Instructions on How to Use GliaTrace Toolbox

by Mahmoud Abdolhoseini
mahmoud.abdolhoseini@newcastle.edu.au

Hunter Medical Research Institute
NSW 2305, Australia

February 2019

Contents

1 Introduction

GliaTrace is a toolbox developed by Mahmoud Abdolhoseini via MATLAB as part of his PhD study at the University of Newcastle. The toolbox is design to perform several image analysis techniques such as segmentation, reconstruction, and quantification on microscopy images including neurons, microglial cells, etc. and model the cell micro-environment as well. This documentation provides the step by step guide on how to apply the toolbox to generate and process the microscopy images. I do appreciate any feedback on this document which makes it easy and clear to follow.

First, some general guidelines about MATLAB features and how to use them are expressed in this section. Then the capabilities of our toolbox are elaborated in the following three main sections: 1- neuron image synthesizer, 2- nucleus segmentation, and 3- cell reconstruction.

MATLAB is a commercial tool founded in 1984. A wide range of toolboxes are combined inside MATLAB to perform engineering and mathematical tasks in a matrix-based environment. MATLAB has its own programming language with straightforward syntax offering a friendly environment to users to develop their own toolboxes and implement their ideas.

A general view of MATLAB development environment (MDE)—2017b—is depicted in Fig. 1. As we can see, there are four distinct sub-windows included in MDE. Depending on different configuration, they might appear in different areas. Here is a brief explanation about them: *'Current Folder'* shows the files and folders that exist in the current directory, *'Command Windows'* is an environment to execute all sorts of commands such as mathematical operations, predefined functions and classes, *'Workspace'* shows all the variables assigned values by user or during execution of an operation, and *'Editor'* contains m-files (MATLAB executable file formats containing commands, functions, classes, etc.) opened in separate tabs.

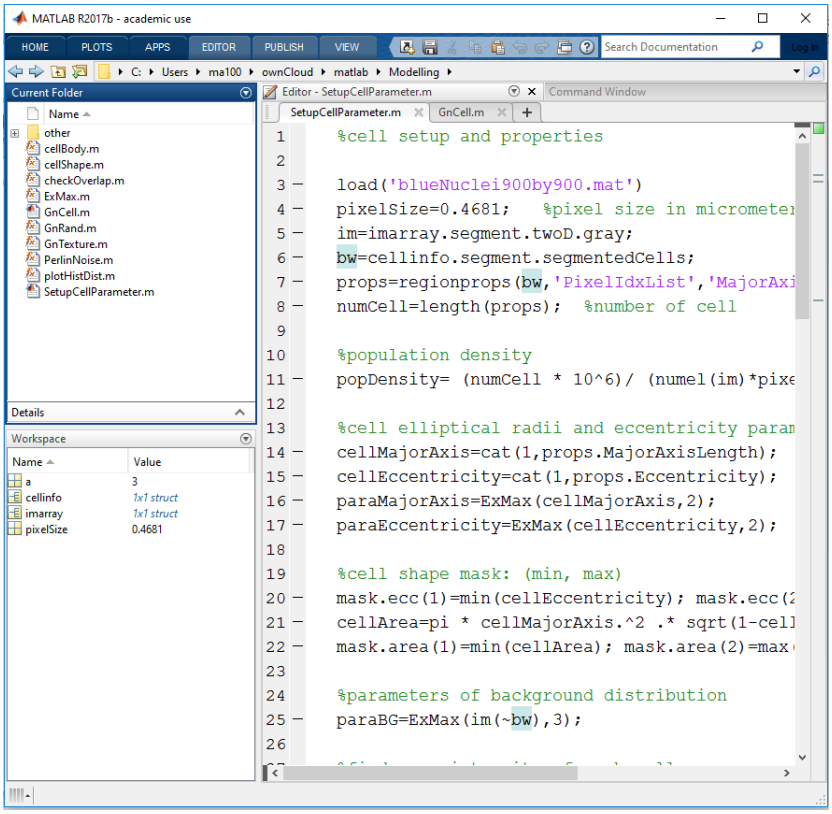


Figure 1: A general view of MATLAB development environment.

2 Neuron Image Synthesizer

In this section, first our motivation for synthesizing neuron images is shortly discussed, then the structure of the algorithms and the code are explained in details making them clear for the user to apply. This section is organized as follows: motivation, algorithms, and application.

2.1 Motivation

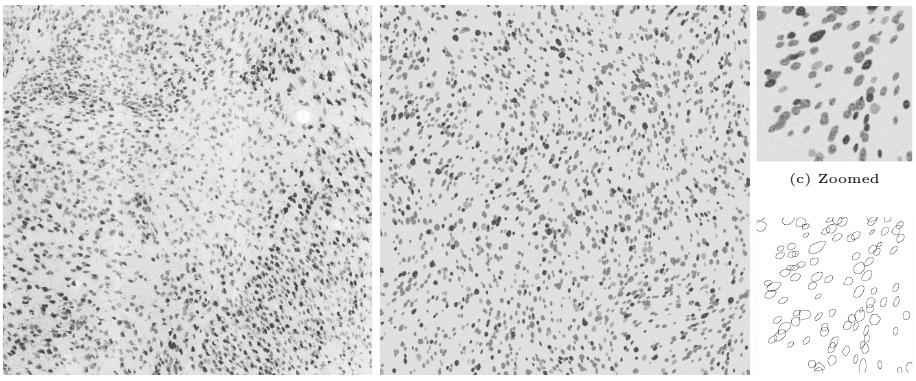
There are so many algorithms developed to analyze microscopy images to extract all sort of information and features out of raw images which lead to better understanding of brain functionalities and disorders. Users always looking for the best software or toolkit—preferably open source—to perform their desired tasks such as segmentation, reconstruction, and quantification on their image datasets. They have to apply different tools on the same dataset with known ground truth to be able to compare their performances and choose the best. There are two different ways of producing ground truth data: manual reconstruction of real data, or synthetically generating datasets with known ground truths.

Obviously, automatic generation of synthetic images with known ground truths has several advantages over the manual reconstruction. First, manual reconstructions prepared by experts are prone to errors, while image microenvironment that are mathematically modeled, provides the means of automatically generating exact ground truth data for a synthetic image dataset via computer. Second, manual reconstruction is time consuming, user dependent, and needs a lot of effort and skills to be acceptable, however a computer framework is able to generate high throughput synthetic image datasets representing different features in a matter of seconds once it is implemented.

2.2 Method and Algorithms

In this section we give a brief explanation on the method and algorithms behind the neuron image synthesizer developed in our toolkit (refer to our paper [1] for a comprehensive explanation). There are two main steps in the implementation of the synthesizer: First generating ground truth data, second generating foreground and background textures and simulating noise.

Elliptical shapes are generated as nuclei via spline interpolation of random points on ellipse perimeter. The parameters of ellipses are estimated by first obtaining the histogram distribution of the major radii and eccentricities via measurement of these parameters in 250 cells. Then, a Gaussian mixture model (GMM) consisting of two Gaussian components is fitted to each histogram. The optimum parameters for GMMs are estimated via expectation maximization (EM) algorithm. Cells are randomly located via uniform distribution, and their population density are controlled by user. The orientations of the neuronal nuclei can also be realized via Perlin noise [2, 3] to mimic the actual behavior of the cells.



(a) Real Image (b) Synthesized Image (c) Zoomed (d) Ground Truth

Figure 2: Synthesized image and its ground truth. (a) A real image of mature neurons stained for NeuN to label nuclei, size 2800×2800 pixels spanning $1310\mu\text{m} \times 1310\mu\text{m}$ physical size, (b) synthesized image generated by our proposed method, (c) zoomed in to the right lower corner of the synthesized image, (d) ground truth generated by our method illustrating the outlines of the nuclei inside the zoomed image.

The textures of the foreground (nuclei) and the background are generated via Perlin noise function [2, 3], and then assigned intensities generated by applying three-components GMM to the real data as follows. 250 cells are categorized into three intensity groups, and their histogram along with that of the background are employed to estimate the GMMs parameters via EM algorithm.

The last modifications to the synthetic texture are blurring and adding noise. Point spread function (PSF) is a characteristic of microscope which specifies the amount of blurring that occurs during imaging. It is common that PSF is theoretically modeled by a Gaussian smoothing filter [4]. The amount of blurring is controlled by the Gaussian kernel variance. The noise of charge coupled device (CCD) detectors is also modeled by additive white Gaussian noise (AWGN) [4].

A synthetic image of neuron nuclei and its ground truth generated via our method, along with the real image are illustrated in Fig. 2 as an example. It can be seen that the behavior and characteristics of the nuclei, e.g. shape, population, and orientation, as well as the foreground and the background texture are well simulated via our method.

2.3 Instructions on How to Generate Synthetic Images

There is a folder called *'Modelling'* inside the toolbox. After navigating to this folder through MATLAB, it must be added to the MATLAB path to make all files and folders inside it known to MATLAB to be able to execute them together. This is done by right clicking on the *'Modelling'* folder inside the *'Current Folder'* sub-window and choosing *'Selected Folders and Subfolders'* as

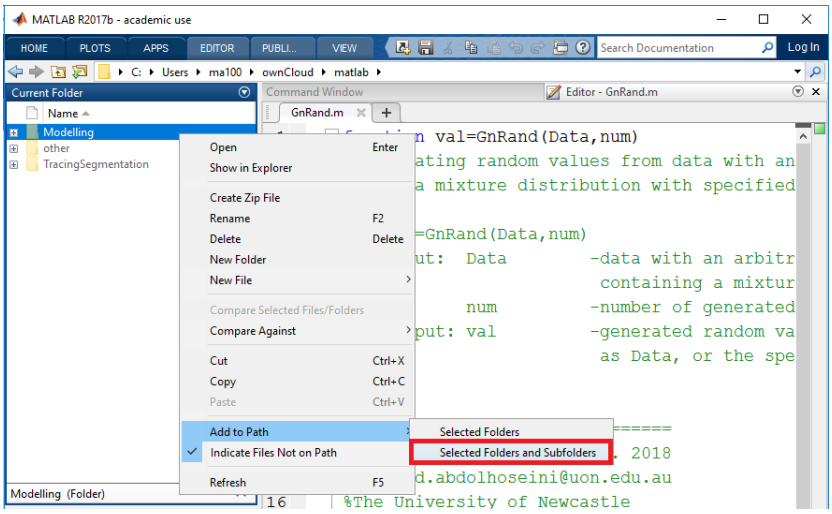


Figure 3: How to add a folder to MATLAB path.

shown in Fig 3. Two files named *'SetupCellParameter'* and *'GnCell'* inside the *Modelling* folder must be executed in order to generate synthetic images. This procedure is explained in the following two steps.

2.3.1 Step 1: Generating Parameters via *SetupCellParameter*

Inside the *Modelling* folder, there is a file named *'SetupCellParameter'* which setups parameters specifically designed to generate **neuron nucleus images** and their ground truths. Therefore, it is required to run *SetupCellParameter* as the first step before running any other file. Two ways to execute a file in MATLAB are either by typing its name in *'Command Windows'* and then pressing the Enter key, or open the file in *'Editor'* by double clicking on it and then pressing F5 key inside the script. Fig. 4 shows the output variables inside *'Workspace'* after running *'SetupCellParameter'*.

Variables specifying cellular features and microenvironment properties are: *'paraBG'*, *'paraFG'*, *'paraEccentricity'*, and *'paraMajorAxis'* store the parameters of the GMMs simulating the background and foreground intensity, the eccentricity and major axis of the nucleus elliptical shape respectively. *'pixelSize'*, *'popDensity'*, *'mask'*, and *'catWeight'* specify image pixel size in micrometer, population density of cells, maximum and minimum sizes of the nucleus eccentricity and area, and the category weights for the nucleus intensity respectively. After generating these parameters, we are ready to move to the next step.

2.3.2 Step 2: Generating Image via *GnCell*

In this step, we generate a synthetic image and its ground truth via *GnCell* based on the variables output by *SetupCellParameter* in Step 1. First, open the

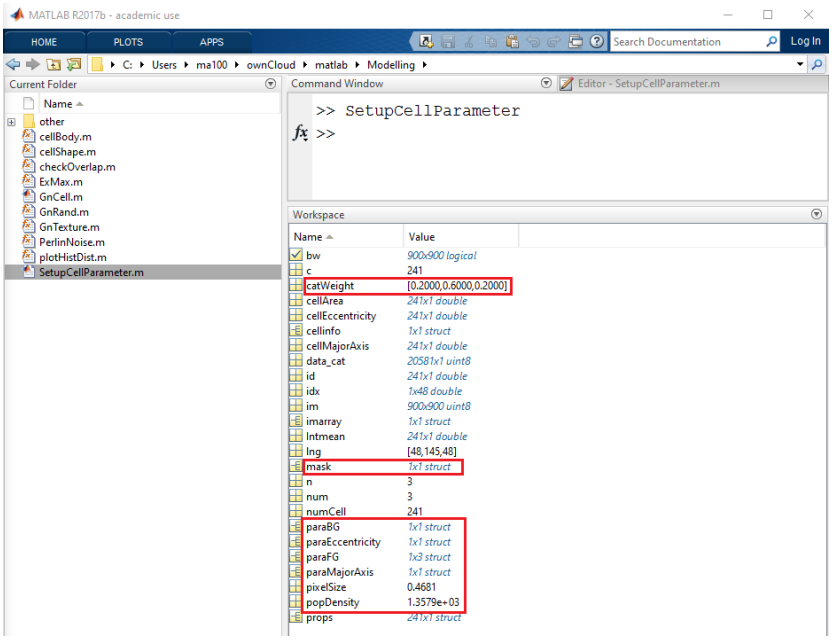


Figure 4: The output of *'SetupCellParameter'*.

file *GnCell* inside *Current Folder*, then appropriate values need to be assigned to the parameters depicted in Fig. 5 as follows.

- `imSize` specifies a desired size for the output image. It must be assigned two values specifying number of image rows and columns as in `[NumberOfRows,NumberOfColumns]`.
- `maxOL` specifies maximum nucleus overlapping. Any value between 0 to 1 specifies the percentage of maximum overlapping allowed. For instance, if `maxOL=0;` it means nucleus overlapping in the output image is not allowed at all, and `maxOL=1;` means maximum of 100% overlapping is allowed.
- `ssOr` is required for the orientation of the nuclei. The less this parameter, the smoother variation in the nucleus orientations.
- `ssFG` is required for the foreground texture. The less the parameter, the smoother the texture.
- `ssBG` is required for the background texture. The less the parameter, the smoother the texture.
- `octave` specifies the desire number of octaves required for Perlin noise. Increasing this parameter will add high frequency noise to the textures.

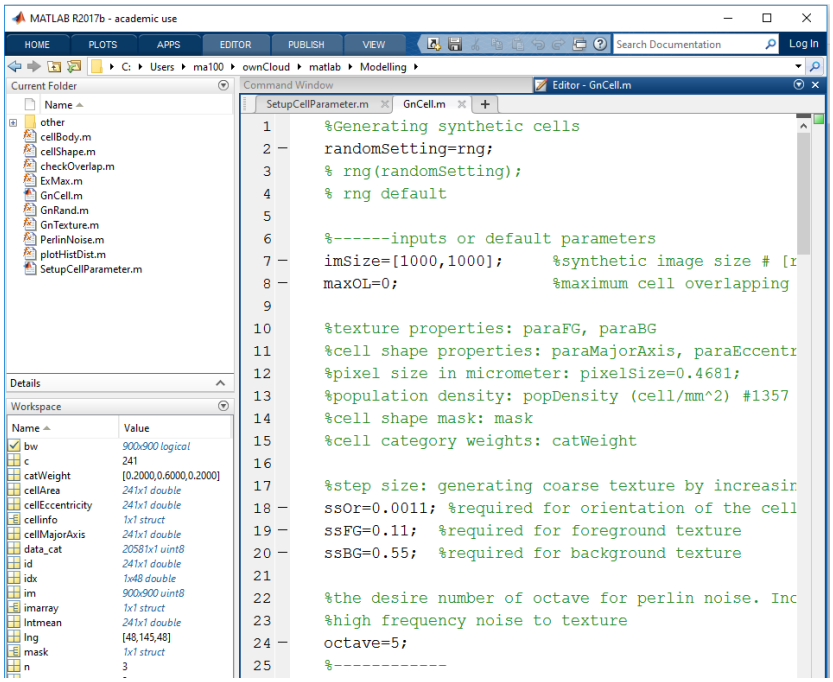


Figure 5: The parameters required to generate synthetic images with desired size, cell shape, and textures via *GnCell*.

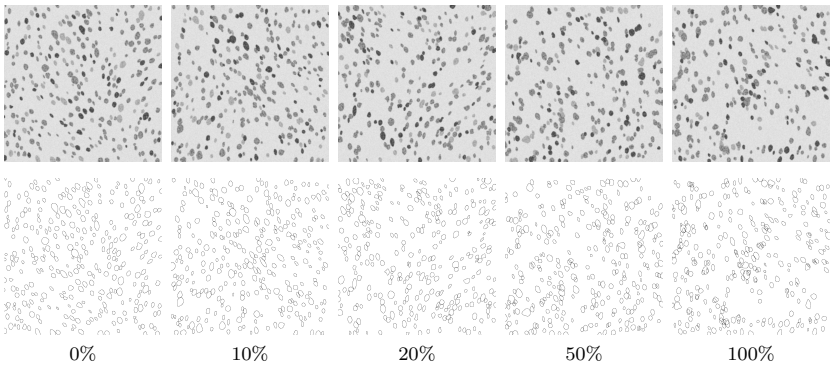


Figure 6: Synthetic images (top row), along with their ground truths (bottom row) generated via *GnCell* for the various overlapping criteria from none to 100%.

2.4 Simulation Examples

2.4.1 Nucleus Overlapping

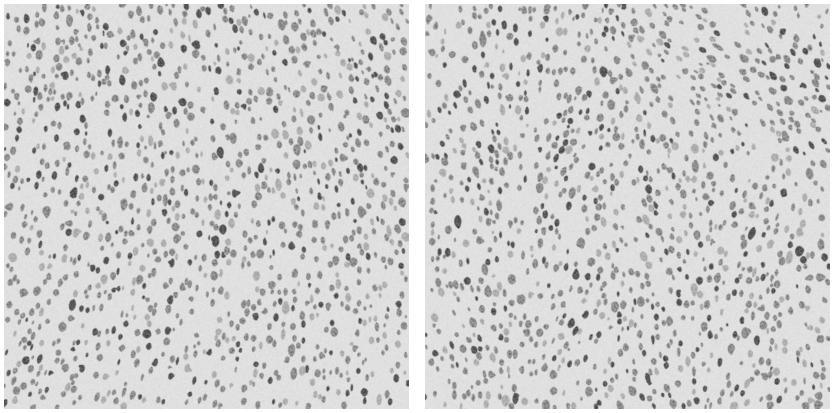
We are going to show how manipulating these parameters effect the image output and cell organization through a few examples in this section. Let's generate nucleus images with the same size, 1000×1000, but different nucleus overlapping: 0, 10%, 20%, 50%, and 100%. Therefore the parameters inside *GnCell* would be `imSize=[1000,1000];` and the rest as default: `ssOr=0.0011; ssFG=0.11; ssBG=0.55;` and `octave=5`. Then, we run *GnCell* five times, each time assigning one of the following values: 0, 0.1, 0.2, 0.5, and 1 to `maxOL` to generate synthetic images including nuclei with the following maximum overlapping: 0, 10%, 20%, 50%, and 100% (see Fig. 6).

2.4.2 Nucleus Orientations

In this example we are going to show how orientations of the nuclei change by tweaking the parameter `ssOr` inside *GnCell*. The orientations of the neighboring nuclei vary smoothly when `ssOr` is small (close to zero), i.e. they elongate in the same direction (see Fig. 7a,b). As this parameter increases, the variations in the nucleus orientations become rougher and random, i.e. neighboring cells might elongate in different directions (see Fig. 7c,d). The example images in Fig. 7 are generated with the following parameters: `imSize=[2000,2000]; maxOL=0; ssFG=0.11; ssBG=0.55; octave=5;` and `ssOr=0.00011,0.0011,0.011,0.11`.

2.4.3 Image Texture

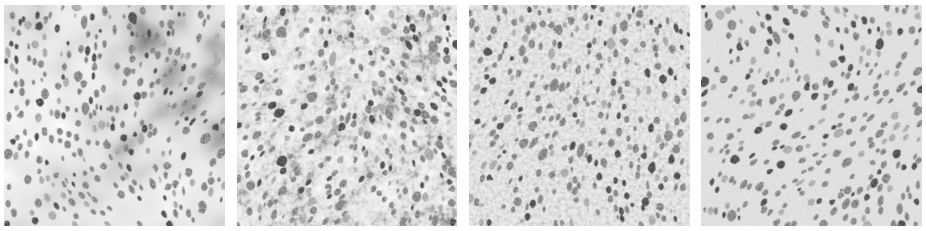
Next, we are going to show how the texture parameters `ssFG` and `ssBG` affect the output image. Same as `ssOr`, by increasing texture parameters, the texture variations move from being smooth to rough. To make it more clear, we



(a) ssOr = 0.00011 (b) ssOr = 0.0011

(c) ssOr = 0.11 (d) ssOr = 0.011

Figure 7: The effect of parameter `ssOr` on the nucleus orientations.



(a) ssBG = 0.00055 (b) ssBG = 0.0055 (c) ssBG = 0.055 (d) ssBG = 0.55

Figure 8: The effect of parameter `ssBG` on the background texture.

keep the foreground parameter, `ssFG`, fixed and apply the changes through the background parameter, `ssBG`, in this example, however changing `ssFG` have a similar effect on the foreground texture. As it can be seen in Fig. 8, as `ssBG` increases, the background texture becomes more random until it looks like a random noise in Fig. 8d. Parameter selection to generate images in Fig. 8 is as follows: `imSize=[1000,1000]; maxOL=0; ssOr=0.0011; ssFG=0.11; octave=5;` and `ssBG=0.00055,0.0055,0.055,0.55`.

3 Nucleus Segmentation

In this section, we introduce the ability of our toolbox in segmenting nuclei of cells such as neuron, Drosophila, and breast cancer from histopathological images. This section is organized in three subsections elaborating our motivation, method, and the instructions on how to use the code with several examples.

3.1 Motivation

Automated segmentation of cell nuclei is the key to gain further insight into cell features and functionality which support computer-aided pathology in early diagnosis of diseases such as breast cancer and brain tumour. In other words, physiological state of a cell can be recognized by its cellular morphology. This makes the segmentation of nuclei, a fundamental step in both inter and intracellular processes, and a popular topic which has drawn great attention in past years [5]. Despite considerable advances in automated segmentation, it still remains a challenging task to split heavily clustered nuclei, due to intensity variations caused by noise and uneven absorption of stains.

3.2 Method Overview

A novel method applicable to variety of histopathological images stained for different proteins, with high speed, accuracy and level of automation is devised and implemented in this toolbox. Our algorithm is initiated by applying a new locally adaptive thresholding method on watershed regions. Followed by a new splitting technique based on multilevel thresholding and the watershed algorithm to separate clustered nuclei. Finalized by a model-based merging step to eliminate oversegmentation and a model-based correction step to improve segmentation results and eliminate small objects. A flowchart of four main stages of our segmentation method is depicted in Fig. 9. Please refer to our journal paper [6] on this section, if you are interested in more details about this method and its application.

3.3 Instructions on How to Segment Nuclei

Navigate to *TracingSegmentation* folder inside the toolbox and add it to the MATLAB path as explained in Subsection 2.3. Open the file *GliaTrace* inside the folder *TracingSegmentation* into the *Editor* window. Inside the file *GliaTrace*, you will see the parameter section at the beginning (see Fig. 10).

The first two parameters: `cellPro.trace` and `cellPro.segment` used to unable (with '1') or disable (with '0') tracing and segmentation features of the toolkit respectively. In this section, we are going to perform segmentation, therefore `cellPro.segment=1`. Next, we assign proper values to the parameters that belong to the segmentation process as depicted in Fig. 10 by the red square entitled 'for segmentation'. Four segmentation parameters starting with `paraS` are defined as follows:

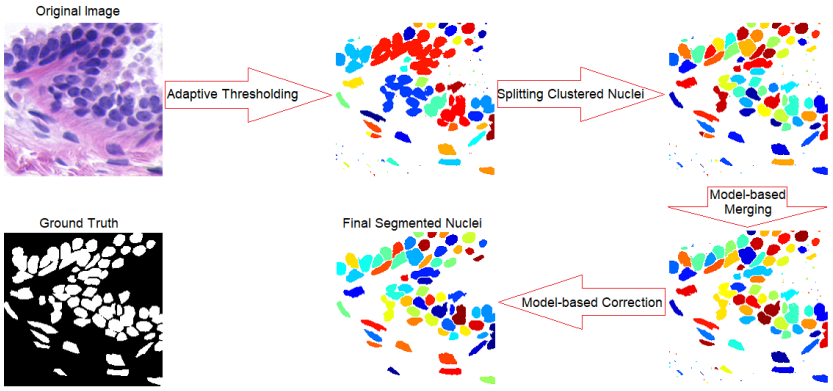


Figure 9: A flowchart of four main stages of our segmentation method. The original image and its ground truth are provided in UCSB dataset of breast cancer [7].

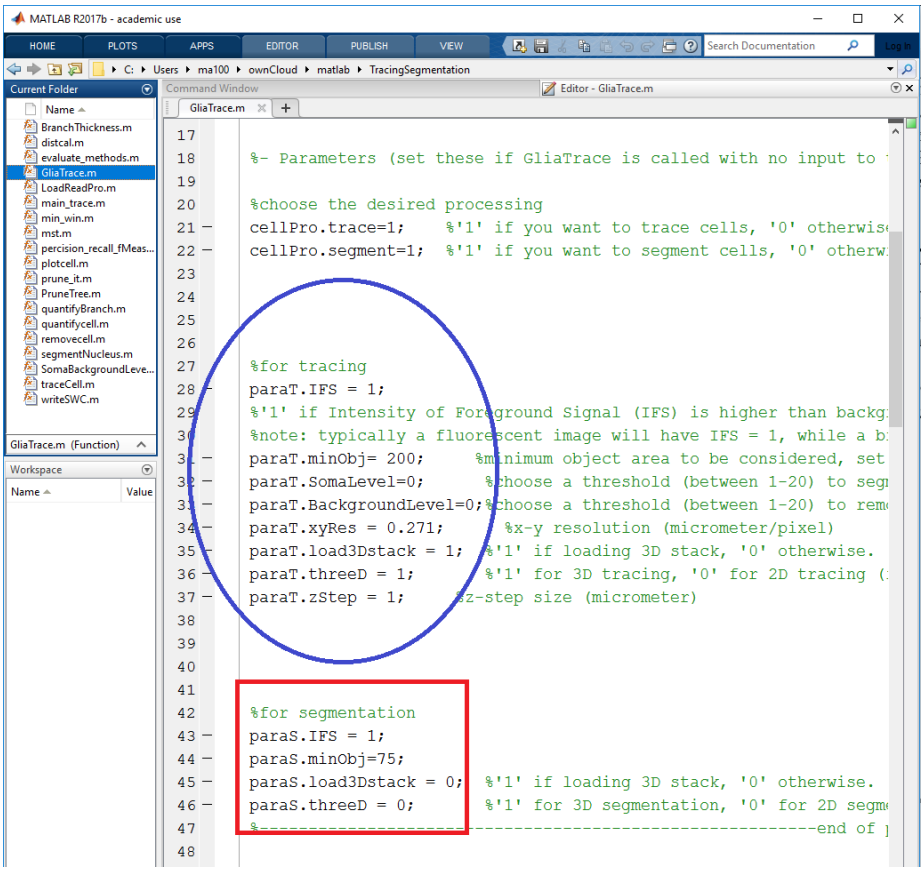


Figure 10: *GliaTrace* parameters.

- `paraS.IFS` must be set to '1' if Intensity of the Foreground Signal (IFS) is higher than the background, and '0' otherwise, i.e. typically for a fluorescent image, we set `paraS.IFS=1`, while for a brightfield image, we set `paraS.IFS=0`.
- `paraS.minObj` specifies the minimum object allowed to be segmented in pixels. Objects less than this parameter are considered as noise and discarded during the process.
- `paraS.load3Dstack` must be set to '1' if loading 3D stack, and '0' otherwise.
- `paraS.threeD` must be set to '1' for 3D segmentation, and '0' for 2D segmentation.

After assigning desired values to the parameters, we run the *GliaTrace* using the following command, `[cellinfo,imarray]=GliaTrace;` written in *Command Window*. Then the program will prompt you to load images desired to be segmented. The rest of the process will be automatically done, and the results will be visualized, and stored in the output arrays (i.e. `cellinfo` and `imarray`). Also, a folder named 'results' will be created in the same directory as the loaded image to save the segmented result in a '.png' image format. An Excel spreadsheet will also be created in the 'results' folder to store the image properties, the parameters set by user, number of found objects, their centroid coordinates and sizes in pixels/voxels (see the following examples).

3.4 Cell Segmentation Examples

3.4.1 Drosophila Kc167 Cell

A dataset of Drosophila Kc167 cells stained for DNA to label nuclei is available here (BBBC007 dataset [8]). We run GliaTrace on this dataset to segment the nuclei with parameter `paraS.minObj=30`. Regarding the dataset, the rest of the parameters are as follows: `paraS.IFS=1` (white foreground and dark background), `paraS.load3Dstack=0`, and `paraS.threeD=0` (not 3D). The segmentation results for five images of this dataset are presented in Fig. 11.

3.4.2 Heavily Clustered Nuclei

The next example is the segmentation of nuclei while they are heavily clustered in their environment. For this image, we ran GliaTrace with `paraS.minObj=50` and the rest of the parameters are the same as previous example. The result is presented in Fig. 12.

3.4.3 2D vs 3D Segmentation

In this example, we compare the results of 2D and 3D segmentation by applying GliaTrace to a 3D image stack of neuron nuclei and its maximum intensity

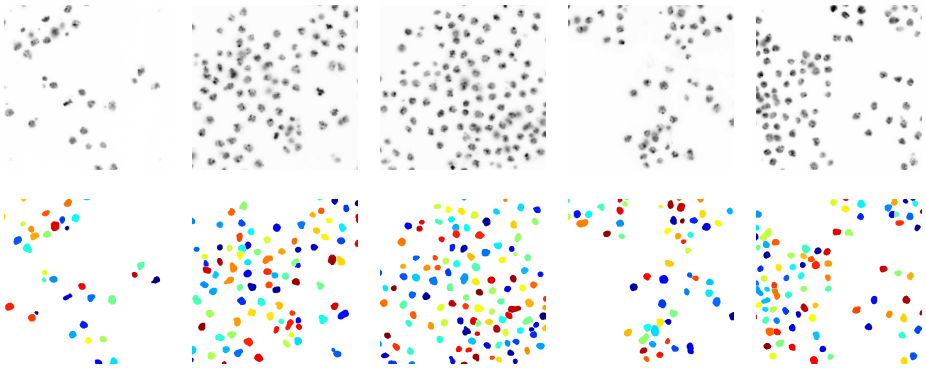
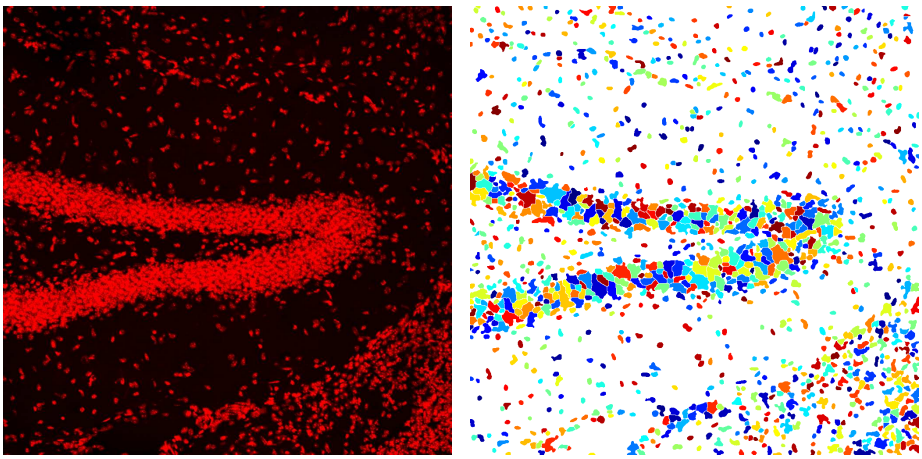


Figure 11: Segmentation of Drosophila Kc167 cell. Top row: original images, bottom row: segmented results. Images are available here (BBBC007 dataset [8]).



(a) Original image (b) Segmented result

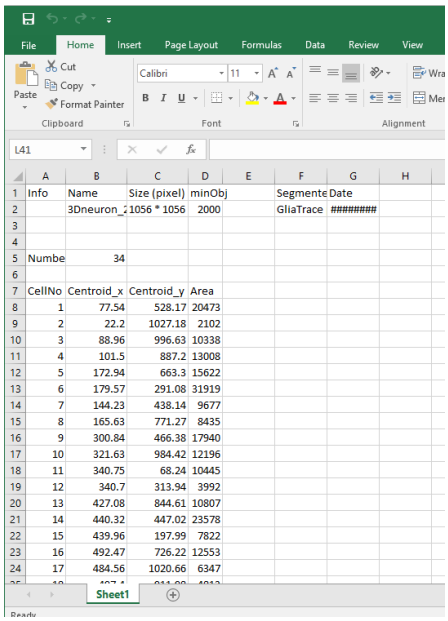
Figure 12: Segmentation of heavily cluster nuclei. Image size: 1024×1024 pixels showing 1mm $\times$ 1mm physical size.

(a) MIP of 3D image (b) Segmented result 2D (c) Segmented result 3D

Figure 13: 2D vs 3D Segmentation. Image size: $1056 \times 1056 \times 12$ voxels spanning 92.26 µm $\times$92.26 µm $\times$16.51 µm physical size.

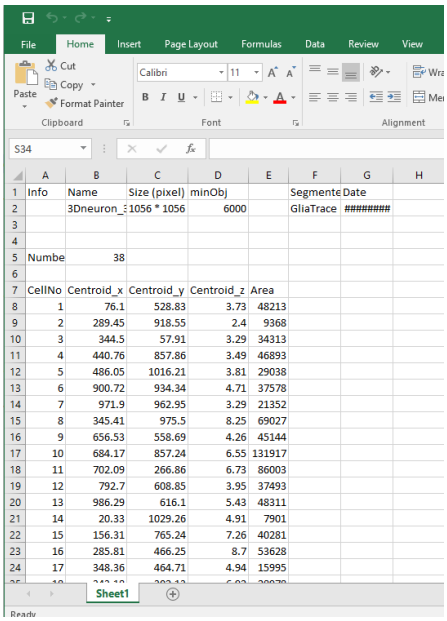
projection (MIP). The neuron nucleus is magnified in this image, therefore a large value must be assigned to `paraS.minObj`, which is `paraS.minObj=2000` for 2D segmentation and `paraS.minObj=6000` for 3D. Since the image is fluorescent and 3D, the following parameters are the same in both case: `paraS.IFS=1`; `paraS.load3Dstack=1`. As for the last parameter, `paraS.threeD`, we have to run the code once with `paraS.threeD=0` for 2D segmentation of the stack MIP, and another time with `paraS.threeD=1` for 3D segmentation. The results of both 2D and 3D segmentation are presented in Fig. 13 for comparison.

Fig. 14 shows the information stored in the Excel spreadsheet inside the folder 'results' created in the same directory as the loaded image. As shown, the spreadsheet includes information such as: image properties, the set parameters, number of found objects, their centroid coordinates and sizes in pixels/voxels.



	A	B	C	D	E	F	G	H
1	Info	Name	Size (pixel)	minObj		Segmente	Date	
2		3Dneuron_2	1056 * 1056	2000		GliaTrace	########	
3								
4								
5	Numbe	34						
6								
7	CellNo	Centroid_x	Centroid_y	Area				
8	1	77.54	528.17	20473				
9	2	22.2	1027.18	2102				
10	3	88.96	996.63	10338				
11	4	101.5	887.2	13008				
12	5	172.94	663.3	15622				
13	6	179.57	291.08	31919				
14	7	144.23	438.14	9677				
15	8	165.63	771.27	8435				
16	9	300.84	466.38	17940				
17	10	321.63	984.42	12196				
18	11	340.75	68.24	10445				
19	12	340.7	313.94	3992				
20	13	427.08	844.61	10807				
21	14	440.32	447.02	23578				
22	15	439.96	197.99	7822				
23	16	492.47	726.22	12553				
24	17	484.56	1020.66	6347				

(a) Spreadsheet for 2D segmentation



	A	B	C	D	E	F	G	H
1	Info	Name	Size (pixel)	minObj		Segmente	Date	
2		3Dneuron_3	1056 * 1056	6000		GliaTrace	#########	
3								
4								
5	Numbe	38						
6								
7	CellNo	Centroid_x	Centroid_y	Centroid_z	Area			
8	1	76.1	528.83	3.73	48213			
9	2	289.45	918.55	2.4	9368			
10	3	344.5	57.91	3.29	34313			
11	4	440.76	857.86	3.49	46893			
12	5	486.05	1016.21	3.81	29038			
13	6	900.72	934.34	4.71	37578			
14	7	971.9	962.95	3.29	21352			
15	8	345.41	975.5	8.25	69027			
16	9	656.53	558.69	4.26	45144			
17	10	684.17	857.24	6.55	131917			
18	11	702.09	266.86	6.73	86003			
19	12	792.7	608.85	3.95	37493			
20	13	986.29	616.1	5.43	48311			
21	14	20.33	1029.26	4.91	7901			
22	15	156.31	765.24	7.26	40281			
23	16	285.81	466.25	8.7	53628			
24	17	348.36	464.71	4.94	15995			

(b) Spreadsheet for 3D segmentation

Figure 14: Information stored in spreadsheet.

4 Cell Reconstruction and Quantification

In this section we explain how to reconstruct cells' body and branches, and quantify their features via *GliaTrace*. The code is designed to reconstruct microglial cells from 2D and 3D microscopy images, however it works on neuronal microenvironment as well. The rest of this section is organized as follows: first, we express our motivation in microglial reconstruction, and explaining a bit about the algorithms behind the scenes, finally give instructions on how to apply the toolbox to reconstruct and quantify the cells.

4.1 Motivation

Microglia have recently been identified to play a central role in modulating synaptic structure, synaptic physiology as well as learning and memory processes. Microglia are not in permanent physical contact with other cells in their micro-environment. Therefore they need to engage in significant structural change to perform any of their known activities [9]. They exhibit distinct morphological transition states when undertaking specific biological functions. Therefore the morphological study of microglia is an important tool in understanding their functionality.

Manual reconstruction and quantification of the cells are very time consuming and prone to error, especially when dealing with 3D image stacks. Despite many promising automated methods in segmentation of blood vessels [10], and reconstruction of neuronal morphology [11], our evaluation results reveal that these methods often fail to accurately capture microglia organizations due to the striking structural differences. Soma size and branch architecture vary considerably between neurons, microglia and astrocytes but most importantly the morphology among microglia cells is highly variable. Neurons resemble a relative stable and reliable tree structure with a primary process, the axon, and multiple dendrites representing the tree trunk and canopy, respectively. Microglial architecture is more defuse and complex with multiple primary and secondary branches not confined to a generalizable structure. Additionally microglia morphology is highly diverse and changes by increasing and decreasing branches. This requires a more sophisticated approach of reconstruction taking into account the varying nature of branch structures and soma sizes.

4.2 Method Overview

We have proposed a novel approach to reconstruct the microglial cells, and quantify their features from 2D/3D image datasets. The segmentation of the soma, and background removal via multilevel thresholding is the first step of our algorithm. Then a tracing step is initiated with the centroid of the somas and connects the prioritized seed points extracted from the cell voxels to build a tree structure for each cell. The thickness of the traced skeleton is estimated as the next step. Finally, the output data is quantified to extract the cell features such as: number of primary branches and branch points, branch length, soma

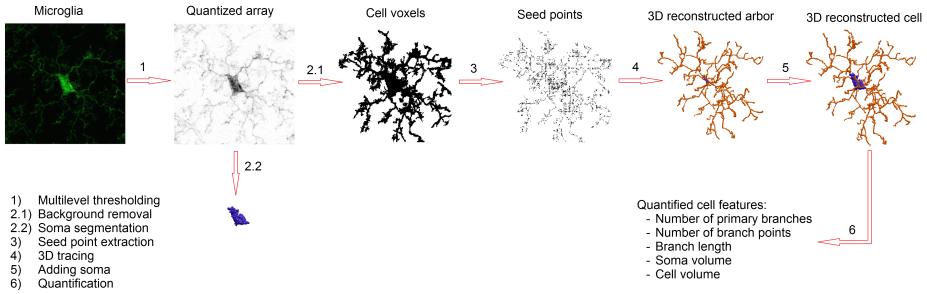


Figure 15: Flowchart of our method for reconstruction and quantification of microglia.

size, and cell size. The whole of the process is automated and results in an objective morphological analysis of microglia with high speed and accuracy. The reconstructed data is written in SWC file format, so that it can be visualized or quantified via many toolboxes designed to read this standard format. Fig. 15 represents these steps via a flowchart. Please refer to our papers [12, 13] on this section, if you are interested in more details about this method and its application.

4.3 Instructions on How to Reconstruct Cells

Navigate to *TracingSegmentation* folder inside the toolbox and add it to the MATLAB path as explained in Subsection 2.3. Open the file *GliaTrace* inside the folder *TracingSegmentation* into the *Editor* window. Inside the file *GliaTrace*, you will see the parameter section at the beginning (see Fig. 10).

The first two parameters: `cellPro.trace` and `cellPro.segment` used to unable (with '1') or disable (with '0') tracing and segmentation features of the toolkit respectively. In this section, we are going to perform reconstruction, therefore `cellPro.trace=1`. Next, we assign proper values to the parameters that belong to the reconstruction process as depicted in Fig. 10 by the blue oval entitled 'for tracing'. Eight tracing parameters starting with `paraT` are defined as follows:

- `paraT.IFS` must be set to '1' if Intensity of the Foreground Signal (IFS) is higher than the background, and '0' otherwise, i.e. typically for a fluorescent image, `paraT.IFS=1`, while for a brightfield image, `paraT.IFS=0`.
- `paraT.minObj` specifies the minimum object (in pixels) allowed to be segmented as soma. Objects less than this parameter are considered as noise and discarded during the process. If you are not sure about this parameter, set it '0', so it will be assigned automatically.
- `paraT.load3Dstack` must be set to '1' if loading 3D stack, and '0' otherwise.

- `paraT.threeD` must be set to '1' for 3D reconstruction, and '0' for 2D.
- `paraT.SomaLevel` is a threshold and must be chosen from a range of 1–20 to segment soma. Leave it '0' for automatic selection.
- `paraT.BackgroundLevel` is a threshold and must be chosen from a range of 1–20 to remove background. Leave it '0' for automatic selection. Note that `paraT.BackgroundLevel` must be greater than `paraT.SomaLevel`, if they have values other than zeros.
- `paraT.xyRes` is x-y resolution of the image in micrometer/pixel.
- `paraT.zStep` is z-step size of the 3D image stack in micrometer.

After assigning desired values to the parameters, we run the *GliaTrace* using the following command, `[cellinfo,imarray]=GliaTrace;` written in *Command Window*. Then the program will prompt you to load images desired to be reconstructed and quantified. The rest of the process will be automatically done, and the output (reconstructed images) will be stored in the output arrays (i.e. `cellinfo` and `imarray`), visualized and saved in a folder named 'results' created in the same directory as the loaded images. The traced cell information will also be stored in the SWC file format ready to be visualized and quantified by several open source applications such as: Neuromantic [14], SharkViewer [15], Vaa3D [16], and L-Measure [17].

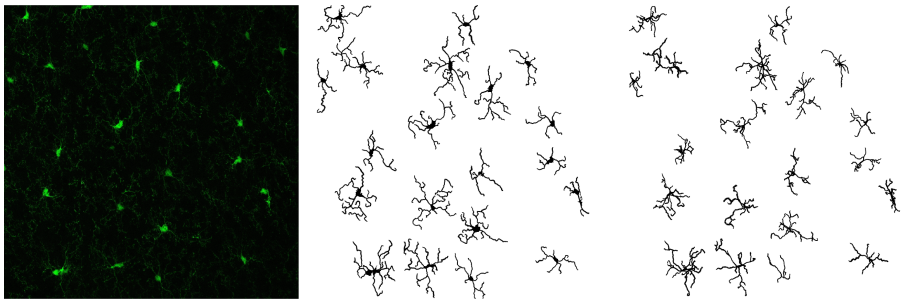
The parameters set by user, and the quantified features of the cells will be stored in an Excel spreadsheet created inside 'results' folder. These features include: number of primary branches and branch points, branch length, cell radius, soma and cell area/volume, soma eccentricity, cell solidity and extent. The following example demonstrate the process and the output results.

4.4 Cell Reconstruction Example

In this section, we explain how to tune the parameters in *GliaTrace* to get the accurate results through an example.

4.4.1 2D vs 3D Reconstruction

In this example, we reconstruct a 3D image stack of microglial cells and quantify their features. We also perform the same process on 2D MIP of this image to compare the results of 2D reconstruction with 3D. The following are the properties of the image in this example: size: $1024 \times 1024 \times 26$ voxels, x-y resolution: 0.271 µm/pixel, z-step size: 1 µm. The following parameters are set in *GliaTrace* to perform the task: `paraT.IFS=1` (the image is fluorescent), `paraT.minObj=0` (to be determined automatically), `paraT.load3Dstack=1` (the image is a 3D stack), `paraT.threeD=1` for 3D reconstruction and `paraT.threeD=0` for 2D, `paraT.SomaLevel=0` (to be selected automatically), `paraT.BackgroundLevel=0` (to be selected automatically), `paraT.xyRes=0.271` and `paraT.zStep=1` (according to the image setting).

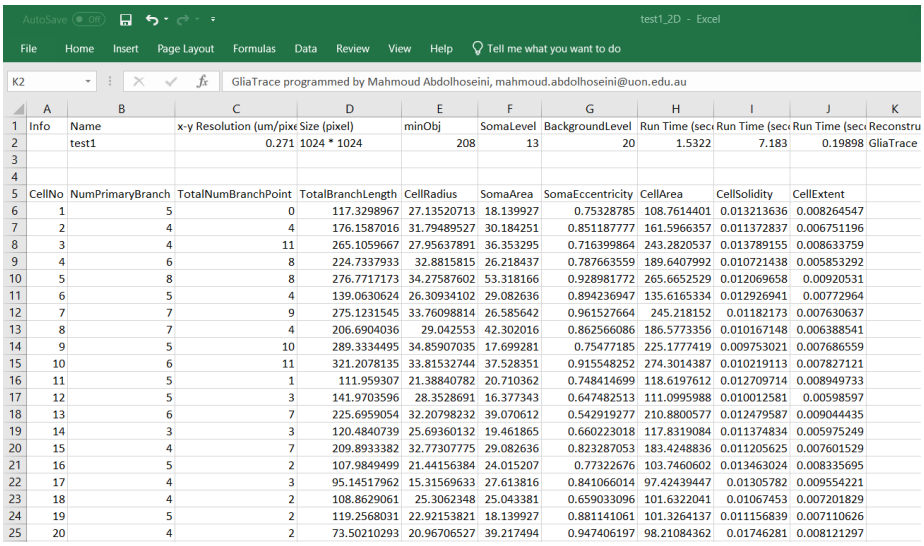


(a) Image (b) 2D Reconstruction (c) 3D Reconstruction

Figure 16: 2D vs 3D Reconstruction. (a) MIP of the original image, (b) 2D reconstruction of the cells, (c) x-y view of the 3D reconstruction.

Both 2D and 3D reconstructed cells are visualized in Fig. 16. Note that the 3D reconstruction illustrated in Fig. 16 is an x-y view of full 3D. For a full 3D visualization, open the SWC output file created in the 'results' folder via an open-source application such as SharkViewer [15].

The quantified features of the reconstructed cells in this example are depicted in Figs. 17 and 18, which belong to 2D and 3D reconstructions respectively. These information are stored in Excel spreadsheets saved in the 'results' folder. Note, length is in micrometers, area in μm^2, and volume in μm^3.



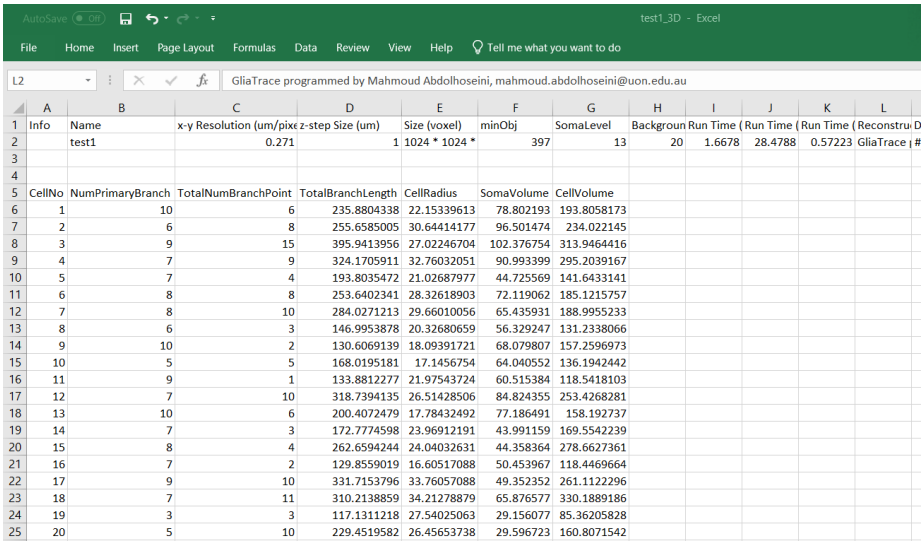
AutoSave Off test1_2D - Excel

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

K2 GliaTrace programmed by Mahmoud Abdolhoseini, mahmoud.abdolhoseini@uon.edu.au

	A	B	C	D	E	F	G	H	I	J	K
1	Info	Name	x-y Resolution (um/pixe	Size (pixel)	minObj	SomaLevel	BackgroundLevel	Run Time (sec	Run Time (sec	Run Time (sec	Reconstru
2		test1	0.271	1024 * 1024	208	13	20	1.5322	7.183	0.19898	GliaTrace
3											
4											
5	CellNo	NumPrimaryBranch	TotalNumBranchPoint	TotalBranchLength	CellRadius	SomaArea	SomaEccentricity	CellArea	CellSolidity	CellExtent	
6	1	5	0	117.3298967	27.13520713	18.139927	0.75328785	108.7614401	0.013213636	0.008264547	
7	2	4	4	176.1587016	31.79489527	30.184251	0.851187777	161.5966357	0.011372837	0.006751196	
8	3	4	11	265.1059667	27.95637891	36.353295	0.716399864	243.2820537	0.013789155	0.008633759	
9	4	6	8	224.7337933	32.8815815	26.218437	0.787663559	189.6407992	0.010721438	0.005853292	
10	5	8	8	276.7717173	34.27587602	53.318166	0.928981772	265.6652529	0.012069658	0.00920531	
11	6	5	4	139.0630624	26.30934102	29.082636	0.894236947	135.6165334	0.012926941	0.00772964	
12	7	7	9	275.1231545	33.76098814	26.585642	0.961527664	245.218152	0.01182173	0.007630637	
13	8	7	4	206.6904036	29.042553	42.302016	0.862566086	186.5773356	0.010167148	0.006388541	
14	9	5	10	289.3334495	34.85907035	17.699281	0.75477185	225.1777419	0.009753021	0.007686559	
15	10	6	11	321.2078135	33.81532744	37.528351	0.915548252	274.3014387	0.010219113	0.007827121	
16	11	5	1	111.959307	21.38840782	20.710362	0.748414699	118.6197612	0.012709714	0.008949733	
17	12	5	3	141.9703596	28.3528691	16.377343	0.647482513	111.0995988	0.010012581	0.00598597	
18	13	6	7	225.6959054	32.20798232	39.070612	0.542919277	210.8800577	0.012479587	0.009044435	
19	14	3	3	120.4840739	25.69360132	19.461865	0.660223018	117.8319084	0.011374834	0.005975249	
20	15	4	7	209.8933382	32.77307775	29.082636	0.823287053	183.4248836	0.011205625	0.007601529	
21	16	5	2	107.9849499	21.44156384	24.015207	0.77322676	103.7460602	0.013463024	0.008335695	
22	17	4	3	95.14517962	15.31569633	27.613816	0.841066014	97.42439447	0.01305782	0.009554221	
23	18	4	2	108.8629061	25.3062348	25.043381	0.659033096	101.6322041	0.01067453	0.007201829	
24	19	5	2	119.2568031	22.92153821	18.139927	0.881141061	101.3264137	0.011156839	0.007110626	
25	20	4	2	73.50210293	20.96706527	39.217494	0.947406197	98.21084362	0.01746281	0.008121297	

Figure 17: Quantified features of the 2D reconstructed cells.



AutoSave Off test1_3D - Excel

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

L2 GliaTrace programmed by Mahmoud Abdolhoseini, mahmoud.abdolhoseini@uon.edu.au

	A	B	C	D	E	F	G	H	I	J	K	L
1	Info	Name	x-y Resolution (um/pixe	z-step Size (um)	Size (voxel)	minObj	SomaLevel	Backgroun	Run Time (	Run Time (	Run Time (	Reconstru D
2		test1	0.271	1	1024 * 1024 *	397	13	20	1.6678	28.4788	0.57223	GliaTrace (#
3												
4												
5	CellNo	NumPrimaryBranch	TotalNumBranchPoint	TotalBranchLength	CellRadius	SomaVolume	CellVolume					
6	1	10	6	235.8804338	22.15339613	78.802193	193.8058173					
7	2	6	8	255.6585005	30.64414177	96.501474	234.022145					
8	3	9	15	395.9413956	27.02246704	102.376754	313.9464416					
9	4	7	9	324.1705911	32.76032051	90.993399	295.2039167					
10	5	7	4	193.8035472	21.02687977	44.725569	141.6433141					
11	6	8	8	253.6402341	28.32618903	72.119062	185.1215757					
12	7	8	10	284.0271213	29.66010056	65.435931	188.9955233					
13	8	6	3	146.9953878	20.32680659	56.329247	131.2338066					
14	9	10	2	130.6069139	18.09391721	68.079807	157.2596973					
15	10	5	5	168.0195181	17.1456754	64.040552	136.1942442					
16	11	9	1	133.8812277	21.97543724	60.515384	118.5418103					
17	12	7	10	318.7394135	26.51428506	84.824355	253.4268281					
18	13	10	6	200.4072479	17.78432492	77.186491	158.192737					
19	14	7	3	172.7774598	23.96912191	43.991159	169.5542239					
20	15	8	4	262.6594244	24.04032631	44.358364	278.6627361					
21	16	7	2	129.8559019	16.60517088	50.453967	118.4469664					
22	17	9	10	331.7153796	33.76057088	49.352352	261.1122296					
23	18	7	11	310.2138859	34.21278879	65.876577	330.1889186					
24	19	3	3	117.1311218	27.54025063	29.156077	85.36205828					
25	20	5	10	229.4519582	26.45653738	29.596723	160.8071542					

Figure 18: Quantified features of the 3D reconstructed cells.

References

[1] M. Abdolhoseini, M. G. Kluge, F. Walker, and S. Johnson, "Neuron image synthesizer via gaussian mixture model and perlin noise," in *IEEE ISBI*, Apr 2019.

[2] K. Perlin, "An image synthesizer," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 287–296, Jul. 1985.

[3] ——, "Improving noise," 2002. [Online]. Available: https://mrl.nyu.edu/~perlin/paper445.pdf

[4] H. Kaufman and A. M. Tekalp, "Survey of estimation techniques in image restoration," *IEEE Control Systems*, vol. 11, no. 1, pp. 16–24, Jan 1991.

[5] E. Meijering, "Cell segmentation: 50 years down the road [life sciences]," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 140–145, 2012.

[6] M. Abdolhoseini, M. G. Kluge, F. R. Walker, and S. J. Johnson, "Segmentation of heavily clustered nuclei from histopathological images," *Scientific Reports*, vol. 9, March 2019.

[7] E. D. Gelasca, J. Byun, B. Obara, and B. S. Manjunath, "Evaluation and benchmark for biological image segmentation," in *ICIP*, 2008, pp. 1816–1819.

[8] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter, "Annotated high-throughput microscopy image sets for validation," *Nature Methods*, vol. 9, p. 637, 2012.

[9] F. R. Walker, S. B. Beynon, K. A. Jones, Z. Zhao, R. Kongsui, M. Cairns, and M. Nilsson, "Dynamic structural remodelling of microglia in health and disease: A review of the models, the signals and the mechanisms," *Brain, Behavior, and Immunity*, vol. 37, pp. 1–14, 2014.

[10] D. Lesage, E. D. Angelini, I. Bloch, and G. Funka-Lea, "A review of 3D vessel lumen segmentation techniques: Models, features and extraction schemes," *Medical Image Analysis*, vol. 13, no. 6, pp. 819–845, 2009.

[11] L. Acciai, P. Soda, and G. Iannello, "Automated neuron tracing methods: An updated account," *Neuroinformatics*, vol. 14, no. 4, pp. 353–367, Oct 2016.

[12] M. Abdolhoseini, M. G. Kluge, F. R. Walker, and S. J. Johnson, "Segmentation, tracing, and quantification of microglial cells from 3d image stacks," *Scientific Reports*, 2019.

[13] M. Abdolhoseini, F. Walker, and S. Johnson, "Automated tracing of microglia using multilevel thresholding and minimum spanning trees," in *IEEE EMBC*, 2016, pp. 1208–1211.

[14] D. Myatt, T. Hadlington, G. Ascoli, and S. Nasuto, "Neuromantic – from semi-manual to semi-automatic reconstruction of neuron morphology," *Frontiers in Neuroinformatics*, vol. 6, p. 4, 2012.

[15] C. Weaver and C. Bruns. (2014) Sharkviewer http://www.janelia.org/sharkviewer.

[16] H. Peng, Z. Ruan, F. Long, J. H. Simpson, and E. W. Myers, "V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets," *Nature biotechnology*, vol. 28, no. 4, pp. 348–53, 04 2010.

[17] R. Scorcioni, S. Polavaram, and G. A. Ascoli, "L-Measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies," *Nat. Protocols*, vol. 3, pp. 866–876, 2008.