NA-MIC
National Alliance for Medical Image Computing
http://www.na-mic.org

MatlabBridge Extension

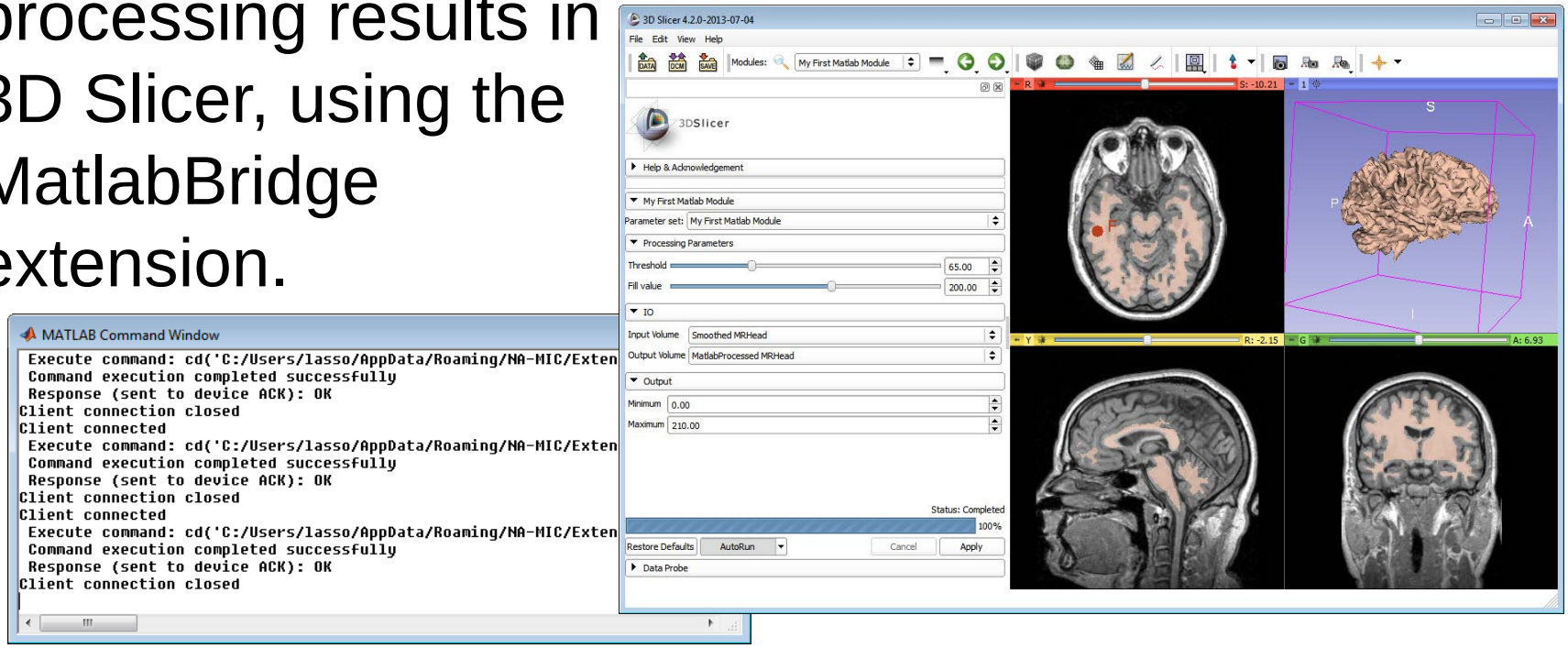
Andras Lasso[1], Jean-Christophe Fillion-Robin[2], Kevin Wang[3], Gabor Fichtinger[1]

[1]Laboratory for Percutaneous Surgery, Queen's University, Canada
[2]Kitware
[3]University Health Network, Toronto, ON, Canada

lasso@cs.queensu.ca

Learning Objective

This tutorial demonstrates how 3D Slicer can be used to run Matlab functions and visualize the processing results in 3D Slicer, using the MatlabBridge extension.

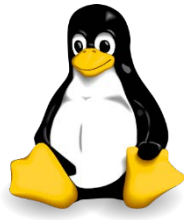


Pre-requisites

- Matlab installed
- A **nightly release** of 3D Slicer installed.
  Download the 3D Slicer installer package from the *nightly build* row at:
  http://www.slicer.org/pages/Downloads/

Platforms

- Windows: fully supported



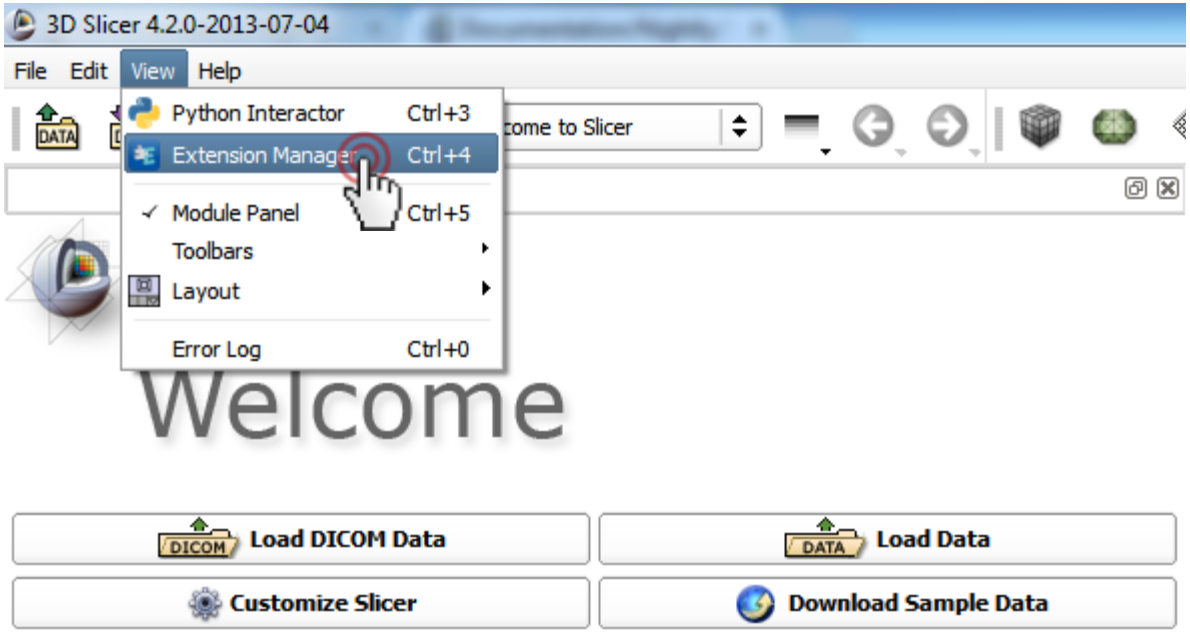
- Linux: not tested extensively, check this page for the current status

- Mac: not tested extensively, check this page for the current status

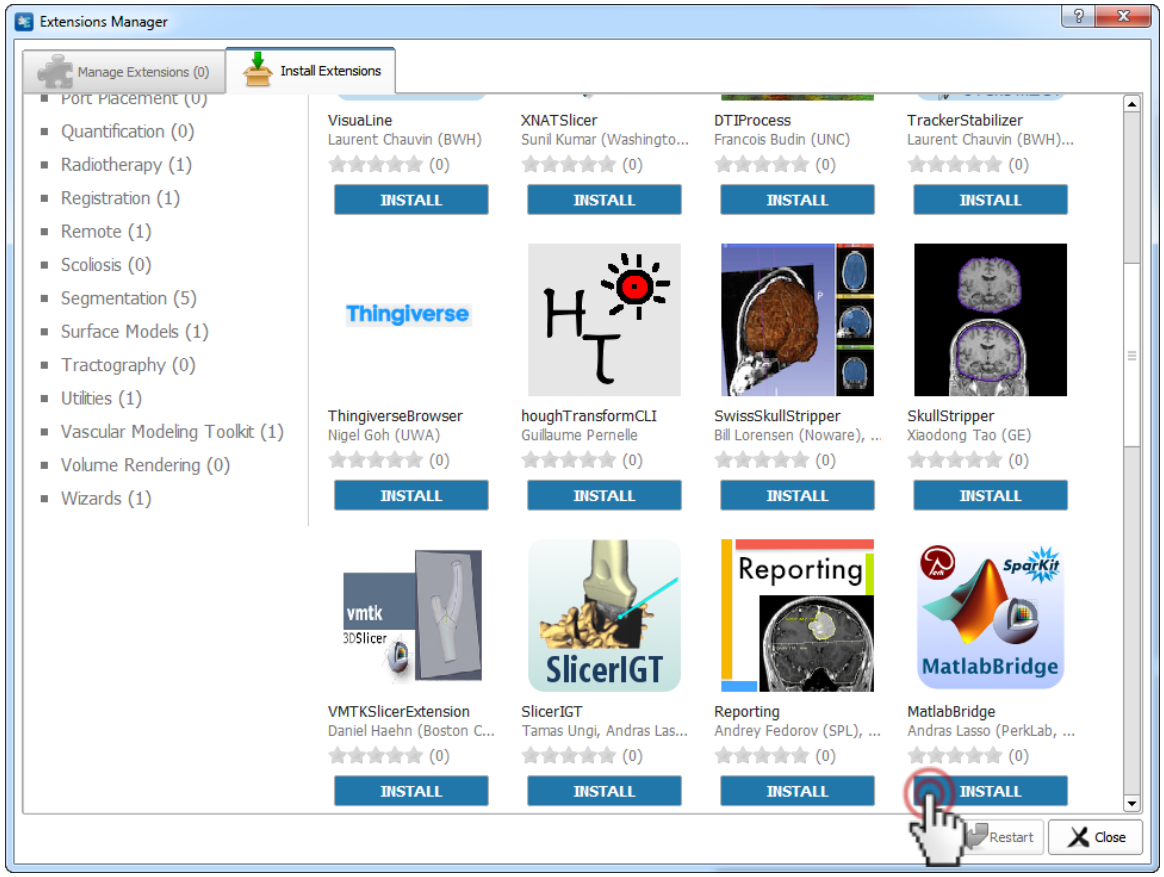
Overview

1. Install and configure
2. Create a simple Matlab module
3. Use your Matlab module
4. Customize your Matlab module

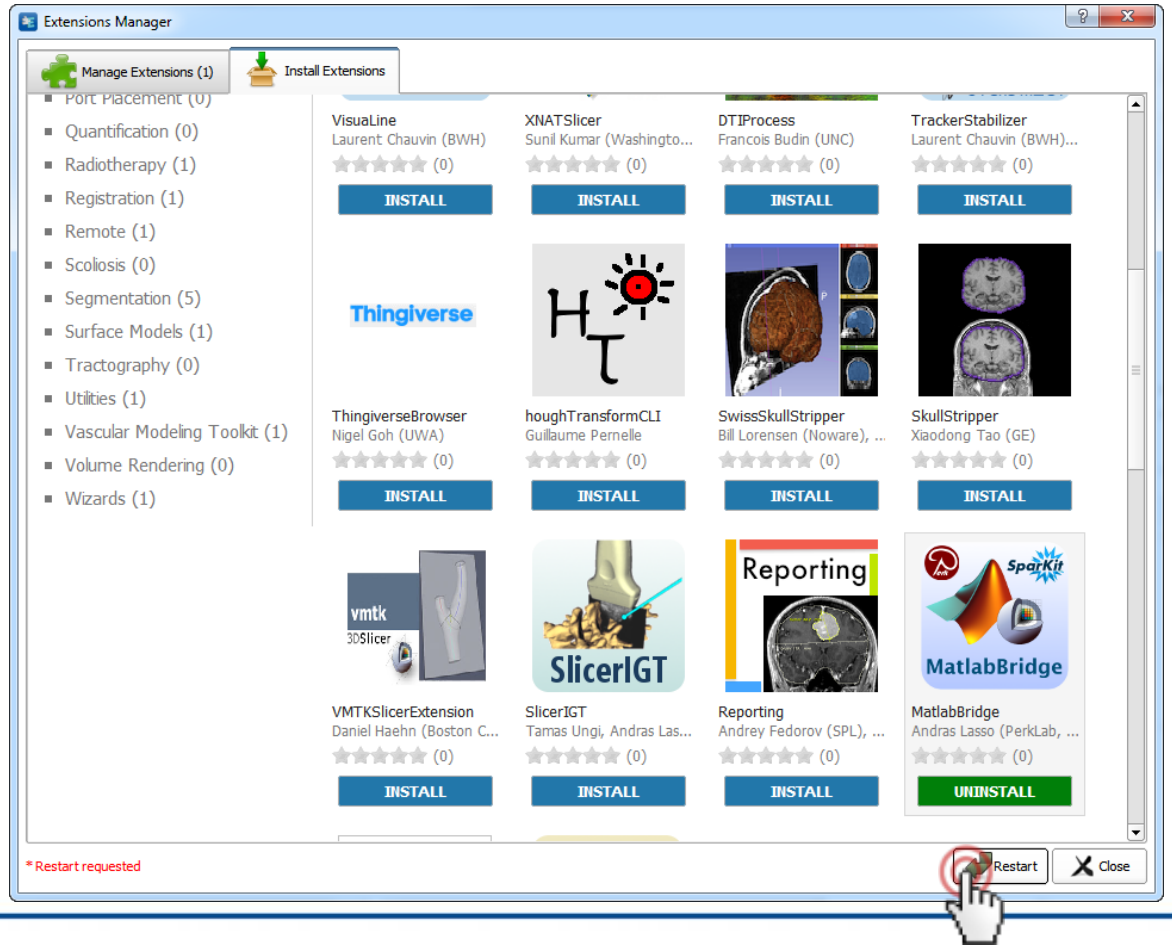
1.1 Install and configure



1.2 Install and configure



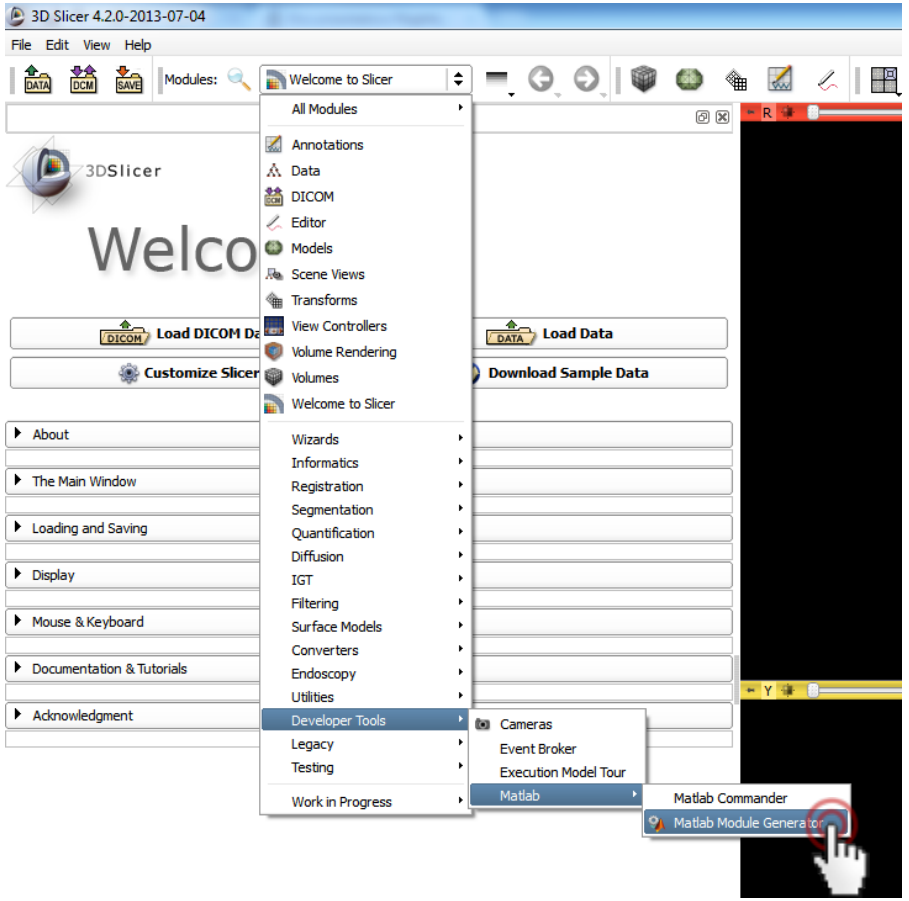
1.3 Install and configure



1.4 Install and configure

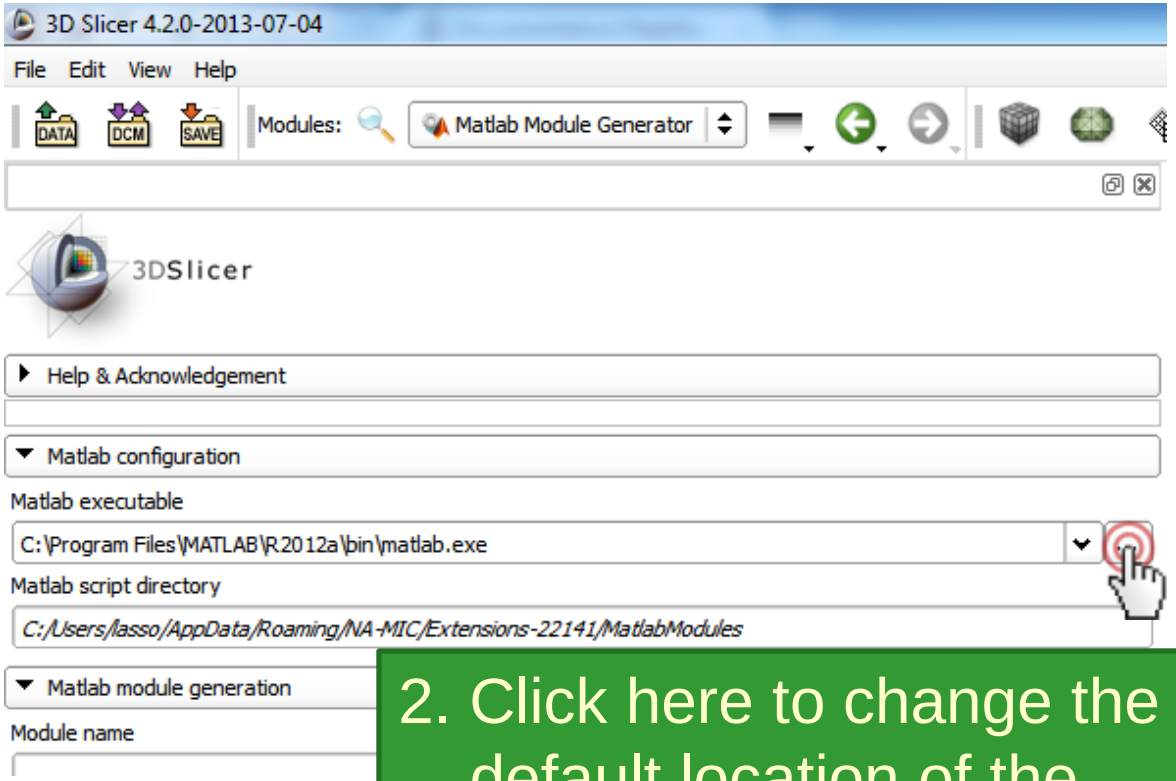
Note: MatlabBridge modules appear under *Developer Tools / Matlab* category in the module list

Start the *Matlab Module Generator* module



1.5 Install and configure

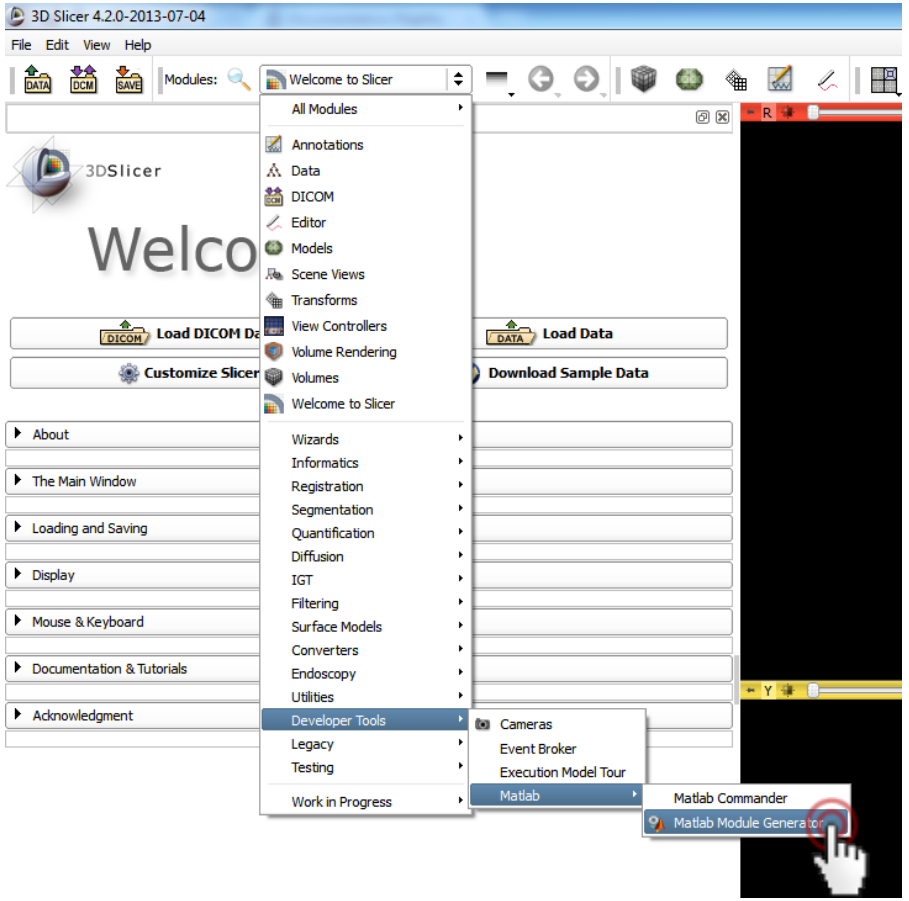
1. Verify that the path to Matlab is correct



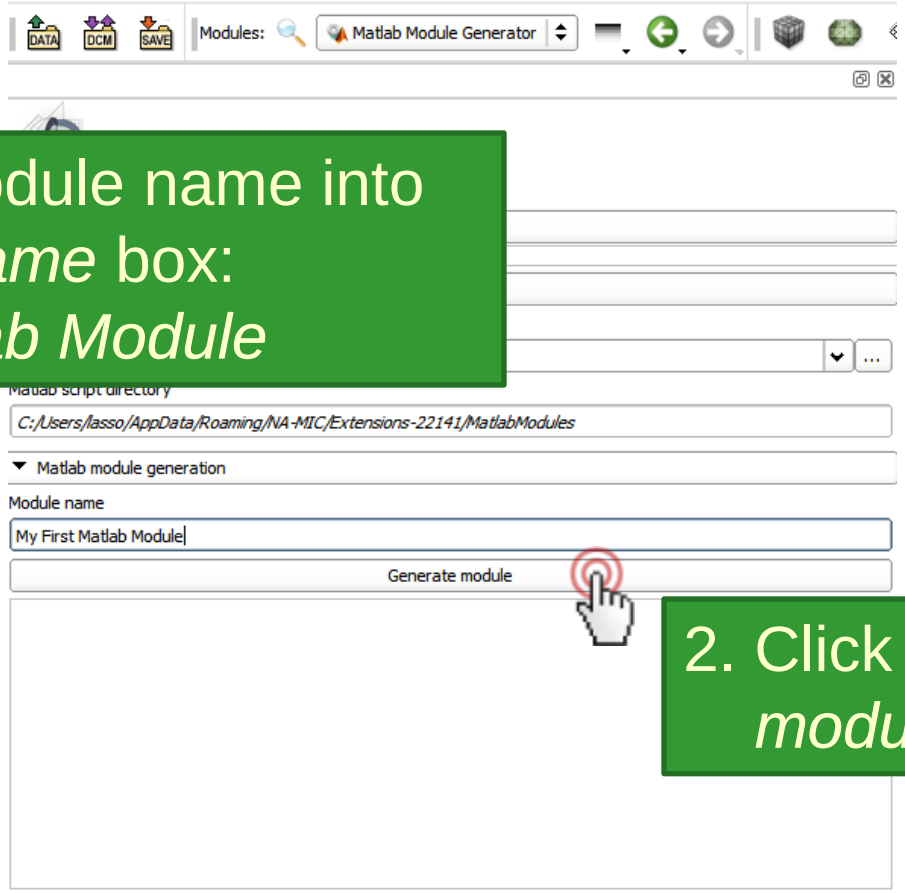
2. Click here to change the default location of the Matlab executable

2.1 Create a Matlab module

Start the *Matlab Module Generator* module



2.2 Create a Matlab module

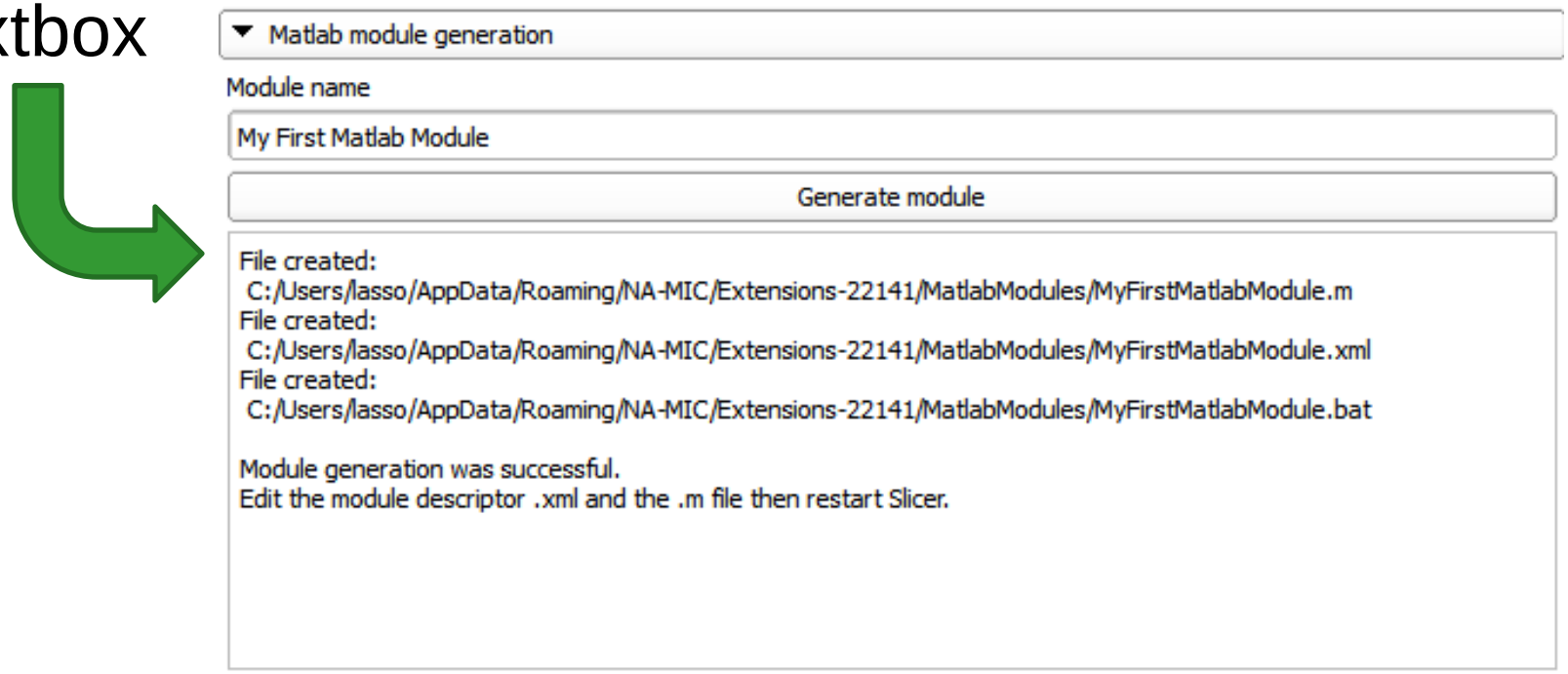


1. Enter your module name into the *Module name* box: *My First Matlab Module*

2. Click *Generate module* button

2.3 Create a Matlab module

Note: status of the module generation and the file path of the generated files are shown in the textbox



2.4 Create a Matlab module

Each Matlab module consists of three files:

- **Matlab script** (*.m): the Matlab function that Slicer calls, *this file has to be customized* by the user to perform all the necessary data input, processing, and output
- **Module descriptor** (*.xml): this XML file defines the graphical user interface that will be displayed for the module in 3D Slicer, *this file has to be customized* for the specific Matlab function
- **Module proxy** (*.bat): this file is generated once and need not to be changed

2.5 Create a Matlab module

Generated module descriptor: *MyFirstMatlabModule.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<executable> <category>Matlab</category> <title>My First Matlab Module</title> …
  <parameters> <label>Processing Parameters</label>
    <integer> <label>Threshold</label> <longflag>threshold</longflag> </integer>
  </parameters>
  <parameters> <label>IO</label>
    <image> <label>Input Volume</label> <longflag>inputvolume</longflag>
      <channel>input</channel> </image>
    <image> <label>Output Volume</label> <longflag>outputvolume</longflag>
      <channel>output</channel> </image>
  </parameters>
  <parameters> <label>Output</label>
    <double> <label>Minimum</label> <name>min</name> <channel>output</channel> </double>
    <double> <label>Maximum</label> <name>max</name> <channel>output</channel> </double>
  </parameters>
</executable>
```

Description of the module (category, title, etc.)

Input numerical value

Input image

Output image

Two output numerical values

2.6 Create a Matlab module

Geenerated Matlab script: *MyFirstMatlabModule.m*

```
% Example function that returns the minimum and maximum voxel
% value in a volume and performs thresholding operation
% on the volume.

function outputParams = MyFirstMatlabModule( inputParams )

img=cli_imageread(inputParams.inputvolume);
```

Input image

```
outputParams.min=min(min(min(img.pixelData)));
outputParams.max=max(max(max(img.pixelData)));
```

Two output numerical values

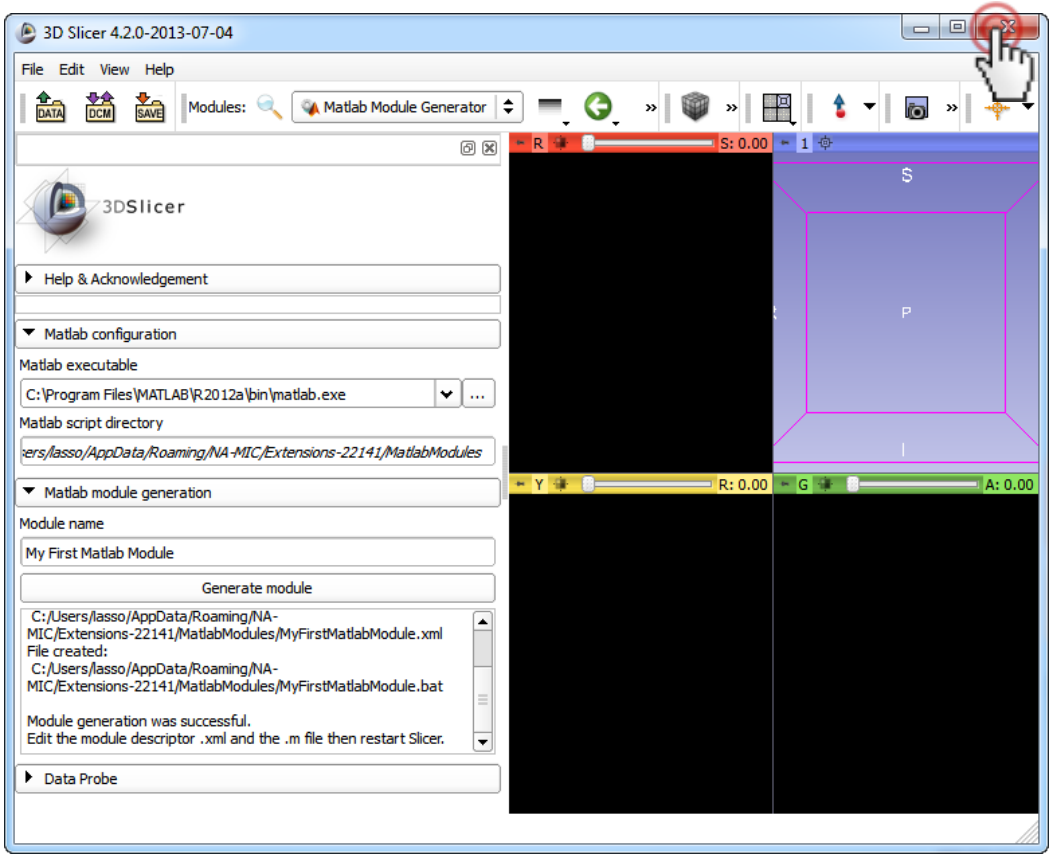
Input numerical value

```
img.pixelData=(double(img.pixelData)>inputParams.threshold)*100;

cli_imagewrite(inputParams.outputvolume, img);
```

Output image

3.1 Use your Matlab module



1. Exit 3D Slicer

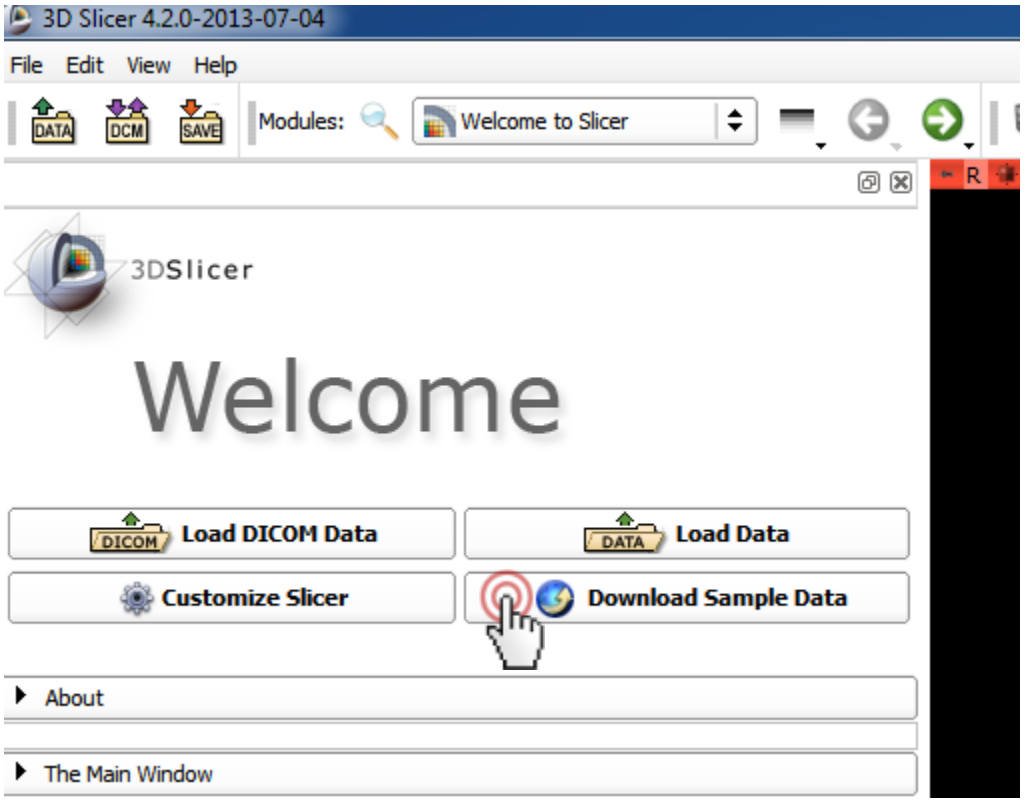
2. Start 3D Slicer

Module graphical user interfaces are created when Slicer is started. So, we need to restart Slicer to see the new module.

3.2 Use your Matlab module

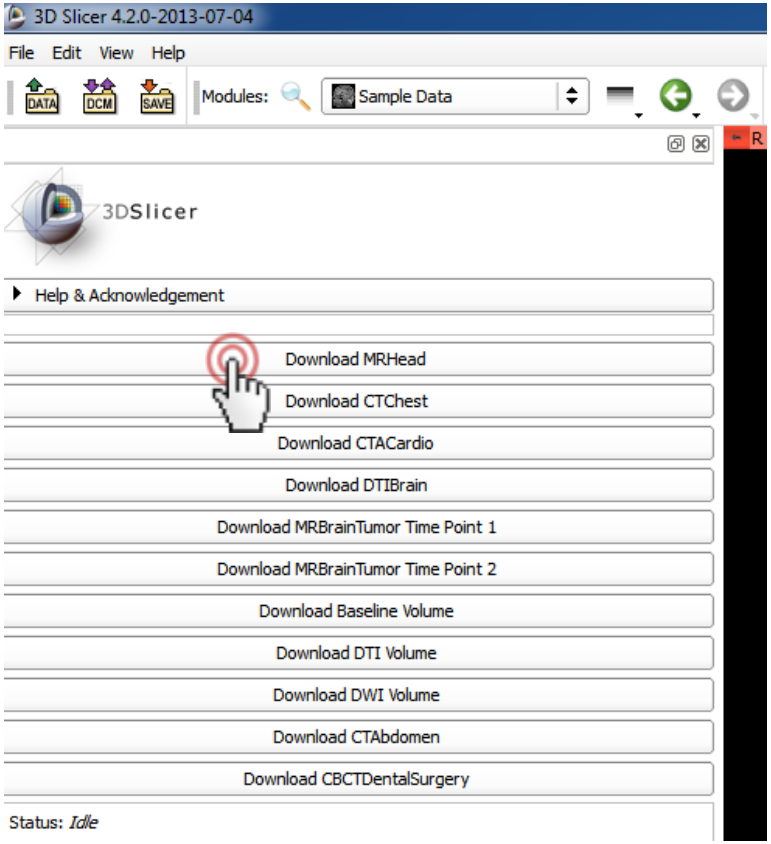
Matlab modules work exactly the same way as other Command-Line Interface (CLI) Slicer modules.

Click *Download Sample Data* module to load some sample data that we will process.

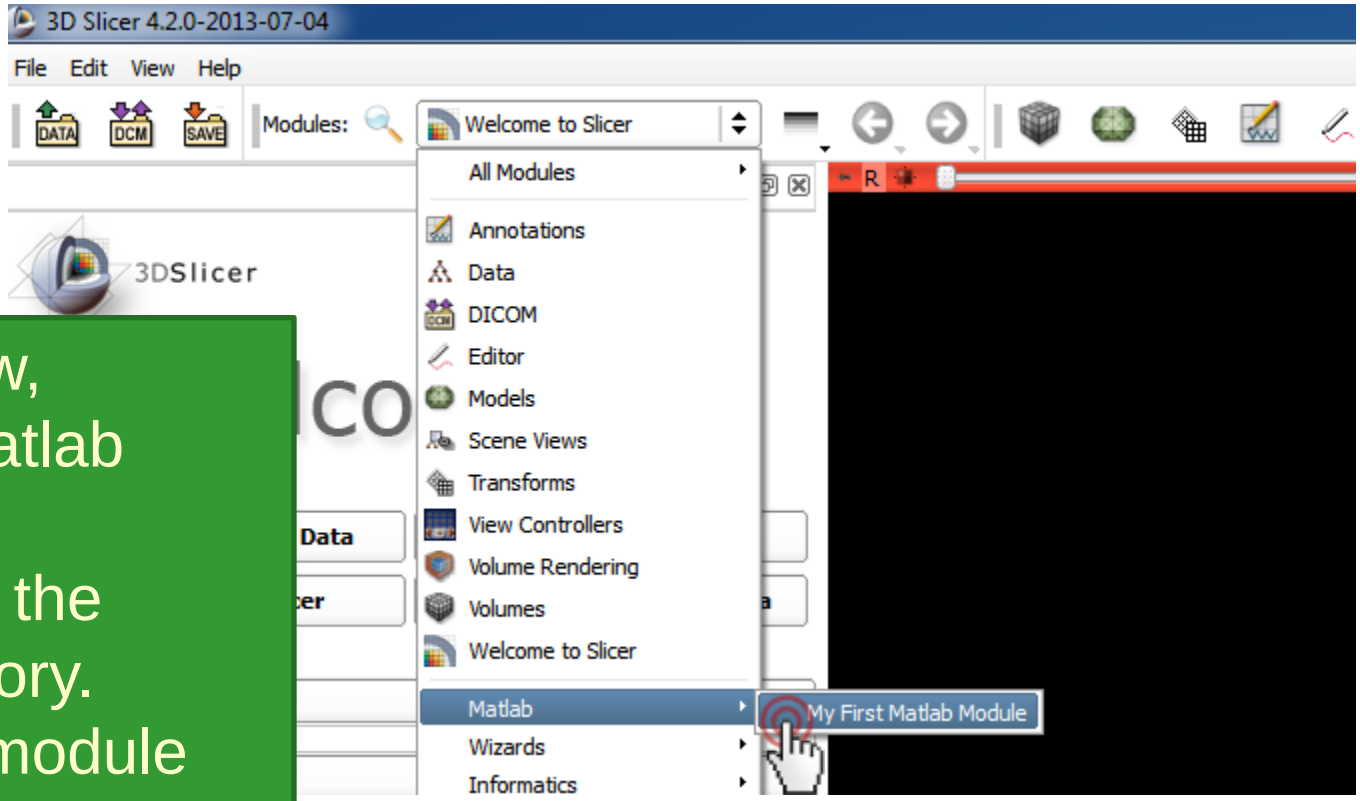


3.3 Use your Matlab module

Click *Download MRHead*



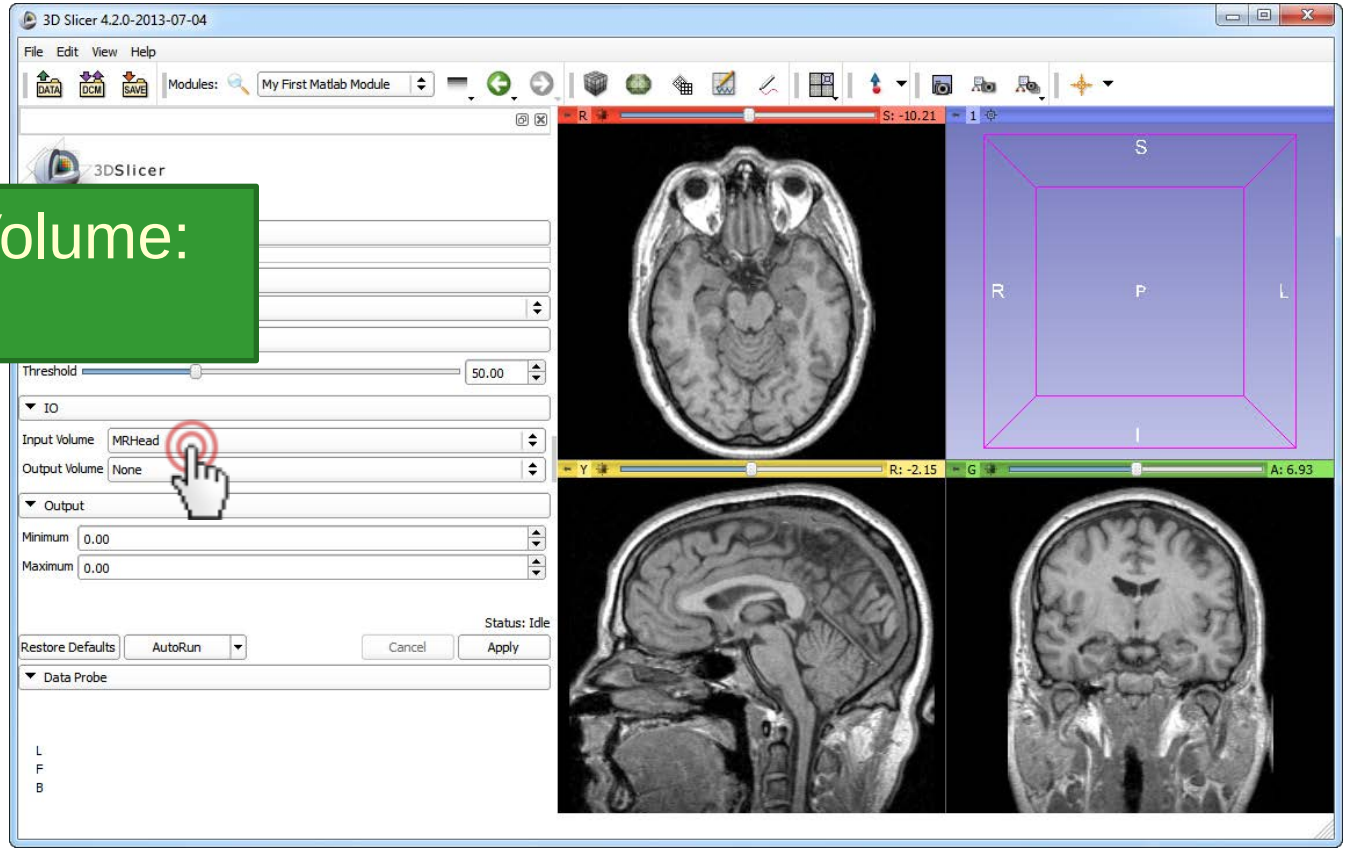
3.4 Use your Matlab module



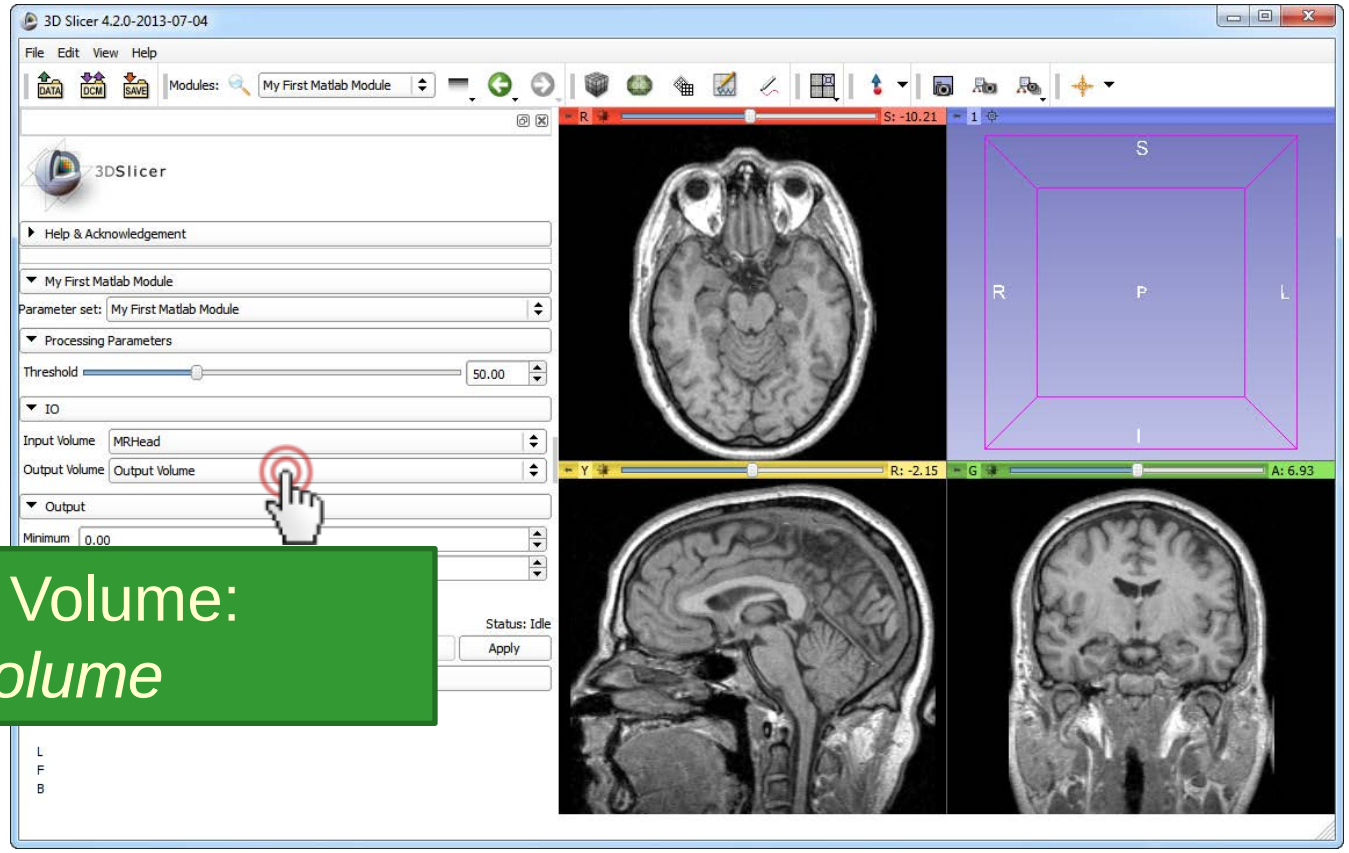
Open our new, generated Matlab module.
It is shown in the *Matlab* category.
Click on the module name to start.

3.5 Use your Matlab module

Select Input Volume:
MRHead

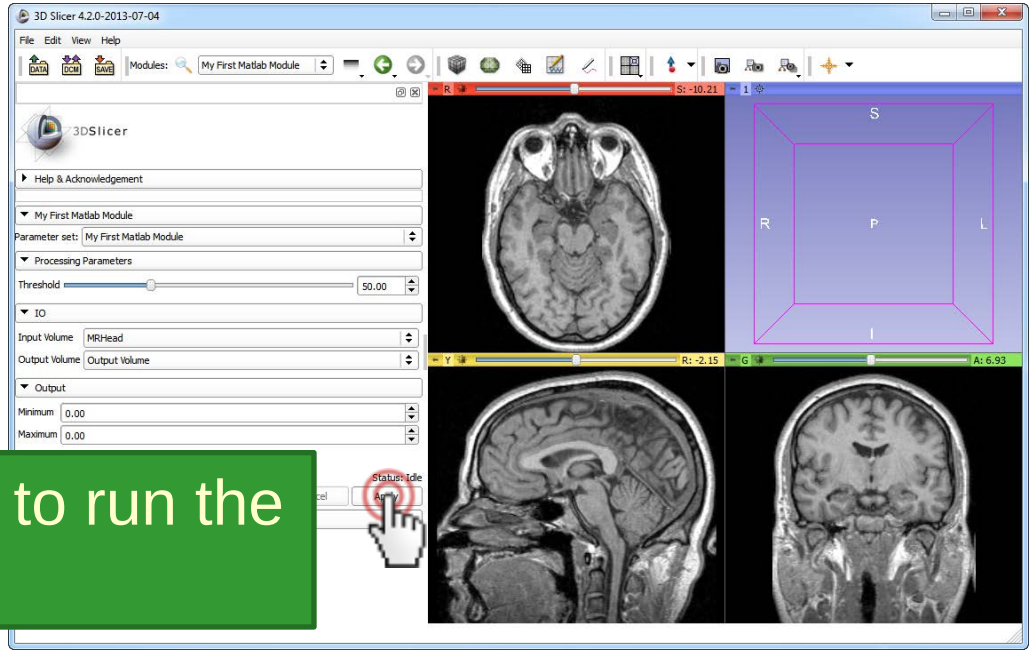


3.6 Use your Matlab module



Select Output Volume:
Create new Volume

3.7 Use your Matlab module



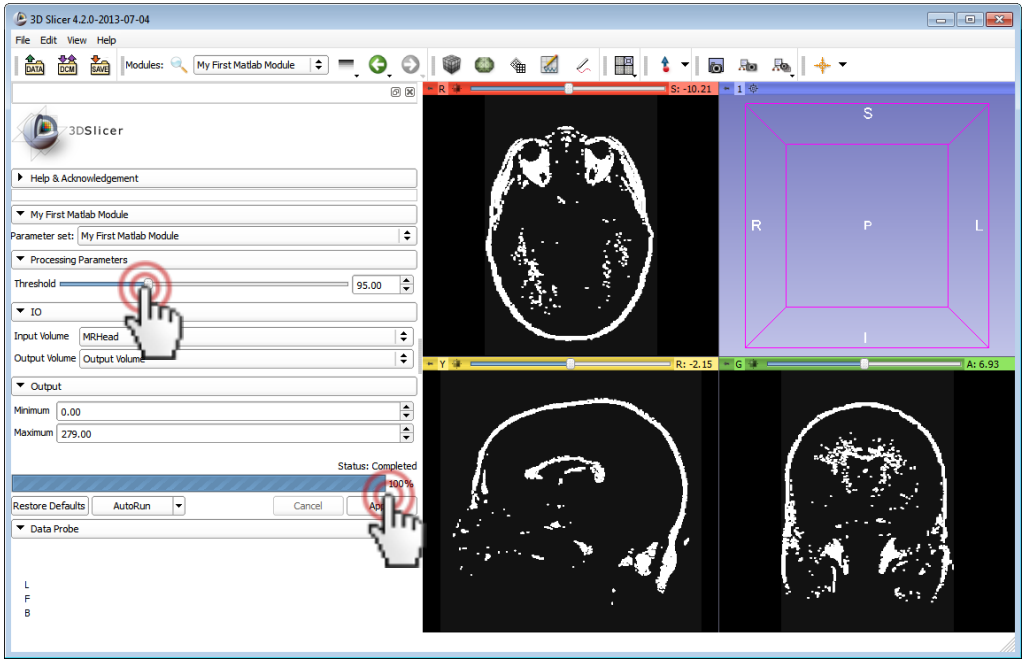
Click *Apply* button to run the module

The first time the *Apply* button is clicked you may have to wait for *Matlab* to start.

3.8 Use your Matlab module

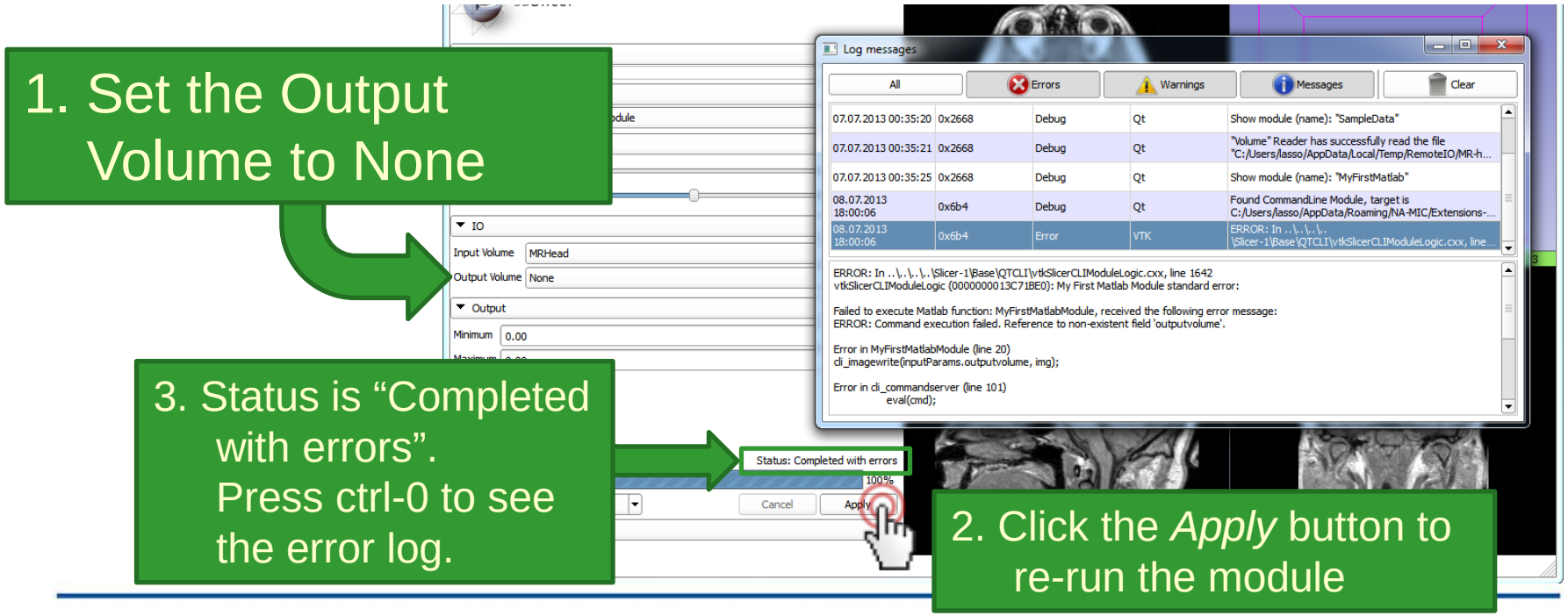
1. Change the *Threshold* value

2. Click the *Apply* button to re-run the module



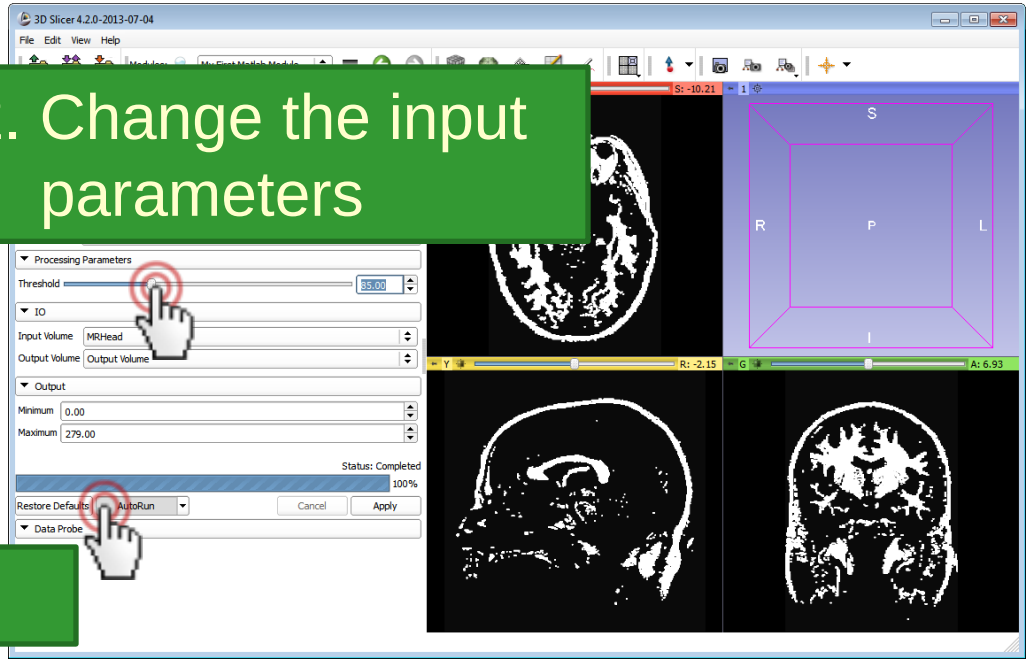
3.9 Use your Matlab module

If an error occurs during the execution of the Matlab function: open the error log in Slicer (View / Error log or ctrl-0) or open the Matlab window to see the error message.



3.10 Use your Matlab module

Enable the AutoRun feature to avoid the need of clicking *Apply* each time an input parameter is changed.

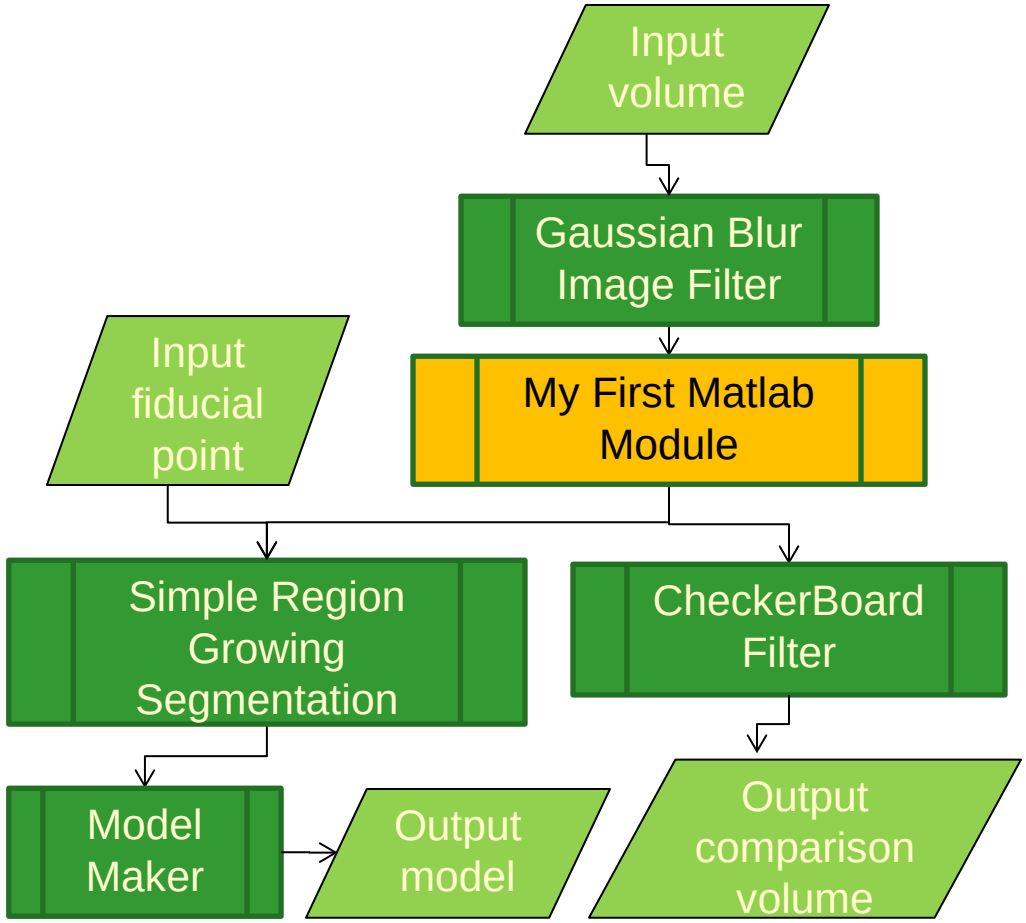


After each change the Matlab function is executed automatically with the new parameters, and the new results are updated on the screen.

3.11 Use your Matlab module

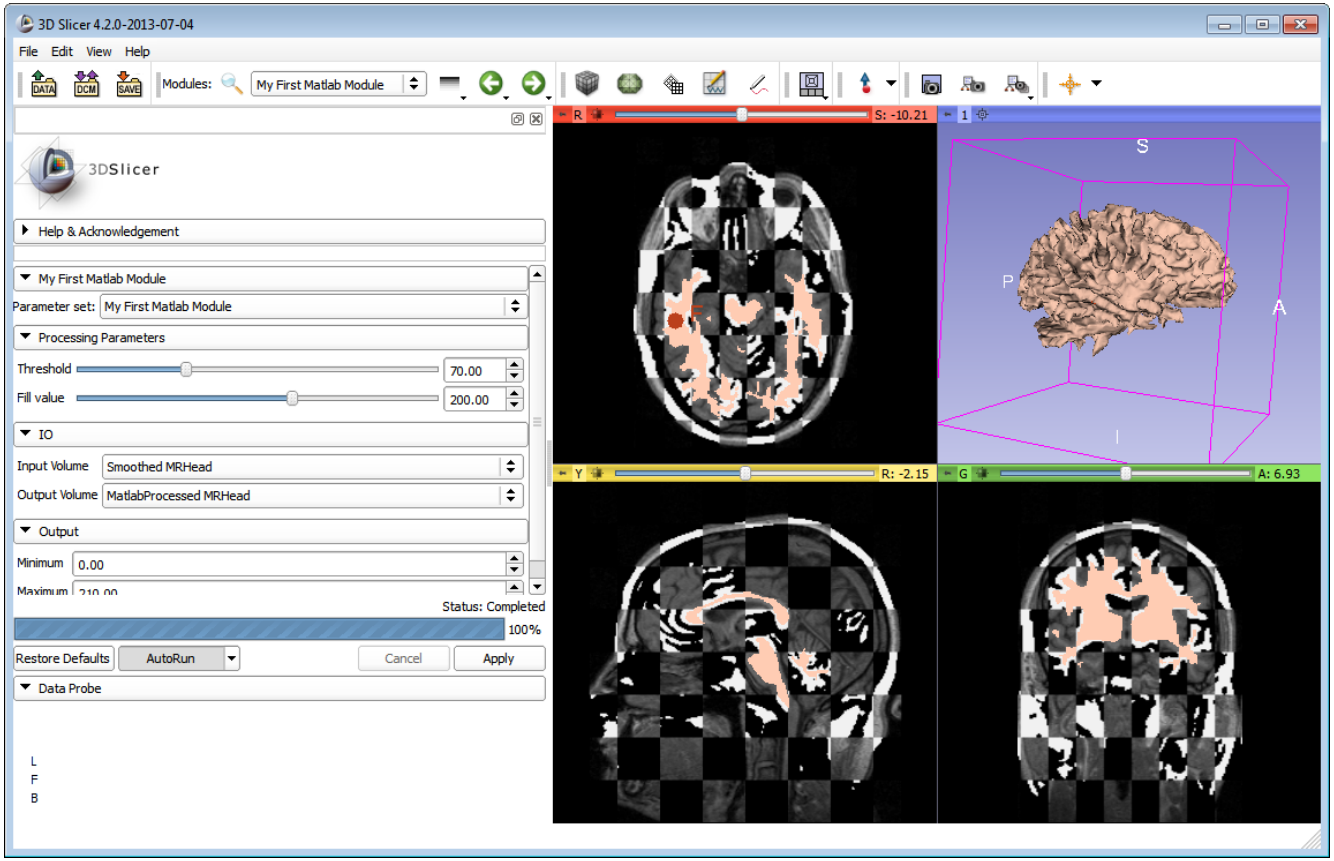
By using *AutoRun*, a pipeline of many pre-processing and post-processing filters and visualization methods can be constructed!

- Choose the output of a CLI module as the input of the subsequent processing module
- Enable *AutoRun on modified input* option of *AutoRun*
- Enable *AutoRun*



3.12 Use your Matlab module

If any input or processing parameter is changed, then the pipeline is refreshed and the updated results are displayed.



4.1 Customize your Matlab module

Example: We would like to introduce one more input parameter, *Fill value.* It is a simple numerical value specifying the value that will be set in the output volume for the values that are above the chosen threshold.

4.2 Customize your Matlab module

Add the following lines in a <*parameters*> section in the *MyFirstMatlabModule.xml* file:

```
<integer>
  <label>Fill value</label>
  <description>
    <![CDATA[Voxels above the threshold will be set to this value]]>
  </description>
  <longflag>fillvalue</longflag>
  <default>200</default>
  <constraints>
    <minimum>-1000</minimum>
    <maximum>1000</maximum>
    <step>5</step>
  </constraints>
</integer>
```

4.3 Customize your Matlab module

Update the Matlab function: replace the line

```
img.pixelData=(double(img.pixelData)>inputParams.threshold)*100;
```

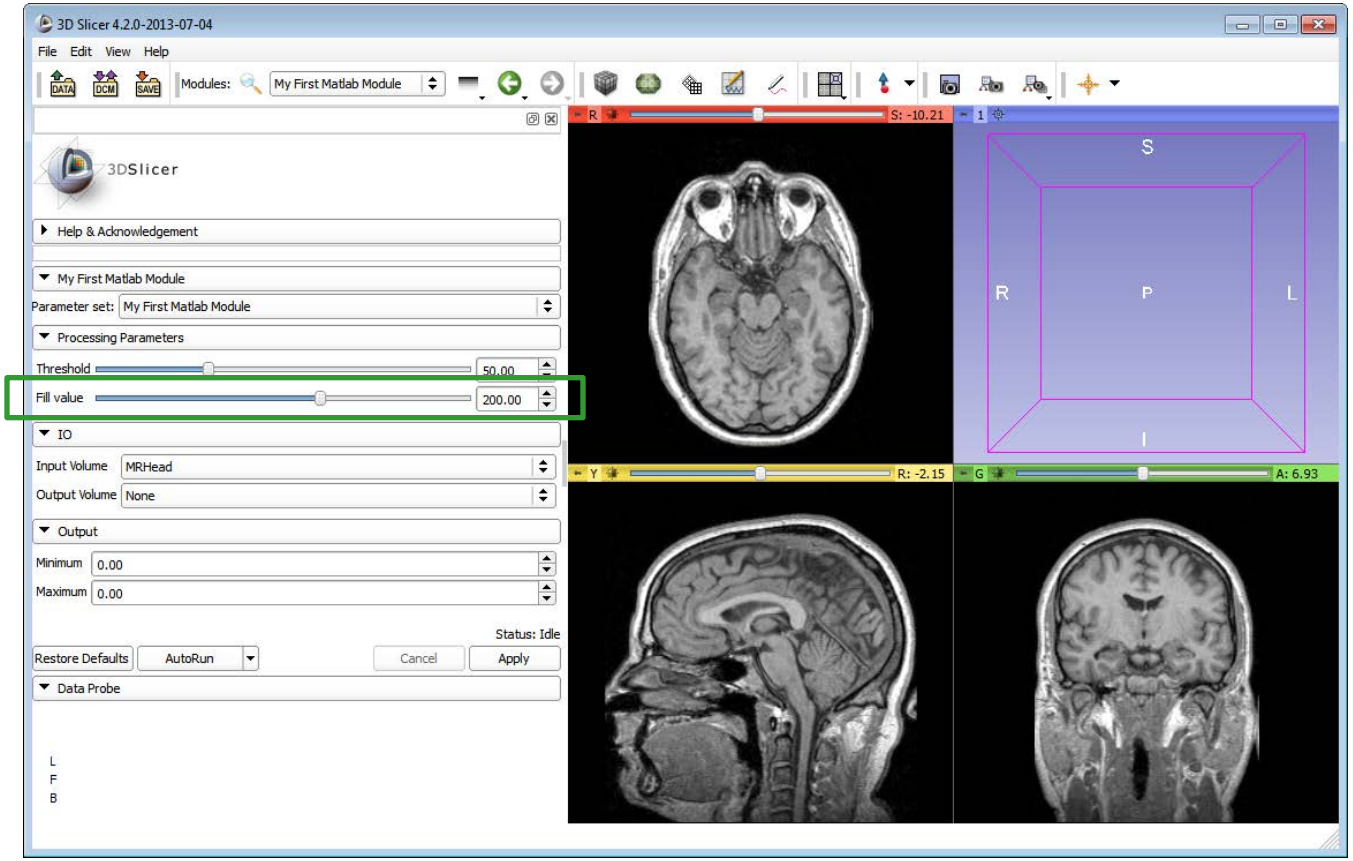
by

```
img.pixelData=(double(img.pixelData)>inputParams.threshold)*inputParams.fillvalue;
```

Restart Slicer to see the updated module graphical user interface and try the module.

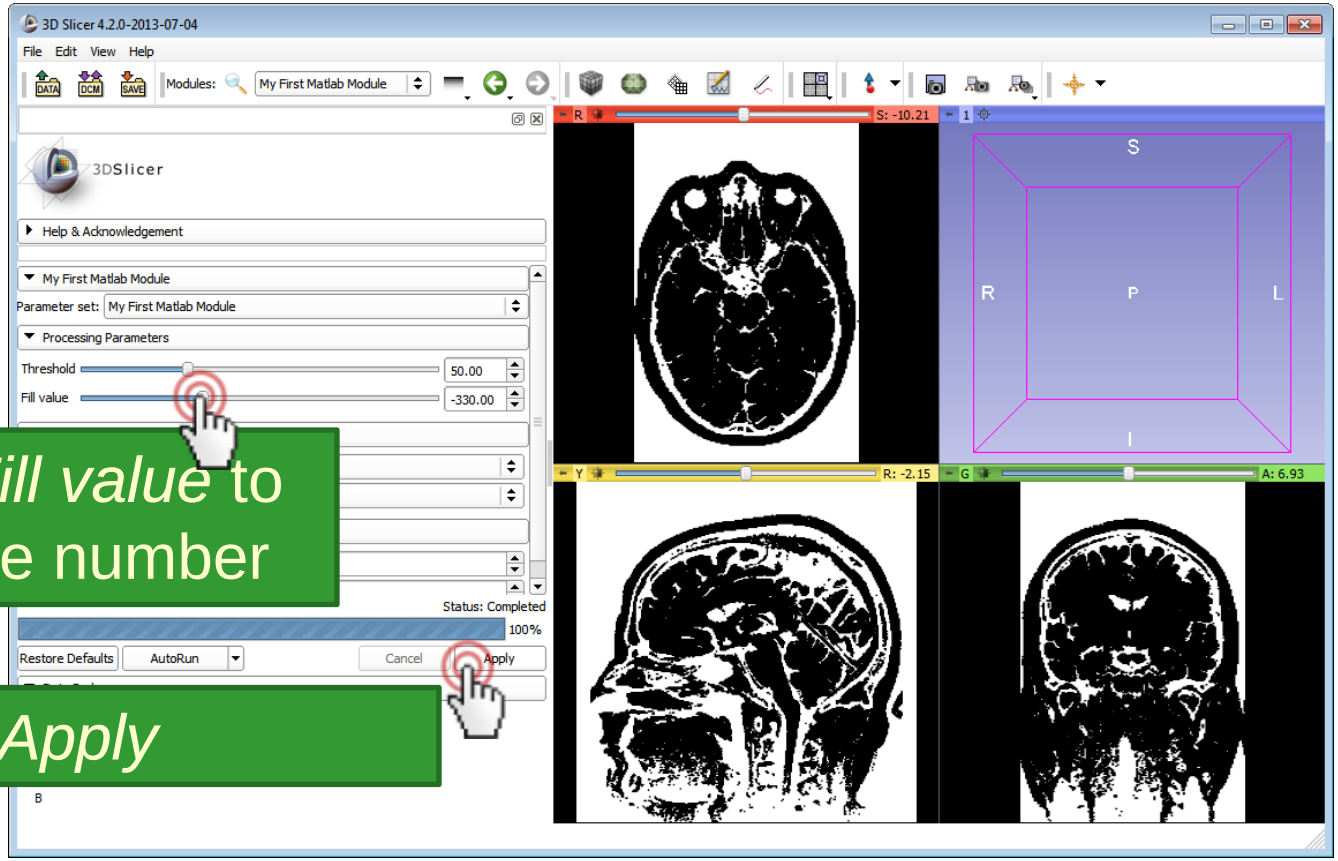
4.4 Customize your Matlab module

The new parameter appears in the user interface.



4.5 Customize your Matlab module

If the fill value is negative then the values above the threshold value will appear darker.



1. Set the *Fill value* to a negative number

2. Click *Apply*

4.6 Customize your Matlab module

- Modifying and testing the Matlab function: simply edit the .m file, save the file, then click *Apply* on the module GUI in Slicer to re-run the module (no need to restart Slicer)
- Setting a breakpoint in the Matlab function:
  - Option A: stop the command server in Matlab (ctrl-c), add breakpoints in the Matlab GUI, restart the cli_commandserver Matlab function
  - Option B: execute a Matlab message using the MatlabCommander module, for example:

```
dbstop in c:\Users\MyUsername\AppData\Roaming\NA-MIC\Extensions-NNNNN\MatlabModules\MyModule.m
```

4.7 Customize your Matlab module

- More information for customizing the XML file:
  - Complete specification of the XML file
  - XML files of all the core Slicer CLI modules
- A few more Matlab module examples
  - Fill around seeds: fills region in a volume around point positions marked in Slicer
  - Landmark registration: computes a linear transform between two point sets marked in Slicer
  - Matlab Bridge parameter passing test: example for using different kind of input and output parameters

Conclusion

- Slicer can be used for running Matlab functions using a convenient graphical user interface.
- Matlab modules behave exactly as any other command-line interface (CLI) module.
- Automatic update feature (*AutoRun*) allows quick visualization of the processing results whenever any input or processing parameter is changed.
- Existing Slicer modules can be used for importing data, pre-processing, post-processing, and visualization.

More information

- Extension documentation page: http://www.slicer.org/slicerWiki/index.php/Documentation/Nightly/Extensions/MatlabBridge
- Add/view bug reports and feature requests at: https://www.assembla.com/spaces/dG15GuCs4r4l4UeJe5cbCb/tickets/report/u783013
- Project webpage: http://www.slicerrt.org/
- PerkLab webpage: http://perk.cs.queensu.ca
- Email contact: Andras Lasso (lasso@cs.queensu.ca)

Acknowledgments

Cancer Care Ontario

SparKit
(Software Platform and Adaptive Radiotherapy Kit)

National Alliance for Medical Image Computing

NIH U54EB005149