Anatomist: a python framework for interactive 3D visualization of neuroimaging data

D Rivière[1,2], D Geffroy[1], I Denghien[1], N Souedet[1,3], Y Cointepas[1,2]

[1]IFR 49, Gif-sur-Yvette, France

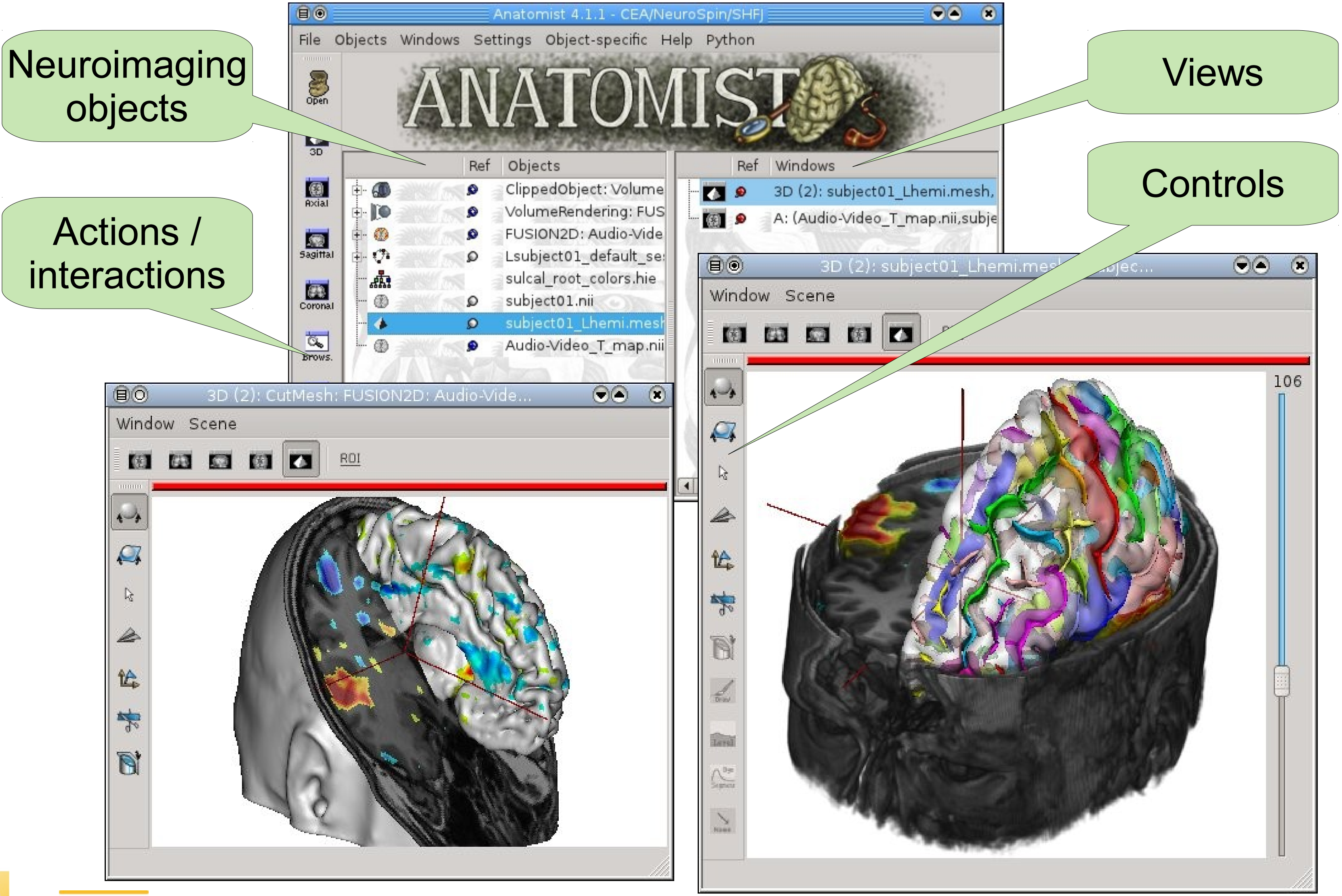
[2]CEA, I2BM, Neurospin, Gif-sur-Yvette, France

[3]CEA, I2BM, MIRCEN, Fontenay-aux-Roses, France

http://brainvisa.info/doc/pyanatomist/sphinx/

Anatomist : overview

Free software: - Anatomist (C++ libraries) is BSD-like
- PyAnatomist is GPL-like (due to PyQt)



Neuroimaging objects

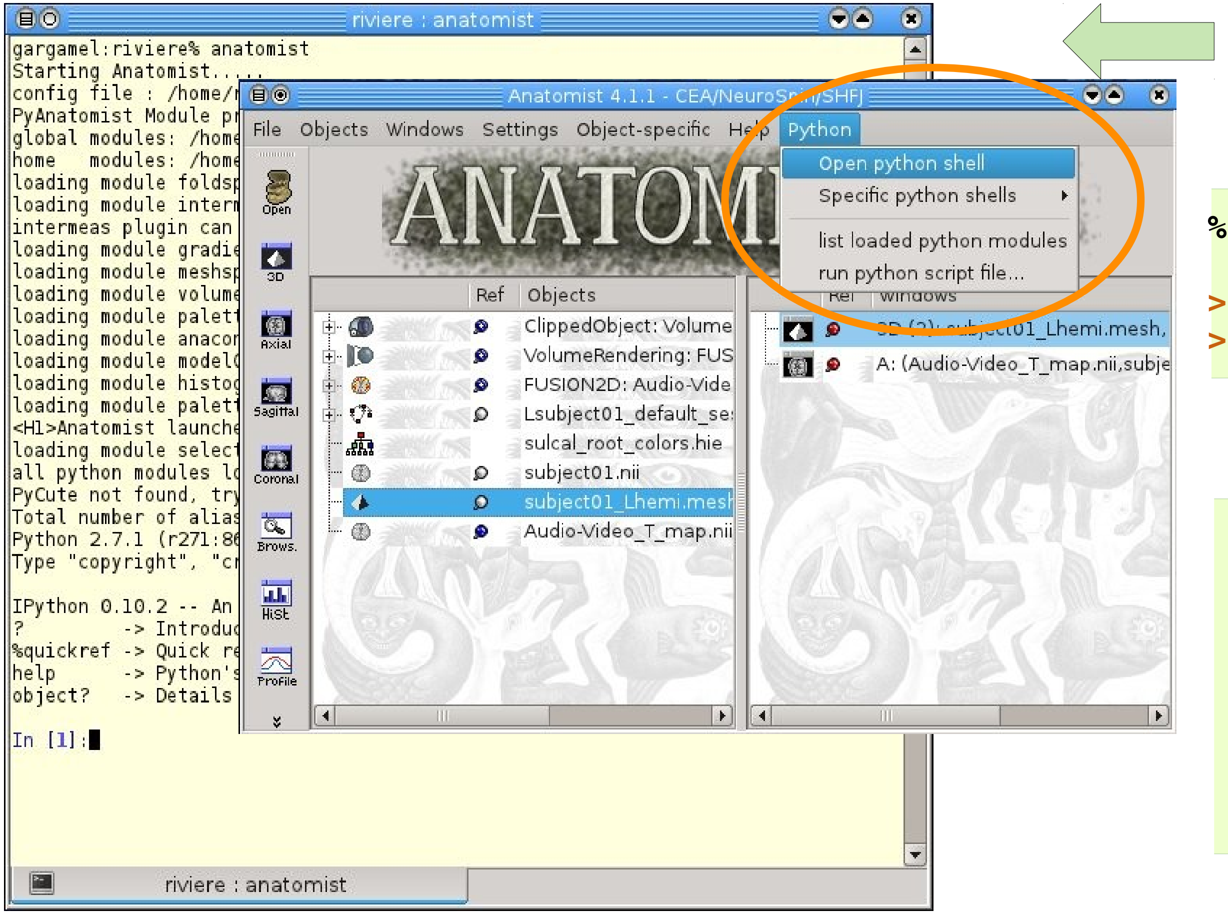
Actions / interactions

Views

Controls

Python scripting

Several ways to enter python scripting mode



```
gargamel:riviere% anatomist
Starting Anatomist.....
config file : /home/r
PyAnatomist Module pr
global modules: /home
home   modules: /home
loading module foldsp
loading module interm
intermeas plugin can
loading module gradie
loading module meshsp
loading module volume
loading module palett
loading module anacor
loading module model(
loading module histog
loading module palett
<H1>Anatomist launche
loading module select
all python modules lo
PyCute not found, try
Total number of alias
Python 2.7.1 (r271:86
Type "copyright", "cr

IPython 0.10.2 -- An
?         -> Introduc
%quickref -> Quick re
help      -> Python's
object?   -> Details

In [1]:
```

- From Anatomist: python menu
- From IPython (use -q4thread option, or –gui=qt with IPython >= 0,11):

```
% ipython -q4thread

>>> import anatomist.api as anatomist_api
>>> a = anatomist_api.Anatomist()
```

- Running a script with a Qt event loop

```
# !/usr/bin/env python

import anatomist.api as anatomit_api
import sys
from PyQt4 import QtGui

qapp = QtGui.QApplication( sys.argv )
a = anatomist_api.Anatomist()
QtGui.qApp.exec_()
```

- In a Python plugin for Anatomist

```
Location : $HOME/.anatomist/python_plugins
In BrainVisa distributions :
<BV_dir>/share/anatomist-<version>/python_plugins
```

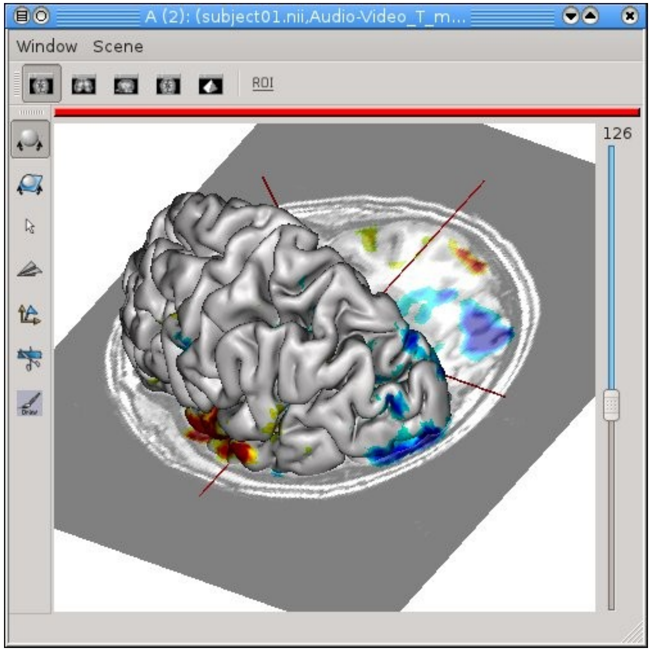
IFR 49

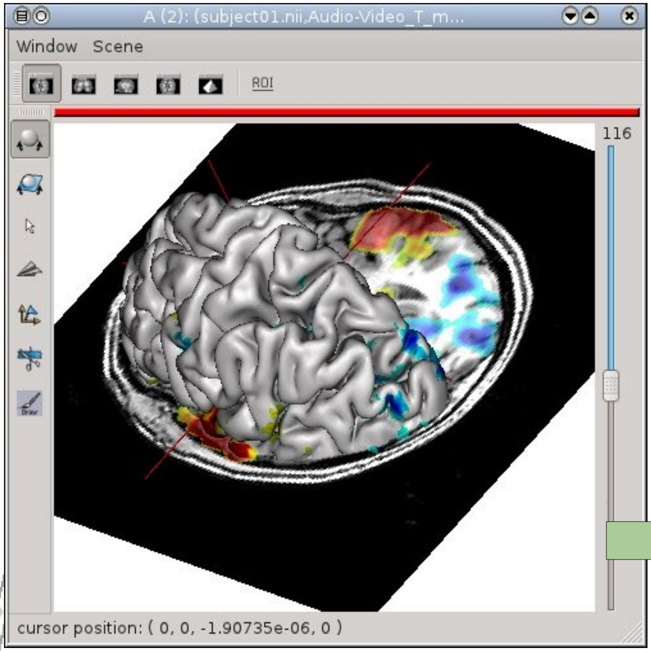
Inserm

The internal commands system

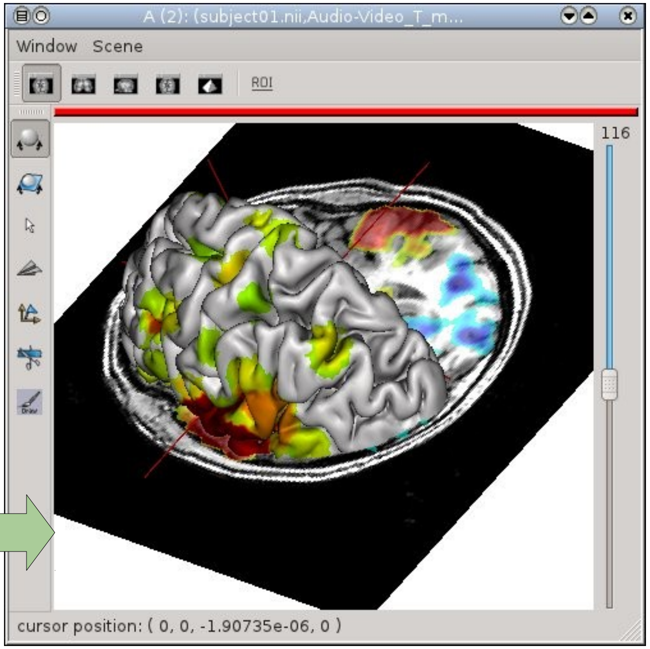
Internal commands system (this interpreter is older than the raise of Python for scientific applications),
http://brainvisa.info/doc/anatomist/html/fr/programmation/commands.html
Most have been ported to the newer Python API

```
>>> a.execute( 'LinkedCursor', window=win,
        position=( 0, 0, 0 ) )
>>> a.execute( 'Fusion2DParams', object=fus2d,
        mode='linear_on_defined', rate=0.5,
        reorder_objects=( vol, vol2 ) )
>>> a.execute( 'Fusion3DParams', object=fus3d,
        method='sphere', submethod='max',
        depth=8., step=0.5 )
>>> a.execute( 'Server', port=40007 )
```





cursor position: (0, 0, -1.90735e-06, 0)



cursor position: (0, 0, -1.90735e-06, 0)

Python API

A single base API for different control modes:

- Socket: control through a network connection
- Direct: access to the C++ library
- Threaded: thread-safe + direct

```
>>> import anatomist
>>> anatomist.setDefaultImplementation(anatomist.SOCKET)
>>> import anatomist.api as anatomist_api
>>> a = anatomist_api.Anatomist()
```

Or:

```
>>> import anatomist.socket.api as ana
>>> a = ana.Anatomist()
```

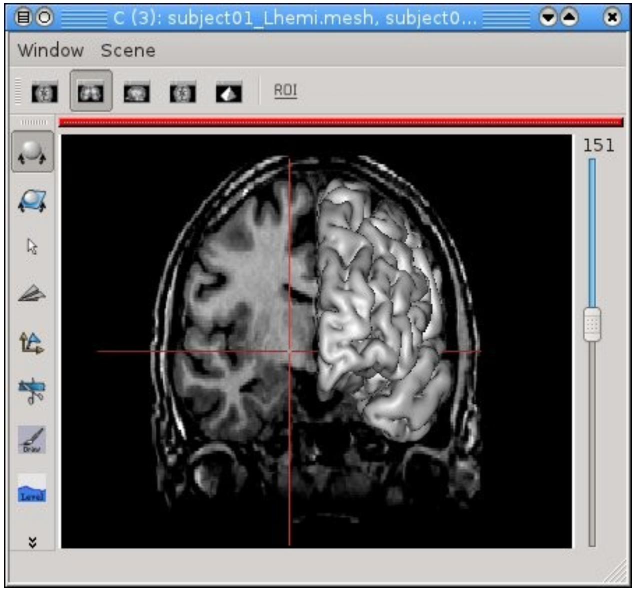
- The direct mode allows additional features (direct memory access)
- The socket mode allows several clients to connect to the same Anatomist, or a client program may pilot several Anatomist instances (on sevral machines)

```
>>> # here we will control 2 anatomist server applications
>>> import anatomist.socket.api as anatomist_api
>>> a1 = anatomist_api.Anatomist() # default is host=localhost, port=40007
>>> a2 = anatomist_api.Anatomist( host='localhost', port=40008, forceNewInstance=True )
>>> w1 = a1.createWindow( '3D' )
>>> w2 = a2.createWindow( 'Browser' )
```

Basic operations

Loading objects, opening views

```
>>> a = anatomist_api.Anatomist()
>>> t1 = a.loadObject( 'volume.nii' )
>>> mesh = a.loadObject( 'mesh.gii' )
>>> win = a.createWindow( 'Coronal' )
>>> win.addObjects( ( t1, mesh ) )
```

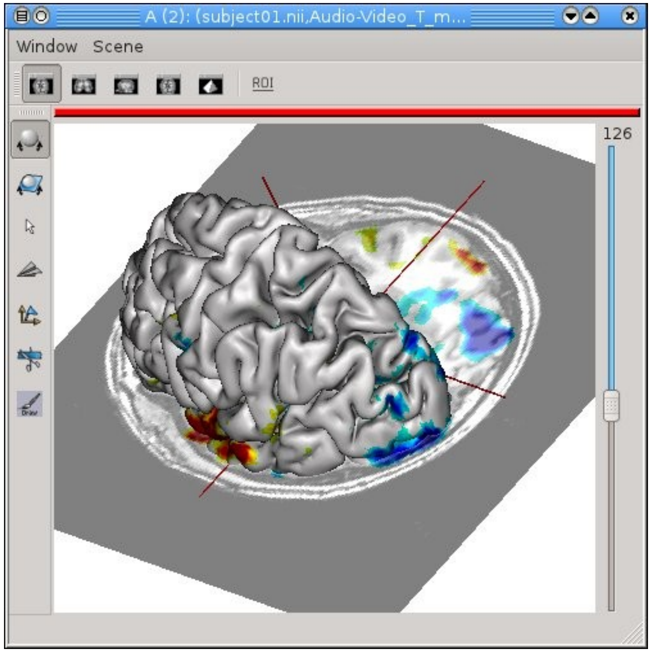


Using colors

```
>>> mesh.setMaterial( diffuse=[ 0.5, 0.5, 1., 0.8 ] )
>>> func = a.loadObject( 'Audio-Video_T_map.nii' )
>>> func.setPalette( 'tvalues100-200-100', MinVal=-4.13,
        maxVal=4.13, absoluteMode=True )
```

The “fusion” system: making new objects

```
>>> # objects may live in different coordinates systems
>>> a.applyBuiltinReferential( ( t1, func, mesh ) )
>>> fus2d = a.fusionObjects( ( t1, func ),
        method='Fusion2DMethod' )
>>> win2 = a.createWindow( 'Axial' )
>>> win2.addObjects( fus2d )
>>> # now another fusion type
>>> fus3d = a.fusionObjects( ( func, mesh ),
        method='Fusion3DMethod' )
>>> win2.addObjects( fus3d )
```



IFR 49

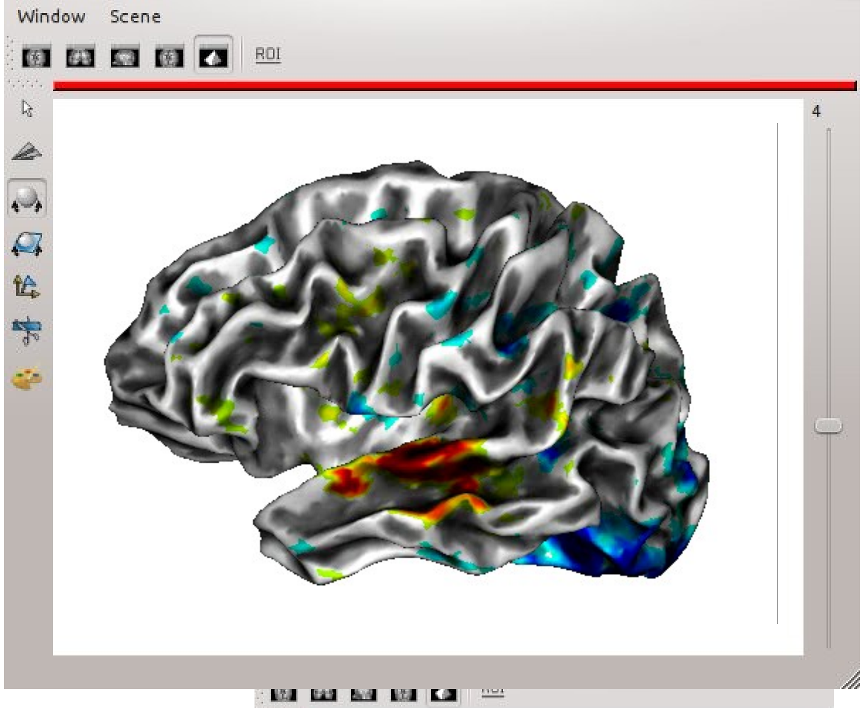
cea

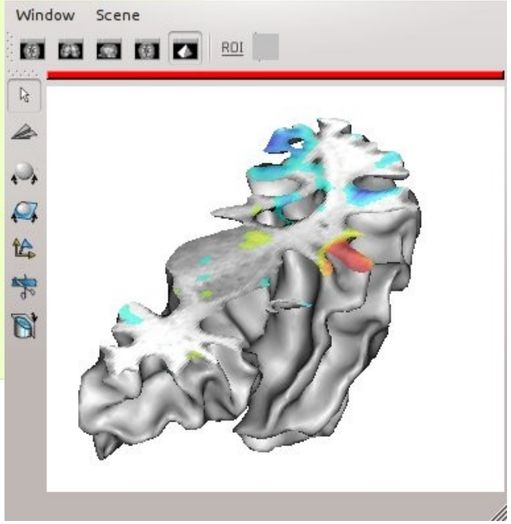
Inserm

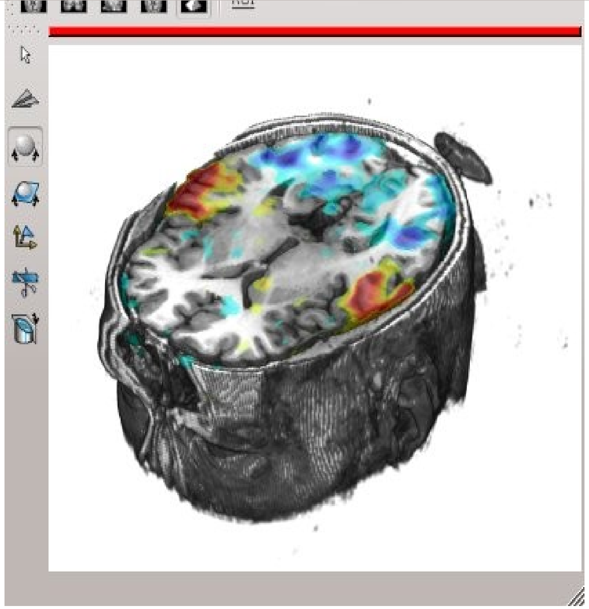
CNRS

Generic “Fusion” mechanism

```
import anatomist.api as anatomist_api

a = anatomist_api.Anatomist()
t1 = a.loadObject( 'subject01.nii' )
white = a.loadObject( 'subject01_Lwhite.mesh' )
func = a.loadObject( 'Audio-Video_T_map.nii' )
inflat = a.loadObject( 'subject01_Lwhite_inflated_4d.mesh' )
curv = a.loadObject( 'subject01_Lwhite_curv.tex' )
a.applyBuiltinReferential( ( white, func ) )
func.setPalette( 'tvalues100-200-100', minVal=-4.13,
    maxVal=-4.13, absoluteMode=True )
curv.setPalette( 'B-W LINEAR', minVal=-0.691, maxVal=0.212,
    absoluteMode=True )
f3 = a.fusionObjects( ( white, func ),
    method='Fusion3DMethod' )
mtex = a.fusionObjects( ( f3, curv ),
    method='FusionMultiTextureMethod' )
tinfl = a.fusionObjects( ( inflat, mtex ),
    method='FusionTexSurfMethod' )
w = a.createWindow( '3D' )
w.addObjects( tinfl )

f2 = a.fusionObjects( ( t1, func ),
    method='Fusion2DMethod' )
cut = a.fusionObjects( ( white, f2 ),
    method='FusionCutMeshMethod' )
volr = a.fusionObjects( ( f2, ),
    method='VolumeRenderingFusionMethod' )
clip = fusionObjects( ( volr, ),
    method='FusionClipMethod' )
w2 = a.createWindow( '3D' )
w2.addObjects( cut )
w3 = a.createWindow( '3D' )
w3.addObjects( clip )
```

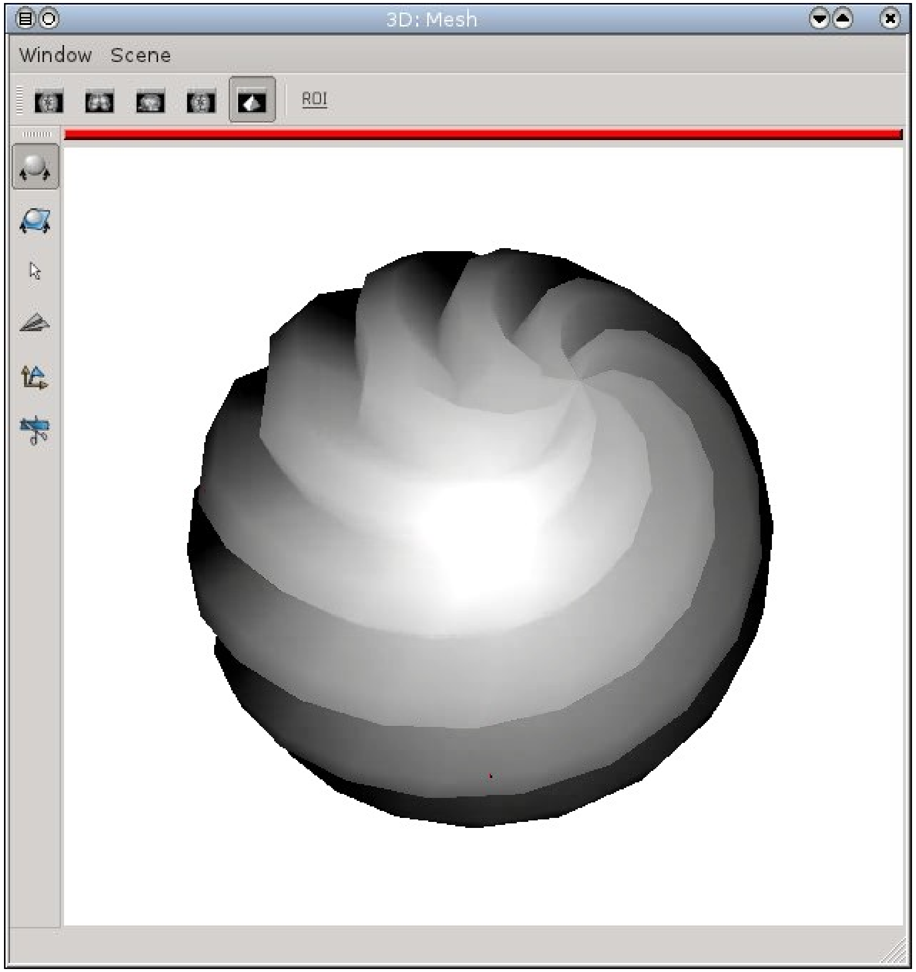






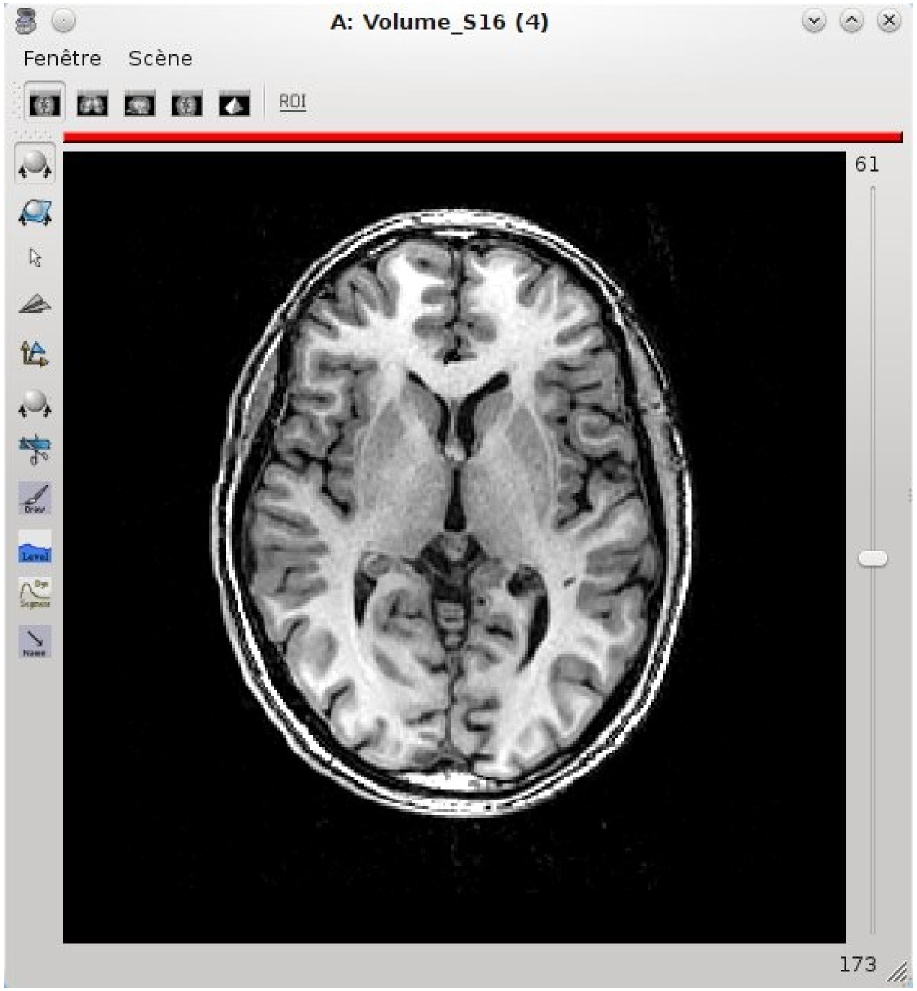
Direct mode features

```
from soma import aims
import time
import os
import anatomist.direct.api as anatomist_api
import sys
from PyQt4 import QtGui

m = aims.SurfaceGenerator.sphere( (0,0,0), 100, 500 )
a = anatomist_api.Anatomist()
# Put the mesh in anatomist
am = a.toAObject( m )
aw = a.createWindow( '3D' )
aw.addObjects( am )
coords = [ aims.Point3df(p) for p in m.vertex() ]
points = xrange( 0, len(coords), 3 )

for i in xrange( 10 ):
  # shrink
  for s in reversed(xrange(100)):
    for p in points:
      m.vertex()[p] = coords[p] * s/100.
    am.setChanged()
    am.notifyObservers()
    QtGui.qApp.processEvents()
    time.sleep( 0.01 )
  # expand
  for s in xrange(100):
    for p in points:
      m.vertex()[p] = coords[p] * s/100.
    am.setChanged()
    am.notifyObservers()
    QtGui.qApp.processEvents()
    time.sleep( 0.01 )
```



Anatomist and numpy

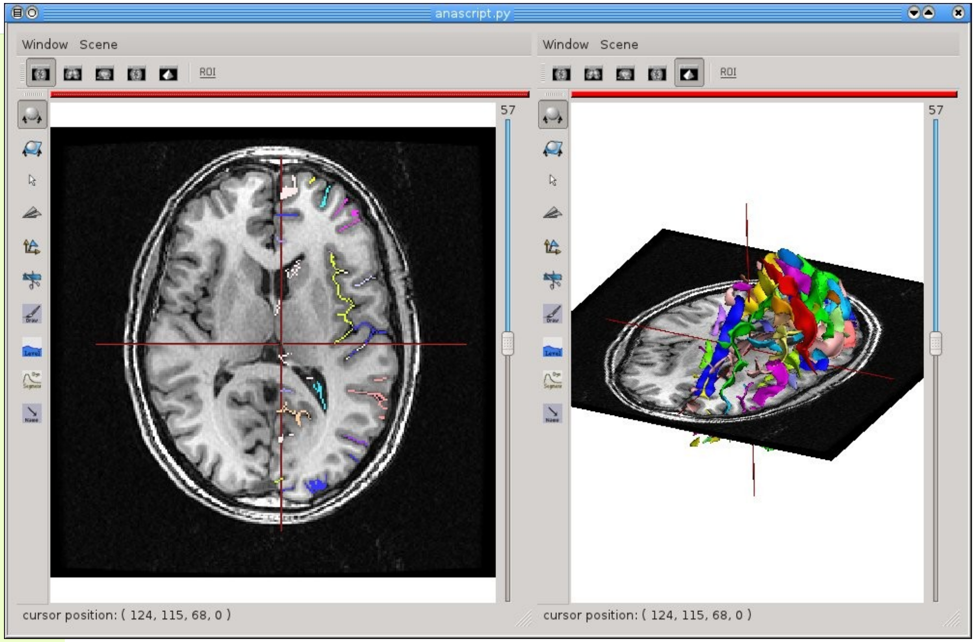
```
import anatomist.direct.api as anatomist_api
from soma import aims
import numpy
from PyQt4 import QtGui

a = anatomist_api.Anatomist()
qApp = QtGui.qApp
vol = aims.read( 'subject01.nii' )
bmask = aims.read( 'brain_subject01.nii' )
masked = aims.Volume( vol )
avol = a.toAObject( vol )
abmask = a.toAObject( bmask )
amasked = a.toAObject( masked )
w = a.createWindow( 'Axial' )
w.addObjects( amasked )
qApp.processEvents()

# get numpy arrays on volume data
abm = numpy.array( bmask, copy=False )
am = numpy.array( masked, copy=False )
av = numpy.array( vol, copy=False )
# masking brain
am[abm==0] = 0
amasked.setChanged()
amasked.notifyObservers()
qApp.processEvents()

# iterative blurring
for i in range( 50 ):
  am[ 1:-1, 1:-1, 1:-1, : ] = ( am[ 1:-1, 1:-1, 1:-1, : ] * 3 \
  + am[ :-2, 1:-1, 1:-1, : ] + am[ 2:, 1:-1, 1:-1, : ] \
  + am[ 1:-1, :-2, 1:-1, : ] + am[ 1:-1, 2:, 1:-1, : ] \
  + am[ 1:-1, 1:-1, :-2, : ] + am[ 1:-1, 1:-1, 2:, : ] ) / 9
  amasked.setChanged()
  amasked.notifyObservers()
  qApp.processEvents()
```



Direct mode and GUI

- Anatomist views internal representations are Qt widgets: using them in custom GUI is allowed

```
#!/usr/bin/env python

import anatomist.direct.api as anatomist_api
from PyQt4 import QtCore, QtGui
import sys

qapp = QtGui.QApplication( sys.argv )
# disable default anatomist main window
a = anatomist_api.Anatomist( '-b' )
# make a custom GUI with 2 views
mainw = QtGui.QMainWindow( None )
grid = QtGui.QWidget( mainw )
mainw.setCentralWidget( grid )
layout = QtGui.QGridLayout( grid )
w1 = a.createWindow( 'Axial' )
layout.addWidget( w1.getInternalRep(), 0, 0 )
w2 = a.createWindow( '3D' )
a.execute( 'LinkedCursor', window=w1, position=(124,115,68) )
layout.addWidget( w2.getInternalRep(), 0, 1 )
mainw.show()
# display something in the views
vol = a.loadObject( 'subject01.nii' )
nomenc = a.loadObject( 'sulci/sulcal_root_colors.hie' )
graph = a.loadObject( \
    'sulci/Lsubject01_default_session_auto.arg' )
a.execute( 'GraphDisplayProperties', objects=[graph],
    nomenclature_property='label' )
a.addObjects( ( vol, graph ), ( w1, w2 ), add_graph_nodes=True )
w2.camera( view_quaternion=[0.508661, 0.133626,
    0.192899, 0.828371], zoom=1.5 )
qapp.exec_()
del w1, w2, graph, vol, a
```



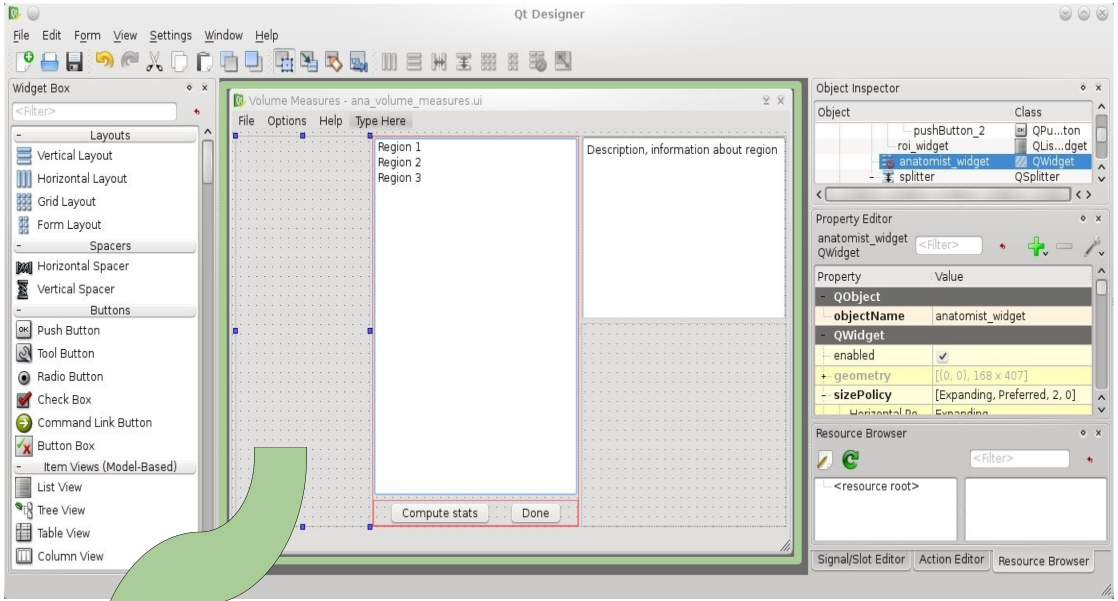
cursor position: (124, 115, 68, 0)

cursor position: (124, 115, 68, 0)

The `getInternalRep()` method of objects/windows grants access to a lower-level API which is the direct bindings to the C++ library API.

Custom GUI using Qt Designer

```
#!/usr/bin/env python

from PyQt4.uic import loadUi
from PyQt4.QtGui import QApplication, QVBoxLayout
import anatomist.direct.api as anatomist_api
Import sys

qapp = QApplication( sys.argv )
ui = 'ana_volume_measures.ui'
mainw = loadUi( ui )
anatomist = anatomist_api.Anatomist( '-b' )

# open an axial window
layout = QVBoxLayout( mainw.anatomist_widget )
ana_window = anatomist.createWindow( 'Axial',
    no_decoration=True )
ana_window.setParent( mainw.anatomist_widget )
layout.addWidget( ana_window.getInternalRep() )

# [...]

mainw.show()
```

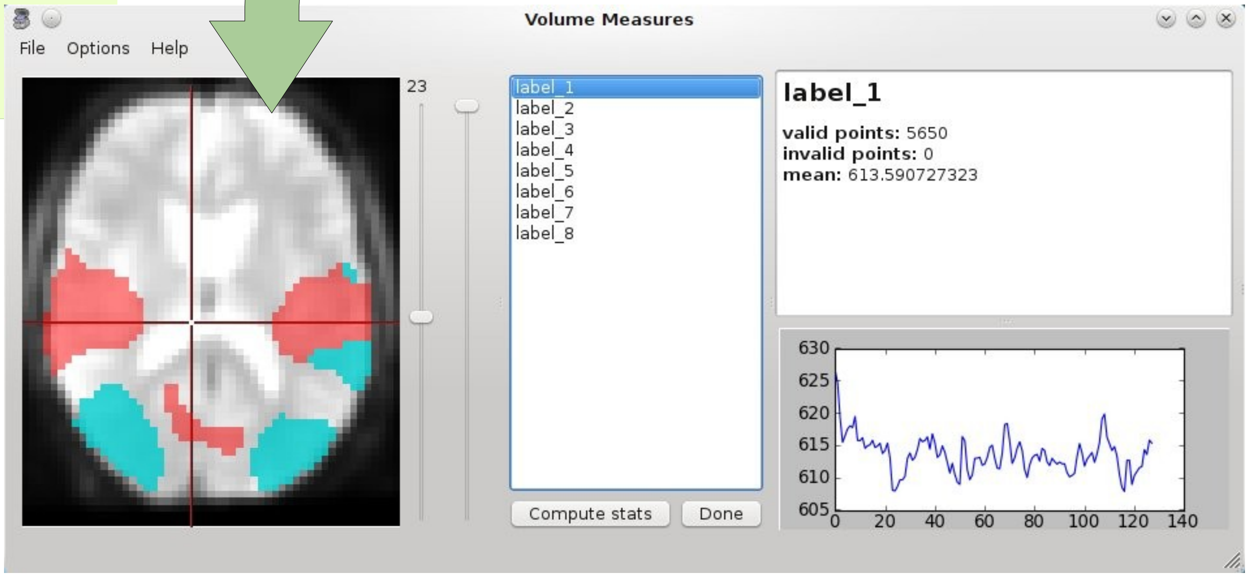


Object	Class
pushButton_2	QPu...ton
roi_widget	QLis...dget
anatomist_widget	QWidget
splitter	QSplitter

Property Editor

anatomist_widget
QWidget

Property	Value
QObject	
objectName	anatomist_widget
QWidget	
enabled	✓
geometry	[(0, 0), 168 x 407]
sizePolicy	[Expanding, Preferred, 2, 0]
Horizontal Po...	Expanding



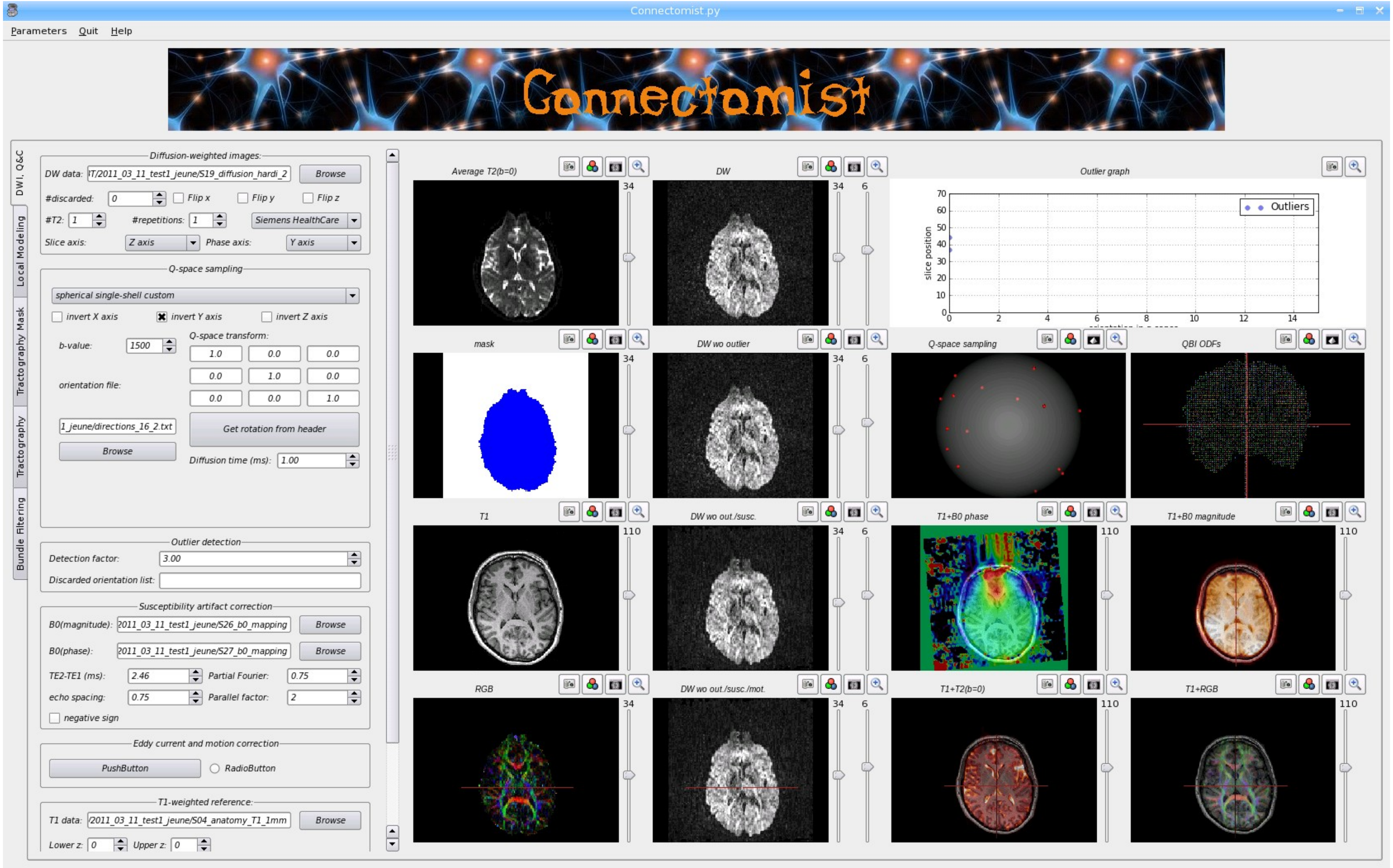
label_1

valid points: 5650
invalid points: 0
mean: 613.590727323

Conclusion

- Anatomist allows many pieces to be combined, python allows to do so quickly and easily.
- Simple programming interface for basic manipulations
- Everything is extensible (down to low-level, via C++ / python classes inheritance)
- Possibility to easily build custom dedicated applications

Conclusion



Diffusion-weighted images:

DW data: T/2011_03_11_test1_jeune/S19_diffusion_hardi_2 Browse

#discarded: 0 □ Flip x □ Flip y □ Flip z

#T2: 1 #repetitions: 1 Siemens HealthCare

Slice axis: Z axis Phase axis: Y axis

Q-space sampling

spherical single-shell custom

□ invert X axis ☒ invert Y axis □ invert Z axis

b-value: 1500

Q-space transform:

1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

orientation file: 1_jeune/directions_16_2.txt Browse

Get rotation from header

Diffusion time (ms): 1.00

Outlier detection

Detection factor: 3.00

Discarded orientation list:

Susceptibility artifact correction

B0(magnitude): 2011_03_11_test1_jeune/S26_b0_mapping Browse

B0(phase): 2011_03_11_test1_jeune/S27_b0_mapping Browse

TE2-TE1 (ms): 2.46 Partial Fourier: 0.75

echo spacing: 0.75 Parallel factor: 2

□ negative sign

Eddy current and motion correction

PushButton ○ RadioButton

T1-weighted reference:

T1 data: /2011_03_11_test1_jeune/S04_anatomy_T1_1mm Browse

Lower z: 0 Upper z: 0

(image: courtesy of C. Poupon et al.)

Any questions ?