DICOMRTTool_manuscript

January 11, 2021

1 DICOM RT Tool Tutorial with Open-Access Data

This notebook demonstrates the various functions and utilities available in the Dicom RT tool Python package (https://github.com/brianmanderson/Dicom_RT_and_Images_to_Mask) by Anderson et. al. It serves as supplementary information for the Technical Paper titled: "Simple Python Module for Conversions between DICOM Images and Radiation Therapy Structures, Masks, and Prediction Arrays" . This notebook works through an example of publicly available brain tumor data of T1-w/FLAIR MRI sequences and corresponding RT structure files with multiple segmented regions of interest. Full information of the publicly available brain tumor data used in this notebook can be found at: https://figshare.com/articles/dataset/Data_from_An_Investigation_of_Machine_Learning_Methods_in_Delta radiomics_Feature_Analysis/9943334. This notebook was written for easy accessibility for beginners to Python programming, medical imaging, and computational analysis. It should take no more than 10-15 minutes to run in it's entirety from scratch. The notebook generates about 10 GB worth of files, so ensure you have adequate space to run it.

The notebook covers the following topics (click to go to section): 1. Getting the data 2. Reading in DICOM and RT struct files and converting to numpy array format 3. Saving arrays to nifti format and reloading them 4. Saving and loading numpy array files 5. Calculating radiomic features 6. Predictions To RT-Structure Example

The notebook assumes you have the following nested directory structure after running cells that download necessary data:

[ ]:
```
"""
Top-level directory/
   DICOMRTTool_manuscript.ipynb
   Example_Data/ <- Generated when you run the cells below
|      Image_Data/
|          Structure/ <- These correspond to the Pre-RT scans
               T1/
|                   Patient number/
|                       RT Struc file (.dcm)
               T2FLAIR/
|                   Patient number/
|                       RT Struc file (.dcm)
|          T1/
|              Post1/
|                   Patient number/
```

```
|                       DICOM image files (.dcm)
|              Post2/
|                  Patient number/
|                       DICOM image files (.dcm)
|              Pre/
|                  Patient number/
|                       DICOM image files (.dcm) <- The images we care about
|          T2FLAIR/
|              Post1/
|                  Patient number/
|                       DICOM image files (.dcm)
|              Post2/
|                  Patient number/
|                       DICOM image files (.dcm)
|              Pre/
|                  Patient number/
|                       DICOM image files (.dcm) <- The images we care about
   Data.zip <- Generated when you run the cells below, downloaded Figshare file
   Nifti_Data/ <- Generated when you run the cells below
|     Image.nii
|     Mask.nii
|     MRN_Path_To_Iteration.xlsx
|     Overall_Data_Examples_(iteration).0.nii.gz
|     Overall_mask_Examples_y(iteration).0.nii.gz
   Numpy_Data/ <- Generated when you run the cells below
|     image.npy
|     mask.npy
"""
```

[1]:
```
%%capture
# Load or install the program, %%capture supresses print statements
!pip install DicomRTTool --upgrade
from DicomRTTool.ReaderWriter import DicomReaderWriter
```

[2]:
```
# importing neccessary libraries

# file mangagment
import os
import zipfile
from six.moves import urllib

# array manipulation and plotting
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# medical image manipulation
```

```
import SimpleITK as sitk
```

[3]:
```
# os.chdir("..") # move up one folder to access src folder, temporary solution
 ↪until PyPI is updated
# cwd = os.getcwd()
# from src.DicomRTTool.ReaderWriter import DicomReaderWriter # temporary while
 ↪working on latest version from github pulls
# os.chdir("Examples") # change working directory back to what it was
# cwd = os.getcwd()
```

1.1 Part 1: Getting the data.

The RT struc files and their corresponding DICOM images can be in the same directory or different directories. Here we show a case where structure files and images are located in different directories. This is a good dataset to work with since its somewhat messy but coherent enough to show power of DICOMRTTool. Many files (pre-RT, post-RT at 2 timepoints) but only pre-RT T1 and FLAIR images have associated RT structure files. Downloading and unzipping the necessary files will take about 10 minutes on most CPUs and takes up about 8 GB of storage. One may visualize these DICOM images using a free commercially available DICOM viewer, such as Radiant (https://www.radiantviewer.com/).

[4]:
```
%%time
data_path = os.path.join('.', 'Example_Data')
if not os.path.isdir(data_path): # create Example_data directory if it doesn't
 ↪exist
    os.mkdir(data_path)

url_img = "https://ndownloader.figshare.com/files/20140100" # brain scans
filename_img = os.path.join(data_path, 'Data.zip')
if not os.path.exists(filename_img): # if zip file doesnt exist download
    print ("Retrieving zipped images...")
    print('Estimated download time is 5 minutes...')
    urllib.request.urlretrieve(url_img, filename_img)
    print('Finished downloading!')
else:
    print ("Zipped images already downloaded.")

if os.path.exists(filename_img):  # If we downloaded the data
    if not os.path.exists(os.path.join(data_path, 'Image_Data')): # and it
 ↪hasn't been unzipped
        print ("Unzipping images...")
        print('Estimated unzip time is 2 minutes')
        z = zipfile.ZipFile(filename_img)
        z.extractall(data_path)
        print ("Done unzipping images.")

print("All required files downloaded and unzipped!") # print when done
```

```
Zipped images already downloaded.
All required files downloaded and unzipped!
Wall time: 4.4 ms
```

[5]:
```python
def display_slices(image, mask, skip=1):
    """
    Displays a series of slices in z-direction that contains the segmented␣
↪regions of interest.
    Ensures all contours are displayed in consistent and different colors.
        Parameters:
            image (array-like): Numpy array of image.
            mask (array-like): Numpy array of mask.
            skip (int): Only print every nth slice, i.e. if 3 only print every␣
↪3rd slice, default 1.
        Returns:
            None (series of in-line plots).
    """

    slice_locations = np.unique(np.where(mask != 0)[0]) # get indexes for where␣
↪there is a contour present
    slice_start = slice_locations[0] # first slice of contour
    slice_end = slice_locations[len(slice_locations)-1] # last slice of contour

    counter = 1

    for img_arr, contour_arr in zip(image[slice_start:slice_end+1],␣
↪mask[slice_start:slice_end+1]): # plot the slices with contours overlayed␣
↪ontop
        if counter % skip == 0: # if current slice is divisible by desired skip␣
↪amount
            masked_contour_arr = np.ma.masked_where(contour_arr == 0,␣
↪contour_arr)
            plt.imshow(img_arr, cmap='gray', interpolation='none')
            plt.imshow(masked_contour_arr, cmap='cool', interpolation='none',␣
↪alpha=0.5, vmin = 1, vmax = np.amax(mask)) # vmax is set as total number of␣
↪contours so same colors can be displayed for each slice
            plt.show()
        counter += 1
```

1.2 Part 2: Reading in DICOM and RT struct files and converting to numpy array format.

The principal on which this set of tools operates on is based on the DicomReaderWriter object. It is instantiated with the contours of interest (and associations) and can then be used to create numpy arrays of images and masks of the format [slices, width, height].

The following code logic is used to demonstrate searching a path and returning indices for matched structures and images (by UID) for arbitrary directory structures (DICOM image files and RT

Struct files not in the same folder). If all necessary structure files are in the same folder as the corresponding images (by UID), one can alternatively use an os.walk through directories of interest and call DicomReaderWriter each time a folder is discovered. For example, I normally use a folder structure MRN -> date of image (pre,mid,post-RT) -> type of scan (MRI, CT, etc.) -> files (DICOM images + RT Struct). However, this approach calls the DicomReaderWriter iteratively, which can be computationally taxing.

[6]:
```
DICOM_path = os.path.join('.', 'Example_Data', 'Image_Data') # folder where␣
 ↪downloaded data was stored
print(DICOM_path)
```

```
.\Example_Data\Image_Data
```

This will walk through all of the folders, and using SimpleITK, will separate them based on SeriesInstanceUIDs.

[7]:
```
%%time
Dicom_reader = DicomReaderWriter(description='Examples', arg_max=True)
print('Estimated 30 seconds, depending on number of cores present in your␣
 ↪computer')
Dicom_reader.walk_through_folders(DICOM_path) # need to define in order to use␣
 ↪all_roi method
```

```
Estimated 30 seconds, depending on number of cores present in your computer
Loading from .\Example_Data\Image_Data\Structure\T1\005Loading from
.\Example_Data\Image_Data\Structure\T1\001
Loading from .\Example_Data\Image_Data\Structure\T1\003

Loading from .\Example_Data\Image_Data\Structure\T1\006
Loading from .\Example_Data\Image_Data\Structure\T1\004
Loading from .\Example_Data\Image_Data\Structure\T1\002
Loading from .\Example_Data\Image_Data\Structure\T1\007
Loading from .\Example_Data\Image_Data\Structure\T1\009
Loading from .\Example_Data\Image_Data\Structure\T1\010Loading from
.\Example_Data\Image_Data\Structure\T1\011

Loading from .\Example_Data\Image_Data\Structure\T1\008Loading from
.\Example_Data\Image_Data\Structure\T1\012

Loading from .\Example_Data\Image_Data\Structure\T2Flair\001
Loading from .\Example_Data\Image_Data\Structure\T2Flair\002
Loading from .\Example_Data\Image_Data\Structure\T2Flair\003
Loading from .\Example_Data\Image_Data\Structure\T2Flair\004
Loading from .\Example_Data\Image_Data\Structure\T2Flair\005Loading from
.\Example_Data\Image_Data\Structure\T2Flair\006

Loading from .\Example_Data\Image_Data\Structure\T2Flair\007
Loading from .\Example_Data\Image_Data\Structure\T2Flair\008
Loading from .\Example_Data\Image_Data\Structure\T2Flair\009Loading from
```

```
.\Example_Data\Image_Data\Structure\T2Flair\010

Loading from .\Example_Data\Image_Data\Structure\T2Flair\011
Loading from .\Example_Data\Image_Data\Structure\T2Flair\012
Loading from .\Example_Data\Image_Data\T1\001
Loading from .\Example_Data\Image_Data\T1\002
Loading from .\Example_Data\Image_Data\T1\003
Loading from .\Example_Data\Image_Data\T1\004
Loading from .\Example_Data\Image_Data\T1\005
Loading from .\Example_Data\Image_Data\T1\006
Loading from .\Example_Data\Image_Data\T1\007
Loading from .\Example_Data\Image_Data\T1\009
Loading from .\Example_Data\Image_Data\T1\010Loading from
.\Example_Data\Image_Data\T1\008

Loading from .\Example_Data\Image_Data\T1\Post1\001
Loading from .\Example_Data\Image_Data\T1\Post1\002Loading from
.\Example_Data\Image_Data\T1\Post1\003
Loading from .\Example_Data\Image_Data\T1\Post1\004

Loading from .\Example_Data\Image_Data\T1\Post1\005Loading from
.\Example_Data\Image_Data\T1\Post1\006

Loading from .\Example_Data\Image_Data\T1\Post1\007
Loading from .\Example_Data\Image_Data\T1\Post1\009
Loading from .\Example_Data\Image_Data\T1\Post1\008
Loading from .\Example_Data\Image_Data\T1\Post1\010
Loading from .\Example_Data\Image_Data\T1\Post1\011
Loading from .\Example_Data\Image_Data\T1\Post1\012
Loading from .\Example_Data\Image_Data\T1\Post2\001
Loading from .\Example_Data\Image_Data\T1\Post2\002
Loading from .\Example_Data\Image_Data\T1\Post2\003
Loading from .\Example_Data\Image_Data\T1\Post2\004
Loading from .\Example_Data\Image_Data\T1\Post2\005
Loading from .\Example_Data\Image_Data\T1\Post2\006
Loading from .\Example_Data\Image_Data\T1\Post2\007
Loading from .\Example_Data\Image_Data\T1\Post2\008
Loading from .\Example_Data\Image_Data\T1\Post2\009
Loading from .\Example_Data\Image_Data\T1\Post2\010
Loading from .\Example_Data\Image_Data\T1\Post2\011
Loading from .\Example_Data\Image_Data\T1\Post2\012
Loading from .\Example_Data\Image_Data\T1\Pre\001
Loading from .\Example_Data\Image_Data\T1\Pre\002
Loading from .\Example_Data\Image_Data\T1\Pre\003
Loading from .\Example_Data\Image_Data\T1\Pre\004
Loading from .\Example_Data\Image_Data\T1\Pre\005
Loading from .\Example_Data\Image_Data\T1\Pre\006
Loading from .\Example_Data\Image_Data\T1\Pre\007
```

```
Loading from .\Example_Data\Image_Data\T1\Pre\008
Loading from .\Example_Data\Image_Data\T1\Pre\009
Loading from .\Example_Data\Image_Data\T1\Pre\010
Loading from .\Example_Data\Image_Data\T1\Pre\011
Loading from .\Example_Data\Image_Data\T1\Pre\012
Loading from .\Example_Data\Image_Data\T2Flair\Post1\001
Loading from .\Example_Data\Image_Data\T2Flair\Post1\002
Loading from .\Example_Data\Image_Data\T2Flair\Post1\003
Loading from .\Example_Data\Image_Data\T2Flair\Post1\004
Loading from .\Example_Data\Image_Data\T2Flair\Post1\005
Loading from .\Example_Data\Image_Data\T2Flair\Post1\006
Loading from .\Example_Data\Image_Data\T2Flair\Post1\007
Loading from .\Example_Data\Image_Data\T2Flair\Post1\008
Loading from .\Example_Data\Image_Data\T2Flair\Post1\009
Loading from .\Example_Data\Image_Data\T2Flair\Post1\010
Loading from .\Example_Data\Image_Data\T2Flair\Post1\011
Loading from .\Example_Data\Image_Data\T2Flair\Post1\012
Loading from .\Example_Data\Image_Data\T2Flair\Post2\001
Loading from .\Example_Data\Image_Data\T2Flair\Post2\002
Loading from .\Example_Data\Image_Data\T2Flair\Post2\003
Loading from .\Example_Data\Image_Data\T2Flair\Post2\004
Loading from .\Example_Data\Image_Data\T2Flair\Post2\005
Loading from .\Example_Data\Image_Data\T2Flair\Post2\006
Loading from .\Example_Data\Image_Data\T2Flair\Post2\007
Loading from .\Example_Data\Image_Data\T2Flair\Post2\008
Loading from .\Example_Data\Image_Data\T2Flair\Post2\009
Loading from .\Example_Data\Image_Data\T2Flair\Post2\010
Loading from .\Example_Data\Image_Data\T2Flair\Post2\011
Loading from .\Example_Data\Image_Data\T2Flair\Post2\012
Loading from .\Example_Data\Image_Data\T2Flair\Pre\001
Loading from .\Example_Data\Image_Data\T2Flair\Pre\002
Loading from .\Example_Data\Image_Data\T2Flair\Pre\003
Loading from .\Example_Data\Image_Data\T2Flair\Pre\004
Loading from .\Example_Data\Image_Data\T2Flair\Pre\005
Loading from .\Example_Data\Image_Data\T2Flair\Pre\006
Loading from .\Example_Data\Image_Data\T2Flair\Pre\007
Loading from .\Example_Data\Image_Data\T2Flair\Pre\008
Loading from .\Example_Data\Image_Data\T2Flair\Pre\009
Loading from .\Example_Data\Image_Data\T2Flair\Pre\010
Loading from .\Example_Data\Image_Data\T2Flair\Pre\011
Loading from .\Example_Data\Image_Data\T2Flair\Pre\012
Compiling dictionaries together…
Index 0, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\002
Index 1, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\004
Index 2, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\006
```

```
Index 3, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\005
Index 4, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\006
Index 5, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\007
Index 6, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\006
Index 7, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\009
Index 8, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\010
Index 9, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\010
Index 10, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\004
Index 11, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\003
Index 12, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\002
Index 13, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\005
Index 14, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\009
Index 15, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\008
Index 16, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\009
Index 17, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\012
Index 18, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\010
Index 19, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\011
Index 20, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\011
Index 21, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\004
Index 22, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\007
Index 23, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\007
Index 24, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\003
Index 25, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\012
Index 26, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\008
```

```
Index 27, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post1\001
Index 28, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\005
Index 29, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\008
Index 30, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\001
Index 31, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Pre\003
Index 32, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post1\001
Index 33, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Pre\002
Index 34, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\005
Index 35, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Pre\005
Index 36, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post1\008
Index 37, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post1\006
Index 38, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\006
Index 39, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Pre\008
Index 40, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post1\005
Index 41, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Pre\001
Index 42, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Pre\009
Index 43, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post1\009
Index 44, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post1\007
Index 45, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Pre\006
Index 46, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Pre\007
Index 47, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\001
Index 48, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post1\002
Index 49, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Pre\004
Index 50, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post1\003
Index 51, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post2\004
Index 52, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post2\001
Index 53, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post1\004
```

```
Index 54, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\011
Index 55, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Pre\012
Index 56, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\011
Index 57, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\008
Index 58, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Pre\010
Index 59, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\010
Index 60, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Pre\011
Index 61, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post1\011
Index 62, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post1\010
Index 63, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\007
Index 64, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post1\012
Index 65, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Pre\012
Index 66, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\009
Index 67, description ax T1 3D 1MM  +c at .\Example_Data\Image_Data\T1\Post2\012
Index 68, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\002
Index 69, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post2\002
Index 70, description Ax T2Flair Propeller at
.\Example_Data\Image_Data\T2Flair\Post2\003
Index 71, description ax T1 3D 1MM fSPGR +C  at
.\Example_Data\Image_Data\T1\Post2\003
Index 72, description None at None
Index 73, description None at None
74 unique series IDs were found. Default is index 0, to change use
set_index(index)
Wall time: 5.31 s
```

[8]:
```
all_rois = Dicom_reader.return_rois(print_rois=True)  # Return a list of all␣
↪rois present, and print them
```

```
The following ROIs were found
rttempglioma
exprttempglioma
brainstem
dose 500[cgy]
dose 1000[cgy]
dose 1200[cgy]
gtvplus2
expltparrecgliom
ltparrecglioma
expltfrontrecao
ltfrontrecao
body
expltfrparrecgbm
ltfrparrecgbm
```

```
explttempglioma
lttempglioma
exprtfrontrecgbm
rtfrontrecgbm
expinfrttemprecg
infrttempgbm
dose 2400[cgy]
expltfrontgbm
ltfrontgbm
exprttemprecglio
rttemprecglioma
rtfrontrecglioma
exprtfrontrecgli
brainstem1
eye, left
eye, right
chiasm
lens, left
lens, right
optic nerve, rig
optic nerve, lef
dose 2500[cgy]
exprttemprecgbm
rttemprecgbm
exprtfrparresxn
right_front_par_
abv
abv_roi
```

As we can see, these ROIs correspond to a variety of structures. In particular, we can see many GBM and glioma structures. Note GBM denotes glioblastoma multiforme (a high grade glioma).

[9]:
```
# Print the locations of all RTs with a certain ROI name, automatically lower␣
 ↪cased
Dicom_reader.where_is_ROI(ROIName='BrAiNsTeM1')
```

```
Contours of brainstem1 are located:
.\Example_Data\Image_Data\Structure\T1\001\RS.CA1756_T13D.dcm
.\Example_Data\Image_Data\Structure\T1\011\RS.GF6065_T13D.dcm
.\Example_Data\Image_Data\Structure\T2Flair\001\RS.CA1756_T2Flair.dcm
.\Example_Data\Image_Data\Structure\T2Flair\011\RS.GF6065_T2Flairdcm.dcm
.\Example_Data\Image_Data\T1\001\RS.CA1756_T13D.dcm
```

[10]:
```
Dicom_reader.which_indexes_have_all_rois()  # Check to see which indexes have␣
 ↪all of the rois we want
# Since we haven't defined anything yet, it prompts you to input a list of␣
 ↪contour names
```

```
You need to first define what ROIs you want, please use
```

```
.set_contour_names_and_associations(roi_list)
```

From these ROIs, we will look for those that describe the following regions of interest: tumor (glioblastoma multiforme only) and high-dose area of radiation therapy.

[11]:
```
Contour_Names = ['tumor', 'high_dose']
# Associations work as {'variant_name': 'desired_name'}
associations = {'dose 1000[cgy]': 'high_dose', 'dose 1200[cgy]': 'high_dose', #␣
 ↪new high dose ROI through association
                'exprtfrontrecgbm': 'tumor', 'rtfrontrecgbm': 'tumor',␣
 ↪'expltfrontgbm': 'tumor', # associating gbms to tumor
                 'ltfrontgbm': 'tumor','infrttempgbm': 'tumor', 'rttemprecgbm':␣
 ↪'tumor',
                'exprttemprecgbm': 'tumor','expltfrparrecgbm': 'tumor',␣
 ↪'ltfrparrecgbm': 'tumor'}
```

[12]:
```
Dicom_reader.set_contour_names_and_associations(Contour_Names=Contour_Names,␣
 ↪associations=associations)
```

```
Lacking ['tumor'] in index 0, location
.\Example_Data\Image_Data\T2Flair\Pre\002. Found ['rttempglioma',
'exprttempglioma', 'brainstem', 'dose 500[cgy]', 'dose 1000[cgy]', 'dose
1200[cgy]', 'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 1, location
.\Example_Data\Image_Data\T2Flair\Post2\004. Found []
Lacking ['tumor'] in index 2, location
.\Example_Data\Image_Data\T2Flair\Pre\006. Found ['brainstem', 'dose 500[cgy]',
'dose 1200[cgy]', 'dose 1000[cgy]', 'expltparrecgliom', 'ltparrecglioma',
'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 3, location
.\Example_Data\Image_Data\T2Flair\Post1\005. Found []
Lacking ['tumor', 'high_dose'] in index 4, location
.\Example_Data\Image_Data\T2Flair\Post1\006. Found []
Lacking ['tumor'] in index 5, location
.\Example_Data\Image_Data\T2Flair\Pre\007. Found ['brainstem', 'dose 500[cgy]',
'dose 1000[cgy]', 'dose 1200[cgy]', 'expltfrontrecao', 'ltfrontrecao', 'body',
'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 6, location
.\Example_Data\Image_Data\T2Flair\Post2\006. Found []
Lacking ['tumor', 'high_dose'] in index 8, location
.\Example_Data\Image_Data\T2Flair\Post2\010. Found []
Lacking ['tumor', 'high_dose'] in index 9, location
.\Example_Data\Image_Data\T2Flair\Post1\010. Found []
Lacking ['tumor'] in index 10, location
.\Example_Data\Image_Data\T2Flair\Pre\004. Found ['brainstem', 'dose 1000[cgy]',
'dose 1200[cgy]', 'dose 500[cgy]', 'explttempglioma', 'lttempglioma',
'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 12, location
```

```
.\Example_Data\Image_Data\T2Flair\Post1\002. Found []
Lacking ['tumor', 'high_dose'] in index 13, location
.\Example_Data\Image_Data\T2Flair\Post2\005. Found []
Lacking ['tumor', 'high_dose'] in index 14, location
.\Example_Data\Image_Data\T2Flair\Post1\009. Found []
Lacking ['tumor', 'high_dose'] in index 15, location
.\Example_Data\Image_Data\T2Flair\Post2\008. Found []
Lacking ['tumor', 'high_dose'] in index 16, location
.\Example_Data\Image_Data\T2Flair\Post2\009. Found []
Lacking ['tumor', 'high_dose'] in index 17, location
.\Example_Data\Image_Data\T2Flair\Post2\012. Found []
Lacking ['tumor', 'high_dose'] in index 19, location
.\Example_Data\Image_Data\T2Flair\Post2\011. Found []
Lacking ['tumor', 'high_dose'] in index 20, location
.\Example_Data\Image_Data\T2Flair\Post1\011. Found []
Lacking ['tumor', 'high_dose'] in index 21, location
.\Example_Data\Image_Data\T2Flair\Post1\004. Found []
Lacking ['tumor', 'high_dose'] in index 22, location
.\Example_Data\Image_Data\T2Flair\Post1\007. Found []
Lacking ['tumor', 'high_dose'] in index 23, location
.\Example_Data\Image_Data\T2Flair\Post2\007. Found []
Lacking ['tumor', 'high_dose'] in index 24, location
.\Example_Data\Image_Data\T2Flair\Post1\003. Found []
Lacking ['tumor', 'high_dose'] in index 25, location
.\Example_Data\Image_Data\T2Flair\Post1\012. Found []
Lacking ['tumor', 'high_dose'] in index 26, location
.\Example_Data\Image_Data\T2Flair\Post1\008. Found []
Lacking ['tumor', 'high_dose'] in index 27, location
.\Example_Data\Image_Data\T2Flair\Post1\001. Found []
Lacking ['tumor'] in index 29, location
.\Example_Data\Image_Data\T2Flair\Pre\008. Found ['brainstem', 'dose 1000[cgy]',
'dose 1200[cgy]', 'dose 500[cgy]', 'exprttemprecglio', 'rttemprecglioma',
'gtvplus2']
Lacking ['tumor'] in index 30, location
.\Example_Data\Image_Data\T2Flair\Pre\001. Found ['rtfrontrecglioma',
'exprtfrontrecgli', 'brainstem1', 'dose 1000[cgy]', 'dose 1200[cgy]', 'dose
500[cgy]', 'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 32, location
.\Example_Data\Image_Data\T1\Post1\001. Found []
Lacking ['tumor'] in index 33, location .\Example_Data\Image_Data\T1\Pre\002.
Found ['exprttempglioma', 'rttempglioma', 'brainstem', 'dose 500[cgy]', 'dose
1000[cgy]', 'dose 1200[cgy]', 'body', 'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 34, location
.\Example_Data\Image_Data\T1\Post2\005. Found []
Lacking ['tumor', 'high_dose'] in index 36, location
.\Example_Data\Image_Data\T1\Post1\008. Found []
Lacking ['tumor', 'high_dose'] in index 37, location
.\Example_Data\Image_Data\T1\Post1\006. Found []
```

```
Lacking ['tumor', 'high_dose'] in index 38, location
.\Example_Data\Image_Data\T1\Post2\006. Found []
Lacking ['tumor'] in index 39, location .\Example_Data\Image_Data\T1\Pre\008.
Found ['eye, left', 'eye, right', 'brainstem', 'chiasm', 'lens, left', 'lens,
right', 'rttemprecglioma', 'exprttemprecglio', 'dose 500[cgy]', 'dose
1000[cgy]', 'dose 1200[cgy]', 'gtvplus2', 'body']
Lacking ['tumor', 'high_dose'] in index 40, location
.\Example_Data\Image_Data\T1\Post1\005. Found []
Lacking ['tumor'] in index 41, location .\Example_Data\Image_Data\T1\Pre\001.
Found ['exprtfrontrecgli', 'rtfrontrecglioma', 'brainstem1', 'dose 500[cgy]',
'dose 1200[cgy]', 'dose 1000[cgy]', 'body', 'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 42, location
.\Example_Data\Image_Data\T1\Pre\009. Found []
Lacking ['tumor', 'high_dose'] in index 43, location
.\Example_Data\Image_Data\T1\Post1\009. Found []
Lacking ['tumor', 'high_dose'] in index 44, location
.\Example_Data\Image_Data\T1\Post1\007. Found []
Lacking ['tumor'] in index 45, location .\Example_Data\Image_Data\T1\Pre\006.
Found ['eye, left', 'lens, right', 'lens, left', 'ltparrecglioma', 'chiasm',
'optic nerve, rig', 'optic nerve, lef', 'expltparrecgliom', 'brainstem', 'eye,
right', 'dose 500[cgy]', 'dose 1000[cgy]', 'dose 1200[cgy]', 'gtvplus2']
Lacking ['tumor'] in index 46, location .\Example_Data\Image_Data\T1\Pre\007.
Found ['brainstem', 'chiasm', 'expltfrontrecao', 'eye, left', 'eye, right',
'lens, left', 'lens, right', 'ltfrontrecao', 'optic nerve, lef', 'optic nerve,
rig', 'dose 2500[cgy]', 'dose 500[cgy]', 'dose 1000[cgy]', 'dose 1200[cgy]',
'gtvplus2', 'body']
Lacking ['tumor', 'high_dose'] in index 47, location
.\Example_Data\Image_Data\T2Flair\Post2\001. Found []
Lacking ['tumor', 'high_dose'] in index 48, location
.\Example_Data\Image_Data\T1\Post1\002. Found []
Lacking ['tumor'] in index 49, location .\Example_Data\Image_Data\T1\Pre\004.
Found ['brainstem', 'dose 1000[cgy]', 'dose 1200[cgy]', 'dose 500[cgy]',
'explttempglioma', 'lttempglioma', 'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 50, location
.\Example_Data\Image_Data\T1\Post1\003. Found []
Lacking ['tumor', 'high_dose'] in index 51, location
.\Example_Data\Image_Data\T1\Post2\004. Found []
Lacking ['tumor', 'high_dose'] in index 52, location
.\Example_Data\Image_Data\T1\Post2\001. Found []
Lacking ['tumor', 'high_dose'] in index 53, location
.\Example_Data\Image_Data\T1\Post1\004. Found []
Lacking ['tumor'] in index 55, location
.\Example_Data\Image_Data\T2Flair\Pre\012. Found ['brainstem', 'chiasm', 'dose
1000[cgy]', 'dose 1200[cgy]', 'dose 500[cgy]', 'exprtfrparresxn', 'gtvplus2',
'right_front_par_']
Lacking ['tumor', 'high_dose'] in index 56, location
.\Example_Data\Image_Data\T1\Post2\011. Found []
Lacking ['tumor', 'high_dose'] in index 57, location
```

```
.\Example_Data\Image_Data\T1\Post2\008. Found []
Lacking ['tumor', 'high_dose'] in index 59, location
.\Example_Data\Image_Data\T1\Post2\010. Found []
Lacking ['tumor', 'high_dose'] in index 61, location
.\Example_Data\Image_Data\T1\Post1\011. Found []
Lacking ['tumor', 'high_dose'] in index 62, location
.\Example_Data\Image_Data\T1\Post1\010. Found []
Lacking ['tumor', 'high_dose'] in index 63, location
.\Example_Data\Image_Data\T1\Post2\007. Found []
Lacking ['tumor', 'high_dose'] in index 64, location
.\Example_Data\Image_Data\T1\Post1\012. Found []
Lacking ['tumor'] in index 65, location .\Example_Data\Image_Data\T1\Pre\012.
Found ['brainstem', 'chiasm', 'exprtfrparresxn', 'right_front_par_', 'body',
'dose 500[cgy]', 'dose 1000[cgy]', 'dose 1200[cgy]', 'abv', 'abv_roi',
'gtvplus2']
Lacking ['tumor', 'high_dose'] in index 66, location
.\Example_Data\Image_Data\T1\Post2\009. Found []
Lacking ['tumor', 'high_dose'] in index 67, location
.\Example_Data\Image_Data\T1\Post2\012. Found []
Lacking ['tumor', 'high_dose'] in index 68, location
.\Example_Data\Image_Data\T2Flair\Post2\002. Found []
Lacking ['tumor', 'high_dose'] in index 69, location
.\Example_Data\Image_Data\T1\Post2\002. Found []
Lacking ['tumor', 'high_dose'] in index 70, location
.\Example_Data\Image_Data\T2Flair\Post2\003. Found []
Lacking ['tumor', 'high_dose'] in index 71, location
.\Example_Data\Image_Data\T1\Post2\003. Found []
```

Note: The module is printing "Found []" because many of the scans (post-1 and post-2 RT) do not have associated structure files. The module recognizes these images exist (unique UIDs) but associated structure files cannot be located for them.

[13]:
```
indexes = Dicom_reader.which_indexes_have_all_rois()  # Check to see which␣
↪indexes have all of the rois we want, now we can see indexes
```

```
The following indexes have all ROIs present
Index 7, located at .\Example_Data\Image_Data\T2Flair\Pre\009
Index 11, located at .\Example_Data\Image_Data\T2Flair\Pre\003
Index 18, located at .\Example_Data\Image_Data\T2Flair\Pre\010
Index 28, located at .\Example_Data\Image_Data\T2Flair\Pre\005
Index 31, located at .\Example_Data\Image_Data\T1\Pre\003
Index 35, located at .\Example_Data\Image_Data\T1\Pre\005
Index 54, located at .\Example_Data\Image_Data\T2Flair\Pre\011
Index 58, located at .\Example_Data\Image_Data\T1\Pre\010
Index 60, located at .\Example_Data\Image_Data\T1\Pre\011
```

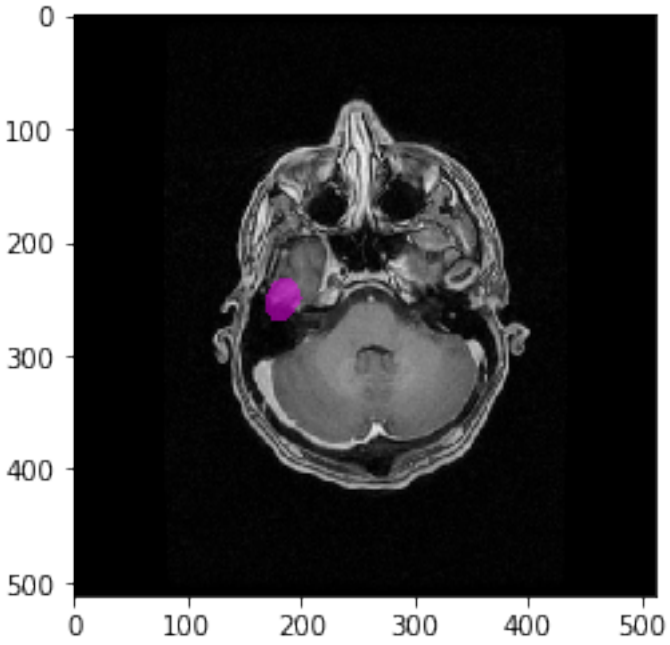
[14]:
```
pt_indx = indexes[-1]
```

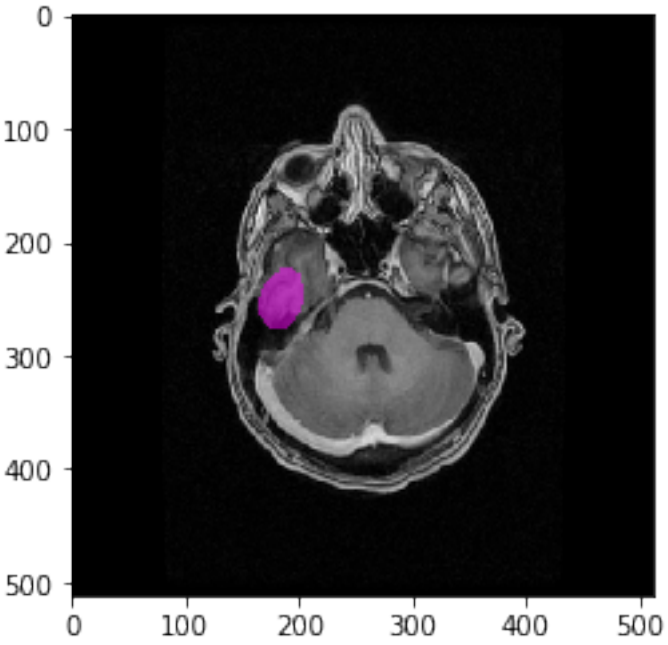
```
Dicom_reader.set_index(pt_indx)  # This index has all the structures,␣
 ↪corresponds to pre-RT T1-w image for patient 011
Dicom_reader.get_images_and_mask()  # Load up the images and mask for the␣
 ↪requested index
```

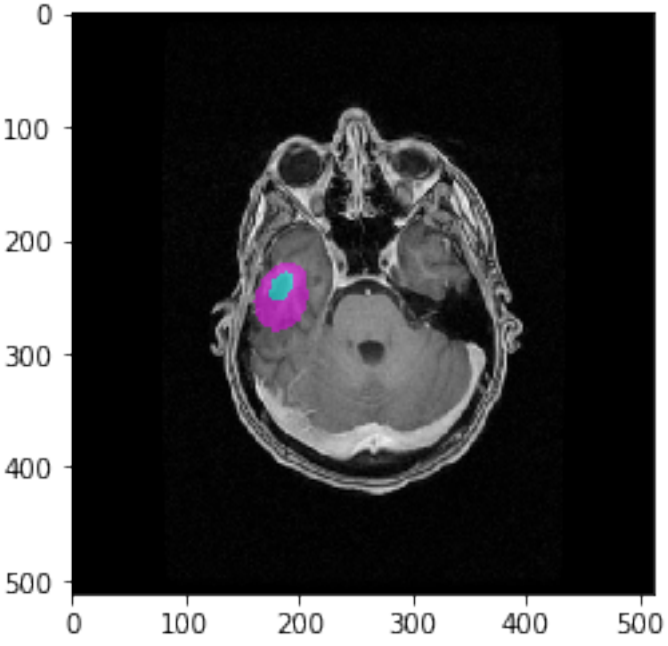
```
Loading images for ax T1 3D 1MM  +c at
 .\Example_Data\Image_Data\T1\Pre\011
```

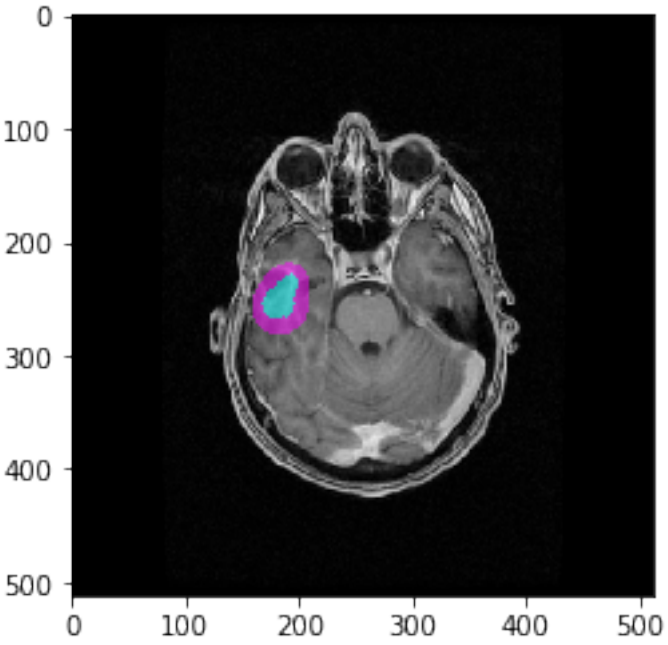
[15]:
```
image = Dicom_reader.ArrayDicom # image array
mask = Dicom_reader.mask # mask array
dicom_sitk_handle = Dicom_reader.dicom_handle # SimpleITK image handle
mask_sitk_handle = Dicom_reader.annotation_handle # SimpleITK mask handle
```

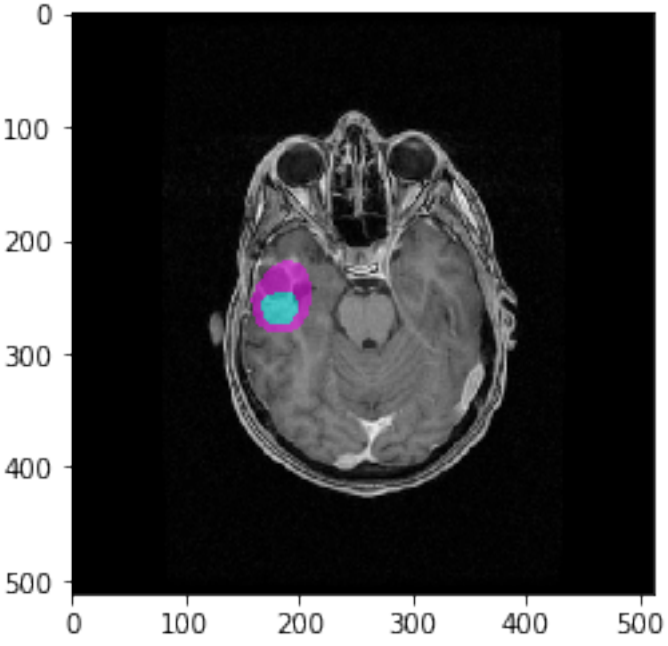
[16]:
```
n_slices_skip = 4
display_slices(image, mask, skip = n_slices_skip) # visualize that our␣
 ↪segmentations were succesfully convereted
```

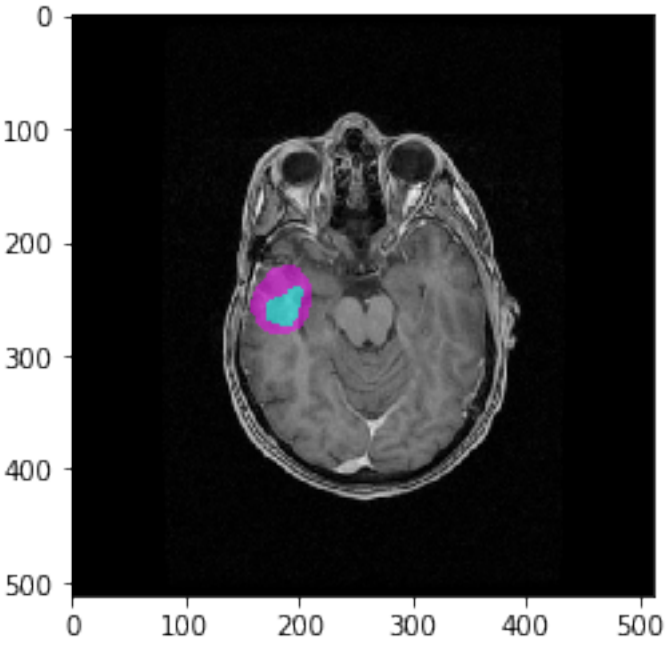


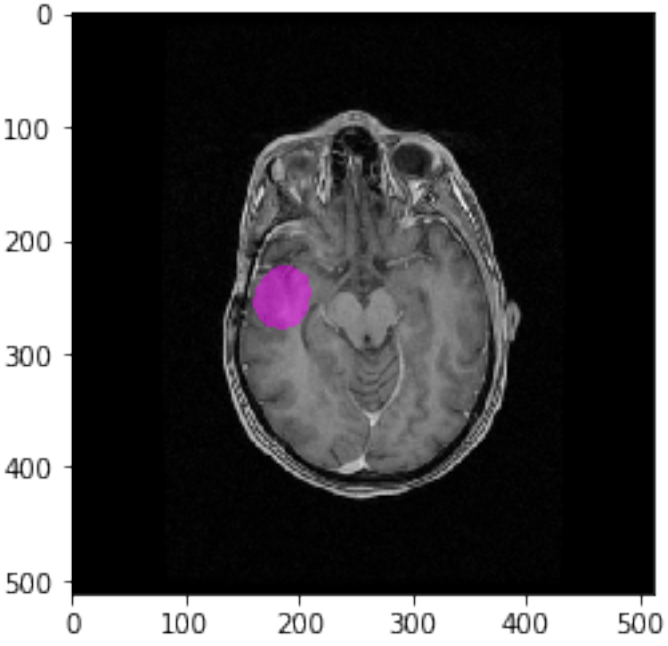

0
100
200
300
400
500
0
100
200
300
400
500

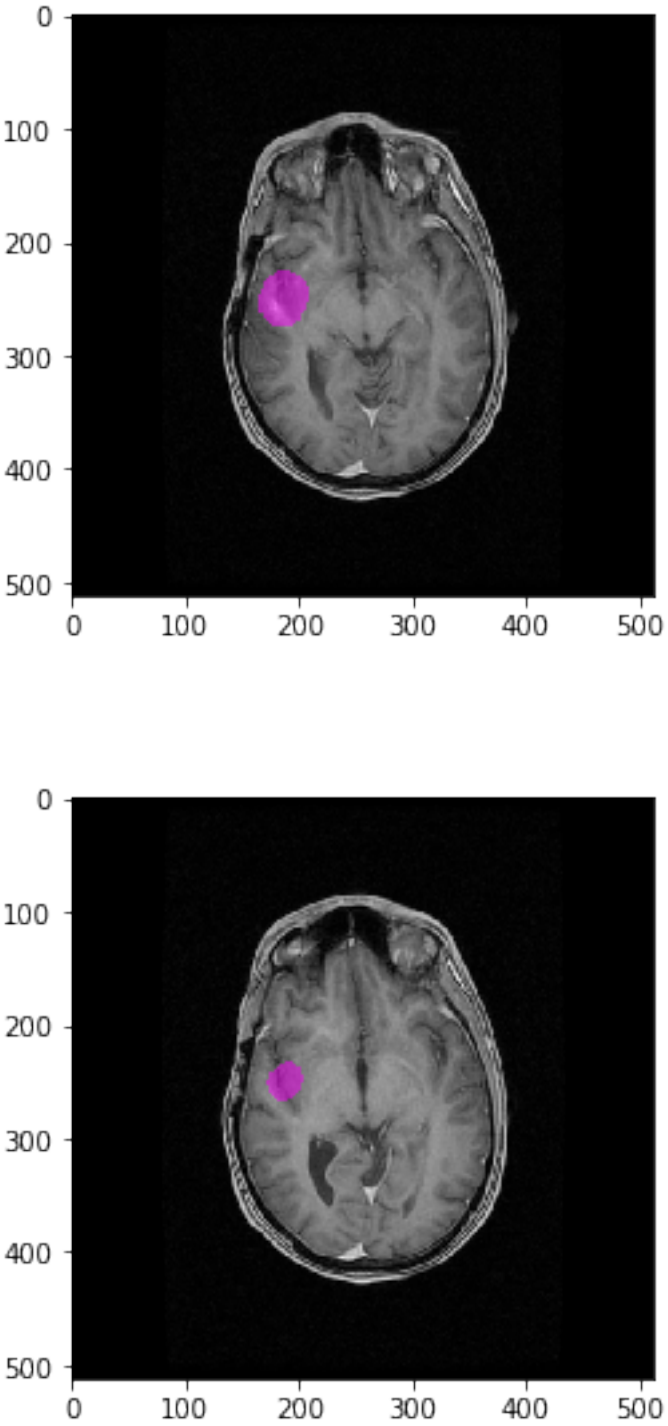

0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500



Note: Cyan color denotes tumor while magenta denotes surrounding area of high-dose radiation. Only displaying 7 slices.

1.3 Part 3: Saving arrays to nifti format.

If you want to use a manual approach, you can view the nifti files easily after running get_images_and_mask(). Saving files as nifti is advisable since spacing information is preserved.

[17]:
```
nifti_path = os.path.join('.', 'Example_Data', 'Nifti_Data') # nifti subfolder
if not os.path.exists(nifti_path):
    os.makedirs(nifti_path)
```

[18]:
```
dicom_sitk_handle = Dicom_reader.dicom_handle # SimpleITK image handle
mask_sitk_handle = Dicom_reader.annotation_handle # SimpleITK mask handle
sitk.WriteImage(dicom_sitk_handle, os.path.join(nifti_path, 'Image.nii'))
sitk.WriteImage(mask_sitk_handle, os.path.join(nifti_path, 'Mask.nii'))
```

One can also use the built in .write_parallel attribute to generate nifti files for all relevant pairs the DicomReaderWriter object has found/generated. In this case there are 9 image/mask pairs for unique UIDs that contain all contours we are interested in. Note a corresponding log excel file in the specified output path. The nifti files are written in the following format: "Overall_Data_{description}_ {iteration}.nii.gz" (image) or "Overall_mask_{description}_ y{iteration}.nii.gz" (mask).

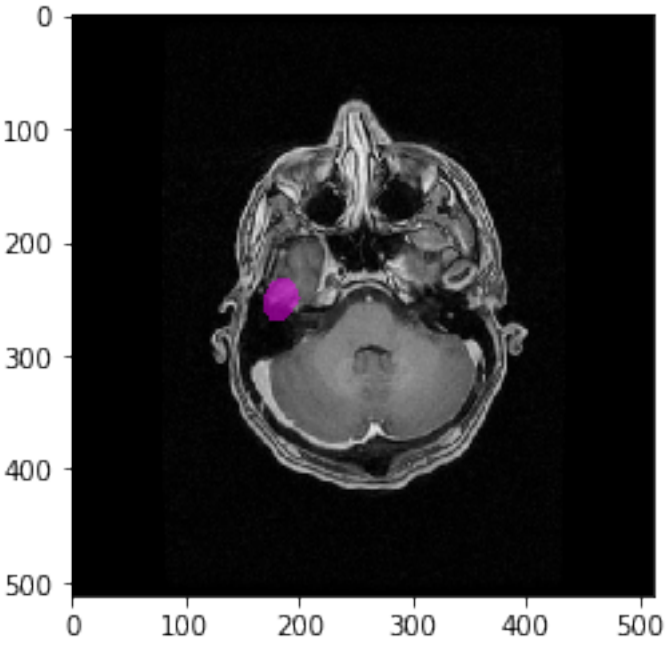
[19]:
```
%%time
%%capture
Dicom_reader.write_parallel(out_path = nifti_path, excel_file = os.path.
 ↪join(nifti_path,'.','MRN_Path_To_Iteration.xlsx'))
```

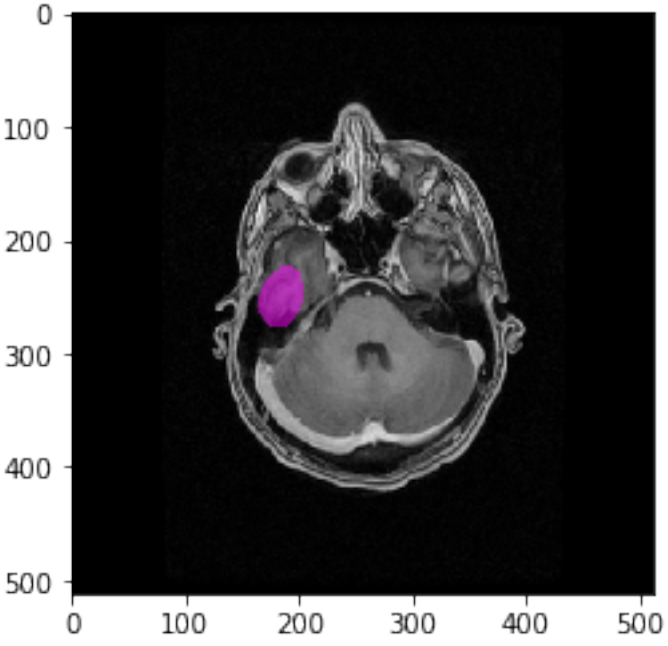
```
Wall time: 662 ms
```

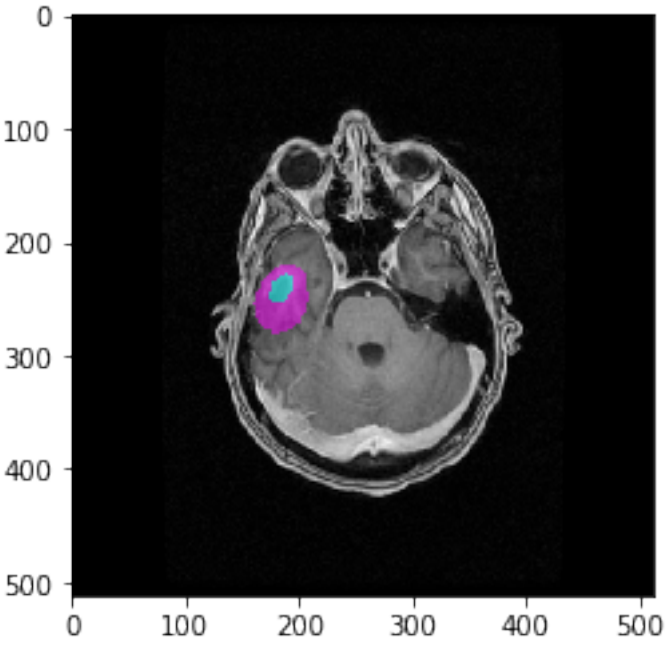
We can now reload the nifti files and disaply them to check that nothing went wrong. You can inspect the other converted files by changing the numerical suffix as per the excel log file ('MRN_Path_To_Iteration.xlsx').

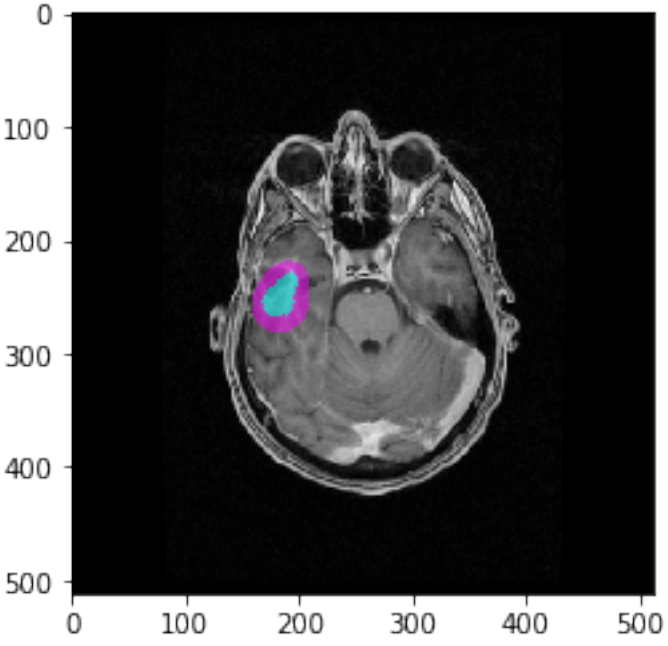
[20]:
```
nifti_image = sitk.ReadImage(os.path.join(nifti_path,"Overall_Data_Examples_8.
 ↪nii.gz")) # reload image
image = sitk.GetArrayFromImage(nifti_image)
nifti_mask = sitk.ReadImage(os.path.join(nifti_path,"Overall_mask_Examples_y8.
 ↪nii.gz")) # reload mask
mask = sitk.GetArrayFromImage(nifti_mask)
```

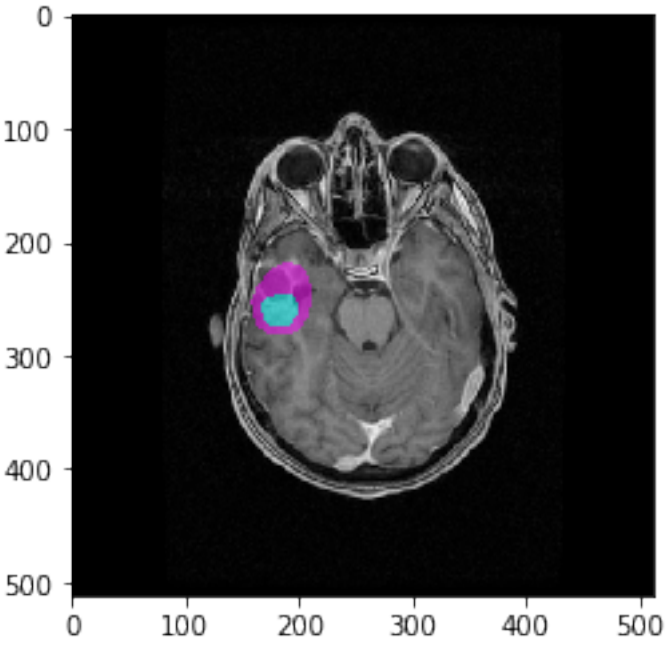
[21]:
```
display_slices(image, mask, skip = n_slices_skip) # visualize that our␣
 ↪segmentations were succesfully convereted from nifti
```

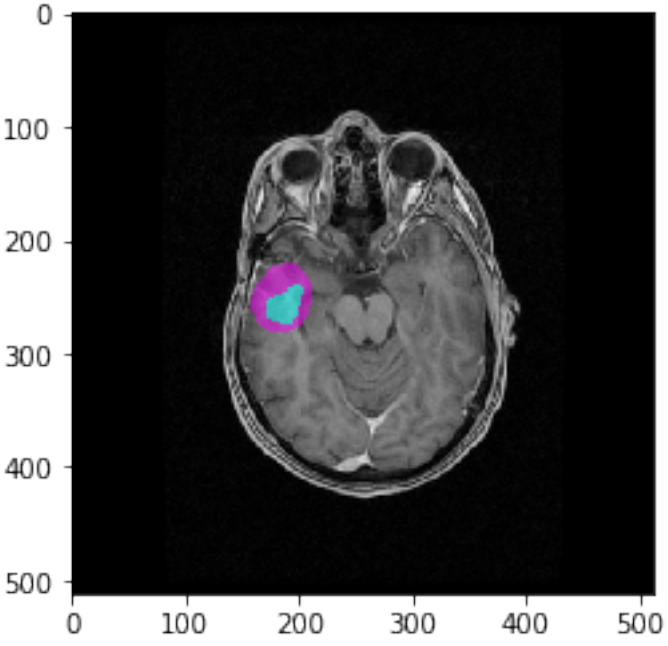

0
100
200
300
400
500
0
100
200
300
400
500

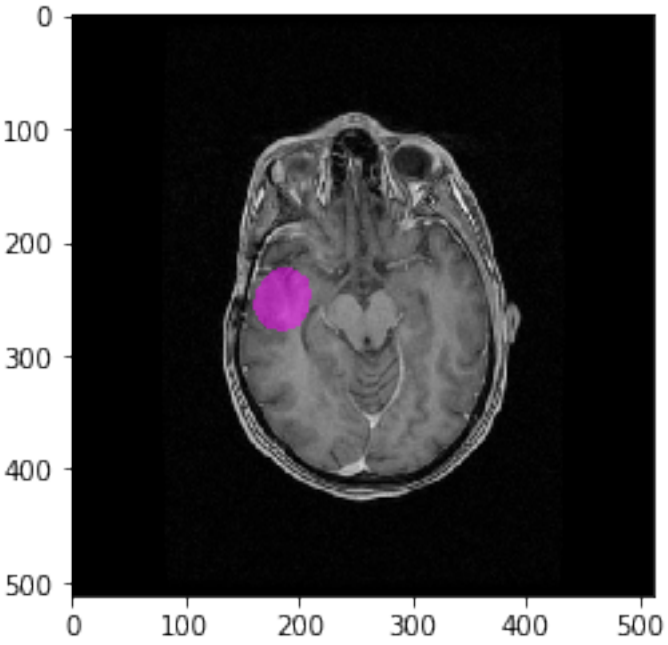

0
100
200
300
400
500
0
100
200
300
400
500

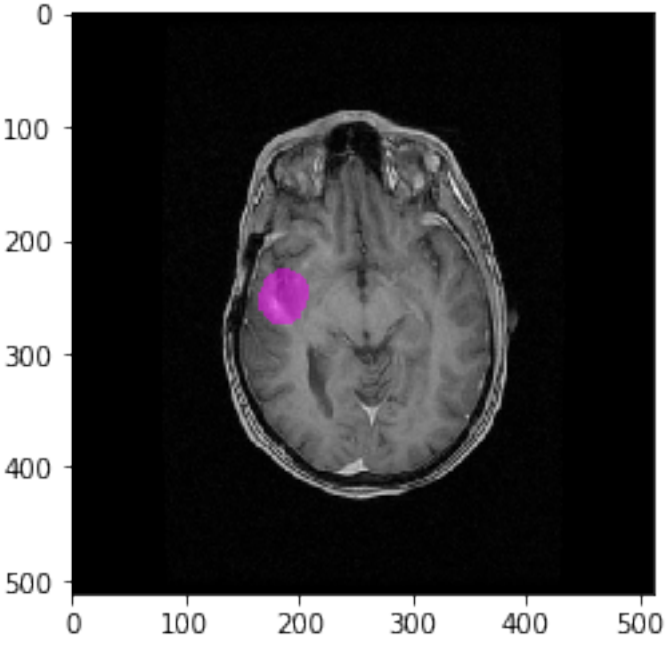

0
100
200
300
400
500
0
100
200
300
400
500

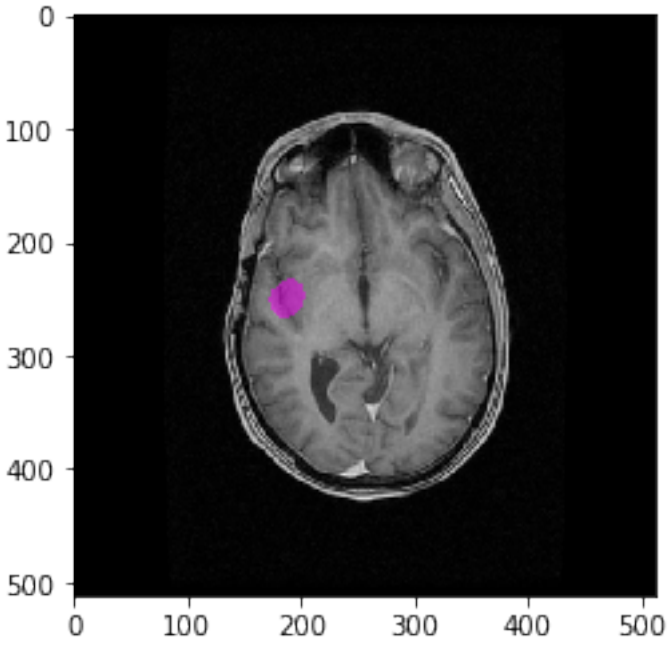

0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500



1.4 Part 4: Saving and loading numpy files for later use.

Finally we can save the numpy arrays themselves to files for later use (so you don't have to reinstantiate the computationally expensive DicomReaderWriter object) and subsequently re-load the numpy arrays.

[22]:
```
numpy_path = os.path.join(data_path, 'Numpy_Data') # go into numpy subfolder
if not os.path.exists(numpy_path):
    os.makedirs(numpy_path)
```

[23]:
```
np.save(os.path.join(numpy_path, 'image'), image) # save the arrays
np.save(os.path.join(numpy_path, 'mask'), mask)
```

[24]:
```
image = np.load(os.path.join(numpy_path,'image.npy')) # load the arrays
mask = np.load(os.path.join(numpy_path,'mask.npy'))
```

1.5 Part 5: Radiomics Use-case Example.

Here we use the popular open-source radiomics library PyRadiomics (https://pyradiomics.readthedocs.io/en/latest/) to calculate radiomic features for our ROIs. In this case, we only calculate a limited number features from the tumor as an illustrative example.

[25]:
```
try:
    from radiomics import featureextractor
except:
    !pip install pyradiomics
```

```
    from radiomics import featureextractor
```

[26]:
```
pd.set_option('display.max_columns', None) # show all columns
```

[27]:
```
%%time
# note: need sitk images (sitk.ReadImage(nifti file)) to plug into PyRadiomics,␣
 ↪preserves spacing

ROI_index = 1 # index for tumor
nifti_mask_tumor = sitk.BinaryThreshold(nifti_mask, lowerThreshold=ROI_index,␣
 ↪upperThreshold=ROI_index) # select only ROI of interest

params = {} # can edit in more params as neccessary
extractor = featureextractor.RadiomicsFeatureExtractor(**params) # instantiate␣
 ↪extractor with parameters
extractor.disableAllFeatures() # in case where only want some features, can␣
 ↪delete disable/enable lines if you want deafult
extractor.enableFeatureClassByName('firstorder')
extractor.enableFeatureClassByName('glcm')
features = {} # empty dictionary
features = extractor.execute(nifti_image, nifti_mask_tumor) # unpack results␣
 ↪into features dictionary
df = pd.DataFrame({k: [v] for k, v in features.items()}) # put dictionary into␣
 ↪a dataframe
```

```
GLCM is symmetrical, therefore Sum Average = 2 * Joint Average, only 1 needs to
be calculated

Wall time: 9.02 s
```

[28]:
```
df # display dataframe to inspect features
```

[28]:
```
  diagnostics_Versions_PyRadiomics diagnostics_Versions_Numpy  \
0                           v3.0.1                      1.19.5

  diagnostics_Versions_SimpleITK diagnostics_Versions_PyWavelet  \
0                          2.0.2                          1.1.1

  diagnostics_Versions_Python  \
0                       3.6.8

                  diagnostics_Configuration_Settings  \
0  {'minimumROIDimensions': 2, 'minimumROISize': …

  diagnostics_Configuration_EnabledImageTypes  \
0                           {'Original': {}}

            diagnostics_Image-original_Hash  \
```

```
0  adcd47a617bf0a6906a361ba78d7e86388bc2fc9

  diagnostics_Image-original_Dimensionality  \
0                                        3D

             diagnostics_Image-original_Spacing  \
0  (0.5859000086784363, 0.5859000086784363, 1.0)

  diagnostics_Image-original_Size  diagnostics_Image-original_Mean  \
0                (512, 512, 192)                       138.125106

   diagnostics_Image-original_Minimum  diagnostics_Image-original_Maximum  \
0                                 0.0                                1936.0

            diagnostics_Mask-original_Hash  \
0  ca99975f7f6ce5b9da272d5b065bb0fd8d40810e

              diagnostics_Mask-original_Spacing  \
0  (0.5859000086784363, 0.5859000086784363, 1.0)

  diagnostics_Mask-original_Size diagnostics_Mask-original_BoundingBox  \
0               (512, 512, 192)            (167, 225, 68, 37, 50, 16)

   diagnostics_Mask-original_VoxelNum  diagnostics_Mask-original_VolumeNum  \
0                               10905                                    1

         diagnostics_Mask-original_CenterOfMassIndex  \
0  (184.79257221458047, 254.31728564878497, 75.06…

             diagnostics_Mask-original_CenterOfMass  \
0  (-41.43703106818862, 1.4954971831525938, -7.49…

  original_firstorder_10Percentile original_firstorder_90Percentile  \
0                            659.0                            1042.0

  original_firstorder_Energy original_firstorder_Entropy  \
0              8464051347.0           4.503961941913401

  original_firstorder_InterquartileRange original_firstorder_Kurtosis  \
0                                  188.0            3.056529328502632

  original_firstorder_Maximum original_firstorder_MeanAbsoluteDeviation  \
0                      1292.0                        114.14490366028481

  original_firstorder_Mean original_firstorder_Median  \
0        869.2987620357634                       888.0
```

```
  original_firstorder_Minimum original_firstorder_Range  \
0                       311.0                     981.0

  original_firstorder_RobustMeanAbsoluteDeviation  \
0                              80.00459220139915

  original_firstorder_RootMeanSquared original_firstorder_Skewness  \
0                    881.000814067669          -0.5230388203250016

  original_firstorder_TotalEnergy original_firstorder_Uniformity  \
0               2905529560.251311           0.051218440447186646

  original_firstorder_Variance original_glcm_Autocorrelation  \
0           20482.096710984642              579.312440764934

  original_glcm_ClusterProminence original_glcm_ClusterShade  \
0               42908.67954319103         -693.8616497595558

  original_glcm_ClusterTendency original_glcm_Contrast  \
0            118.47853188255739       8.74050977809287

  original_glcm_Correlation original_glcm_DifferenceAverage  \
0        0.8617988560957099               2.128732075933059

  original_glcm_DifferenceEntropy original_glcm_DifferenceVariance  \
0              2.6510141072718554               3.9429016300371384

     original_glcm_Id   original_glcm_Idm  original_glcm_Idmn  \
0  0.4588122607941779  0.3890481464071379  0.9946965137645681

    original_glcm_Idn    original_glcm_Imc1  original_glcm_Imc2  \
0  0.9515499470890418  -0.25514812891031574  0.9358524694499407

  original_glcm_InverseVariance original_glcm_JointAverage  \
0           0.37803667377261313         23.491920944502194

  original_glcm_JointEnergy original_glcm_JointEntropy   original_glcm_MCC  \
0      0.006641162131651515          7.807733854014718  0.8696978190928191

  original_glcm_MaximumProbability original_glcm_SumAverage  \
0              0.01625971801846341        46.98384188900442

  original_glcm_SumEntropy original_glcm_SumSquares
0         5.42170370906024        31.80476041516255
```

Numerical results for radiomic features shown here are consistent with importing nifti files as image and label map in 3D Slicer (https://www.slicer.org/) and using Radiomics extension (https://www.slicer.org/wiki/Documentation/Nightly/Extensions/Radiomics).

1.6 Part 6: Predictions To RT-Structure Example

Here we will provide a simple example for converting a predicted NumPy array of a square into a Dicom RT-Structure file

[29]:
```
RT_path = os.path.join('Example_Data', 'RT_Structures')
if not os.path.exists(RT_path):
    os.makedirs(RT_path)
```

First, we will create a fake prediction, it will be the same size as the image NumPy array

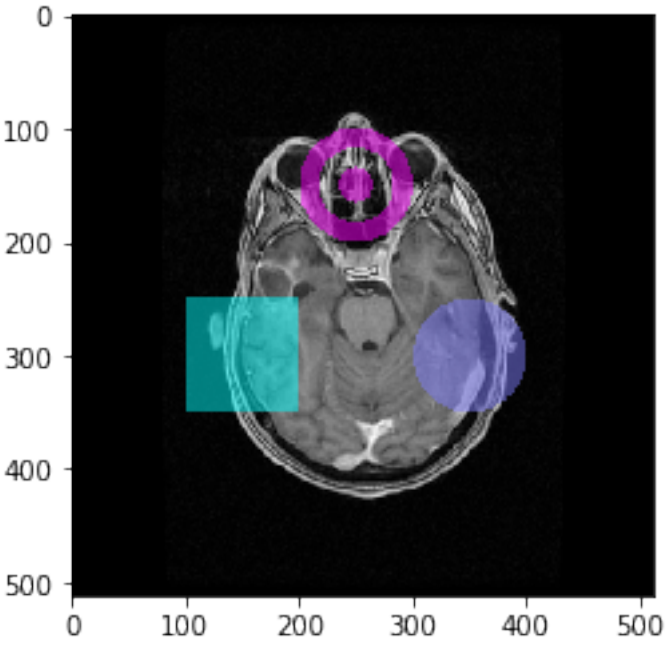
[30]:
```
image = Dicom_reader.ArrayDicom
```

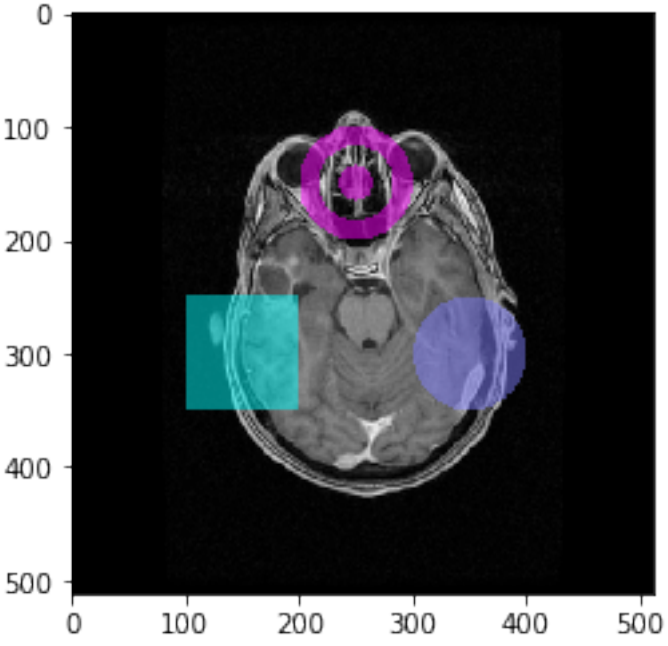
Now, deep learning model typically create segmentations in the format of (z_images, rows, cols, # of classes)

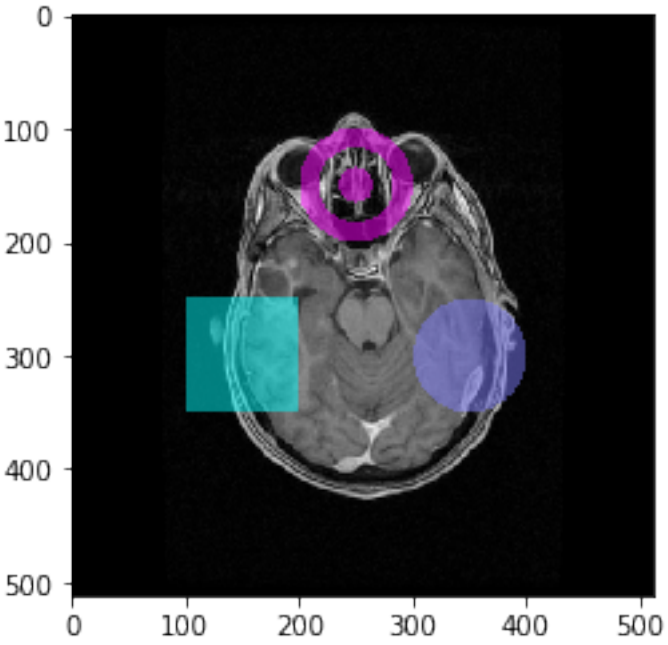
[31]:
```
def create_circular_mask(h, w, center=None, radius=None):

    if center is None: # use the middle of the image
        center = (int(w/2), int(h/2))
    if radius is None: # use the smallest distance between the center and image␣
 ↪walls
        radius = min(center[0], center[1], w-center[0], h-center[1])

    Y, X = np.ogrid[:h, :w]
    dist_from_center = np.sqrt((X - center[0])**2 + (Y-center[1])**2)

    mask = dist_from_center <= radius
    return mask
```

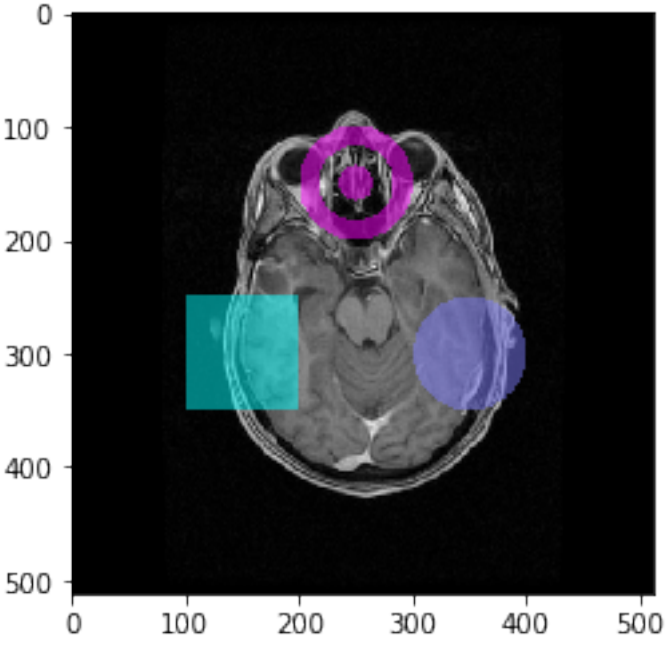
[32]:
```
predictions = np.zeros(image.shape + (4,))  # Four classes: background, square,␣
 ↪circle, target
predictions.shape
predictions[75:80, 250:350, 100:200, 1] = 1  # Here we are drawing a square
predictions[75:80, 250:350, 300:400, 2] += create_circular_mask(100, 100,␣
 ↪center=None, radius=50).astype('int')
predictions[75:80, 100:200, 200:300, 3] += create_circular_mask(100, 100,␣
 ↪center=None, radius=50).astype('int')
predictions[75:80, 100:200, 200:300, 3] -= create_circular_mask(100, 100,␣
 ↪center=None, radius=33).astype('int')
predictions[75:80, 100:200, 200:300, 3] += create_circular_mask(100, 100,␣
 ↪center=None, radius=15).astype('int')
```

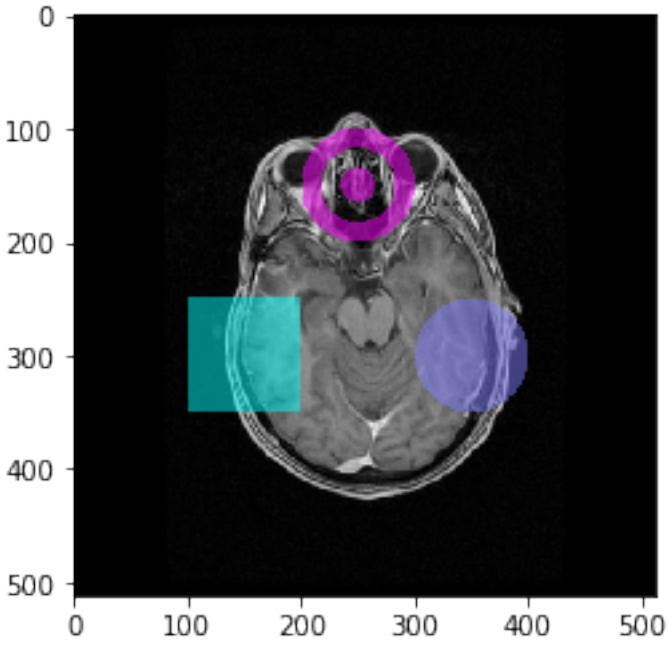
[33]:
```
display_slices(image, np.argmax(predictions, axis=-1), skip = 1) # visualize␣
 ↪our square on the image
```


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500



Convert the NumPy arrays into RT-Structure

[34]:
```
Dicom_reader.prediction_array_to_RT(prediction_array=predictions,␣
↪output_dir=RT_path,
                                    ROI_Names=['square', 'circle', 'target'])
```

```
Running off a template
Writing data for square
Writing data for circle
Writing data for target
Writing out data…Example_Data\RT_Structures
Finished!
```

2 Final notes

2.0.1 I hope that this code has been useful, if you have any suggestions or problems, please open an issue ticket or merge request on the Github: https://github.com/brianmanderson/Dicom_RT_and_Images_to_Mask

Thank you!

[ ]: