Preamble

Hello everyone, I made this repository because it’s the proper way of managing codes/scripts/functions/programs and whatnot. Often there are many edits to scripts, and this is a good way of keeping track of those edits. The scripts that are included here were written simply to help me manage dynamic CT data and as they are now, are written specifically for managing the data collected on January 8th, 2020. The scripts can be roughly grouped into three sections.

Python scripts end in .py, MATLAB scripts end in .m ImageJ macros end in .txt or .ijm if written in Fiji. I like to name functions that are called to with a ‘f\_’ in front of the name e.g. ‘f\_function.py’. This is because I wouldn’t edit a called function whereas most of scripts I would edit ‘on the spot’.

I like to ‘comment out’ a section of my script with an if statement. For example, I will always have a variable ‘run = 1’ at the beginning of my script. A section that I ‘comment out’ will come after ‘if run == 0’.

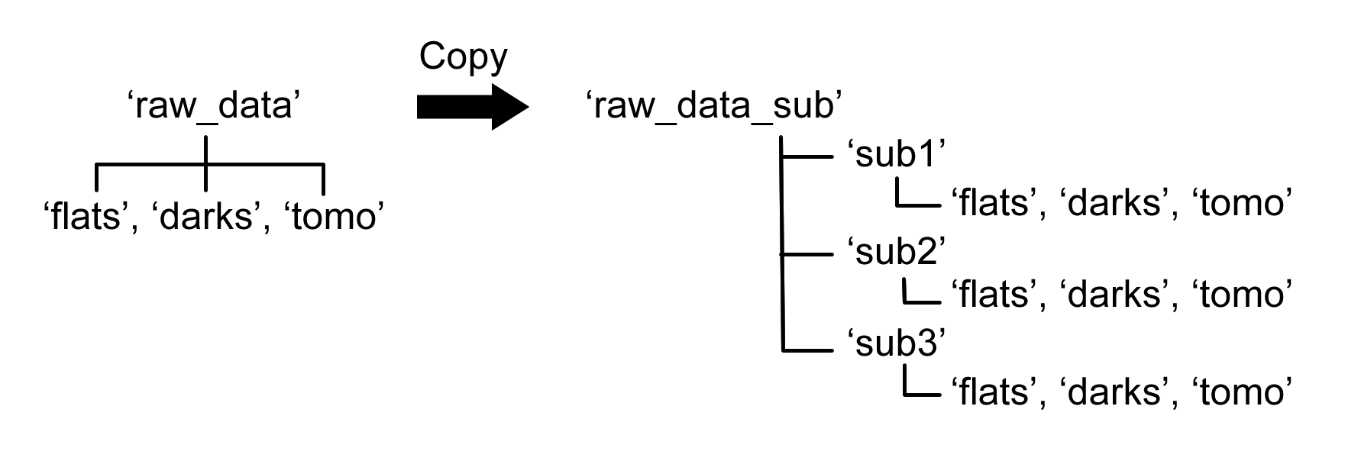
# Preprocessing

## Unwanted Files

The script ‘RemoveExcess.py’ will comb through all subdirectories and delete problematic files like ‘Thumbs.db’, ‘reco.params’, or ‘im\_00001.tif.rec’.

## Creating Subfolders

‘MakeSubfolders.py’ is a way of organizing the large number of projections acquired for dynamic CT. This script needs to be placed in the raw data directory before the darks, flats, and tomo. It copies files from the original directory into a new directory, the raw data is left untouched.



The inputs are the name of the raw data directory, the number of projections per CT, the starting projection (i.e., the first time point), the last projection (i.e., the last point), and step between stating and last (i.e., the temporal resolution).

# Reconstruction

## UFO

There are many ways to reconstruct the data. EZ\_UFO can be used for reconstruction, but because there are so many time points across different datasets, I like to use ‘RunUFO.py’ because once I know the parameters, I can run it in a loop and automate this whole process. ‘RunUFO.py’ needs to be placed in the same directory as the subdirectory created by in section 1.2.

The inputs are the name of the dataset, the size of the reconstructed volume, the rotation center, whether to reconstruct the whole volume, a single slice, or a thick slice, and the number of projections per CT.

## Tomopy

'RunTomopy.py' along with 'f\_readstack.py' and 'f\_paganin.py' are used to reconstruct using Tomopy. All three scripts should be kept in the same directory as the directories containing 'flats', 'darks', and 'tomo'. This is useful when the BMIT server is not available. To use, create the following environment in anaconda:

conda create -n tomopy python=3.6.12

conda activate tomopy

conda install -c conda-forge tomopy=1.9.1

conda install -c conda-forge dxchange=0.1.5

conda install h5py=3.1.0

It should be fairly straightforward, for more information read here:

https://tomopy.readthedocs.io/en/latest/api.html

# Postprocessing

## Move Files out of ‘sli’

‘MoveFiles.py’ is a script that moves reconstructed images out of the ‘sli’ folder that automatically gets made. This is useful when reconstructed single slices in different locations e.g. the 300th slice. For dynamic data, each time point is in a different directory, the 300th slice at different time points are each in a different directory. Rather than moving them out individually, this script copies them into a central directory.

## Rotate and Reslice

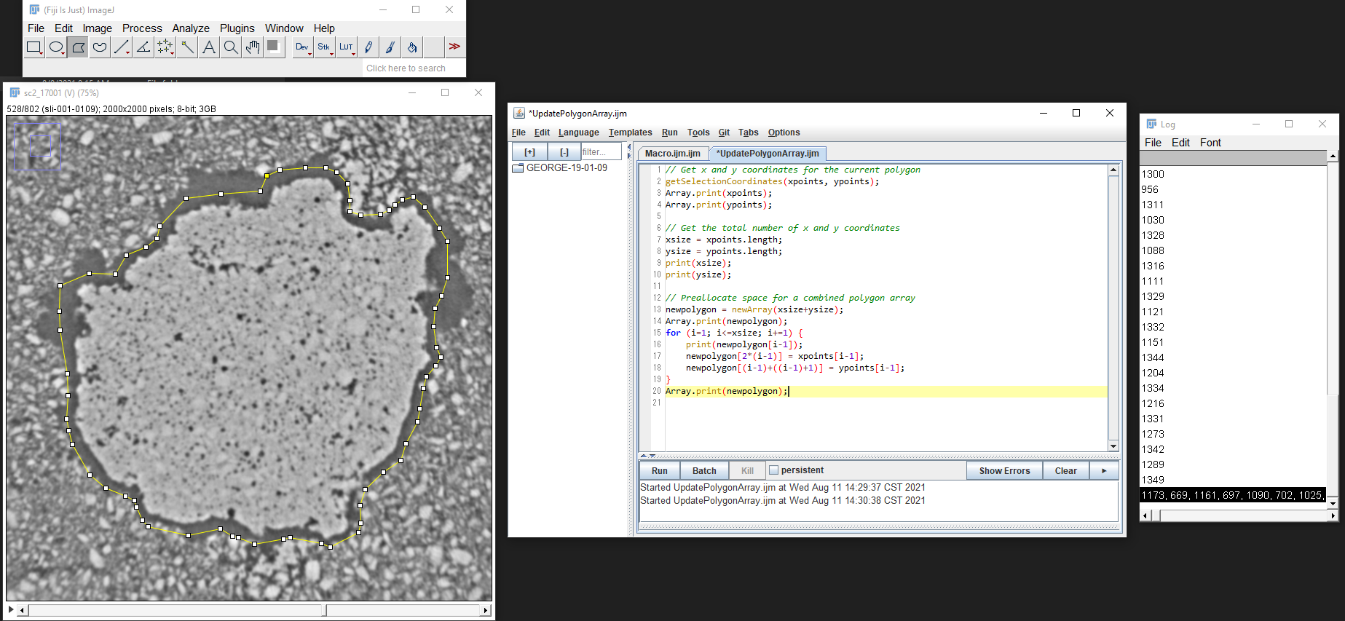
‘Rotate.imj’ and ‘Reslice.imj’ perform singular tasks. ‘Rotate.imj’ rotates the images in a folder so that they are all the same orientation. ‘Reslice.imj’ saves the orthogonal views of a volume.

Both are ImageJ macros and in the input section, type in the directory that contains the images or the working directory. As they are now, they run as a loop and the input directories change for each run, so I have them all written in an array and index to the necessary value.

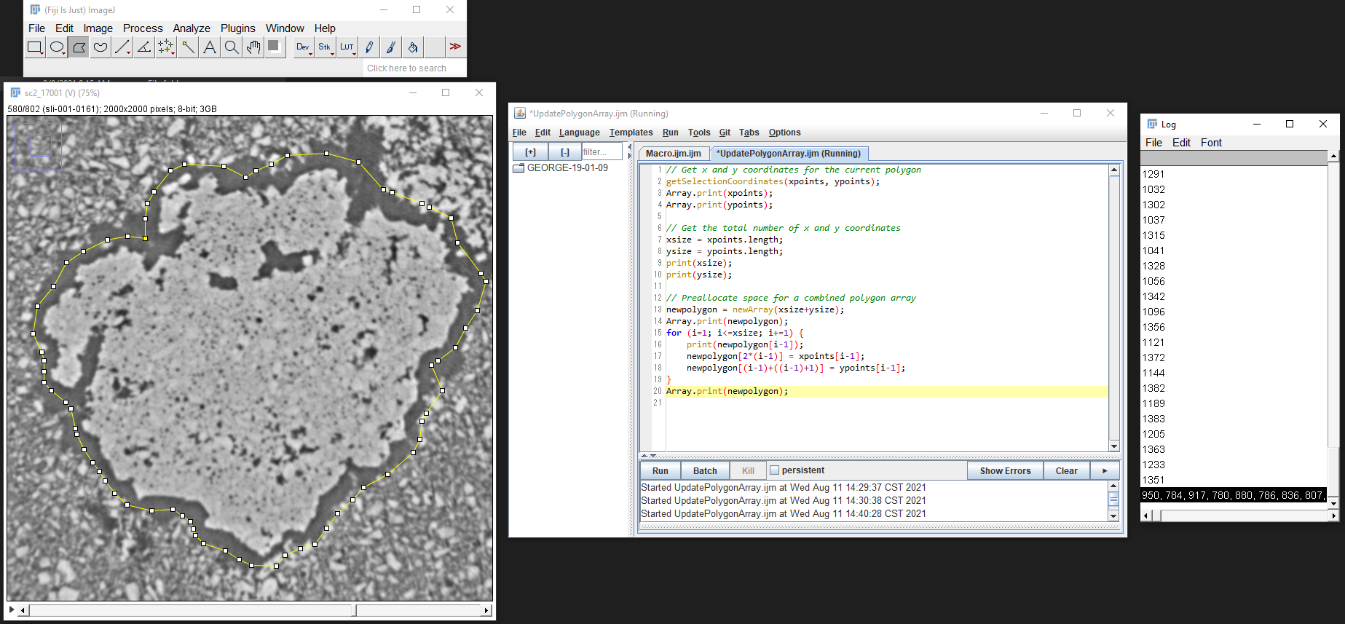
## Polygon Tool for Segmentation

‘UpdatePolygonArray.imj’ is a macro that returns an array of x and y values for whatever polygon is drawn in ImageJ or Fiji. This is useful for keeping track of segmenting.

Open an image and use the polygon tool draw around the ROI. Run ‘UpdatePolygonArray.imj’ and the last line will be an array of x and y values that make of the currently drawn polygon.



On a different position either in the stack or at a different time, update the polygon and run the macro again. The last output is a new array of x and y coordinates that correspond to the updated polygon.



## Calculate Stack ROI Area

‘CalcualteThresholdAreas.imj’ is a macro that calculates the thresholded area of each slice in a stack and then saves that information in a text file.