IPBME6UE INF_BA_VZ: Exercise #1- Segmentation

Spring 2022

Group Work

For this exercise, we encourage teamwork! You can work either by yourself or in pairs (2 people working together equally and submitting one notebook for the both of you).

Who said medical imaging had to be lonely?

Person 1:

YOUR NAME HERE

Person 2:

YOUR NAME HERE

iPython Notebooks: A lesson in reproducibility

For those of you that are unfamiliar, iPython notebooks are interactive Python sessions that allow you to intersperse code, raw text, and markdown in a seamless manner. The next paragraph will teach you to use them.

The words you are reading right now are modifiable. Double-click me to change the text that you're reading. These modifiable boxes are called cells. They will be used to write answers to our written questions.

Double-click the box above me now, and go ahead and fill in your name (and your partner's as well, if applicable).

Cells can contain executable code as well. If you're running the Jupyter Notebook program, You can change the type of cell by highlighting the cell of interest and going to `Cell -> Cell Type -> Markdown/Code`, and then clicking the desired cell type. If you're running the newer JupyterLab program, just click the drop-down menu next to the play, stop, and refresh icons above this window (along the top of the tabs detailing the current files that are open) and change the cell type directly there.

You can add cells using Insert in the Jupyter Notebook program and the `'+'` icon in JupyterLab.

The best thing about notebooks is that you can run small components of your code in one cell to make sure they work before putting them together to make a larger component. Make as many cells as you want to play and experiment, but please delete them if they are not part of your final submission. To delete a cell, highlight the cell and going to `Edit -> Delete Cells`, or use the keyboard shortcut of pressing the 'D' button twice. Make sure to not do this by mistake, but if you do, you can `"Redo Cell Operation"` from the respective menus in the different programs.

Another nice thing about notebooks - the entire homework assignment is self-contained in this file! Please put all your functions and classes into the cells of this notebook, and please write clean code with at least some annotation to help us follow your thought process. Once you're done, export the notebook to a `.pdf` file and submit (on Teams?).

In research, it is important that code is readable and reproducible. Notebooks are a natural first step toward both goals.

Setup

Run the code in the following cells by single clicking on them and then press CTRL+ENTER (SHIFT+ENTER on Mac) or click the play button in the menu bar (the one to the left of the stop button and to the right of the up and down arrows).

The following first cell loads all the Python dependencies for this homework. It's OK if you get an error at first.

In [ ]:
```
import numpy as np
import pydicom as dicom
import matplotlib.pyplot as plt
import os
import skimage

from pathlib import Path

%matplotlib inline
```

Did you get an error like this?

```
### ModuleNotFoundError: No module named 'skimage'/'numpy'/'pydicom'?
```

It means you need to install some more dependencies. For example, PyDicom is not usually included.

If you are using Anaconda - Here is how to install PyDicom. If you don't have an Anaconda Python distribution, try with `pip`. You may have to use `sudo` like this:

```
$ sudo pip3 install pydicom
```

But try first without `sudo`.

Here is some PyDicom documentation which might be useful later on in the assignment.

The other dependencies like scikit-image are installed similarly.

You may need to do the same for `skimage`:

```
$ sudo pip3 install scikit-image
```

Note: You can also install packages from within the Jupyter kernal. Try running the code in the cell below, which attempts to install NumPy:

In [ ]:
```
# Install a pip package in the current Jupyter kernel
import sys
!{sys.executable} -m pip install numpy
```

```
Requirement already satisfied: numpy in /home/liad/.virtualenvs/krems/lib/py
thon3.8/site-packages (1.22.2)
```

The Data

The data consist of sets of DICOM images that hold completely anonymized chest CT scans (see section “Reading in the Data”). DICOM (Digital Imaging and Communications in Medicine) is a standardized format for transmitting medical image data. Each DICOM file is composed of two parts: the image data as well as a header giving you a lot of metadata about the patient and specifics about the imaging parameters (e.g. space between slices).

You can choose any of the patients in the dataset to prototype your code, but you are expected to generalize your algorithm to work for at least ten different patients to show robustness. The more you do, the more you’ll prove your algorithm.

First, let's make sure you can read in the data.

In the cell below, replace the `scans_path` string variable with the (relative or absolute) path to the place where you put the data. Particularly, look for the folder that contains scans. Each scan is in its own folder and has a name, e.g. 0d7a1bc2-352b-41f3-882c-1c8ad9f89847. `scans_path` should be the folder that contains these oddly named folders. Once you do this, you can run the cell below to look at all the metadata associated with that slice.

It should look like this:

```
In [6]: print ds

(0008, 0005) Specific Character Set             CS: 'ISO_IR 100'
(0008, 0016) SOP Class UID                      UI: CT Image Storage
(0008, 0018) SOP Instance UID                   UI: 1.2.840.113654.2.55.884774032949208935931647381951158593 72
(0008, 0060) Modality                           CS: 'CT'
(0008, 103e) Series Description                 LO: 'Axial'
(0010, 0010) Patient's Name                     PN: '0a0c32c9e08cc2ea76a71649de56be6d'
(0010, 0020) Patient ID                         LO: '0a0c32c9e08cc2ea76a71649de56be6d'
(0010, 0030) Patient's Birth Date               DA: '19000101'
(0018, 0060) KVP                                DS: ''
(0020, 000d) Study Instance UID                 UI: 2.25.60037070027156423276159501017920151735078954137544798194660
(0020, 000e) Series Instance UID                UI: 2.25.58703274222857573910779974742342423982066946347485459782406
(0020, 0011) Series Number                      IS: '1'
(0020, 0012) Acquisition Number                 IS: '1'
(0020, 0013) Instance Number                    IS: '67'
(0020, 0020) Patient Orientation                CS: ''
(0020, 0032) Image Position (Patient)           DS: ['-160.100006', '-142.500000', '-148.410004']
(0020, 0037) Image Orientation (Patient)        DS: ['1', '0', '0', '0', '1', '0']
(0020, 0052) Frame of Reference UID             UI: 2.25.108164247919496335220585254710613811374002699175185374855 73
(0020, 1040) Position Reference Indicator       LO: 'SN'
(0020, 1041) Slice Location                     DS: '-148.410004'
(0028, 0002) Samples per Pixel                  US: 1
(0028, 0004) Photometric Interpretation         CS: 'MONOCHROME2'
(0028, 0006) Planar Configuration               US: 0
(0028, 0010) Rows                               US: 512
(0028, 0011) Columns                            US: 512
(0028, 0030) Pixel Spacing                      DS: ['0.664062', '0.664062']
(0028, 0100) Bits Allocated                     US: 16
(0028, 0101) Bits Stored                        US: 12
(0028, 0102) High Bit                           US: 11
(0028, 0103) Pixel Representation               US: 0
(0028, 0120) Pixel Padding Value                US: 0
(0028, 1050) Window Center                      DS: '40'
(0028, 1051) Window Width                       DS: '400'
(0028, 1052) Rescale Intercept                  DS: '-1024'
(0028, 1053) Rescale Slope                      DS: '1'
(0028, 1054) Rescale Type                       LO: 'HU'
(7fe0, 0010) Pixel Data                         OW: Array of 524288 bytes
```

In [ ]:
```
# Each chest CT scan is a folder with a name like R_num that within contains
# Assign the path to such folder to the variable scans_path

# TODO: Change this variable to match the folder where your data is located!
scans_path = "FULL_PATH_TO/YOUR_FOLDER_LOCATION/LungCT-Diagnostic Pre-Surger
list_of_scans = os.listdir(scans_path)

# for figuring out the controls lets experiment with slice 122 of slice 2
scan_num = 2
scan_path = os.path.join(scans_path, list_of_scans[scan_num])
list_of_slices = os.listdir(scan_path)
slice_num = 42
slice_path = os.path.join(scan_path, list_of_slices[slice_num])

# read in the full path to the file as ds
ds=dicom.read_file(slice_path) # you may have to use pydicom instead of dico
print(ds)
```

```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 194
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: CT Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.3.6.1.4.1.14519.5.2.
1.4320.5030.332046026331924823710503218304
(0002, 0010) Transfer Syntax UID                 UI: Implicit VR Little Endi
an
(0002, 0012) Implementation Class UID            UI: 1.2.40.0.13.1.1.1
(0002, 0013) Implementation Version Name         SH: 'dcm4che-1.4.34'
-------------------------------------------------
(0008, 0008) Image Type                          CS: ['ORIGINAL', 'PRIMARY',
'AXIAL', 'CT_SOM5 SPI']
(0008, 0016) SOP Class UID                       UI: CT Image Storage
(0008, 0018) SOP Instance UID                    UI: 1.3.6.1.4.1.14519.5.2.
1.4320.5030.332046026331924823710503218304
(0008, 0020) Study Date                          DA: '19980803'
(0008, 0021) Series Date                         DA: '19980803'
(0008, 0022) Acquisition Date                    DA: '19980803'
(0008, 0023) Content Date                        DA: '19980803'
(0008, 0030) Study Time                          TM: '085302.84000'
(0008, 0031) Series Time                         TM: '085438.81000'
(0008, 0032) Acquisition Time                    TM: '085619.020672'
(0008, 0033) Content Time                        TM: '085619.020672'
(0008, 0050) Accession Number                    SH: '6551273001978417'
(0008, 0054) Retrieve AE Title                   AE: 'AE_TITLE'
(0008, 0060) Modality                            CS: 'CT'
(0008, 0070) Manufacturer                        LO: 'SIEMENS'
(0008, 0090) Referring Physician's Name          PN: ''
(0008, 1030) Study Description                   LO: 'Diagnostic Pre-Surgery
Contrast Enhanced CT'
(0008, 1032)  Procedure Code Sequence  1 item(s) ----
   (0008, 0100) Code Value                          SH: 'CT Thorax w/Cont'
   (0008, 0102) Coding Scheme Designator            SH: 'CT Thorax w/Cont'
   (0008, 0104) Code Meaning                        LO: 'CT Thorax w/Contras
t'
   ---------
(0008, 103e) Series Description                  LO: '= NONE ='
(0008, 1080) Admitting Diagnoses Description     LO: ''
(0008, 1090) Manufacturer's Model Name           LO: 'Sensation 16'
(0008, 2111) Derivation Description              ST: ''
(0009, 0010) Private Creator                     LO: 'SIEMENS CT VA1 DUMMY'
(0009, 0011) Private Creator                     LO: 'EMAGEON STUDY HOME'
(0010, 0010) Patient's Name                      PN: 'R_043'
(0010, 0020) Patient ID                          LO: 'R_043'
(0010, 0030) Patient's Birth Date                DA: ''
(0010, 0040) Patient's Sex                       CS: 'O'
(0010, 1010) Patient's Age                       AS: ''
(0010, 1020) Patient's Size                      DS: None
(0010, 1030) Patient's Weight                    DS: None
(0010, 2160) Ethnic Group                        SH: ''
(0010, 2180) Occupation                          SH: ''
(0010, 21a0) Smoking Status                      CS: 'UNKNOWN'
(0010, 21b0) Additional Patient History          LT: ''
(0010, 21c0) Pregnancy Status                    US: 4
(0010, 4000) Patient Comments                    LT: ''
(0012, 0062) Patient Identity Removed            CS: 'YES'
(0012, 0063) De-identification Method            LO: 'Per DICOM PS 3.15 Anne
xE. Details in 0012,0064'
```

```
(0012, 0064)  De-identification Method Code Sequence  7 item(s) ----
   (0008, 0100) Code Value                          SH: '113100'
   (0008, 0102) Coding Scheme Designator            SH: 'DCM'
   (0008, 0104) Code Meaning                        LO: 'Basic Application C
onfidentiality Profile'
   ---------
   (0008, 0100) Code Value                          SH: '113101'
   (0008, 0102) Coding Scheme Designator            SH: 'DCM'
   (0008, 0104) Code Meaning                        LO: 'Clean Pixel Data Op
tion'
   ---------
   (0008, 0100) Code Value                          SH: '113105'
   (0008, 0102) Coding Scheme Designator            SH: 'DCM'
   (0008, 0104) Code Meaning                        LO: 'Clean Descriptors O
ption'
   ---------
   (0008, 0100) Code Value                          SH: '113107'
   (0008, 0102) Coding Scheme Designator            SH: 'DCM'
   (0008, 0104) Code Meaning                        LO: 'Retain Longitudinal
With Modified Dates Option'
   ---------
   (0008, 0100) Code Value                          SH: '113108'
   (0008, 0102) Coding Scheme Designator            SH: 'DCM'
   (0008, 0104) Code Meaning                        LO: 'Retain Patient Char
acteristics Option'
   ---------
   (0008, 0100) Code Value                          SH: '113109'
   (0008, 0102) Coding Scheme Designator            SH: 'DCM'
   (0008, 0104) Code Meaning                        LO: 'Retain Device Ident
ity Option'
   ---------
   (0008, 0100) Code Value                          SH: '113111'
   (0008, 0102) Coding Scheme Designator            SH: 'DCM'
   (0008, 0104) Code Meaning                        LO: 'Retain Safe Private
Option'
   ---------
(0013, 0010) Private Creator                        LO: 'CTP'
(0013, 1010) Private tag data                       UN: b'LungCT-Diagnosis'
(0013, 1013) Private tag data                       UN: b'43205030'
(0018, 0015) Body Part Examined                     CS: 'LUNG'
(0018, 0050) Slice Thickness                        DS: '3.0'
(0018, 0060) KVP                                    DS: '120.0'
(0018, 0081) Echo Time                              DS: None
(0018, 0088) Spacing Between Slices                 DS: None
(0018, 0090) Data Collection Diameter               DS: '500.0'
(0018, 1000) Device Serial Number                   LO: ''
(0018, 1020) Software Versions                      LO: 'syngo CT 2006G'
(0018, 1030) Protocol Name                          LO: '1_3mm_THORAX_W'
(0018, 1100) Reconstruction Diameter                DS: '350.0'
(0018, 1110) Distance Source to Detector            DS: '1040.0'
(0018, 1111) Distance Source to Patient             DS: '570.0'
(0018, 1120) Gantry/Detector Tilt                   DS: '0.0'
(0018, 1130) Table Height                           DS: '168.0'
(0018, 1140) Rotation Direction                     CS: 'CW'
(0018, 1150) Exposure Time                          IS: '500'
(0018, 1151) X-Ray Tube Current                     IS: '433'
(0018, 1152) Exposure                               IS: '216'
(0018, 1160) Filter Type                            SH: '0'
(0018, 1170) Generator Power                        IS: '44'
```

```
(0018, 1190) Focal Spot(s)                        DS: '1.2'
(0018, 1200) Date of Last Calibration             DA: '19980803'
(0018, 1201) Time of Last Calibration             TM: '064448.000000'
(0018, 1210) Convolution Kernel                   SH: 'B40f'
(0018, 5100) Patient Position                     CS: 'HFS'
(0018, 5101) View Position                        CS: ''
(0019, 0010) Private Creator                      LO: 'SIEMENS CT VA0  COAD'
(0019, 1090) [Osteo Offset]                       DS: '0.0'
(0019, 1092) [Osteo Regression Line Slope]        DS: '0.966'
(0019, 1093) [Osteo Regression Line Intercept]    DS: '0.88'
(0019, 1096) [Osteo Phantom Number]               IS: '0'
(0019, 10b0) [Feed per Rotation]                  DS: '12.0'
(0020, 000d) Study Instance UID                   UI: 1.3.6.1.4.1.14519.5.2.
1.4320.5030.227672022111453893046049224932
(0020, 000e) Series Instance UID                  UI: 1.3.6.1.4.1.14519.5.2.
1.4320.5030.167938164374243671184910060739
(0020, 0010) Study ID                             SH: ''
(0020, 0011) Series Number                        IS: '2'
(0020, 0012) Acquisition Number                   IS: '2'
(0020, 0013) Instance Number                      IS: '7'
(0020, 0020) Patient Orientation                  CS: ''
(0020, 0032) Image Position (Patient)             DS: [-161.658203125, -342.6
58203125, 410]
(0020, 0037) Image Orientation (Patient)          DS: [1, 0, 0, 0, 1, 0]
(0020, 0052) Frame of Reference UID               UI: 1.3.6.1.4.1.14519.5.2.
1.4320.5030.161351896859611677292005139036
(0020, 0060) Laterality                           CS: ''
(0020, 1002) Images in Acquisition                IS: '0'
(0020, 1040) Position Reference Indicator         LO: ''
(0020, 1041) Slice Location                       DS: '410.0'
(0020, 4000) Image Comments                       LT: ''
(0021, 0010) Private Creator                      LO: 'SIEMENS MED'
(0021, 1011) [Target]                             DS: [13, 0]
(0028, 0002) Samples per Pixel                    US: 1
(0028, 0004) Photometric Interpretation           CS: 'MONOCHROME2'
(0028, 0006) Planar Configuration                 US: 0
(0028, 0010) Rows                                 US: 512
(0028, 0011) Columns                              US: 512
(0028, 0030) Pixel Spacing                        DS: [0.68359375, 0.6835937
5]
(0028, 0034) Pixel Aspect Ratio                   IS: [0, 0]
(0028, 0100) Bits Allocated                       US: 16
(0028, 0101) Bits Stored                          US: 12
(0028, 0102) High Bit                             US: 11
(0028, 0103) Pixel Representation                 US: 0
(0028, 0106) Smallest Image Pixel Value           US: 0
(0028, 0107) Largest Image Pixel Value            US: 3088
(0028, 0303) Longitudinal Temporal Information M CS: 'MODIFIED'
(0028, 1050) Window Center                        DS: '40.0'
(0028, 1051) Window Width                         DS: '400.0'
(0028, 1052) Rescale Intercept                    DS: '-1024.0'
(0028, 1053) Rescale Slope                        DS: '1.0'
(0028, 1054) Rescale Type                         LO: ''
(0028, 1055) Window Center & Width Explanation    LO: ['WINDOW1', 'WINDOW2']
(0028, 2110) Lossy Image Compression              CS: ''
(0029, 0010) Private Creator                      LO: 'SIEMENS CSA HEADER'
(0029, 0011) Private Creator                      LO: 'SIEMENS MEDCOM HEADER'
(0029, 1131) [PMTF Information 1]                 LO: '4.0.5659670'
(0029, 1132) [PMTF Information 2]                 UL: 524288
```

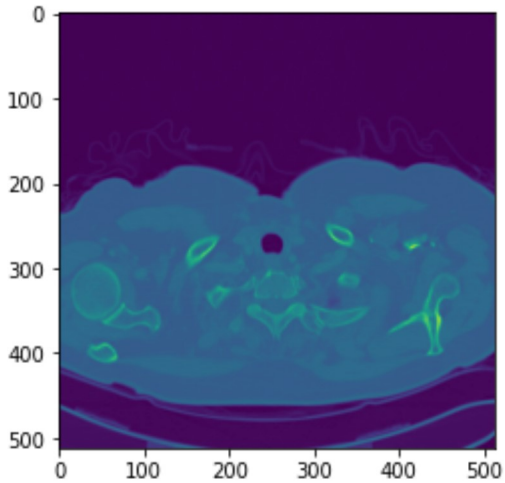
```
(0029, 1133) [PMTF Information 3]               UL: 0
(0029, 1134) [PMTF Information 4]               CS: 'DB TO DICOM'
(0032, 000c) Study Priority ID                  CS: 'LOW'
(0032, 1030) Reason for Study                   LO: ''
(0032, 1060) Requested Procedure Description    LO: 'CT Thorax w/Contrast'
(0032, 1064)  Requested Procedure Code Sequence 1 item(s) ----
   (0008, 0100) Code Value                         SH: 'CT Thorax w/Cont'
   (0008, 0102) Coding Scheme Designator           SH: 'CT Thorax w/Cont'
   (0008, 0104) Code Meaning                       LO: 'CT Thorax w/Contras
t'
   ---------
(0032, 4000) Study Comments                     LT: ''
(7fe0, 0010) Pixel Data                         OW: Array of 524288 element
s
```

That's a lot of metadata, right? Don't be scared. Next, let's actually look at the slice.

We can see that there are many data fields in a DICOM file. There's a lot of patient information stored in a DICOM - name, birthdate, and so on. For obvious privacy reasons, this data has been completely anonymized.

Run the following cell to view the image associated with this person.

In [ ]:
```
rawimg= ds.pixel_array
plt.imshow(rawimg, cmap='viridis')
plt.show()
print(type(rawimg), np.mean(ds.pixel_array), rawimg.dtype)
```

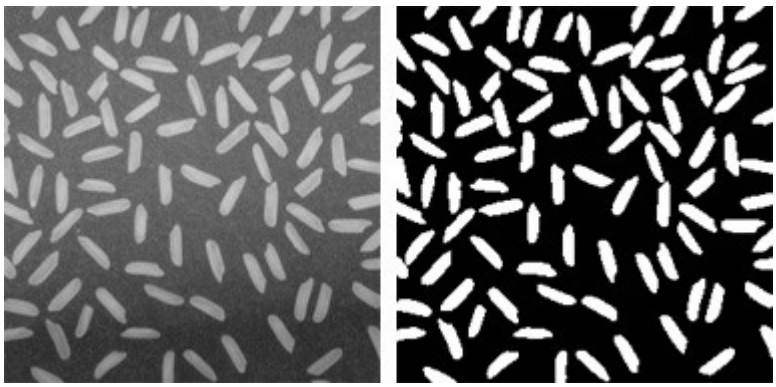



```
<class 'numpy.ndarray'> 545.1571922302246 uint16
```

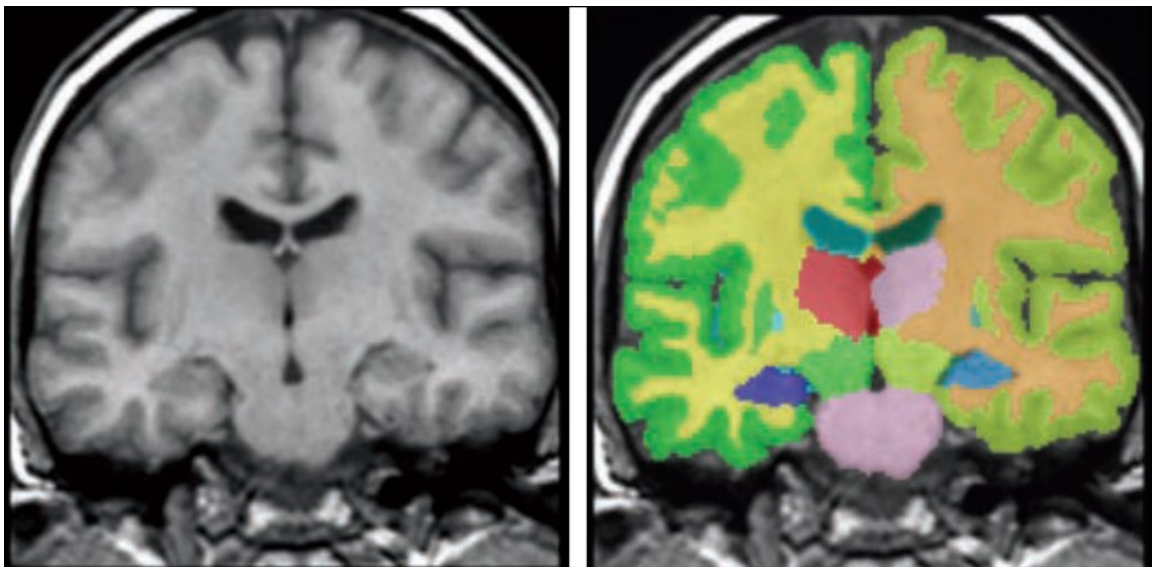
We're all set. On to the problem set!

Introduction

Segmentation is the process of dividing a given image into sections. This can be binary segmentation (if you wanted to separate the image into foreground and background) or a multilabel segmentation if you wanted to label many different parts of an image (see figures below for examples of each). Segmentation is an extremely interesting problem in image analysis, and it has yet to be solved perfectly. From an algorithmic point of view, we can approach the problem using tools such as graph cuts, watershed, flood fill / region growing, statistical set separation, etc... From a machine learning point of view, we can think of the problem as a pixel classification problem that can be solved with some supervised learning algorithm (assuming we had some ground truth training data) or an unsupervised learning algorithm (such as k-means clustering).



(a) Binary Segmentation - A binary segmentation of some grains of rice. We can use segmentation to create a binary mask that allows us to separate the rice (foreground) from the darker background.



(b) Multilabel Segmentation - A multilabel segmentation of a coronal section from a magnetic resonance image of the brain. We can see that different colors represent different sections of the brain. For example, the white matter on the left side of the brain is labeled

sections of the brain. For example, the white matter on the left side of the brain is labeled with a different color than the white matter on the right side of the brain.

A lot of radiological image analysis begins with segmentation. For example, to quantitatively describe a lung nodule, you may want to collect data on its edge sharpness, its total volume, and/or the mean intensity of voxels in the nodule. However, to do this, we have to first be able to segment the nodule and cleanly define its boundaries. This can get extremely hairy. For example, in the chest, it might be hard to teach a program to segment a nodule in the lung from a seed voxel. It's important to make sure the nodule segmentation doesn't bleed into the pericardium, as the voxel intensities for a nodule and the fleshy bits in the pericardium are very similar. We can try and restrict this through lung field segmentation (basically, we want to go through our chest CT scans and segment out the lungs) and thus define a region of interest, but this gets increasingly difficult as more and more problems pop up.

This assignment will help you understand many core concepts taught in the lectures and hopefully integrate the lectures into a medically relevant application. The goals of this project include:

a. Understanding medical image data format - specifically, DICOM.

b. Understanding thresholding techniques - specifically, Otsu's Threshold.

c. Understanding morphological image analysis techniques, and expanding this understanding into 3D.

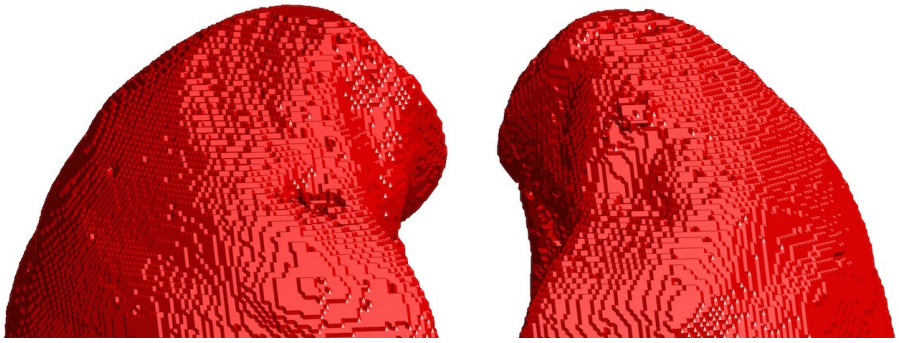
d. Understanding image convolution and expanding it into efficient 3D.

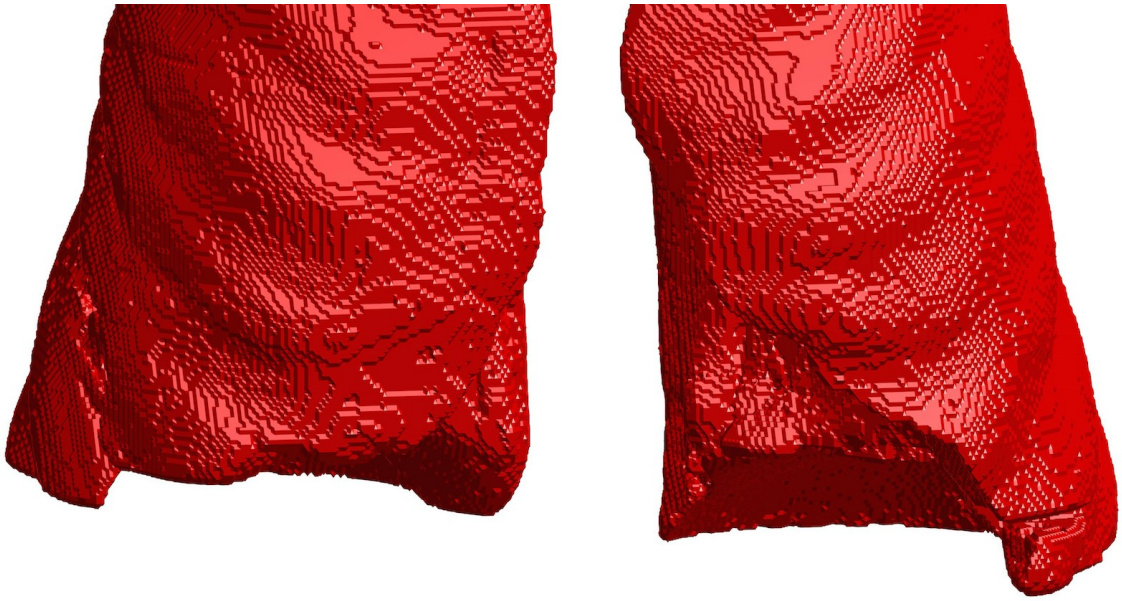
e. Understanding visualization in 3D.

f. Embracing creativity and exploring image analysis.

These can seem daunting tasks for a first timer, but we promise the final results will be amazingly cool! Stick with us, and as always, don't hesitate to ask for help.

The main goal of this assignment is to segment out the lungs, and only the lungs, from these CT slices, and then create a 3D rendering of the lungs to give yourself some way to qualitatively assess how good your segmentation is. Basically, given a chest CT series, you will be creating something like this:

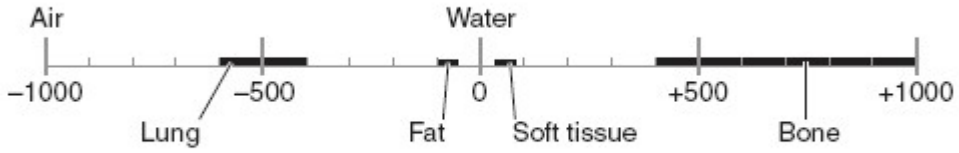




Step 1: (5 points)

Some important DICOM fields like $RescaleIntercept$ and $RescaleSlope$ determine how the image pixel values should be interpreted. These metadata are critical for quantitative imaging methods like CT. The default raw pixel values are arbitrary units returned from the actual machine used and may differ based on the scanner manufacturer. We'd like to convert these raw values to Hounsfield units, which you learned about in lecture, in order to have some standardization among all of our CT scans. The conversion formula is:

$$\text{Hounsfield Units} = RescaleSlope \times \text{Raw Image} + RescaleIntercept$$



Bone	+400 → +1000
Soft tissue	+40 → +80
Water	0
Fat	−60 → −100
Lung	−400 → −600
Air	−1000

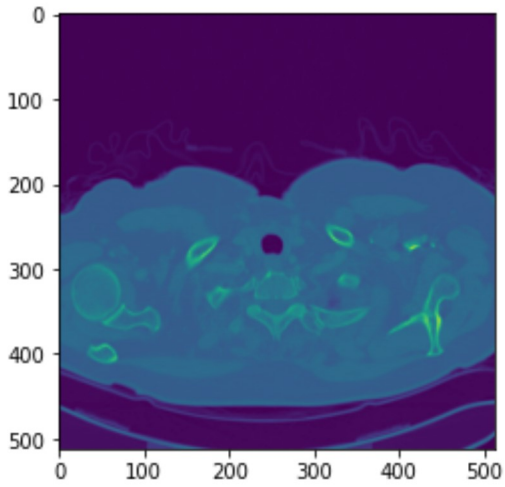
Fig. 1.4 *The Hounsfield scale of CT numbers.*

Deliverable:

- Read in the raw data for a CT slice, and convert its pixel values into Hounsfield units

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 2 cells for this question, but on

################################################################################
```

```
--- Image stats ---
max: 3088, min: 0, mean: 545.1571922302246, stddev: 502.05132322225245
--- HU Image stats ---
max: 2064.0, min: -1024.0, mean: -478.8428077697754, stddev: 502.05132322225
245
--- Difference ---
max: 1024.0, min: 1024.0, mean: 1024.0, stddev: 0.0
```



Step 2: (25 points)

As you may have noticed, we live in a 3D world. The next main part of your problem will be to go from all the individual slices of the CT scan (each one stored as a .dcm file) into a 3D volume. Basically, you'll need to figure out some way to order these slices in the right order. Read in a few of the DICOM images from your patient, and try using different DICOM fields as a sorting key. You may find one of them works well in sorting the slices in the correct order. Try looking up what that field means in the DICOM standard. *Hint: you may want to initialize a blank 3D matrix called* `volCT` *or something.*

Here is the pseudocode for one way (you are free to do your own) of creating this volume and filling it:

Algorithm 1: Creating Volume
Relevant Commands: `os.listdir`, `dicom.read_file`, `numpy.zeros`, `numpy.shape`, `sort`

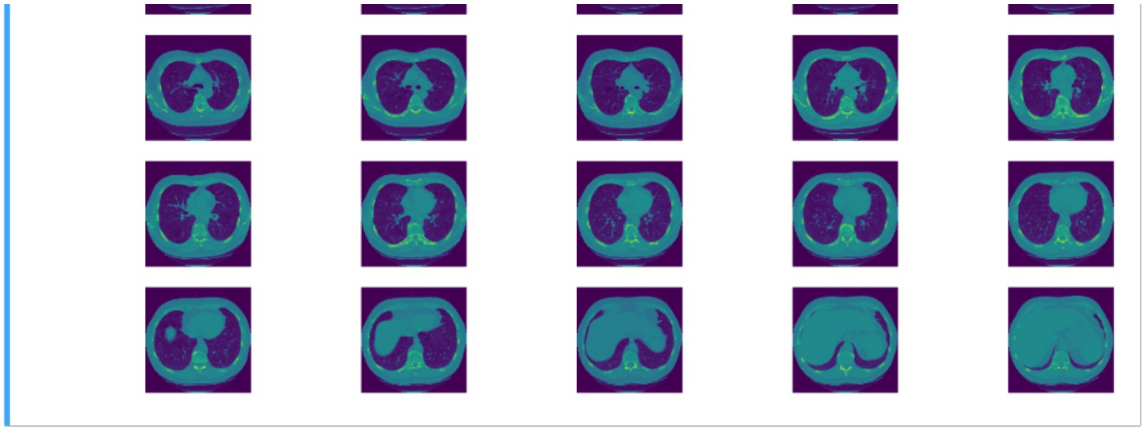
---

initialize `vol` = matrix of size [#pix$_{width}$, #pix$_{height}$, #slices]
`for` each .dcm in folder:
  Read in .dcm
  Store number from header that acts as proxy for slice order.
`endfor`
`order` the saved numbers to index each DICOM file by slice order.
`for` each .dcm in folder:
  `s` = Slice Order of DICOM image
  Read in image as `img`
  `vol(:, :, s) = img;`
`endfor`

Visualize the NumPy volume you create from the stacks. Feel free to do this within Python with something like matplotlib.

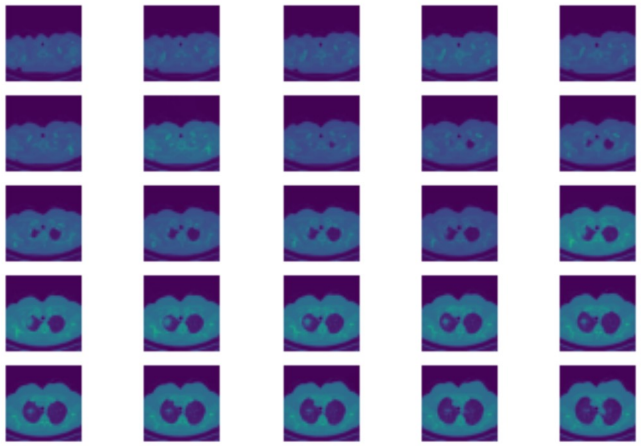
Deliverable:

Make the 3D volume and display 25 of the slices in correct order for us, like this:



In [ ]:
```
### WRITE CODE IN HERE. You can have up to 4 cells for this question, but on

##########################################################################
```

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 4 cells for this question, but on



##########################################################################
```



Step 3 (5 points)

If you dont already know about numerical types, take a look here and/or here.

Deliverable:

What is the default numerical type used to store raw image data in DICOM files? Answer in the cell below.

In the coding cell below that one, convert all the voxels in your volume to a **float32** type number between 0.0 and 1.0; output the min, max, and dtype (datatype) of the voxels in your volume before and after you convert.

Answer: YOUR ANSWER TO STEP 3 PART 1 HERE

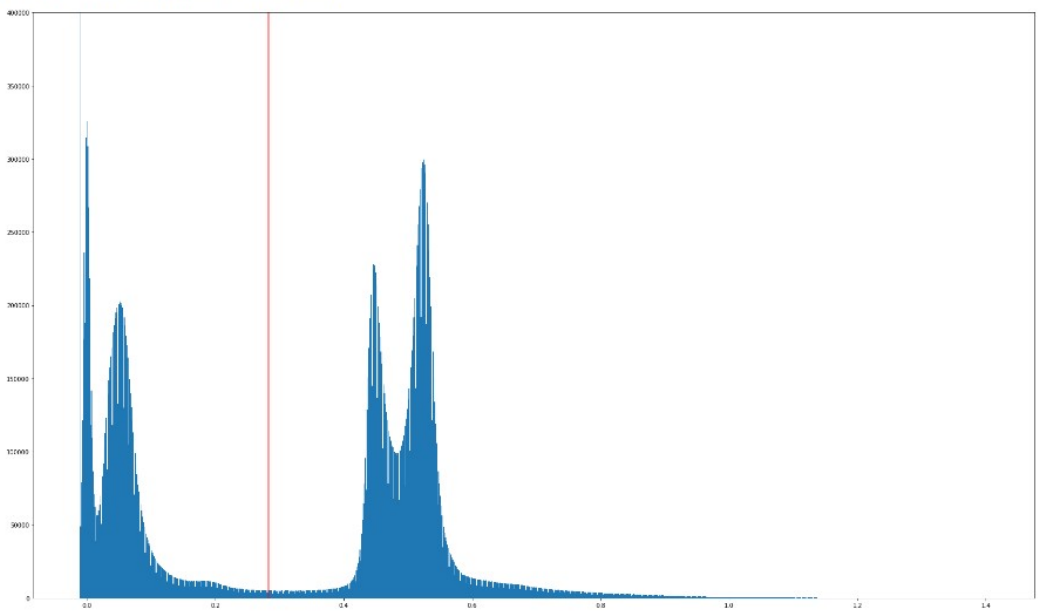
In [ ]:
```
### WRITE CODE IN HERE. You can have up to 2 cells for this question, but on



##########################################################################
```

```
Before normalization: min - 0, max - 4095 dtype - uint16
After normalization: min - 0.0, max - 1.0 dtype - float32
```

Step 4 (10 points)

Now that we have our 3D matrix of lung CT data, we can try to segment out our lung. We know from lecture that the pixel values of CT scans are given in Hounsfield units, where lower Hounsfield units correspond to low density materials (like air) that are not highly attenuative for X-rays and higher Hounsfield units correspond to highly attenuative materials, like bone.

```
In [23]: _ = plt.hist(vol.reshape([-1]), 1000)
         plt.ylim([0,400000])
         plt.plot([val,val], [0,400000], 'r')
Out[23]: [<matplotlib.lines.Line2D at 0x4e6a62b0>]
```



That red line separating the two groups of pixels is the Otsu's threshold. We can find a nice separating value of our two modes with this Otsu's method.

Deliverable:

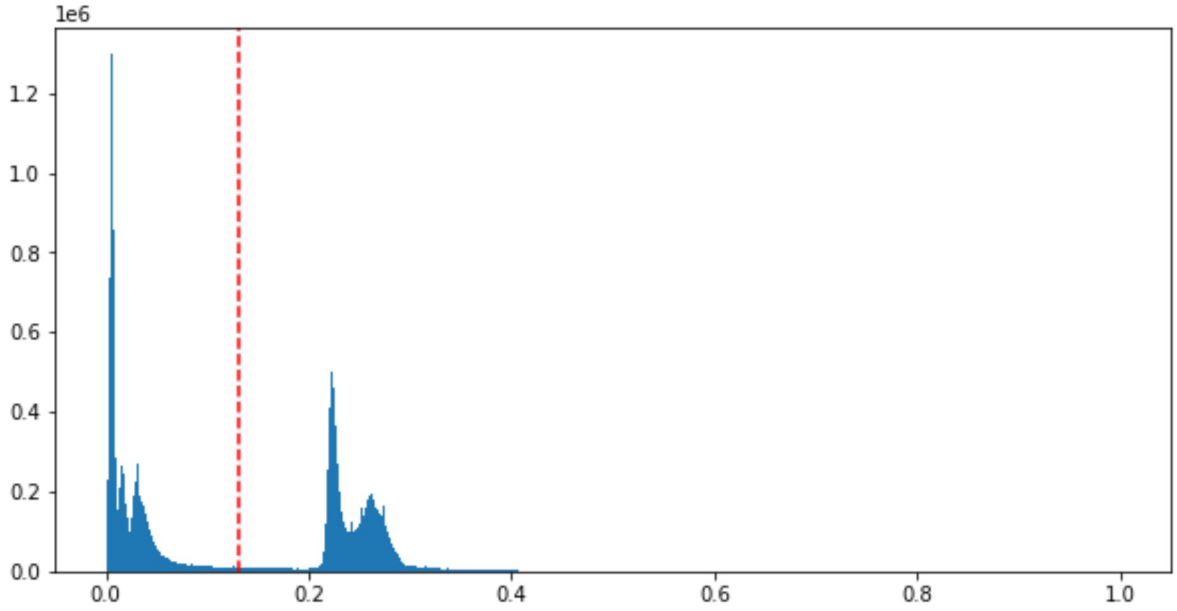
A histogram of the Hounsfield units from a typical CT scan will be significantly bimodal! Why?

Answer: YOUR ANSWER TO STEP 4 PART 1 HERE

Next, find the Otsu's threshold in your volume of pixels and plot the histogram of of your pixels with this line like the example figure. You will find these links (1, 2, and 3) useful if you have not done this before.

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 2 cells for this question, but on



################################################################################
```

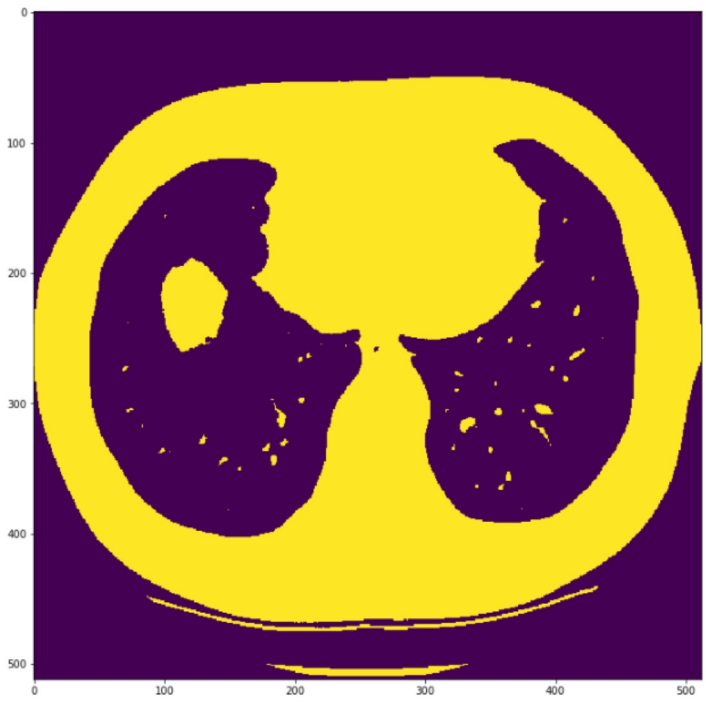
Out[ ]:
```
<matplotlib.lines.Line2D at 0x7fcf321c8790>
```



Step 5 (10 points)

We can now take a look at the performance of Otsu's threshold. We want all the pixels less than Otsu's threshold since the lung is in the darker (low Hounsfield units) part of the image.

```
In [27]: val = filters.threshold_otsu(vol[vol>0])
         plt.imshow(vol[:,:,100] > val)
         plt.rcParams['figure.figsize'] = (15,9)
```



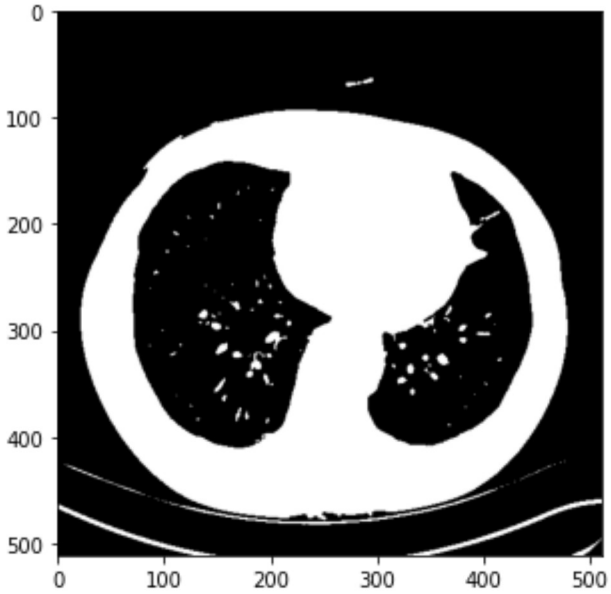
Despite the crude binary cutoff, we can see that this is really close to what we want for a final result, albeit a little noise.

Deliverable:

Display for us one slice of the CT scan showing a binary image after applying thresholding using Otsu's method.

```
In [ ]: ### WRITE CODE IN HERE. You can have up to 2 cells for this question, but on

        ##########################################################################
```

```
Out[ ]: <matplotlib.image.AxesImage at 0x7fcf3188b1c0>
```


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

Step 6 (30 points)

We can see our dark lungs surrounded by lighter-colored tissue, which is then again surrounded by the darkness of the 0-pixels. Our lung segmentation is thus, interestingly, completely separated from the segmentation of the outside air. These two sets of binary-connected components are completely unconnected.

> Can you think of a simple way of differentiating between the dark lungs and the dark air outside the body?

Here's one way to do it - we assume that the outside air will always be touching the edge of the image and the lungs will not:

Algorithm 2: Segmentation

Relevant Commands: `skimage.filters.threshold_otsu`, `skimage.measure.label`

```
Find Otsu's threshold of vol, thresh.
Make binary volume mask of anything below thresh, volBW.
for each slice s:
   Find all binary connected components of slice s.
   for all connected components i:
      if connected component i touches the edge of the image:
         delete connected component i.
      endif
   endfor
endfor
```

However, the segmentation is still rough around the edges. There are a few pixels of noise around the main lung (components that weren't touching the edges but still aren't part of the lung). Luckily for us, these bits are relatively small in size. We can, therefore, quickly filter them out based on their volume. We want to choose connected components whose volume is larger than some threshold V. You can set V arbitrarily (most noise will have pixel counts in the double digits or maybe low hundreds), as long as the lung volume should have a pixel count several orders of magnitude higher. Here would be a possible pseudo-algorithm for this task:

Algorithm 3: Volume Filter

```
Find all the 3D connected components in your volBW.
for each connected component i:
   if connected component's volume < V:
```

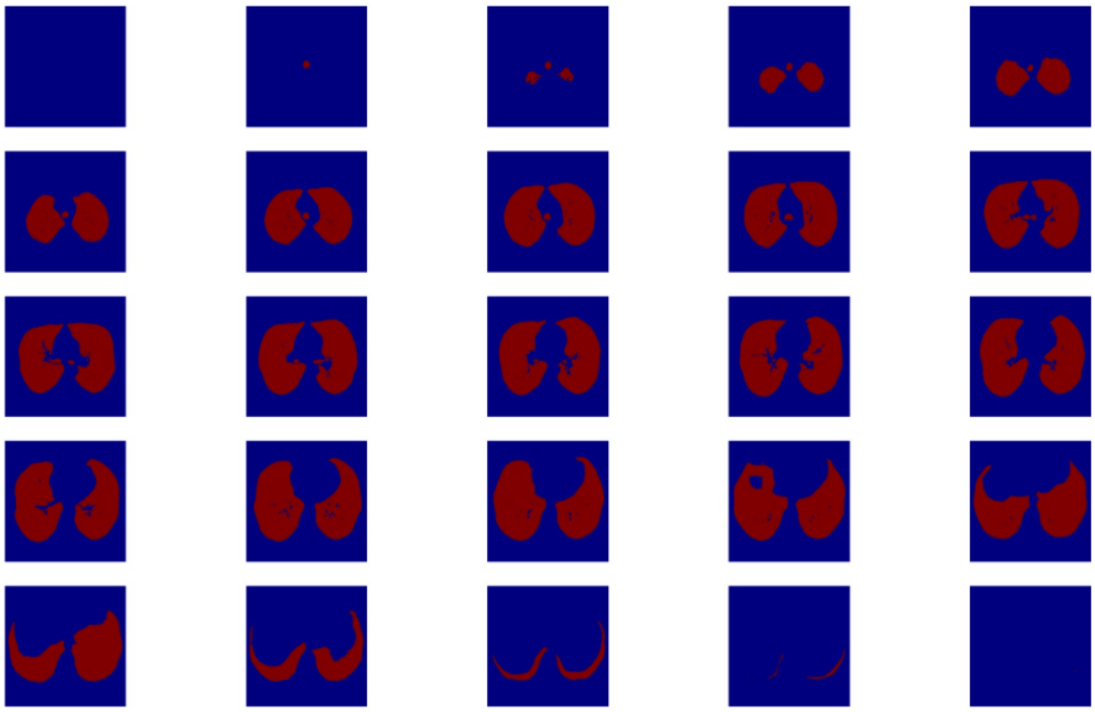
Delete Connected Component i.
endif
endfor

After this, we just want to smooth out the edges. The segmentation of our lung looks very ragged because it wasn't able to pick out the lighter colored blood vessels in the lung. We can do this smoothing with a binary closing algorithm, provided by the `scikit-image` package (`skimage.morphology.binary_closing`). It works for both 2-dimensional and 3-dimensional binary matrices.

Deliverable:

Implement the steps described here and display for us an ordered sample of the cuts from your boolean CT scan volume that has been smoothed, with only the pixels of the lung and trachea as 1/true, and everything else as 0/False, like this:

```
fix,ax = plt.subplots(5,5)
for i in range(5):
    for j in range(5):
        ax[i,j].imshow(smoothed_ovol[:,:,(5*i+j)*5])
        ax[i,j].axis('off')
plt.rcParams['figure.figsize'] = (30,18)
```



In [ ]:
```
### WRITE CODE IN HERE. You can have up to 5 cells for this question, but on

##########################################################################
```

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 5 cells for this question, but on
```

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 5 cells for this question, but on


################################################
```

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 5 cells for this question, but on

################################################################################
```

Step 7 (5 points each, 15 points total)

Show that the segmentation works on 3 more scans.

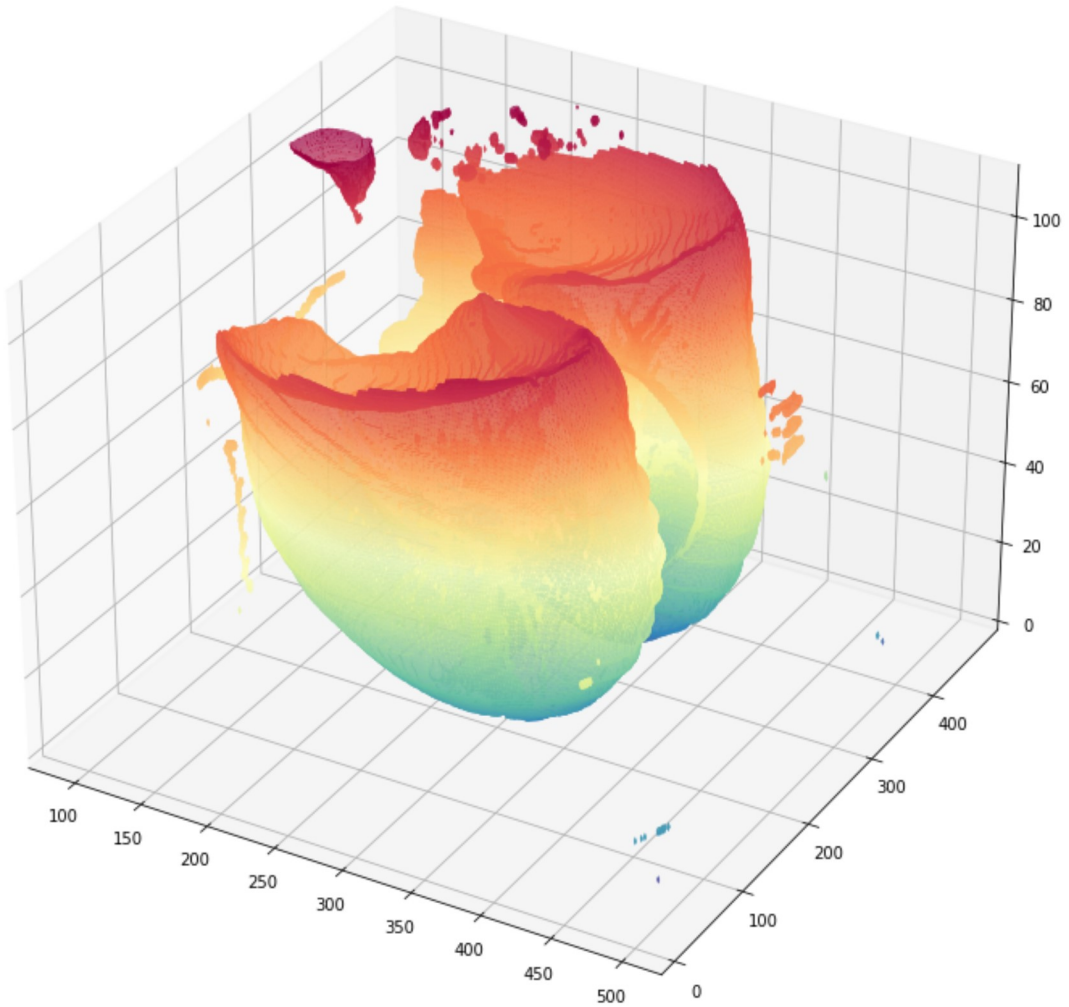
Deliverable:

Same as Step 6, but with different scans.

Step 8 (Bonus 15 points, 5 points each)

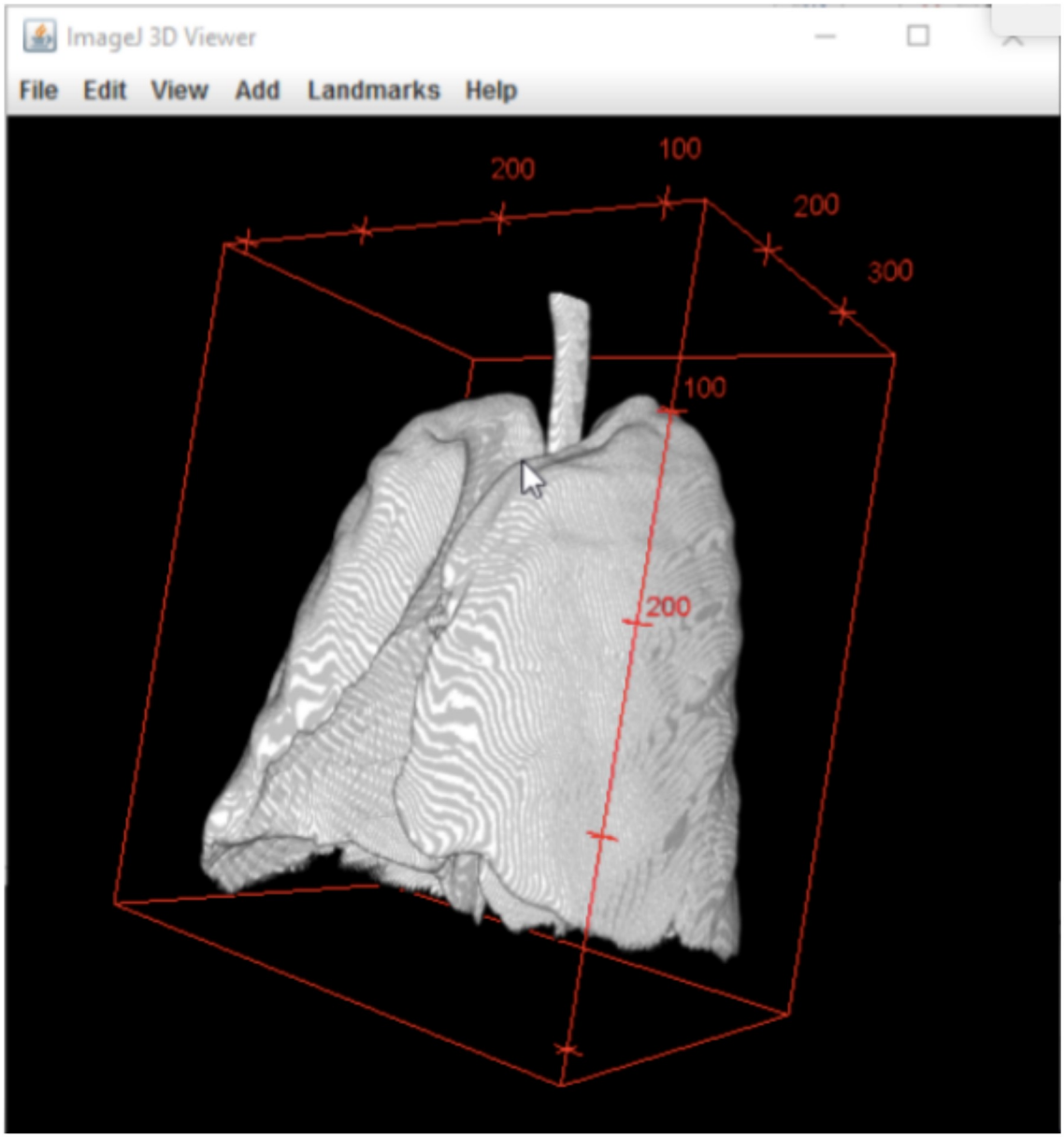
If you've finished everything above, you've already done a full fledged segmentation. However, for an extra challenge, you can try your hand at:

1. Removing just the trachea so that your segmentation is just on the left and right lung without any of the trachea or bronchi.
2. Separating out the two halves of the lung into left/right lungs.
3. There are a ton of python packages that help you visualize the lung as a whole. We highly recommend looking up some **marching cube algorithms**. You can also stay with matplotlib and scikit-image. Which should output a figure like this:



Another, easier, way to go would probably be using something like Fiji. Simply download Fiji (available for Windows, Mac, and Linux). You can save all your 3D segmentation as binary image slices. Then, using Fiji, simply read in your image sequence (be sure to name your

image slices. Then, using Fiji, simply read in your image sequence (be sure to name your images some numeric count or something so that Fiji knows what order to read the images in). Then, go to `Plugins -> 3D Viewer`. This should give you a figure like what you see below:



Show us that this works on at least two scans.

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 5 cells for this question, but on

##########################################################################
```

In [ ]:
```
### WRITE CODE IN HERE. You can have up to 5 cells for this question, but on

##########################################################################
```