The Journal of Open Source Software

Brightest path tracing: A Python package to trace the brightest path in 2D and 3D images

Vasudha Jha[1,2,3] **and Robert H. Cudmore**[3¶]

1 Department of Radiation Oncology, Stanford University School of Medicine, Stanford, CA, USA **2** Department of Computer Science, University of California, Davis, CA, USA **3** Department of Physiology and Membrane Biology, University of California Davis School of Medicine, Davis, CA, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Aoife Hughes

Reviewers:

- @jingpengw
- @haesleinhuepf

Submitted: 17 July 2023
Published: unpublished

Summary

Brightest path tracing is a widely used image processing technique in several fields including biology, geography, and geology. However, despite the availability of many image processing libraries in Python, few offer an out-of-the-box implementation of a brightest path tracing algorithm. This paper presents a Python package, `brightest-path-lib`, that efficiently finds the path with maximum brightness between points in a 2D or 3D image. An example graphical user interface is provided as a Napari plugin. Taken together, the package and plugin provide a powerful and extensible tool for users to efficiently trace structures of interest in 2D or 3D images, regardless of the type of structure being analyzed.

Statement of Need

Brightest path tracing is an image processing technique which involves identifying the brightest points in an image and then connecting them to form a path. This path can provide valuable information about the structure and organization of the object being traced, such as their connectivity and morphology. Brightest path tracing is commonly used in neuroscience and medical imaging to trace structures such as neurons or blood vessels, aiding in disease diagnosis and treatment.

Python has a thriving ecosystem of image processing libraries that provide a broad range of functionalities, from basic operations like image manipulation and saving to advanced tasks like object detection and machine learning-based image analysis. Some of the widely used image processing libraries in Python include OpenCV (Bradski, 2000), scikit-image (Walt et al., 2014), imageio (Klein et al., 2023), and Pygame (Pygame, 2023). Although these libraries support implementing search algorithms such as A* (Hart et al., 1968) and Dijkstra's algorithm (Dijkstra, 1959) to find the shortest path between two points in an image, they lack built-in support for brightest path tracing. While other software tools like ImageJ (Abràmoff et al., 2004) offer solutions for neuronal tracing, such as the Simple Neurite Tracer (Arshadi et al., 2021; Longair et al., 2011), they are implemented in Java and may pose challenges when integrating with Python-based workflows. Here, we present the `brightest-path-lib`, a Python package designed to fill this gap by providing native Python implementations of informed search algorithms tailored specifically for brightest path tracing, making it easier to integrate into existing Python-based image processing pipelines.

Results

The `brightest-path-lib` package is available on PyPi for installation using pip and the source code is provided on GitHub (https://github.com/mapmanager/brightest-path-lib). The package includes efficient algorithms to trace the path between points with the maximum brightness in 2D and 3D images (Figure 1). Two search algorithms are provided including A* Search and Bidirectional Search (Pijls & Post, 2009; Wink et al., 2000).

Detailed application programming interface (API) and user documentation as well as example workflows are provided, enabling users to quickly get started with the package and understand its functionalities. Finally, the use of open-source licensing, and Python's extensive usage in the scientific community ensures a vast community of users and developers who can contribute to the package 's continued growth and improvement.

To allow non-programming users to utilize the `brightest-path-lib` package, we have developed a Napari (Napari, 2023) plugin called `Napari Tracer Plugin` (available at https://github.com/mapmanager/napari-tracing) that integrates the `brightest-path-lib` to provide a graphical user interface (GUI) for real-time visualization of the path and the image, making it easier to understand the object's structure and organization being traced (Figure 1). Finally, tracings are saved in the SWC neuron morphology file format (Stockley et al., 1993).

Discussion

Together, the `brightest-path-lib` package and the Napari plugin provide a powerful set of tools for both programmers and end-users to efficiently trace structures of interest. Moreover, the plugin is designed to work seamlessly with other Napari plugins, allowing for more integrated and streamlined workflows. This feature further extends the capabilities of the `brightest-path-lib` and `Napari Tracer Plugin` beyond their core functionalities, providing users with a more comprehensive suite of tools for image analysis and visualization.

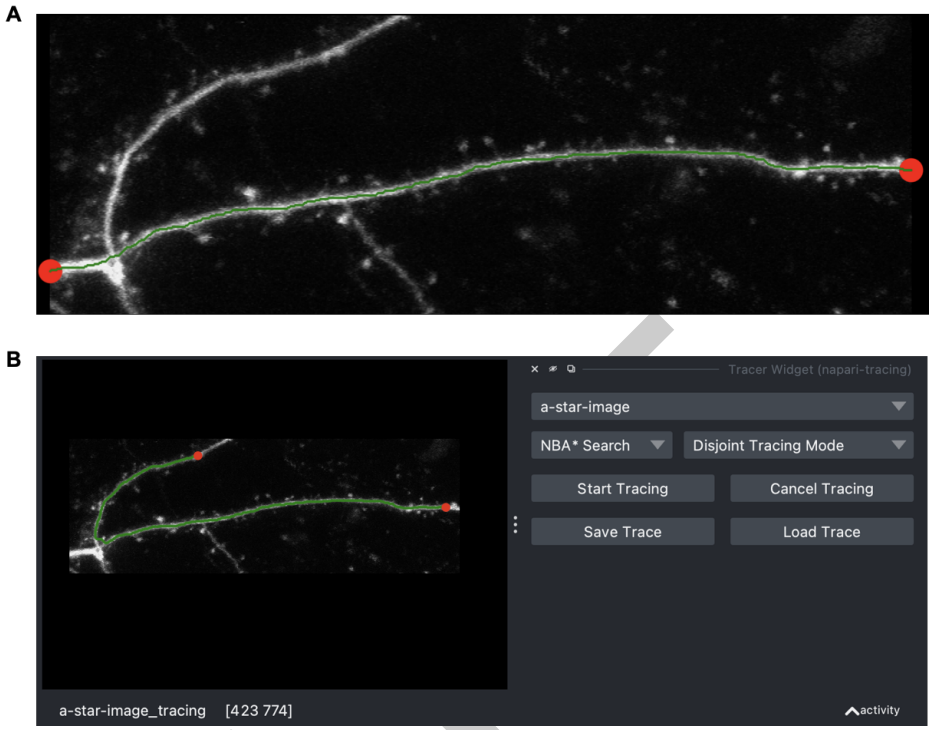

Figure 1: Example results of brightest-path tracing of a neuronal dendrite. (A) An example tracing performed from a Python script. The start and stop points are red circles, the final brightest path is in green. The grayscale image is a maximal intensity projection of a 3D image volume. (B) An example screenshot of the Napari Tracing Plugin. On the left is the image, start and stop points (red), and the final brightest path (green). On the right is the user interface for tracing including controls to select the tracing algorithm, start and cancel the tracing, and finally to save and load the tracing as an SWC file.

Acknowledgements

VJ and RHC were supported by an NIH grant (1RF1MH123206-01A1) and a Chan Zuckerberg Initiative grant (2022-252611).

References

Abràmoff, M. D., Magalhães, P. J., & Ram, S. J. (2004). Image processing with ImageJ. *Biophotonics International*, *11*(7), 36–42.

Arshadi, C., Günther, U., Eddison, M., Harrington, K. I. S., & Ferreira, T. A. (2021). SNT: A unifying toolbox for quantification of neuronal anatomy. *Nat Methods*, *18*(4), 374–377. https://doi.org/10.1038/s41592-021-01105-7

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*, 269–271. https://doi.org/10.1007/BF01386390

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

Klein, A., Wallkötter, S., Silvester, S., Tanbakuchi, A., actions-user, Müller, P., Nunez-Iglesias, J., Harfouche, M., Dennis, Lee, A., McCormick, M., Ischr, OrganicIrradiation, Rai, A.,

Ladegaard, A., Smith, T. D., Kemenade, H. van, Vaillant, G., jackwalker64, … Inggs, G. (2023). *Imageio/imageio: v2.31.1* (Version v2.31.1). Zenodo. https://doi.org/10.5281/zenodo.8025955

Longair, M. H., Baker, D. A., & Armstrong, J. D. (2011). Simple neurite tracer: Open source software for reconstruction, visualization and analysis of neuronal processes. *Bioinformatics*, *27*(17), 2453–2454. https://doi.org/10.1093/bioinformatics/btr390

Napari. (2023). https://www.napari.org/

Pijls, W., & Post, H. (2009). *Yet another bidirectional algorithm for shortest paths*.

Pygame. (2023). https://www.pygame.org/

Stockley, E., Cole, H., Brown, A., & Wheal, H. (1993). A system for quantitative morphological measurement and electrotonic modelling of neurons: Three-dimensional reconstruction. *Journal of Neuroscience Methods*, *47*(1-2), 39–51. https://doi.org/10.1016/0165-0270(93)90020-R

Walt, S. van der, Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., & contributors, the scikit-image. (2014). Scikit-image: Image processing in python. *PeerJ*, *2*, e453. https://doi.org/10.7717/peerj.453

Wink, O., Niessen, W. J., & Viergever, M. A. (2000). Minimum cost path determination using a simple heuristic function. *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, *3*, 998–1001 vol.3. https://doi.org/10.1109/ICPR.2000.903713