Long-term day-by-day tracking of microvascular networks sprouting in fibrin gels: from detailed morphological analyses to general growth rules.

Katarzyna O. Rojek[1], Antoni Wrzos[2], Stanisław Żukowski[2, 3], Michał Bogdan[1], Maciej Lisicki[2], Piotr Szymczak[2], and Jan Guzowski[1]

[1]Institute of Physical Chemistry, Polish Academy of Sciences, Warsaw, Poland
[2]Institute of Theoretical Physics, Faculty of Physics, University of Warsaw, Warsaw, Poland
[3]Laboratoire Matière et Systèmes Complexes (MSC), UMR 7057, CNRS & Université Paris Cité, Paris, France

Comparative Study of Bead Sprouting Morphology Analysis Tools

1 Introduction

The purpose of this supplementary information is to offer a comprehensive comparison of the tool we developed for the analysis of sprouting angiogenesis with existing tools. The primary aim of this comparison is to evaluate how well these available tools, in the forms they were originally shared, performed in our specific case. It's important to note that our intention is not to criticize the presented tools. It is entirely possible that there are optimal ways to adjust their parameters, of which we may not be aware, which could lead to improved performance.

Additionally, we want to clarify that we were not aware of the existence of the majority of these tools when we commenced this project in late 2020. At that time, it was challenging to anticipate the specific difficulties we would encounter. Consequently, we decided that developing our own software from scratch was the most suitable approach.

We will compare our tool with three others: two open-source options— the Sprout Analysis plugin for ImageJ [1] and the SproutAngio tool written in Python [2]; and one paid tool, IKOSA, which utilizes the machine learning model Spheroid Sprouting Assay v2.1.0 [3]. The tests for IKOSA were conducted using a free trial version. Our starting point for all these tools is an 8-bit max-pooled (projected) 2D image, as shown in Figure 1, selected from our dataset. This image was not chosen with a bias towards any particular tool but rather as a challenging test case that may pose difficulties for all of the tested softwares.

All processing will be carried out using a single core of an AMD Ryzen 5 5600x processor. It's worth noting that our software allows us to run multiple tasks in parallel, which provides an advantage when dealing with larger datasets containing numerous images. We will evaluate the tools in several categories, including execution time (estimated order of magnitude, as obtaining exact values in some cases may not be feasible), the quality of results in terms of images and metrics, and the overall user experience (ease of use, challenges encountered, strengths, and weaknesses).

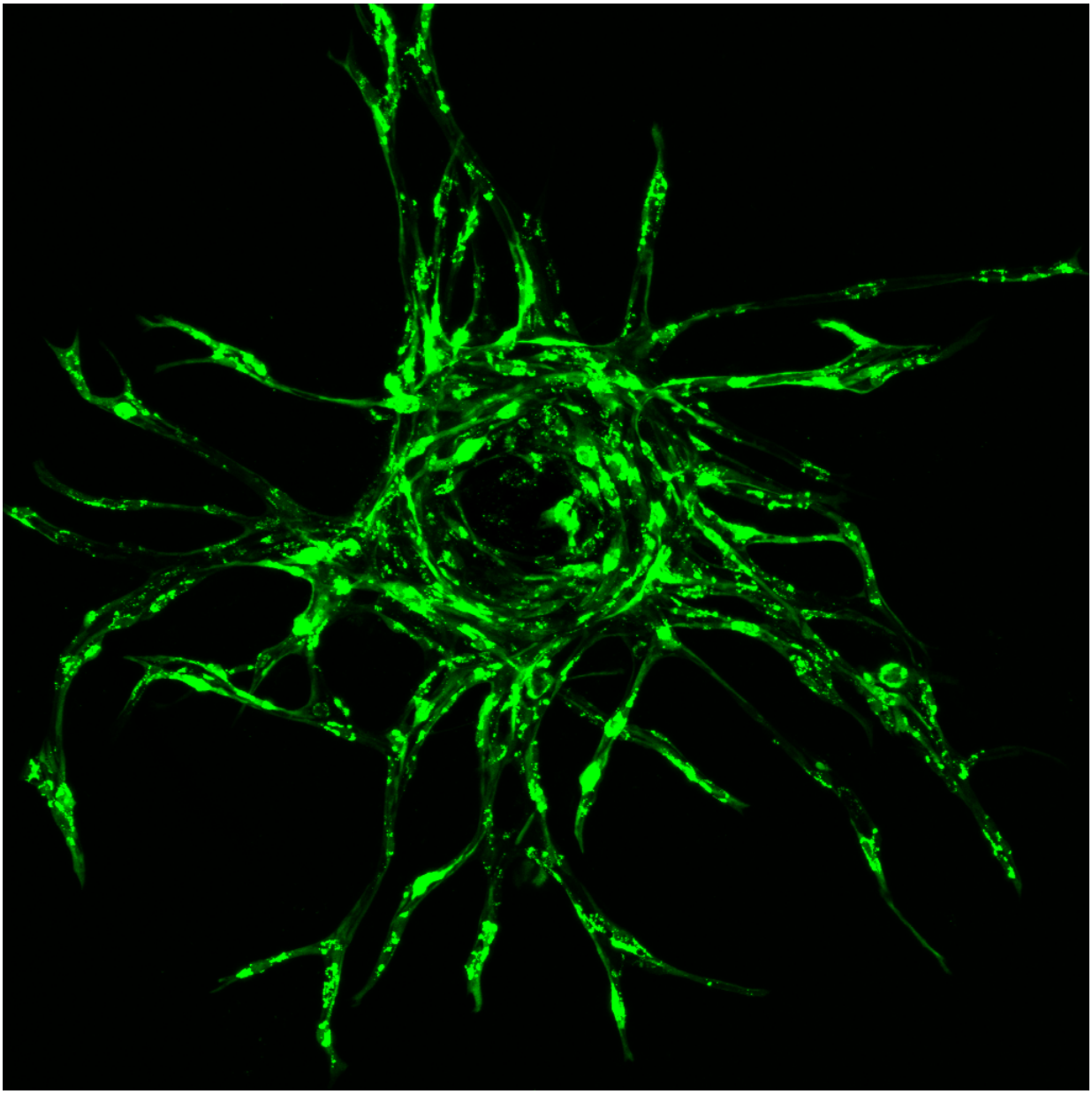

Figure 1: Endothelial cell bead sprouting on day 9 under the influence of a 50 ng/ml VEGF concentration.

2 Proposed software results and performance

2.1 Timing

We conducted timing measurements using the magic function %time in Jupyter Notebook. The computation process took approximately 120 milliseconds when including full serialization, which encompasses saving the segmentation mask (Figure 2), the skeleton mask (Figure 3), and pickling the graph object (pickling refers to the Python operation of saving custom objects in binary format). This approach also allows for convenient access to all necessary data from memory after the computation.

Furthermore, we offer the option to run the software with additional debugging graph information, such as images displaying graph connectivity, the bead mask, skeleton, and bifurcation angles (Figure 4). However, it's worth noting that due to the relatively slow font rendering, the computation time increases to approximately 1 second.

2.2 Graphical output

Upon examining Figure 2, it becomes apparent that part of the structure is being cut during segmentation, resulting in the incomplete reconstruction of the entire network. This truncation occurs because

the largest connected component is selected. While it's possible to omit this step, doing so would leave unwanted disconnected portions in the image. Figure 3 displays the full skeleton prior to the pruning of excessively short branches. The graph with connectivity is depicted in Figure 4.

It's worth noting that there is a specific case to consider, where a junction between branches b181 and b191 at the bottom of the image is missing (as seen in Figure 4). This can be attributed to a rare case, which is elucidated using Figures 5 and 6. Junctions j83 and j84 (depicted in Figure 5) are situated extremely close to each other, and the software does not allow for branches shorter than one pixel. Consequently, after deleting branch b184, junction j83 is regarded as a redundant tip junction (highlighted in red), and junction j84 between branches b190 and b188 is merged (appearing in violet). This situation is more clearly illustrated in Figure 6, where the involved branches are marked in cyan on the skeleton mask prior to pruning excessively short branches. It's important to emphasize that such cases are relatively infrequent, thus they have a small impact on the overall statistics.

2.3 Evaluated metrics

The computed metrics are provided in Listing 1, presented in μm units where applicable. These metrics represent the direct output from the code. The 'sm' (single metrics) discussed in this article include: area A, area_circle A_C, total_length L, r_max (maximum tip extent from the bead r_m), area_by_length λ, nof_primary_branch N_{pb}, and bifurcation_generation G.

Although additional single metrics are available, many of them can be derived from the aforementioned metrics or from histogram metrics 'hm'. Some metrics are applied to the graph before lifting, such as nof_tips. We have chosen to retain these metrics in this form, as further code development will require careful planning. Relevant histogram metrics 'hm' include:

- length_rank2 L_{bif}: Lengths of bifurcated segments, which is important and includes primary branches where the bifurcation condition is met.
- length_rank1: Lengths of tip segments L_{tip}. The number of entries in this statistic provides the correct count of tips N_{tip} on the pruned graph.
- length_primary: Lengths of the primary branches.
- angle ϕ: Bifurcation angles.

Please note that some entries were skipped or not discussed for the sake of clarity. Additionally, it's important to acknowledge that the sum of the number of entries from length_rank1 + length_rank2 + length_primary may not equal nof_branch in general, as length_rank2 can contain some primary branches. However, we are aware of small defect related to the imperfect detection of very short primary branches close to the surface of a bead, which may result in a minor statistical error in the number of segments.

Listing 1: Metrics computed using the proposed software relevant to the benchmark network.


```
{'sm': {'area': 171068.19032499997,
  'area_circle': 104121.75269999998,
  'total_length': 11424.855072122935,
  'length_avg': 89.96819804029533,
  'length_std': 81.2966378986271,
  'r_max': 657.1432145658965,
  'r_avg': 280.8429672447792,
  'r_std': 219.42989478637466,
  'tortuosity_avg': 1.101993464968535,
  'tortuosity_std': 0.1688595066182397,
  'nof_branch': 84,
  'area_by_length': 14.973335700547537,
  'lacunarity': 7.321845364513845,
  'nof_primary_branch': 21,
  'nof_tips': 47,
  'bifurcation_ratio': 1.1622714288988771,
  'bifurcation_generation': 2.162271428898877},
 'hm': {
```

```
'length_rank2': array([ 29.90467604,  65.95821823,  17.6591901 ,  64.23562656,
        14.29307585, 241.34242617,   2.51      ,  48.78192239,
        84.1003026 ,  22.98370417, 152.08471172,  18.08983802,
       262.88756855, 129.45868281, 141.91949271,  99.80627448,
        16.4041901 , 100.75676042, 249.45578854, 191.9298345 ,
        73.63483626, 178.83433073, 135.91206302,  59.89854219,
        72.23321823,  24.66935208,  18.60967604, 138.33287291,
        60.29224635, 188.93175866,  12.6391901 , 104.48481666,
         7.83451406, 227.0225276 , 167.34967252, 129.46386439,
        39.17257031,   3.765     , 102.40546458, 158.57595052,
        58.12370417,  11.295     ,  97.04400677]),
'length_rank1': array([ 68.86192239,  63.66354219,  80.2461125 , 110.88595052,
        51.41805625,  76.48629408, 111.27965469,  81.80562656,
        78.29289427,  57.7453026 ,  49.94773229,  61.36886615,
       125.53060521,  62.6761125 , 225.01706302, 127.77303489,
       160.00933073, 185.94898541,  38.65273229,  43.45740833,
        49.30176042, 398.84059993, 348.99851731]),
'length_primary': array([115.42305625,  24.97386615,  39.04643646,  20.25838021,
         8.56967604,   9.30483802,  16.61951406, 263.97430937,
        94.57095052,  11.07967604, 132.09999824,   4.80467604,
        23.10983802,   4.28483802,  47.86838021, 120.53224635,
        13.805     ,  13.37435208,   9.08951406, 153.87576718,
        61.11659844]),
'r': array([631.06661396, 609.19230595, 569.85845128, 525.95431441,
       463.37210544, 438.64180036, 300.23629161, 364.46245818,
       325.10556386, 291.35737922, 288.55449507, 347.17256246,
       523.75767703, 397.50230569, 220.10853913, 214.72239479,
       262.80191447, 393.36192756, 545.07035401, 243.93855686,
       596.56210817, 349.00058499, 617.95211081, 412.98844687,
       657.14321457, 468.57068304, 482.39202795, 516.92747695,
       248.83520214, 209.19385657, 320.82130342, 448.86386915,
       414.10055765, 243.18843255, 388.5902364 , 331.9706301 ,
       208.87362806, 343.26028062, 426.0572512 , 343.6454924 ,
       405.66795181, 424.66500032, 493.39858543, 610.22689276,
       397.72215276, 490.49953499, 598.41921687]),

'angle': array([107.28149837,  49.69912247,  90.9640207 ,  28.52134122,
        32.67823737,  69.00471432,  64.94831328,  38.16055155,
        40.29999669,  85.3768021 ,  89.97647957,  33.4165814 ,
        52.84170322,  78.22425844,  45.71318864,  59.85861445,
        92.07163954,  88.99491399,  96.42125961,  43.57958006,
        46.06865378]),}}
```

Figure 2: Segmentation mask of the endothelial cell network.

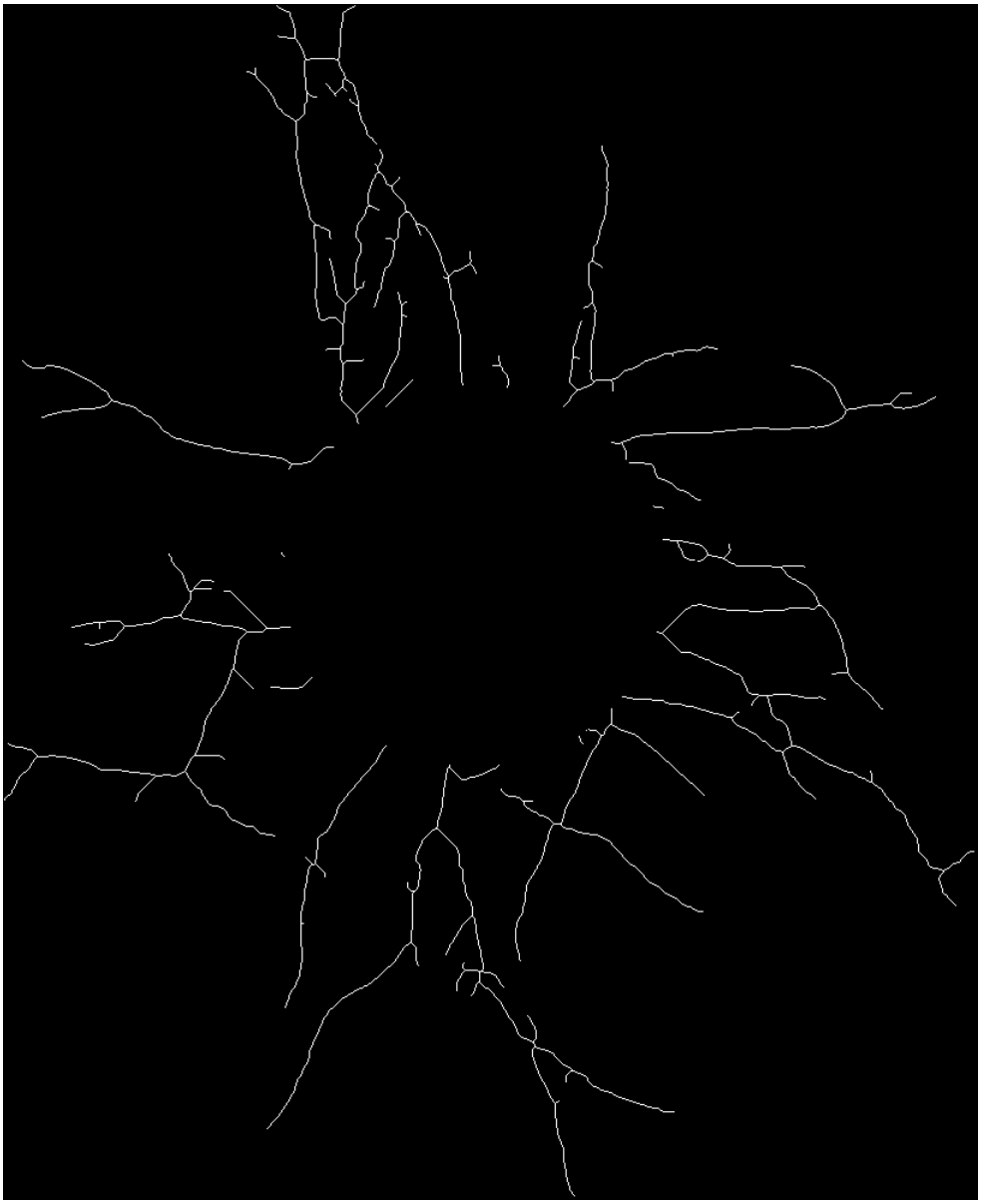

Figure 3: Skeleton of the network with a central bead mask applied. This skeleton includes all segments, including those that are later pruned.

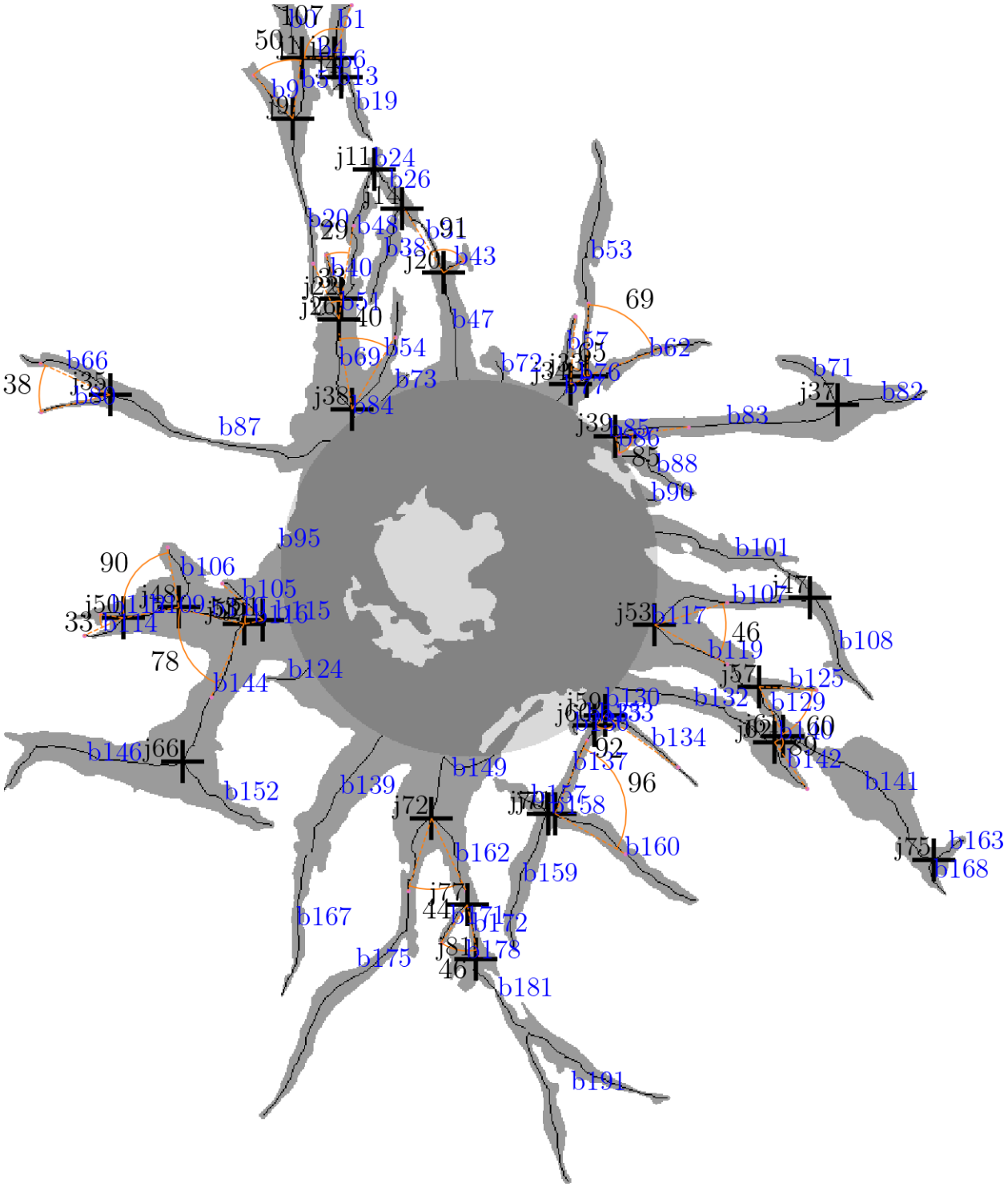

Figure 4: Graph presenting connectivity, bifurcation angles, segmentation mask, pruned skeleton and bead mask.

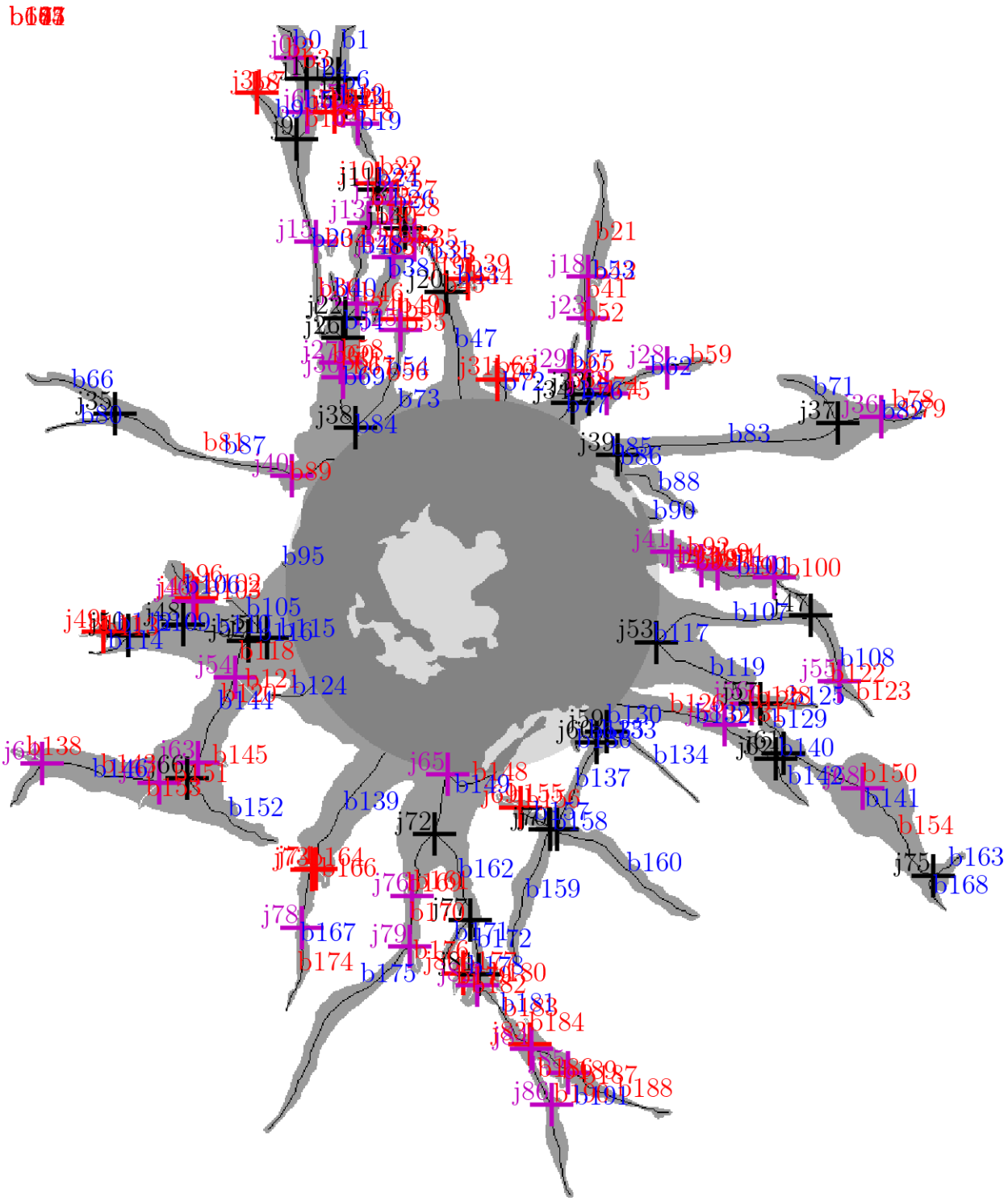

Figure 5: Graph during the lifting (pruning) process, where some branches vanish from the connectivity (indicated in red) and are replaced by the blue ones. Red junctions represent redundant tip junctions, while violet junctions indicate the merging of branches.

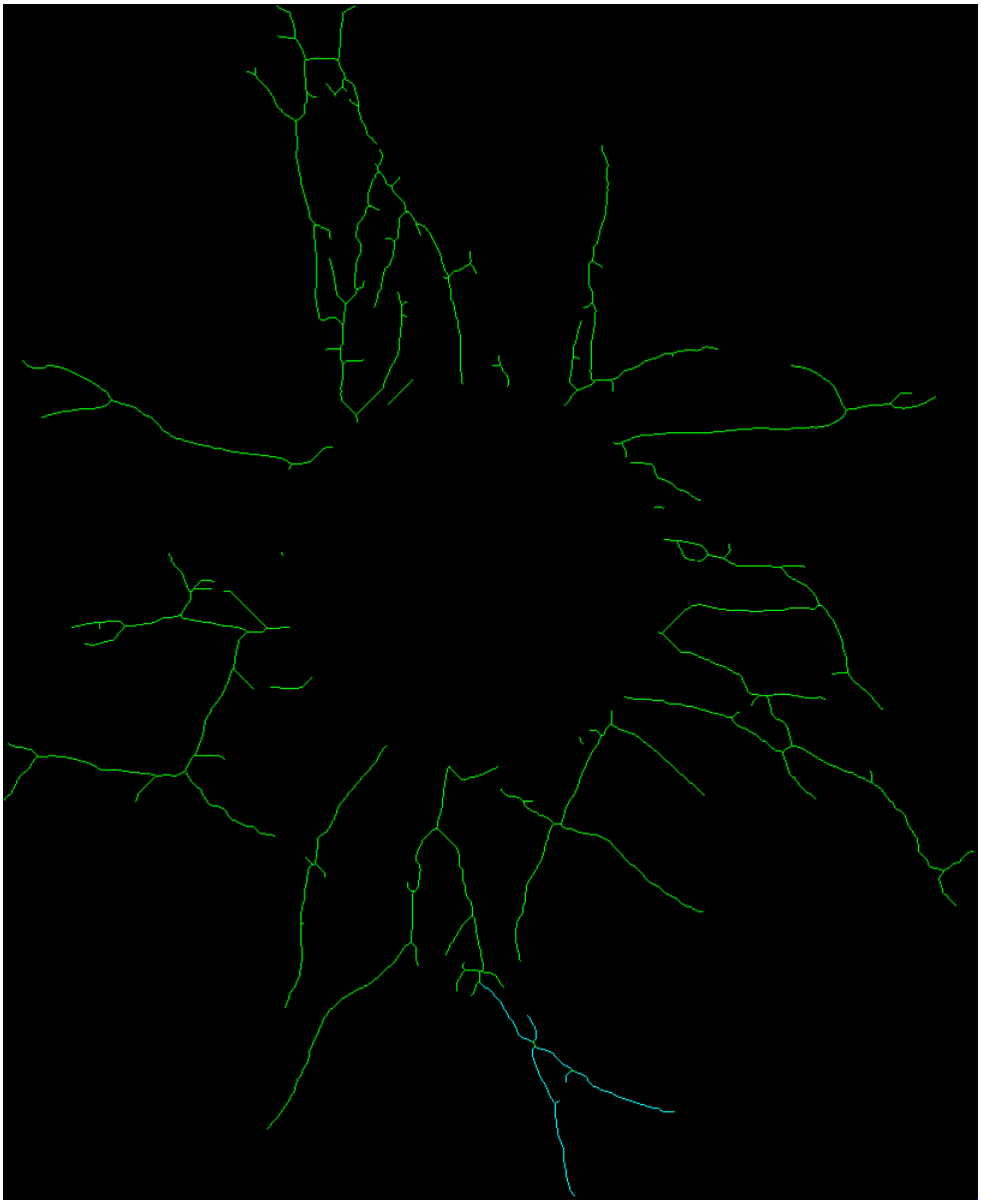

Figure 6: Skeleton with the cyan part where one junction is absent.

3 Sprout Analyzer plugin for ImageJ

3.1 Timing

Measuring the execution time of this software posed a challenge, as its primary audience comprises GUI users. Nonetheless, it exhibited a relatively responsive performance. It is worth noting that a significant portion of the total time consumed could be attributed to rendering and launching ImageJ windows, which renders a fair comparison impractical.

A rough estimate, favoring the Sprout Analyzer, suggests that the software operates within a similar time frame for computational tasks as the our software. However, when considering the entire rendering process, the complete operation takes a few seconds.

3.2 Graphical output

The results obtained from the analysis are depicted in Figures 7, 8, and 9. The segmentation results (Figure 7) bear a resemblance to the outcomes produced by our software, which is a reasonable observation. However, there are certain aspects that could be improved upon: a portion of the structure is truncated, some disconnections are present, and certain gaps are left uncovered, which may lead to the formation

of artificial loops in the skeletons. Depending on subsequent processing steps, these results might still be valid.

Furthermore, the software generates a nuclei mask (Figure 8, a feature not included in our software. This mask allows for the measurement of various metrics within the most densely stained regions of the cells, typically corresponding to internal cell organelles.

Regrettably, the bead mask (Figure 9) appears to be an invalid result, despite our attempts to adjust the initial parameters for bead mask identification. It is possible that this issue arises from our misinterpretation or misuse of the software's functionality.

3.3 Evaluated metrics

The results generated by the software are presented in Table 1. Although the names of the metrics imply that these are related to sprouts, it appears that they are more pertinent to the nuclei image (as depicted in Figure 8). When considering the results in the context of a nuclei image, they seem plausible; however, they are not applicable to our specific dataset.

This discrepancy may stem from a misunderstanding of the software's functionality on our part, but it is more likely that the software is not producing the correct output for our data. This discrepancy could be attributed to the option within the software to independently label channels for the sprout and nuclei images, while our dataset consists of only one channel.

n(beads)	1
n(sprouts)	6
n(cells)	233
Total sprout area (pixels^2)	255
Total network length (pixels)	706.35238
Average sprout length (pixels)	117.7254
Average sprout width (pixels)	0.36101
Average junctions per sprout	0
Cell density (1/pixels^2)	0.91373

Table 1: Metrics from the Sprout Analyzer ImageJ.

3.4 User experience

In general, utilizing this software was straightforward. The installation process merely required ImageJ, followed by the download of the plugin via the built-in package manager. Conducting the analysis itself was intuitive, involving a three-step process during which users could adjust various parameters. While we initially used default settings, we later engaged in some experimentation to enhance results.

The software features a simple graphical user interface (GUI) that enables users to view the outcomes of the analysis, including computed metrics. In its current form, it is tailored more towards individuals without programming backgrounds who intend to analyze a limited number of images. For more extensive analyses, it becomes necessary to develop a macro that can process data using the plugin, a task that may pose some difficulty.

Furthermore, it's important to note that the software is both free and open-source, making it advantageous for those interested in extending its functionality. Regrettably, in our specific case, the software did not function as expected. Additionally, making custom extensions would be challenging due to our limited experience in crafting ImageJ macros.

Figure 7: Segmentation mask of the endothelial cell network from the Sprout Analyzer ImageJ.

Figure 8: Nuclei mask obtained from the Sprout Analyzer ImageJ.

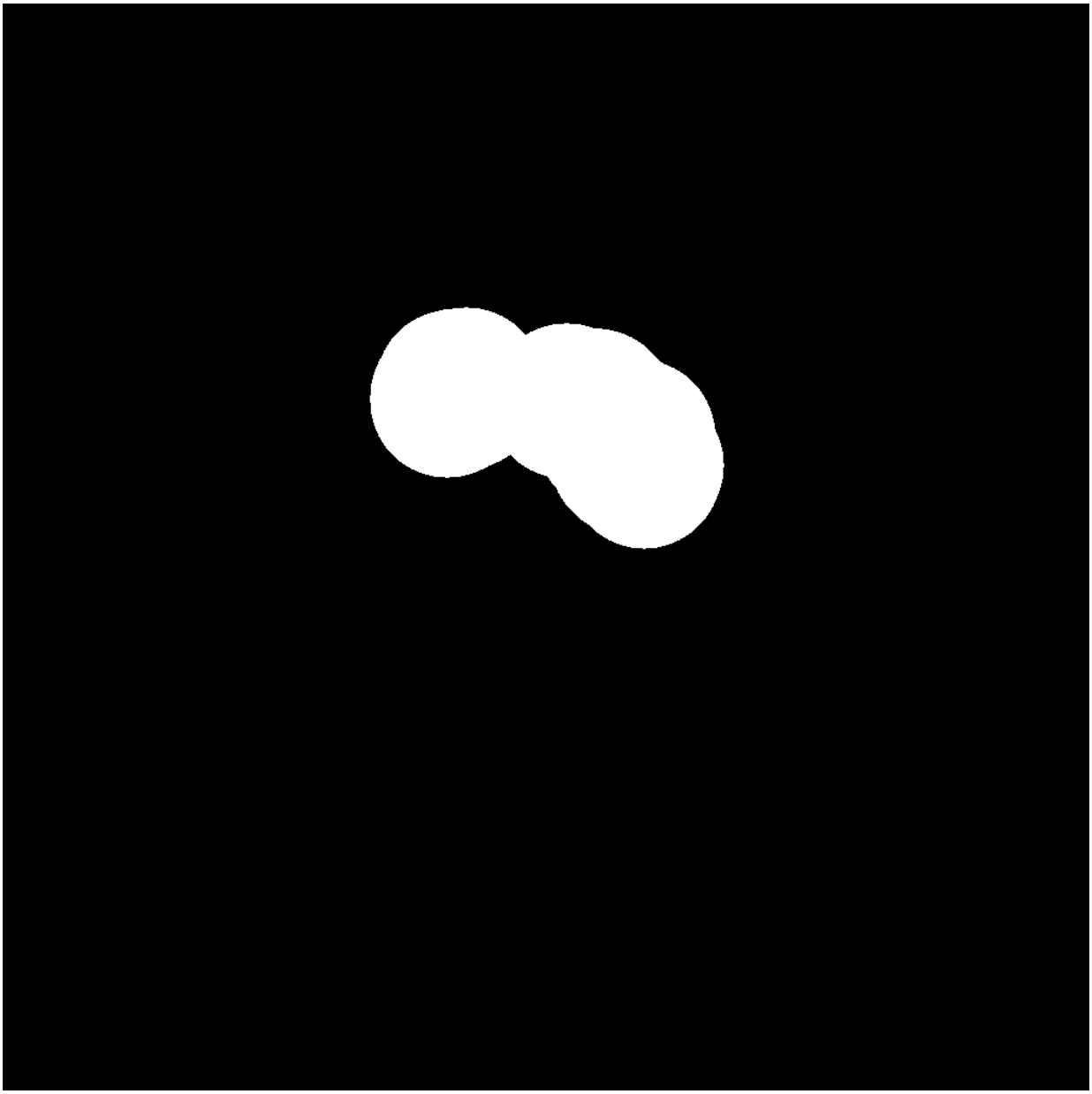

Figure 9: Bead mask returned by the Sprout Analyzer ImageJ. The mask does not match the original bead.

4 IKOSA Spheroid Sprouting Assay v2.1.0

4.1 Timing

Similar to the case with Sprout Analyzer ImageJ, precise timing in this application, which is web-based and primarily designed for an enhanced user experience through its graphical user interface (GUI), proves to be challenging. Several factors necessitate consideration, including the time required for navigating through various windows, data submission to the queue, potential wait times in the queue for resource allocation, and the actual computational processes, followed by the transmission of results.

For a single image, the elapsed time ranged from several seconds to over a dozen seconds after initiating data submission for computation. However, it remains challenging to precisely gauge the duration of the processing phase, as it can be influenced by various factors, including internet connection speed. Consequently, a detailed timing analysis falls beyond the scope of our capabilities.

4.2 Graphical output

The segmentation results are presented in Figure 10. In all software solutions, including this one, complete network reconstruction was not achieved. However, the software effectively assigns distinct colors and numerical labels to the sprouts. In some instances, there may be debate regarding the accurate division of specific sprouts, but it's worth noting that the network under consideration is not straightforward. In the region of sprout segmentation, the software correctly identifies skeletons, and the bead mask is accurately detected.

4.3 Evaluated metrics

Results are presented in the Table 2. They look reasonable and are comparable (order of magnitude after re-scaling by factor 1.255) with our results. The main metrics are area of sprouts and total length, two parameters describing bead and number of sprouts, witch is close to definition of primary branches. While there are fewer metrics available than those offered by our software, the authors of this model can certainly expand it, creating an impressive tool based on machine learning, which is currently performing well.

roi_size [Px^2]	1.048576e+06
number_of_sprouts	1.100000e+01
sprouts_total_length [Px]	5.370472e+03
sprouts_total_area [Px^2]	8.429200e+04
body_area [Px^2]	6.544000e+04
body_circularity	5.705830e-01

Table 2: Metrics from the IKOSA software

4.4 User experience

This software boasts one of the best graphical interfaces, characterized by its modern web-based architecture, excelling in data storage, computation, result review, and comprehensive documentation. Additionally, it provides computational resources as part of its solution. One drawback of this approach is its closed nature, limiting control over the computational process. While it offers the option to train models, it necessitates extensive data preparation, which can be a substantial undertaking. Furthermore, this is a paid tool, which may present a drawback in scientific applications, with costs potentially increasing based on the volume of calculations.

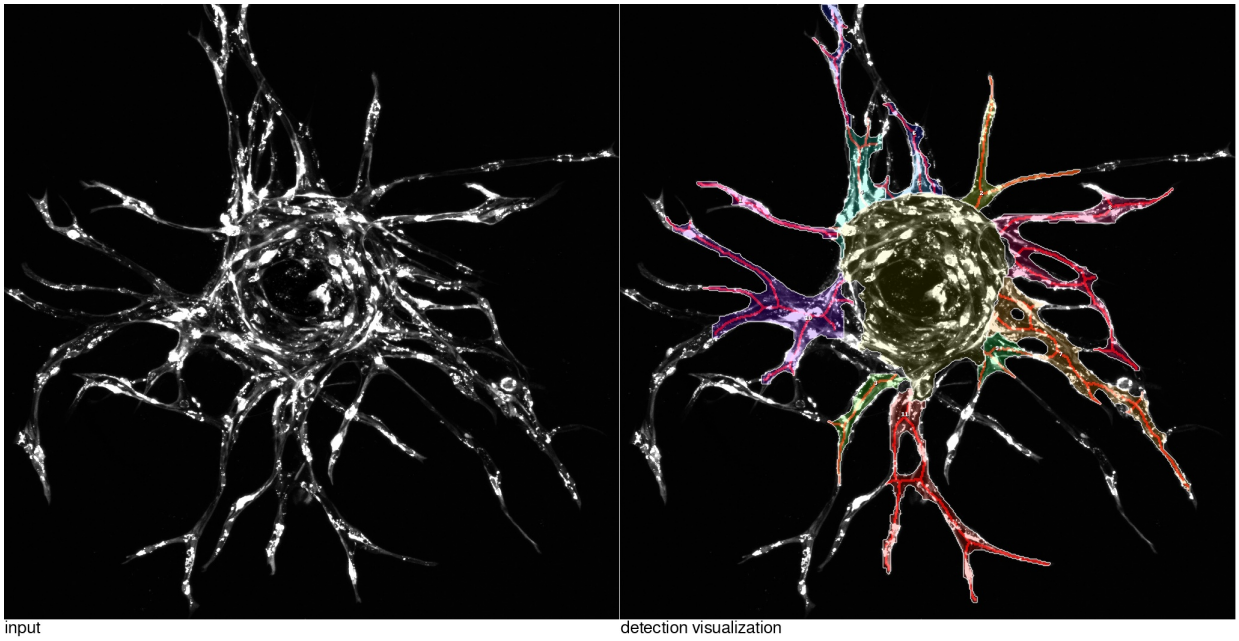

Figure 10: The segmentation mask generated by the IKOSA machine learning tool shows detected sprouts, each marked with distinct colors and labeled with numbers. Corresponding skeletons are also identified, and the bead mask is outlined with a polygon.

5 SproutAngio Python tool

5.1 Preliminary discussion

This tool stood out as one of the most promising options due to its implementation in the Python programming language and because it compares results with Sprout Analyzer from ImageJ. However, it has a specific requirement for two-channel images containing sprouts and nuclei. Without such data, its utilization becomes less straightforward. Despite this limitation, we made an effort to conduct a basic comparison to assess the software's performance and results. This software performs the main analysis using 3D data from the nuclei channel, with elements of the 2D max projection using the sprout channel.

We concur with the authors of SproutAngio in acknowledging that 2D projection introduces simplifications and leads to the loss of information. However, the applicability of 3D processing depends on the characteristics of the data being processed. We observed that the dataset provided by the SproutAngio authors is approximately 2-3 times denser in the z-direction, making 3D processing more suitable.

It is important to note that a comprehensive analysis should ideally incorporate time information, denoted as (X, Y, Z, t). However, neither SproutAngio nor our software utilizes information from previous or subsequent frames during the segmentation process.

The distinct advantage of operating in a 2D context following maximum projection (similar to SproutAngio's approach, at least partially) lies in significantly accelerated processing compared to 3D counterparts. To illustrate, we tested SproutAngio on the publicly available image group4-05.chi (Figure 11), containing two channels with dimensions 43x1024x1024. The computation took approximately 3 minutes and 15 seconds, resulting in Figure 12. This processing time is 1.5 times longer than what our software requires to process an entire dataset comprising 2638 images (using 5 out of 6 cores), each with, on average, more intricate networks. Even when giving SproutAngio the benefit of the doubt, our software remains approximately three orders of magnitude faster in terms of the number of processed images, while still providing abundant morphological information about the networks.

Given the evident challenge of developing efficient 3D segmentation software, at the time we were starting development, we opted for a more modest approach in our software. To provide perspective on the computation time scale, consider that if we were to recompute everything 100 times (which is approximately accurate, considering all our tests), and each image required 200 seconds of computation, we would need to wait for $\frac{200}{3600 \cdot 24} \cdot 2638 \cdot 100 = 610$ days instead of just a few hours.

5.2 Timing

We utilized the magic function %time within the Jupyter Notebook environment for benchmarking. In this comparison, we focus exclusively on step 2 of the SproutAngio code (Listing 2), which aligns with our segmentation definition. We experimented by providing a maximum-projected image (Figure 1) as input, employing 2D filtering instead of the original 3D filtering approach (utilizing the same method as in the original code). The recorded computational time, with the 'napri' library for visualization disabled, was approximately 9 seconds, slightly less.

Performing the same computation on publicly available data group4-05.chi with the unaltered code took around 50 seconds, slightly less. It appears that a substantial portion of the processing time is allocated to line 29, which involves median filtering. Turning off the median filtering, time has been reduced to a bit less than 9 seconds (result in Figure 13). Notably, results without filtering exhibit considerable similarity, as the final step involves maximum projection, effectively mitigating the impact of such filtering. Applying Gaussian blur, or similar smoothing method after max projection, would lead to similar effect as with the 3D median filtering. Consequently, employing median filtering before max projection appears unnecessary, as it mainly increases computational complexity.

Listing 2: 3D median filtering and max projection in SproutAngio.

```
1
2  # 1- Basic parameters:
3  image_path = 'test_data/VEGFA_10ng.czi'
4  median_filtering = (7,7,7)
5
6  (...)
7
8  # 2- Sprout segmentation:
9  # Importing necessary libraries and defining functions for the analysis:
10
11 (...)
12
13 with CziFile(image_path) as im:
14 image_array = im.asarray()
15
16 # checking the shape of the array
17 print(image_array.shape)
18
19 # defining the correct channels into new arrays based on the above (3) shape
20 im_sprout = image_array[0,0,0,0,:,:,:,0]
21 im_nuclei = image_array[0,0,0,1,:,:,:,0]
22
23 # Opening napari
24 import napari
25 viewer = napari.view_image(im_sprout, name='im_sprout')
26
27 # median filtering the sprout channel
28 med_sprout = im_sprout
29 med_sprout = scipy.ndimage.median_filter(med_sprout, size=median_filtering)
30
31 viewer.add_image(med_sprout, name="median_filtered_sprout")
32
33 # taking the maximum projection of the sprout channel and
34 #checking if median filtering size is ok
35 max_proj_0 = np.max(im_sprout, axis=0)
36 viewer.add_image(max_proj_0, name="max_projection_sprout")
37
38 max_projection = np.max(med_sprout, axis=0)
```

5.3 Graphical output

Starting from the pure implementation, we obtained the segmentation illustrated in Figure 14 (again using benchmark image from our dataset). This segmentation proved ineffective due to the use of the Otsu threshold, which was also tested in our approach. To address this issue, we substituted the Otsu method with a simple average threshold, consistent with our software's approach (Figure 15). The resulting

segmentation exhibits high quality, closely resembling our own results. It applies a consistent coloration scheme to each sprout, akin to the ICOSA software. Additionally, it offers the flexibility to adjust the bead radius, enabling the exclusion of more sprouts, which can potentially be equal to number of primary branches in our analysis.

The results obtained from the group4-05 image are presented in Figure 12. Once again, the segmentation process performed effectively. However, in a distinct case, specifically group1-07, numerous errors stemming from the 'napri' library were encountered, including division by zero errors. This issue might be attributed to the underdeveloped nature of the network in this particular image.

We were unable to conduct testing on the latter section of the code using benchmark image, as it would necessitate significant modifications, especially given that our dataset lacks a separate channel for nuclei. Consequently, further comparisons might yield misleading results. The authors of SproutAngio have introduced novel methods for measuring the width of sprouts, utilizing nuclei and the Euclidean distance transform (EDT). While we have not extensively examined this section of the code, a preliminary assessment of the results presented in Table 3 suggests its validity.

total skeleton length	437.000000
average skeleton length	109.250000
sprout number	4.000000
nuclei number	95.000000
tip width	30.780433
stalk width	35.618997
root width	46.062441
sprout volume	709204.000000
average paired nuclei distance	34.271619

Table 3: Metrics from the SproutAngio for the group4-05 data

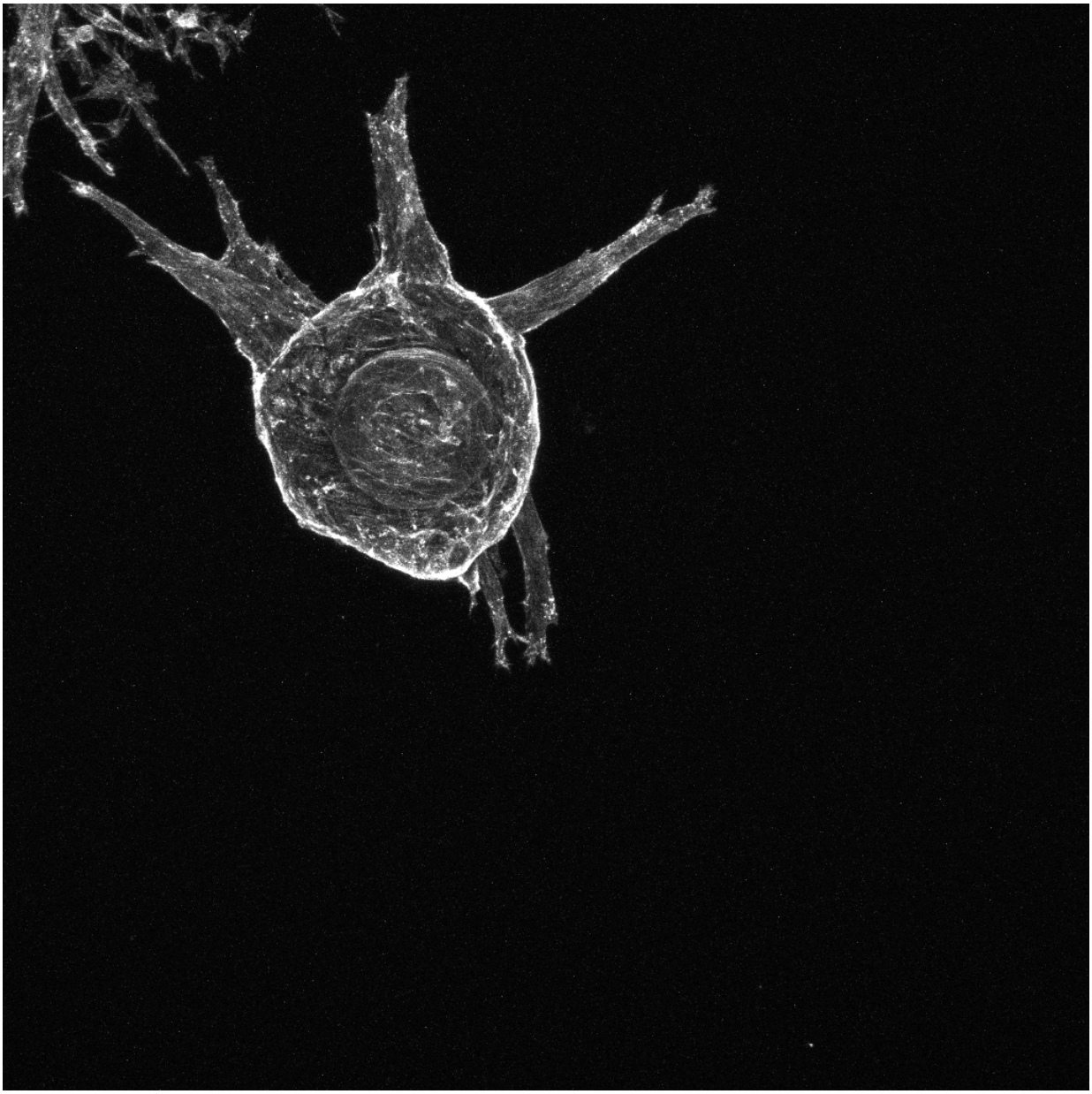

Figure 11: Max-projected bead sprouting data from group 4-05, sourced from the SproutAngio authors database.

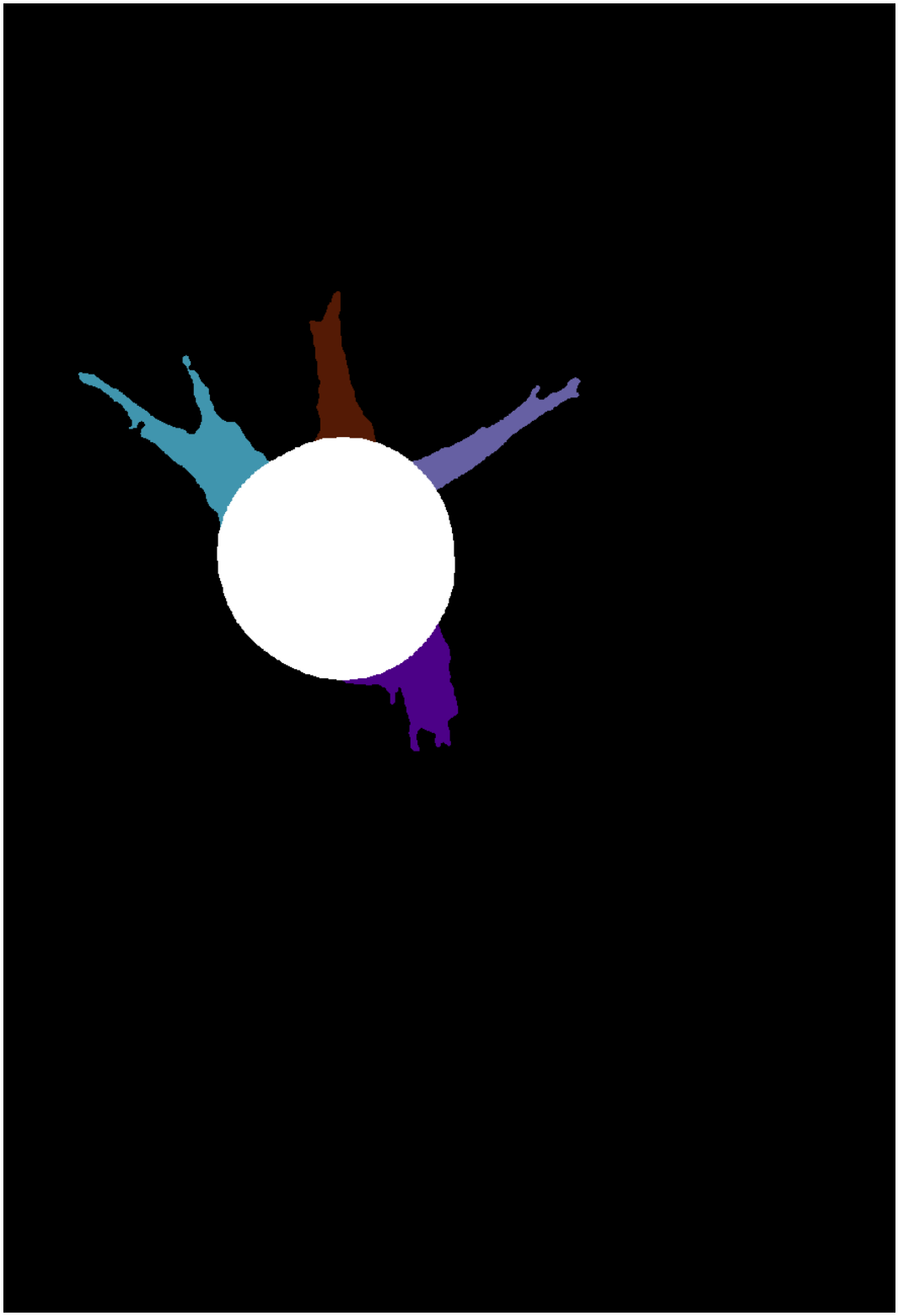

Figure 12: The segmentation of the image group4-01 is based on data provided by the authors of SproutAngio.

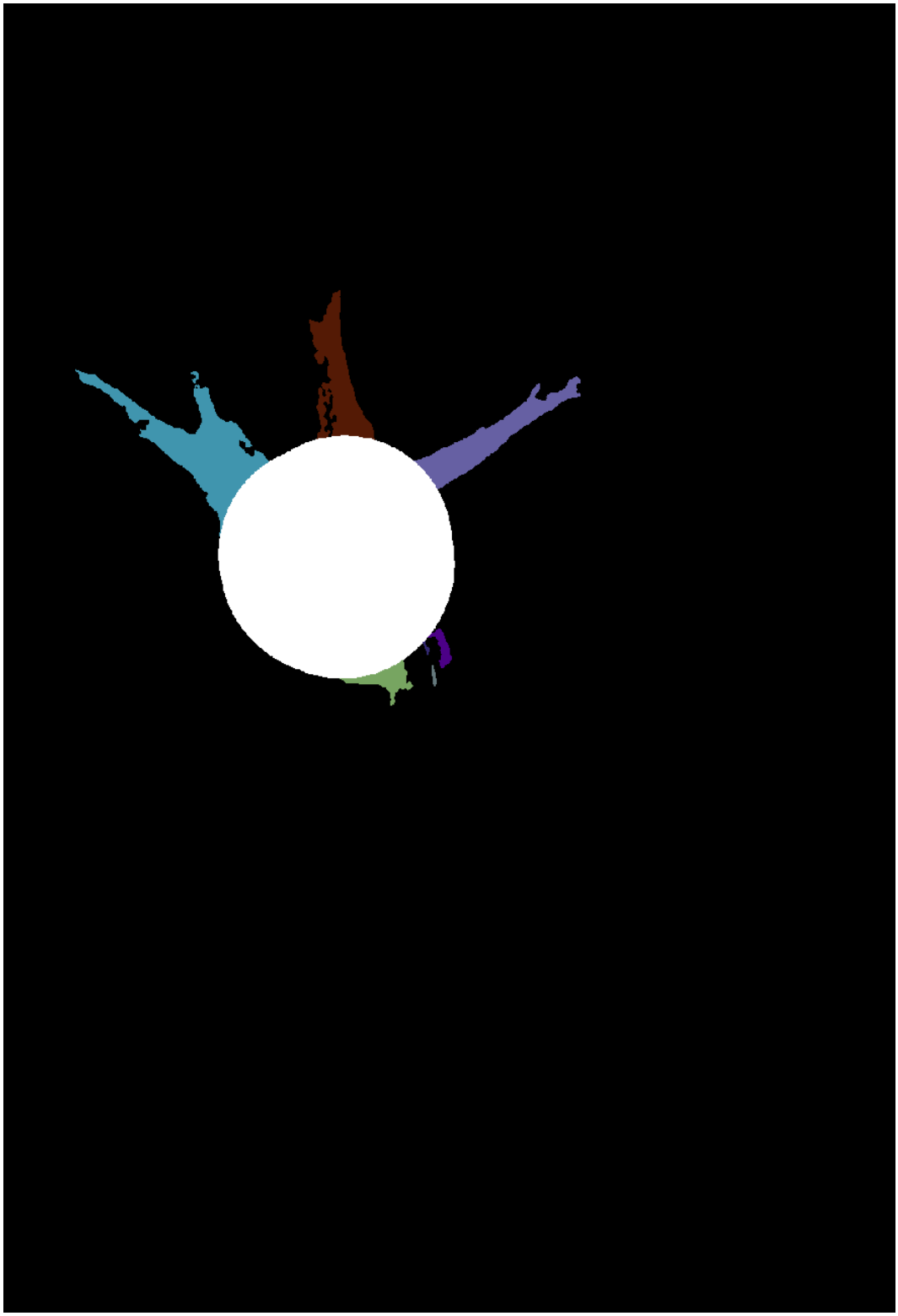

Figure 13: The segmentation of the image group4-01 is based on data provided by the authors of SproutAngio with median filtering turned off. Small holes and irregularities are present comparing to the variant with the median filtering (Figure 12).

Figure 14: The segmentation mask generated using SproutAngio with the Otsu threshold, using the max-projected benchmark image as input.

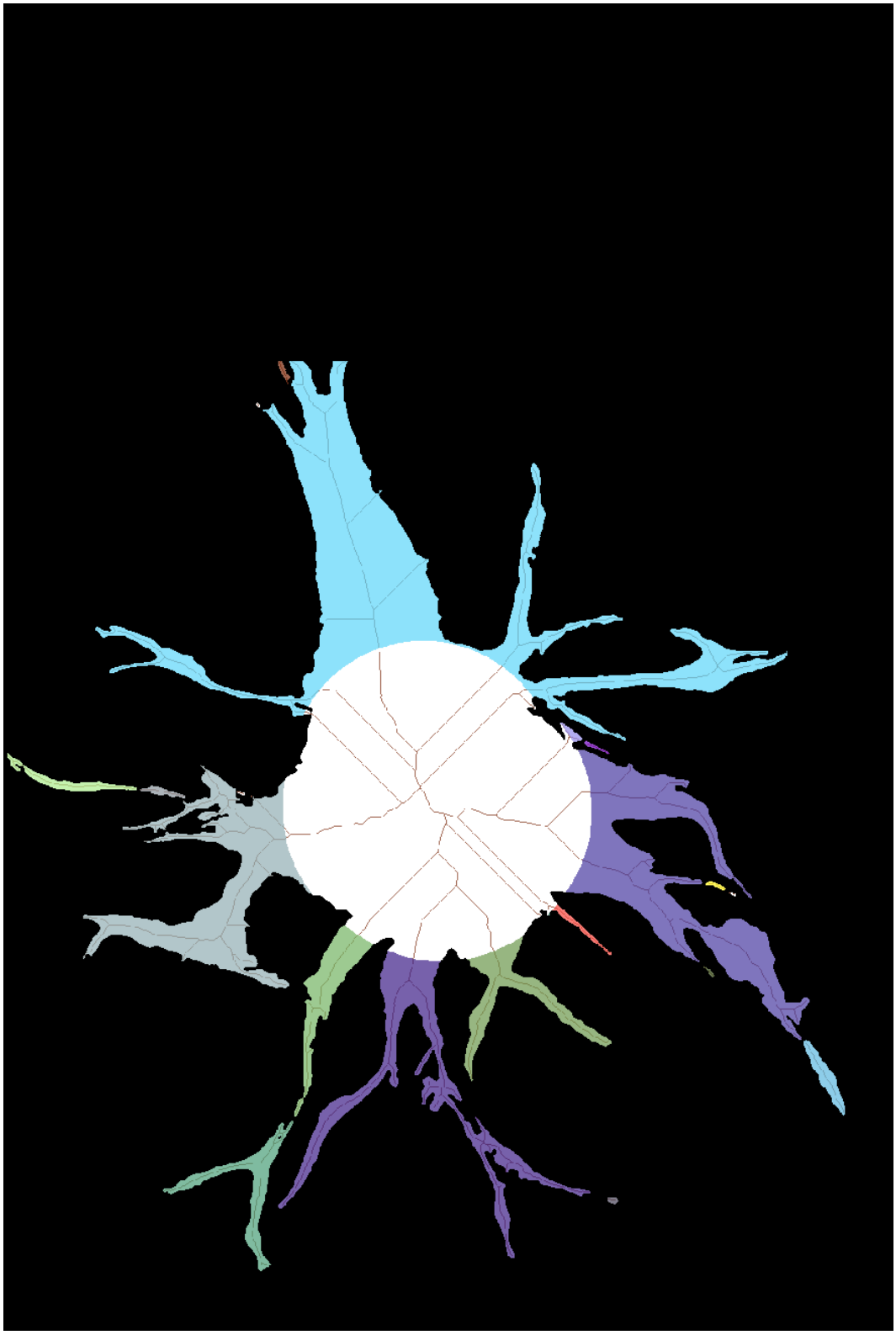

Figure 15: The segmentation mask generated using SproutAngio with the average threshold, using the max-projected benchmark image as input.

References

[1] J. Eglinger, H. Karsjens, and E. Lammert. "Quantitative assessment of angiogenesis and pericyte coverage in human cell-derived vascular sprouts". In: *Inflamm Regener 37, 2* (2017). DOI: `https://doi.org/10.1186/s41232-016-0033-2`.

[2] M. Beter et al. “SproutAngio: an open-source bioimage informatics tool for quantitative analysis of sprouting angiogenesis and lumen space”. In: *Sci Rep 13, 7279* (2023). DOI: https://doi.org/10.1038/s41598-023-33090-6.

[3] *IKOSA AI*. URL: https://www.kmlvision.com/our-offerings/ikosa-ai/. (accessed: 21.09.2023).