Instructions for using imageJ macros and MATLAB code from:

Whitelaw et al. 2020, eNeuro

For live imaging pre-processing and analysis of 4D image stacks

Questions? Problems? Comments? Suggestions? Email me at **Brendan.whitelaw@gmail.com**

-Brendan Whitelaw, Majewska Lab, University of Rochester Neuroscience

**Scripts for Live Imaging analysis**

Pre-processing 4D images in FIJI/ImageJ: scripts for batch processing

* Each asks for input folder, output folder, file extension, and options.
* Processes each image in folder (and subfolders) and saves processed image another folder

3Dcorrect\_fixedZ.ijm:

* Input: directory containing raw 4D (or 5D) TIFF stacks
* Output: directory to save 3D drift-corrected images. Saves as same name with ‘\_3Dcorrect’ appended.
* Main: applies Correct 3D Drift plugin to each image
* Optional:
  + Despeckle (median filter)
  + Guassian smoothing

MaxZproj\_batch.ijm:

* Input: directory containing drift-corrected 4D stacks (i.e. output of 3Dcorrect\_fixedZ.ijm)
* Output: directory for maximum Z-projection images (‘xyt’ movies). Saves with ‘\_maxZ[lower]to[upper]’ appended to old file name
* Main: maximum Z projection of slices manually inputted by the user
  + For each image, the original stack is displayed. The use scrolls through, decides which slices to include, and types in the top and bottom slices for the projection
* Optional:
  + Despeckle (median filter)
  + Guassian smoothing

processing\_timeseries.ijm:

* Input: directory containing time-series image stack (‘xyt’; i.e. output of MaxZproj\_batch.ijm)
* Output: directory for processed time series images. Saves with ‘\_processed’ addended to old file name.
* Optional:
  + MultiStackReg: an additional registration in 2D, which can be helpful if there is still slight movement in the image field
  + Autocrop: uses the ‘Select Bounding Box’ to crop stack such that any blank space from the drift correction is eliminated
  + Trim Edges: removes 10-pixel thickness edges from all sides
  + Bleach correction (histogram method): corrects for photobleaching using histogram method.

Matlab Scripts for Running velocity and convergence analysis:

* *Add the folder containing the functions to your path in MATLAB*
* **Important:** the function TIFFStack has some issues when sharing. Try deleting it and re-downloading it straight from:
* [*https://www.mathworks.com/matlabcentral/fileexchange/32025-dylanmuir-tiffstack*](https://www.mathworks.com/matlabcentral/fileexchange/32025-dylanmuir-tiffstack)
* **Directional\_main\_postthresh.m** will call all the other necessary functions to analyze the data.
* **Open this script, and hit RUN to get started!!!**
* Requires ‘Image Processing Toolbox’ and ‘Computer Vision Toolbox’; Run with MATLAB 2019a (I don’t know if earlier versions will work)
* Analysis is divided into 5 separate steps (labeled 0-4)
* Look for old files will determine if the (a) images have been loaded, (b) ROIs selected, (c) and the optic flow vectors calculated; so they don’t have to be redone when tweaking analysis.
* Re-analyze vectors: if changing any of the analysis parameters, keep this checked so that it updates the files
* After each step in analysis, the resulting data (e.g. thresholded image as an array in MATLAB) is saved as a MATLAB variable in a separate file for each image. More variables are added to this results file with each step.

1. Loading ‘xyt’ image stack and thresholding image
   1. If Manual threshold is selected, a window with a sliding bar will pop up.
      1. Uses ‘thresh\_tool’ function (google that and mathworks for info)
   2. If not, the next step will ask you to select a folder with the thresholded images. These should have similar names to the original image (i.e. should be in the same alphabetical order, since that is how matlab orders them when it imports them)
      1. Don’t necessarily have to be logical pixels. Uses MATLAB imbinarize function if not already logical; theoretically just needs to be very obvious what is signal vs. background.
2. ROI Selection: follow the instructions printed in command window
   1. For pipet experiment: select the tip of the pipet; then outline the pipet to exclude from analysis.
   2. For laser injury: outline the auto-fluorescent core at the center of the injury
   3. Note: if you want to use a different time point for selecting ROIs, go to line 44 of ‘ROIselection.m’:
      1. 44 > (imshow(stack(:,:,1),[0 0.3]) 🡪 change the ‘1’ to a different time point
3. generate\_flowvectors: run optic flow on movies to generate velocity vectors.
   1. Also transform velocity vector array so that it’s in reference to the central point: from (dx, dy) to (towards, orthogonal) velocities
   2. Uses a rotation transformation matrix
   3. No user input needed
   4. NOTE: Generates **BIG** file sizes
4. analyze\_flowvectors: calculates average directional velocity and convergence
   1. Restrict analysis to circle: if unchecked, run analysis on entire image. If checked, will restrict analysis to a donut region between circles of radii (in pixels) inputted
      1. Will always exclude the ROIs drawn previously)
   2. Convergence thickness: thickness of the ‘donut’ shaped ROI directly around the injury (laser injury) or the radius of the circle around the pipet tip (pipet experiments) to use for convergence analysis
   3. Mask is applied using the thresholded microglial images to refine analysis to only components of microglia (vs. ‘blank’ space)
5. quiver\_flowvectors: generates color-coded quiver plots using flow vectors
   1. Downsample: because velocity vectors are calculated for every pixel, it does not make sense to plot every single one.
   2. Saves .tif movie of grayscale image with velocity vectors overlaid, color coded (green = net towards; red = net away);

Example Settings window:

Graphical user interface, application

Description automatically generated

* Outputs of program:
  + New directories:
    - ‘flow\_results’: For each movie: ‘results’ file (.mat) containing important intermediate matlab variables created throughout the program. Useful when re-analyzing to save time. BIG files though
    - ‘quiver’: contains quiver plot movies for each file
  + ‘all\_results-[DATE].mat’ file: **most important/useful** containing relevant outputs as workspace variables;
    - input\_names: list of initial file names; this is the order in which data is output in the other folders; each ‘n’, or sample/video
    - settings: choices you used previously. Especially important if you’re analyzing different distances
    - velocity\_avg\_mat: ‘t’ by ‘n’ matrix: the average directional velocity at each time point, for each sample.
      * NOTE: first time point is meaningless for average velocity, because the optic flow calculation compares this image to a ‘blank’ image as the previous time point; ‘t’ is number of time points/image frames
      * Normalized to the amount of microglial pixels in the ROI
    - con\_size\_mat: ‘1’ by ‘n’ matrix: size of area used to normalize amount of convergence
    - con\_frac\_mat: ‘t’ by ‘n’ matrix: stands for convergence fraction matrix: fraction of ROI occupied by microglial processes (based on thresholded image) at a given time point
    - con\_frac\_mat\_adj: ‘t’ by ‘n’ matrix: con\_frac\_mat substracted by the value at the first time point to account for any debris or processes initially in the ROI.

Motility\_Analysis\_batch\_bsw: for running motility analysis on xyt .tif movies

* Much simpler than velocity analysis
* Easy 2-step process:
  + Manual thresholding: similar interface as velocity; uses thresh\_tool also
  + Pixel-based motility analysis
* Output:
  + New directories:
    - ‘Thresholded\_images’: tif files containing thersholded movies
    - ‘Thresholded\_stacks’: .mat files containing matlab variable, an array representing thresholded movies
  + mot\_results-[DATE].mat:
    - input\_names: list of initial file names; this is the order in which data is output in the other folders; each ‘n’, or sample/video
    - motility\_index: ‘t’ by ‘n’ matrix of motility index
    - motility\_avg: ‘n’-size vector containing the average motility index across timepoints for each sample
    - coverage: ‘n’-size vector containing the coverage/surveillance for each sample