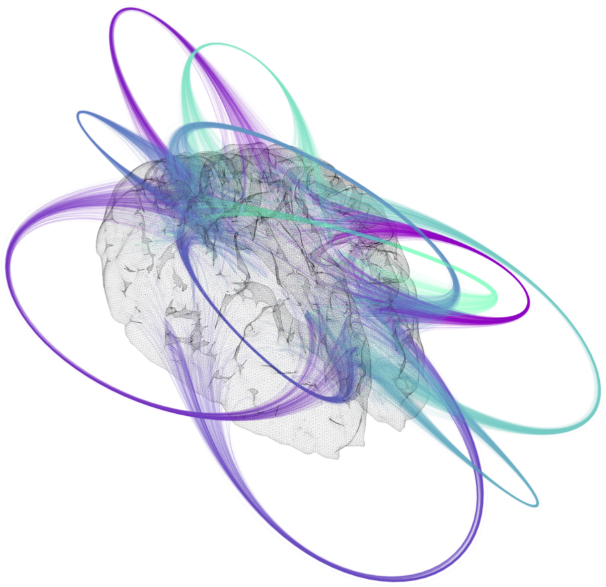
mpi.is-brain-connectivity-plugin Documentation

Release 0.1

Lennart Bramlage, Raffi Enficiaud

May 08, 2018

Contents:



CHAPTER 1

Gallery

A set of pictures generated in Paraview and the brain-connectivity plugin is collected here. Feel free to send additional pictures to the authors/maintainers of the plugin.

CHAPTER 2

Installation

The installation is done in 2 steps:

1. first installing the package (preferably in a python virtual environment),
2. then generating the plugin that points to the installed package, and register the plugin into Paraview. This is needed only if you want to use the Paraview plugin. Otherwise installing the package alone will let you use the utility functions.

Things to consider:

The installation procedure for the package and the plugin depend on the version of Paraview in use.

2.1 Identify your Paraview and Python installations

In case you want to use the visualization plugin and the package functions inside Paraview, you will need to identify the version of Paraview you are using. The most important aspect is to

- identify if this is a standalone version of Paraview
- if your version uses the same Python interpreter as the one of your package.

In case the installation of Paraview is standalone, Paraview may use its own Python interpreter (through `pvpython` or `pvbatch`), while your package may be using another Python interpreter that is eg. the one that defaults on your operating system.

Should this be the case, the interpreters in use in the plugin and the package would be different. This is usually not an issue, as the package code (as of May 2018) is pure Python and does not bring any compiled python extension. However, in case you start developing additional extensions and those contain python extensions, an ABI problem may occur and will potentially lead to a crash during the execution of the plugin.

To be safe, it is recommended to install a version of Paraview that uses the OS Python interpreter, and to use the same interpreter for installing our package.

See the *Platform specific details* for more information.

2.2 Package installation

The package preferably installs in a virtual environment. Also since Paraview is using Python2, we suggest sticking to Python2. For installing the package, you need at least `virtualenv` and `pip`. Once there, let's first create a virtual environment like this (in a terminal):

```
mkdir ~/.venvs
virtualenv --system-site-packages ~/.venvs/venv_paraview_brain
. ~/.venvs/venv_paraview_brain/bin/activate
```

and then navigate to the path where you cloned the package, and type

```
cd <brain-connectivity-visualization-path>
pip install .
```

After the installation, you should see the package with `pip list`:

```
pip list
Package                          Version
-------------------------------- ---------
[...]
brain-connectivity-visualization 1.0
```

2.3 Plugin installation

For installing the plugin, we first generate a binding between Paraview and the package in the form of an `.xml` file, and we then install/register this file in Paraview as a plugin. This `.xml` will act as a binding between the relevant functions in our package and Paraview.

There is no streamlined procedure for installing a python plugin into Paraview, and/or to use an external python library. Paraview has different installation flavour that differ according to the platform or the way you installed the tool. We tried to make the procedure as easy as possible.

2.3.1 Generating the XML plugin file

Installing the package will bring you a script `generate_brain_connectivity_plugin` that will be used for generating a plugin file in the `.xml` format. This little tool converts the entry point of the plugin into an XML file that can be installed/registered into Paraview.

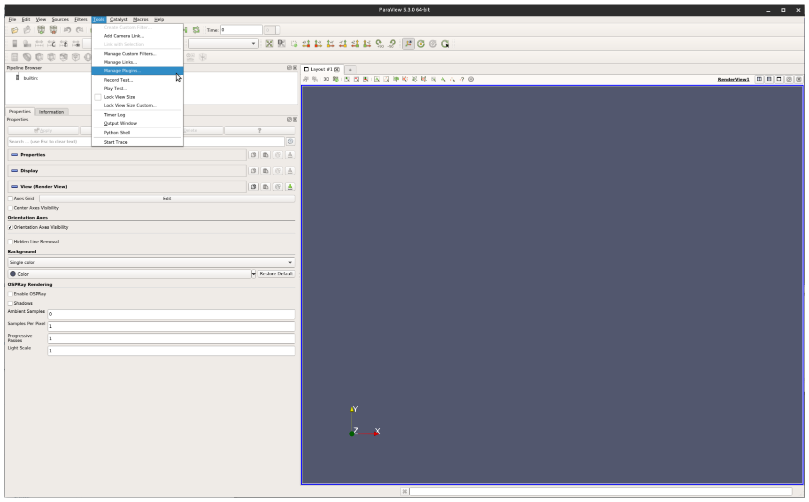
To call the tool, run the following from the command line terminal:

```
generate_brain_connectivity_plugin mpi_brain_plugin.xml
```

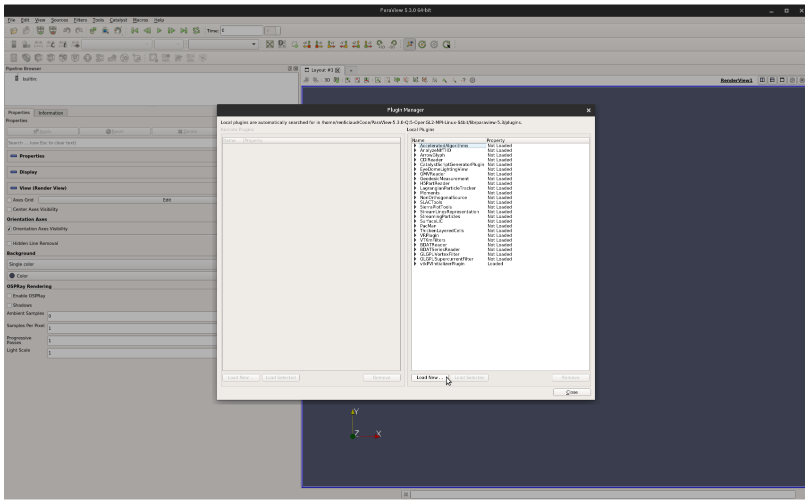
2.3.2 Registering the XML plugin file

This plugin should then be imported into Paraview

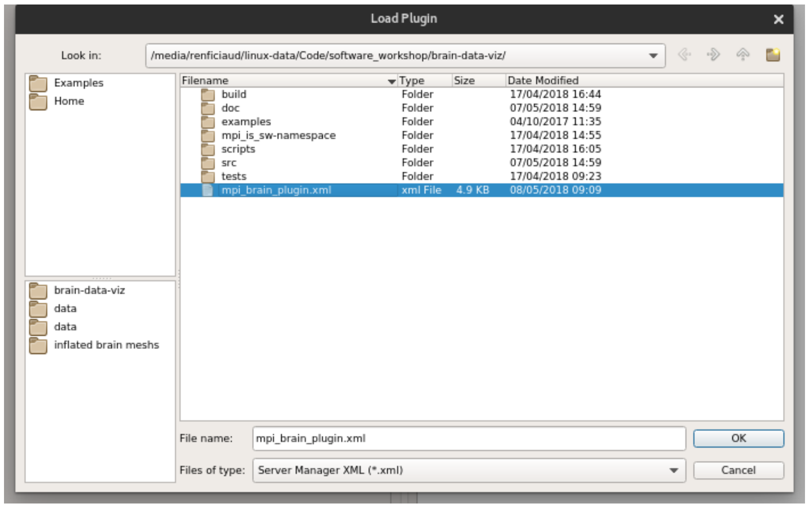
1. Open the menu from `Tools > Manage Plugins`:



2. On the custom filter window, click on *Load New*:



3. Select the `.xml` file that you generated earlier, make sure that the file filter of the selection dialog shows `.xml`:


Load Plugin

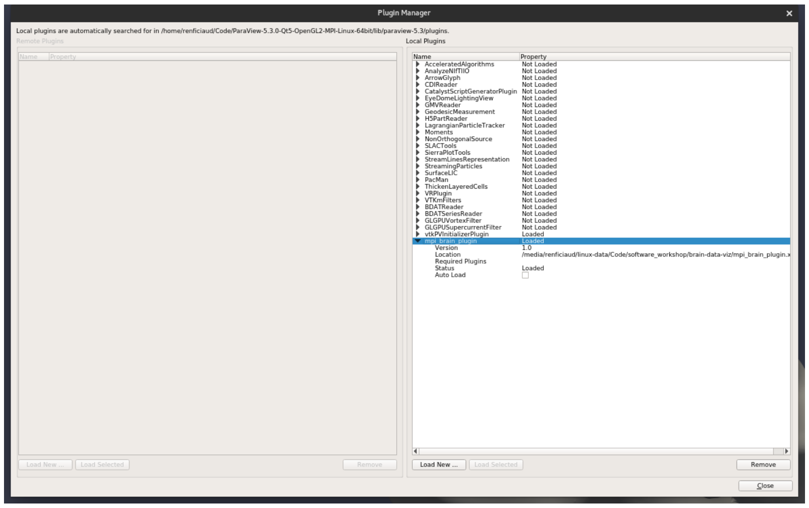
Look in: /media/renficiaud/linux-data/Code/software_workshop/brain-data-viz/

Filename	Type	Size	Date Modified
build	Folder		17/04/2018 16:44
doc	Folder		07/05/2018 14:59
examples	Folder		04/10/2017 11:35
mpi_is_sw-namespace	Folder		17/04/2018 14:55
scripts	Folder		17/04/2018 16:05
src	Folder		07/05/2018 14:59
tests	Folder		17/04/2018 09:23
mpi_brain_plugin.xml	xml File	4.9 KB	08/05/2018 09:09

File name: mpi_brain_plugin.xml

Files of type: Server Manager XML (*.xml)

4. You should then see the plugin on the list of plugins, make sure the plugin is **loaded**. You may check the `autoload` option for loading the plugin the next time you start Paraview.


Plugin Manager

Local plugins are automatically searched for in /home/renficiaud/Code/ParaView-5.3.0-Qt5-OpenGL2-MPI-Linux-64bit/lib/paraview-5.3/plugins.

Local Plugins

Name	Property
AcceleratedAlgorithms	Not Loaded
AnalyzeNIfTIIO	Not Loaded
ArrowGlyph	Not Loaded
CDIReader	Not Loaded
CatalystScriptGeneratorPlugin	Not Loaded
EyeDomeLightingView	Not Loaded
GMVReader	Not Loaded
GeodesicMeasurement	Not Loaded
H5PartReader	Not Loaded
LagrangianParticleTracker	Not Loaded
Moments	Not Loaded
NonOrthogonalSource	Not Loaded
SLACTools	Not Loaded
SierraPlotTools	Not Loaded
StreamLinesRepresentation	Not Loaded
StreamingParticles	Not Loaded
SurfaceLIC	Not Loaded
PacMan	Not Loaded
ThickenLayeredCells	Not Loaded
VRPlugin	Not Loaded
VTKmFilters	Not Loaded
BDATReader	Not Loaded
BDATSeriesReader	Not Loaded
GLGPUVortexFilter	Not Loaded
GLGPUSupercurrentFilter	Not Loaded
vtkPVInitializerPlugin	Loaded
mpi_brain_plugin	Loaded
Version	1.0
Location	/media/renficiaud/linux-data/Code/software_workshop/brain-data-viz/mpi_brain_plugin.x
Required Plugins	
Status	Loaded
Auto Load	☐

Warning: Paraview will need the `.xml` file you generated every time it needs to run your plugin. In case you move this file to some other physical location on disk, you will be needing to redo the installation steps above after having uninstalled the plugin.

Also the plugin will be using the location of the python virtual environment where the package is installed. This will let you update the package (and the plugin functionalities) without registering the plugin again, but will prevent you from removing the virtual environment.

Note: In case you save your session in Paraview for later reuse/load, the plugin needs to be loaded **prior** to loading the session. This is why the `autoload` option is convenient.

Tip: You may have noticed the message `Local plugins are automatically searched for in <some-path>` at the top of the plugin list dialog box. If you have a standalone installation of Paraview, you may generate the `.xml` file directly there and then skip the registration and loading of the plugin.

2.3.3 Platform specific installation details

OSX

Paraview on OSX relies on the OSX system Python (2.7.X at the current day). This means that it is possible to install the package and the plugins from a unique location using a python virtual environment, and this will be the preferred method on this platform.

> **Warning:** If another version of Python is in use from the command line, eg. after an installation with Homebrew (either Python2 or Python3), it would mean that you may fall in the case mentioned earlier, and it would be safer to make the OSX system Python the default in that case.

The installation of Paraview is often located in `/Application`` folder, and it is not possible to write into the Paraview application folder. Therefor the installation on OSX requires a virtual environment or a global installation.

Linux

The Linux installation is either:

1. the one that comes with the operating system package manager, such as *aptitude*. In that case, it is likely that this installation uses the system Python and you will be using the same Python interpreter for Paraview and your package installation,
2. or the one that can be downloaded from the Paraview website, that is often more recent, but that also may use an embedded/shipped Python interpreter that will differ from the one of your operating system.

In first case, the procedure with virtual environments can be safely followed. In the second case, and **if the version of Python is different than the one of the operating system**, an additional step outside of the virtual environment should be taken:

1. Suppose the standalone version of Paraview is located `$PARAVIEW_PATH`, and the visualization package has been cloned to `<brain-connectivity-visualization-path>`
2. Then install the package within the Paraview Python environment by typing the following in a terminal

```
cd <brain-connectivity-visualization-path>
$PARAVIEW_PATH/bin/pvpython setup.py install
```

The previous commands will install the package inside the Python installation shipped with Paraview. In this case the virtual environment installation is not necessary, as Paraview will be using only the package installed in its own environment. In all cases, the registration of the plugin remains the same.

you may need the virtual environment installation described above only if you need to perform some processing using the utility functions from a regular command line. The package will then have two installation location: the Paraview Python environement and your virtual environment.

CHAPTER 3

Paraview plugin

We describe in this page how to use the plugin inside Paraview:

# the plugin acts on a mesh of point representing the surface of the brain # several visualization algorithm and parameters are shown in the user interface.

3.1 Input data

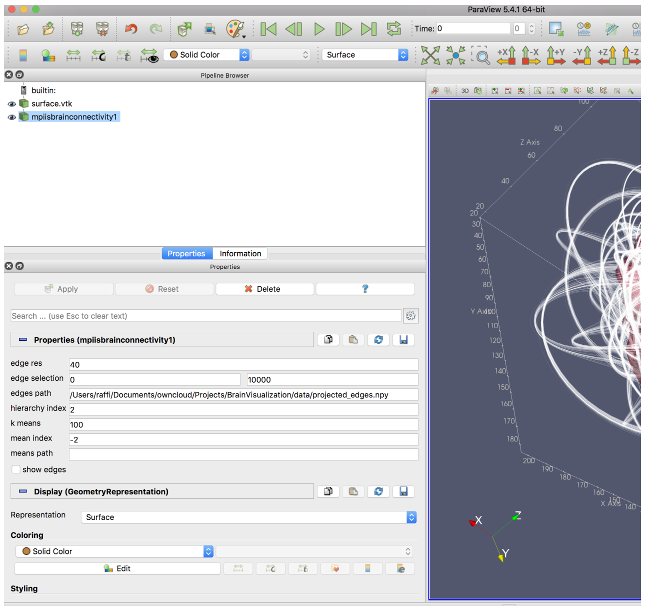
In order to function, the visualization needs:

- the mesh on which computation will be performed. This is usually a mesh file that Paraview can understand,
- the connectivity file, that contains the topology of connection network in the brain. This connectivity file is a mapping between pair of mesh locations and a weight.

The way the connectivity file is created is detailed in another section.

3.2 Parameters

The parameters exposed to the user by the plugin are shown in the picture below:



Properties (mpiisbrainconnectivity1)

edge res	40	
edge selection	0	10000
edges path	/Users/raffi/Documents/owncloud/Projects/BrainVisualization/data/projected_edges.npy	
hierarchy index	2	
k means	100	
mean index	-2	
means path		

☐ show edges

Display (GeometryRepresentation)

Representation: Surface

Coloring: Solid Color

The meaning of the parameters are:

- `Edge res`: indicates the edges resolution, ie. the number of intermediate steps a trajectory connecting two sites is divided into. The higher, the smoother the trajectory but also the heavier the on-screen drawing,
- `Edge selection`: a range of edges on which the processing will be performed. Any edge falling outside of this range is ignored in the visualization and the computation,
- `Edge path`: the **absolute** path of the file containing the definition of the topology,
- `show edges`: if checked,

CHAPTER 4

Computation utilities

Along the paraview plugin, several functions have been developed. Those utilities have different purposes:

- make off-line computations in order to speed-up the visualization
- support the writing of new visualization functions

4.1 Off-line computations

4.1.1 kMeans and hierarchical kMeans

The kMeans algorithm currently being in use within the plugin is a naive implementation using a distance function that is specific to the needs of the visualization. Accelerated algorithms exist such as the *[Elkan]* and that involves a lower number of computations and distance estimations. The complexity of the kMeans is at best polynomial in the number of points.

The hierarchical version of the kMeans runs the kMeans at each level of the hierarchy, and has also a complexity

4.1.2 Alignment

The data to visualize should be in the correct format in order to use the plugin.

4.2 Bibliographic references

4.3 References

This package contains a set of python tools for assisting in the visualization of brain connectivity information. The package contains:

- a Paraview plugin, that is a parametrizable plugin running in a Paraview visualization pipeline. This plugin shows the connectivity of the brain sites, in a raw or summarized fashion,
- utility functions for converting files and performing off-line computation on the brain connectivity

CHAPTER 5

References

This package was developed by Lennart Bramlage and Raffi Enficiaud in the *Software Workshop* Central Scientific Facility of the Max Planck Institute for Intelligent Systems, in Tübingen, Germany.

CHAPTER 6

Installation

The installation procedures are described on the page *Installation*.

CHAPTER 7

How to use the plugin

The usage of the plugin is detailed in the page *Paraview plugin*.

CHAPTER 8

Utility function

The additional tools are described in the page *Computation utilities*.

CHAPTER 9

License

The code contained in this repository is released under the MIT license and copyright is owned by

- Max Planck Society
- Lennart Bramlage
- and Raffi Enficiaud.

See the *LICENSE.txt* file at the root folder of the repository for more information.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Bibliography

[Elkan] Elkan: Using the triangle inequality to accelerate k-means, ICML‘03