OpenAnalyser

Alexander Schlemmer

April 28, 2010

Contents

1 Introduction

OpenAnalyser is a Java program for scientifc data analysis. It was especially designed to analyse cardiovascular florescence imaging data. OpenAnalyser provides a simple, yet powerful plugin system, which allows the extension of the program functionality with with user contributed code, including analysis algorithms, custom import functionality and visualization. An automatic log mechanism keeps track of every step during analysis and enables the user to reproduce results that have been previously obtained. The program has been developed at the Max Planck Research Group Biomedical Physics located at the Max Planck Institute for Dynamics and Self-Organization in Göttingen and is released under GNU GPL Version 3.

2 Installation

2.1 Normal Installation (Binary Distribution)

Simply unzip "OpenAnalyser.zip" in a directory of your choice. OpenAnalyser automatically installs itself in your home-directory when starting the program for the first time (see 3).

3 Running OpenAnalyser

3.1 Via the .jar-file

On systems with a correctly installed java virtual machine it is possible to run "OpenAnalyser.jar" (by simply single-/double-clicking it in your file browser). On some systems it might be necessary to select "Open with" →<Name of your Java-Runtime-Environment, e.g. "OpenJDK Java 6 Runtime"> instead of "Open". It will bring up a dialog where you can configure the Xmx-option.

3.2 Command Line

This method is mostly system-independent. A shell is available on most operating systems.

1. Open a shell (Linux: konsole, gnome-terminal, ...; MacOS: terminal; Windows: MS DOS Prompt).
2. Change the current directory (cd) to the root directory of OpenAnalyser, e.g. the directory where the file "OpenAnalyser.jar" is stored.
3. Run the command:
   `java -Xmx1024m -jar OpenAnalyser.jar`

The Xmx-option instructs the virtual machine to reserve 1GiB of memory for the program, which is a good choice for lots of data analysis. If you don't need that much (or don't have so much RAM), try "-Xmx512m" or even less. A common mistake is to forget the "m" after size which causes the VM to reserve very few memory (and probably cause crashes).

4 Quitting OpenAnalyser

OpenAnalyser can be quit using the button "Quit" in the file menu.

5 OpenAnalyser Windows

5.1 Main Window

The main window is the container for all windows used inside of OpenAnalyser. The main menu contains the following entries:

- File
  - **New**
    Clears the workspace and starts a new workspace file. The program asks for confirmation before clearing the workspace.
  - **Import**
    Brings up the import dialog which is used to import raw data into the workspace (See section 6.1).
  - **Save**
    Save the workspace to a file. Workspace files are zip-containers containing raw and xml data. The default file-extension is "oaw" which stands for "OpenAnalyser Workspace".
  - **Load**
    Load the workspace from a file.
  - **Quit**
    Closes OpenAnalyser.
- Tools
  - **Reload all Plugins**
    Reloads all OpenAnalyser plugins. The workspace is saved to a temporary location. Afterwards all classes contained in the plugin directory are reloaded (or loaded if they are found for the first time) and the workspace is reloaded from the temporary location. This feature is useful for plugin developers who want to quickly try out changes without having to restart the whole program.
  - Settings
- Windows

  - Workspace
  - Ranges
  - Messges
  - Queue
  - Property Editor
  - Window List
  - Close All
- Help
  - Info

5.2 The Log-Window

The log-window is the center of OpenAnalyser. It stores every piece of information created in the analysis and allows to perform operations on them.

The Workspace-Window provides an overview of the log entries that have been created and processed. The table contains three columns, namely "Description" and "Type" and "Info". The description can be directly changed by editing the text in the corresponding column. The most important functions can be accesed via the right-click menu for the list contained in the workspace menu. For being able to run plugins on specific log-entries it is important to ensure that the log-entry is selected. The description of the log-entry is then highlighted blue.

5.2.1 Log-Entry-Menu

Right clicking on a selection of log-entries or clicking the button labelled "Actions" in the Log-Window opens the log-entry-menu. The Log-Entry-Menu looks very similar to figure 1 most of the time. The first entry is the description of the log-entry (as entered in the corresponding column).

The following action is available for all log-entries:

- Delete: Deletes the selected log-entries (without asking questions)

The following action is available for datasets only:

- Create Plot: create a new plot out of the selected datasets

The following action is available for datasets:

- Show Plot: shows a previously created plot or creates and displays a temporary plot depending on the type of log entries

The following action is available for Video-Datasets only:

- Image Display Settings: configure the selected Image- / Video-Dataset (see section 4)

The following action is available for plots only:

- Edit Plot: configure the selected plot

Functions are available for all types of log-entries and listed below the second seperator in the menu. They are ordered by package-name and class-name.

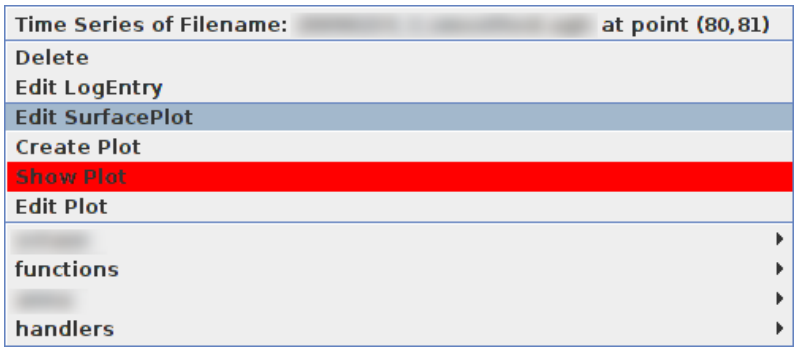


Figure 1: Log-Entry-Menu for a selected timeseries. The default action for this log-entry-type is highlighted in red.

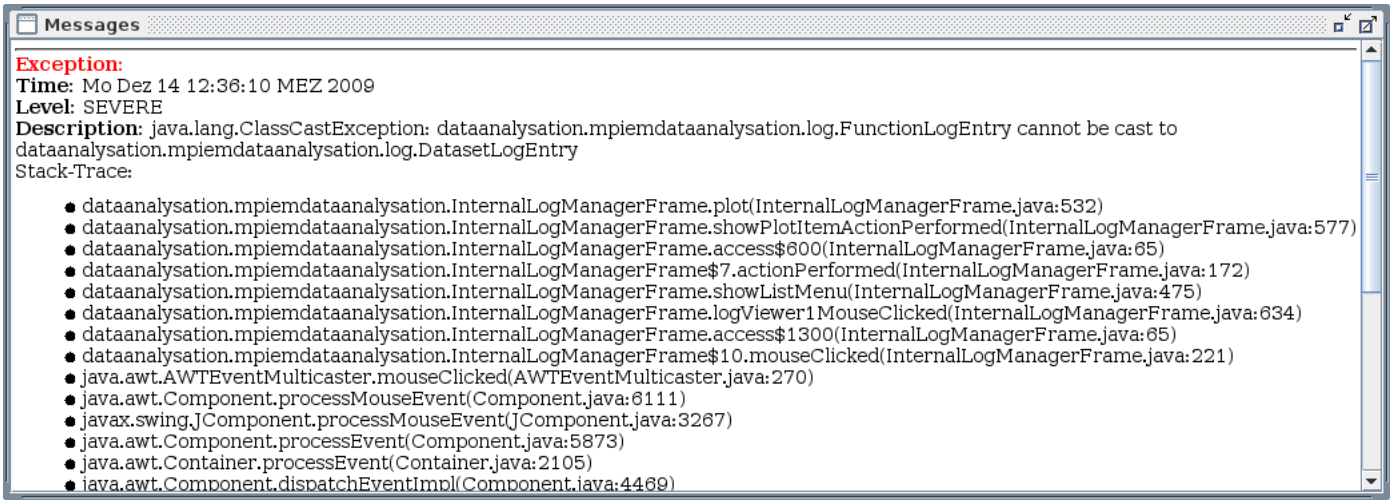


Figure 2: The message window displaying detailed information about an exception

5.3 Message Window

The message window is used to report textual feedback to the user. Plugins can use this window to display simple output. Furthermore it is used to report errors.

5.3.1 Error messages

The message window is used to display java error message in a nicely formatted way. Each error message includes:

- Timestamp
- Level
- Description
- Stack-Trace
- (Cause)

5.4 Ranges Window

The Ranges window is the global configuration window for ranges that can be used for functions. Selections inside of surface plots are sent directly to this window. There are six entries:

1. **X1:** X-Coordinate of the upper left point of the selected rectangle.
2. **Y1:** Y-Coordinate of the upper left point of the selected rectangle.
3. **X2:** X-Coordinate of the lower right point of the selected rectangle.
4. **Y2:** Y-Coordinate of the lower right point of the selected rectangle.
5. **T1:** The number of the first selected frame (frame numbering starts with 0). See 5.4.1 for details.
6. **T2:** The number of the first frame after the last selected frame (frame numbering starts with 0). See 5.4.1 for details.

5.4.1 Frame ordering

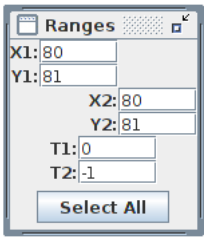
Frame ordering and temporal ranges used for the manipulation of video surface plot datasets corresponds to most programming languages, e.g. the first frame has number 0 and the last frame is the number of frames minus 1. Similarly a range is made of two frame numbers where the first frame number (T1) is inclusive and the second frame number (T2) is exclusive. Negative numbers are counted "in reverse", so that "-1" corresponds to the last selectable value +1. This behaviour makes it easy to select whole dimensions, e.g. the whole temporal range is selected by setting **T1** to 0 and **T2** to -1. The button "Select All" can be used to select everything, i.e. to set **X1=Y1=T1=0**, **X2=Y2=T2=-1**.

5.4.2 Range selection

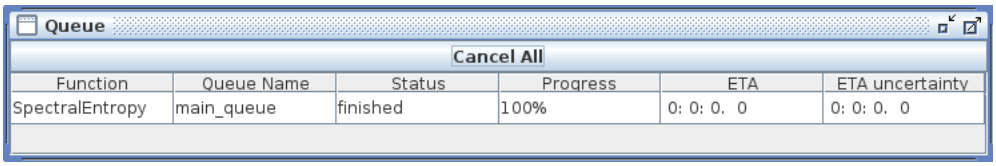
There are two types of selections for surface plots:

1. **Point selection:** Clicking inside a surface plot sets the upper left and the lower right points simultaneously to the same coordinates. I.e. a point is a rectangle with zero width and height from the point of view of the ranges. A point selection is marked by an orange "+" inside of the plot panel.
2. **Rectangle selection:** Dragging a rectangle sets the upper left and the lower right coordinates to the corresponding coordinates of the selected rectangle. The rectangle is visualized by an orange rectangle inside of the plot panel.

Additionally double clicking inside of a plot panel does not only zoom the view to fit the contents, but also selects the whole spatial area (i.e. **X1=Y1=0**, **X2=Y2=-1**). The button "Select All" can be used to set set **X1=Y1=T1=0**, **X2=Y2=T2=-1**.



(a) Ranges window



(b) Queue Window

Figure 3: Main Windows

5.4.3 How range selections are used by functions

How selections are used by functions depends mainly on the function itself. The convention is that filters (functions that manipulate data in place) should use the ranges as input- / output-ranges for filters, i.e. the region to which the filter is applied. Functions that do not manipulate data in place should consider the ranges as input-ranges, i.e. generate their results only from data contained in the selected range. However functions cannot be forced to do so and the function-documentation should be taken into account in case of doubt.

6 Important Dialogs

6.1 Import Video

Importing Data normally involves three steps:

- Select a filename. Three methods can be used to do this:
  - Enter the filename into the text field named "Filename(s)"
  - Use the "Open File" dialog by pressing the button labelled "..."
  - Select a favorite by choosing it from the list
- Select an import plugin: OpenAnalyser tries to guess the appropriate import plugin by analysing the file. Import plugins that are recognized to work for the file light up green. The first recognized import plugin in the list is selected automatically. Import plugins are organized in cathegories. There are four radio buttons above the list of available plugins which filter the list:
  - All: Show all installed import plugins
  - Video: Show video import plugins only. Video import plugins are plugins which read data and create a VideoDataset (or a similar custom dataset type).
  - Image: Show image import plugins only.Image import plugins are plugins which read data and create a ImageDataset (or a similar custom dataset type).
  - Time-Series: Show time series import plugins only. Time series plugins are plugins which read data and create a TimeSeries (or a similar custom dataset type).
- After pressing "OK" a dialog wil show up which allows the setting of parameters needed by the import plugin. After pressing "OK" again the data will be imported into the workspace.

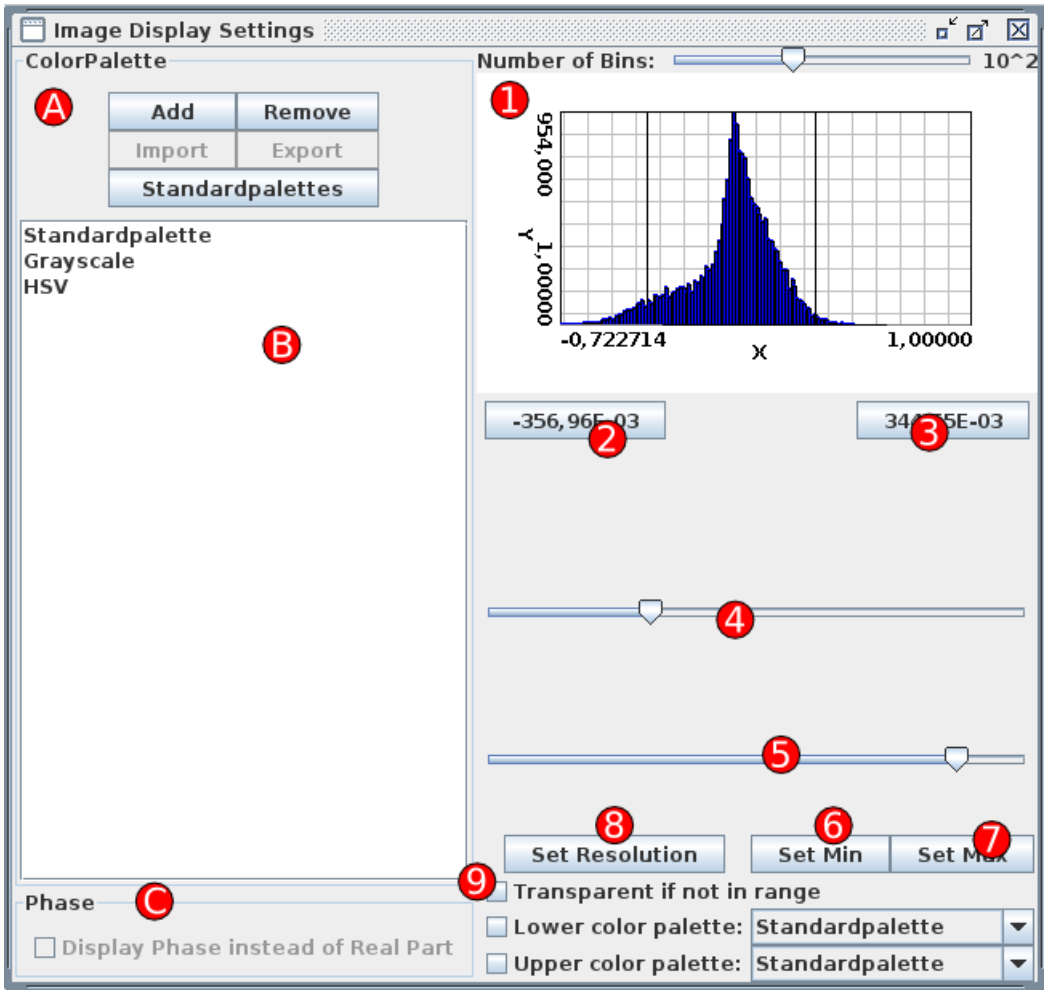


Figure 4: Configuration of a surface plot

6.1.0.1 Favorites Favorites are a practical feature to store often used filenames without having to reselect / reenter them in future sessions. Simply click the button "Add to favorites" in order to add the contents of the filename-text-field as a new favorite file. The "Remove"-button removes favorites without asking. Clicking in the list puts the selected line into the filename text-field. Favorites are stored in the file ".openanalyser.xml" in the user's home-directory.

6.2 Image Display Settings

Images and videos are displayed by mapping values contained in them to a colormap. Sometimes it is necessary to select the range of values which are mapped to this colormap. Settings related to the display of surface plots can be configured by choosing the menu entry "Image Display Settings" while the corresponding surface plot dataset is selected (highlighted in blue). Figure 4 shows a typical view of the "Image Display Settings"-window.

6.2.0.2 Explanation of the labelled controls:

1) **Histogram:** Displays a histogram created from the values of one image of the surface plot. The image that had been visible when this window was opened is used for this histogram. The two black lines in the plot mark the positions of the contrast sliders (see 6.2.0.2). The behaviour of this plot panel is described in section 7.1. The slider labelled "Number of bins" can be used to adjust the number of bins in the histogram.

2) **Upper limit:** Use this button to set the lower limit of the range manually. This will affect the position of the contrast sliders. Allowed values for this limit depend on the setting of the resolution.

3) **Lower limit:** Use this button to set the upper limit of the range manually. This will affect the position of the contrast sliders. Allowed values for this limit depend on the setting of the resolution.

4 and 5) **Contrast Sliders:** The contrast sliders are used to control the range which is mapped to the color palette. The upper slider controls the lower limit and the lower slider controls the upper limit of the range.

6) **Minimum:** Use this button to set the minimum value of the contrast sliders. Allowed values for this limit depend on the setting of the resolution.

7) **Maximum:** Use this button to set the maximum value of the contrast sliders. Allowed values for this limit depend on the setting of the resolution.

8) **Resolution:** Use this button to set the resolution of the contrast sliders.

9) **Transparency:** If this checkbox is checked values in the surface plot higher than the upper limit or lower than the upper limit are not displayed. Otherwise the topmost and the bottommost colors in the color palette are used for values that are outside of the specified range.
Lower color palette: If this checkbox is checked values smaller than the lower limit are mapped to a different colorpalette (which can be selected by the drop down menu).
Upper color palette: If this checkbox is checked values larger than the upper limit are mapped to a different colorpalette (which can be selected by the drop down menu).

A) **Color Palettes:** These controls are used to control the list of available colormaps.

B) **Color Palette List:** The active color map for the surface plot is selected here.

C) **Phase Display:** This checkbox is only available if the dataset contains complex data. In this case the display of the phase (the angle of the complex number) can be switched on with this checkbox.

6.3 Universal Property Editor

This editor, also known as the Quercus-editor is used for several tasks including editing the properties of log entries and setting parameters for function-plugins. A typical display looks like figure 5.

6.4 Preferences

Figure 6 shows the preferences-window which can be accesed via the main menu ("Tools"→"Preferences"). Preferences are saved on clicking "Save". These preferences are stored in the user's home-directory in the file ".openanalyser.xml".

6.4.1 Directories

There are four important paths which can be configured in this dialog:

- Main path: The root directory of your OpenAnalyser-installation. This is the folder where your "OpenAnalyser.jar" should be stored if you are not a developer.
- Plugin path: This is the root directory for your plugins. This directory should normally point to <Main path> + "plugins". If this directory is not configured correctly no functions will appear in the log-entry-menu (section 5.2.1).
- Library path: The library-folder is the folder where additional libraries, containing for example separate mathematical routines, are stored. This should be set to <Main path> + "lib" in most cases.

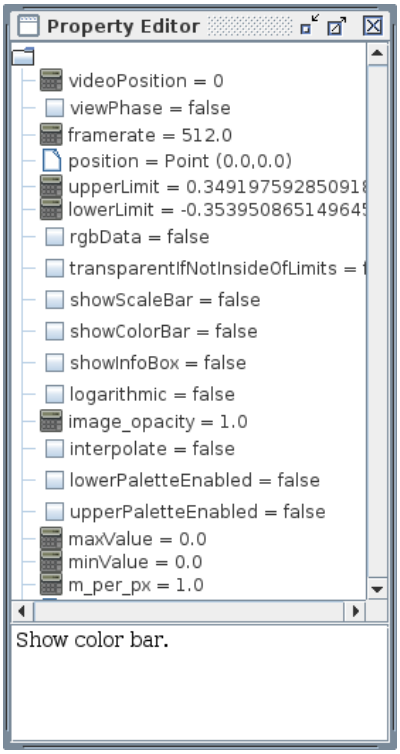


Figure 5: Property-Editor in action

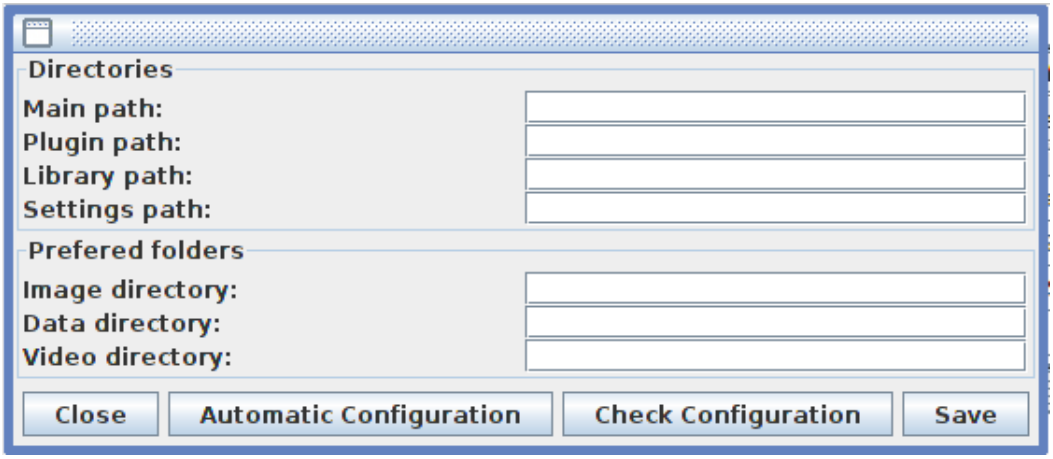


Figure 6: The preferences window

- Settings path: The settings folder is necessary to store information that is needed in multiple sessions of OpenAnalyser and is independent from individual data-files. The folder is usually stored in the user's home-directory (e.g. "/home/username/" on linux and mac) and named ".emanalyser". Its contents should be:
  - The color-palettes: colorPalettes.dat
  - The last window-layout: lastWindowLayout.xml
  - The workspace is saved on closing OpenAnalyser in this file: lastSession.oaw
  - A directory "tempData" containing output produced by the Octave-Output-plugin

6.4.2 Prefered Folders

Prefered folders are those folders initially set when a dialog for specifying file names is opened. Three different folders are distinguished:

- Image directory: used for image-export
- Data directory: used for video-import (see section 6.1)
- Video directory: used for video-export

6.4.3 Automatic Configuration

This invokes a method that tries to configure the important directories automatically. It usually works for the binary-installation of OpenAnalyser.

6.4.4 Check Configuration

This button is useful for debugging. It reports the status of most files and directories required (or recommended) by OpenAnalyser. The output is dumped to the message-window (see section 5.3).

7 Special windows

7.1 PlotPanel

The PlotPanel is probably the most used component in OpenAnalyser. It can be used to display the following plot-types and combinations (superpositions) of them:

1. XY-Plots / Histograms
2. Static Surface-Plots
3. Video Surface-Plots
4. Vector-Plots

This component is also used in the Image Display Settings (see section 6.2) as a display for histograms.

7.1.1 Interaction

Hint: Interaction has been developed for linux (Ubuntu 9.10, with OpenJDK version 1.6.1) and works there. On your operating system some of the possibilities mentioned below might not work!

Abbreviations used:

- LMB = Left Mouse Button
- MMB = Middle Mouse Button
- RMB = Right Mouse Button

Action	Linux
Move the visible region	MMB + Drag **OR** ALT + LMB + Drag **OR** CTRL + LMB + Drag
Zoom into a selection region	SHIFT + LMB + Drag to select a rectangle
Select a rectangle	Drag to select a rectangle
Select a point	Click a point using LMB

8 Function-Plugins

OpenAnalyser is equipped with several plugins useful for analysing scientific video data, especially excitable media. Plugins are java-class-files, a special kind of binary file (called bytecode), with the file-extension ".class". Plugins are organized in packages, which are represented by a folder structure on the filesystem.

8.1 Plugins in detail

It follows a list of most plugins included in the OpenAnalyser-distribution. The required input log entries are listed in sections "*Input*", correspondingly "*Output*" notes the types of output created by the function. If there are multiple possible input types they are numbered. "*Ranges*" notes which range parameters (see section 5.4.2) are used by the plugin. Sections "*Additional Parameters*" explains additional input parameters which may be queried by the function (using Quercus, see section 6.3) and their default values. Optional input / output (i.e. output that depends for example on an additional parameter) is written in square brackets.

8.1.0.1 Notation:

1. Coordinates:
   (a) Two-dimensional: (x, y)
   (b) Three-dimensional: (x, y, t)
2. Ranges:
   (a) Rectangle: $[x1, y1, x2, y2]$
   (b) Temporal range: $[t1, t2]$
   (c) Three-dimensional range: $[x1, y1, t1, x2, y2, t2]$

8.1.1 Standard-Plugins

8.1.1.1 TimeSeries

Input: VideoSurfacePlotDataset

Output: PlotDataset

Ranges: $X1$, $Y1$, $T1$, $T2$

Additional Parameters: None

Description:

This function extracts a timeseries from a video at coordinates ($X1$, $X2$) between $T1$ and $T2$ and stores the points ($time$,$value$) in a PlotDataset. The framerate from the video is copied.

8.1.1.2 PowerSpectrum

Input (1): VideoSurfacePlotDataset

Input (2): PlotDataset

Output: PlotDataset

Ranges: $X1$, $Y1$, $T1$, $T2$

Additional Parameters: None

Description:

(1) This function extracts a timeseries from a video at coordinates ($X1$, $X2$) between $T1$ and $T2$, creates a power spectrum out of it and stores the points ($frequency$,$value$) in a PlotDataset. The framerate from the video is copied.

(2) This function creates a power spectrum out of a given timeseries and stores the points ($frequency$,$value$) in a PlotDataset. The framerate from the timeseries is copied.

8.1.1.3 PseudoECG

Input: VideoSurfacePlotDataset

Output: PlotDataset

Ranges: X1, Y1, X2, Y2, T1, T2

Additional Parameters: None

Description:

This function creates a PlotDataset containing the points ($time$,$value$) where value is the arithmetic mean of the values contained in the rectangle [$X1$,$Y1$,$X2$,$Y2$] taken from the video at time $time$. The temporal range $[T1, T2]$ is taken into account as input range.

8.1.1.4 SpectralEntropy

Input: PlotDataset

Output: None (Message only)

Ranges: None

Additional Parameters: *nzust*

Description:

This function computes the spectral entropy out of a power spectrum (which means that the input-PlotDataset has to be a PowerSpectrum, in order to return useful results). No input range is taken into account. The spectral entropy is displayed as a message (see section 5.3).

Details:

Data in the power spectrum is binned using *nzust* bins. Afterwards the spectral entropy is computed according to the formula:

$$H = \sum_{p \neq 0} p \cdot \log_2(p) \tag{1}$$

where p is an element of the binned data.

8.1.1.5 DominantFrequencyMap

Input: VideoSurfacePlotDataset

Output: StaticSurfacePlotDataset [, StaticSurfacePlotDataset]

Ranges: $X1$, $Y1$, $X2$, $Y2$, $T1$, $T2$

Additional Parameters: *createAmplitudeMapToo*, *threshold*

Description:

Creates a map of dominant frequencies for the input dataset. The dominant frequency is the frequency with the highest amplitude for a given timeseries. Only timeseries contained in $[X1, Y1, T1, X2, Y2, T2]$ are used and the corresponding output dataset(s) will have the spatial size of this cuboid. If *createAmplitudeMapToo* is set to true, a second map containing the amplitudes of the dominant frequencies for each time series is created. The parameter *threshold* can be used to ignore noise: Only frequencies will be displayed where the corresponding amplitude is larger than *threshold* (they will be set to NaN otherwise).

8.1.1.6 ActivationMap

Input: VideoSurfacePlotDataset

Output: StaticSurfacePlotDataset

Ranges: $X1$, $Y1$, $X2$, $Y2$, $T1$, $T2$

Additional Parameters: *threshold*, *interpolate*, *derivations*

Description:

For each timeseries contained in $[X1, Y1, T1, X2, Y2, T2]$ the activation-time is calculated and displayed on a map. The activation-time is assumed as the time t_{act} when a value of the timeseries rises above $mean + std \cdot threshold$ or drops below $mean - std \cdot threshold$ for the first time. *mean* is the arithmetic mean of all values of the timeseries and *std* is its standard deviation. *derivations* is the number of difference-operations applied to each timeseries before the activation-time is determined. If *interpolate* is set to true a linear interpolation between t_{act} and $t_{act} - 1$ is used in order to approximate the activation time more accurately.

8.1.1.7 ActivationCurve

Input: SurfacePlotDataset

Output: PlotDataset, PlotDataset

Ranges: None

Additional Parameters: None

Description:

Creates an activation curve out of an activation map. An activation curve shows the number of already activated points (i.e. the number of points that have already been hit by a wave of excitation) for each time step (see section 8.1.1.6, too). The second PlotDataset that is created by this function is the corresponding activation difference curve which displays the number of activated points at a given time. In order to achieve good activation difference curves the source activation map should be discrete, so it should be created with $interpolate = false$.

8.1.2 Normalization

8.1.2.1 Normalization (Filter)

Input: VideoSurfacePlotDataset

Output: None

Ranges: $X1$, $Y1$, $X2$, $Y2$, $T1$, $T2$

Additional Parameters: lowerLimit, upperLimit, scope

Description:

Each point is normalized using the formula

$$p_{new} := (p_{old} - min) \cdot \frac{upperLimit - lowerLimit}{max - min} + lowerLimit \tag{2}$$

min and max are the minimum and maximum values of

1. each time series if $scope = pointwise$
2. each frame if $scope = framewise$

9 Plugin-Developement

Detailed information about the classes needed for plugin development can be found in the javadoc-documentation of OpenAnalyser. The examples-package of OpenAnalyserPlugins contains many useful examples.

9.1 Quick-Start-Tutorial

This section is meant for those who want to jump right into plugin-development. A good way to get used to the mechanisms is to take a look at an easy example plugin:

```
1  import de.mpg.ds.openanalyser.dataset.*;
2  import de.mpg.ds.openanalyser.math.*;
3  import de.mpg.ds.openanalyser.plugin.baseclasses.*;
4
5  public class ExamplePlugin extends GeneralFunction {
6          public void call () {
7                  // Receive parameters from the main program.
8                  // Parameters include the spatial and the temporal
9                  // selections from the ranges window.
10                 DestinationParameters params = getParameters();
11
12                 // Add a message to the message window.
13                 addMessage("This is just a simple example.");
14
15                 // Create a new xy-plot and give it a name.
16                 PlotDataset plot = new PlotDataset("Some numbers.");
17                 // Add some numbers.
18                 for (int i=0; i<42; i++)
19                         plot.addPoint(new DoubleXY(i, i*i));
20
21                 // Add the plot to the workspace.
22                 addResult(plot);
23         }
24 }
```

This code describes a simple plugin which takes no input parameters and displays a message and creates a new xy-plot-dataset when called. Simply save the above code to a file named "ExamplePlugin.java", place it in your plugin-directory and compile it using "javac". As soon as the compilation process is finished you find the class-file inside of your plugin directory. In the open program you can now deselect everything and click "Actions" in the workspace-window. The example-plugin will be listed somewhere at the bottom of the popup-menu.

9.1.1 Input datasets

Plugins which take no datasets as input are normally used to create simulations or similar data-producing programs. In most cases analysis of data requires the input of one or more datasets from the workspace. The plugin system of OpenAnalyser implements a mechanism which works similar to function overloading in programming languages: The plugin system looks inside of the java class for functions named "call" and checks if the parameters of that function correspond to the selected datasets in the workspace. If this is true the function is invoked via reflection.

We want to depict this behaviour with another example plugin:

```
1  import de.mpg.ds.openanalyser.dataset.*;
2  import de.mpg.ds.openanalyser.math.*;
3  import de.mpg.ds.openanalyser.plugin.baseclasses.*;
4
5  public class ExamplePlugin extends GeneralFunction {
6          public void call () {
7                  // Do something without a dataset.
8          }
9
10         public void call (VideoDataset video) {
11                 // Do something, operating on a VideoDataset.
12         }
13
14         public void call (PlotDataset dataset) {
15                 // Do something, taking the data from the xy-plot-dataset into account.
16         }
```

```
17
18          public void call (VideoDataset video, TimeSeriesDataset timeSeries) {
19                  // The plugin invokes this function, if the plugin is applied to a selection
20                  // containing a VideoDataset and a TimeSeriesDataset in this order.
21          }
22  }
```

The plugin can be applied to the following selections:

- Nothing selected
- A VideoDataset selected
- A PlotDataset selected, note that TimeSeriesDataset extends PlotDataset, so it is possible to supply a TimeSeriesDataset too
- A selection containing a VideoDataset as the first entry and a TimeSeriesDataset as the second entry

In each case a different call-function is invoked which is specialized for the selected dataset / dataset combination.