SOFTWARE TOOL ARTICLE

REVISED GeNePy3D: a quantitative geometry python toolbox for bioimaging [version 2; peer review: 2 approved]

Minh-Son Phan, Anatole Chessel

Laboratory of Optics and Biosciences, CNRS, INSERM, Ecole polytechniqe, Institut polytechnique de Paris, Palaiseau, 91120, France

v2 **First published:** 26 Nov 2020, **9**:1374
https://doi.org/10.12688/f1000research.27395.1
Latest published: 17 Jun 2021, **9**:1374
https://doi.org/10.12688/f1000research.27395.2

Abstract

The advent of large-scale fluorescence and electronic microscopy techniques along with maturing image analysis is giving life sciences a deluge of geometrical objects in 2D/3D(+t) to deal with. These objects take the form of large scale, localised, precise, single cell, quantitative data such as cells' positions, shapes, trajectories or lineages, axon traces in whole brains atlases or varied intracellular protein localisations, often in multiple experimental conditions. The data mining of those geometrical objects requires a variety of mathematical and computational tools of diverse accessibility and complexity. Here we present a new Python library for quantitative 3D geometry called GeNePy3D which helps handle and mine information and knowledge from geometric data, providing a unified application programming interface (API) to methods from several domains including computational geometry, scale space methods or spatial statistics. By framing this library as generically as possible, and by linking it to as many state-of-the-art reference algorithms and projects as needed, we help render those often specialist methods accessible to a larger community. We exemplify the usefulness of the GeNePy3D toolbox by re-analysing a recently published whole-brain zebrafish neuronal atlas, with other applications and examples available online. Along with an open source, documented and exemplified code, we release reusable containers to allow for convenient and wide usability and increased reproducibility.

Keywords

bioimage informatics, quantitative geometry, computational geometry, workflow, python

This article is included in the NEUBIAS - the Bioimage Analysts Network gateway.

Open Peer Review

Reviewer Status ✓ ✓

	Invited Reviewers	
	1	**2**
version 2 (revision) 17 Jun 2021	✓ report	
version 1 26 Nov 2020	? report	✓ report

1. **Virginie Uhlmann**, European Bioinformatics Institute (EMBL-EBI), Cambridge, UK

2. **Yizhi Wang**, Virginia Polytechnic Institute and State University, Arlington, USA

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Anatole Chessel (anatole.chessel@polytechnique.edu)

Author roles: Phan MS: Data Curation, Methodology, Software, Visualization, Writing – Original Draft Preparation; **Chessel A**: Conceptualization, Data Curation, Methodology, Software, Supervision, Writing – Original Draft Preparation

Competing interests: No competing interests were disclosed.

Grant information: This work was supported by the Morphoscope Equipex (ANR-11-EQPX-0029 Morphoscope2) and the France BioImaging national infrastructure (ANR-10-INBS-04 France BioImaging).
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Phan MS and Chessel A. **GeNePy3D: a quantitative geometry python toolbox for bioimaging [version 2; peer review: 2 approved]** F1000Research 2021, **9**:1374 https://doi.org/10.12688/f1000research.27395.2

First published: 26 Nov 2020, **9**:1374 https://doi.org/10.12688/f1000research.27395.1

REVISED **Amendments from Version 1**

Small updates to answer reviewer comments include removal of reference to 'large scale', a clarification of reasoning behind licensing choices and removal of mention of alignment algorithms. We also updated the title, to remove mention of large scale, and updated affiliation of one of the authors.

Any further responses from the reviewers can be found at the end of the article

Introduction

Bioimage informatics aims at bringing microscopy into quantitative biology, associating higher level information to pixels to answer complex biological questions. In particular machine learning based techniques[1] are easing the image analysis step, extracting geometrical objects from multidimensional images. But the next step, transforming that geometrical information into biological knowledge, involves a very diverse set of algorithmic tools in distinct communities, from spatial statistics[2,3] to computational geometry[4,5] or neuroinformatics[6]. Similarly, the software ecosystem around geometrical data analysis is very diverse and heterogeneous, with reference algorithm implementation spread across languages (Spatstat[7] for spatial statistics in R, CGAL[8] for computational geometry in C++) or across module in python (scipy[9] for generic algorithms, anytree[10] for trees, trimesh[11] for meshes etc), a lack of generic geometric data exchange format and standard graphical tools like Fiji[12] and Icy[13] being limited in the flexibility of the analysis easily available. To address this problem, we propose GeNePy3D[14,15], a python library meant as a 'middleware' library to facilitate building data analysis workflows for geometrical objects by providing one convenient API for geometrical data I/O, conversion and interaction between geometrical objects and access to many common and less common algorithm. We will introduce below the architecture of the library and show one example workflow, re-analysing a published dataset of zebrafish brain neuronal traces by combining traces and brain region to extract quantitative metrics per region.

Methods

Architecture

GeNePy3D[14,15] was designed with any computational-minded life scientist as target user, to provide a simple and homogeneous API. GeNePy3D consists of four main objects (Figure 1) corresponding to four basic geometrical objects of interest: Points (cells or intracellular object positions...), Curve (particles tracks, neurite branches, microtubules...), Tree (neuronal traces, dividing cell tracks) and Surface (cell surface or other tissue level structure...). Each of them has its own attributes, functions and I/O. We provide ways to transform between them, (decomposing a Tree into sequences of Curve, or converting Points into the Surface that enclose them). Interaction between objects of the same/different classes are also available (optimal transport-based distance between two Points, intersection between Curve and Surface, etc.) Altogether, GeNePy3D offers a unified and seamless way to analyse complex geometrical biological data.

Implementation

GeNePy3D is implemented in Python, taking advantage of a high-level programming language with simple syntax and many open-source packages. We reused algorithms and functions available from various recognised packages when possible, and developed our own implementation when needed, within a unique interface. Most of the packages we link to are available from the Python package Index (PyPi) and can be easily installed via Python package manager (pip). Figure 1 lists out some functions with colors denoting the package used. Beyond standard ones, more specific ones includes AnyTree for tree manipulation, TriMesh for surface manipulation or ScikitLearn for machine learning tasks. Other feature are listed as optional, as they come from harder to install or less recognized

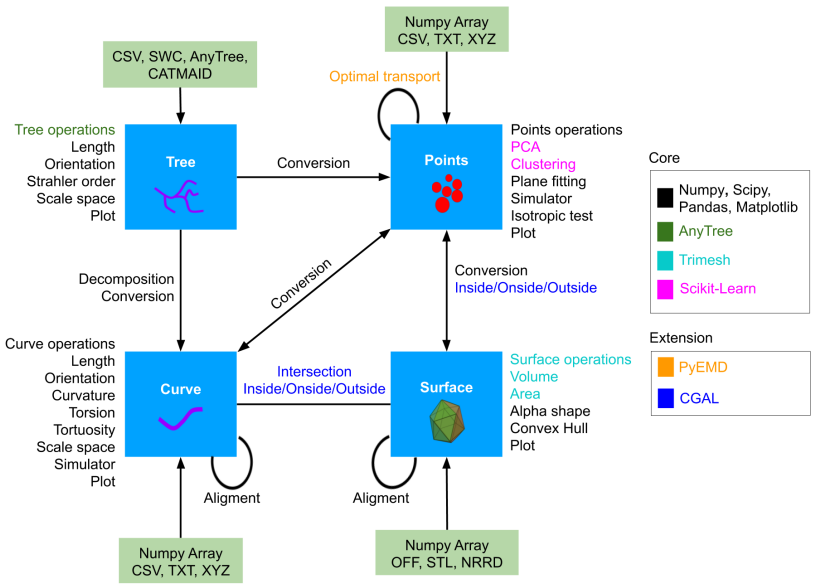

Figure 1. GeNePy3D architecture. The library is structured around four main classes for four principal geometrical objects, and propose various functions acting on them or converting between them, either implemented anew or linking to recognized library.

sources, including the C++ library CGAL, only partially available in Python, for generic object interaction in 3D, or the optimal transport method implemented in PyEMD. Some original development available in GeNePy3d include an algorithm to compute local 3D scale we recently published[16]. Many common input/output formats are supported including SWC for Tree, CSV, XYZ for Points/Curve and STL and OFF for Surface. We release the library in two packages for licensing issues (see licenses below).

Operation

GeNePy3D works with Python 3.6. Details of the specific software requirements, documentation including the installation instruction and Python notebooks examples can be accessed via https://genepy3d.gitlab.io. Example pipelines using GeNePy3D are run using Jupyter notebooks. To ease the use and deployment of GeNePy3D we provide ready to use docker containers at https://gitlab.com/genepy3d/genepy3d_dockers.

Use case

To exemplify the use of GeNePy3D[14,15], we reanalyzed a recently published dataset containing up to 2000 traced neurons across the whole brain of larval zebrafish[17]. The authors annotated 36 symmetric regions and established a connectivity atlas for the neurons within these regions. Figure 2A illustrates a possible workflow using GeNePy3D for reanalyzing the dataset. The inputs consist of neuronal traces in SWC formats and a 3D volume in NRRD format containing different annotated labels for the 36 brain regions. The traces are imported into GeNePy3D under Tree objects, while the regions are reconstructed into

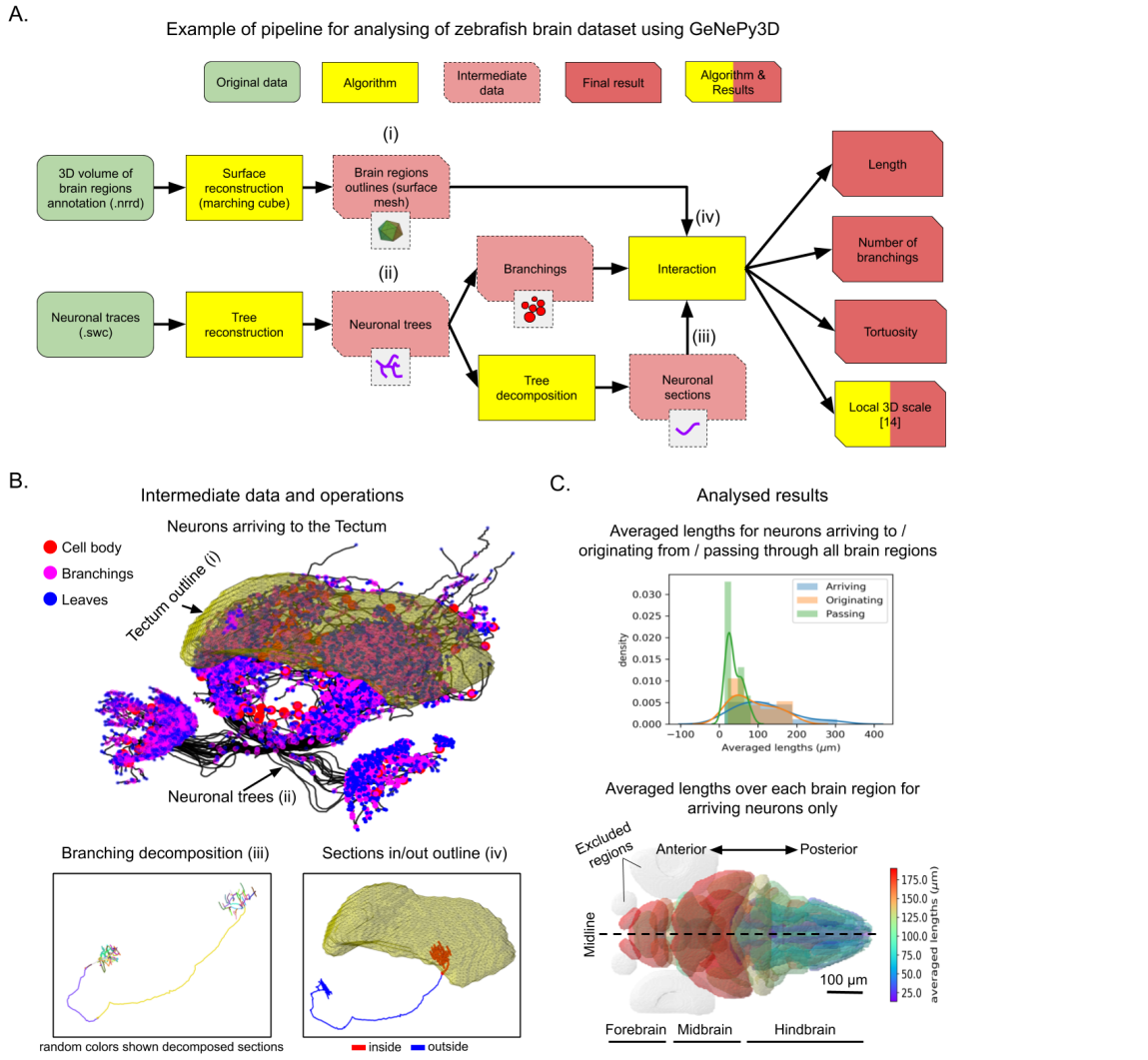

Figure 2. Example workflow for analysing of Larval zebrafish brain dataset[17] with GeNePy3D. (**A**) Workflow schema. (**B**) Example of intermediate data and operations from the workflow: outline surface of the Tectum and all neurons arriving to it (top), decomposition of a neuronal tree into sections (displayed with random colors) based on branching positions (bottom left), and computing of neuronal sections inside/outside the Tectum (bottom right). (**C**) Resulting quantifications: distribution of average neuronal lengths for groups of neurons arriving to/originating from/passing all brain regions (top), and heat map of averaged neuronal lengths over each brain region for group of neurons arriving to the brain regions (bottom). The regions with small number of arriving neurons (< 10 neurons) are excluded (in gray). The letters (i-iv) in (**B**) illustrate some steps in (**A**).

Surface objects using marching cube algorithm. Figure 2B top illustrates the outline of the Tectum along with all neuronal traces arriving to this brain region. We then extracted branching point positions from the neuronal traces (Tree→Points), decomposed them into sections (Tree→Curves) and checked whether the branching points or curve sections lies within or outside each region (interaction with Surface). Examples of decomposing the traces, computing sections inside and outside the Tectum region are shown in Figure 2B bottom. Finally, we measured within the brain regions neuronal lengths, number of branching points, tortuosities (proportion of length over distance between two end points of the curve), and local 3D scales[16] (scale at which the curve transforms to 3D).

Part of the resulting quantification obtained are shown Figure 2C. The top graph shows a longer neuronal length on averaged for groups of neurons arriving to and originating from the regions compared to ones passing through. Figure 2C bottom shows a map of the averaged neuronal length for each brain regions for arriving neurons showing that neurons coming from fore- and midbrain are much longer than those from hindbrain. Detail of all processing steps and additional quantified results can be found at https://gitlab.com/genepy3d/genepy3d_examples/-/tree/master/zebrafish_atlas.

Conclusions

The advent of machine learning and developments in biological imaging is leading to numerous geometrical datasets, and GeNePy3d[14,15] aims at enabling complex analysis workflows based on those objects. But as in other aspects of bioimage informatics, the key will be for the community to work together and define common formats and structures for region of interests and geometric objects to ease the interactions between the various visualisation, data management or analysis tools, and convert raw images to biological knowledge. GeNePy3d is ready to become a component of that ecosystem.

Data availability

Source data

The data used for Figure 2 has been published in https://fishatlas.neuro.mpg.de. To download the traces, we choose 'single axons', 'connect without logging in', chose 'Kunst *et al*. 2019' in publications; once all neurons are loaded the download option appears.

Software availability

GeNePy3D is hosted at: https://genepy3d.gitlab.io and easily installable through the PyPi tool.

Source code available at: https://gitlab.com/genepy3d.

Archived source code at time of publication:

- GeNePy3D: https://doi.org/10.5281/zenodo.4269466[14].
- GeNePy3D_GPL: https://doi.org/10.5281/zenodo.4269484[15].

License: The library is distributed as two packages. The main package GeNePy3D[14] is under a BSD 3-Clause Licence, while features that necessitate linking to GPL-licensed code are distributed separately in GeNePy3D_GPL[15], under the GNU General Public License v3.0.

We wanted to release GeNePy3D under a BSD license but could not avoid the use of some GPL license software, forcing us to such a solution. Practical consequences should be minimal in most circumstances thanks to modern python package management.

The source code for the analysis of Figure 2 is available at https://gitlab.com/genepy3d/genepy3d_examples/-/tree/master/zebrafish_atlas.

Acknowledgements

The authors wish to acknowledge Jean Livet, Emmanuel Beaurepaire and Katherine Matho for fruitful discussions and the collaboration that gave the incentive to develop this library, and Debora Keller-Olivier for reading the manuscript. This publication was supported by COST Action NEUBIAS (CA15124), funded by COST (European Cooperation in Science and Technology.

References

1. Moen E, Bannon D, Kudo T, *et al.*: **Deep learning for cellular image analysis.** *Nat Methods.* 2019; **16**(12): 1233–1246.
PubMed Abstract | Publisher Full Text

2. Chessel A, Dodgson J, Carazo-Salas RE: **Spherical spatial statistics for 3D fluorescence video-microscopy.** In *Proceedings-International Symposiumon Biomedical Imaging.* 2012.
Publisher Full Text

3. Lagache T, Grassart A, Dallongeville S, *et al.*: **Mapping molecular assemblies with fluorescence microscopy and object-based spatial statistics.** *Nat Commun.* 2018; **9**(1): 698.
PubMed Abstract | Publisher Full Text | Free Full Text

4. Chessel A, Cinquin B, Bardin S, *et al.*: **Computational Geometry-Based Scale-Space and Modal Image Decomposition Application to Light Video-Microscopy Imaging.** volume 5567 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
Reference Source

5. Kashiwagi Y, Higashi T, Obashi K, *et al.*: **Computational geometry analysis of dendritic spines by structured illumination microscopy.** *Nat Commun.* 2019; **10**(1): 1285.
PubMed Abstract | Publisher Full Text | Free Full Text

6. Bates AS, Manton JD, Jagannathan SR, *et al.*: **The natverse, a versatile toolbox for combining and analysing neuroanatomical data.** *eLife.* 2020; **9**: e53350.
PubMed Abstract | Publisher Full Text | Free Full Text

7. Baddeley A, Turner R: **spatstat: An R Package for Analyzing Spatial Point Patterns.** *J Stat Softw.* 2005; **12**(6).
Publisher Full Text

8. The CGAL Project: **CGAL User and Reference Manual.** CGAL Editorial Board,

5.1 edition, 2020.
Reference Source

9. Virtanen P, Gommers R, Oliphant TE, *et al.*: **SciPy 1.0: fundamental algorithms for scientific computing in Python.** *Nat Methods.* 2020; **17**(3): 261–272.
PubMed Abstract | Publisher Full Text | Free Full Text

10. **Anytree.** 2016.
Reference Source

11. Dawson-Haggerty, *et al.*: **Trimesh.** 2019.
Reference Source

12. Schindelin J, Arganda-Carreras I, Frise E, *et al.*: **Fiji: An open-source platform for biological-image analysis.** *Nat Methods.* 2012; **9**(7): 676–682.
PubMed Abstract | Publisher Full Text | Free Full Text

13. de Chaumont F, Dallongeville S, Olivo-Marin JC: **ICY: a new open-source community image processing software.** In *IEEE International Symposiumon Biomedical Imaging(ISBI).* 2011.
Publisher Full Text

14. Phan MS, Chessel A: **Genepy3d.** October 2020.
http://www.doi.org/10.5281/zenodo.4269466

15. Phan MS, Chessel A: **Genepy3d_gpl.** October 2020.
http://www.doi.org/10.5281/zenodo.4269484

16. Phan MS, Matho K, Livet J: **Emmanuel Beaurepaire, and Anatole Chessel. A scale-space approach for 3D neuronal traces analysis.** *bioRxiv.* 2020.
Publisher Full Text

17. Kunst M, Laurell E, Mokayes N, *et al.*: **A Cellular-Resolution Atlas of the Larval Zebrafish Brain.** *Neuron.* 2019; **103**(1): 21–38.e5.
PubMed Abstract | Publisher Full Text

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 2

Reviewer Report 24 June 2021

https://doi.org/10.5256/f1000research.57818.r87836

✔ **Virginie Uhlmann**
Wellcome Genome Campus, European Bioinformatics Institute (EMBL-EBI), Cambridge, UK

Thank you for providing in-depth clarifications to all of my points and amending the paper accordingly. I look forward to following up on future GeNePy3D developments.

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioimage analysis

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 16 March 2021

https://doi.org/10.5256/f1000research.30274.r80682

✔ **Yizhi Wang**
Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Arlington, Virginia, USA

The author design a Python package to analyze the various geometric objects extracted from microscopy images of the brain. The package combines tools from computational geometry,

spatial statistic, and other fields into a unified API. The usefulness of the package is demonstrated by an example of re-analyzing the zebrafish brain region and neuron traces.

The tool should be very useful in gaining insights from the microscopy imaging data. The package is adequately documented, and the Docker file makes it easier to use for many users.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioimage informatics, machine learning

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 15 December 2020

https://doi.org/10.5256/f1000research.30274.r75563

?

Virginie Uhlmann
Wellcome Genome Campus, European Bioinformatics Institute (EMBL-EBI), Cambridge, UK

The authors describe GeNePy3D, a Python toolbox that facilitates the processing and quantification of geometrical objects extracted from images. The goal is for this package to bring together the methods provided in various existing geometrical processing libraries such as PyEMD, Trimesh, or AnyTree. The provided example reanalyzing the larval zebrafish connectome

dataset of Kunst *et al.* is well-chosen and convincing. Overall, GeNePy3D is an absolutely relevant software that is likely to be impactful in the bioimage analysis community.

Major comments:

- The authors mention Fiji and Icy, which are indeed widely used to quantify geometry from bioimages. They however do not spell out clearly how they envision GeNePy3D to interact with these GUI-based (and Java-based!) alternatives. More details on this aspect should be provided.

- The GitHub repo should be better documented, in particular when it comes to describing the methods used in functions such as a curve or surface alignment (hence the "sufficient details of the code" point above flagged as "partly").

- The article's title emphasizes the "large scale" aspect of GeNePy3D. From the implementation's description, I am under the impression that the scaling ability of this package comes "for free" from the fact that numpy, scipy, pandas, etc all scale extremely well. If there is more and specific efforts have been put into developing methods in a specific manner so as to allow processing of large datasets, this aspect should be discussed in more detail in the implementation section. If not and if this really is simply a consequence of using well-developed Python libraries, I would suggest downplaying a bit the "large scale" aspect of the toolbox.

Minor comments:

- The package name, GeNePy3D, is poorly chosen for two reasons: 1) the meaning of the acronym is unclear, 2) there exists already at least 3 different Python packages called genepy, all containing entirely unrelated algorithms. I would strongly suggest coming up with a more self-descriptive and less overused name.

- I am no expert in software licensing, but I worry that the two-licenses solution adopted here may be unnecessarily confusing for the end-user. Wouldn't there be a way to package the entirety of the library under a single BSD3/GPL license, or license it all under the most restrictive of the two if applicable? If having two repos under two licenses really is the only possible solution, some clear explanation of why this is so should be provided in the article.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioimage analysis

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 14 Jun 2021

Anatole Chessel, Ecole polytechniqe, Institut polytechnique de Paris, Palaiseau, France

The authors describe GeNePy3D, a Python toolbox that facilitates the processing and quantification of geometrical objects extracted from images. The goal is for this package to bring together the methods provided in various existing geometrical processing libraries such as PyEMD, Trimesh, or AnyTree. The provided example reanalyzing the larval zebrafish connectome dataset of Kunst *et al.* is well-chosen and convincing. Overall, GeNePy3D is an absolutely relevant software that is likely to be impactful in the bioimage analysis community.

We thank the reviewer for this positive overall assessment.

Major comments:

- The authors mention Fiji and Icy, which are indeed widely used to quantify geometry from bioimages. They however do not spell out clearly how they envision GeNePy3D to interact with these GUI-based (and Java-based!) alternatives. More details on this aspect should be provided.

Indeed Fiji and icy, and more generally the java-based ecosystem of bioimage informatics tools is very important in the community. The python-based ecosystem is becoming very important as well, with developments around napari and scikit image for example. Linking those two ecosystems is very important to allow heterogeneous workflows, and various solutions have been proposed such as https://github.com/imagej/pyimagej; but we feel this question is generally beyond the scope of this work, which presents a new python library and thus is usable wherever python is usable. For geometrical objects specifically, the development of a common exchange format would go a long way into allowing interactions between genepy3d and the java world, and is definitely part of the roadmap of future development, in genepy3d or elsewhere.

- The GitHub repo should be better documented, in particular when it comes to describing the methods used in functions such as a curve or surface alignment (hence the "sufficient details of the code" point above flagged as "partly").

We improved and reformatted the documentation, to make it easier to access and follow. The main documentation, with tutorials and generic information is still at genepy3d.gitlab.io. The reference documentation of the API is now hosted on readthedocs, separately for the two

packages: genepy3d.readthedocs.io and genepy3d_gpl.readthedocs.io.
Specifically for the alignment functions: that aspect is still essentially lacking in the library as is it, mainly because we had no occasion to implement it. There is one 'align' function for curves, which is quite specific and is now better documented. There is a huge bibliography on those topics and many available implementations; it is something that would be very useful to have in the library and that we do plan to tackle in the future. We removed explicit mention of alignment algorithms in the text, to avoid misleading the reader.

- The article's title emphasizes the "large scale" aspect of GeNePy3D. From the implementation's description, I am under the impression that the scaling ability of this package comes "for free" from the fact that numpy, scipy, pandas, etc all scale extremely well. If there is more and specific efforts have been put into developing methods in a specific manner so as to allow processing of large datasets, this aspect should be discussed in more detail in the implementation section. If not and if this really is simply a consequence of using well-developed Python libraries, I would suggest downplaying a bit the "large scale" aspect of the toolbox.

The original applications were on large scale images, hence the 'large scale' in the title, but it is true that the genepy3d library itself does not provide any specific development for large scale processing. We propose to drop 'large scale' from the title to reflect that point, if that is possible at this stage of the publication process.

Minor comments:

- The package name, GeNePy3D, is poorly chosen for two reasons: 1) the meaning of the acronym is unclear, 2) there exists already at least 3 different Python packages called genepy, all containing entirely unrelated algorithms. I would strongly suggest coming up with a more self-descriptive and less overused name.

Finding a name for a project is complicated and we agree that, in retrospect, better choices most likely exist. It originally stood for Geometry of Neuron in Python in 3D. It might be clearer when heard (something like 'geeneepaï'). We may indeed change it in the future, if it gains traction and the user and developer base expands; when going out of alpha for example.

- I am no expert in software licensing, but I worry that the two-licenses solution adopted here may be unnecessarily confusing for the end-user. Wouldn't there be a way to package the entirety of the library under a single BSD3/GPL license, or license it all under the most restrictive of the two if applicable? If having two repos under two licenses really is the only possible solution, some clear explanation of why this is so should be provided in the article.

We would have welcomed another solution but we do not know of a clean, legal way to mix, in a project under BSD, both GPL and BSD code. Since we want the bulk of the library to be under a BSD license for compatibility with the rest of the python ecosystem (see also this argument for BSD in scientific code: https://www.astrobetter.com/blog/2014/03/10/the-whys-and-hows-of-licensing-scientific-code/), this means corralling out GPL bits. We will try to make them as small as possible, and the added complexity is mitigated by modern python package management, which makes it trivial to install additional packages. We added a sentence to explain this reasoning in the text.

Competing Interests: No competing interests were disclosed.