K.N. Toosi University
of Technology
1928

Machine Vision &
Medical Image
Processing Lab.
2000

User Instruction Manual

How to Display and Analyze Medical Images in MedVisPy

Version 2.0.5

Presented by
Kimia Afshari, Amirreza Kazemloo
medvispy@gmail.com
mvmip.ee.kntu.ac.ir, medvispy.ee.kntu.ac.ir

July 2023

This page intentionally left blank.

Software Description

Descriptions

MedVisPy is a software platform for the analysis and rendering of three-dimensional medical images and for research in image guided therapy. It is a product of Machine Vision and Medical Image Processing (MVMIP) Lab. from Faculty of Electrical and Computer Engineering at K.N. Toosi University of Technology.

The preliminary version of this software (0.1.0) has been released in 12/30/2020. This document is for software version **2.0.5**.

New Features

- **Tools**
  A new "Tools" option has been added to the menu. It incorporates various modules for neonatal brain image analysis including (see pages 27-34 in this manual for more details):
  – **MR atlas creation**: constructs probability maps of infant cranial bone, scalp, and CSF, which can be employed to extract the skull from MR cerebral images.
  – **Fast thresholding**: extracts different components from the result of the FSL[1] program's Fast module.
  – **Post processing**: applies morphological and Gaussian filters to medical images (e.g., the output of Fast thresholding) for smoother results.

- **3D visualization improvement:**
  The software's 3D visualization module has been upgraded as follows:
  – The software now automatically detects the range of image values and scales it to the normal range for better graphical 3D visualization.
  – The value of three sources of light (Ambient, Diffuse, and Specular) can now be tuned to improve the depth and shadowing of 3D rendering.
  – The 3D rendering method has been simplified to reduce GPU and CPU usage.
  – The displacement and rotation of 3D renderings are now smoother and faster.

[1] Smith, S. M. (2002). Fast robust automated brain extraction. Human brain mapping, 17(3), 143-155. https://doi.org/10.1002/hbm.10062

System Requirements

This platform supports Linux 16.04 and later and Windows 10 operating systems with Minimum Ram of 4 GB.
Default plugins need also Microsoft Visual C++ Redistributable for running on windows. This will be downloaded through installation.

This page intentionally left blank.

Table of Content

Illustration Index

This page intentionally left blank.

Introduction

The MedVisPy desktop software, developed in Python, is designed to assist clinicians and medical specialists in analyzing and processing medical (brain and lung) images. It includes an IPython console that enables users to execute tasks by calling corresponding commands, as well as a batch processing feature for more efficient data processing. The software also allows to add custom plugins as an internal extension. It provides a user-friendly interface that simplifies the processes and eliminates the need for switching between various programs.

There are two versions available: MedVisPy-B for neonatal brain MRI segmentation and MedvisPy-L for lung CT images processing. The researchers who use MedVisPy-B are encouraged to refer and cite the following paper for more information:

E. Hokmabadi et al., "Skull and scalp segmentation in neonatal cerebral MRI using subject-specific probability models" Bioarxiv, 2023.

The researchers who use MedVisPy-L are encouraged to refer and cite the following paper for more information:

N. Delfan et al., "CT-LungNet: A Deep Learning Framework for Precise Lung Tissue Segmentation in 3D Thoracic CT Scans", Arxiv, 2023.

To download the installation and sample data files, please consult the software webpage at:

https:\\medvispy.ee.kntu.ac.ir

To download the software, you are requested to fill and submit a simple form to MedVisPy group. After completion and submission of the form, you will receive an email containing the links to download MedVisPy2.0.5 installation file and MedVisPy sample data files. By default, the MedVisPy installation file will install the software in the following path in Windows:

C:\Program Files (x86)

You are also encouraged to download MedVisPySampleData.rar file and extract it in an arbitrary (Read/Write) Path as follows:

SampleData_Path\MedVisPySampleData

This folder includes two subfolders containing neonatal cerebral MRI and adult Thoracic CT scans, respectively:

SampleData_Path\MedVisPySampleData\Brain
SampleData_Path\MedVisPySampleData\Lung

MedVisPy GUI

The following figure shows the graphical interface of MedVisPy software. As illustrated, there are several tabs on the top row of the GUI, and some icons in the middle and left side which correspond to the graphical tools. The command console is placed at the bottom row of the GUI. Finally, the graphical window is placed at the right side and above the command console.

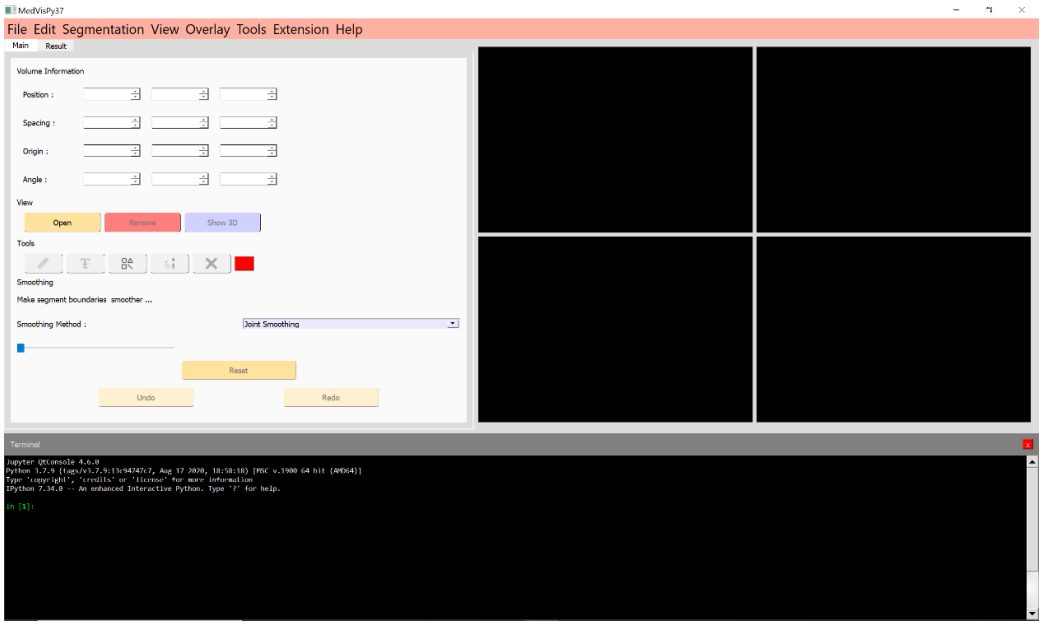


Illustration 1: Graphical interface of MedVisPy software

The different parts of the software including the menu tabs, the graphical icons and command console are described in the following sections.

Display Images

Open File

To read and display DICOM or NifTi file image you can select file path of the image.

As shown below in menu in tab “File” select “Open File”, then a dialog will open and select your path.

With Ctrl+Shift+O also it works.

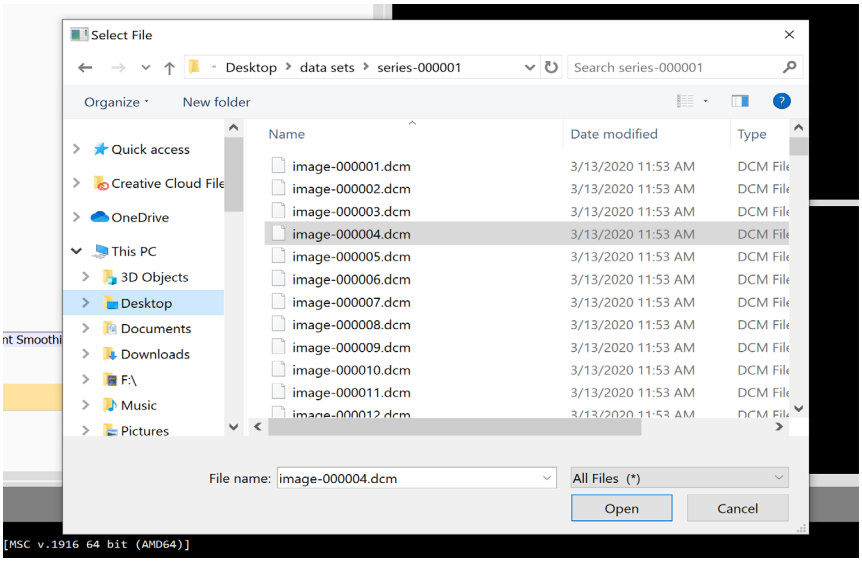


Illustration 2: File selection with GUI

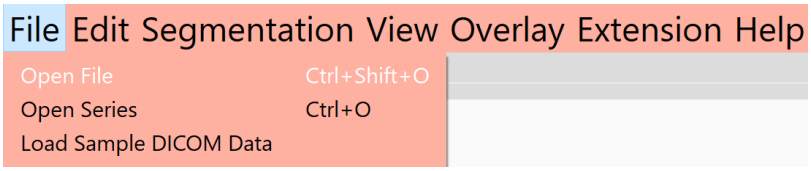


Illustration 3: File opening with GUI

You can simply use command line to open a volume with “openFile “command:

```
handler.openFile(file path)
```

Open Data

To display DICOM series you can select directory of **DICOM** volume images.

You have multi choices. As shown below in menu in tab “File” select “Open Data”, then a dialog opens and you may choose your directory. Or you can open by clicking on “Open” button in the panel.

It also works with Ctrl+O.

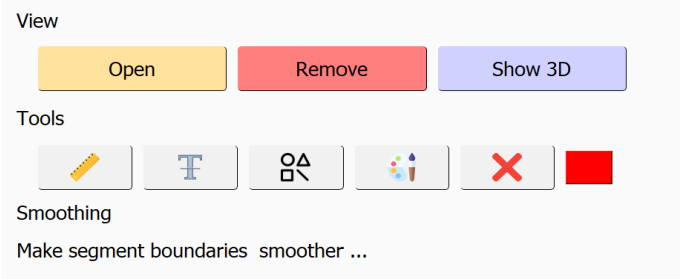


Illustration 4: Data opening with GUI

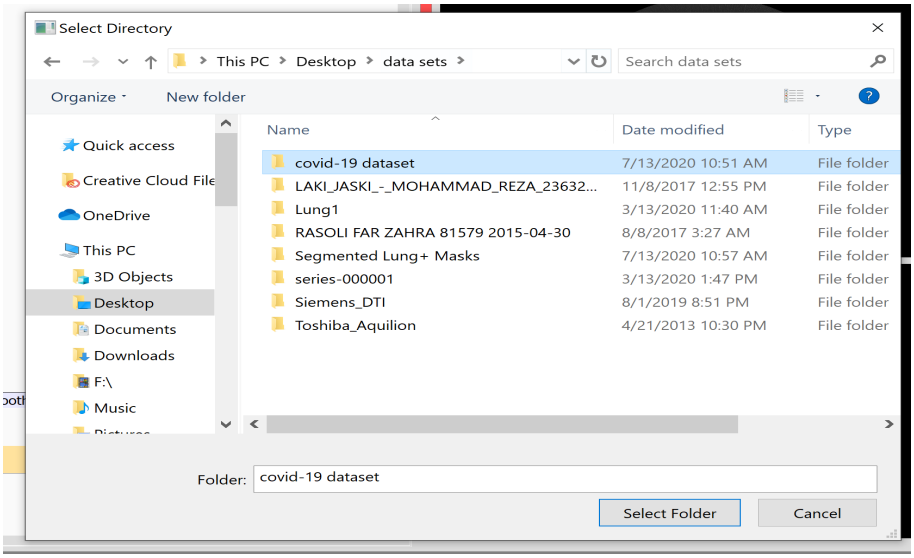


Illustration 5: Directory selection with GUI

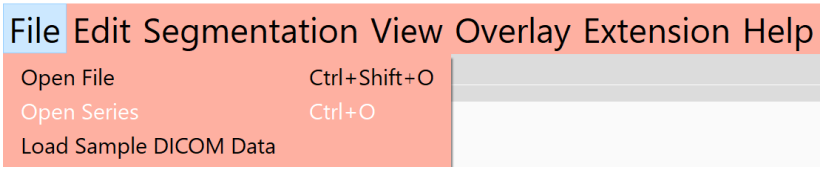


Illustration 6: Series opening with GUI

You can simply use command line to open a volume with " openData " command :

```
handler.openData(directory  path)
```

Show 3D

To have 3D view of opened image data you should call show_3d method as command or use "Show 3D" button in the left panel.

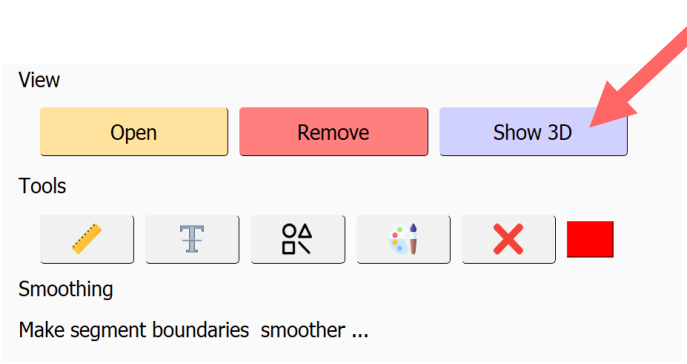
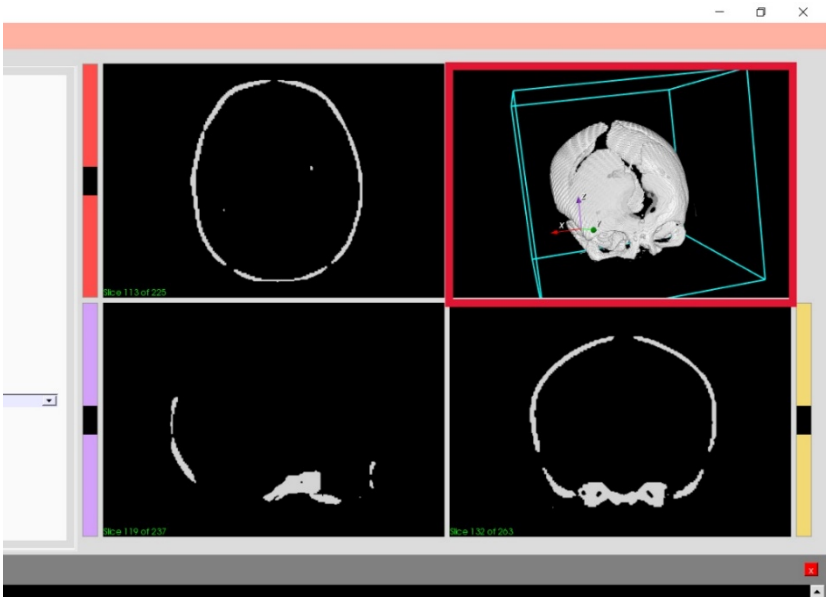


Illustration 7: 3D display with GUI

With command:

```
handler.show3D()
```

The 3D image will be shown in right-upper cell that would be similar below:



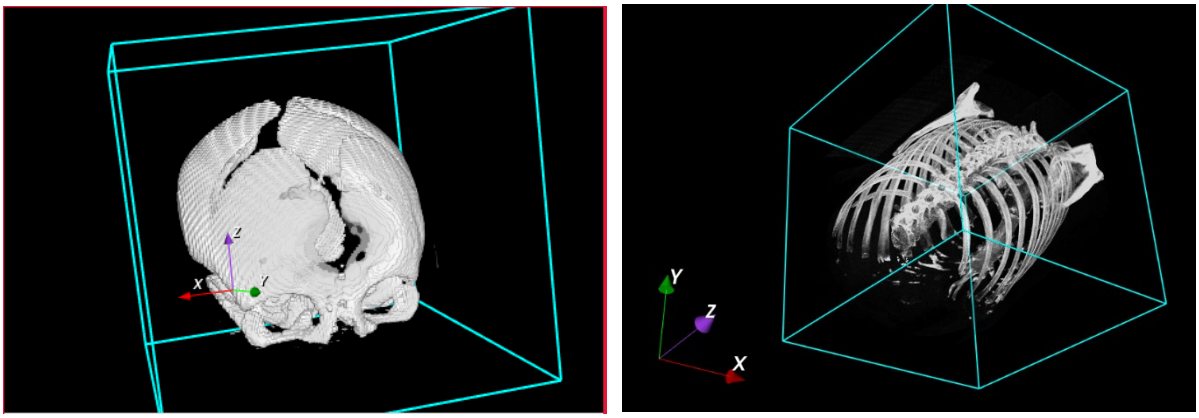


Illustration 8: 3D display

Save Volume

If you want to save your volume you can select “save” item from menu or use command line.

First go tab ”File” choose “Save” and then select the directory to which you want to be saved.

You can also use Ctrl+S key.

For using with command, you should write “saveData” :

```
handler.saveVolume(directory | file path)
```

*Notice that the volume is by default saved in .nii format

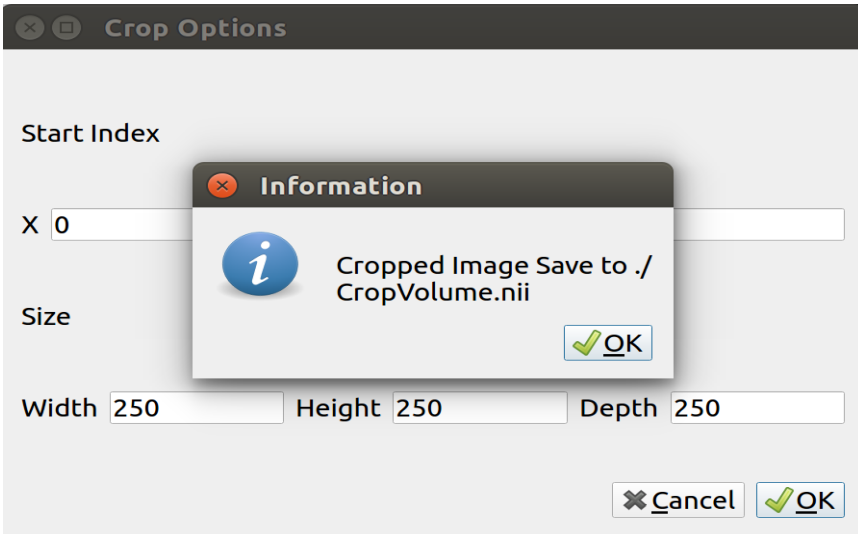


Illustration 9: Data saving

Save Meta

If you want to save your meta information such as added measure tools, points, draws on images, you can select “save meta” item from menu or use command line. Meta is type of .mhd or .mha. by default, it is saved in .mhd format or you can save your meta with .mha with command line. For this purpose give your desired directory and function’s input, then the filename and it’s extension.

First select tab ”File” choose “Save Meta” and then select the directory to which you want to be saved.

For using with command, execute “saveMeta” shown below:

```
handler.saveMeta(directory | file path)
```

Filters and Processes

Volume Information

When an image data is opened, in the panel at the left side of the software, there is a volume info section that displays opened volume slice, spacing and origin. By this section boxes you can simply change and see your desired property (shown by the arrow in illustration 9).

For changing slice, you can also do with each window sliders (shown by the arrow in illustration 10).

There is also available orientation for 3D volume that will be displayed when you show 3D volume.

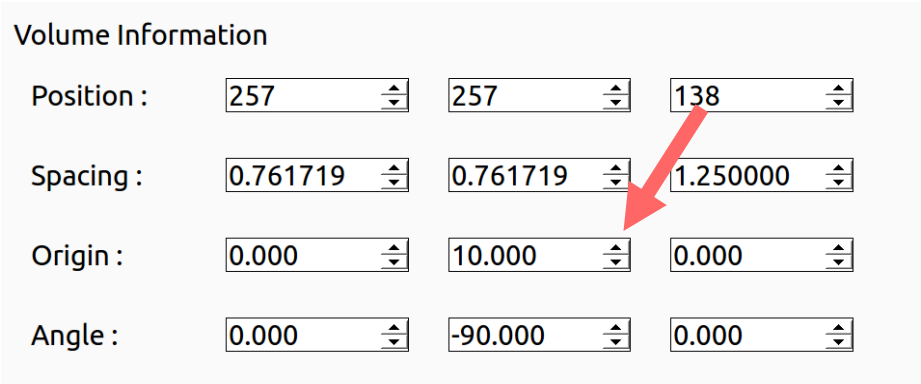


Illustration 10: Change slices origin by panel boxes

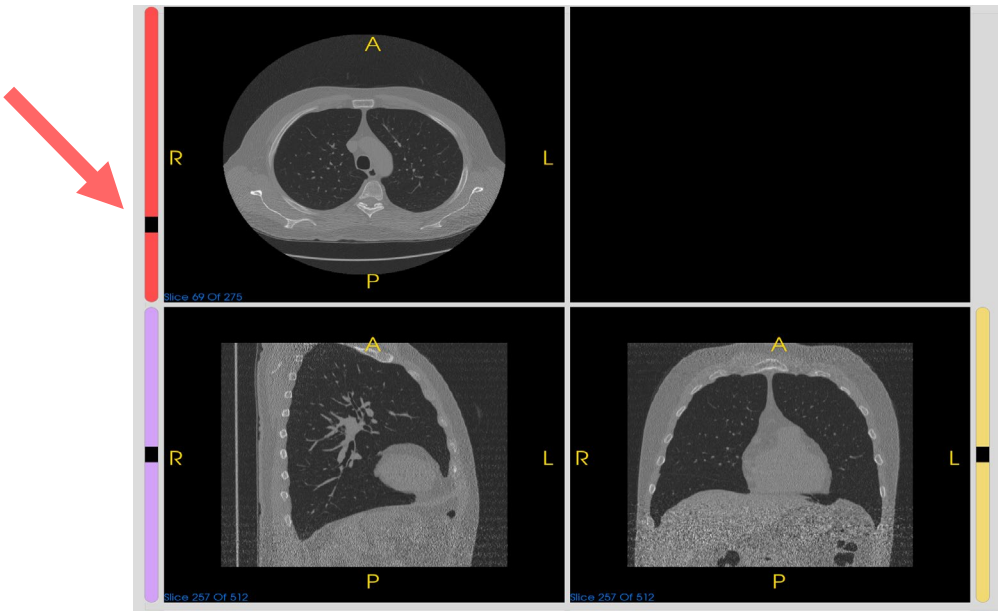


Illustration 2: Moving through slices by window sliders

Crop Volume

Sometimes we have a data and only want to process a special part of it so it's better not to study the whole data. For more speed you can use only your region of interest. It enables you to cut your region of interest (ROI) and process it.

In MedVisPy you can simply do it from "Edit" menu tab.

You can also use Ctrl+C key.

For using with command, you may write "cropData":

```
handler.cropVolume(start index , size)
```

An example has been shown on next page.

Example:

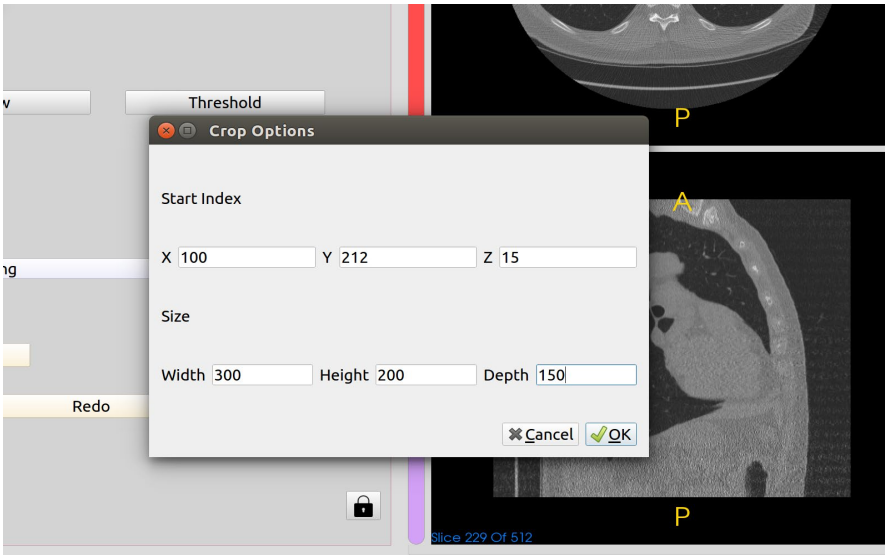


Illustration 3: ROI selection with GUI

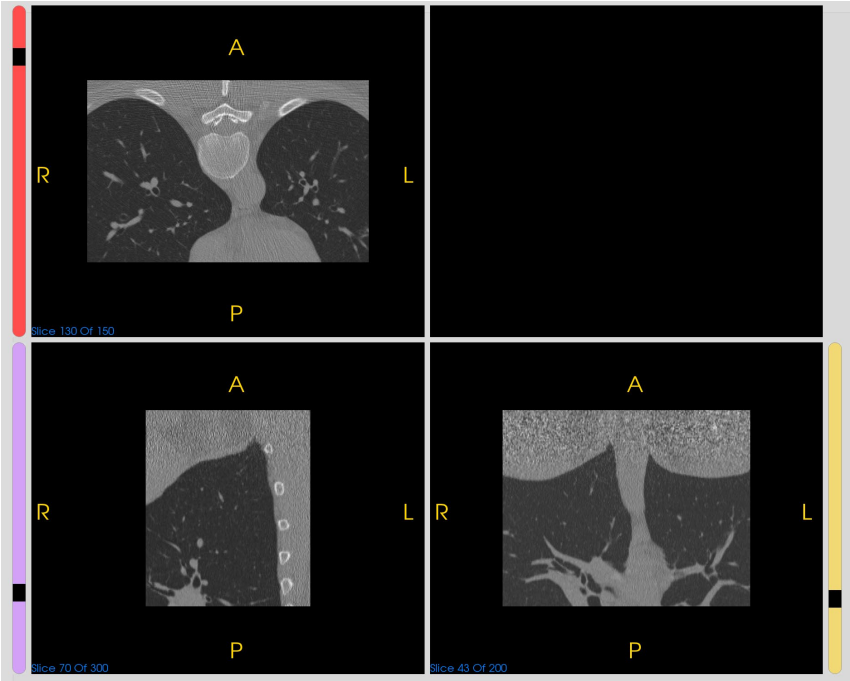


Illustration 4: ROI output

Otsu Thresholding

Otsu thresholding is a well-known operation for segmenting images. It automatically finds the threshold and applies to the current image. It is especially good for primary lung segmentation.

You can choose from "Segmentation" menu tab.

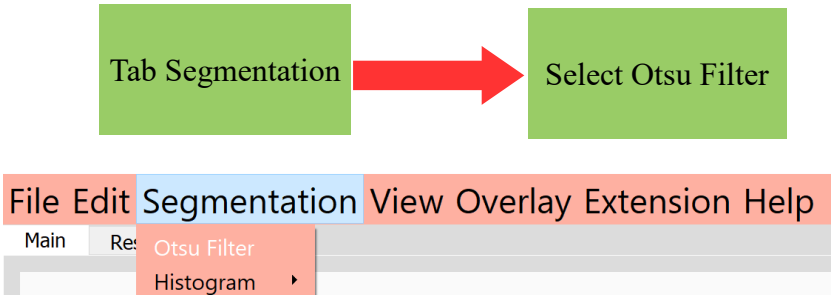


Illustration 5: Otsu thresholding with GUI

For using with command you should write "otsuThreshold":

```
handler.otsuThreshold()
```

In the following, first a sample lung image is displayed and then the segmented image is shown as a result of Otsu thresholding.

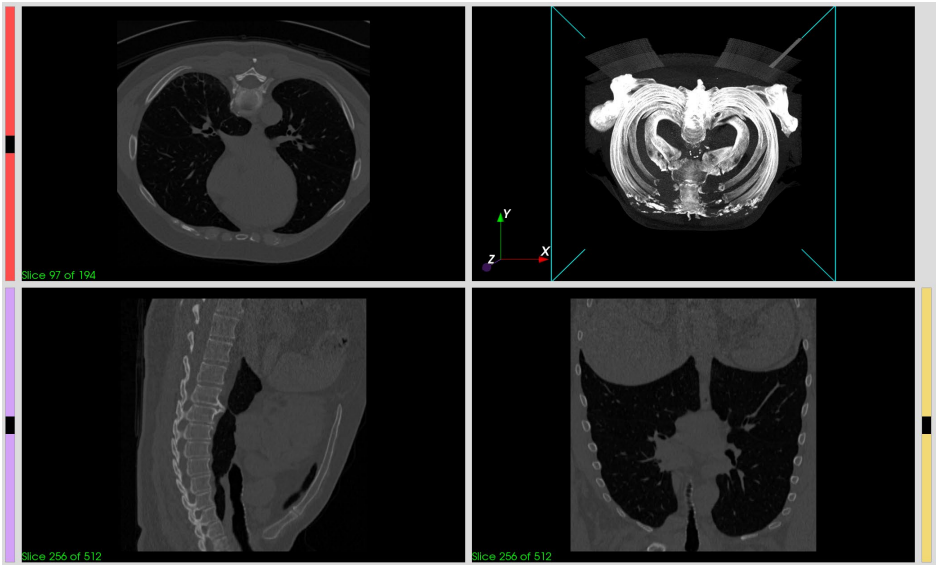


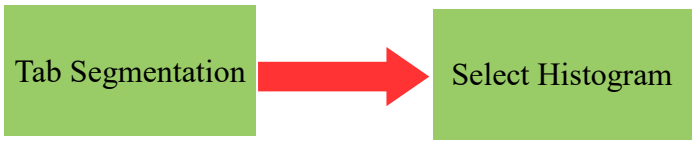
Illustration 6: Original image before Otsu thresholding

Illustration 16: Lung segmented image using Otsu thresholding

Histogram

A histogram is a graphical display of the image intensity distribution using bars of different heights. In a histogram, each bar groups numbers into ranges. Taller bars show that more data falls in that range. A histogram displays the shape and discrete distribution of data intensity.

To see your whole data intensity distribution, select menu from "Segmentation" tab, choose histogram option.



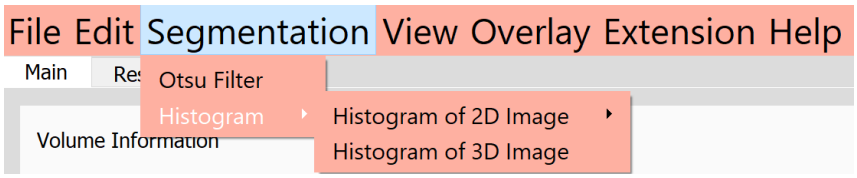


Illustration 17: Displaying histogram with GUI

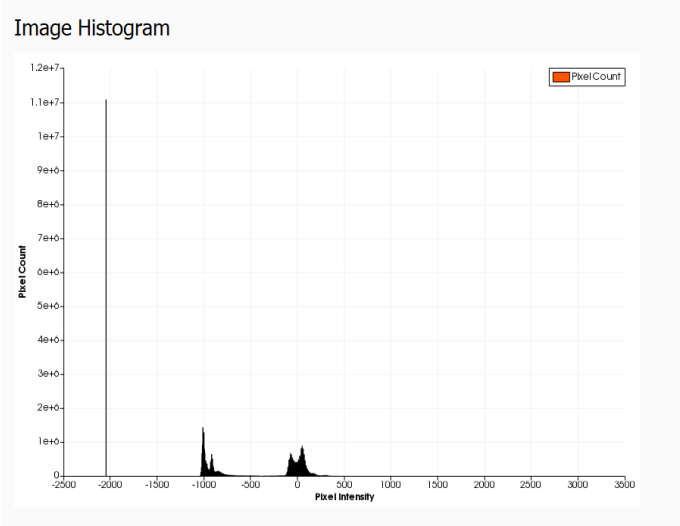


Illustration 18: Histogram of whole data

For viewing a special slice of a plane use bellow command. Input 0 is for Sagittal, 1 is for Coronal, 2 is for Axial plane and 3 is a default value and is for whole data histogram.

```
handler.displayHistogram(PLANE_CODE)
```

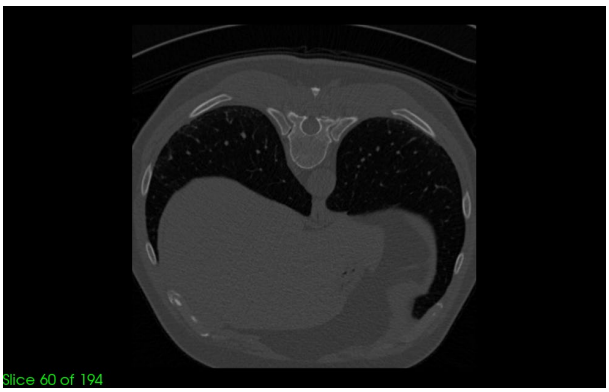


Illustration 19: A sample of axial slice

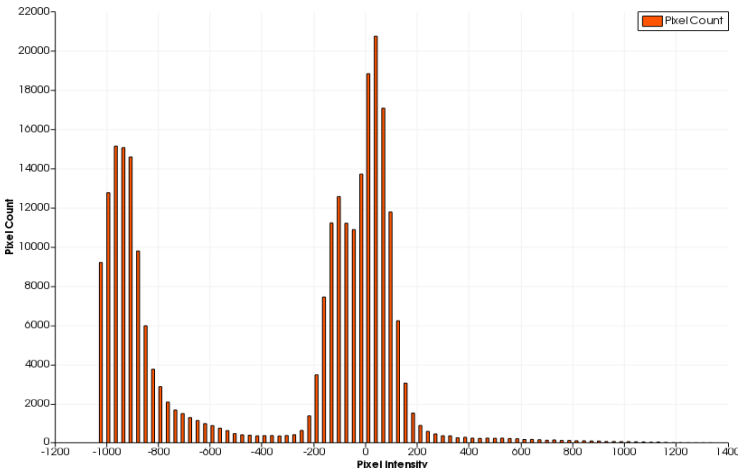


Illustration 20: Histogram of sample axial slice

Graphical Tools

There are some graphical tools which can be accessed via a specific interface placed in the graphical Sometimes you

Measure Tool

Sometimes you need to measure distance between to points of your image data. There is a measurement tool that provides you a ruler to choose two points and find their distance. That's available both in 2D and 3D image.

To use this tool, select measure button in left panel and double click on surface you want to select the start point then click on again to select end point. So now you can see their distance by millimeter(mm) unit. You can also move and expand the ruler. For ending the measurement click on cancel button in the panel.

You can save this measurement information to meta file

by selecting save meta.

Here is an example of measuring distance between two points on axial slice:

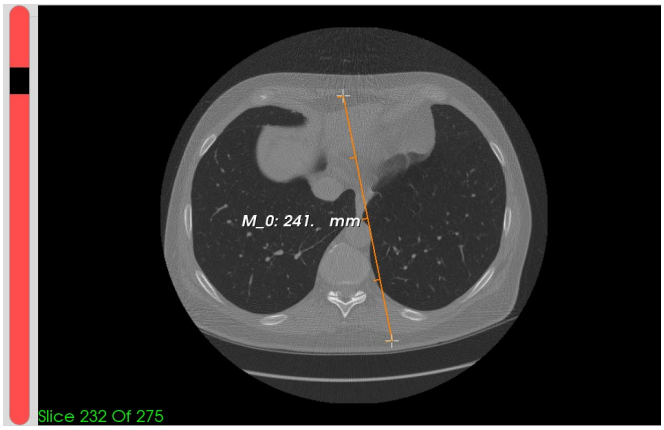


Illustration 21: 2D distance measurement

Here is an example of measuring distance on 3D data too:

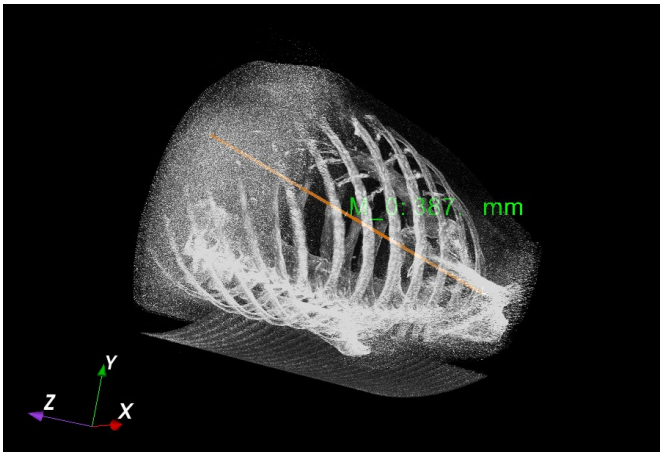


Illustration 22: 3D distance measurement

Annotation Tool

Text Annotation is the practice and the result of adding a note or gloss to an image. By text annotation you can add extra information about a point on image.

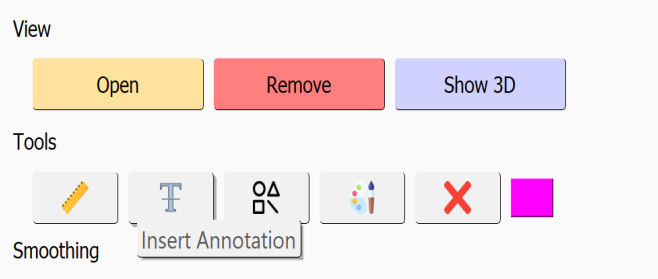


Illustration 23: Annotation tool

To insert text annotation, click on annotation tool in user panel on left side. This enables annotation tool then click on the desired area to insert. Change the annotation value by double clicking on after disabling tools. You can also change its color-by-color changer tool. Pick your desired color then select your meta object.

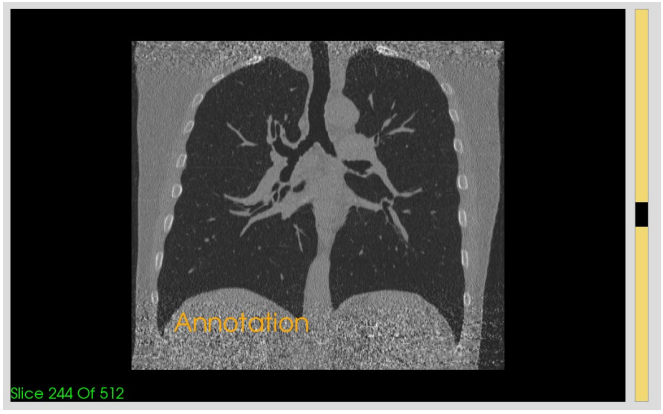


Illustration 24: Sample annotation on image

Draw Box

Sometimes you need to select a region and wants a toll to draw a box for you. You can use draw shapes tool to draw a box on you desired area. You can change its color and drag to move. It is also removable.

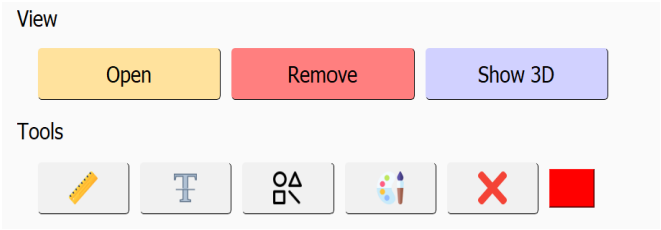


Illustration 25: Drawing tool

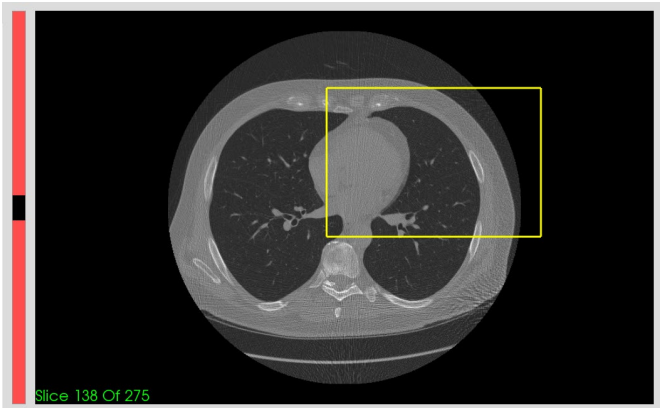


Illustration 26: Drawing box on image

Change Color

Change meta objects color by using this tool. Choose color by picking it from palette, then click on your desired object.

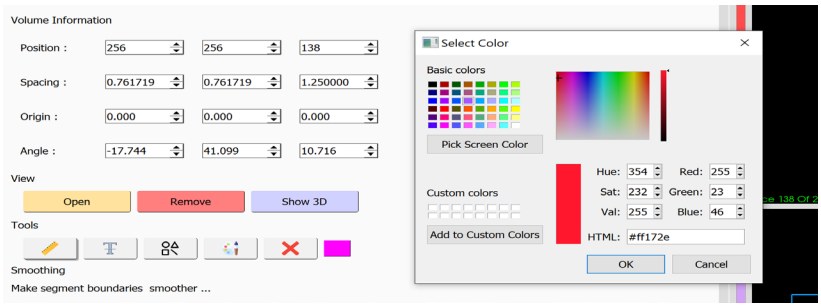


Illustration 27: Color picking

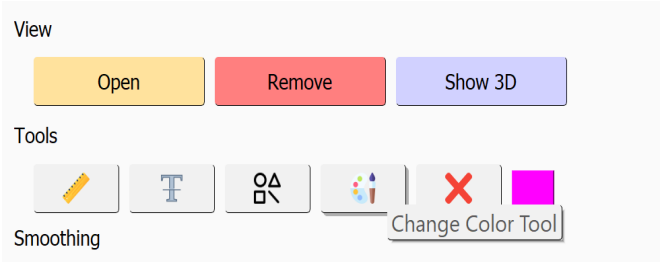


Illustration 28: Color change tool

Remove Tool

Remove tool is for removing added meta objects on image. Enable it by cross button in panel then click on object you wanna delete from scene.

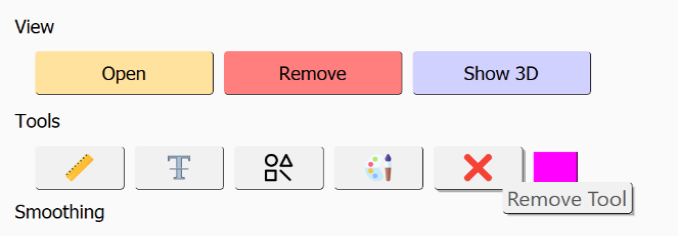


Illustration 29: Removal tool

Neonatal brain image analysis

Creation of subject-specific probability models

To create subject-specific probability models from MRI images, based on the paper "Skull and scalp segmentation in neonatal cerebral MRI using subject-specific probability models" by Hokmabadi et al. (2023), you can follow the steps outlined below:

1) Open the file, as shown in the image below:

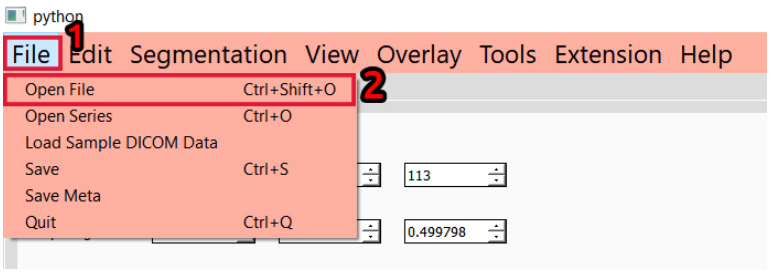


Illustration 30: Open Data with GUI

Then from the open dialog, similar to the following image, select the type of your image (you can use .dcm as default or .nii) :

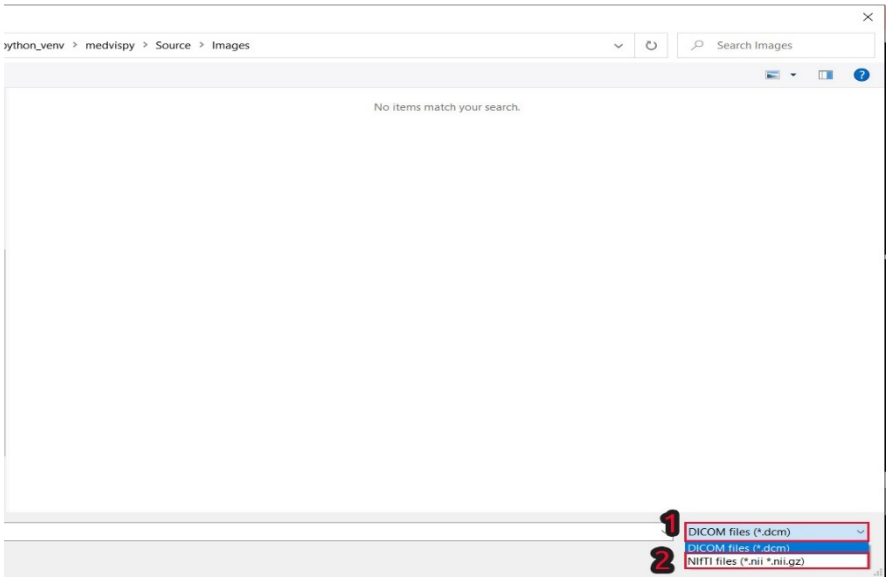


Illustration 31: File opening with GUI

Some sample images are provided in the installation file which can be found in the following path:

SampleData_path > MedVisPySampleData > Brain

Once you have selected your desired image, click on the "OK" button. 2) This will open your image and visualizes it in the graphic windows on the right side of the GUI. To proceed, go to the top menu and click on "Tools", and then click on "MRI Atlas Creation", as shown in the image below.

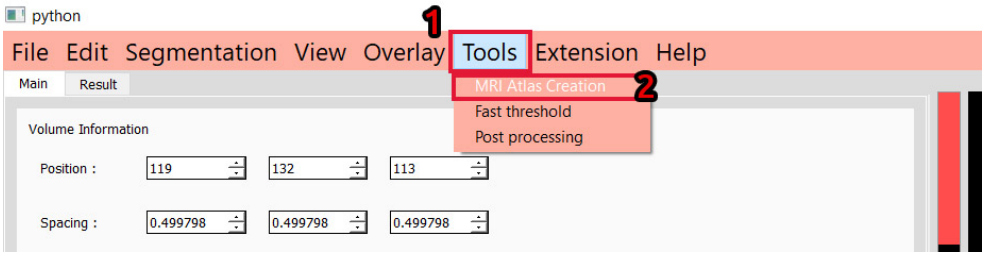


Illustration 32: "MRI Atlas Creation" tab

Once the Dialog box appears, specify the final threshold that will be applied to the skull and scalp probability atlases. The default value is 0.4, but you can use higher thresholds based on your image resolution and experience.

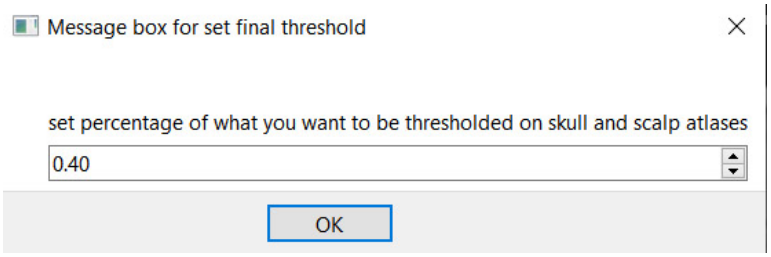


Illustration 33: Specify the final threshold

Depending on your image resolution, a dialog box will appear as shown below. If the resolution of your image is high (intra-slice resolution<0.7×0.7 mm^2), it will ask you whether you want to resample your image (decreasing the size of the image/increasing the within slice pixel spacing). If you choose yes, the program will resample your image and create a template based on the resampled image. Otherwise, it will continue without resampling, but it will take about ten times longer to complete its work. The processing time for segmenting (creating skull, scalp, and CSF tissue probability maps) of an input image with the minimum intra-slice pixel size of 0.5 mm^2 is about 30-40 minutes depending on the system's computational power.

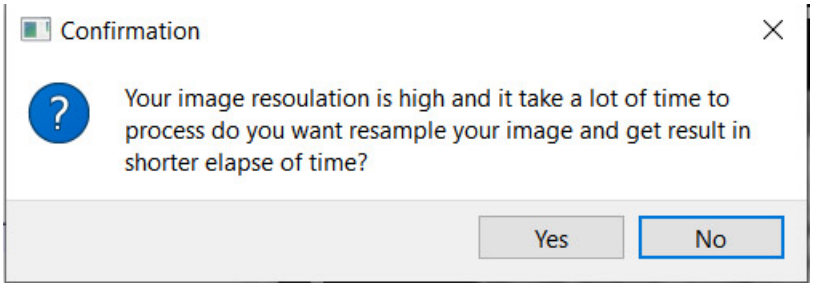


Illustration 34: A dialog box for resampling image

The program will then display an image indicating the ongoing processing.

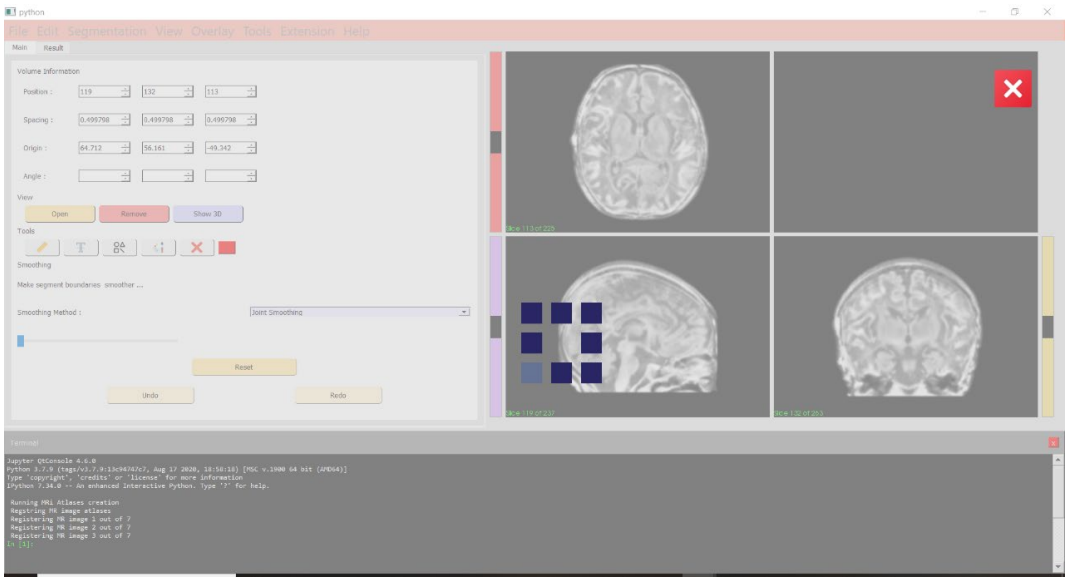


Illustration 35: Ongoing processing window

3) During processing, the program creates two folders "Intermediate_Results" in which the intermediate files are stored and "Final_Atlases" in which the final atlases will be located:

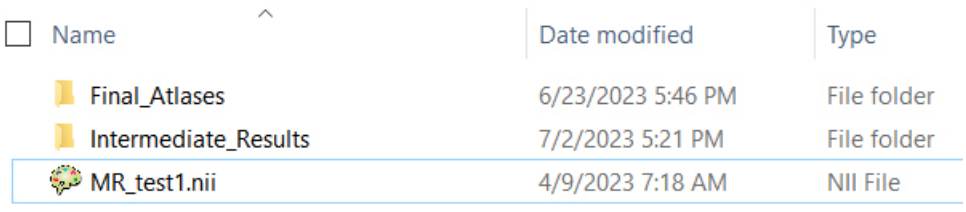


Illustration 36: The path of the intermediate results and final atlases

4) The “Intermediate_Results” folder contains several sub-folders as shown below:

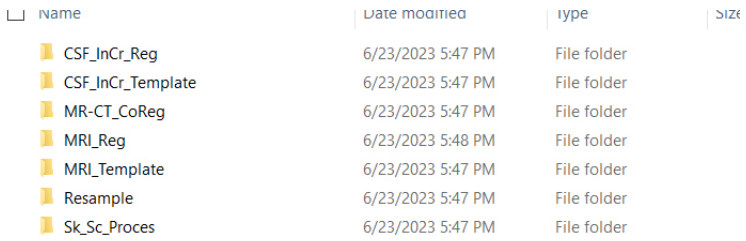


Illustration 37: The “Intermediate_Results” subfolders

These sub-folders are created and filled in the following order:

a) Resample: contains a resampled version of the original image. It enables the user to open the resampled image and use it as the input image for segmentation.
b) MRI_Reg: contains the database MR images that have been registered[2] to the input image by MedVisPy.
c) CSF_InCr_Reg: contains the database CSF and intracranial images corresponding to each database MR image after being aligned to the input image using the same tansformation matrix as used to register the database MR image.
d) MRI_Template: contains the template created based on the registered image in the MRI_Reg folder.
e) CSF_InCr_Template: contains the CSF and intracranial MRI templates. To create them, first all the transformation matrices which were resulted during the process of MR template creation are applied to database CSF and intracranial MR images. Second, the registered CSF and intracranial images are averaged to achieve CSF and intracranial MRI templates.
f) MR-CT_CoReg: contains three subfolders as listed below:

- InCr_CoReg: Contains intracranial of CT images that have been registered to the created intracranial MRI templet, as described in the previous section. Additionally, the resulted warping matrices are available.
- WholeHead_Reg: Contains the CT images that have been undergone the same transformations as in InCr_CoReg.

[2] All deformations for registration and creating templates in the "MRI Atlas Creation tool” are applied using the ANTs software, which was developed within the framework of the ITK open-source tool. This information is available in the following paper: "Avants, B. B., Tustison, N. J., Stauffer, M., Song, G., Wu, B., & Gee,J. C. (2014). The Insight ToolKit image registration framework. Front Neuroinform, 8, 44. https://doi.org/10.3389/fninf.2014.00044".

– Final_Reg: The resulted CT images in WholeHead_Reg folder are registered again to a whole head MRI template. The resulted images and warping matrices are stored in this folder.

g) Sk_Sc_proces: contains three subfolders:

– "CT_Template": contains the CT template which has been constructed using Final_Reg images resulted from the previous section.

– "Temporary_Scalp_files" and "Temporary_Skull_files": All the transformations that have been employed in the "MR-CT_CoReg" folder will be applied to the scalp and skull files of the database CT images and the results will be stored in these folders.

– Sk_Sc_Temp: contains the final probability maps of skull and scalp that were resulted by applying transformation files of CT_template subfolder and taking their average.

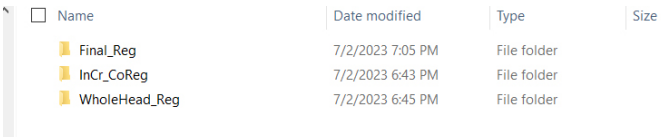


Illustration 38: "MR-CT_CoReg" Folder

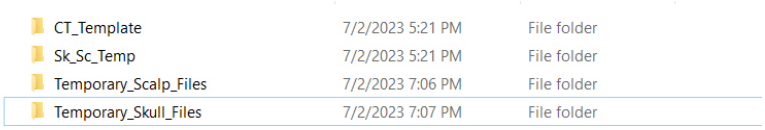


Illustration 39: "Sk_Sc_Temp" Folder

5) During program execution, its status is shown via information messages in command console window.

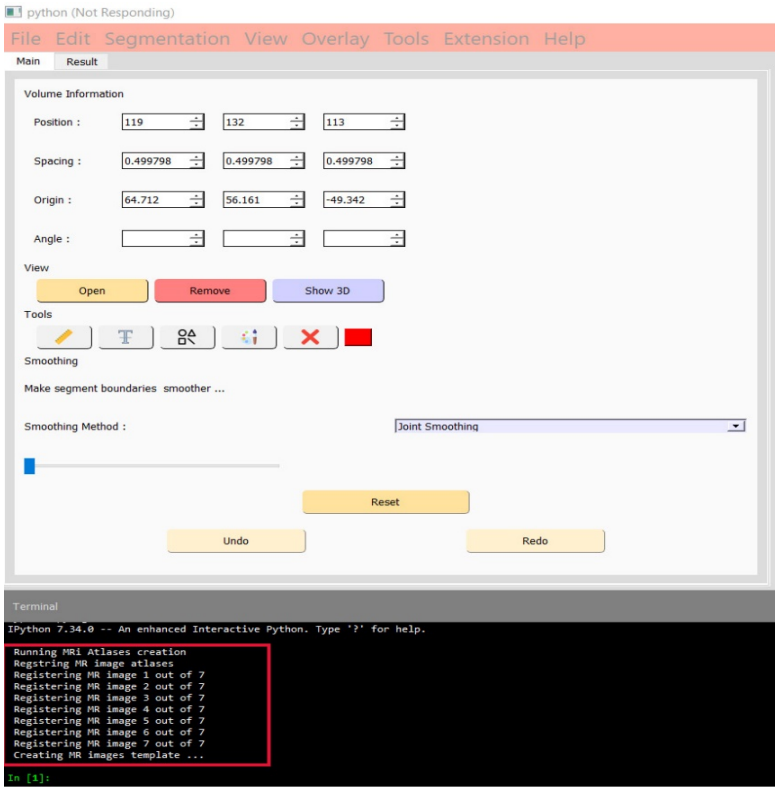


Illustration 40: The information messages in command console window show the status of the program execution

The following figure shows some information messages in the console window

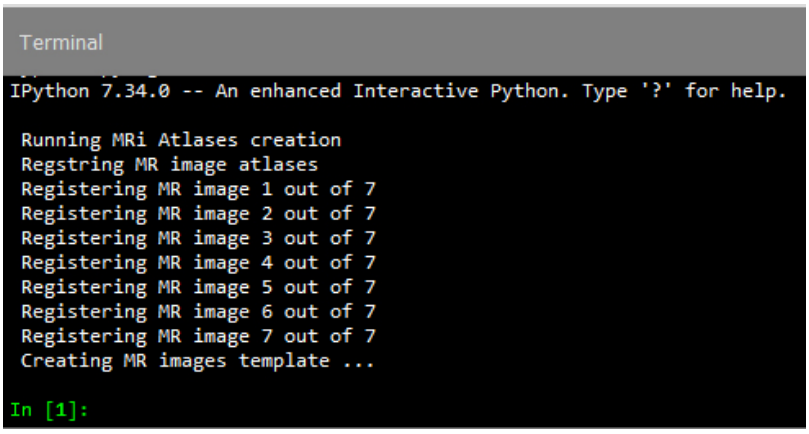


Illustration 41: The stage of processing

6) After completion of the program execution, you will receive the following pop-up message that gives you some information about the total processing time and the path of resulted subject specific skull, scalp and CSF atlases:

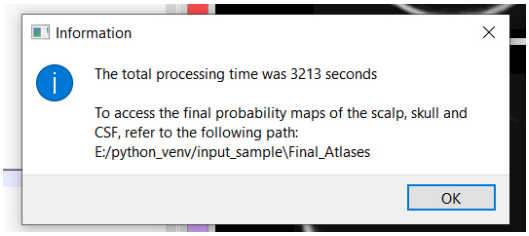


Illustration 42: The program termination pop-up message.

7) You can access (via the delivered path) the final skull, scalp and CSF probability maps that have been already post-processed and are ready to be used in the next step. The post processing includes suppressing the low probability pixels (thresholding) and smoothing.

Illustration 43: The final probability maps

8) After getting the final result (skull, scalp and CSF probability maps) use the below code in FSL:

```
fast -t 1 -n 3 -H 0.1 -I 4 -l 10.0 -a
/usr/local/fsl/etc/flirtsch/ident.mat -A <path_to_csf_template>
<path_to_skull_template> <path_to_scalp _template> -P
<path_to_csf_template> <path_to_skull_template> <path_to_scalp
_template> -o <path_to_mri_file> <path_to_mri_file>
```

and from the resulted images take <MRI_file_name_seg.nii> and open it using MedVisPy. Click on “Fast threshold” from “Tools” menu as shown below:

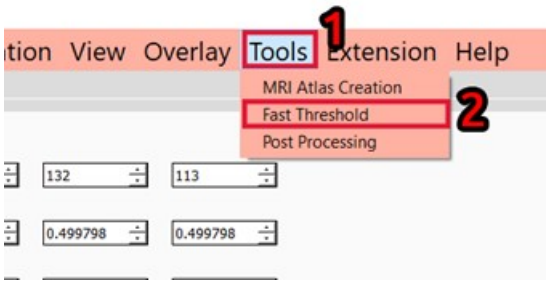


Illustration 44: “Fast threshold” tab

It will create a folder called “FSL_threshold_Result” in the main path as follows:

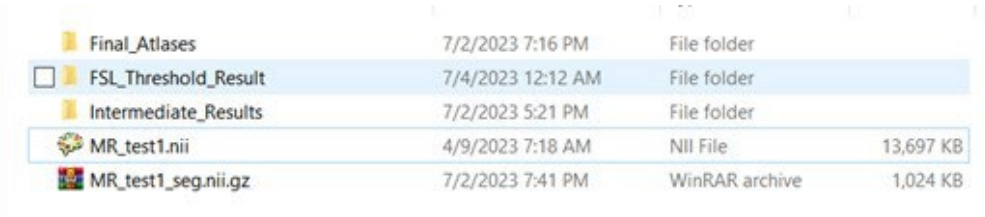


Illustration 45: “FSL_Threshold_Result” Folder

The FSL_threshold_Result folder contains three binary files corresponding to scalp, skull and CSF.

9) For post-processing, open your resulted skull binary image and from “tools” click on “Post processing” as shown below:

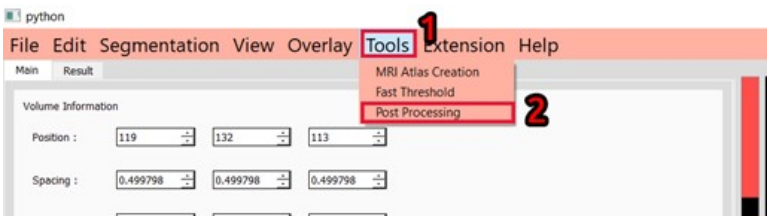


Illustration 46: “Post Processing” tab

It will create a folder called “Post_Processing_Result” in the main path which contains the final segmentation results.

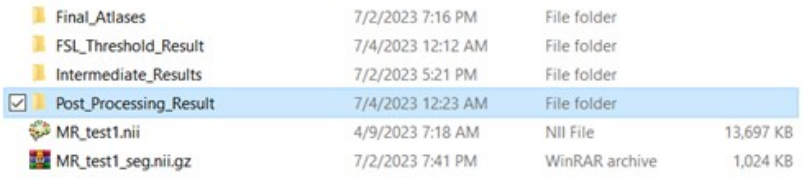


Illustration 47: “Post_Processing_Result” Folder

The following figures show the final probability maps of the skull and scalp that have been resulted through the “MRI Atlas Creation” tab on “Tools” menu.

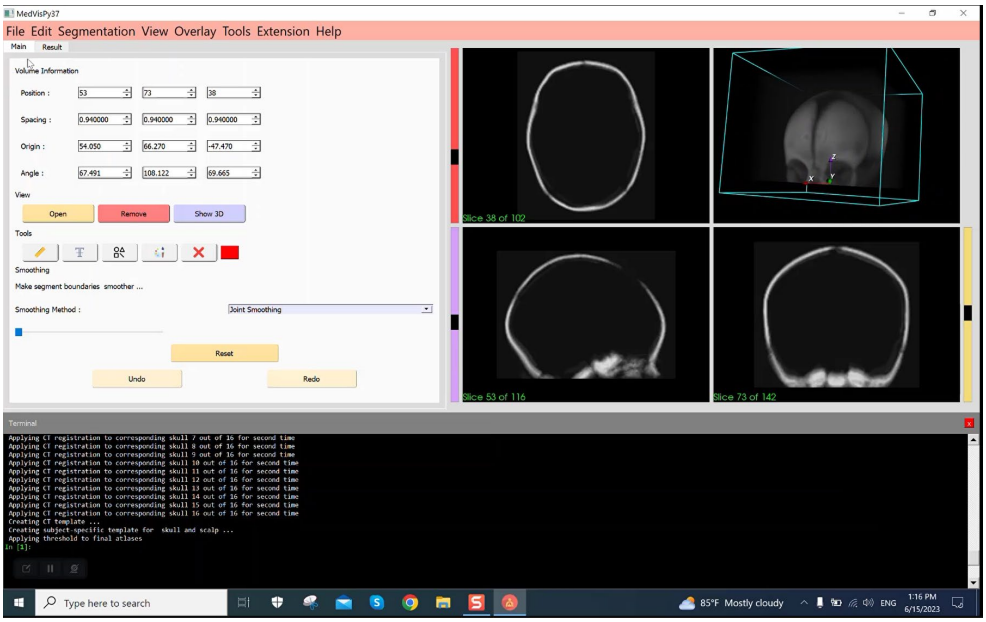


Illustration 48: Probability map of the skull after program execution.

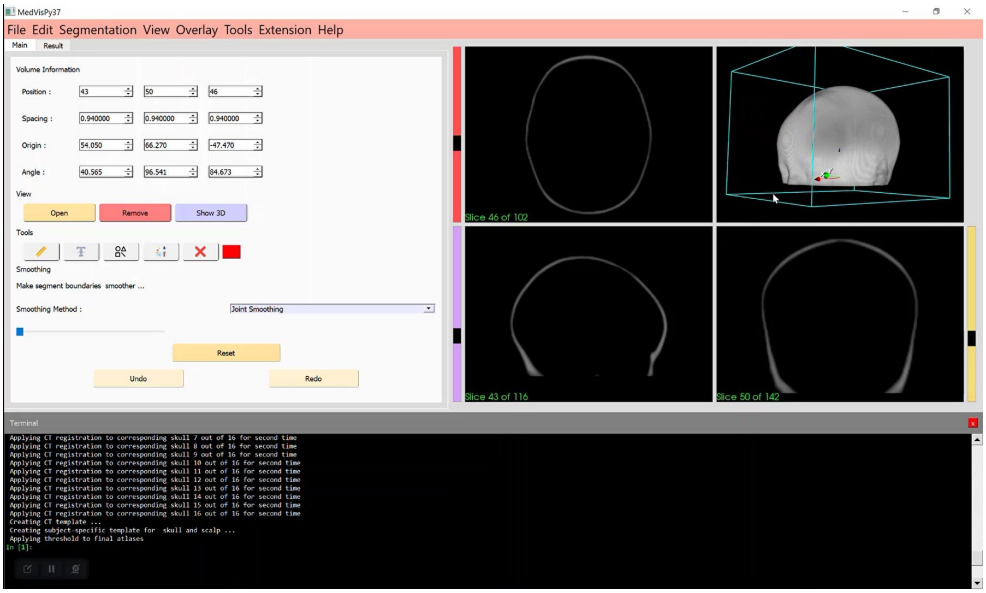


Illustration 49: Probability map of the scalp after finishing the process

Extensions

Plugins

A plugin is a software add-on that is installed on a program, enhancing its capabilities. To add plugin you have to extend it from **Plugin** class and implement **perform_operation** method. Finally put your plugin files in LOCAL APPDATA in Medvispy directory, inside “Plugin_Manger/Plugins” directory.

```
class Plugin(object):
    """Base class that each plugin must inherit from. within this class
    you must define the methods that all of your plugins must implement
    """
    __metaclass__ = abc.ABCMeta

    def __init__(self):
        """
        description: str
            name of the plugin which is appeared in menu bar ,extension section
        type: str
            type of the plugin which is appeared in menu bar ,extension section
        inputArray: data numpy array
            input image numpy array
        outputArray: list
            list of dictionaries
        slices_2D: bool
            image data consists of 2d slices or is a 3d data in general
        """
        self.description = 'UNKNOWN'
        self.type = 'UNKNOWN'
        self.inputArray = []
        self.outputArray = []
        self.slices_2D = False

    @abc.abstractmethod
    def perform_operation(self, argument):
        """The method that we expect all plugins to implement. This is the
        method that our framework will call
        argument: numpy array
            2D or 3D array input
        Return: list
            outputs as a list of dictionaries
        """
```

Illustration 50: Abstract plugin class

Plugin has some fields, description, type, input array, output array and slices_2D. Set plugin’s description to be appeared with that name in app menu bar and set type to be added in right category.

inputArray field is a numpy array of image data that can be 2D or 3D image.

outputArray is a list of dictionaries which contains outputs of each slice. If there is no output, return an empty dictionary.

slice_2D field shows that image consists of some 2d files or is a 3d image in general. By default is False.

Perform_operation method's argument is an input numpy array of image data. This method performs your plugin on inputArray and finally returns data as list of dictionaries.

Plugins must return a list of dictionaries with some key-values. Each list item represents data for each slice. For example if we have 3 separate slices, the list must contain 3 dictionary items. If we have a 3d image that contain 3 slices inside, we have a list that contains only one dictionary item. This is how slices_2D field can help.

Now let's examine keys of dictionaries. If a key contains "data" substring, the value of that key will be saved as a image data with NIfTI format. If the key is "img_data", its value will be shown on software windows.
Other keys are considered as a meta and they will be saved in meta files. If image is 3D it has only one meta otherwise, it has one meta file per 2D slice.
** attention** : values must be a numpy array.

Batch Processing

Most important advantage of plugins is ability to run plugin on a batch of data. This is done by command "runPlugin" and running as batch or not will be declared by batch flag argument. This function takes five arguments as inputs. First argument is plugin name in string, inputPath flag is the path of batch data directory, outputPath is the path of output to store, batch flag means running as a batch or not and excelItem flag represents that store values of excelItem to a excel file.

```
handler.runPlugin(pluginName, inputPath="", batch=False,
excelItem=None, outputPath=None)
```

Each image data must be in a separate directory. Put your single NIfTI file or multiple DICOM files in a single subdirectory. inputPath is the outer directory that contains all these subdirectories.

For example:

Data

F1

IM00001.dcm

IM00002.dcm

IM00003.dcm

IM00004.dcm

F2

img_data.nii

F3

img_data.nii

inputpath is Data path that has 3 data. Each data is located in a separate subdirectory.

In batch processing the perform_operation output data are saved in a different binary meta file with its key values fields with the name of the single image file that plugin was run on. These meta files can be easily exported by MedVisPy software.

If your key includes "data" substring, this data will be stored as image data in NIfTI format not in meta file. So if your plugin returns image data you can set its key name with substring of "data".

ExcelItem: you can easily save your desired returned label of your plugin in an excel file. In excel file, label_name filed is considered as columns and label values are added to its related column. Each item in label array will be put in different rows.
For example if we have 3 items in label, we have 3 rows which means we have 3 slices.
In multi image files like DICOMs, we have multiple labels not multiple items in label array. So it is also stored in multiple rows as multiple slices.
For each row we have name of that slice too. If we have multiple files, we have the name of the files, if we have a single file, we add slice number to the name of that file.

Example 1

Sample data sets are available in "Source/SampleData" directory. Let's run LungSegmentation plugin on data1 and data2 as a batch.

Consider command bellow:

```
handler.runPlugin("LungSegmentation", "Source\\SampleData",
                  outputPath="G:\\AppOutput",batch=True)
```

Set pluginName to LungSegmentation, pluginDir flag to directory that holds all data like SampleData, batch flag to True.

If we don't set the output path argument by default it would create a folder in base input directory and store the output. But here we've set "G:\\AppOutput".

The plugin outputs are stored in "G:\\AppOutput" directory as .nii files. To see the outputs open the .nii files. Here we've opened image data file to see the segmented image.

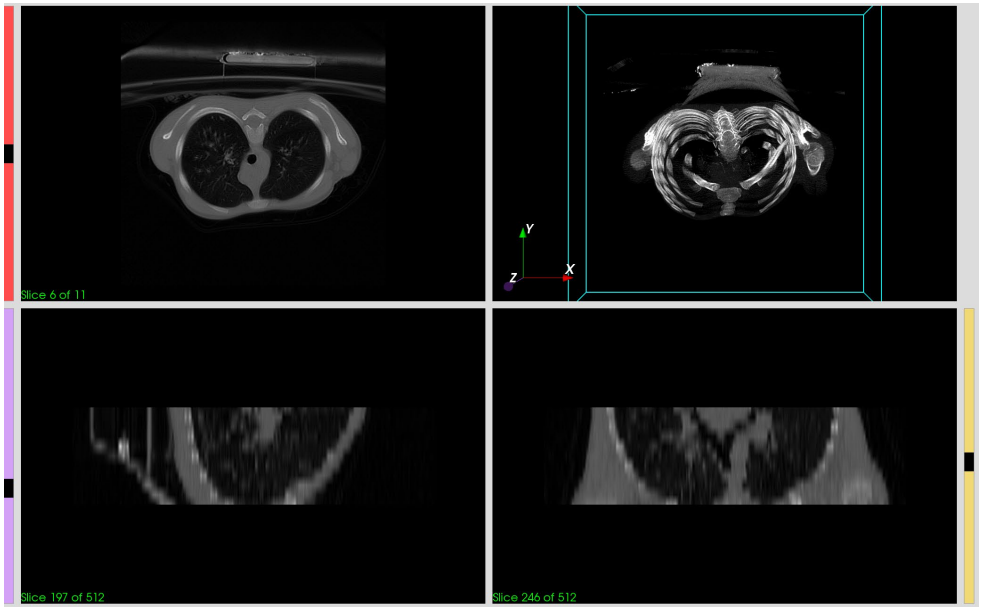


Illustration 5110: Original image data1

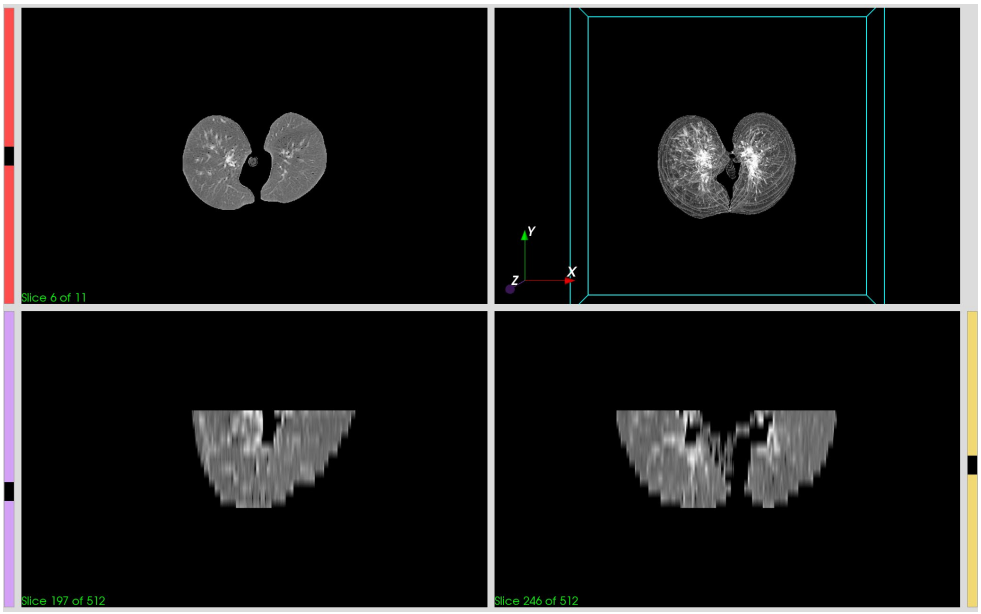


Illustration 52: Segmented image data1

Example 2

Let's run LungSegmentation plugin on data in F1 and F2 direcories as a batch. Both F1 and F2 directories are located in MyData directory. F1 and F2 also contains DICOM images.

Consider command bellow:

```
handler.runPlugin("LungSegmentation", "D:\\MyData",
batch=True)
```

Here is content of MyData directory. It has two subdirectories:

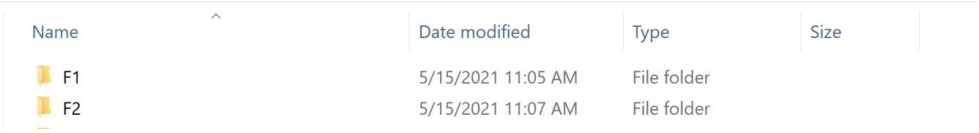


Illustration 53: MyData directory

Here is content of F1 subdirectory:

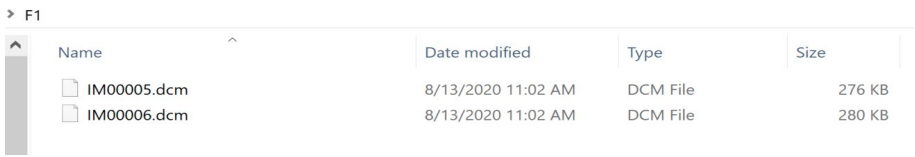


Illustration 54: F1 directory

The result of running LungSegmentation on MyData directory is shown below. It stores each data in its directory.

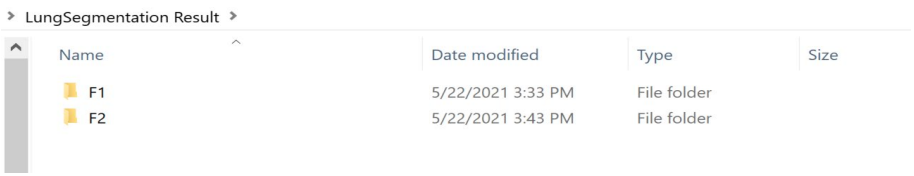


Illustration 55: Output directory

The plugin returns both mask and segmented images. They stored in NIfTI format.

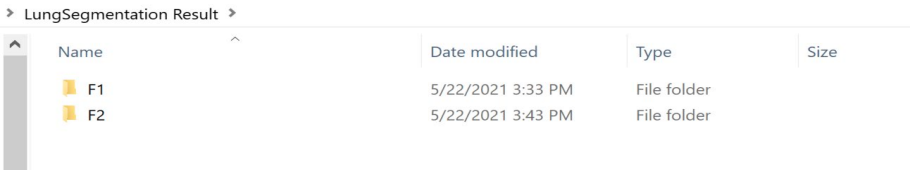


Illustration 56: Output directory

Here is the output of running plugin on F1 data.

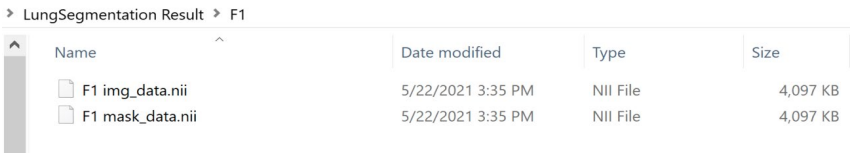


Illustration 57: F1 output directory

Example 3

Let's run LungPatternClassification plugin on NIfTI images data (F1.nii , F2.nii) in MyData as a batch. They are a 3D image data with 2 slices. So the output must be two meta files in each directory with the name of the input file. Each slice has an image overlay that is stored in one separate file. You can open these meta files to add image overlay to your image.

Consider command bellow:

```
handler.runPlugin("LungPatternClassification", "D:\\MyData", batch=True)
```

It creates a meta file in input directory.

Later you can open this meta file with "Load image overlay" option from menu.

Illustration 58: Load image overlay menu

Select meta file of slice 1:

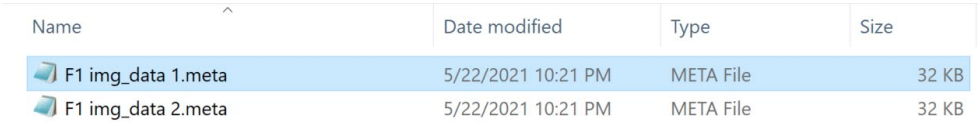


Illustration 59: Meta file selection

The classified image is shown below with blue, red and green labels.

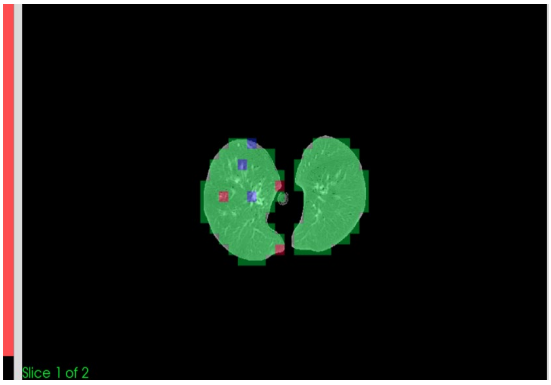


Illustration 60: Classified image overlay

From the overlay image it can be realized that the green areas are the largest areas of the image, means that most parts of the image is normal.

In the next page, also the pie chart of classified image is shown to illustrate statistical proportion of the labels.

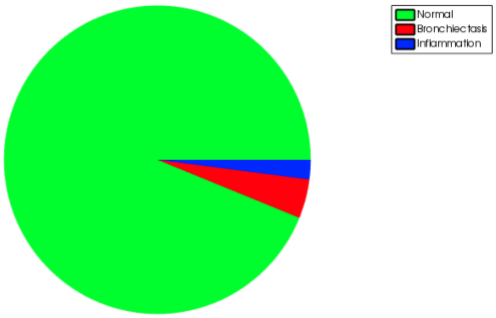


Illustration 61: Show proportion of output labels in pie chart

Example 4

Here we have a single file “B1 img_data.nii” that contains 2 slices. Let’s run LungPatternClassification plugin as a batch on “G1” directory that contains a single file “B1 img_data.nii”.

Consider command bellow:

```
handler.runPlugin(“LungPatternClassification”,
“Source\\SampleData\\ G1”, batch=True, excelItem=“labels”)
```

If we execute command above, it creates a directory called “LungPatternClassification Result” that inside of it, we have a single meta and an excel file.

Name	Date modified	Type	Size
B1 img_data.meta	6/1/2021 4:28 PM	META File	10 KB
B1 img_data.xlsx	6/1/2021 4:28 PM	Microsoft Excel W...	5 KB

Illustration 62: LungPatternClassification result on B1 img_data

Now let’s analyze the content of the meta file:

```
# vtk DataFile Version 4.2
vtk output
ASCII
DATASET STRUCTURED_POINTS
FIELD FieldData 8
MedVisPy_Version 1 1 string
2.0.4

labels 1024 2 vtktypeint64
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

Illustration 63: Content of B1 img_data meta file

Here we have 8 field data with different keys and values. First field specifies the version of the meta file. Second field is labels. It has two labels items. This represents that we have two slices that each slice has 1024 label values.

In runPlugin command we've set excelItem argument to "labels", means that it should store the frequency of the label field values of each returned item to the excel file. This excel file is used for statistical analysis.

As shown bellow, excel file has 2 rows with the names of "B1 img_data_s0" as first slice of B1 image_data and "B1 img_data_s1" as second slice of B1 image_data.

In each row we have three columns that specifies the frequency of each classification label.

In Lung Pattern Classification plugin, the lung image will be classified through three labels that are represented by Green, Red and Blue colors.

	A	B	C	D	E
1	**Slice Name**	**Green**	**Red**	**Blue**	
2	B1 img_data_S0	1018	3	3	
3	B1 img_data_S1	1018	6	0	
4					

Illustration 64: Content of B1 img_data excel file

Console

IPython Console Description

The **IPython Console** allows you to execute commands and enter, interact with and visualize data inside any number of fully featured IPython interpreters. With this Console besides of python commands you can execute each software function available with user interface.

Example:

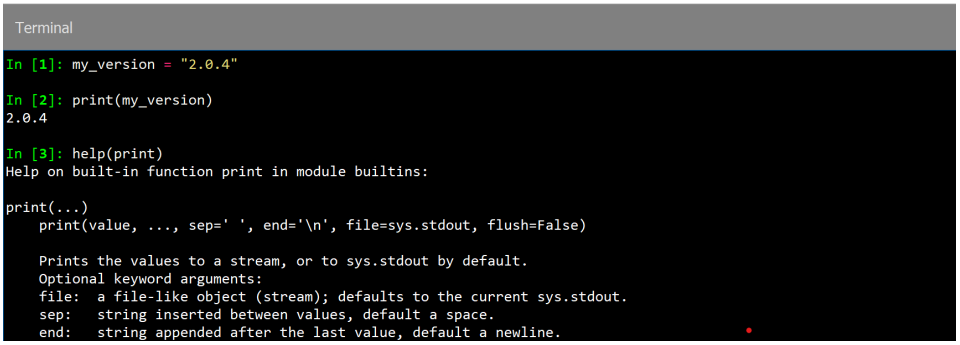


Illustration 65: Schema of IPython console

You can hide console by cross button in the right top of the console and then reverse it by using:

menubar > view > show console

Commands

All functions you can do graphically can be done by executing its related command.

Here's list of all commands you can use :

- openFile("FILE PATH")
  // Open and Display Image File
- openData("DIR PATH")
  // Open and Display Image Series
- show3D()
  // Display 3D of Opened Image Data
- displayHistogram(PLANE_CODE)
  // Display Histogram of Desired Plane
- saveData("PATH")

// Save Current Volume

- runPlugin(`pluginName, inputPath="", batch=False, excelItem=None, outputPath=None`)

// Run plugin on a single data or a batch of data

- saveMeta("FILE NAME")

// Save Meta Information

- cropData(START_POINT, SIZE)

// Select Region of Interest

- otsuThreshold()
  // Find Threshold using Otsu and Apply to Image
- help()

// Display How to Use a Command

- helpHelp()

// Display How to Get Help of Method

*** You have to run above commands by instance of Handler object like this:**

handler.openData()

(handler object is created before. No need to recreate.)

Some Python Commands

- help(function name)

// Display docstring of Method

- pwd()

// Display Working Directory

- dir()

// Display A List of the Files and
// SubFolders Contained in Current Directory

Appearance

Theme

You can change software theme using view item from menu bar. You can switch between light theme and dark theme.

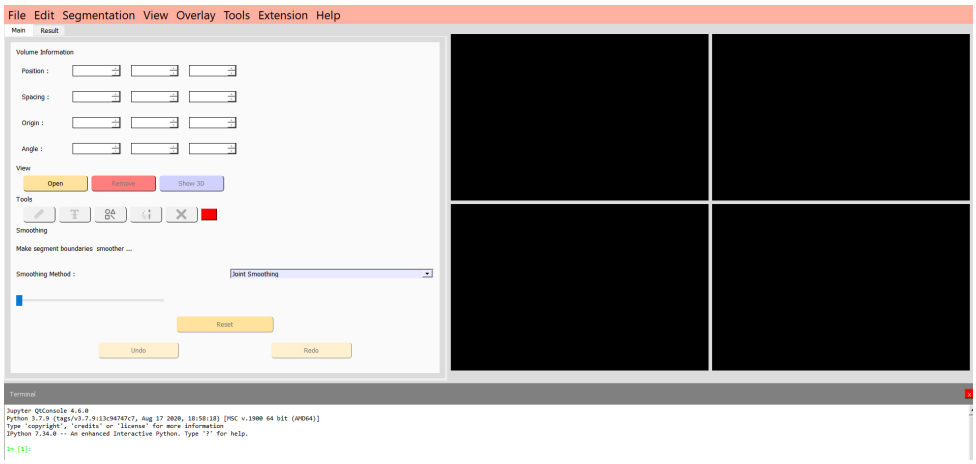


Illustration 66: Software light theme

Overview

MedVisPy Schema

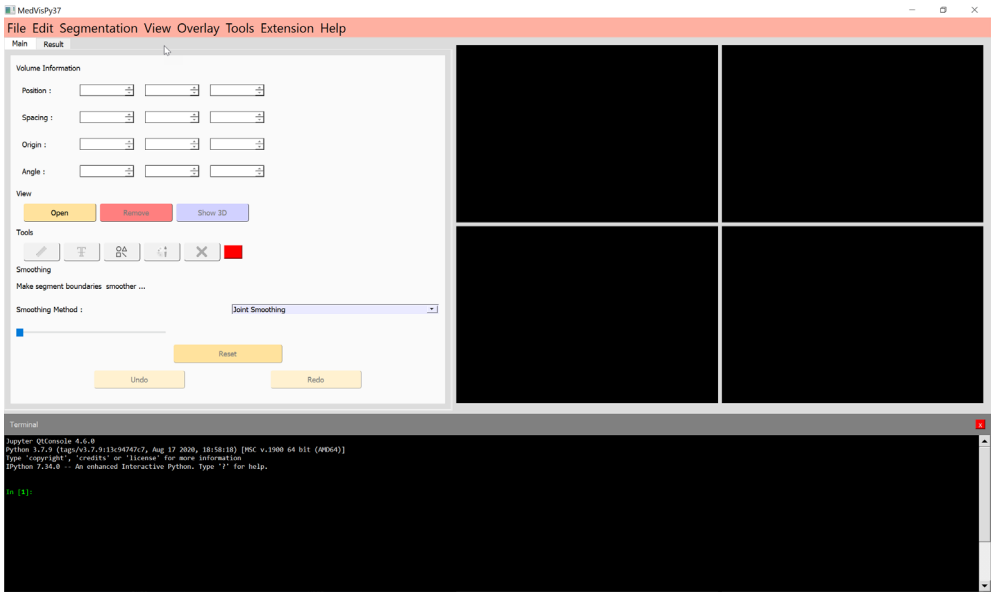


Illustration 67: Schema before opening image data

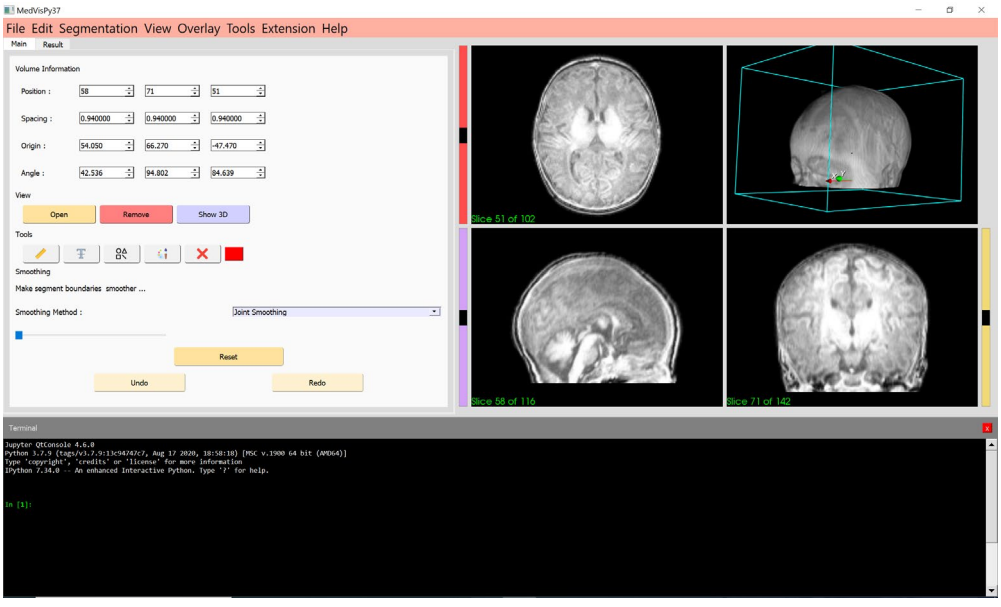


Illustration 68: Schema after opening image data

MedVisPy is able to

Display medical images in 2D Slices and a 3D volume. It provides some tools for preprocessing and applying some filters on image. At the end, you can save and reopen the volume.
The Software supports also for CT images such as NifTI and DICOM formats.
The other important feature is that MedVisPy helps users who are more comfortable to work with commands, do all they can do graphically by providing a console.
MedVisPy enables users to implement their custom plugins and easily add them to the software and work with them.
It can also run plugin on a batch of data. It saves a lot of works and time. You can either work with software at the same time. Batch processing is running parallel so there is no conflict and you do not need to wait for process to be finished.