Building Software Systems for Image-Guided Robot-Assisted Interventions with SlicerROS2

04/19/2023

Overview

Part 1: AI segmentation from tracked ultrasound images

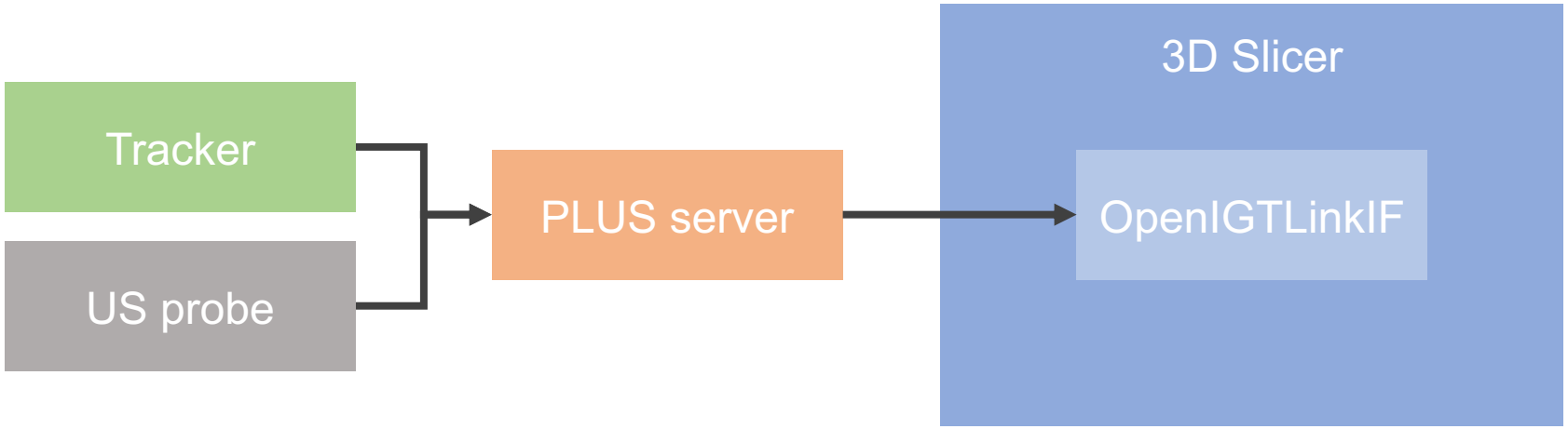
Part 2: Intro to SlicerROS2

Part 3: Image registration and path planning in Slicer

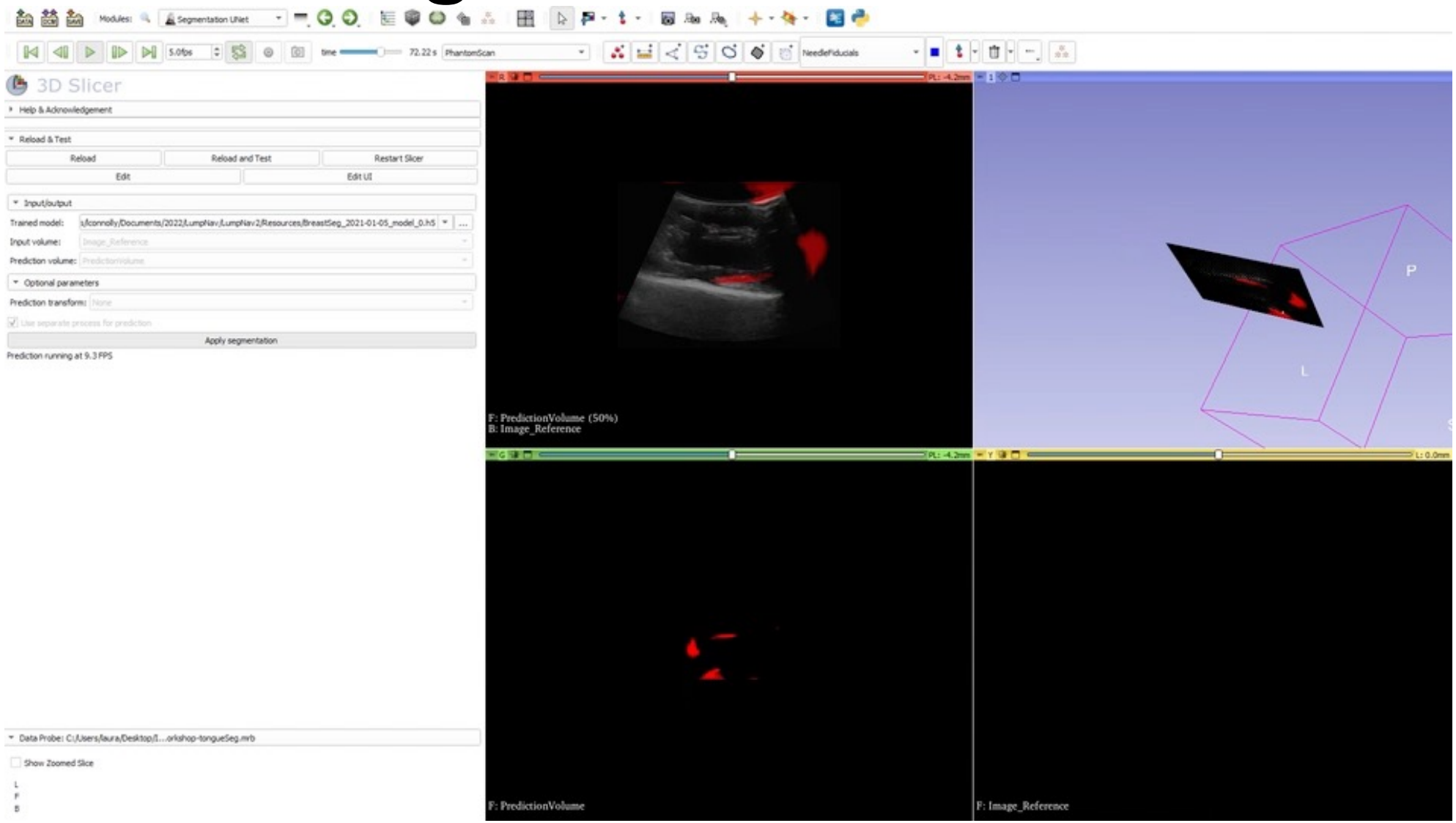
Part 1: AI segmentation from tracked ultrasound images

How to capture tracked ultrasound in 3D Slicer

- Make sure your ultrasound and tracker are PLUS compatible by checking here for a list of supported devices: https://plustoolkit.github.io/features
- For this demo we used an Aurora tracker and a Clarius US probe
- For more information, check out the SlicerIGT tutorials here: https://www.slicerigt.org/wp/user-tutorial/



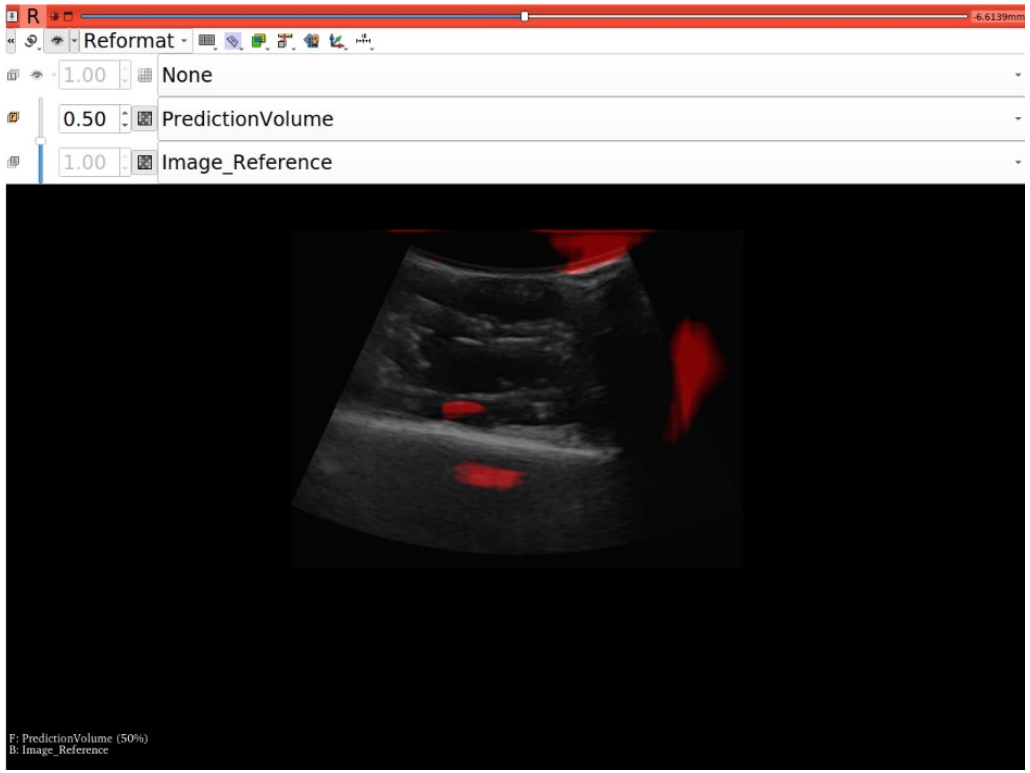
AI segmentation in Slicer



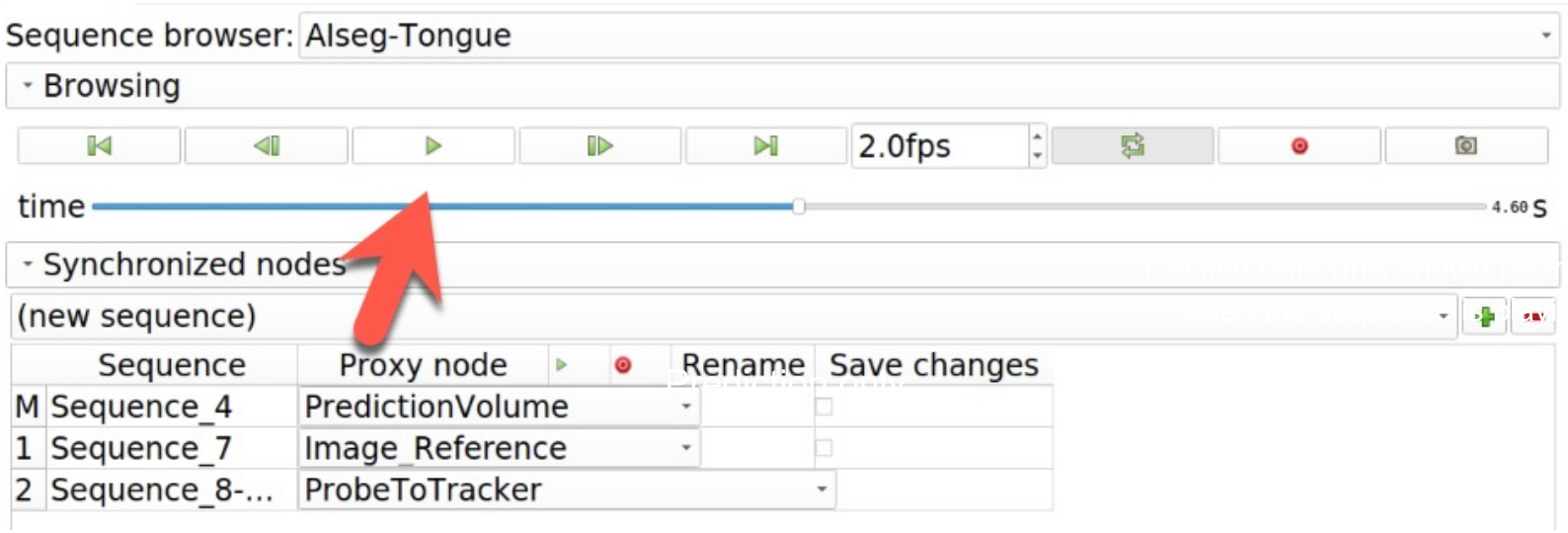

R
IRP: 194.4mm
F: PredictionVolume (50%)
B: Image_Reference

AI tumor segmentation from ultrasound

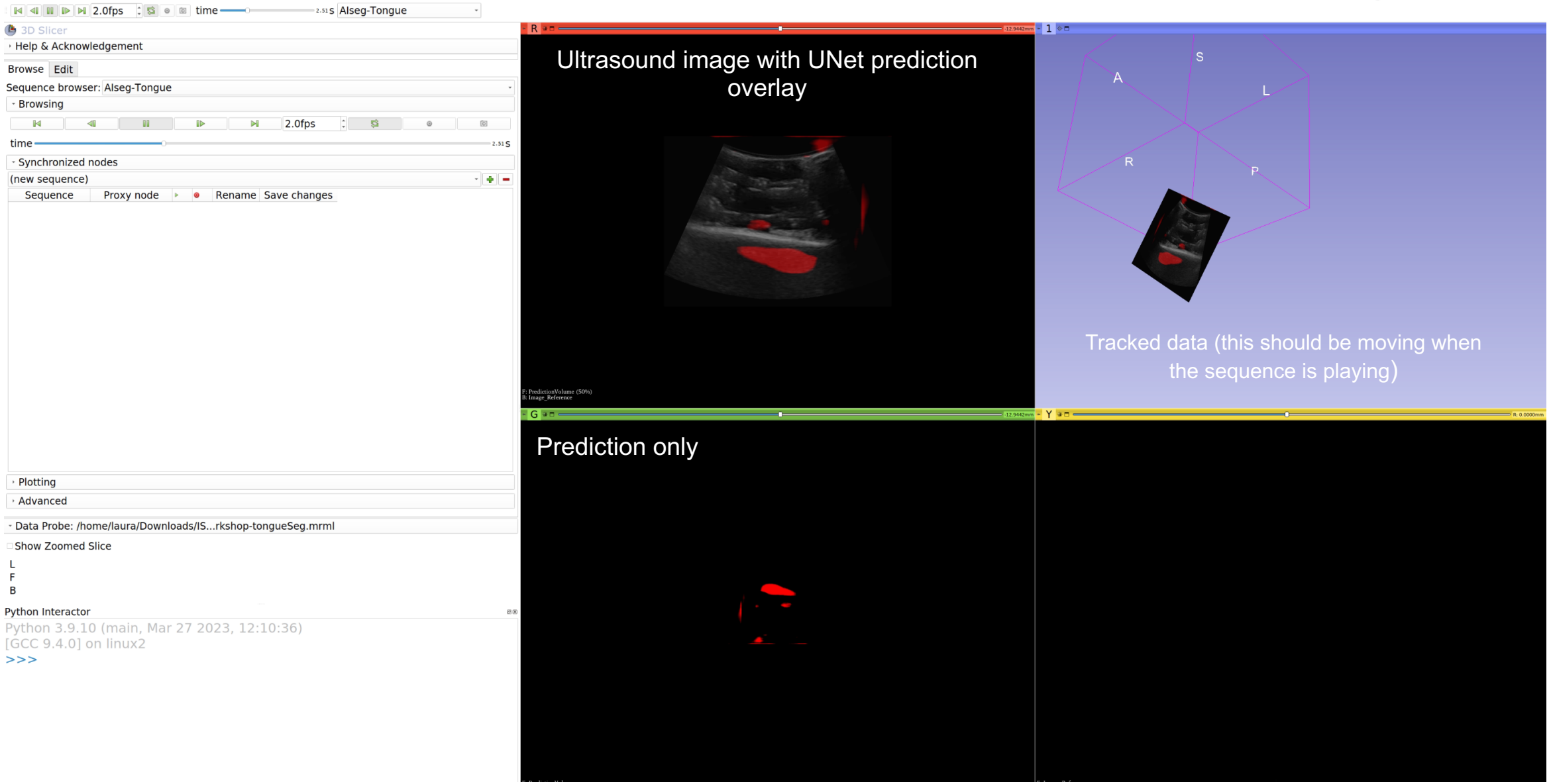
1. Load the scene “ISMRWorkshop-tongueSeg.mrb”
2. Update your slice views (see below):



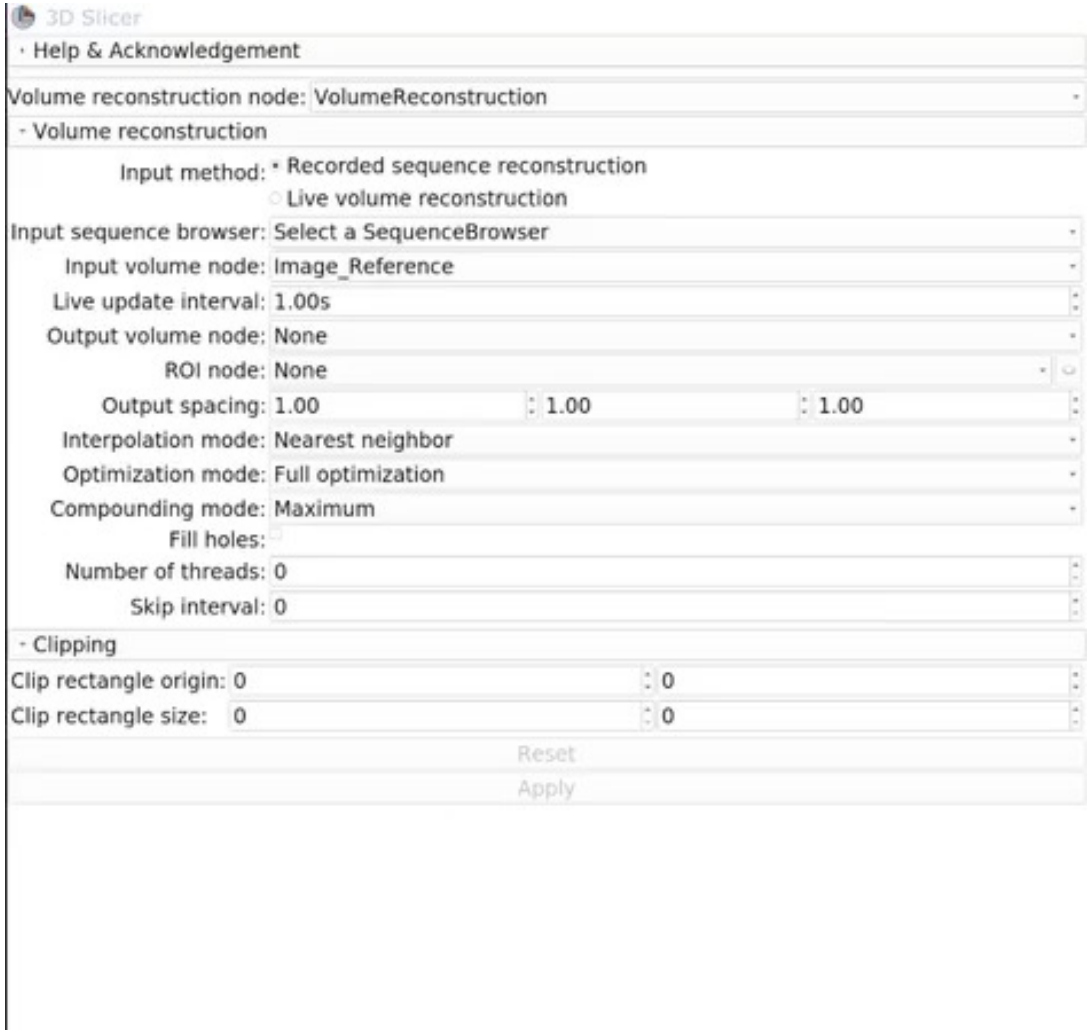
3. Playback the tracked US data and live prediction using the *Sequences* module



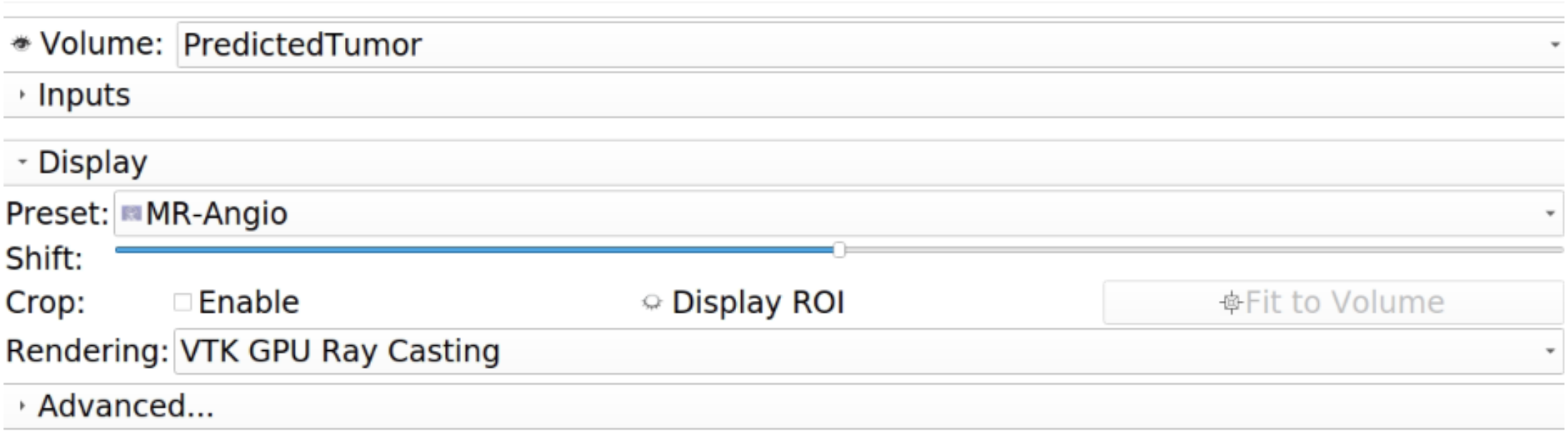
References: https://www.youtube.com/watch?v=WyscpAee3vw – Tamas Ungi


2.0fps
time
2.51 s
Alseg-Tongue
3D Slicer
Help & Acknowledgement
Browse
Edit
Sequence browser: Alseg-Tongue
Browsing
2.0fps
time
2.51 s
Synchronized nodes
(new sequence)
Sequence
Proxy node
Rename
Save changes
Plotting
Advanced
Data Probe: /home/laura/Downloads/IS...rkshop-tongueSeg.mrml
Show Zoomed Slice
L
F
B
Python Interactor
Python 3.9.10 (main, Mar 27 2023, 12:10:36)
[GCC 9.4.0] on linux2
>>>
Ultrasound image with UNet prediction overlay
S
A
L
R
P
Tracked data (this should be moving when the sequence is playing)
Prediction only

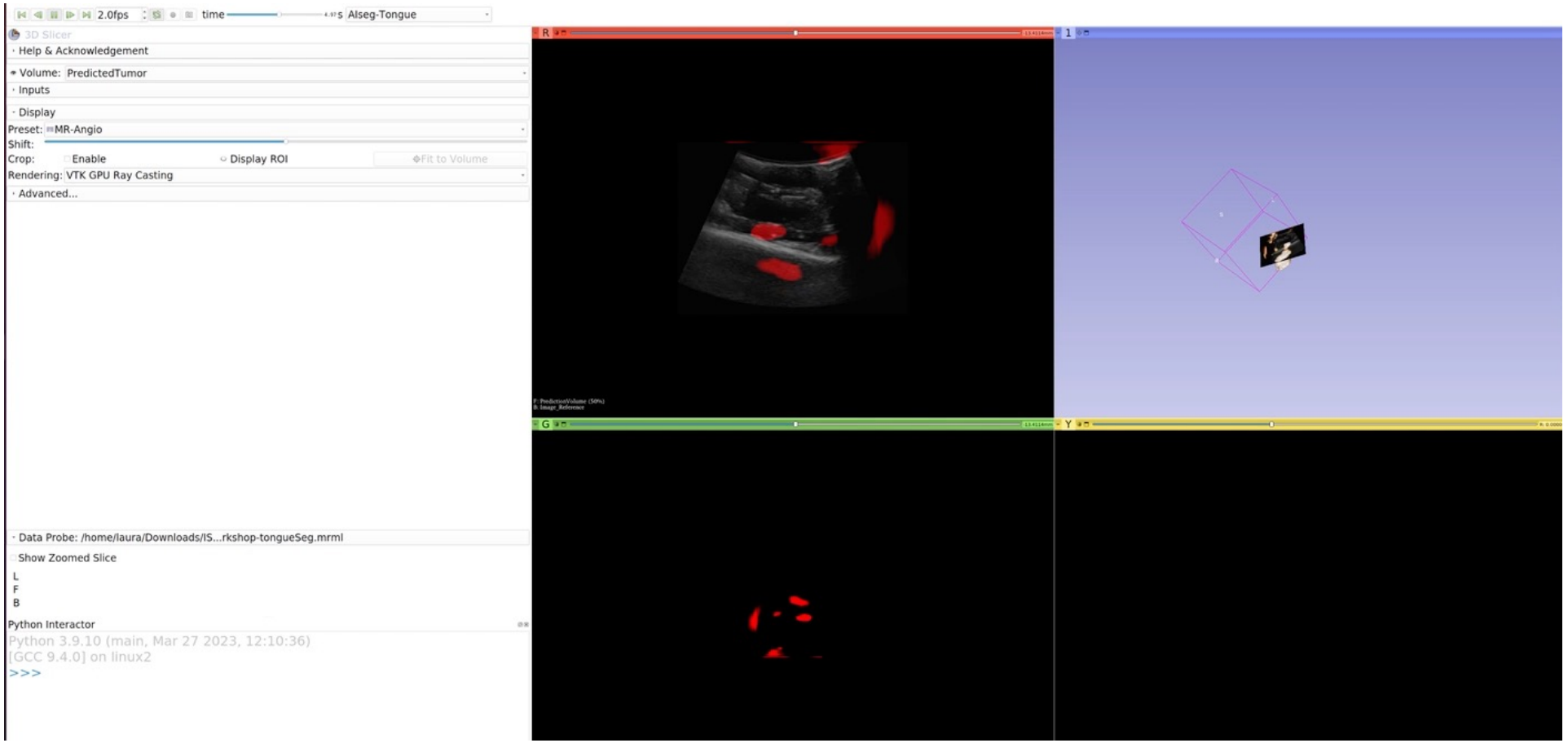
4. Switch to the *Volume Reconstruction* and select the following:



4. Finally, switch to the *Volume Rendering* module and select the following:



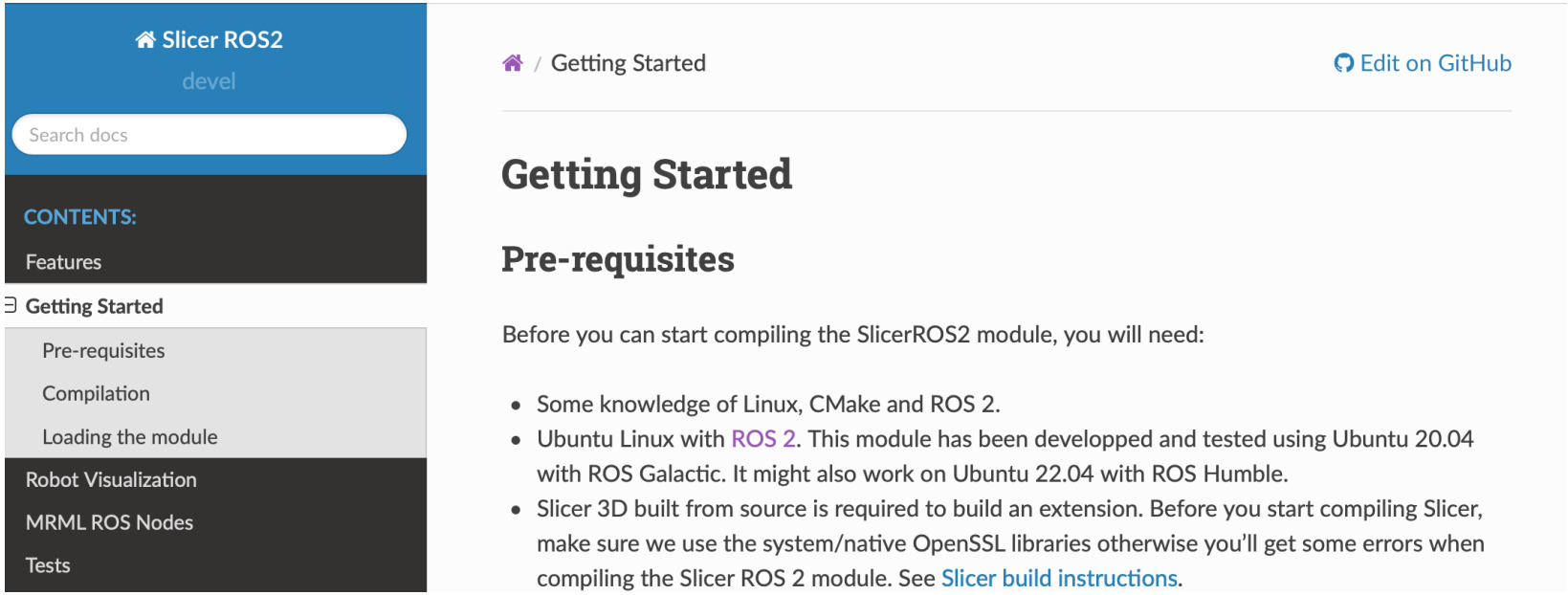
Make sure the “AISeg-tongue” sequence is playing and you should see something like this:



Part 2: Intro to SlicerROS2

Getting started on SlicerROS2

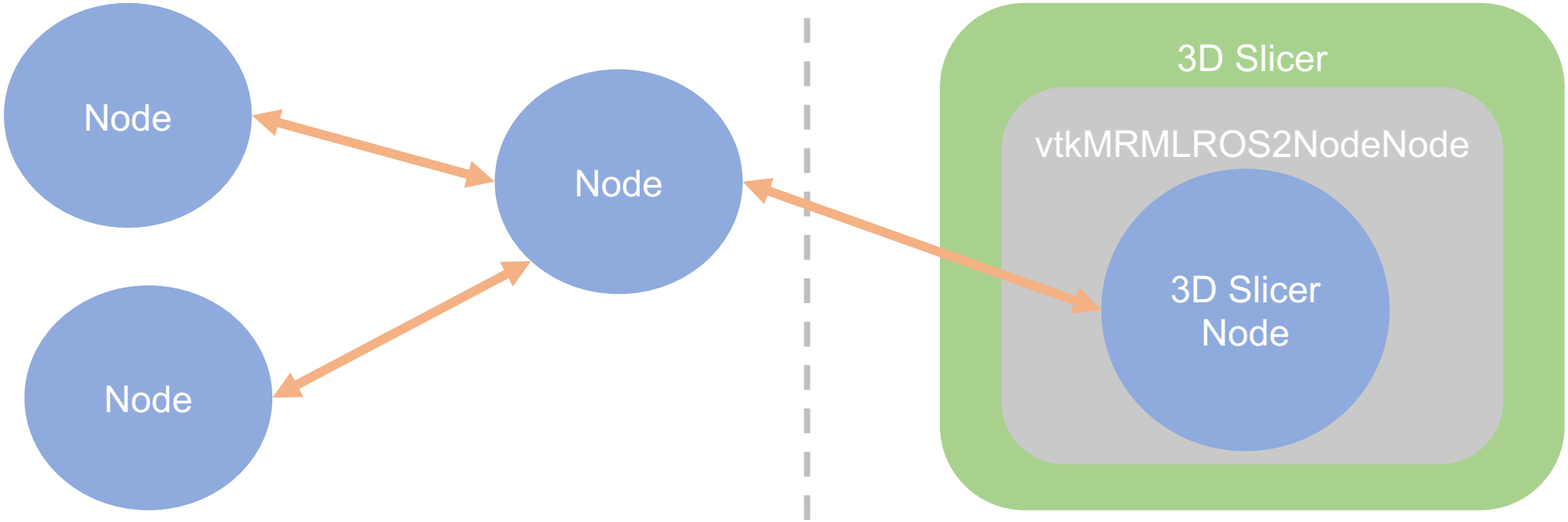
- SlicerROS2 has been built in your docker container
- The instructions for installing on your own computer, and a lot of the examples we'll go through today are available here: https://slicer-ros2.readthedocs.io/en/devel/index.html



How does it work?

3D Slicer is treated like a ROS node

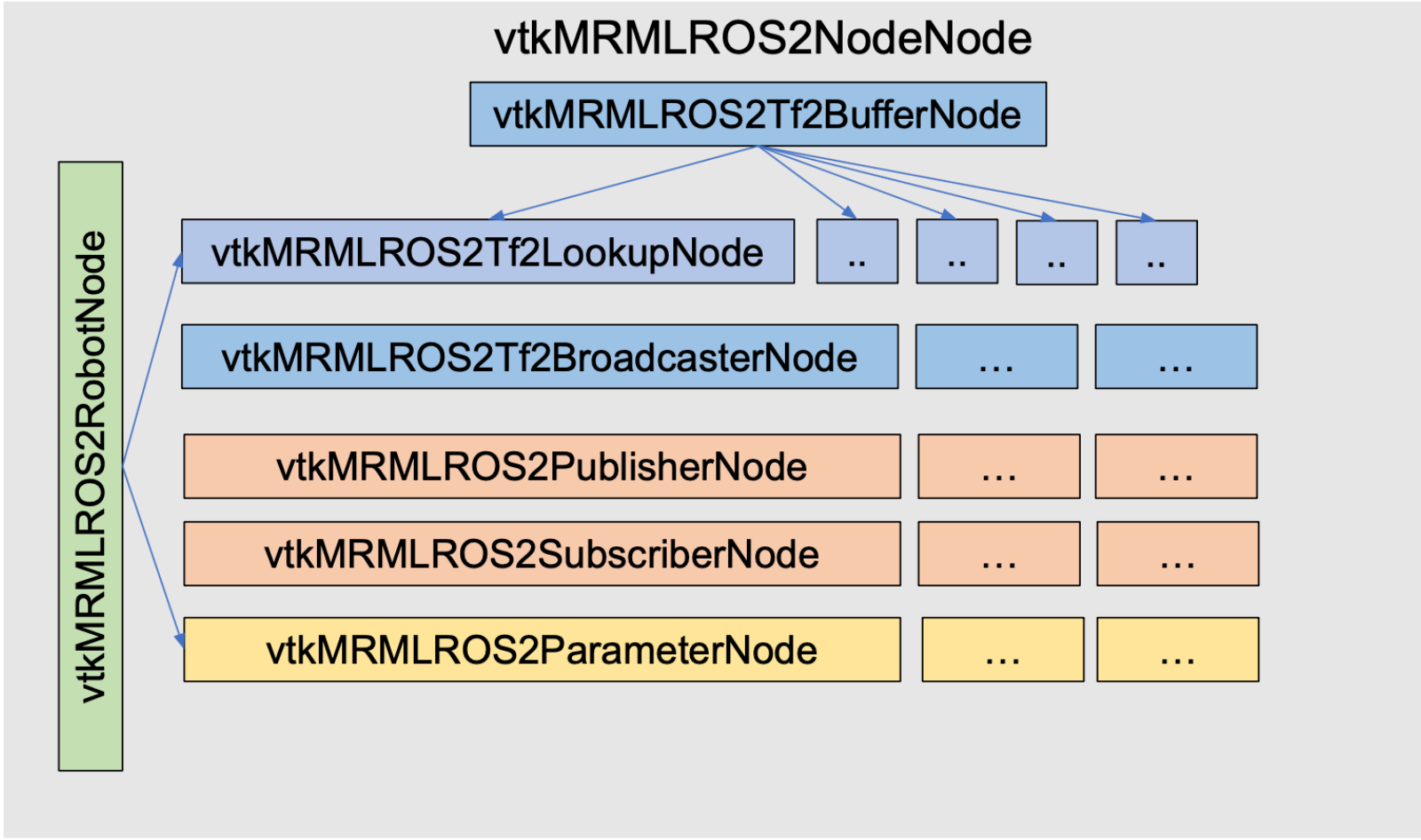
ROS communication mechanisms are packaged as MRML nodes



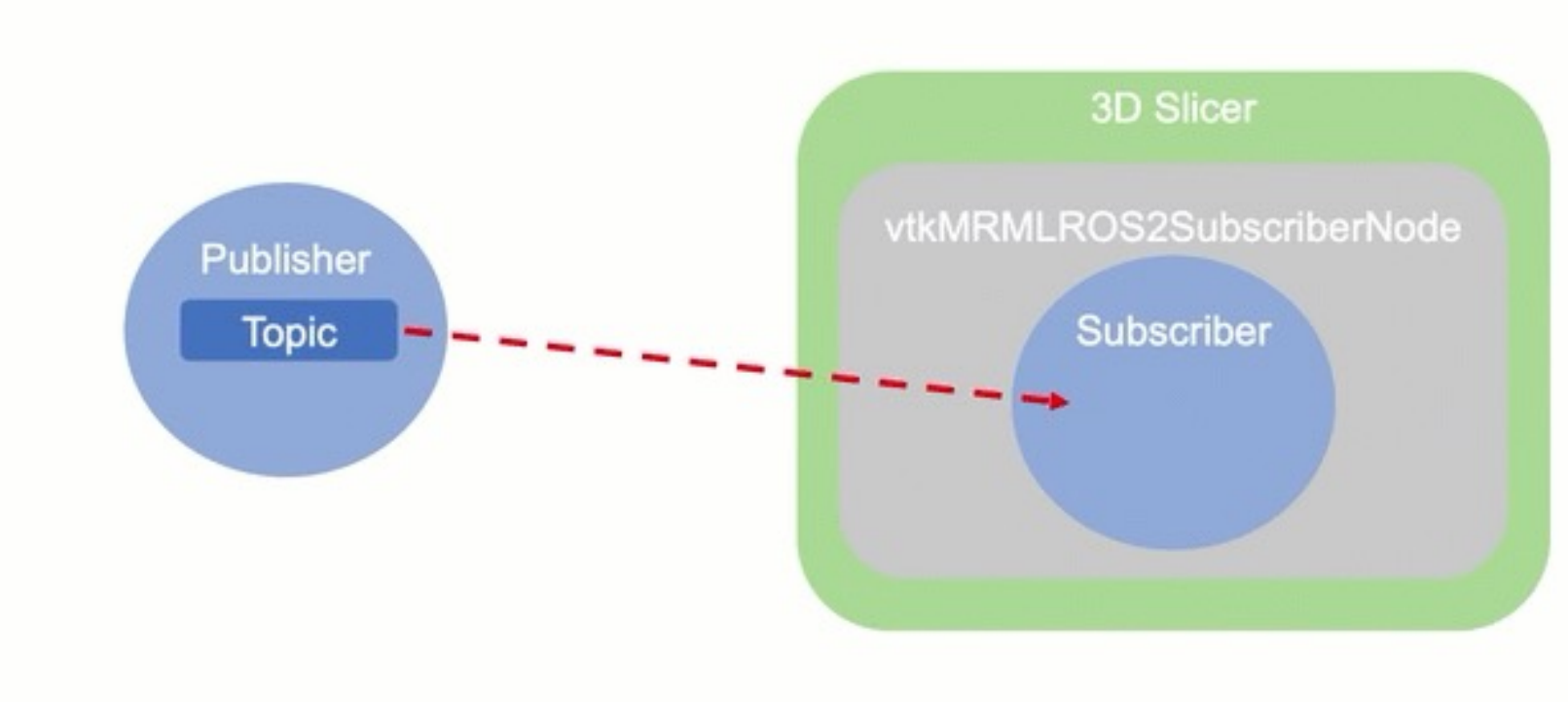
ROS side

3D Slicer side

SlicerROS2 MRML nodes: Overview



SlicerROS2 MRML nodes: Subscribers

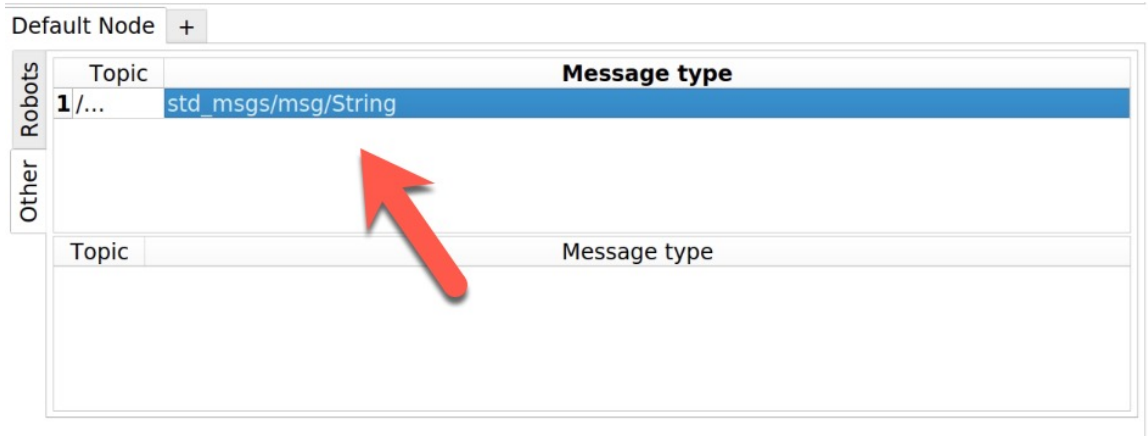


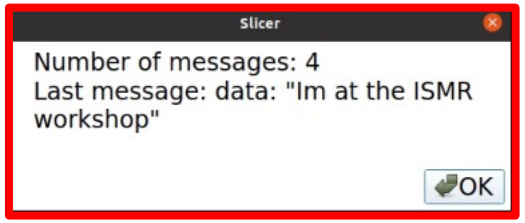
SlicerROS2 MRML nodes: Subscribers

Create your own subscriber

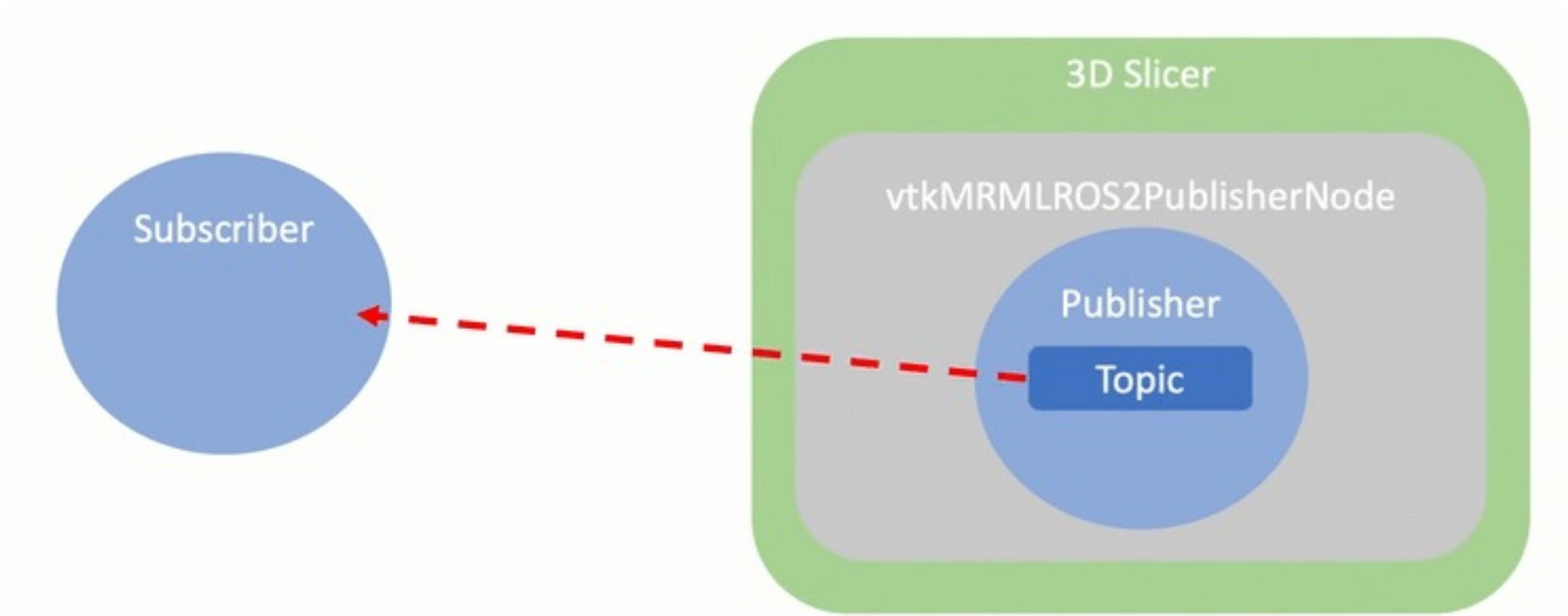
```
rosLogic = slicer.util.getModuleLogic('ROS2')
rosNode = rosLogic.GetDefaultROS2Node()
subString = rosNode.CreateAndAddSubscriberNode('vtkMRMLROS2SubscriberStringNode', '/my_string')
# run `ros2 topic pub /my_string` to send a string
m_string = sub.GetLastMessage()
```

```
ros2 topic pub /my_string std_msgs/msg/String "{data: 'Im at the ISMR workshop'}"
```





SlicerROS2 MRML nodes: Publishers

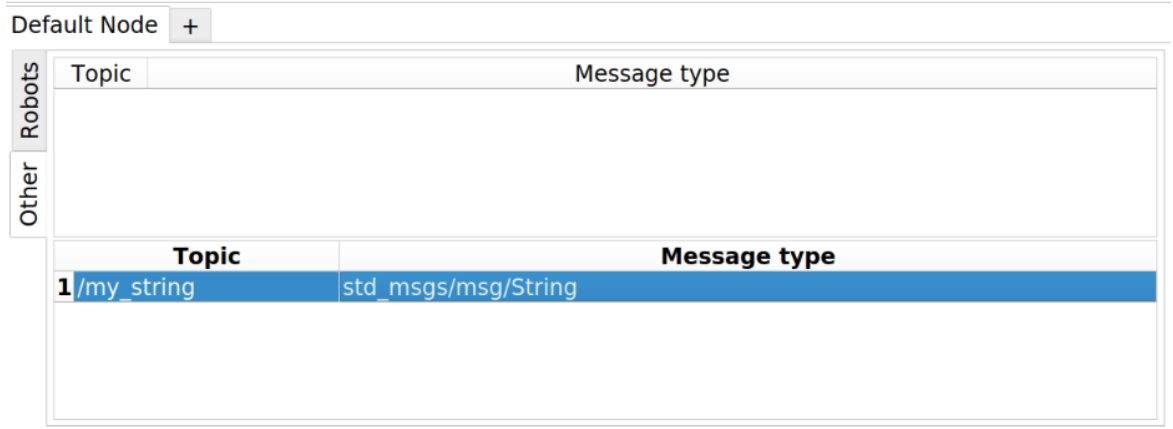


SlicerROS2 MRML nodes: Publishers

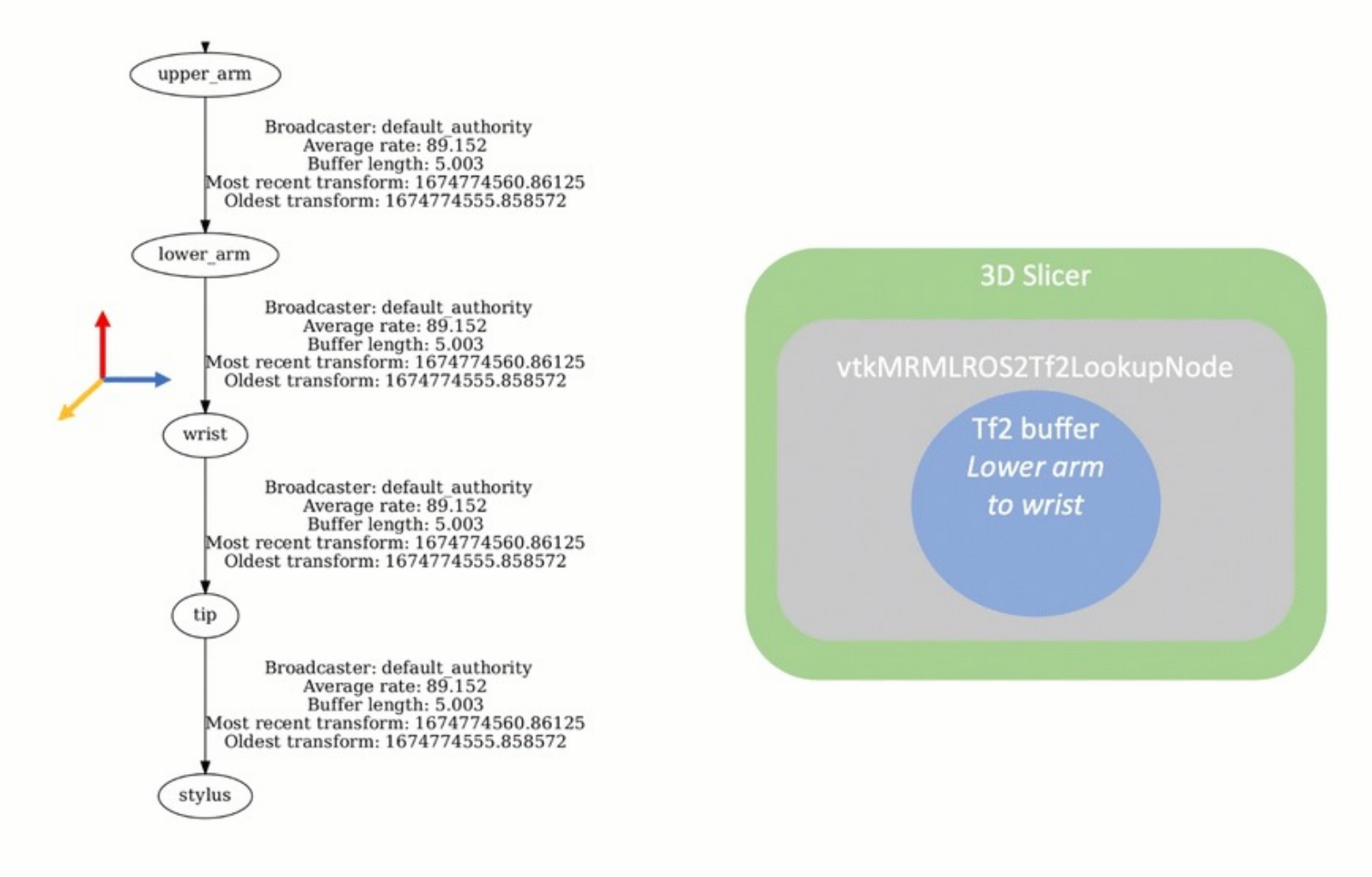
Create your own publisher

```
rosLogic = slicer.util.getModuleLogic('ROS2')
rosNode = rosLogic.GetDefaultROS2Node()
pubString = rosNode.CreateAndAddPublisherNode('vtkMRMLROS2PublisherStringNode', '/my_string')
# run `ros2 topic echo /my_string` in a terminal to see the output
pubString.Publish('my first string')
```

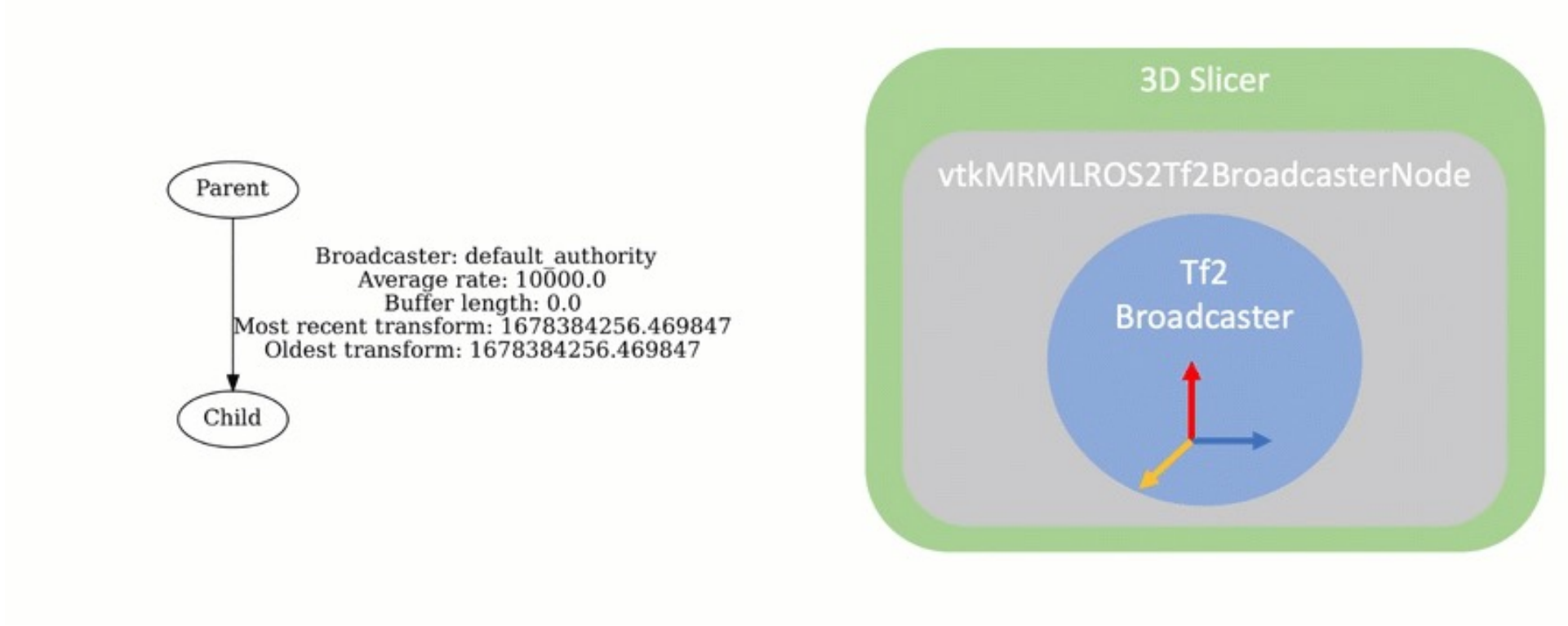
```
laura@appendix:~/dev/dvrk$ ros2 topic echo /my_string
1681784581.195066 [0]       ros2: selected interface "lo" is not multicast-capable: disabli
data: Im at the ISMR workshop
---
```



SlicerROS2 MRML nodes: Tf2 Lookup



SlicerROS2 MRML nodes: Tf2 Broadcaster

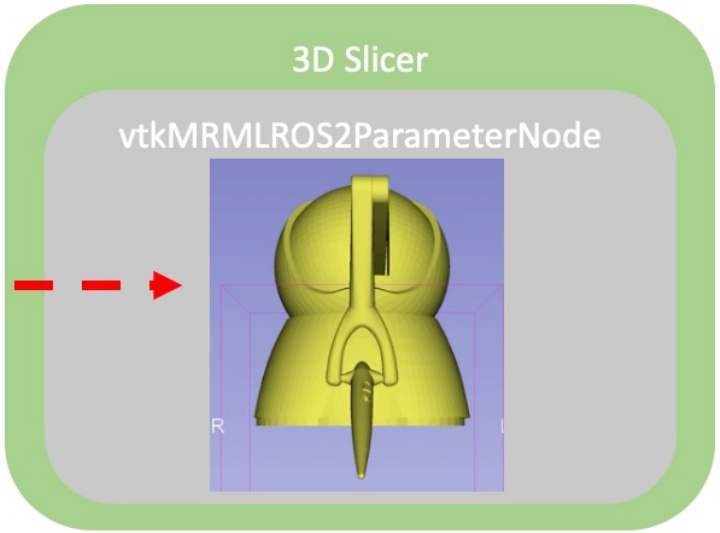


SlicerROS2 MRML nodes: Parameter

```xml
<?xml version="1.0" encoding="utf-8"?>
<robot name="phantom_omni" xmlns:xacro="http://www.ros.org/wiki/xacro">

  <!-- Materials -->
  <material name="metal_seamed">
    <color rgba="0.4627450980392157 0.4666666666666667 0.4509803921568628 1.0" />
  </material>
  <material name="blue">
    <color rgba="0.0 0.0 0.8 1.0" />
  </material>
  <material name="white">
    <color rgba="1.0 1.0 1.0 1.0" />
  </material>

  <!-- Links -->
  <link name="base" >
    <visual>
      <origin xyz="0 -0.02 0" />
      <geometry>
        <mesh filename="package://saw_sensable_phantom_models/meshes/base.stl" />
      </geometry>
      <material name="metal_seamed"/>
    </visual>
  </link>
  <link name="torso" >
    <visual>
      <origin xyz="0 0 0.036" rpy="-1.5707 0 0"/>
      <geometry>
        <mesh filename="package://saw_sensable_phantom_models/meshes/torso.stl" />
      </geometry>
      <material name="blue"/>
    </visual>
  </link>
```



Conversion methods

Slicer type	ROS type	SlicerROS2 "name"
std::string	std_msgs::msg::String	String
bool	std_msgs::msg::Bool	Bool
int	std_msgs::msg::Int64	Int
double	std_msgs::msg::Float64	Double;
vtkIntArray	std_msgs::msg::Int64MultiArray	IntArray
vtkDoubleArray	std_msgs::msg::Float64MultiArray	DoubleArray
vtkTable	std_msgs::msg::Int64MultiArray	IntTable
vtkTable	std_msgs::msg::Float64MultiArray	DoubleTable
vtkMatrix4x4	geometry_msgs::msg::PoseStamped	PoseStamped
vtkDoubleArray	geometry_msgs::msg::WrenchStamped	WrenchStamped
vtkTransformCollection	geometry_msgs::msg::PoseArray	PoseArray

We use this naming convention for publishers/ subscribers:
vtkMRMLROS2PublisherxxxxNode
vtkMRMLROS2SubscriberxxxxNode

How to add your own custom publishers / subscribers:

Select a ROS type and 3D Slicer counterpart (vtk / or native C++ types) and write your conversion function:

SlicerToROS2 for publishers: vs ROS2ToSlicer for subscribers

```
void vtkSlicerToROS2(vtkMatrix4x4 * input, cisst_msgs::msg::CartesianImpedanceGains & result,
                     const std::shared_ptr<rclcpp::Node> &) // the input should be something related to the closest point on the volume
{
  // stiffness = elasticity
  // damping = viscosity
  result.pos_stiff_neg.x = 0.0;
  result.pos_stiff_pos.x = 0.0;
  result.pos_damping_neg.x = 0.0;
  result.pos_damping_pos.x = 0.0;
  result.pos_stiff_neg.y = 0.0;
  result.pos_stiff_pos.y = 0.0;
  result.pos_damping_neg.y = 0.0;
  result.pos_damping_pos.y = 0.0;
  // These gains are preconfigured for our application - they can be modified according to your device/ specific needs
  result.pos_stiff_neg.z = -10.0;
  result.pos_stiff_pos.z = 10.0;
  result.pos_damping_neg.z = -20.0;
  result.pos_damping_pos.z = 20.0;
```

Initialize the new macros and register your new MRML node class:

```
VTK_MRML_ROS_PUBLISHER_VTK_H(vtkMatrix4x4, CartesianImpedanceGains);
```

```
VTK_MRML_ROS_PUBLISHER_VTK_CXX(vtkMatrix4x4, cisst_msgs::msg::CartesianImpedanceGains, CartesianImpedanceGains);
```

```
this->GetMRMLScene()->RegisterNodeClass(vtkSmartPointer<vtkMRMLROS2PublisherCartesianImpedanceGainsNode>::New());
```

Example application: Virtual fixture

Initialize MRML objects:

```
self.sub = self.ros2Node.CreateAndAddSubscriberNode("vtkMRMLROS2SubscriberPoseStampedNode", "/arm/measured_cp")
self.pub = self.ros2Node.CreateAndAddPublisherNode("vtkMRMLROS2PublisherPoseStampedNode", "/arm/servo_cp")
self.wrenchpub = self.ros2Node.CreateAndAddPublisherNode("vtkMRMLROS2PublisherWrenchStampedNode",
                                                         "/arm/servo_cf")
```

Add observer:

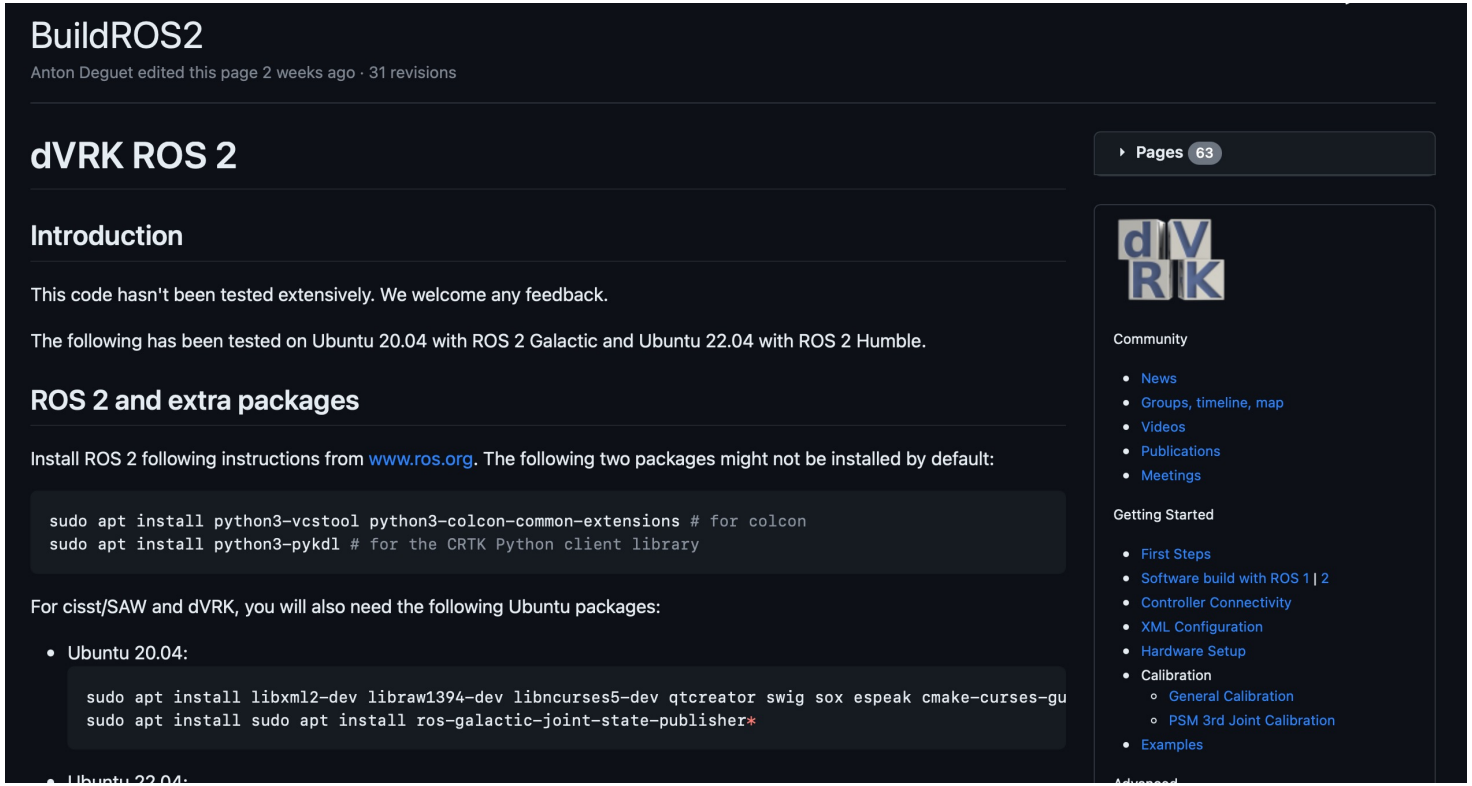
```
self.addObserver(self.logic.breachWarningNode, vtk.vtkCommand.ModifiedEvent, self.logic.breachDetection)
```

```
if (self.breachWarningNode.IsToolTipInsideModel()):
    pose = vtk.vtkMatrix4x4()
    self.sub.GetLastMessage(pose)
    self.pub.Publish(pose)
else:
    darr = vtk.vtkDoubleArray()
    self.wrenchpub.Publish(darr)
```

Setup the ROS2 dVRK package

Instructions found here:

https://github.com/jhu-dvrk/sawIntuitiveResearchKit/wiki/BuildROS2



BuildROS2

Anton Deguet edited this page 2 weeks ago · 31 revisions

dVRK ROS 2

Introduction

This code hasn't been tested extensively. We welcome any feedback.

The following has been tested on Ubuntu 20.04 with ROS 2 Galactic and Ubuntu 22.04 with ROS 2 Humble.

ROS 2 and extra packages

Install ROS 2 following instructions from www.ros.org. The following two packages might not be installed by default:

```
sudo apt install python3-vcstool python3-colcon-common-extensions # for colcon
sudo apt install python3-pykdl # for the CRTK Python client library
```

For cisst/SAW and dVRK, you will also need the following Ubuntu packages:

- Ubuntu 20.04:

```
sudo apt install libxml2-dev libraw1394-dev libncurses5-dev qtcreator swig sox espeak cmake-curses-gu
sudo apt install sudo apt install ros-galactic-joint-state-publisher*
```

- Ubuntu 22.04:

Pages 63

Community

- News
- Groups, timeline, map
- Videos
- Publications
- Meetings

Getting Started

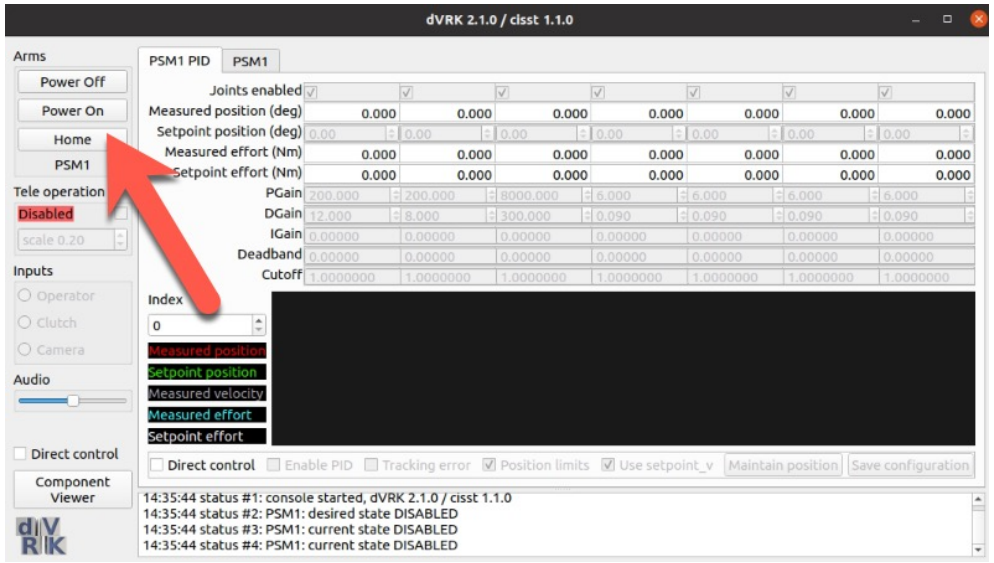
- First Steps
- Software build with ROS 1 | 2
- Controller Connectivity
- XML Configuration
- Hardware Setup
- Calibration
  - General Calibration
  - PSM 3rd Joint Calibration
- Examples

Visualize the dVRK in 3D Slicer

1. Launch the virtual PSM

```
source ~/ros2_ws/install/setup.bash
cd ~/ros2_ws/install/sawIntuitiveResearchKitAll/share/sawIntuitiveResearchKit
ros2 run dvrk_robot dvrk_console_json -j share/console/console-PSM1_KIN_SIMULATED.json
```

2. Select “Power on”



Launch 3D Slicer

3. Start your robot state publisher:

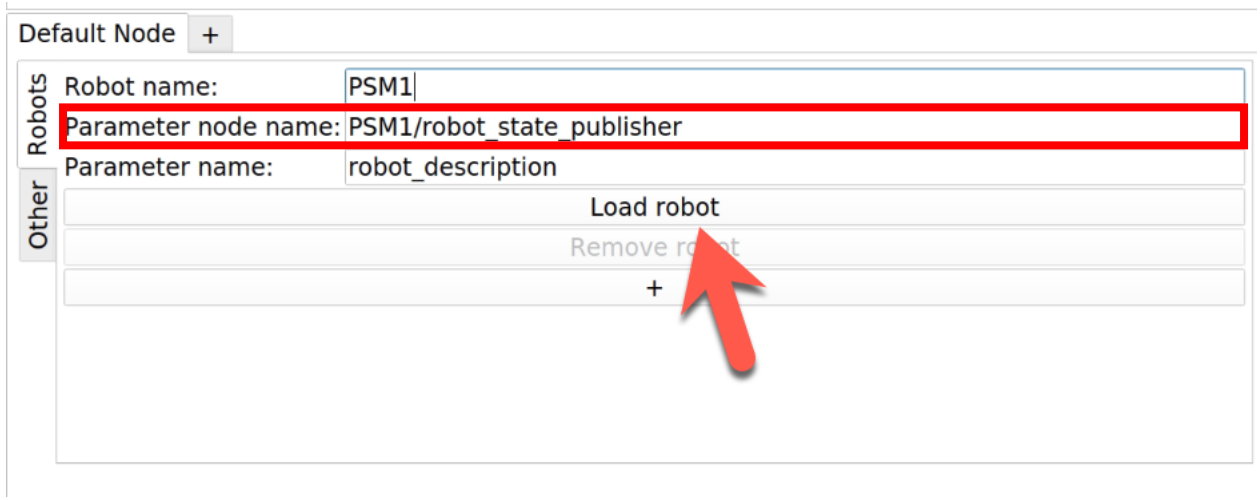
```
source ~/ros2_ws/install/setup.bash
ros2 launch dvrk_model dvrk_state_publisher.launch.py arm:=PSM1
```

4. Launch 3D Slicer from a terminal:

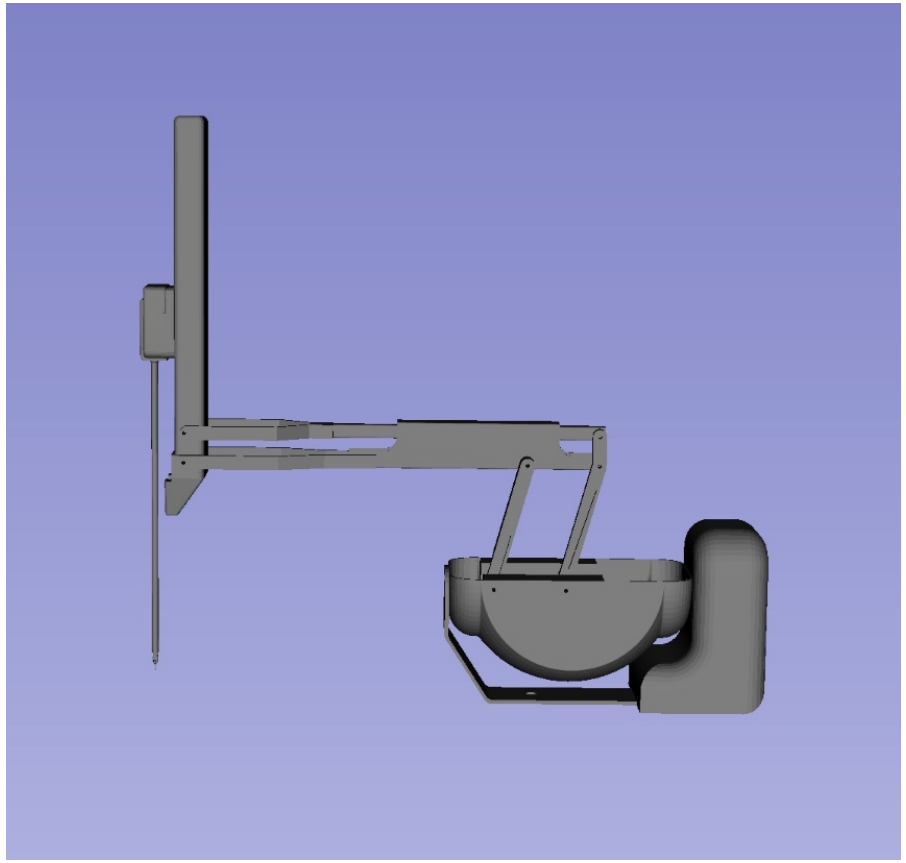
```
laura@appendix:~/dev/Slicer-SuperBuild-Release$ source /opt/ros/galactic/setup.bash
laura@appendix:~/dev/Slicer-SuperBuild-Release$ source ~/dev/ros2_ws/install/setup.bash
laura@appendix:~/dev/Slicer-SuperBuild-Release$ ./Slicer
```

Visualize the robot

Set up the names correctly:



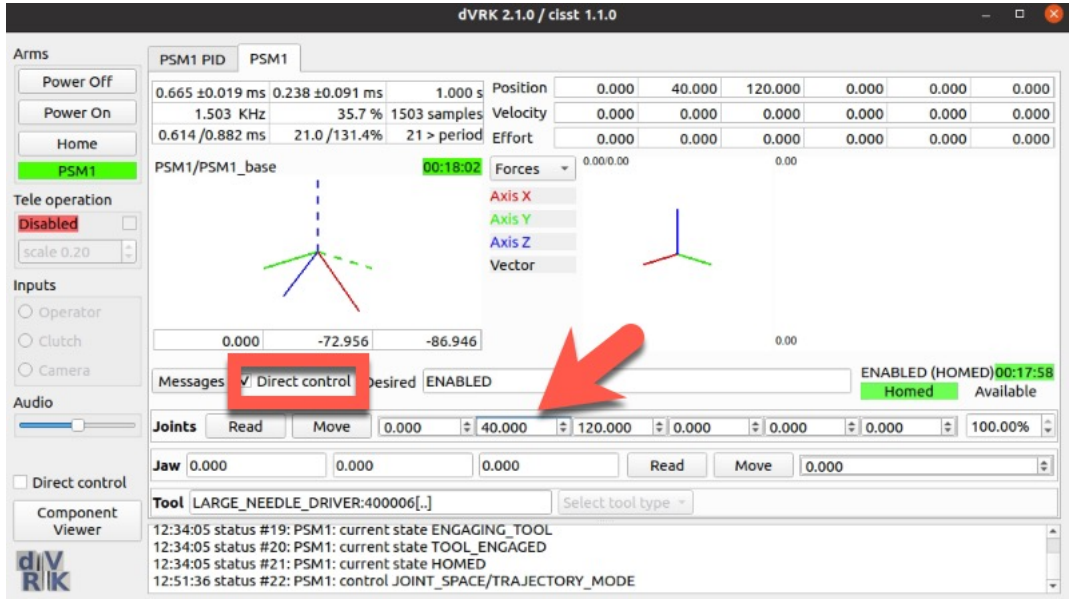
You should see this in the 3D viewer:



You may need to zoom out to see this! Scroll towards you on your mouse to do so

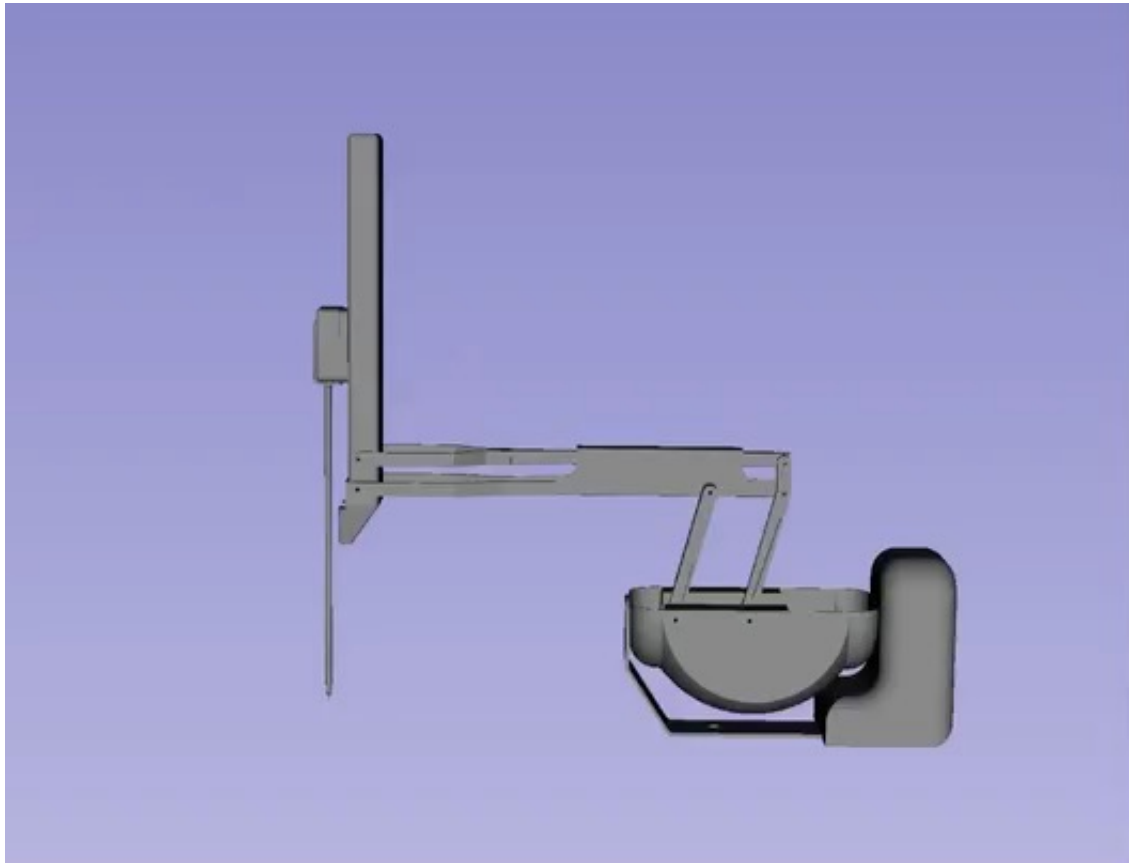
Move the robot

Direct control:



Or you can use this script:

```
source ~/ros2_ws/install/setup.bash
ros2 run dvrk_python dvrk_arm_test.py -a PSM1
```

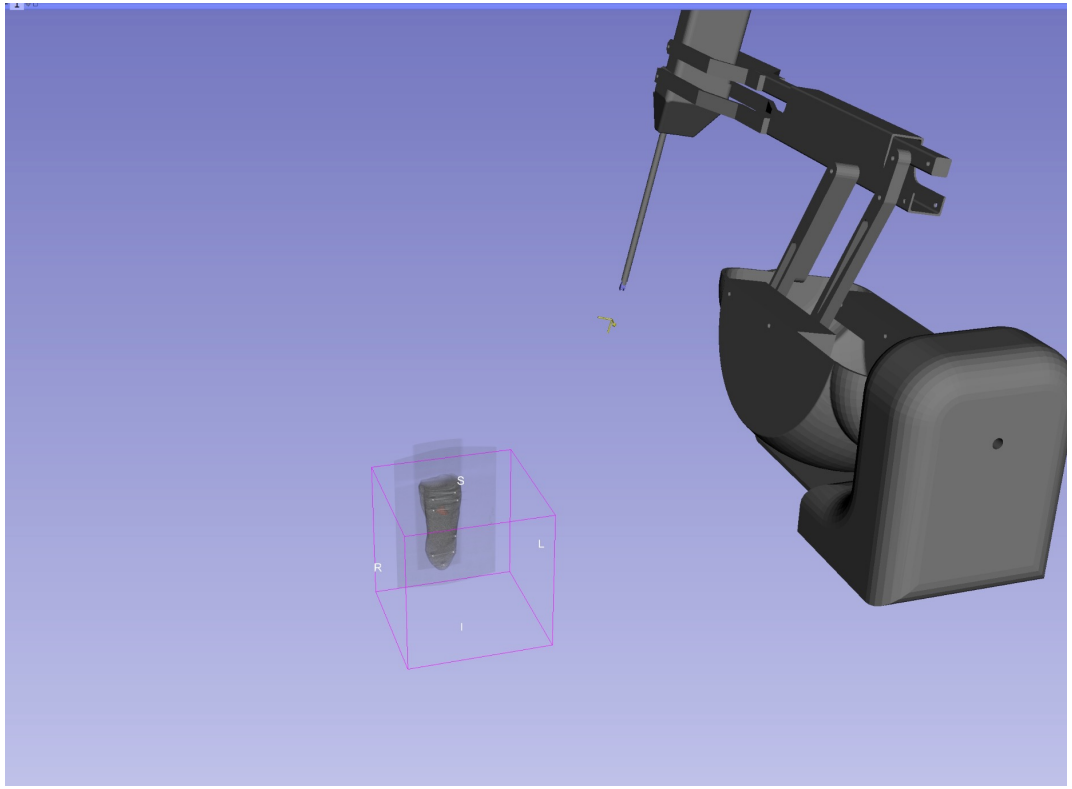


Part 3: Image registration and path planning in Slicer

Load the scene into 3D Slicer

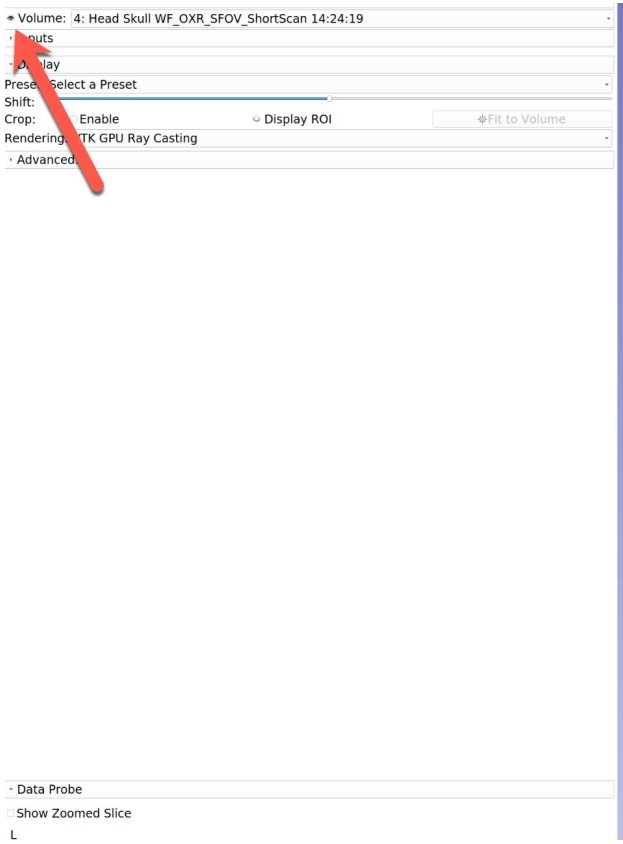
1. Drag and drop the scene called “RegisteredScene-dVRKToTongue.mrb” into 3D Slicer

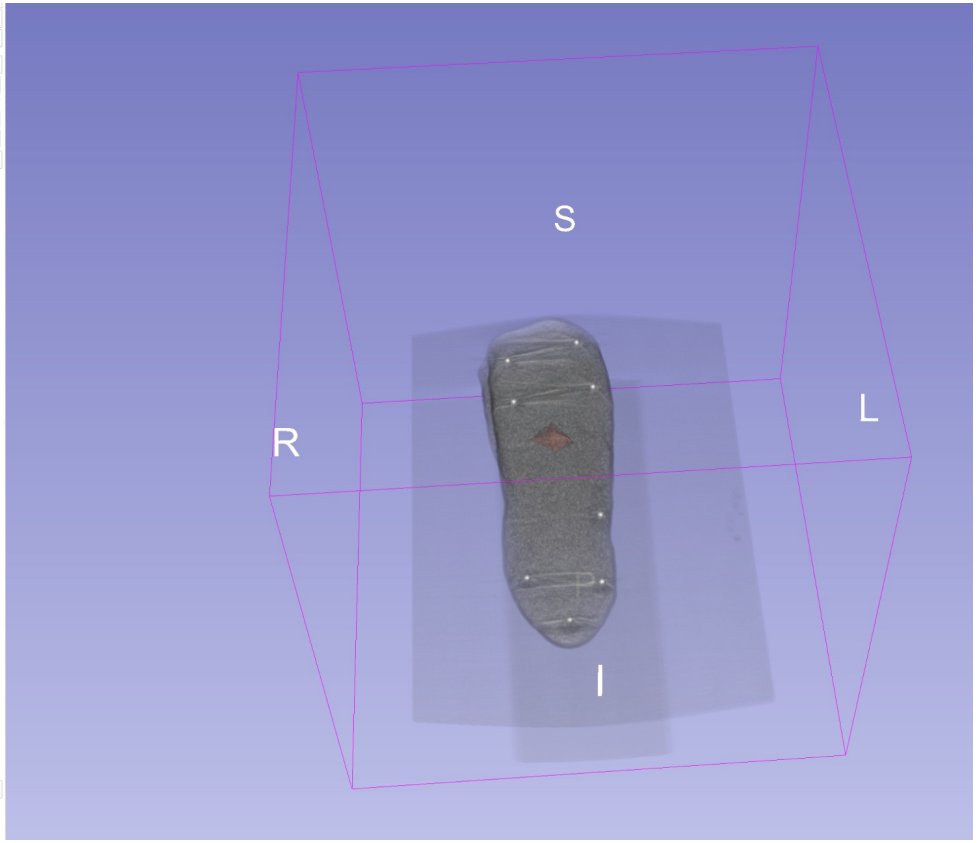
You should see something like this:



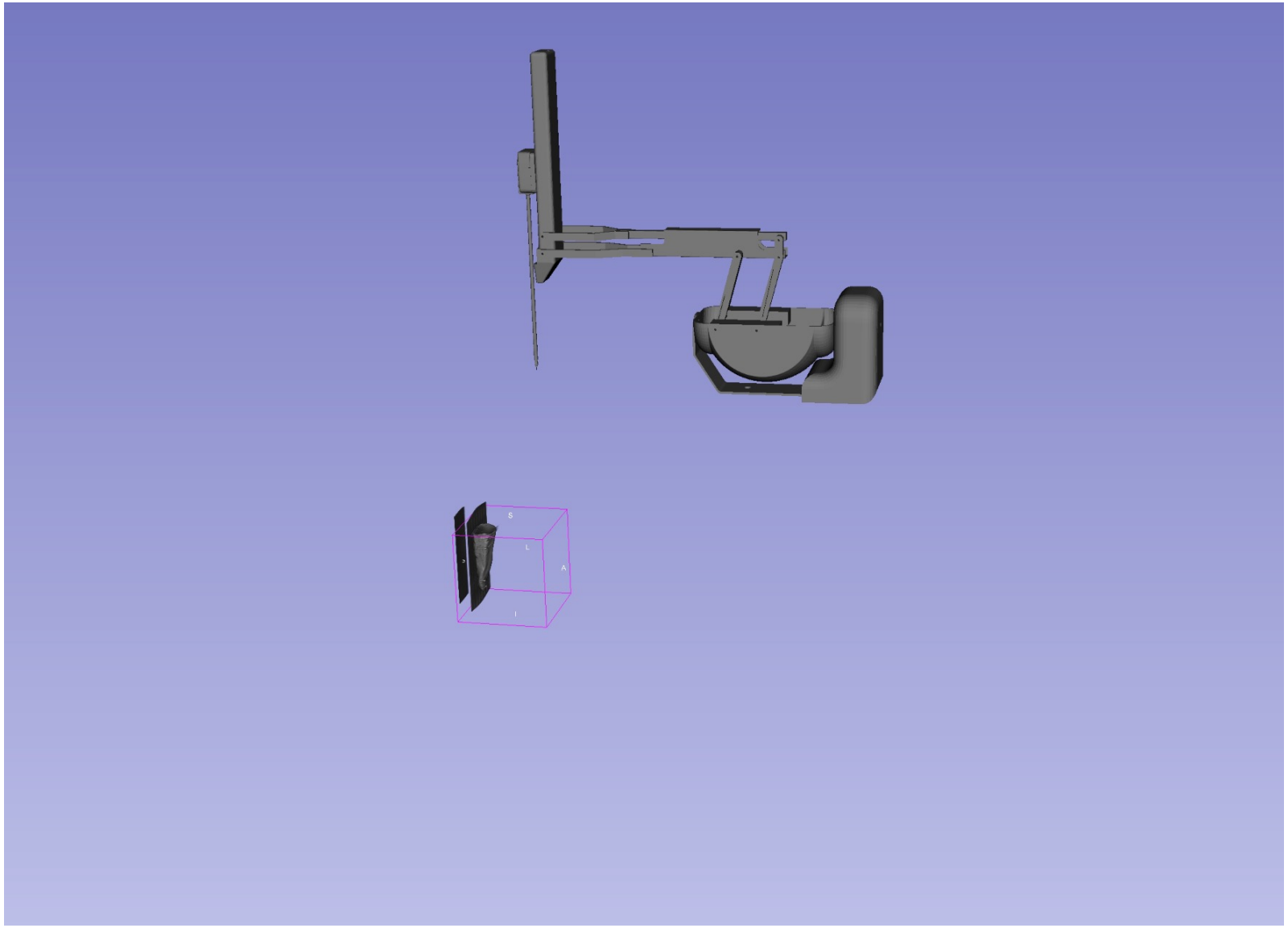
Get Familiar with the scene

2. Switch to the *Volume Rendering* module and press the eyeball on the top left to hide and show the tumor in the tongue



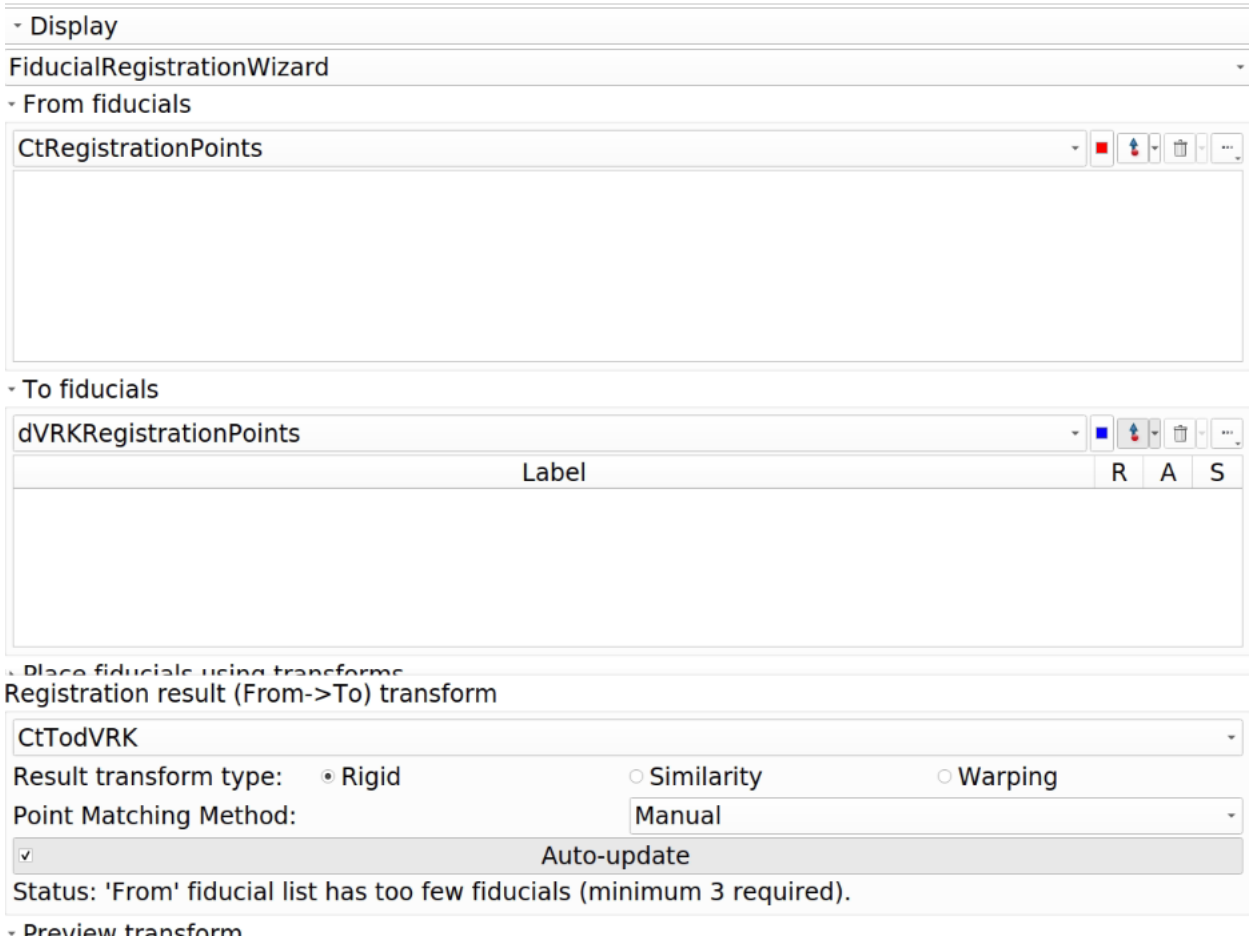


The image and the robot are not in the same coordinate system yet



***We've already done this for you*

3. Switch to the *Fiducial Registration Wizard* module and create:

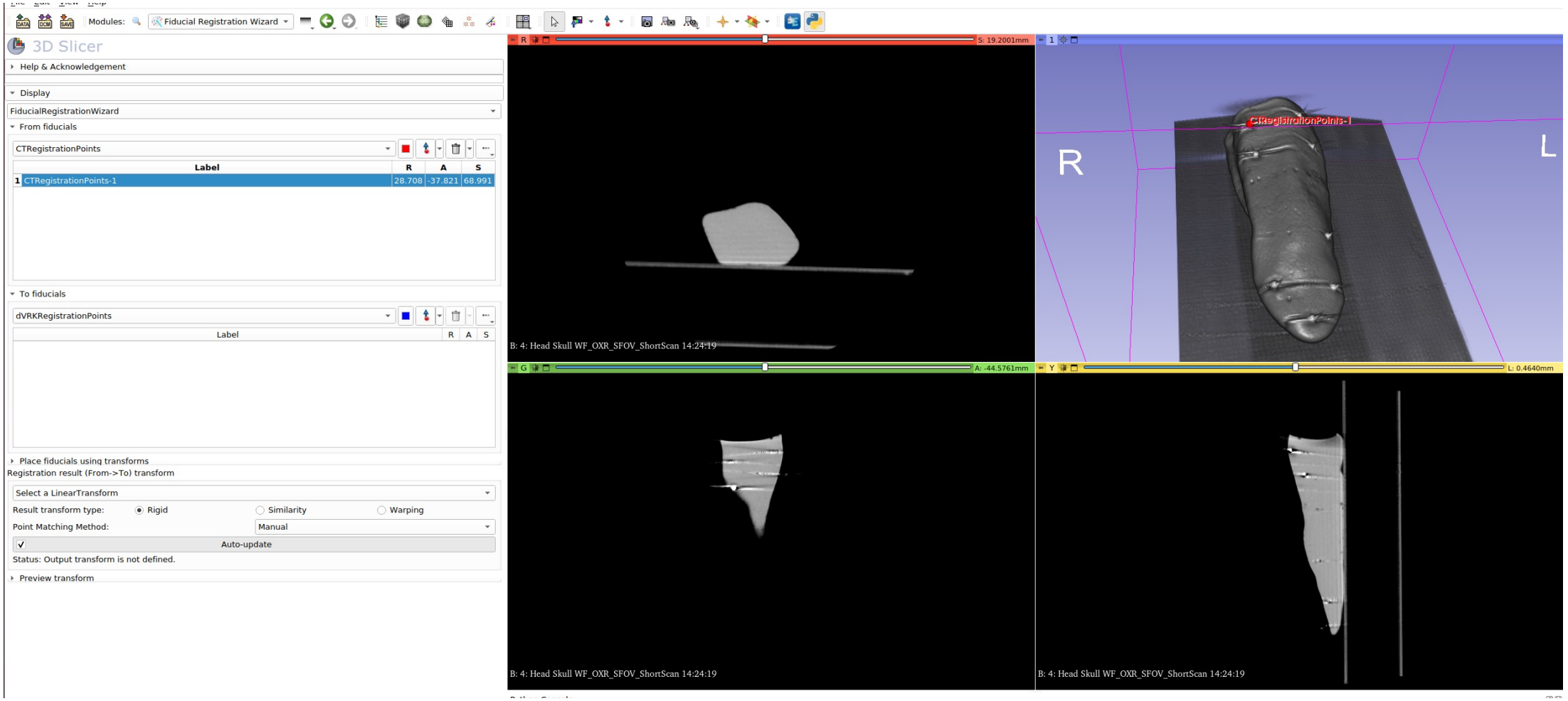


CT registration points

dVRK registration points

CtTodVRK

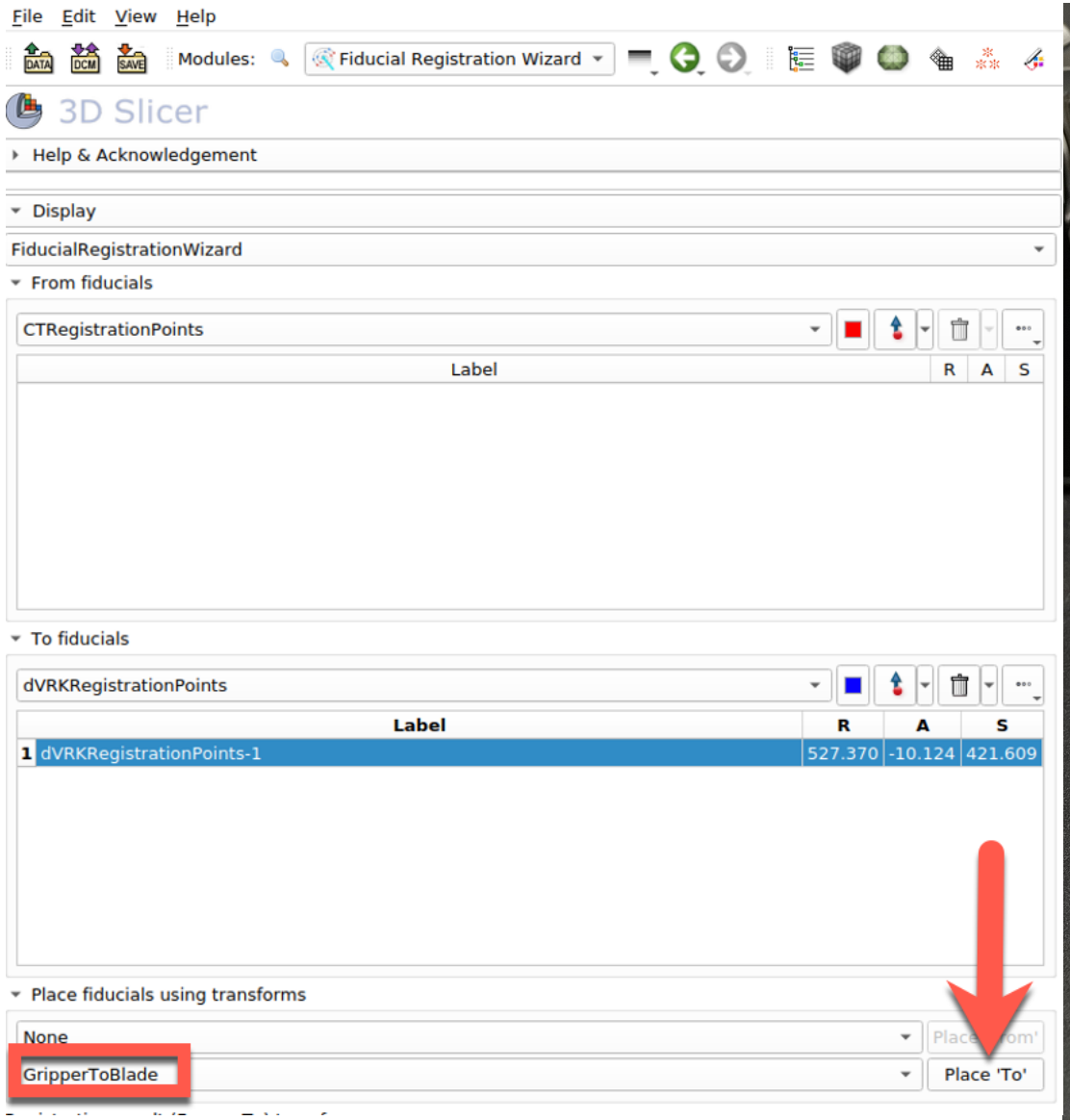
Place a fiducial on the CT bead in the volume rendering using the 3D viewer

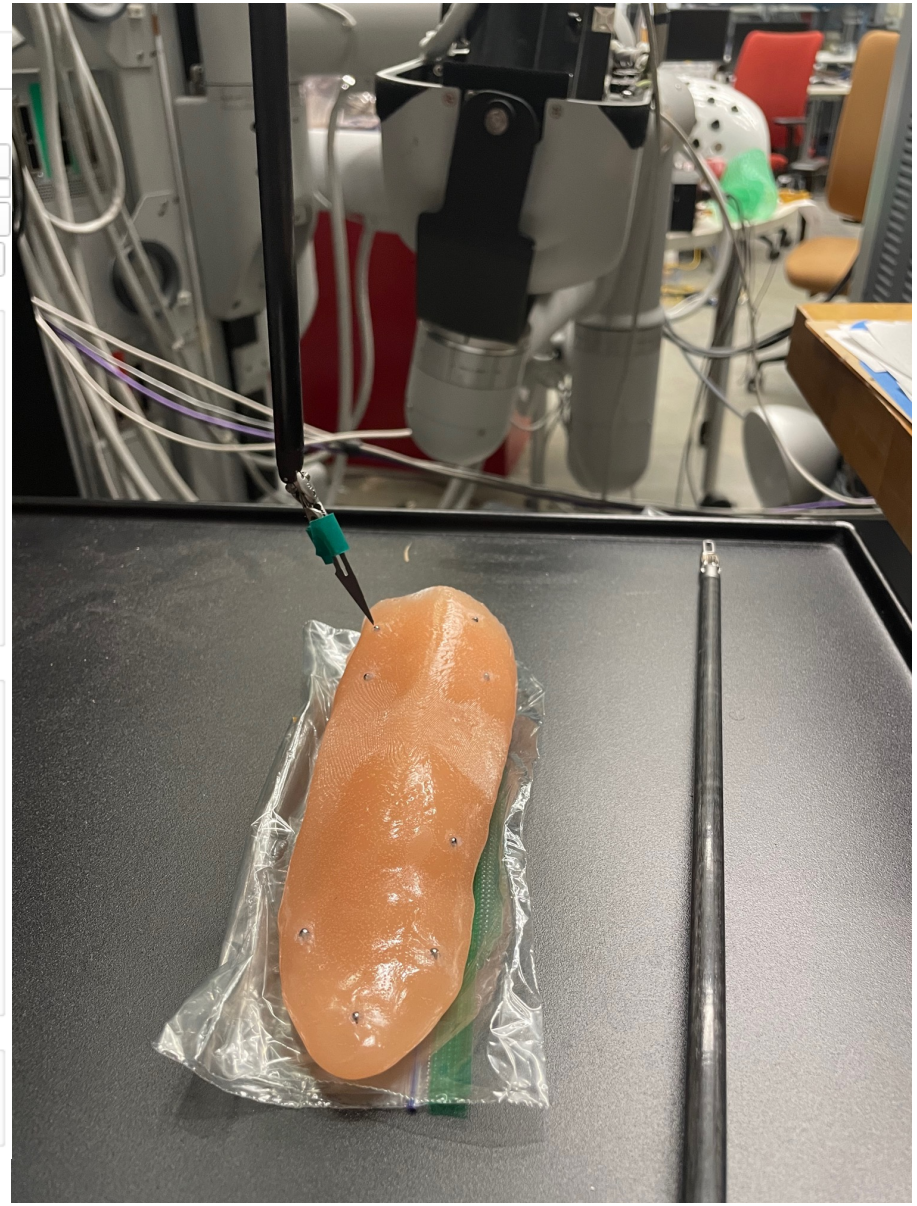


GripperToBlade is a transform that we append to the tree to access the position of the tip of the blade

Move the robot to the same point and press:

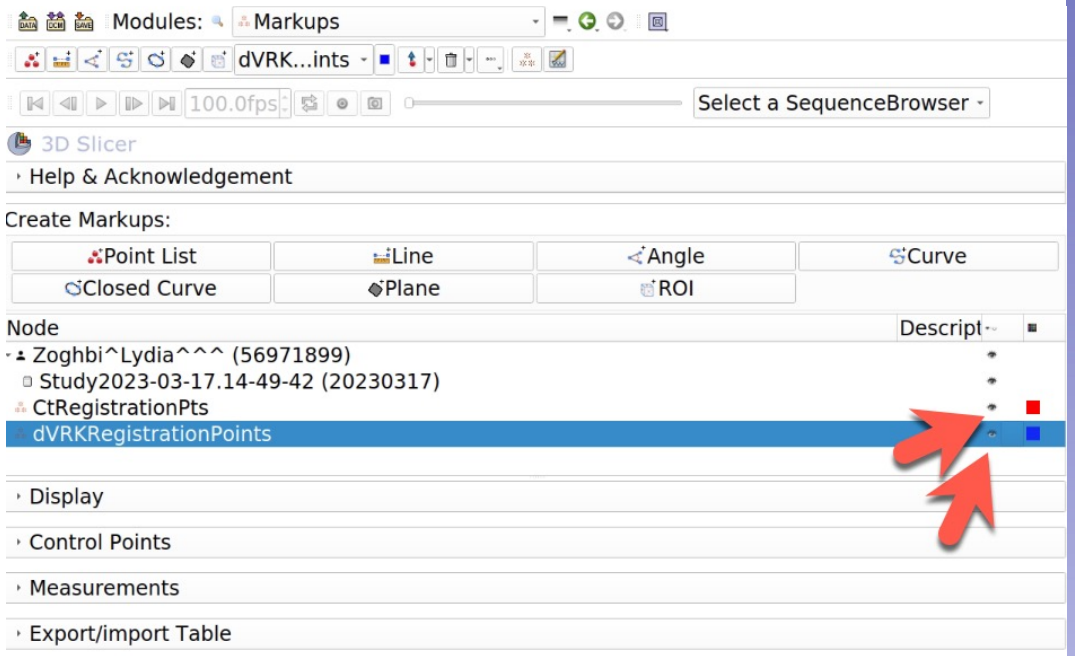
Place ‘To’

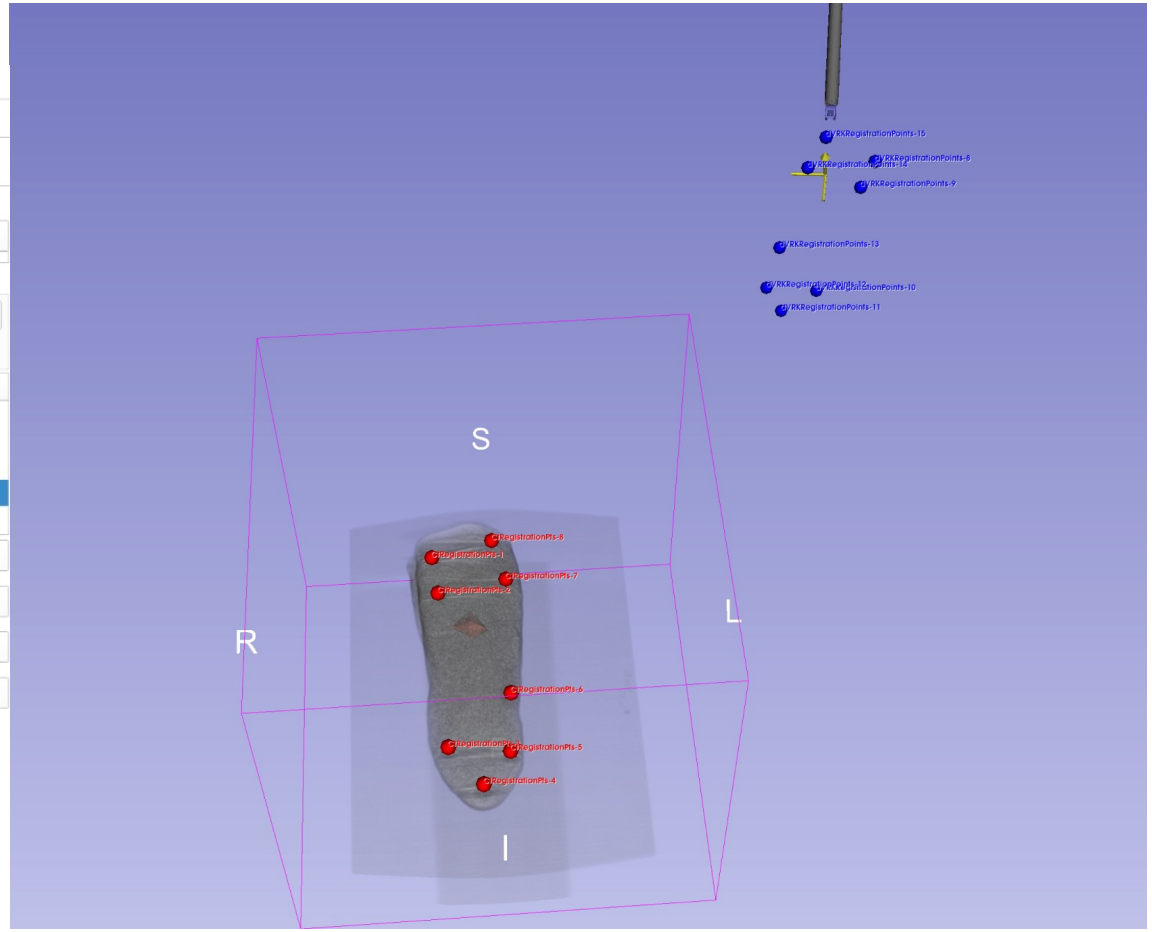




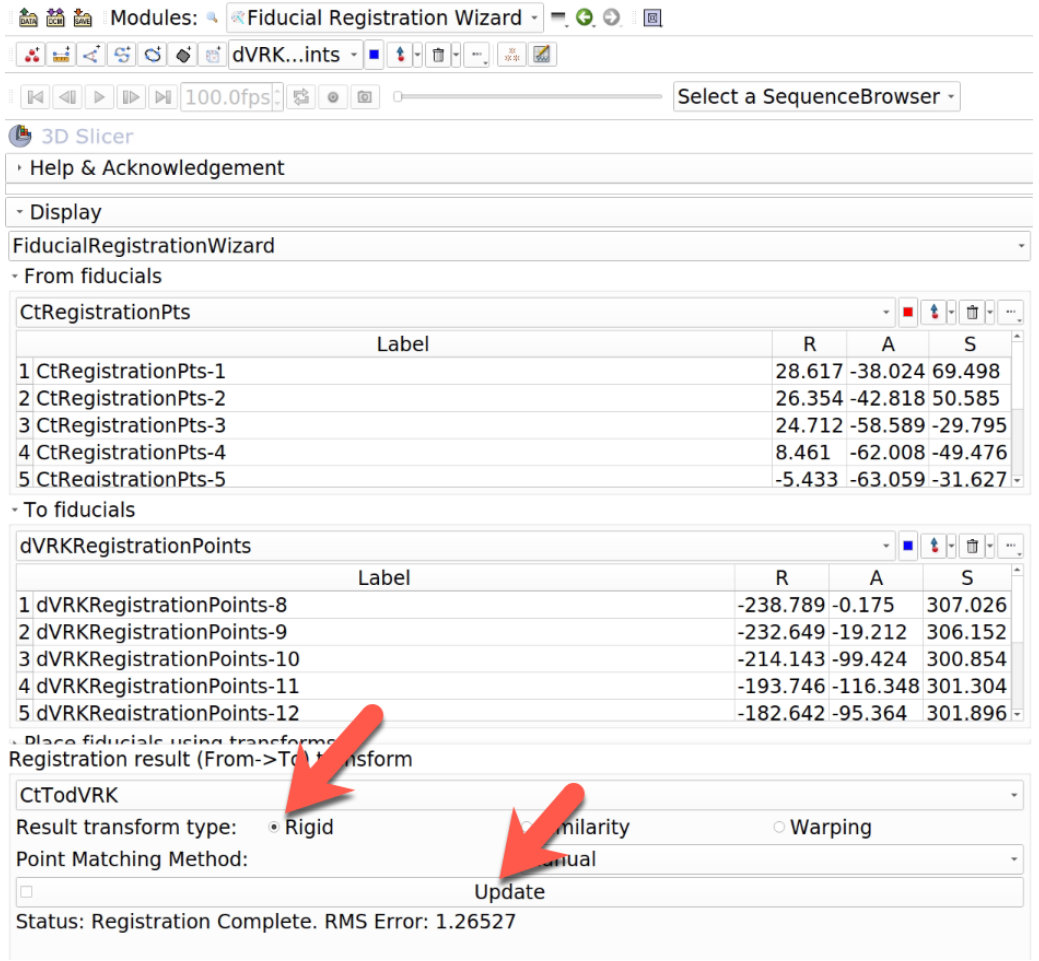
Repeat this for all the points! Make sure the order of points matches

4. Switch to the *Markups* module to visualize these points

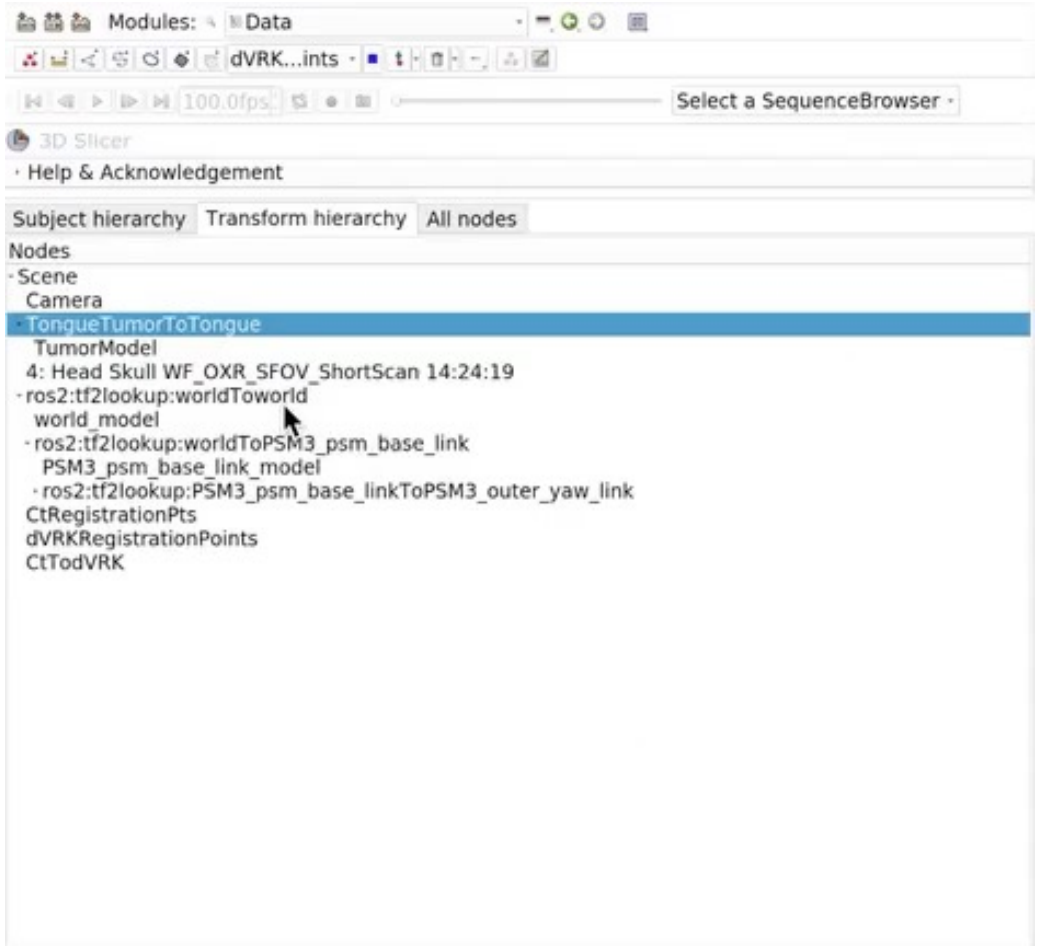




5. Switch back to *Fiducial Registration Wizard* and update the transform



6. Switch to the *Data* module and move the Ct image and the tumor under this new transformation



Now we’re registered!