## Overview

Neuronal self avoidance is a feature that prevents the neuron from fusing its parts with itself. In particular, dendrites avoid “crossovers” where a dendrite reconnects with another dendrite within the same neuron. Studies of this behavior require quantification of this phenomenon through counting the total number of dendrite crossovers which requires extraneous manual labor, as neurons exist in a 3D space and analysis of the whole space is required in order to accurately assess whether there exists dendrite crossover at a particular point. So far, ImageJ’s ability to navigate layered images and count markers have been used to count the number of dendrite crossovers on a neuron, however an application made specifically to facilitate this process has not been created yet. I sought to create such an application using the MATLAB platform that helps navigate the user through each potential dendrite crossover point and allows them to determine whether crossovers exist.

The approach of this application is to work with a 2D projection of the 3D space. It will detect any areas on the image closed off by lines that exist, navigate the user through each area, allow them to determine whether crossovers exist around these areas, and count/extract crossover points on the neuron. The reasoning behind this approach is that: when two lines intersect at two or more points, areas of the image will be isolated from the rest of the image. Due to dendrites all originating from the same point (soma), for every crossover point there exists some arbitrary two lines that can be drawn such that the two lines intersect at more than one point. One of these points will most likely be the product of dendrite branching where the dendrite branches from one branch to multiple, therefore the vast majority of areas closed off on the image will result in one potential crossover point and one branching point.

The efficacy of this application still needs to be determined by comparing the dendrite crossover counts obtained through manually counting using imageJ and using this application.

## Required Components

Tiffread: [https://www.mathworks.com/matlabcentral/fileexchange/60037-tiffread](https://www.mathworks.com/matlabcentral/fileexchange/60037-tiffread%5C)

Image Processing Toolbox

Mapping Toolbox

## Features

User input for names of files that will be used: stack file name and projection file name:

* Stack file name is the name of the tiff file z-stack image. Only use 8-bit tiff files, and include the file name extension in the textbox.
* Projection file name is the name of the projection image file. Only use conventional image types, png recommended. Include the file name extension.
* “LOAD” button loads the two files according to the text in the text boxes

FullViewImage:

* The large image on the left side, shows the full view of the neuron with different colors signifying different closed off areas.

Thresholding values:

* Brightness threshold: minimum pixel brightness required for the pixel to be registered for analysis.
* Pixel area threshold: minimum area required to be counted and considered for dendrite crossovers.
* “Update” button: update the image analysis program with the new thresholding parameters.

Crossover?

* Yes and No buttons: tells the application that the area we are currently looking at is or is not an area.
* +1 or -1 buttons: for areas that have more than one crossover or if a mistake was made, these buttons compensate for these scenarios.

TotalCrossoverCounter:

* Shows how many potential crossovers there are currently.

Specific view images on the right:

* Colored: shows the currently selected area in the colored view.
* Original: shows the currently selected area as it was loaded into the application (black and white).
* “Specify Area” button: allows for user input of a selection of a specific area on the colored image to inspect, shows up to 12 images on the bottom of the selected area.
* “Play Video” button: plays a video using the Original image area that iterates through all of the layers of the stacked image of the area that is currently selected.

Specified area images:

* 12 images that show the different depths of an area specified using the “Specify Area” button.

After inspecting all areas, a message will appear showing the total number of crossovers counted. An output excel file will also be generated in the currently active folder that shows the coordinates of the center of all areas that had one or more crossovers on them.

## Instructions

1. Image processing using imageJ (projection image)
   1. Clear out foreign dendrites/neurons/objects from the image
   2. Manually count dendrite crossovers at areas of the image that have overlaps with other neurons and remove that part of the image (clear such that that part of the image is black)
   3. Save the image as a format that the application can use
2. (stack image)
   1. Change data type to 8-bit and save as tiff file
   2. Split color channels such that only the one with dendrites is saved
3. Input images into application
   1. Fill out

## Notes

* \*\*\*BOUNDS ERROR: inspect manually and move on, will be fixed soon
* Currently, the first version of the application supports up to 12 images for the stacked image, if more is needed then the “specify area” feature is unusable, but the video will still work.
* Future versions/alternative versions could support more than 12 images.
* Different screen sizes seem to disrupt the formatting of the application, different versions of the application can potentially be made to accommodate for this.
* Only 8-bit tiff images work with tiffread. The data type of tiff images can be changed using imageJ.
* When inspecting an area, especially large areas, it is imperative that the user does not dismiss the area after finding evidence of lack of crossover, but rather consider all of the possible crossover points that might exist on that area.
* It would be beneficial to have a simple, fixed criteria in mind when inspecting a neuron, for example: as long as a potential crossover point isn’t a dendrite branching point and the two lines in question exists with maximum brightness at that point in the same depth then that can be classified as a crossover point. As long as these criteria stay consistent, results should stay consistent as well.
* It is better to manually count crossover points at locations where dendrites of our target neuron overlaps with neighboring neurons and remove that part from the projection image completely than try to work with those areas in the application.
* Current thresholding settings were set with 1024 x 1024 pixels images in mind. If higher or lower resolution images are used, thresholding settings should be changed.
* Due to the layout of the application, square images would work best.
* If, after loading the application, the application doesn’t fit on the screen properly, click on the edge of the application window and move it manually so that it fits.
* Don’t “completely” threshold the image (completely binarize the image) on imageJ, as it can be difficult to work with in MATLAB afterwards. Use the thresholding settings in matlab as it gives the option to view the original projection image.
* The assumption this project is based on only holds when the lines are complete and does not break midway through.