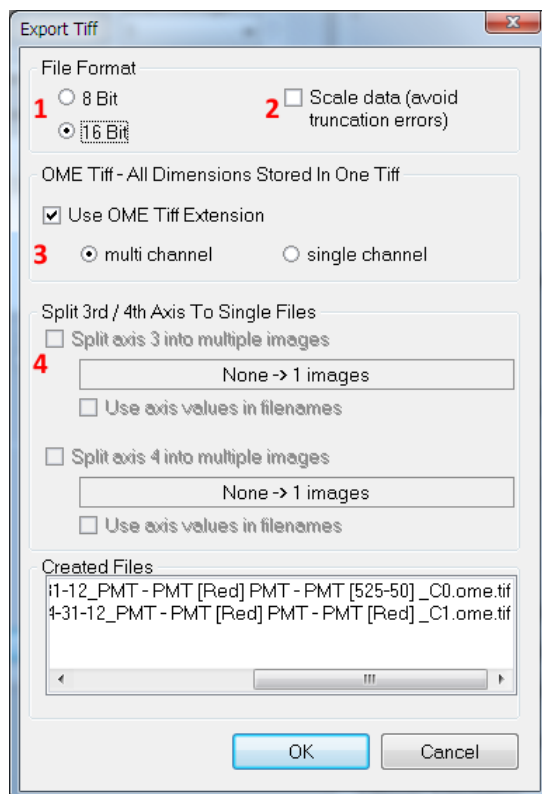
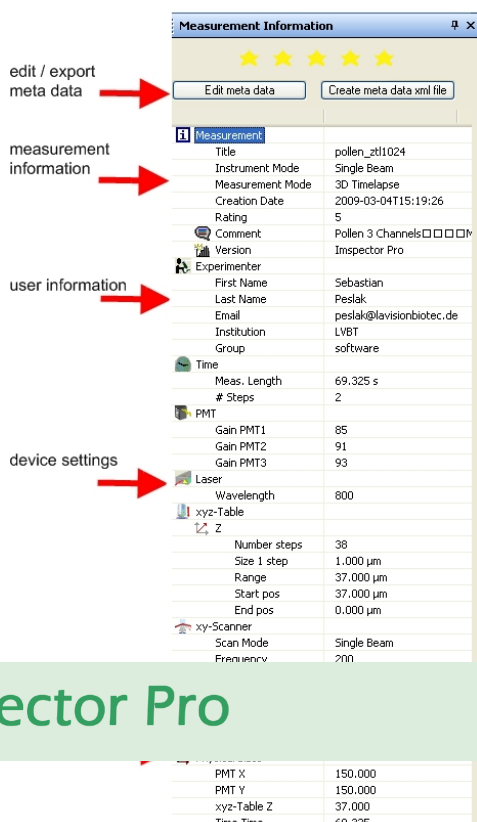
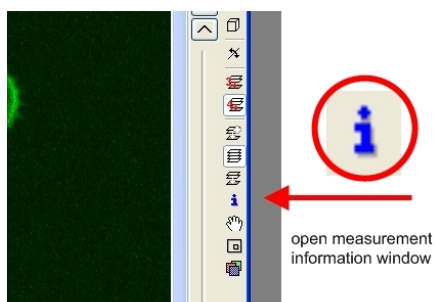


Figure 19: Export Settings.



Some of these values can be changed by the user. By clicking "edit meta data" a new dialog appears (see figure 21):

In this dialog you can change the experimenter settings (name, email, etc.). You can also select a default user. By clicking "Set default user" the "user settings dialog" opens where you can select or add a default user. These user settings are stored to all documents and OME tiffs created by imspecter.

You can change some measurement information or add a measurement description. By clicking "copy tree values" all information of the measurement window are stored as description of the measurement (If you export your stacks as OME tiff, this description is shown in other software, that supports OME e.g. ImageJ or Imaris).

You can also import meta data from another stack. By clicking the "..." button you can

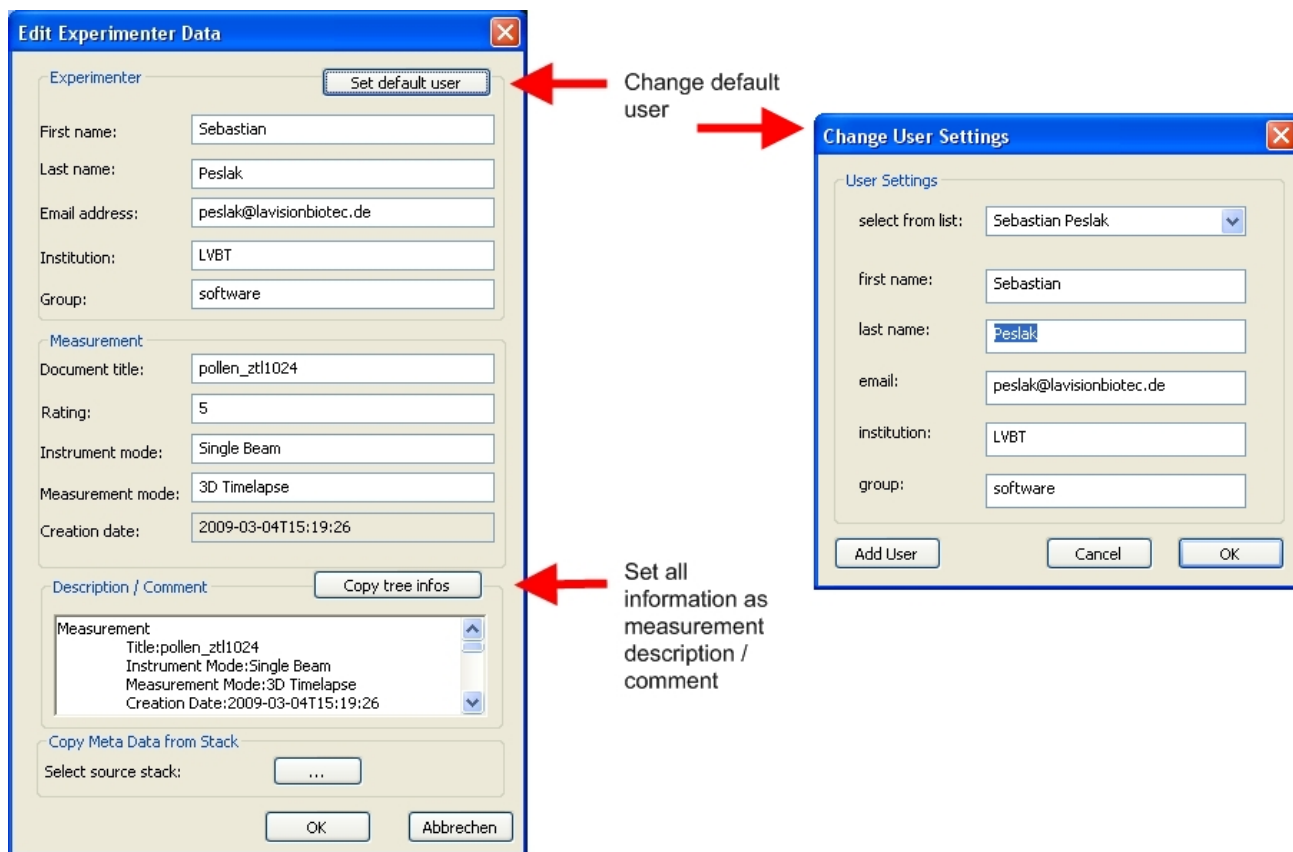


Figure 21: Edit meta data.

select the stack the data should be copied from.

You can export all meta data by clicking "create meta data XML file" at the top right of the measurement information window.

The exported file includes all information that is stored into the OME tiffs created by inspector. It can be read with a XML reader, text editor or web browser.

## 7.10 Batch Conversion

To export any number of Inspector '.msr' documents to the Ome Tiff file format you can use the *batch conversion*. This can be found in the menu "Edit" "batch conversion".

You can browse for files or drag and drop files directly from your windows system. The files are exported to 16 bit multi channel Ome tiffs. For every document a new folder with the name of the document will be created inside of the suggested or user specified output directory.

## 7.11 Export All

You can export the stacks of a document to tiff all at once by using the *Export All* function. It can be started in the menu "Edit" or on its tool bar icon.

## 7.12 Display Shadow Cursors

If a measurement has several channels you can display the mouse cursor in all channels simultaneously.

You can activate the 'shadow cursors' by clicking the mouse wheel while the original cursor moves over one of the channels. A cursor that follows the original mouse cursor position appears in every channel. There is also shown the current count value of the stack at the bottom of the cursors. After clicking the mouse wheel again, the cursors are fixed and don't follow the original cursor any more.

After clicking the mouse wheel a third time the shadow cursors disappear.

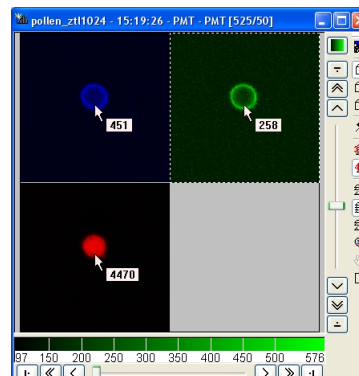


Figure 22: Shadow Cursors.

## 8 Handling Windows

### 8.1 Editing a Windows Size and Position

A window is activated by a mouse left click on the headline. To move it to another position on the workspace do a left-click on the headline, keep the mouse button pressed and drag the window to the desired position. The size of the window frame can be adjusted by left-clicking on the boarder of it, keeping the mouse button pressed and dragging the mouse until the desired size is reached.

### 8.2 Organisation of Several Windows

The organisation of several data windows on the workspace of ImSpector can be done the popup menu window. The function cascade aligns all data windows starting in the upper left corner. Cascade and fit arranges windows and assigns the same size to all of them. The function tile puts all window in parallel.

## 9 Description of Devices

### 9.1 Scanner

#### 9.1.1 Introduction

This section introduces the scanner dialog window in ImSpector Pro software package. First, it covers different scan modes and second, it describes the use of EOMs and Power Modulation Package.

#### 9.1.2 XY-Scanner Dialog Window

##### Scan Modes

The XY-Scanner Dialog Window includes several tabs for specific scanner operations. Figure 23 shows the XY-Scanner dialog window.

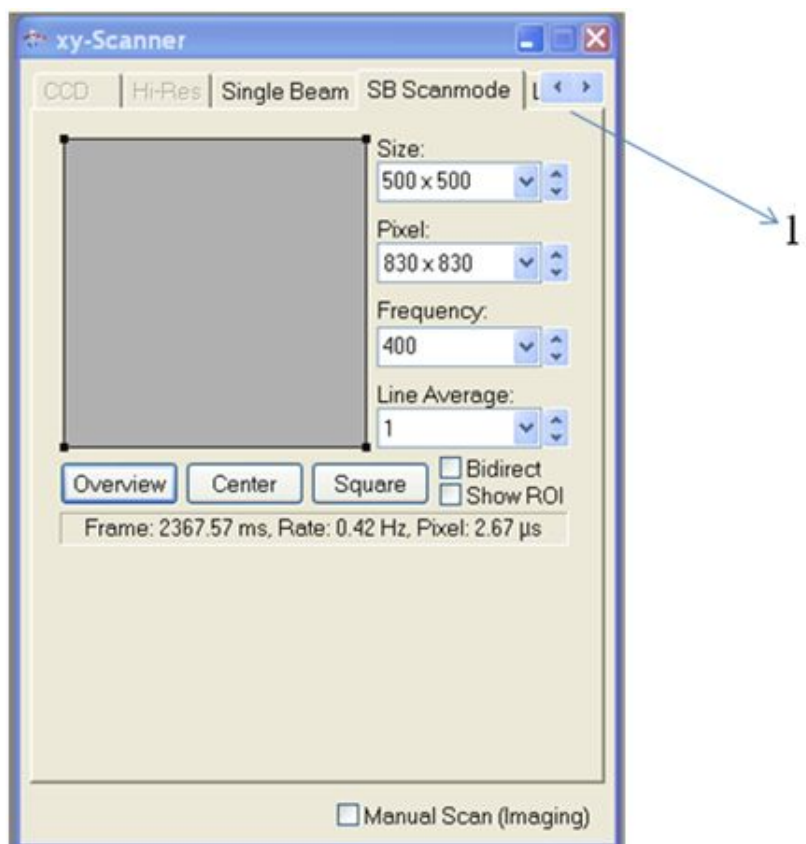


Figure 23: The XY-Scanner dialog window

Each tab in the XY-Scanner dialog window activates a specific scan mode.

- CCD  
Optional, if the instrument is equipped with a CCD camera and/or the beam splitter.
- Hi-Res  
Optional package for the multi-beam version of the TriM Scope.
- Single Beam  
Provides predefined FOVs, pixel resolutions and scan velocities

- SB Scanmode  
Provides user definable FOVs, pixel resolution, scan velocities and zoom functionality
- Line Scan  
Provides various modes for fast line imaging

## Single Beam Mode

Figure 24 shows the Single Beam dialog window. This mode provides predefined Scanfield Dimensions [1], Pixel Resolutions [2], Scan Frequencies [3] and # Multi Lines [4].

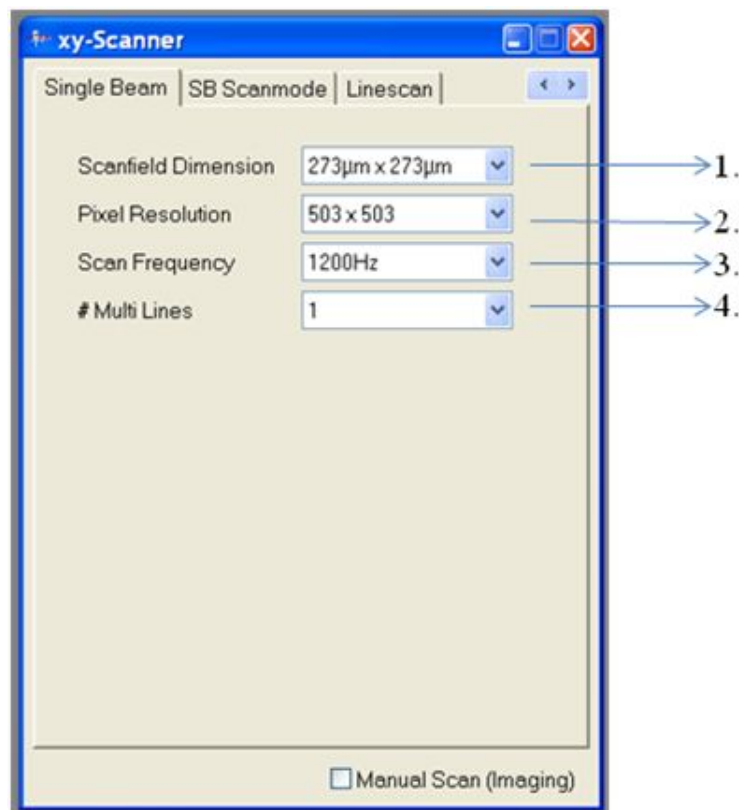


Figure 24: Single Beam scanner dialog

1. Scanfield Dimension: are fixed and predefined values. The dimensions in  $\mu\text{m}$  are determined by the magnification of the objective lens and the scanning angle of the galvo scanner mirrors. When choosing another objective lens the objective lens setting has to be changed in the hardware settings (toolbar: hardware, configure, xy-scanner settings, magnification)
2. Pixel Resolution: # of pixel in x and y direction.
3. Scan Frequency: this is the line frequency (the fast scanning axis).
4. # Multi Lines: the # indicates the repetitions of each fast line scan. The signal will be added up for each Pixel.  

$$I_{\#} = I_1 + I_2 + \dots + I_n; \text{ for } \# = 1 \text{ to } n; I_i \text{ is the intensity in line scan } i$$

Figure 24 shows a single beam scan definition that delivers  $273 \times 273 \mu\text{m}^2$  scanfield, 1200 Hz line frequency [1200 lines/second in one direction, fly back takes 15 scanner tics, each



scanner tic is  $23\mu\text{s}$ ] and  $503 \times 503$  pixel resolution. This will result in a frame time of:

$$503 [\text{\# of lines}] \times ((1/1200\text{Hz}) + (15 \times 23\mu\text{s})) = 594\text{ms}$$

Higher line frequency results in shorter frame times.

When choosing 3 for instance in the # Multi Lines box the frame time will increase by a factor of 3  $\rightarrow 3 \times 503 [\text{\# of lines}] \times ((1/1200\text{Hz}) + (15 \times 23\mu\text{s})) = 1.78\text{s}$ .

## SB Scan Mode

Figure 25 shows the SB scan mode dialog. This dialog allows the user to define scan Size [1], Pixel resolution [2], fast line Frequency [3] and Line Averaging [4].

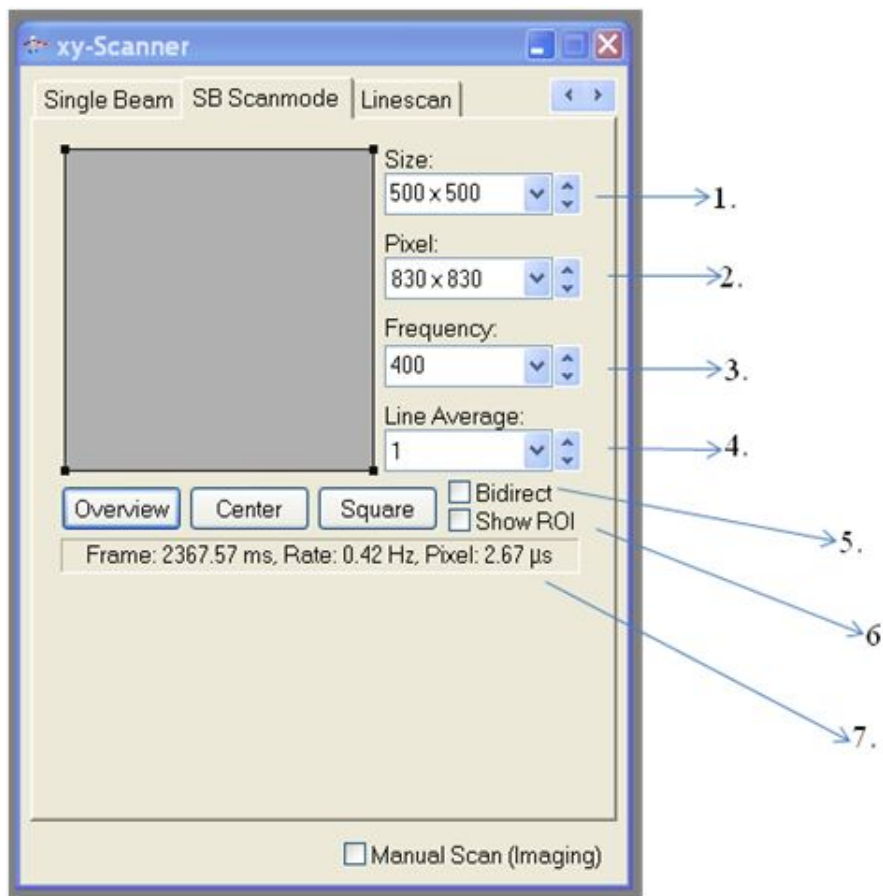


Figure 25: SB scan mode

1. Size: scan field in  $[\mu\text{m}]$ : e.g.  $500\mu\text{m} \times 500\mu\text{m}$  The dimensions in  $[\mu\text{m}]$  are determined by the magnification of the objective lens and the scanning angle of the galvo scanner mirrors. When choosing another objective lens the objective lens setting has to be changed in the hardware settings (toolbar: hardware, configure, xy-scanner settings, magnification)
2. Pixel: pixel resolution in [1]: e.g.  $830 \times 830$  pixel Pressing the "shift" button on the keyboard and clicking the increment arrow in the dialog box increase the number of pixel.

3. Frequency: fast line frequency in [Hz]: e.g. 400Hz
4. Line Average: the # indicates the repetitions of each fast line scan. The signal will be added up for each Pixel xy.  
 $I_{\#} = I_1 + I_2 + \dots + I_n$ ; for # = 1 to n;  $I_i$  is the intensity in line scan i
5. Bidirectional scanning: Activating the check box will allow bidirectional scanning mode that will be explained in the next chapter. This option requests the optional Targeted Path Scanning module.
6. Show ROI: Activating the check box will show the selected ROI in the document.
7. The software shows final frame parameters in this box: frame time [ms], frame rate [Hz], pixel dwell time [ $\mu$ s] will be displayed. The real frame rate might drop down when the images are displayed during the measurement.
  - Clicking the tab Overview gives the largest scan field that is provided by the galvo scanner.
  - Clicking the tab Center the ROI will be centered in the max. FOV.
  - Clicking the tab Square will change a rectangular ROI to a squared one.

Marking the ROI in the image document [1 in fig. 26] defines the ROI for the next scan.

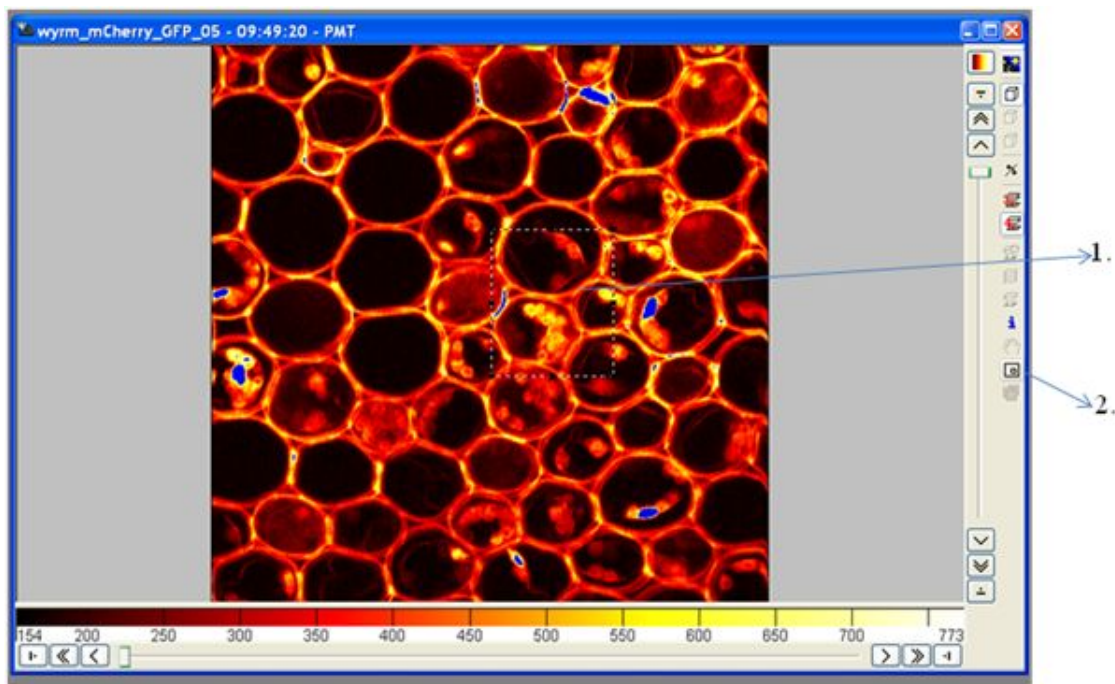


Figure 26: Choosing a ROI for the scanner dialog from the document

Clicking the tab [2 in fig. 26] "select ROI" will generate the ROI in  $\mu$ m in the scanner dialog. The ROI frame can be moved with the left mouse bottom.

### Scanning Parameters

In single beam scan mode the system allows a user definable set of scanning parameters. Nevertheless the scanning parameters are limited by the mechanical and electrical microscope capabilities. The following tables show maximum values.

## Scan Rates

Pixel resolution: 250 x 250

Max. # of pixel: 14500 x 14500 in unidirectional scan mode  
1000 x 1000 in bidirectional scan mode and/or when position feedback is activated (requests Targeted Path Scanning Module)

FOV/ $\mu\text{m}^2$ (20x Objective lens)	Unidirectional			Bidirectional		
	Max scanner frequency / Hz	Min frame time / ms	Max frame rate / Hz	Max scanner frequency / Hz	Min frame time / ms	Max frame rate / Hz
500 x 500	800	397	2.5	2000	129	7.7
400 x 400	1200	294	3.4	2500	106	9.3
300 x 300	1600	241	4.1	3000	89	11.1
200 x 200	2000	207	4.8	3500	78	12.8
100 x 100	2400	190	5.2	4000	66	15.0
50 x 50	2600	178	5.6	4000	66	15.0

## Line Scan

Before using the line scan dialog an image (e.g. 500 x 500  $\mu\text{m}^2$ ) image has to be acquired. After activating the line scan dialog (see fig. 27) individual scan lines can be defined by using the left mouse button.

Fig. 28 shows different line scan settings. The functionality of the tabs in the dialog window are:

- **Timelapse:** this entry defines the repetitions of all predefined lines and points, e. g. 100 cycles of the displayed four lines.
- **Line Averaging:** When choosing values  $>1$  the cycle of predefined points and lines will be repeated and the intensity for each pixel will added up before jumping to the next time step in the timelapse series.
- **Loop Freq:** This is the frequency of one individual scan cycle for the defined number of lines and points. The shortest scanner cycle time is determined by the infinite scanner tic of 23  $\mu\text{s}$ . Therefore the scanner cycle time is always a multiple of 23  $\mu\text{s}$ . Each jump between two events (line or point) needs 6 scanner tics or 138  $\mu\text{s}$ .
- **Pixel Time:** Pixel dwell time in  $\mu\text{s}$ . The pixel dwell time will not change the loop frequency, which is constant. The higher the pixel dwell time the fewer pixels will be scanned.
- **Line speed:** This is the constant speed in [ $\mu\text{m}/\mu\text{s}$ ] for one line. When changing the line speed the speed for each line will be constant and all other parameters like Loop Frequency, number of pixel and the time for one line will be changed.
- **Pixel shift:** This value will be calculated by the software.
- **Auto Zoom:** Activating the check box Auto Zoom will zoom into the loop of lines (and points).
- **Vert:** The scan line will be rotated in vertical direction.
- **Horz:** The scan line will be rotated in horizontal direction.

- Up: The scan line will go one position up in the list of scanned lines (and points).
- Down: The scan line will go one position down in the list of scanned lines (and points).
- Flip: The scan line will be rotated by 180°.
- Del: The scan line will be deleted.
- Eval: see next chapter.
- Scan Path: see next chapter.

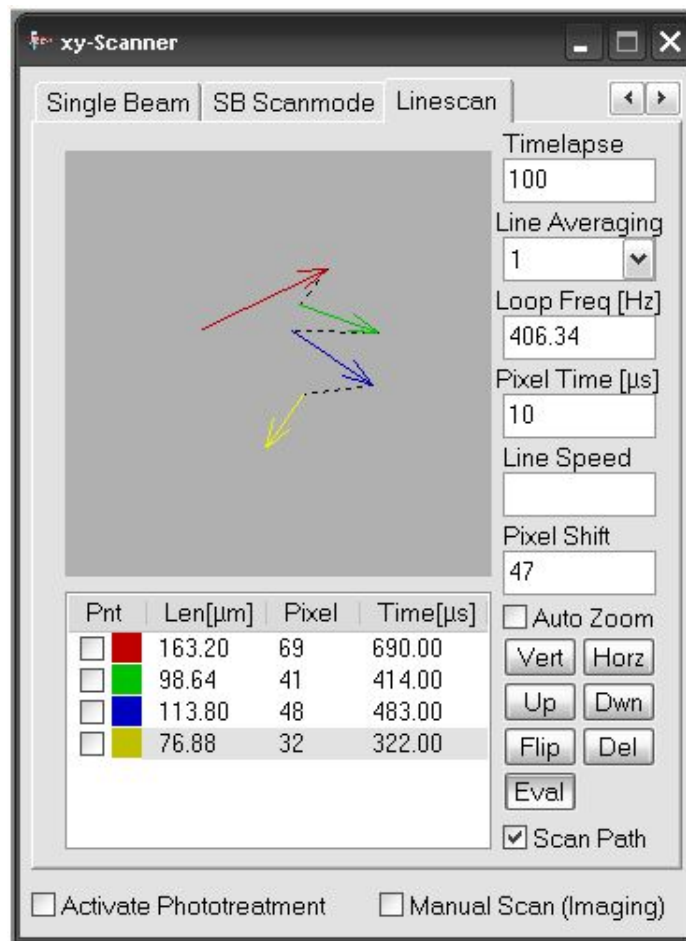


Figure 27: Line scanning

Activating the check boxes Pnt will reduce the line to a point. The point will be the starting point of the line. Len [ $\mu m$ ] displays the length of the line in [ $\mu m$ ]. Pixel displays the number of pixel and Time defines the time for these specific lines in [ $\mu s$ ]. When double clicking on the line one gets access to the following Line Position dialog window (fig.: 28):

This dialog allows the definition in physical units [ $\mu m$ ]. Starting and end point of each line can be changed. Activating the check box Point will generate a point.

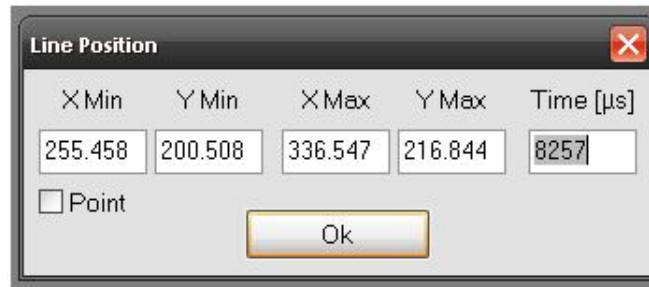


Figure 28: Line Position dialog window

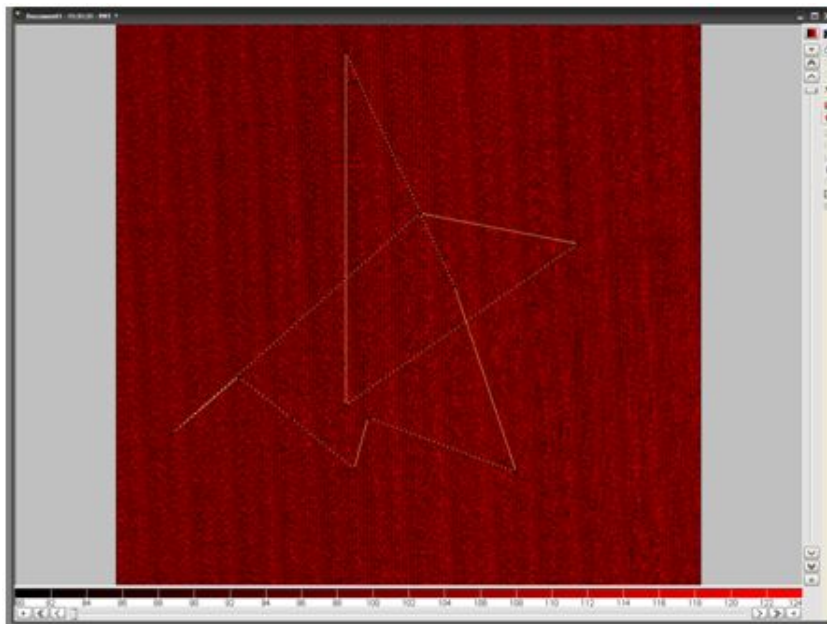


Figure 29: Different lines

## Defining and modifying Lines or Points

Lines and points are defined and modified in the document window (see fig. 29). Therefore an image has to be displayed in a document window and the mouse cursor has to be moved into the document window. Lines are always displayed as solid lines, points as black point and jumps between points or lines as dashed lines.

### Creating the first Line or Point

Press the left mouse button and draw a line. If the line should be converted to a point the Pnt check box has to be activated in the xy-Scanner dialog window.

### Creating an additional Line or Point

Move the mouse cursor on the dashed line until the cursor changes to a cross, press the shift button of the keyboard and draw a new line. If the shift button is not pressed the new line starts from the dashed line. If the new line should be converted to a point one the Pnt check box has to be activated in the xy-Scanner dialog window.

### Moving a Line

Move the mouse cursor on a solid line until the cursor changes to two crossed arrows, press the left mouse button and move the line.

### Stretching a Line

Move the mouse arrow on the end of a solid line until the cursor changes to a double arrow and stretch the line.

## Bidirectional Scanning

The bidirectional scanning requests the optional **Targeted Path Scanning Module**.

### Activation Bidirectional Scanning

By activating the Bidirect check box in the xy-Scanner dialog window the bidirectional scanning mode is turned on. As the bidirectional scanning requests the position feedback signal one has to activate it by a double click in the **PMT** dialog window next to samples/pix (see Fig. 30).

### Adjusting Bidirectional Scanning

For getting smooth images one has to adjust the phase shift between back and forth scan direction. The phase shift depends on the scanner frequency. Therefore the phase adjustment has to be checked when changing the scan frequency. The phase shift has to be modified by the slider Samples/pix in the PMT dialog window.

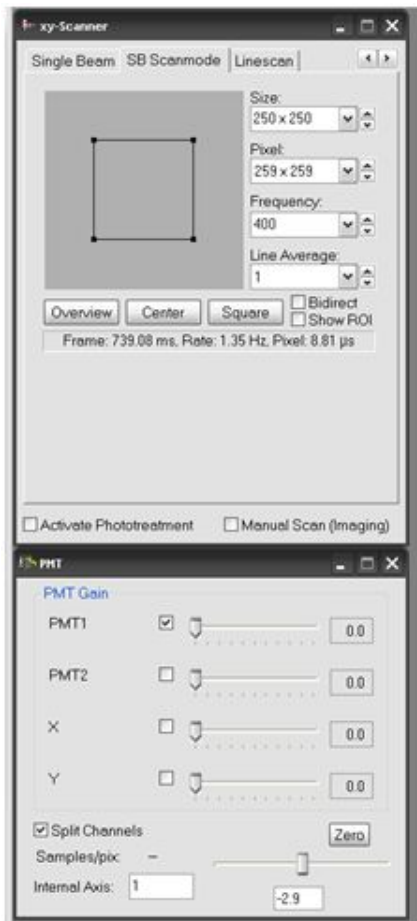


Figure 30: Activation of the position feedback from the scanner

## Increasing the Scanning Speed in Bidirectional Scanning Mode by activating an internal Axis

The internal axis method is the fastest acquisition method for a time lapse experiment in bidirectional scanning mode. Recorded data are stored in the computer RAM and will not be displayed until the time lapse is finished. The number of non displayed images will be defined in the Internal Axis box in the PMT dialog window.

## Using the position feedback in the Line Scan Mode

The use of the position feedback signal requests the optional **Targeted Path Scanning Module**.

The Targeted Patch Scanning module gives the real scanner position during a scan cycle. Therefore the defined scan cycle and the real scan cycle can be compared to check if the most interested areas have been scanned or treated.

## Defining a Scan Cycle

See the previous chapter.

## Comparing Defined and Real Scan Cycle without Laser

First, activate the Scan Path check box in the xy-Scanner dialog window. While activating the Scan Path check box the scanner will run one cycle without opening the laser shutter. The real and the defined scan path will be displayed in new document window, which can be overlaid with the image window (see fig. 31). Now, one could start to optimize the scan path definition to scan/treat the interesting position within the image frame.

## Measurement and displaying Data

Pressing the Start button in the Measurement Wizard window will start the scan cycle. After starting the measurement two windows are displayed (see figure 32). The first document window shows the image and the defined scan cycle while the second document window displays the intensity during the scan cycle. The x-axis (3. in figure 32) is the time axis during one cycle and the y-axis (1. in figure 32) is the time axis of time lapse experiment.

When opening an ROI in the second document window (that is the one that shows the intensity during the scan cycle) the real scanner position will be showed in the first document window that shows the image frame.

### 9.1.3 Single Beam Scanning with EOM

While using the EOM the optional **Power Modulation Package** is required. An EOM (Electro Optical Modulator) allows the rapid change of the laser power (roughly  $<1\mu s$ ). Typical EOM applications are:



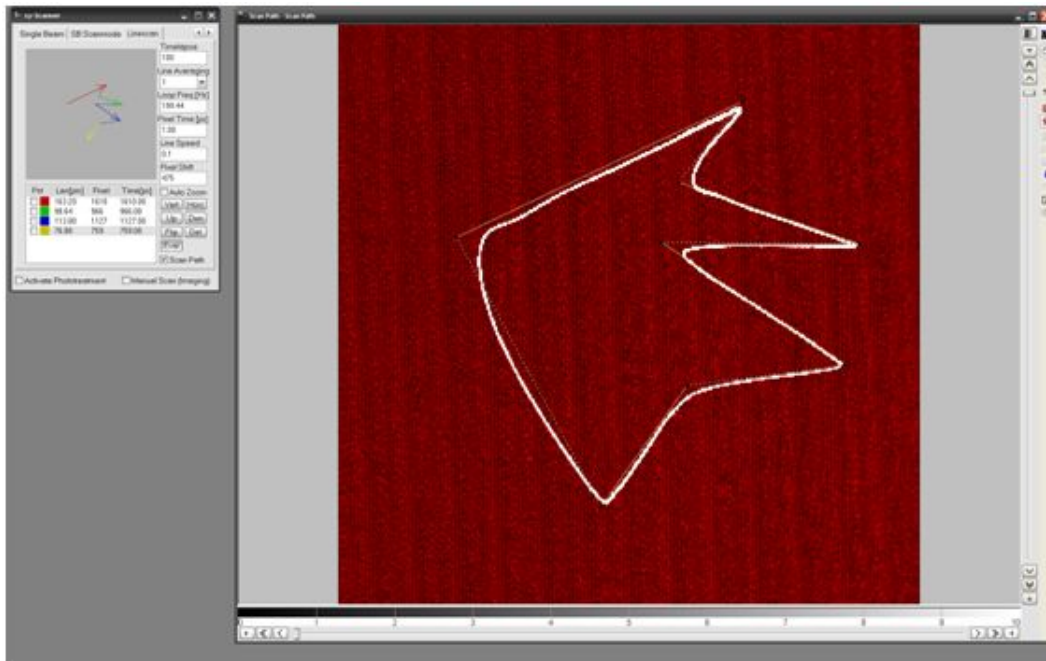


Figure 31: Calculated and predefined scan path

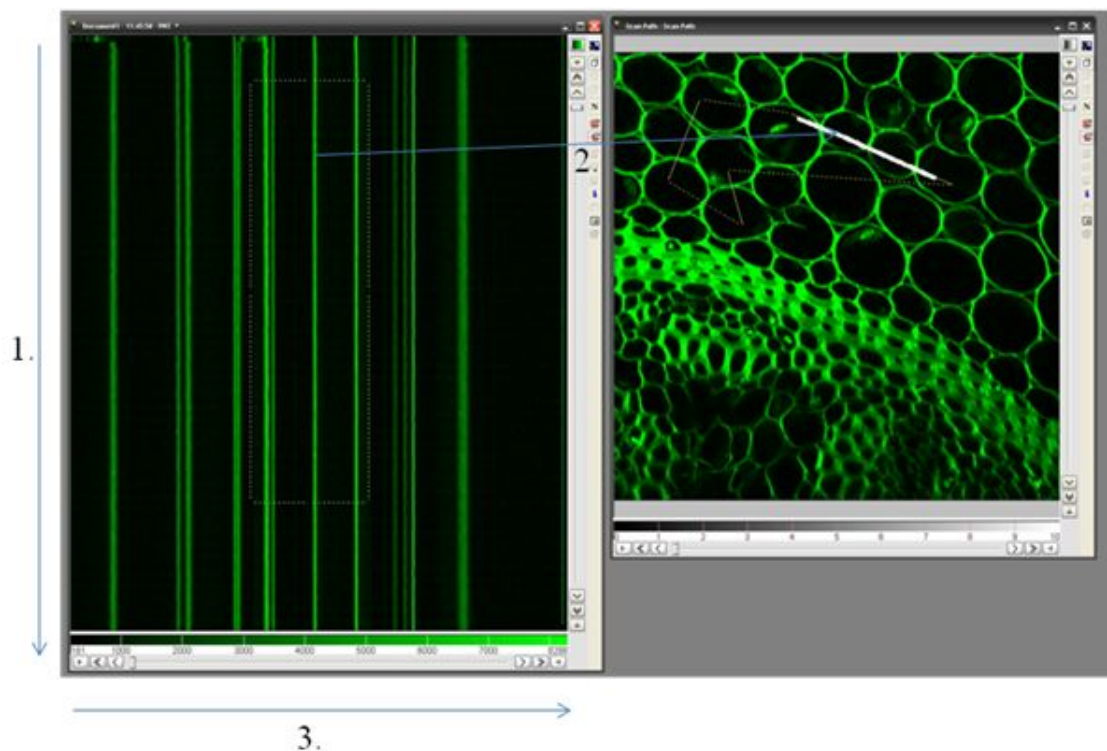


Figure 32: Intensity during a scan cycle (left window) and the pre-acquisition image (right window)

- Photo bleaching
- Photo activation
- Photo stimulation
- Uncaging
- ...

#### 9.1.4 The EOM Dialog Window

Fig. 33 shows the EOM dialog window, which provides the following functionality:

- Select From Image: The selected ROI in the image document window will be selected for the treatment. The selected ROI could be a rectangle, circle, polygon, or a free form [2. in figure 33].
- ROI: displays the number of pixel of the chosen ROI [1. in figure 33].
- Start Treatment:
  - a. Activating the "after # Imaging" check box will start the treatment procedure.
  - b. The treatment will start after the chosen # of frames and
  - c. will be continued for the mentioned # of frames.
- The Manual Start button will start the measurement without data acquisition.
- EOM:
  - a. The left pull down menu allows the following settings:
    - a.i. Always on: in Always On mode the laser power can be set by the Imaging slider and will be constant all the time.
    - a.ii. Auto: in Auto mode the right pull down menu and the treatment mode will be activated.
    - a.iii. Off: in Off mode the laser will be blocked all the time.
  - b. The right pull down menu is only active if the left pull down menu is in Auto mode and allows the following settings:
    - b.i. Fullfield: in Fullfield mode the full field will be imaged with the set laser imaging laser power. For scanner fly back and treatment the laser power will be set to zero.
    - b.ii. ROI: ROI mode is like Fullfield mode but limited to the ROI that is chosen in the EOM dialog window.
    - b.iii. ROI Treat: In ROI Treat mode the full FOV will be imaged with the set laser power, during the scanner fly back period the laser power is set to zero. The treatment is limited to the ROI and the laser power will be set to the chosen value.
  - c. Imaging Slider: sets the laser Power for the imaging. [4. in figure 33]
  - d. Treatment Slider: sets the laser Power for treatment. [5. in figure 33]

**In the EOM treatment mode the position feedback signal has to be turned off!**

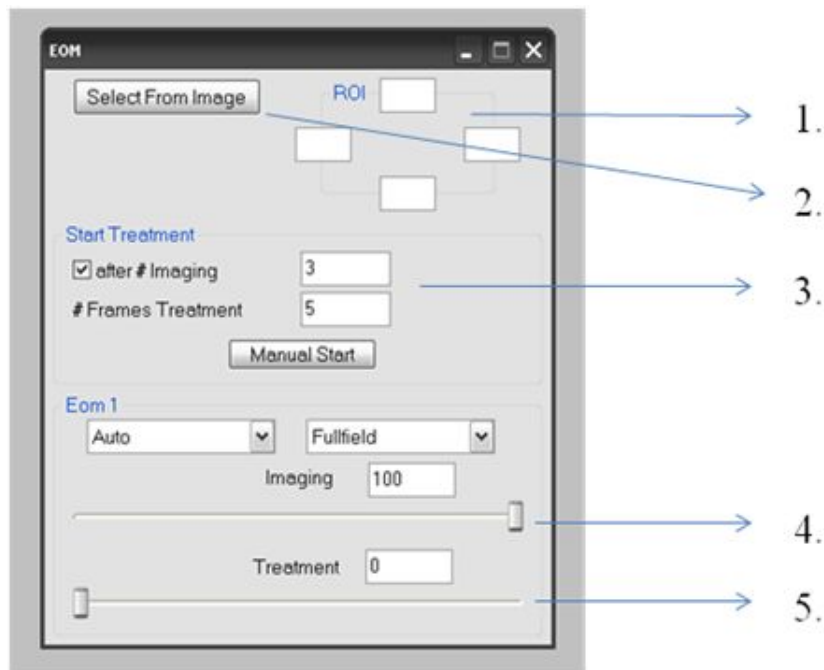


Figure 33: EOM dialog window

### Line Scanning Fly back Blanking

During fly back blanking in line scanning the laser power will be turned off by the EOM when the scanner flies back. The fly back blanking procedure is as follows:

- Acquire an image with activated feedback
- Choose the loop of lines and points in the xy-Scanner dialog window
- Evaluate and activate the scan
- Overlay both images
- Choose Auto and ROI in the EOM scanner dialog
- Start the measurement in the Measurement Wizard window

### Scanner Treatment

The Phototreatment will be activated in the xy-Scanner dialog window (see fig. 34). In Phototreatment mode the scanner switches rapidly between different scan fields. The laser power could be set individually fore each scan field (imaging/treatment). The meanings of the controls in fig. 34 are as follows:

1. Activate Phototreatment: activating the check box will activate the Phototreatment mode
2. At Position: If the checkbox is activated the Phototreatment will be active at the actual position in multidimensional stacks.
3. After # Images: If the checkbox At Position is not activated the Phototreatment will start after the mentioned number of images.

4. # Treatments: The number of treated ROIs will be set in this box.
5. This list shows all parameters of the treated ROIs. Up to six ROIs can be defined.
6. Show Scanmode: Clicking the toggle switch will show treatment area or imaging FOV.
7. Zero Gain: If the checkbox is activated the PMT gain will be set to zero automatically during the treatment.

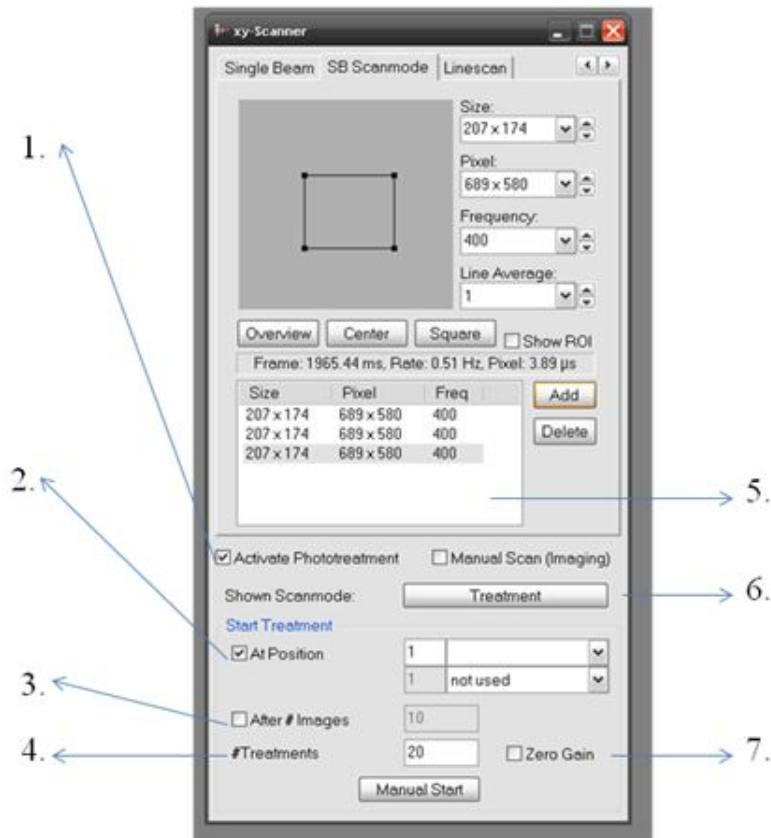


Figure 34: Scanner Treatment

## 9.2 Cloud Scanner

The Cloud Scanner device is able to split a single beam into multiple. With that a special kind of excitation of the sample is possible.

The device supports two scan modes. The default *VoxelFill* scan and a so-called *Object Scan* mode.

### 9.2.1 Life Dialog

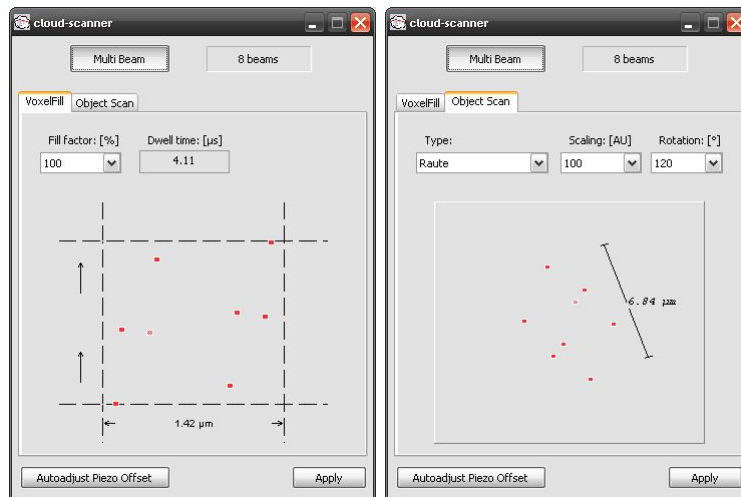


Figure 35: Cloud Scanner Life Dialog

The dialog has a button to enable or disable its activation for a measurement. Next to it, there is an information box showing how many beams are generated.

The *VoxelFill* tab presents the default scan mode. The only option is to set the fill factor for the pixel to place the beams.

The button *Autoadjust Piezo Offset* recalibrates the beam alignment of a piezo to the origin beam.

The other button *Apply* sets the hardware for a measurement. This is to minimize the initialization time of the measurement.

The *Object Scan* tab enables, in contrast to the scanner pixel of the default mode, a free scan. There can be set user-defined types. *These should be configured with the LaVision BioTec service.* A type-specific scale and rotation may be chosen.

## 9.2.2 Hardware Dialog



Figure 36: Cloud Scanner Hardware Dialog

The VoxelFill device connects three hardware parts. The controller and motor via serial lines and optionally an internal camera via usb. The latter is not required for the device to function properly and can be ignored.

The *Vector offset error*, *Piezo zero pass wait time* and *Calibration* button are for the *LaVision BioTec* service.

### 9.3 Horizontal Two Photon microscopy

The Horizontal Two Photon device, abbr. H2P, performs tomographic imaging of a sample.

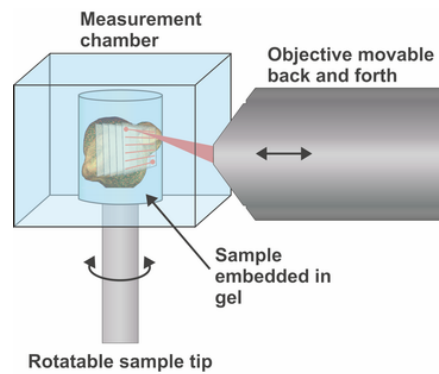


Figure 37: System Schema

#### 9.3.1 Life Dialog

The dialog is split into the overview of the current system status and the tabs below controlling the position, zero offset and ranges for the sample to be imaged. It is possible to use the *z adaptation* for the radius axis.

#### 9.3.2 SLOT mode

The rotation and radius axis ranges can be set freely with the software.

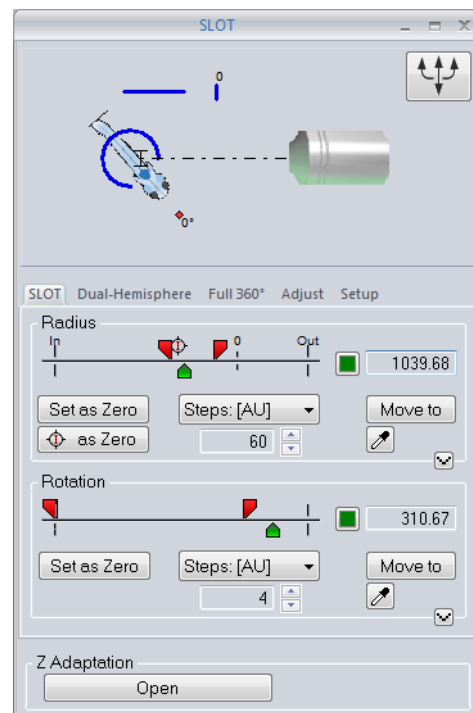


Figure 38: Life Dialog - SLOT mode

### 9.3.3 Dual-Hemisphere mode

The rotation axis range is fixed at 180° and two steps.

The radius axis can be set freely.

Note: For this mode *only* the radius axis must be chosen as H2P measurement axis! Axes from other devices can still be used.

Images of both hemispheres are placed into one stack. The images of the first hemisphere are placed top to center into the stack. The images of the second hemisphere are placed bottom to center into the stack. The images of the second hemisphere are not post-processed for performance reasons. The user may choose a tool to flip them after the measurement.

If this is not wanted, the user should use the SLOT mode [9.3.1].

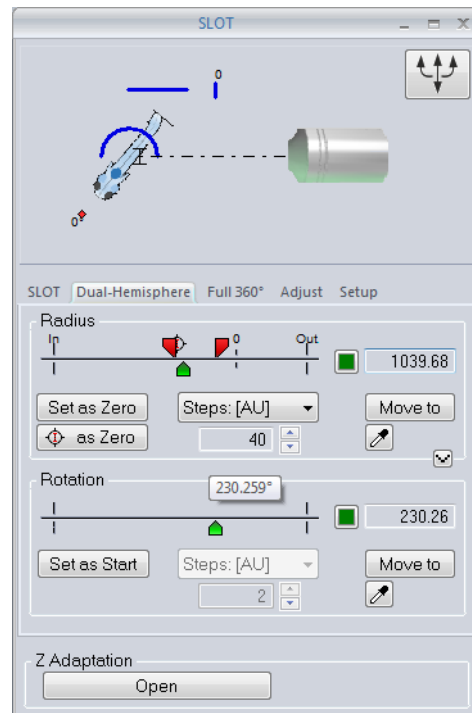


Figure 39: Life Dialog - Dual-Hemisphere mode

### 9.3.4 Full 360° mode

The rotation axis range is fixed at 360°. Its steps can be set freely.

The radius axis can be set freely.

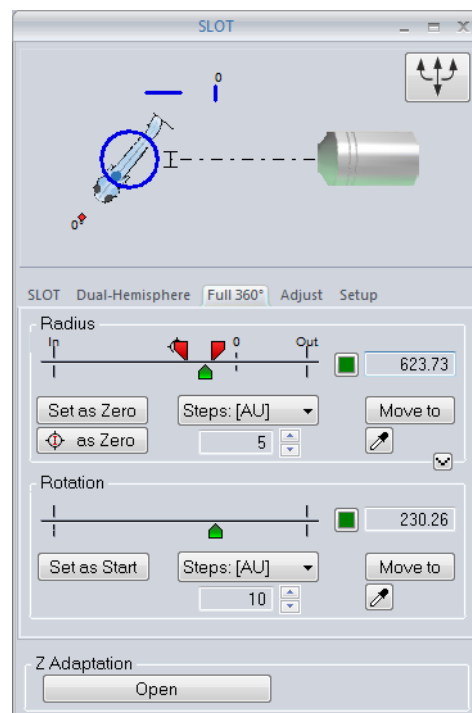


Figure 40: Life Dialog - Full 360° mode



### 9.3.5 Advanced Settings pull-down menu

Axis parameters are to be set manually.

The button *Take* sets the parameter to the current position, which can be seen at the green thumb of the slider.

The button *Move to* places the current position to the parameters value.

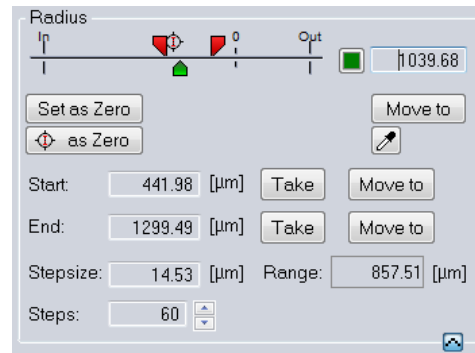


Figure 41: Life Dialog - Advanced Settings pull-down

### 9.3.6 Adjust

This tab is to tilt the sample, so that its rotation axis has proper horizontal and vertical directions. Therefore, four rotation positions are predefined, where the sample can be tilted.

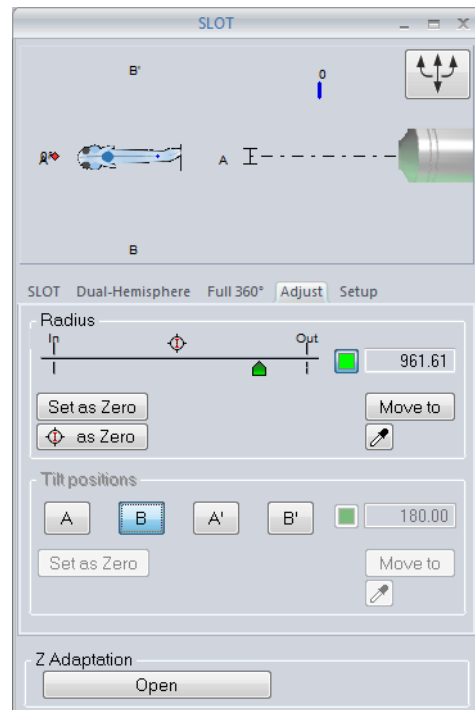


Figure 42: Life Dialog - Adjust

### 9.3.7 Setup

This tab consists of two groupboxes to reflect system parameters in the *Adjustment* groupbox and graphic settings for the overview in the *Overview setup* groupbox.

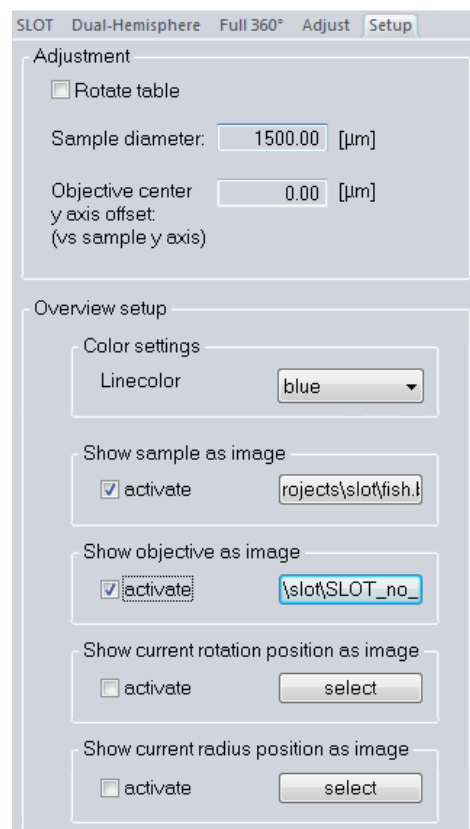


Figure 43: Life Dialog - Setup

### 9.3.8 Hardware Dialog

This dialog is divided into 4 groups. The motor configuration, the rotation axis and radius axis setup and the hardware calibration.

The H2P device connects two hardware parts. The rotation and radius motor via serial lines. It is possible to uncheck the motor position feedback for measurements, where speed is more important than positioning accuracy. Below, the initialization status is shown.

The rotation axis setup shows its stepping range and its main adjustment positions, where the sample may be tilted. The button *Take* is for the *LaVision BioTec service*. In case of lost reference of the rotation motor, the button *check if referenced* can be used to reference the motor again.

The radius axis setup shows the ideal hardware stepping position of its center of rotation and its stepping range. Usually, the possible range is far lower. *If you set a range too wide, you may destroy the hardware!*

Below, you can set the current range to be controlled in the life dialog. In the figure there is a range of 5120 microns.

The last part is for hardware calibration. There is the mapping for steps vs microns, the real center of axis position as seen in the sample image taken after tilting the sample. Press button *Take* to set this position once sample is prepared.

Below is the working distance of the objective and the rotation direction, which can be changed.

Hardware Settings: SLOT

Rotation Motor: (none)

Radius Motor: (none)

Position feedback (for measurement only)

Rangestart feedback (for measurement only)

Initialization successful!

Rotation axis setup check if referenced

0° 0 360° 12800

Adjust A 3200 Take Adjust B 9600 Take

Radius axis setup

Axis center of rotation: 12000 [Step]

Axis movement range for all objectives:

Min 0 [Step] Max 27500 [Step]

Axis movement range for current objective:

Out 18000 [Step] In 6000 [Step]

2560 [µm] -2560 [µm]

Hardware calibration

Radius axis: 2.344 [Steps/µm]

Radius axis zero position: 2650.6 [µm] Take

(rotation radius = 0) 11787 [Step]

Objective working distance (WD): 2000 [µm]

Rotation: clockwise direction

Figure 44: Hardware Dialog

## 9.4 Motorized XY Table

The motorized XY table device allows automatic movement of your sample. You can control the movement with its joystick or with the Life dialog.

You can also use the device for multi position scans or stitching measurements.

The XY Table has three different life dialogs: One visual representation for multi position and mosaic mode measurements. The other two are simple dialogs. One for multi position and one for mosaic mode measurements. In the hardware dialog you can select which dialog to show.

### 9.4.1 Visual Scan Life Dialog

The Visual Scan Life Dialog is initialized with an empty stage overview. That view is centered on the current position of the table.

Above is a tab bar to select either Mosaic or Multiposition mode. The table device communicates with the measurement wizard, so that the mode change triggers a measurement mode change of the wizard if properly configured - and vice versa!

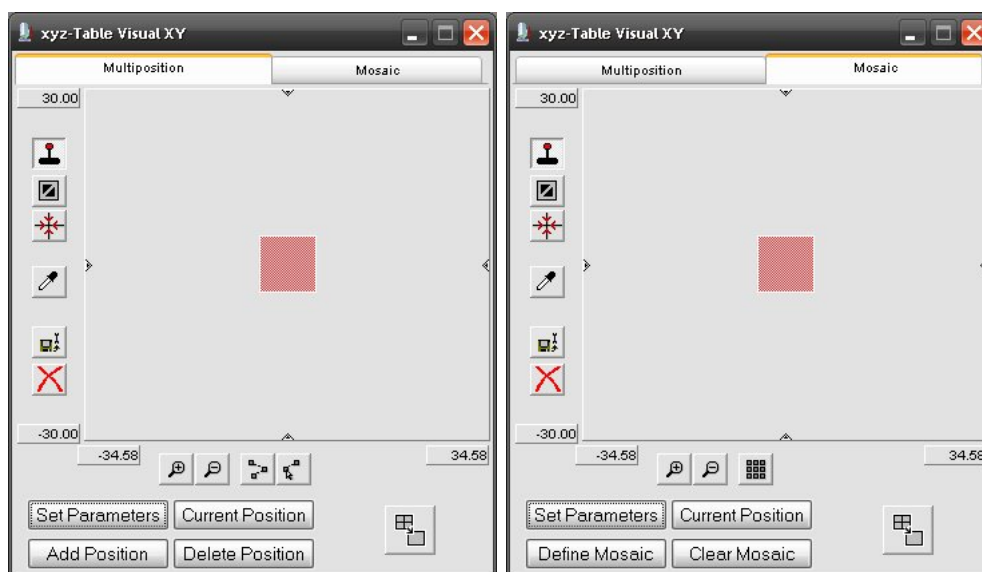

















Figure 45: Visual Stitching Dialog.

```
axisconfig.ini:  
[AC02]  
LABEL=3D Mosaic  
AXIS_FIRST=xyz-Table X,Split  
AXIS_SECOND=xyz-Table Y,Split  
AXIS_THIRD=xyz-Table Z  
AXIS_FOURTH=  
ICONFILE=3D_Filter_150.bmp  
COMMENT=StitchingMosaic  
  
[AC03]  
LABEL=3D Multiposition  
AXIS_FIRST=xyz-Table X,Split  
AXIS_SECOND=xyz-Table Z  
AXIS_THIRD=
```

```
AXIS_FOURTH=  
ICONFILE=3D_Filter_150.bmp  
COMMENT=StitchingPosition
```

On the left and bottom of the view are edit boxes to set the size of the stage overview. The width and height of the view stays consistent! There are also many icon buttons explained in the following list:

-  Enable or disable the joystick to move the current position.
-  Zoom to the range of positions.
-  Set current position as origin.
-  Select a stack window to take its image properties.
-  Load positions of last session.
-  Delete all positions of this mode.
-  Stop all stage movement immediately; measurements are aborted.
-  Zoom into the view.
-  Zoom out of the view.
-  Shift mosaic positions to the multiposition mode. (*mosaic mode only*)
-  Show ascending measurement order of positions. (*multiposition mode only*)
-  Define ascending measurement order to shorten table movements by clicking on the positions. (*multiposition mode only*)
-  Activate Z offset. (*multiposition mode only*)
-  Activate manual position offsets[9.4.1].
-  Open the "Stitching" dialog. [9.4.2]

## Mosaic mode buttons

- "Set Parameters" - Open dialog for fine tuning of the measurement.

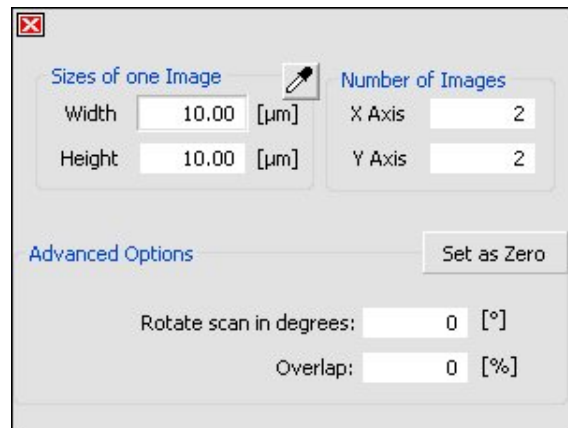


Figure 46: Stitching Option Dialog.

With this dialog you can set the image properties and the number of images taken on each axis. As advanced options you can set an overlap for the images as well as the rotation of the mosaic. The button "Set as Zero" sets the current position as origin.

- "Current Position" - Open dialog for editing the current position.



Figure 47: Current Position Dialog.

Here you can edit the x and y coordinates of the current position. The "Move to" button updates the table. Again, there are the image properties options and the "Set as Zero" button.

- "Define Mosaic" - Create or extend a mosaic bounded by the current position.  
If there is no mosaic it creates a single tile mosaic at the current position.
- "Clear Mosaic" - Deletes the mosaic.

## Multiposition mode buttons

- "Set Parameters" - Open dialog for editing the active position.



Figure 48: Active Position Dialog.

There are edit boxes for the number (of the measurement order) of the position and its x and y coordinates. You can select the positions with the spin buttons to the right. There are also the image properties options.

Changing the coordinates simultaneously updates the position.

Changing the number of the position followed by pressing the *return key* shifts the position to that number (of the measurement order).

- "Current Position" - Open dialog for editing the current position.




Figure 49: Current Position Dialog.

Here you can edit the x and y coordinates of the current position. The "Move to" button updates the table. Again, there are the image properties options and the "Set as Zero" button.

A direct access to the positions is possible by clicking on the grey tinged "No." edit field.

- "Add Position" - Set the current position as a multiposition mode position, if there is none yet!
- "Delete Position" - Remove active position from multiposition mode positions.

-  Show ascending measurement order of positions.

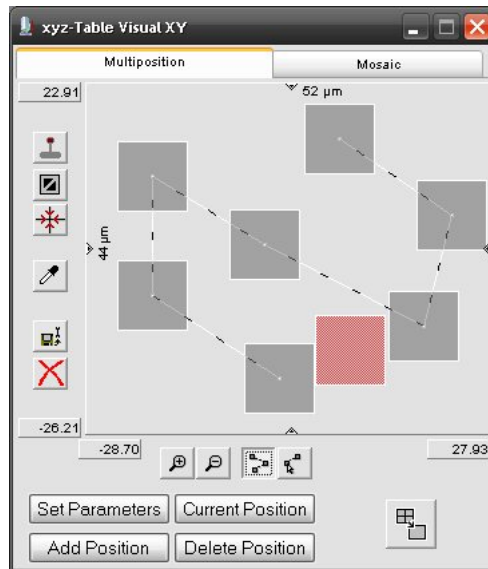



Figure 50: Current Position Path.

-  Define ascending measurement order to shorten table movements by clicking on the positions.

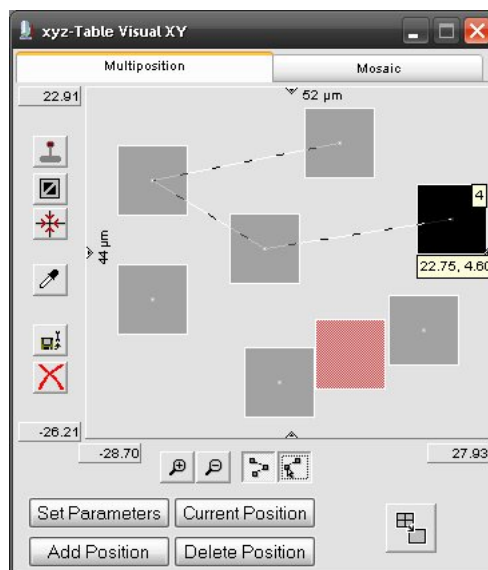



Figure 51: Current Position Path define.

-  Activate different Z offsets for different positions (*multiposition mode only*).

After activating, the positions dialog expands: for each position a different Z offset can be defined. By clicking the "Set" Button the current position of the Z stage is taken over as start value for the selected position - the offset is shown in the edit box.



Every position can have a different z offset. You can also enter the offset manually in the edit box. By clicking "Reset" the offset of the active position is set to zero. By clicking "Reset all" the offsets of all positions are set to zero.

*The number of steps and the stepsize are the same for all positions. These settings can only be changed in the Z dialog.*

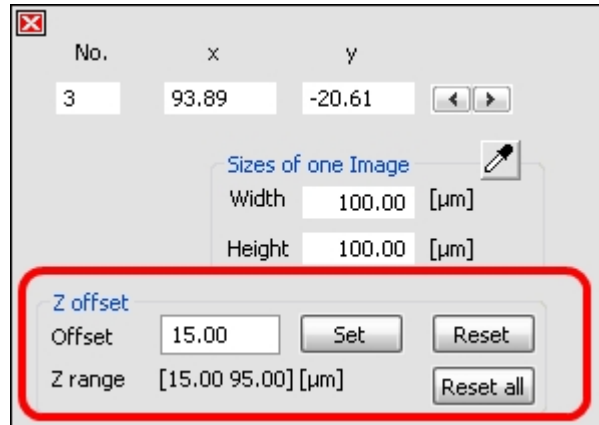



Figure 52: Position Z Offset.

- 
 Activate manual position offsets.
 

With manual position offsets it is possible to adjust the current X,Y or Z position during a measurement. If the sample moves or gets out of focus you can activate manual position offsets to correct the position. When manual position offsets is activated a new dialog opens:

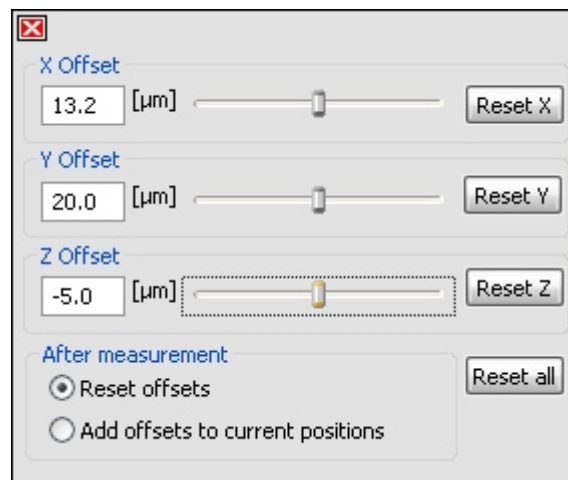


Figure 53: Offset dialog.

The offset can be adjusted by moving the slider for each dimension. For safety reasons there is a limit of maximal offset. This limit must be specified in the hardware dialog of the stage. To activate the offset function check 'Allow XYZ offsets' in the hardware dialog.

After the measurement finishes the offsets are either reset (*'Reset offsets'*) or added (*'Add offsets to current positions'*) to the current stage positions.

For writing own tracking or focus functions, the manual offset can also be accessed via the Python Virtual Device (9.20).

### Mouse and keyboard interactions

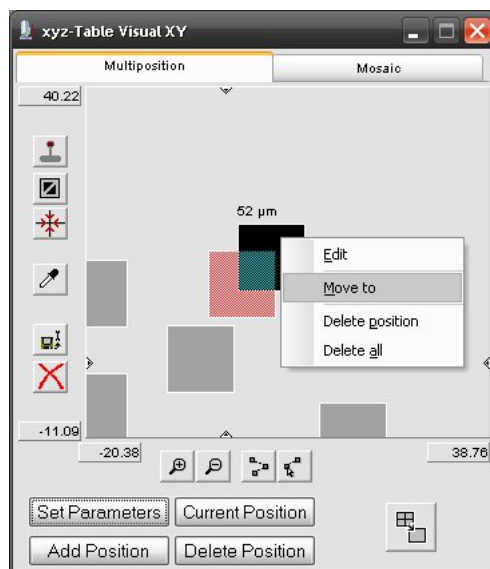
If you hover the mosaic, black dots appear at the edges and corners to resize the mosaic. To actually change the mosaic you have to hover a dot, press the mouse button and move the mouse. When that happens the mosaic changes its colour.



Figure 54: Visual Stitching Dialog ROI.

You can click on a position to activate and highlight it. If the left mouse button is pressed on a position, the current position or the stage overview and moved the respective object shifts. At the mosaic mode the complete mosaic shifts, not just the activated position. This is also possible via keyboard with the *up*, *down*, *left* and *right* keys. Either without or with one of the *shift* or *control* keys.

At the multiposition mode, it is possible to right-click on a position to open a context menu. There are entries for editing the position, moving the current position to that position, deleting that position and deleting all positions.



### 9.4.2 The Stitching Dialog

Figure 55: Current Position Contextmenu.

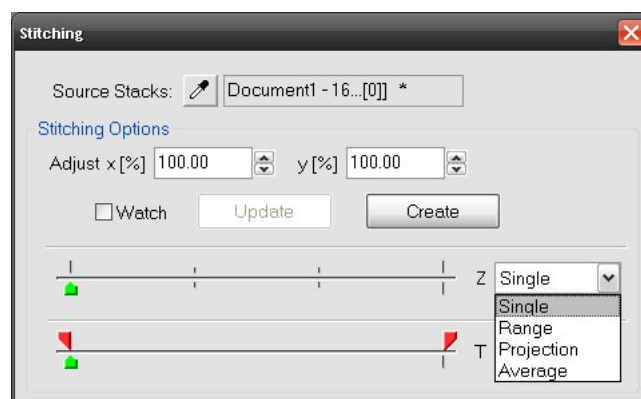



Figure 56: The Stitching Dialog.

You can select a source stack view with the  pipette icon button. In the "Stitching Options" frame you can adjust the physical size for the stitched result. If the source stack view is a measurement document you can activate a watch before starting a measurement to update the stitched result automatically. The "Update" button does the before-mentioned once. The "Create" button generates the stitched stacks into a new view.

*How to stitch Inspector mosaic data with the freely available image processing tool **fiji** see these stitching instructions[13.1]*

### 9.4.3 XY Position Scan Life Dialog [old dialog]

The table position can be changed by mousewheel, joystick or direct input in the edit boxes in the top left. If you use the joystick, the current X and Y table positions are also shown in these boxes.

By clicking the "Add" button the current position is added to the scan list. In the [Measurement Wizard](#) you can select the "X-Axis" of this device. If you select it, during your measurement the table goes to these positions one after another.

If you click at a list entry, the table moves to its position. To delete a position from the list,

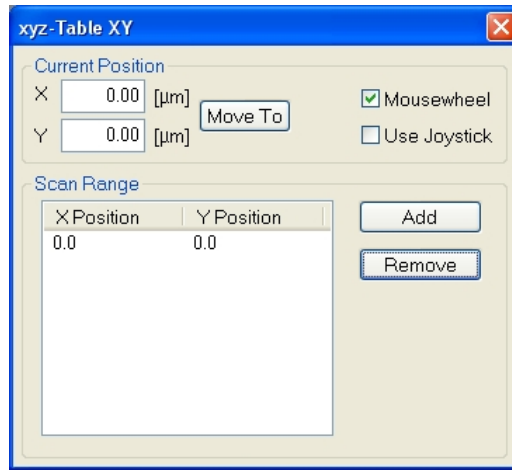


Figure 57: XY Position Scan Dialog.

click the “Remove” button.

#### 9.4.4 XY Mosaic Scan Life Dialog [old dialog]

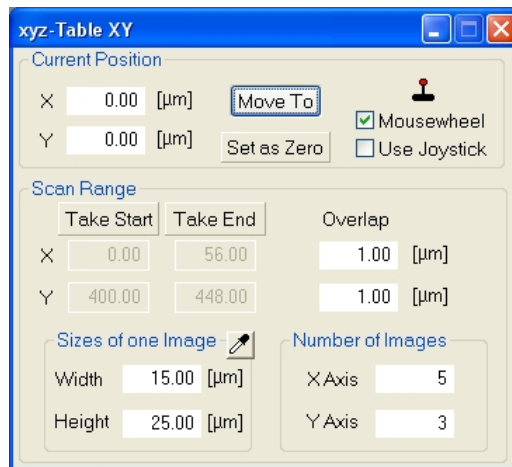


Figure 58: XY Mosaic Scan Dialog.

Like in the position scan dialog you can select your position by mousewheel, joystick or direct input. The start position for your stitching measurement can be set by clicking “Take Start”. The End Position is calculated automatically.

To get the correct calculation you have to enter the physical sizes of one image. You can use the pipette button to copy the sizes of your current stack. The number of images for X and Y direction can be entered in the input boxes on the bottom right. If you wish, you can also enter an overlap between two stitching frames.

## 9.4.5 Hardware Dialog

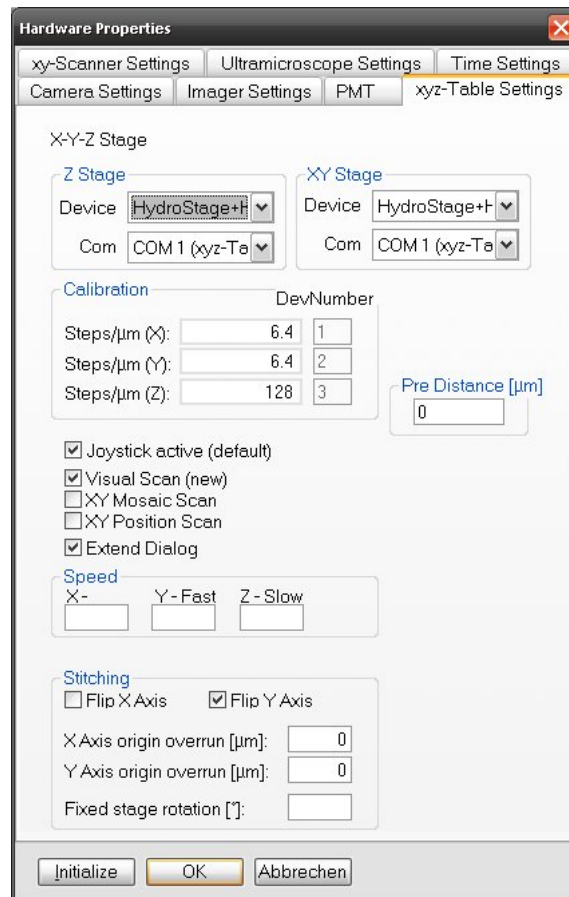


Figure 59: XYZ Hardware Dialog

In the hardware dialog is set up which kind of table device you are using. The comport connection, the table step ratio and specific table speed values are stated here. These values shouldn't be changed by the user!

You can decide which table mode you want to use:

- Check "Visual Scan" if you want to use the new visually represented life dialog for mosaic and multiposition scans.
- Check "XY Mosaic Scan" if you want to use the simple mosaic scan life dialog.
- Check "XY Position Scan" if you want to use the simple position scan life dialog.
- Check "Extend dialog" to see the complete stitching dialog. If it isn't checked, the life dialog don't show stitching specific entries.

After changing these values, you have to click 'Hardware'→'Save Settings' and restart Im-spector to see the new dialog.

You can also select, if the joystick should be used as your default control device. Even if it's not your default control, you can always activate it in the life dialog.

At the bottom of the dialog you'll find the 'Flip X' and 'Flip Y' boxes. You may have to check these values if the stitched stacks of your measurement aren't sorted correctly (see above).

The following boxes should not be changed by the user!

The 'X Axis origin overrun' and 'Y Axis origin overrun' boxes are for Ultramicroscopes displacements for mosaic/multiposition scans. The 'Fixed stage rotation' box is usually for misaligned cameras.

## 9.5 Motorized Z Stepper

### 9.5.1 Life Dialog

The life dialog of the Motorized Z Stepper is organized in three parts:

- **Current Position**  
Set the current and its zero position. Choose mousewheel or joystick to control.
- **Scan Range**  
Within this group box the range to scan with its stepsize and number of images is configured.
- **Z Power Adaptation**  
This opens a new dialog to set the z power increment for available attenuators. *This is not available for the Ultramicroscope!*

### 9.5.2 Hardware Dialog

See [9.4.5] from Motorized XY Table for a description.

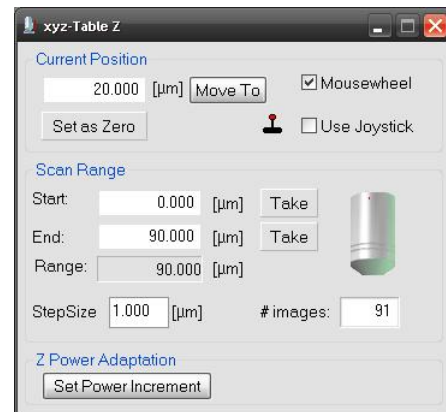


Figure 60: Motorized Z Stepper Life Dialog

### 9.5.3 Z Power Adaptation

With this extension dialog appearing in the Z Stepper Life dialog up to three attenuators can be controlled. The Z Power Increment can be set via five kinds of functions: Points, Linear, QuadraticFixed, Quadratic, Exponential.

- *Points* - This is a linear point-to-point function. The points can be added or removed by the user. The green point, which represents the actual power and z position, is used to add a point.
- All other types implement the function from start to end position of the Z Stepper Life dialog. The quadratic and exponential functions have a helper position to generate its coefficients.

The Z Power Diagram for an attenuator consists of a *power slider* to set the actual power of the selected attenuator, an optional *power upper bound*, a *description field*, a *profile* that represents the power (y axis) over the z range (x axis) and four spin buttons. These spin buttons increment or decrement the power for the selected point (top), marked with a circle, all points (bottom), as well as a left-sided or right-sided relative linear power change over the z range, where the end of the other side is fixed.

*Tip: Keep the spin buttons pressed for a continuous movement.*

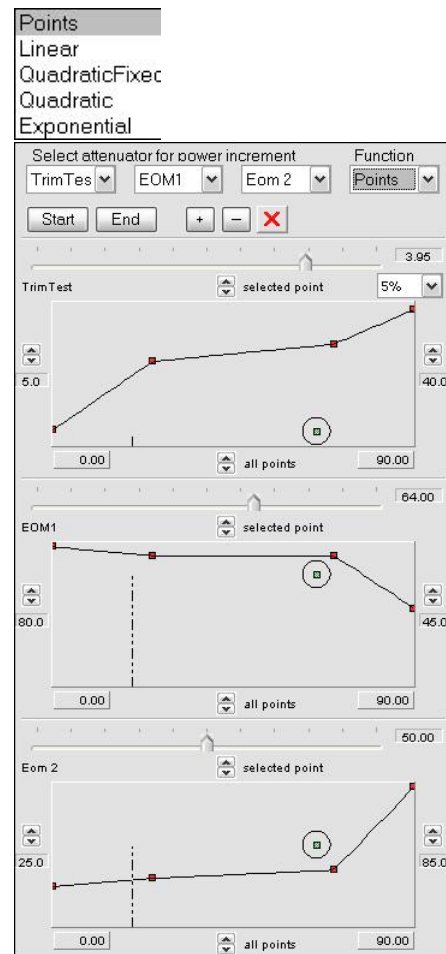


Figure 61: Z Power Adaptation

## 9.6 Pifoc Z Stepper

### 9.6.1 Life Dialog

The life dialog of the Pifoc is organized in five parts:

- **Current Position**  
Set the current and its zero position.
- **Scan Range**  
Within this group box the range to scan with its stepsize and number of images is configured.
- **Slider Control**  
It displays the current settings.
- **Command Mode**  
The command mode manages how to receive a trigger for the next step.  
  
The *fast* modes can be set via measurement wizard by special measurement modes. These are written in the axisconfig.ini file.
- **Z Power Adaptation**  
This opens a new dialog to set the z power increment for available attenuators. See [9.5.3] from Motorized Z Stepper.

The following command modes are available:

- **serial**  
Classic mode where the Pifoc receives triggers over the serial line.
- **frame trigger**  
Free mode where the *PMT* internal axis generates FRAME triggers that the Pifoc has to receive.
- **fast z stack pmt**  
This mode is coupled with the *PMT* internal axis. Its size is taken from the Pifoc life dialog.
- **fast linescan x-z**  
This mode is coupled with the scanner *Linescan* life dialog timelapse field. The Pifoc has to receive LINE triggers.

Optionally, a field to edit the start delay time for the fast modes is shown. There it is possible to tell the measurement to wait for the Pifoc to reach the start position. This happens when you do *Auto-Repeat* measurements.

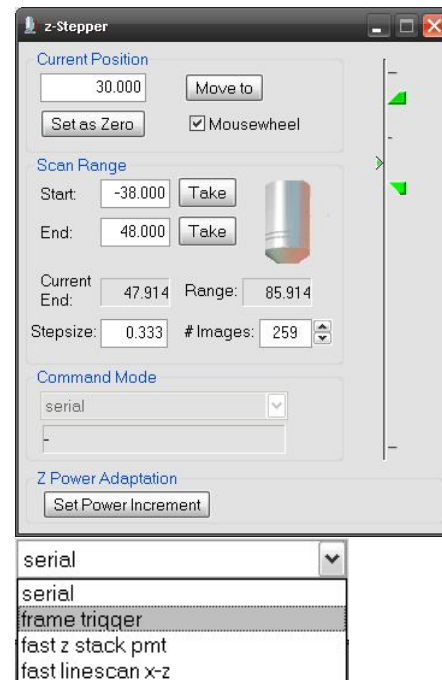


Figure 62: Pifoc Z Stepper Life Dialog



## 9.6.2 Hardware Dialog

The Pifoc Z Stepper is configured by the EcoStepperle device tab.

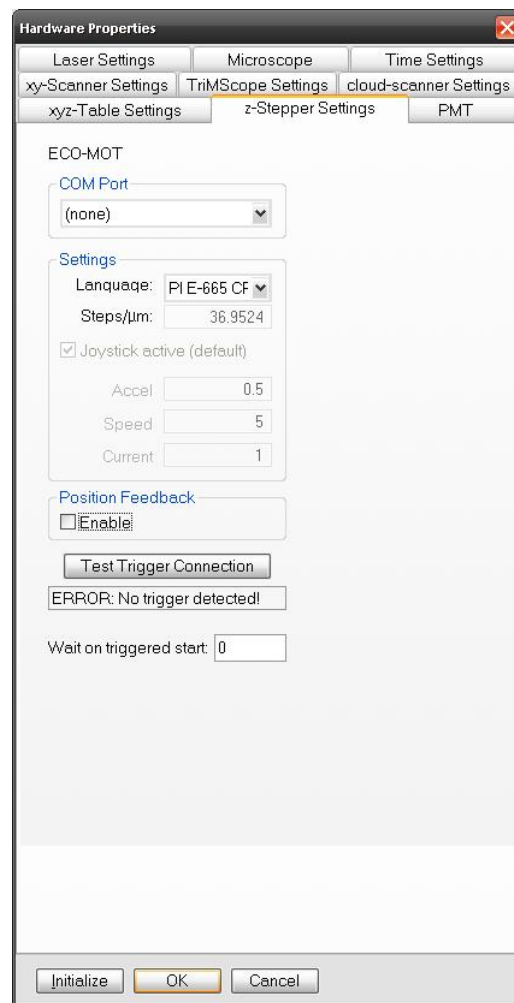


Figure 63: Pifoc Z Stepper Hardware Dialog

Select the language *PI E-665 CR* to enable the Pifoc life dialog.

Enable position feedback to wait for the hardware to end its movement on change. *This is very slow.*

If this language is selected, a button to test the hardware trigger setup appears. With the help of hardware triggers, fast movements along the z axis are possible.

With a big range to scan, the hardware may not be able to reach the start position within a trigger interval of a fast mode. Therefore, it is possible to set a wait time [ms].

## 9.7 FilterWheel

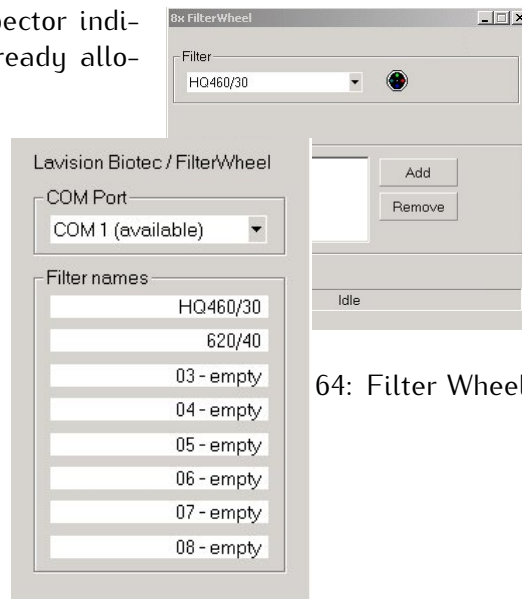
### 9.7.1 Life Dialog

**Filter:** List of installed filters. The names can be changed in the hardware settings.

**Selected Filter:** List of currently selected filters (up to five). These filters will be used sequentially in a measurement.

## 9.7.2 Hardware Dialog

**COM Port:** Select COM port which is connected to the filter wheel. Inspector indicates which COM ports are already allocated by other devices.



64: Filter Wheel Life Dialog.

Figure 65: Filter Wheel Hardware Settings.

**Filter Names:** When you change or load filters into the filter wheel, give them a name here to identify them during a measurement. Axis will be labelled with these strings.

## 9.8 Time Driver

To create a time lapse measurement you can use the *time driver*. For example using a camera, the number of steps set in the *time driver dialog* lets the camera take the specified amount of images. The time between 2 images is either the given wait time or zero, if the checkbox "*Fast Mode*" is checked.

### 9.8.1 Details of the dialog

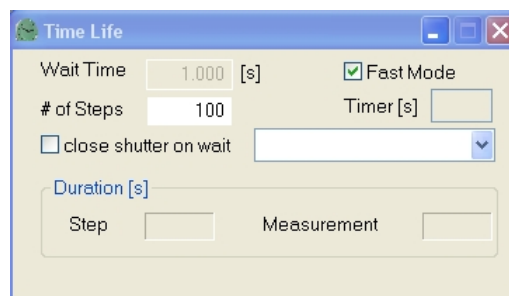


Figure 66: The TimeDriver life dialog.

- Wait Time: The time the driver waits, before the next step is started
- Fast Mode: The next step is started immediately
- # of Steps: Number of steps on the time axis
- Timer: Shows the remaining time till the next step is started
- Close shutter on wait: The shutter (if present) is closed during the driver's wait time (useful if the wait time is long)
- Duration: Shows the duration of the last step and of the measurement

## 9.9 Imager QE

### 9.9.1 Life Dialog

**Exposure Time:** Set the exposure time in milliseconds

**Binning:** Set the binning parameter. (from no binning to 8x8 binning). The counts of several pixels (e.g. 2x2 or 4x4) are combined and displayed as one, thus decreasing resolution and image size. Using higher binning factors normally speeds up the readout time.

**New Background Image:** Take a new background image with the current exposure time.

**Use Background:** If checked, the background image is subtracted from each incoming image.

**New Ratio Image:** Make a new ratio image.

**Use Ratio:** If checked, every pixel in the current image is multiplied by the ratio of the maximum count and the corresponding pixel count of the ratio image.

If both correction options are checked, the background image is subtracted first (from the current and the ratio image) and then the ratio calculation is done.

**ROI:** This shows the current region of interest of the CCD camera. To change the ROI make a new rectangle selection in the measurement window and select Selection→Set as New ROI.

**Full Frame:** Press this button to reset the ROI to full frame.

**Readout Time [ms]:** This field contains the CCD readout time in milliseconds reported by the camera.

**fps\*:** Regarding current exposure time and readout time, here the fastest possible frames per second is shown—if the ccd is driven in internal trigger mode (continuous mode). For changing the trigger mode see Imager QE/hardware settings.

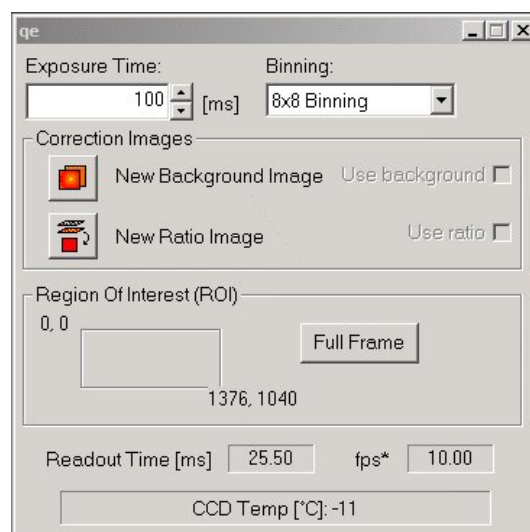


Figure 67: ImagerQE Life Dialog.

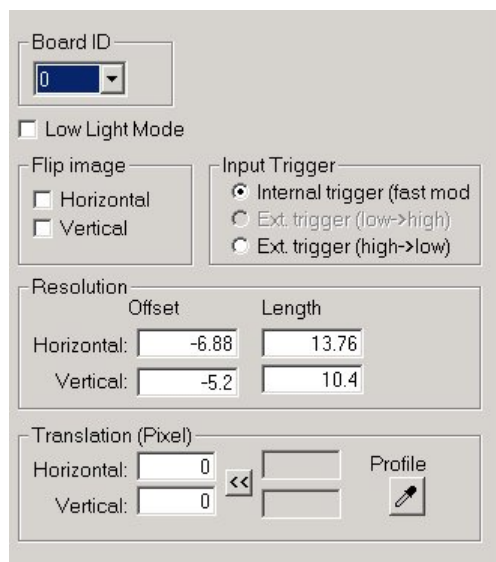


Figure 68: ImageQE Hardware Dialog.

**Board ID:** When using multiple Imager QEs enter the id of the PCI board: 0, 1, 2, ...

**Low Light Mode:** Using the low light mode sets the ccd gain to 2 e<sup>-</sup> / count instead of 4 e<sup>-</sup> / count. Also the quantum efficiency for wavelengths >500nm is increased significantly.

**Flip Image:** Mirrors the image in selected directions.

**Input Trigger:** Internal Trigger lets the Camera run in the continuous mode. No external trigger is regarded. The camera exposures during readout. Fastest mode. In External Trigger mode the camera waits for an external event (e.g. XY scanner) and then starts the exposure sequence. This mode does not exposure during readout and is less fast. However, the camera is exactly synchronized to other experiment devices.

**Resolution:** Defines the dimension of the ccds pixels. These values are used when switching Impector to display physical units.

**Translation (Pixel):** used to align multiple ccds viewports to eachother. One may pickup a translation profile for each ccd. The device driver calculates from these profiles a translation vector which can be applied by clicking the arrow button. Also one can enter the translation vector (horizontal/vertical component) manually.

## 9.10 Ixon

### 9.10.1 Life Dialog

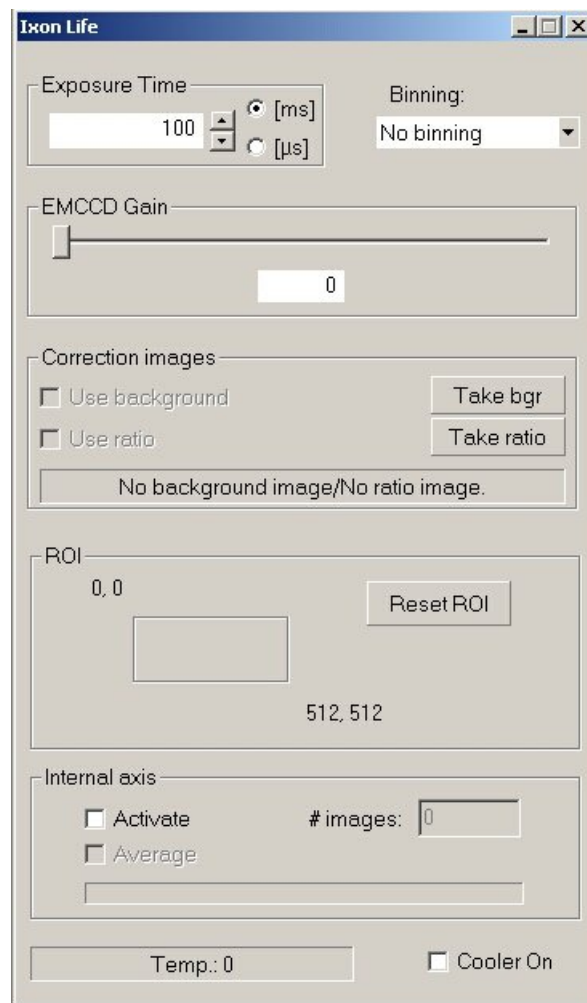


Figure 69: The Ixon Life dialog.

**Exposure Time:** Set the exposure time. Dependend on the selection you made, the exposure time is either in milli seconds or in micro seconds.

**Binning:** Set the binning parameter. (from no binning to 8x8 binning). The counts of several pixels (e.g. 2x2 or 4x4) are combined and displayed as one, thus decreasing resolution and image size. Using higher binning factors normally speeds up the readout time.

**EMCCD Gain:** The gain amplifies the normal CCD output. The higher the gain value, the higher the amplification and thus, the higher the photon detection. If the gain is set high enough, single electron event detection is possible. But note that the use of the gain adds some additional noise to the measured signal due to the statistical nature of the multiplication process. When the signal is high enough to be above noise level, there is probably no need to use a gain. On the other hand, when the signal is lost in noise, the only possibility to see a signal is using the gain.

**New Background Image:** Take a new background image with the current exposure time.

**Use Background:** If checked, the background image is subtracted from each incoming image.

**New Ratio Image:** Make a new ratio image.

**Use Ratio:** If checked, every pixel in the current image is multiplied by the ratio of the maximum count and the corresponding pixel count of the ratio image.

If both correction options are checked, the background image is subtracted first (from the current and the ratio image) and then the ratio calculation is done.

**ROI:** This shows the current region of interest of the CCD camera. To change the ROI make a new rectangle selection in the measurement window and select Selection→Set as New ROI.

**Reset ROI:** Press this button to reset the ROI to full frame.

**Internal Axis - Activate:** When the internal axis is activated, the given number of images is recorded, before they are displayed. This results in a stack of images. Using the internal axis of the Ixon camera is the fastest mode available for imaging.

**Internal Axis - Average:** When this option is active, Inspector will record the given number of images and calculate the average image out of the image stack. The result is a single, averaged image.

**Cooler On:** Switch the cooling of the ccd chip on and off.

**Temp:** Displays the actual temperature of the ccd chip.

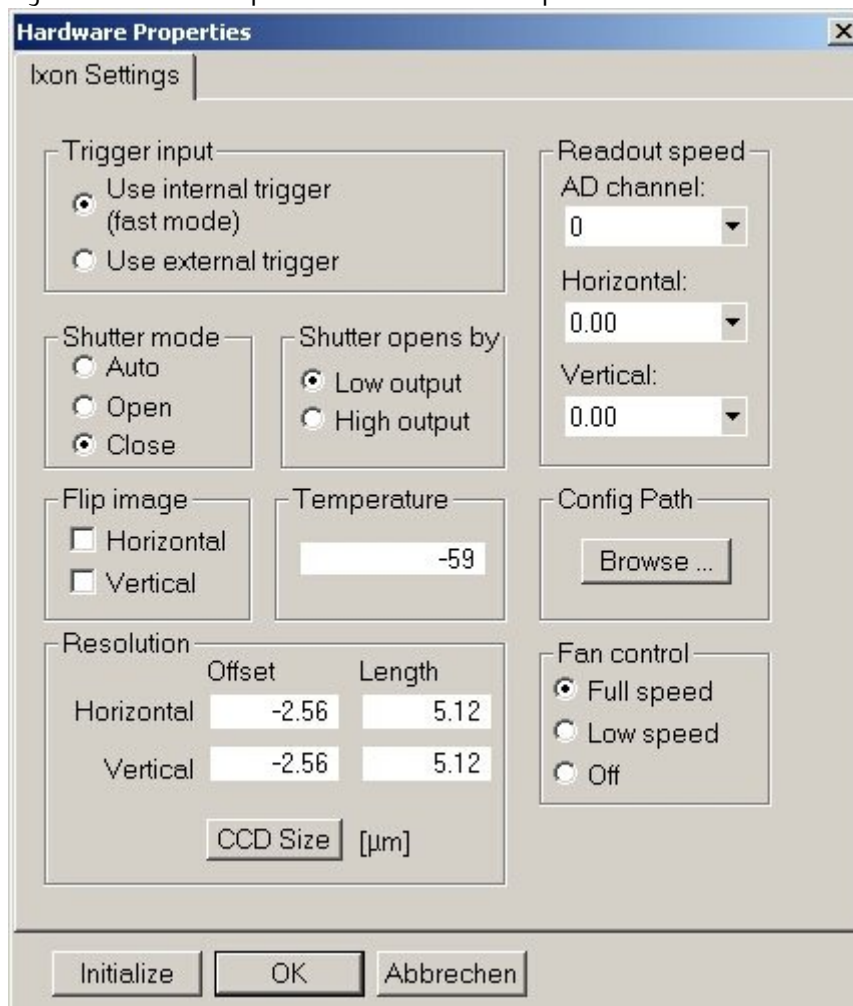


Figure 70: The Ixon hardware dialog.

**Use internal trigger:** Image acquisition will start immediately, when you run a measurement. The software will generate a trigger internally that starts the image acquisition.

**Use external trigger:** The image acquisition will be synchronized with an external trigger pulse. An image is taken with each trigger pulse.

**Shutter mode Auto:** Let the Andor device driver decide, when the shutter will be opened and when the shutter closes.

**Shutter mode Open:** The shutter will stay open before, during and after data acquisition. This is useful if you want to take a series of images.

**Shutter mode Close:** The shutter will stay closed before, during and after data acquisition.

Use this option if you want to take a dark background scan.

**Shutter opens by - Low output/High output:** This option lets you define, when the shutter should open/close. If you select Low output, the shutter will close when a high electrical signal is used and open, when a low electrical signal arrives.

Thus, the shutter logic can be inverted with this feature. The normal option should be High output.

**Readout speed:** The AndorMCD system allows you to set the speed at which charge is shifted horizontally and vertically on the CCD.

The vertical shift speed is the speed at which each row of the CCD chip is shifted vertically into the shift registers.

The horizontal shift speed is the speed at which the shift registers are shifted horizontally into the A/D converter. This also defines the speed at which the signal is digitized via the onboard A/D converter.

The horizontal shift speed can be defined for each available A/D channel.

**Flip image:** Automatically flip the image horizontally or vertically after data acquisition. **Temperature:** Define the temperature at which the CCD chip will operate.

**Config Path:** Choose the path, where the Ixon driver and config files reside.

**Resolution:** Here you can define the real dimension of the visible camera image. In Illustration 15 on page 27 the visible camera image is  $5.12 \mu\text{m}$  in width and height. The virtual axis that is laid into the image starts at  $-2.56$  and runs to  $2.56$ , thus the middle is exactly at  $(0/0)$ . The offset defines the left beginning of the axis.

**Fan Control:** The internal fan of the Ixon camera can run at two different speeds, or it can be switched off to eliminate vibrations.

## 9.11 PixelFly

### 9.11.1 Life Dialog

**Exposure Time:** Set the exposure time. Depending on the selection you made in the hardware dialog, the exposure time is either in milli seconds or in micro seconds.

**Binning:** Set the binning parameter. (from no binning to  $2 \times 2$  binning). The counts of several pixels (e.g.  $1 \times 2$  or  $2 \times 2$ ) are combined and displayed as one, thus decreasing resolution and image size. Using higher binning factors normally speeds up the readout time.

**Take background:** Take a new background image with the current exposure time. Use Background: If checked, the background image is subtracted from each incoming image.



Figure 71: PixelFly

### 9.11.2 Hardware Dialog

#### Trigger

**Software Trigger** With the software trigger, the camera will start recording immediately when a measurement is run. There will be no synchronization with other hardware devices at all.

**External Trigger** In external trigger mode, the camera will synchronize itself to an external trigger input. The recording of a new image will only start, when a trigger signal is received from any external source.

If there are no external sources available, but the camera should be synchronized to the measurement loop, this driver can generate its own trigger signal, either over the LPT port, or even internally with a simulated software trigger.

**Camera Mode** The Pixelfly camera support image acquisition in two different modes.

In single image mode there will be only one image recorded at each trigger impulse. The exposuretime can be from  $10\mu s$  to  $10000\mu s$ .

In video mode, a sequence of images is recorded at each trigger impulse. The exposuretime lies between 1ms and 65000ms.

The Low Light mode amplifies wave lengths over 500ns.

**Flip Image** According to the setting, the image will be flipped horizontally or vertically after acquisition.

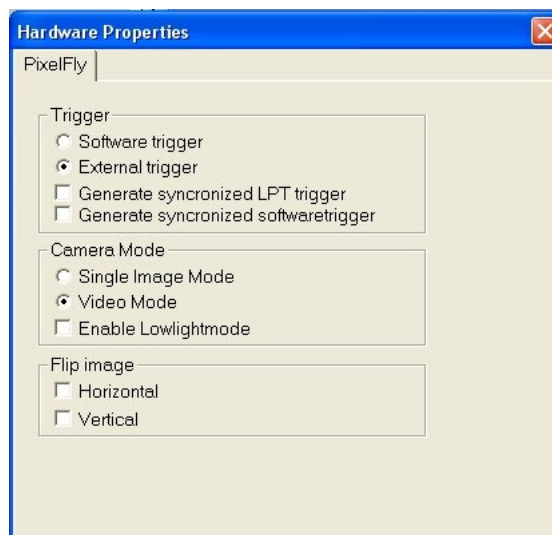


Figure 72: The Pixelfly Hardware Dialog.

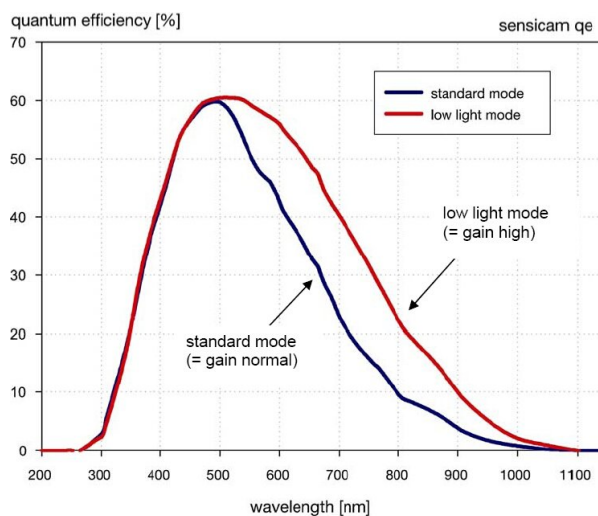


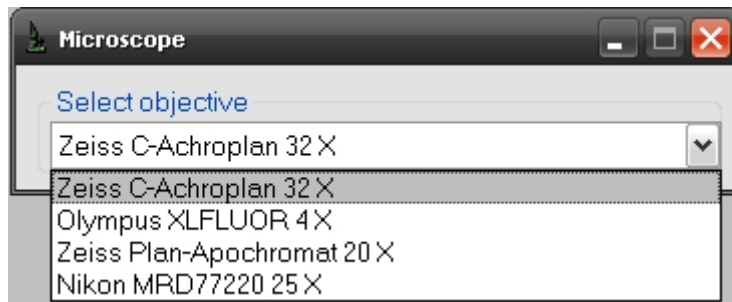
Figure 73: The Low Light Mode of the Pixelfly. Wave lengths over 500nm are amplified.

## 9.12 Microscope Device

The Microscope Device manages different magnification settings of your microscope system. You can select which objective you are using and impector calculates the correct physical sizes automatically.



### 9.12.1 Life dialog



In the life dialog select the objective you are using.

## 9.12.2 Hardware dialog

**Hardware Properties** ✕

xyz-Table Settings   Autofocus   PixelFly   PMT  
xy-Scanner Settings   Laser Settings   **TriMScope Settings**   EOM  
z-Stepper Settings   **Microscope**   Time Settings

**Set microscope stand**

TriM Scope I  
 TriM Scope II

**Tube length / magnification**

Tube length     
System magnification      [mm]  
 Use for scanner device

**Change objective settings**

**Objectives**

<input checked="" type="checkbox"/> Zeiss C-Achroplan 32 X	<b>Manufacturer</b> <input type="text" value="Zeiss"/>
<input type="checkbox"/> Olympus XLFLUOR 4 X	<b>Name</b> <input type="text" value="C-Achroplan"/>
<input type="checkbox"/> Zeiss Plan-Apochromat 20 X	<b>Magnification</b> <input type="text" value="32.00"/>
<input type="checkbox"/> Nikon MRD77220 25 X	<b>NA</b> <input type="text" value="0.85"/>

**original tubelength**

**immersion medium**  
 ▼

In the hardware dialog the microscope stand must be selected: Trim Scope I or Trim Scope II. To each stand a standard tube length value is loaded. If needed this value can be adjusted manually.

By combining the selected objective with the microscope stand (respectively its tube length) the *system magnification* is calculated. This magnification should be used for the scanner device to generate the correct physical sizes. For this purpose you can activate "Use for scanner device". (If it is activated the magnification cannot be adjusted in the scanner device any more.)

The available objectives are shown in the list below. You can modify, add or delete objectives here. These information are stored in the "objective.ini" file.

### 9.12.3 ini file

This file is stored in the config folder of ImSpector. It contains the following entries:

```
[0]
Manufacturer=Zeiss
Name=C-Achroplan
Magnification=32
NA=0.85
Immersion=water
Tubelength=164.5
[...]
[3]

Manufacturer = Nikon
Name = MRD77220
Magnification = 25
NA = 1.1
Immersion = water
Tubelength = 164.5
```

In this file a list of objectives is specified. To each entry a manufacturer, name, magnification, immersion medium and tube length is given. The tube length depends on the manufacturer of the objective.

## 9.13 Autofocus

For recalibrating and keeping the correct Z position during long time measurements you can use the *Autofocus device*.

The device can be used with special *autofocus* hardware and with normal PMT hardware. During the measurement the device calculates an offset, that is used by the Z device of the system to keep the correct Z position.

### 9.13.1 General Settings

In the reconfig.ini: add a special *autofocus* entry. This entry lists which devices of the system are used during a *autofocus* calibration step. Depending on the devices of the system, the entry might look like this:

```
reconfig.ini:
[...]
[Autofocus]
SWITCHON=PMT,xy-scanner,Autofocus,xyz-Table
SWITCHOFF=Imager,Ixon,Filterwheel
```

In the **hardware dialog**: Select which device should be used for the *autofocus* device. You can activate if a logfile should be created. During a measurement the calculated *autofocus* offsets are stored into this file. If special *autofocus* hardware is installed on your system, check "use autofocus hardware".

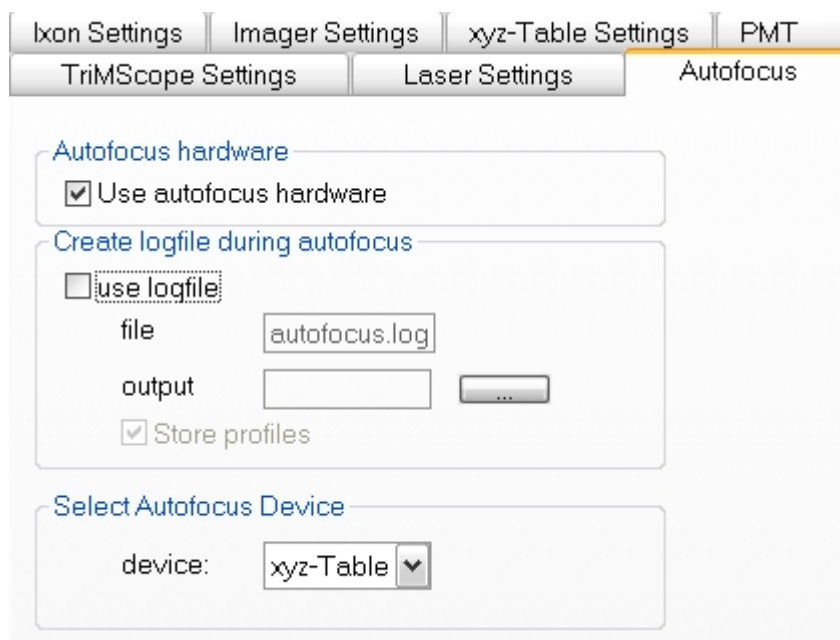


Figure 74: Autofocus Hardware Dialog.

In the **life dialog**: Select at which position in your measurement an *autofocus* recalibration step should be start. Therefore select an axis and a step number in the top right of the dialog. If "Use autofocus" is checked, an *autofocus* step starts when the specified position is reached.

For using *autofocus* a reference profile must be created. The profile is created from a Z stack. This stack should contain a special characteristic or structure that can be seen in the profile.

### 9.13.2 Creating reference profile

There are two different ways for creating reference profiles. When using the special *autofocus* hardware in the life dialog the tab "Scan" must be selected. When using the standard PMT hardware the tab "From profile" must be selected.

#### autofocus hardware

- Select "Scan" tab.
- Activate *autofocus* mode.
- Start a measurement (Video).

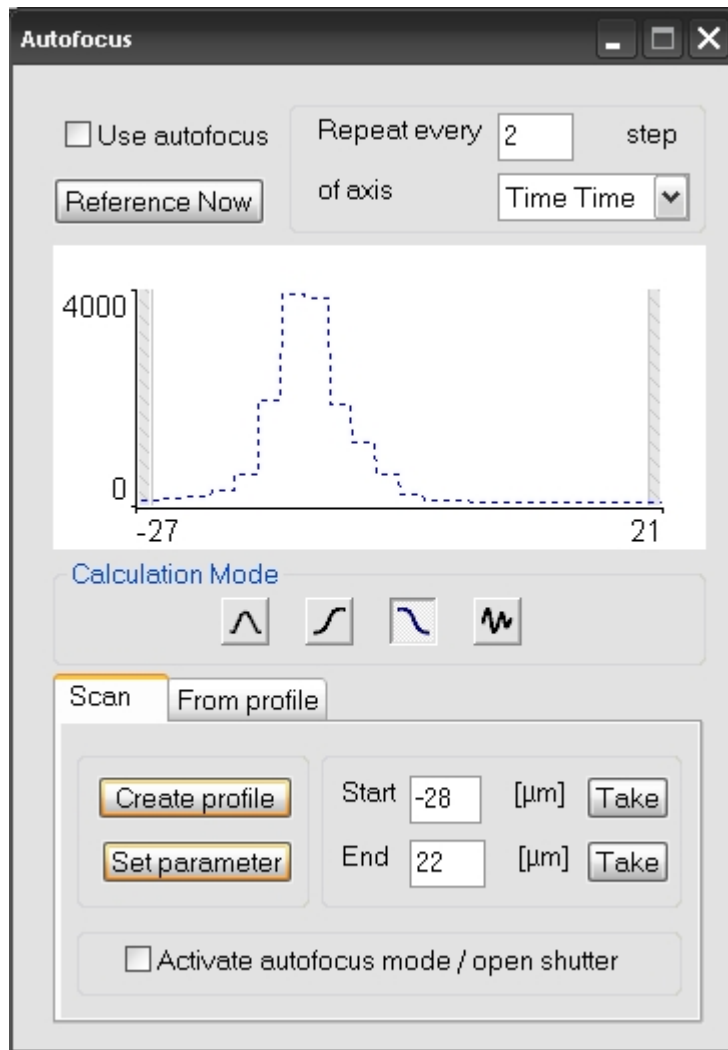


Figure 75: Autofocus Life Dialog.

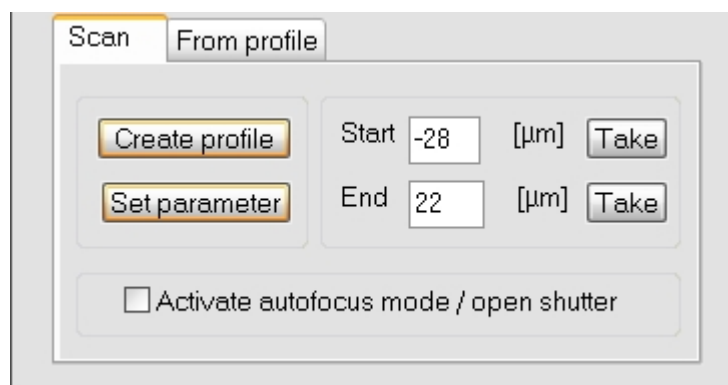


Figure 76: Autofocus Scan Tab.

- With the Z device: Go through the sample, use "Take" buttons to select start and end position of the reference profile.
- Stop the measurement. *Autofocus* mode can be deactivated.
- Click "Create profile" to generate the reference profile. The profile is shown in the life dialog.

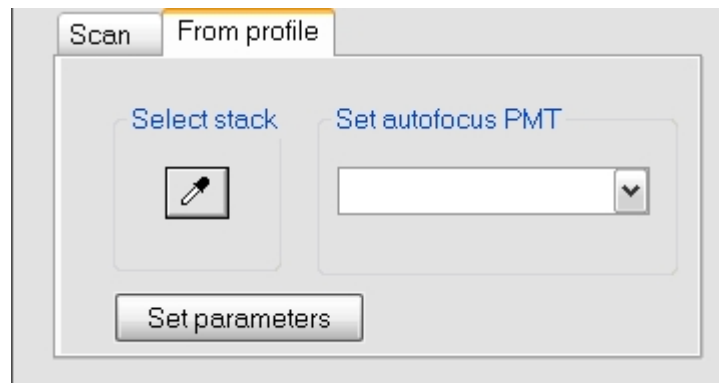


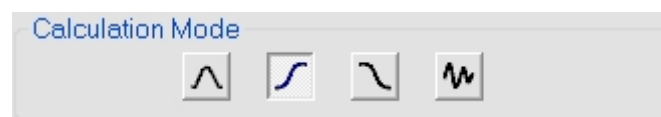
Figure 77: Autofocus Profile Tab.





#### standard hardware

- Select "*From profile*".
- Select PMT that should be used for *autofocus*.
- Create a Z stack that can be used as reference.
- Select stack by using the pipet button: the reference profile is shown in the life dialog.

#### 9.13.3 Calculation modes

During an *autofocus* step another profile is created. The *autofocus* offset is calculated by the reference profile and this new one. There are four different ways how the *autofocus* offset is calculated:



-  The Z position of the peak value of both profiles is compared.
-  The Z position of the maximum rise of the profiles is compared.
-  The Z position of the maximum descent of the profiles is compared.
-  The peak value of a cross-correlated profile is used.

### 9.13.4 Additional Settings

There are additional settings for the *autofocus* device:

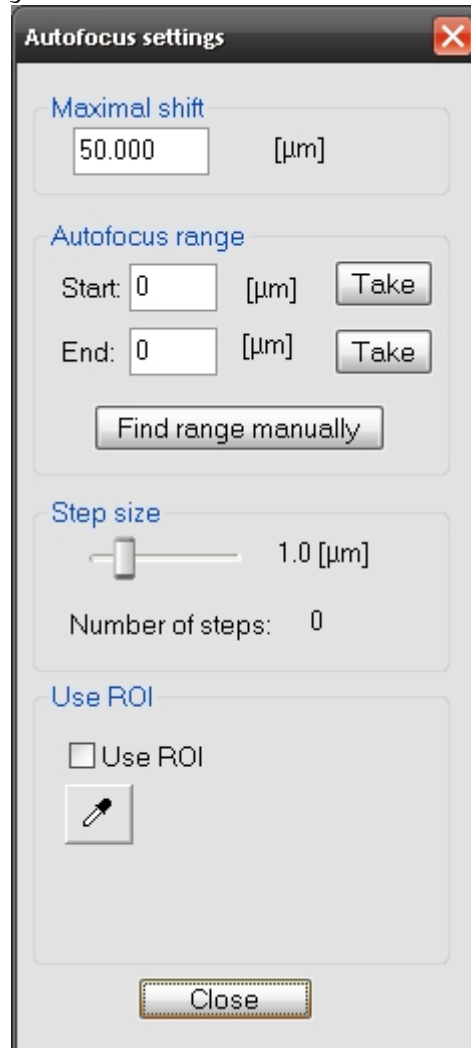


Figure 78: Additional Autofocus Settings.

- The maximum shift value can be specified: If a higher *autofocus* shift is calculated the shift is limited to this value.
- The *autofocus* range can be specified manually or by using the "Take" buttons.
- By clicking "Find range manually" a measurement starts with the defined *autofocus* hardware: The profile is created on the fly while the Z level is moved through the sample.
- The step size can be specified. Depending on the step size the number of steps changes. Default step size is  $1\mu\text{m}$ .
- Specify a ROI that is used for creating the profiles. Select a ROI in your stack and use the pipet button to copy it into the dialog.

### 9.14 Optical Delay

To overlay the beam of different lasers and to modify this overlay, the *Optical Delay* device can be used.

### 9.14.1 Hardware Dialog

The following settings can be done in the hardware dialog [79] of the device:

The screenshot shows the 'OpticalDelay Settings' dialog box. It is divided into several sections, each with a red number indicating its order:

- 1 Device status:** A text box containing 'Initialization successful!'.
- 2 Select com port:** A dropdown menu showing 'COM 1 (available)'.
- 3 Set delay:** A text box with '0.0020832999' and a label '[ps] delay / step'.
- 4 Range settings:** Two radio buttons for 'minimum position' (selected) and 'maximum position'. Below them is a slider, a text box with '0', and a 'Set' button.
- 5 Laser settings:** A section for 'Laser 1' with a dropdown menu set to 'InSight'. It includes two rows of settings: 'Lambda.Center1' with value '800' [nm] and 'Delta.lambda1' with value '0.400' [ps/100 nm].
- 6 Laser settings:** A section for 'Laser 2' with a dropdown menu set to 'MaiTai'. It includes two rows of settings: 'Lambda.Center1' with value '110' [nm] and 'Delta.lambda1' with value '0.300' [ps/100 nm].

Figure 79: Optical Delay Hardware Dialog.

1. Device status: hardware information is shown here.
2. Select Com Port: Specify which com port is used for the device.
3. Set delay: Sets the delay time in picoseconds of one step of the device motor.
4. Range settings: sets the range of the slider used in the life dialog. Select minimum / maximum button to see the current settings. Click on *Set* to keep the slider position for the currently selected position.



5. Laser settings 1: Here the laser 1 and its standard wavelength (*lambda center 1*) must be specified. Delta lambda 1 specifies the adjustment of the delay line according to the actually used wavelength. The computed dispersion correction is shown in the life dialog.
6. Laser settings 2: Same settings as above for second laser.

### 9.14.2 Life Dialog

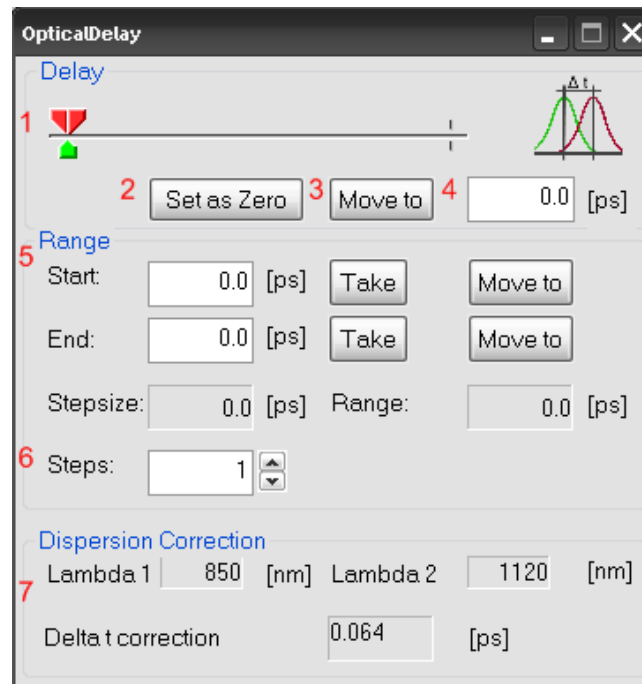


Figure 80: Optical Delay Hardware Dialog.

The *Optical Delay* device can be used with a fixed delay or with a delay range that is adjusted during a measurement. The optical delay device provides an own data axis that can be selected in the [Measurement Wizard](#). The following settings can be done in the life dialog [80]:

1. The currently used delay is specified by the green slider. The value is shown in (4). If using a range, it is specified by the two red sliders.
2. Set as zero: the current position of the green slider is set as position 0.
3. Move to: By clicking on this button, the delay is set to the value, specified in (4).
4. Current position of the green slider is shown here. Values can be entered here manually, to move to a specified position.
5. Range settings: Start & end values of the range can be specified here. By clicking on the upper 'Take' button the currently selected delay value is taken over as start value. By clicking on the lower 'Take' button the value is taken as end value. By clicking on 'Move To' the currently used delay value is set directly to the start (or end) value.
6. Steps: Number of steps of the range is specified here.

7. Dispersion Correction: For using other wavelengths than specified in the hardware dialog ( $\lambda$  center 1 and 2) a dispersion correction is computed automatically. How much correction is computed can be specified in the hardware dialog.

## 9.15 TCSPC

The TCSPC acquires fluorescence lifetimes.

There are two devices available. The FLIM x16 detector, controlled via TDC driver, and the dual-channel detectors FLIM X1/X2, controlled via scTDC driver.

The following description is for the FLIM X1/X2 detectors. Though, the FLIM x16 detector is controlled in a similar way.

### 9.15.1 Life Dialog

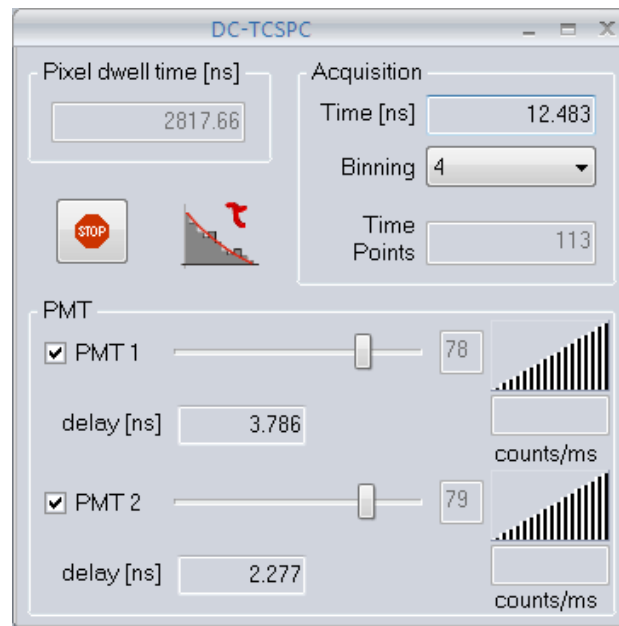


Figure 81: TCSPC Life Dialog

In the upper part are the time parameters. The lower part holds the controls for the PMTs: the gain slider, its delay time and the measured counts per millisecond.

Since the PMTs are very sensitive, **the user must take caution** of the control of the slider. Pushed beyond their limits, which is usually around the gain of 80, **they take damage!** *The countrate view on the right reflects this by turning yellow closing in onto the limit and red if passed!*

The countrate view shows two bars for one measured timepoint. The lighter one represents the count received within the detector. The darker one represents the count received within the hardware driver.

The **STOP** button sets the gain to zero *anytime*.

## 9.15.2 Hardware Dialog



Figure 82: TCSPC Hardware Dialog

The FLIM detector connects to the hardware via USB 3.0.

The dialog consists of configuration parameters, scanner setup, its PMT and measurement parameters. There are also advanced configurations for the hardware, which require a software reinitialization of the device - see at the bottom left corner.

At the top of the dialog is the hardware driver version used by ImSpector Pro. Below

is the *Time bin size*, which is the rasterization of the timeslices. To the right is the configuration file, which is used as a base for the hardware initialization. See the *Advanced box* for user adjustable parameters.

The *Scanner box* shows the current setup of the scanner.

The *PMT box* sets up the colormap used to visualize the images in the document. The gain should be set to the one of the *specification document* shipped with the hardware. The modulo should be around 12500 to 15000 with a 80MHz laser.

If there is a laser device configured, it is possible to set a mapping for modulo vs. wavelength. To activate this the wavelength must be set in the laser life dialog. Then, the modulo can be set for this wavelength in the TCSPC hardware dialog. A mapping can be removed by setting the wavelength and then the modulo to zero; a message box appears, if the selected wavelength is in the map! *A wavelength, which is inbetween two mappings, will not be added to the map!*

The *Measurement PMT box* holds the colormap scaling range from zero to the given maximum, which will be applied to the colormap of the PMT image in the document. Zero means that no scaling range will be applied.

The *Measurement general box* has three settings. The checkbox *Show lifetime overview* enables the visualization of summed-up timeslices. The *cooling time* blocks a new measurement for the given time in seconds to relax the PMT and high-voltage electric circuits. The *data timeout* is the data transmission timeout in milliseconds. This is an extra time to wait for the data to transmit over the usb line.

The *Advanced box* enables the user to overwrite hardware settings which are usually set in the configuration file. The *Auto modulo* configuration sets whether to **always** use the hardware calculated modulo value or a user given modulo; see above. The trigger edges for frame, line and pixel trigger can also be adjusted. The *Pixel timer mode* has 3 options:

- 0 - internal pixel timer: classic mode [default]
- 1 - pixel timer: no correction
- 2 - pixel timer: correction from data flow

The *Start divider* divides the laser reference frequency trigger to the internally used frequency. The internally used frequency is 5MHz. Therefore is 16 perfectly fine for lasers with 80MHz. The *Debug level* is for service use only.

The *Debug box* enables the user to set the USB latency. By default, the hardware requires a time to wait for its initialization to measure a lifetime. Now, that this extra time is not always wanted, it can be set to zero here. *The user must know that with this the hardware runs outside of its specification!*

The measurement logging option is for debugging purposes only.

### 9.15.3 Troubleshooting

- **USB cable length must not be longer than 5m! Active HUBs or Repeater/Cable extensions are used by the users responsibility. Such setups fail in most interesting ways.**
- **Measurement hangs - without other image generating devices** (like the PMT device)
  1. Check pixel(optional), line, frame and laser reference frequency trigger.
  2. Add the TCSPC device to
    - 'Hardware > Configure > xy-scanner > Sync 5'
  3. Check the scanner and its settings.
  4. If it still hangs set the following parameters:
    - 'Hardware > Configure > xy-scanner > Sync 5: TCSPC'
    - 'Hardware > Configure > TCSPC > Advanced: Auto modulo: enable'
    - 'Hardware > Configure > TCSPC > Advanced: Debug level: 3'

Initialize the device several times and confirm each time it has initialized in the newly opened debug window the value of the modulo. If it is stable, the laser reference frequency trigger is detected and the other trigger lines must be checked. Or the threshold needs to be tweaked.

- **Measurement hangs - with other image generating devices** (like the PMT device)
  1. The PMT device must be registered before the TCSPC device! The PMT must be set for the scanner *sync 5*.
    - 'Hardware > Configure > xy-scanner > Sync 5: PMT, TCSPC'
  2. The TCSPC device may not wait for its hardware to initialize for a lifetime measurement.
    - 'Hardware > Configure > TCSPC > Debug: enable'
    - 'Hardware > Configure > TCSPC > Debug: USB latency: 0'

*The user must know that with this the hardware runs outside of its specification!*

- **Variadic delays between image acquisition**
  1. Set the data transmission timeout to a higher value.
    - 'Hardware > Configure > TCSPC > Measurement general: Data timeout'
- **Zero count image | Sporadic dark images**
  1. This can be seen in the image as well as in the life dialog countrate representation. This is usually caused by the the laser *being off* and/or its laser reference frequency trigger *being not connected or deformed* so that the detector does not recognize it. Another cause can be a *too low* or *too high* gain setting. **A too high gain setting causes damage to the PMT!**
  2. A bad configured EOM can cause this.
- **Zero count images for a number of steps before data is acquired**

The cause of this is a too small *cooling time* for the PMT and high-voltage circuits.

Increase the setting of:

  - 'Hardware > Configure > TCSPC > Measurement general: cooling time'

- **PMT image OK, TCSPC image partly shown or corrupted**

Disable PMT feedback. If both images look alike then, the scanner parameters and its coefficients must be calculated!

- **Count rate drops**

1. The PMT gain is set *too high*.

In this case the lighter and darker bar of the measured timepoint of the countrate representation will drop close to zero.

2. The USB port of the computer, to which the USB cable is plugged into, does not support the specification 3.0.

In this case the bars will differ for the measured timepoint.

- **Noise peak within lifetime histogram**

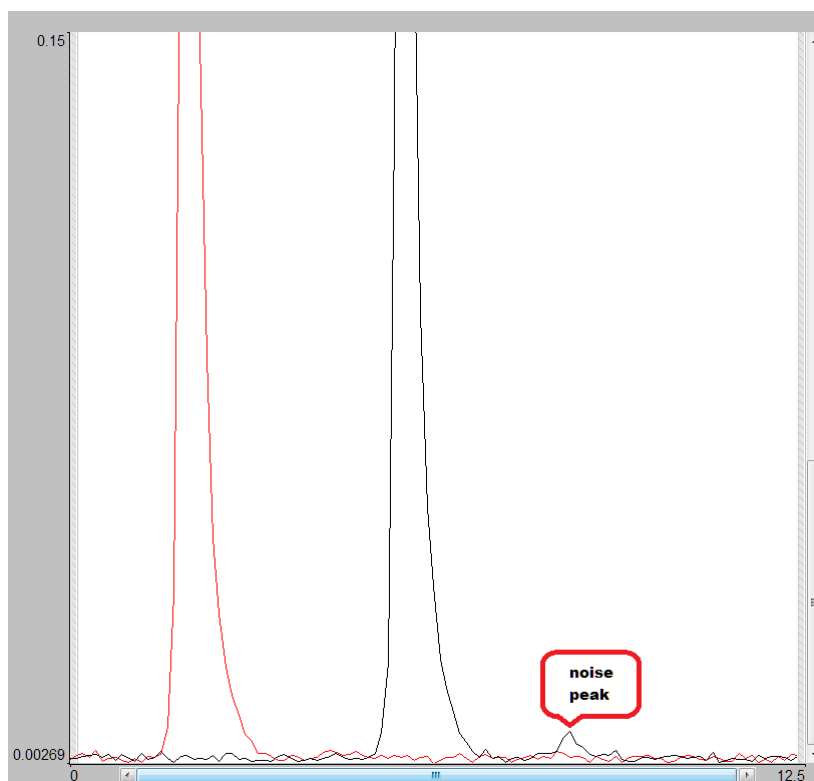


Figure 83: TCSPC - laser ref. freq. trigger cable crosstalk

The red lifetime period is **OK**. The black lifetime period has the **noise peak**.

This is caused by the lifetime peak being too close to the start trigger in conjunction with the length(impedance) of the laser reference frequency trigger cable. This can be circumvented by switching the laser reference frequency trigger cable with one of another length. The user will have to adjust the PMT delays afterwards.

## 9.16 Trigger Sequencer

The Trigger Sequencer driver applies a user-defined sequence of pulses and power thereof to connected devices. There are also two trigger input ports attached to the hardware which the sequence can process.

### 9.16.1 Life Dialog

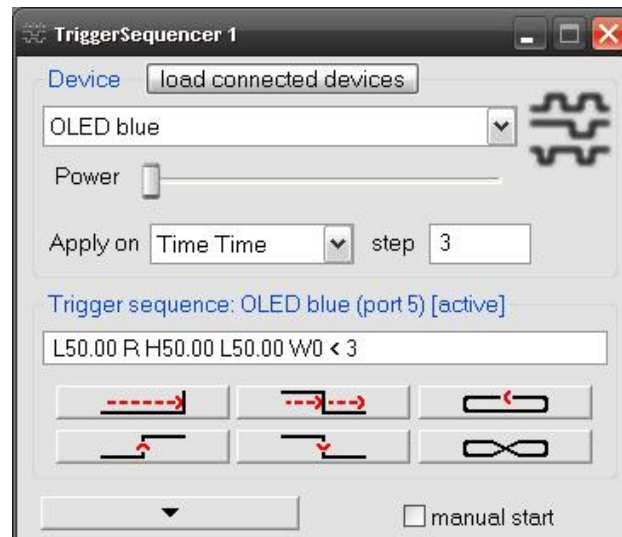








Figure 84: Trigger Sequencer Life Dialog

With the Life Dialog the user configures the devices. The power which is applied to the device and on which axis and step the sequence is executed. The button  scans connected hardware and resets the devices of the document to the newly found ones. On the lower half the user builds the sequence.

The following functions are available:

-  Delay [L50.00]: low level 50 ms
-  Pulse [H50.00 L50.00]: high level 50 ms followed by low level 50 ms
-  Loop [R < 1]: Repeat functions inbetween 'R' and '<' once (On the figure above the functions are repeated 3 times.)
-  Wait for trigger rising: [W0] at port 1 or 2; [W1] at port 1; [W2] at port 2;
-  Wait for trigger falling : [w0] at port 1 or 2; [w1] at port 1; [w2] at port 2;
-  Continuous [:]: Run sequence from start to this token endlessly.

The values above are defaults which should be edited to the users needs.

*A yellow background color of the sequence edit field indicates an error! Doubleclick on the trigger sequencer image at the top-right highlights the bad token.*

At the bottom of the dialog there is the button  to show all devices and their configuration. The check box  manual start starts and stops the trigger sequences for testing purposes.



## Use of keywords in a sequence



Figure 85: Keywords

Keywords can be attached to low and high level tokens of a sequence with a description and range parameter. With that a corresponding slider is added to the Life Dialog to set the value of its token in the sequence.

As the figure shows, multiple tokens of a sequence can respond to the same keyword. With a negative range, the value of that one token is reversed to the thumb of the slider.

## Special keywords in a sequence

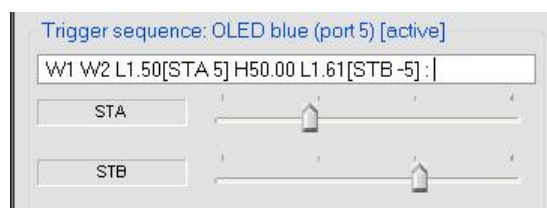


Figure 86: Special keywords

Special Keywords update the values of the attached tokens at the start of a measurement and on internal update routines. *It has to be taken care about the range. The range parameter cuts off higher values!*

These are the special keywords:

- STA: read frame time of treatment scanner 1
- STB: read treatment time of treatment scanner 1
- STC: read frame time of treatment scanner 2
- STD: read treatment time of treatment scanner 2

## 9.16.2 Hardware Dialog

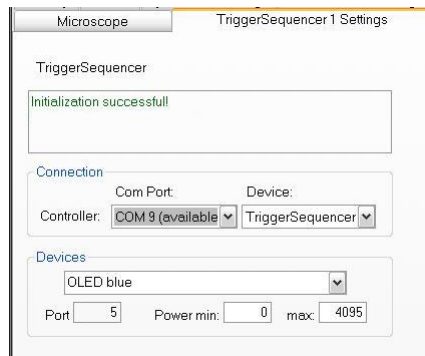


Figure 87: Trigger Sequencer Hardware Dialog

The Trigger Sequencer device connects to the hardware via serial line.

The dialog shows which devices are connected to what ports of the hardware. It is also possible to set the range of motor steps for the minimum and maximum power values needed for non-linear devices, for example.

### 9.16.3 Examples

Now follow two simplified examples of use-cases for the trigger sequencer. The first example shuts the PMTs while the Andor Mosaic treats a sample with a LED. The second example shuts the PMTs while a scanner treats a sample.

#### Example 1: Andor Mosaic treatment with a LED

Version 1:

In this example the LED waits for the shutter to be closed, then treats. Therefore two ports of the Trigger Sequencer must be configured. One for the LED and one for the shutter. The port for the shutter, in this example port 4, must be set in the life dialog to axis *Time Time* and step 10. The port for the LED, in this example port 1, must be set to axis *Start* and step 1. This port setup enables dynamic trigger handling; usually starting the sequence with 'W#' or 'w#' to wait for rising or falling input signal to actually start the sequence.

The sequence for the shutter on port 4 is *L15 H45 L0.05* which waits for 15 ms, then shuts the PMTs for the treatment and finally opens again. At the very moment the PMTs shut the LED port gets triggered and its sequence *W1 L4 H0.05 w1 H4 L0.05*, which waited for the shutter port signal by rising edge, starts. It waits for the shutter to actually close by 4 ms, then starts the treatment until the shutter port signals to open the shutter by falling edge. The LED still treats the 4 ms which the shutter actually needs to open again.

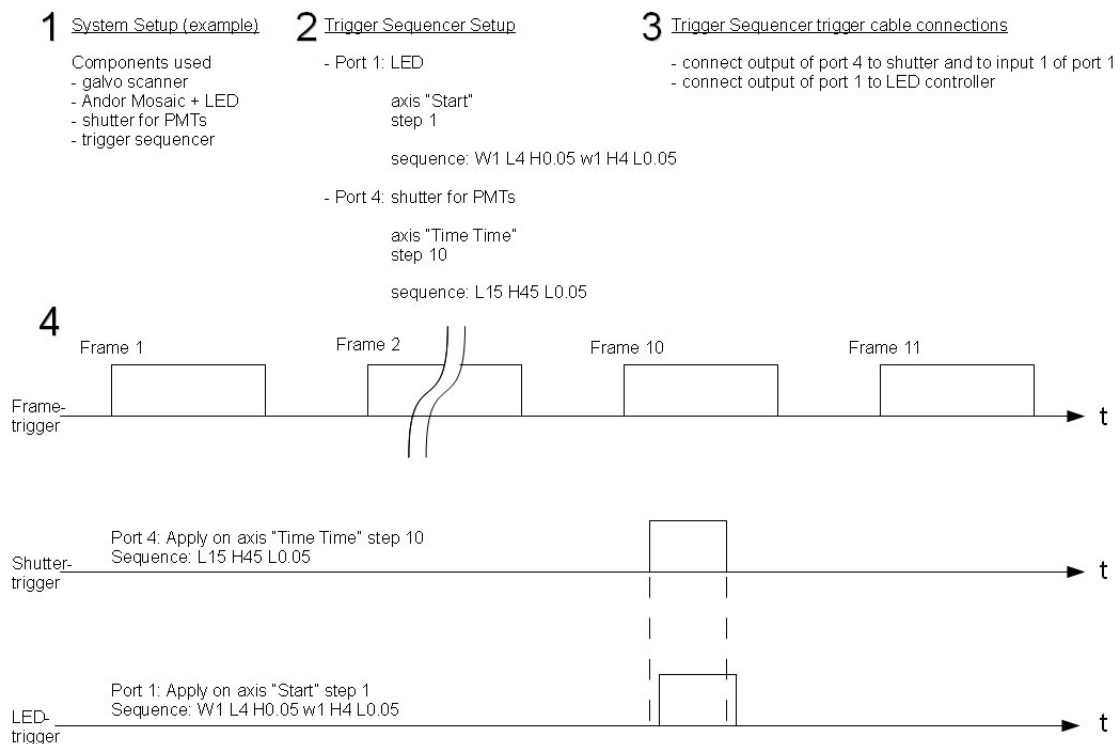


Figure 88: Trigger Sequencer Example 1 Version 1

Version 2:

In this version the shutter port needs to be tweaked for a *resonant scanner*. Its axis must be set to *Start* and step to 1. The sequence should look like *R W1 < 10 L15 H45 L0.05*, which waits for 10 frame triggers to pass before the actual sequence starts.

*Note: A resonant scanner requires every port to have its axis set to Start and step to 1.*

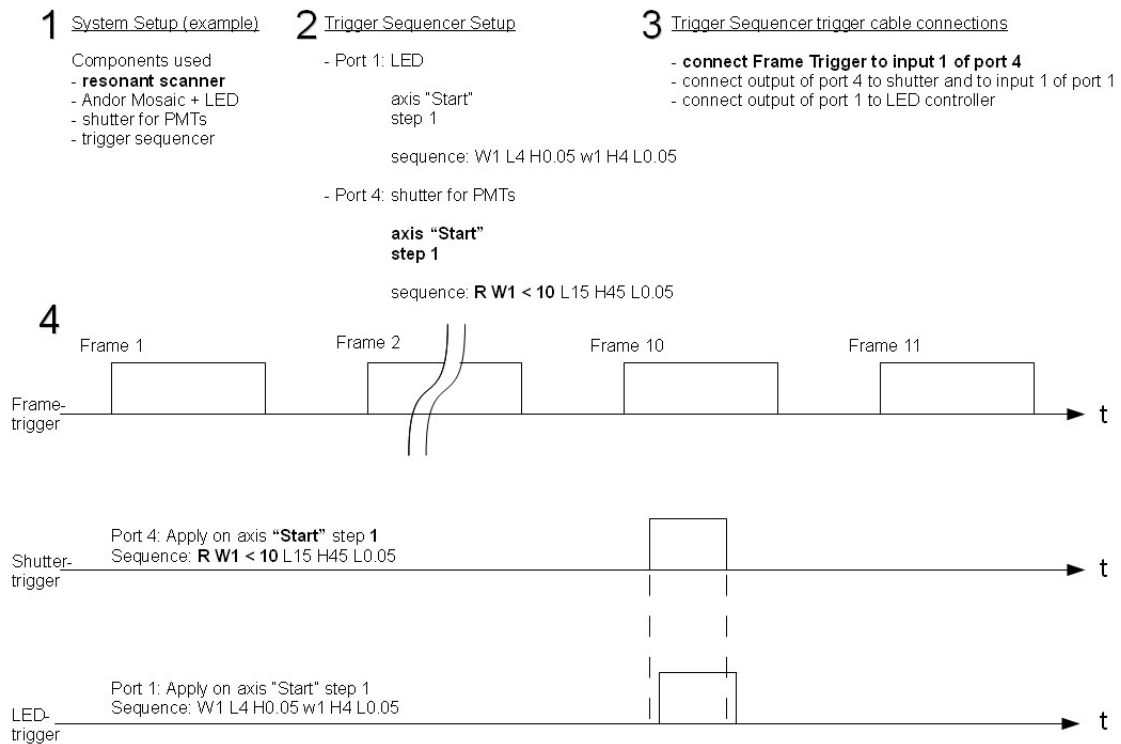


Figure 89: Trigger Sequencer Example 1 Version 2

Version 3:

This version utilizes keywords to delay the shutter and LED with the help of sliders in the life dialog and special keywords to read the treatment time of the scanner for the correct treatment times. The delay of the shutter enables the treatment to happen at a specific time of or position in the frame. The LED delay sets the time, which the LED allows the shutter to actually close before the treatment starts, so that you can start the treatment before the shutter has completely closed.

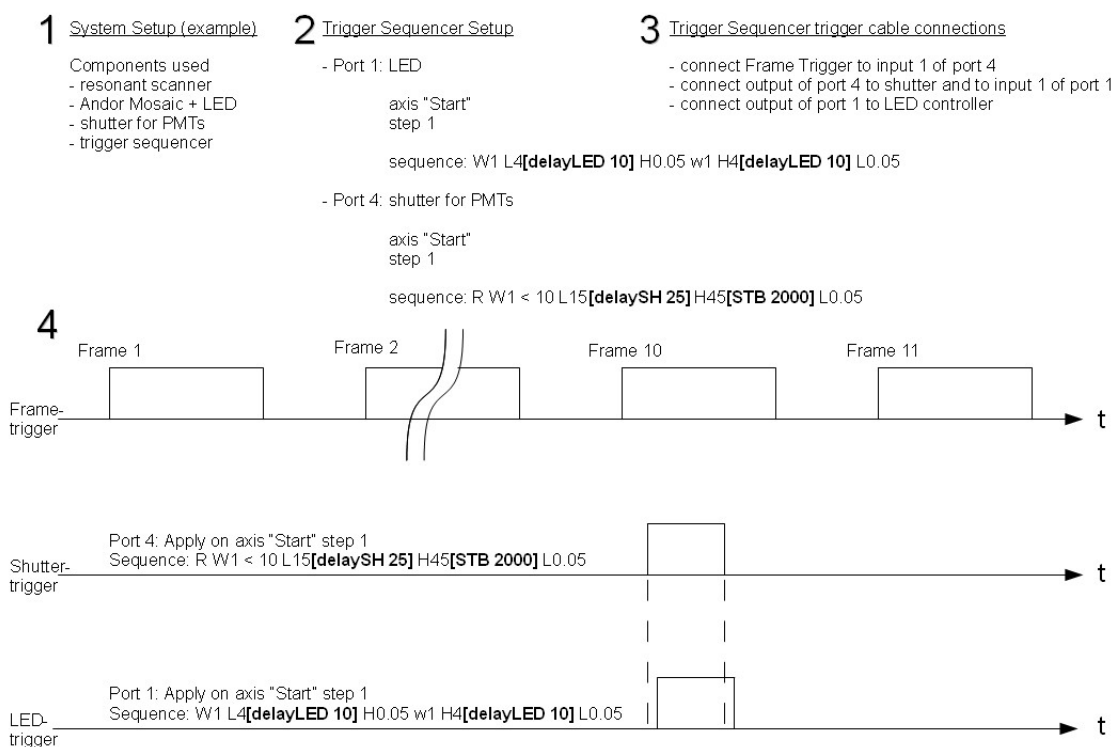


Figure 90: Trigger Sequencer Example 1 Version 3

Example 2: scanner treatment

## Version 1:

With this example the scanners treat a sample. Only one port is required for the shutter. Its axis must be again set to *Start* and step to 1. The sequence *W2 W1 H45 [STB 2000] L0.05* is a bit longer. It waits for the pre-treatment signal of the scanner sync 7 port, then the actual treatment signal of the scanner sync 2 port. After that the sequence starts to treat.

The former extra delay for the shutter must now be configured in the menubar

— Hardware > Configure > Scanner > Frap Delays  
————— > second editbox: <shutter closing time>

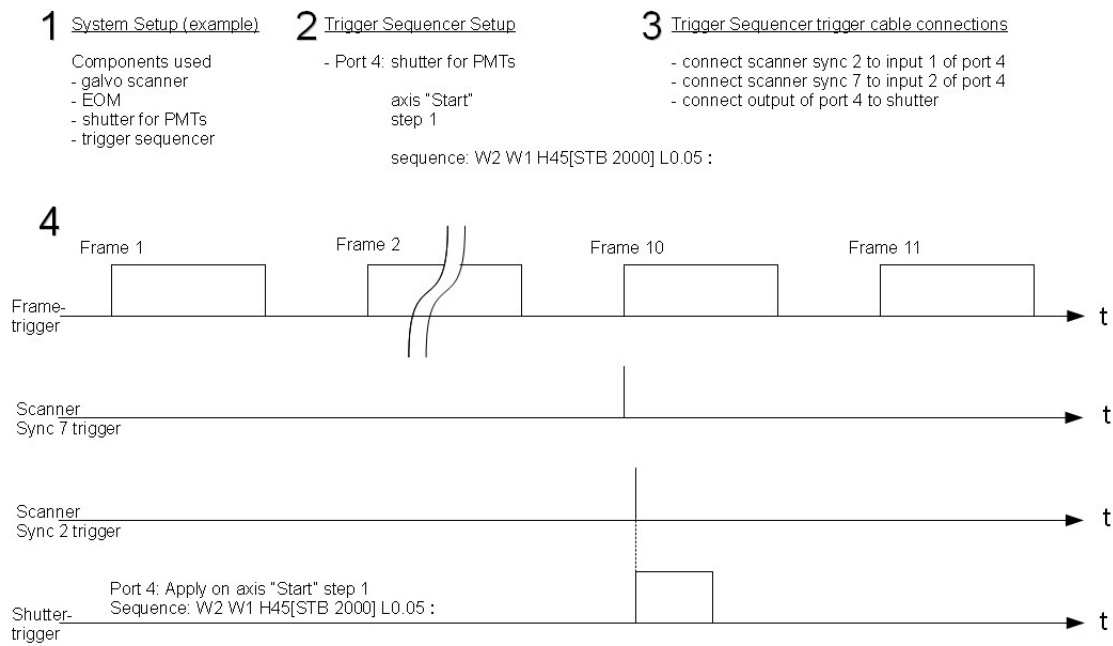


Figure 91: Trigger Sequencer Example 2 Version 1

## Version 2:

With this version two scanner do treatments. Each has an external pre-treatment signal connected, which may very well come from the sample or from a custom setup of the user. The output of the two used Trigger sequencer ports need to be coupled by an electronic logic gate (type OR) to keep the shutter closed for both treatments, assumed that the shutter has only one input port.

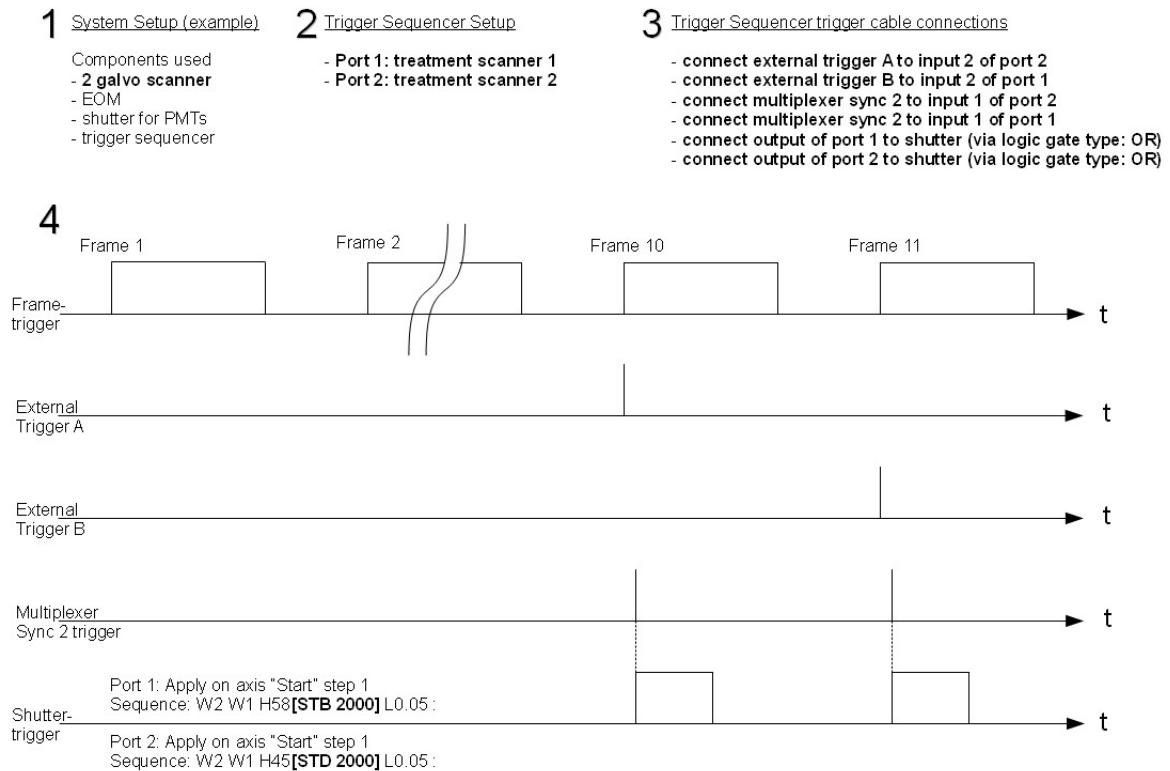


Figure 92: Trigger Sequencer Example 2 Version 2

## 9.17 Andor Mosaic

The *Andor mosaic* device allows to illuminate user defined regions of interest in the specimen simultaneously. Three different LEDs can be used for excitation. To expose the correct area with the correct LED the mosaic device has to be linked with the *Trigger Sequencer* device.

### 9.17.1 Life Dialog

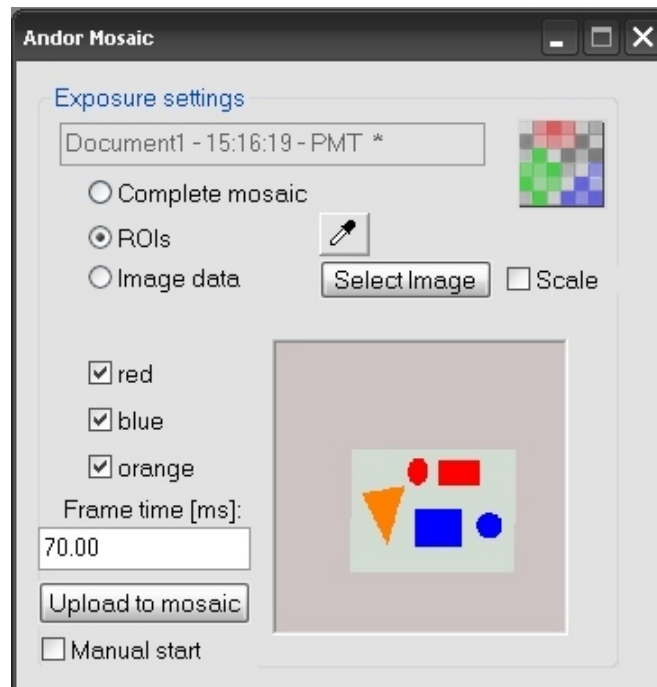




Figure 93: Andor mosaic life dialog

In the life dialog the current exposure area is shown. You can decide which area of the *mosaic* should illuminate:

- Complete mosaic : The complete *mosaic* area is activated.
- ROIs : ROIs in the image stack define the illumination area.
- Image data : Image is used as illumination mask. Choose image file by clicking 'Select image'.
- A sequence of different images can be uploaded via the [Python](#) interface.

When *ROIs* is selected use pipet button  to pick a source window for the ROI data. This adds three different ROIs to that window: So, for each LED the illumination area can be defined separately. For each LED multiple ROIs can be selected.

After defining the *mosaic* exposure area, the settings must be uploaded to the *mosaic* device by clicking .

Every time the ROIs are changed or another image data is selected, the new exposure data must be uploaded manually.

The *mosaic* can run in *internal* and *external trigger mode*.



- **Internal trigger:** the trigger is generated by the *mosaic* device. When running a measurement, the *mosaic* is started automatically and starts illumination immediately. If a sequence is uploaded, the different frames change automatically. The exposure time for a frame can be set in the life dialog ('Frame time' setting).
- **External trigger:** the *mosaic* waits for external trigger input. The external trigger is used for exposure time and for changing the frames of a sequence:
  - The *mosaic* exposures as long as it gets '*high*' signal input by the trigger.
  - Every time the trigger changes from low to high the next frame of the sequence is displayed.

The trigger mode can be set in the *hardware settings*.

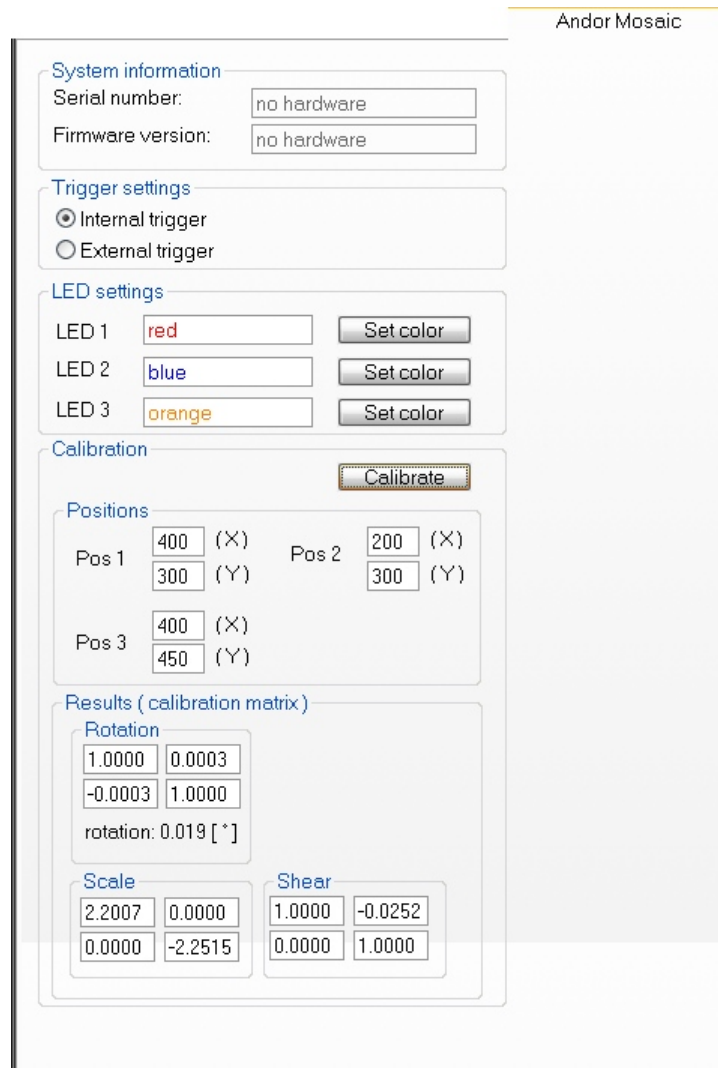
The *mosaic* runs in ***continuous sequence mode***, i.e. when the last frame of a sequence was shown it starts again with the first one.

The three different ROIs are uploaded as a sequence, so the frames can be changed by trigger. When *internal trigger* mode is activated, the frames are changed automatically.

When running a measurement the *mosaic* exposure is started automatically. In *internal trigger* mode the *mosaic* starts immediately, in *external mode* it waits for trigger input.

In *video mode* or when no measurement is running, the *mosaic* can be started by clicking '*Manual start*'. In *external trigger* mode '*Manual start*' also waits for the trigger signal.

## 9.17.2 Hardware Dialog



Andor Mosaic

System information  
Serial number: no hardware  
Firmware version: no hardware

Trigger settings  
 Internal trigger  
 External trigger

LED settings  
LED 1: red [Set color]  
LED 2: blue [Set color]  
LED 3: orange [Set color]

Calibration  
[Calibrate]

Positions  
Pos 1: 400 (X), 300 (Y)  
Pos 2: 200 (X), 300 (Y)  
Pos 3: 400 (X), 450 (Y)

Results (calibration matrix)  
Rotation  
1.0000 0.0003  
-0.0003 1.0000  
rotation: 0.019 [°]  
Scale  
2.2007 0.0000  
0.0000 -2.2515  
Shear  
1.0000 -0.0252  
0.0000 1.0000

Figure 94: Andor mosaic hardware dialog

The Trigger settings are done in the hardware dialog, *internal* and *external* trigger mode can be selected.

Name and color for the different LEDs can be specified in the section '*LED settings*'. By clicking '*Set color*' a color for the LED can be defined. Name and color are shown in the life dialog.

Calibrating the mosaic to the camera can be done in the section '*Calibration*'.

The calibration positions only have to be changed, if the field of view of the camera does not overlap with the *mosaic* exposure area. By clicking '*Calibrate*' the *mosaic* calibration is done automatically. The calibration results are stored in the file '*calibration.ini*' that is located in the *config* directory of *ImInspector*.

The calibration results are also shown in the dialog: rotation, scale and shear of camera to the *mosaic* are displayed.

When calibrating the system, a camera instrument mode must be selected. Make sure to set LEDs to internal trigger mode.

Calibration should be done each time when the objective was changed or the camera was moved. After calibrating the camera to the *mosaic*, the camera should also be calibrated against the scanner device. This can be done in the hardware settings of the scanner.

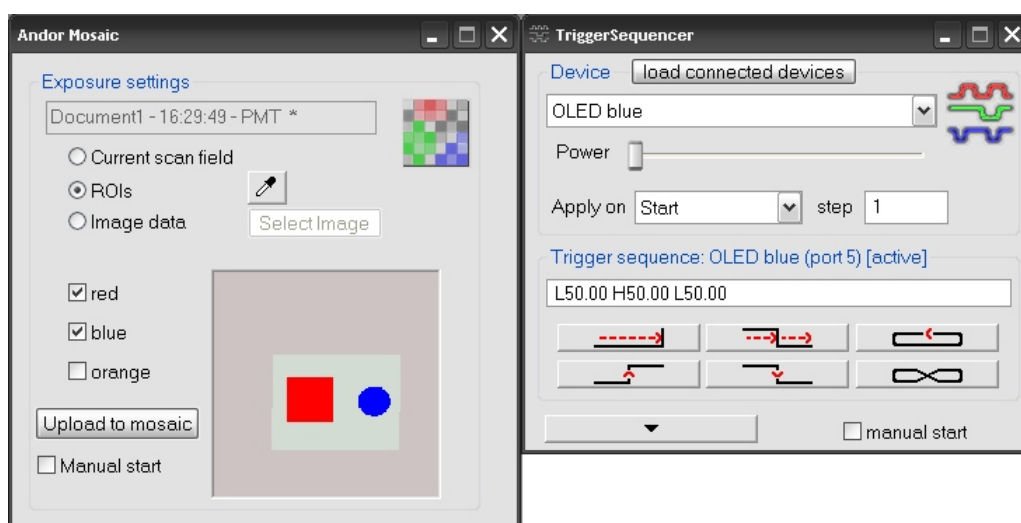
### 9.17.3 Trigger Sequencer settings

To illuminate the correct exposure area at the correct time the *mosaic* device must be synced with the **Trigger Sequencer** device. The *mosaic* device and the LED controllers have to be set to 'external trigger' mode.

The trigger sequencer permits different output lines. Four of these lines must be used when running the mosaic with different LEDs: For each LED and for the *mosaic* device itself a single output sequence must be configured.

When different LEDs are selected in the *mosaic*, the trigger sequencer has to control which LED actually is used.

**The *mosaic* device defines the different areas but cannot control the LED that is used: The trigger sequencer must be used to activate the correct LED at the correct time. Each LED and the *mosaic* itself need a sequence-setup in the trigger sequencer.**



Example:

Figure 95: Example, Syncing *mosaic* with trigger sequencer

Two different ROIs are selected: the red square and the blue circle. These areas are defined by the user.

The illumination should start at the beginning of the measurement. Both LEDs should illuminate the sample for 50 ms. Since two LEDs and the *mosaic* are used three different sequences must be defined in the trigger sequencer.

1. Select trigger output that is connected to the red LED, e.g. 'trigger i' or 'OLED red'. The sequence can be set to **H50 L50 L50** this switches on the LED for 50 ms, then it is switched off.
2. Select trigger output for the blue LED; e.g 'trigger ii' or 'OLED blue'. The sequence can be set to **L50 H50 L50** this switches on the LED for 50 ms after 50 ms wait time, then it is switched off again.
3. Select trigger output for the *mosaic*; e.g. 'trigger iii'. The *mosaic* illuminates as long as it gets 'high' trigger input. To change the exposure area it must receive a 'low' input interrupt. The sequence can be set to **H49.9 L0.1 H49.9 L0.1** this switches on the exposure area of the red square for 49.9 ms. Then it is switched off for 0.1 ms. With the next high trigger the exposure area is changed to the blue circle. This area is open for 49.9 ms. At the same time the blue LED is switched on. So exposure area and LED are synchronized.

To start the illumination at the beginning of the measurement '*apply on start*'

Apply on  step  in the trigger sequencer must be selected. This must be done for all trigger sequences that are used.

#### 9.17.4 Python

The *mosaic* device can be configured via the python interface of *Inspector*. It can be used in both: [Python Virtual Device](#) and [Python Script Editor](#).

*Inspector* provides these functions to set up the *mosaic* device:

- - `Inspector/IS.MosaicAddFrame()`
- - `Inspector/IS.MosaicClearFrames`
- - `Inspector/IS.MosaicSetExposureTime`

In the Python script editor *IS*. methods are used. In the virtual device *Inspector*. methods are used.

`Inspector/IS.MosaicAddFrame()` Sets exposure area for *mosaic* device manually. By running this function a new frame is uploaded to the *mosaic* frame memory. Up to 138 different frames can be uploaded.

*Syntax:* `Inspector/IS.MosaicAddFrame(data,scale)`

*Parameter:*

1. array data: exposure mask for the *mosaic*
2. bool scale: True: mask is scaled to current image/(scanner) size, False: mask is scaled to *mosaic mirror* dimensions

**NOTE:** data must be numpy array type boolean: False/0 *mosaic* is not exposed, True/1 *mosaic* is exposed.

```
» data = np.array(np.zeros((600,800)), dtype = bool)
» x,y = data.shape
```

```
»data[150:x,100:150] = 1
»data[350:x,200:250] = 1
»
» IS.MosaicAddFrame(data,1)
```

**Inspector/IS.MosaicClearFrames** Resets frame memory of *mosaic*.

*Syntax:* Inspector/IS.MosaicClearFrames()  
» **IS.MosaicClearFrames()**

**Inspector/IS.MosaicSetExposureTime** Sets exposure time for the *mosaic* when running in internal mode.

*Syntax:* Inspector/IS.MosaicSetExposureTime(float exptime)

*Parameter:*

- float exptime: exposure time for the *mosaic* (internal trigger mode only!)

```
» IS.MosaicSetExposureTime(33.33)
```

## 9.18 LED

This Inspector device can be used to control the *XCite XLED 1*.

### 9.18.1 Life Dialog

In the life dialog each LED can be switched on and off separately. The power for each LED can be adjusted with the sliders. The *XLED 1* allows values between 5 % and 100 %.

All LEDs can be activated and deactivated at once by using the *All on* and *All off* buttons at the top the dialog.

After starting Inspector the remote control of the device is locked. To unlock the control uncheck *lock remote*.

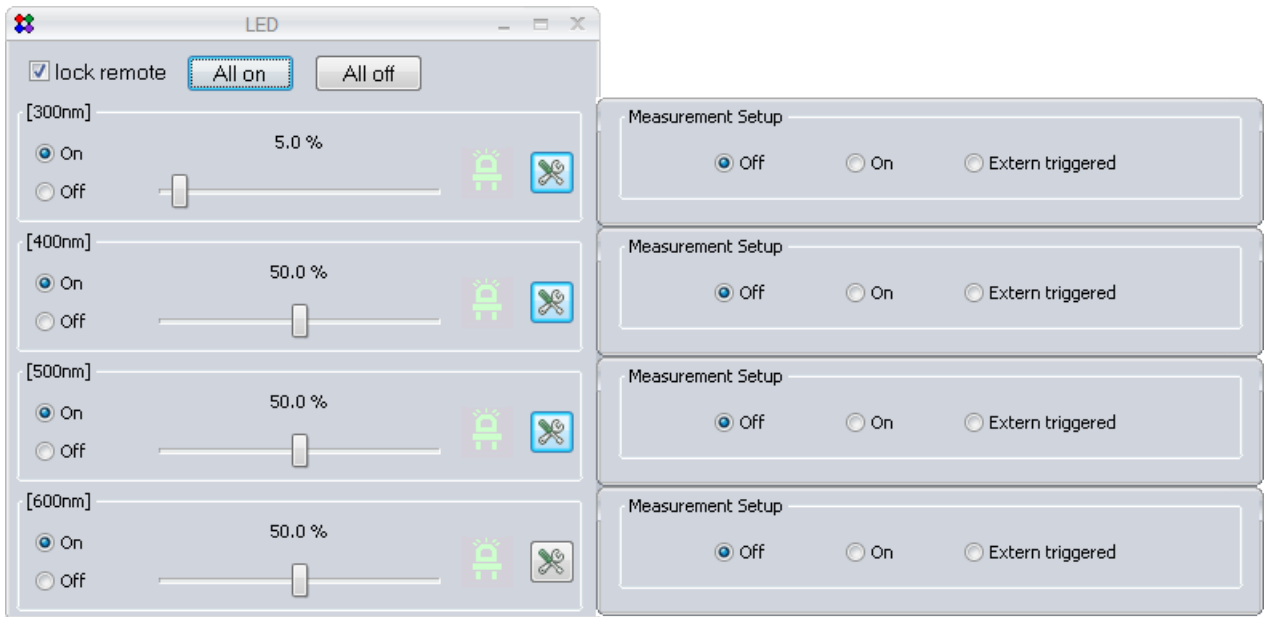



Figure 96: LED life dialog

As default setting the device automatically shuts off all LEDs when starting a measurement.

For each LED can be set up how to behave during a measurement:

- **Off** LED off during a measurement (default)
- **On** LED active during a measurement (wide field imaging)
- **Extern triggered** LED active on trigger signal (used for treatments).

To set up the measurement behaviour click on the settings button .

If the setting button is not shown in the life dialog, the function to use the LEDs during a measurement is disabled. It can be enabled in the hardware dialog.

For treatments an external trigger signal must be used. To generate this the [Trigger Sequencer](#) can be used. With the trigger sequencer can be specified which, when and how the LEDs are activated.

### 9.18.2 Hardware Dialog

In the hardware dialog the com port of the device must be selected.

When the LEDs should be used during a measurement, this function must be activated by checking '*Allow active LEDs during measurement*'. If it is not selected, the LEDs are switched off at the start of a measurement.

### 9.18.3 Python interface

The LED device can also be accessed via the [Python Script Editor](#) and [Python Virtual Device](#). The *ImInspector lvbt* python module provides a LED class (*lvbt.led*). Each LED can

be accessed independently. (e.g. `led1 = lvbt.led(1)`) the `lvbt.led` class includes the following methods:

- `led.id()` Returns ID of LED.
- `led.numLED()` Returns number of LEDs.
- `led.off()` Turns off LED.
- `led.on()` Turns on LED.
- `led.power()` Returns power of LED.
- `led.power( float power)` Sets power of LED.

## 9.19 Adaptive Optics

The *adaptive optics* module allows to correct tissue induced aberrations with a deformable mirror.

The optimization bases on deforming the mirror according to different *Zernike polynomials*. Further information about Zernike polynomials see:

[http://en.wikipedia.org/wiki/Zernike\\_polynomials](http://en.wikipedia.org/wiki/Zernike_polynomials).

The following polynomials can be used in *ImSpector*:

Number	Name	Formula
1	Piston	1
2	Tilt X	$\sqrt{4} \cdot \rho \cdot \cos(\phi)$
3	Tilt Y	$\sqrt{4} \cdot \rho \cdot \sin(\phi)$
4	Defocus	$\sqrt{3} \cdot (2 \cdot \rho^2 - 1)$
5	Astigmatism X	$\sqrt{6} \cdot \rho^2 \cdot \sin(2 \cdot \phi)$
6	Astigmatism Y	$\sqrt{6} \cdot \rho^2 \cdot \cos(2 \cdot \phi)$
7	Coma X	$\sqrt{8} \cdot (3 \cdot \rho^3 - 2 \cdot \rho) \cdot \sin(\phi)$
8	Coma Y	$\sqrt{8} \cdot (3 \cdot \rho^3 - 2 \cdot \rho) \cdot \cos(\phi)$
9	Trefoil Y	$\sqrt{8} \cdot \rho^3 \cdot \sin(3 \cdot \phi)$
10	Trefoil X	$\sqrt{8} \cdot \rho^3 \cdot \cos(3 \cdot \phi)$
11	Spherical Aberration	$\sqrt{5} \cdot (6 \cdot \rho^4 - 6 \cdot \rho^2 + 1)$
12	Secondary Astigmatism X	$\sqrt{10} \cdot (4 \cdot \rho^4 - 3 \cdot \rho^2) \cdot \cos(2 \cdot \phi)$
13	Secondary Astigmatism Y	$\sqrt{10} \cdot (4 \cdot \rho^4 - 3 \cdot \rho^2) \cdot \sin(2 \cdot \phi)$
14	Tetrafoil X	$\sqrt{10} \cdot \rho^4 \cdot \cos(4 \cdot \phi)$
15	Tetrafoil Y	$\sqrt{10} \cdot \rho^4 \cdot \sin(4 \cdot \phi)$
16	Secondary Coma X	$\sqrt{12} \cdot (10 \cdot \rho^5 - 12 \cdot \rho^3 + 3 \cdot \rho) \cdot \cos(\phi)$
17	Secondary Coma Y	$\sqrt{12} \cdot (10 \cdot \rho^5 - 12 \cdot \rho^3 + 3 \cdot \rho) \cdot \sin(\phi)$
18	Secondary Trefoil X	$\sqrt{12} \cdot (5 \cdot \rho^5 - 4 \cdot \rho^3) \cdot \cos(3 \cdot \phi)$
19	Secondary Trefoil Y	$\sqrt{12} \cdot (5 \cdot \rho^5 - 4 \cdot \rho^3) \cdot \sin(3 \cdot \phi)$
20	Pentafoil X	$\sqrt{(12)} \cdot \rho^5 \cdot \cos(5 \cdot \phi)$
21	Pentafoil Y	$\sqrt{(12)} \cdot \rho^5 \cdot \sin(5 \cdot \phi)$
22	Tertiary Speherical Aberration	$\sqrt{7} \cdot (20 \cdot \rho^6 - 30 \cdot \rho^4 + 12 \cdot \rho^2 - 1)$
23	Tertiary Astigmatism Y	$\sqrt{14} \cdot (15 \cdot \rho^6 - 20 \cdot \rho^4 + 6 \cdot \rho^2) \cdot \sin(2 \cdot \phi)$
24	Tertiary Astigmatism X	$\sqrt{14} \cdot (15 \cdot \rho^6 - 20 \cdot \rho^4 + 6 \cdot \rho^2) \cdot \cos(2 \cdot \phi)$
25	Secondary Tetrafoil Y	$\sqrt{14} \cdot (6 \cdot \rho^6 - 5 \cdot \rho^4) \cdot \sin(4 \cdot \phi)$
26	Secondary Tetrafoil X	$\sqrt{14} \cdot (6 \cdot \rho^6 - 5 \cdot \rho^4) \cdot \cos(4 \cdot \phi)$

Figure 97: Zernike polynomials used in ImSpector



## 9.19.1 Hardware Dialog



Figure 98: Adaptive Optics Hardware Settings.

Following settings can be done in the hardware dialog:

1. Status information: Device initialization status.
2. Polynomial Setting: Select up to 8 different Zernike polynomials. The specified polynomials can be accessed via slider in the life dialog. For each slider a name/id can be specified here. This name is shown in the life dialog to identify the slider.
3. Radius: Influence-radius of polynomial, can be set from 0.7 to 1.0. Smaller values only change the inner parts of the deformable mirror, value 1 adjusts the complete mirror.
4. Slider range: Factor that is multiplied to Zernike polynomials (Polynomials are stored normalized).
5. Z-Device: Table/Stepper device that is currently used.
6. Restore Factory Settings: If default values are misaligned, the correct factory settings can be restored by clicking this button.

### 9.19.2 Life Dialog

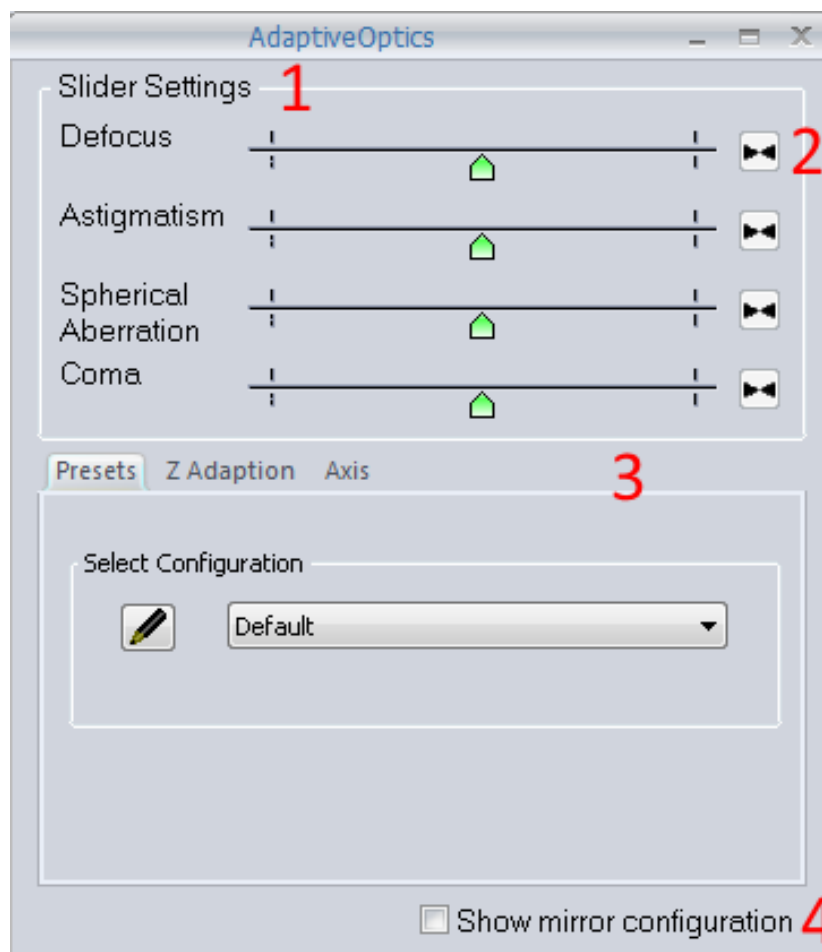




Figure 99: Adaptive Optics Life Dialog.

In the life dialog following settings can be done:

1. Zernike-Slider: slider that are specified in the hardware dialog are shown here. When moving a slider the mirror configuration is directly changed according to the Zernike-values it represents, see [Effect Of Astigmatism Slider](#).

2. Reset-Button: next to each slider a small button is shown . Click on it to reset the slider to its origin position.
3. Three different tabs are shown here:
  - [Presets Tab](#)
  - [Z Adaptation Tab](#)
  - [Axis Tab](#)
4. Mirror Configuration: When checking this box the [Mirror Configuration Dialog](#) opens.,

**Presets Tab** Here different preset configurations can be selected. To store modifications, adding or deleting configurations open the [Voltage Configuration Dialog](#) by clicking this  button.

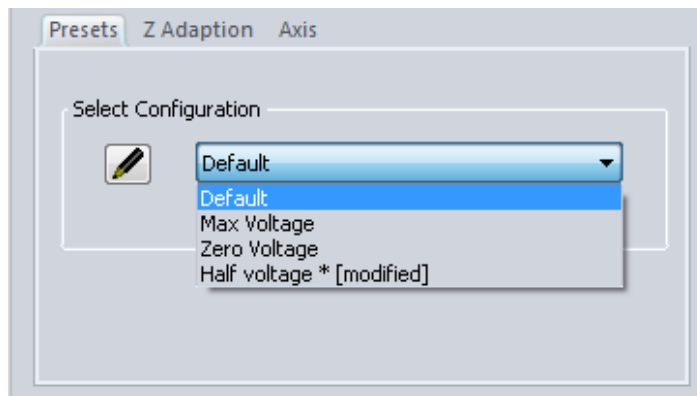


Figure 100: Presets Tab.

**Z Adaptation Tab** For acquiring Z-stacks the mirror configuration can be adapted automatically according to the different z positions.

Following settings can be done here:

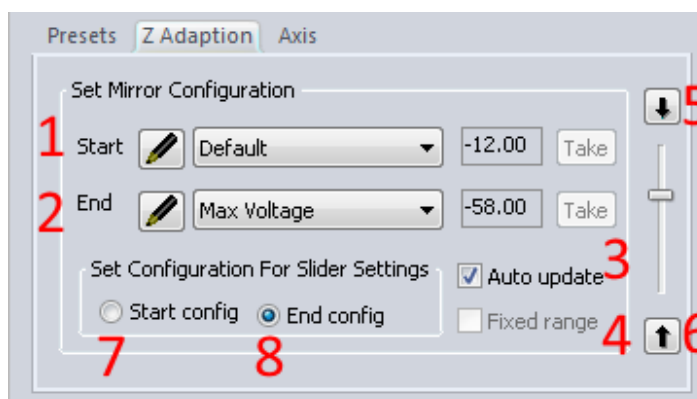



Figure 101: Z Adaptation Tab.

1. Select **start configuration**: This configuration is used at the begin of the Z-stack. You can use a set from list. This set can be modified with the sliders above. Make sure 'Start config' (7) is selected when moving the sliders. To store modifications permanently use the [Voltage Configuration Dialog](#) by clicking this  button.

2. Select **end configuration**: This configuration is used at the end of the Z-stack. Modifying this set can be done independently from the start configuration with the sliders. Make sure the radio-button 'End config' (8) is selected when moving the sliders. Storing the modified configurations can also be done with the [Voltage Configuration Dialog](#). The voltage settings between start and end configuration are adapted linearly according to the current Z position.
3. Auto Update: When 'Auto Update' is checked the z-range defined in the Z-stepper dialog is used automatically as start and end position for this dialog, too. The z-Stepper that is connected to the adaptive optics module has to be selected in the [Hardware Dialog](#).
4. Fixed range: When checking 'Fixed range' the currently used range between start and end position is kept permanently. When changing the start position, the end position is updated automatically (and vice versa). Using fixed range is only possible if 'auto update' is not used.
5. Extend calculation above start position if this button is clicked, the voltage adaption is not stopped when going above the start position (↑), but is extrapolated linearly, otherwise the adaption is stopped (↓).
6. Extend calculation below end position: if this button is clicked, the voltage adaption is not stopped when going below the end position (↓), but is extrapolated linearly, otherwise the adaption is stopped (↑)

**Axis Tab** The adaptive optics module provides a data axis that can be selected in the [Measurement Wizard](#). With this axis it is possible to change the mirror configuration automatically during a measurement according to a specified Zernike slider.

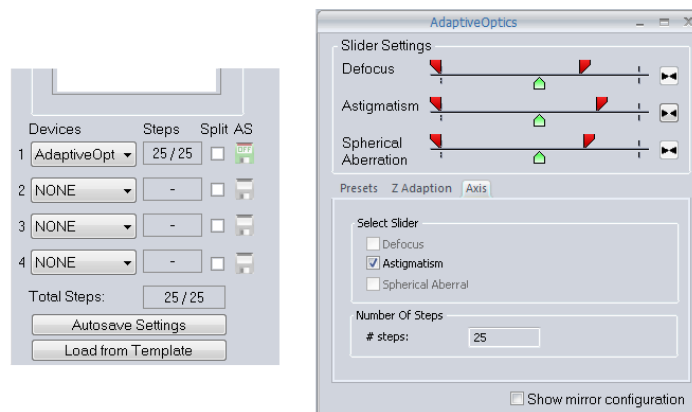


Figure 102: Axis Settings.

These settings can be done with Axis tab:

1. Define position range with the red sliders. These sliders appear when opening the axis tab.
2. Define slider that is used for the data axis by clicking the corresponding checkbox. Only one slider can be selected.
3. Define number of steps of the axis.

**Mirror Configuration Dialog** In the mirror configuration dialog the current voltage settings of the mirror is shown. It has two tabs: The 'Value' tab shows the voltage setting as a number, the 'Positions' tab shows the voltage settings in a graphical representation. If the [Z Adaptation Tab](#) is selected in the life dialog, in the Mirror config dialog can be set up to show the configuration of the start position, end position, or current Z-position.

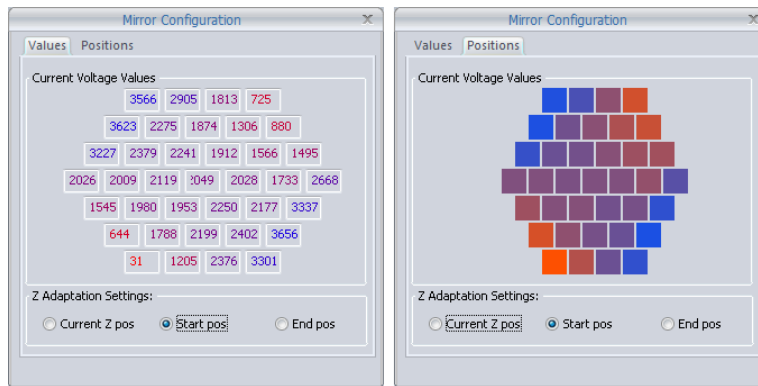


Figure 103: Mirror Configuration.

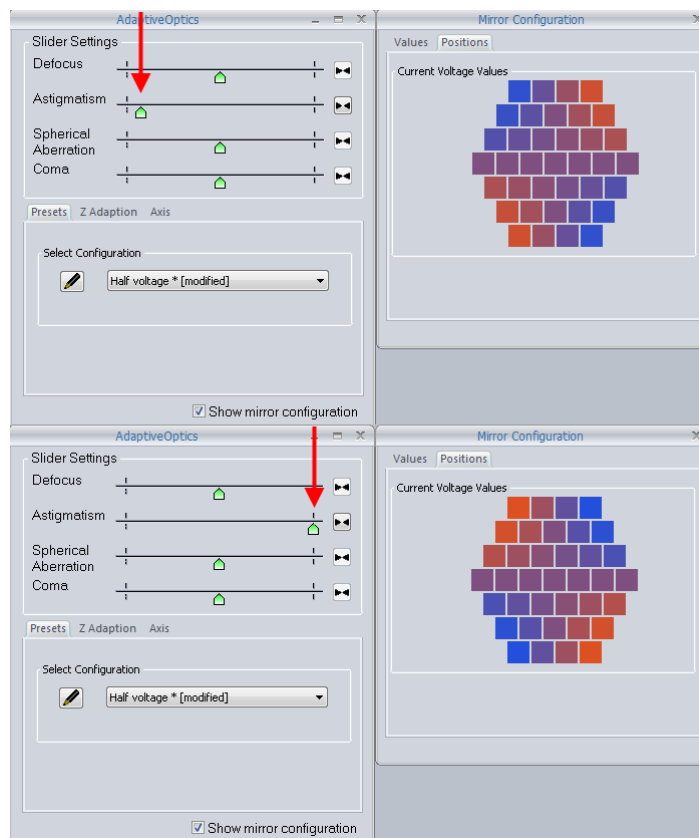


Figure 104: Effect Of Astigmatism Slider.

**Voltage Configuration Dialog** This dialog can be used to store the current mirror configuration permanently or to modify the selected configuration. The default configuration can be set here.

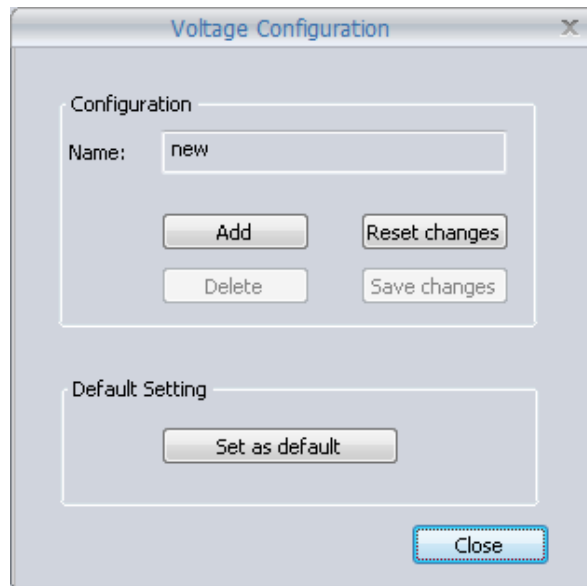


Figure 105: Voltage Configuration Dialog.

- Click  to add the current configuration. Name must be specified above.
- Click  to discard all modifications.
- Click  to remove the currently selected set from the list (not possible for 'Default' set).
- Click  to save modifications of the current set (not possible for 'Default' set).
- Click  to set the current configuration as default.

If accidentally a wrong configuration is stored as default you can reset the default settings in the [Hardware Dialog](#) (restore factory settings).

All voltage sets are stored in an ini file that can be found in the config directory:  
*'Config/adaptive optics/adaptive optics.ini.'*

### 9.19.3 Python Interface

The adaptive optics module can also be controlled via the [Python Virtual Device](#) or the [Python Script Editor](#). Own optimization scripts or data analysis can be done with the python interface. Therefore Inspector provides some methods for accessing the adaptive optics module:

- - `Inspector.AdaptiveOpticsGetSettings ()`
- - `Inspector.AdaptiveOpticsApplySettings ()`
- - `Inspector.AdaptiveOpticsGetMax ()`
- - `Inspector.AdaptiveOpticsGetFactorySettings ()`
- - `Inspector.AdaptiveOpticsSetFactorySettings (voltage)`
- - `Inspector.AdaptiveOpticsGetSliderRange ()`

- - `Inspector.AdaptiveOpticsGetMaxSliderRange()`
- - `Inspector.AdaptiveOpticsSetSlider()`

In the [Python Script Editor](#) 'Inspector:' must be replaced by 'IS.'

Further information can be found with the python info message functions [Inspector.Info\(\)](#) and [IS.Info\(\)](#)

### `Inspector.AdaptiveOpticsGetSettings()`

Syntax: `Inspector.AdaptiveOpticsGetSettings()`

Returns: Pointer to the array with the voltage settings.

`Inspector.AdaptiveOpticsApplySettings()` Uploads the current voltage settings to the adaptive optics device.

Syntax: `Inspector.AdaptiveOpticsApplySettings()`

`Inspector.AdaptiveOpticsApplySettings()` Uploads the current voltage settings to the adaptive optics device.

Syntax: `Inspector.AdaptiveOpticsApplySettings()`

### `Inspector.AdaptiveOpticsGetMax()`

Syntax: `Inspector.AdaptiveOpticsApplySettings()`

Returns: The maximum voltage amplitude of the adaptive optics device (normally this is 4095).

### `Inspector.AdaptiveOpticsGetFactorySettings()`

Syntax: `Inspector.AdaptiveOpticsGetFactorySettings()`

Returns: Voltage array with factory settings.

### `Inspector.AdaptiveOpticsSetFactorySettings(voltage)`

Syntax: `Inspector.AdaptiveOpticsSetFactorySettings(voltage)`

Parameter: vector voltage: vector with voltage values that is stored as factory settings (37 values!)

**Be careful: factory settings will be overwritten permanently!**

### `Inspector.AdaptiveOpticsGetSliderRange()`

Syntax: `Inspector.AdaptiveOpticsGetSliderRange(slider)`

Parameter: int slider: No. of slider (0-7)

Returns: min,max values of specified slider. Range can be set up with red boundaries in the 'axis' tab!

### **Inspector.AdaptiveOpticsGetMaxSliderRange()**

Syntax: **Inspector.AdaptiveOpticsGetMaxSliderRange()**

Returns: Returns: min,max values of Zernike slider that is specified in the hardware settings

### **Inspector.AdaptiveOpticsSetSlider()**

Syntax: **Inspector.AdaptiveOpticsSetSlider(int slider, float position)**

Paramter:   1. int slider: No. of slider (0-7)  
              2. float position: position the slider is set to.



## 9.20 Python Virtual Device

The idea behind the *Python Virtual Device* is to have a python script that behaves like a physical device. The *Virtual Device* gives the user the possibility to write drivers for own devices or to do data analysis during a measurement. The *Virtual Device* can also create own image data and/or register a data axis.

The *Virtual Device* behaves like any other *Inspector* device and can be fully integrated in the measuring process.

The *Virtual Device* supports *Numerical Python (Numpy)* and *Scientific Python (Scipy)*. *Numpy* and *Scipy* are fundamental packages for scientific computing in Python and must be installed manually to get fully use of the *Virtual Device*.

Further information see <http://www.numpy.org/> and <http://www.scipy.org/>.

*Inspector* provides some python methods that can be used for data handling or device communication.

- - `Inspector.InitStack()`
  
- - `Inspector.GetDataPtr()`
  
- - `Inspector.ClearOutput()`
  
- - `Inspector.GetOutput()`
  
- - `Inspector.MoveZ(device, position)`
  
- - `Inspector.GetTablePosition(device)`
  
- - `Inspector.SetTablePosition(device, x position, y position, max movement)`
  
- - `Inspector.TableSetOffsetX(device, offset)`
  
- - `Inspector.TableSetOffsetY(device, offset)`
  
- - `Inspector.TableSetOffsetZ(device, offset)`
  
- - `Inspector.Info()`

- The adaptive optics module can also be controlled via [Python Interface](#)

Further information see [Inspector Methods For Python Device](#).

### 9.20.1 The measurement loop

Since the Virtual Device behaves like a normal *Inspector* device, some methods must be implemented in the script. The following methods are called automatically by the inspector framework when a measurement runs:

OnBegMeas(\*args)

OnEndMeas(\*args)

OnUpdMeas(\*args)

OnBegStep(\*args)

OnUpdStep(\*args)

OnEndStep(\*args)

During a run of a measurement, *Inspector* calls those methods in this order:

#### Start 1. OnBegMeas(\*args)

- Called at the begin of the measurement loop.
- Initializations should be done here.
- If image data is generated, the data stacks must be initialized here.
- If the data axis is used, start position, end position and number of sept can be identified here.

#### Start 2. OnUpdMeas(\*args)

- Called after OnBegMeas().
- Further Initializations can be done here.

#### Loop 1. OnBegStep(\*args)

- First called after OnUpdMeas().
- Begin of 'step-loop' of *Inspector*: Until all steps are processed, this routine is called at the begin of a new step.
- If the data axis is used, the current position can be identified here.

### Loop 2. OnUpdStep(\*args)

- Called after OnBegStep().

### Loop 3. OnEndStep(\*args)

- Called after OnUpdStep.
- All data stacks of a measurement can be accessed here.
- If image data is generated a data pointer to the current slice of the stack is provided here.

### End 1. OnEndMeas(\*args)

- Called at the end of a measurement.
- Deinitializations should be done here.

*All steps must be implemented in the script as python methods  
(see example [9.20.6]).*

The return value of each method must either be '1' which indicates 'OK' or '0' which indicates an **error**.

After an error is returned by a method, *Inspector* stops the complete measurement immediately.

#### 9.20.2 Hardware Dialog

In the hardware dialog the script that is used for the *Virtual Device* must be selected.

#### 9.20.3 Life Dialog

The *Virtual Device* can use a data axis, that can be selected in the *Inspector Measurement Wizard*. The physical sizes of this axis (start and end values) and the number of steps can be specified in the life dialog.

In the script the standard python **print** command can be used. The output is shown in an own window that can be opened in the life dialog. All script errors are displayed in this window, too.

By clicking the button '*Reload script*' the current script that is specified in the hardware dialog is reloaded. After changing the python script in an external editor click this button to update the last changes in *Inspector*.

E.g., if your script implements a z-stage control, you can enter a start position (*From*), a end position (*To*) and the number of steps between.

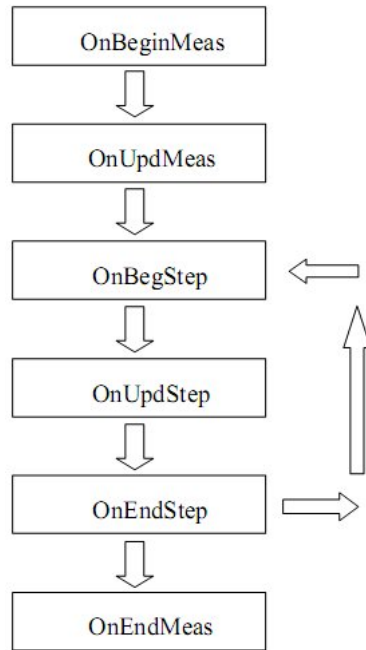


Figure 106: The measurement loop.

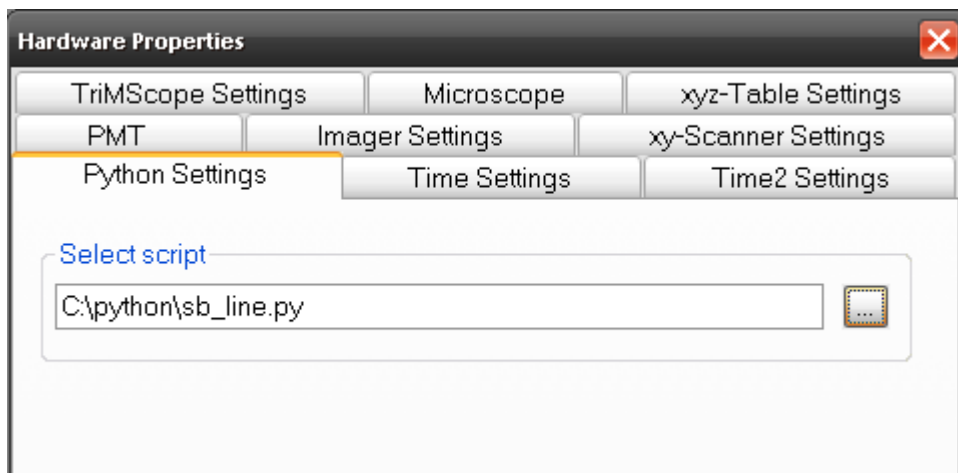


Figure 107: Virtual Device - Hardware Dialog.

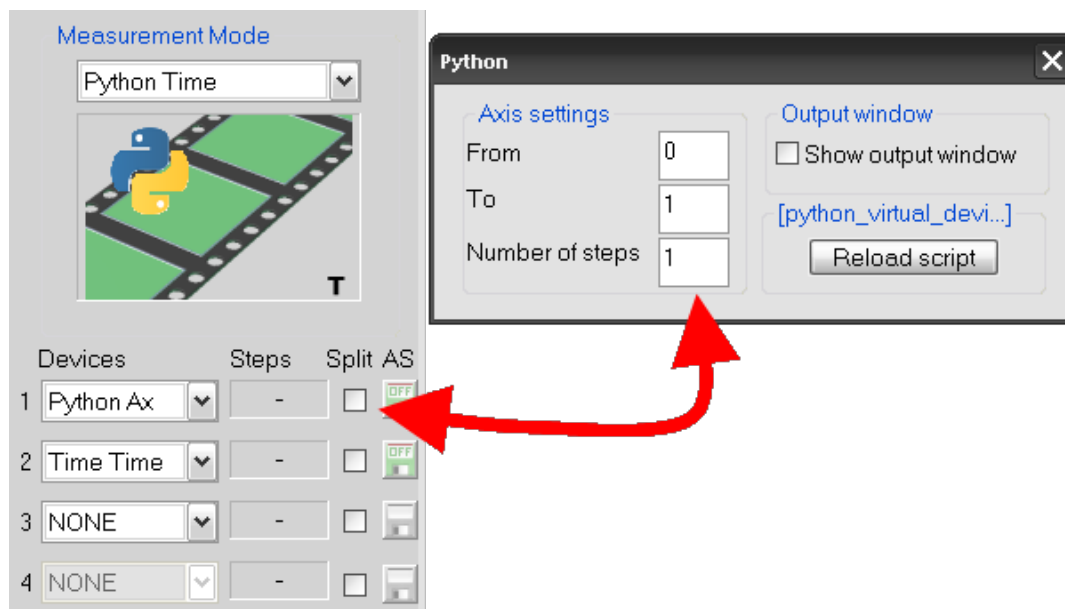


Figure 108: Virtual Device - Life Dialog.

#### 9.20.4 Methods Parameters

The parameters in the life dialog are passed to the python script in *OnBegMeas(\*args)*. The order is: *From*, *To*, *NumberOfSteps*. How the values can be read out by the script is shown in the following here:

```

1 def OnBegMeas(*args):
2
3     ValueFrom = args[0]
4     ValueTo = args[1]
5     NumberOfSteps = args[2]
6     return 1

```

*Inspector* calculates the current position of the device for each step and gives it as a parameter in the *OnBegStep(\*args)* to the script. The current step is also passed to the script. The order is: *currentPosition*, *currentStep*.

```

1 def OnBegStep(*args):
2
3     CurrentPosition=args[0]
4     CurrentStep=args[1]
5     return 1

```

In *OnEndStep()* all data stacks of the current measurement can be accessed. Pointer to the data stacks - among other information - are passed to the script and can directly be transformed into *numpy* arrays. The data stacks can be accessed this way:

```

1 def OnEndStep(*args):
2
3     stacks = args[0]           # different stacks
4     resolutions = args[1]     # pixel sizes for each stack
5     lengths = args[2]        # physical sizes for each stack
6     offsets = args[3]        # offsets for each stack
7
8     AxisStep = args[4]       # boolean if the python axis has moved

```

```

9         # since the last step
10
11     stack_0 = np.array(stacks[0], copy = False)
12     stack_1 = np.array(stacks[1], copy = False)

```

### 9.20.5 Inspector Methods For Python Device

*Inspector* provides some python methods that can be used for data handling or device communication.

To use these methods the *Inspector* package must imported at the beginning of the python script (**import Inspector**)

The following methods are included in that package:

#### **Inspector.InitStack()**

If the *Virtual Device* generates image data, this function must be used to init an empty data stack.

*Syntax:*

**Inspector.InitStack(int size\_x, float length\_x, float offset\_x [,int size\_y, float length\_y, float offset\_y], int size\_z, float length\_z, float offset\_z)**

1D, 2D and 3D stacks can be initialized, for each dimension pixel-size, physical-size must physical-offset must be given.

*parameter: (3,6 or 9 parameter must be given.)*

- size\_x: pixel resolution x
- float length\_x: physical length x
- float offset\_x: physical offset x[
- int size\_y: pixel resolution y
- float length\_y: physical length y
- float offset\_y: physical offset y[
- int size\_z: pixel resolution z
- float length\_z: physical length z
- float offset\_z: physical offset z]]

*Returns: False* if data stack could not be initialized.

```

» size_x = 512
» size_y = 512
»
» length_x= 400.0

```

```

» length_y = 400.0
»
» offset_x = 0.0
» offset_y = 0.0
»
» Inspector.InitStack(size_x, length_x, offset_x, size_y, length_y, offset_y)

```

NOTE: Inspector.InitStack() must be called in the 'OnBegMeas()' of your script!

### Inspector.GetDataPtr()

If the *Virtual Device* generates image data, 'GetDataPtr' returns pointer to current slice of data stack.

*Syntax:* Inspector.GetDataPtr()

```

»data = np.array(Inspector.GetDataPtr(), copy = False) » noise = np.random.randn
(size_x, size_y)
» data[:, :] = noise[:, :] here your own data can be generated !

```

NOTE: Inspector.GetDataPtr() must be called in the 'OnEndStep()' of your script!

### Inspector.ClearOutput()

*Syntax:* Inspector.ClearOutput()

Clears text in output window.

### Inspector.GetOutput()

*Syntax:* Inspector.GetOutput()

Returns text of output window.

```

»results = Inspector.GetOutput ()

```

### Inspector.MoveZ(device,position)

Moves Z device to new position.

*Syntax:* IS.MoveZ(string device,float position)

*Paramter:*

1. string device: name of z device
2. float position: new position

NOTE: The selected position is limited by the range defined in the Z-Dialog!

```

» IS.MoveZ('xyz-Table', 10.0)

```

### **Inspector.GetTablePosition(device)**

Get current x and y position of table device.

*Syntax:* Inspector.GetTablePosition(string device)

*Parameter:* string device: name of table device

*Returns:* tuple with current table x and y position,  
False if table device could not be found or movement is out of allowed range.

» **x,y = Inspector.GetTablePosition('XYZ-Table Z')**

### **Inspector.SetTablePosition(device,x position, y position, max movement)**

Sets new x and y position for table device.

*Syntax:* Inspector.SetTablePosition(string device,float x, float y, float max\_movement)

parameters:

1. string device: name of table device
2. float x: new x position
3. float y: new x position
4. float max\_movement: movement limiter for safety reasons

*Returns False* if table device could not be found or movement is out of allowed range.

### **Inspector.TableSetOffsetX(device,offset)**

Sets X offset of table device.

*Syntax:* Inspector.TableSetOffsetX(string device, float offset)

*Parameter:*

1. string device: name of table device
2. float offset: offset for X positions, maximal offset must be specified in the hardware dialog of the stage

» **Inspector.TableSetOffsetX('XYZ-Table Z',10.0)**

### **Inspector.TableSetOffsetY(device,offset)**

Sets Y offset of table device.

*Syntax:* Inspector.TableSetOffsetY(string device, float offset)

*Parameter:*

1. string device: name of table device
2. float offset: offset for Y positions, maximal offset must be specified in the hardware dialog of the stage

» **Inspector.TableSetOffsetY('XYZ-Table Y',10.0)**



### **Inspector.TableSetOffsetZ(device,offset)**

Sets Z offset of table device.

*Syntax:* Inspector.TableSetOffsetZ(string device, float offset)

*Parameter:*

1. string device: name of table device
2. float offset: offset for Z positions, maximal offset must be specified in the hardware dialog of the stage

» **Inspector.TableSetOffsetZ('XYZ-Table Z',10.0)**

*If using different devices for XY and Z (e.g. intra-vital stage and Pifoc, position offset Z must be given to XY-stage!)*

### **Inspector.Info()**

Returns overview of *Inspector* methods for *Virtual Device*.

*Syntax:* Inspector.Info()

*Returns:* string with info message.

» **print Inspector.Info()**

## 9.20.6 Examples: Generating Image Data, Using Own Serial Device

In this script is shown how to generate own image data with the *Virtual Device*.

```
1
2 import Inspector # import inspector modules
3 import time      # import python time module
4 import numpy as np # import numerical python module
5 import scipy     # import scientific python module
6 from scipy import ndimage # import ndimage from scipy
7
8 # global variables
9 global lena # image data
10
11
12 # OnBegMeas:
13 #
14 # function is called at the begin of an inspector measurement,
15 # Initialisations should be done here
16 # Inspector data stacks can be initialized here
17
18 def OnBegMeas(*args):
19
20     global lena
21
22     lena = scipy.misc.lena() # load 'Lena' image data from scipy
23
24     size_x, size_y = lena.shape # pixel size, must be integer value
25
26     length_x = float(size_x) # physical size, must be float value
27     length_y = float(size_y)
```

```

28
29     offset_x = 0.0    # offset , must be float value
30     offset_y = 0.0
31
32     # print "measurement start"
33
34
35
36     #     INIT STACK (generate image data with python virtual device)
37
38     # 1D, 2D, 3D stacks can be initialized! Every dimension needs pixel-
        size , physical-size and physical offset
39
40     #command to initialize data stack in inspector,
41     Inspector.InitStack(size_x , length_x , offset_x , size_y , length_y , offset_y)
42
43     return 1
44
45
46     #function is called directly after OnBegMeas
47 def OnUpdMeas(*args):
48     return 1
49
50     #function is called for each axis step, if python axis is used in the
        measurement Wizard, the current physical position and step position
        is given in the function arguments
51 def OnBegStep(*args):
52
53     # python axis positions:
54     pos = args[0]
55     step = args[1]
56
57     #print "Pos:\t" + str(pos)
58     #print "Step:\t" + str(step)
59
60     return 1
61
62
63     # function is called for each axis step after OnBegStep
64 def OnUpdStep(*args):
65     return 1
66
67
68     # OnEndStep:
69     #
70     # function is called for each axis step after OnUpdStep:
71     # Image data should be generated here, a pointer to the correct position
        of the data stack can received by this command:
72     #     'data = np.array(Inspector.GetDataPtr() , copy = False) '
73     #
74     # Pointer to all data stacks that are generated in the measurement (also
        from other devices) are given in the arguments
75     #
76     # If python axis has moved since the last step can be seen in the
        arguments
77
78     def OnEndStep(*args):
79
80     # pointer to all data stacks that are generated in current measurement

```

```

81
82     stacks = args[0]           # different stacks
83     resolutions = args[1]     # pixel sizes for each stack
84     lengths = args[2]        # physical sizes for each stack
85     offsets = args[3]        # offsets for each stack
86
87     AxisStep = args[4]       # boolean if the python axis has moved since the
                               # last step
88
89
90     size_y, size_x = lena.shape
91
92
93     #     get data pointer to current slice of data stack
94
95     data = np.array(Inspector.GetDataPtr(), copy = False)
96
97     # copy lena image data to current slice (and add noise to the image),
98
99     # here your own data can be generated !
100
101     noise = np.random.randint(85, size= (size_x, size_y))
102
103     data[:, :] = lena[:, :] + noise[:, :]
104
105     return 1
106
107
108 # function is called at the end of a measurement
109 # de-initialisations should be done here
110 def OnEndMeas(*args):
111     #print "finish "
112     return 1

```

In this example serial communication via python is shown. The script opens a com port and moves a filter wheel during the run of an *ImSpector* measurement. The commands used to control the filter wheel are taken from our real filter wheel that can be delivered with our *TrimScope*.

```
1 import serial
2
3 ## global variables declaration
4 global myComPort
5 global ser
6 global comError
7
8 ## define the device commands
9 global devMoveTo
10 global devRef
11
12 ## initialize global variables
13 myComPort = 3
14 comError = 0
15
16 devRef = 0x80
17 devMoveTo = 0xc0
18
19 def sendDeviceCommand(command):
20     global comError
21     if(not comError):
22         try:
23             ser.write(command)
24             answer = ser.read(10)
25             if((ord(answer[0]) & 0xc0) != 0x40):
26                 raise
27
28         except serial.SerialException:
29             ser.close()
30             comError = 1
31
32 def initDevice():
33     global myComPort
34     global ser
35     global comError
36     try:
37         comError = 0
38         comport = "COM" + str(myComPort)
39         ser = serial.Serial(comport, timeout=0)
40         command = chr(devRef)
41         sendDeviceCommand(command)
42     except serial.SerialException:
43         ser.close()
44         comError = 1
45         print "Could not open comport"
46     except:
47         print "Command rejected"
48
49 def OnBegMeas(*args):
50     initDevice()
51     return 1
52
53 def OnUpdMeas(*args):
54     return 1
```

```

55
56 def OnEndMeas(*args):
57     global comError
58     if (not comError):
59         ser.close()
60     return 1
61
62 def OnBegStep(*args):
63     global comError
64     if (len(args) > 0):
65         newpos = int(args[0])
66         command = devMoveTo | newpos
67         command = chr(command)
68         try:
69             sendDeviceCommand(command)
70         except:
71             print "Command rejected"
72             return 0
73     return 1
74
75 def OnUpdStep(*args):
76     return 1
77
78 def OnEndStep(*args):
79     return 1

```

### 9.20.7 Error handling

Syntax errors and runtime errors are shown in the output window of the life dialog. If a runtime error occurs the script is stopped immediately.

Sometimes it makes sense to implement an own error handling in the script: e.g. when serial communication is used and it is important to open and close com ports properly. Error handling can be done by try...except blocks in the python script.

```

1 def OnBegMeas(*args):
2     try:
3         .... some functionality
4     except:
5         .... handle any error

```

**The measurement is stopped immediately if a script routine returns 0.**

For example if com port opening fails:

```

1 def OnBegMeas(*args):
2
3     port = OpenComPort(4)
4
5     if not port:
6         print "Could not open communication port"
7         return 0
8
9     return 1

```

## 9.20.8 Add Instrument Mode & Measurement Mode

If the python device generates image data, it is useful to set up an own *Instrument mode*.

The Instrument mode can be added in the *reconfig.ini* file that is located in the Config folder of *ImSpector*.

Open the file with a text editor and add a mode similar to this:

```
[REC02]
LABEL=Python Device
SWITCHON=Python
SWITCHOFF=Neo,PMT
ICONFILE=python_device.bmp
```

If the data axis is used, it might be useful to set up a *Measurement mode* for the device.

Measurement modes can be edited in the *axisconfig.ini* that is located in the Config folder of *ImSpector*.

Open the file with a text editor and add a mode similar to this:

```
[AC01]
LABEL=PYTHON-TIMELAPSE
AXIS_FIRST= PYTHON
AXIS_SECOND= TIME TIME
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=Time_150.bmp
COMMENT=Time Series
```

## 10 Description of Plugins

ImInspector ships with a variety of plugins covering various numerical evaluation functions. A plugin can either be started over its corresponding menu entry in the "Analysis" menu or over an existing tool bar button.



Figure 109: The main menu with highlighted Analysis menu.

### 10.1 Arithmetics

Arithmetic operations can be used on a single data stack or to process two data stacks. All functions mentioned in this chapter do not only operate on the two data axes visible in the data window. They also regard the hidden data axis. All arithmetic operations can be called from the dialog arithmetic which is divided into a part called stacks and a part called operation. To execute an operation the user has to choose the kind of operation from the pull-down menu and enter the desired parameter (threshold, factor or offset). Then the data stacks must be chosen on which the selected operation is used. To choose a data stack move the mouse pointer to the button at the right of the data name box and press the left mouse button. Then drag the mouse pointer on the desired data window and do a left click again. The name of the selected data stack should appear in the box. A function is executed by pressing the button go at the bottom of the dialogue. The resulting data stack appears in a new data window.

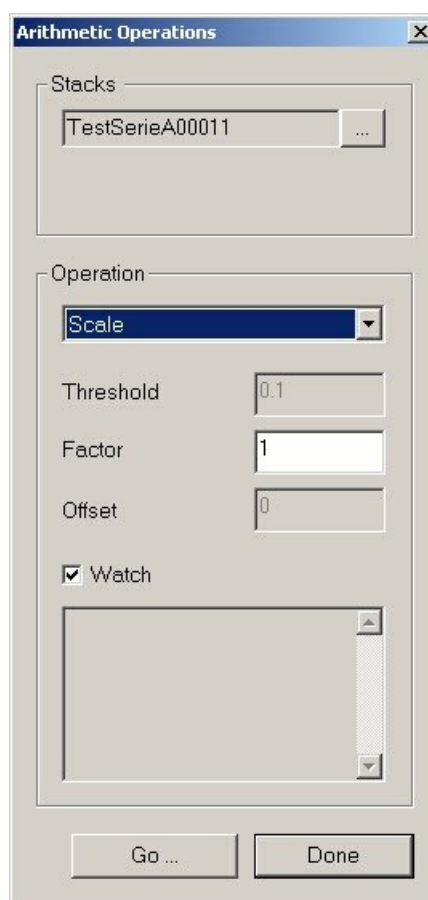


Figure 110: The arithmetics dialog.

#### 10.1.1 Arithmetic Operations on a Single Data Stack

- **Scale** Each pixel is multiplied with a user-defined factor. Multiplying an integer stack with a float value (e.g. 1.25) produces a float output stack.
- **Offset** A user-defined offset is added to each pixel. Handling an integer stack with a float offset produces a float stack.
- **Invert** All pixel values are inverted. The user has to define a threshold that determines which pixel values are regarded.

- **Normalize** A data stack is normalized to a user-defined value.
- **Subtract background** The lowest pixel value contained in the data stack is subtracted from all pixels.

### 10.1.2 Arithmetic Operations for Combining Two Data Stacks

The premise to execute operations that combine two data stacks is that the dimensions of the stacks match. Two data stacks are combined by processing corresponding pixel values.

- **Add** (add up two different data stacks)
- **Subtract** (subtract one data stack from another)
- **Multiply** (multiply two data stacks with each other)
- **Divide** (divide one data stack through another)
- **Function** Calculate the pixel by a user defined function.  
In this function the following variables could be used:  
x, y, z, t: the position of the pixel in physical units  
X, Y, Z, T: length of the stack in physical units  
r: Count of the first source stack  
R: Count of the second source stack  
s, u, v, w: the coordinates of the pixel (0 ... resolution - 1)  
S, U, V, W: the resolution of the stack  
The function can consist of the following operators, constant or mathematical functions:  
+, -, \*, /, ^, pi, abs(), exp(), ceil(), floor(), ln(), sqrt(), sinh(), cosh(), tanh(), heaviside(), sgn(), delta(), erf(), si(), ci(), sin(), cos(), tan(), asin(), acos(), atan(), poidev(), gaussdev(), rand(), atan2(), max(), min()

### 10.2 Delete Negative

Selecting the menu entry "Delete Negative" removes all pixel from the image stack that are below zero by setting them to zero. Negative pixel can occur after the subtraction of two images or similar operations. Note that deleting the negative pixel from the stack, the stack's original data will be changed.

### 10.3 Subtract Background

After pressing on "Subtract Background", the minimum value of the image is searched. This minimum is used as the background value and it will be subtracted from every pixel in the stack. Note that with the subtraction of the background, the stack's original data will be changed.

### 10.4 Interpolation

The dialog interpolation allows to change the resolution on any of the four data axes. The original resolution is displayed in brackets left to the edit field where the user can enter a new resolution value. Depending on the user's choice, the axis is stretched or compressed. The stretching or compression can be done with one of the following algorithms:

1. Lagrange interpolation



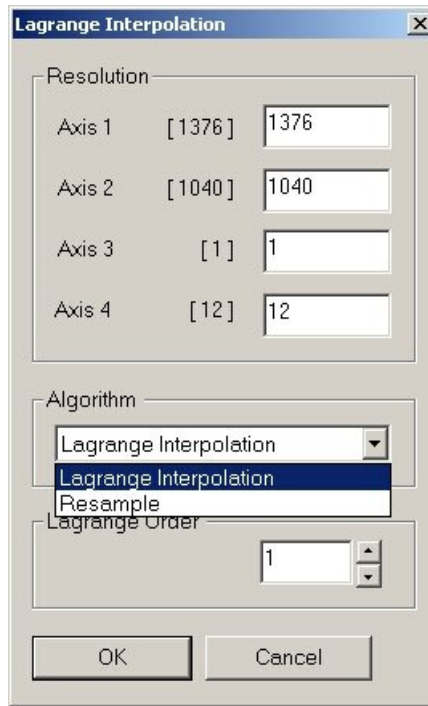


Figure 111: The interpolation dialog.

## 2. Resample

If you chose Lagrange interpolation, the Lagrange Order can be given in the spin box below.

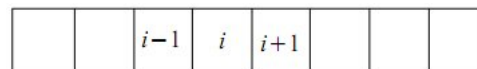
## 10.5 Smoothing

The dialog "Smooth" allows to smooth the data on any of the four axes. Select the data stack which should be smoothed. Depending on the stack's dimensions, it is possible to select up to four axis for smoothing.

Illustration 113 shows the result of the stack displayed in illustration 112.

### 10.5.1 How it works

The algorithm used for smoothing in this plugin is rather straightforward. It simply calculates a weighted sum over the direct surrounding pixels at each position on an axis.



$$p_i = \frac{p_i}{2} + \frac{p_{i-1}}{4} + \frac{p_{i+1}}{4}$$

Figure 114: The smoothing algorithm.

The method can best be shown at one axis. The pixel on position  $i$  is replaced by the sum of its left and right neighbor with itself where the neighbors are multiplied with 0.25 and the pixel its self is multiplied by 0.5.

## 10.6 Exponential Fit

This plugin can be found inside the Inspector Menubar in Analysis→Flim on the tab "Exponential Fit".

Base for an exponential fit is an time series based image stack in which the fluorescence behavior of a substance should be analysed. As shown in illustration 35 a z-profile is drawn

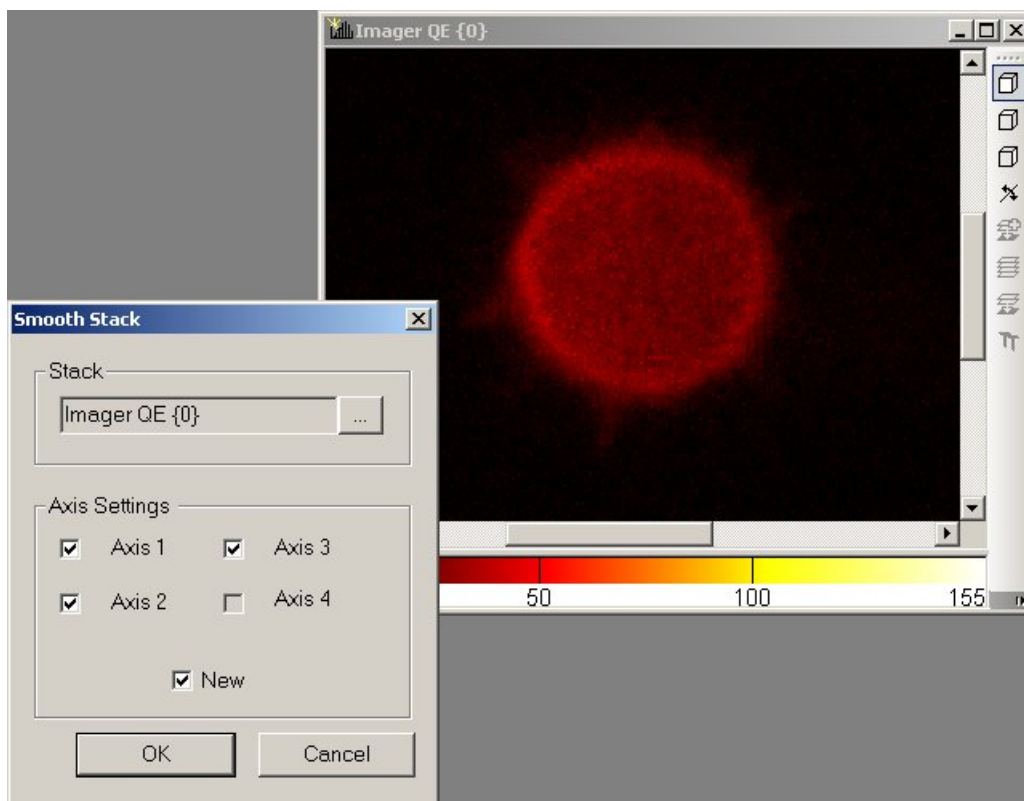


Figure 112: The smooth dialog with three axis selected.

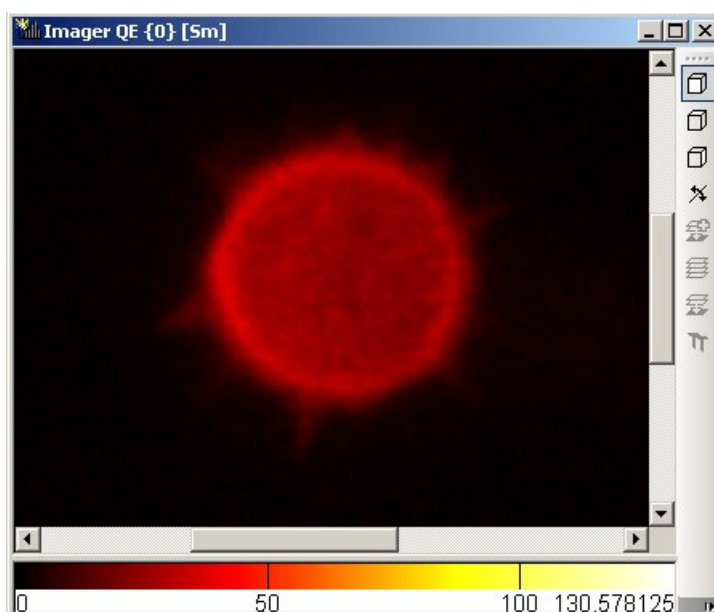


Figure 113: The smooth dialog with three axis selected.

right to the image stack. The profile shows the fluorescence decay over a short period of time. This decay behaves much like an exponential curve of the form:

$$a \cdot e^{\frac{-t}{k}} + b$$

In the plugin window, choose the stack for which you want to calculate an exponential fit and choose the axis which should be fitted. Activate the option "Scale", "Decay" and/or "Offset"

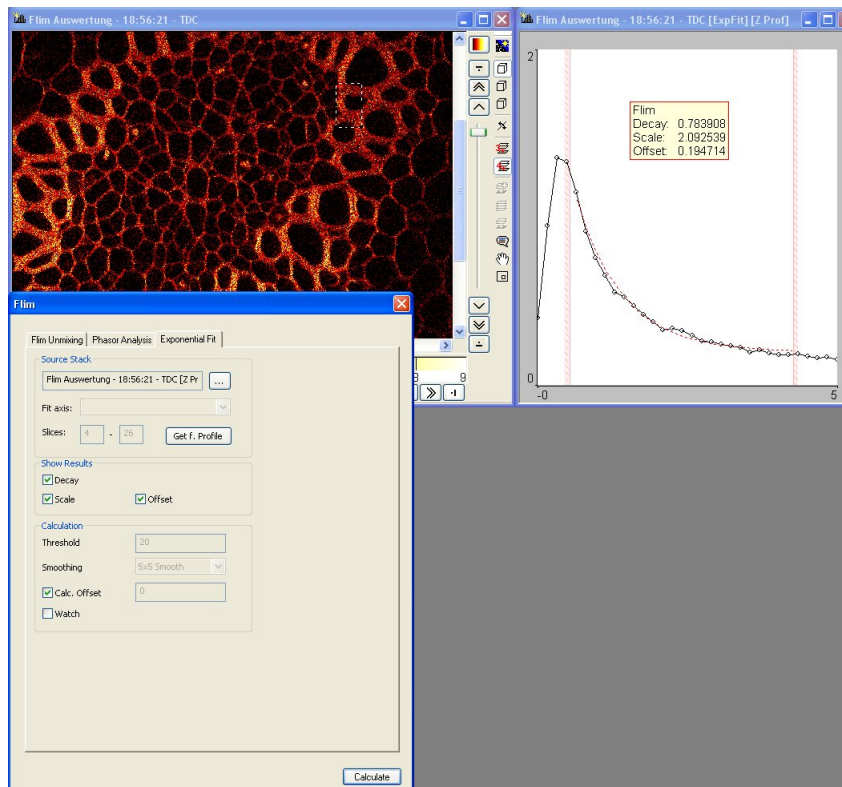


Figure 115: The exponential fit of fluorescence behavior of a substance.

for the result image. The scale option activates the calculation of the part, the offset option the  $b$  part of the formula  $a \cdot e^{-\frac{t}{k}} + b$  and the Decay option calculates the the  $k$ . To evaluate the quality of the fit, the plugin is also able to fit profiles. Just select a profile as source stack. The desired results are shown in the graph window. The limits in the graph window may reduce the fitted region. It is reasonable to activate the "Watch" option to recalculate the fit, if the ROI of the source profile changes. The calculation of the offset takes time. For real time flim it is possible to deactivate it. The plugin will operate with the entered value. The offset is the count value towards the fluorescent signal converges. Note that in this case the fitting is less exact. The threshold is another possibility to increase the speed. Only those pixel of the stack are calculated, which count values of the first image (the brightest one) are above the entered threshold. Its value should be set higher than the background but lower than the interesting structure.

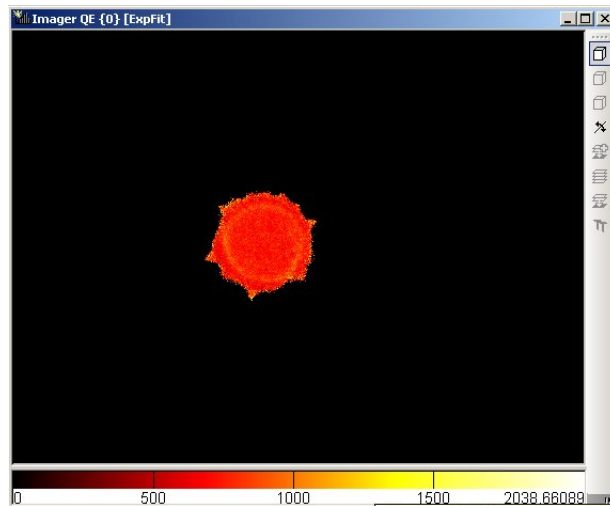


Figure 116: The result of an exponential fit where the  $k$  of the formular  $a \cdot e^{-\frac{t}{k}} + b$  have been calculated. Painted are the  $k$ -values. The darker the pixels in this result image, the faster the fluorescence signal decreased over time in this part of the substrate.

## 10.7 Filter Box

The FilterBox plugin provides algorithms for smoothing an image stack. Smoothing an image means the reduction of the noise in the image. To do this, select the image stack you want to smooth under "Source Stack". Then open the drop down box under "Filter Options" and choose the desired algorithm. Currently, one can choose between two Algorithms, mean and median. Both of them can be applied in two different forms, first as a 3x3 square and second as a 5x5 square.

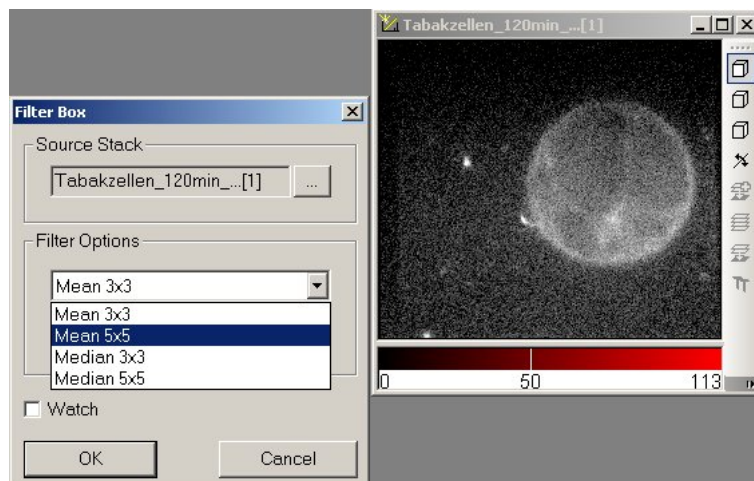


Figure 117: The FilterBox plugin.

The result of a mean 5x5 filter is displayed in Illustration 118.

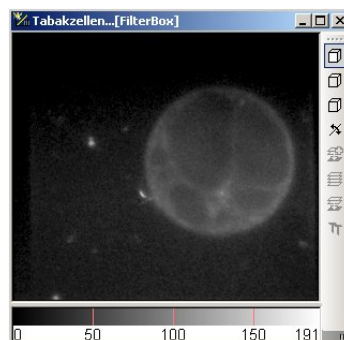


Figure 118: Result of the 5x5 mean filter.

## 10.8 The mean filter in detail

The idea of mean filtering is simply to replace each pixel value in an image with the mean (average) value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings.

Let  $x$  be a position on the  $x$ -axis of the image and  $y$  the position of the  $y$ -axis of the image. Let  $i_{x,y}$  be the pixel at the position  $(x, y)$  and let  $s$  be the  $s \times s$  surrounding of the pixel  $i_{x,y}$  with  $s = 3$ . The mean

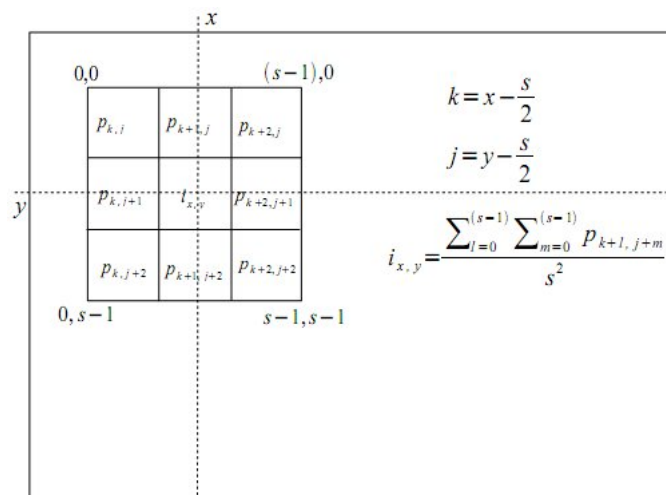


Figure 119: The mean filter in detail

filter recalculates each pixel of the image by adding every pixel of the  $s \times s$  surrounding. This sum is divided by  $s^2$ . The result then replaces the original pixel value of  $i_{x,y}$ .

## 10.9 The median filter in detail

The median filter is also used to reduce noise in an image, just like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image.

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:  
115, 119, 120, 123, 124,  
125, 126, 127, 150

Median value: 124

Figure 120: The median filter sorts all values of a surrounding and takes the center value as the median.

Like the mean filter, the median filter considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighboring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used.) Figure 120 illustrates an example calculation.

## 10.10 Fit Session

### 10.10.1 Example

This plugin affords to fit a model function, for example an exponential function like  $f(x) = a \cdot \exp(-k \cdot x)$ , to a profile of some measured data. This section contains an example of how to fit a lifetimeprofile of flim data with the fit session plugin step-by-step.

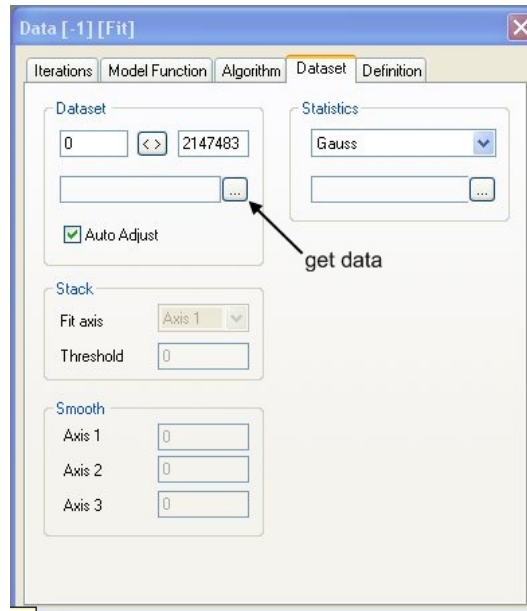


Figure 121: After open the fit session plugin.

After you have opened the fit session plugin you find some tabs to change the settings of this plugin. Inside the tab named "Dataset" exists an area which is also called "Dataset". Here you can find a button to pick up a profile from a profile window (see arrow at figure 121). After you have chosen some data it is recommended to limit this data to an area that fits best to your model function. In our case we want to fit an exponential function, so we limit the data to the exponential decay of the profile. This can be done by slide the limits of the profile window to the favorite position and then check the "Auto Adjust" checkbox inside the "Dataset" area. For an example of this step see figure 122.

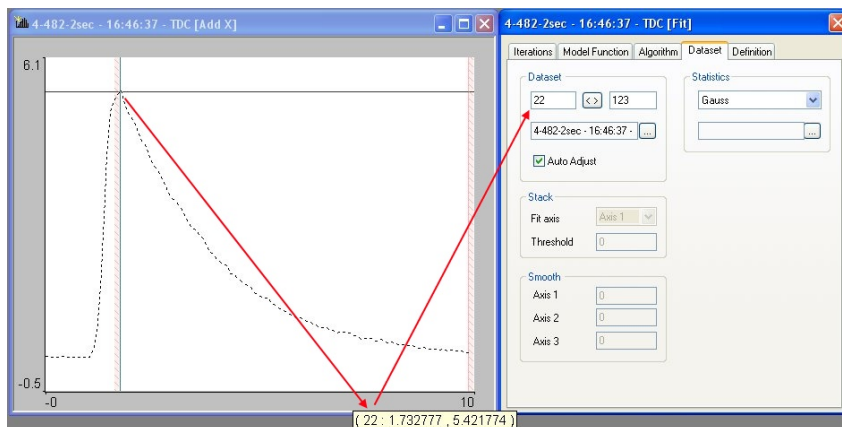


Figure 122: Set the limits.

In the next step we change the tab to the one called "Algorithm" (see figure 123). Here we check the checkbox "Auto Update" in the area "Automation" and the checkbox "Include" in the

area "Background". The last setting puts an extra parameter inside our model function to calculate the "background" or offset of the profile.

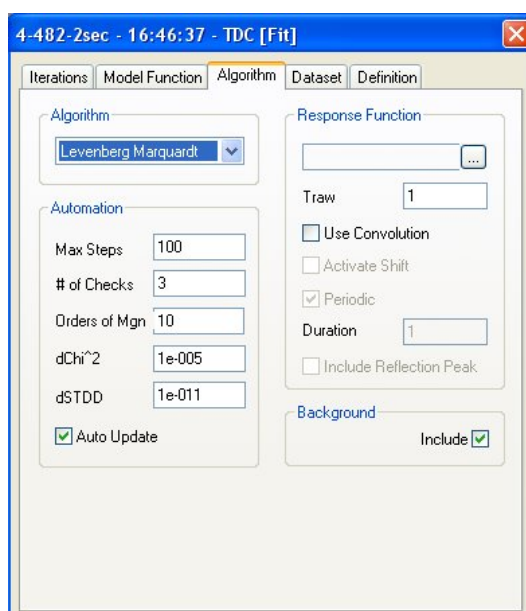


Figure 123: The "Algorithm" tab.

At next we choose a model function in the tab "Model Function". We like to fit an mono-exponential function, thus we choose the function exp1 in the category "Exponentials". For bi-exponential functions you can choose exp2 and for tri-exponentials exp3. The parameter that will be calculated in the exp1 function are  $a_i$  and  $k_i$  (see figure 124).

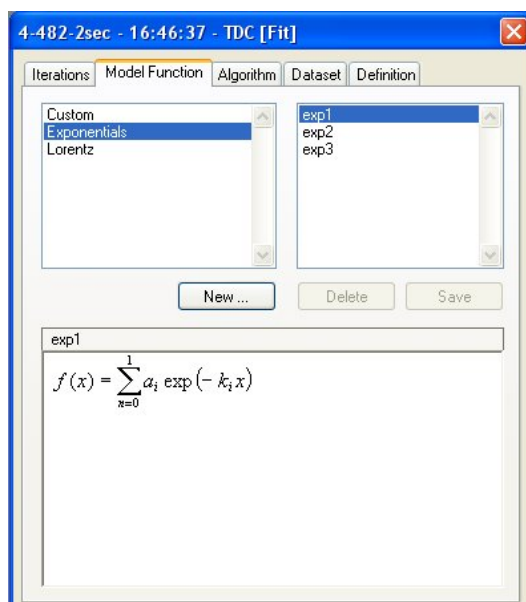


Figure 124: Choose a model function.

The last step is the fit of the functions. For this we change to the "Iterations" tab and press the "Auto" button. The result is shown in figure 125.

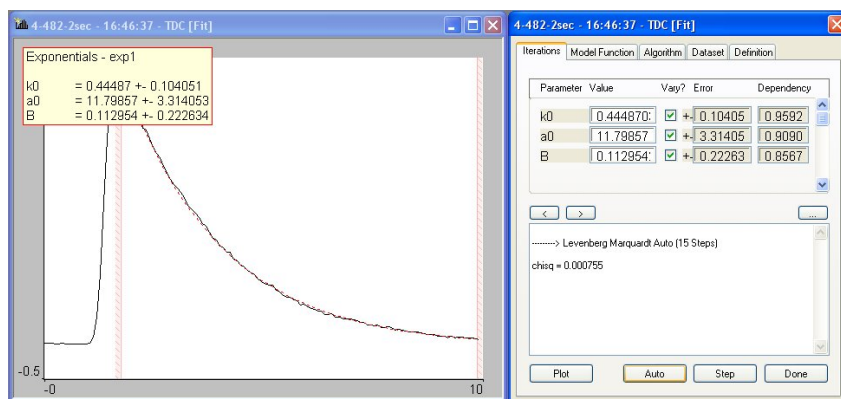


Figure 125: Result of the fit calculation.

The parameter B is the background or offset parameter, which was chosen to be calculated inside the "Algorithm" tab by checking the "Include" checkbox. Parameter  $a_0$  is the scale value of the exponential function. With the help of the last parameter  $k_0$  it is possible to calculate the decay or lifetime of this profile. Our model function has the form  $f(x) = a_0 \cdot \exp(-k_0 \cdot x)$ . A lifetime exponential function is of the form  $f(x) = a_0 \cdot \exp(-\frac{1}{l} \cdot x)$  where  $l$  is the lifetime (see also section 10.6). According to this function the lifetime is:  $l = \frac{1}{k_0}$

Thus the lifetime of the example profile is:  $l = \frac{1}{0.444870} = 2.2478$

### 10.11 Profile Fit

### 10.12 Profile Multiply

With this plugin you can multiply/divide a stack along any dimension with a profile that comes with the same resolution. Examples are z-drop correction for a depth scan, bleaching compensation along the time axis, ...

Pointing to a profile that doesn't fit the resolution of the selected axis leads to an error message. You may also just select a stack and a profile. If the profile's dimension fits any axis of the stack this axis is selected automatically.

### 10.13 Spectral Unmixing

After taking images with a spectrometer, you get an image stack of e.g. 32 (depending on your spectrometer) different images, each belonging to one spectrum. Filtering out certain parts of an image belonging to one spectrum is the task of this plugin. The plugin can be found under the analysis menu or in the toolbar under the icon shown in figure 127:



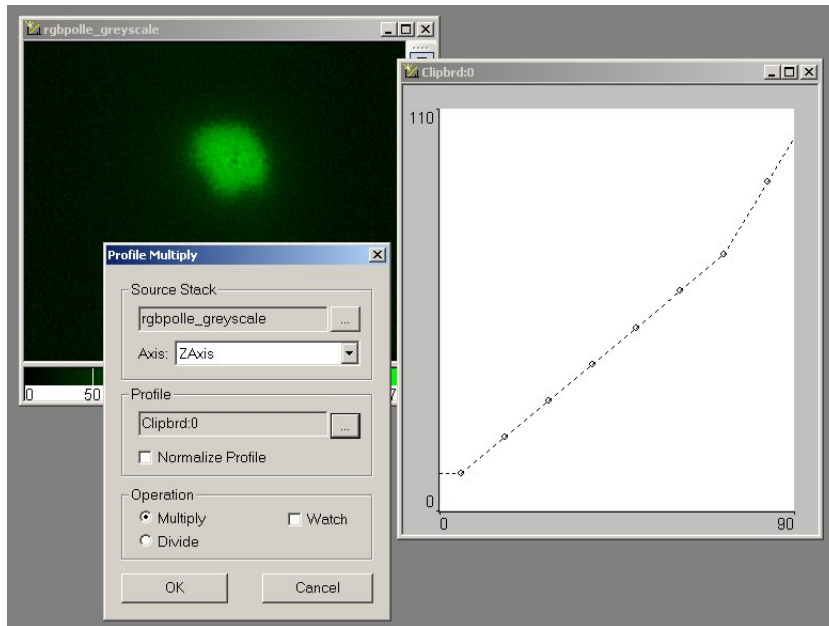


Figure 126:



Figure 127:

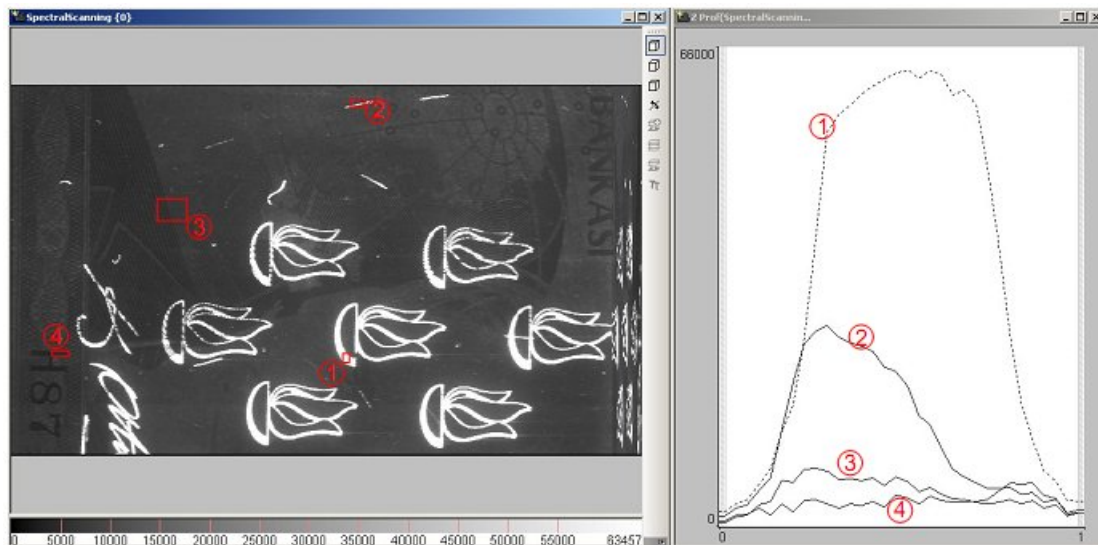


Figure 128: A spectral scan of a banknote with four different profiles.

Illustration 128 shows a spectral scan of a banknote. Four different profiles have been drawn, the first in the bright area of the tulip like symbol, the second in the security mark, the third describes the background and the fourth was defined in the number of the banknote. Now, the "Spectral Unmixing" dialog is opened and the different profiles are assigned to different colors. In this example, profile 1 gets the red color, profile 2 the green, the background profile 3 gets the gray color and the profile 4 the blue color. When the "OK" button is pressed, the plugin now checks each pixel in the image stack and assigns it to one of the spectrums belonging to the selected profiles. As a result, one gets four new images, one for each active profile. In this case, pixels belonging to the spectrum of profile one will be

drawn in red and appear in image one, pixels belonging to the spectrum of profile two will be drawn in green and appear in image two, etc. Illustration 130 shows the result of spectral unmixing of the image stack in Illustration 128 in gallery mode. It clearly shows the differentiation of the spectrums. To leave the gallery mode, right click into the image and select "Manage Channels". Uncheck the "Gallery Mode" from this submenu.

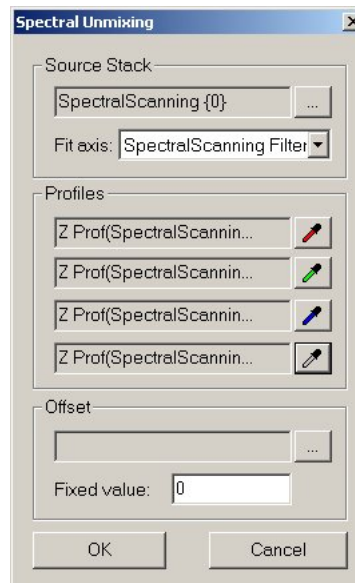


Figure 129: The spectral unmixing dialog.

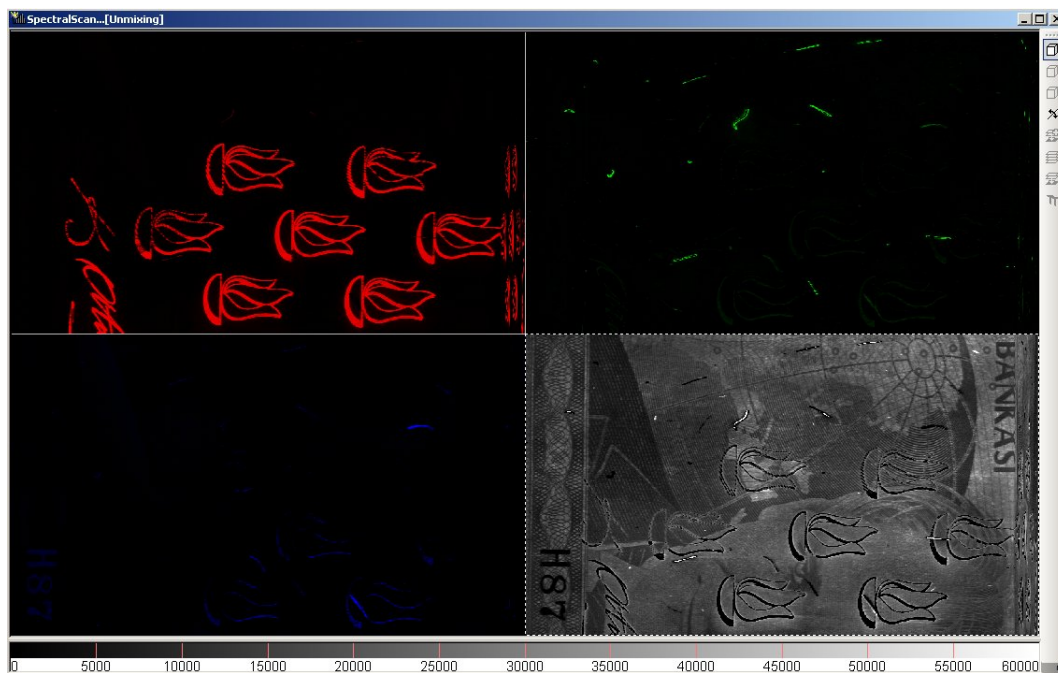


Figure 130: The result of spectral unmixing.

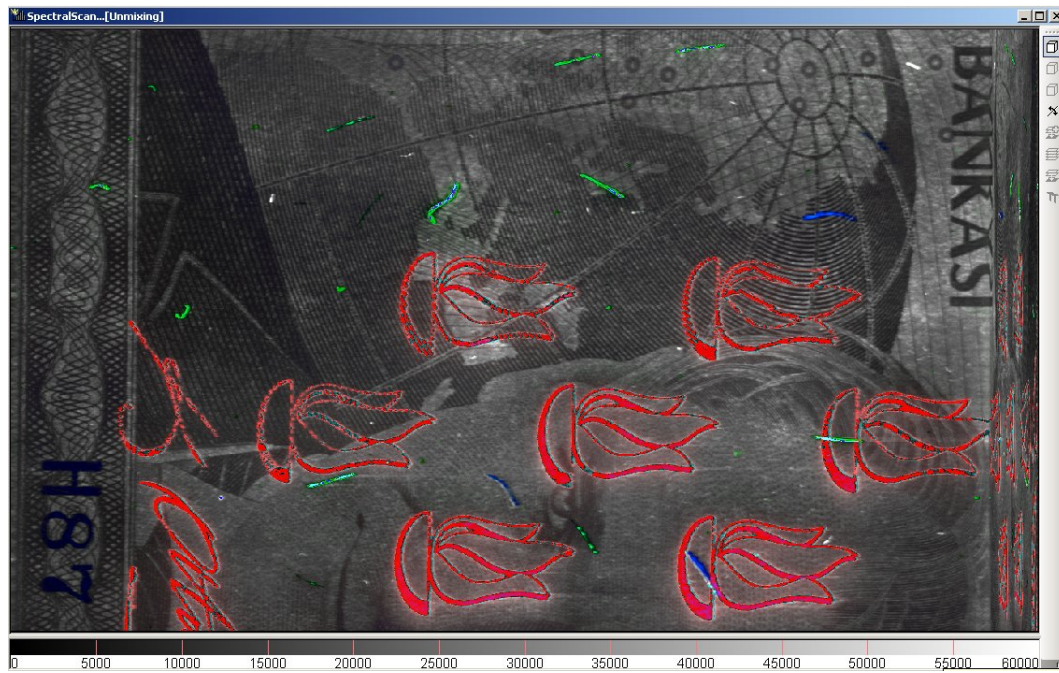


Figure 131: The combined image of the result shown in figure 130

## 10.14 Image Series Viewer

The Image Series Viewer provides a convenient way of displaying an image series that was previously saved with the TimeDriver (See ??). The other feature is to process the image series. There is the possibility to select a certain region of interest or a specific range of the available axes from the series. That can then be imported into a new image stack or exported into video or (Ome-)Tiff Formats.

### 10.14.1 Starting the plugin

Unlike other plugins, this one has no entry in the "Analysis" menu. It can be started over its toolbar button and its entry in the "Edit" menu.



Figure 132: Main menu with highlighted Image Series Viewer toolbar button.

### 10.14.2 Opening a series

Firstly, select your image series on the upper field in the "Stack Search Window" see illustration 133. Click on the "+" next to the drive and select the folder where an image series was stored. If an image series is found in a directory it appears in the middle field of the "Stack Search Window".

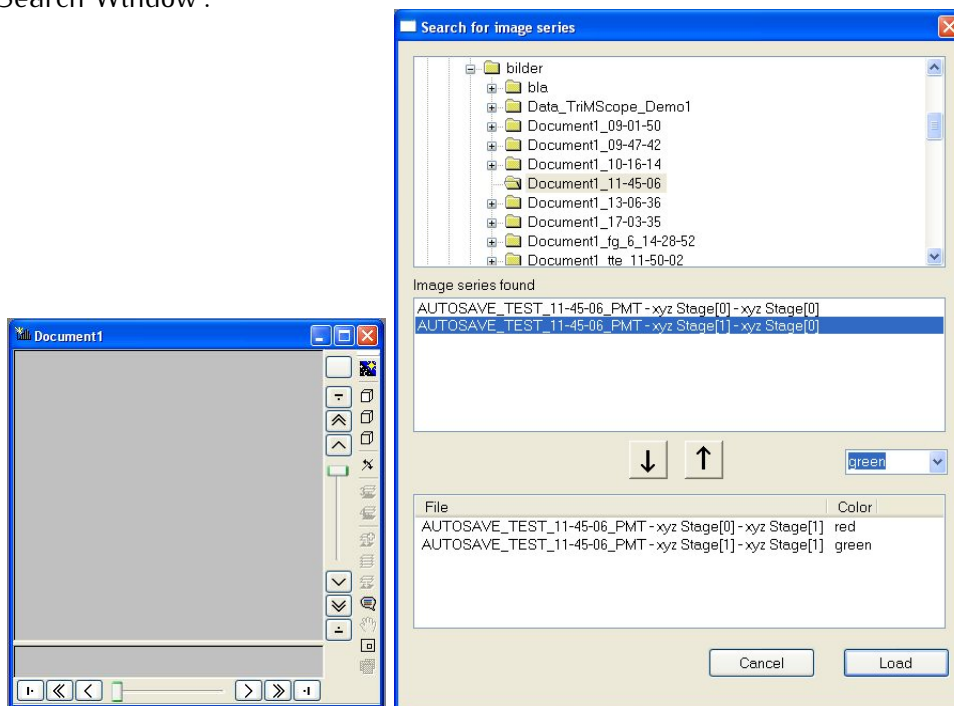


Figure 133: The Stack Search Window with a blank document.

Now, you can select one or more channels of one image series and press the downarrow to put the selected series in the bottom field. Choosing different series with varying image sizes fails to load or exhibits strange behaviour. It is possible in this field to select one of the channels separately and choose a colormap in the right drop down menu for the selected series. If no colormap is chosen the colormap fire appears by default. At last, the button "Load" below should be pressed to load the series. The "Stack Search Window" closes and

the first image of your series are loaded into the document window. Alternatively, you can double-click on one series to load that one series.

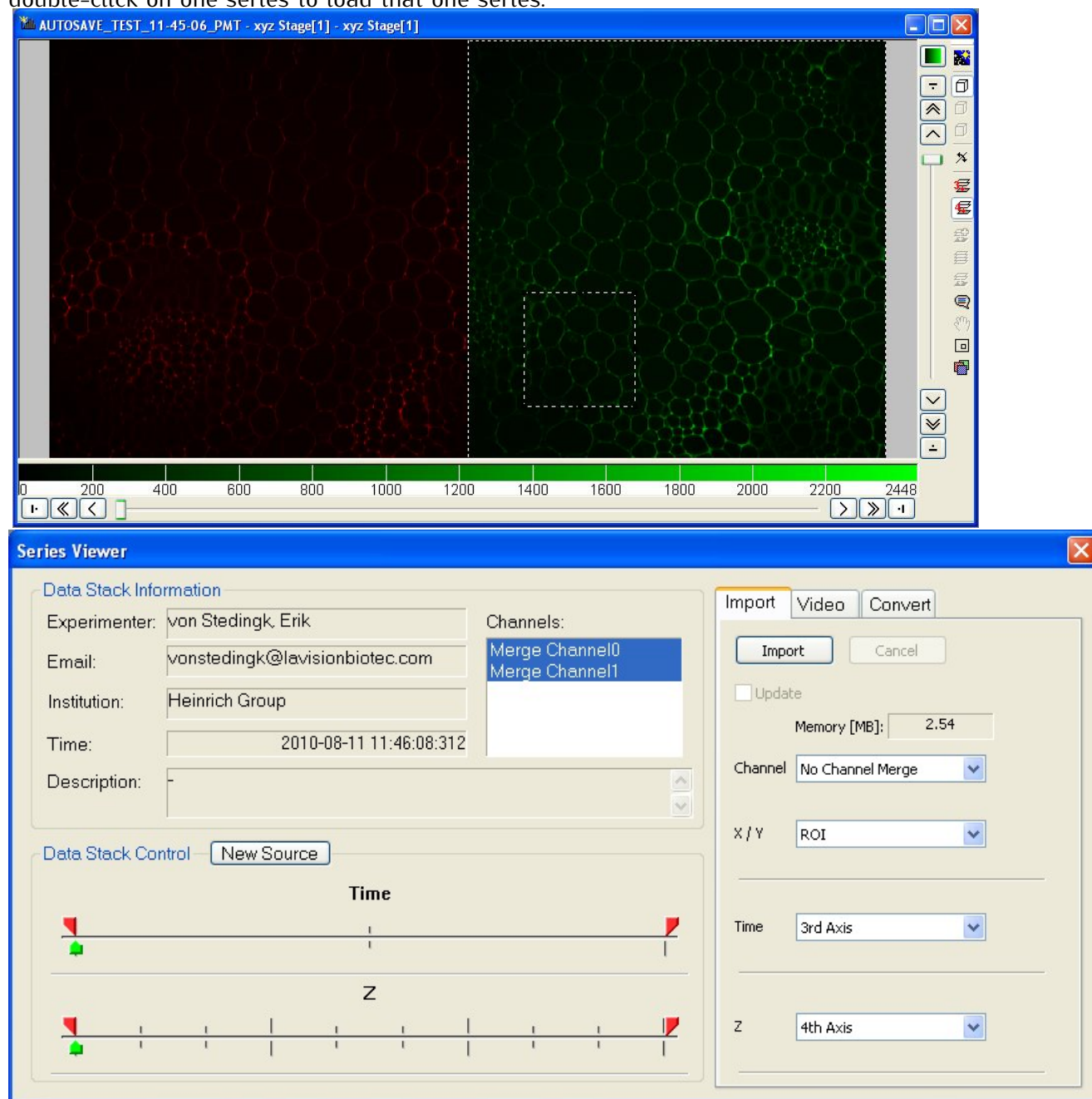


Figure 134: The series control panel with image window.

### 10.14.3 The control panel

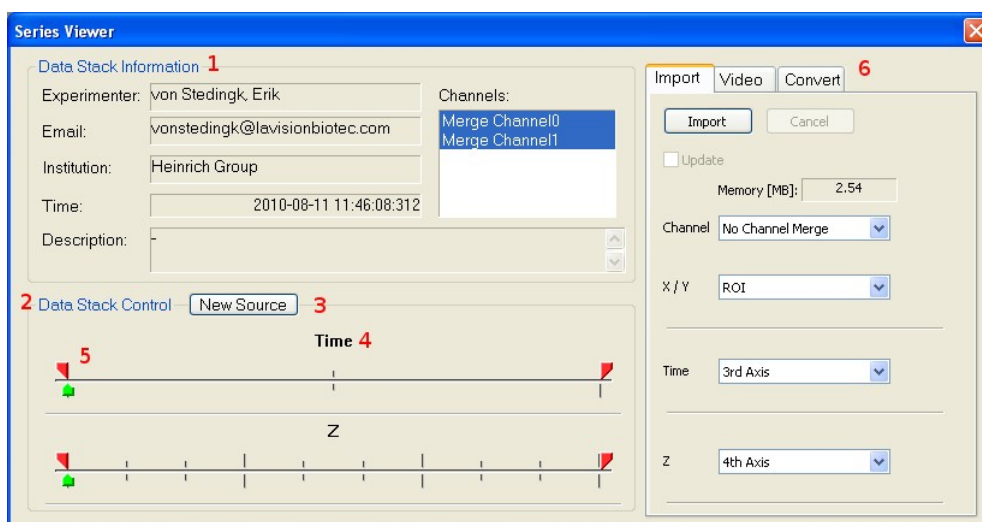


Figure 135: Images series control panel.

Figure 135.1 There is Data Stack Information taken from Ome metadata.

Figure 135.2 The Data Stack Control frame presents slider controls for devices and their axes from the image series.

Figure 135.3 The "New Source" button opens the "Stack Search Window" to load new series.

Figure 135.4 There is an identification for each slider control.

Figure 135.5 The thumbs manage the image series data. (Later more.)

Figure 135.6 The tabs process the series in different ways.

#### 10.14.4 Importing a series - the Import tab

Here you can create a new image stack with the "Import" button. The "Cancel" button stops the import process if some option needs a change. The "Update" checkbox enables re-generation of the destination image stack on changes to slider control axes as well as to the region of interest of the source image stack. The "Memory" editbox shows the calculated size of the destination stack. The "Channel" pull-down list has different types of merging the various channels of the viewer. The "X / Y" pull-down list chooses between "Full Frame" and "ROI". Both options are available with and without Binning. Each slider control pull-down list chooses between the "Single Frame", "3rd Axis" and "4th Axis". It is also possible to choose an operation type to be applied on an axis: "Projection", "Add Up" and "Average".

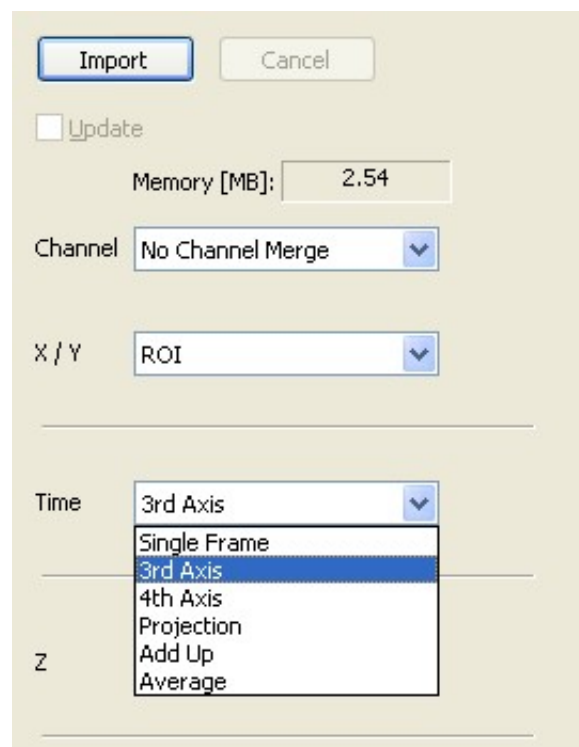


Figure 136: Import tab.

### 10.14.5 Encoding a video from a series - the Video tab

On this tab you can create a raw AVI video or a compressed MP4, DivX or Mpeg video. For this you have to choose a filename and a location with the "..." button ( 1 ). When the encode is finished it is possible to view the video via the "Play video" button. To control what is and how it is to be encoded, set the option for "X / Y". Next to that choose what axis should be the "Video Axis". Optionally, choose on what axes what operation should happen.

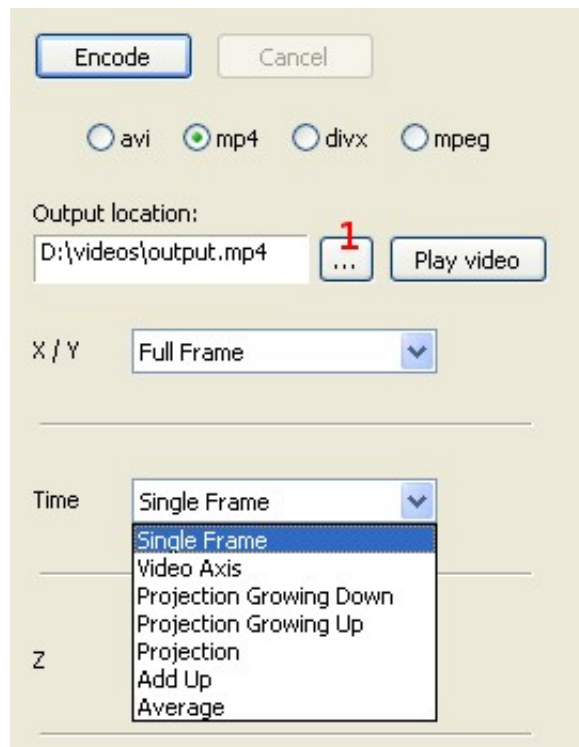


Figure 137: Video tab.

### 10.14.6 Exporting a (Ome-)Tiff from a series - the Convert tab

The last tab lets you export the series to a single Ome-Tiff file, multi-channel Ome-Tiff files - if multiple channels have been loaded - or a simple Tiff file without Ome metadata - grayscaled or colored. Once again you choose the filename and a location with the "..." button ( 1 ). If the export is done it is possible to view the Tiff file with the "Open tiff" button. To control what is and how it is to be exported, set the option for "X / Y". Next to that choose what axis should be the "Tiff Z - Axis" and "Tiff T - Axis", if necessary. Optionally, choose on what axes what operation should happen. Finally, what files will be generated from the export are shown in the "output files" frame, if you press the down-arrow button.

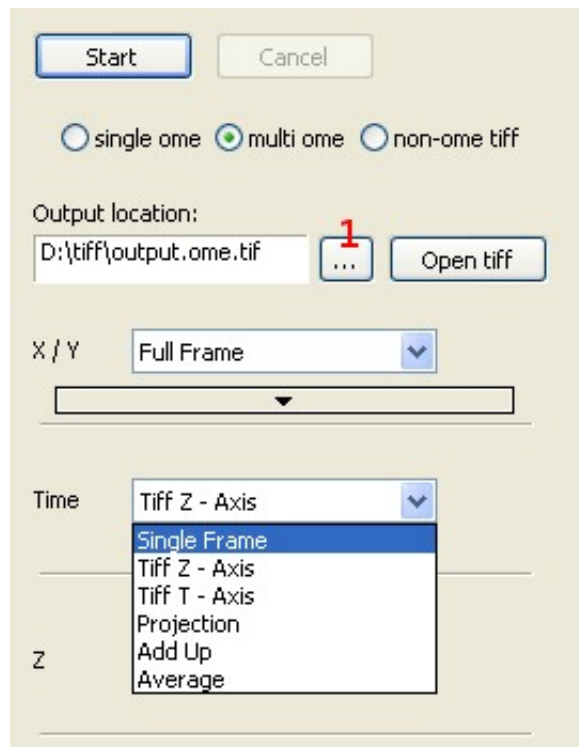


Figure 138: Convert tab.

### 10.14.7 Stitching a series - the Import tab again

Select "Stitching" as merge type on the "Channel" pull-down list. An icon button will appear to the right. The "Stitching" dialog [9.4.2] opens via that button. Only this way the dialog connects with the image series viewer to get its data and therefore that dialog does not show its own slider control(s).



Figure 139: Stitching via Import tab.

### 10.14.8 The Slider Control(s)

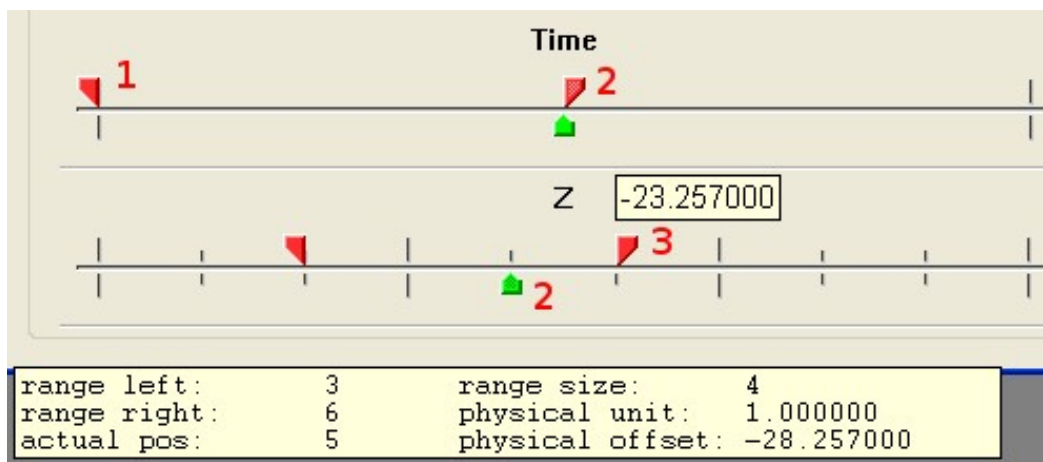


Figure 140: S.

A short description of the thumbs to the right.

1. A thumb to span the range of interest.
2. A selected thumb of a slider control is highlighted.
3. A hovered thumb displays its physical value of the axis.
4. A tooltip displays an overview of the slider control if a thumb is hovered.

Now that you got a brief overview of the "Image Series Viewer" and its features, it is time to explain how to control the thumbs of the slider controls. With the red thumbs you can span a range of a device or one of its axes that take part in the above-mentioned processes. The green thumb moves through the source image stack of the loaded image series. Next to the obvious moves of the thumbs, by clicking on them and moving the mouse, there are other ways to control them.

1. Click and release on the track of the thumbs to let the thumb jump to that position.
2. Click and hold pressed on the track between the red thumbs to shift their range.
3. If the range of the red thumbs is too narrow to click between the thumbs to shift, it is then possible to move the range by clicking in the upper third of one of the red thumbs. If the cursor is within that part the thumbs are "highlighted".



It is also possible to control the thumbs of the slider controls by keyboard - but only if the "Image Series Viewer" Dialog is focused, and not the tab dialogs or other windows.

1. With the TAB key or Shift+TAB combination the next or previous thumb is selected.
2. With the LEFT and UP key or RIGHT and DOWN key the thumb is moved by one backward or forward. With the combination of Shift and one of these keys the thumb is moved by ten.

The last possible way to control the thumbs is with the mouse-wheel. Select the thumb to move, click with the mouse-wheel on the slider control and steer the wheel to move the thumb.

## 10.15 Averaging

The Averaging plugin can be used to build an average stack of a stack that is updated frequently, e.g. a stack of a camera measurement. It also allows to build the average with a sixteen fold oversampling. Instead of building the average the plugin can also be used to add up the data of the updated stack.

If the window that includes the selected stack contains more than one stack, result stacks for all of these stacks are built.

### 10.15.1 Using the plugin

The plugin can be found in the menu "Analysis", sub menu "Live Arithmetics" "Averaging".

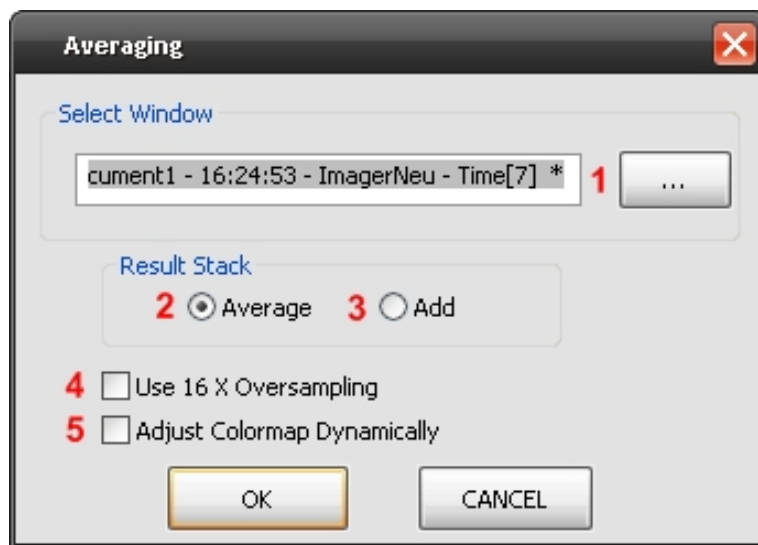


Figure 141: Averaging dialog.

In the plugin dialog the window whose stacks should be averaged or added up can be selected (1). As default the window of the last active stack is selected. The name of the selected stack is shown in the edit box.

You can choose between two kinds of result stacks: a stack that builds the average of the source stack (2), or one that adds up the data of the updated source (3). If "Average" is selected it is possible to use sixteen fold oversampling and/or dynamic colormap adjustment.

By activating the checkbox "Use 16 X Oversampling" (4) at the initial start of the plugin you can choose if sixteen fold oversampling should be used. Therefore the values of the source stack are multiplied by 16 and then used for building the average. This is only possible for source stacks of the data type 'integer' and 'word'.

By activating the checkbox "Adjust Colormap Dynamically" (5) the properties of the colormap of the selected source stack are taken over to the averaged stack. Changes of the colormap of the selected stack are adapted dynamically to the colormap of the result stack. If sixteen fold oversampling is activated, the range of the values of result stack corresponds to the range of the source stack multiplied by 16. Both settings can only be selected if an average stack is built.

If the window of the selected stack contains more than one stack, all settings are taken over to the other result stacks. By clicking "OK" a new window is created which contains the

new stacks. On every update of the source stack the result stack is recalculated.

After opening the plugin dialog again (with 'Ctrl e'), the setting of the 'adjust colormap' checkbox can be changed; the 'use oversampling' checkbox can only be activated at the initial start before new stacks are created. Changing the colormap setting is only possible if an average stack is built.

You can also select another source window. If you select a stack from another window, a new result window containing the averaged or added stacks of the new source will be created. After selecting a new window it is possible to set "Average" or "Add" and to activate the sixteen fold oversampling. If a stack of the same window as the old source is selected these settings remain deactivated and no new window is created.

## 10.16 Phasorplot

The phasorplot plugin uses the lifetime profiles of a TDC datastack to compute a phasor. This plugin can be found inside the Inspector Menubar in Analysis→Flim on the tab "Phasor Analysis".

### 10.16.1 Profile basics

The lifetime profiles differs in four variables mainly. The offset, which is caused by noise during the measurement and leads to a higher count rate, so the profiles seems to be lifted up like in figure 142.

Different delays could be caused by cutting a part of the stack out (for example with the Truncate Stack Tool) for further analysis. It leads to a right or left shift of the whole profile like in figure 143. For further information about the delay see "Get a Delayvalue" in the section 10.16.4.

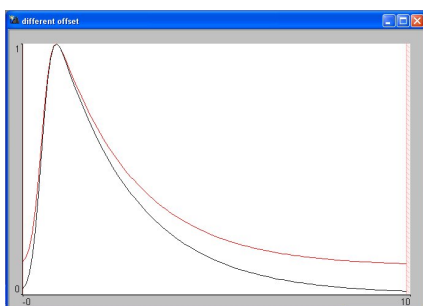


Figure 142: Different offsets.

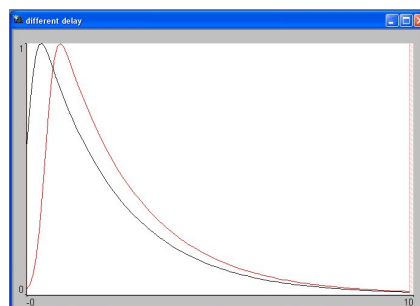


Figure 143: Different delays.

Profiles with different variances looks different in its width. Like in figure 144.

Different lifetimes are shown on figure 145.

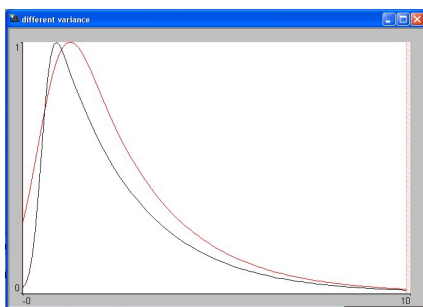


Figure 144: Different variances.

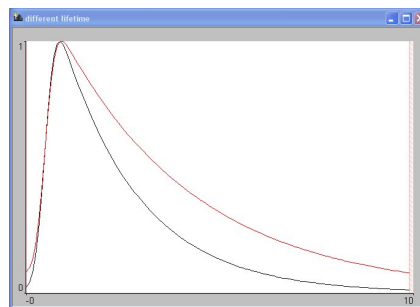


Figure 145: Different lifetimes.

The offset, delay and variance are generally identical for all profiles inside a datastack.

### 10.16.2 Start



Figure 146: Upper part of the phasorplot dialog.

At the top of the phasorplotplugin dialog are three controls (see figure 146). The button in the middle allows you to get a pipette with which you can choose a flim datastack by clicking on it. After that procedure the edit field will show the name of your selected datastack. The watch check box can be used to update the phasorplot during a measurement if it's checked.

### 10.16.3 Plotproperties

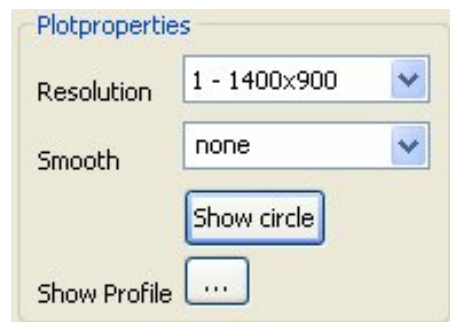


Figure 147: Plotproperties in the phasorplot dialog.

The section Plotproperties on the phasorplot dialog tab allows to change properties of the phasorplot (see figure 147). The first control afford to change the resolution of the phasorplot. The following values are available:

350x225  
700x450  
1400x900  
2800x1800

The value 1400x900 is the default resolution.

By using the controll "smooth" it is possible to smooth the phasorplot with a weighted average smooth. Internally not the phasorplot itself will be smoothed but the data behind - the sourcestack. Values are:

3x3  
5x5  
none

The default value is none.

The button "Show circle" allows to get a semicircle. All data with monoexponential lifetime have their plot on this semicircle. Figure 148 shows a phasorplot with semicircle and monoexponential lifetimes.

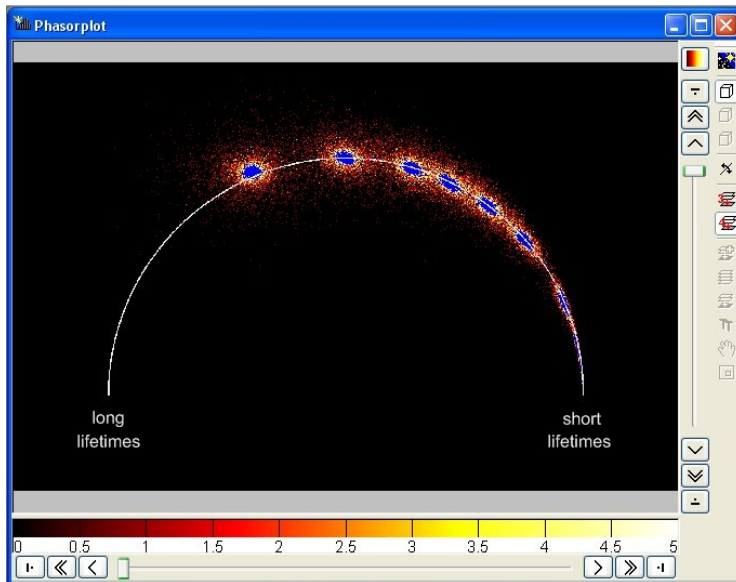


Figure 148: Phasorplot with different lifetimes.

The last button in the Plotproperty section is the profile button. By pressing this button a pipette appears to choose a lifetime profile. This creates a new stack inside the phasorplot, which contains a small circle marking the position of the profile inside the phasorplot (see figure 149). To delete this stack you can switch on the gallery mode (Ctrl+g), activate the favored stack and press Del (see also the shortcut list section 11). For corrections of the profiles' phasorplot see section 10.16.4.

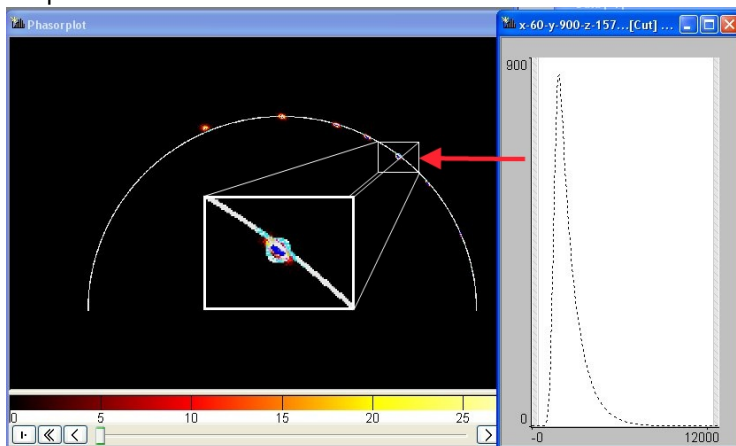


Figure 149: Phasorplot with a profilecircle.

#### 10.16.4 Phasorcorrection



Figure 150: Phasorcorrection area.

In most cases a Phasorcorrection might be very useful or even necessary. The Phasorcorrection area (figure 150) consists of two possibilities to correct a phasorplot. The first one is the offsetcorrection. Figure 151 shows the phasorplot of profiles of the same lifetime but different offsets. The offset causes a linear shift of the phasorplot. To solve this problem automatically you can use the "Correct" button, which computes an approximated offset value. It is also possible to enter an offset value manually and press the "New" button.

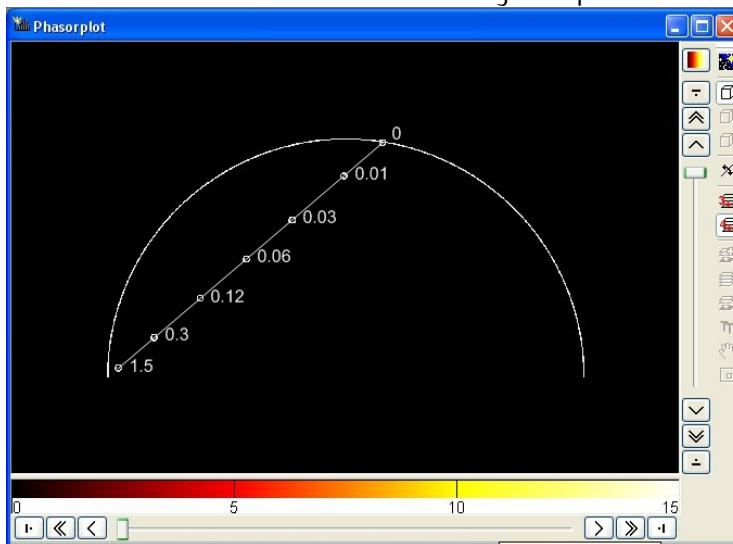


Figure 151: Different offsets.

The next correction method is the delaycorrection. A bad delayvalue causes a rotation of the phasorplot. Figure 152 shows the phasorplot of some profiles of the different lifetimes, same offset and different delay values. All small circles on a semicircle have the same delay but different lifetimes. As well as the offsetcorrection, you can solve the problem with the "Correct" button or by enter a delay value and press "New".

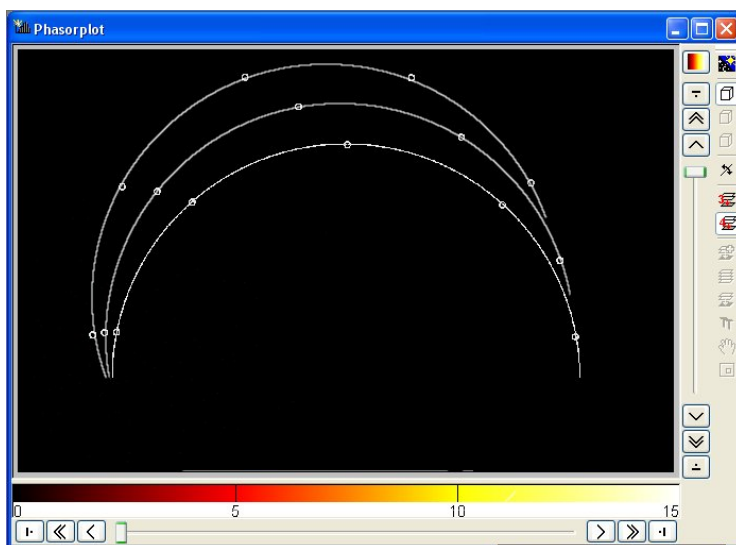


Figure 152: Different delays.

The last checkbox button "Correct Profile" causes a correction of the phasorplot also for the profiles picked with the "Show Profile" button from the Plotproperty area.

**Get a Delayvalue** A delay value could be detected with the help of the Flim Unmixing plugin. At first it is to mention that there is a dependency between the variance of a profile and an adequate delayvalue for the phasorplot (see figure 153 for an example).

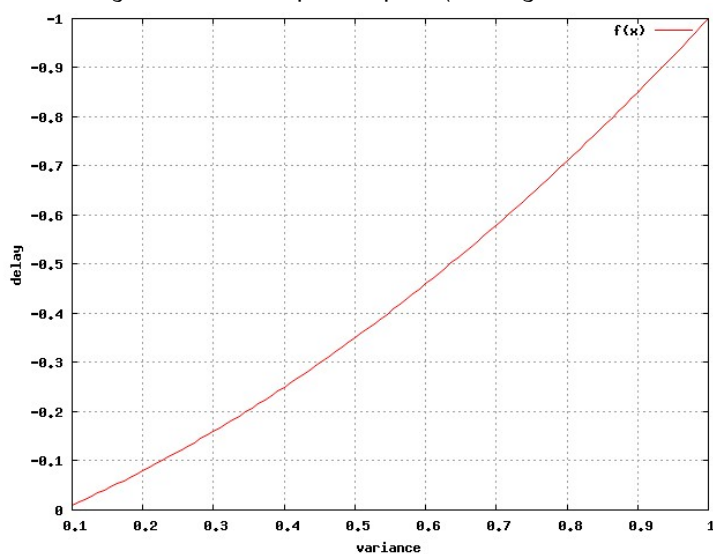


Figure 153: Variance delay dependency for phasorplot.

With the help of this function and the Flim Unmixing plugin you can try to calculate the delay manually. Figure 154 shows a manual fitting with the Flim Unmixing plugin. The dedected values are:

lifetime: 2.4  
 delay: 1.4  
 variance: 0.2  
 offset: 0.005

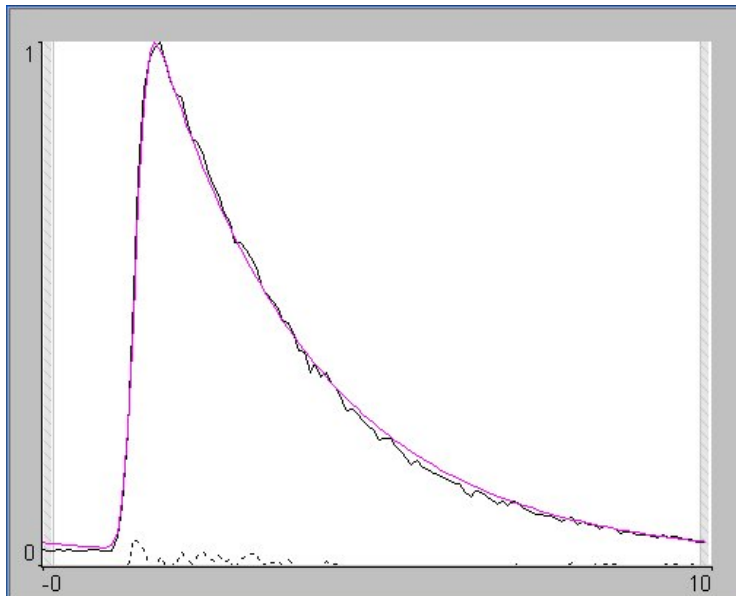


Figure 154: Flim Unmixing manual fitting.

Taking into account that a good delay for the variance of 0.2 would be  $-0.08$ , the correction value must be  $0.08 + 1.4 = 1.48$ . The Figures 155 and 156 show the difference between no correction, an automatic correction and a manual correction with the Flim Unmixing plugin for some example data. Thereby the difference between the automatic and the manual correction depends on the data.

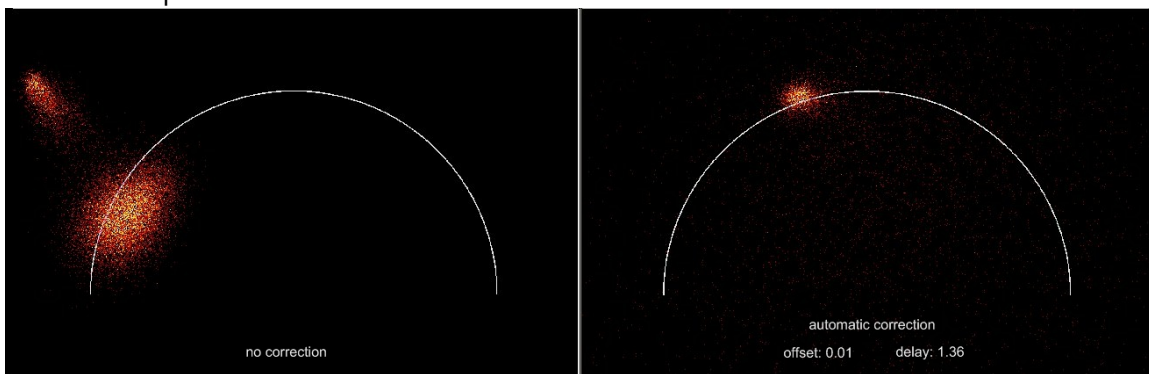


Figure 155: Two phasorplots without and with phasorcorrection.

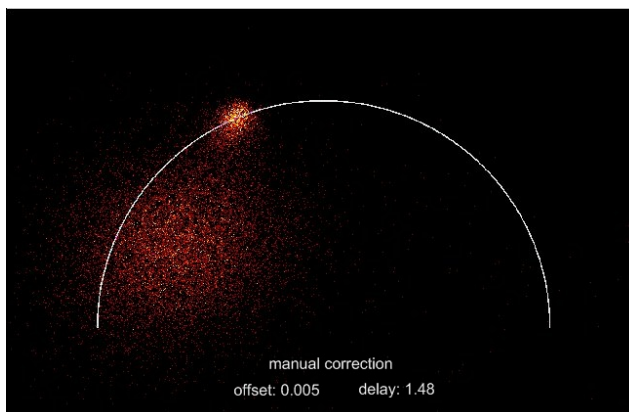


Figure 156: A phasorplot with manual phasorcorrection.



### 10.16.5 Data

The "Data" area contains the two buttons Sum and Histogram.

The Sum-button creates a summation stack of the source stack like the "plus"-button in the document toolbar in section 5.1.3. However if an area inside the phasorplot is bounded by a rectangle the corresponding pixels inside the summationstack are set to a higher intensity (see figure 157).

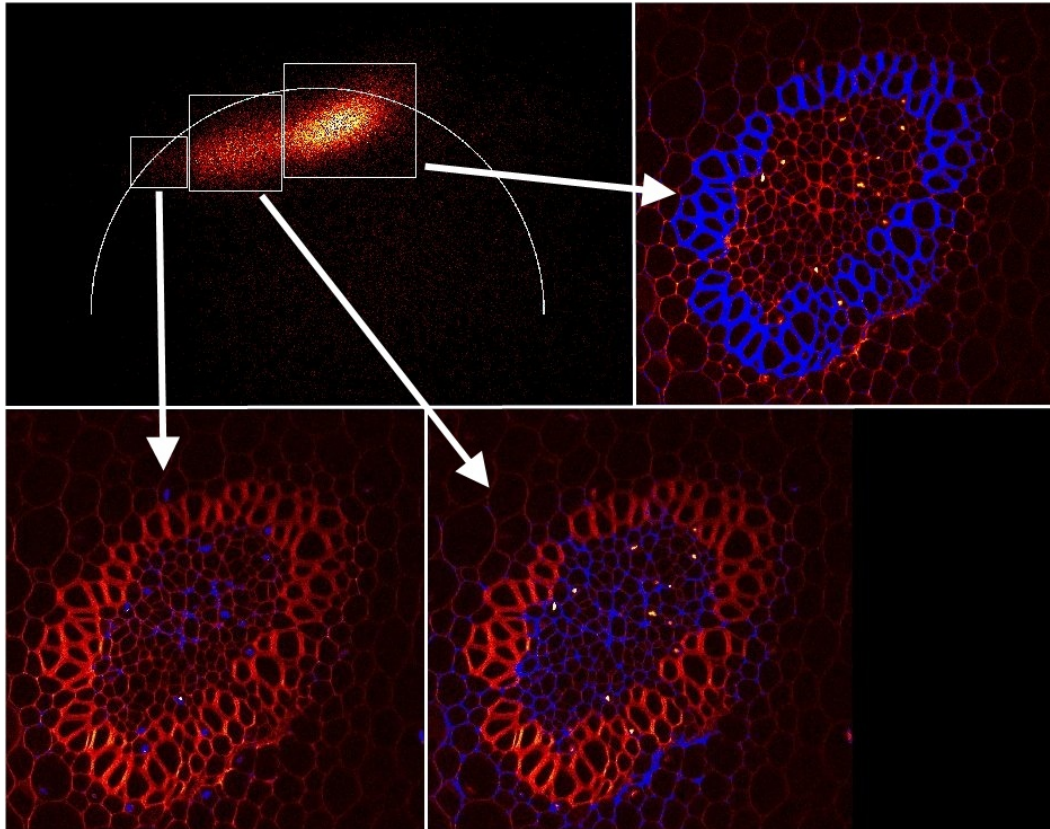


Figure 157: The Phasorplot with different summation stacks.

The Histogram-button opens a histogram of the summation stack. The horizontal axis represents countvalue of the pixels. The vertical axis is the amount of these pixels. Figure 158 shows the histogram for the adjacent phasorplot. As one can see there are a lot of pixels with a small count value, represented by a peak at the beginning of the histogram. Small counts often represents the noise inside the stack. By moving the left and right limit borders of the histogram all pixel with counts outside this border can be deleted out of the phasorplot (see figure 159).

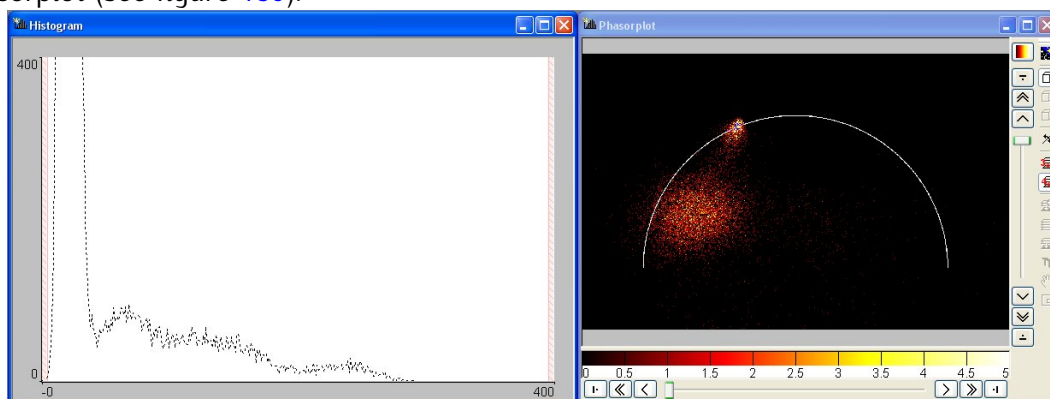


Figure 158: Histogram with borders.

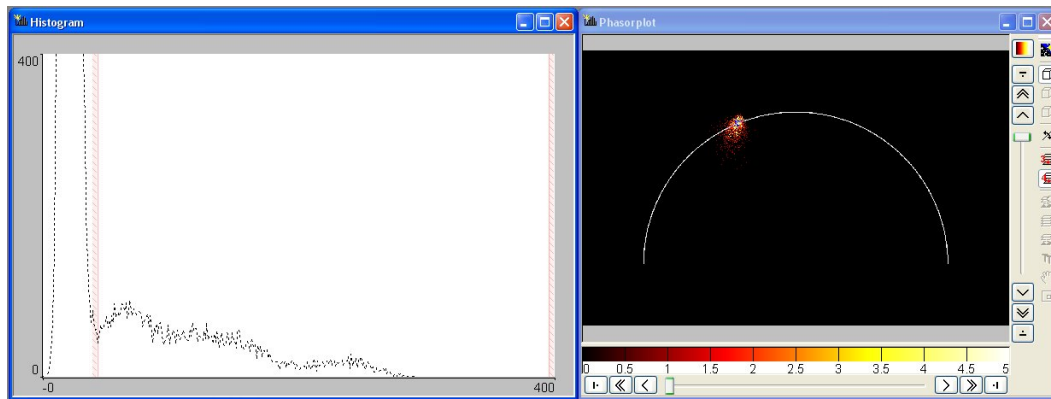


Figure 159: Histogram with shifted border and phasorplot with deleted pixels.

### 10.16.6 Create Stack

With the area "Create Stack" a datastack can be divided into its different lifetimes by pigmenting them with different colors. To do this, the user has to create a rectangle inside the phasorplot and check one of the checkboxes called 1st, 2nd or 3rd Roi like in figure 160. This represents the first chosen area of the source datastack it is possible to choose three altogether. To get a second area in a different color, a second rectangle of a different area is needed. It is important to create a new rectangle instead of just moving the old one to the right position. For the third checkbox it is also possible to choose no rectangle. This leads to the situation, that all areas inside the phasorplot but outside the rectangles will be represented by the last color. After using the "Create" button a result will have been computed. An example is shown in figure 161.

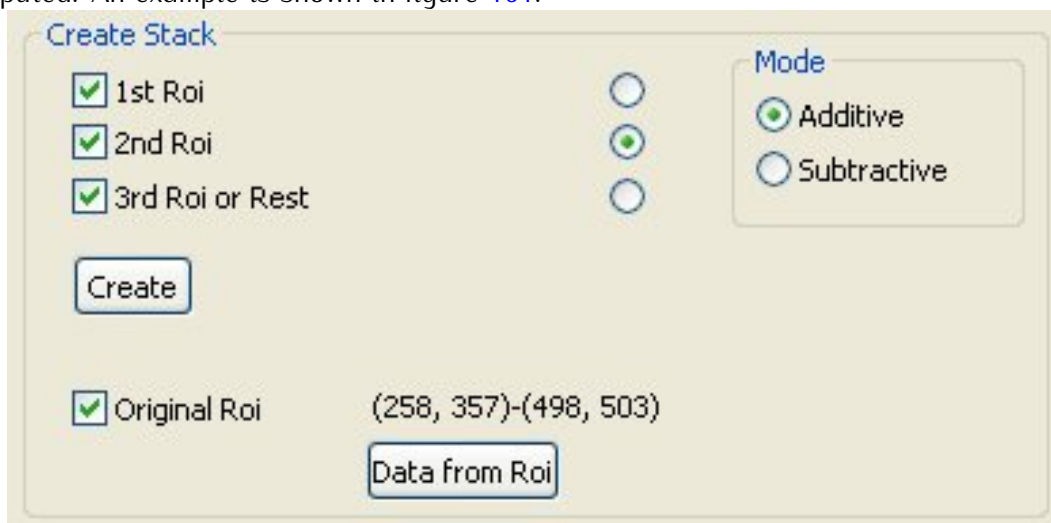


Figure 160: The "Create Stack" area inside the phasorplot.

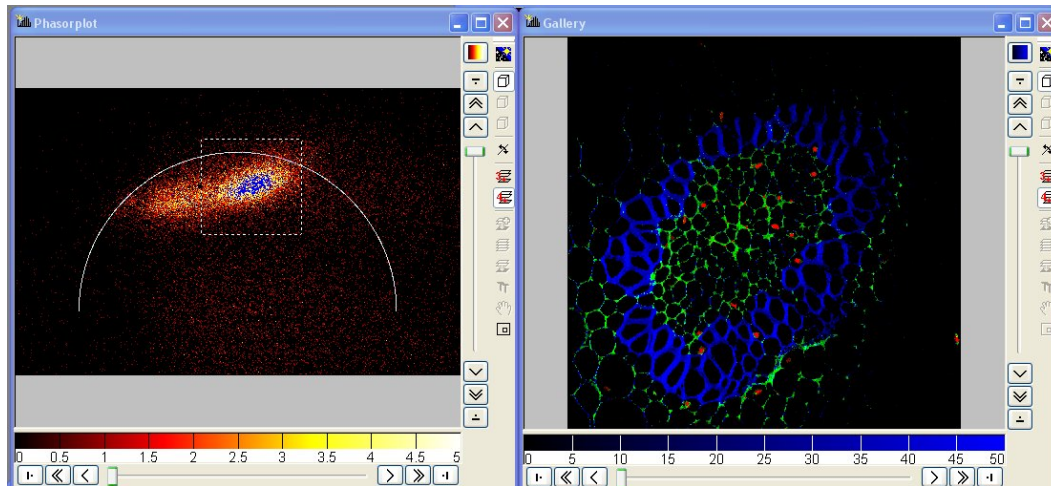


Figure 161: The result of the Create Stack operation in additive mode.

After creating the new stack with different colorareas it is also possible to change the position of the rectangles inside the phasorplot and thereby changing the coloration of the datastack. To get the particular rectangles of the areas the readibuttons beside the checkboxes could be clicked. This is the reason why it is necessary to create a new rectangle for each chosen checkbox earlier. The last possibility to modify this stack is to change its mode. The additive mode leads to different colors for each rectangle inside the phasorplot (see 161), pixels outside this rectangles are not shown. In the subtractive mode these pixels are colored gray (see 162). To get a good result in the subtractive mode it is necessary to equalize the colormaps of all three datastacks (see 11);

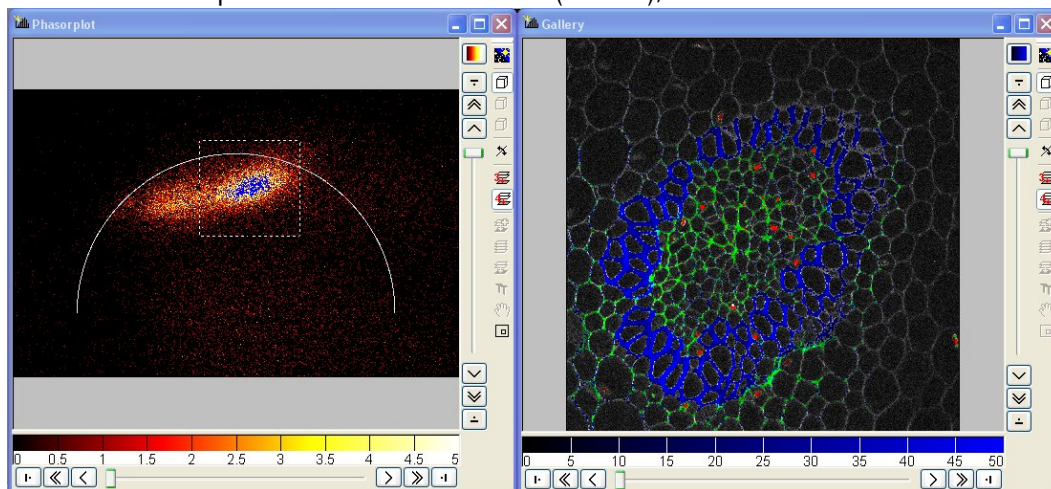


Figure 162: The result of the Create Stack operation in subtractive mode.

The last option inside the "Create Stack" area is to get a part of the source datastack for further analysis. For this we need to create a rectangle inside the phasorplot and check the checkbox called "Original Roi" and press the button Data from Roi. The result for an example is shown at figure 163.

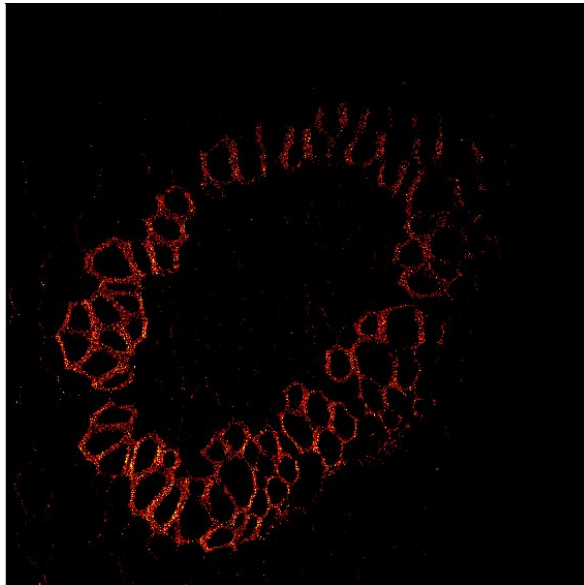


Figure 163: Part of the original source stack.

## 10.17 Laser Adjustment

This plugin is controlled by a xml-file located inside the global Config-directory. The xml-file is called "PDQ80S1.xml". If isn't present and Imspector can't find it, Imspector is going to create a new one, with some example data.

### 10.17.1 Thorlabs

The following listing shows an example for the xml-file.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <SENSORDATA>
4     <HUBS>2</HUBS>
5     <COORDINATEDLG>
6         <NAME>Beamshaper</NAME>
7         <SENSOR name="Sensor 1">
8             <ORIENTATION>90</ORIENTATION>
9             <SENSORID>0</SENSORID>
10        </SENSOR>
11        <SENSOR name="Sensor 2">
12            <ORIENTATION>0</ORIENTATION>
13            <SENSORID>1</SENSORID>
14        </SENSOR>
15    </COORDINATEDLG>
16    <COORDINATEDLG>
17        <NAME>Scanhead</NAME>
18        <SENSOR name="Sensor 3">
19            <ORIENTATION>90</ORIENTATION>
20            <SENSORID>2</SENSORID>
21        </SENSOR>
22        <SENSOR name="Sensor 4">
23            <ORIENTATION>180</ORIENTATION>
24            <SENSORID>3</SENSORID>
25        </SENSOR>
26    </COORDINATEDLG>
27 </COORDINATEDLG>

```

```

28         <NAME>OPO-Beamshaper</NAME>
29         <SENSOR name="Sensor 5">
30             <ORIENTATION>0</ORIENTATION>
31             <SENSORID>4</SENSORID>
32         </SENSOR>
33         <SENSOR name="Sensor 6">
34             <ORIENTATION>0</ORIENTATION>
35             <SENSORID>5</SENSORID>
36         </SENSOR>
37     </COORDINATEDLG>
38 </SENSORDATA>

```

### Description of the tags:

- <SENSORDATA>: This tag contains all other instructions.
- <HUBS>: At this tag the number of hubs is mentioned. If you got more than four 4-Q-Diodes, than you got two Hubs and the number inside this tag must be 2 else it must be 1.
- <COORDINATEDLG>: This tag contains the Name of the coordination window and the sensors shown on this window. It is possible to create four coordination dialogs.
- <NAME>: The Name of the coordination window.
- <SENSOR name="Sensor">: One Sensor shown on the coordination window. The Name can be chosen freely.
- <ORIENTATION>: With this tag the displayed orientation of the 4-Q-Diodes can be changed. Valid values are 0, 90, 180, 270.
- <SENSORID>: Every sensor got his own sensorid, it starts at 0 and increments to the number of 4-Q-Diodes - 1.

### 10.17.2 Lavisision Biotec

The xml configuration for the Lavisision Biotec sensors are similar to the configuration of the Thorlabs sensors. The following listing shows the differences.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <SENSORDATA>
4     <COM>1</COM>
5     <SHOW>1</SHOW>
6     <COORDINATEDLG>
7         <NAME>Beamshaper</NAME>
8         <SENSOR name="Sensor 1">
9             <DEVICE>1</DEVICE>
10            <BUS>1</BUS>
11            <ORIENTATION>90</ORIENTATION>
12            <SENSORID>0</SENSORID>
13        </SENSOR>
14        <SENSOR name="Sensor 2">
15            <DEVICE>2</DEVICE>
16            <BUS>1</BUS>
17            <ORIENTATION>0</ORIENTATION>
18            <SENSORID>1</SENSORID>
19        </SENSOR>
20    </COORDINATEDLG>

```

```

21     <COORDINATEDLG>
22         <NAME>Scanhead</NAME>
23         <SENSOR name="Sensor 3">
24             <DEVICE>0</DEVICE>
25             <BUS>2</BUS>
26             <ORIENTATION>90</ORIENTATION>
27             <SENSORID>2</SENSORID>
28         </SENSOR>
29         <SENSOR name="Sensor 4">
30             <DEVICE>1</DEVICE>
31             <BUS>2</BUS>
32             <ORIENTATION>180</ORIENTATION>
33             <SENSORID>3</SENSORID>
34         </SENSOR>
35     </COORDINATEDLG>
36 </SENSORDATA>

```

### Description of the tags:

- <COM>: Comport of the sensors.
- <SHOW>: This tag is optional. If it's in the file the bus and device addresses of the sensors will be shown inside imspecter.
- <DEVICE>: Devicenumber of this sensor (between 0 and 15). This tag is optional but recommended. Use the SHOW tag to let imspecter show the bus and device addresses for the sensors for help.
- <BUS>: Busnumber of this sensor (between 1 and 3). This tag is optional but recommended. Use the SHOW tag to let imspecter show the bus and device addresses for the sensors for help.

## 11 List of Shortcuts

Shortcut	Environment	Explanation
Alt+0	main	Show Workspace
Ctrl+n (n=1..9)	image	n gallery columns
Ctrl+x/c/v	all	Cut/copy/paste
Alt+c	image	Open colormap manipulation window
Ctrl+Alt+c	image	Copy colormaps of active window
Ctrl+Alt+v	image	Apply copied colormaps to active window
Ctrl+d	graph	Display Style ...
Ctrl+e	image	Open plugin of document
Ctrl+g	image	Toggle gallery mode
Ctrl+Shift+g	all	New empty graphwindow
Ctrl+h	image	Set hidden axis
Ctrl+Alt+h	main	Select new help directory
Ctrl+k	image	Keep Colormap Scaling
Ctrl+l	graph	Set graph limits from current selection
Ctrl+m	main	Comment popup
Ctrl+o	main	Import Data
Ctrl+p	main	Print
Ctrl+r	image	Rename documents
Ctrl+Shift+r	image	Set as New Roi
Ctrl+s	image	Save file
Ctrl+Shift+s	main	New stack window
Ctrl+t	image	Change Stack size
Ctrl+q	image	Equalize Colormap
Ctrl++/-	image	Zoom in/out
Ctrl+Shift+c/v	image	Copy/paste selection rectangle (not its contents)
Del	image	Delete stack
arrow keys	image	Select stacks
Ctrl+arrow up/down	image	Move in line or rectangle selection history
F1	all	Get context help if available
F10	image	Fit Min
Ctrl+F10	image	Fit Min image
Alt+F10	image	Fit Min Max image
Ctrl+Shift+F10	image	Open Document Menu

F5	image	Fit image
F9	image	Fit Max
Ctrl+F9	image	Fit Max image
Alt+F9	image	Fit Min Max
PageUp/Down	image	Next/previous slice (right slider)
Ctrl+PageUp/Down	image	Next/previous layer (bottom slider)
Shift+PageUp/Down	image	Next/previous slice (jump ten)
Ctrl+Shift+PageUp/Down	image	Next/previous layer (jump ten)
Tab	main	Toggle life dialogs
Ctrl+Tab	main	Show Documentlist (choose document by holding Ctrl and repeat pressing Tab)
Ctrl+w	image	Close document
<b>Shortcut</b>	<b>Environment</b>	<b>Explanation</b>
Ctrl+left/right mouse button	image	Drag and drop a stack from one document in an other (copy)
Ctrl+Alt+left mouse button	image	Drag and drop a stack from one document in an other (duplicate)
Ctrl+mouse wheel	image	Zoom in/out
mouse wheel	image	Next/previous slice (right slider)
Shift+mouse wheel	image	Next/previous layer (bottom slider)



## 12 Python Script Editor

### 12.1 Introduction

Inspector includes a scripting package, which is based on the freely available programming language *Python* (<http://www.python.org/>).

Inspector integrates the *Python* language and extends it with some own commands to give access to its data stacks and devices.

(see: [Inspector lvbt Python Module](#)).

Use `print lvbt.info()` to get documentation or `help(lvbt)` to get latest auto generated documentation.

With the *Python Script Editor* the user can write own algorithms for controlling the measuring process, data analysis or data manipulation. The measurement can be configured within the script editor by adjusting all parameters of the integrated hardware. Different measurements - with different settings - can be started one after the other. Data analysis or data manipulation can be done automatically after a measurement finishes.

For analysis and manipulation, the data can be transformed into "*numpy arrays*":

*Numerical Python and Scientific Python are fundamental packages for scientific computing in Python.*

*To use them in Inspector the packages must be installed manually !*

*Further information see [How To Install Numpy](#)  
<http://www.numpy.org/>, <http://www.scipy.org/>*

The *Python Script Editor* in Inspector uses *Python* version 2.7. The basic python packages are installed automatically, when selected in the setup programm. [How To Install Numpy](#) shows how to install numerical python or other python packages.

### 12.2 The Script Editor

To open the *Python Script Editor* click the python symbol in the Inspector toolbar.



Figure 164: To open the editor click this button.

Figure [165] shows the script editor window. Major component is the editor itself. The python script is entered here.

The script editor also consists of a toolbar, a menu section and a message window.

The user can switch between output and editor window with the tabs at the bottom. Message output is done with the python **print** command.

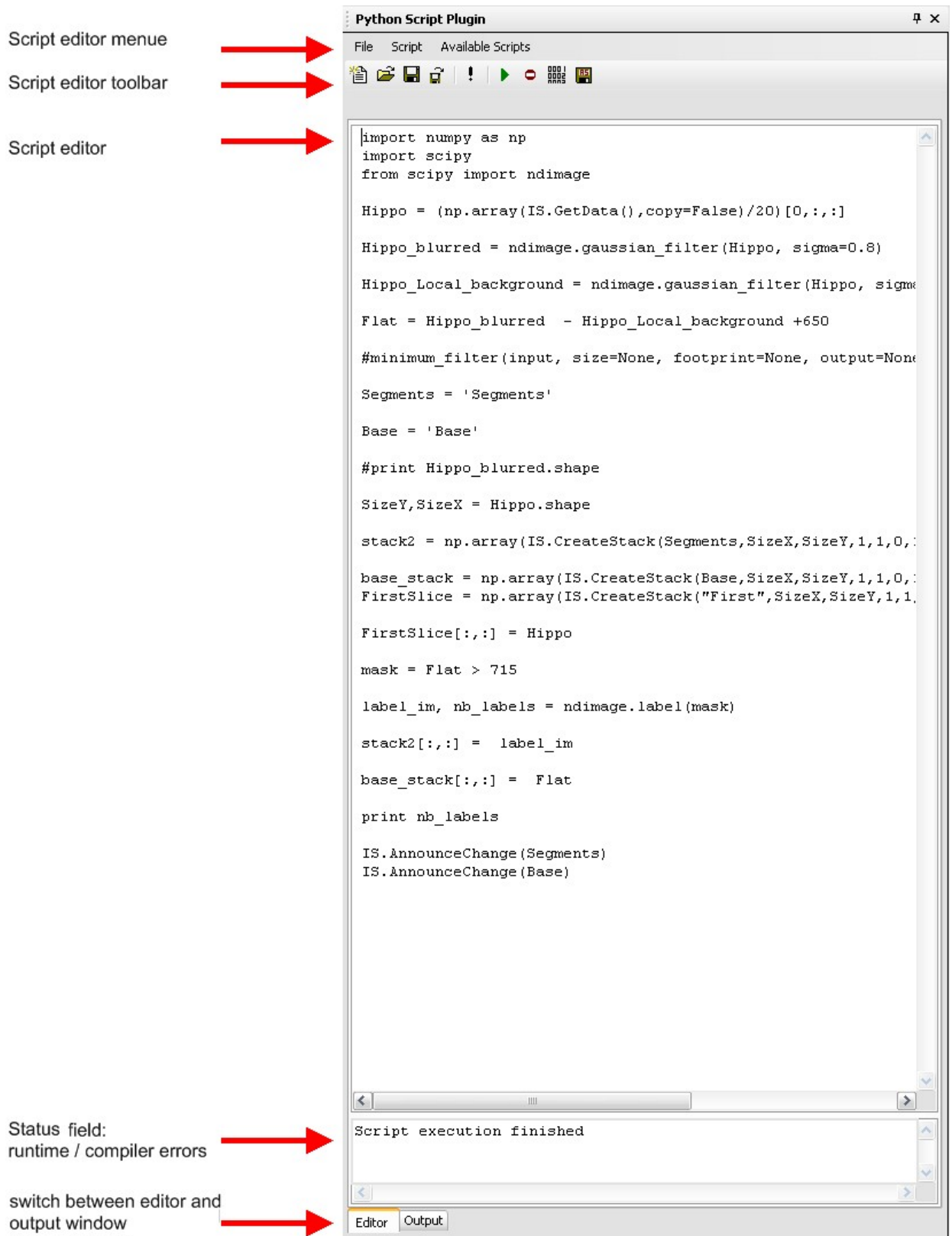













Figure 165: The Python Script Editor.





Figure 166: Toolbar.

Toolbar explanation:

1.  Create a new script
2.  Open an existing script from disk
3.  Store script into inspector document
4.  Export script to disk
5.  Check syntax of script without running
6.  Run the script
7.  Stop script execution immediately
8.  Show line numbers beside script editor
9.  Activate autosave script
10.  Watch script file

With  different python scripts can be stored into the same Inspector document. The different scripts can be accessed in the menu "Available Scripts". To edit this list of scripts click "Available Scripts" → "manage scripts".

When autosave script  is activated the current script is saved to disk every time the script execution is started. The output file can be specified in the menu "Script" → "autosave script" → "select autosave path".

Watch script file . Click this button to select a python script from disk. The file is automatically reloaded when the script is started. Use this feature if you want to use an external editor.

### 12.3 Inspector lvbt Python Module

For analysis, data access or measurement settings Inspector provides a python module called **lvbt**. This module includes several functions and classes that permit direct access to data stacks or hardware devices.

The lvbt python module can also be used in the [Python Virtual Device](#)

Information about the deprecated **IS** module see: [Inspector IS Module](#).

Classes & Functions

### 12.3.1 Overview

Functions:

- [[lvbt.info](#)]
- [[lvbt.createStack](#)]
- [[lvbt.createGraph](#)]
- [[lvbt.getROIs](#)]
- [[lvbt.devices](#)]

Classes:

- [[lvbt.adaptiveOptics](#)]
- [[lvbt.andorMosaic](#)]
- [[lvbt.autosave](#)]
- [[lvbt.dataStack](#)]
- [[lvbt.document](#)]
- [[lvbt.horizontalTwoPhoton](#)]
- [[lvbt.io](#)]
- [[lvbt.led](#)]
- [[lvbt.measurement](#)]
- [[lvbt.obis](#)]
- [[lvbt.scanner](#)]
- [[lvbt.table](#)]
- [[lvbt.triggerSequencer](#)]
- [[lvbt.trimscope](#)]
- [[lvbt.ultra](#)]
- [[lvbt.viewer](#)]
- [[lvbt.viewer.slider](#)]

### 12.3.2 Global Functions

**lvbt.info**

lvbt.info()

| Returns info message

**lvbt.createStack**

lvbt.createStack(string name, numpy array data[, window no.]

| Creates new Inspector DataStack and returns dataStack object [->[lvbt.dataStack](#)]

### **lvbt.createGraph**

lvbt.createGraph(string name, int size[, window no.])

| Creates new ImInspector Graph and returns ->dataStack object [->[lvbt.dataStack](#)]

### **lvbt.getROIs**

lvbt.getROIs([int window no.])

| Returns list with ROI positions (if no argument is given, pipette is used)

### **lvbt.devices**

lvbt.getROIs([int window no.])

| Returns list with names and types of all registered hardware devices

## 12.3.3 Classes

### **lvbt.adaptiveOptics**

lvbt.adaptiveOptics()

| Returns Adaptive Optics object.

Methods of adaptive optics object:

- getCurRange(int slider)  
| Returns current range of slider (red slider positions)
- getMaxRange(int slider)  
| Returns maximum range of slider
- maxValue()  
| Returns maximum voltage value that can be applied to mirror
- setFactory()  
| Stores current factory voltage array permanently(values stored in ini file)
- setMirror()  
| Uploads current voltage settings to mirror
- setSlider(int slider, int pos)  
| Sets current position (green slider position)

Members of adaptive optics object:

- factory  
| Data pointer to factory settings array (numpy array). Be careful with editing!
- voltage  
| Data pointer to voltage array (numpy array)

## **lvbt.andorMosaic**

lvbt.andorMosaic()

| Returns Andor Mosaic object.

Methods of Andor Mosaic object:

- clear()  
| Resets exposure memory
- setArray(numpy array exposure, bool scale )  
| Sets mosaic exposure array
- setExposureTime(float time)  
| Sets mosaic exposure time

## **lvbt.autosave**

lvbt.autosave()

| Returns Autosave object.

Static methods of autosave object:

- createNewFolder(bool activate)  
| Activate / Deactivate new folder creation for every measurement
- getLastDirectory() -> str :  
| Returns directory of last autosave measurement
- metaDataFirst(bool activate)  
| Activate / Deactivate writing meta to first file only
- setDirectory(string directory)  
| Sets autosave directory
- setPrefix(string prefix)  
| Set autosave prefix
- snapshot(bool activate)  
| Activate / Deactivate autosave for snapshot
- useCompression(bool activate)  
| Activate / Deactivate Tiff compression
- useOme(bool activate)  
| Activate / Deactivate OME tiff extension
- useShortNames(bool activate)  
| Activate / Deactivate short file names

## **lvbt.dataStack**

lvbt.dataStack(string name of data stack)

| Returns data stack object (no parameter: pipette is used)

Methods of data stack object:

- announceChange()  
| Updates view and depending stacks, call this after changing the data
- color(int color)  
Set colormap / LUT of stack (parameter color (0 - 13))
  0. default (fire)
  1. red
  2. green
  3. blue
  4. fire
  5. gray
  6. beta
  7. gamma
  8. delta
  9. cyclic
  10. grinv
  11. coldwarm
  12. warmcold
- comment()  
| Returns comment of data stack (stored in meta data)
- comment(string comment)  
| Sets comment of data stack (stored in meta data)
- export(string path, string name)  
Exports stack to Tiff  
| (single channel only, use measurement->export for multichannel export)
- isROI(int x, int y)  
| Check if position is part of ROI
- label()  
| Returns label of data stack
- label(string label)  
| Sets label of data stack
- rois()  
| Returns list of rois
- window()  
| Returns window number of data stack

Members of data stack object:

- data  
| Data pointer of stack (numpy array)
- length  
| Physical length



- lineScanPosX  
| Line Scan positions X (if linescan stack)
- lineScanPosY  
| Line Scan positions Y (if linescan stack)
- offset  
| Physical offset
- res  
| Pixel resolution

### **lvbt.document**

lvbt.document()

Returns document object

Static methods of data stack object:

- save()  
| Opens save document routine
- saveDefaultSettings()  
| Saves current document settings as default
- stacks()  
| Returns list with all all stacks of document

### **lvbt.horizontalTwoPhoton**

lvbt.horizontalTwoPhoton()

Returns Horizontal Two Photon object

Methods of data stack object:

- radius(float radius)  
| Sets radius
- radius() | Returns radius
- radiusRange()  
| Returns radius range
- radiusRange(float r1, float r2)  
| Sets radius range (returns range that is reached)
- rotation(float rotation)  
| Sets rotation
- rotation() | Returns rotation
- rotationRange() | Returns rotation range
- rotationRange(float r1, float r2 ) | Sets rotation range (returns range that is reached)

## **lvbt.io**

lvbt.io()

Returns input/output object

Static methods of io object:

- ask(string message)  
| Opens message box
- clearOutput()  
| Clears text of output window
- getDocFileName(...)  
| Returns file name of current document
- getFile()  
| Opens file select window
- input(string message) | Opens input message box and returns value (as string) given by user

## **lvbt.led**

lvbt.led()

Returns LED object

Methods of LED object:

- id()  
| Returns ID of LED
- numLED()  
| Returns number of LEDs
- off()  
| Turns off LED
- on()  
| Turns on LED
- power()  
| Returns power of LED
- power( float power)  
| Sets power of LED

## **lvbt.measurement**

lvbt.measurement(string name of measurement)

Returns measurement object (different measurements can be set up in the 'workspace' (open workspace: [ALT-0]))

Methods of measurement object:

- export(str path [,str filename])  
| Exports measurement data to multichannel OME Tiff

- `getProperty(string prop label)`  
| Returns property value
- `run()`  
| Starts measurement
- `runAsync()`  
| Starts measurement asynchronously
- `setProperty(string prop label, double value)` `setProperty(string prop label, int value)`  
`setProperty(string prop label, string value)`  
Sets property

Members of measurement object:

- `name`  
| Name of measurement

### **lvbt.obis**

`lvbt.obis()`

Returns obis object.

Members of obis object:

- `off()`  
| Turns off laser
- `on()`  
| Turns on laser

### **lvbt.scanner**

`lvbt.scanner(string device name)`

Returns scanner object (parameter name of device)

Methods of scanner object:

- `moveTo(int x, int y)`  
| [maintenance] Moves scanner to xy position
- `offset_x(int offset)`  
| [maintenance] Set X offset
- `offset_y(int offset)`  
| [maintenance] Set Y offset
- `rotate(int angle)`  
| [maintenance] Rotates scanner

## **lvbt.table**

lvbt.table(string device name)

Returns Table Object (parameter name of device)

Methods of table object:

- motor()  
| Returns Motor object (8MS table only)
- xy()  
| Returns current xy positions
- xy( tuple(x,y))  
| Moves table to given xy positions
- xy( float x, float y)  
| Moves table to given xy positions
- z()  
| Returns current z position
- (float z)  
| Moves table to given z positions  
| Position must be inside in the min max range defined in table dialog!

## **lvbt.triggerSequencer**

lvbt.triggerSequencer()

Returns triggerSequencer object

Methods of triggerSequencer object:

- activate(int port)  
| Activates port of trigger sequencer
- deactivate(int port)  
| Deactivates port of trigger sequencerr
- getActive(int port)  
| Returns active trigger port
- getAxis(int port)  
| Returns port axis name
- getData()  
| Returns data of all ports
- getName(int port)  
| Returns port name
- getPortData(int port)  
| Returns all data of specified port
- getPower(int port)  
| Returns power trigger sequencer

- `getSequence(int port)`  
| Returns port sequence
- `getSteps(int port)`  
| Returns port steps
- `refreshHardware()`  
| Refreshs hardware
- `resetHardware()`  
| Resets hardware
- `setAxis(int port, string axis)`  
| Sets axis of port
- `setKeywordPosition(int port, string key, string value)`  
| Sets Keyword position of port
- `setKeywordRange(int port, string key, string value)`  
| Sets Keyword range of port
- `setName(int port, string name)`  
| Sets name for port
- `setPower(int port, float value)`  
| Sets power of port
- `setSequence(int port, string value)`  
| Sets sequence of port
- `setSteps(int port, list steps)`  
| Sets steps of port
- `start(int port)`  
| Starts trigger sequencer
- `stop(int port)`  
| Stops trigger sequencer

### **lvbt.trimscope**

`lvbt.trimscope()`

Returns trimscope object

Methods of trimscope object:

- `getOpo()`  
| Returns GetOpo attenuator value
- `getOpo1()`  
| Returns GetOpo1 attenuator value
- `getOpo2()`  
| Returns GetOpo2 attenuator value
- `getTiSa()`  
| Returns TiSa attenuator value

- `getTiSa2()`  
| Returns TiSa2 attenuator value
- `motor()`  
| Returns Motor object
- `setOpo(float power)`  
| Sets OPO attenuator
- `setOpo1(float power)`  
| Sets OPO1 attenuator
- `setOpo2(float power)`  
| Sets OPO2 attenuator
- `setTiSa(float power)`  
| Sets TiSa attenuator
- `setTiSa2(float power)` | Sets TiSa2 attenuator

### **lvbt.ultra**

`lvbt.ultra()`

Returns ultramicroscope object

Methods of ultra object:

- `contrastParameter(bool autoparam, int offset, int median, int mean)`  
| Sets contrast parameter of Ultramicroscope
- `motor()`  
| Returns Motor object
- `test()` | Ultra test routine

Members of ultramicroscope object:

- `heatmap` | Data pointer to heatmap array (numpy array)

### **lvbt.viewer**

`lvbt.viewer()`

Returns [Image Series Viewer](#) object

Methods of viewer object:

- `importedStacks()`  
| Returns list of imported stacks of series viewer
- `numSlider()`  
| Returns number of sliders
- `stacks()`  
| Returns list with stacks of series viewer

## lvbt.viewer.slider

lvbt.slider.viewer()

Returns slider object of image series viewer

**slider = lvbt.viewer().slider(int pos)**

Methods of slider object:

- curRange()  
| Returns selected range of slider (red boundaries)
- name()  
| Returns Name of slider
- physicalOffset()  
| Returns physical offset of slider
- physicalUnit()  
| Returns physical unit of slider
- pos1()  
| Returns left position of slider
- pos1(int pos)  
| Sets position left
- pos2()  
| Returns right position of slider
- pos2(int pos)  
| Sets position right
- posCurrent()  
| Returns current (green) position of slider
- posCurrent(int pos)  
| Sets current (green) position
- range() | Returns complete range of slider

## 12.4 Script Examples

Script examples can be loaded with the *Python Examples Dialog*. It can be opened in the python menubar *Available Scripts* → *Python Examples*.

When clicking *Load script* the selected script is directly loaded into the script editor. The scripts can also be found in the *config* directory of Inspector (*.. config\examples\python*).

### 12.4.1 Script Example - Two Measurements

This script runs two different measurements alternately and saves the data of the measurement directly on disk. The two measurements must be defined in the *Inspector workspace*. This Script - among others - can be found in the *Examples Dialog*.

```
1 #
2 # Run two different measurements alternately
3 #
4 # Export data to c:/aa (as OME tiff)
5 #
```

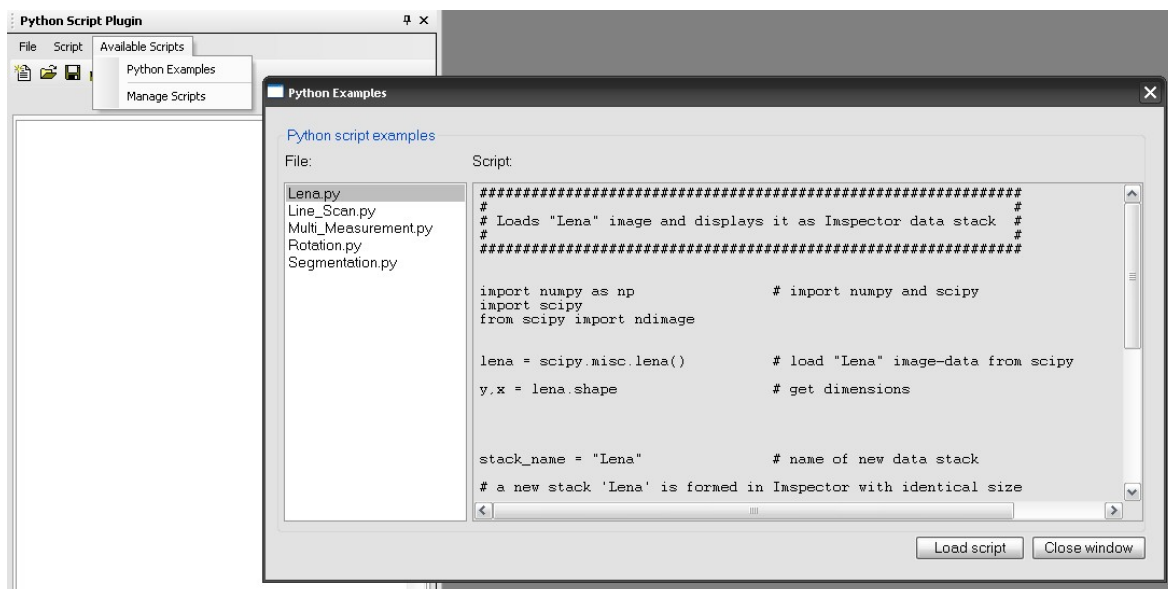


Figure 167: Python Examples.

```

6 # 2 Measurements must be defined in the Workspace![ALT-0] !
7 #
8
9 import lvbt
10 import numpy as np
11
12 output = "c:/aa"
13 iterations = 10
14
15 meas1 = lvbt.measurement("Measurement 1")
16 meas2 = lvbt.measurement("Measurement 2")
17
18 for i in range(iterations):
19
20     meas1.run() # start measurement 1
21     meas1.export(output)
22
23     meas2.run() # start measurement 2
24     meas2.export(output)

```

## 12.5 The PySerial extension

### 12.5.1 Using the PySerial extension

As with all other *Python* extensions, the *PySerial* extension must first be imported. Thus, one of the first lines of an Inspector script should be:

```
import serial.
```

Some examples:

Open port 0 at "9600,8,N,1", no timeout

```

1 import serial
2 ser = serial.Serial(0) #open first serial port
3 print ser.portstr     #check which port was really used

```



```

4 ser.write("hello")           #write a string
5 ser.close()                  #close port

```

Open port 0 at "19200,8,N,1", 1s timeout

```

1 ser = serial.Serial('/dev/ttyS1', 19200, timeout=1)
2 x = ser.read()              #read one byte
3 s = ser.read(10)           #read up to ten bytes (timeout)
4 line = ser.readline()      #read a '\n' terminated line
5 ser.close()

```

Open second port at "38400,8,E,1", non blocking HW handshaking

```

1 ser = serial.Serial(1, 38400, timeout=0, parity=serial.PARITY_EVEN,
    rtscts=1)
2 s = ser.read(100)          #read up to one hundred bytes
3                            #or as much is in the buffer

```

### Get a Serial instance and configure/open it later

```
1 ser = serial.Serial()
```

Now, a Serial instance is build and can be configured.

```

1 ser.baudrate = 19200
2 ser.port = 0
3 ser.open()
4 print ser.isOpen()      -> True
5 ser.close()
6 print ser.isOpen()     -> False

```

Be carefully when using "readline". Do specify a timeout when opening the serial port otherwise it could block forever if no newline character is received. Also note that "readlines" only works with a timeout. "readlines" depends on having a timeout and interprets that as EOF (end of file). It raises an exception if the port is not opened correctly.

### Parameters for the Serial class

```

1 ser = serial.Serial(
2     port=None,                #number of device, numbering starts at
3                               #zero. if everything fails, the user
4                               #can specify a device string, note
5                               #that this isn't portable anymore
6                               #if no port is specified an unconfigured
7                               #an closed serial port object is created
8     baudrate=9600,           #baudrate
9     bytesize=EIGHTBITS,     #number of databits
10    parity=PARITY_NONE,      #enable parity checking
11    stopbits=STOPBITS_ONE,   #number of stopbits
12    timeout=None,            #set a timeout value, None for waiting
13                               forever
14    xonxoff=0,                #enable software flow control
15    rtscts=0,                #enable RTS/CTS flow control
16 )

```

The port is immediately opened on object creation, if a port is given. It is not opened if port is None.

Options for read timeout:

```
timeout=None      # wait forever
timeout=0        # non-blocking mode (return immediately on read)
timeout=x        # set timeout to x seconds (float allowed)
```

### Methods of Serial instances

```
open()           # open port
close()          # close port immediately
setBaudrate(baudrate) # change baudrate on an open port
inWaiting()     # return the number of chars in the receive buffer
read(size=1)    # read "size" characters
write(s)        # write the string s to the port
flushInput()    # flush input buffer, discarding all it's contents
flushOutput()   # flush output buffer, abort output
sendBreak()     # send break condition
setRTS(level=1) # set RTS line to specified logic level
setDTR(level=1) # set DTR line to specified logic level
getCTS()        # return the state of the CTS line
getDSR()        # return the state of the DSR line
getRI()         # return the state of the RI line
getCD()         # return the state of the CD line
```

### Attributes of Serial instances    Read Only:

```
portstr         # device name
BAUDRATES       # list of valid baudrates
BYTESIZES       # list of valid byte sizes
PARITIES        # list of valid parities
STOPBITS        # list of valid stop bit widths
```

New values can be assigned to the following attributes, the port will be reconfigured, even if it's opened at that time:

```
port           # port name/number as set by the user
baudrate       # current baudrate setting
bytesize      # bytesize in bits
parity         # parity setting
stopbits       # stop bit with (1,2)
timeout        # timeout setting
xonxoff        # if Xon/Xoff flow control is enabled
rtscts         # if hardware flow control is enabled
```

### Exceptions

serial.SerialException Constants

parity:

```
serial.PARITY_NONE
serial.PARITY_EVEN
```

serial.PARITY\_ODD

stopbits:

serial.STOPBITS\_ONE  
serial.STOPBITS\_TWO

bytesize:

serial.FIVEBITS  
serial.SIXBITS  
serial.SEVENBITS  
serial.EIGHTBITS

### 12.5.2 Example: Create a trigger with PySerial and ImSpector

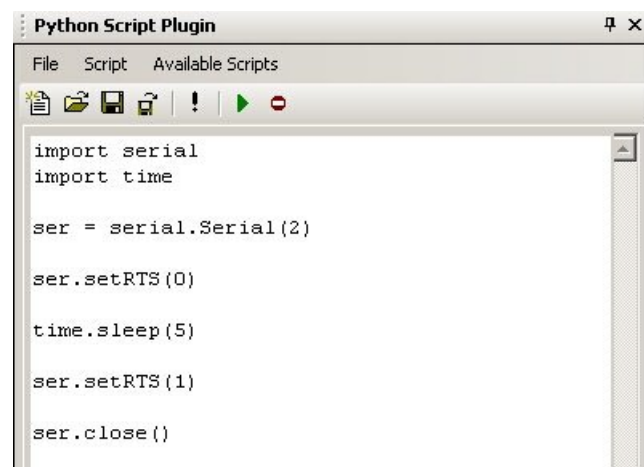


Figure 168: Open a com port and use its RTS line for a trigger signal.

In this small example, the RTS line of the serial communication interface is used to generate a trigger.

```
1 import serial    # import the serial extension
2 import time      # import the time extension
3
4 ser = serial.Serial(2) # create a serial object
5
6 ser.setRTS(0)      # set the RTS line to logical 0
7
8 time.Sleep(5)     # wait 5 seconds
9
10 ser.setRTS(1)     # set the RTS line to logical 1
11
12 ser.close()      # close the port
```

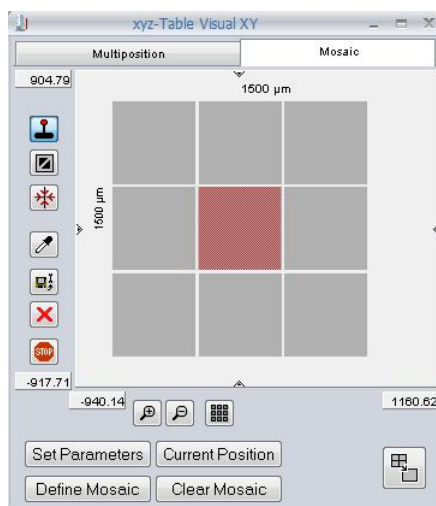
## 13 Appendix


### 13.1 How To Stitch Inspector Mosaic Image Data With Fiji

Inspector mosaic data can be stitched using the freely available image processing tool *fiji* (further information about *fiji* see: <http://fiji.sc/Fiji>). Here is shown how to acquire the data in Inspector and how to stitch it with *fiji*.

#### 13.1.1 ImSpector Set Up

1. Define the mosaic in the life dialog of the [Motorized XY Table](#).



2.  Start the measurement and acquire the data.
3. Export the data: When [Autosave](#) is used the data automatically is stored in the directory defined in the autosave settings. If autosave is not used the data must be exported manually to **OME tiff** using the [Export To File](#) function.

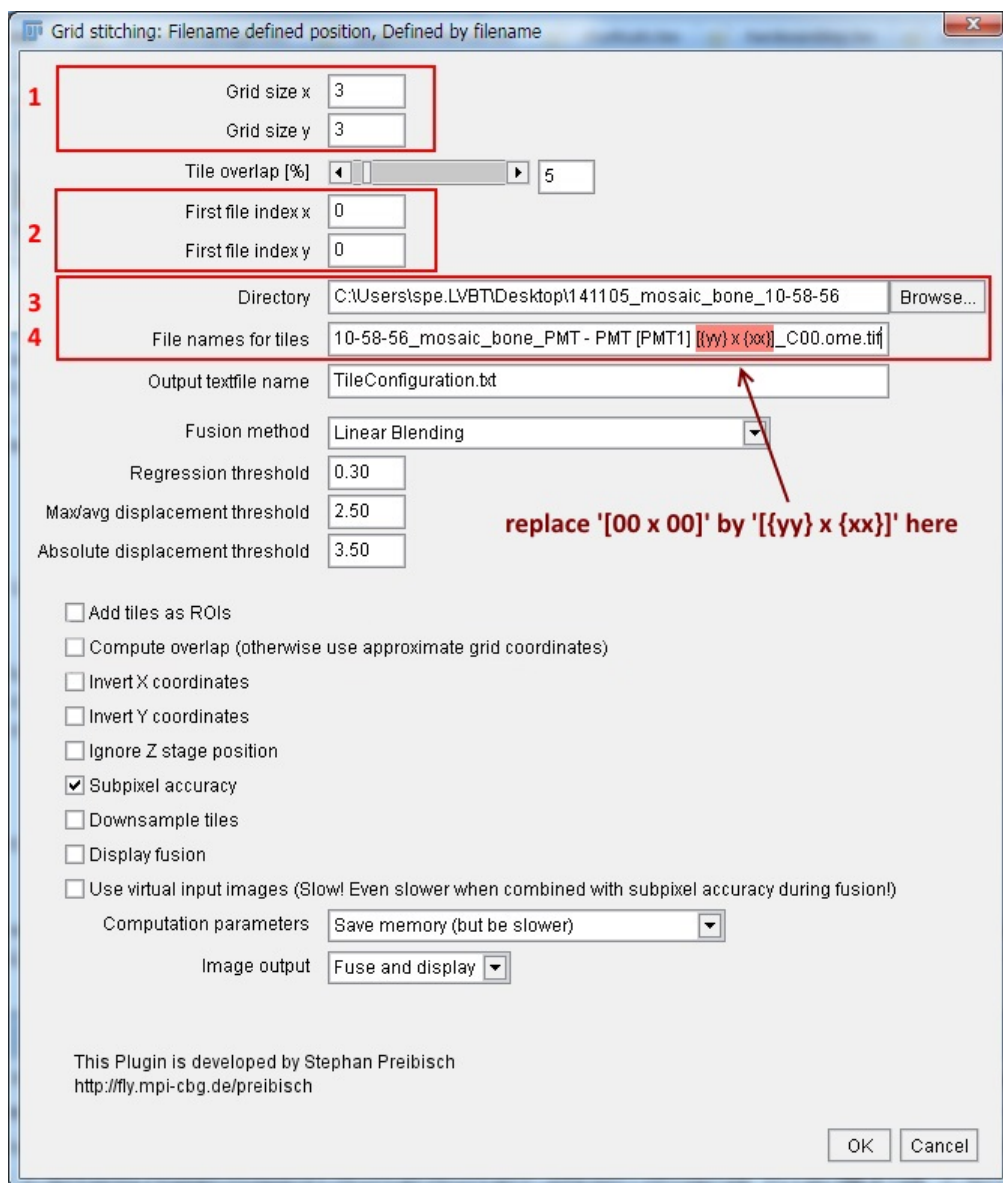
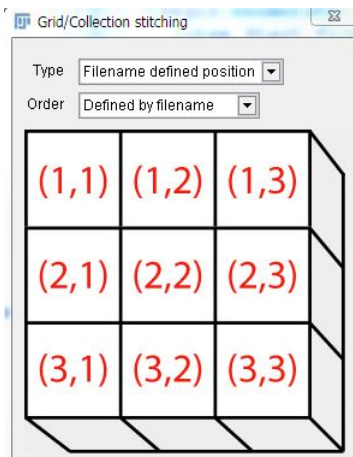
#### 13.1.2 Stitching Data With Fiji

Up to five dimensional data can be stitched with *fiji*. For the stitching this plugin should be used:

'Plugins' → 'Stitching' → 'Grid/Collection Stitching'

How to stitch:

- Start *fiji*
- Open the stitching plugin '**Grid/Collection Stitching**'
- Select Type: 'Filename defined position', Select Order: 'Defined by filename' and click 'OK'.
- A new dialog opens. Several settings have to be done here.
  1. Define Grid size used in the measurement.



2. Set 'First file index x' and 'First file index y' to 0
3. Specify directory where the tiff files are located
4. Copy & paste name of first tiff file of this series. Replace [00 x 00] by [{yy} x {xx}].  
Make sure to copy the complete file name, including the .ome.tif file extension.
5. Click 'OK' to start computation
6. To get better stitching results you can vary the computation parameter.

***Please note: The file names differ,***

- if autosave is not used or***
- if autosave is used, but it's not activated for the X and Y axis in the wizard***

***Then instead of***

[00 x 00]

[xyz – Table [0] [xyz – Table [0]]

***is used.***

***This must be replaced by***

[xyz – Table[{y}][xyz – Table[{x}]

- The plugin is developed by *Stephan Preibisch*, further information see [http://fiji.sc/wiki/index.php/Image\\_Stitching#Grid.2FCollection\\_Stitching](http://fiji.sc/wiki/index.php/Image_Stitching#Grid.2FCollection_Stitching) and <http://fly.mpi-cbg.de/~preibisch/>

## 13.2 Python

### 13.2.1 How To Install Numpy

***Numerical Python (NumPy) and Scientific Python (SciPy) are fundamental packages for scientific computing in Python.***

***To use them in ImInspector the packages must be installed manually !***

***Numpy must be installed for using the ImInspector lvbt module***

The NumPy package bundles Intel MKL library which we are not allowed to redistribute. Therefore you have to follow these steps to use NumPy and SciPy within the ImInspector Python scripting editor:

#### Simple Shortcut

***Simple shortcut for installing numpy (64 bit only):***

- download site-packages from our website: <http://www.lavisionbiotec.com/software/site-packages.zip>

- unzip everything.
- replace site-packages directory in C:\Program Files\InspectorPro\Python27\Lib with the unzipped site-packages directory.

### Standard Way

- Download NumPy: <http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>  
numpy-MKL-X.Y.Z.win-amd64-py2.7.exe
- Download SciPy: <http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy>  
scipy-X.Y.Z.win-amd64-py2.7.exe

These files must be extracted manually, so you might have to install a zipping tool:

- Download *WinRAR* or *7Zip* to extract the .exe files <http://www.7-zip.org/> or <http://www.rarsoft.com/download.htm>
- Install *WinRAR/7Zip* and keep the check with integration into the windows explorer and cascading menu.
- Extract the files: Rightclick the .exe files
  - Context Menu > WinRAR > extract here
  - Or Context Menu > 7Zip > extract here
- Open the extracted files
- Enter directory PLATLIB and copy the .egg-info files and the two folders numpy and scipy to C:\Program Files\InspectorPro\Python27\Lib\site-packages

**You may repeat this procedure for any other compatible Python package!**

### 13.2.2 Inspector IS Module

Although Inspector provides with the [Inspector lvbt Python Module](#) a python interface to its data stacks and devices, the old - deprecated - IS module is still available. These are the IS module commands:

- - `IS.RunMeasurement (measurement) [13.3.1]`
- - `IS.SaveMeasurementData (measurement, path, [Filename]) [13.3.2]`
- - `IS.SetProperty ( measurement, property, integer value) [13.3.3]`
- - `IS.SetProperty ( measurement, property, float value) [13.3.3]`
- - `IS.SetProperty ( measurement, property, string value) [13.3.3]`
- - `IS.GetProperty ( measurement, property, string value) [13.3.4]`
- - `IS.Say (message) [13.3.5]`
- - `IS.ExportToTiff (sourcePath, destinationPath) [13.3.6]`
- - `IS.GetFile () [13.3.7]`
- - `IS.GetDocFileName () [13.3.8]`

- - `IS.GetData([name])` [13.3.9]
- - `IS.GetGraph([name])` [13.3.11]
- - `IS.GetGraphs([name])` [13.3.12]
- - `IS.GetDataOfWindow(window)` [13.3.10]
- - `IS.GetStackDims([name])` [13.3.13]
- - `IS.CreateStack(name, size, type, [window])` [13.3.14]
- - `IS.CreateGraph(name, size, [window])` [13.3.15]
- - `IS.CreateProfile(window, stack, mode, rect)` [13.3.16]
- - `IS.SetLength(length[, name])` [13.3.17]
- - `IS.SetOffset(offset[, name])` [13.3.18]
- - `IS.GetStackNames()` [13.3.19]
- - `IS.GetStackNamesOfWindow(window)` [13.3.20]
- - `IS.AnnounceChange(name)` [13.3.21]
- - `IS.Ask(message)` [13.3.22]
- - `IS.GetInt()` [13.3.23]
- - `IS.GetFloat()` [13.3.24]
- - `IS.GetString()` [13.3.25]
- - `IS.MoveZ(device, position)` [13.3.26]
- - `IS.GetTablePosition(device)` [13.3.27]
- - `IS.SetTablePosition(device, x, y, max_movement)` [13.3.28]
- - `IS.ExtractROIs(name, source window, source stack, mode, [destination window])` [13.3.29]
- - `IS.GetROIPosition(window, stack)` [13.3.30]
- - `IS.GetROIPositions(window, stack)` [13.3.31]
- - `IS.GetLineScanPositions(window, stack)` [13.3.32]
- - `IS.ClearOutput()` [13.3.33]
- - `IS.SetComment(comment)` [13.3.34]
- - `IS.info()` [13.3.35]

To print a message in the output window of the script editor use the standard python **print** command!



## 13.3 Inspector Python Commands

### 13.3.1 IS.RunMeasurement(measurement)

Starts a prepared Inspector measurement.

*Syntax:* `IS.RunMeasurement(string measurement)`

*Parameter:* name of measurement.

You can define different measurements in Inspector.

In the **Inspector workspace** measurements can be added or deleted. To open the workspace click *ALT-0* or menu "View" → "Workspace".

Here, all prepared measurements are listed [169](#). If you want to run such a measurement from the script, call `IS.RunMeasurement("Measurement 1")`. You can add a measurement by right clicking on an existing measurement and selecting "New Measurement" in the context menu.

```
» IS.RunMeasurement('Measurement 1')
```

```
» IS.RunMeasurement('Measurement 2')
```

### 13.3.2 IS.SaveMeasurementData(measurement, path, [filename])

Save measurement data to specified location.

*Syntax:*

`IS.SaveMeasurementData(string measurement, string path, [string filename])`

*Parameter:*

1. string measurement: name of measurement.
2. string path: location on file system.
3. optional: [string file: name of file that is created.]

If no file name is given, a standard file name is used.

```
» IS.SaveMeasurementData('Measurement 1', 'C:\Data')
```

### 13.3.3 IS.SetProperty( measurement, property, integer/float/string value)

To change a setting in your measurement manually you can use the `IS.SetProperty` command.

*Syntax:*

`IS.SetProperty( string measurement, string property, integer/float/string value)`

*Parameter:*

1. string measurement: name of measurement
2. string property: name of property

3. string/float/int value: new value of property

**NOTE:** To change the property correctly you have to pass the correct data type as new value:

- string properties need string value.
- float properties need float values.
- integer properties need integer values.

Property names and types can be seen in the Inspector workspace. To open the workspace click *ALT-0* or menu "View" → "Workspace".

```
» IS.SetProperty('Measurement 1', 'Imager Exposure Time', 50) [170]
```

### 13.3.4 IS.GetProperty( measurement, property, string value)

Returns property value.

*Syntax:* **IS.GetProperty( string measurement, string property, string type)**

*Parameter:*

- string measurement: name of measurement
- string property: name of property
- string type: type of property: either 'string', 'int' or 'float'

```
» exp = IS.GetProperty('Measurement 1', 'Imager Exposure Time', 'int')
```

### 13.3.5 IS.Say(message)

Displays message in message box.

*Syntax:* **IS.Say(string message)**

*Parameter:* string message: text that is displayed.

```
» IS.Say('text message') [171]
```

### 13.3.6 IS.ExportToTiff(sourcePath, destinationPath)

Exports all stacks of all *.msr* files that are found in the *sourcePath* directory into tiff and saves the results in the *destinationPath* directory. For every *.msr* file that is found a new folder in the destination directory is created. If the *sourcePath* is a *single .msr* file, just this document with all of its stacks is exported to tiff.

*Syntax:*

**IS.ExportToTiff(string source path,string destination path)**

*Parameter:*

1. string sourcePath: source directory where to look for *.msr* documents
2. string destinationPath: destination directory where exported files are stored

The tiff files are saved with "ome-xml" metadata information (see: [www.openmicroscopy.org](http://www.openmicroscopy.org)).

```
» IS.ExportToTiff('C:\msrDocuments', 'C:\TiffFiles') [172]
```

### 13.3.7 IS.GetFile()

Opens a file selection window.

*Syntax:* IS.GetFile()

*Returns:* String with path of selected file.

```
» filename = IS.GetFile()
```

### 13.3.8 IS.GetDocFileName()

Returns string with file name (path+name) of current document.

*Syntax:* IS.GetDocFileName()

```
» » filename = IS.GetDocFileName()
```

### 13.3.9 IS.GetData([name])

IS.GetData() permits direct access to Inspector data stacks for analysis and manipulation. *The data can directly be transformed into numpy arrays.*

*Syntax:* IS.GetData([string stack name])

*Parameter:* None or name of data stack (as string)

If no parameter is given, the inspector pipet is used to select a data stack manually. When using the pipet the user selects on runtime which data stack should be used.

*Returns:* Pointer to image data.

The data can be transformed into a *numpy array* this way:

```
» stack = numpy.array(IS.GetData('document1'), copy=False) # returns  
pointer to stack 'document 1'
```

```
» stack = numpy.array(IS.GetData(), copy=False) # pipet is used: user  
selects stack
```

**NOTE:** Numpy arrays are upside-down!

The last dimension of the stack is stored as first dimension in numpy-array!

- 4. dimension of stack -> 1. dimension of numpy array
- 3. dimension of stack -> 2. dimension of numpy array
- 2. dimension of stack -> 3. dimension of numpy array
- 1. dimension of stack -> 4. dimension of numpy array

Example:

3D stack:

```
» import numpy as np
»
» stack = np.array(IS.GetData(), copy=False)
»
» X = stack.shape[2]
» Y = stack.shape[1]
» Z = stack.shape[0]

» Z, Y, X = stack.shape
```

4D stack:

```
» import numpy as np
»
» stack = np.array(IS.GetData(), copy=False)
»
» X = stack.shape[3]
» Y = stack.shape[2]
» Z = stack.shape[1]
» T = stack.shaoe[0]

» T, Z, Y, X = stack.shape
```

Further information about numpy see:

To use *numpy* and *scipy*, both packages must be installed manually (See [How To Install Numpy](#)).

### 13.3.10 IS.GetDataOfWindow(window)

Permits direct access to all data stacks of specified window, (stacks can be transformed directly into numpy arrays)

*Syntax:* IS.GetDataOfWindow(int window)

*Paramter:* int window: number of window.

*Returns:* Python tuple with pointer to data stacks.

**NOTE:** The window numbers can be displayed by clicking *Window* → *Show window numbers* in imspector menu bar.

The number of returned data stacks can be find out by using the python len() command!

```
» data = GetDataOfWindow(1)

» num = len(data)
» stack1 = np.array(data[0], copy=False)
```

```
» stack2 = np.array(data[0], copy=False)
```

### 13.3.11 IS.GetGraph([name])

Permits direct access to graph data.

*Syntax:* IS.GetGraph([string stack name])

*Parameter:* none or name of data stack (as string)

If no parameter is given, the impspector pipet is used to select a Graph manually.

*Returns:* tuple with: x coordinates, y coordinates, graph length, graph offset

**NOTE:** Only those data is returned that is specified by red limits in graph view.

```
» x,y,length,offset = IS.GetGraph()
```

### 13.3.12 IS.GetGraphs([name])

Permits direct access to all graphs of specified data window.

*Syntax:* IS.GetGraphs([int window])

*Parameter:* int window: window of window with graphs.

*Returns:* tuple with tuples of x coordinates, y coordinates, graph length, graph offset of all graphs in specified window.

**NOTE:** Only those data is returned that is specified by red limits in graph view.

```
» graphs = IS.GetGraphs(1)  
» for g in graphs:  
»   x,y,length,offset = g
```

### 13.3.13 IS.GetStackDims([name])

Returns pixel sizes, physical sizes and offsets of specified data stack.

*Syntax:* IS.GetStackDims([name])

*Parameter:* string name: none or name of data stack

If no parameter is given, the impspector pipet is used to select a data stack manually.

*Returns:* 3 tupels: pixel resolution, physical sizes and offsets  
Each tuple contains the sizes for each dimension.

```
» dims = GetStackDims('document 1')  
» pixel = dims[0]  
» physSize = dims[1]
```

```
» offset = dims[2]
```

**NOTE:** dimension order is upside down, according to *numpy* dimension order!

### 13.3.14 IS.CreateStack(name,size,type,[window])

Creates a new data stack and displays data in specified window.

*Syntax:* IS.CreateStack(string name, int tuple size,int type,[int window])

*Parameter:*

1. string name: name of new data stack.
2. int tuple size: x,y,z,t dimension of stack.
3. int type: data type of stack:  
0: unsigned int,  
1: long,  
2: double
4. int window: stack is shown in specified window, if empty, stack is shown in new window

*Returns:* Pointer to new data stack.

The data pointer can be transformed into numpy array.

```
» size = (256,256,100,10) # dimension of new stack  
» newStack = np.array(IS.CreateStack("test", size, 0, 0), copy=False)
```

### 13.3.15 IS.CreateGraph(name,size,[window])

Creates empty graph (attached to data stack that can be filled).

*Syntax:* IS.CreateGraph(string name,int size,[int window])

*Parameter:*

1. string name: name of graph (data stack)
2. int size: size of graph (data stack) (only one dimension!)
3. int window: stack is shown in specified window, if empty or window is not a graph view, stack is shown in new window

*Returns:* Pointer to data stack of the graph, can be transformed into numpy array.

```
» graph = np.array(IS.CreateGraph('test', 50, 1) copy = False)
```

### 13.3.16 IS.CreateProfile(window,stack,mode,rect)

Creates profile graph.

*Syntax:* IS.CreateProfile(int window,int stack,int mode,int tuple rect)

*Parameter:*

1. int window: source window
2. int stack: source stack
3. int mode: profile mode: 0: X Add, 1: YAdd, 2: Hist, 3: Z profile
4. int tuple rect: rect positions (left,top,right,bottom)

```
» rect = (110, 80, 130, 100)  
» IS.CreateProfile(0, 0, 3, rect)
```

### 13.3.17 IS.SetLength(length[,name])

Sets physical sizes of data stack.

*Syntax:* **IS.SetLength(int tuple length, [string name])**

*Parameter:*

1. int tuple with sizes for each dimension
2. string name: name of data stack

If no name is given, pipet is used.

**NOTE:** dimension order is upside down, according to *numpy* dimension order.

4D stack:

```
» newLength = (10, 100, 255, 255)  
» IS.SetLength(newLength)
```

### 13.3.18 IS.SetOffset(offset [name])

Sets offset of data stack.

*Syntax:* **IS.SetOffset(int tuple offset, [string name])**

*Parameter:*

1. int tuple with offset for each dimension
2. string name: name of data stack

If no name is given, pipet is used.

**NOTE:** dimension order is upside down, according to *numpy* dimension order.

4D stack:

```
» newOffset = (10, 100, 255, 255)  
» IS.SetLength(newOffset)
```

### 13.3.19 IS.GetStackNames()

Get Stack names of all data stack of the current document.

*Syntax:* IS.GetStackNames()

*Returns:* Tuple with stack names of current document

```
s = IS.GetStackNames()  
n = len(s) # number of stacks  
name1 = s[0] # name of first data stack of document  
name2 = s[1] # name of second data stack of document
```

### 13.3.20 IS.GetStackNamesOfWindow(window)

Get Stack names of all data stacks of specified window.

*Syntax:* IS.GetStackNamesOfWindow(window)

*Parameter:* int window: number of window.

*Returns:* tuple with stack names of specified window.

```
» names = IS.GetStackNamesOfWindow(0)  
» numStacks = len(names)  
» name1 = names[0]  
» name2 = names[1]  
» name3 = names[2]
```

**NOTE:** The window numbers can be displayed by clicking *Window* → *Show window numbers* in inspector menu bar.

### 13.3.21 IS.AnnounceChange(name)

Updates all views, that are connected to specified data stack, should be called after data has changed.

*Syntax:* IS.AnnounceChange(string name)

*Parameter:* string name: name of data stack.

```
» IS.AnnounceChange("doc 1")
```

### 13.3.22 IS.Ask(message)

Requests 'YES/NO/CANCEL' from user via dialog.

*Syntax:* IS.Ask(string message)

*Parameter:* string message: message that is shown in dialog.

*Returns:* boolean



The script is stopped if user clicks 'cancel'.

```
» IS.Ask("Compute again ???")
```

### 13.3.23 IS.GetInt()

Requests input from user via dialog.

*Syntax:* **IS.GetInt([string message])**

*Parameter:* [string message: optional message that is displayed in dialog.]

*Returns:* integer value.

```
» i = IS.GetInt () # integer input
```

### 13.3.24 IS.GetFloat()

Requests input from user via dialog.

*Syntax:* **IS.GetFloat([string message])**

*Parameter:* [string message: optional message that is displayed in dialog.]

*Returns:* float value.

```
» i = IS.GetFloat () # float input
```

### 13.3.25 IS.GetString()

Requests input from user via dialog.

*Syntax:* **IS.GetString([string message])**

*Parameter:* [string message: optional message that is displayed in dialog.]

*Returns:* string value.

```
» i = IS.GetString () # string input
```

### 13.3.26 IS.MoveZ(device,position)

Moves Z device to new position.

*Syntax:* **IS.MoveZ(string device,float position)**

*Parameter:*

1. string device: name of z device
2. float position: new position

**NOTE:** The selected position is limited by the range defined in the Z-Dialog!

```
» IS.MoveZ ('xyz-Table' , 10.0)
```

### 13.3.27 IS.GetTablePosition('device')

Get current x and y position of table device.

*Syntax:* IS.GetTablePosition(string device))

*Parameter:* string device: name of table device.

*Returns:* tuple with current table x and y position.

```
» x,y = IS.GetTablePosition('XYZ-Table Z')
```

### 13.3.28 IS.SetTablePosition('device',x,y,max\_movement)

Sets new x and y position for table device.

*Syntax:* IS.SetTablePosition(string device,float x, float y, float max\_movement)

*Parameter:*

- string device: name of table device
- float x: new x position
- float y: new x position
- float max\_movement: movement limiter for safety reasons

```
» x,y = IS.GetTablePosition('xyz-Table')  
» IS.SetTablePosition('xyz-Table', x + 100, y + 125, 500)
```

### 13.3.29 IS.ExtractROIs(name,source window,source stack,mode, [destination window])

Extract data of ROIs and displays it as new stack.

*Syntax:*

IS.ExtractROIs(string name, int source window,int source stack, int mode, [int destination window])

*Parameter:*

1. string name: name of (new) stack, data should be stored in
2. int source window: window number of source window
3. int source stack: pos of source stack in the source window
4. int mode: extraction mode: 0 = labeling, 1 = binary, 2 = copy data
5. int destination window: window no where result stack should be displayed (if empty, source window is used)

```
»IS.ExtractROIs('rois', 1, 0, 0)
```

### 13.3.30 IS.GetROIPosition(window,stack)

Get Bounding box position of current ROI.

*Syntax:*IS.GetROIPosition(int window, int stack)

*Parameter:*

1. int window: window number of source window
2. int stack: pos of source stack in the source window

*Returns:* Tuple with ROI bounding box positions (left,top,right,bottom)

```
»roi = IS.GetROIPosition(0,0)
```

### 13.3.31 IS.GetROIPositions(window,stack)

Get Bounding box position of all ROIs of specified window.

*Syntax:*IS.GetROIPositions(int window, int stack)

*Parameter:*

1. int window: window number of source window
2. int stack: pos of source stack in the source window

*Returns:* Tuple with tuples with ROI bounding box positions (left,top,right,bottom)

```
»rois = IS.GetROIPositions(0,0) »roi = rois[0]
```

### 13.3.32 IS.GetLineScanPositions([stack])

Get actual line scan positions of line scan stack.

*Syntax:*IS.GetLineScanPositions([string name])

*Parameter:* none or name of data stack (as string)

If no parameter is given, the impector pipet is used to select a data stack manually.

*Returns:* Tuple with line scan coordinates (x & y positions)

```
»» linescan = IS.GetLineScanPositions() » x = linescan[0] # x coordinates  
» y = linescan[1] # y coordinates
```

**NOTE:** returned values are physical positions! Might be transformed into pixel positions for further analysis!

### 13.3.33 IS.ClearOutput()

Clears text in output window.

*Syntax:* IS.ClearOutput()

### 13.3.34 IS.SetComment(comment)

Sets comment for all stacks of current document.

After exporting data stacks as OME tiff, this comment directly is shown in other software. Comment is stored in data stacks and can be seen in *Inspector Measurement Information*[7.9](#).

*Syntax:* `IS.SetComment(string comment)`

*Parameter:* string comment: new comment for all stacks.

```
» IS.SetComment ('TEST')
```

### 13.3.35 IS.Info()

Returns overview of inspector IS methods.

*Syntax:* `IS.Info()`

*Returns:* string with info message.

```
» print IS.Info()
```

### 13.3.36 Screenshots

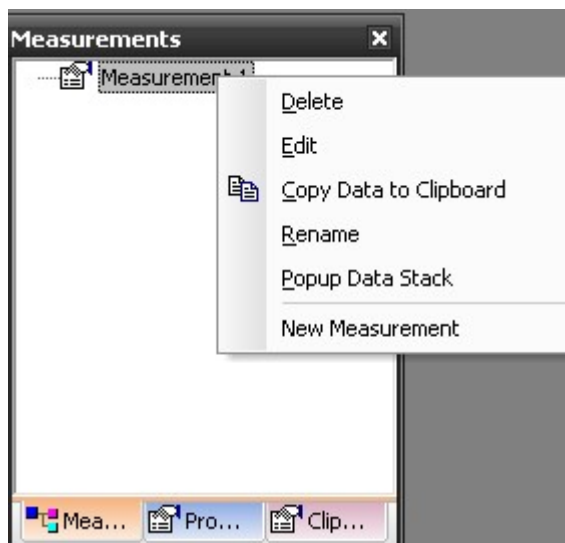


Figure 169: Workspace.

Properties		
Property	Type	Value
<input type="checkbox"/> Time Offset	float	12.00
<input type="checkbox"/> Microscope Is Active	int	0
<input type="checkbox"/> Microscope Objective Pos	int	0
<input type="checkbox"/> Endoscope End	float	0.00
<input type="checkbox"/> Endoscope Is Active	int	0
<input type="checkbox"/> Endoscope R Stepsize	float	0.00
<input type="checkbox"/> Endoscope Rot Fixpoint	float	0.00
<input type="checkbox"/> Endoscope Rot Offset [°]	float	0.00
<input type="checkbox"/> Endoscope Rot Range [°]	float	0.00
<input type="checkbox"/> Endoscope Rot Resolution	int	0
<input type="checkbox"/> Endoscope Start	float	0.00
<input type="checkbox"/> Ixon Num. img. to average	int	1
<input type="checkbox"/> Ixon T Axis	int	0
<input type="checkbox"/> Ixon T Length	float	0.00
<input type="checkbox"/> Ixon T Offset	float	0.00
<input type="checkbox"/> Ixon T Resolution	int	0
<input type="checkbox"/> Ixon X Bin	int	1
<input type="checkbox"/> Ixon X Len	float	5.12
<input type="checkbox"/> Ixon X Off	float	0.00
<input type="checkbox"/> Ixon X Pos	int	0

Figure 170: Properties.

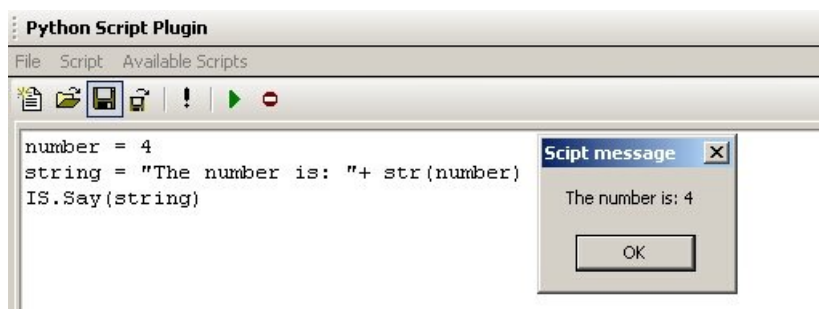


Figure 171: Sample of Say.

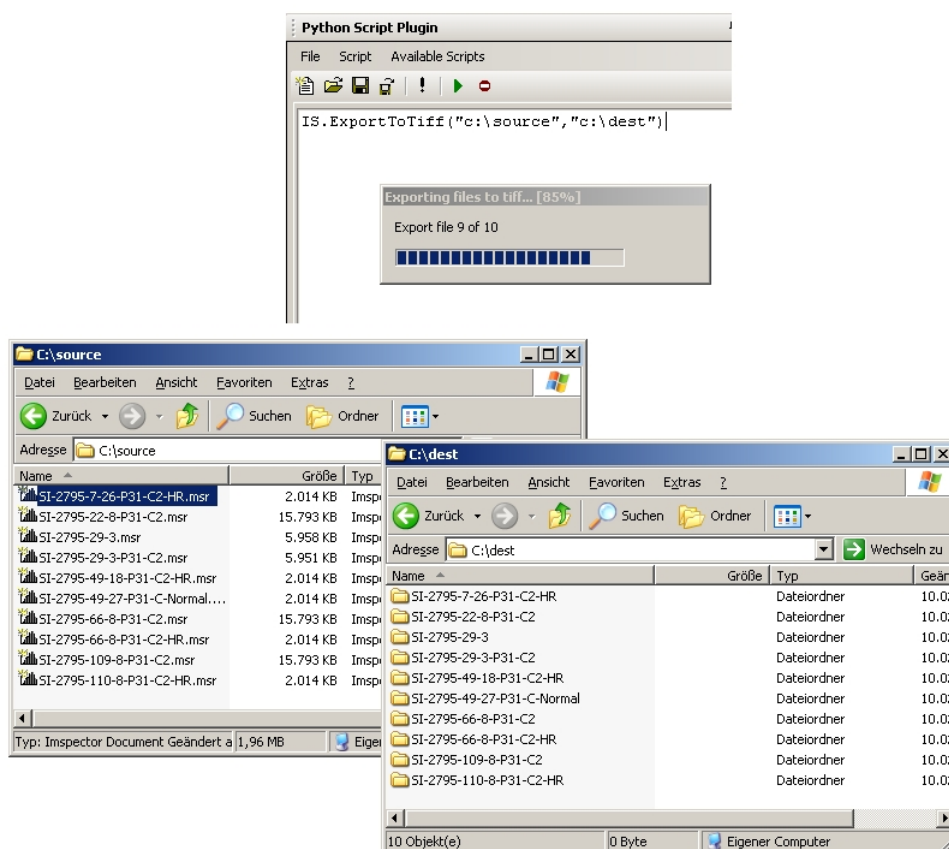


Figure 172: Example of Tiff Export: in the source folder 10 .msr files are found.

### 13.4 ImInspector Multi User Setup

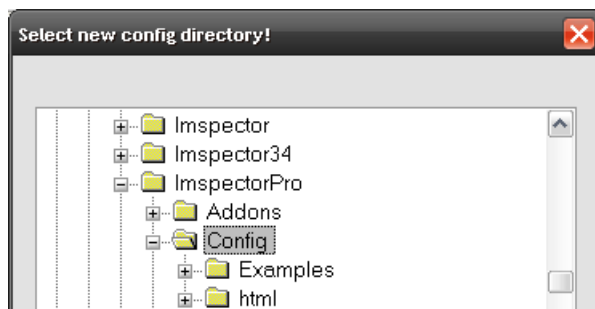
ImInspector can be used in a multi user facility. Please keep in mind some points when setting up the system.

- All microscope settings are located in an one directory (config directory).
- Each Windows user has an own configuration directory. The location of this directory is stored in the Windows Registry. It can be changed in the ImInspector software.
- For using the microscope correctly it is important that the user has fully read and

write permissions to the configuration directory.

Before creating a new user find the location of the configuration directory of the standard user:

- Log in as standard user.
- Start *Inspector*.
- The configuration directory can be located here: Click 'Edit'->'Set Config Dir':



- The dialog shows the current configuration directory. Keep in mind this location.

Then new users can be added to the windows system. Configure *Inspector* for all new users:

- Log in as new user.
- Copy the configuration of the standard user to a folder located in the user home area.
- Start *Inspector*.
- Click 'Edit'->'Set Config Dir' and select the copied directory in the home area. Click 'OK' and restart *Inspector*. DO NOT click 'hardware'->'save settings'

If all users should use the same microscope setup without the possibility to adjust anything personally, the same config directory can be used for all user accounts. This config directory must be located somewhere all users have read and write permissions! Select this directory when setting up the *Inspector* configuration for the new users.

## 13.5 axisconfig.ini templates

### 13.5.1 Burst

```
# PROPSET_XX sets properties of devices for the current mode (burst related)
#
# PROPRESET_XX sets properties of devices for all other modes (burst related)
#
#
# This mode simply encourages the use of the internal pmt axis
#
# _____
#
#
#
```

```

#
# example: PROPSET_00=PMT  M Resolution,int,30
# Devicename: PMT
# Propertyname:  M Resolution
# Type: int
# Value: 30
#
# INFO: Preset time burst to 30 steps
# Keep the 2 spaces between device and property name!
#
# _____
#
#
#
# For property names see Workspace Document (Alt + 0) in ImInspector
#
# DLGON=Dialogname1,Dialogname2,... switches the dialog on
# DLGOFF=Dialogname1,Dialogname2,... switches the dialog off
#
# RECONFIGS=REC01,RECXX allows the mode for reconfig.ini mode 01, XX
#
#
#
#
[AC01] # FIX THE INDEX NUMBER
LABEL=Timelapse Burst
AXIS_FIRST=
AXIS_SECOND=
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=Time_Burst_150.bmp
COMMENT=Time Burst Series
PROPSET_00=PMT  M Resolution,int,30

```

### 13.5.2 Pifoc

```

# PROPSET_XX sets properties of devices for the current mode (Pifoc related)
#
# PROPRESET_XX sets properties of devices for all other modes (Pifoc related)
# i.e.: reset Pifoc into serial line mode
#
# _____
#
#
#
# example: PROPSET_00=Pifoc Pifoc Measurement Mode,int,0
# Devicename: Pifoc
# Propertyname: Pifoc Measurement Mode
# Type: int

```



```

# Value: 0
#
# INFO: 0: serial communication, no trigger, normal ImSpector loop
# 1: frame trigger, normal ImSpector loop, no serial communication
# 2: fast z-stack pmt, frame trigger, skip ImSpector loop
# 3: fast linescan x-z, line trigger, skip ImSpector loop
#
# example: PROPSET_01=xy-Scanner Frap Active,int,0
# Devicename: xy-Scanner
# Propertyname: Frap Active
# Type: int
# Value: 0
#
#
#
# THIS PROPERTY IS IGNORED FOR NOW! NEEDS A PROPER USECASE!
#
#
# example: PROPSET_02=Pifoc Pifoc Measurement Modes allowed,int,4
# Devicename: Pifoc
# Propertyname: Pifoc Measurement Modes allowed
# Type: int
# Value: 3
#
# INFO: Value is the sum of corresponding "Pifoc Measurement Mode"
# represented as bit position.
# Made available in Dialog|Command Mode Combo Box.
#
#
#
# IMPORTANT
#
#
# example: PROPRESET_00=Pifoc Pifoc Measurement Mode,int,0
# Devicename: Pifoc
# Propertyname: Pifoc Measurement Mode
# Type: int
# Value: 0
#
# INFO: Make sure to reset properties to safe values,
# so that the hardware is not left in "unknown" state.
#
# i.e.: reset Pifoc into serial line mode, so that
#       it does not move on FRAME/LINE triggers by accident.
#
#
#
#
#

```

```

#
# For property names see Workspace Document (Alt + 0) in ImSpector
#
# DLGON=Dialogname1,Dialogname2,... switches the dialog on
# DLGOFF=Dialogname1,Dialogname2,... switches the dialog off
#
# RECONFIGS=REC01,RECXX allows the mode for reconfig.ini mode 01, XX
#
#
#
#

```

```

[AC05] # FIX THE INDEX NUMBER
LABEL=Pifoc Z
AXIS_FIRST=Pifoc Z
AXIS_SECOND=
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=PiFoc_150.bmp
DLGON=z-Stepper
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,1
#PROPSET_01=z-Stepper Pifoc Measurement Modes allowed,int,1
PROPPRESET_00=z-Stepper Pifoc Measurement Mode,int,0

```

```

[AC06] # FIX THE INDEX NUMBER
LABEL=Pifoc Z-T
AXIS_FIRST=Pifoc Z
AXIS_SECOND=Time Time
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=PiFoc_Time_Time_150.bmp
DLGON=z-Stepper
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,1
#PROPSET_01=z-Stepper Pifoc Measurement Modes allowed,int,1
PROPPRESET_00=z-Stepper Pifoc Measurement Mode,int,0

```

```

[AC05] # FIX THE INDEX NUMBER
LABEL=Pifoc Fast Z
AXIS_FIRST=
AXIS_SECOND=
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=PiFoc_Fast_Z_Burst_150.bmp
DLGON=z-Stepper
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,2
#PROPSET_01=z-Stepper Pifoc Measurement Modes allowed,int,2
PROPPRESET_00=z-Stepper Pifoc Measurement Mode,int,0

```

```

[AC06] # FIX THE INDEX NUMBER
LABEL=Pifoc Fast Z-T
AXIS_FIRST=Time Time
AXIS_SECOND=

```

```

AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=PiFoc_Fast_Z_Time_Burst_150.bmp
DLGON=z-Stepper
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,2
#PROPSET_01=z-Stepper Pifoc Measurement Modes allowed,int,2
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,0

```

```

[AC07] # FIX THE INDEX NUMBER
LABEL=Pifoc Fast Linescan
AXIS_FIRST=
AXIS_SECOND=
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=PiFoc_Line_Z_150.bmp
DLGON=z-Stepper
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,3
PROPSET_01=xy-Scanner Frap Active,int,0
#PROPSET_02=z-Stepper Pifoc Measurement Modes allowed,int,4
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,0

```

```

[AC08] # FIX THE INDEX NUMBER
LABEL=Pifoc Fast Linescan-T
AXIS_FIRST=Time Time
AXIS_SECOND=
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=PiFoc_Line_Z_Time_Time_150.bmp
DLGON=z-Stepper
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,3
PROPSET_01=xy-Scanner Frap Active,int,0
#PROPSET_02=z-Stepper Pifoc Measurement Modes allowed,int,4
PROPSET_00=z-Stepper Pifoc Measurement Mode,int,0

```

### 13.5.3 Snapshot

```

# PROPSET_XX sets properties of devices for the current mode (snapshot related)
#
# PROPSET_XX sets properties of devices for all other modes (snapshot related)
# i.e.: reset scanner modes
#
# _____
#
#
#
#
# example: PROPSET_00=xy-Scanner Modes,CString,"Linescan"
# Devicename: xy-Scanner
# Propertyname: Modes
# Type: CString
# Value: "Linescan"
#

```

```

# INFO: Modes: "CCD", "Hi-Res", "Single Beam", "SB Scanmode", "Linescan"
# Multiple modes may be added in a comma-separated list, see below.
#
#
#
#
# IMPORTANT
#
#
# example: PROPRESET_00=xy-Scanner Modes,CString,""
# Devicename: xy-Scanner
# Propertyname: Modes
# Type: CString
# Value: ""
#
# INFO: Make sure to reset properties to safe values,
# so that the hardware is not left in "unknown" state.
#
# i.e.: reset to allow all scanner modes
#
#
#
#
# For property names see Workspace Document (Alt + 0) in ImSpector
#
# DLGON=Dialogname1,Dialogname2,... switches the dialog on
# DLGOFF=Dialogname1,Dialogname2,... switches the dialog off
#
# RECONFIGS=REC01,RECXX allows the mode for reconfig.ini mode 01, XX
#
#
#
#
[AC00] # FIX THE INDEX NUMBER
LABEL=Snapshot Linescan
AXIS_FIRST=
AXIS_SECOND=
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=snapshot_linescan_150.bmp
COMMENT=Single Linescan Image
PROPSET_00=xy-Scanner Modes,CString,"Linescan"
PROPRESET_00=xy-Scanner Modes,CString,""

[AC01] # FIX THE INDEX NUMBER
LABEL=Snapshot PMT
AXIS_FIRST=

```

```

AXIS_SECOND=
AXIS_THIRD=
AXIS_FOURTH=
ICONFILE=snapshot_150.bmp
COMMENT=Single PMT Image
RECONFIGS=REC01
PROPSET_00=xy-Scanner Modes,CString,"Single Beam,SB Scanmode"
PROPSET_00=xy-Scanner Modes,CString,""

```

### 13.5.4 Stitching

```

# PROPSET_XX sets properties of devices for the current mode (stitching related)
#
# PROPSET_XX sets properties of devices for all other modes (stitching related)
#
# _____
#
#
#
# Please report any interesting properties!
#
#
#
# For property names see Workspace Document (Alt + 0) in ImInspector
#
# DLGON=Dialogname1,Dialogname2,... switches the dialog on
# DLGOFF=Dialogname1,Dialogname2,... switches the dialog off
#
# RECONFIGS=REC01,RECXX allows the mode for reconfig.ini mode 01, XX
#
#
#
#

[AC05] # FIX THE INDEX NUMBER
LABEL=3D Mosaic
AXIS_FIRST=xyz-Table X,Split
AXIS_SECOND=xyz-Table Y,Split
AXIS_THIRD=xyz-Table Z
AXIS_FOURTH=
ICONFILE=3D_stitch_150.bmp
COMMENT=StitchingMosaic
DLGON=xyz-Table Visual XY

[AC06] # FIX THE INDEX NUMBER
LABEL=3D Multiposition
AXIS_FIRST=xyz-Table X,Split
AXIS_SECOND=xyz-Table Z
AXIS_THIRD=
AXIS_FOURTH=

```

```
ICONFILE=3D_stitch_pos_150.bmp  
COMMENT=StitchingPosition  
DLGON=xyz-Table Visual XY
```