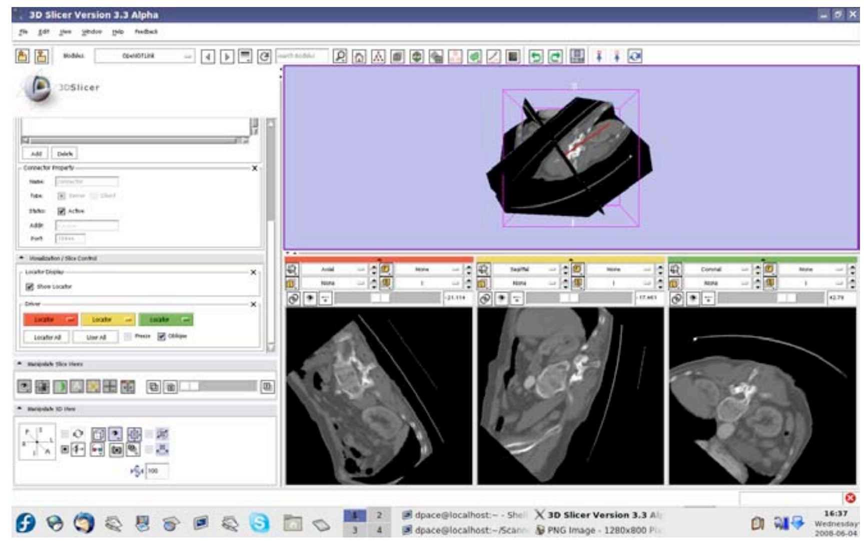
# Image Guided Therapy in Slicer3

Advanced Tutorial on Navigation using OpenIGTLink




Danielle Pace, B.CmpH

# Acknowledgements

**Surgical Planning Lab, Harvard Medical School**

Junichi Tokuda, Haiying Liu, Nobuhiko Hata, Steve Pieper, Ron Kikinis

**National Alliance for Medical Image Computing**

**National Center for Image-Guided Therapy**

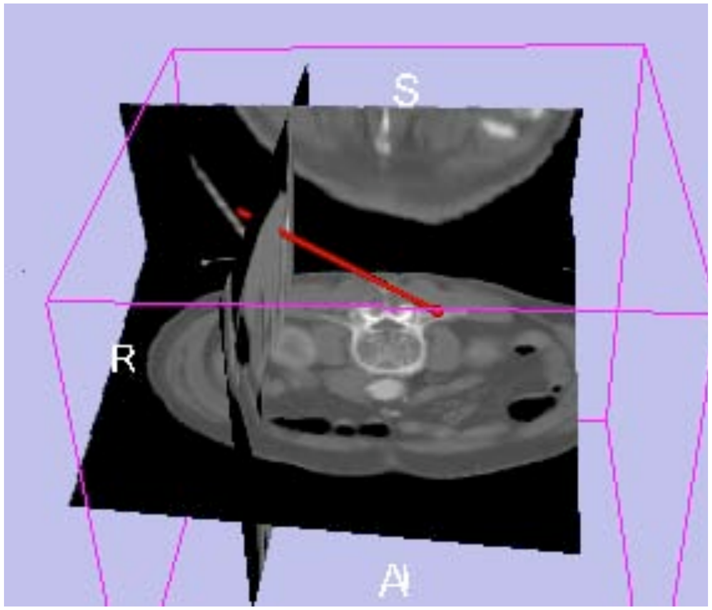
**Robarts Research Institute**

Chris Wedlake

# Learning objective

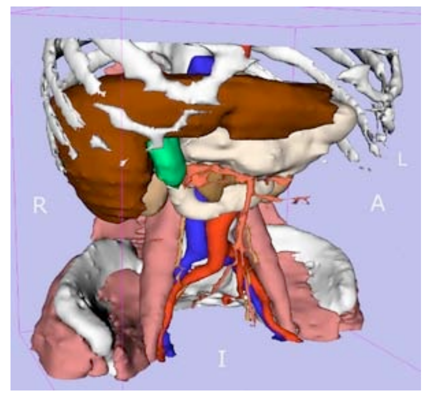
Following this tutorial, you will:

- Understand how to connect actual tracking devices with Slicer3 using the OpenIGTLink module
- Learn the details of the OpenIGTLink protocol



# Material

- This course requires *either* the SPL-PNL brain atlas or the SPL abdominal atlas:




**Brain and abdominal atlases:**
http://wiki.na-mic.org/Wiki/index.php/IGT:ToolKit/Datasets

# Required software

This tutorial requires the OpenIGTLink Slicer3 module, IGSTK and the IGSTKSandbox:

- For all three of these, you have the choice of either downloading a precompiled version (binary) **OR** building it yourself from the source code

For installation instructions, see the wiki page at http://wiki.na-mic.org/Wiki/index.php/IGT:ToolKit/Navigation-with-Aurora

*Disclaimer: It is the responsibility of the user of 3D Slicer to comply with both the terms of the license and with the applicable laws, regulations and rules.*

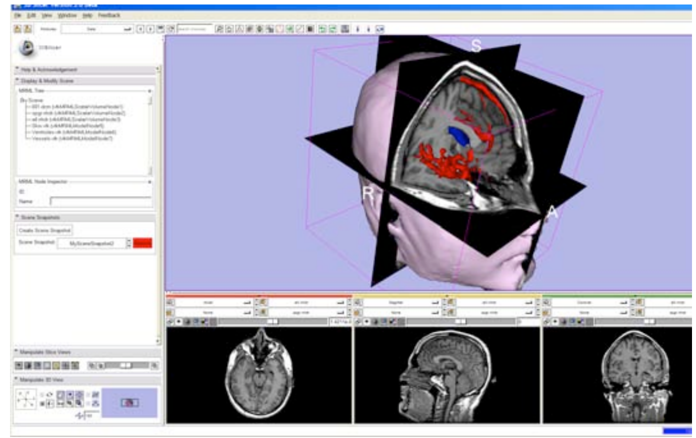
# Required hardware

- This tutorial requires an NDI Aurora tracking device and a tracked tool
- If your computer does not have a serial port, you will also need a serial-to-USB converter

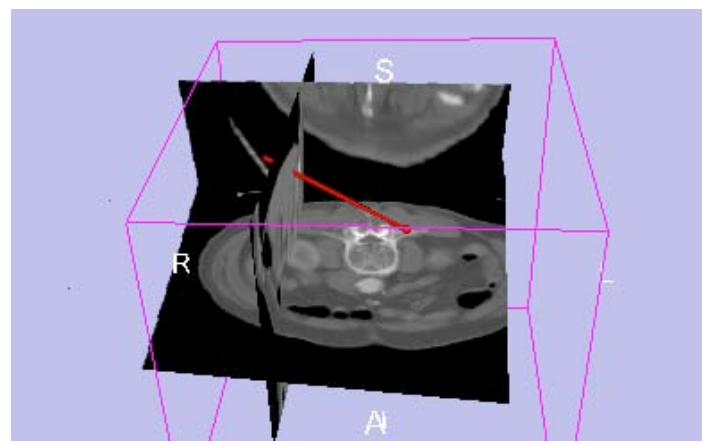
Courtesy www.ndigital.com

# Prerequisites

- Data Loading and Visualization in Slicer3:
  http://wiki.na-mic.org/Wiki/index.php/Slicer:Workshops:Slicer3_Training



- Basic Navigation Tutorial:
  http://wiki.na-mic.org/Wiki/index.php/IGT:ToolKit/Navigation-tutorial

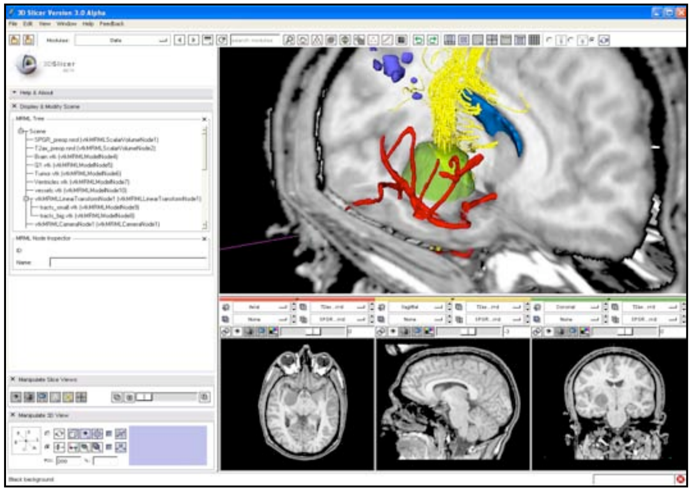


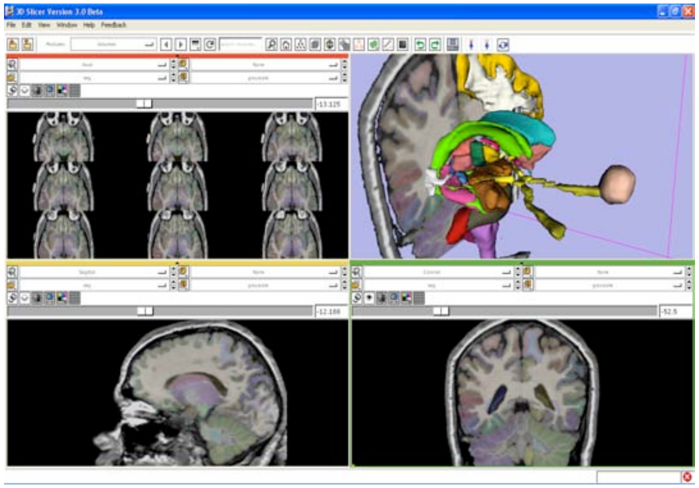
# Tutorial outline

1. **Introduction to surgical navigation**
2. Interfacing Slicer3 with external devices using OpenIGTLink
3. The OpenIGTLink protocol
4. Hands-on navigation using the NDI Aurora tracking device
5. Examples of OpenIGTLink in use

# 3D Slicer

- Integrates algorithms and utilities for medical image computing research and Image Guided Therapy into a single framework
- Is both an end-user application and a platform for research
- The precompiled program and the source code are both freely downloadable





Courtesy R. Kikinis

# Image Guided Therapy (IGT) in Slicer3

Slicer3 has extensive support for IGT, including:

- Visualization
- Registration
- Segmentation
- Model making
- Diffusion Tensor Imaging
- Quantification
- Filtering
- Interfacing to imaging devices, trackers and medical robots

} **Focus of this tutorial**

# Navigation in IGT

- Determining the positions and orientations of surgical tools using a tracking system
- Displaying virtual representations of those tools on the screen for the surgeon

# Navigation in IGT

- **Selected clinical uses:**
  - Real-time update of tool position and orientation in augmented reality environments (ex. for minimally-invasive cardiac surgery)
  - Image-to-patient registration using tracked pointer tools (ex. for total hip replacement surgery)
  - Image-to-patient registration using tracked intraoperative imaging devices (ex. ultrasound)

In order to perform navigation, software must be able to receive position and orientation data from tracking devices!

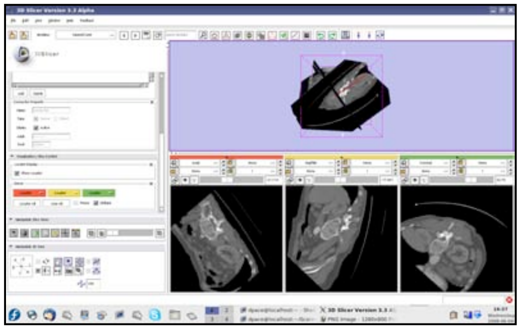
# Tutorial outline

1. Introduction to surgical navigation
2. **Interfacing Slicer3 with external devices using OpenIGTLink**
3. The OpenIGTLink protocol
4. Hands-on navigation using the NDI Aurora tracking device
5. Examples of OpenIGTLink in use

# What is OpenIGTLink?

- OpenIGTLink is a communication protocol that allows Slicer3 to communicate with external devices

# What is OpenIGTLink?

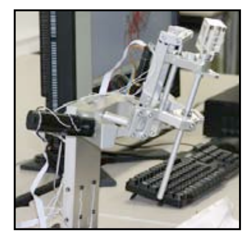


**Slicer3 OpenIGTLink module**

**OpenIGTLink** ⟷ **Imaging devices** (ex MRI, US)

**OpenIGTLink** ⟷ **Tracking devices**

**OpenIGTLink** ⟷ **Medical robots**

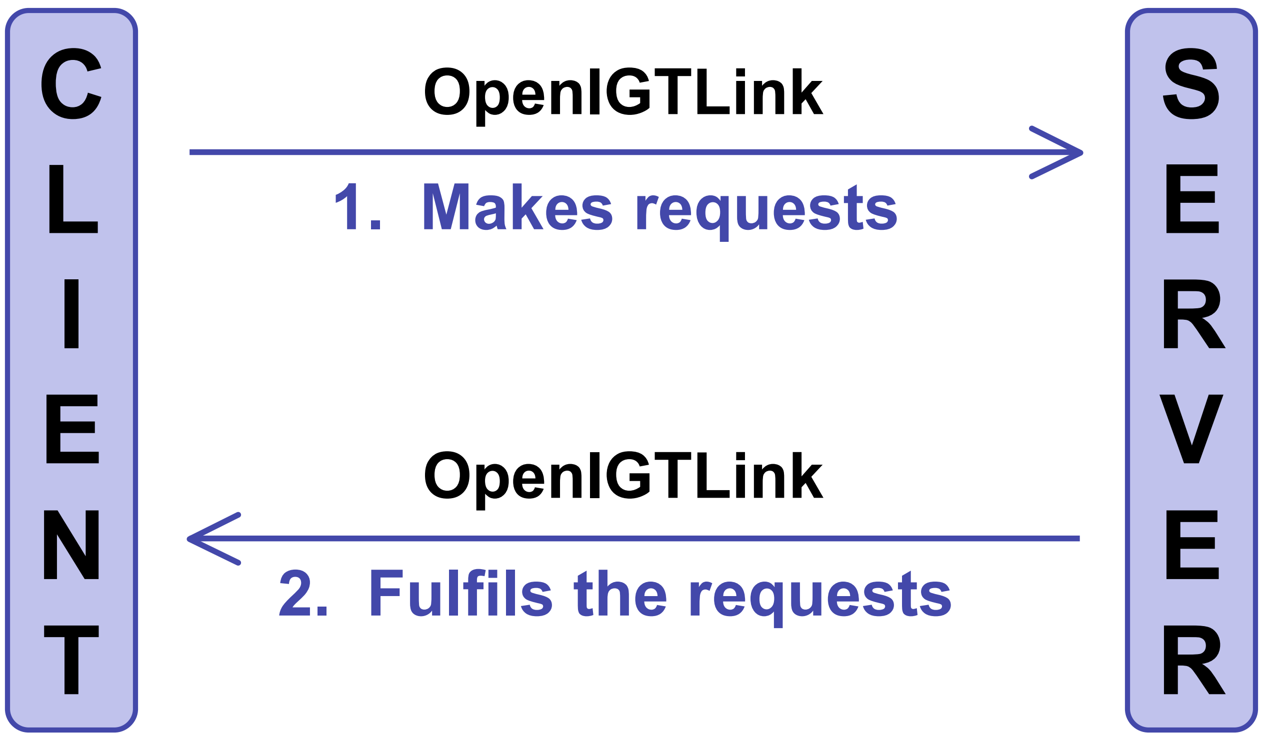




Courtesy www.ncigt.org, www.ndigital.com, www.slicer.org

# OpenIGTLink

- OpenIGTLink uses a “Client-Server” architecture.



# OpenIGTLink

- Surgical robot example:

CLIENT

OpenIGTLink →

**1. Makes requests**

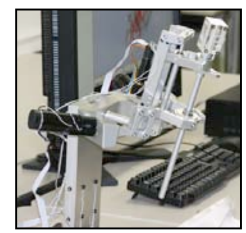
Ex. Get current coordinate

OpenIGTLink ←

**2. Fulfils the requests**

Ex. Returns current coordinate

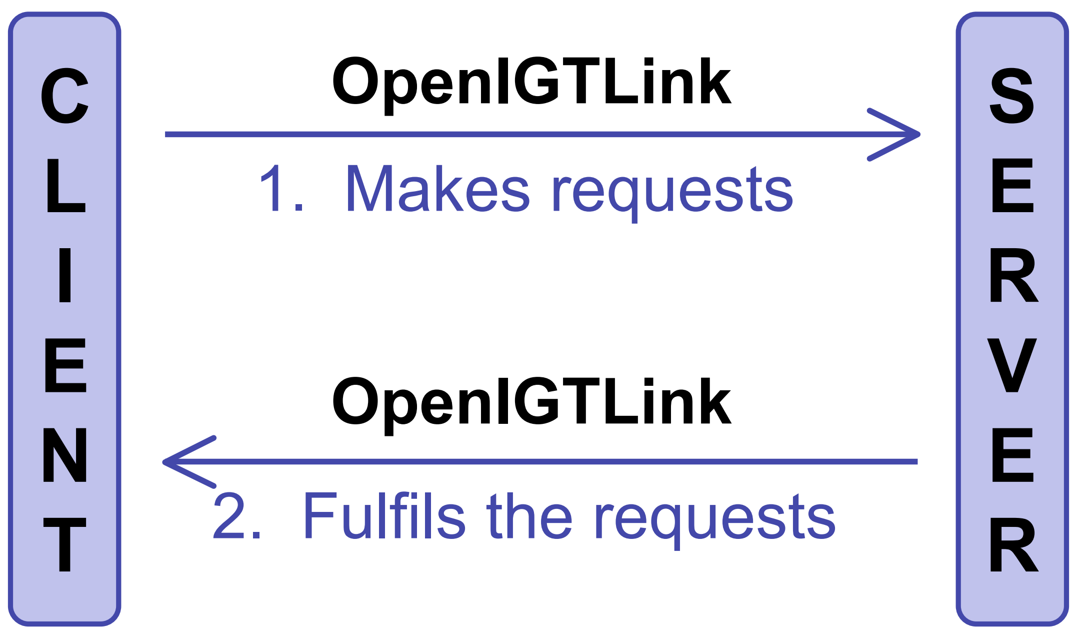
SERVER



Courtesy www.slicer.org

# OpenIGTLink

- The OpenIGTLink protocol specifies the structure of the messages sent between the client and the server
- Slicer3 can be either the client or the server, depending on the application

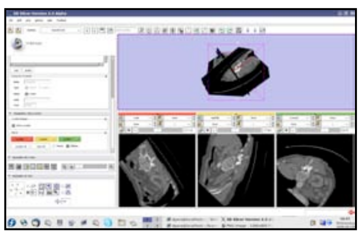


# The OpenIGTLink module in Slicer3

- OpenIGTLink is a protocol
- There is an OpenIGTLink module in Slicer3 that implements the protocol so that Slicer3 can communicate with external devices

# OpenIGTLink and IGSTK

- IGSTK = Image-Guided Surgery Tool Kit
- OpenIGTLink functionality has been added to IGSTK: you can now use IGSTK to write programs that interact with both Slicer3 and the physical device



**Slicer3**
(client or server)

**OpenIGTLink**

Image-Guided Surgery Toolkit

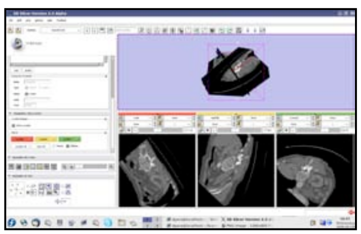
**IGSTK program**
(client or server)


Courtesy www.ndigital.com

**Device ex. tracker**

# igstkAuroraTrackerToolObserverToOpenIGTLinkRelayTest

- This tutorial uses an NDI Aurora tracking device to demonstrate Slicer3's navigation capabilities
- The `igstkAuroraTrackerToolObserverToOpenIGTLinkRelayTest` IGSTK test acts as the **client** to send the tracker data to Slicer3 over OpenIGTLink



**Slicer3**
(server)

**OpenIGTLink**

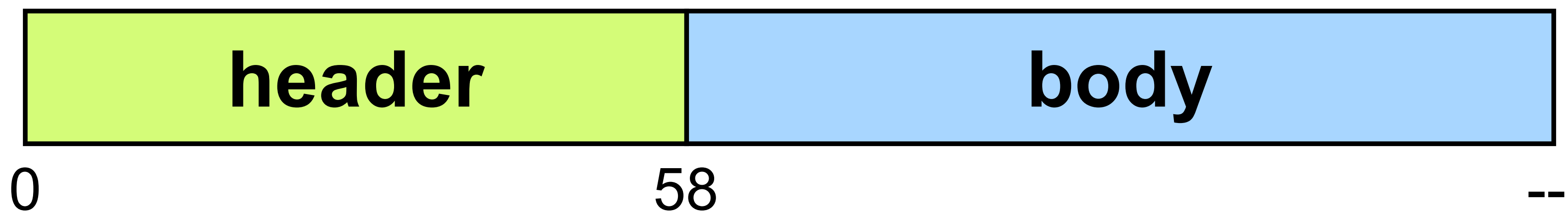
**igstkAurora TrackerTool ObserverTo OpenIGTLink RelayTest**
(client)

**NDI Aurora tracker**

# Tutorial outline

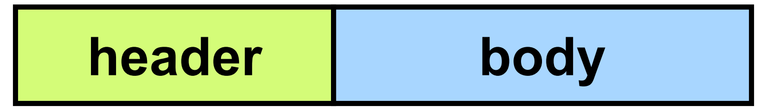
1. Introduction to surgical navigation
2. Interfacing Slicer3 with external devices using OpenIGTLink
3. **The OpenIGTLink protocol**
4. Hands-on navigation using the NDI Aurora tracking device
5. Examples of OpenIGTLink in use

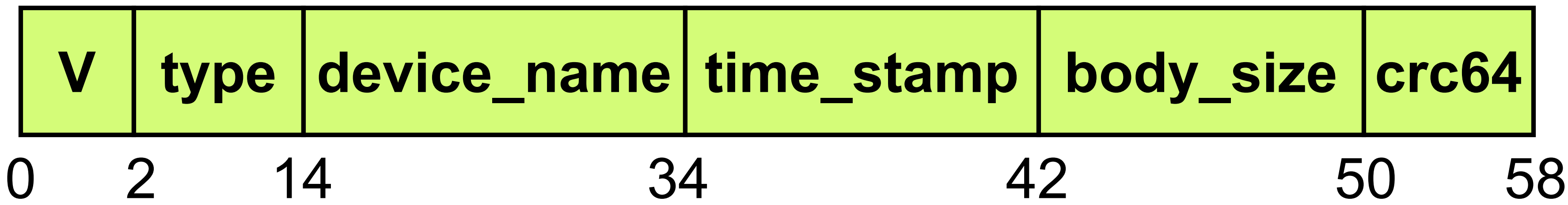
# The OpenIGTLink protocol




- The **header** gives information about the structure of the body
  - The header is always the same length
  - The header is fixed to big endian
- The **body** contains commands, queries or data
  - The body is of variable length
  - The body may be big endian or little endian

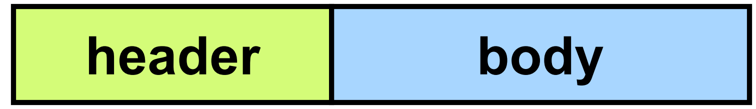
# Header structure



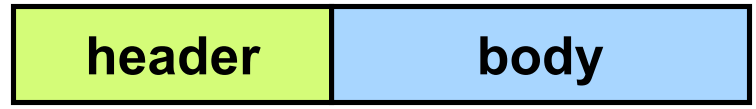

- **V** = version number of the OpenIGTLink protocol
- **type** = type of message, ex. IMAGE or GET_POSITION
- **device_name** = name of the data source (ex. each port on a 4-port NDI Aurora would have a unique name)
- **time_stamp** = timestamp for the message, or 0 if unused
- **body_size** = size of the message's body, in bytes
- **crc64** = checksum

# Body structure




- Recall that the **type** component of the message's header specifies the type of the message
- Messages can be either data, queries or commands:
  - **Data** (ex. IMAGE) can be sent from either the client to the server or from the server to the client
  - **Queries and commands** (ex. GET_STATUS and MOVE_TO) are sent from the client to the server and can optionally include parameters

# Body structure




- The rest of the slides in this section describe the body structure for some example data and command OpenIGTLink messages

- For a more detailed description of the OpenIGTLink protocol, see: http://wiki.na-mic.org/Wiki/index.php/OpenIGTLink/Protocol

# IMAGE data

| header | body |
|---|---|

| image data header | image data |
|---|---|

58 130 --

- The body of IMAGE data contains:
  - **Image data header:** describes the image data
  - **Image data:** intensity values for the image

# IMAGE - image data header

| Data | Type | Description |
|---|---|---|
| V | Unsigned short | Version number |
| T | 8 bit unsigned int | Image type (scalar or vector) |
| S | 8 bit unsigned int | Scalar type (ex uint8, float64…) |
| E | 8 bit unsigned int | Endianness for image data |
| O | 8 bit unsigned int | Image coordinate (RAS or LPS) |
| RI, RJ, RK | 16 bit unsigned int | # pixels in each direction |
| PX, PY, PZ | 32 bit float | Image center position |
| TX, TY, TZ | 32 bit float | Transverse vector for ‘i’ index |
| SX, SY, SZ | 32 bit float | Transverse vector for ‘j’ index |
| NX, NY, NZ | 32 bit float | Normal vector (direction of ‘k’ index) |
| DI, DJ, DK | 16 bit unsigned int | Starting index of subvolume |
| DRI, DRJ, DRK | 16 bit unsigned int | Number of pixels in subvolume |

# IMAGE - image data

| Data | Type | Description |
| --- | --- | --- |
| IMAGE_DATA | Binary image data | Intensity values for the image data |

# TRANSFORM data

header | body

| $a_{00}$ | $a_{10}$ | $a_{20}$ | $a_{01}$ | $a_{11}$ | $a_{21}$ | $a_{02}$ | $a_{12}$ | $a_{22}$ | $a_{03}$ | $a_{13}$ | $a_{23}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 58 | 62 | 66 | 70 | 74 | 78 | 82 | 86 | 90 | 94 | 98 | 102 | 106 |

- TRANSFORM data is a list of 4-byte floats specifying the top three rows of a 4x4 transformation matrix

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# POSITION data

| header | body |
|---|---|

| x | y | z | ox | oy | oz | W |
|---|---|---|---|---|---|---|

58 62 66 70 74 78 82 86

- (x, y, z) = position (three 4-byte floats)
- (ox, oy, oz, W) = normalized orientation quaternion (four 4-byte floats)

# STATUS data

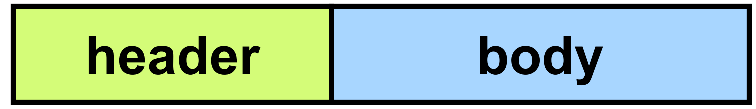
| header | body |
| --- | --- |

| C | Sub code | Status name | Status message |
| --- | --- | --- | --- |

58 60 68 88 --

- **C** = Status code, ex. 1 for “OK” and 7 = “time out / connection lost”
- **Sub code** = sub code for the error, ex. 0x200 is “file not found”
- **Error name** = character string, ex. “starting up”
- **Status message** (optional) = optional English description

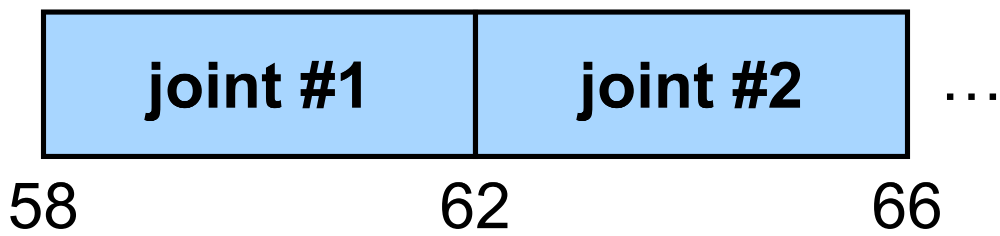
# GET_STATUS query




- The GET_STATUS command has no parameters, so the length of the body is zero
- The server will return a STATUS data packet

# GET_VELOCITY query

header | body



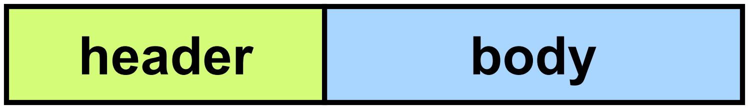

- **Parameters:** one or more 32-bit integers representing the specific joints
- The robot will return a joint velocity for each specified joint

**This is a JHU-BRP robot example**

- Slicer3 = client
- Robot = server

For more information:
http://www.na-mic.org/Wiki/index.php/OpenIGTLink/Protocol/JHUBRP

# STOP command




- The STOP command has no parameters, so the length of the body is zero
- The robot will stop moving

**This is a JHU-BRP robot example**

- Slicer3 = client
- Robot = server

For more information: http://www.na-mic.org/Wiki/index.php/OpenIGTLink/Protocol/JHUBRP

# MOVE_TO command

| header | body |
|---|---|

| x | y | z | ox | oy | oz | W |
|---|---|---|---|---|---|---|

58 62 66 70 74 78 82 86

- **Parameters:** three 4-byte floats indicating the position (**x**, **y**, **z**) and four 4-byte floats indicating the normalized orientation quaternion (**ox**, **oy**, **oz** , **W**)
- The robot will move to this position and orientation, and will return its status

**This is a JHU-BRP robot example**

- Slicer3 = client
- Robot = server

For more information: http://www.na-mic.org/Wiki/index.php/OpenIGTLink/Protocol/JHUBRP

# Tutorial outline

1. Introduction to surgical navigation
2. Interfacing Slicer3 with external devices using OpenIGTLink
3. The OpenIGTLink protocol
4. **Hands-on navigation using the NDI Aurora tracking device**
5. Examples of OpenIGTLink in use

# Hands-on navigation

- Using an NDI Aurora tracking device, you will learn how to:
  - Set up an OpenIGTLink connection between an actual tracking device and Slicer3
  - Show the resulting transforms using the Slicer3 “locator”
  - Reslice image volumes using the tracker transform

# Note

- Although the screenshots used in this tutorial use the SPL abdominal atlas, the SPL-PNL brain atlas can also be used

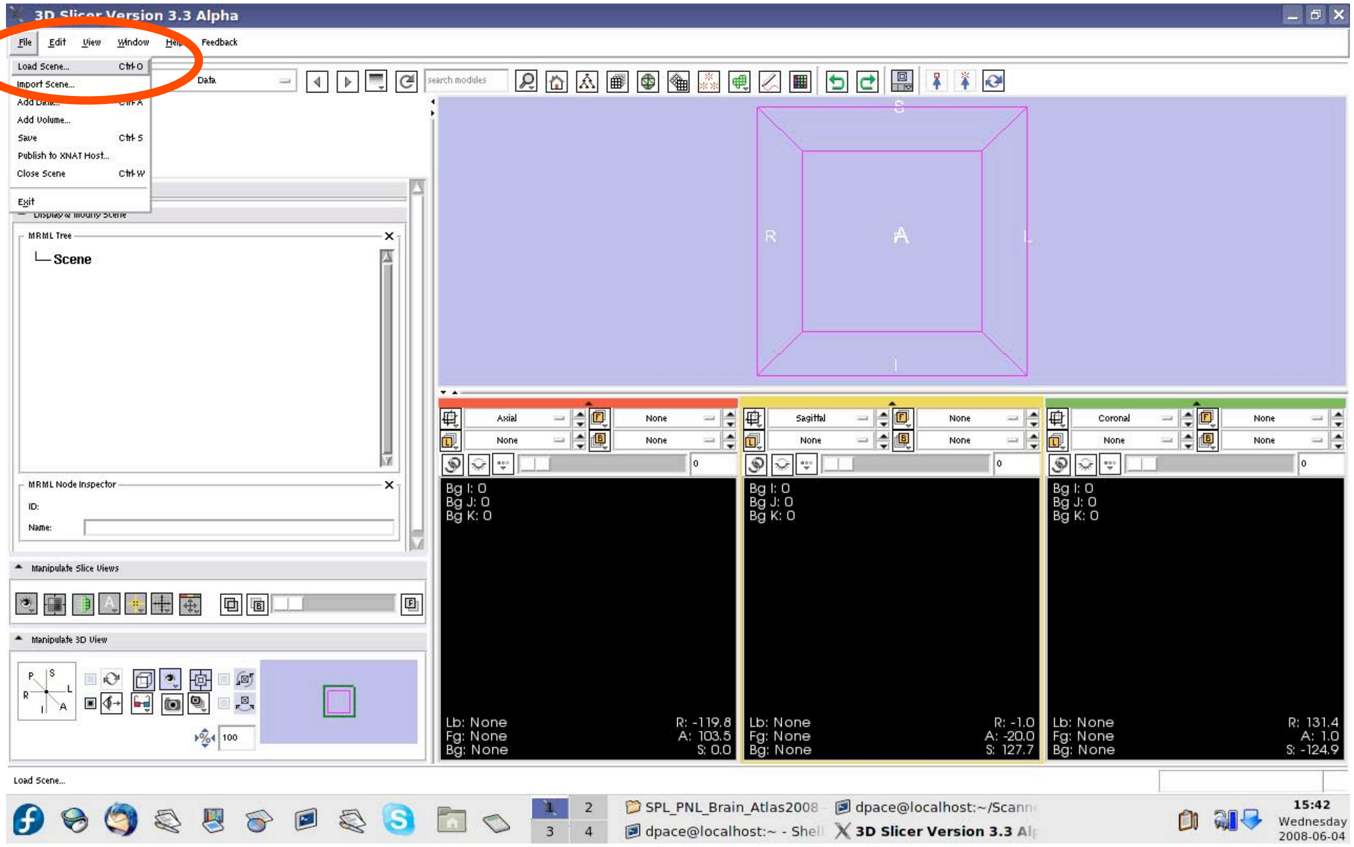
# Set up the NDI Aurora device

Connect the NDI Aurora device to your computer:

- Turn the control unit on
- Connect the field generator to the control unit
- Connect your tool to the sensor interface unit (analog-to-digital converter), and plug the sensor interface unit into **port 1** on the control unit
- Connect the control unit to **serial port 0** on your computer, or into any USB port if you are using a serial-to-USB converter

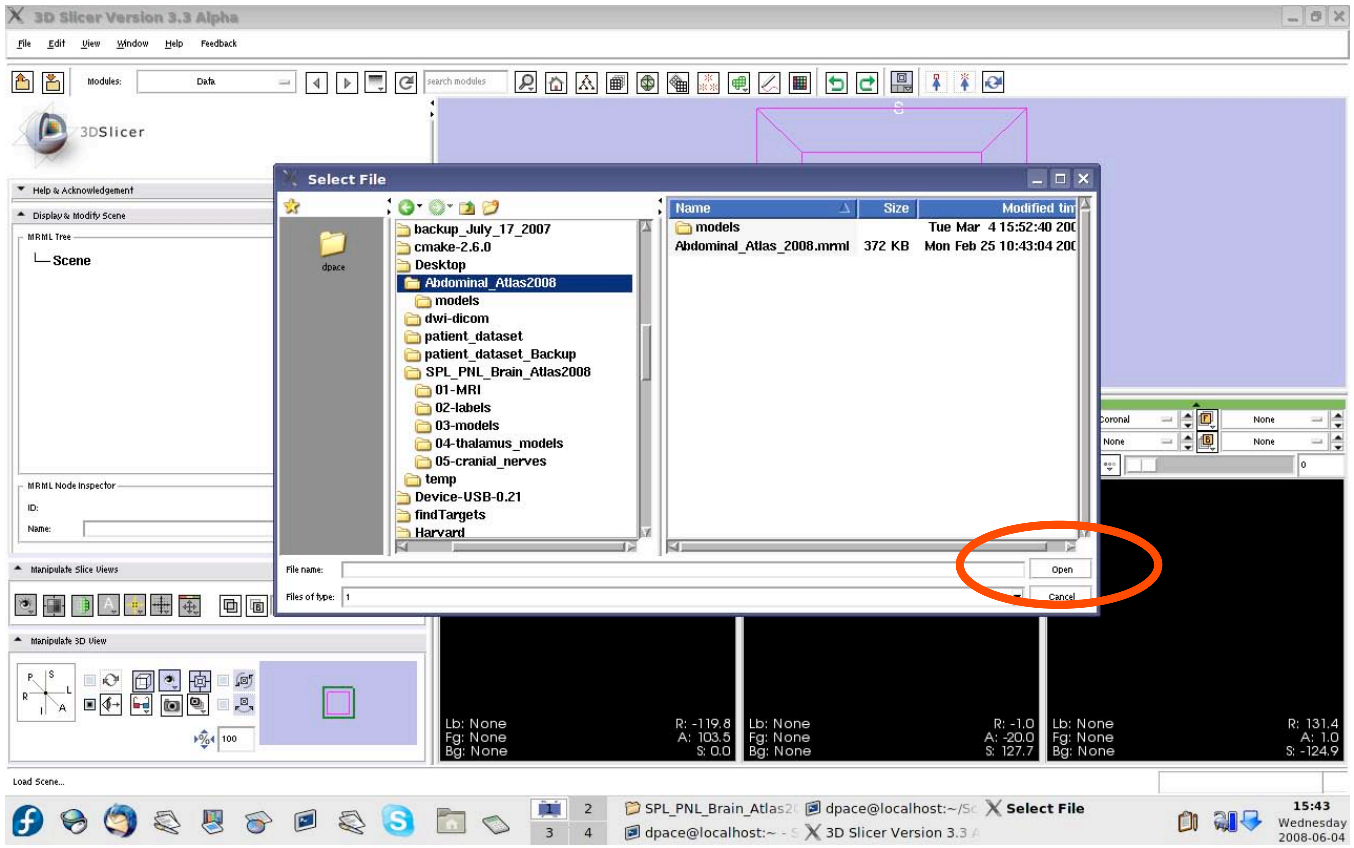
# Load the atlas

Click on File -> Load Scene



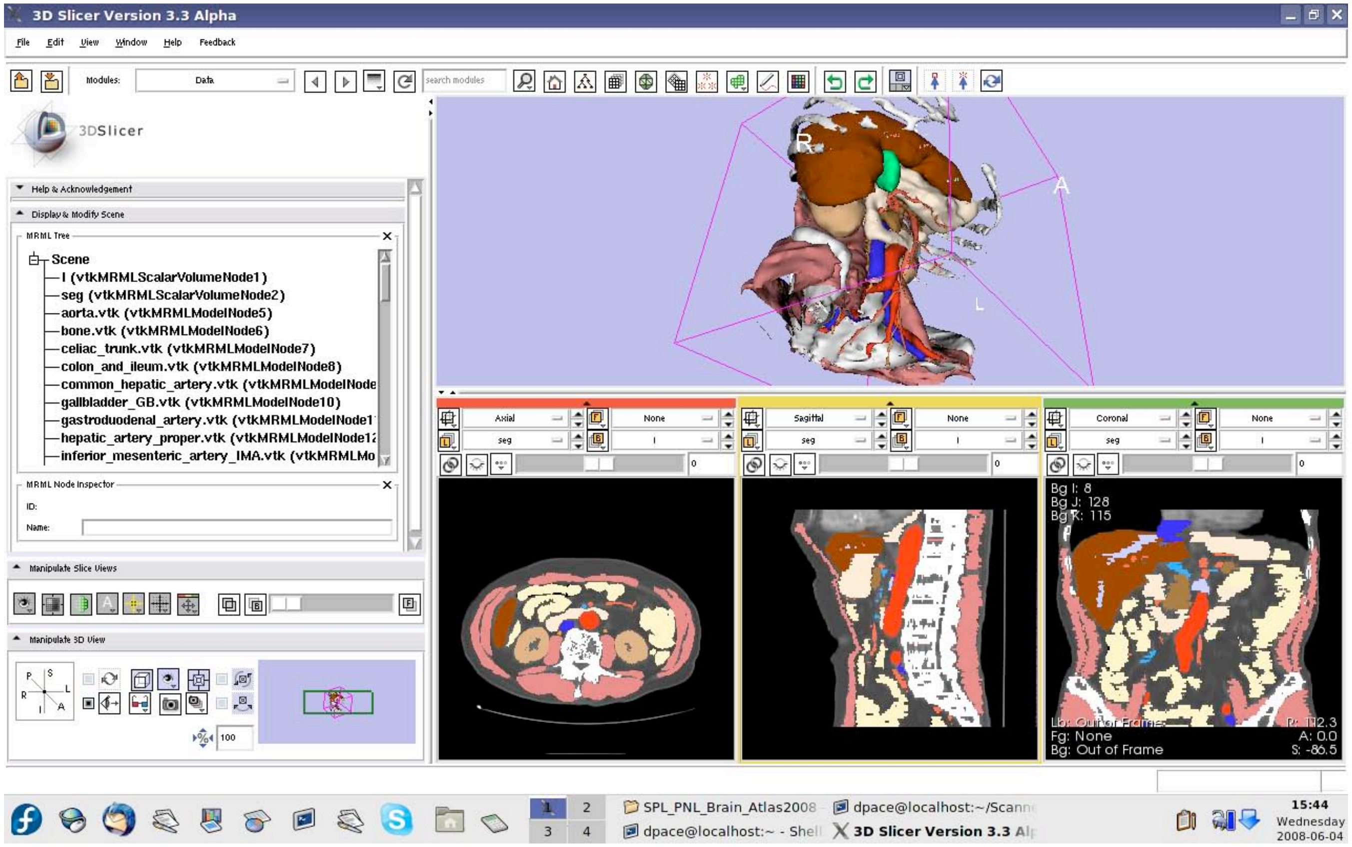
# Load the atlas

Select the scene file for the atlas (brain_atlas_ 2008.mrml or Abdominal_ Atlas_2008) and click "Open"



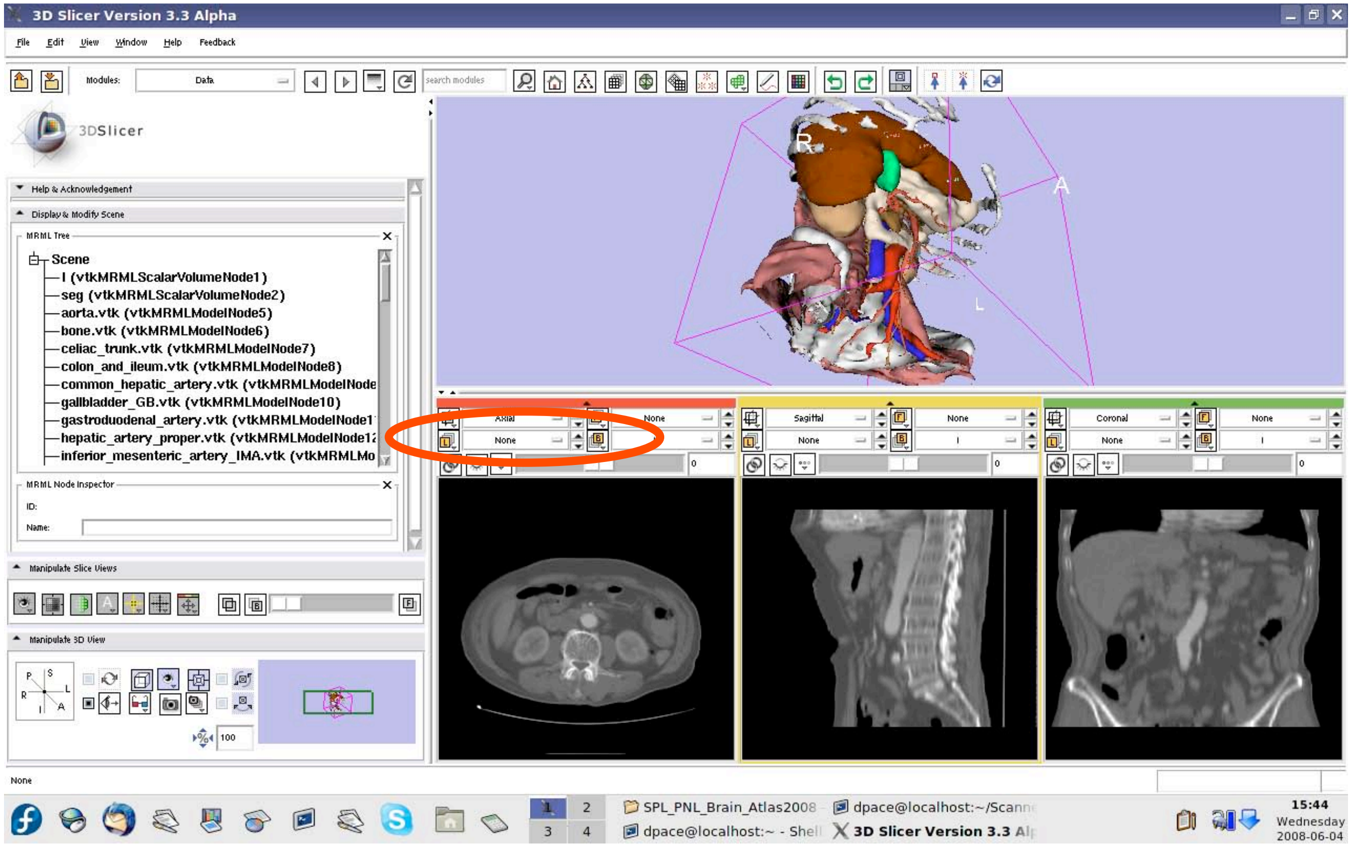
# Load the atlas

All of the atlas components are shown in the MRML scene within the Data module



# Load the atlas

If you are using the abdominal atlas, change the label map to "None"

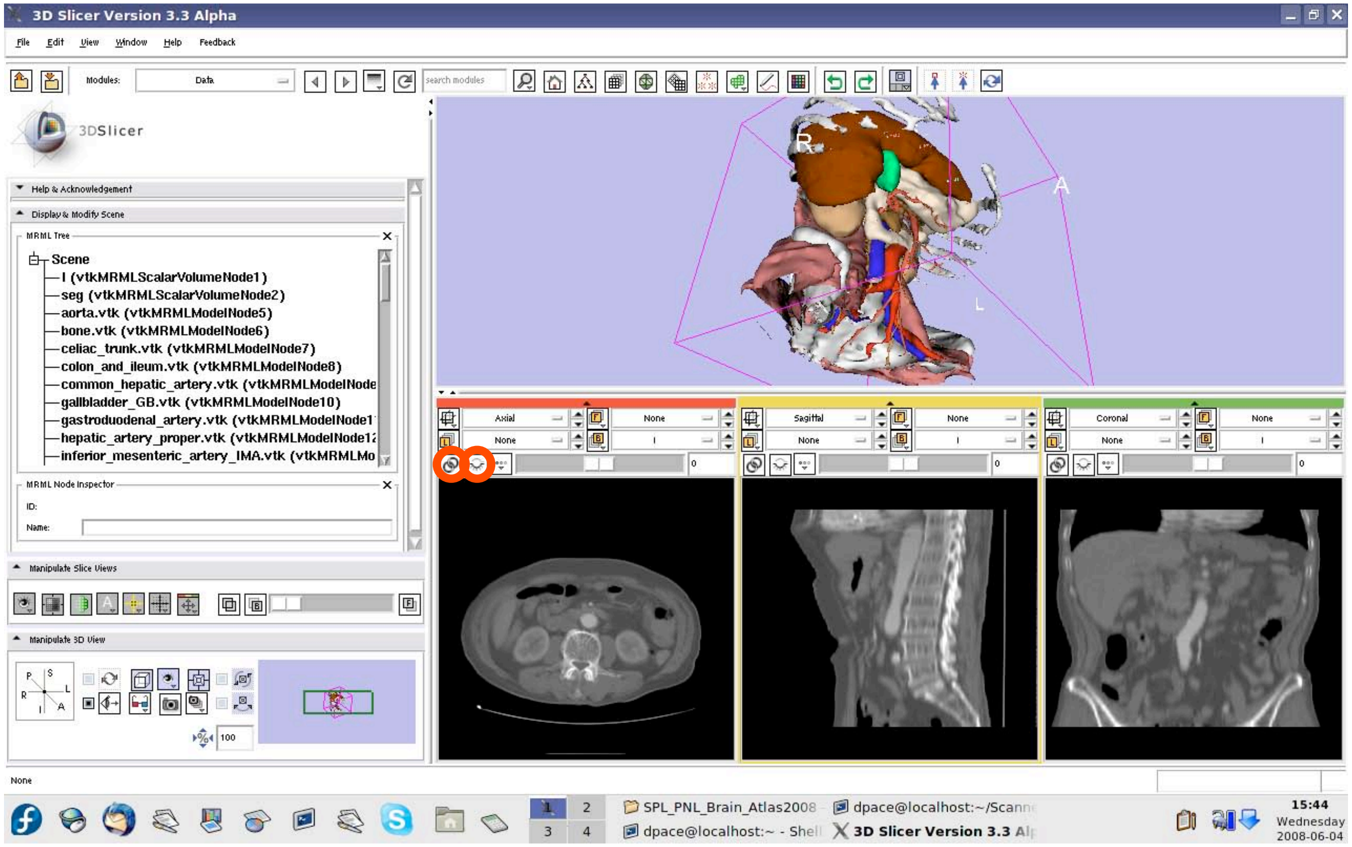


# Load the atlas

If you are using the brain atlas, turn off the visibility of the images:

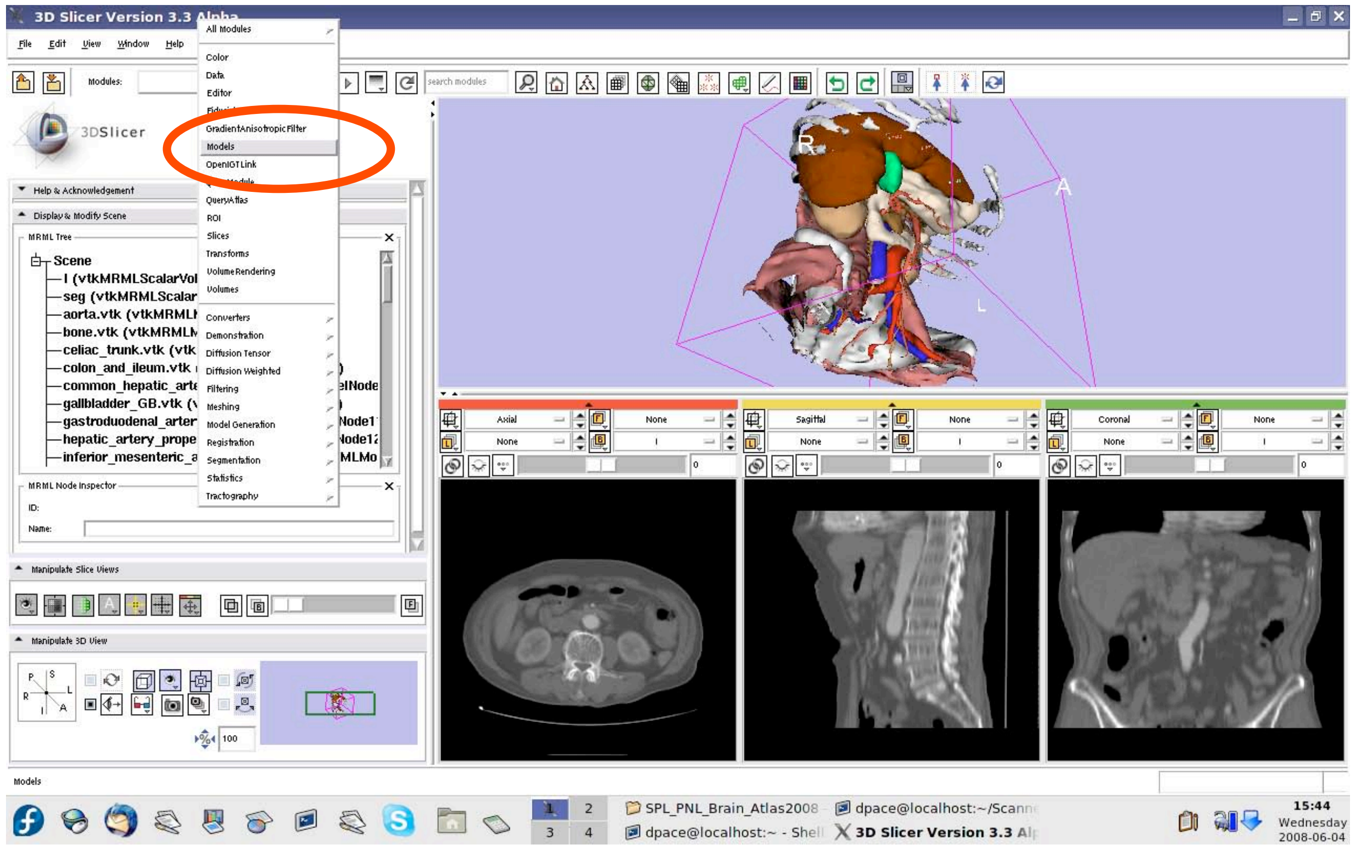
Click the “Link” button

Click the “Visibility” button



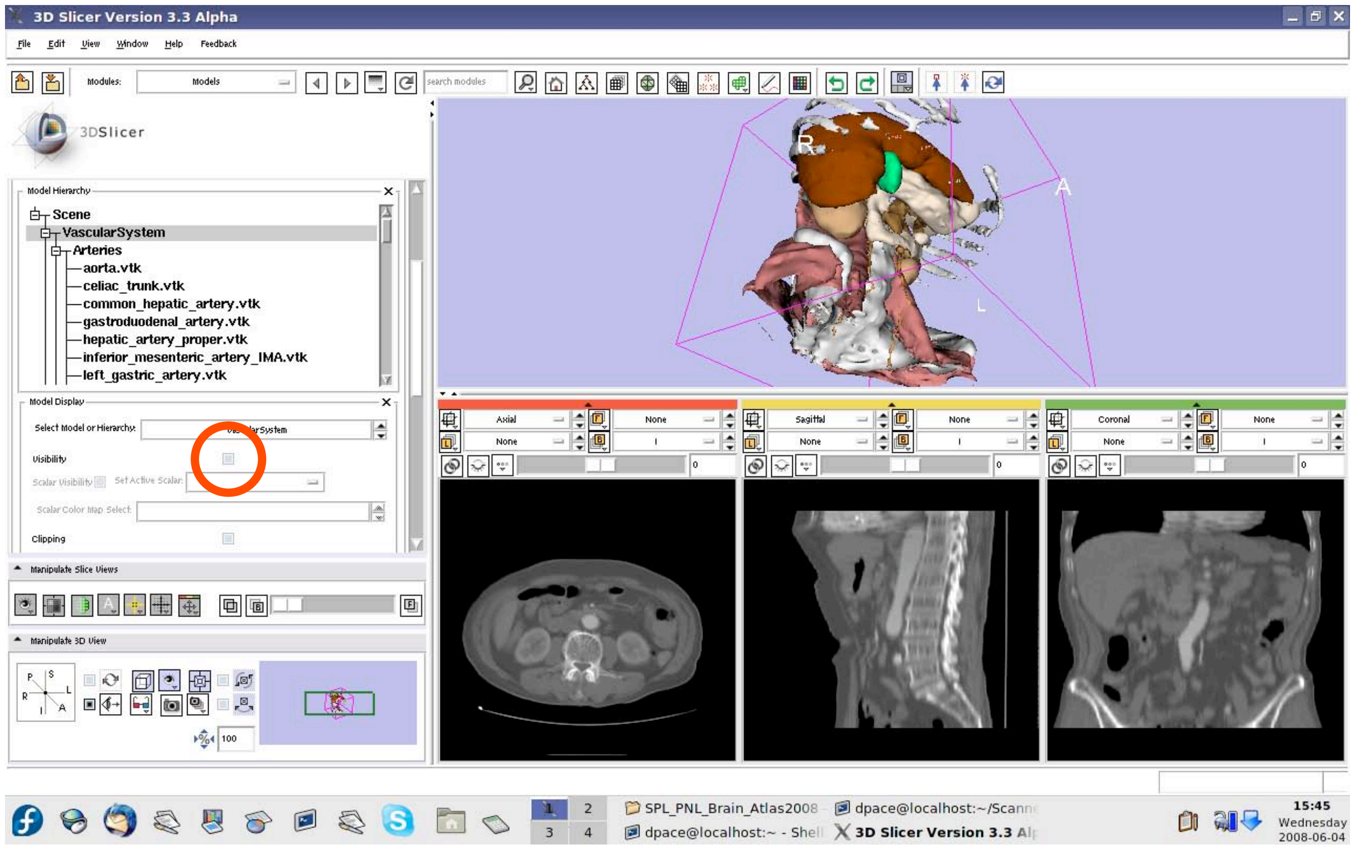
# Make the models invisible

Open the Models module



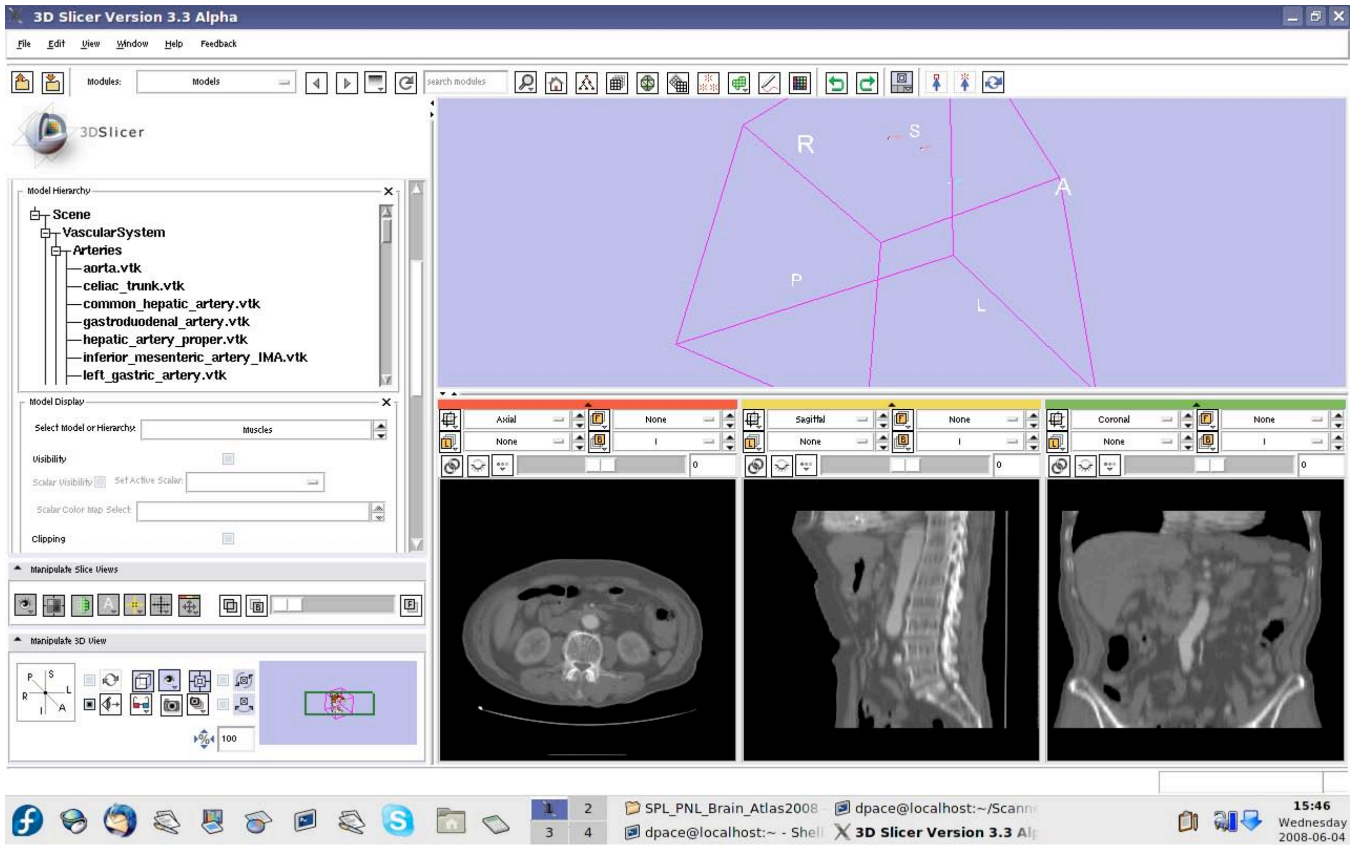
# Make the models invisible

For each of the major headings in the model hierarchy, turn the visibility off



# Make the models invisible

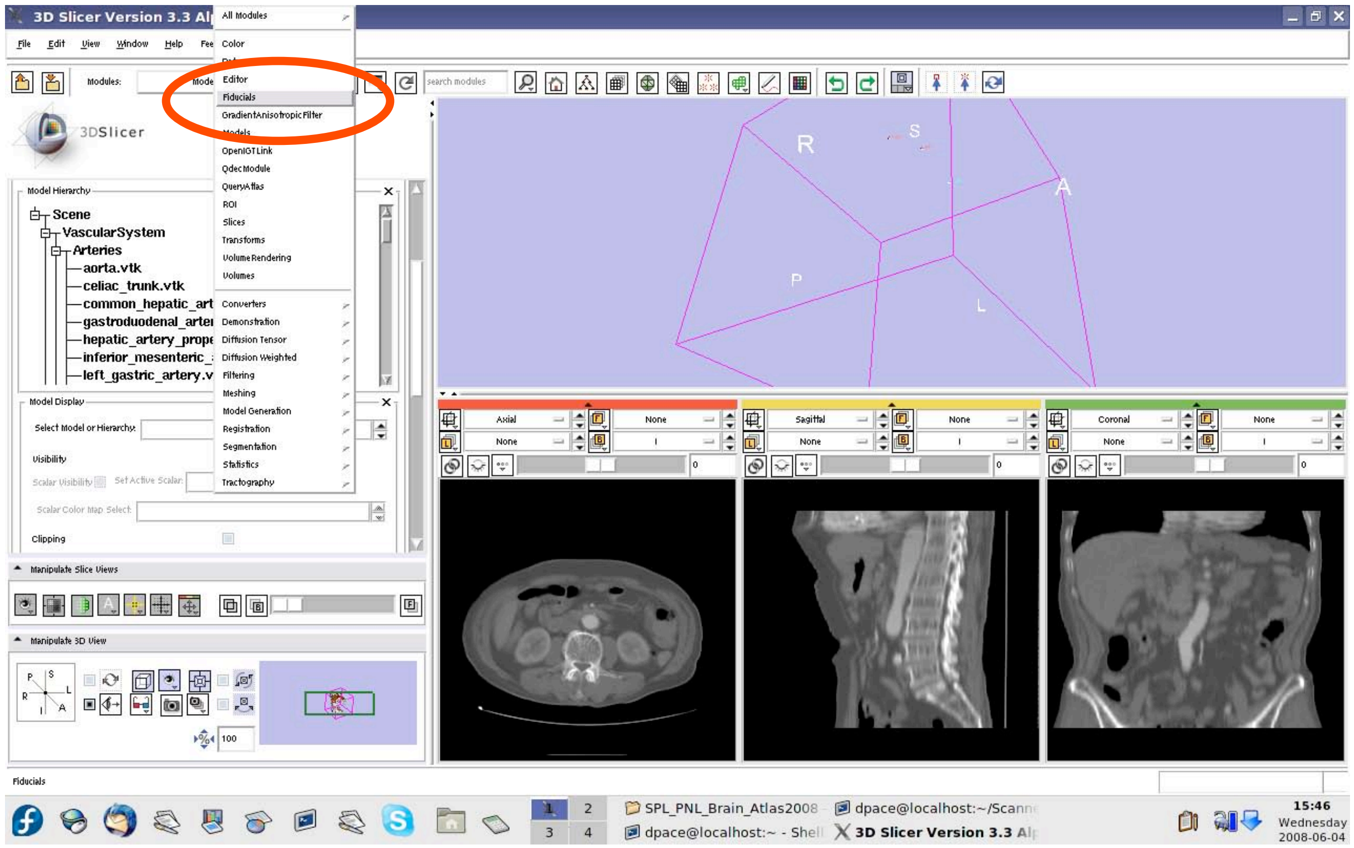
When you are finished, no models will be shown



3DSlicer

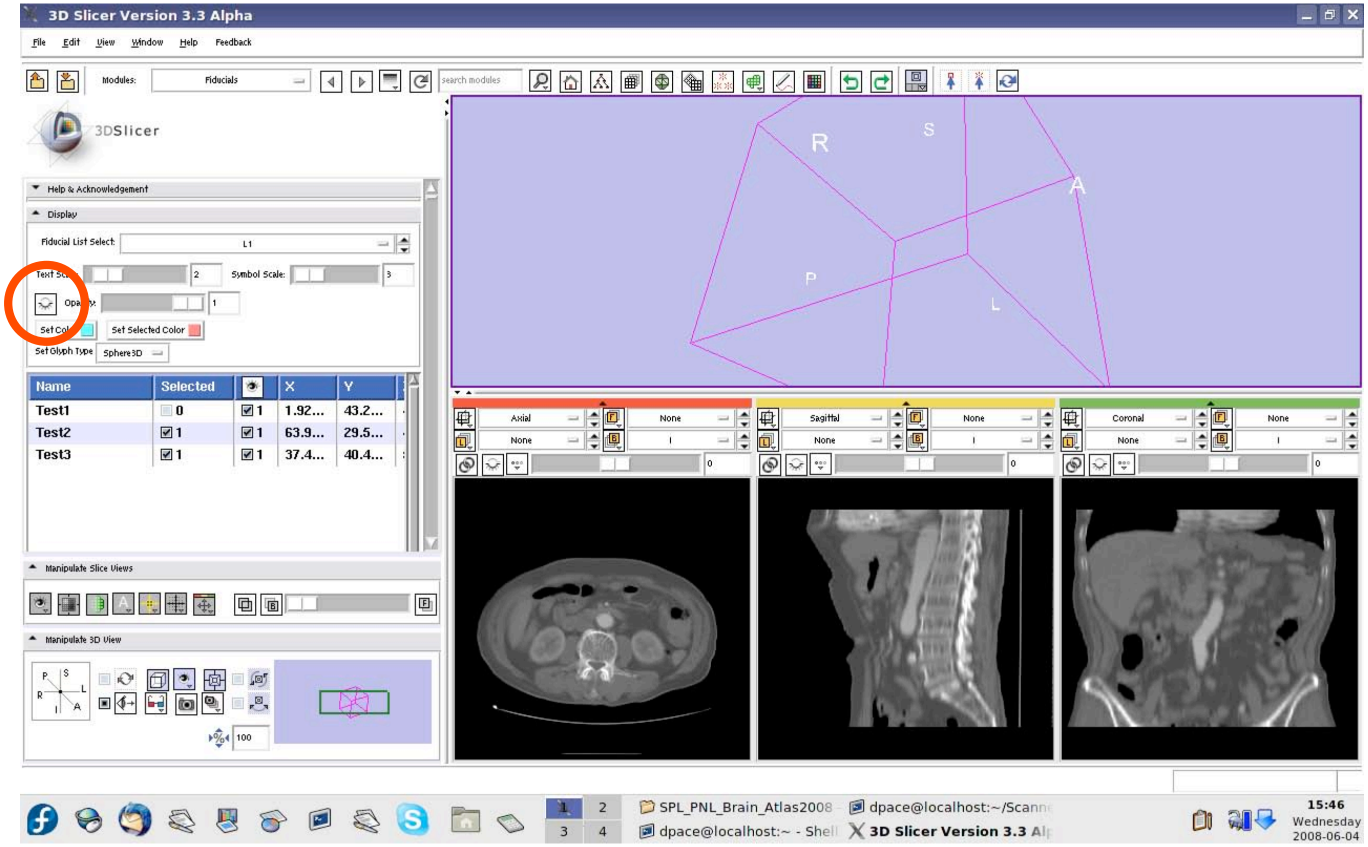
# Make the fiducials invisible

If you are using the abdominal atlas, open the Fiducials module



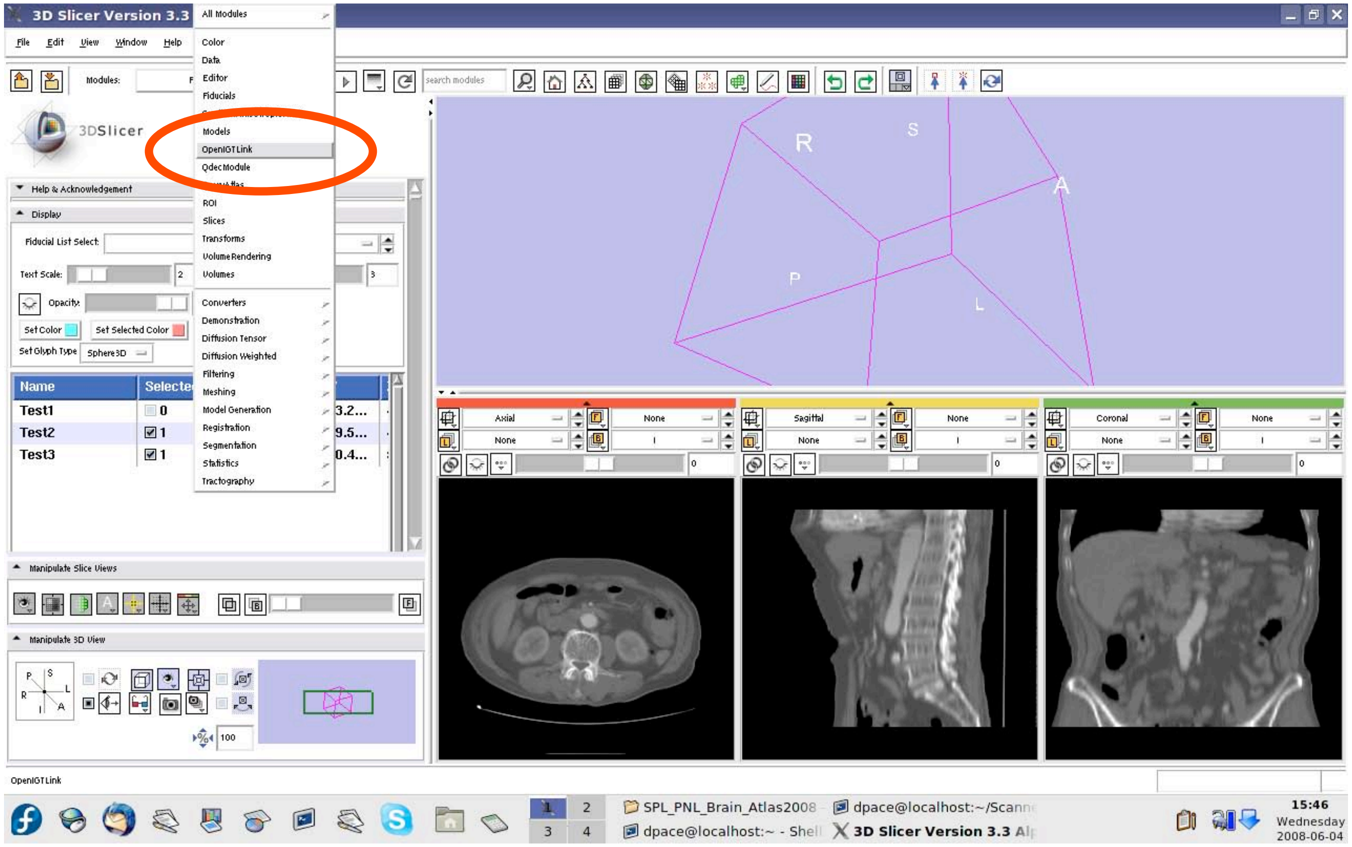
# Make the fiducials invisible

If you are using the abdominal atlas, turn off the visibility of the fiducials



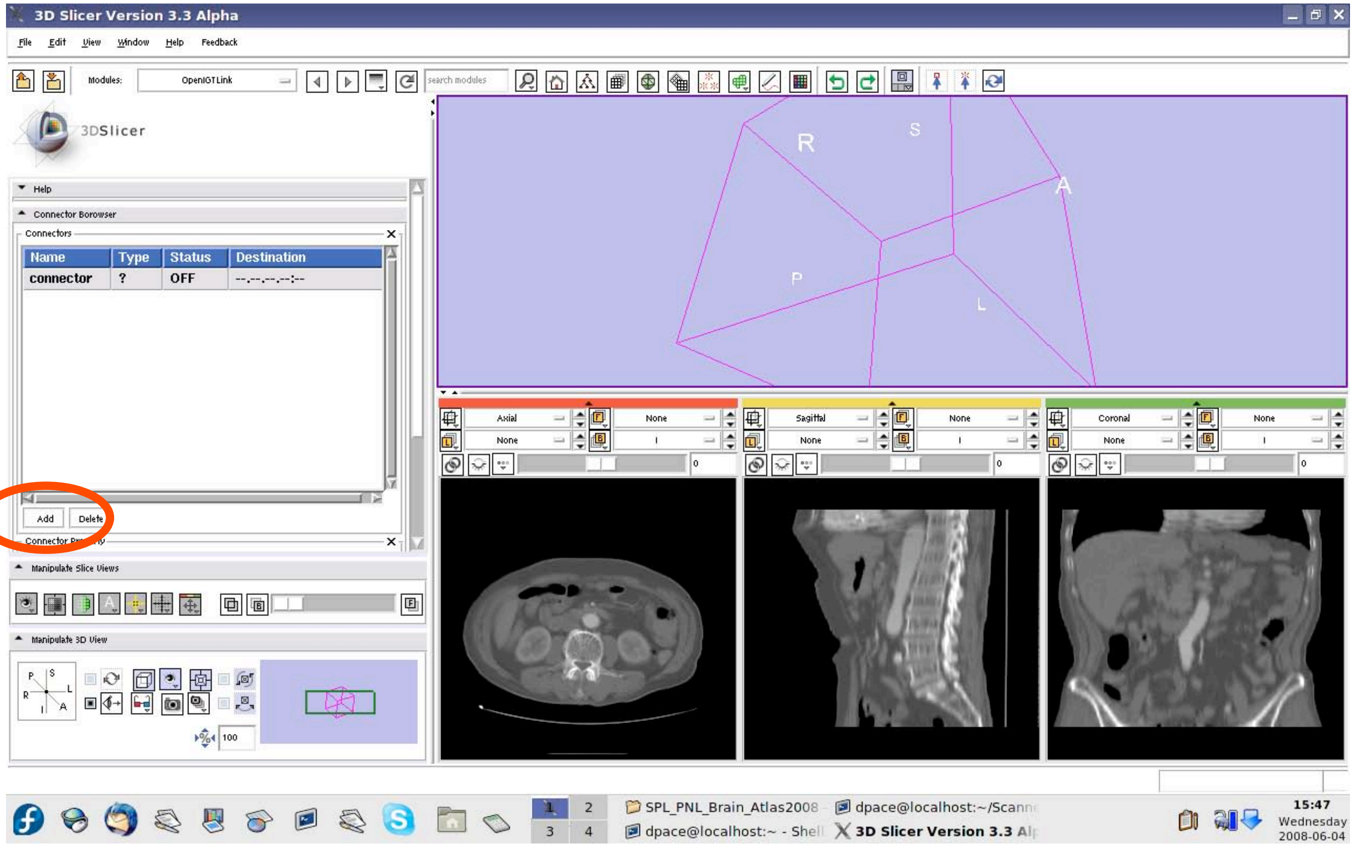
# Set up the OpenIGTLink connection

Open the OpenIGTLink module



# Set up the OpenIGTLink connection

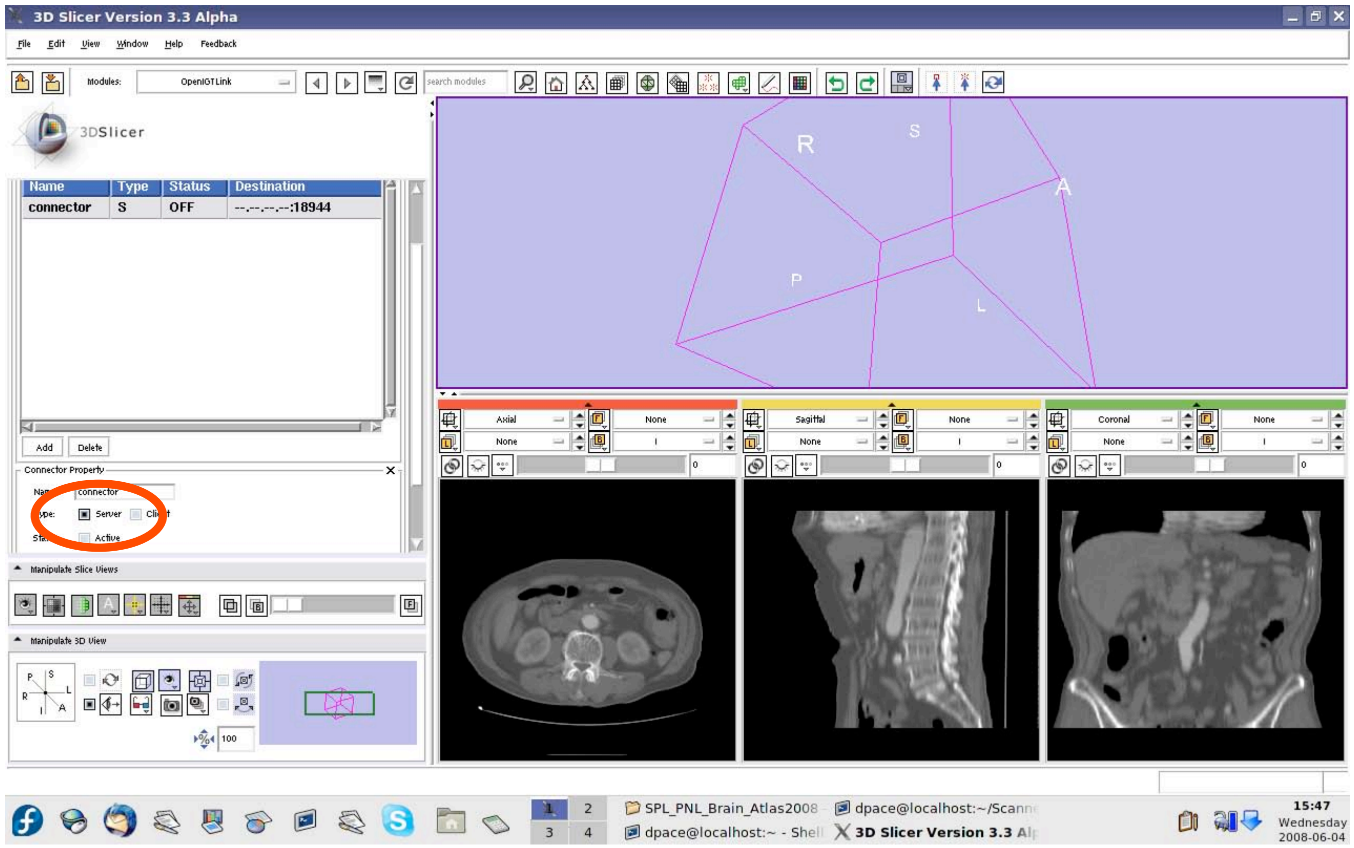
The Connectors pane shows the OpenIGTLink connections that Slicer3 is connected to




Add a new connection by clicking the "Add" button

# Set up the OpenIGTLink connection

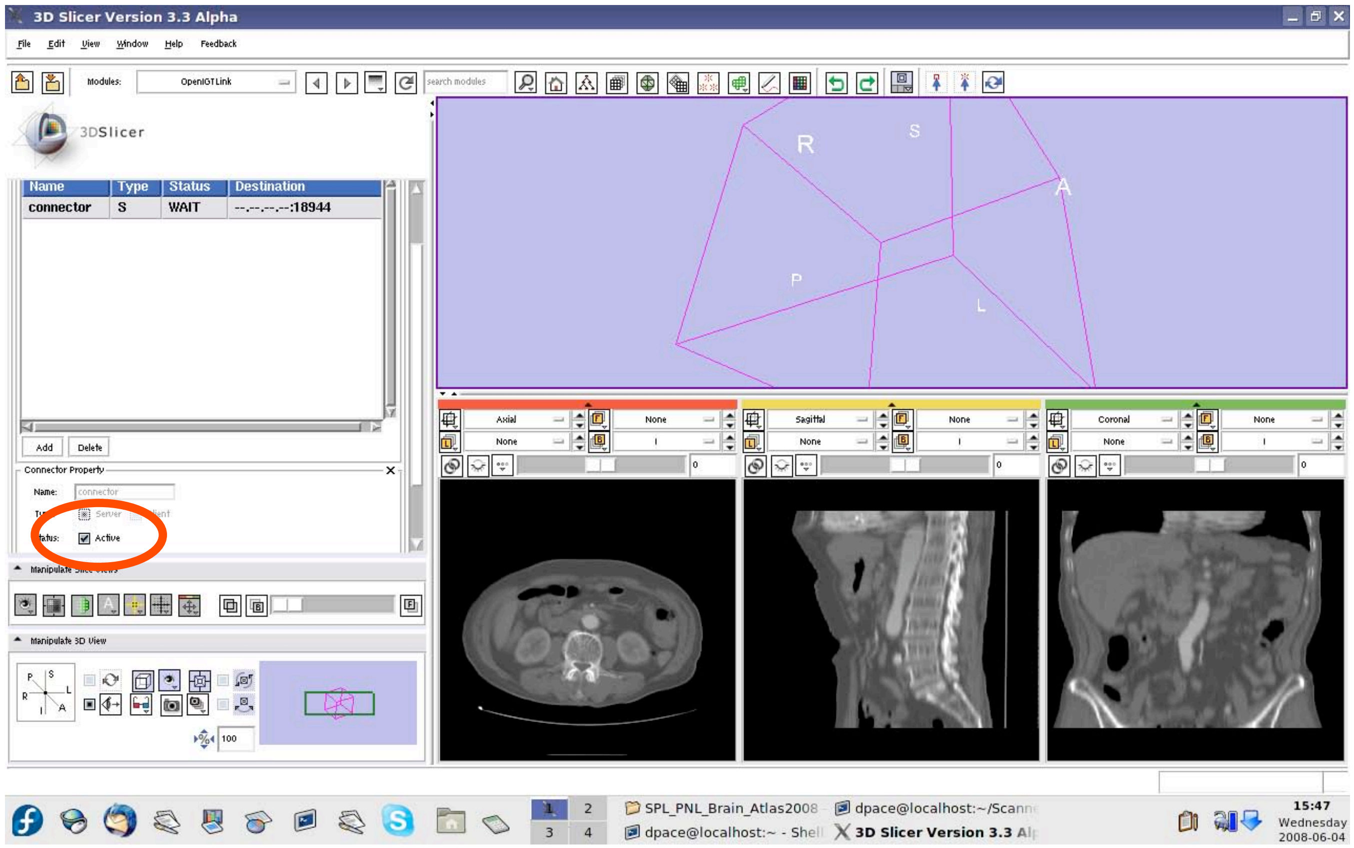
Set Slicer3 to be the server by clicking on the Server box




Note that the connector type is now set to “S” instead of “?”

# Set up the OpenIGTLink connection

Make the connection active by clicking on the "Active" button

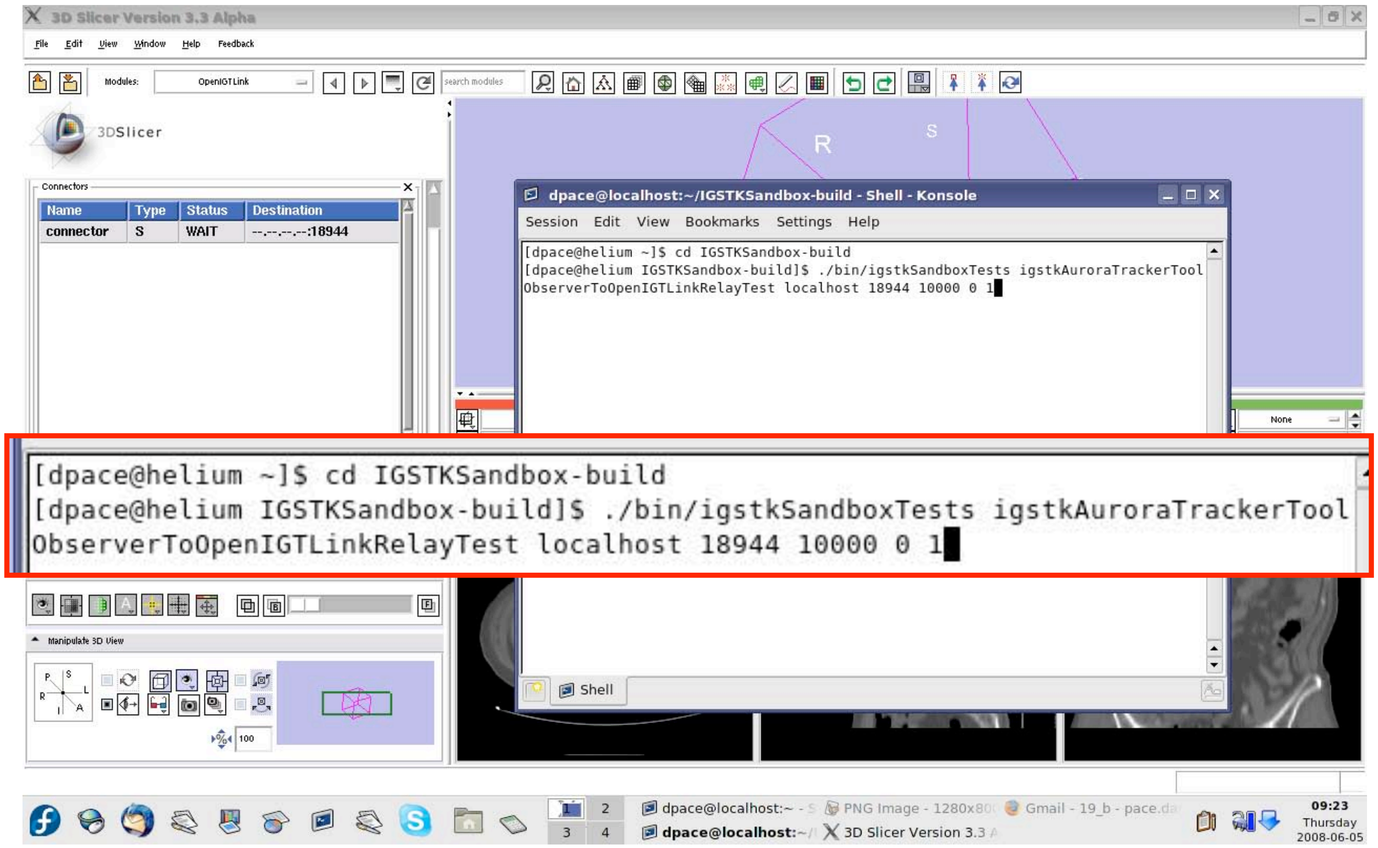



Note that the connector status is now set to "WAIT" instead of "OFF"

# Start the IGSTK test program

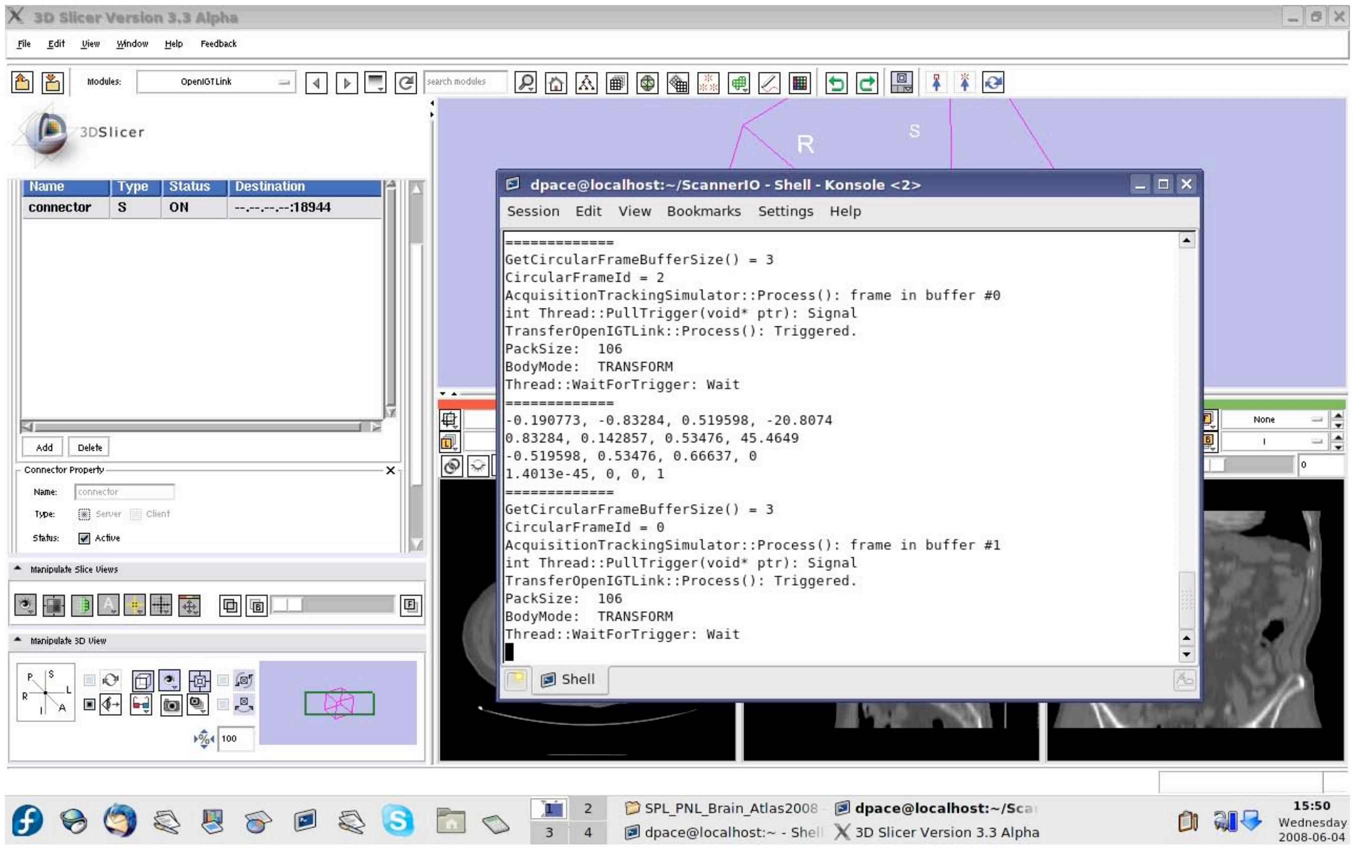
Run the IGSTK test program:

- localhost = the host name
- 18944 = the port number
- 10000 = # of transforms to send
- 0 = serial port number that the Aurora is connected to
- 1 = # of frames per second



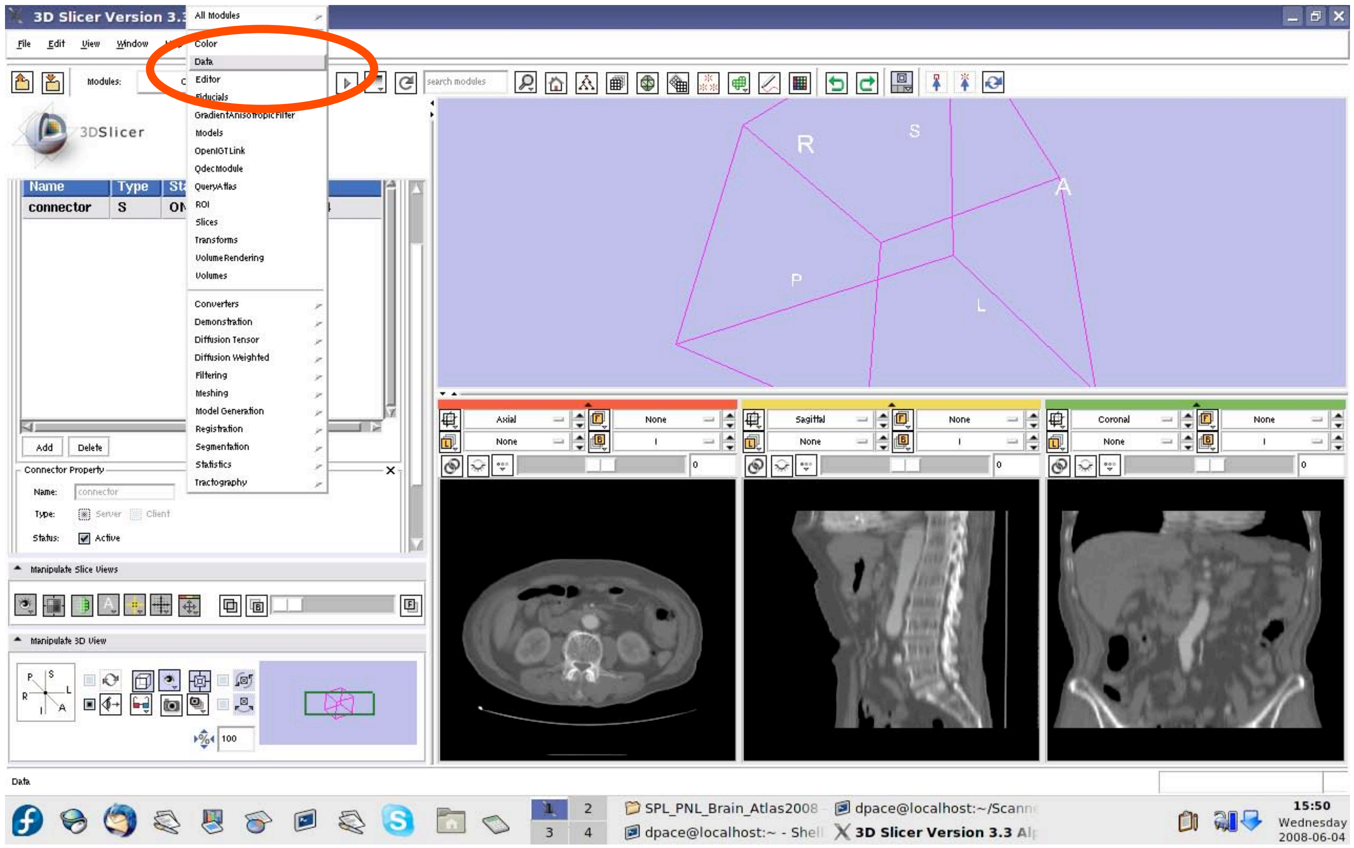
# Start the IGSTK test program

The transforms being sent are written to the terminal as you move the tool



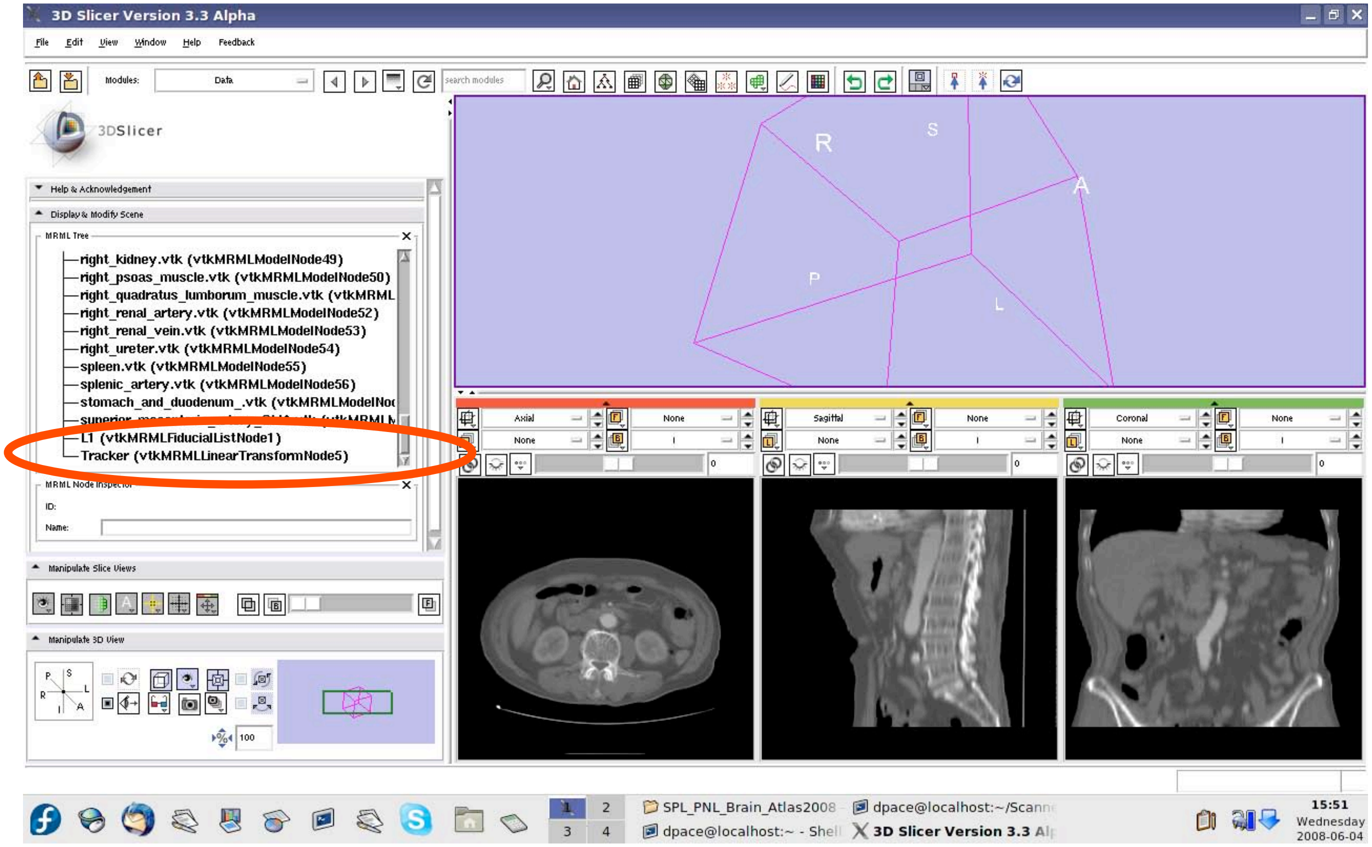
# Start the IGSTK test program

Open the Data module



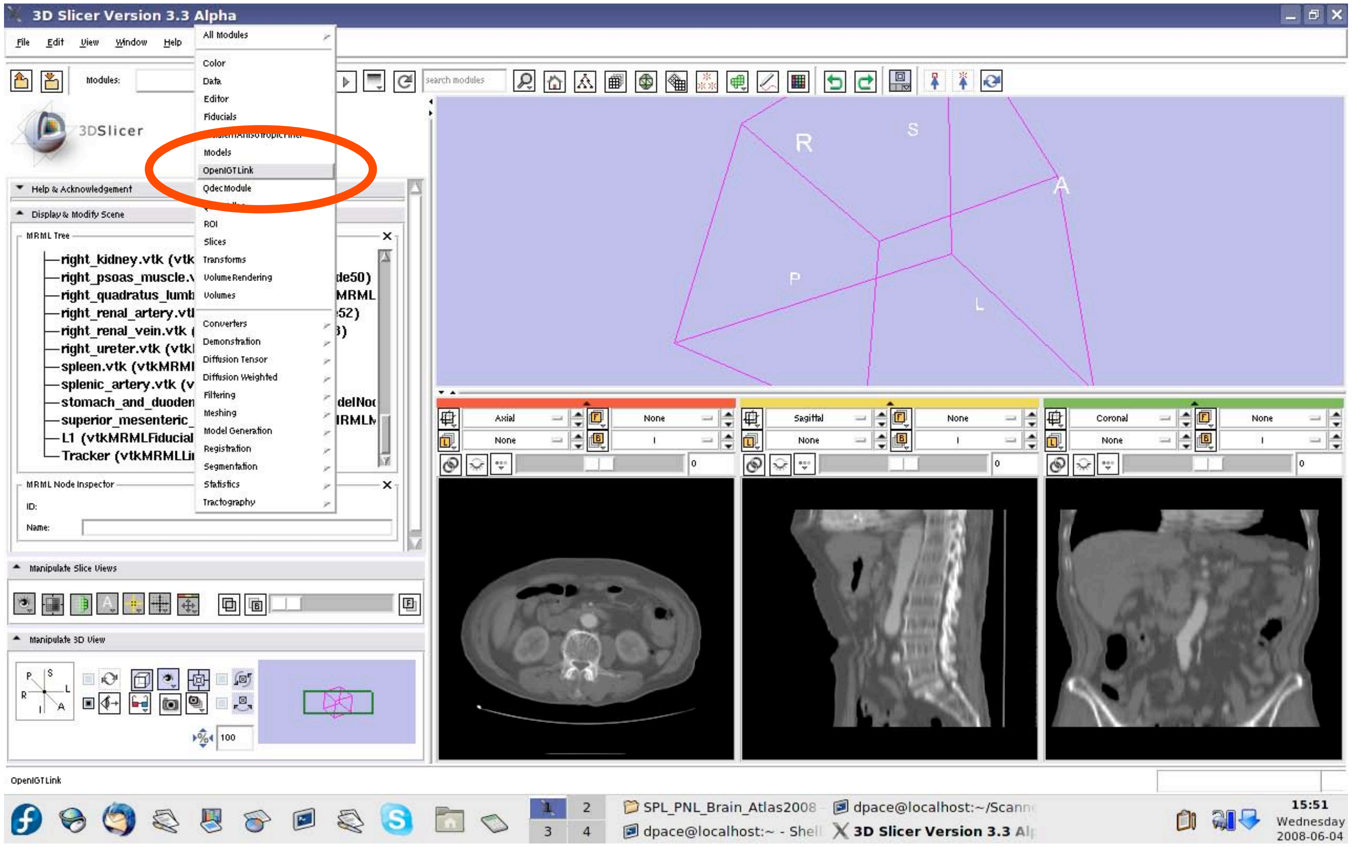
# Start the IGSTK test program

The new tracker node is a transform node - you can see it at the bottom of the MRML tree



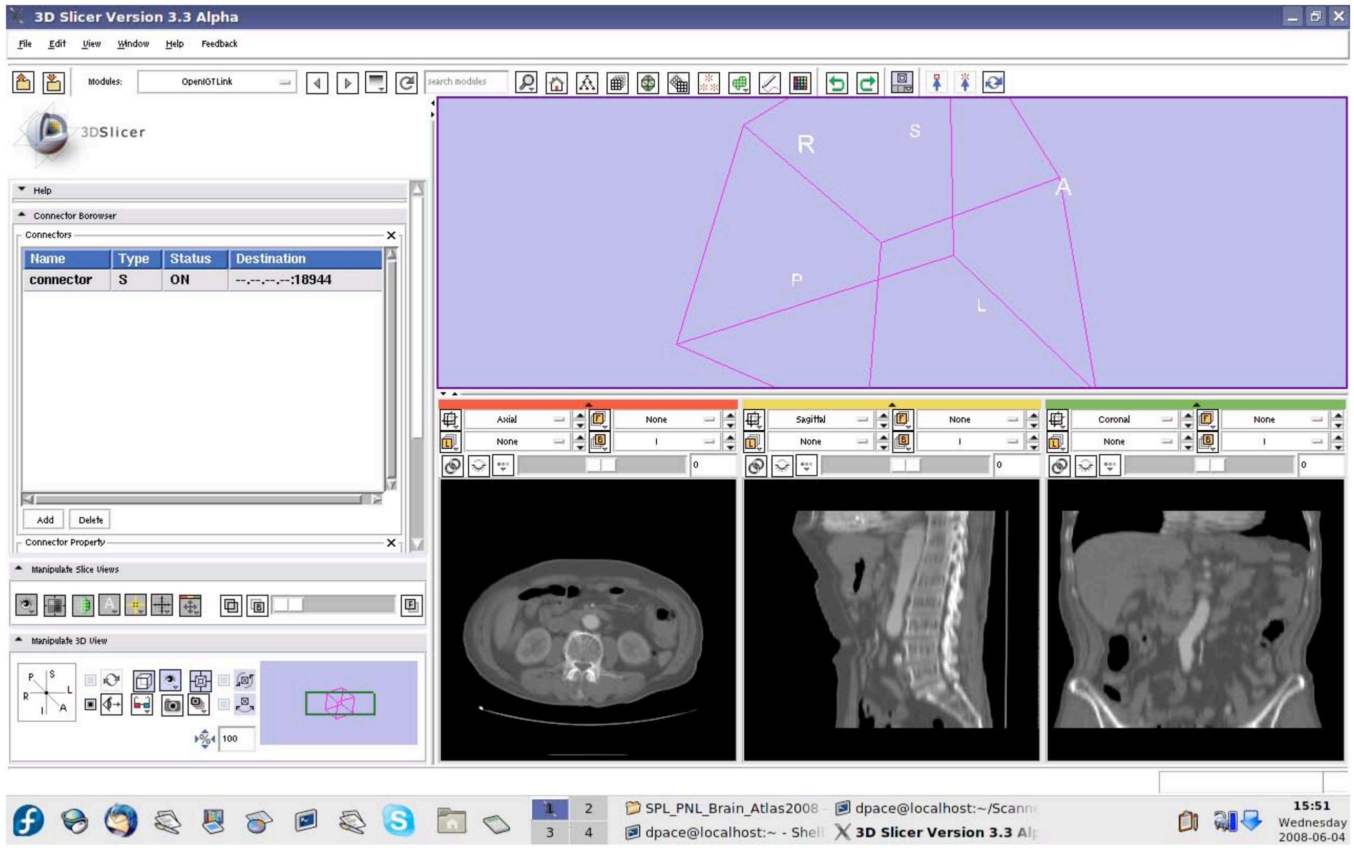
# Start the IGSTK test program

Open the OpenIGTLink module



# Start the IGSTK test program

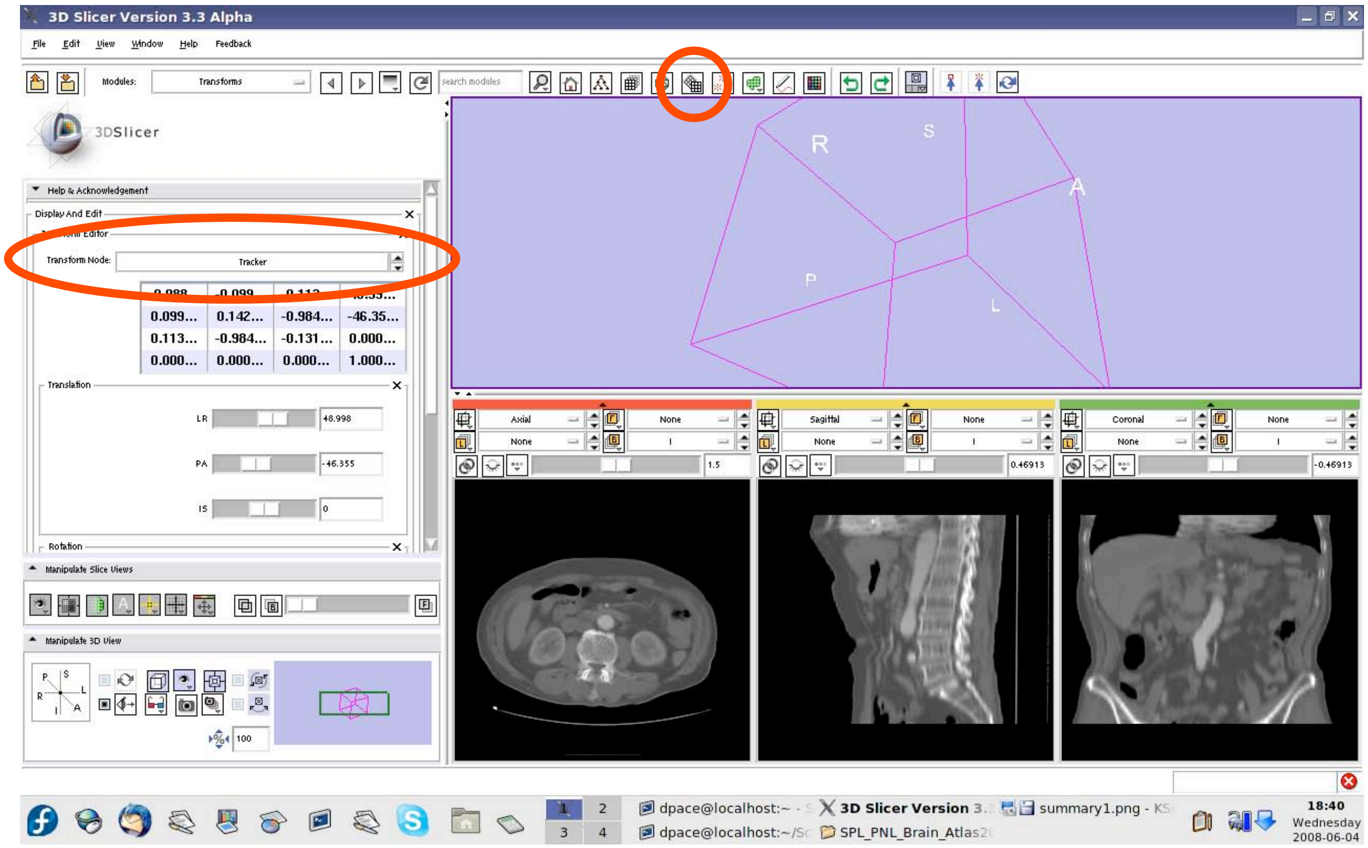
Note that the connector status is now set to “ON” instead of “WAIT”



# Start the IGSTK test program

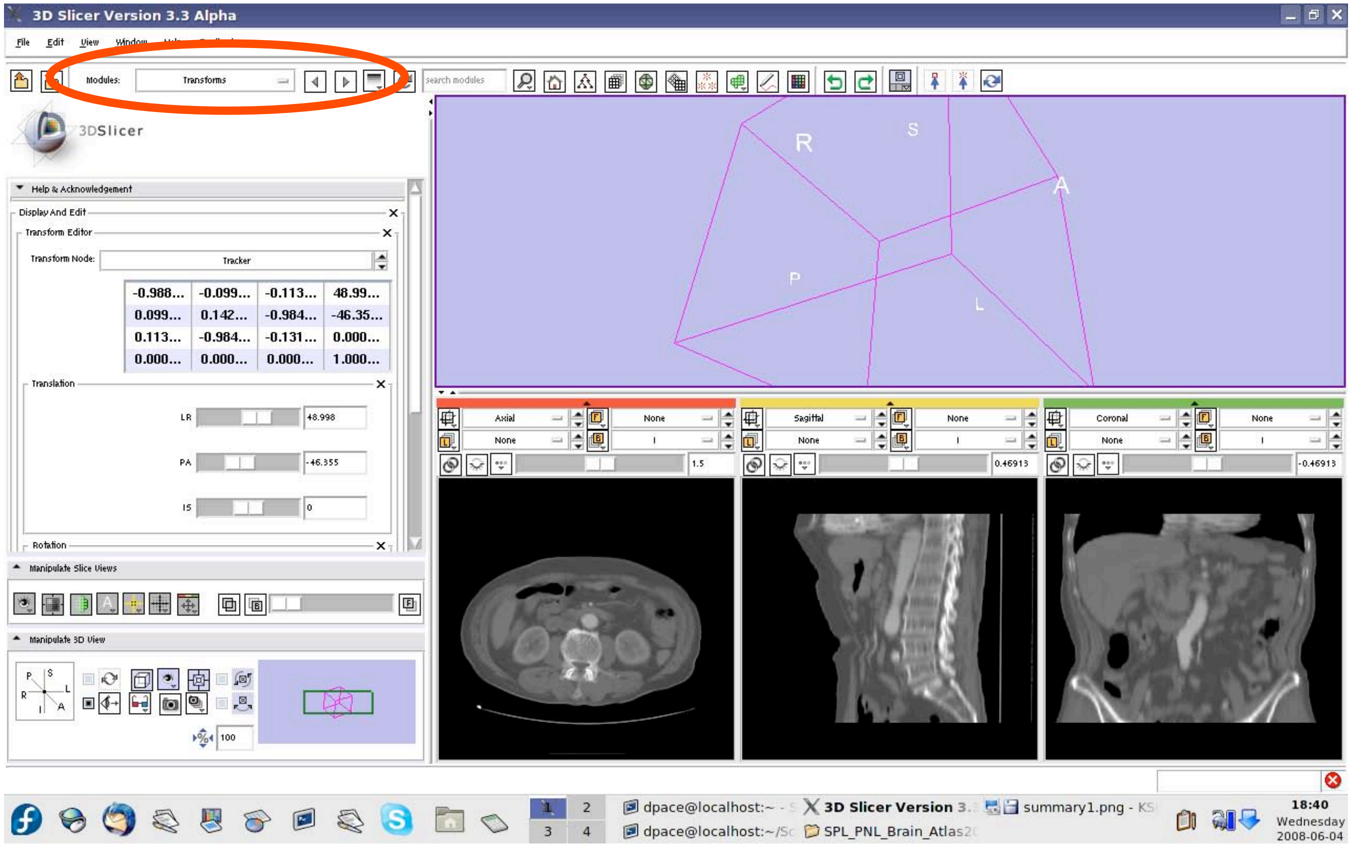
Open the Transforms module

Click on the new Tracker transform to see the changing transformation matrix



# Start the IGSTK test program

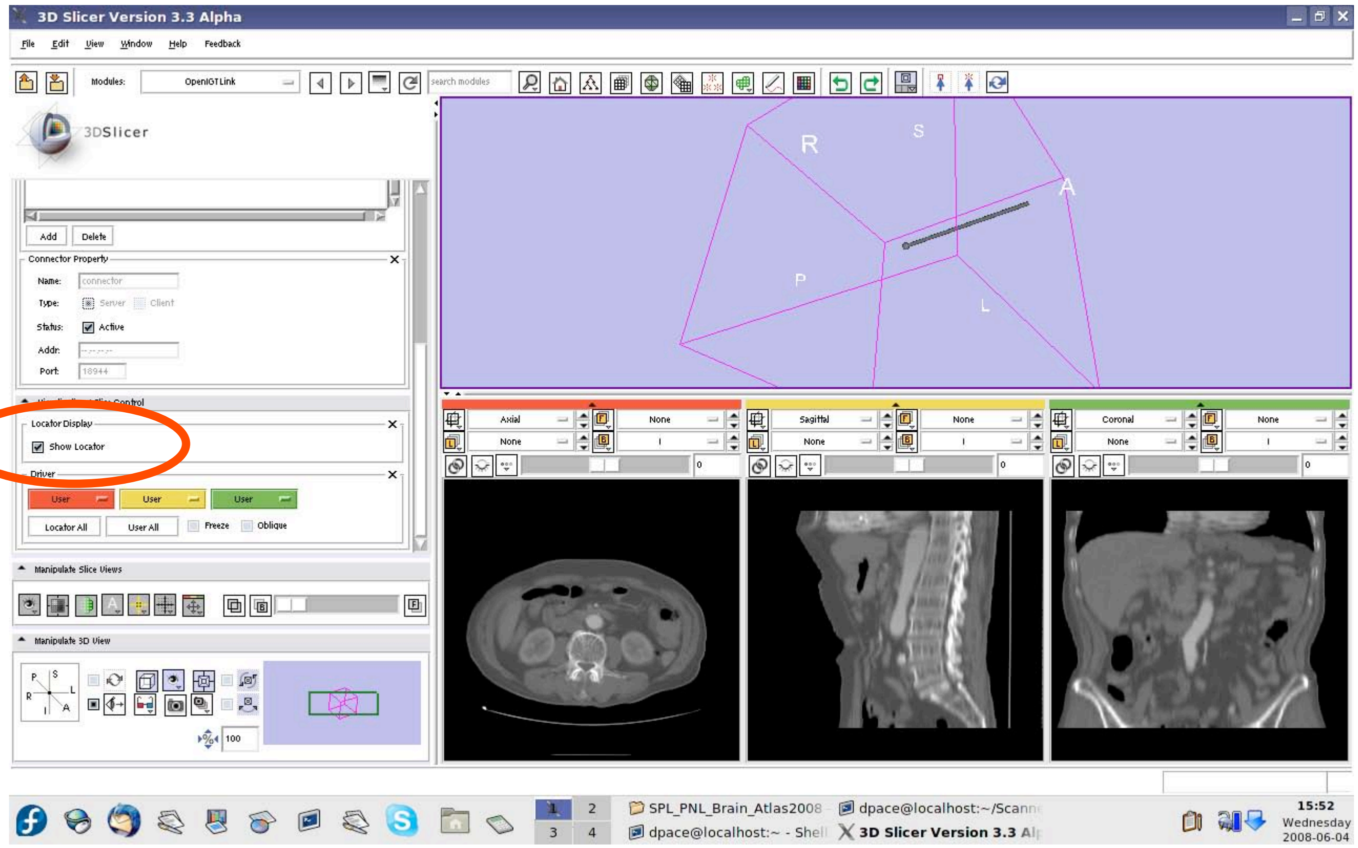
Open the OpenIGTLink module



# Show the transform using the locator

In the Visualization/ Slice Control pane, click the "Show Locator" button

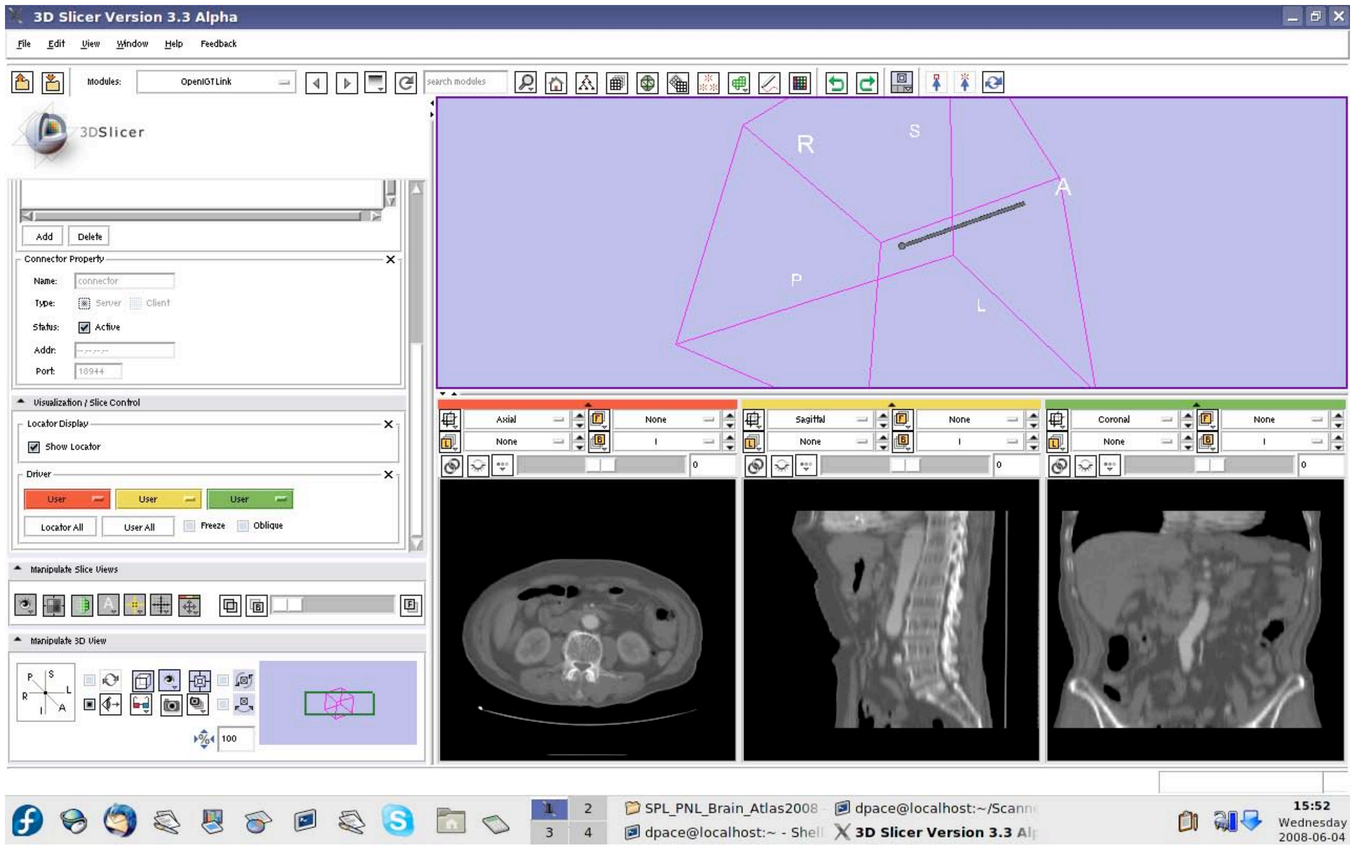
If the locator does not appear, make sure that the IGTLocator model is set to "visible" in the Models module



3DSlicer

# Show the transform using the locator

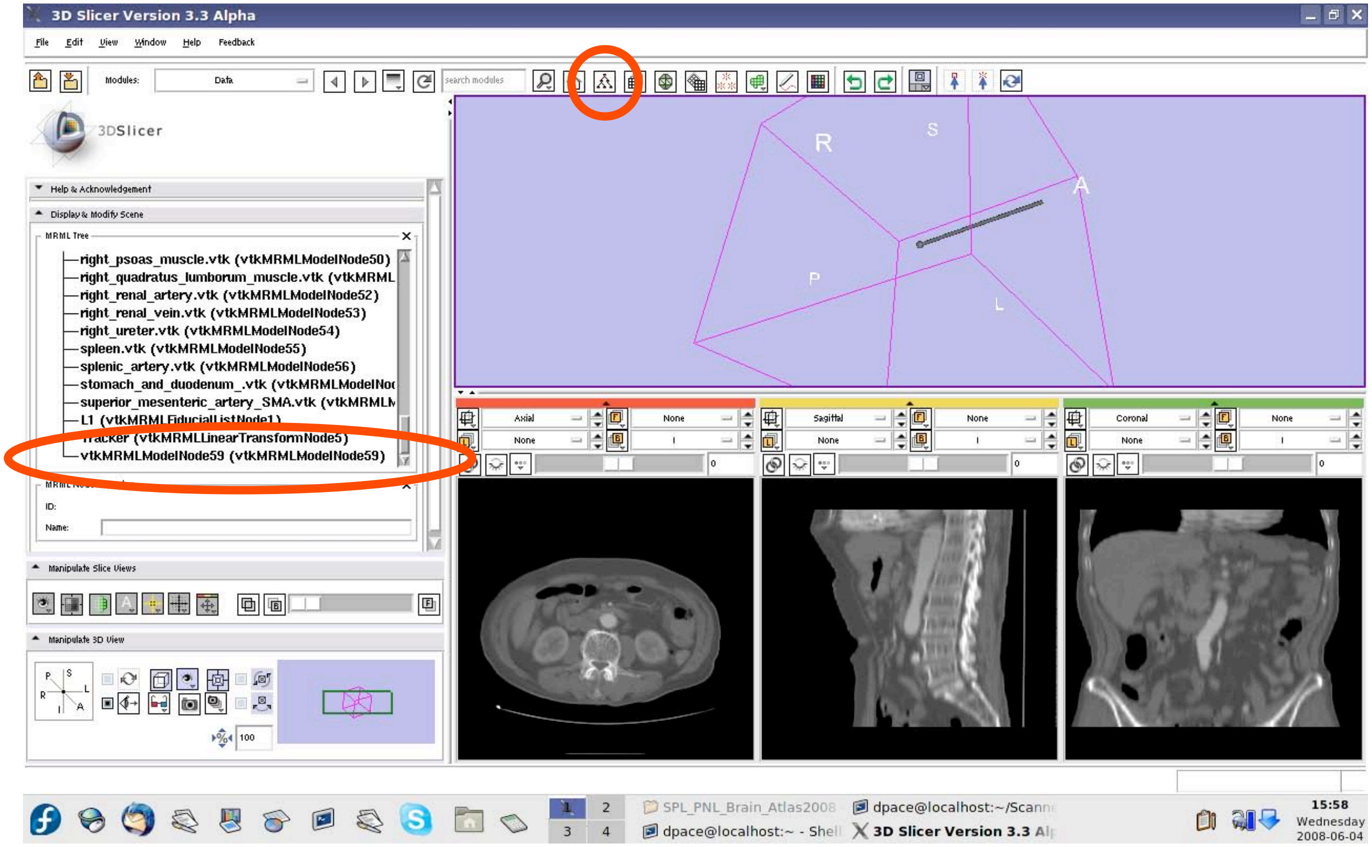
The round end shows the tool's position, and the cylinder shows the tool's orientation



# Show the transform using the locator

Open the Data module

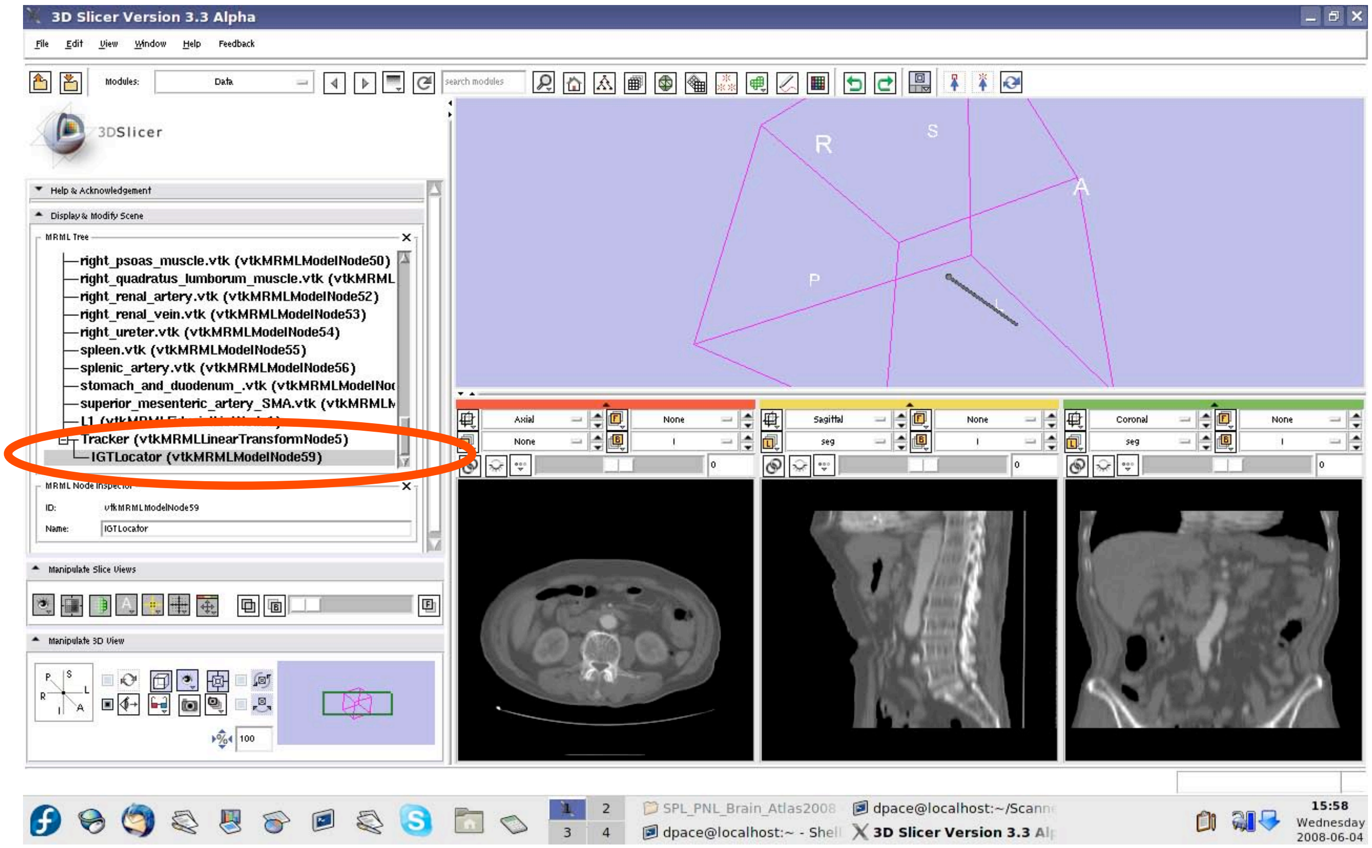
The new locator node is a model node at the bottom of the MRML tree



# Show the transform using the locator

Drag the locator node under the Tracker node

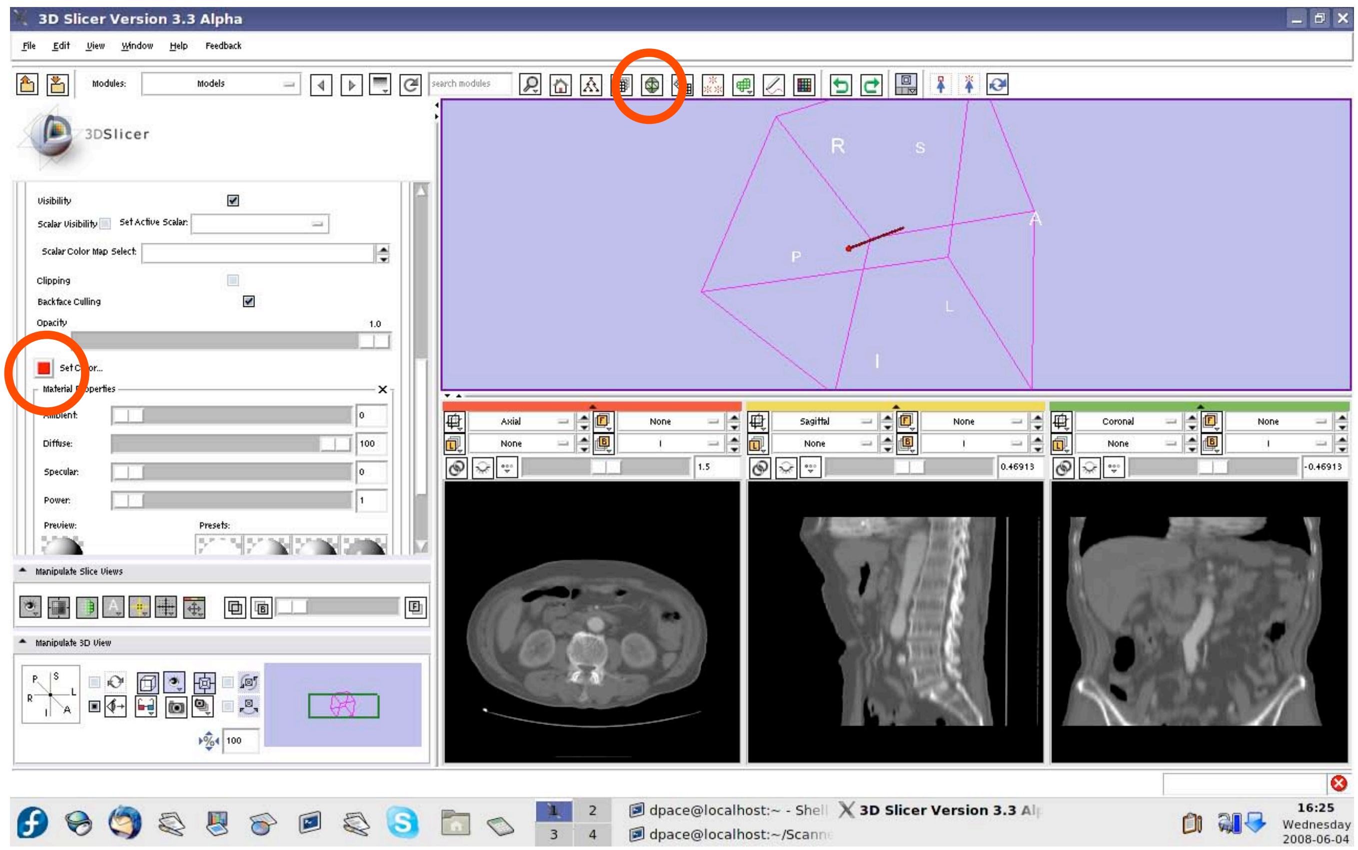
The Tracker transform is now applied to the locator model - it will move according to the transforms from the tracker simulator



# Reslice the images using the tracker transform

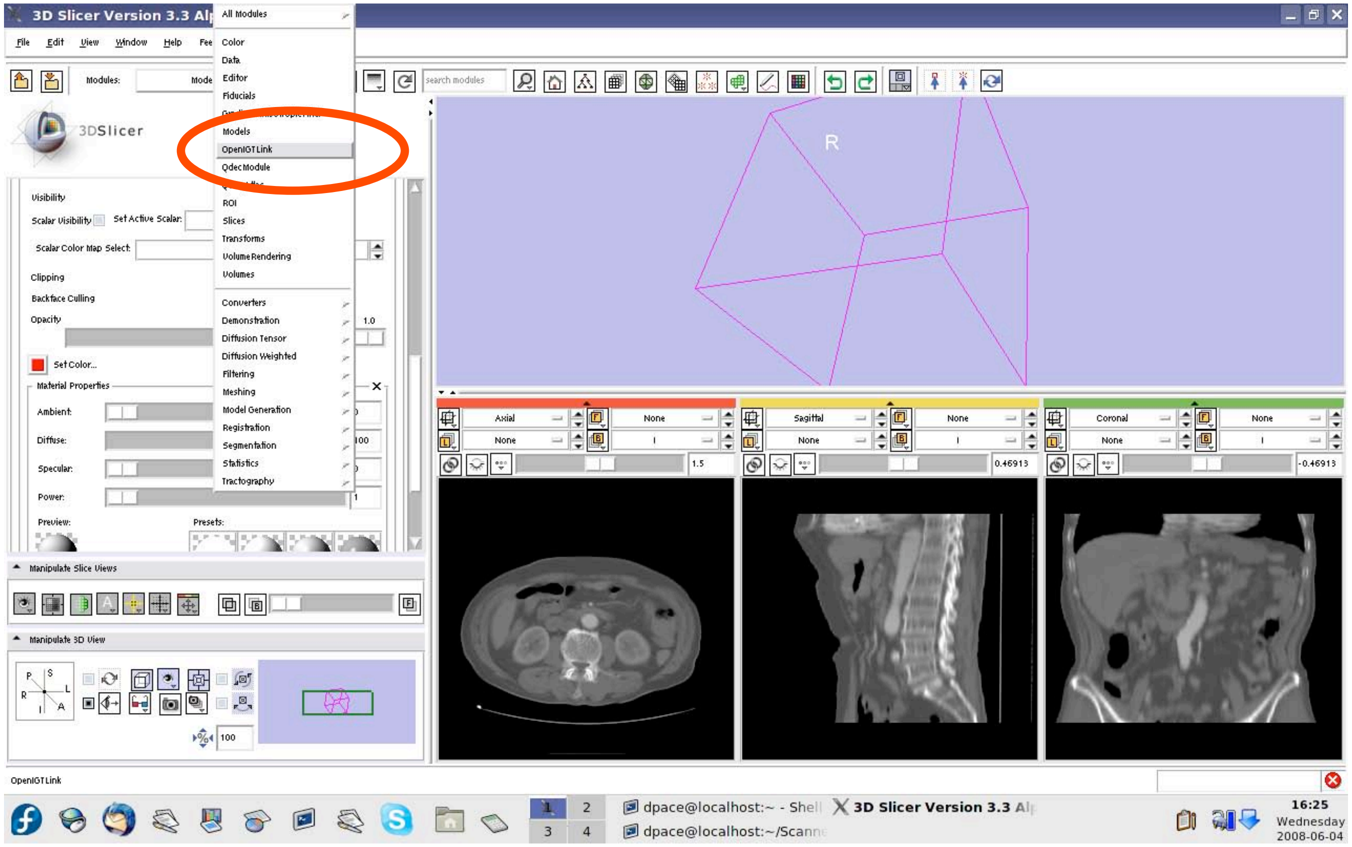
Open the Models module

Select the IGTLocator model as the selected model and change its colour to red



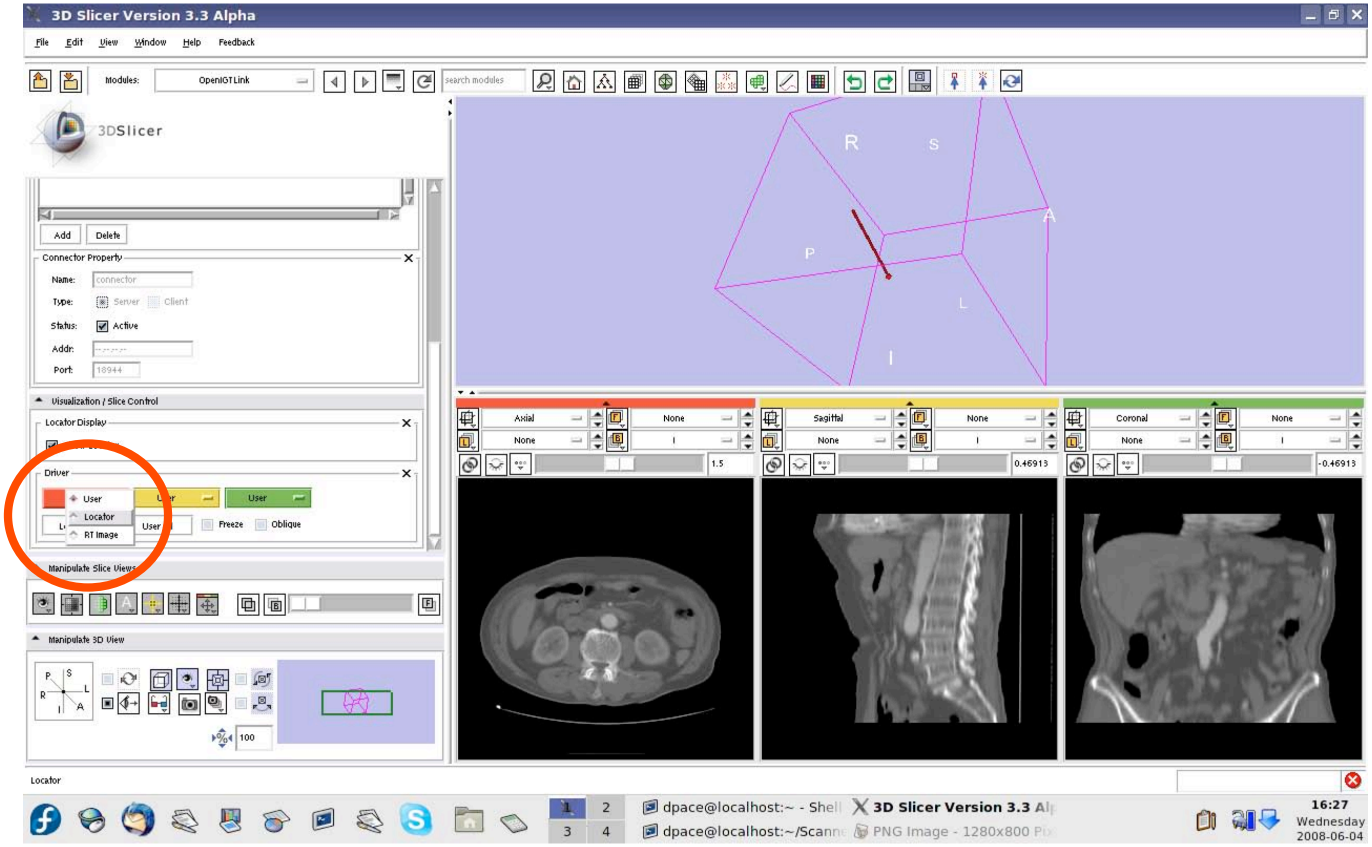
# Reslice the images using the tracker transform

Open the OpenIGTLink module



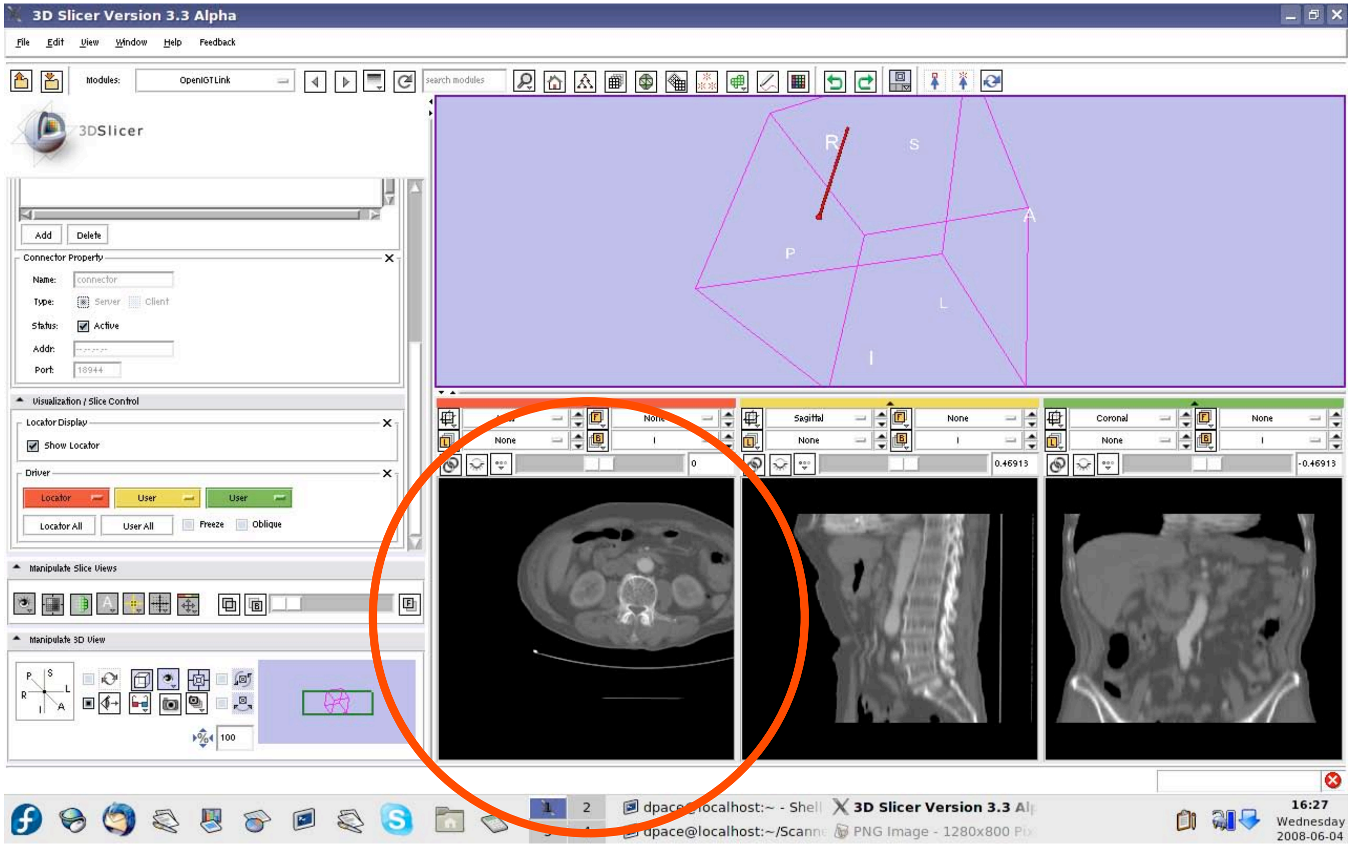
# Reslice the images using the tracker transform

Set the driver for the red (axial) slice to "Locator"



# Reslice the images using the tracker transform

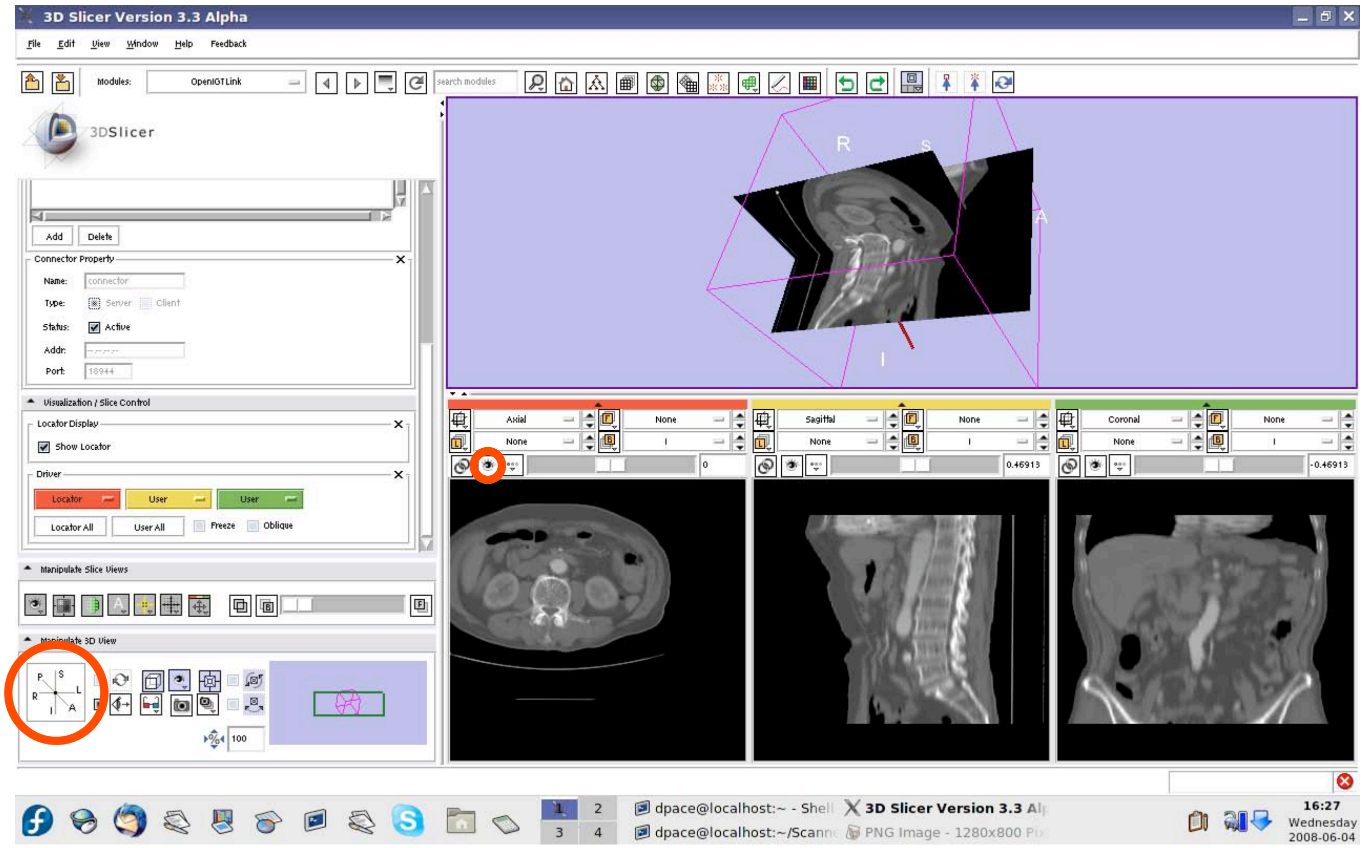
The axial slice moves as the locator moves



# Reslice the images using the tracker transform

Click on the “visibility” button

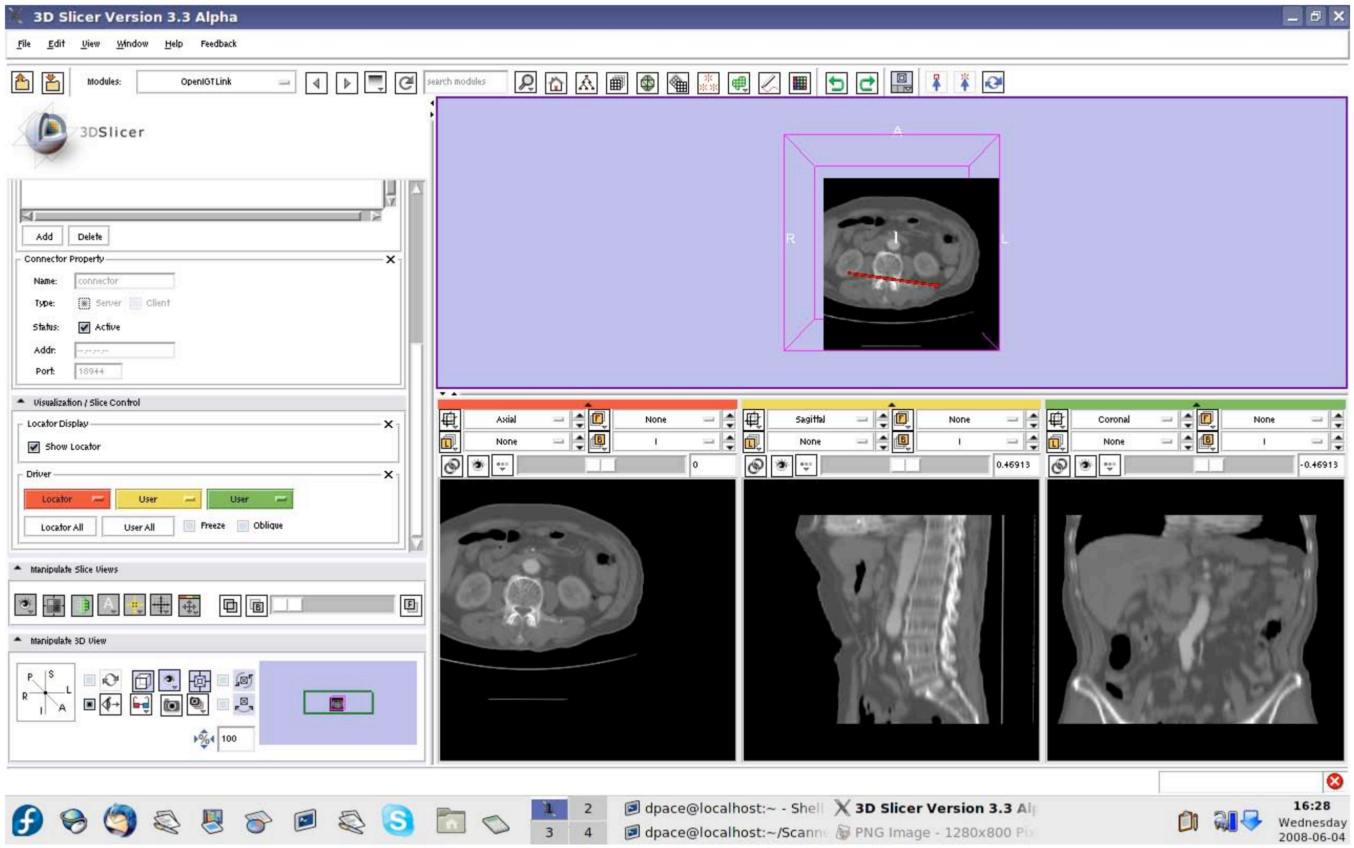
Change the view in the 3D viewer by clicking on the “I” (inferior) button on the “Manipulate 3D View” pane



# Reslice the images using the tracker transform

Note that the axial slice moves as you move the tool

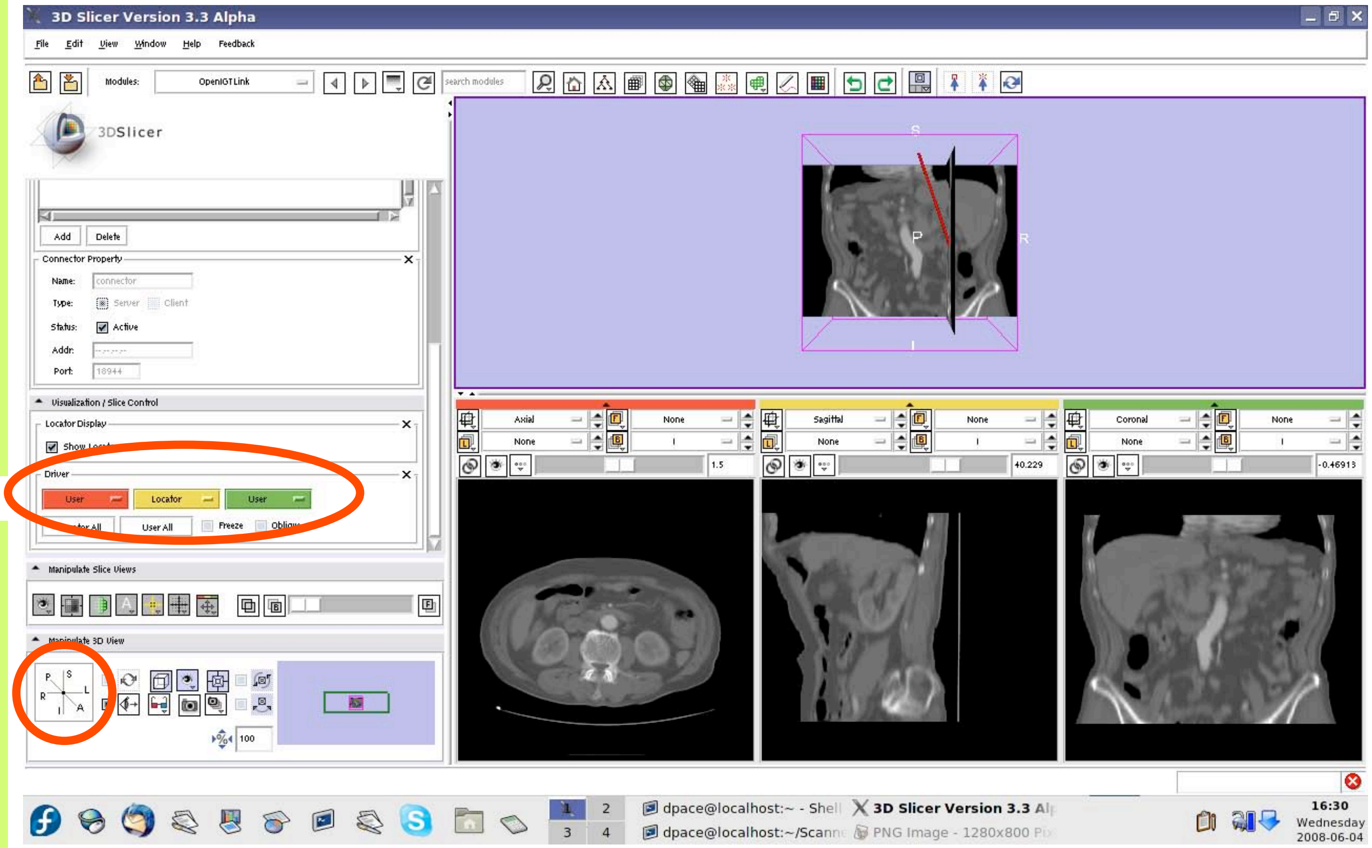
This is because the image origin in the left-right direction is set to the locator's position in the left-right direction



# Reslice the images using the tracker transform

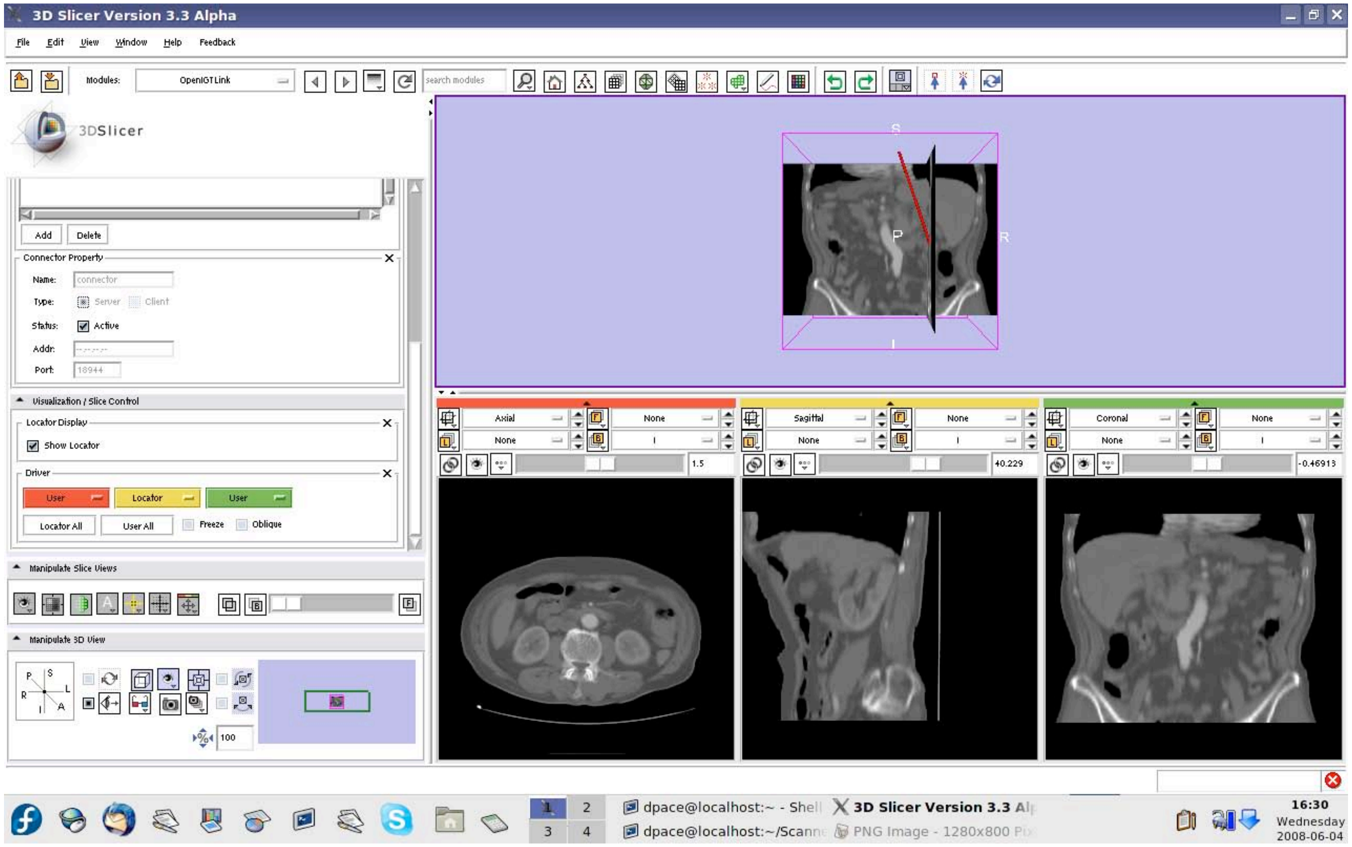
Set the driver for the red (axial) slice to “User” and the driver for the yellow (sagittal slice) to “Locator”

Click on the “P” (posterior) button on the “Manipulate 3D View” pane



# Reslice the images using the tracker transform

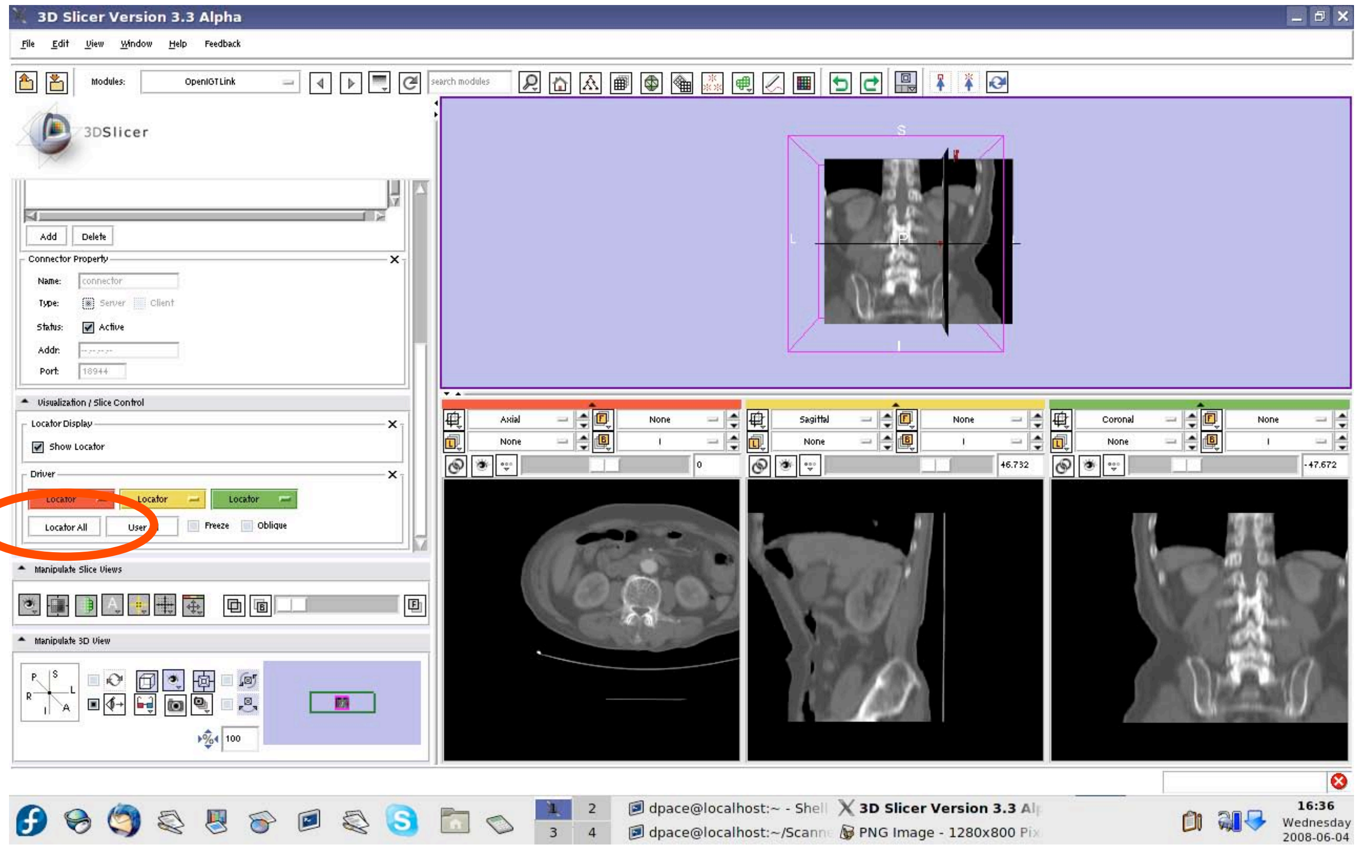
Note that the sagittal slice moves from left to right as the tool moves



# Reslice the images using the tracker transform

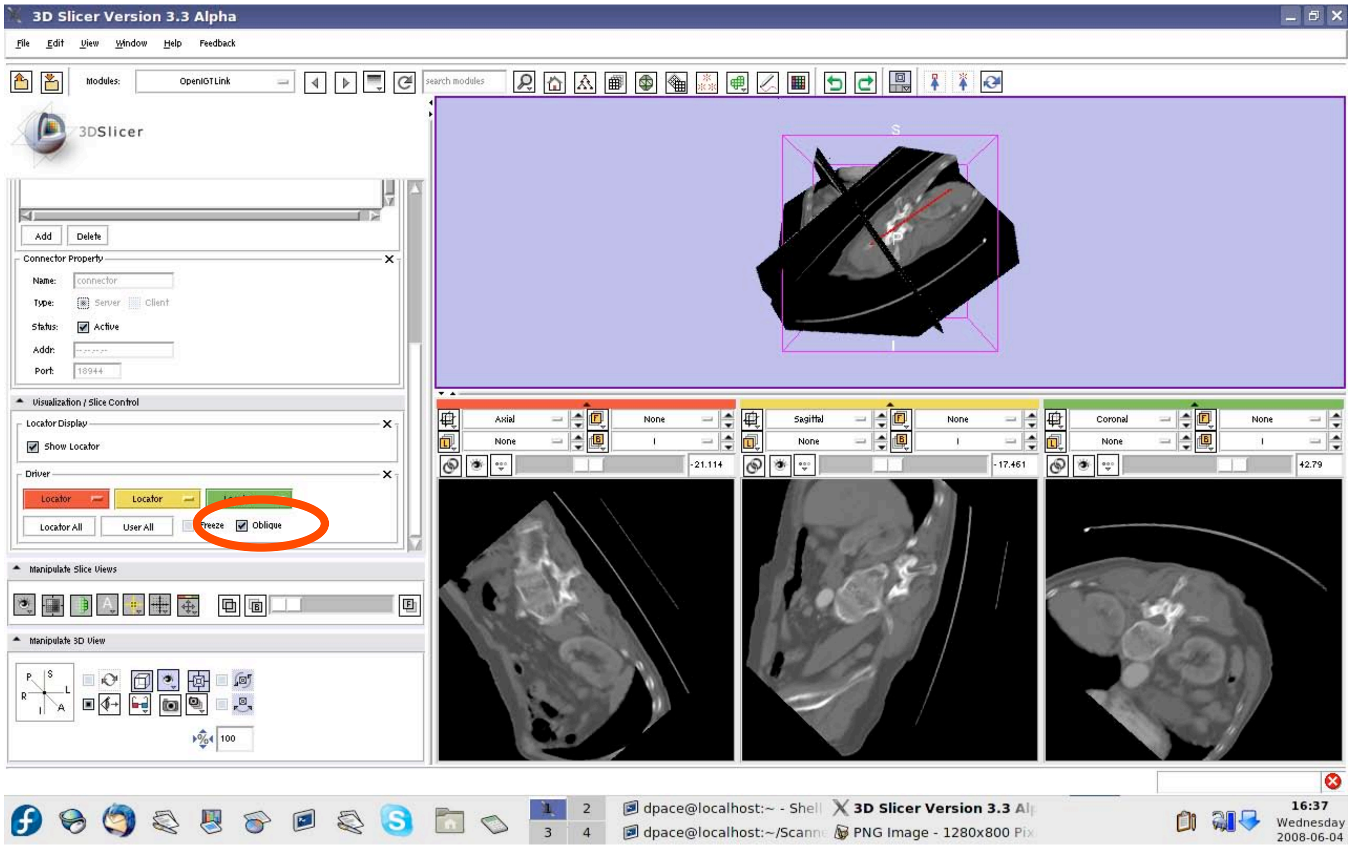
You can click on the "Locator All" button to set the driver to "Locator" for all of the slice views.

The image origin is set to the locator's position.



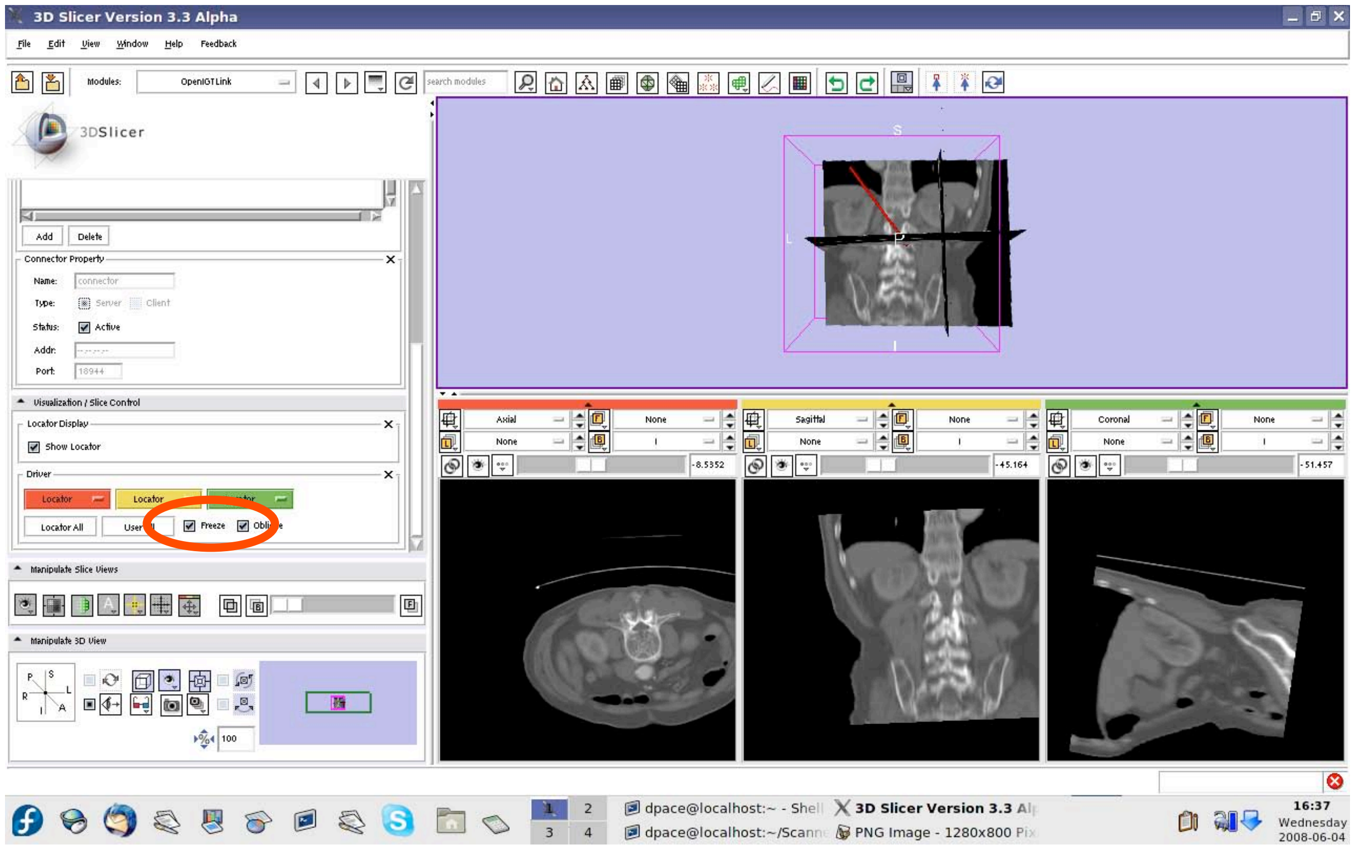
# Reslice the images using the tracker transform

Check the “oblique” box to slice the image volume according to the tool’s orientation - the coordinate system is setup so that one axis is parallel to the locator’s orientation



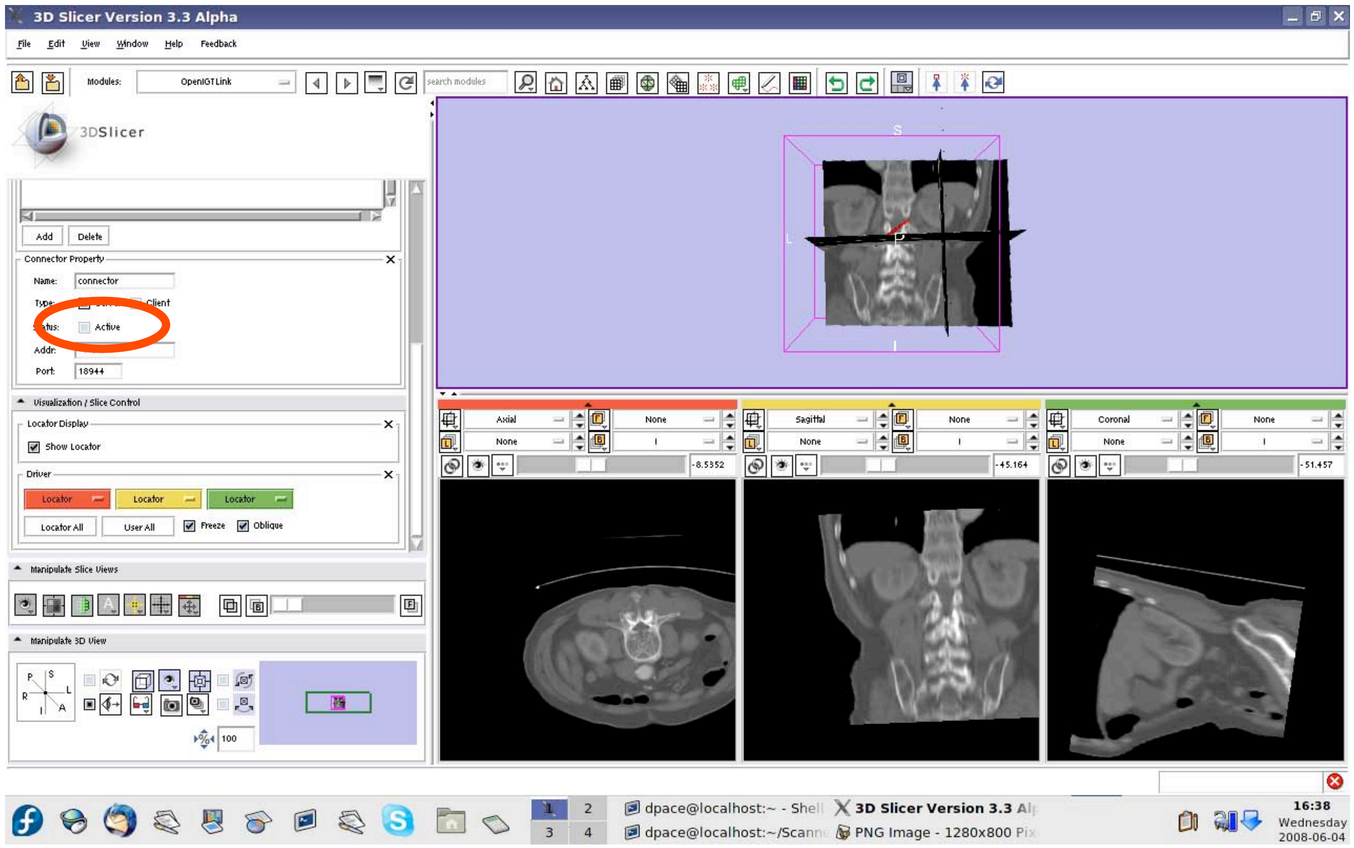
# Reslice the images using the tracker transform

Check the "Freeze" box to freeze the images in both the 3D Viewer and the three slices viewers (the locator keeps moving)



# Turn off the OpenIGTLink connection

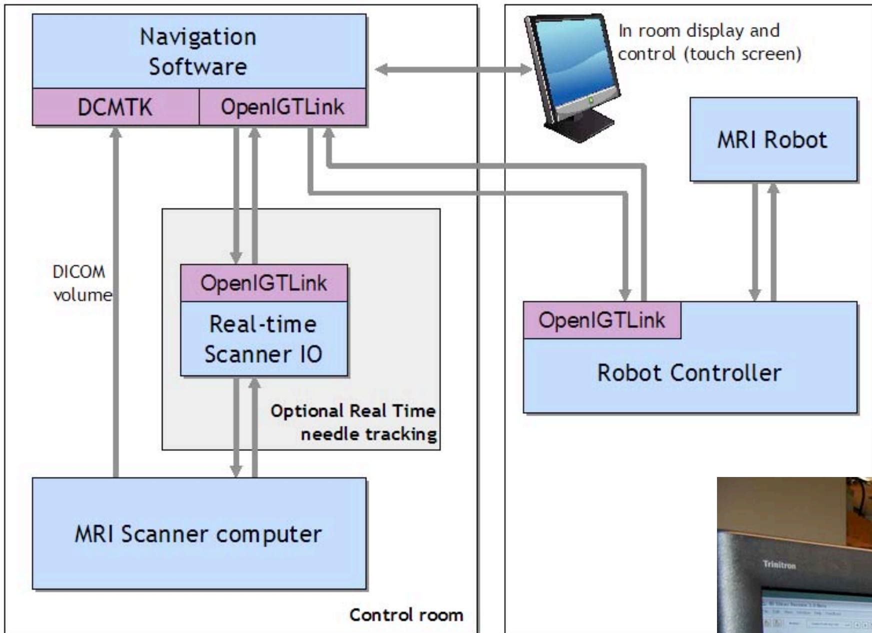
Click on the “Active” box to disconnect the OpenIGTLink connection



# Tutorial outline

1. Introduction to surgical navigation
2. Interfacing Slicer3 with external devices using OpenIGTLink
3. The OpenIGTLink protocol
4. Hands-on navigation using the NDI Aurora tracking device
5. **Examples of OpenIGTLink in use**

# Examples of OpenIGTLink in use




Prostate biopsy robot under MRI-guidance

Volume-rendered 4D ultrasound

# Overview

- In this tutorial, you learned:
  - How the OpenIGTLink protocol standardizes messages between Slicer3 and external devices
  - How to set up OpenIGTLink connections using an actual tracking device
  - How to visualize the tracker transforms
  - How to reslice image volumes using the tracker transforms
  - How OpenIGTLink is currently being used in practice

# Conclusions

- Slicer3 can interact with common devices used in Image Guided Therapy
- OpenIGTLink is evolving technology - expect lots of active development!
- Slicer3 is free open-source software that allows IGT researchers to share algorithms and work within a common framework

# For more information…

- For a description of the OpenIGTLink protocol: http://www.na-mic.org/Wiki/index.php/OpenIGTLink