Segmenting and Evaluating Intracranial Hemorrhage Region in Non-Contrast Computed Tomography

Christy Johnny - UCID: 30061368, Emma Tholl - UCID: 30066559,
Jessica Ritchie - UCID: 30071655, Tomin Mattam - UCID: 30071513

Introduction

Motivation

Intracranial hemorrhage (ICH) is a life-threatening condition when bleeding happens within the skull due to trauma, stroke, and aneurysm rupture [1]. Annually, more than 20,000 individuals in the United States die of intracranial hemorrhage [2]. The ability to accurately detect and segment ICH from Non-Contrast Computed Tomography (NCCT) images is critical for early diagnosis and treatment of patients. Manually segmenting ICH is a time-consuming and subjective process that can lead to inconsistencies and errors. Therefore, there is a need for research to develop automated methods for ICH segmentation.

The motivation for this project is to develop a tool that can accurately segment ICH from NCCT images using advanced algorithms. Our goal is to improve the accuracy and efficiency of ICH segmentation, thereby improving patient outcomes and reducing the workload of medical professionals. This objective was inspired by the 2022 Instance Challenge, which aims to promote the development of automated methods for ICH segmentation [3].

Background and Objectives

Non-Contrast Computed Tomography (NCCT) is a commonly used imaging modality to diagnose ICH [4]. However, the accurate segmentation of ICH from NCCT images is challenging due to the complex and variable nature of the hemorrhage. Previous methods for ICH segmentation like manual segmentation and thresholding have been shown to be inaccurate and time-consuming. Recent advancements in machine learning techniques, such as Chan-Vese and Simple Linear Iterative Clustering, have shown improved efficiency and accuracy [5]. Therefore, we will develop a program that will use these methods to segment and evaluate ICH from NCCT images. The tool will be evaluated using the relative volume difference (RVD) metric. The results of this project will contribute to the development of more accurate and effective methods for ICH segmentation, which will have an important clinical impact.

Theory

Our project uses advanced algorithms for ICH segmentation from NCCT images. We employ window/level filtering to enhance contrast between the brain and the ICH, based on experiments with various filtering techniques. We utilize three segmentation methods: threshold-based, Chan-Vese, and Simple Linear Iterative Clustering (SLIC). Threshold-based segmentation is simple but may not capture the complex nature of the ICH. Chan-Vese separates the image into two classes but may not capture detailed information due to limited ability to handle complex shapes. SLIC, a clustering-based method, allows for more detailed and accurate segmentation of ICH. We evaluate the accuracy of segmentation using Relative Volume Difference (RVD) calculations, which provides a quantitative assessment of segmentation results.

Materials and Methods

Data

Our project uses a publicly available dataset from kaggle [6]. This dataset contains 82 NNCT brain images in jpg format. Each image has approximately 30 slices. Masks were provided for 318 image slices containing hemorrhages. Images and slices had to be manually selected for analysis because they needed to contain a visible hemorrhage for segmentation and an associated mask. The following files were used in our analysis: 049, 050, 051, 052, 053.

Tools

The following tools and packages were used in our analysis:

Table 1. Description of Tools and Data Used

Category	Tool	Version	Description
Hardware	Personal Computers	N/A	All code was run on personal computers without any issues.
Programming Framework	Python - Jupyter Notebook	6.4.5	Jupyter Notebook is a web-based open-source software for developing Python code. This was used as the framework for our project [7].
Version Control	Github Desktop	3.2.1	Github was used to track changes in code.
Data Loading	NiBabel Library	5.0.1	NiBabel is a Python library that was used to help us read and import the NIFTI file images from our dataset [8].
Filtering	SciPy Library ● ndimage.gaussian_filter() ● ndimage.sobel() ● ndimage.laplace() Scikit-image ● denoise_nl_means() 3D Slicer	SciPy 1.7.1 Scikit-image 0.18.3 3D Slicer 5.2.2	Filtering was performed using functions from SciPy and Scikit-image libraries. 3D Slicer was used to visualize the effects of altering the window and level on an image and selecting image slices for segmentation.
Segmentation	Scikit-image ● Threshold based ● segmentation.chan_vese() ● skimage.segmentation	0.18.3	Segmentation was performed manually using a threshold and with Chan-Vese and SLIC algorithms from the Scikit library. The Chan-Vese method groups pixels with low variance, while SLIC clusters pixels

	.slic()		based on their similar intensity and proximity (both common in medical imaging) [9]
Calculations	Scikit-image ● measure.label ● measure.regionprops	0.18.3	Scikit-image functions were used to calculate the volume of both the brain and hemorrhage. The measure.label function labels the hemorrhage and brain regions, while the measure.regionprops function calculates the volume. The RVD will be calculated using the formula: RVD = (Volume of Hemorrhage Region - Volume of Brain Region) / (Volume of Brain Region).
Visualization Tools	Matplotlib	3.4.3	Matplotlib was used for 2D image visualization as it is a powerful, easy-to-use library built on NumPy arrays, designed to work with advanced Python libraries, and supports interactive plotting [10].

Processing Pipeline

The below image shows our project approach for a single image. The first step was to pre-process our data. We identified 4 pre-processing techniques to attempt on our images. We included window & level to adjust the contrast, one low-pass filter to remove noise and two high-pass filters to perform edge detection. We included two high-pass filters because we assumed we would need to perform edge detection prior to segmentation. Each technique would be inspected manually and the best technique or combination of techniques would be used in our segmentation procedure.

Our second step was segmentation. We needed to segment both the brain and hemorrhage to calculate the relative volume difference. Three segmentation methods were chosen: Threshold based, Chan-Vese and Simple Linear Iterative Clustering (SLIC). The threshold method was included because of its simplicity and use in prior notebooks. The Chan-Vese method was selected because it is commonly used for medical image segmentation and works by separating an image into two classes with the lowest variance [9]. Finally, SLIC was chosen because it uses k-means which is a commonly used clustering technique [11]. SLIC can also segment each image into a user-defined number of regions. The best segmentation approaches would be used to calculate the Relative Volume Difference (RVD).

Using our segmented brain and hemorrhage we need to calculate our RVD. RVD was chosen because hemorrhage volume is a common prognosis tool and one of the two metrics used to evaluate entries in the 2022 Instance Challenge [3], [12]. The RVD will be calculated for our segmentation procedure and compared against the RVD of the mask. The segmentation technique that produces an RVD closest to the mask will be our final procedure.

Finally, we will apply this segmentation procedure to all 5 of our images. This step will determine if our segmentation approach can be automated or not. The segmentation can be automated if the hemorrhage is correctly isolated from the brain without manually adjusting any parameters or steps.

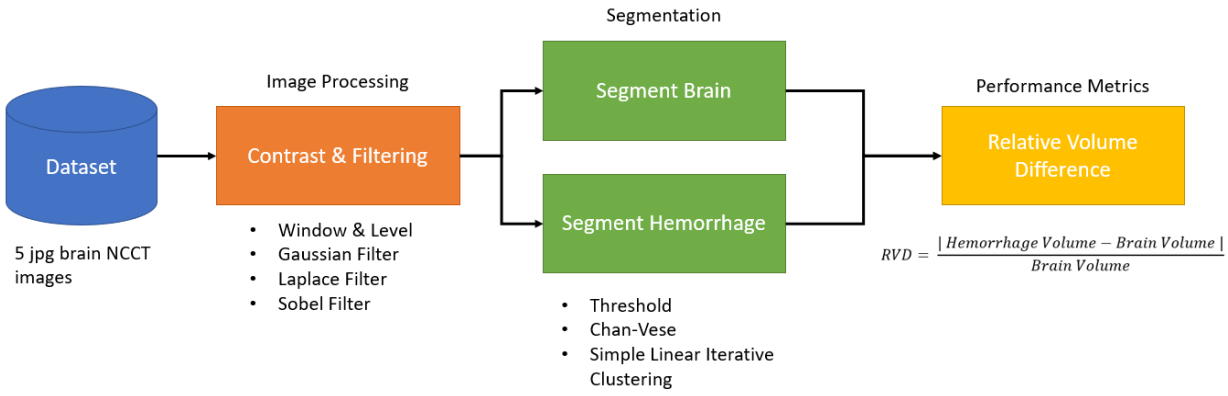


Figure 1. Processing Pipeline

Results

The results from this project can be generalized into three parts: filtering, segmentation, and RVD calculations. In this section, these parts will be analyzed and described to highlight the results gathered from this project.

Filtering

Various filtering techniques were applied to determine which was the most effective for viewing the ICH. The different techniques we explored included window/level, median, sobel, high pass filter, etc. Based on observation, the filter that provided the best result was window/level. Different values were experimented with and we decided on 130 for the window and 65 for the level. This allowed for much better contrast between the brain and ICH, as seen in Figure 2 below. On the left is the original image taken from the dataset, whereas on the right, is the same image with window and leveling applied. A limitation we experienced during this step included leaving the image slicing as manual. This limitation is explained more in the discussion section below.

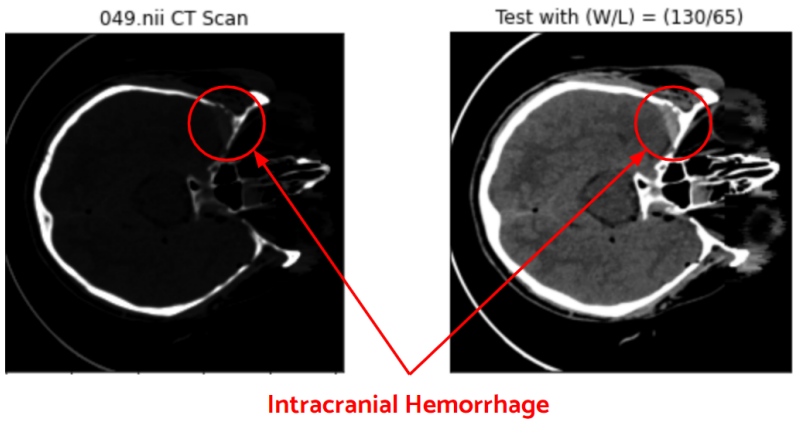


Figure 2. Image from Dataset Before and After Applying Window and Level

Segmentation

To decide which segmentation method was the most effective we experimented with three different kinds: threshold-based, Chan-Vese, and SLIC. Based on observation, SLIC was the best method at segmenting the ICH. Threshold-based segmentation was unable to distinguish between the ICH and the perimeter of the skull. Whereas, Chan-Vese could only segment into two sections meaning it was unable to capture any detail of the ICH. In Figure 3 below, we have included 6 images showing the original image, filtering image, ideal mask (provided by the dataset), and the results from all three segmentation models. It is clear from examining this result that SLIC was the best segmentation model. More results gathered from working with different images can be found in Appendix A. A limitation we experienced during this step included leaving the ICH region for segmentation in the SLIC model as manual. This limitation is explained more in the discussion section below.

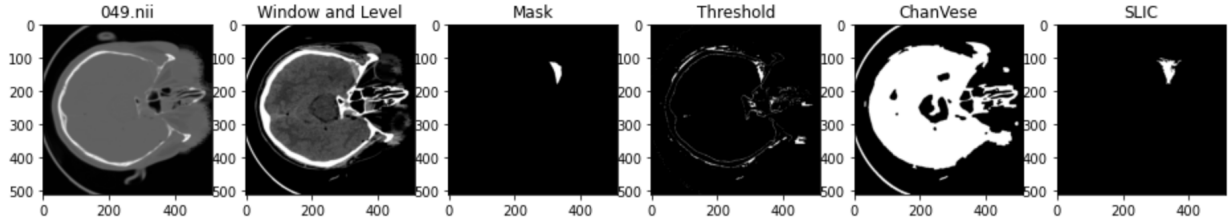


Figure 3. Results from Applying Three Segmentation Methods

RVD Calculations

For this part of the results, RVD was calculated for the ideal mask, threshold-based and SLIC model. We excluded the Chan-Vese model from this calculation because it was obviously not effective in segmenting the ICH. By calculating these values we could validate that the SLIC model can more accurately segment the ICH. As seen in Table X below, RVD calculated for the SLIC model was very similar to the RVD calculated for the ideal mask. As for the threshold-based model, the RVD value was always roughly the same because the same threshold range was used for each image. In other words, this analysis helped confirm that the SLIC method was the most effective in segmenting the ICH.

Table 2. Calculated RVD Values

File Number	Ideal Mask	Threshold-Based	Threshold % Error	SLIC	SLIC % Error
049.nii	0.995	0.999…	0.40	0.991	0.40
050.nii	0.997	0.999…	0.20	0.994	0.30
051.nii	0.988	0.999…	1.11	0.989	0.10
052.nii	0.988	0.999…	1.11	0.983	0.51
053.nii	0.974	0.999…	2.57	0.968	0.62

Discussion

Our project was aimed at developing an automated segmentation procedure for ICH in NCCT images. This segmentation procedure should be able to accurately isolate a hemorrhage for use in RVD calculations and prognosis, and be automated to reduce time and resources spent on image processing. The results of our segmentation procedure partially meets these aims. Our chosen segmentation procedure accurately isolates the hemorrhage which is evident by very similar RVD values to the mask. However, when we applied our procedure to multiple images in our dataset we were not able to isolate the hemorrhage automatically. This is because the hemorrhage is not always stored in the same region from the SLIC segmentation. Therefore this procedure is not suitable for automation. However, this procedure is still beneficial by providing a framework that requires minimal adjustments to achieve an accurate segmentation.

The RVD results for the 5 images are shown in the above table. The threshold based method resulted in the same value (0.999) for each image. This is because the threshold method had a very small range of values resulting in few segmented pixels. The fewer pixels included in the segmentation resulted in negligible hemorrhage volume and RVD values close to one. In comparison, the SLIC segmentation was based on k-means clustering and would pick up hemorrhage pixels that were not within the chosen threshold range. This resulted in a greater hemorrhage volume and RVD values further from one. Comparing the RVD values from the mask to the threshold and SLIC methods, it is obvious the SLIC method is better. The threshold method is only better for file 050 when the true RVD is close to one. However, the SLIC segmentation was able to adapt to each image. The results of our SLIC segmentation were very high quality with an average percent error of 0.39%. The SLIC segmentation also performed better visually as seen in Figure 2 and more images in Appendix A. The threshold segmentation would always include part of the skull, while the SLIC method could isolate only the hemorrhage.

The two main limitations we experienced when completing this project were time and automation. Time was a limitation because all group members are balancing the rest of their workload while completing this project. With more time, we may have been able to expand the scope of this project and add further functionality. With that being said, the requirements were still satisfied and the results gathered were

considered a success. As for automation, it was difficult to write code that was efficient and produced optimal results. When working on this project there were two steps that we tried to automate but decided they were better left manual. These two steps were slicing the image and choosing the segmentation region for the SLIC model. When these steps were automatic, the program was unable to use the optimal slice that displayed the ICH best and also was unable to segment it well because the program did not know where the ICH region was. By converting these two steps back to manual, the efficiency of the program was lowered but the results displayed were more accurate.

Below are recommendations we would pass along to others looking to perform a similar project. These suggestions are not ways to advance our project but more-so how to increase the likelihood of success in similar projects:

1. Focus on a single image to start and then scale up. By doing this, we were able to create a strong foundation and prove that our code worked before looking at multiple images.
2. Experiment with multiple segmentation methods. By trying more than one model, you learn more about how certain methods work as well as which ones are more effective.
3. Try to find a dataset that has an ideal mask. In the dataset we used, we were provided with the ideal mask from CT scans which helped us quickly understand what the ideal segmentation result would look like and saved us the hassle of trying to create our own ideal result.

Future work for enhancing this project could include:

1. Using Dice Similarity Coefficient (DSC) could be an additional step to validate the performance of the segmentation methods. This is a performance metric used to determine the difference between the ground-truth and its corresponding prediction.
2. Another way to advance this project would include developing more automated code through the use of machine learning. By incorporating machine learning, parts of this project that remained manual could be automated to increase the efficiency of the segmentation.
3. Lastly, different segmentation models could be investigated to compare more results and better understand how various methods affect the segmentation results.

Conclusion

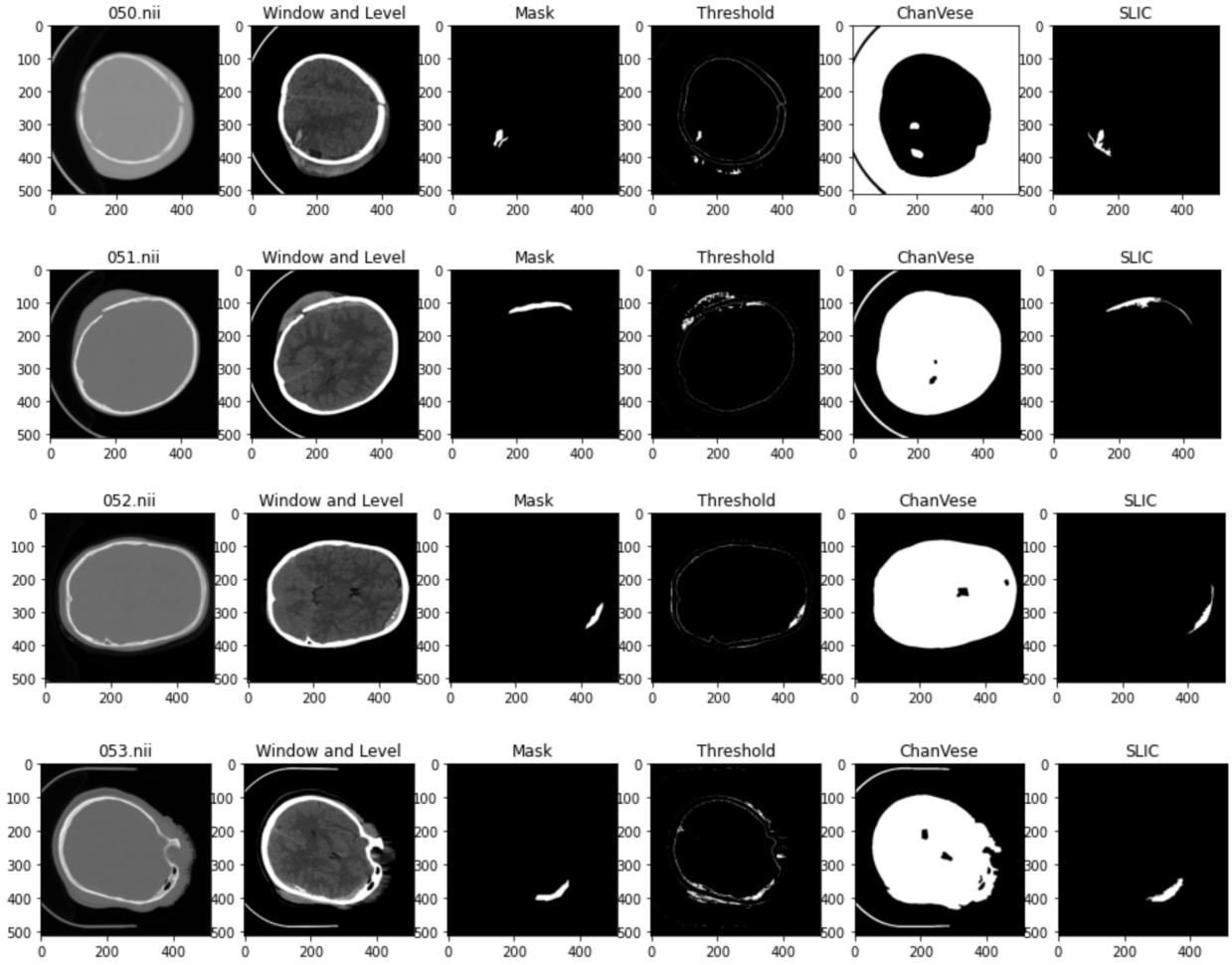
Intracranial hemorrhages (ICH) is a life-threatening condition that requires early diagnosis and treatment for improved patient outcomes. Manual segmentation of ICH from Non-Contrast Computed Tomography (NCCT) images can be time-consuming and subjective, leading to inconsistencies and errors. Therefore, the development of automated methods for ICH segmentation is crucial. In this project, we aimed to develop a tool using advanced algorithms to accurately segment ICH from NCCT images, with the goal of improving the accuracy and efficiency of ICH segmentation.

We utilized a publicly available dataset from Kaggle, and our processing pipeline involved pre-processing techniques, including window/level adjustment for contrast, low-pass and high-pass filtering for noise removal and edge detection, followed by segmentation using three methods: threshold-based, Chan-Vese, and Simple Linear Iterative Clustering (SLIC). The Relative Volume Difference (RVD) was used for evaluation, as it is commonly used for prognosis and was one of the evaluation metrics in the 2022 Instance Challenge.

Our results showed that window/level adjustment provided the best contrast for viewing ICH, with values of 130 for window and 65 for level. Among the segmentation methods, SLIC was the most effective in accurately segmenting ICH from NCCT images, as it could capture the details of the hemorrhage without including the perimeter of the skull. The RVD calculated for our segmentation procedure was compared against the RVD of the mask, and the segmentation technique that produced the RVD closest to the mask was identified as the final procedure.
Our tool has significant clinical implications, as it contributes to the development of more accurate and efficient methods for ICH segmentation, which can aid in early diagnosis and treatment planning for patients with intracranial hemorrhages. However, there are limitations in our project, including manual image slicing and a limited dataset. Future work could involve automated image slicing and further validation on larger and more diverse datasets. In conclusion, our project developed a tool for accurate ICH segmentation from NCCT images using advanced algorithms, with SLIC identified as the most effective segmentation method. The results of this project contribute to the field of medical image analysis and have the potential to improve patient outcomes in the diagnosis and treatment of intracranial hemorrhages. Further research and validation are warranted to refine and optimize the tool for clinical use.

Appendix

Appendix A: Results from Applying Three Segmentation Methods to Different Images



Appendix B: Jupyter Notebook Code

Please see the following pages

Final Results

April 13, 2023

1 Bmen509 Final Results

1.1 Group 10

[1]:
```
import numpy as np
# Set numpy to print only 2 decimal digits for neatness
np.set_printoptions(precision=2, suppress=True)
import os
import nibabel as nib
from nibabel.testing import data_path
import matplotlib.pyplot as plt
import pandas as pd

from skimage import data, img_as_float
from skimage.segmentation import chan_vese, slic, mark_boundaries
from skimage.color import label2rgb
from os.path import exists
from skimage import measure
```

[2]:
```
# Function definitions

def window_level_function(image, window, level):

    i = image.astype(np.double)

    max_val = 255
    min_val = 0

    upper_lim = level + (0.5*window)
    lower_lim = level - (0.5*window)

    m = (max_val-min_val)/window
    b = max_val - (m*upper_lim)

    j = np.empty(shape=(512, 512))

    for x in range(511):
        for y in range(511):
```

```
            if i[x,y] < lower_lim:
                j[x,y] = 0
            elif i[x,y] > upper_lim:
                j[x,y] = 255
            else:
                j[x,y] = m*(i[x,y]) + b

    image = j

    return image.astype(np.uint8)


def segment_threshold(img, lower_lim, upper_lim):

    out_img = np.zeros_like(img,'uint8')

    for x in range(511):
        for y in range(511):
            if img[x,y] < lower_lim or img[x,y] > upper_lim:
                out_img[x,y] = 0
            else:
                out_img[x,y] = 1

    return out_img

def segment_slic(img, num, region):

    boundaries = slic(img, n_segments=num) #max_num_iter=100)

    out_img = np.zeros_like(boundaries,'uint8')
    out_img[boundaries == region] = 1
    out_img[boundaries != region] = 0

    return out_img

def RVD_calc(brain, hemo):

    labels1 = measure.label(brain, background=0)  # Identify adjacent pixels
    labels2 = measure.label(hemo, background=0)

    brain_reg = measure.regionprops(labels1)  # Convert lables into regions
    hemo_reg = measure.regionprops(labels2)

    if len(hemo_reg)==0:
        return 0

    # Calculate volumes of hemorrhage and brain regions
```

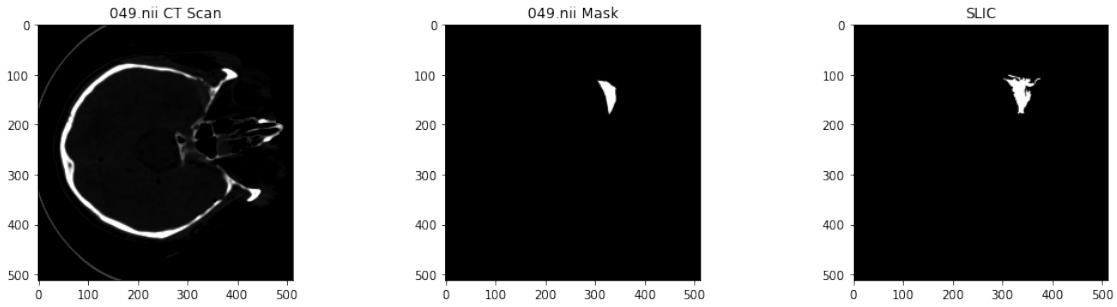
```
    # Choose the index for the largest region (background is excluded)
    brain_vol = brain_reg[1].area
    hem_vol = hemo_reg[0].area

    # Calculate RVD
    rvd = abs(hem_vol - brain_vol) / brain_vol

    return rvd
```

[3]:
```
slice_array = []
num_segments = []
region = []
```

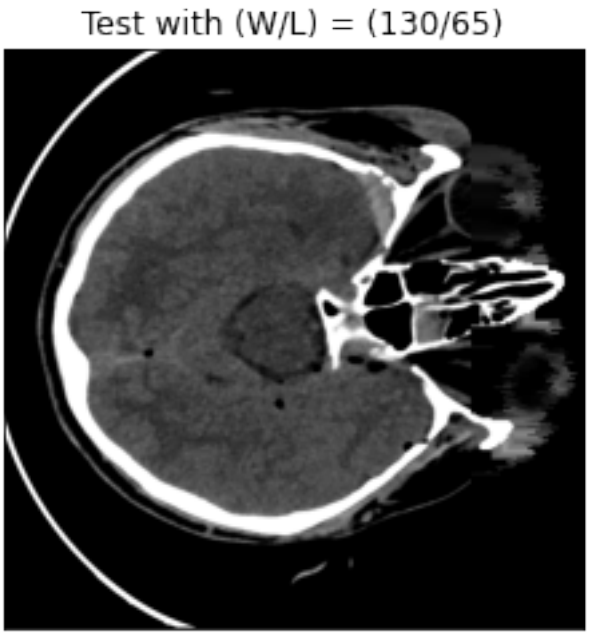
[4]:
```
data_path = path = '.\Data\ct_scans'
file = '049.nii'
example_ni1 = os.path.join(data_path, file)
n1_img = nib.load(example_ni1).get_fdata()

data_path = path = './Data/masks'
example_nimask1 = os.path.join(data_path, file)
n1_mask_img = nib.load(example_nimask1).get_fdata()

slice_val = 14
slice_array.append(slice_val)

num_seg = 133
reg = 17
slic_img = segment_slic(n1_img[:,:, slice_val], num_seg, reg)

num_segments.append(num_seg)
region.append(reg)

plt.figure(figsize=(16, 16))

plt.subplot(131)
n1_img = n1_img[:,:, slice_val]
plt.imshow(n1_img, cmap='gray', vmin=0,vmax=1000)
plt.title(str(file) + ' CT Scan')

plt.subplot(132)
n1_mask_img = n1_mask_img[:,:, slice_val]
plt.imshow(n1_mask_img, cmap='gray')
plt.title(str(file) + ' Mask')
plt.subplots_adjust(wspace=0.6, hspace=0.6)

plt.subplot(133)
plt.imshow(slic_img, cmap='gray')
```

```
plt.title('SLIC')

plt.show()
```

```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```



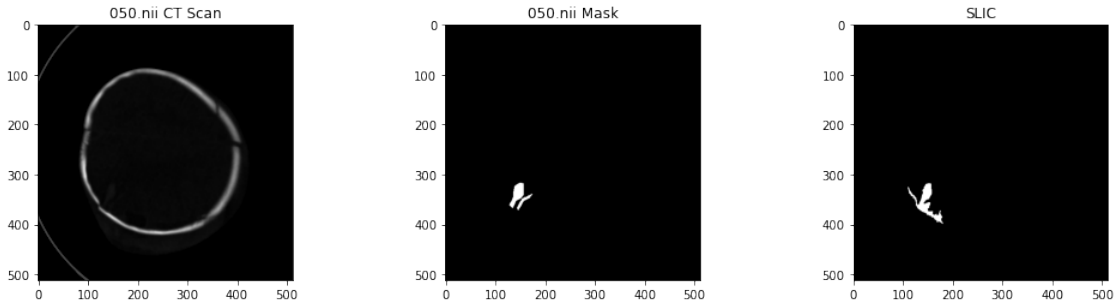
[5]:
```
pairs = {
    'Test': (130, 65)
}

#i = 1
#plt.subplots(len(pairs), 2, figsize=(20,20))
for key in pairs:
    window, level = pairs[key]

    new_image = window_level_function(n1_img, window, level)

    #plt.subplot(len(pairs), 2, 2*i-1)
    plt.imshow(new_image, cmap='gray')
    plt.title('{} with (W/L) = ({}/{})'.format(key, window, level))
    plt.xticks([])
    plt.yticks([])

    #plt.subplot(len(pairs), 2, 2*i)
    #plt.hist(new_image.ravel(), bins=256)

    #i += 1
plt.show()
```



[6]:
```
data_path = path = '.\Data\ct_scans'
file = '050.nii'
example_ni1 = os.path.join(data_path, file)
n1_img = nib.load(example_ni1).get_fdata()

data_path = path = './Data/masks'
example_nimask1 = os.path.join(data_path, file)
n1_mask_img = nib.load(example_nimask1).get_fdata()

slice_val = 20
slice_array.append(slice_val)

num_seg = 140
reg = 57
slic_img = segment_slic(n1_img[:,:, slice_val], num_seg, reg)

num_segments.append(num_seg)
region.append(reg)

plt.figure(figsize=(16, 16))

plt.subplot(131)
n1_img = n1_img[:,:, slice_val]
plt.imshow(n1_img, cmap='gray', vmin=0,vmax=1000)
plt.title(str(file) + ' CT Scan')
```

```
plt.subplot(132)
n1_mask_img = n1_mask_img[:,:, slice_val]
plt.imshow(n1_mask_img, cmap='gray')
plt.title(str(file) + ' Mask')
plt.subplots_adjust(wspace=0.6, hspace=0.6)

plt.subplot(133)
plt.imshow(slic_img, cmap='gray')
plt.title('SLIC')

plt.show()
```

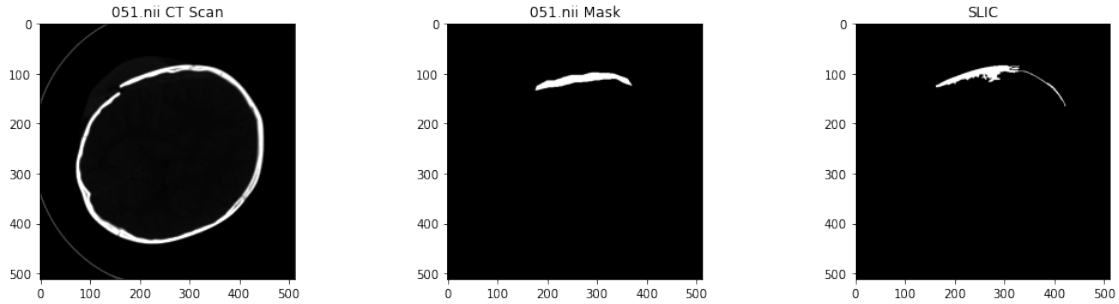
```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```



[7]:
```
data_path = path = '.\Data\ct_scans'
file = '051.nii'
example_ni1 = os.path.join(data_path, file)
n1_img = nib.load(example_ni1).get_fdata()

data_path = path = './Data/masks'
example_nimask1 = os.path.join(data_path, file)
n1_mask_img = nib.load(example_nimask1).get_fdata()

slice_val = 29
slice_array.append(slice_val)

num_seg = 135
reg = 18
slic_img = segment_slic(n1_img[:,:, slice_val], num_seg, reg)

num_segments.append(num_seg)
```

```
region.append(reg)

plt.figure(figsize=(16, 16))

plt.subplot(131)
n1_img = n1_img[:,:, slice_val]
plt.imshow(n1_img, cmap='gray', vmin=0,vmax=1000)
plt.title(str(file) + ' CT Scan')

plt.subplot(132)
n1_mask_img = n1_mask_img[:,:, slice_val]
plt.imshow(n1_mask_img, cmap='gray')
plt.title(str(file) + ' Mask')
plt.subplots_adjust(wspace=0.6, hspace=0.6)

plt.subplot(133)
plt.imshow(slic_img, cmap='gray')
plt.title('SLIC')

plt.show()
```

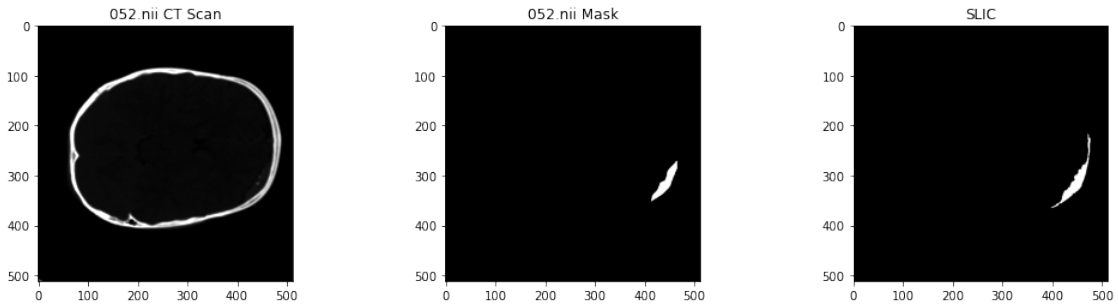
```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```



[8]:
```
data_path = path = '.\Data\ct_scans'
file = '052.nii'
example_ni1 = os.path.join(data_path, file)
n1_img = nib.load(example_ni1).get_fdata()

data_path = path = './Data/masks'
example_nimask1 = os.path.join(data_path, file)
n1_mask_img = nib.load(example_nimask1).get_fdata()
```

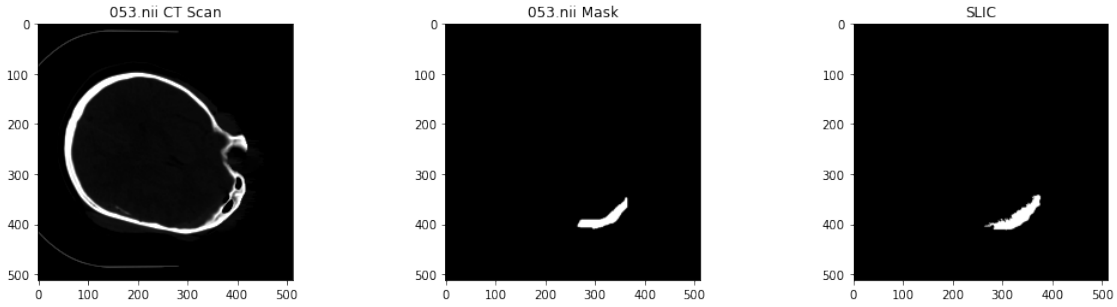
```
slice_val = 15
slice_array.append(slice_val)

num_seg = 120
reg = 31
slic_img = segment_slic(n1_img[:,:, slice_val], num_seg, reg)

num_segments.append(num_seg)
region.append(reg)

plt.figure(figsize=(16, 16))

plt.subplot(131)
n1_img = n1_img[:,:, slice_val]
plt.imshow(n1_img, cmap='gray', vmin=0,vmax=1000)
plt.title(str(file) + ' CT Scan')

plt.subplot(132)
n1_mask_img = n1_mask_img[:,:, slice_val]
plt.imshow(n1_mask_img, cmap='gray')
plt.title(str(file) + ' Mask')
plt.subplots_adjust(wspace=0.6, hspace=0.6)

plt.subplot(133)
plt.imshow(slic_img, cmap='gray')
plt.title('SLIC')

plt.show()
```

```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```



[9]:
```
data_path = path = '.\Data\ct_scans'
file = '053.nii'
example_ni1 = os.path.join(data_path, file)
n1_img = nib.load(example_ni1).get_fdata()

data_path = path = './Data/masks'
example_nimask1 = os.path.join(data_path, file)
n1_mask_img = nib.load(example_nimask1).get_fdata()

slice_val = 22
slice_array.append(slice_val)

num_seg = 85
reg = 29
slic_img = segment_slic(n1_img[:,:, slice_val], num_seg, reg)

num_segments.append(num_seg)
region.append(reg)

plt.figure(figsize=(16, 16))

plt.subplot(131)
n1_img = n1_img[:,:, slice_val]
plt.imshow(n1_img, cmap='gray', vmin=0,vmax=1000)
plt.title(str(file) + ' CT Scan')

plt.subplot(132)
n1_mask_img = n1_mask_img[:,:, slice_val]
plt.imshow(n1_mask_img, cmap='gray')
plt.title(str(file) + ' Mask')
plt.subplots_adjust(wspace=0.6, hspace=0.6)

plt.subplot(133)
plt.imshow(slic_img, cmap='gray')
plt.title('SLIC')

plt.show()
```

```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```



[10]:
```
print(slice_array)
print(num_segments)
print(region)
```

```
[14, 20, 29, 15, 22]
[133, 140, 135, 120, 85]
[17, 57, 18, 31, 29]
```

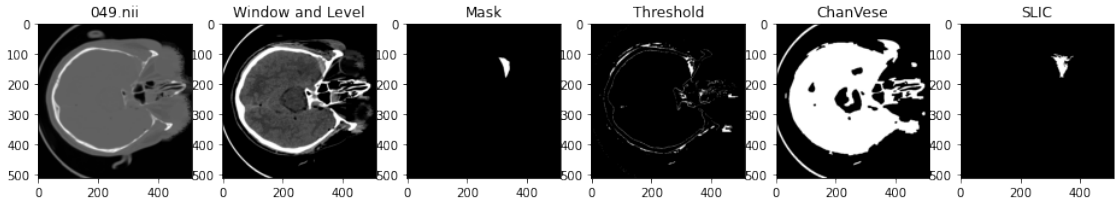
[11]:
```
# Loop over all files

path = os.getcwd() + '/Data/ct_scans/'

#files = os.listdir(path)
files = ['049.nii', '050.nii', '051.nii', '052.nii', '053.nii']

RVD = pd.DataFrame(columns=['File','Ideal','Threshold','SLIC'])

i = 0

for file in files:

    # Find image mask. If it doesn't exist continue to next iteration.
#     if not(exists(os.getcwd() + '/Data/masks/' + file)):
#         continue

    # Load file
    img = nib.load(path + file).get_fdata()
    img_original = img
    img = img[:,:, slice_array[i]]

    # Load mask
    mask_img = nib.load(os.getcwd() + '/Data/masks/' + file).get_fdata()
    mask_img = mask_img[:,:, slice_array[i]]

    # Window and Level
    new_img = window_level_function(img, 130, 65)
```

```
    # Threshold Based
    brain = segment_threshold(new_img, 0, 240)
    threshold_img = segment_threshold(new_img, 120, 160)

    # Chan-Vese
    cv_img = chan_vese(new_img, mu=0.2, lambda1=1, lambda2=1, tol=1e-3, dt=0.2,␣
↪extended_output=True) #, max_num_iter=200,)

    # SLIC Based
    # Use original image for SLIC segmentation
    # Not every hemorrage will be region 24!
    slic_img = segment_slic(img, num_segments[i], region[i])

    temp = pd.DataFrame({'File': [file],
                         'Ideal': [RVD_calc(brain, mask_img)],
                         'Threshold': [RVD_calc(brain, threshold_img)],
                         'ChanVese': [RVD_calc(brain, cv_img[0])],
                         'SLIC': [RVD_calc(brain, slic_img)],
                         'Average': [(RVD_calc(brain, threshold_img) +␣
↪RVD_calc(brain, cv_img[0]) + RVD_calc(brain, slic_img))/3]})

    print(temp)
    RVD = RVD.append(temp)

    i = i + 1

    plt.figure(figsize=(16, 16))
    plt.subplot(161)
    plt.imshow(img, cmap='gray')
    plt.title(str(file))

    plt.subplot(162)
    plt.imshow(new_img, cmap='gray')
    plt.title('Window and Level')

    plt.subplot(163)
    plt.imshow(mask_img, cmap='gray')
    plt.title('Mask')

    plt.subplot(164)
    plt.imshow(threshold_img, cmap='gray')
    plt.title('Threshold')

    plt.subplot(165)
    plt.imshow(cv_img[0], cmap='gray')
    plt.title('ChanVese')
```

```
    plt.subplot(166)
    plt.imshow(slic_img, cmap='gray')
    plt.title('SLIC')

    plt.show()

    del img, mask_img, new_img, brain, threshold_img, cv_img, slic_img, temp

    averages = RVD.mean(axis=0)
print('\nAverage RVD values:')
print(averages)
```

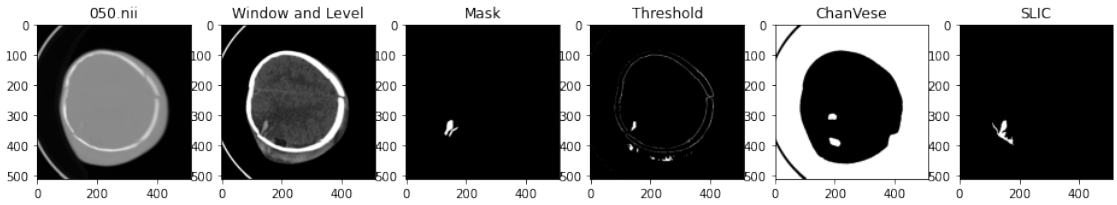
```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```

	File	Ideal	Threshold	ChanVese	SLIC	Average
0	049.nii	0.994882	0.999996	0.993385	0.991362	0.994914



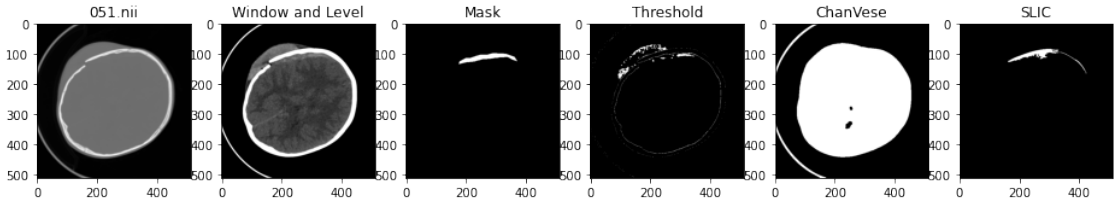

```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2200365756.py:83:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  averages = RVD.mean(axis=0)
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```

	File	Ideal	Threshold	ChanVese	SLIC	Average
0	050.nii	0.996813	0.999996	0.985031	0.99377	0.992932



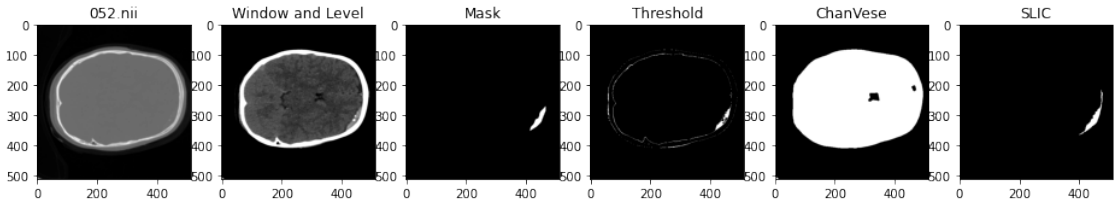

```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2200365756.py:83:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  averages = RVD.mean(axis=0)
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```

	File	Ideal	Threshold	ChanVese	SLIC	Average
0	051.nii	0.988396	0.999996	0.993257	0.988796	0.994016



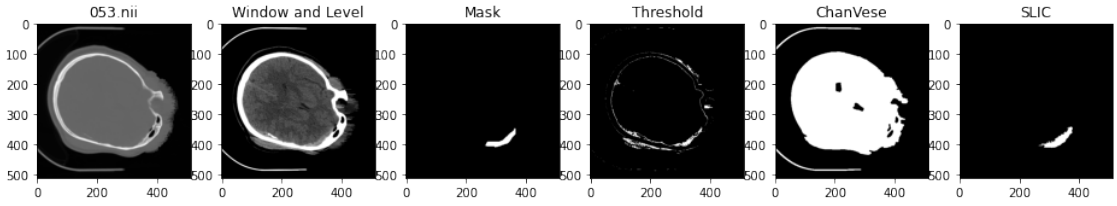

```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2200365756.py:83:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  averages = RVD.mean(axis=0)
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```

	File	Ideal	Threshold	ChanVese	SLIC	Average
0	052.nii	0.987572	0.999939	0.183181	0.983371	0.722164




```
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2200365756.py:83:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  averages = RVD.mean(axis=0)
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2990673429.py:47:
FutureWarning: skimage.measure.label's indexing starts from 0. In future version
it will start from 1. To disable this warning, explicitely set the `start_label`
parameter to 1.
  boundaries = slic(img, n_segments=num) #max_num_iter=100)
```

	File	Ideal	Threshold	ChanVese	SLIC	Average
0	053.nii	0.974147	0.999972	0.978465	0.967933	0.982123




```
Average RVD values:
Ideal        0.988362
Threshold    0.999980
SLIC         0.985046
ChanVese     0.826664
Average      0.937230
dtype: float64
C:\Users\jnrri\AppData\Local\Temp/ipykernel_10532/2200365756.py:83:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  averages = RVD.mean(axis=0)
```

[12]:
```
print(RVD)
```

	File	Ideal	Threshold	SLIC	ChanVese	Average
0	049.nii	0.994882	0.999996	0.991362	0.993385	0.994914
0	050.nii	0.996813	0.999996	0.993770	0.985031	0.992932
0	051.nii	0.988396	0.999996	0.988796	0.993257	0.994016
0	052.nii	0.987572	0.999939	0.983371	0.183181	0.722164
0	053.nii	0.974147	0.999972	0.967933	0.978465	0.982123

[ ]:

References

[1] "Brain Bleed/Hemorrhage (Intracranial Hemorrhage): Causes, Symptoms, Treatment," *Cleveland Clinic*. https://my.clevelandclinic.org/health/diseases/14480-brain-bleed-hemorrhage-intracranial-hemorrhage (accessed Apr. 11, 2023).

[2] "Intracranial Hemorrhage: Background, Pathophysiology, Epidemiology," Feb. 2022, Accessed: Apr. 11, 2023. [Online]. Available: https://emedicine.medscape.com/article/1163977-overview

[3] "INSTANCE2022 - Grand Challenge," *grand-challenge.org*. https://instance.grand-challenge.org/Dataset/ (accessed Mar. 02, 2023).

[4] A. Patel *et al.*, "Intracerebral Haemorrhage Segmentation in Non-Contrast CT," *Sci. Rep.*, vol. 9, no. 1, p. 17858, Nov. 2019, doi: 10.1038/s41598-019-54491-6.

[5] "An implementation of Otsu thresholding and the Chan–Vese method on the PCO segmentation of ultrasound images | IEEE Conference Publication | IEEE Xplore." https://ieeexplore-ieee-org.ezproxy.lib.ucalgary.ca/abstract/document/9309595 (accessed Apr. 11, 2023).

[6] "Brain CT Images with Intracranial Hemorrhage Masks." https://www.kaggle.com/datasets/vbookshelf/computed-tomography-ct-images (accessed Mar. 02, 2023).

[7] "Project Jupyter." https://jupyter.org (accessed Mar. 02, 2023).

[8] "Neuroimaging in Python — NiBabel 5.0.0 documentation." https://nipy.org/nibabel/ (accessed Mar. 02, 2023).

[9] "Image Segmentation using Python's scikit-image module," *GeeksforGeeks*, Aug. 23, 2021. https://www.geeksforgeeks.org/image-segmentation-using-pythons-scikit-image-module/ (accessed Mar. 02, 2023).

[10] "Matplotlib — Visualization with Python." https://matplotlib.org/ (accessed Mar. 02, 2023).

[11] E. Ecosystem (LEDU), "Understanding K-means Clustering in Machine Learning," *Medium*, Sep. 12, 2018. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 (accessed Apr. 07, 2023).

[12] J. P. Broderick, T. G. Brott, J. E. Duldner, T. Tomsick, and G. Huster, "Volume of intracerebral hemorrhage. A powerful and easy-to-use predictor of 30-day mortality.," *Stroke*, vol. 24, no. 7, pp. 987–993, Jul. 1993, doi: 10.1161/01.STR.24.7.987.