PyLocator Documentation

Release 0.30

Thorsten Kranz

April 11, 2012

CONTENTS

CHAPTER

ONE

LOCALIZATION OF EEG-ELECTRODES FROM MRI-VOLUMES

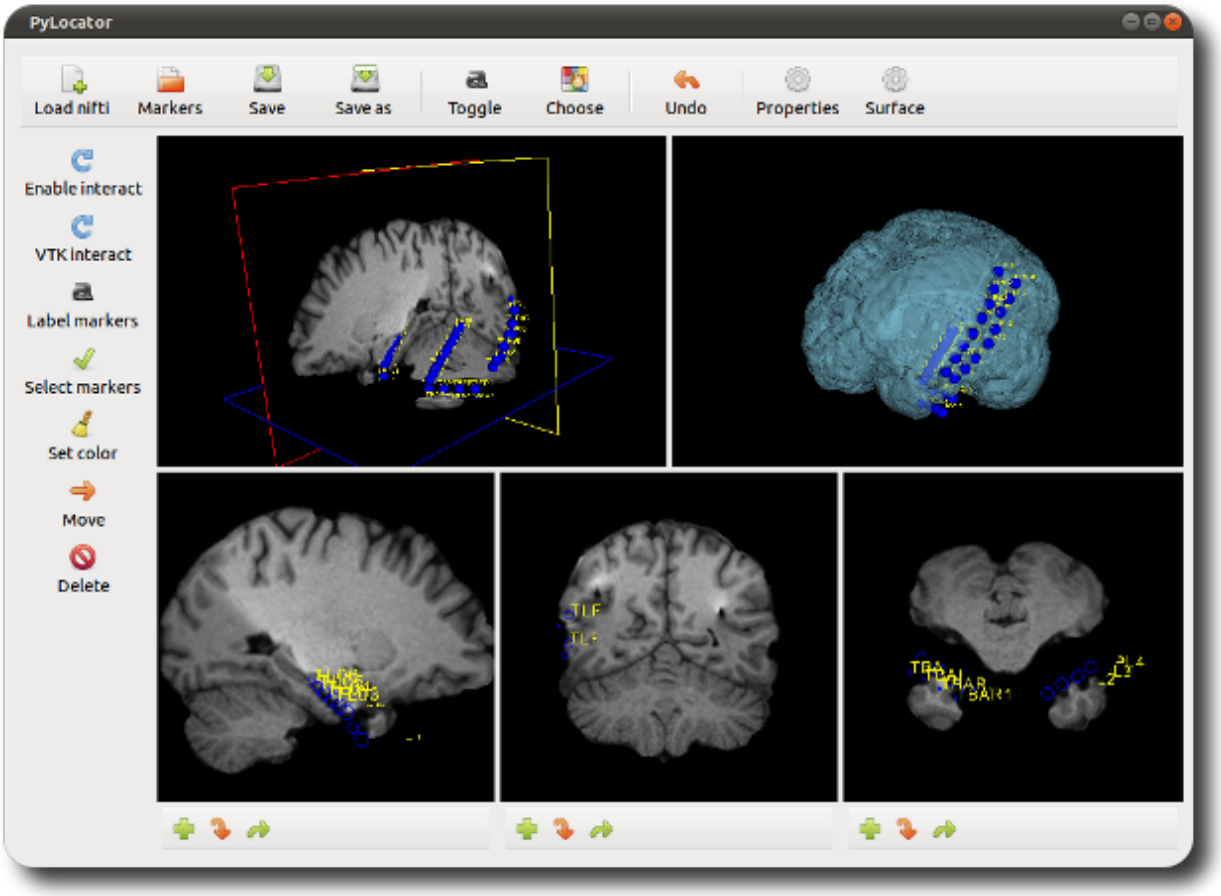
PyLocator is a little program for localizing EEG electrodes from MR-images. It uses VTK to show a neat GUI for marking electrode locations in 3d-space.

It is based on previous work by John D. Hunter and Michael Castell as part of pbrain, now maintained by Eli Albert. PyLocator works as a stand-alone program without any dependencies from pbrain.

CHAPTER

TWO

SCREENSHOT



CHAPTER

THREE

CONTENTS

3.1 Installation

3.1.1 Dependencies

PyLocator relies on a bunch of libraries:

- VTK: 3d-visualization
- nibabel: Reading the Nifti-format for MRI data
- NumPy: Sophisticated array-types for Python
- GTK+: For the GUI.
- GTK+ OpenGL Extension

On a Debian-like environement, these dependencies can usually be resolved via a simple:

```
sudo apt-get install python-vtk python-nibabel python-numpy python-gtk2 python-gtkglext1
```

This should prepare your system for PyLocator. On Windows and OS X, things are a little bit more complicated, but Python distributions like EPD or Python(x,y) should be helpful here. The main problem will be to get **gtkgtlext** working. If you have any hints, e.g., binary packages, please let me know.

Depending on your configuration, nibabel has to be downloaded separately.

Can you help with detailed instructions for these operating systems? Please tell me!

3.1.2 How to download

The source code of PyLocator is kept in a public GIT repository:

http://www.github.com/nipy/PyLocator

You can simply clone this repository via:

```
git clone git://github.com/nipy/PyLocator.git
```

Alternatively, you can download a tarball that is updated once in a while from

http://pylocator.thorstenkranz.de/download/pylocator-0.30.dev.tar.gz

Extract this archive using your preferred archive manager or in a terminal using something like:

```
tar xfvz pylocator-0.30.dev.tar.gz
```

3.1.3 Installing

Once you obtained the source code, enter the PyLocator-directory and type:

```
python setup.py build
python setup.py install --user # For per-user installation
# or
sudo python setup.py install # system-wide installation
```

After these steps, the package *pylocator* should be properly installed. You can then run the program by running:

```
python ~/.local/lib/python2.?/site-packages/pylocator/pylocator.py
```

in case of a per-user installation or:

```
python /usr/local/lib/python2.?/site-packages/pylocator/pylocator.py
```

or similar in case of a system-wide installation. Replace the *2.?* by your python version number.

This solution isn't perfect yet, I'll clean it up soon. Of course you can create some little bash-script that calls this for you and put it into ~/bin/pylocator or similar:

```
#! /bin/bash
python ~/.local/lib/python2.?/site-packages/pylocator/pylocator.py $@
```

3.2 Tutorial

This section shows a basic usage-example and thus introduces the main concepts of Pylocator. It is arranged as a step-by-step instruction and uses a dataset which is available here: post2std_brain.nii.gz.

3.2.1 Loading the MR image

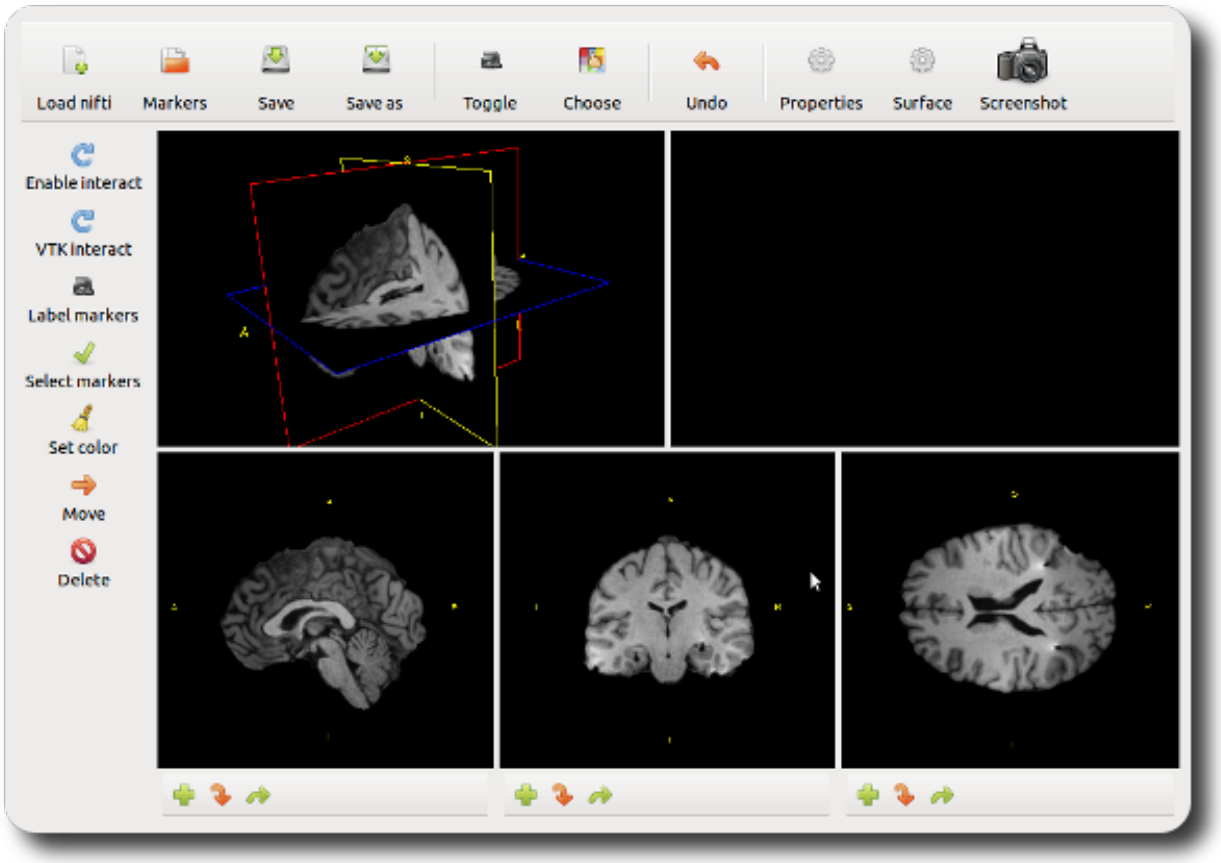
After starting the program (refer to *Installation* for details) you'll first see a dialog prompting for a Nifti-file. Choose the "post2std_brain.nii.gz" file, downloaded from above.

Note: The MR image is a post-implantation image containing two temporal depth electrodes, temporo-lateral strip electrodes and temporo-basal electrodes on either side.

It was preprocessed using FSL:

1. first, a pre-implantation image was brain-extracted and normalized to a MNI152_T1_1mm-template. The transformation matrix was saved.
2. then the post-implantaion image was normalized to the pre-implantation image, also saving the matrix
3. these two matrices where then concatenated; using this new matrix the post-implantation image was transformed to the MNI template.
4. finally, the image was brain-extracted.

Then the MRI is loaded and the main window of PyLocator pops up. It consists of two toolbars (a main toolbar at the top and a secondary at the left side) and 5 subwindows. Four of these windows come to live immediately..


Load nifti
Markers
Save
Save as
Toggle
Choose
Undo
Properties
Surface
Screenshot
Enable interact
VTK interact
Label markers
Select markers
Set color
Move
Delete

The lower row shows three slice views of the MR volume (let's call them *slice widgets* from now on). Exactly the same cut planes are also visible in the upper-left subwindow (subsequently called *planes widget*), but arranged in a 3d setting.

3.2.2 Controlling the views

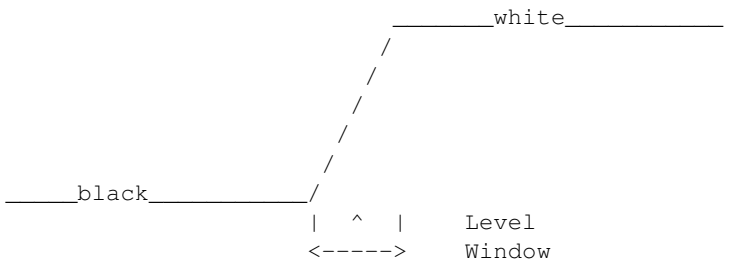
Getting used to interacting with these widgets is maybe the hardest time using PyLocator. It's typical VTK style: you'll need all three mouse buttons and your keyboard, and their respective behaviour depends on where you use them. But just go ahead and try, once you get a feeling, it works like a charm.

While you're in "Interact Mode" (button *Enable Interact* in the left toolbar, default), the controls basically are:

Planes widget

Left mouse button When clicked on one of the planes, the corresponding coordinates and intensity are displayed. If you drag this button somewhere not on any plane, you can rotate the virtual camera around the planes object. If you additionally hold down the Ctrl-Key, the camera will roll around its viewing direction.

Right mouse button When dragged on-plane, you can adjust the thresholds of the colormap for all planes. Drag left/right for adjusting the **level** (center of the colormap-range), drag up/down to change the **window** (width of the colormap-range). To clarify this concept, let's look at a little "diagram":



When dragged off-plane, you can move your camera into / out of the scene.

Middle mouse button When dragging on-plane you can move the planes. Dragging at the edges rotates the plane. Off-plane you can translate the focal point of the camera.

Slice widgets These slices are automatically synchronized to the cut planes in the planes widget. The right and middle mouse button behave mailny the same as *off-plane* in the plane widget. Additionally, you can use the mouse wheel to slowly move the slices back and forth.

3.2.3 Marking electrodes

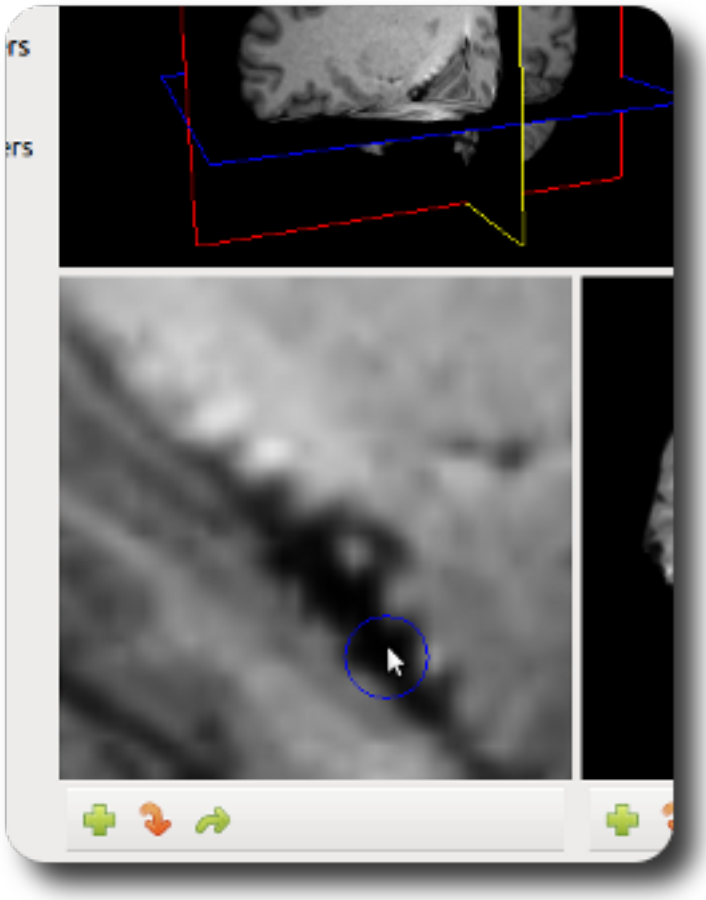
PyLocator uses Markers to help in the localization of electrodes. Every Marker is represented as a small sphere in 3d-space. In the plane widget you can see these spheres, while in the slice widgets they are visible as small circles (if the slice intersects the sphere).

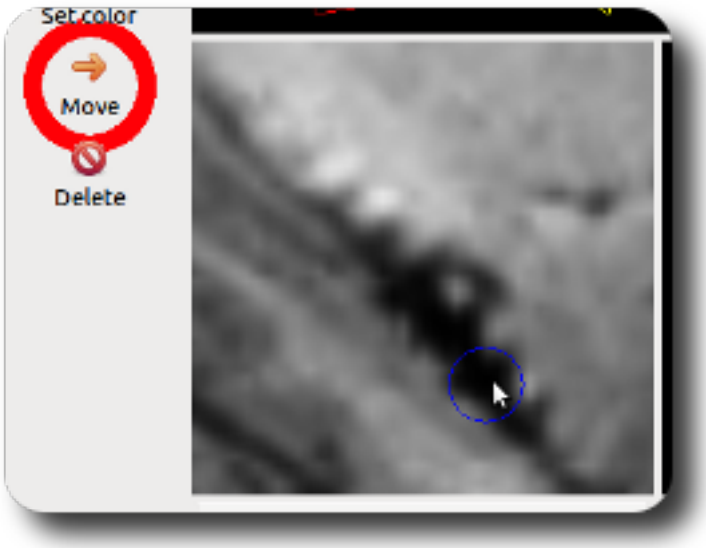
Inserting markers

First, move the cut planes so that you can see the electrode you want to mark. Use the controls as described above to zoom in. Move you mouse above the electrode location (in one of the slice widgets) and press the button **I** (as in *insert*) on your keyboard. A blue circle appears,

Congratulations! You just inserted your first electrode marker.

If you aren't satisfied with it's location, use the "Move" tool to adjust it




Set color
Move
Delete

You might have noticed that after inserting markers, also the fifth subwindow isn't empty anymore. You can see the little spheres also there. Let's call it the **surface widget**. We will explain it later.

Labeling the markers

After having marked some electrodes, you can assign labels to the electrodes. Use the corresponding tool from the toolbar, and the click on one of the spheres in the plane widget or one of the circles in the slice widgets.

A little dialogue will pop up, you can enter a label, hit O.k. Afterwards, the label shows up as yellow text next to the marker.

You can reedit the labels anytime later using the same method.

Saving markers to file

Finally, when you want to export the electrode locations, you can save them as a simple text file to disk. Use the **Save to**-button from the main toolbar. Choose a directory and filename and your done.

electrode_locations.txt:

```
TBPR1,22.3044795975,-5.109097651,-37.8967504764,3.0,0.0,0.0,1.0
TBPR2,31.4296973957,-10.476872826,-37.8967504764,3.0,0.0,0.0,1.0
TBPR3,40.0181376763,-16.918203037,-37.8967504764,3.0,0.0,0.0,1.0
TBPR4,49.1433554745,-22.82275573,-37.8967504764,3.0,0.0,0.0,1.0
TL01,-25.062664026,-6.375814565,-29.9888764281,3.0,0.0,0.0,1.0
TL02,-25.632289616,-10.015602878,-26.9951491745,3.0,0.0,0.0,1.0
TL03,-26.63892536,-13.301689266,-23.680670446,3.0,0.0,0.0,1.0
TL04,-27.604637459,-17.54492081,-20.3931495947,3.0,0.0,0.0,1.0
TL05,-28.415850937,-21.109297964,-17.1291834355,3.0,0.0,0.0,1.0
...
```

The first column contains the labels you assigned to the markers, the next three columns are the indices / coordinates. Columns 5 to 8 can be ignored, they contain the marker size and its color.

Note: In v0.1 of PyLocator, the affine transformation was not applied for visualization. It was rather used at save time to create a second file. If you decided to use "electrode_locations.txt", the file "electrode_locations.txt.conv" used to be created in the directory. While the first one contained the voxel-indices where the marks were set (as floats, due to interpolation) the second one had the coordinates in scanner space [1] as obtained by the affine transform stored in the Nifti file.

Here is an example how the old files might have looked like:

electrode_locations.txt:

```
TBPR1,67.6955204025,120.890902349,34.1032495236,3.0,0.0,0.0,1.0
TBPR2,58.5703026043,115.523127174,34.1032495236,3.0,0.0,0.0,1.0
TBPR3,49.9818623237,109.081796963,34.1032495236,3.0,0.0,0.0,1.0
TBPR4,40.8566445255,103.17724427,34.1032495236,3.0,0.0,0.0,1.0
TL01,115.062664026,119.624185435,42.0111235719,3.0,0.0,0.0,1.0
TL02,115.632289616,115.984397122,45.0048508255,3.0,0.0,0.0,1.0
TL03,116.63892536,112.698310734,48.319329554,3.0,0.0,0.0,1.0
TL04,117.604637459,108.45507919,51.6068504053,3.0,0.0,0.0,1.0
TL05,118.415850937,104.890702036,54.8708165645,3.0,0.0,0.0,1.0
...
```

[1] If you normalized the MRI image to some standard brain (the tutorial file is normalized to MNI152, T1, 1mm voxel size) these coordinates where in standard space, e.g. MNI coordinates

If you reload the same image again sometime later, you can also load these files back into the program to recover all markers.

Note: When loading markers from disk, be careful with old files: if they where created with PyLocator version < 0.2, **do choose the .conv-file**. Otherwise the locations will be messed up

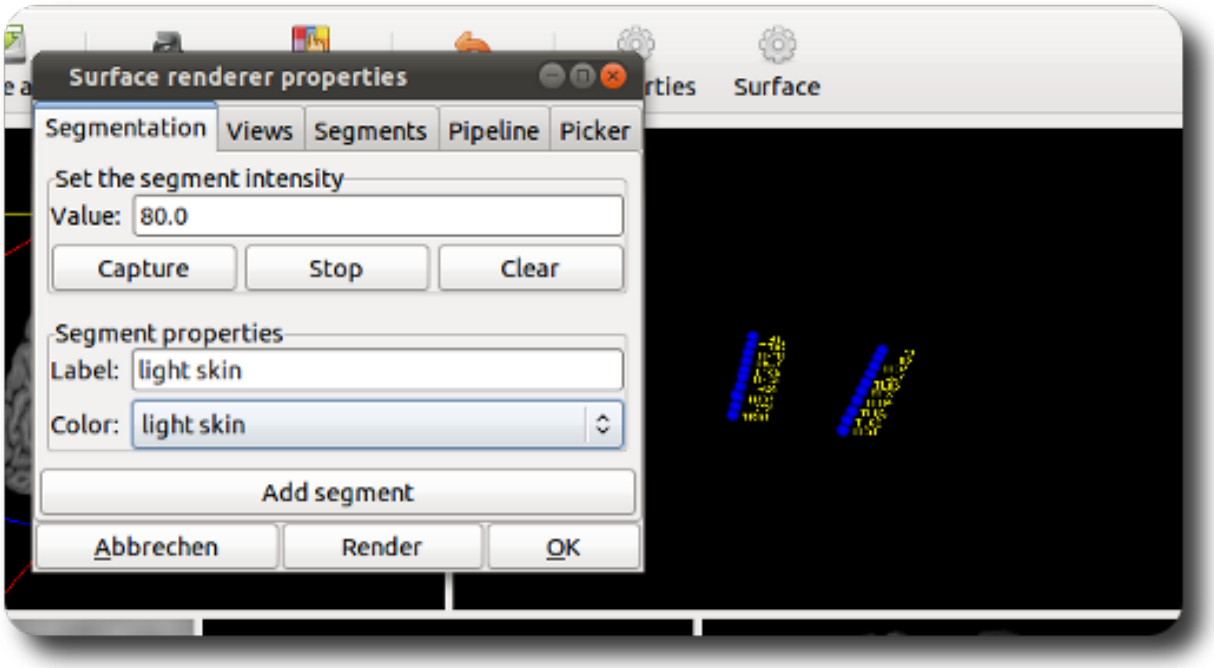
3.2.4 Rendering 3d-surface

Now let's move our attention to the upper right subwindow, the **surface widget**. You already see some electrode markers inside it and can use the same controls as for the planes widget (only setting window/level doesn't make sense here).

We use this subwindow to render iso-surfaces for our volumetric data. [2] This is especially helpful for locating

1. subdural electrodes, like strips and grids, in brain extracted MR images and
2. surface-electrodes in simultaneous EEG / fMRI experiments.

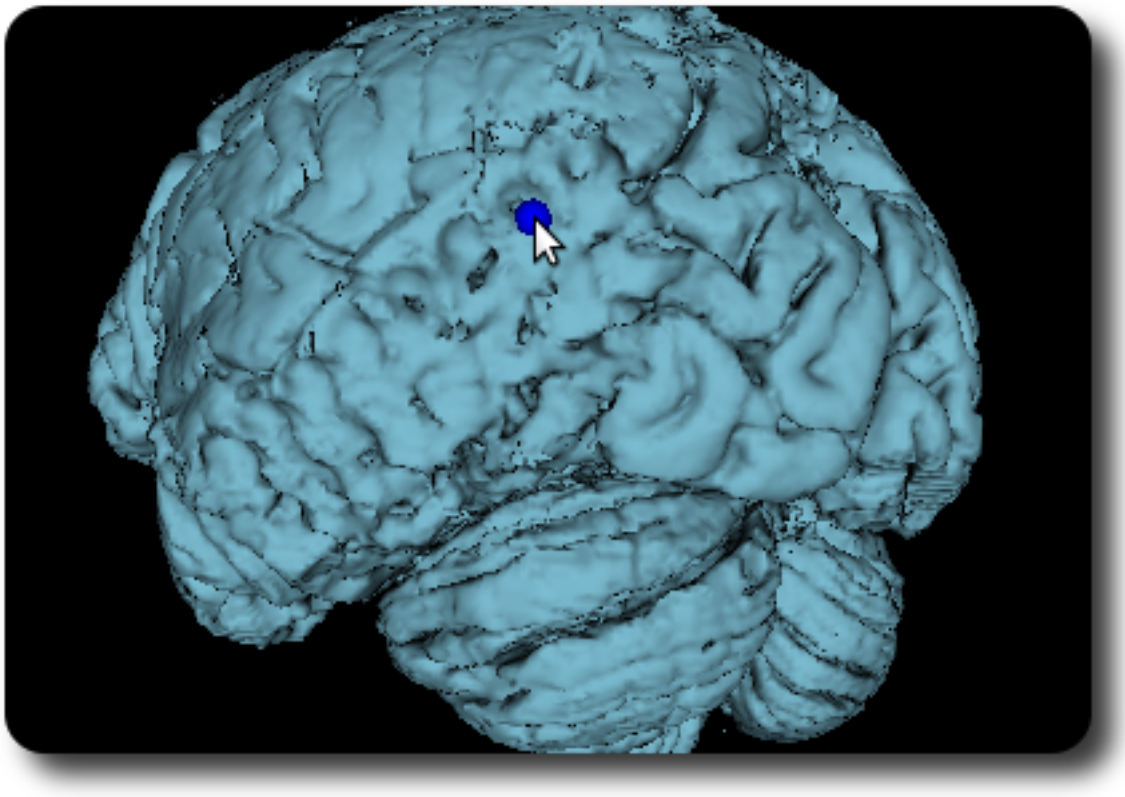
To create a iso-surface, choose the button "Surface" from the main toolbar. A dialogue shows up where we can make all necessary settings. We have to choose a threshold value for the iso surface. For now, you can accept the default value.



Click on "Add segment" (we could render more than one iso surface, but we won't do for now) and then "Render". After a short while, you'll see your iso surface inside the surface widget. You can see the gyri and sulci and - if you search a little bit - you can find the locations of the subdural electrodes as additional "bumps".

What comes in handy now is that you can insert makers also here just as you can inside the slice widgets: Move the mouse cursor above the "bump" you want to mark and hit **I**. Another little sphere appears, just at the point on the surface you were pointing at.

[2] Iso surfaces are the 3d analogy to contour plots.



Again, you can correct the marker locations within the slice widgets, label them in the planes widget or one of the slice widgets, and finally save all markers to disc.

3.2.5 Taking screenshots

A feature recently added to PyLocator is its ability to take screenshots of the 3d-widgets. In contrast to using an external program for doing so, we can achieve a higher quality using VTK.

Click on the button "Screenshot" in the main toolbar. A dialog appears (see above). Here, you can pick a filename pattern (**it is important to keep the %03i within the pattern, as an automatically incremented counter is added here**). You can also have a pattern proposed by PyLocator, is will be based on the name of the MRI Nifti file.

Next, choose your desired magnification. The currently rendered images in each widget will be resampled accordingly by VTK, resulting in a higher resolution than a-posteriori resizing a normal screenshot.

You can use the buttons in the dialog to take photos of individual widgets or of all widgets.

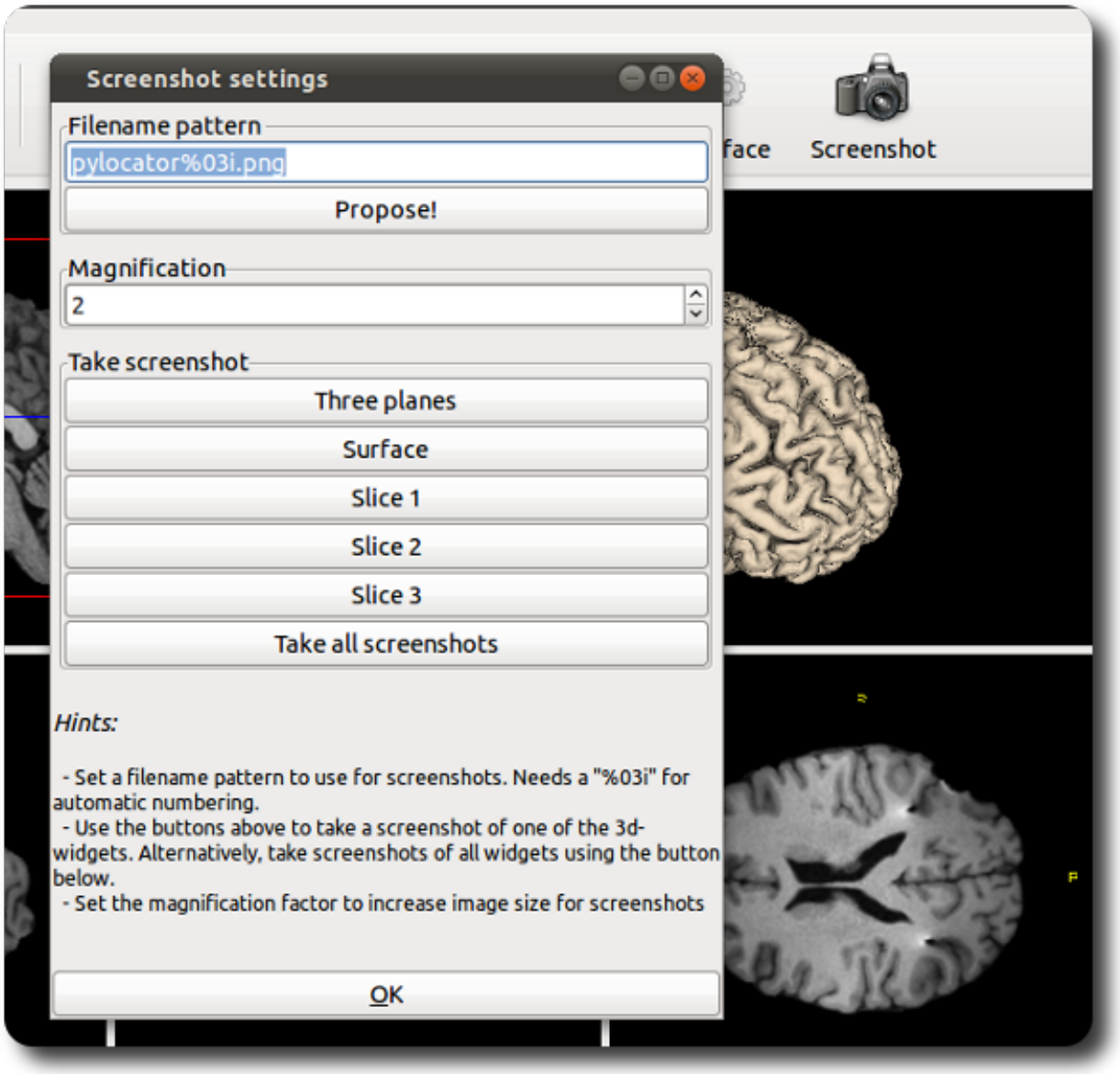
3.3 Frequently Asked Questions

Under construction...

Information on how to obtain and configure PyLocator correctly can be obtained in the *Installation* section.

To get a feeling of what PyLocator can do and in order to get used to the controls, we recommend our little *Tutorial*.

The *Frequently Asked Questions* will start with some little basis of obvious questions and will then grow with time.


Screenshot settings
Filename pattern
pylocator%03i.png
Propose!
Magnification
2
Take screenshot
Three planes
Surface
Slice 1
Slice 2
Slice 3
Take all screenshots
Hints:
- Set a filename pattern to use for screenshots. Needs a "%03i" for automatic numbering.
- Use the buttons above to take a screenshot of one of the 3d-widgets. Alternatively, take screenshots of all widgets using the button below.
- Set the magnification factor to increase image size for screenshots
OK
face
Screenshot

CHAPTER

FOUR

LICENSE

PyLocator is distributed under a BSD-style license, and thus is free software (both in the sense of free beer and free speech). Details can be found in the LICENSE-file distributed with the source code.

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*