CSE 554A Project Report - **Neuron Segmentation in 2p FLIM**

Sangwook Suh (455105)
sangwooksuh@wustl.edu

1. Accomplished Features:
    a. Segmentation tool that outputs segmented neurons as a stack of binary masks, given path to data and number of neurons
    b. Evaluation tool for quantitative analysis that outputs average intersection over union (IOU)
    c. Everything is a fully automated command line tool without GUI
2. Implemented Wishlist Features:
    a. Use data from both intensity and lifetime videos to increase solution quality
    b. Give user control over threshold value
    c. Add evaluation as an option to segmentation tool
3. Core Algorithm
    a. Preprocessing:
        i. Load all data as 2D arrays
        ii. For each frame:
            1. Remove borders (hard-coded to given data)
            2. Separate into 5 channels so that each pixel has one value:
                a. Intensity
                b. Lifetime - Red
                c. Lifetime - Green
                d. Lifetime - Blue
                e. Lifetime - average
            3. Scale data in frame to zero mean and unit variance
            4. Threshold (default 1 if no user input)
    b. Get Solutions:
        i. For each frame
            1. Get the union of pairwise intersections of all 5 channels:
                a. Equivalent to selecting all pixels that are ‘True’ in at least 2 of the 5 given channels.
            2. Open (r=2) & Close (r=1) the result from step b. i. 1. (r ← radius)
            3. Get n largest components (n ← number of neurons)
            4. Fill holes (invert, get largest component: background, invert)
            5. Close (r=1) & Dilate (r=1); r ← radius
            6. Return result
    c. Save result in ‘solutions’
    d. If `-e`, `--evaluate` options selected, load ‘ground truth’ data and calculate average IOU

4. Significant Implementations & Work
    a. Implemented morphology operations (erode, dilate, open, close) with ‘square 3x3’ struct with basic NumPy
    b. Implemented connectivity algorithms (flood, label connected components, get n largest connected components) with basic NumPy
    c. Developed core algorithm & evaluation tool
5. Quantitative Analysis:
    a. Results (from presentation slides):

Threshold	EASY	MEDIUM	HARD
0.25	0.326	0.740	0.674
0.5	0.332	0.764	0.722
0.75	0.777	0.785	0.644
1	0.785	0.801	0.540
1.1	0.779	0.806	0.416
1.2	0.761	0.814	0.187

    b. Takeaways:
        i. Overall, not amazing but reasonable results
        ii. ‘Medium’ problem was the easiest: spherical neuron shape was easier to segment compared to irregular shaped neurons in ‘Easy’ and ‘Hard’ problems
        iii. Mostly difficulties with removing ‘tail’ portion of the neurons
            1. Tail can be removed with heavy use of morphological operations but this approach scales badly with non-spherical neurons
        iv. For certain problems, specific channels have high noise
            1. Giving user control to only use a subset of channels instead of all data may improve tool’s flexibility, but this has not been implemented.

6. Future Work
    a. Give user more control
        i. Only a user specified subset of channels

    ii. Weight certain channels more than others
    iii. Set different thresholds for different channels
  b. Source code documentations for functions
  c. Handle bad input
  d. Increase usability
    i. Option to save evaluation data on a per frame basis as .csv or .xlsx
    ii. Option to make a detailed report identifying outlier frames based on evaluation
  e. Fix known issues & bugs below
7. Known Issues & Bugs
  a. Program tested on Mac OS Sonoma 14.1.1, cannot guarantee results on other environments.
  b. Recursion depth
    i. Typically, Mac Terminal has a recursion depth limit of 1000, which leads to a recursion depth limit reached error with the 'flood' function. On Jupyter notebook, this limit is typically set as 3000, which is sufficient for the image provided (160x160). But for higher resolution images, the 'flood' algorithm may hit the recursion limit and produce an error.
    ii. To mitigate this bug, the program sets the system's recursion limit to 1,000,000 if it is lower than 1,000,000.
  c. Crash when threshold too high
    i. If a given threshold is so high that there is no 'object' component after thresholding in a given frame, the program crashes because at some point the 2D arrays become flattened, possibly due to some nuances in NumPy default behavior with differing inputs on things like bitwise and logical operations. There was not sufficient time to properly identify and address this bug. With given data, thresholds of 1.2 or lower seemed to be safe.
  d. Border cropping hard-coded
    i. Assumes the image has a border exactly as the given data, such that the image is contained in rows 40-199 and columns 114-273. The 'crop(img)' function needs to be modified for data with different dimensions or borders than the given data.
8. Dependencies
  a. Submission includes environment.yml file
  b. Python 3.11.5:
    i. Modules: os, sys, argparse, itertools
  c. Packages:
    i. NumPy 1.26.0
    ii. pillow 10.0.1
    iii. natsort 7.1.1