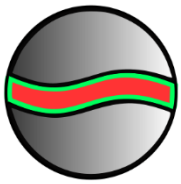
SFU FCReL

2018



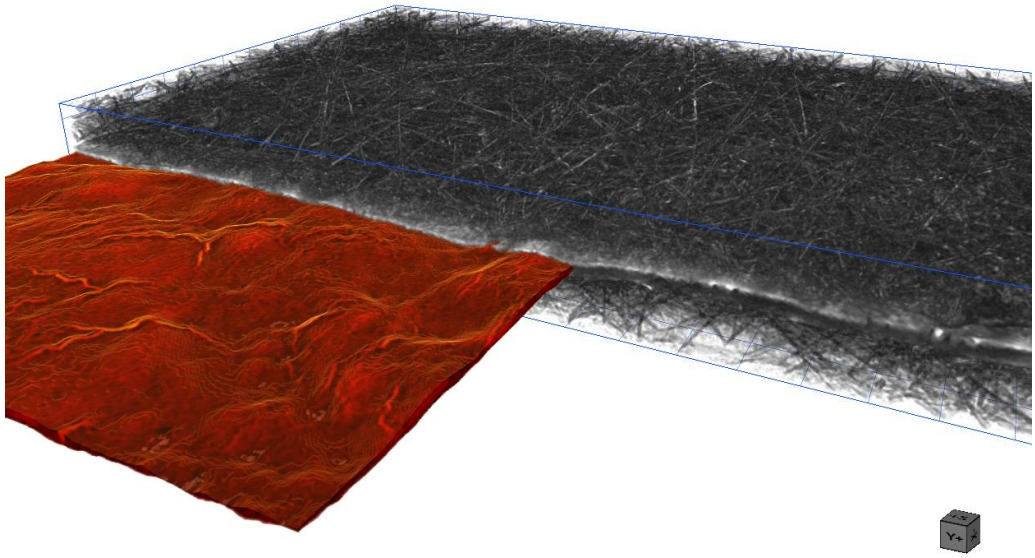
MemSeg

CUSTOM TOOL FOR SEGMENTATION OF PEFC MEMBRANE

ROBIN WHITE

Purpose

The location and greyscale value of the membrane in polymer electrolyte fuel cells following XCT imaging is often such that segmentation using conventional means is time consuming, inaccurate, and requires significant user knowledge of image processing steps. This makes for repeatability and consistency difficult. MemSeg aims to solve many of the problems through it's user interface and automated operation, and rapidly allow for segmentation of membrane layer from XCT image sets



Above shows output of MemSeg followed by visualization in DragonFly from ORS.

Use

MemSeg will output 5 image files following successful completion: A binary image stack of the full location of the membrane (ignoring cracks), a thickness image of the local thickness of the membrane (in pixels), interpolated boundary image stacks of the catalyst layers (used to determine membrane location – useful for confirming accuracy), a masked greyscale image (used in the above figure), as well as a single slice which conforms to the wave-y nature of the membrane (as opposed to a rigid slice). See below for details. This latter image file is particularly useful when looking at cracks through the membrane.

Installation

For installation instructions, please see the ReadMe located on the project Github.

Workflow

File management

For simplicity it is easiest if you create a new directory called <sample>_data (or whatever). In there, place the cropped and tilt/rotation corrected greyscale image stack. For details on this processing step,

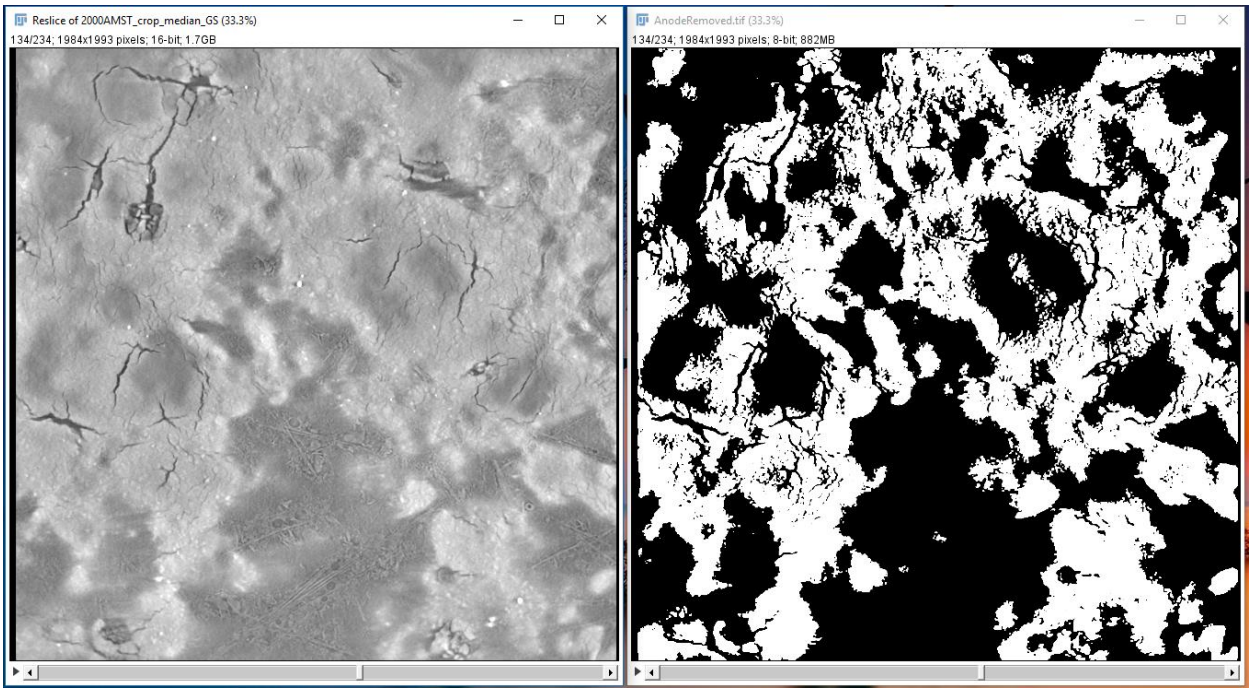
refer to Manual Stack Alignment section 2.2, 2.3, and 2.4; see https://github.com/robintwhite/NXCT/tree/master/ImageJ_macros. Call this <sample>_GS.tif . This will be helpful later and keeps the workflow organized.

Note < > indicates editable text for the user to choose. Of course, this is just a suggestion.

Boundaries using FIJI/ImageJ

Using this greyscale (GS) image stack in the <sample>_data directory, it is now necessary to find our boundaries. For this we will use the electrode layers (anode and cathode), since these are generally easier to differentiate from other components due to their high attenuation and bright greyscale values. To do this we will use CathodeSeparator and AnodeSeparator in Fiji. For a tutorial on using Cathode Separator please see the Cathode Separator tutorial documentation in the ImageJ_macros folder.

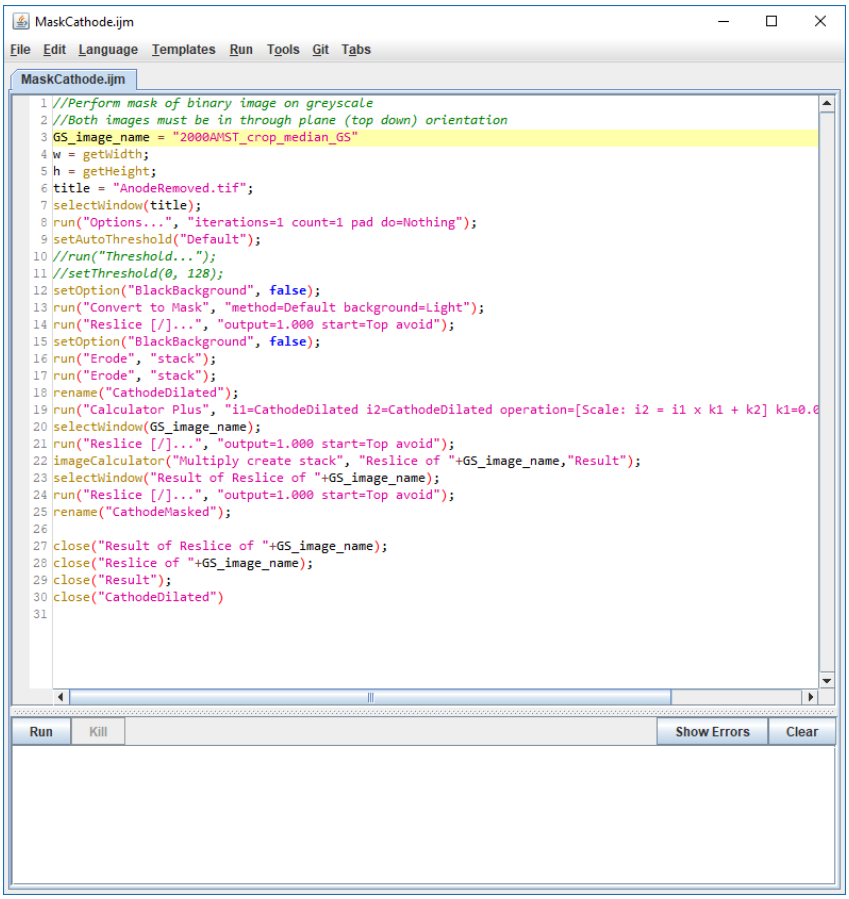
At this stage you should have an image stack called AnodeRemoved which is a binary image stack of **just** the cathode catalyst layer.



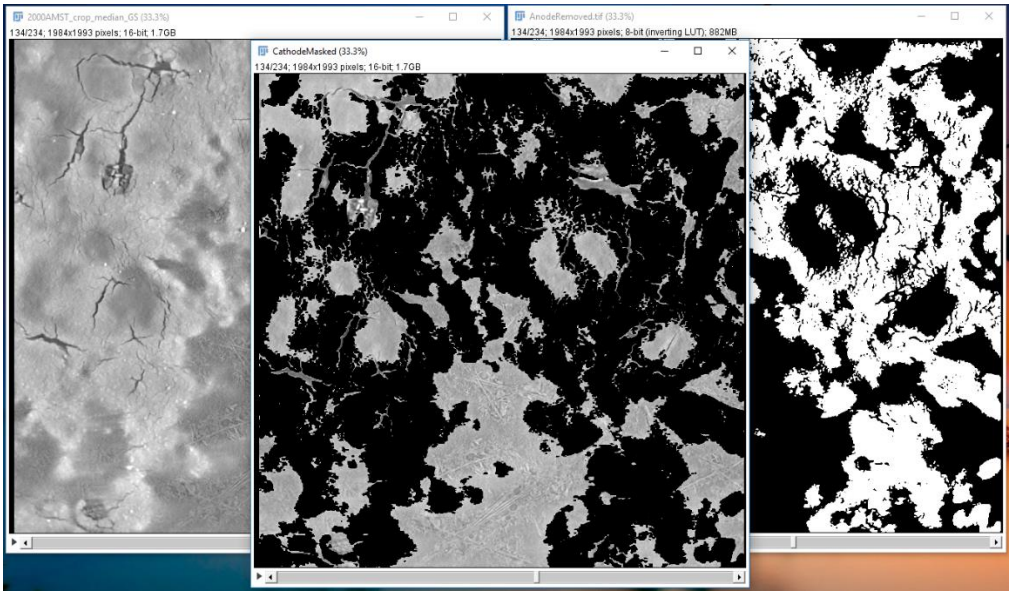
We will now use this to remove the cathode from the greyscale image to allow us to segment the anode layer.

From NXCT-master\ImageJ_macros, drag and drop MaskCathode.ijm into Fiji so that it opens in the editor (see below).

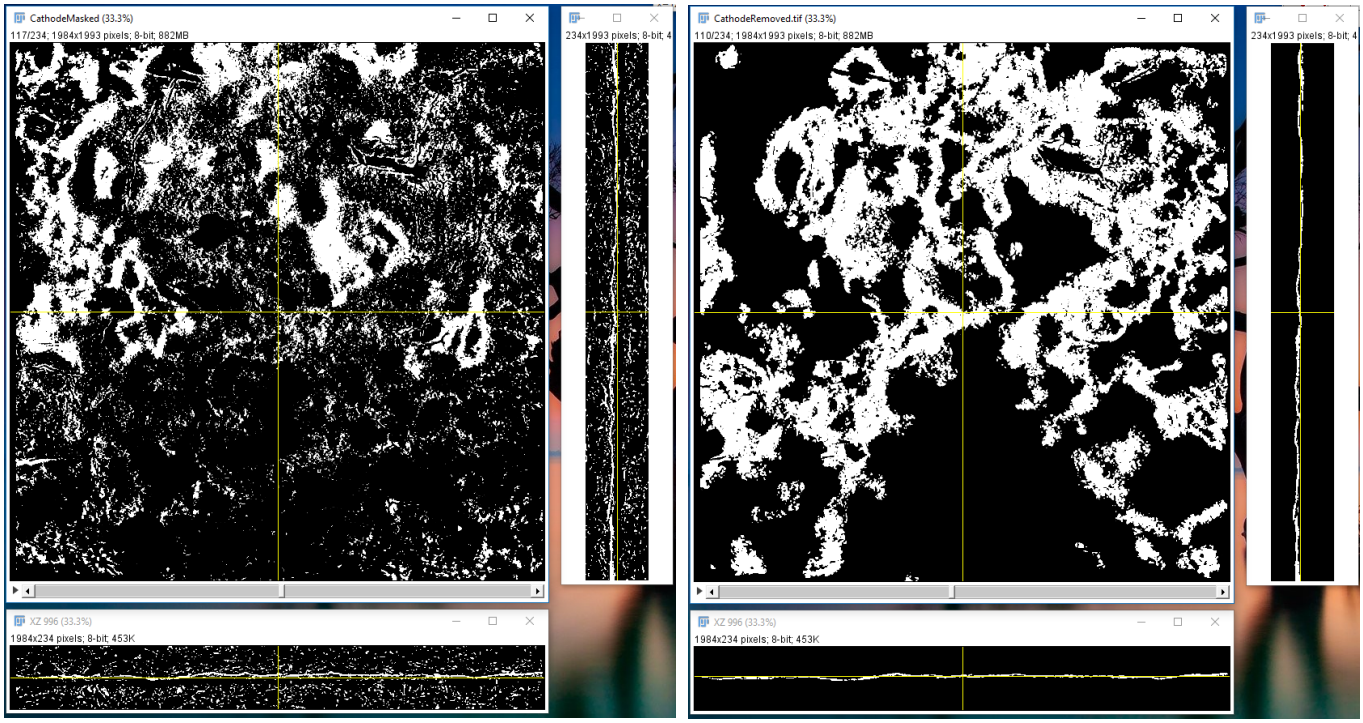
Next, change the GS_image_name in the text editor to the name of your open greyscale image. To help remove the chance of errors, if the current name of the greyscale image has spaces or .tif in the name, rename this image by right-clicking on the image and selecting Rename. (Above, my image is Reslice of 2000AMST_crop_median_GS, I therefore changed it to just 2000AMST_crop_median_GS). The binary image name should be AnodeRemoved.tif (if saved) and shouldn't be changed as MemSeg will look for this file name later.



Once the image name has been updated, hit Run; this will run the macro and display several images, but after finishing there will be a new image called CathodeMasked (shown below). As you can see the cathode catalyst is now black.



This is now the image stack which you can threshold and segment out the anode catalyst layer, following the same procedure as in the Cathode Separator tutorial. You may find it useful to save/duplicate this stack. First, threshold for the anode, as best as we can. The anode is usually more difficult to isolate. What is important, is that it forms a 'good' boundary, even if there are some gaps. MemSeg performs interpolation to try to get around that. Then run AnodeSeparator. Be sure if you did duplicate CathodeMasked, do not use the image name ending in -1 as described in the Cathode Separator tutorial. Rename it if necessary. You will most likely have to use a fairly high threshold value following gaussian blurring. In this case, I used a value of 162. It is ok for the anode to be sparse.



Left, following applying a threshold to CathodeMasked. Right, after running AnodeSeparator. Be sure to save the above image as CathodeRemoved.tif.

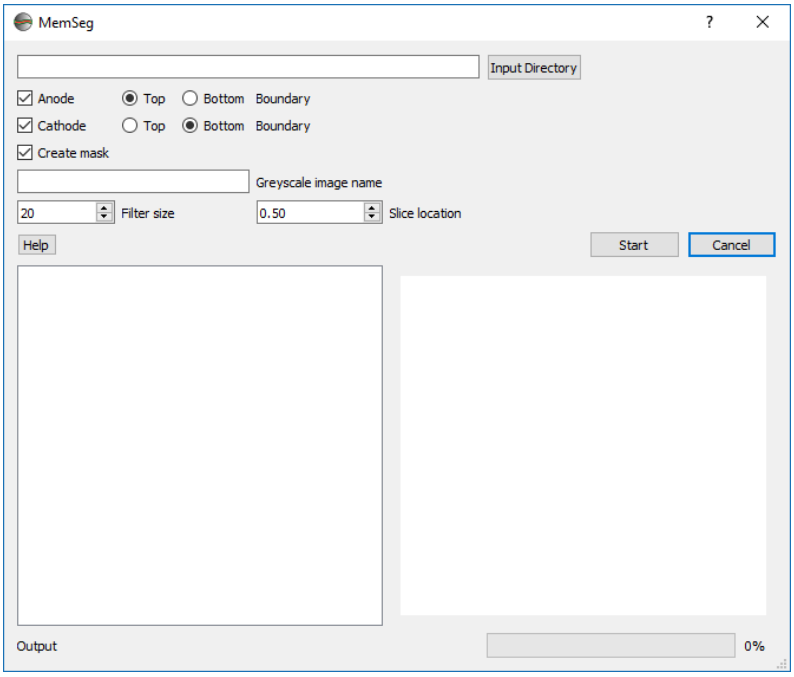
Note: You should now have 3 image stacks, <sample>_GS.tif, AnodeRemoved.tif, and CathodeRemoved.tif. It is imperative that these are in the <sample>_data folder, and also that they are all in the through-plane direction (top-down). Other files can be in this folder, but these three must be in there.

We are now ready to run MemSeg!

MemSeg

As discussed in the installation instructions, to start MemSeg you must be in the virtual environment, initiated by **activate MemSeg** from the command prompt, as well as in the MemSeg directory. From the command prompt, now type **python main.py**. Please see the installation instructions if you need further assistance.

You should now see the main window



Input Directory: Folder containing the data of interest with the 3 image stacks from above.

Anode, Cathode: MemSeg allows the user to only obtain one boundary if they wish. To perform full membrane segmentation, both Anode and Cathode must be selected.

Boundary: The boundary with which each corresponds. If Anode will be the top boundary (of the membrane), then Top is selected, Cathode will be bottom – or vice-versa

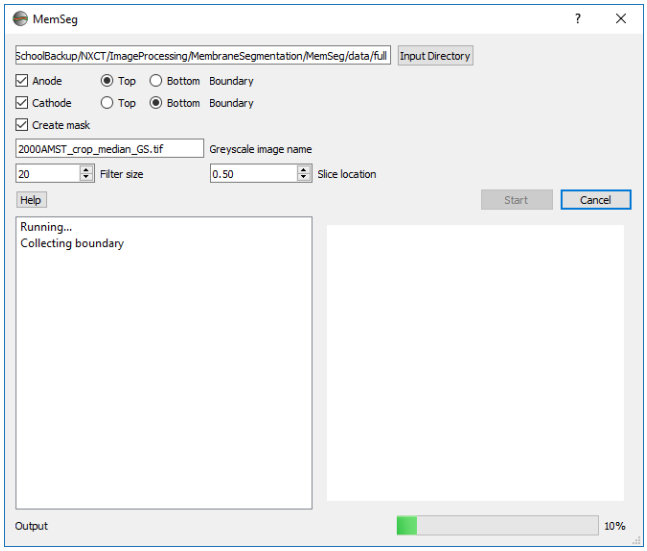
Create mask: If a greyscale mask is wanted by the user, otherwise just a binary image stack and thickness image will be output.

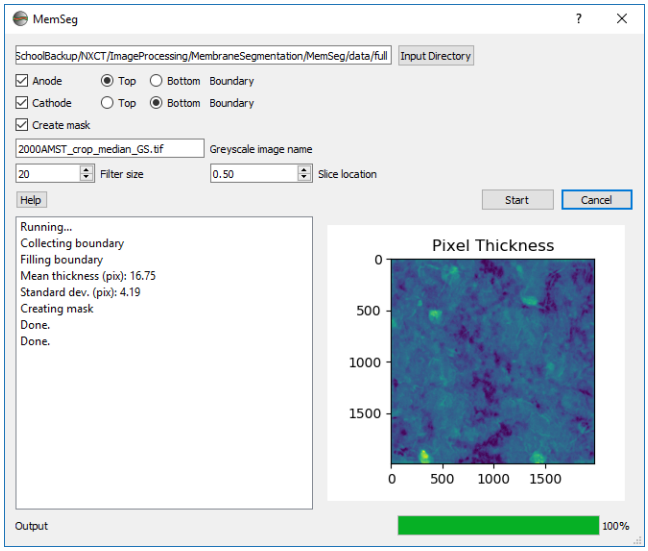
Greyscale image name: If the naming convention and image location has been followed as described above with the original greyscale image stack ending in GS.tif, this will automatically populate. Otherwise type in the name of the image stack. It must still be in the same folder as the input directory however.

Filter size: An option for the smoothing factor, default = 20

Slice location: If Create mask is selected, this will output a 2D image at the fraction between top and bottom boundaries through the membrane. This follows the undulation of the membrane, unlike a single stack slice.

Once started with the necessary options you should have something as below.



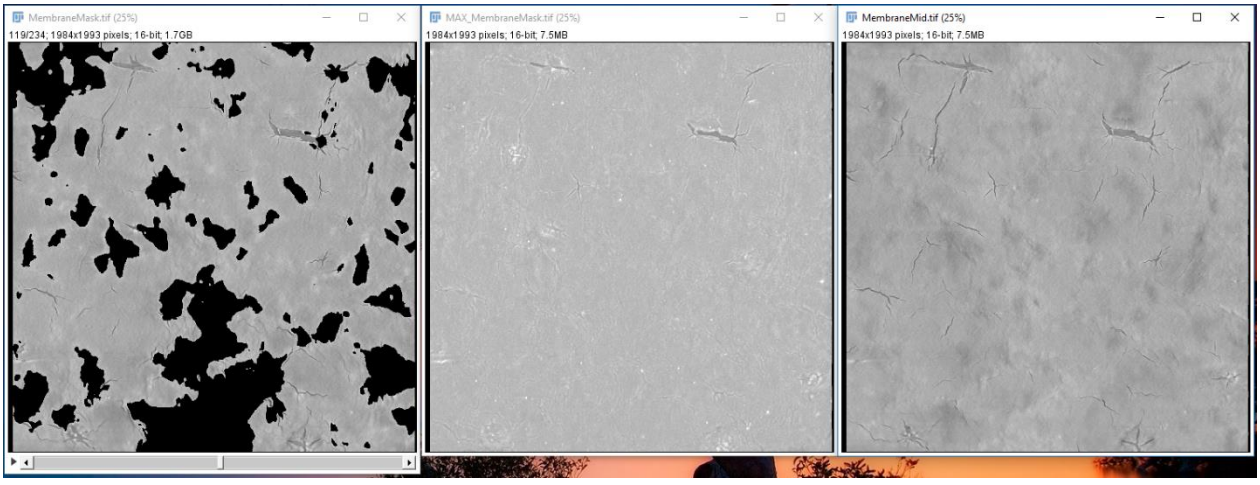


Full stack (about 1.5Gb in size) will take about 3 minutes to perform the full processing, including mask of greyscale values. If an error occurs, the user should be prompted, or the output can be seen in the terminal for debugging.

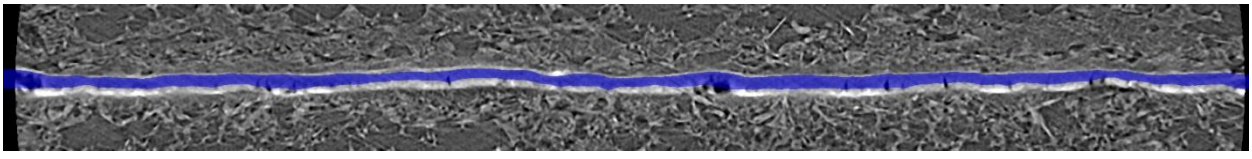
The output images will be in the same directory as the input.

Dynamic slice location

Shown below are the comparisons of taking a single stack slice through the masked membrane, the max project of the masked membrane, and the dynamic slice output image from MemSeg (left to right, respectively).



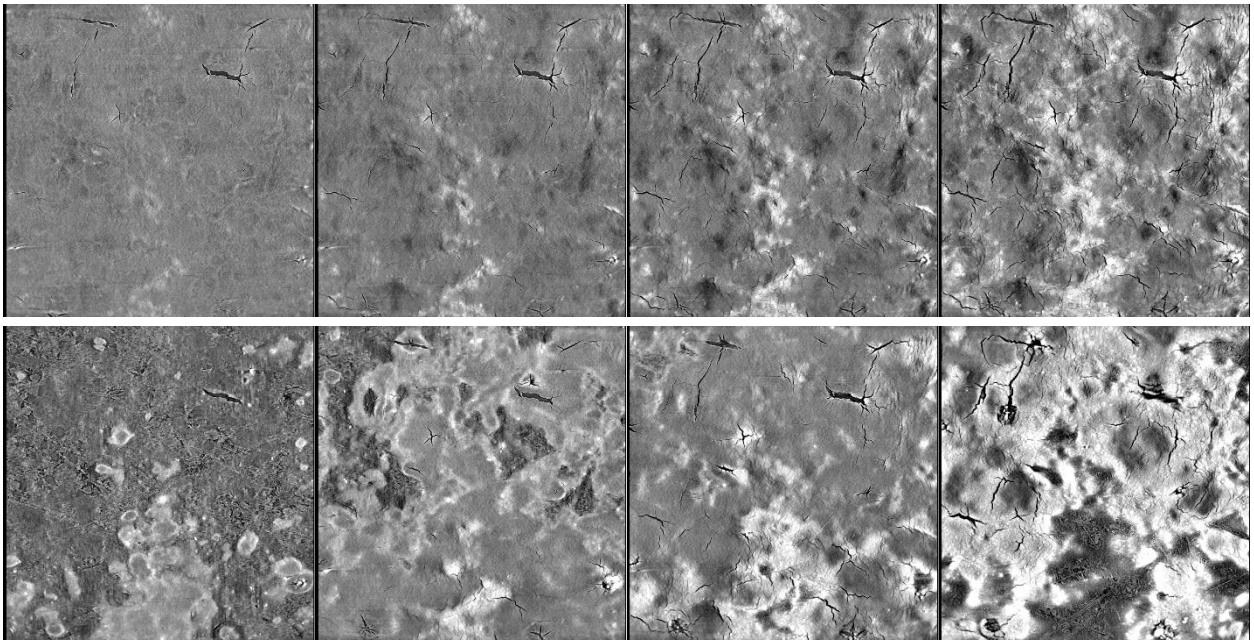
Binary stack overlay



Update

V0.1.1 – 15/02/2018

An option to perform dynamic slice through the full region between boundaries, following the contours and undulations has been added. A single slice will still be output provided 'create mask' is selected as well. The full dynamic slice will create a stack which falls in the range of thickness given by the user. For example: 0.2, 0.6 will start the dynamic slice at a location that is 0.2*local_thickness away from the top boundary and extend to a location that is 0.6*local_thickness away, taking the specified number of steps to get there. If number of steps is left as 0, the maximum thickness number of steps is taken. The images below show the comparison between dynamic slice (top), and rigid slice (bottom). As can be seen, the full membrane is constantly viewed throughout the images for dynamic slicing but for rigid slicing only portions of the membrane can be captured in each image due to the wavey nature of the membrane.



Disclaimers

- Accuracy is only as good as the catalyst layer segmentation.

- Minimal error may be introduced from the interpolation step. Check _interp_boundary.tiff images to compare against greyscale image. Adjustments to the filter size may help.

- Preprocessing on the original greyscale image to improve catalyst layer segmentation have not been investigated fully yet, however may improve some accuracy in results.

- With some practice, a full workflow can take ~30 minutes; from initially loading the raw tiff stack from XCT to final output with cathode layers and membrane fully segmented. The catalyst segmentation is the longest step. Once the boundaries are obtained, MemSeg will only take ~4 minutes to complete the process.

Possible future improvements

- In the current version there is no option to load previously obtained interp_boundary to skip a step if the user just wants to re-do only part of the process.

- MemSeg will automatically overwrite images in the same folder, a prompt could be added if the user wants to not have this happen

- Error reporting improvement

- Closed package as .exe

- Combined segmentation and crack analysis