User Manual and Documentation

OneClickCut – A 3D Slicer Extension

Guoqing Bao
Supervisor: Sidong Liu; Weidong Cai;
The University of Sydney

Catalogue

1. How to Install?

1.1 Add OneClickCut Module to 3D Slicer

Firstly, open 3D Slicer, select “Edit” menu and then select “Application settings”:

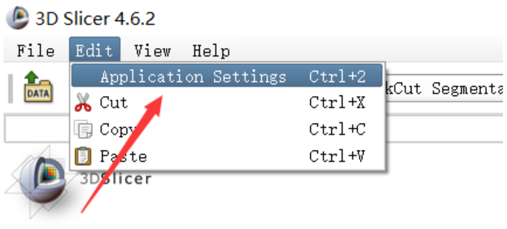
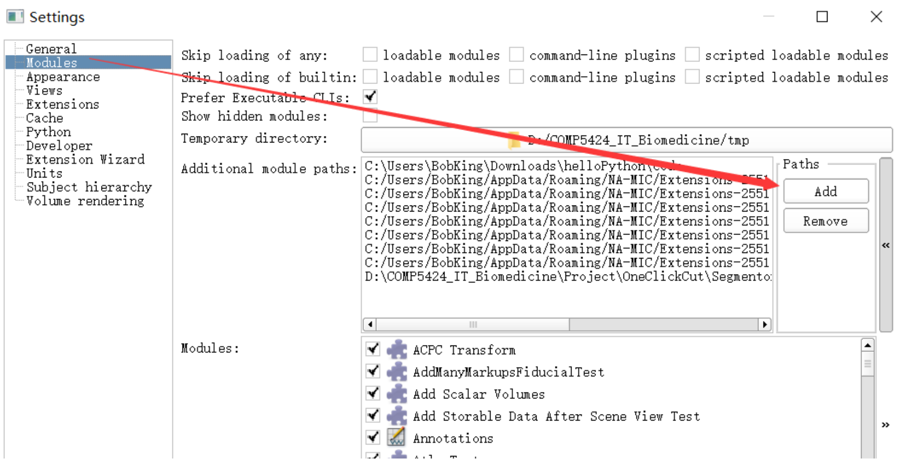


Figure 1 Application settings in menu

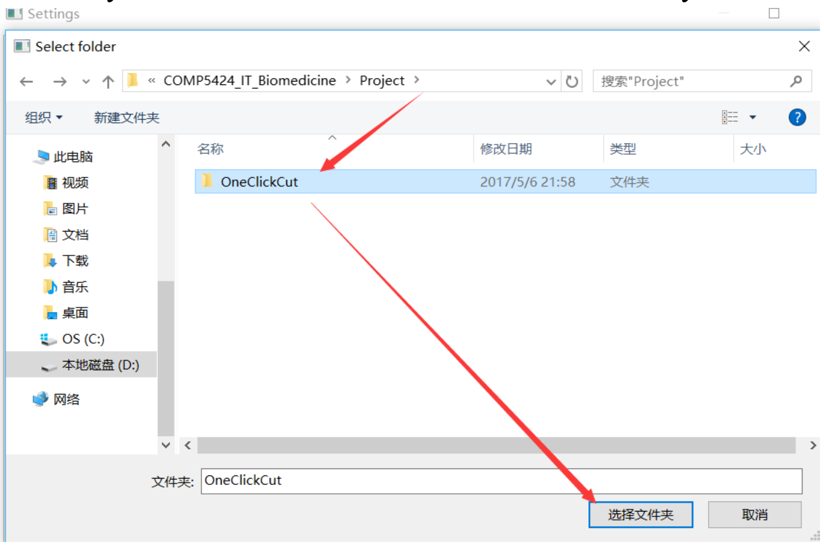
In the “Settings” dialog, select “Modules” tab and then click “Add” button:



```
Skip loading of any:        [ ] loadable modules  [ ] command-line plugins  [ ] scripted loadable modules
Skip loading of builtin:    [ ] loadable modules  [ ] command-line plugins  [ ] scripted loadable modules
Prefer Executable CLIs:     [✓]
Show hidden modules:        [ ]
Temporary directory:        D:/COMP5424_IT_Biomedicine/tmp
Additional module paths:    C:\Users\BobKing\Downloads\helloPython\cod
                            C:/Users/BobKing/AppData/Roaming/NA-MIC/Extensions-2551
                            C:/Users/BobKing/AppData/Roaming/NA-MIC/Extensions-2551
                            C:/Users/BobKing/AppData/Roaming/NA-MIC/Extensions-2551
                            C:/Users/BobKing/AppData/Roaming/NA-MIC/Extensions-2551
                            C:/Users/BobKing/AppData/Roaming/NA-MIC/Extensions-2551
                            C:/Users/BobKing/AppData/Roaming/NA-MIC/Extensions-2551
                            C:/Users/BobKing/AppData/Roaming/NA-MIC/Extensions-2551
                            D:\COMP5424_IT_Biomedicine\Project\OneClickCut\Segmento
Modules:                    [✓] ACPC Transform
                            [✓] AddManyMarkupsFiducialTest
                            [✓] Add Scalar Volumes
                            [✓] Add Storable Data After Scene View Test
                            [✓] Annotations
```

Figure 2 Settings dialog

Choose the directory of OneClickCut and Click “Select Directory”:



名称	修改日期	类型	大小
OneClickCut	2017/5/6 21:58	文件夹	

文件夹: OneClickCut

Figure 3 Install extension from folder

1.2 Find OneClickCut Segmentation Extension after Restart 3D Slicer

After you added the OneClickCut module, you need restart 3D Slicer.

After your restarting 3D Slicer, you can find OneClickCut at Modules (Submenu in Segmentation):

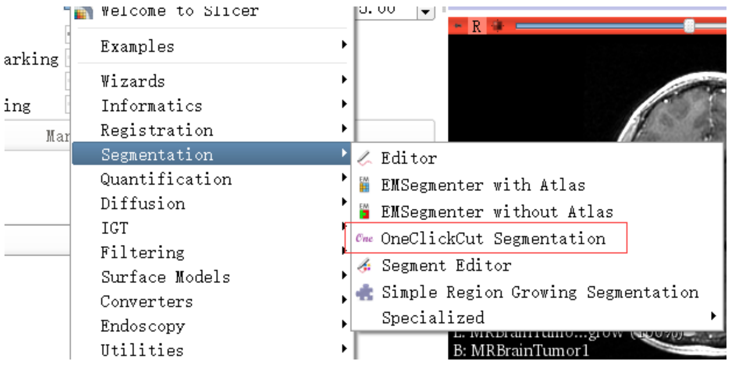
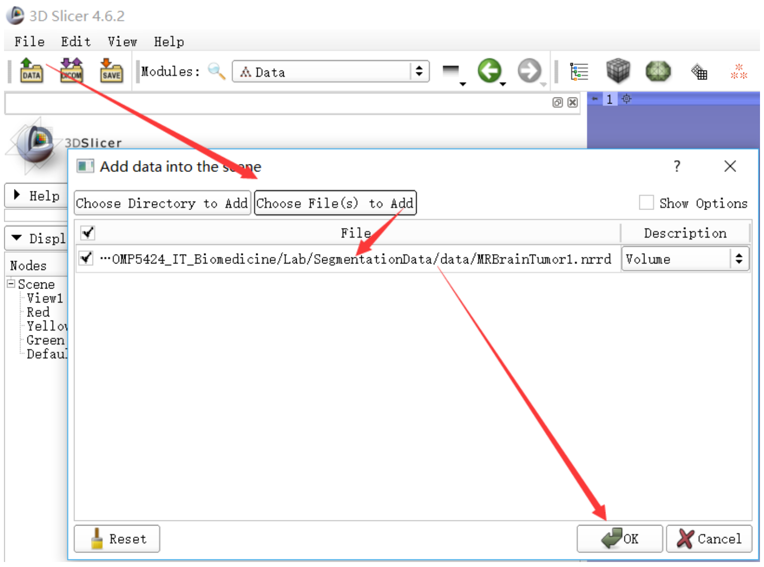


Figure 4 OneClickCut Segmentation tool in 3D Slicer

2. How to use?

2.1 Loading Data

Firstly, you need to load scene as what you did in the past:



…OMP5424_IT_Biomedicine/Lab/SegmentationData/data/MRBrainTumor1.nrrd Volume

Figure 5 Loading data

Click the “Add Data” button in the toolbox or select “Add Data” in “File” menu; then choose a file, like “MRBrainTumor1.nrrd”, then click “OK”.

After you load the scene, find the module called “OneClickCut Segmentation” and select it.

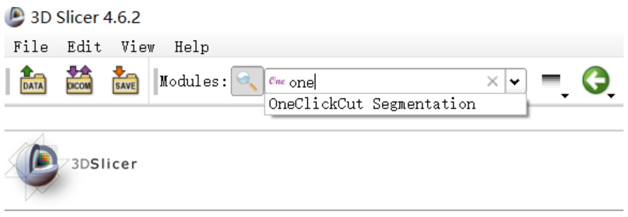


Figure 6 Find OneClickCut module

2.2 Marking seeds

Then, move your mouse to the yellow slice, the mouse will become like the left picture below:

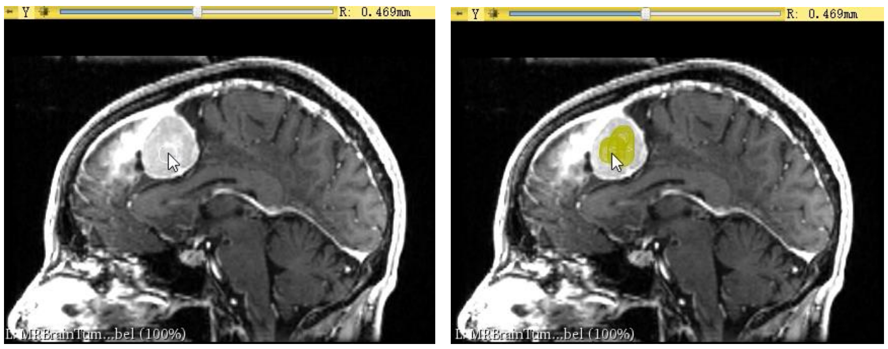


Figure 7 Marking seeds by the user

2.3 Performing Segmentation

You can directly click the region that you want to segment or you click and move the mouse to select a larger region, like the right picture above; the segmentation process will complete in 3 seconds after you release your mouse. The result will like this:

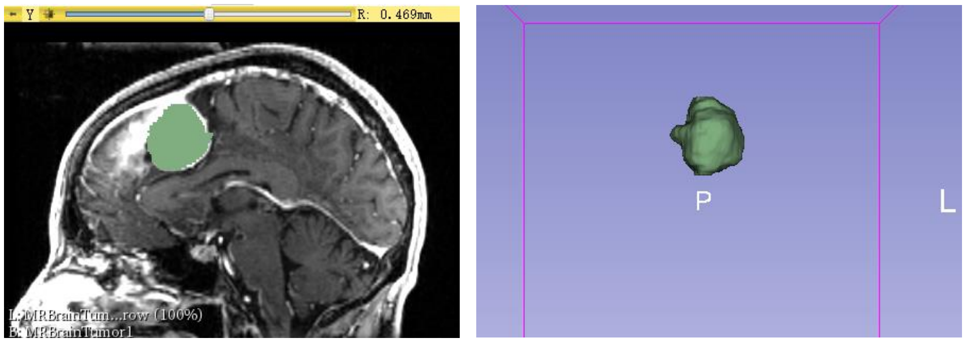


Figure 8 Segmentation results

3. Advantages of OneClickCut?

3.1 Segmentation Results Comparison

Firstly, we compare segmentation results of default GrowCutEffect, FastGrowCutEffect, Region Growing with intensity range and our OneClickCut:

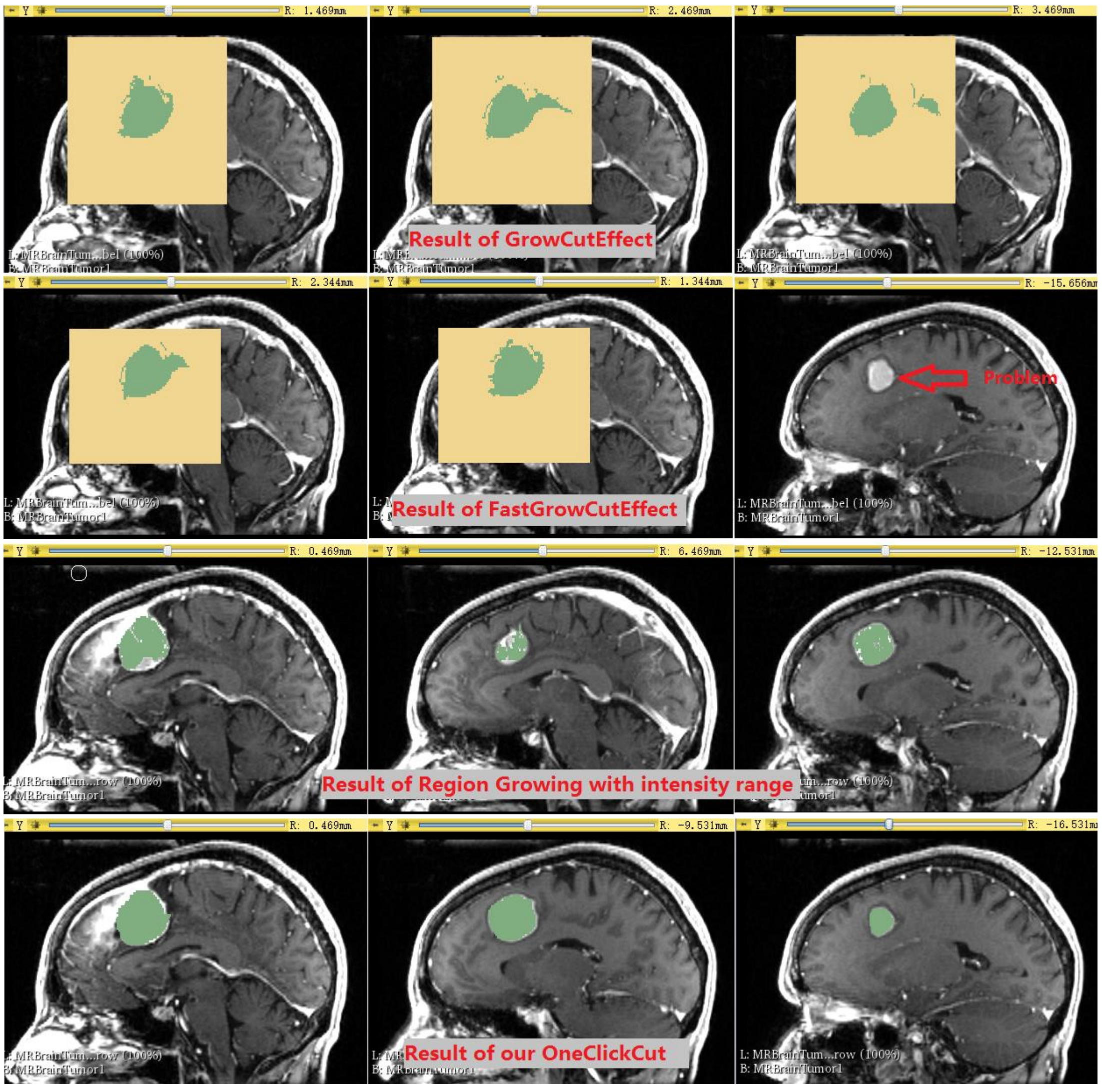

Result of GrowCutEffect
Result of FastGrowCutEffect
Problem
Result of Region Growing with intensity range
Result of our OneClickCut

Figure 9 Comparison segmentation result of OneClickCut with others

From the above picture, we can see that our segmentation extension achieved much better results than default GrowCutEffect, which provided by 3D Slicer and the other segmentation tool (FastGrowCutEffect). Out results are much smoother, the boundary between the tumor and surrounding tissue are much clear and accurate than the result of existing segmentation tools.

3.2 3D Model Comparison

Next, we compare 3D model created by different segmentation tools:

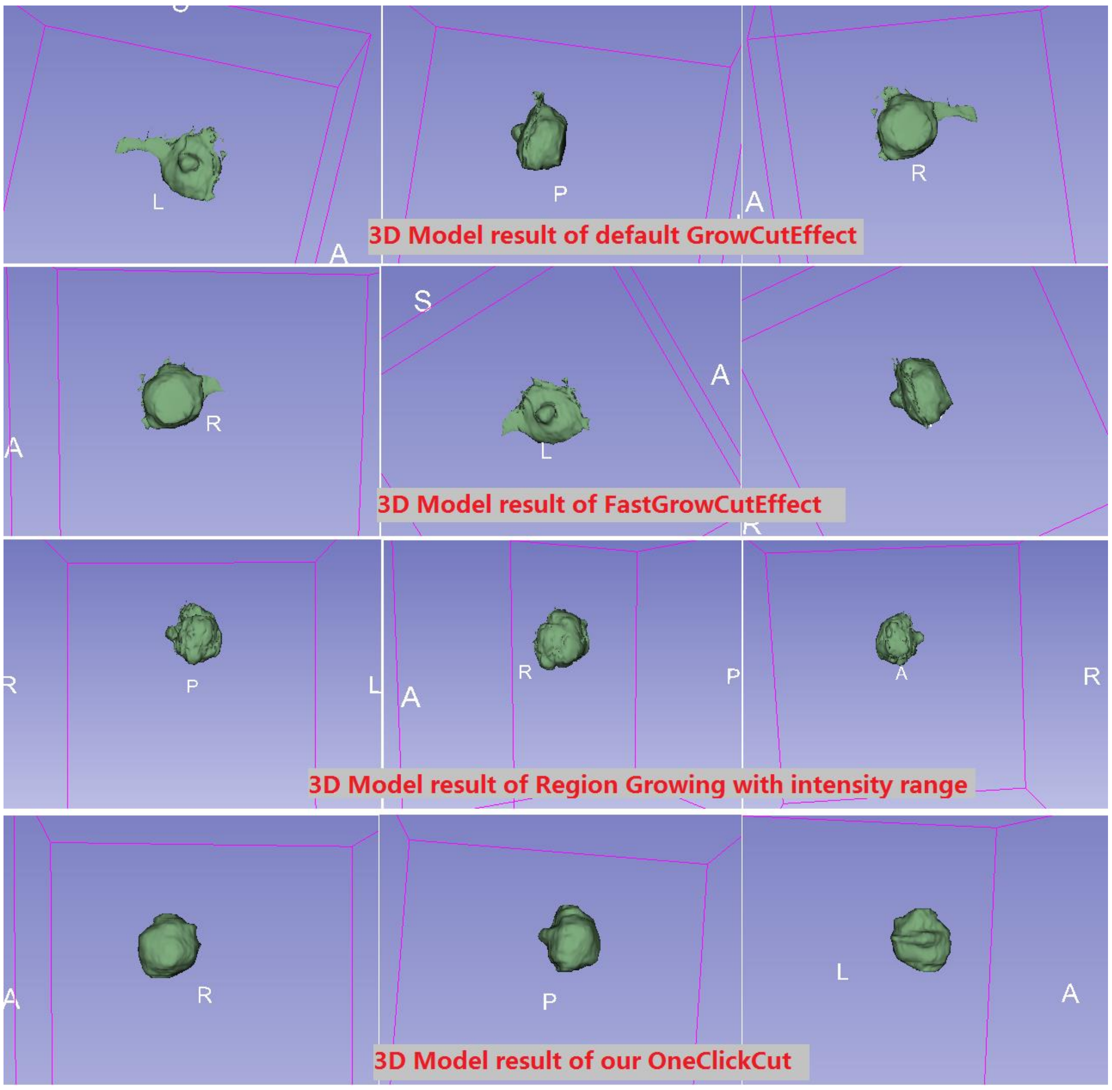

3D Model result of default GrowCutEffect
3D Model result of FastGrowCutEffect
3D Model result of Region Growing with intensity range
3D Model result of our OneClickCut
L
P
R
A
S

Figure 10 Comparison 3D Model generated by OneClickCut with others

Obviously, the 3D models created from the segmentation results by default GrowCutEffect and third-party FastGrowCutEffect are inferior to the 3D model that automatically created by our OneClickCut extension.

3.3 Time Complexity Comparison

Finally, we compare the time costs of different segmentation tools:

Tools\Time	**Segmentation time**	**Model building time**	**Total time**
GrowCutEffect	13.5 s	1.6 s	15.1 s
FastGrowCutEffect	1.2 s	1.6 s	2.8 s
Region Growing with intensity range	1.1 s	1.2 s	2.3 s
OneClickCut	1.1 s	1.2 s	2.3 s

**Note: Total time of GrowCutEffect and FastGrowCutEffect does not include time costs of user's hands-on operation, like choosing suitable paint size, marking the seeds and prepare results for building a 3D model; Total time of OneClickCut include all those manual operation time.*

From the above results, we can see that OneClickCut still achieved the best performance although we include time costs of user's hands-on operation. We don't consider user's operation time into the total time of default GrowCutEffect and FastGrowCutEffect since those time costs may be very different from one user to another, it depends on the user's skill level in using the 3D slicer. For Region Growing with intensity range, the result of this algorithm is too rough to be used in actual application.

4. System and UI design

4.1 Architecture design

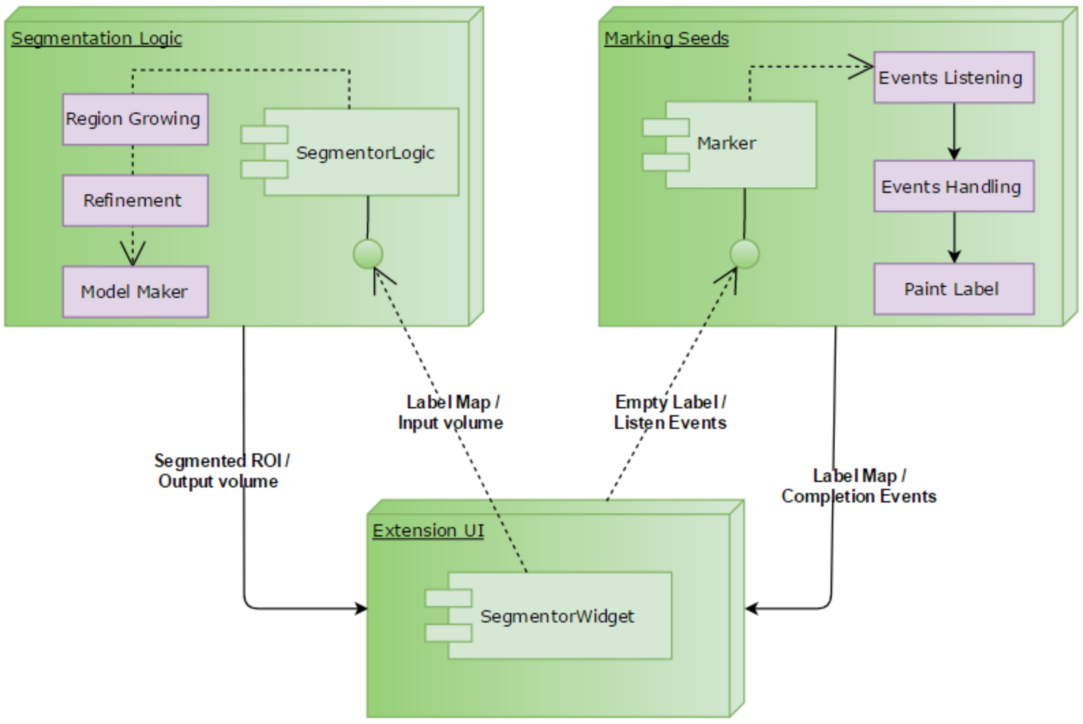


Figure 11 Architecture design of OneClickCut Extension

The above figure shows the architecture of our OneClickCut extension, it consists three modules: SegmentorWidget, Marker and SegmentorLogic.

The SegmentorWidget is the UI of this extension, it uses "Marker" to processing users' seed marking events, after a marker is created, the SegmentorWidget monitors the Scene events, if new scene is loaded, it will automatically create an empty label map for input volume and pass it to the "Marker", then tells the "Marker" to listen the user's mouse events.

The “Marker” listens to the mouse event and process such event, “Paint Effects” are applied to response to the user’s marking action, like the left picture of Figure 2:

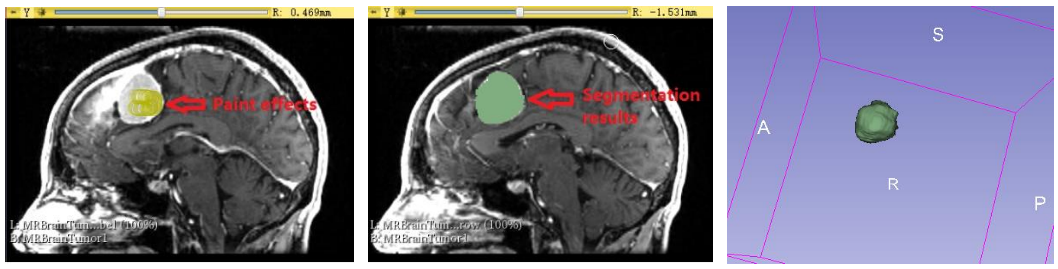


Figure 12 Segmentation steps

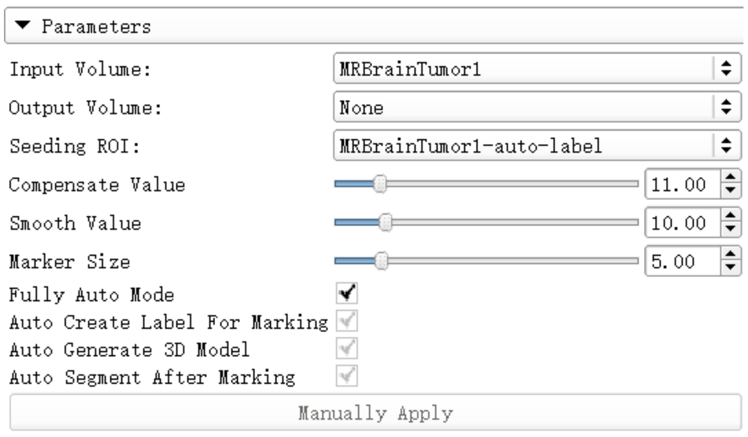
After the user released the mouse, the “Marker” will paint the label map and pass the painted label map back to the UI (the “SegmentorWidget”).

After received the marking completion event and painted label map, the “SegmentorWidget” will pass the input volume and painted label map to the “SegmentorLogic”, the SegmentorLogic will do below tasks to segment the 3D ROI:

1) Firstly, the SegmentorLogic will call Region Growing algorithm to segment the initial ROI.
2) Then, refinement methods are used to improve the initial segmentation results. (See 3.2 Segmentation method) The refinement results like the middle picture in Figure 2.
3) Using the refinement results as 3D label map to create a 3D model and present it to the 3D viewer, like the right picture in Figure 2.

4.2 UI design

Since we want to streamline the segmentation process and minimize the difficulties of using our extension, the UI is designed to reduce the intervention from end users, most of the parameters are automatically selected or predefined:



Parameters

Input Volume:	MRBrainTumor1
Output Volume:	None
Seeding ROI:	MRBrainTumor1-auto-label
Compensate Value	11.00
Smooth Value	10.00
Marker Size	5.00
Fully Auto Mode	✓
Auto Create Label For Marking	✓
Auto Generate 3D Model	✓
Auto Segment After Marking	✓

Figure 13 User interface of OneClickCut

This extension can monitor users' actions in 3D slicer (listen to the scene events), the "Input volume" and "Seeding ROI" are automatically selected and created after user loaded the data or user entered this extension. However, the user can change input volume if they think the current selection is not favorable. The "Output volume" is automatically created after segmentation, it is not required to predefine before segmentation. That can reduce the complexities of using this segmentation extension.

The "Compensate value" is used for compensating a value to the intensity range of Region Growing algorithm; the "Smooth value" indicates the degree of smoothing when creating 3D Model.

The "Marker size" is predefined to 5 (pixels radius) by default, this size is determined by our experience in using other segmentation tools, we found that other segmentation tools set this value too large, the user need to change it every time when they using it, but, our default setting of Marker size, 5-pixels-radius, is suitable for most cases, although the user can change the value by themselves.

We set "Fully Auto Mode" to be checked by default to minimize users' intervention, so that this extension will "Auto Create Label for Marking", "Auto Segment after Marking" and "Auto Create 3D Model". However, the user can uncheck "Fully Auto Mode", that will make the user determine that whether it automatically create label map or not, whether it automatically do segmentation after marking the seeds or not, whether it automatically create 3D model or not, if the user unchecked one of them, such as "Auto Segment after Marking" is unchecked, the "Manually Apply" button is enabled, then the user can click this "Manually Apply" button to do segmentation after finished seeds marking. Similarly, if the user unchecked "Auto Create 3D Model", the 3D model will not be created after segmentation.

5. Interfaces and code analysis

5.1 Source files analysis

```
#
Segmentor.py
#
```

The Segmentor Module, Include 2 Classes:

"Segmentor" is the Scripted Loadable module, used for setting product info.

"SegmentorWidget" is the UI of this product, handle events (with the help of Marker class). SegmentorWidget have a member called "marker" which is the instance of Marker, used for selecting seeds.

```
#
Marker.py
#
```
Include Marker class which implement simple paint effect.
This class used for marking the seeds, modified from the Editor module of 3D Slicer; UI module will call the listen function of this class, after listening the mouse event; this class will process the user input event, handle the marking seeds function.

```
#
SegmentorLogic.py
#
```
The actual implementation 3D segmentation of extension OneClickCut; the main segmentation logic is implemented in "run" function; the UI model will call the run function after user marked the initial seeds; region-grow(grow-cut) algorithm was first used, then Opencv was used for refinement of the output of grow-cut, then the 3D model was presented in 3D view.

```
#
SegmentorUtils.py
#
```
Include the helper function for region growing algorithm.

```
#
Cv2.pyd
#
```
Is the OpenCV python module, which is loaded by OneClickCut at runtime. OneClickCut will use some functions from this module for postprocessing.

```
#
Testing/ MRBrainTumor1.nrrd
#
```
Used for testing, which will be loaded by OneClickCut at runtime if you click "Reload & Test" button.

5.2 Functions and Interfaces

```
#
Class SegmentorWidget
#
def enter(self):
```
Usage: After the user entered this extension, we need to automatically create a label for user to marking (if Auto Mode and need to create that label). Then we listen the events,

such as mouse event, scene event.

```
def exit(self):
```
Usage: Remove mouse and scene listener after user leaved this extension.

```
def onSceneClose(self, caller=None, event=None):
```
Usage: When scene closed, drop the listener.

```
def createLabel(self):
```
Usage: If Auto Create Label is checked, automatically create a label for user to mark.

```
def onPaintBefore(self):
```
Usage: Marker will call this function to get label node, marker will paint on this label.

```
def onEndPaint(self):
```
Usage: Marker would call this function after user have marked the seeds.

```
def onMarkerSizeChanged(self):
```
Usage: The marker size changed by user, recreate the marker with that size.

```
def updateLabelNode(self, caller=None, event=None):
```
Usage: Used for detection of volume changes, after user reselect the input volume or changed the scene, it the label map is not created yet, create it automatically.

```
def onApplyButton(self):
```
Usage: Call the logic module to do real works when user clicked apply button (only happens in manual mode) or being called after marking completed (happens in auto mode).

```
#
Class Marker
#
def setup(self, parent, radius):
```
Usage: Used to initialize the marker. Such as initializing the paint and paint-effect, prepare event listeners.

```
def createGlyph(self, polyData):
```
Usage: Create a poly graph, used for marking the seeds, mark a poly graph seed area after user click instead of just mark one pixel.

```
def listen(self):
```
Usage: Listen the user event, such as mouse event, processEvent function will handle

these events.

```
def dropListen(self):
```
Usage: After user leave this extension or closed the scene, we drop the event listener, that means don't listen the mouse event.

```
def processEvent(self, caller=None, event=None):
```
Usage: Process the marking event, after listen to the 3D volume, the mouse button down, mouse move and mouse release event will help us to paint the label (marking seeds).

```
def paintAddPoint(self, x, y):
```
Usage: Call this function when painting, if delayed painting, show the paint effects, otherwise do real painting (on label map).

```
def paintFeedback(self):
```
Usage: Show the paint effects when painting.

```
def paintApply(self):
```
Usage: Do the real paint work, will first get label node, then call paintBrush do real paint on label map, then show paint effects.

```
def paintBrush(self,labelNode, x, y):
```
Usage: This function will do real painting; it uses a brush to paint the seeds instead of draw single pixel. Paint label is 1 by default.

```
#
Class SegmentorLogic
#
def run(self, inputVolume,seedingROI, paintSize, automodel):
```
Usage: The actual segmentation algorithms implemented in this function: firstly, use grow-cut to make initial ROI; then, use Opencv (convex hull, morphological operation) to refine the ROI; finally, present the ROI model in 3D view.

```
def growCut(self, inputVolumeData, seedingROIData,
outputROIData):
```
Usage: Region growing algorithm in 3D space, modified from grow-cut in 3D slicer. However, the result of this grow-cut algorithm is rough, need further refinement.

```
def makeModel(self,volumeNode):
```
Usage: This function is modified from "Model Maker" in 3D slicer, used to create a model using the command line module, based on the current editor parameters.

```
#
Class SegmentorTest
#
def runTest(self):
```
Usage: After “Test” button clicked, this function will be called, used for testing the OneClickCut.

```
def test(self):
```
Usage: Main test function, include loading test data, testing paint effect, marking the seeds, testing segmentation.