A medical image processing software – Based on ITK-SNAPS

Zhang Ke 16307100128
Feng MengDi 16307100076
Zhang YuQing 16307130308

1 Introduction

1.1 Previous work and our inspiration

Our project was inspired by a software application used to segment structures in 3D medical images called ITK-SNAPS, which is free, open-source, and multi-platform. ITK-SNAPS is the product of a decade-long collaboration between Paul Yushkevich, Ph.D., of the Penn Image Computing and Science Laboratory (PICSL) at the University of Pennsylvania, and Guido Gerig, Ph.D., of the Scientific Computing and Imaging Institute (SCI) at the University of Utah.

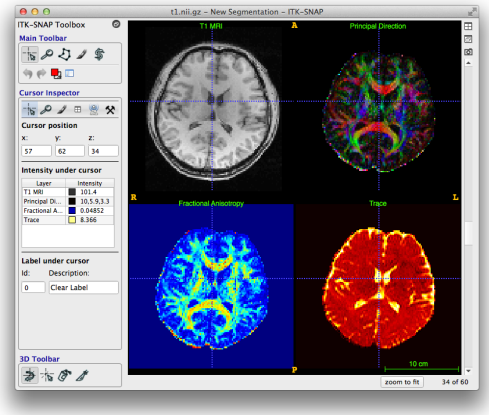

Figure 1: ITK-SNAPS

ITK-SNAP provides semi-automatic segmentation using active contour methods, as well as manual delineation and image navigation. In addition to these core functions, ITK-SNAP offers many supporting utilities. Some of the core advantages of ITK-SNAP include:

- Linked cursor for seamless 3D navigation
- Manual segmentation in three orthogonal planes at once
- A modern graphical user interface based on Qt
- Support for many different 3D image formats, including NIfTI and DICOM
- Support for concurrent, linked viewing, and segmentation of multiple images
- Support for color, multi-channel, and time-variant images

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

- 3D cut-plane tool for fast post-processing of segmentation results
- Extensive tutorial and video documentation

1.2 Brief introduction of our work

In this project ,we aim to create a tool ,like ITK-SNAP, that would be dedicated to two specific function, segmentation and 3D show, and would be easy to use and learn. Besides some basic functions in ITK-SNAPs, our tool is able to color the tumor and show the contour of tumor in both 2D and 3D ways. Our tool provides semi-automatic segmentation using active contour methods, as well as manual delineation and image navigation.It can support for different 3D image formats, including *mha and *nii.

Our design also emphasizes interaction and ease of use, with the bulk of the development effort dedicated to the user interface.Our project is based on Python3 developing environment , and the GUI is implemented by PyQt5 .

2 Function Outline

EXE: pan.Baidu: `https://pan.baidu.com/s/1tLaI9fw0pE0-t3KTDzfvIw`
Extraction code:9bez

Table 1: Function outline

Menu	Function	Notes
File	open	Open file
	save As	Save as *mha
	save As	Save as *nii
	save As	Save segmentation result(*mha)
	Display	Display slices in 3*4 window
Edit	Contrast	Adjust contrast
	Brightness	Adjust brightness
	Maxmize	Adjust size
Segmentation	Outline	Show the contour of the brain
	Manual segmentation	Manually mark the tumor point
	Active manual segmentation	Dynamic display of segmentation results
	Threshold	Gray value binarization
	Manual segmentation outline	Draw segmentation contour
	Active Manual segmentation outline	Dynamic display of segmentation contour
3D	Active 3D	Show 3D segmentation results=
	Colorful 3D	Show 3D structure of brain
	Slicer 3D	Show 3D slices

3 Algorithms of some Functions

3.1 File

3.1.1 open file

In pyqt5,we can use QFileDialog.getOpenFileName to open the file we want.

```
imgName, imgType = QFileDialog.getOpenFileName(self, "", "*.mha;;*.nii.gz
    ;;*.nii;;All␣Files(*)")
```

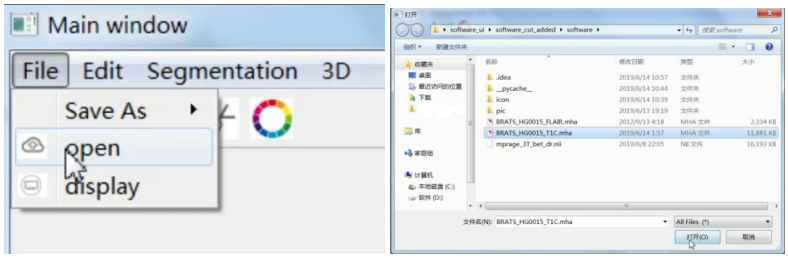

Figure 2: Open file

After opening *mha or *nii file, we need to read them. There are 2 simple ways to read the file, we can either use the package ”SimpleITK” or ”nibable”

```
import nibabel as nib
img = nib.load(filename)
data_array=img.get_fdata()

import SimpleITK as sitk
img = sitk.ReadImage(filename)
data_array = sitk.GetArrayFromImage(img)
```

Then in order to show the slicer ,we need to convert the multidimensional data to three 2D arrays.We select the array for S,A,C side respectively.

Input is the path we choose and output is a two-dimension array.

```
def read_img(path):
    img = sitk.ReadImage(path)
    data = sitk.GetArrayFromImage(img)
    data = data - np.mean(data)
    data = np.divide(data, data.max())
    data = np.multiply(data, 255)
    return data

def return_S_array(path, S):
    data = slice.read_img(path)
    img = data[S, :, :]
    return img
```

See the following interface, in which the lower right corner is a 3D window (you can drag the mouse to rotate and other operations), and the remaining three are three slices based on rectangular coordinate system.

3.1.2 save as

First, we choose a directory to save the file. In pyqt5,we can use QFileDialog.getExistingDirectory to choose the folder we want.

We obtain two format of saving and converting: one is *mha the other one is *nii . we can use ” sitk.WriteImage ” to change the format.

```
sitk.WriteImage(image, os.path.join(directory1, name + ".nii"))
```

3.1.3 Save Segmentation Result

After you have done the segmentation, you may save the result by clicking ”File-> Save Segmentation Result”. Then you could find a *mha file in the directory you choose. In this ”segmentation-result.mha”, the tumor part is painted white.

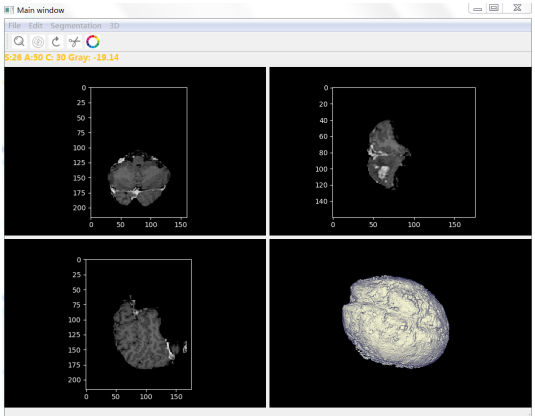

Figure 3: interface

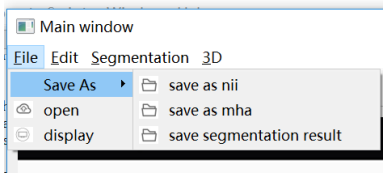

Figure 4: Save file

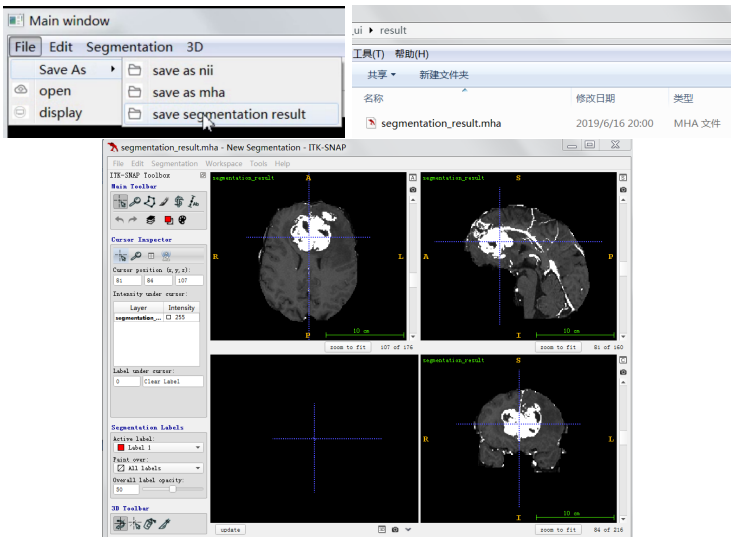

Figure 5: Save segmentation result

3.1.4 Display

n File -> Display, we can jump to the 4* 3 window for a better look and feel. We can choose to get the 12 slicers from 3 directions.

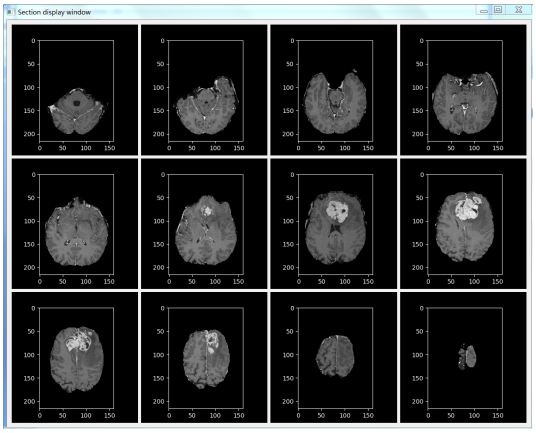

Figure 6: Display slices of specific dimension

3.2 Edit

We can make the following edits to the three static images in order to have a clearer look on the tumor.

3.2.1 Contrast

ALGORITHMS:
Input and output both are two-dimension arrays . We use the " piecewise linear functions " to adjust contrast based on the image grayscale matrix.

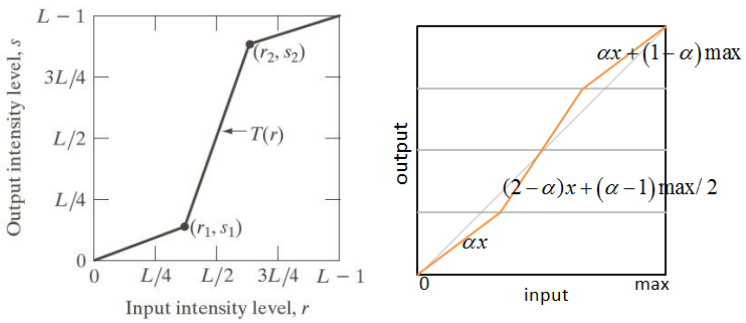

Figure 7: Piecewise linear functions

We conduct symmetry conversion, and divide the grayscale data into 4 parts: max = the max of the grayscales in the array For each grayscale in the array(we just call it "x" here):

```
def contrast(array, alpha):
    newarray = array
    m = array.max()
    for i in range(array.shape[0]):
        for j in range(array.shape[1]):
            if (array[i][j] < m / 4):
                newarray[i][j] = alpha * array[i][j]
```

```
            elif (array[i][j] > m * 3 / 4):
                newarray[i][j] = alpha * array[i][j] + (1 - alpha) * m
            else:
                newarray[i][j] = (2 - alpha) * array[i][j] + (alpha - 1)
                    * m / 2
    return newarray
```

HOW TO USE:
Click "Edit->contrast", and input a contrast you want.(Attention: <1 means decreasing contrast, >1 means increasing contrast). As shown in the figure below, contrast is set to 10.

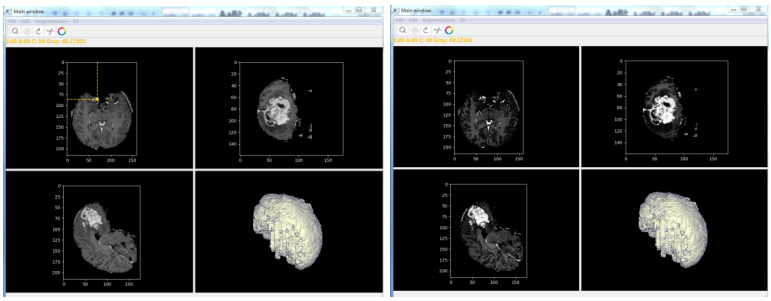

Figure 8: Adjust Contrast

3.2.2 Brightness

ALGORITHMS:
Input and output both are two-dimension arrays . We use the " Power-Law transformation " to adjust contrast based on the image grayscale matrix.

$$s = cr^y$$

```
def bright(array, brightness):
    gamma = 1 / brightness
    max_num = array.max()
    if max_num > 0:
        newarray = max_num * (array / max_num) ** gamma
    else:
        newarray = max_num * (array) ** gamma
    return newarray
```

HOW TO USE:
Click "Edit->brightness", and input a brightness you want. (Attention:0-1 means decreasing brightness, >1 means increasing brightness). As shown in the figure below, brightness is set to 2.

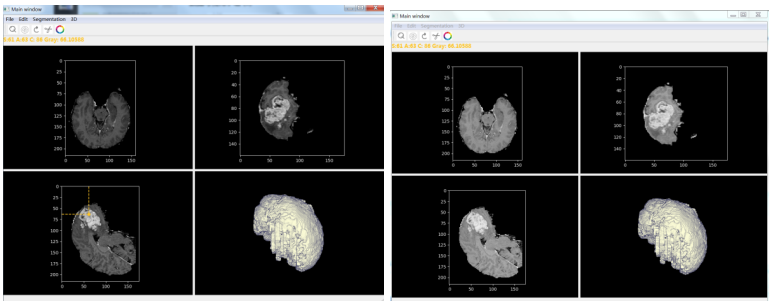

Figure 9: Adjust brightness

3.2.3 Maxmize

ALGORITHMS:
Input and output both are two-dimension arrays. Use "transform.rescale()" to enlarge the picture. Then we refine the enlarged picture in the same original window based on the part we wish to enlarge.

```
def resize(array, alpha):
    newarray = transform.rescale(array, alpha)
    return newarray

def enlarge(x,y, array, alpha):
    newarray = adjust.resize(array, alpha)
    returnarray = newarray[int((alpha - 1) * y):int((alpha - 1) * y) +
        array.shape[0],
                    int((alpha - 1) * x):int((alpha - 1) * x) + array.
                        shape[1]]
    return returnarray
```

Figure 10: Enlage function

HOW TO USE:
First click where you want to enlarge the image. Click the magnifying glass button on the toolbar or Edit->maximize for local magnification. The effect is shown below.

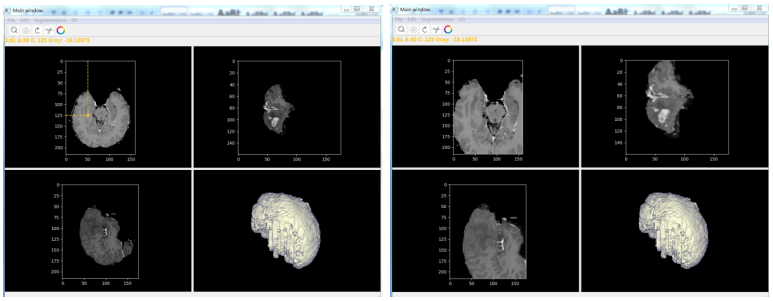

Figure 11: Adjust size

3.3 Segmentation

3.3.1 Selected tumor point

ALGORITHMS: For the array converted from *mha, the array is subtracted from the mean and divided by the maximum value then multiplied by 255, so that the gray values in each dimension of the image are more balanced. **HOW TO USE:** First, switch to the 3d view, and you can adjust the contrast or other functions so that all three views can show clear tumors as much as possible. Hint: How to tell if it's a tumor? – By looking at the grayscale at the top left coordinate, the gray scale at the tumor part will be larger (typically >100). Click the switch mode button to switch to manual annotation mode.

Figure 12: Switch message

After seeing the following mode, click the "color" button, and select the color you like. Click the tumor location to mark them. Hint: You need to click on all 3 sides and you may click a few more points.

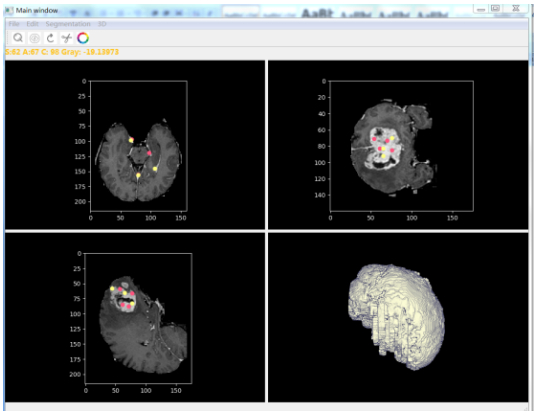

Figure 13: Manual mark points

3.3.2 Manual Segmentation

ALGORITHMS:

- We first conduct a equalization in grayscale data.

```
data = data - np.mean(data)
data = np.divide(data, data.max())
data = np.multiply(data, 255)
```

- Input a threshold. Then we can get the tumor range.
  If the max of grayscale of our click is 100, and threshold we input is 40. Then the grayscale of tumor is 60-140

```
tumor_range = [max([self.targets, self.targeta, self.targetc]) -
    self.manualsegthresh,
max([self.targets, self.targeta, self.targetc]) + self.
    manualsegthresh]
```

- Based on the tumor range, we can draw scatter points on it.

```
def manualseg_scatters(self):
        self.replots()
        nrows = self.s.shape[0]
        ncols = self.s.shape[1]
        x_scatters = []
        y_scatters = []
        for x in range(nrows):
            for y in range(ncols):
                if abs(self.s[x][y] -max([self.targets,self.
                    targeta,self.targetc])) < self.manualsegthresh:
                    x_scatters.append(y)
```

```
                        y_scatters.append(x)
            self.axs.scatter(x_scatters, y_scatters, c=self.color,s=1)
            self.axs.figure.canvas.draw()
```

HOW TO USE: After the dot notation, click "Segmentation->manual segmentation", and input the threshold(usually pick 30-50). Generate segmentation image after confirmation. Now that the range of tumor has spread to the background, other functions can be explored.

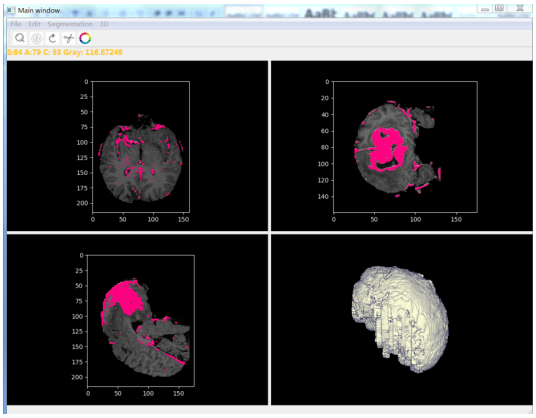

Figure 14: Manual segmentation

3.3.3 Generate Manual Segmentation Gif

HOW TO USE: Click Segmentation->Active Manual Segmentation, then you can see the gifs generated by drawing points in the tumor area.

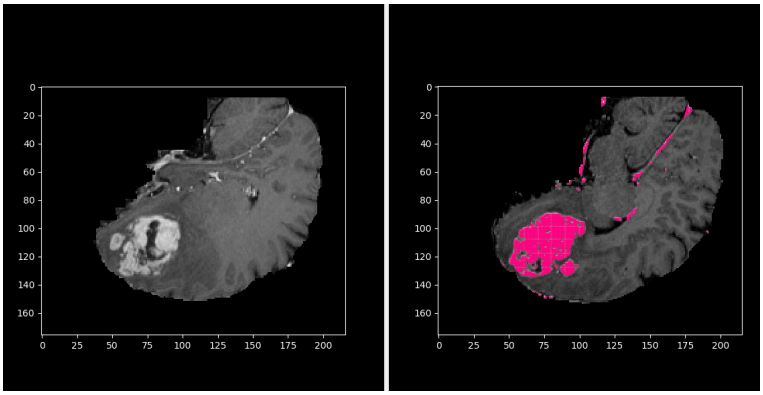

Figure 15: Manual segmentation gif

3.3.4 Showing tumor contour

ALGORITHMS: Basically, the algorithms for this part is quite similar with "outline ". And we can draw scatter points on the tumor contour.

```
def outline_scatters(self):
       self.replots()
       nrows = self.tmps.shape[0]
       ncols = self.tmps.shape[1]
       x_scatters = []
       y_scatters = []
```

```
        for x in range(nrows):
            for y in range(ncols):
                if self.tmps[x][y] >120:
                    x_scatters.append(y)
                    y_scatters.append(x)
        self.axs.scatter(x_scatters, y_scatters, c=self.color,s=0.1)
        self.axs.figure.canvas.draw()
```

HOW TO USE: Click "Segmentation -> Manual Segmentation Outline". There is no color inside the colored spot area. The edges of the painted area remain colored so that the edges of the tumor are visible.

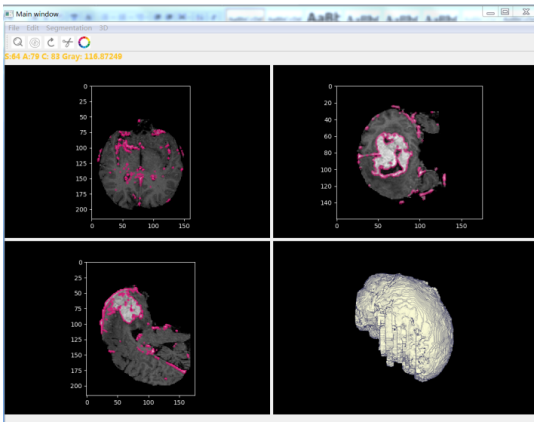

Figure 16: Manual Segmentation contour

3.3.5 Showing tumor contour Gifs

HOW TO USE: Click "Segmentation->Activate Manual Segmentation outline", then you can see the gifs generated by drawing the tumor contour.

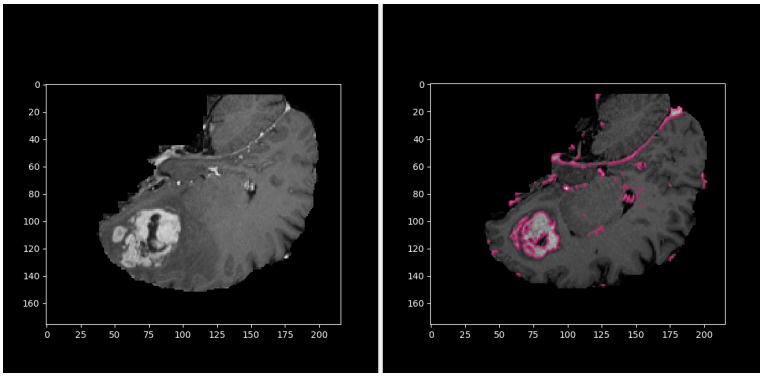

Figure 17: Activate Manual Segmentation outliner

3.3.6 Outline

ALGORITHMS: Aiming to obtain the clear contour of the brain structure:

- Average the difference between adjacent gray values in x and y dimensions.
- Calculate the square root of the sum of square

```
def outline(gray_array):
    rows, cols = gray_array.shape
    mag = np.zeros([rows, cols], np.float)
    for x in range(1, rows - 1):
        for y in range(1, cols - 1):
            gx = 0.5 * gray_array[x + 1, y] - 0.5 * gray_array[x - 1, y]
            gy = 0.5 * gray_array[x, y + 1] - 0.5 * gray_array[x, y - 1]
            mag[x, y] = np.power(gx * gx + gy * gy, 0.5)
    return mag
```

HOW TO USE: Click "Segmentation->outline", we can see the contour of the brain in black and white.

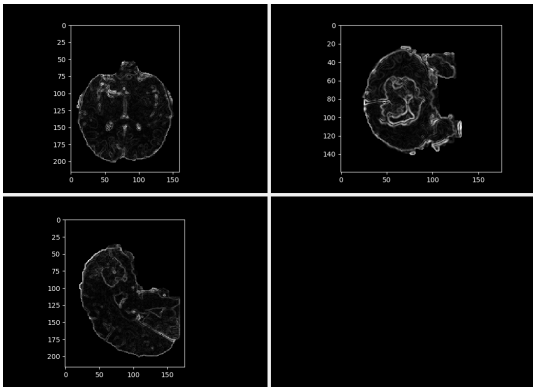

Figure 18: Activate Manual Segmentation outliner

3.3.7 Threshold

ALGORITHMS Use the threshold package of OpenCV to carry out this function:

```
def threshold(img,th,choice="TOZERO"):
    #img = histogram_balanced(img)
    if choice=='BINARY':
        ret, thresh = cv2.threshold(img, th, 255, cv2.THRESH_BINARY)
    if choice == 'BINARY_INV':
        ret, thresh = cv2.threshold(img, th, 255, cv2.THRESH_BINARY_INV)
    if choice == 'TRUNC':
        ret, thresh = cv2.threshold(img, th, 255, cv2.THRESH_TRUNC)
    if choice == 'TOZERO':
        ret, thresh = cv2.threshold(img, th, 255, cv2.THRESH_TOZERO)
    if choice == 'TOZERO_INV':
        ret, thresh = cv2.threshold(img, th, 255, cv2.THRESH_TOZERO_INV)
    return thresh
```

HOW TO USE: Click "Segmentation-> threshold", we can see the contour more clearly. Switch to 3D mode and click a point in the figure (such as tumor point) before beginning to "threshold segmentation". You can set the threshold through the gray value of this point

3.4 3D Display

3.4.1 colourful 3D

ALGORITHMS: We use MarchingCubes algorithms to paint the 3D image.

- First we need to find the distribution of points in the brain.
- Then find the contour lines to draw.
- Use marching cubes algorithm to carry out surface rendering

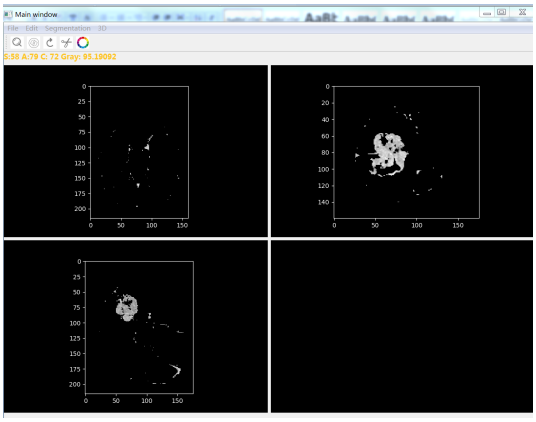

Figure 19: Threshold result

- Concatenating Triangle Strips
- Mapper is established to map geometric data to image data
- set the property(such as color, opacity, ambient, diffuse, specular and so forth)
- set background

We draw a histogram to grasp the distribution of brain point values.

- Flatten the array of brain image
- Plot histogram of the flatten list and find the top 3 highest bins

```
def show_3D_brain(path,colorlist = [round(random.random(),3),round(random
    .random(),3),round(random.random(),3),0.2]):
    img = sitk.ReadImage(path)
    dims = img.GetSize()
    spacing = img.GetSpacing()
    brain_data = sitk.GetArrayFromImage(img)

    image = vtk.vtkImageData()
    image.SetDimensions(dims[2],dims[1],dims[0]) # set dimension
    image.SetSpacing(spacing[0],spacing[1],spacing[2]) # set spacing in X
        ,Y,Z
    image.SetOrigin(0,0,0)

    if vtk.VTK_MAJOR_VERSION <= 5:
        image.SetNumberOfScalarComponets(1)
        image.SetScalarTypeToDouble()
    else:
        image.AllocateScalars(vtk.VTK_DOUBLE,1)

    # find distribution of brain
    n1, bins1, patches1 = plt.hist(brain_data.flatten())
    plt.clf()
    n1 = list(n1)
    bins1 = list(bins1)
    max_index1 = n1.index(max(n1))
    # find equal value line
    brain_isopleth = int((bins1[max_index1] + bins1[max_index1 + 1]) / 2)

    for z in range(dims[0]):
        for y in range(dims[1]):
            for x in range(dims[2]):
                scalardata = brain_data[x][y][z]
                image.SetScalarComponentFromDouble(x,y,z,0,scalardata)
```

```
32
33      # marching cubes
34      Extractor = vtk.vtkMarchingCubes()
35      Extractor.SetInputData(image)
36      Extractor.SetValue(0,brain_isopleth)
37
38      # triangle connection
39      stripper = vtk.vtkStripper()
40      stripper.SetInputConnection(Extractor.GetOutputPort())
41
42      #mapper setting
43      mapper = vtk.vtkPolyDataMapper()
44      mapper.SetInputConnection(stripper.GetOutputPort())
45      actor = vtk.vtkActor()
46      actor.SetMapper(mapper)
47
48      # property setting
49      actor.GetProperty().SetColor(colorlist[0],colorlist[1],colorlist[2])
50      actor.GetProperty().SetOpacity(colorlist[3])
51      actor.GetProperty().SetAmbient(0.25)
52      actor.GetProperty().SetDiffuse(0.6)
53      actor.GetProperty().SetSpecular(1.0)
54
55      #scene setting
56      ren = vtk.vtkRenderer()
57      ren.SetBackground(0,0,0)    # set backgroundcolor
58      ren.AddActor(actor) # Add actor
59      return ren
60  cv2.threshold(img, th, 255, cv2.THRESH_TOZERO_INV)
61      return thresh
```

HOW TO USE: Click ”3D-> colourful 3D”, You get a rough 3D map of the tumor on the left and a detailed 3D map of the brain on the right. You are allowed to execute this operation as long as you open the file. Hint: the color is randomly generated, and if the GIF color is ”extremely” ugly, please don't mind to try again.

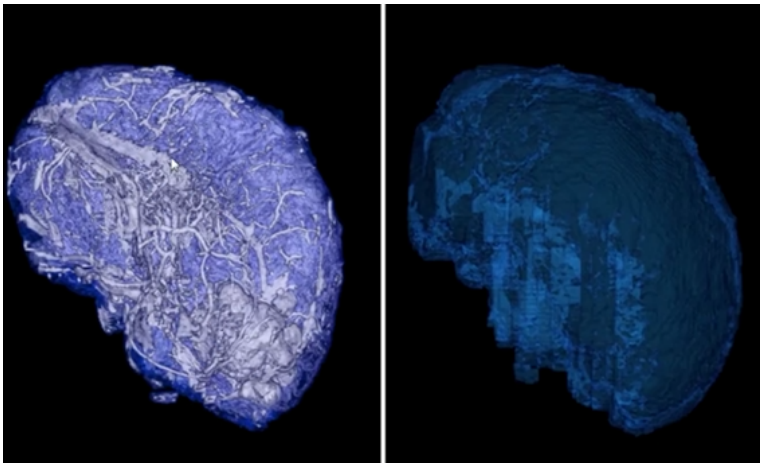

Figure 20: Rough 3D map

3.4.2 Active 3D

ALGORITHMS: Similarly we use Marching Cubes algorithms to paint the 3D image. And in this part, we receive the tumor points obtained by manual segmentation, map these points onto the original image and get their values to find the largest distribution, which is the location of the tumor. **HOW TO USE:** This step is generated after the segmentation. Click ”3D-> Active 3D”, select global color and transparency and tumor color and transparency,

and wait to generate a GIF later. You can also choose to or not to save the 3D result. If

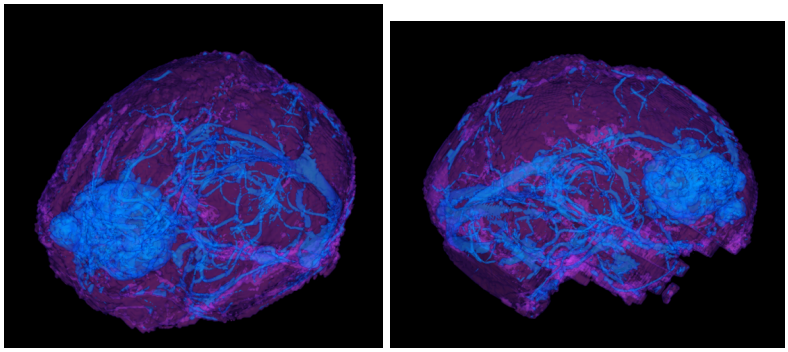

Figure 21: 3D segmentation map

you input "yes", and you are expected to find a "brain.stl" and a "tumor.stl" in your chosen folder.

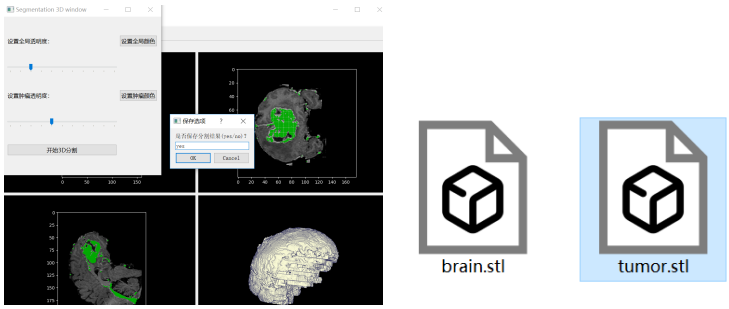

Figure 22: Save 3D result

3.4.3 Slicer 3D

ALGORITHMS: The algorithms is quite similar with above, but we can choose the range of A,C,S side, so that we can see the thin slice in 3D.

```
# if S has been moved, A cannot be moved
if S_low != 0 or S_up != dims[0]:
    A_low = 0
    A_up = dims[1]
elif (int(float(A_low)) >= int(float(A_up))) or (int(float(A_low)) < 0)
    or (int(float(A_up)) > dims[0]):
    A_low = 0
    A_up = dims[1]
else:
    A_low = int(float(A_low))
    A_up = int(float(A_up))

# if S or A has been moved,C cannot be moved
if  (S_low != 0 or S_up != dims[0]) or (A_low != 0 or A_up != dims[1]):
    C_low = 0
    C_up = dims[2]
elif (int(float(C_low)) >= int(float(C_up))) or (int(float(C_low)) < 0)
    or (int(float(C_up)) > dims[0]):
    C_low = 0
    C_up = dims[2]
else:
    C_low = int(float(C_low))
    C_up = int(float(C_up))
```

HOW TO USE: Click "3D-> Slicer 3D". Set the maximum and minimum values for each side when you see the following screens. The smaller the difference of cutting surface, the thinner the slice. After confirming the slicer segmentation, 3D slices can be seen.

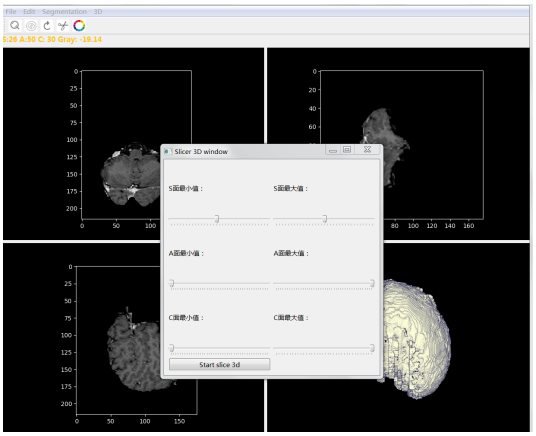

Figure 23: 3D slicer setting

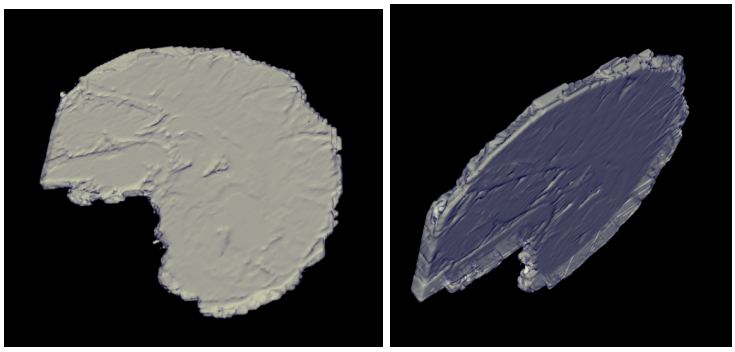

Figure 24: 3D slicer

4 User guide

We move this part to the file readme.pdf.

5 Conclusion and Expectation

5.1 Project Feature

- Complete and good-looking UI – Provides various forms of interaction.
- Multiformat read-write – Provides read-write in *Nii, *Mha format.
- Excellent capacity of tumor-localizing – It can be useful in treating tumors and other diseases
- Combine 2D and 3D – There is a comprehensive display of 2D and 3D, displaying image files from all aspects

5.2 Future Plans

- Optimize user experience – Simplify the logic of the interactive system and optimize the algorithm to make the operation smoother.

- Neural network segmentation – If there is an appropriate training set, automatical neural network segmentation can be reached.
- More functions – Such as 2D ranging, 3D ranging, image annotation and so forth.

6 Group division of work

EXE: pan.Baidu: `https://pan.baidu.com/s/1tLaI9fw0pE0-t3KTDzfvIw`
Extraction code:9bez

- **Zhang Ke:**
  Codes: UI and segmentation part Others: Report

- **Feng Mengdi:**
  Codes: Image reading and 3D display part Others: PPT

- **Zhang Yuqing:**
  Codes: Image adjustment Others: Report, showing video and user's manual

7 References

- ITK SNAP Tutorial http://www.itksnap.org/pmwiki/pmwiki.php
- 3D Slicer Tutorial https://www.slicer.org/
- A deep learning approach for brain tumor MRI segmentation https://github.com/Issam28/Brain-tumor-segmentation
- Lorensen VTK example https://lorensen.github.io/VTKExamples/site/Python/
- Draw with VTK and Python https://cloud.tencent.com/developer/news/386145
- VTK draws contours https://blog.51cto.com/weiyuqingcheng/2113496
- 3D reconstruction of medical images based on surface rendering technology https://www.docin.com/p-2021064302.html
- PyQt5-Chinese-tutorial https://github.com/maicss/PyQt5-Chinese-tutorial