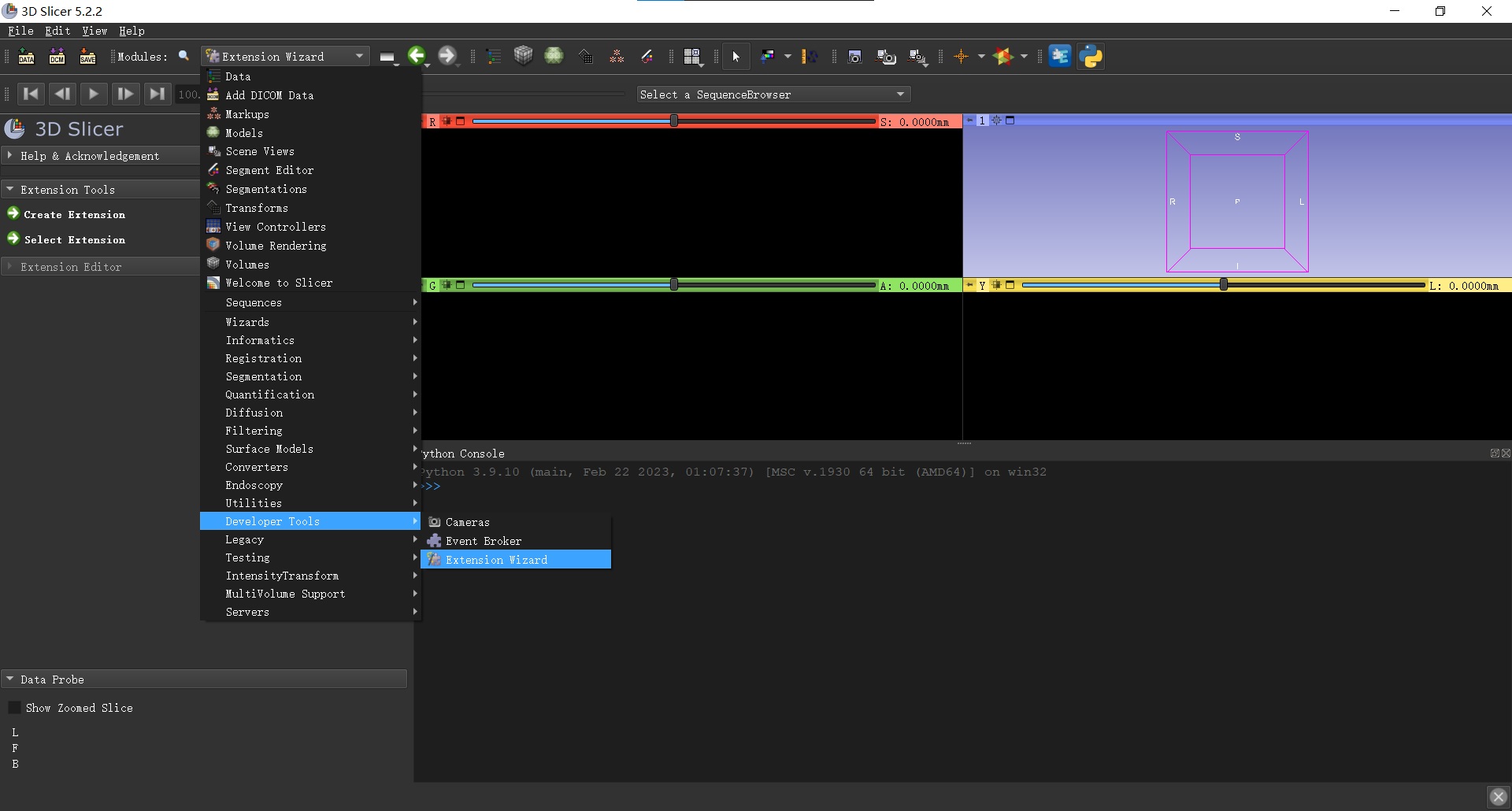
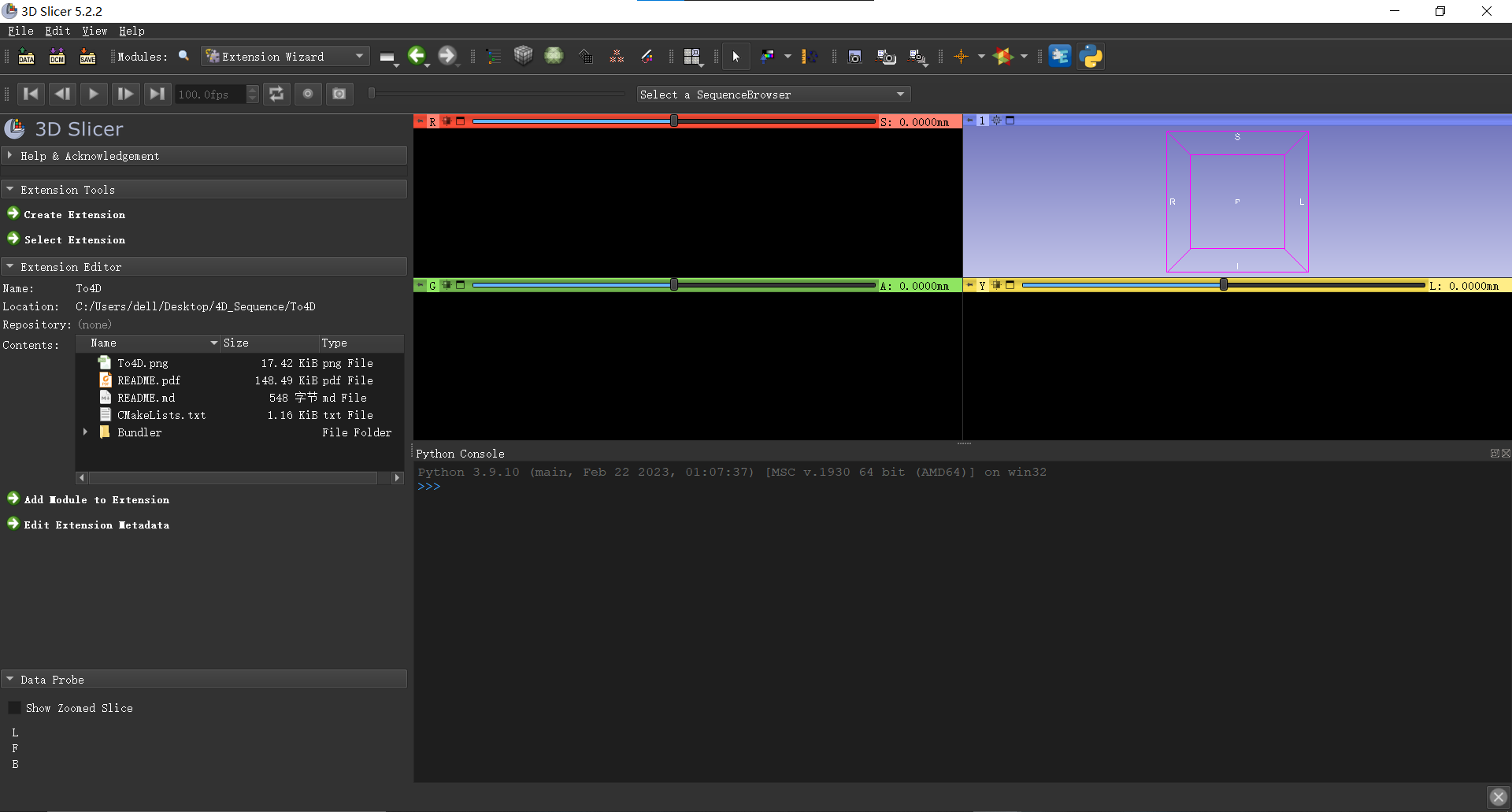
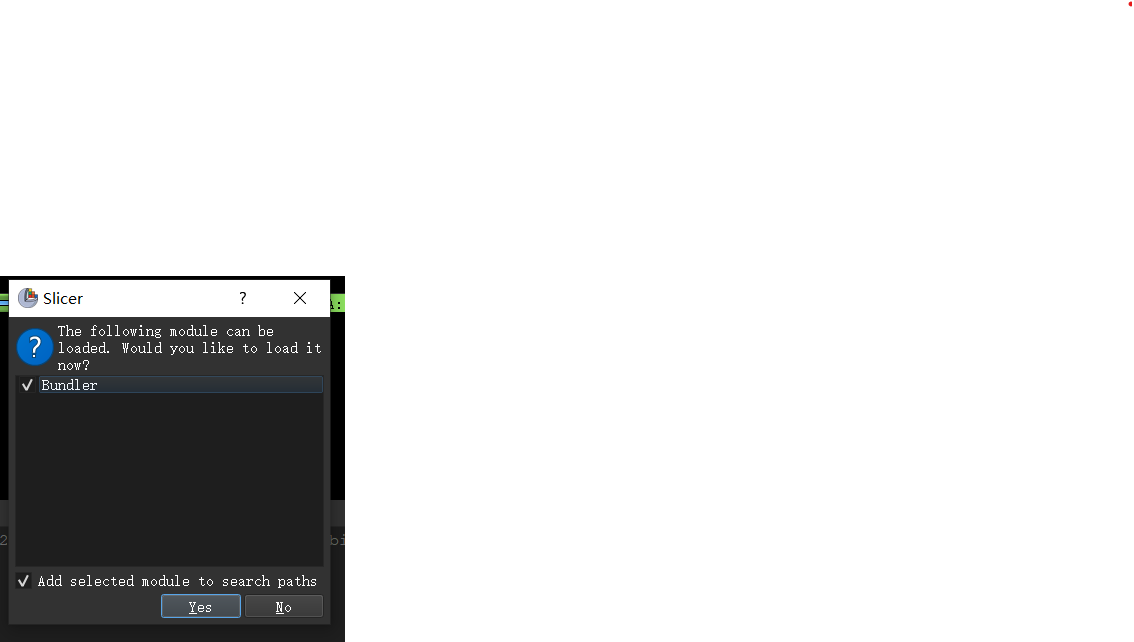
安装流程：

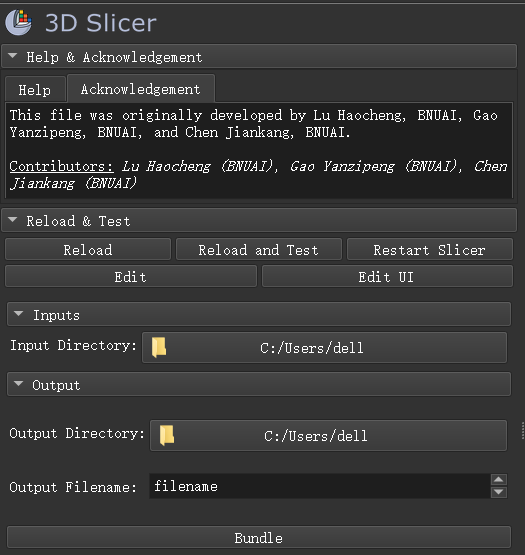
1. 下拉菜单Modules->Developer Tools->Extension Wizard
2. 左侧面板Extension Tools->Select Extension->选择拓展所在目录



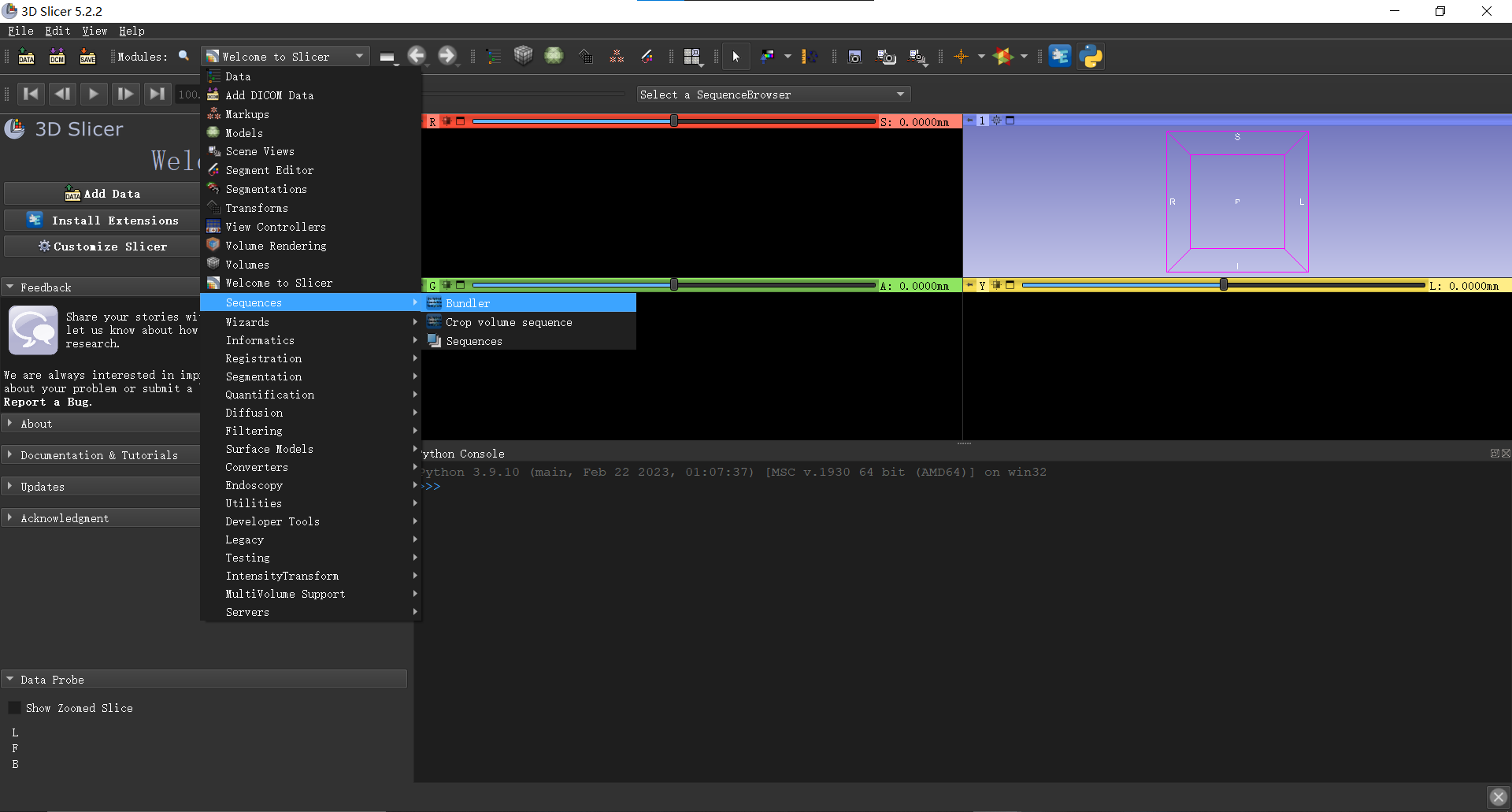
1. 如下窗口，确认复选框均被勾选后单击Yes。



1. 重启3D Slicer后拓展安装完成。

使用流程：

1. Modules->Sequences->Bundler



1. 操作界面

Input Directory可以选取待处理的Volumes所在的文件夹。

Output Directory可以选取存放生成的Sequence所在的文件夹。

Output Filename中可以填写生成的文件名（不含文件后缀）。

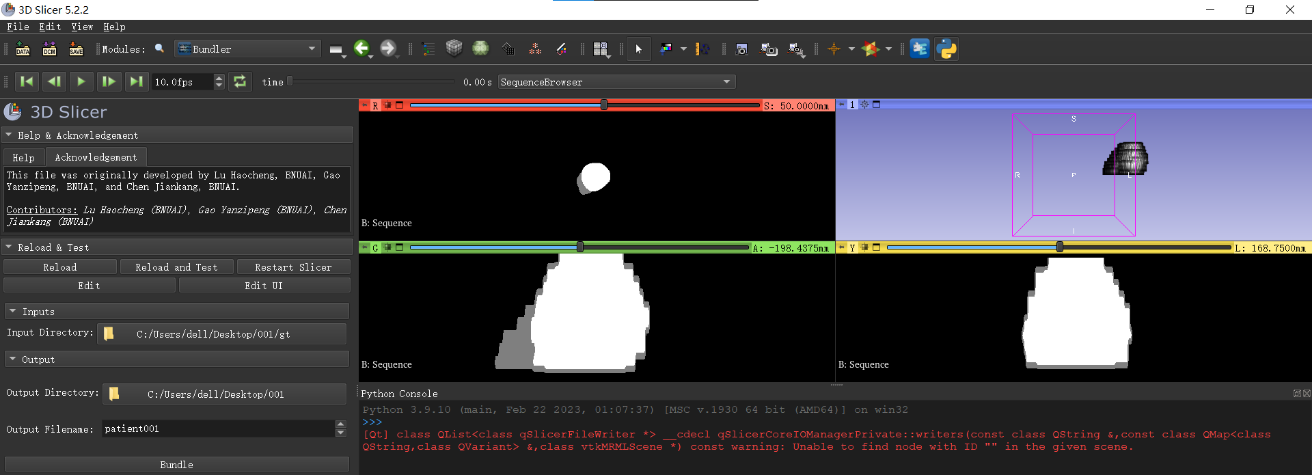
单击Bundle后等待处理完成即可，此后右侧视图会出现合成好的模型。

1. 实例演示

C:\Users\dell\Desktop\001\gt, C:\Users\dell\Desktop\002\gt两个文件夹中分别存放着一个病人心脏的30个格式为.nii.gz的Volume文件。

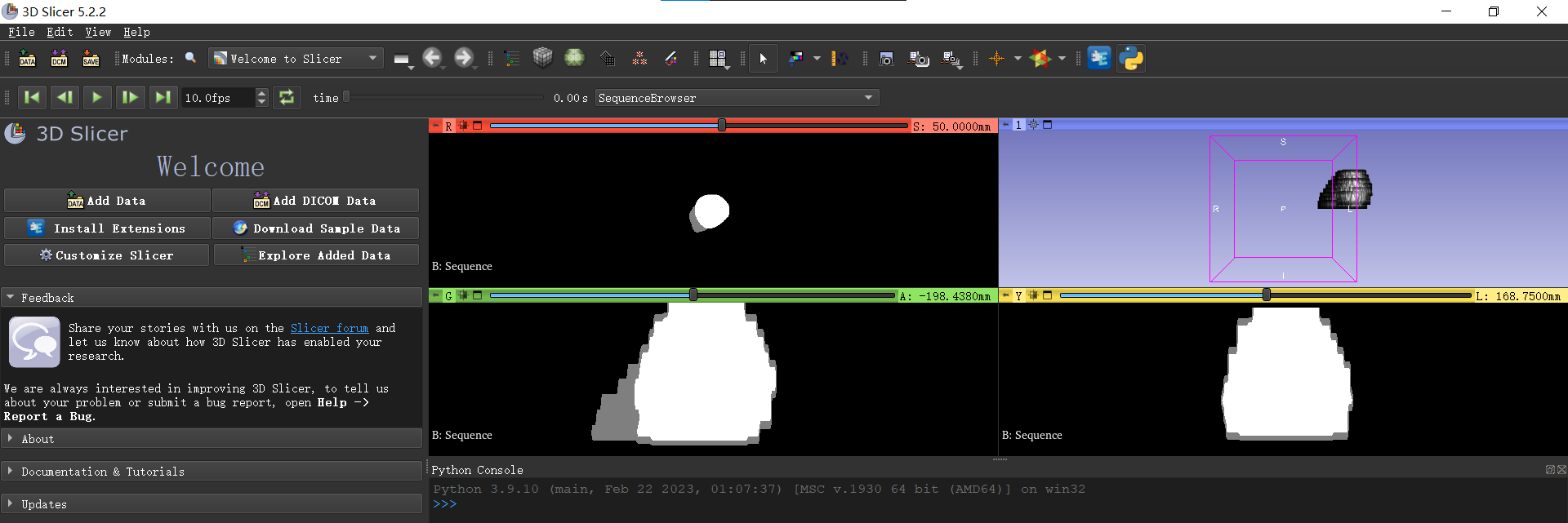
（由于4D文件具有动画效果，以下对相同文件的比较中，将时间轴调为同一时间点）

001文件夹合成效果：

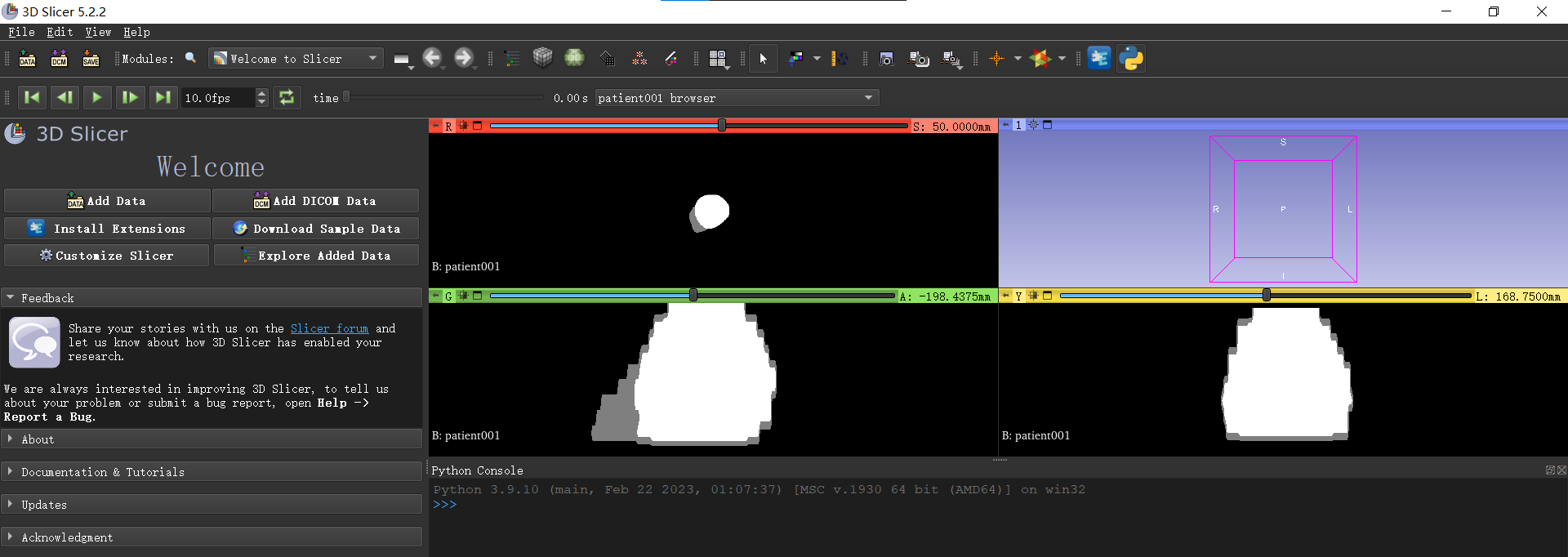
生成两个文件（不相互依赖）



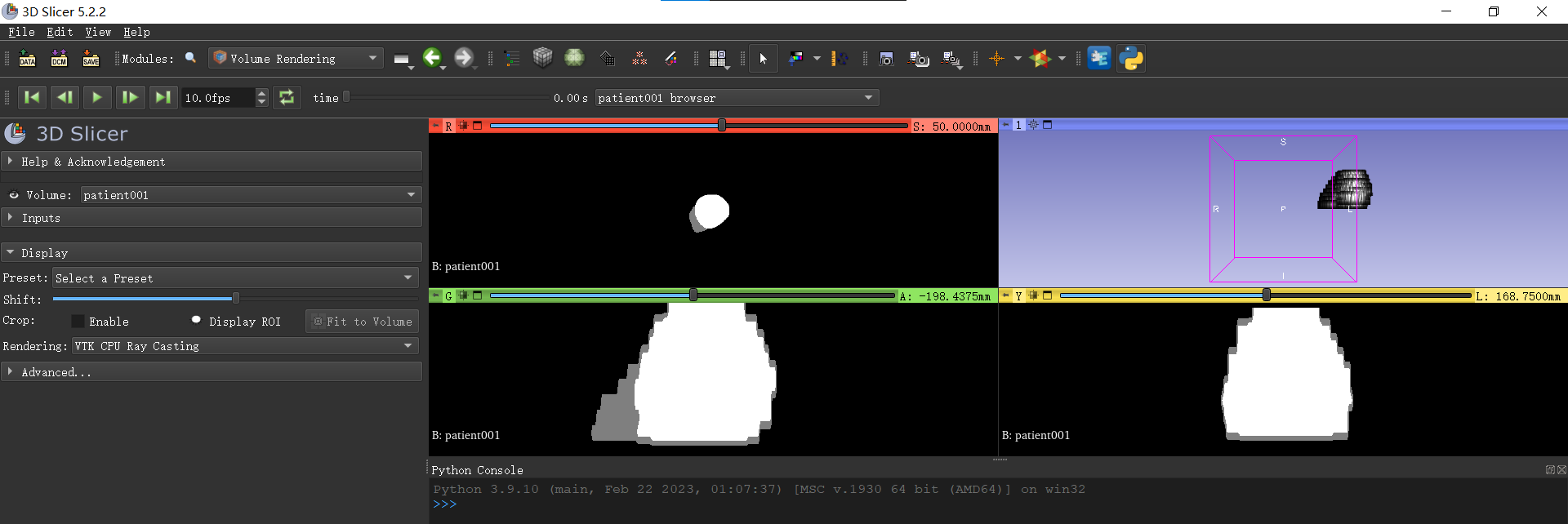
打开patient001.mrb



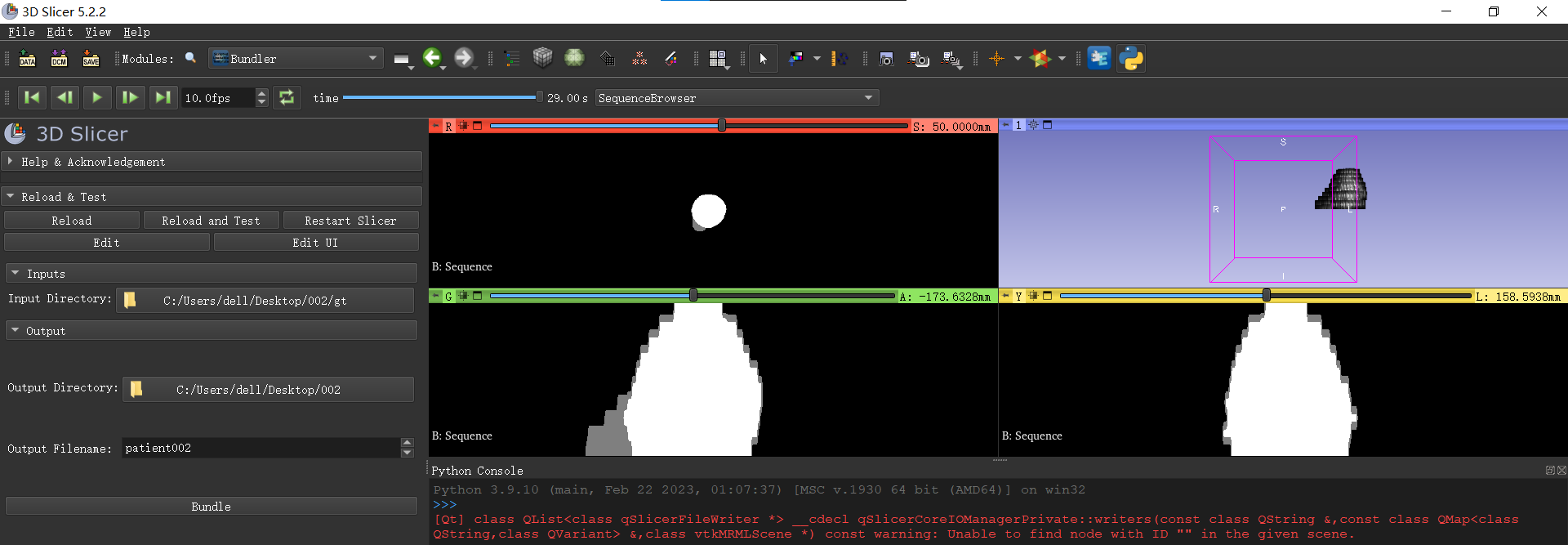
打开patient001.seq.nrrd



Modules->Volume Rendering，将Volume旁的“眼睛”图标设置为睁开状态。

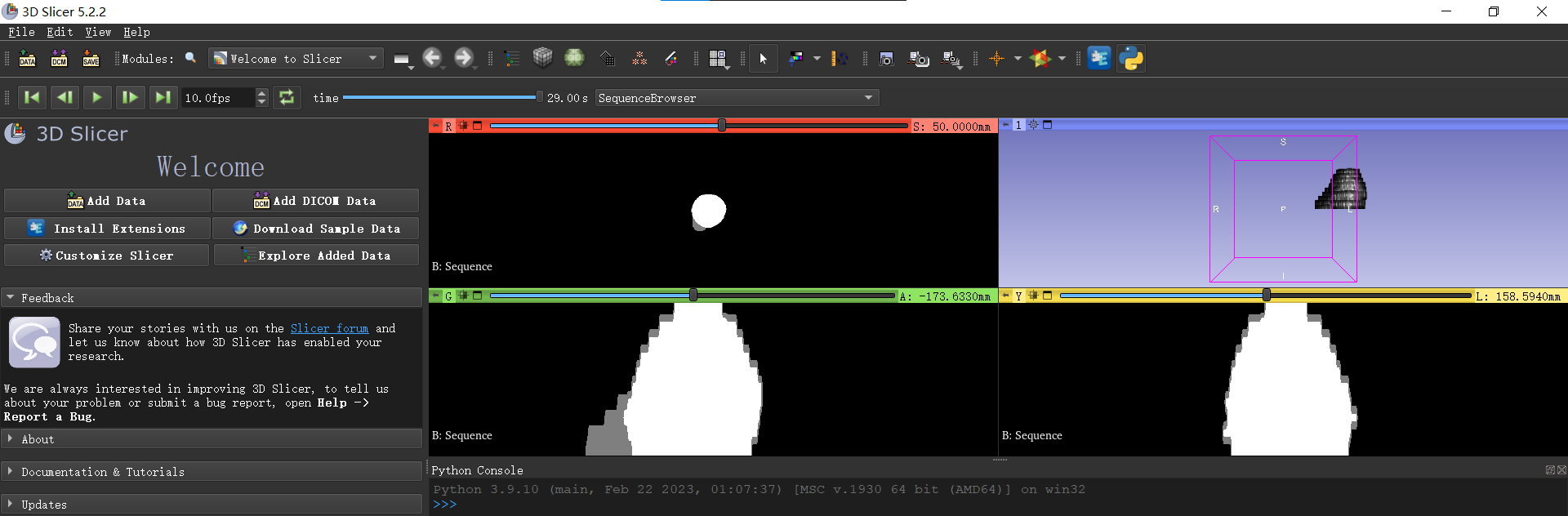


对于002文件夹，附上合成效果：

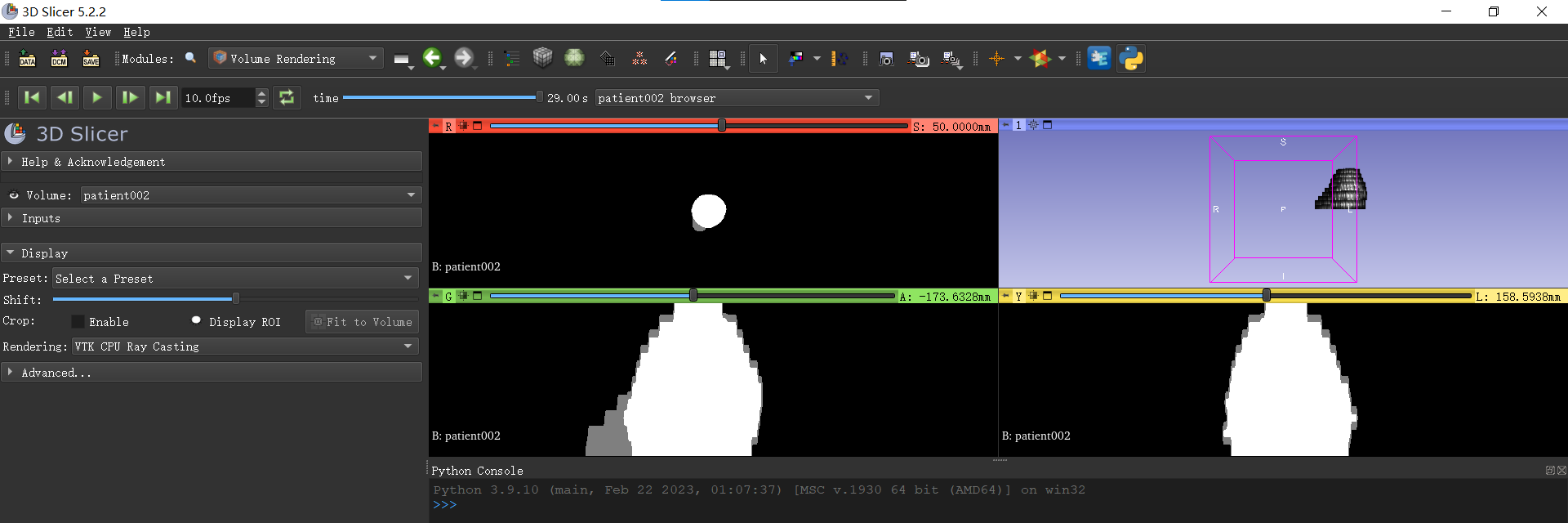




patient002.mrb



patient002.seq.nrrd



拓展的核心逻辑：

    def **process**(self, inputPath, outputPath, outputName):

**"""**

**Run the processing algorithm.**

**Can be used without GUI widget.**

**:param inputPath: directory holds the volumes**

**:param outputPath: directory to hold the sequence file**

**:param outputName: the name sequence saved as**

**"""**

        if not inputPath or not outputPath:

            raise **ValueError**("Input or output path is invalid")

        import **time**

        startTime = **time**.**time**()

**logging**.**info**('Processing started')

***# Define the path to the folder containing the input volumes***

*# fileFolderPath = r"D:\Files\Temp\input\gt"*

        fileFolderPath = inputPath

***# Load the input volumes into Slicer***

        volumeNodes = []

        for fileName in **os**.**listdir**(fileFolderPath):

            filePath = **os**.path.**join**(fileFolderPath, fileName)

            volumeNode = **slicer**.util.loadVolume(filePath)

            volumeNodes.**append**(volumeNode)

***# Create a new sequence node in Slicer and add the loaded volumes to it***

        sequenceNode = **slicer**.mrmlScene.AddNewNodeByClass("vtkMRMLSequenceNode")

        for i, volumeNode in **enumerate**(volumeNodes):

            sequenceNode.SetDataNodeAtValue(volumeNode, **str**(i))

***# Save the sequence node as an nrrd file***

*# nrrdFilePath = os.path.join(r"D:\Files\Temp", "out.seq.nrrd")*

*# mrbFilePath = os.path.join(r"D:\Files\Temp", "out.mrb")*

        nrrdFilePath = **os**.path.**join**(outputPath, outputName+".seq.nrrd")

        mrbFilePath = **os**.path.**join**(outputPath, outputName+".mrb")

***# Create the volume rendering for the sequence browser node***

        vrLogic = **slicer**.modules.volumerendering.logic()

        sequenceBrowserNode = **slicer**.mrmlScene.AddNewNodeByClass("vtkMRMLSequenceBrowserNode")

        sequenceBrowserNode.AddSynchronizedSequenceNode(sequenceNode)

        proxyVolumeNode = sequenceBrowserNode.GetProxyNode(sequenceNode)

        vrLogic.CreateDefaultVolumeRenderingNodes(proxyVolumeNode)

        vrDisplayNode = **slicer**.mrmlScene.GetFirstNodeByClass("vtkMRMLVolumeRenderingDisplayNode")

        volumePropertyNode = vrDisplayNode.GetVolumePropertyNode()

        volumePropertyNode.Copy(vrLogic.GetPresetByName('CT-Cardiac'))

***# Set the volume rendering node to synchronize with volume display node***

        vrDisplayNode.SetFollowVolumeDisplayNode(True)

        vrDisplayNode.SetVisibility(True)

***# Remove the loaded volumes from Slicer***

        for volumeNode in volumeNodes:

**slicer**.mrmlScene.RemoveNode(volumeNode)

***# Set the output file path for the sequence browser node and start playback***

        sequenceBrowserNode.SetSelectedItemNumber(0)

*# sequenceBrowserNode.SetRecordingFilePath(niftiFilePath)*

        sequenceBrowserNode.SetPlaybackActive(True)

**slicer**.util.saveNode(sequenceNode, nrrdFilePath)

**slicer**.util.saveScene(mrbFilePath)

        stopTime = **time**.**time**()

**logging**.**info**(f'Processing completed in {stopTime-startTime:.2f} seconds')