Journal of Visualized Experiments

Automatic identification of dendritic branches and their orientation

--Manuscript Draft--

Article Type:	Invited Methods Article - JoVE Produced Video
Manuscript Number:	JoVE62679R2
Full Title:	Automatic identification of dendritic branches and their orientation
Corresponding Author:	Refael Minnes Ariel University Faculty of Natural Sciences Ariel, ISRAEL
Corresponding Author's Institution:	Ariel University Faculty of Natural Sciences
Corresponding Author E-Mail:	refaelm@ariel.ac.il
Order of Authors:	Inbar Dahari
	Danny Baranes
	Refael Minnes
Additional Information:	
Question	**Response**
Please specify the section of the submitted manuscript.	Neuroscience
Please indicate whether this article will be Standard Access or Open Access.	Standard Access ($1400)
Please indicate the **city, state/province, and country** where this article will be **filmed**. Please do not use abbreviations.	Ariel, Israel
Please confirm that you have read and agree to the terms and conditions of the author license agreement that applies below:	I agree to the Author License Agreement
Please provide any comments to the journal here.	
Please confirm that you have read and agree to the terms and conditions of the video release that applies below:	I agree to the Video Release

TITLE:

Automatic Identification of Dendritic Branches and Their Orientation

AUTHORS AND AFFILIATIONS:

Inbar Dahari[1], Danny Baranes[1], Refael Minnes[2]

[1]Department of Molecular Biology, Ariel University, Ariel, 4070000, Israel
[2]Department of Physics, Ariel University, Ariel, 4070000, Israel

Email addresses of co-authors:

Inbar Dahari (inbar.dahari@msmail.ariel.ac.il)
Danny Baranes (dannyb@ariel.ac.il)

Corresponding author:

Refael Minnes (refaelm@ariel.ac.il)

SUMMARY:

Presented is a computational tool that allows simple and direct automatic measurement of orientations of neuronal dendritic branches from 2D fluorescence images.

ABSTRACT:

The structure of neuronal dendritic trees plays a key role in the integration of synaptic inputs in neurons. Therefore, characterization of the morphology of dendrites is essential for a better understanding of neuronal function. However, the complexity of dendritic trees, both when isolated and especially when located within neuronal networks, has not been completely understood. We developed a new computational tool, **SOA** (Segmentation and Orientation Analysis), which allows automatic measurement of the orientation of dendritic branches from fluorescence images of 2D neuronal cultures. SOA, written in Python, uses segmentation to distinguish dendritic branches from the image background and accumulates a database on the spatial direction of each branch. The database is then used to calculate morphological parameters such as the directional distribution of dendritic branches in a network and the prevalence of parallel dendritic branch growth. The data obtained can be used to detect structural changes in dendrites in response to neuronal activity and to biological and pharmacological stimuli.

INTRODUCTION:

Dendritic morphogenesis is a central subject in neuroscience, as the structure of the dendritic tree affects the computational properties of synaptic integration in neurons[1,2,3]. Moreover, morphological abnormalities and modifications in dendritic branches are implicated in degenerative and neuro-developmental disorders[4,5,6]. In neuronal cultures where dendritic ramification can be more readily visualized, the interactions between non-sister dendritic branches regulate the sites and extent of synaptic clustering along the branches, a behavior that may affect synaptic coactivity and plasticity[7,8,9]. Therefore, characterization of the morphological parameters of the dendritic tree using two-dimensional (2D) neuronal cultures is advantageous for understanding dendritic morphogenesis and functionality of single and networks of neurons. Yet, this is a challenging task because dendritic branches form a complex mesh even in “simplified” 2D neuronal cultures.

Several tools have been developed to automatically trace and analyze dendritic structures[10,11,12,13]. However, most of these tools are designed for 3D neuronal networks and are naturally too complex to use with 2D networks. In contrast, less advanced morphological analysis tools typically involve a significant component of computer-assisted manual labor, which is very time-consuming and susceptible to operator bias[14]. Existing semi-automatic tools, such as ’ImageJ’[15] (an NIH open-source image processing package with a vast collection of community-developed biological image analysis tools), largely reduce user manual labor. However, some manual interventions are still needed during image processing, and the quality of the segmentation can be less than desirable.

This paper presents the SOA, a simple automated tool that allows direct segmentation and orientation analysis of dendritic branches within 2D neuronal networks. The SOA can detect various line-like objects in 2D images and characterize their morphological properties. Here, we used the SOA for segmenting dendritic branches in 2D fluorescence images of dendritic networks in culture. The software identifies the dendritic branches and successfully performs measurements of morphological parameters such as parallelism and spatial distribution. The SOA can be easily adapted for the analysis of cellular processes of other cell types and for studying non-biological networks.

PROTOCOL:

NOTE: The Israeli Ministry of Health approved the use of mice under protocol IL-218-01-21 for the ethical use of experimental animals. SOA is only compatible with Windows 10 only and Python 3.9. It is available as an open-source code: https://github.com/inbar2748/DendriteProject. At this link, there is also a README.DM file that has directions for downloading the software, a link to the software's website, and a requirements file containing information on the required versions of all the packages. Additional examples of analysis performed using the software have been provided there as well.

[Place **Figure 1** here]

1. Open the SOA application.

1.1. Open the URL address: https://mega.nz/folder/bKZhmY4I#4WAaec4biiGt4_1lJlL4WA, find the **SOA.zip** zipped folder, and download the ZIP file by double-clicking.

1.2. Unzip the folder by right-clicking on **SOA.zip** and choose **Extract Files.** Observe the **Extraction Path and Options** window that opens and the **Destination Address** text box that displays the path for the extracted files. To extract to a different location, click on one of the folders in the window's right panel to make it the destination folder. Click **OK** to extract the files to that folder.

1.3. Open the extracted SOA file and double-click on **SOA.exe**. Wait for a black window to open, after which the application will appear.

[Place **Figure 2** here]

2. Open an image to analyze.

2.1. In the **SOA Viewer Upload** menu bar | select **Choose File** | choose an image from the computer files | click on it (.png .jpg .tif .bmp files only) | **Open** | observe the path of the file | **Next.**

3. Segmentation optimization

NOTE: In the **SOA Viewer Properties** menu bar, change the values of the selected parameters to adjust the segmentation process settings. A detailed description of the parameters, such as the **Threshold**, is given in the **Supplemental Material**.

3.1. In **Edges**, adjust the threshold for the display by selecting the **Threshold** and entering a number.

NOTE: The lower the number for the threshold, the more lines are detected. Threshold is a number that ranges from 0 to 255. The default value has been set to 0.

3.2. In **Merge Lines:**

3.2.1. Adjust the minimum distance to merge for the display by selecting the **Min distance to merge** and entering a number.

NOTE: The **Min distance to merge** ranges from 0 to 30 pixels. The default value is set to 20.

3.2.2. Adjust the minimum angle to merge for the display by selecting the **Min angle to merge** and entering a number.

NOTE: The **Min angle to merge** ranges from 0 to 30°. The default value is set to 10.

3.3. Click on **Create Preview Segmentation Image**.

NOTE: A preview image of the segmentation results will be displayed according to the updated values. In addition, the number of lines before merging and the number of lines after merging will be displayed.

3.4. Change the parameters to achieve maximum identification of segments. If there is a need to change the Properties, click on the **Close window** button and follow steps 3.1–3.4.

4. Create the output files.

4.1. Press **OK** to visualize the segmentation images and the analyzing graphs. Observe the window that appears for selecting a location where the .xlsx file will be saved.

4.2. insert a file name | Choose **Save** | wait for the .xlsx file with data to be created and saved.

NOTE: In addition to the .xlsx file, the following files will be automatically displayed: a file that presents the original image, the line recognition image, the final image of the segmentation, and three analysis graphs.

5. Navigation toolbar

NOTE: A navigation toolbar is included in all figure windows and can be used to navigate through the data set. Each of the buttons at the bottom of the toolbar is described below.

5.1. To navigate back and forth between previously defined views, use the **Forward** and **Back** buttons.

NOTE: The **Home**, **Forward**, and **Back** buttons are similar to the **Home**, **Forward**, and **Back** controls on a web browser. **Home** returns to the default screen, the original image.

5.2. Use the **Zoom** button to pan and zoom. To activate panning and zooming, press the **Zoom** button, then move the mouse to a desired location in the image.

5.2.1. To pan the figure, press and hold the left mouse button while dragging it to a new position. Release the mouse button, and the selected point in the image will appear in the new position. While panning, hold down the **x** or **y** keys to restrict the motion to the x or y axes, respectively.

5.2.2. To zoom, hold down the right mouse button and drag it to a new location. Move right to zoom in on the x-axis and move left to zoom out on the x-axis. Do the same for the y-axis and up/down motions. When zooming, note that the point under the mouse remains stationary, allowing to zoom in or out around that point. Use the modifier keys **x**, **y**, or **CONTROL** to limit the zoom to the x, y, or aspect ratio preserve, respectively.

5.3. To activate the **Zoom-to-rectangle mode**, click the **Zoom-to-rectangle** button. Place the cursor over the image and press the left mouse button. Drag the mouse to a new location while holding the button to define a rectangular region.

NOTE: The axes view limits will be zoomed to the defined region when the left mouse button is pressed. The axes view limits will be zoomed out when the right mouse button is pressed, placing the original axes in the defined region.

5.4. Use the **Subplot-configuration** tool to configure the appearance of the subplot.

NOTE: The left, right, top, and bottom sides of the subplot, as well as the space between rows and columns, can be stretched or compressed.

5.5. To open a file save dialog, click the **Save** button and save the file in the following formats: .png, .ps, .eps, .svg, or .pdf.

REPRESENTATIVE RESULTS:

A representative analysis was performed on images of dendritic networks in culture. Cells were extracted as described by Baranes *et al*.[16,17]. Briefly, hippocampal cells were extracted from the brains of postnatal rats and cultivated on 2D glass coverslips for 1–2 weeks. The cultures were then fixed and stained through indirect immunofluorescence using an antibody against the dendritic protein marker, microtubule-associated protein 2 (MAP2). Images of dendritic networks were collected using a fluorescence microscope, and 10 images were processed using SOA.

Figure 1 shows a typical SOA workflow for the analysis of a dendritic network. The input is a 2D fluorescent microscope image. Image segmentation is performed in two stages: the first stage is the identification of dendritic branches as lines, and the second stage is merging of relevant lines according to distance and direction criteria determined by the user. After segmentation, spatial information is collected for each identified dendritic branch. Then, the following information is extracted from the data of all dendritic branches in the image: Parallel/Non-Parallel Classification, average length of parallel vs. non-parallel dendritic branches, distance between parallel dendritic branches, and angular distribution.

Figure 2 illustrates the workflow laid out in **Figure 1** applied to a representative 2D image of a dendritic network labeled with a fluorescent anti-MAP2 antibody. A Graphical User Interface (GUI) was developed in which the user can select and upload an image out of a set of files (Step 1). Then, the user can modify the Edges settings (Step 2) and Merge Lines settings (Step 3). Following that, lines representing the identified dendritic branches are detected (Step 2) and then merged, based on the proximity of the segmented lines and their directional differences (Step 3). The GUI enables the comparison of the original image to the segmented image (Step 3) and provides real-time monitoring of the effect of any changes to the segmentation settings. The spatial information collected is then analyzed, and the results are presented as graphs or tables (**Figure 3, Figure 4, Figure 5,** and **Figure 6**)

Figure 3 demonstrates how dendritic branches are classified as growing parallel (**Figure 3A**) and non-parallel (**Figure 3B**). All segments classified as "non-parallel" are not parallel to any other segment. Because dendritic branches do not actually extend in straight lines, there was a need to provide some degree of freedom for the definition of parallel growth. To achieve that, the orientation of a particular branch was measured, and then a range of angles around the measured orientation was permitted for parallelism. This range is fixed for each image and depends on the number of lines detected; however, it cannot exceed 10° (a detailed description of the parallelism sorting process is given in **Supplemental Material**/Analysis, section 1). Then, the fitness of the orientation of all other branches to this angle range was examined. Once the analysis was completed, the number of parallel branches within the tested range was extracted and plotted in a frequency graph (**Figure 3C**).

To understand whether the extent of parallel growth among the dendritic branches is random or directed, the results of the graph in **Figure 3C** were compared to those extracted from simulation of random growth of lines of the same number as those of the dendritic branches in the cultures (**Figure 3D**). SOA then measures the distances among the parallel branches (**Figure 4**) as well as the lengths of parallel and non-parallel dendritic branches. **Figure 5** displays bar graphs of the lengths of the non-parallel dendritic branches (**Figure 5A**) and parallel dendritic branches (**Figure 5B**) and their average lengths. To determine whether preferential growth directions exist, SOA displays a distribution histogram of the growth angles of the dendritic branches (**Figure 6**). Such a presentation allows for rapid identification of preferred growth directions and identification of specific dendritic branches in each group (by ID number) (**Figure 6A**).

FIGURE AND TABLE LEGENDS:

Figure 1: SOA workflow for segmentation and growth direction analysis. Shown are the processing steps of fluorescent images of dendritic networks and data analysis. The 2D image is uploaded, segmented (in two steps: dendritic branches are detected as lines, and then the relevant lines are merged), and the spatial information of each dendritic branch is obtained. The data are collected for all the dendritic branches in the image. Finally, the data are analyzed to give the desired morphological parameters. Abbreviation: SOA = segmentation and orientation analysis.

Figure 1: Example of a workflow using the SOA's GUI. Left column: GUI sections of the workflow. Middle column: image of a dendritic network, processed during the workflow (Scale bar: 20 µm). Right column: magnification of the area marked by a red rectangle in the images of the middle column (Scale bar: 4 µm). Step 1: Selection and uploading of an image. Step 2: The first stage of segmentation is the detection of lines that represent the identified dendritic branches. Step 3: The second stage of segmentation is the proximity-based merger of segment lining in individual dendritic branches. The settings of all steps can be modified. Abbreviations: SOA = segmentation and orientation analysis; GUI = graphical user interface.

Figure 3: Classification of dendritic branch parallel growth vs. random simulation. Dendritic branches of the image in **Figure 2B** identified by SOA were classified as (**A**) parallel and (**B**) non-parallel. (**C**) The number of parallel dendritic branches in each angle range checked was collected and divided into groups of pairs/triples/quartets as explained in **Supplemental Material**/Analysis, section 1. The occurrence frequency of each group is shown in the graph. (**D**) The results of grouping parallel lines extracted from a random line distribution simulation. Abbreviation: SOA = segmentation and orientation analysis.

Figure 2: SOA measurement of the distance among parallel dendritic branches. Shown is an example of how SOA displays the distance among branches grown in parallel. Each dendrite detected receives a unique number (ID number). The SOA measures the minimum distance between each pair of parallel dendrites. A detailed description can be found in **Supplemental Material**/Analysis, section 1. Examples: 1. Dendritic branch ID=2 is at a distance of 60 µm from another branch. 2. Dendrite ID=17 has two parallel branches at distances of 60 µm and 13.7 µm. Abbreviation: SOA = segmentation and orientation analysis.

Figure 5: SOA's display of the length distribution of parallel vs. non-parallel dendritic branches. Shown is a frequency plot distribution of SOA's comparison of the lengths of (**A**) non-parallel and (**B**) parallel dendritic branches, as well as the average lengths. Abbreviation: SOA = segmentation and orientation analysis.

Figure 6: SOA's display of the angular distribution of dendritic branches. (**A**) A graphical representation of each identified line according to its growth angle. (**B**) A display of **A** that allows quick recognition of preferred growth directions. Combining **A** and **B**, each dendritic branch can be allocated to a specific growth direction category. This form of representation of the distribution of dendritic growth directions allows rapid identification of preferred growth directions (**B**: the higher the column, this growth direction is more preferred) and directions in which dendrites do not grow (**B**: at an angle where there is no column, dendrites do not grow in the same direction). Abbreviation: SOA = segmentation and orientation analysis.

DISCUSSION:

Effective strategies for extracting morphological information from 2D images are urgently required to keep up with biological imaging data. Although imaging data can be generated in hours, in-depth analysis of the images takes a long time. As a result, image processing has clearly become a major obstacle in many fields. This is due in part to the high complexity of the data, especially when dealing with biological samples. Furthermore, as many users lack specialized programming and image processing skills, automated tools that allow image processing to be done in an easy and user-friendly manner are needed. That is why SOA is expected to be useful. The representative results demonstrate the automated segmentation that enables identification of the dendritic branches in the image in a few simple steps, despite the complexity of the dendritic network. The work process is simple and intuitive, and diverse spatial information is obtained immediately and without effort. Several algorithms were employed to detect individual dendritic branches in the fluorescence images and conduct multiple analyses on the results. A detailed description of the algorithms can be found in the **Supplemental Material** and information on the system requirements. This paper will briefly present the algorithms and their role in the software.

Segmentation

Segmentation was the most challenging part of this project. In segmentation, a digital image is converted into several segments (image objects). The aim of segmentation is to identify objects in the image and thus make them more understandable and available. Here, segmentation was utilized to identify dendritic branches and isolate them from the background. The image segmentation process was split into two stages: the first stage involved the detection of dendritic branches as lines, and the second stage involved merging related lines based on distance and direction criteria set by the user.

Dendritic information and analysis

Information on the location, angle, and length of each line detected was collected. The software performs the following analyses on the data obtained from all the lines identified in the image: 1. Dendritic Branch Classification (Parallel/Non-Parallel) 2. The average length of parallel vs. non-parallel dendritic branches was measured. 3. Angular distribution measurement and display 4. Distance measurement between parallel dendritic branches

The SOA's user interface allows users to upload an image from the computer's files. It also allows for adjustments of segmentation settings. Because each dendritic network image is unique, we suggest "tinkering" with the settings to achieve the best possible segmentation. The user interface allows for comparison of the original and the segmented images and immediate monitoring of the effect of any change in the settings on the segmentation.

After modifying all settings, SOA creates the final figure of the segmentation that shows each of the identified dendritic branches. The SOA generates graphs of the analyses performed and a .xlsx file with all the data.

The output of the SOA can be utilized as a starting point for the input of tools for further analysis. For example, we are now developing software that calculates the average parallelism value for dendrites that have received various treatments using SOA output on a large number of images.

Summary

SOA is an automated tool for identification, segmentation, and extraction of important morphological information from images of complex 2D line networks and has a user-friendly and intuitive interface. In this work, the use of SOA was introduced through an example of analysis of dendritic networks. SOA can be used for the analysis of other types of 2D cellular networks, such as networks of different neural and non-neural cells, intracellular complex structures such as those generated by the cytoskeleton and non-biological networks, e.g., nanotubes and more. SOA was developed for a very specific purpose, and it is important to know its advantages and limitations. SOA's limitations include the fact that it is only suitable for 2D image analysis and not for 3D image analysis. SOA can only be used to analyze images with objects that are line-like. Moreover, the information obtained from the software is limited to the spatial information of the identified dendrites and to the specific analyses described here. Additional analyses are not performed by the SOA. The software's key advantages are its simplicity and user-friendliness. The software allows complex images to be analyzed quickly and in a few easy steps. Moreover, SOA is flexible and readily adjustable; hence, its analytic capacity can be expanded beyond morphometry and be beneficial for other applications.

ACKNOWLEDGMENTS:

The authors would like to thank Dr. Orly Weiss for the preparation of the culture images.

DISCLOSURES:

The authors declare that they have no competing financial interests.

REFERENCES:

1. Ferrante, M., Migliore, M., Ascoli, G. Functional impact of dendritic branch-point morphology. *Journal of Neuroscience*. **33** (5), 2156–2165 (2013).

2. Spruston, N. Pyramidal neurons: dendritic structure and synaptic integration. *Nature Reviews Neuroscience*. **9** (3), 206–221 (2008).

3. Chklovskii, D. Synaptic Connectivity and Neuronal Morphology: Two Sides of the Same Coin. *Neuron*. **43** (5), 609–617 (2004).

4. Chapleau, C., Larimore, J., Theibert, A., Pozzo-Miller, L. Modulation of dendritic spine development and plasticity by BDNF and vesicular trafficking: fundamental roles in neurodevelopmental disorders associated with mental retardation and autism. *Journal of Neurodevelopmental Disorders*. **1** (3), 185–196 (2009).

5. Irwin, S. Dendritic Spine Structural Anomalies in Fragile-X Mental Retardation Syndrome. *Cerebral Cortex*. **10** (10), 1038–1044 (2000).

6. Kaufmann, W. Dendritic anomalies in disorders associated with mental retardation. *Cerebral Cortex*. **10** (10), 981–991 (2000).

7. Pinchas, M., Baranes, D. Dendritic branch intersections are structurally regulated targets for efficient axonal wiring and synaptic clustering. *PLoS ONE*. **8** (12), e82083 (2013).

8. Cove, J., Blinder, P., Baranes, D. Contacts among non-sister dendritic branches at bifurcations shape neighboring dendritic branches and pattern their synaptic inputs. *Brain Research*. **1251**, 30–41 (2009).

9. Blinder, P., Cove, J., Foox, M., Baranes, D. Convergence among non-sister dendritic branches: An activity-controlled mean to strengthen network connectivity. *PLoS ONE*. **3** (11), e3782 (2008).

10. Glaser, J., Glaser, E. Neuron imaging with neurolucida — A PC-based system for image combining microscopy. *Computerized Medical Imaging and Graphics*. **14** (5), 307–317 (1990).

11. Scorcioni, R., Polavaram, S., Ascoli, G. L-Measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. *Nature Protocols*. **3** (5), 866–876 (2008).

12. Torben-Nielsen, B. An efficient and extendable python library to analyze neuronal morphologies. *Neuroinformatics*. **12** (4), 619–622 (2014).

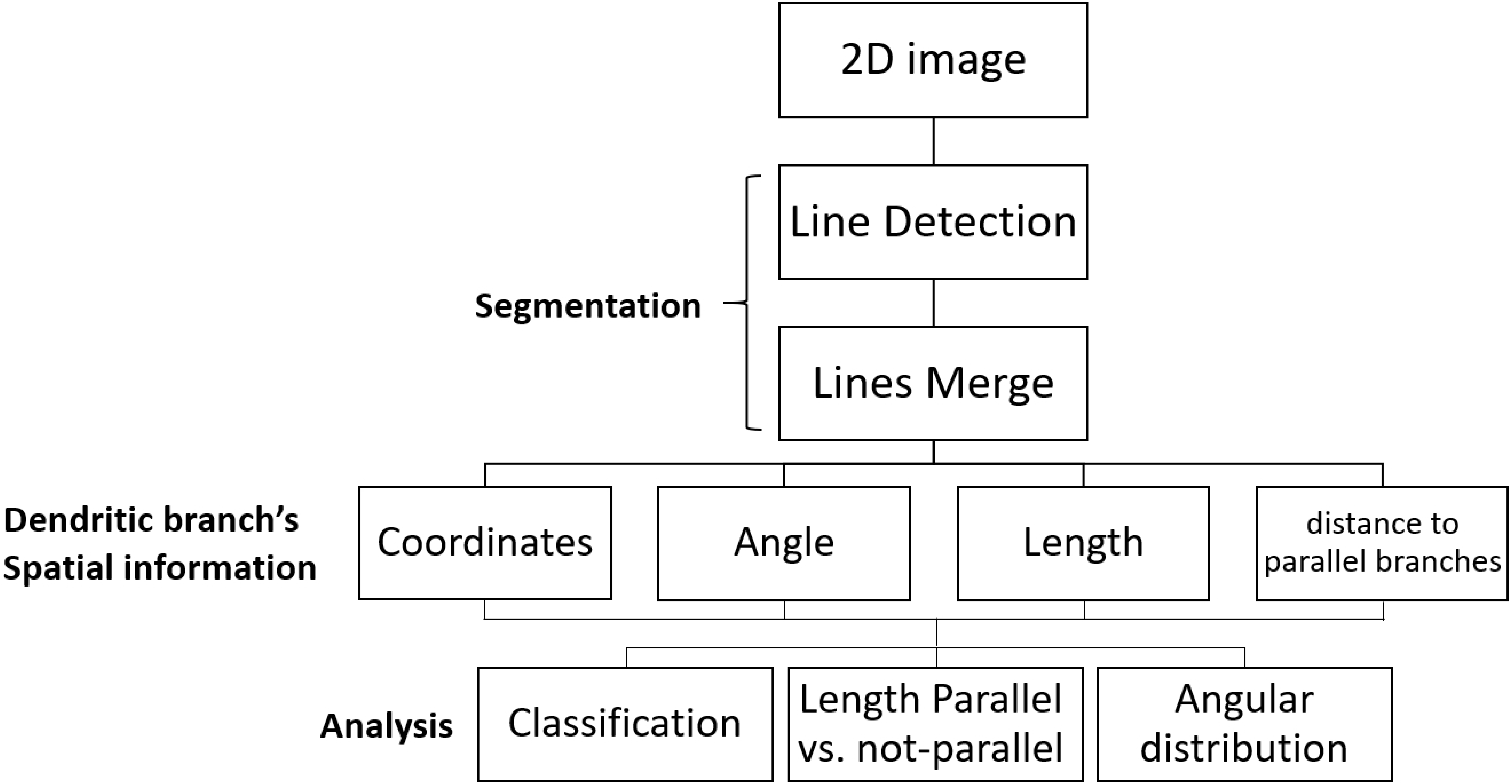
13. Parekh, R., Ascoli, G. Neuronal morphology goes digital: A research hub for cellular and system neuroscience. *Neuron*. **78** (1), 206 (2013).

14. Cheng, J., Zhou, X., Sabatini, B.L., Wong, S.T.C. NeuronIQ: A novel computational approach for automatic dendrite SPINES detection and analysis. *2007 IEEE/NIH Life Science Systems and Applications Workshop*. 168–171 (2007).

15. NIH. ImageJ. Image processing and analysis in Java. https://imagej.nih.gov/ij/ (2021).

16. Peretz, H., Talpalar, A.E., Vago, R., Baranes, D. Superior survival and durability of neurons and astrocytes on 3-dimensional aragonite biomatrices. *Tissue Engineering*. **13**, 461–472 (2007).

17. Morad, T.I., Hendler, R.M., Weiss, O.E., Canji, E.A., Merfeld, I., Dubinsky, Z., Minnes, R., Francis, Y.I., Baranes, D. Gliosis of astrocytes cultivated on coral skeleton is regulated by the matrix surface topography. *Biomedical Materials*. **14** (4), 045005 (2019).



2D image

Segmentation

Line Detection

Lines Merge

Dendritic branch's Spatial information

Coordinates

Angle

Length

distance to parallel branches

Analysis

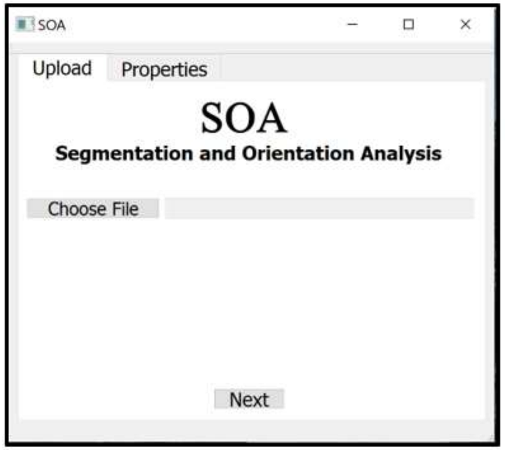
Classification

Length Parallel vs. not-parallel

Angular distribution

Figure 2

Step 1:
Image upload



SOA

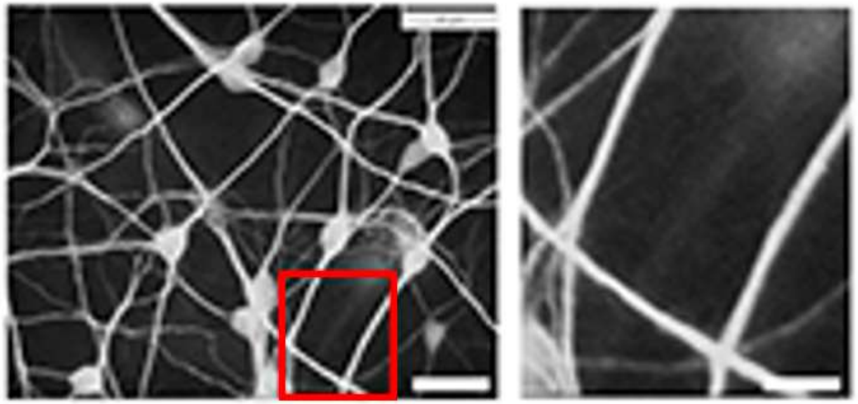
Upload | Properties

SOA

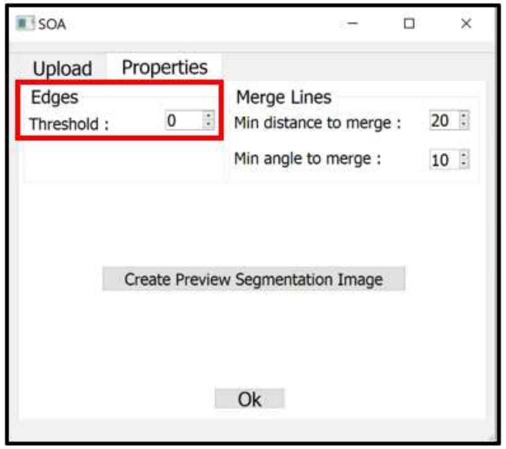
Segmentation and Orientation Analysis

Choose File

Next



Step 2: Segmentation
Line detection



SOA

Upload | Properties

Edges

Threshold : 0

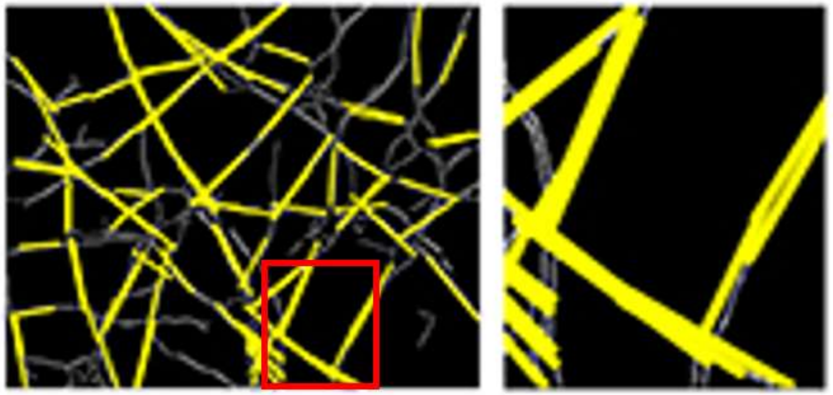
Merge Lines

Min distance to merge : 20

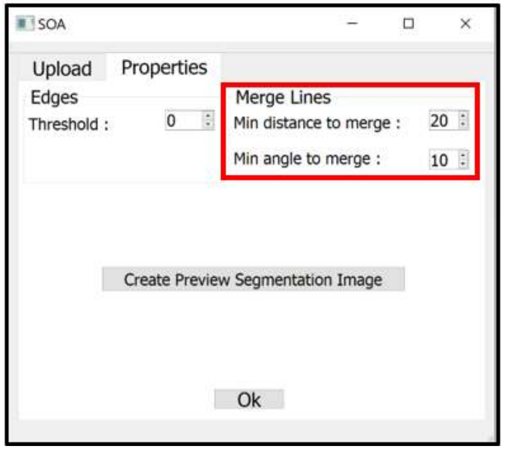
Min angle to merge : 10

Create Preview Segmentation Image

Ok



Step 3: Segmentation
Line merging



SOA

Upload | Properties

Edges

Threshold : 0

Merge Lines

Min distance to merge : 20

Min angle to merge : 10

Create Preview Segmentation Image

Ok

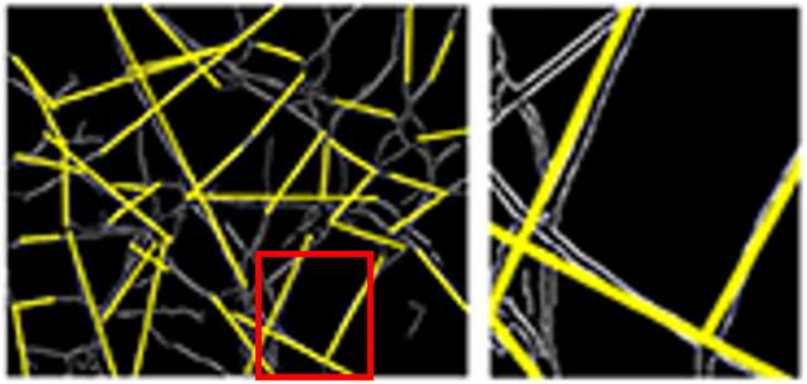
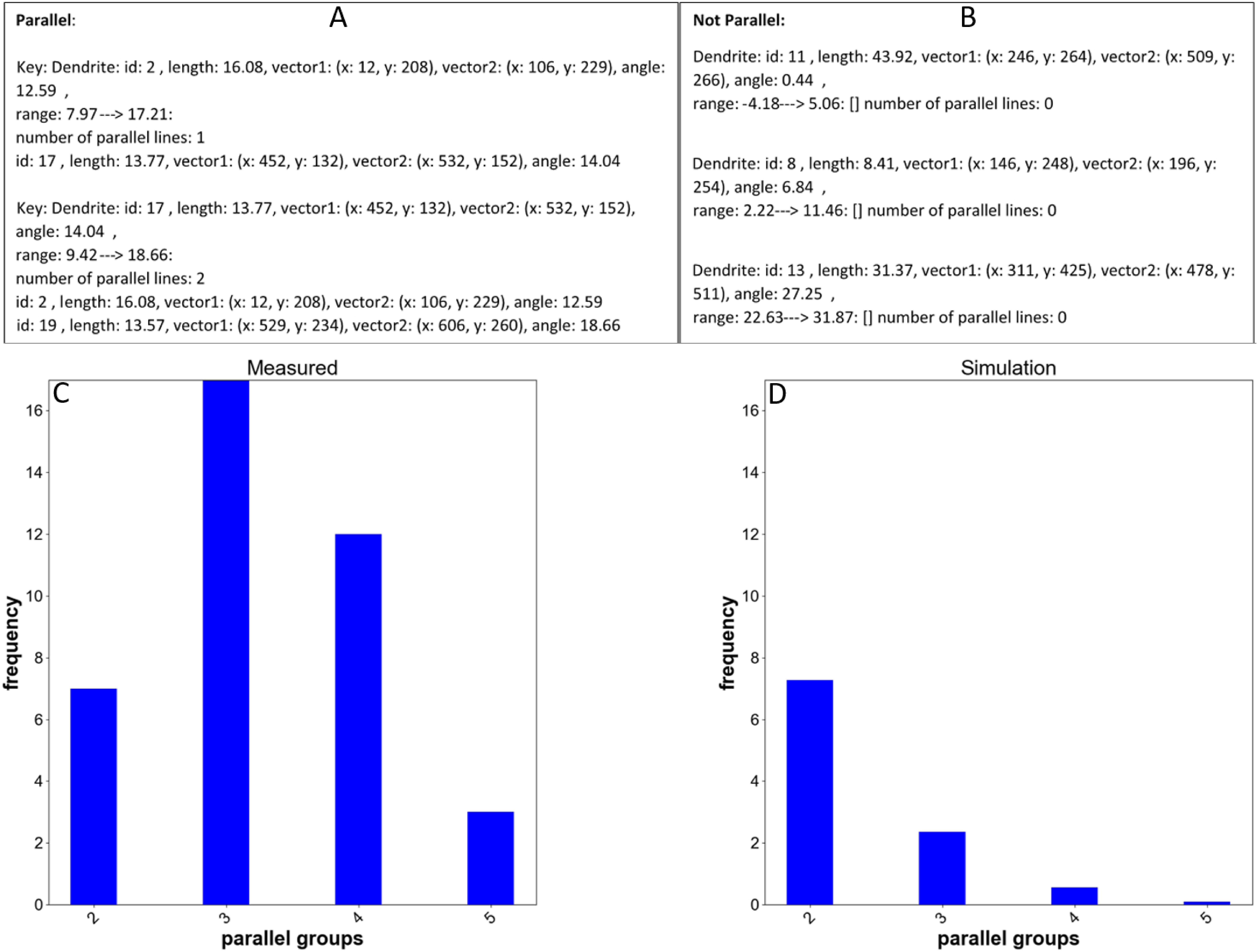


Figure 3



A

Parallel:

```
Key: Dendrite: id: 2 , length: 16.08, vector1: (x: 12, y: 208), vector2: (x: 106, y: 229), angle:
12.59 ,
range: 7.97---> 17.21:
number of parallel lines: 1
id: 17 , length: 13.77, vector1: (x: 452, y: 132), vector2: (x: 532, y: 152), angle: 14.04

Key: Dendrite: id: 17 , length: 13.77, vector1: (x: 452, y: 132), vector2: (x: 532, y: 152),
angle: 14.04 ,
range: 9.42---> 18.66:
number of parallel lines: 2
id: 2 , length: 16.08, vector1: (x: 12, y: 208), vector2: (x: 106, y: 229), angle: 12.59
 id: 19 , length: 13.57, vector1: (x: 529, y: 234), vector2: (x: 606, y: 260), angle: 18.66
```

B

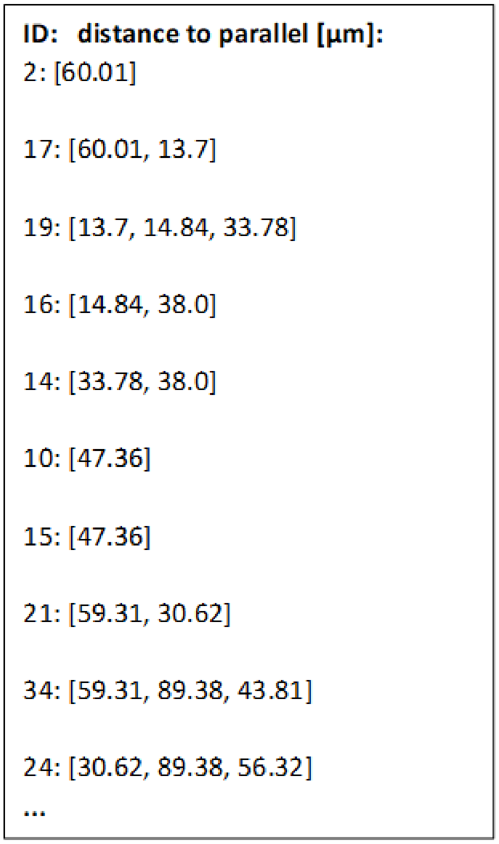
Not Parallel:

```
Dendrite: id: 11 , length: 43.92, vector1: (x: 246, y: 264), vector2: (x: 509, y:
266), angle: 0.44 ,
range: -4.18---> 5.06: [] number of parallel lines: 0


Dendrite: id: 8 , length: 8.41, vector1: (x: 146, y: 248), vector2: (x: 196, y:
254), angle: 6.84 ,
range: 2.22---> 11.46: [] number of parallel lines: 0


Dendrite: id: 13 , length: 31.37, vector1: (x: 311, y: 425), vector2: (x: 478, y:
511), angle: 27.25 ,
range: 22.63---> 31.87: [] number of parallel lines: 0
```

Figure 4



```
ID:   distance to parallel [µm]:
2: [60.01]

17: [60.01, 13.7]

19: [13.7, 14.84, 33.78]

16: [14.84, 38.0]

14: [33.78, 38.0]

10: [47.36]

15: [47.36]

21: [59.31, 30.62]

34: [59.31, 89.38, 43.81]

24: [30.62, 89.38, 56.32]
...
```

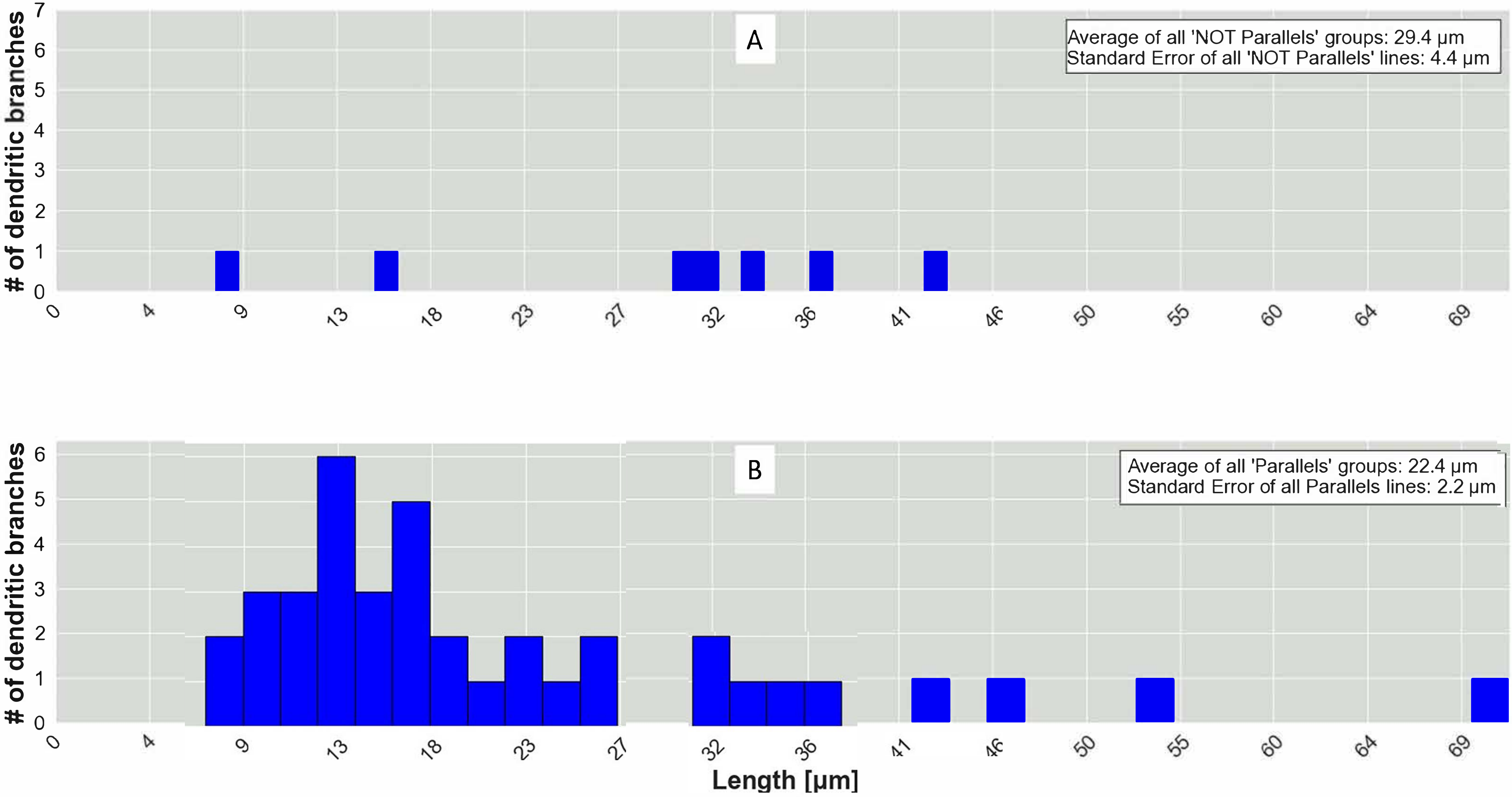

A
Average of all 'NOT Parallels' groups: 29.4 µm
Standard Error of all 'NOT Parallels' lines: 4.4 µm
# of dendritic branches
0
1
2
3
4
5
6
7
0
4
9
13
18
23
27
32
36
41
46
50
55
60
64
69
B
Average of all 'Parallels' groups: 22.4 µm
Standard Error of all Parallels lines: 2.2 µm
# of dendritic branches
0
1
2
3
4
5
6
0
4
9
13
18
23
27
32
36
41
46
50
55
60
64
69
Length [µm]

Figure 6

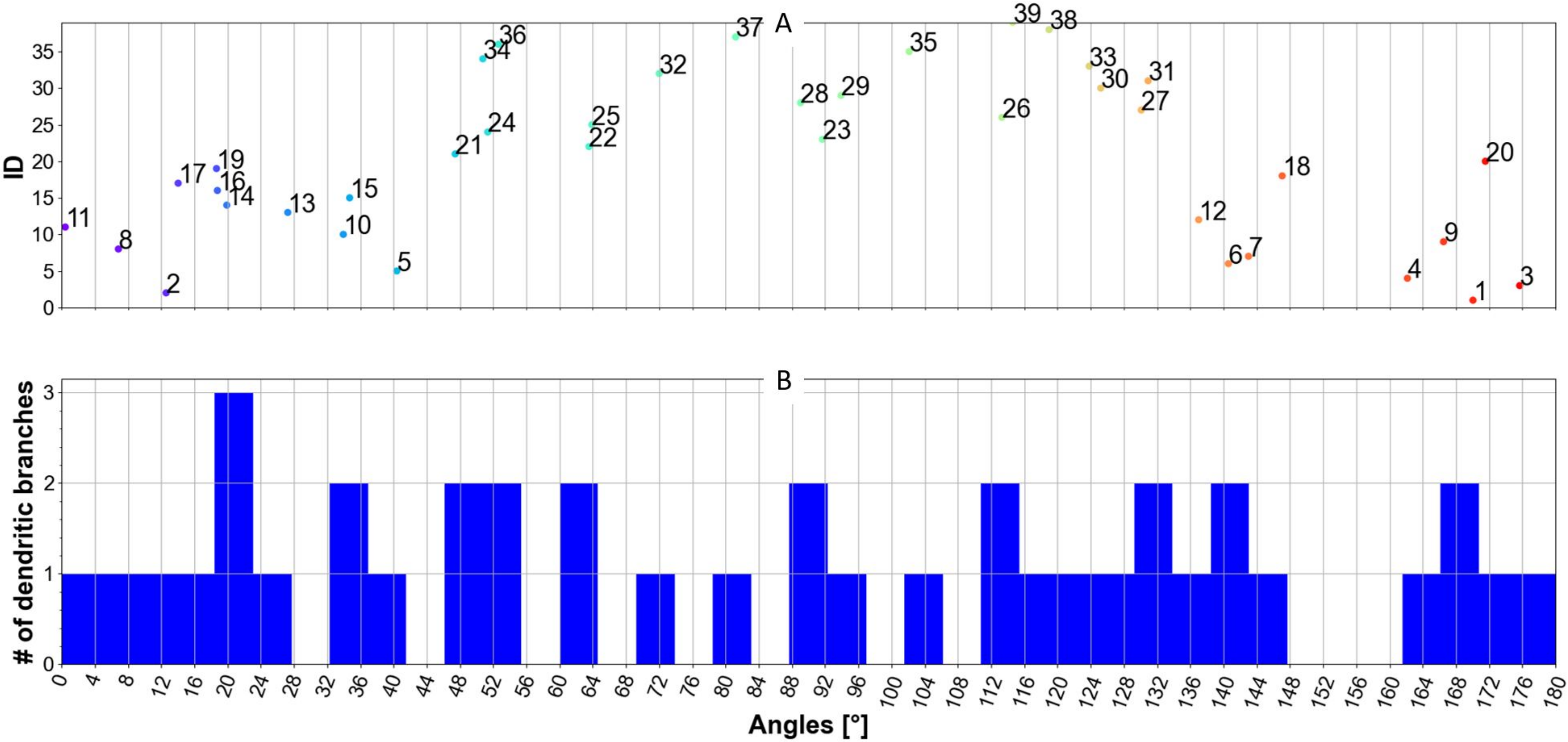

A
ID
0
5
10
15
20
25
30
35
11
8
2
17
19
16
14
13
15
10
5
21
24
34
36
25
22
32
37
28
29
23
35
26
39
38
33
30
31
27
18
12
6
7
4
9
1
3
20
B
# of dendritic branches
0
1
2
3
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100 104 108 112 116 120 124 128 132 136 140 144 148 152 156 160 164 168 172 176 180
Angles [°]

Table of Materials

Click here to access/download

Table of Materials

JoVE_Materials (2).xls

Point-by-Point Response to Reviewers and Editor

In the following, we address reviewer #1 and editor's comments:

We will refer only to the comments reviewer #1 noted as "not-addressed".

Reviewer's and editor's comments in red, our response in black.

Reviewer #1:

Major Concerns:

- The authors refused to perform any kind of comparison to other tools used in the field (like ImageJ). Neither on the level of the actual dendritic classification performance, nor on the level of the software design. The only information that I and future readers of this article have in this regard, is the author's claim that other software either requires a "significant component of computer-assisted manual labor" or that "the quality of the segmentation is less than desirable". No specific examples are provided however that support these statements. Thus, the comparison to other tools remains vague and users just have to test themselves which software works best for their problems.

It is important to mention that the purpose of this work is not to compare the tool we have developed with other tools but to present the tool we have developed and describe how it is used.
For the claim "significant component of computer-assisted manual labor" we have a reference (reference 14).
The claim: "the quality of the segmentation is less than desirable" is based on our experience and impression.

We rewrote the following in *Introduction*: "*However, based on our experience, some manual interventions are still needed during image processing and the quality of the segmentation is less than desirable.*"

- The authors made no efforts to provide more analysis functionalities with their software. In my opinion, SOA might be a proper choice for the detection of dendritic tree segments in 2D images, but not for any further study or analysis. Currently, both the visualization and the descriptive statistics it provides are too basic in order to recommend SOA as an analysis tool. For example, I suggested calculating anisotropy values, which could provide crucial information about the orientation preferences of dendritic branches and would surely be valuable to many users of the software. Also, I suggested providing a 2D visualization of the pairwise orientation differences between dendritic trees. The authors decided to ignore those suggestions. Thus, SOA does not provide many actual functionalities for analysing or studying the morphological structure of 2D dendritic networks beyond basic descriptive statistics on their length, parallelism and orientation.

The goal of the article was to describe the software we developed, as well as its capabilities and applications for our purposes. For our purposes we needed only the analysis we presented in the manuscript.

We rewrote the following in *Discussion/Summary*: " *We developed SOA for a very specific purpose, and it is important to know its advantages and limitations. SOA's limitations include the fact that it is only suitable for 2D image analysis and not for 3D image analysis. The SOA can only be used to analyze images with objects that are line-like. And the information obtained from the software is limited to the spatial information of the identified dendrites and to the specific analyzes we have specified. Additional analyses are not performed by the SOA. The software's key advantages are its simplicity and user-friendliness. The software allows complex images to be analyzed quickly and in a few easy steps. Moreover, SOA is flexible and readily adjustable, so its analytic capacity can be expanded beyond morphometry and be beneficial for other applications.*"

- No concrete statements were added that suggest how the output of SOA may be used in subsequent analyses. Is the output format of SOA appropriate for further analysis via another software? Or are users required to transform the output themselves? At least 1 or 2 examples or statements should be provided here that explain how the software can be used in combination with other tools to overcome the lack of functionalities for network analyses in SOA itself.

The software output are image files (.png) and a xlsx file. If you want to build a tool that will perform additional analyzes based on SOA output, the tool must use these files as a basis for its input.

We rewrote the following in *Discussion*: "*The SOA output can be used as a basis for input of tools for further analysis. For example: Currently, we are working on a software that uses SOA output on a large number of images to calculate the average parallelism value for dendrites that have undergone various treatments.*"

- A discussion of the advantages/disadvantages of the software is still missing, the version requirements of the required software packages are still unknown, and some of the parameter/algorithm descriptions are still not transparent to me. For example, the authors describe in discussion section 1.1.2.4 that the "Threshold 1" (see Fig. 2) is the minimum intensity value at which image pixels are recognized as belonging to an edge. However, in their code on github (https://github.com/inbar2748/DendriteProject/blob/master/main_solution_update.py), this threshold is used as follows: "cv.Canny(blur, 50, self.p_threshold1, None, 3)". The signature of the opencv-python function Canny, however, looks as follows:

"defCanny(image,threshold1,threshold2,edges=None,apertureSize=None,L2gradient=None)". To me, this looks as if self.p_threshold1 is actually used as threshold2 and thus rather controls the upper limit of the edge thinning procedure. In their documentation in section 1.1.2.4, the authors claim that this parameter is set to "None", however. As another example, the Hough transform call made by SOA looks as follows: "cv.HoughLinesP(dst, 1, np.pi / 180, 50, None, 50, 10)". As can be seen, this function has several arguments, all of which are set to certain default values that cannot be changed by the user but have an impact on the detection of dendritic line segments. It should be addressed why these default values are good choices for any kind of 2D dendritic network image. If they are not, it should be clarified what kind of images can be well processed with these settings, or the settings have to be made adjustable. As a final example, the authors use a Gaussian blurring on the input data before feeding it into the Canny algorithm ("blur = cv.GaussianBlur(src, (5, 5), 0)"). However, according to the open-cv documentation (https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html), this step seems to be performed by the Canny algorithm anyway? This is also how I would understand the description provided by the authors in the discussion (section 1.1.2.1)

Based on your comments we have provided additional information on the points you have specified: (1) We have expanded the discussion of the advantages/disadvantages of the software in the *Discussion/Summary* (2) We have added details version requirements of the required software packages in the *Supplemental Material/System*. (3) We expanded on the parameters / algorithms you specified.

As for the Threshold, there was indeed a confusion that caused a mix in the positions of the arguments in the function. This was corrected, and it was found that there was no significant effect on the results following the mistake.

The default value of Threshold 2 has been updated according to Canny's recommendation.

We rewrote the following in *Supplemental Material*: " *In SOA, this function will have only one variable: Threshold1, which we call Threshold. For the Threshold1 parameter we must select different values to match the unique properties of each image. It is worth noting that the lines become more visible as the chosen number decreases, and the higher the number, the less background noise there is. Threshold2 is, on the other hand, calculated automatically based on Threshold1. Threshold2 should be three times higher than Threshold1 according to Canny's recommendation. As long as Threshold1 is less than 85, Threshold2 will receive a value three times that of Threshold1. Threshold2 will have a fixed value of 255 if Threshold1 is greater and/or equal to 85*."

As for the Gaussian blurring, you are correct and it was fixed. Since we wanted to display the image in one of the figures after blurring we used the GaussianBlur function (src, (5, 5), 0) directly on the image.

We fixed it and now Canny's algorithm gets the image we are analyzing and directly and automatically performs the blur as part of the Canny's method.

- No substantial improvements have been made regarding the language. I suggest a complete check for grammatical and spelling errors by a native English speaker.

The article was reviewed by a scientific / linguistic editor.

- The github provides barely any information apart from the raw code. There is no information on how to install the software or the software requirements and the code is not documented at all. As it stands, it is very inconvenient for users to navigate the repository.

We have added a README.DM file which includes instructions for downloading the software and a link to the website where the software is located. And the requirements file has been updated and we have added information on the required versions of all the packages.

We added the following note in Protocol:

"*NOTE: SOA is only compatible with Windows 10 only and Python 3.9.*
SOA is available as an open-source code:
https://github.com/inbar2748/DendriteProject.
At the link there is also a README.DM file that has directions for downloading the software, a link to the software's website, and a requirements file that contains information on the required versions of all the packages.
Additional examples of analysis performed using the software can be seen there as well."

Minor Concerns:

- The cited tool/paper describes a software specific to dendritic spine detection, not for dendritic orientation analysis. Besides, the tool describes automatized spine detection algorithms, and does not seem to require time-consuming, manual labor at all. I asked for a reference that supports the statement in Line 45, which the authors did not provide.

The cited paper claims that computer-assisted neuronal analysis is time consuming:

"*However, the analysis of neuron images generated by either modality has remained largely manual. Even with computer assistance, such analysis is still extremely time-consuming, and subject to user bias, i.e., results cannot be easily confirmed by other investigators.*" (second paragraph of the *Introduction* of reference 14)

- The authors were not able to point out any concrete issue with other dendritic orientation detection software that SOA addresses. Thus, it remains unclear to the reader where they could benefit from using SOA instead of another tool. I asked the authors to provide a single concrete example that demonstrates where tools like ImageJ require manual interventions or struggle to correctly recognize dendrites, but SOA does not. After all, this appears to be the motivation for developing SOA, according to the introduction. The authors refused to provide such an example (see above).

We tried to use the ImageJ for the segmentation of our images but the results we got were not good enough (there was a significant difference between the dendrites that we saw by eye and the dendrites that were recognized by ImageJ). We are not writing a review of ImageJ, so we do not think we need to describe all the problems in using it for our purpose.

- "Neural networks" is a widely used term that can be understood in multiple ways (e.g. artificial neural networks, synaptic wiring diagrams, …). I asked the authors to be more specific at this point, such that the reader immediately knows what the topic of the following sentences is. This was not done by the authors.

The sentence following the sentence the reviewer refers to describes the kind of images relevant to the SOA:

"The SOA can detect various line-like objects in 2D images." Hence, the relevant "neural networks" are 2D neuronal cultures.

- I asked the authors to provide a minor explanation for an important parameter in their software. Instead, they referred to the discussion section. This reference should at least be included in the text, such that readers know where to look for a definition of that parameter and do not just have to wonder whether this will be explained in a subsequent part of the manuscript.

Before the first mention of the "Threshold" we have added a reference to the detailed explanation: "*NOTE: In the SOA Viewer Properties menu bar, change the values in the selected parameters to adjust the segmentation process settings. A detailed description of the parameters, such as the Threshold, is given in Supplemental data section.*"

- The lines I was referring to state that a range of angles can be defined in the GUI that permit to classify dendritic branches as "parallel" vs. "non-parallel". However, in Fig. 2, step 3 and the protocol reference stated above, an angle range is described that is used for line merging. It is not clear to me why these angle ranges should be the same and whether they actually are in SOA. This should be properly described.

You are right. There is a range of angles for the merging of the lines and a range of angles for classifying dendritic branches as "parallel" vs. "non-parallel". The sentence you mentioned was indeed out of place. The sentence was deleted and we rewrote the following:

"This range is fixed for each image and depends on the number of lines detected."

- Unfortunately, I still do not know what Figure 4 shows. It does not seem to be average values but some kind of distance measures (Euclidean distance, I guess?). It is unclear to me what the IDs refer to. Single dendrites, branches, or parallel groups? Also, there exists no legend for Figure 4. I recommend that the authors check the labeling of their figure legends. There are two legends for Fig. 1 and none for Fig. 4.

I do not know why you did not find the legend of Figure 4. The explanation for the figure appears there. Following your comments, we have added details to the description in the legend:

*"**Figure 4. SOA measurement of the distance among parallel dendritic branches.** Shown is an example of how SOA displays the distance among branches grown in parallel. Each dendrite detected receives a unique number (ID number). The SOA measures the minimum distance between each pair of parallel dendrites. A detailed description can be found in Supplemental data /Analysis, section 1. Examples: 1. Dendritic branch ID=2 is at a distance of 60 µm from another branch. 2. Dendrite ID=17 has two parallel branches at distances of 60µm and 13.7 µm."*

- The grouping mechanism/algorithm is still untransparent. For example, it is unclear why the dendrite with ID 2 is just parallel to ID 17, even though ID 17 is parallel to both ID 2 and ID 19. A reference to section 3.1 should be provided here such that the reader can find the information necessary to understand that figure. Also, I asked for a 2D matrix depiction showing the pairwise orientation differences between dendrites (and optimally also which dendrites are grouped as parallel). This has not been provided by the authors.

Since each dendrite's direction is defined by a range of angles and not just one angle, the situation you described may arise, that a dendrite will be parallel to a second dendrite but not to dendrites parallel to the second dendrites.

I added a reference to the detailed description in Supplemental Material/Analysis, section 1.

Figure 6 shows the angular distribution of the branches. This form of presentation describes in a good and simple way the distribution of angles and is, in our opinion, superior to other forms of presentation.

- The authors refused to provide angular distribution plots instead of histograms and also no further statistics such as the anisotropy value that I suggested. The statistics that were reported instead (average and standard deviation of dendritic lengths) might not be well suited, since the histogram in Fig. 5 indicates that the lengths of these branches are not distributed according to a Gaussian distribution.

Editor: Please address this comment on presentation in your figure legend or in the results section.

We have tried different graphs and in our opinion the presentation we have chosen is clear and allows the reader to easily understand which directions are preferred (if any). In the legend of Figure 6 We have added an explanation of how to identify the preferred growth directions and directions in which there is no growth of dendrites:

"***Figure 6. SOA's display of angular distribution of dendritic branches.*** *(A) A graphical representation of each identified line according to its growth angle. (B) A display of (A) that allows quick recognition of preferred growth directions. Combining (A) and (B), each dendritic branch can be allocated to a specific growth direction category. This form of representation of the distribution of dendritic growth directions allows rapid identification of preferred growth directions (Fig. 6B the higher the column, this growth direction is more preferred) and directions in which dendrites do not grow (in Fig. 6B at an angle where there is no column, dendrites do not grow in the same direction).*"

- In lines 150-151 of the original manuscript, it appears to me like the authors claim that this setting can indeed be adjusted. If not, this sentence has to be changed.

You are right. The sentence you mentioned was indeed out of place. The sentence was deleted, and we rewrote the following:

"*This range is fixed for each image and depends on the number of lines detected but it cannot exceed 10 degrees (a detailed description of the parallelism sorting process is given in the Discussion/Analysis, section 1).*"

- The authors refer to section 3.2, but I think they meant to refer to section 3.1. The line parallelism metric (180/n where n refers to the number of lines), appears to be a very liberal definition of "parallel". What are typical values of n? According to Fig. 3, the "number of parallel lines" could be as low as 1 or 2. Clearly, two dendritic branches with an angular difference of 90 are not parallel. Please clarify what the number of lines actually refers to, to make this classification transparent. Also, I think that this classification metric should be somehow adjustable for users.

You are right. The reference should be to section 3.1 (now *Supplemental data /Analysis, section 1).*

Regarding the criterion for parallelism, you're also correct. Because in our images the number of dendrites is high, we did not see the need to set an upper limit for the parallelism range, but in order to adapt the software to images with fewer detected lines, we limited the range of parallelism to 10 degrees (±5 degrees). See previous comment and rewrite in Supplemental data:

"Since line parallelism is measured by comparing angles, we defined parallelism in this study as having a deviation of ±angle(=180/(number of lines)), as long as this deviation does not exceed 5 degrees."

- The authors should provide a proper reference for OpenCV Python, e.g. by referring to their homepage or github.

We added a reference to the OpenCV Python homepage:

"18. Home - OpenCV. OpenCV at <https://opencv.org/> (2021)."

- The arguments of the function are not explained, making it unclear to the reader why (5, 5) and 0 are good default arguments for every possible input image. This is important, since these parameters of the line detection process cannot be adjusted. I assume that (5, 5) represents the variance of a 2D Gaussian smoothing kernel. Not being able to adjust this setting is an issue, because it requires images to have a resolution and level of detail for which this particular smoothing strength is appropriate.

We addressed this issue in our response to your comment in the *Major Concerns*.

- Line 240: The sentence I was referring to has not been changed. I suggest to remove the sentence "It is useful for removing noise, it eliminates high frequency content (e.g: noise, edges) from the image, resulting in clear edges", and instead write "GaussianBlur (src, (5, 5), 0) convolves the image with a 2D Gaussian kernel, thus removing high spatial frequencies from the image.

We rewrote the sentence as you suggested.

- The Hysteresis procedure is, however, a central process of the Canny edge detection method (https://en.wikipedia.org/wiki/Canny_edge_detector). Also, it does require two thresholds to adjust, whereas SOA only allows adjusting the lower threshold for edge detection. This way, it seems like users can only adjust which edges are to be definitely removed, but not how strong edges should be to be definitely kept. This limits the flexibility of the software the options users have to optimize the edge detection for their specific input images.

We addressed this issue in our response to your comment in the *Major Concerns*.

- The authors have not specified the version requirements of these packages. Can all Python packages listed above have the most current version? Or are specific versions required? Does SOA run for all versions of Python 3 or just specific ones?

We added the version requirements for all the software and packages in *Supplemental Material/System*:

*"**System***

SOA runs on Windows 10 using Python 3.9 with the following main packages:

- *OpenCV library 4.5.2.54 - designed to solve computer vision problems.*
- *NumPy 1.20.3 - fundamental package for scientific computing library.*
- *Matplotlib 3.4.2 - a Python 2D plotting library.*
- *PyQt5 5.15.4 - to manage the GUI.*
- *Xlsxwriter 1.4.3 - a Python module for creating Excel XLSX files.*
- *Seaborn 0.11.1 - Python's Statistical statistical Data data Visualization visualization library.*
- *Scipy 1.6.3 - Python-based ecosystem of open-source software for mathematics, science, and engineering.*
- *matplotlib-scalebar 0.7.2 - artist for matplotlib to display a scale bar."*

Editorial Comments:

1. The organization of sections has to be according to JoVE’s template but concepts and technical terms must still be explained before describing them further.

The organization of the sections in the article is according to JoVE’s template and all the concepts and technical terms are explained (or a reference to the explanation is given) before describing them further.

2. Please highlight up to 3 pages of the Protocol (including headings and spacing) that identifies the essential steps of the protocol for the video, i.e., the steps that should be visualized to tell the most cohesive story of the Protocol. This will ensure that filming will be completed in one day. Remember that non-highlighted Protocol steps will remain in the manuscript, and therefore will still be available to the reader.

Done.

3. Please revise the text, especially in the protocol, to avoid the use of any personal pronouns (e.g., "we", "you", "our" etc.).

Done.

4. If you want to keep this as a step, consider something like “Change the parameters to reach …”

   Otherwise, make this a note and add it to the previous note on line 122 (you cannot have multiple notes between steps.

Done.

5. Please revise the text, especially in the protocol, to avoid the use of any personal pronouns (e.g., "we", "you", "our" etc.).

Done.

6. Please ensure that all text in the protocol section is written in the imperative tense as if telling someone how to do the technique (e.g., “Do this,” “Ensure that,” etc.). The actions should be described in the imperative tense in complete sentences wherever possible. Avoid usage of phrases such as “could be,” “should be,” and “would be” throughout the Protocol. Any text that cannot be written in the imperative tense may be added as a “Note.” However, notes should be concise and used sparingly. Please include all safety procedures and use of hoods, etc.

Done

7. If you want to keep this as a step, consider something like “Change the parameters to reach …”. Otherwise, make this a note and add it to the previous note on line 122 (you cannot have multiple notes between steps).

Done.

8. The Protocol should be made up almost entirely of discrete steps without large paragraphs of text between sections. Please simplify the Protocol so that individual steps contain only 2-3 actions per step and a maximum of 4 sentences per step.

Done

9. Rephrase to remove all personal pronouns esp in protocol (no “you”, “we” “our” …).

Done

10. Please add information to the Table of Materials before uploading it. It could serve as a handy reference with all software programs, mice, reagents, plasticware, glassware, solvents, equipment used in this protocol.

There were no mice, reagents, plasticware, glassware, solvents or any other equipment used in this protocol. The Table of Materials contains only software programs.

11. If these involve actions (button clicks and other actions: typing command lines…), consider moving this information into the protocol, writing the steps in imperative tense. This numbering you have used here is for protocol steps. For the discussion, please revise the Discussion to explicitly cover the following in detail in 3-6 paragraphs with citations:

a) Critical steps within the protocol
b) Any modifications and troubleshooting of the technique
c) Any limitations of the technique
d) The significance with respect to existing methods
e) Any future applications of the technique

Done

Thanks to the reviewers and editor for their hard work and insightful comments; it's apparent that the article would have been less good without their input.

Sincerely,

Dr. Refael Minnes (corresponding author)

Department of Physics, Ariel University.

40700, Ariel, Israel

Cell: (972)-54-6254094

Office: (972)-3-6453140

Email: refaelm@ariel.ac.il

Supplemental Coding Files

Click here to access/download

Supplemental Coding Files

Supplemental Material.docx