# Automated Test Platform for Validation of Intracranial Aneurysm Segmentation Algorithms


## Abstract

Intracranial aneurysms are a cerebrovascular disorder which contributes to significant morbidity and mortality globally. Aneurysms in the cerebral vasculature are often identified incidentally on cross-sectional imaging, and methods for automated detection and segmentation of intracranial aneurysms have become an active area of research. However, these methods must be validated on a sample data set in order to assess their accuracy, a task which has previously required individual authors to each create their own testing apparatus. This paper describes an open-source benchmarking tool which executes the aneurysm segmentation algorithm under test on a set of pre-segmented aneurysms and measures its accuracy. The tool supports mesh-based and voxel-based representations and includes rendering capability to allow visualization of the aneurysm and vessel geometry. It has been optimized to minimize runtime overhead and is available under MIT license at https://github.com/brysonh/AnSegTest/tree/main.


## Introduction

Intracranial aneurysms are a disorder of the cerebrovascular system in which dilatation of the vessel wall creates a localized deformation. These deformations can be found in approximately 3.2% of the global population and can result in neurological symptoms due to compression of adjacent structures as the aneurysm expands [7]. Additionally, intracranial aneurysms pose a risk of rupturing, which can result in a life-threatening subarachnoid hemorrhage with significant morbidity and mortality [13]. Early detection of the aneurysm before rupture can allow for

timely intervention to prevent rupture from occurring. However, intracranial aneurysms are often clinically silent and are frequently identified incidentally on imaging studies obtained for another indication [3]. In recent years, automated detection of aneurysms in imaging studies has become an active area of research, with a number of methods proposed to algorithmically identify aneurysms in cross-sectional medical imaging [8][9][2]. An associated task in automated processing of intracranial aneurysms is aneurysm segmentation, in which a boundary is created to delineate the border between aneurysm and healthy vessel. A number of methods have been proposed for this task, and new approaches are continuing to be developed [12][11]. As part of development of these algorithms, it is necessary to validate the performance of the proposed method, which is typically accomplished by running the method on a set of test data consisting of aneurysms which have been manually segmented by a human [1]. The segmentation generated by the automated method can then be compared to the true segmentation and the accuracy of the method quantified. In the past, most researchers have developed their own test apparatus, requiring significant effort and extending the development period [4][10]. This work describes an open-source test platform for such algorithms which removes this duplicated effort to allow more efficient development of aneurysm segmentation algorithms.

**Functionality**

The test platform consists of two primary components: a test script and a repository of test data. During a test, the test script sequentially retrieves files from the data repository, performs necessary processing to prepare the data for testing, runs the function under test using the data, compares the resulting calculated segmentation to the true segmentation, and stores the result of the comparison. The test script is written in pure Python and does not require compilation prior

to use. It makes use of several freely available Python modules including Numpy [5] and Scipy [16] which must be installed prior to use, and is launched via a command-line interface through which required parameters are passed as arguments. A full description of the script's interface and usage is included with the test platform. The script supports both mesh-based and voxel-based geometry, and the mode can be selected by the user depending on the needs of their specific algorithm. The script's output is saved as a CSV file and contains the calculated similarity measures for each data file used in the test. Available similarly measures include the Jaccard Index (JI) and Dice-Sorensen Coefficient (DSC), which are commonly used to describe the similarity between two sets [14]. They produce a value between zero and one, with zero representing completely disjoint sets and one representing identical sets, as per equations 1 and 2 below. In voxel mode, the sets consist of the voxels which belong to each segmentation, with set magnitude determined by the number of voxels in each set. In mesh mode, the sets consist of the faces which belong to each segmentation, with set magnitude determined by the total surface area of faces which are members of the set, allowing weighting of faces based on their size.

(1) $DSC(A,B) = \frac{2|A \cap B|}{|A|+|B|}$

*Equation 1: Formula used to calculate Dice-Sorensen Coefficient, where A and B are sets of voxels or faces*

(2) $JI(A,B) = \frac{|A \cap B|}{|A \cup B|}$

*Equation 2: Formula used to calculate Jaccard Index, where A and B are sets of voxels or faces*

The test script also includes rendering capability which can optionally be activated to visualize the aneurysm geometry and segmentation results. If rendering is enabled, a new window is opened following execution of the test script which displays a 3-dimensional view of each aneurysm in the test data set, with coloration indicating the calculated and actual segmentations. The user can rotate, zoom, and pan the rendering using mouse inputs, and a graphical interface allows the user to select which test data file to view. An example of the interface is shown in figure 1 below. Rendering is performed on the user's graphics processing resources using the OpenGL interface via the Pyglet module, which serves as a Python wrapper around the system's native graphics interface [6].

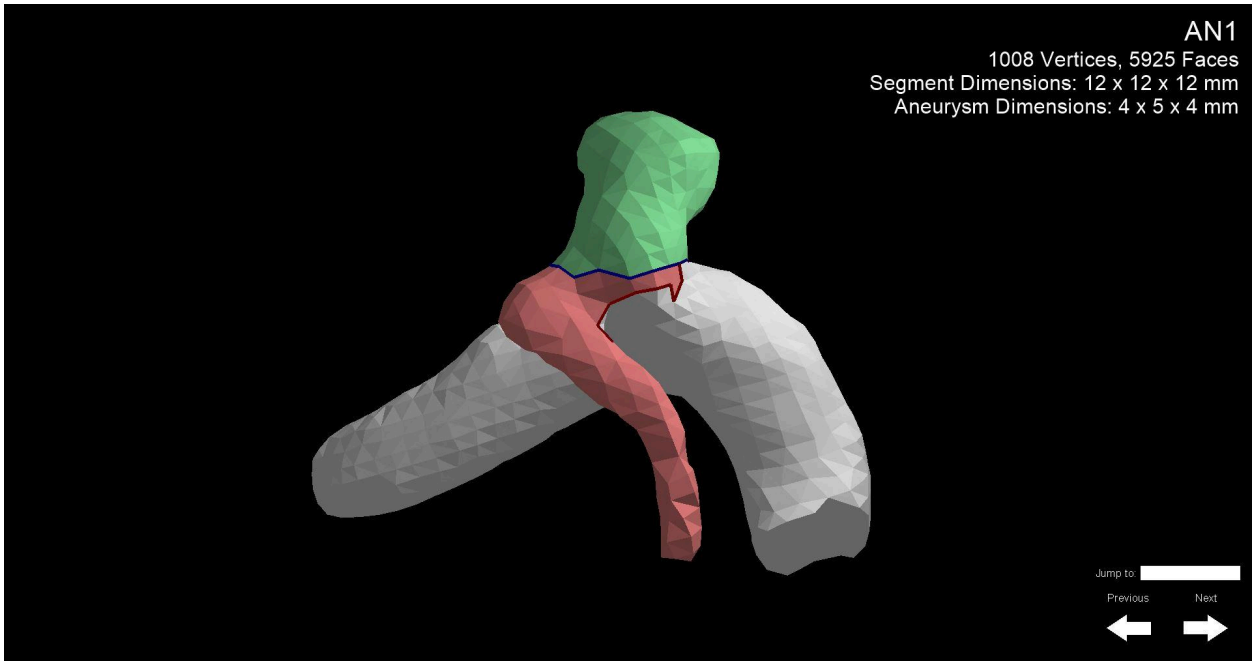


*Figure 1: Graphical user interface available to view renderings of aneurysm segmentations. Areas which belong to both the calculated and true segmentation are colored green, areas which belong only to the calculated segmentation are red, areas which belong only to the true*

*segmentation are blue, and areas which belong to neither are white. The controls on the bottom right can be used to navigate through the available aneurysm test files.*

The other component of the test platform is the repository of test data. This consists of 112 examples of intracranial aneurysms obtained from MRA images and published in the IntrA dataset [17], which also provides expert-verified manual segmentations of the aneurysms. The IntrA dataset uses the Wavefront format to denote the 3-dimensional structure of the aneurysm as a series of vertices connected by a mesh of triangular faces. For segmentation algorithms which operate on mesh-based representations of vessel geometry, this data format can be used directly. For segmentation algorithms which operate on a voxel representation of vessel geometry, the mesh representations from the IntrA dataset have also been converted into a voxel format, allowing the user to select the appropriate format for their use case. Examples of both mesh and voxel representations of aneurysms are shown in figures 2 and 3 below.

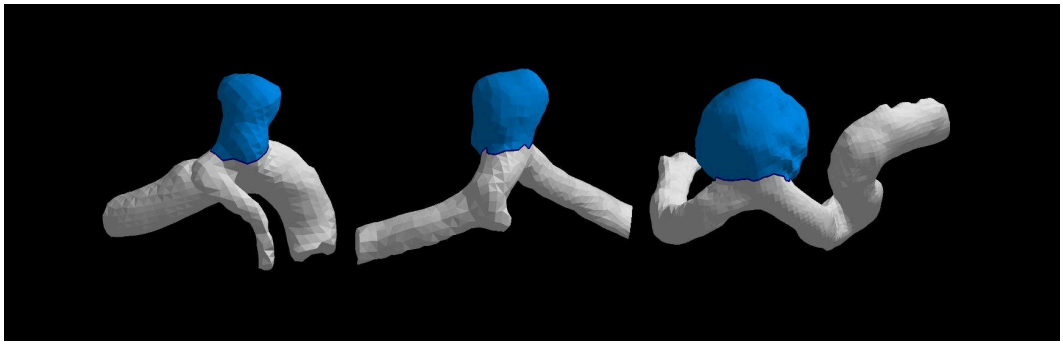

*Figure 2: examples of aneurysm geometry and segmentation data using a mesh-based representation*

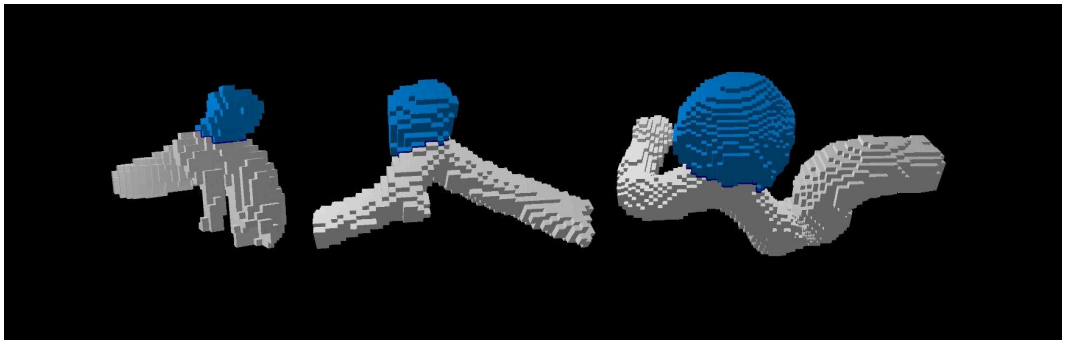

*Figure 3: examples of aneurysm geometry and segmentation data using a voxel-based representation*

**Validation**

The test platform has undergone testing to ensure it functions as expected and produces valid results. During development, unit testing was performed on each function to identify issues early and reduce the testing burden once development was completed. Following development, validation was performed on the calculated similarity measures to ensure they produce accurate results. To accomplish this, a phantom consisting of either voxels or triangular mesh faces depending on the test data mode was created as input to the program, and two arbitrary divisions along the x-axis were chosen. One of these divisions was chosen as the boundary for the test's true segmentation, with all points on the phantom with x coordinates less than this boundary belonging to the true segmentation. Similarly, the other division was chosen as the boundary for the calculated segmentation, with all points less than this boundary belonging to the calculated segmentation. Based on this geometry, the ideal Jaccard Index and Dice-Sorensen Coefficient are given by Equations 3 and 4 for voxel mode and Equations 5 and 6 for mesh mode. In these equations, $x_t$ denotes the x coordinate of the true segmentation boundary, $x_c$ denotes the x

coordinate of the calculated segmentation boundary, and y and z denote the prism size in the y and z dimensions, respectively

$$(3)\ DSC_{Ideal}(A,B) = \frac{2(2(min(x_t,x_c)(y)) + 2(min(x_t,x_c)(z-2)) + (y-2)(z-2))}{(2(x_t)(y) + 2(x_t)(z-2) + (y-2)(z-2)) + (2(x_c)(y) + 2(x_c)(z-2) + (y-2)(z-2))}$$

*Equation 3: Ideal DSC for a rectangular prism in voxel representation*

$$(4)\ JI_{Ideal}(A,B) = \frac{2(min(x_t,x_c)(y)) + 2(min(x_t,x_c)(z-2)) + (y-2)(z-2)}{2(max(x_t,x_c)(y)) + 2(max(x_t,x_c)(z-2)) + (y-2)(z-2)}$$

*Equation 4: Ideal JI for a rectangular prism in voxel representation*

$$(5)\ DSC_{Ideal}(A,B) = \frac{2(2(min(x_t,x_c)(y)) + 2(min(x_t,x_c)(z)) + (y)(z))}{(2(x_t)(y) + 2(x_t)(z) + (y)(z)) + (2(x_c)(y) + 2(x_c)(z) + (y)(z))}$$

*Equation 3: Ideal DSC for a rectangular prism in mesh representation*

$$(6)\ JI_{Ideal}(A,B) = \frac{2(min(x_t,x_c)(y)) + 2(min(x_t,x_c)(z)) + (y)(z)}{2(max(x_t,x_c)(y)) + 2(max(x_t,x_c)(z)) + (y)(z)}$$

*Equation 4: Ideal JI for a rectangular prism in mesh representation*

The phantom and its divisions are then used as input for the test script and similarities are calculated and compared to the ideal results derived from the above equations. Edge cases such as empty sets are tested directly, and then the two divisions are swept across the surface in a nested fashion at regular intervals, with testing performed at each increment. A rendering of the test setup is shown in figure 4 below.

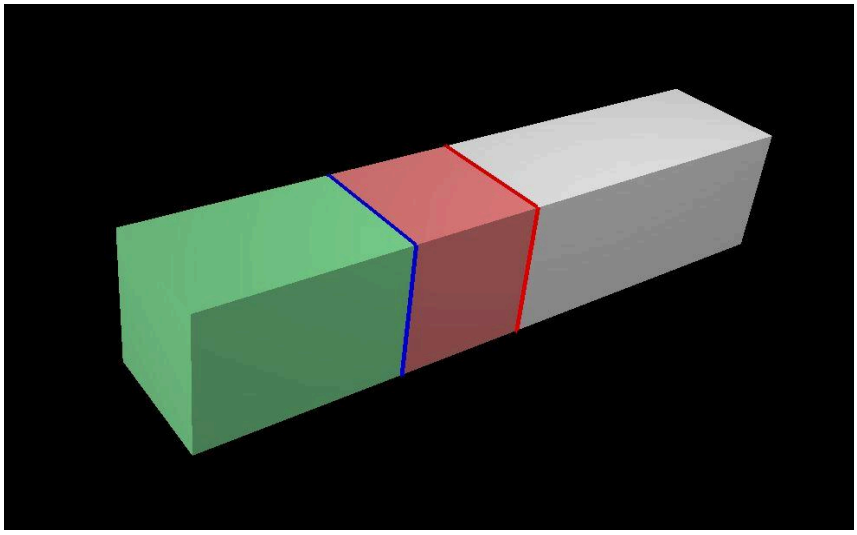

*Figure 4: phantom used for validation of similarity measure calculation. The blue line represents the true segmentation, the red line represents the calculated segmentation, the green area represents the intersection, the red area represents faces which belong only to the calculated segmentation, and the white area represents faces which belong to neither segmentation*

All tests completed successfully, with maximum error less than 1e-15 attributable to floating-point imprecision during mesh surface area calculation. Validation of the test data preprocessing and interface with the function under test was performed using a simulated algorithm which generated a randomized boundary when given input data and returned a segmentation based on this boundary. Tests were run in both mesh and voxel mode using separate simulated algorithms, and all test data files were used for every test. Edge conditions including returning a segmentation which selected no points and returning a segmentation which selected every point were tested directly.

**Performance**

Throughout development of the test platform, optimizations were made to reduce execution time in order to minimize the test platform's impact on overall testing time. Benchmarking was performed using an AMD Ryzen 9 5900HS at 3.3GHz, and all values are rounded to the nearest millisecond. Although the test script is written in Python, it makes use of the Numpy module for most operations involving large arrays of voxels or vertices, which results in a substantial performance improvement as Numpy utilizes precompiled C functions which execute significantly faster than a comparable Python function [15]. The test platform also utilizes caching in order to improve runtime by eliminating repetitive computations. The first time the test script is executed, it reads the test data from the files in which it is stored and processes it into data structures which contain all of the necessary information to perform the test and render the results. These data structures are then written to nonvolatile memory and can be loaded directly by the test script without additional processing the next time the test script is executed. If one of the data files used to create the cached data structure is modified, the test script will identify that the cached data is stale and recalculate the cache structure for that specific file. To evaluate the performance improvement from implementing caching, a mock function under test was created which returns the worst-case result of a segmentation containing the entire input set, the test script was executed with this mock function as its target, and the execution times for each file in the data repository were measured. In voxel mode without caching, each file took an average of 474 ms to test, with execution times ranging between 122 ms and 1410 ms. Implementing caching in voxel mode reduced this execution time to an average of 89 ms, with a minimum and maximum of 31 ms and 263 ms, respectively, demonstrating an 81% decrease in total execution time overhead. Each file in the cache utilizes 4.53 MB of storage space on

average, resulting in an additional 507 MB of disk utilization with caching enabled. In mesh mode without caching, each file took an average of 547 ms to process, with a minimum of 91 ms and a maximum of 4416 ms. Utilizing caching in mesh mode reduces average processing time to 187 ms, with a minimum of 40 ms and maximum of 1280 ms, a 65% reduction. The cache files occupy an average of 2.49 MB each of file space, requiring 279 MB total for the 112 files included in the test data set. Histograms of execution time in mesh and voxel mode are shown in figures 5 and 6 below.

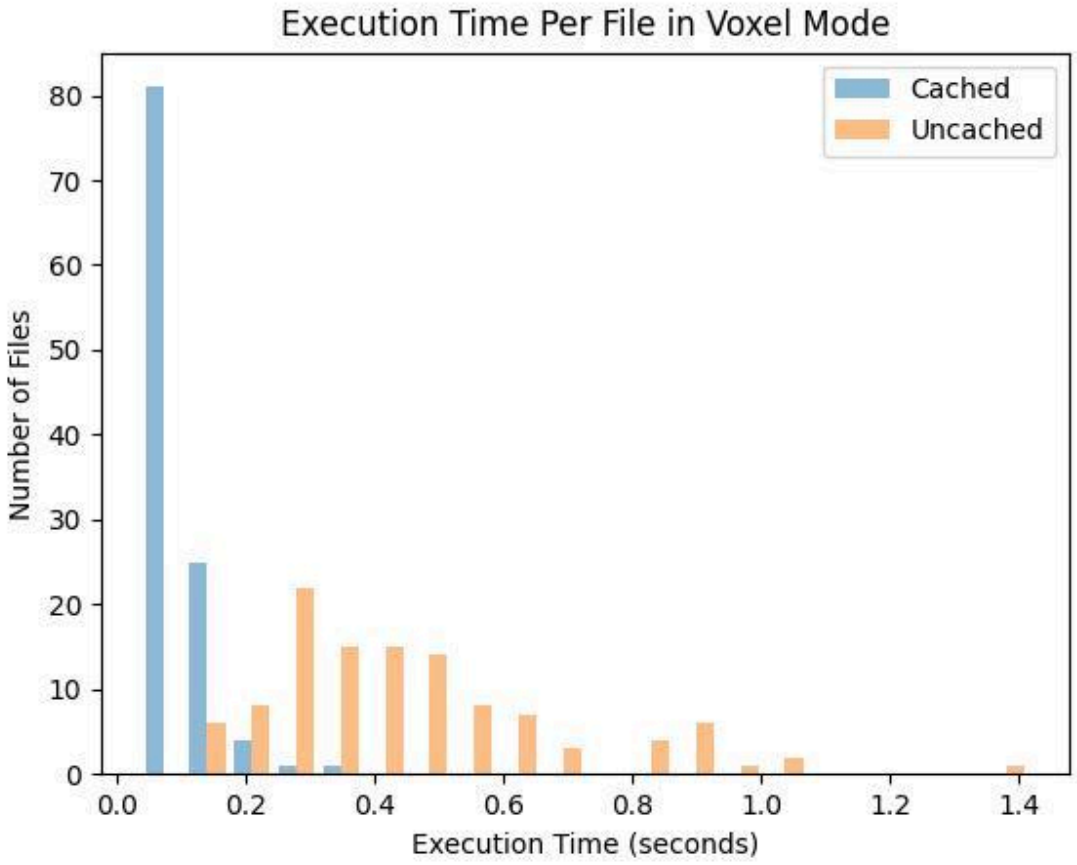

*Figure 5: comparison of execution times in voxel mode between cached and uncached executions of the test script. Utilizing caching for the test data objects results in a significant performance improvement.*

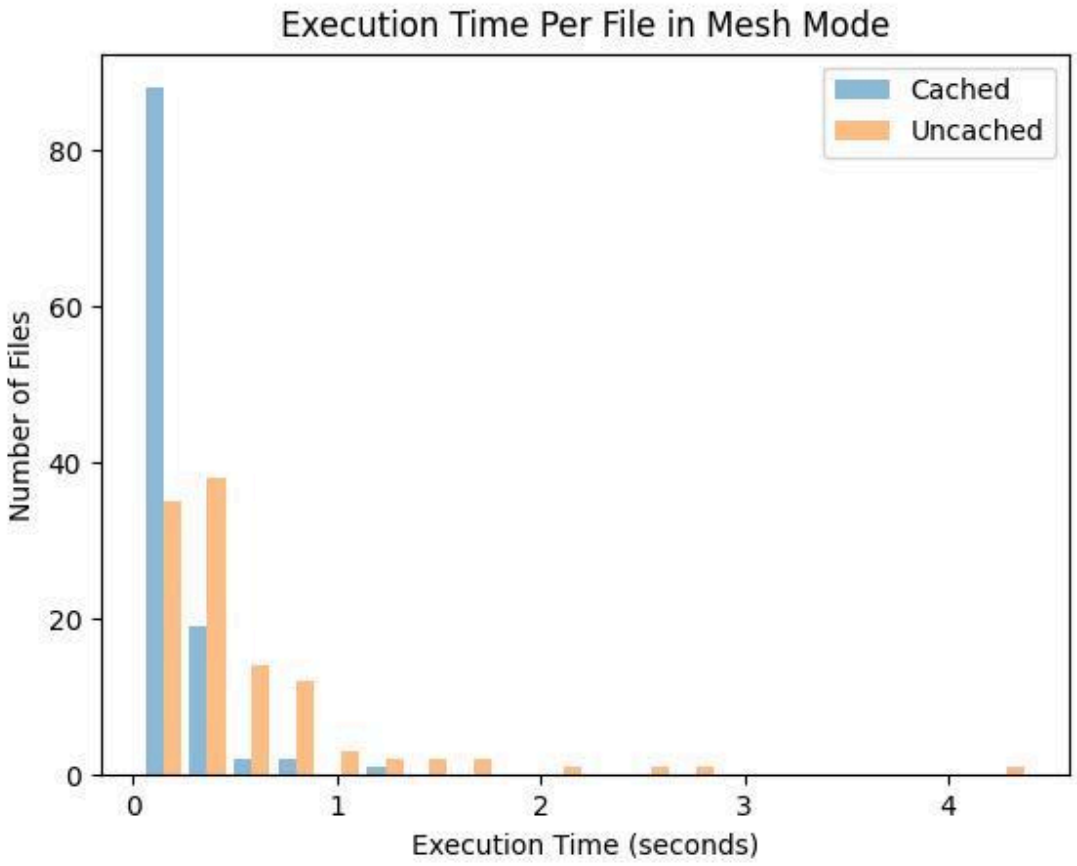


*Figure 6: comparison of execution times in mesh mode between cached and uncached executions of the test script. Performance is significantly increased, but to a lesser degree than in voxel mode, with a 65% reduction in execution time.*

## Conclusion

This paper describes a test platform intended to reduce the testing overhead required to develop aneurysm segmentation methods. It provides a library of expert-segmented test data available in both mesh and voxel representations on which to test a new algorithm, as well as a test program to perform the test, calculate results, and provide a rendering of the outcome to aid in debugging and iterative improvement. It has been validated and optimized, and is made freely available for download.

## References


[1] Cárdenes, R., Larrabide, I., Román, L. S., & Frangi, A. F. (2012). Performance Assessment of Isolation Methods for geometrical cerebral aneurysm analysis. *Medical & Biological Engineering & Computing, 51*(3), 343–352. https://doi.org/10.1007/s11517-012-1003-8

[2] Din, M., Agarwal, S., Grzeda, M., Wood, D. A., Modat, M., & Booth, T. C. (2022). Detection of cerebral aneurysms using Artificial Intelligence: A systematic review and meta-analysis. *Journal of NeuroInterventional Surgery, 15*(3), 262–271. https://doi.org/10.1136/jnis-2022-019456

[3] Faluk M, Das JM, De Jesus O. Saccular Aneurysm. [Updated 2024 Mar 7]. In: *StatPearls [Internet]*. Treasure Island (FL): StatPearls Publishing; 2024 Jan-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK557867/

[4] Firouzian, A., Manniesing, R., Flach, Z. H., Risselada, R., van Kooten, F., Sturkenboom, M. C. J. M., van der Lugt, A., & Niessen, W. J. (2011). Intracranial aneurysm segmentation in 3D CT angiography: Method and quantitative validation

with and without prior noise filtering. *European Journal of Radiology, 79*(2), 299–304. https://doi.org/10.1016/j.ejrad.2010.02.015

[5] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature, 585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[6] Holkner, A. (n.d.). *The OpenGL interface*. Pyglet Documentation. https://pyglet.readthedocs.io/en/latest/programming_guide/gl.html

[7] Jersey AM, Foster DM. Cerebral Aneurysm. [Updated 2023 Apr 3]. In: *StatPearls [Internet]*. Treasure Island (FL): StatPearls Publishing; 2024 Jan-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK507902/

[8] Kohout, J., Chiarini, A., Clapworthy, G. J., & Klajnšek, G. (2013). Aneurysm identification by analysis of the blood-vessel skeleton. *Computer Methods and Programs in Biomedicine, 109*(1), 32–47. https://doi.org/10.1016/j.cmpb.2012.08.018

[9] Kuwabara, M., Ikawa, F., Sakamoto, S., Okazaki, T., Ishii, D., Hosogai, M., Maeda, Y., Chiku, M., Kitamura, N., Choppin, A., Takamiya, D., Shimahara, Y., Nakayama, T., Kurisu, K., & Horie, N. (2023). Effectiveness of tuning an artificial intelligence algorithm for cerebral aneurysm diagnosis: A study of 10,000 consecutive cases. *Scientific Reports, 13*(1). https://doi.org/10.1038/s41598-023-43418-x

[10] Law, M. W., & Chung, A. C. (2007). Vessel and intracranial aneurysm segmentation using multi-range filters and local variances. *Medical Image Computing and*

*Computer-Assisted Intervention – MICCAI 2007, 10*(Pt 1), 866–874. https://doi.org/10.1007/978-3-540-75757-3_105

[11] Li, T., An, X., Di, Y., He, J., Liu, S., & Ming, D. (2022). Segmentation method of cerebral aneurysms based on entropy selection strategy. *Entropy, 24*(8), 1062. https://doi.org/10.3390/e24081062

[12] Nishi, H., Cancelliere, N. M., Rustici, A., Charbonnier, G., Chan, V., Spears, J., Marotta, T. R., & Mendes Pereira, V. (2023). Deep learning-based cerebral aneurysm segmentation and morphological analysis with three-dimensional rotational angiography. *Journal of NeuroInterventional Surgery*, *16*(2), 197–203. https://doi.org/10.1136/jnis-2023-020192

[13] Orz, Y., & AlYamany, M. (2015). The impact of size and location on rupture of intracranial aneurysms. *Asian Journal of Neurosurgery, 10*(01), 26–31. https://doi.org/10.4103/1793-5482.144159

[14] Survarachakan, S., Prasad, P. J., Naseem, R., Pérez de Frutos, J., Kumar, R. P., Langø, T., Alaya Cheikh, F., Elle, O. J., & Lindseth, F. (2022). Deep learning for image-based liver analysis — a comprehensive review focusing on malignant lesions. *Artificial Intelligence in Medicine, 130*, 102331. https://doi.org/10.1016/j.artmed.2022.102331

[15] van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering, 13*(2), 22–30. https://doi.org/10.1109/mcse.2011.37

[16] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R., Jones, E., Kern, R., Larson, E.,

… Vázquez-Baeza, Y. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods, 17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[17] Yang, X., Xia, D., Kin, T., & Igarashi, T. (2020). Intra: 3D intracranial aneurysm dataset for Deep Learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr42600.2020.00273