Intro & Overview:

NOTE: Please please please read the entire supplement of the the paper so that you are aware of the potential artifacts that can occur in this type of analysis and how to control / correct for them!

There are two primary ways you can go about using this code, each with drawbacks and advantages:

A. Run the full processing pipeline as used in the paper.

   Pros: Less matlab knowledge needed.
   Cons: Lots! This was written in a fairly problem-specific manner, and has lots of strict requirements on image data format, type and quality, and necessary software etc. (requirements detailed below). It's only been tested one the type of data in the paper and it's not all too likely it will work without adjustment on new data.

B. Use the individual algorithm components in your own pipeline

   Pros: much more flexibility, much more likely to actually work on your data.
   Cons: need to code your own pipeline.

Instructions for using both of these approaches are found below in their respective sections. In both cases you should first look at the setup section. Also, since the results will likely differ for your data there are instructions for visualizing the results from each approach to check if everything worked.

General Setup:

1. Download the code zip file here.
2. Unzip the code, preferably in the matlab home directory (the directory matlab goes to when it starts up).
3. Add the code folder and sub-folders to the path:
   a. Type "pathtool" into the matlab prompt and press enter.
   b. In the window that comes up click the "Add with subfolders…" and select the folder that you unzipped the code to.

Requirements (READ ALL THESE!!!!):

- Data format: To use the full pipeline the images must be .STK files, one file per-channel per-time-point, with each channel in it's own folder and all channels folders in a directory

named “images”. See the example data if this is confusing. Yeah, I know, I told you there were some pretty specific requirements!!

- Hardware: For the example data you’ll need about 10GB of free RAM. Obviously, other datasets may require more or less.
- Matlab: This has been tested with matlab versions from up to r2015a. It was developed on r2012b-r2014a but I haven’t gone back and thoroughly tested on these older versions.
    - Matlab toolboxes: You’ll need these matlab toolboxes to run the full pipeline:
        - image processing
        - bioinformatics
        - curve fitting
        - optimization
        - statistics
- OS: Has been tested under windows 7 and ubuntu 12.04. Probably won’t run under Mac OSX
- Imaris: If you want to be able to check if the segmentation actually worked (certainly not a given with new data) by overlaying it on the images, or if you want to visualize the skeletonization etrc. you’ll need Imaris. Specifically, for the skeletonizatoin and surface mesh visualization you will need version 7.4 or *earlier.* For visualizing the cortical intensity and curvature samples in voxel space you can use later versions (tested with up through 7.7).
    - Imaris requirements: You will need to have the ImarisXT module to connect to matlab.

NOTE: These requirements are necessary to run the full processing pipeline (method A in the Intro above). To Use just the individual components (method B in intro above) the requirements will vary.

Running the full processing pipeline:

1. Download the sample data here.
2. Open the example script by typing “edit runExampleDatasets.m” at the matlab prompt and then pressing enter.
3. Look at the script. This has all the basic commands you’ll need to run the processing pipeline. You can copy-paste these into your own script if you want to change the settings etc.
4. Run the script by typing “runExampleDatasets.m” at the matlab prompt and pressing enter. Wait a while - if it finishes and no errors pop up (warnings are OK) then you can proceed to visualization.

NOTE: The function analyzeMOvieMaskedIntensities currently only displays figures using the local maximum intensity, so is not the same as in the paper. To get the type of figures used in the paper you will want to combine multiple cells using postProcess3DMovieArrayMaskedIntensities as described below.

Visualizing results from full pipeline in Imaris:

NOTE: See requirements above!

1. Be sure you have already run the processing pipeline on the dataset you want to visualize
2. Open up the MovieData object describing the dataset to visualize. Change the matlab directory to the folder containing the movieData.mat file and then run:

MD = MovieData.load('movieData.mat',0);

3. To view the segmentation, surface mesh and skeletonization you will need to have Imaris 7.4 or earlier installed (and NOT have any later versions installed. Yes, i know it's a pain in the ass...), then run:

viewMovieImaris(MD)

4. To visualize the surface curvature samples and intensity samples in voxel space (NOTE: this can be done with later versions of imaris, but requires that the analyzeMovieMaskedIntensities analysis have been run), type this in matlab and hit enter:

viewMovieCurvatureImaris(MD)

Running combined analysis on multiple datasets:

NOTE: This is a more advanced step and these instructions are pretty rough currently. Sorry, please add comments where confusion occurs.

1. Create an array of MovieData objects (each MovieData object describes a single dataset) using the setupMovieArray:

MA = setupMovieArray;

2. Specify a directory to save the results to (if you run multiple combined analyses you will want to run each with a different output directory):

p.OutputDirectory = NAME OF SOME DIRECTORY TO SAVE FIGURES TO

3. Pass this array to one of the combined processing functions. These functions are all named something like “postProcess3DMovieArray ….” . They’re also all pretty poorly documented and sloppily coded. One example is:

postProcess3DMovieArrayMaskedIntensities(MA,p)

Using individual algorithm components:

Luckily, all of this sloppy code was written in a way that separates the input/output from the core algorithms, so you can use these as you see fit in your own analysis pipeline. All of these components are in the code directory, but I list some of interest with notes below.

NOTE: Most of these functions actually have some amount of documentation in the header, so check that for more information.

Components of interest:

- Segmentation:
    - Core segmentation algorithm: “huntersFancySegmentation3.m”
        - Yes, that’s the actual name of it. This takes in a 3D image matrix and returns a binary mask matrix of the same size. It probably won’t work on your data.
    - Segmentation post-processing: “postProcess3DMask.m”
        - This fixes minor errors in the segmentation produced by the core segmentation function. Also pretty specific to my data, but can be used a s a model for something that actually works for yours.
- Skeletonization:
    - Skeletonization algorithm: “skeleton3D”
        - This is a .m wrapper for a c function I ported into a mex file, from the publication cited in the original paper. Takes in a 3D binary mask matri and returns a 3D binary skeleton matrix.
    - Skeleton to graph conversion: “skel2graph.”

        - This converts the 3D binary skeleton matrix into a graph, described by vertices connected by edges.
- Surface mesh creation & curvature calc:
    - Basic mesh creation: “isosurface.m” (matlab built-in function)
    - Mesh smoothing and curvature calc: “analyze3DMaskGeometry.m”
        - Takes in a 3D mask matrix and returns a bunch of stuff, including the surface mesh and the curvature at each face on this mesh. Including, importantly, the locally averaged surface curvatures which are much less noisy.
            - NOTE: You will want to make your mask voxels symmetric before using this function, with the “make3DImageVoxelsSymmetric” function
            - NOTE: that while the averaging radius in the full pipeline is specified in nanometers, in this function it is specified in pixels.
- Intensity & curvature correlation:
    - core function: “analyze3DImageMaskedIntensities”
        - Takes in a bunch of stuff (images, mask, skeleton graph, mask properties etc) and outputs a structure with various correlations between geometryand intensity within the images.
            - NOTE: again you will want to make the voxels in your image isotropic using the “make3DImageVoxelsSymmetric” function before calling this function.

Visualizing results from individual algorithm components:

While it’s always good to overlay your results on the raw image data to get some subjective sense of whether the analysis is working, these functions will let you visualize individual components from the analysis within matlab without needing imaris.

- Masks/Segmentation:
    - show3DMask - view a 3D binary mask matrix as a surface mesh
    - spy3d - view a 3D binary mask matrix as a 3d plot *a la* spy.m
    - NOTE: you can also view an image in 3D using viewStack but it’s REALLY slow . It’s one alternative to using imaris to check the segmentation though.
- Skeletonization:
    - plotSkel - view a skeleton graph as a 3D plot
    - spy3d - view a 3D binary skeleton matrix as a 3D plot
- Surface curvature / meshing:
    - showMaskSurfaceProp - shows a 3D mesh mask surface color-coded with the local curvature:
        - NOTE: add “LA” in front of each curvature to view the (less noisy) locally averaged curavatures, e.g. showMaskSurfaceProp(maskProp,’LAmean’)