Author Guidelines for Software Submissions

Content

- Mission and Scope of Software Submissions
- Types of Software Submissions
- Requirements for Software Submissions
- Preparing your Software submission
- Reviewing and selection criteria
- Sample Research Object

Mission and Scope of Software Submissions

Aperture recognizes the essential importance of software to the advancement of science, especially in neuroimaging research. When submitting Software Research Objects to Aperture, authors are welcome and encouraged to think broadly beyond the traditional PDF of “Software” and “Toolbox” papers.

As for any Aperture research objects, software submissions will be selected and reviewed based on their quality, impact, relevance, and their ability to increase connectivity among researchers in the field. The software should fall within the interest of the OHBM mission to advance the understanding of the anatomical and functional organization of the human brain with a strong focus on neuroimaging. Research Objects involving animal studies are also welcome.

Submissions that report minor plug-ins for existing software or minor advancements of existing algorithms with limited breadth or impact are unlikely to be selected for peer review. However, submissions representing important changes (even if small) in a previous toolbox/library, but with great impact and utility, are very much welcome.

Almost every active researcher in the field of neuroimaging develops software in one way or another, and many research labs have their specialized in-house software. Aperture aims to publish code that is likely to have a broader utility (beyond one’s own lab). Due to the extensive workload for reviewers to assess code, not every submission may make it to peer review unless it is deemed to be of broader utility to the community.
Types of Code Submissions

Types of code submissions can be, but are not limited to the categories below:

**Software and toolboxes:** Authors can submit stand-alone software packages and toolboxes for neuroimaging analyses.

Examples: AFNI, TDT Decoding toolbox, PMC Toolbox, CVMANOVA toolbox, etc.

**Notebooks and pipelines:** Authors can submit Notebooks (E.g., Jupyter, R) and scripts (e.g. Python, Matlab/Octave markup code). While Aperture would prefer open source software (i.e., Python, Octave), Matlab is welcome too (as it can be widely distributed and can be compiled). The editorial team will consider submissions of non-open software on a case by case basis. Notebooks can be educational in nature, they can contain pedagogical tutorials, they can reveal insights into specific analysis steps, and/or they can be entire processing pipelines. Submissions should include as much supporting information as possible. Documentation should be included within the notebook/mockup file.

Examples: BrainWavelet Matlab filters, Jupyter Notebook for simulating fMRI PSF, Python Pipeline HyPyP for hyperscanning, BV Notebooks, PrF fitting in Python, etc.

**Apps and Web tools:** Submissions can be smartphone apps for Android/iPhones and/or cloud-based or browser-based software. Such apps could, for example, serve an educational purpose regarding neuroimaging data acquisition, data reconstruction, brain science, or other topics.

Examples: Phone apps Kspapp, BrainTutor. Web-based analysis tools: MRI-Cloud, GiraffeTools, BioimageSuite, etc.

**Brain imaging data acquisition and reconstruction code:** Submissions can be open code related to MRI sequences, MR-signal simulation, and/or MRI image reconstruction (or other brain imaging modalities). Non-open sequence code that is confined to the programming environment of commercial vendors does not fall within the scope of Aperture’s code submissions.

Examples: ODIN, Pulseq, TOPPE, JEMRIS, Gadgetron programs, etc.
Requirements of Software Submissions

Authors are encouraged to submit software that is open-source and accessible via a trusted repository. Please link to this repository using the “Research Object Types” Form in the “Forms” tab on the Submission Platform. While any software that may have significant impact on the field may be submitted, Aperture encourages authors to submit software that fulfills the following criteria. Deviations from these criteria should be explained in the cover letter.

1. **Open source**: Software should be Open Source as defined by the [https://opensource.org/osd](https://opensource.org/osd). If the software is not or cannot be open source, you are asked to provide the reason and the implications/interests involved.

2. **License**: Aperture software submissions require that the open source code is legally usable with an explicit license attached. Open source code is encouraged to be under an Open Source Initiative approved license. [Full list of acceptable licenses](#).

3. **No cost to readers**: The software should be freely available for the reviewers and readers of Aperture. The software should ideally be executable in a free coding environment. Software submissions for non-free environments (e.g. MATLAB, IDL, etc.) will be considered, but may make reviewing more difficult.

4. **Tested and Validated**: At this time Aperture will only accept submission of software that have been heavily tested and validated. Authors should provide a detailed description of the test results and share the tested data.

5. **Dependencies**: All dependencies of the submitted code must be documented, they must be listed with their respective software version, and they must be openly available too.

6. **Timing of software availability**: The software needs to be available at the time of submission, so that Reviewers can test the software and potentially inspect the code.

7. **Code versioning**: The code should be versioned. Only the original time-stamped Research Object at the time of submission will be peer-reviewed.

8. **Reliable software hosting**: The software should be hosted on a trusted repository that is both reliable and will maintain open access on a long-term basis (for at least 5 years). Authors should host the software in good faith that users will be able to access the software in the years to come. If the hosting site goes off-line, the authors need to resubmit/edit the Research Object. Hosting code via lab-operated, or personal websites is not recommended.

9. **Anonymous access**: The software should be available for peer-review without the necessity of the reviewer to disclose her/his identity. Examples of reliable and acceptable code hosting platforms are: GitHub, Zenodo, GitLab, SourceForge, NITRC, Figshare, Docker Hub, etc.
10. **Accompanying data:** Authors may submit any relevant data along with the software submission (i.e., as examples or tutorials). The accompanying data need to comply with the General Data Protection Regulation (GDPR) of the EU or the laws governed by the country where informed consent was obtained. The Authors should explain the reasons for sharing the data along with the code and any relevant implications to consider.

11. **Code efficiency:** Inefficient software has financial costs for users and environmental costs for us all. Authors are encouraged to optimise their code prior to submission such that these costs are minimised.

---

**Preparing your Software submission**

Authors will log onto the Aperture Journal Submission Dashboard to begin the submission process. When you are ready to submit your Research Object, please use your OHBM member username and password to log into the Aperture submission page. If you are not an OHBM member, you will have the option to create an Aperture profile.

Once we receive your Research Object, you will be notified via email. You will not be able to edit your Research Object at any time during the review process unless directed to do so by the Journal Manager.

**Cover Letter:** Please include a cover letter with your submission. You will be asked to upload your cover letter as a separate document in the submission platform. This letter should be addressed to Aperture’s Editorial Team and include any relevant information you feel will aid the Editors in the consideration of your work.

**Title:** Titles are limited to 300 characters and should contain major keywords. Please do not use Subscript, Superscript, boldface and/or italics in your title.

**Authors:** The first author or first listed author should be the main contributor of any code or software submitted. Contributions from co-developers should be acknowledged.

**Comprehensive Summary:** All Software submissions must include a comprehensive summary that will serve as a “front page” of your submission. This summary should describe the main purpose of the software to Aperture’s readership and should place the software into context. This is different from the cover letter to the Editor. Keep the lay-public in mind as this summary will be the first thing audiences will see when visiting your publication on the Aperture site.
There is no particular requirement regarding the format or style of your summary (feel free to structure your summary as you see fit). Enter your comprehensive summary text in the Summary/Full Research Object step in the submission process.

For software submissions, we recommend that your summary should cover the following (if applicable):

- **Visual summary**: Consider this as a “graphical abstract” of the work. Choose a figure that characterizes the main purpose/achievement of the software. The visual summary should “read” well and avoid elements that may distract the reader. See the figure guidelines for specific image requirements.

- **Relevance of the software**: The summary should include a clear description about the potential relevance and significance of the software to the field. It should be clear where it can be applied. Explicit mentioning of applications and types of application scenarios are encouraged.

- **Purpose of the software**: The summary should clearly state the purpose of the software. It should be mentioned what kind of problem it solves and what its (research) application is. For example, is the software intended as a core tool to be integrated in analysis workflows? Is the software primarily instructive or pedagogical in nature?

- **Uniqueness and Novelty**: If there exist alternative tools solving a similar problem, this should be mentioned in the summary, stating any key differences between these.

- **Generalisability and validity**: It should be clear from the summary whether the software is applicable to a wide spectrum of conditions.

- **Target audience of the software**: It should be clear from the summary who the intended audience of the code is.

- **Quality of the software**: The summary should contain a brief statement regarding the quality of the software (examples, documentation, tests, validations, continuous integration). If applicable, authors should briefly state planned extensions, code maintenance, how users can report issues, and/or future planned developments of the software. Known limitations, in terms of methods or data type, should be mentioned.

- **Link to code**: The summary should include a link to the hosting platform of the code and/or the project's web site.

- **Users Manual**: Submission of software should include a reference to the softwares manual. It should be clear to the users how to install and use the software.

- **Requirements and dependencies**: The summary should also contain the main requirements for the software, including operating system (E.g., platform independent or Unix only) and programming language (E.g., C++, or Python 2 and higher).
• **Other links (if applicable):** Additional supporting materials (e.g. example data) can be described in the summary.

• **References:** If software is reliant on other existing neuroimaging software, this should be clearly described.

---

**Reviewing criteria**

- Reviewers are encouraged to provide their main impression of the software, including whether it is in line with the mission of Aperture (including quality and impact of submissions), and whether it adds to an already existing software base.
- Reviewers are invited to summarize the aim and context of the software in a short paragraph. This shows the editor they have read and understood the software submission.
- Reviewers should make an assessment of the software and whether it follows the intended scope of the Software submission as expressed in the comprehensive summary. For example, if authors claim that the code is readable and generally applicable, the reviewers are tasked to assess these claims.
- The reviewer should download, install, and execute the code and test it in reasonable scenarios. The reviewer is not required to read every line of code. However, the reviewer is welcome to assess any parts of the software as they feel necessary.
- Reviewers should assess both the technical soundness as well as the scientific appropriateness of the implemented algorithms.
- Reviewers are allowed to comment on the coding style, keeping in mind that individual programmers might have different coding conventions. It is possible that inelegant code can still have great value if it performs a certain function well.
- Reviewers are invited to comment on the level of documentation, presence of developer guidelines (for new contributions), ease of extensibility, and/or whether the code has high (or low) efficiency (e.g., execution time, CPU usage, RAM usage, etc.). However, reviewers should bear in mind that code may have substantial value even if it is not doing its very best on one or more of these factors.
- Reviewers should not feel responsible to look for bugs in the software. However, if bugs are identified, they can be reported to the authors. If the general code is sound, but let down by too many bugs for general applications, recommend to the editor that the author(s) have their code validated on more test cases.
- While conventional plagiarism guidelines of research articles' text are not straightforwardly applicable to code, reviewers should indicate if they suspect plagiarism of code or fraud, or have any other ethical concerns.
• Reviewers are invited to see if Authors have credited all contributors appropriately (either with authorship or clear acknowledgement). Authors may be given a chance to explain any exceptions.
Sample Research Object:

Title: CleanTheBrain: Browser-based noise removal of brain data
Authors: DaVinci, L.
Link to code: https://github.com/cleanthebrain
Comprehensive summary:

CleanTheBrain is an open-source browser-based analysis tool to remove unwanted signals in brain imaging data. Virtually all brain imaging data are limited by unwanted signal sources (a.k.a. noise). Here, we present an analysis software, CleanTheBrain, which can apply powerful noise cleaning tools on brain imaging data for a number of neuroimaging modalities. The cleaning engine decomposes the brain data and then filters all signal components that do not match features of neurally driven activity. As such, CleanTheBrain removes physiological noise, signal/image artefacts, and it works particularly well to remove random thermal noise. The techniques that are implemented in CleanTheBrain are documented elsewhere (see https://github.com/cleanthebrain); what we provide here is the implementation of a streamlined, easy-to-use tool that has been extensively tested on a variety of datasets.

The code has been tested and validated on structural MRI, functional MRI (1,2) and EEG (3). The code is well documented and written to be easily readable and editable by the users. It can be freely and anonymously used under the GNU (GPL-3.0) license. It runs in any web browser of all operating platforms and mobile devices, and has dependencies to HTML4 (or higher). User instructions with tutorials and example data to test the tool are available on https://github.com/cleanthebrain/cleanthebrain. The code is version controlled and long term supported. Users are invited to report bugs and request features via https://github.com/cleanthebrain/cleanthebrain/issues.

References: