German Medical Data Sciences 2021: Digital Medicine: Recognize - Understand - Heal
R. Röhrig et al. (Eds.)

doi:10.3233/SHTI210556

openMNGlab: Data Analysis Framework for Microneurography – A Technical Report

Fabian SCHLEBUSCH [a], Frederic KEHREIN [a], Rainer RÖHRIG [a], Barbara NAMER[b, 1] and Ekaterina KUTAFINA [a, c, 1, 2]

[a] *Institute of Medical Informatics, Medical Faculty, RWTH Aachen University, Aachen, Germany*
[b] *Junior Research Group Neuroscience, Interdisciplinary Center for Clinical Research Within the Faculty of Medicine, RWTH Aachen University, Aachen, Germany*
[c] *Faculty of Applied Mathematics, AGH University of Science and Technology, Krakow, Poland*

Abstract. openMNGlab is an open-source software framework for data analysis, tailored for the specific needs of microneurography – a type of electrophysiological technique particularly important for research on peripheral neural fibers coding. Currently, openMNGlab loads data from Spike2 and Dapsys, which are two major data acquisition solutions. By building on top of the Neo software, openMNGlab can be easily extended to handle the most common electrophysiological data formats. Furthermore, it provides methods for data visualization, fiber tracking, and a modular feature database to extract features for data analysis and machine learning.

Keywords. Microneurography, neurophysiology, membrane potential, data science

1. Introduction

Microneurography (MNG) is a domain of electrophysiology investigating the responses of peripheral nerve fibers in awake humans using needle electrodes inserted into a peripheral nerve. MNG studies are crucial for understanding sensory coding and helping patients with neurological conditions, such as neuropathic pain [1]. Before the analysis of the neural code of the very thin C-fibers, which are responsible for signaling pain, itch and temperature, is possible, individual nerve fiber signals are manually identified, aligned to an individual nerve fiber and annotated in the recording with a semi-automatic "marking" technique [1] based on the activity-related slowing of their conduction velocity.

Fully automatic solutions for detection and sorting of action potentials to enable tracking of responses from a single fiber over time, are necessary to improve the

[1] Barbara Namer and Ekaterina Kutafina contributed equally as senior authors.

[2] Corresponding author: Dr. Ekaterina Kutafina, PhD. Address: Institute of Medical Informatics, Medical Faculty, RWTH Aachen University, Pauwelsstraße 30, 52074 Aachen., Germany. Email: ekutafina@ukaachen.de.

efficiency of data analysis. However, before the methodologies known from other domains of computational neuroscience (e.g., spike clustering) can be applied, there is a need to develop software solutions for loading, processing, and storing of MNG data.

Furthermore, scientific research relies on cooperation between different work groups. Exchange of data but also practices and software for its analysis leads to new findings. However, the exchange of data and especially source code for analysis is in practice often impeded by differences in the used hardware, software, and study protocols. Therefore, to overcome these difficulties, we developed a software framework called openMNGlab. The development of openMNGlab was guided by the FAIR (findability, accessibility, interoperability, reusability) principles [2]. Our Python-based framework has a modular design and uses the Neo package [3] as it provides efficient data structures for electrophysiological data. Moreover, by using the Neo library, we try to ensure compatibility with open-source electrophysiology software such as SpykeViewer [4] or Elephant [5].

The current version of openMNGlab already provides fundamental methods for MNG data analysis, with a focus on machine learning that we believe might lead to novel neuroscientific insight through using the full potential of large data collections.

2. State of the Art

Currently, a variety of different recording software for MNG exists, each with their own strengths, weaknesses, and file formats. Further adding to the problem, some software products may be discontined. In the absence of software that can read these specific file formats, this would lead to a loss of data.

The electrophysiology community is developing solutions for these problems. The Neo software package [3] is a powerful Python library to load electrophysiological data from different file formats and organize the data in memory. However, it is not specific to the requirements of MNG data analysis. To the best of our knowledge, there is no other software which aims at providing loading functionality for common file formats used in MNG while also being specifically tailored towards MNG-specific data analysis.

3. Concept

We aim to provide a software framework for analysis of data which has been generated in MNG experiments. To enable data analysis, the framework has to offer functionality for loading data from different sources, e.g., MNG data acquisition tools. Furthermore, the data should be available in a unified format after loading, so that the source code for data analysis is independent of the data origin and file format.

Apart from that, our software must be tailored to MNG by providing methods, classes and visualizations, which are specific to MNG research. This includes modeling of MNG-specific entities such as action potentials, electrical or mechanical stimulation, or nerve fibers attributes.

4. Implementation

We implemented openMNGlab in the Python programming language. The software framework is an open-source project and publicly available[3]. As our software framework is implemented in Python, it will run on different operating systems and we can resort to a plethora of existing software packages.

4.1. Input and Output

We build upon the Neo package for processing of electrophysiological data which provides classes for importing electrophysiology data from different file formats. File formats supported by Neo include, e.g., Spike2 and OpenEphys formats which are particularly relevant for MNG data analysis in our associated research groups. In a first step, our framework supports the import of Spike2 recording files using the corresponding Neo importers. Import methods for other file formats can be added at any time if necessary. Given Neo's IO class for a certain file format, implementation of a simple method which performs minor pre-processing steps such as proper annotation of the Neo objects, suffices (see Figure 1).

Furthermore, our software can import Dapsys [6] recordings, albeit not in the Dapsys-specific .dps format as we have no information about the structure of this format. Instead, recordings must be exported to comma separated value (csv) files using the Dapsys software. Particularly on German operating systems due to incompatibilities with the local number format, additional pre-processing is necessary so that the csv-files are valid. The (fixed) files are then read by an import method and a Neo-segment with corresponding channel objects is generated.

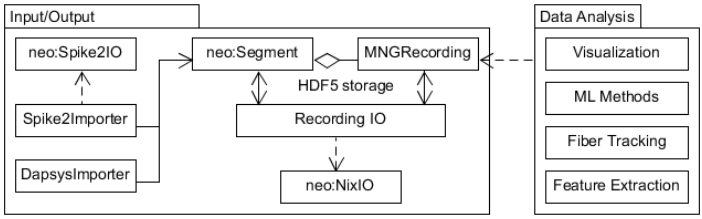

Figure 1. UML class diagram of the openMNGlab classes for loading and storing of recordings. The Spike2 importer uses Neo's Spike2IO to load the raw format as a Neo segment and performs further pre-processing steps. The Dapsys importer directly instantiates a Neo segment with the correct attributes. The resulting Neo segments are wrapped by the MNG recording class, which is used as an abstraction layer for the data analysis methods. The recording IO manages reading and writing of recordings in the HDF5 format, and transparently uses Neo's NixIO class.

After a recording has been loaded into memory as a Neo model, it can be manipulated and analyzed. If any changes should be made permanent, the Neo model can be written to the disk in the HDF5 format. Such files can then again be loaded into memory. HDF5 is a standardized format for storage of scientific data and may be supported for many years. Therefore, we recommend that all recording files should be

[3] The OpenMNGlab git repository is available at https://git.rwth-aachen.de/fabian.schlebusch/imi-neuro

loaded once and stored to HDF5, to ensure long-term readability and abstraction from the original formats.

4.1.1. Transformation to MNG-specific format

To facilitate analysis of MNG recordings, we provide a more specific data format which transparently uses Neo for efficient data storage and access. Data which has been read using Neo's IO classes typically needs to be further processed and made conformant with our unified data representation for MNG. In case of Spike2 this includes, e.g., splitting event channels by their different event markers into separate event channels. The required preprocessing depends strongly on the specifics of the loaded file format.

Most importantly, all channels must be annotated with a unique channel identifier and a channel type. Possible channel types include electrical stimulation channels, mechanical stimulation which is used to identify the receptive fields in MNG, or raw signals. Such semantic information is not represented in the Neo data structure but is important for MNG data analysis. Therefore, we emphasize these differences in our MNG-specific data representation.

To facilitate access to the channels by their assigned ID, all import methods are required to maintain an ID map. This ID map links the channel names as displayed in, e.g., Spike2 to the identifiers assigned by the import method.

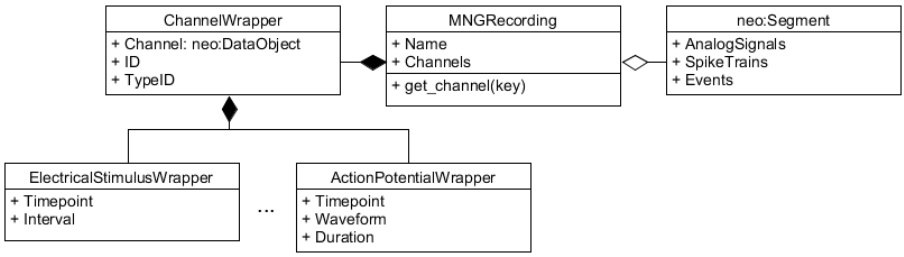

Figure 2. Simplified class diagram of openMNGlab's wrapper classes. We omit wrappers for mechanical stimulation, additional electrical stimulation etc. here. The wrapper classes provide simple access to single entities or events of an MNG recording. Action potential wrappers can also be used for accessing the feature database (see section 4.3). New wrapper classes for more specific types of events and entities can and will be added.

4.2. MNG-specific Data Format and Classes

We provide wrapper classes for the Neo models (see Figure 2) to enable easy access to the relevant entities in MNG data analysis. There are wrapper classes for the recording itself, for action potentials, electrical stimulation pulses, as well as other relevant entities and events. The wrapper classes are instantiated when the MNG recording's entity lists are accessed. Upon creation, these classes transparently use the annotations added to the Neo objects during data import. Consequently, import methods for different file formats must always store this information in the same attributes and fields. The main advantage is that the wrapper classes contain semantically relevant and specific information for, e.g., electrical stimulation in the context of MNG. Therefore, we add another abstraction layer from the Neo objects and facilitate scientific programming and programming cooperation in the context of MNG. On those occasions where Neo's structures are

beneficial, they can of course still be accessed since they exist in parallel to the MNG-specific data structure.

Apart from wrapper classes for Neo objects, we also provide classes and methods which in turn build upon our data structure. For example, we model action potential tracks (see Figure 3) as defined by Turnquist and Namer [7]. Data analysis code can use these classes and methods which facilitates data analysis even more, by reducing the required programming effort.

4.3. Feature Extraction

We provide a modular programming interface for feature extraction, mainly to enable machine learning and regression analysis. At the center of our implementation is the feature database to which new feature extractors can be registered. New feature extractors must implement the provided interface. Then, feature expressions for each action potential are calculated using the methods defined by the feature extractors. Feature expressions for individual action potentials can be retrieved from the feature database by providing instances of action potential wrappers or indices. As some features may be computationally expensive, the feature database can be written to a permanent storage as YAML files and numpy objects.

Currently, we have implemented some simple features such as the latency of an action potential measured from the regular electrical stimulation, but also more complex ones like a sliding window that counts the number of action potentials in distinct time intervals preceding an action potential. We allow different data types including floats and multidimensional arrays.

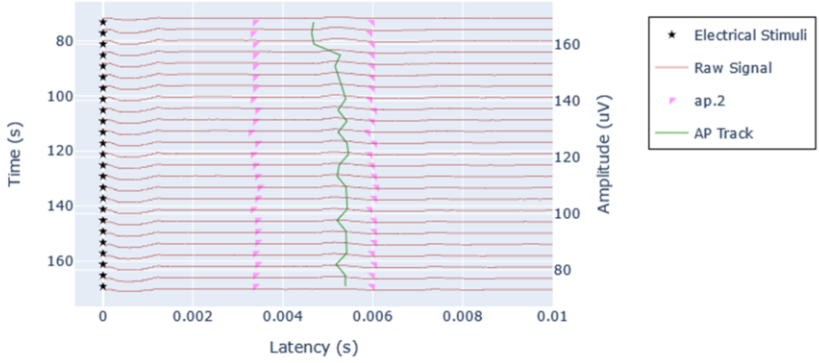

Figure 3. Example of a waterfall plot created from a Spike2 recording on animal data. The stars indicate electrical stimulation at constant time intervals along the left y-axis. In x-direction, the raw signal is plotted and the amplitude is shown on the right y-axis. The onset and offset of each action potential as registered by Spike2 is marked using triangles. Finally, the plot includes the AP track which has been identified by the correlation-based tracking algorithm as proposed by Turnquist et al. [7].

4.4. Visualizations

We provide methods for visualizations that are commonly needed in MNG data analysis. For example, waterfall plots aim at visualizing the latency between an electrical stimulus and an action potential that was triggered by this electrical stimulus. The electrical stimuli are plotted along the y-axis, with the post-stimulus signal being plotted along the

x-axis (see Figure 3). Such a plot can conveniently be generated from our internal data representation. Furthermore, usage of modern plotting libraries such as plotly allows interaction with the plots, including zooming and panning.

5. Lessons Learned / Discussion

We have found that using the Neo package for file import and internal representation of MNG data is very efficient. Neo's existing import methods are convenient from a software development perspective. As Neo offers import functionality for Spike2 files and other file formats which are used to record MNG data, usage of the Neo package avoids additional programming work. Still, as Neo cannot read Dapsys files we had to implement the import of Dapsys files using a workaround via csv-files. The ability to import Spike2 and Dapsys files into a unified representation allows us to apply algorithms for data analysis and visualization without the need to make adaptions to the source code. By adding import methods for further formats, this advantage might become even more apparent.

In the long term, we aim to import Dapsys dps-files directly. This can be realized either by contributing a Dapsys IO class to the Neo project or by providing our own solution within openMNGlab.

Furthermore, we found that providing more specific wrapper classes for MNG data analysis facilitates working with the Neo representations in an MNG context. These wrapper classes convey semantics as they correspond to actual MNG entities. The modular structure of openMNGlab allows for the addition of new wrapper classes if necessary. Thereby, development of algorithms and analysis pipelines in an MNG context becomes more efficient because relevant events and entities can be directly and conveniently accessed.

Finally, the design of our modular feature database enables for fast and easy addition of new features and feature extraction methods for machine learning in MNG. Some features have already been implemented [8], and new features will be added depending on future machine learning tasks.

In the next step, we will implement an improved fiber tracking algorithm for MNG data, based on track correlation as described by Turnquist et al. [7]. The algorithm will be extended with wavelet filters and semi-supervised learning approaches.

Concurrently, we are working on metadata storage and sharing in the context of MNG, using the odML metadata markup language [9]. In the long term, accessing metadata from the framework to retrieve information about a recording may allow for further automatization of MNG data analysis pipelines.

6. Conclusion

With openMNGlab, we propose a software framework which allows researchers to import microneurographic recordings of different formats so that they can perform data analysis and produce visualizations. One main advantage is that data analysts do not need to care about the specifics of the used file formats once an import method for this format exists within the framework. Furthermore, the unified data structure allows for easier coding collaboration and data exchange between multiple work groups. For data import and its internal representation, our software framework builds upon the Neo package for

electrophysiological data. As Neo does not support the Dapsys data acquisition software yet, we provide our own import methods. To facilitate data analysis in an MNG context, we introduce MNG semantics by extending the Neo data structure with wrappers for MNG-specific entities. On top of that, we provide methods for MNG data analysis, as well as methods for data visualization. With our feature database implementation, we also lay the foundation for machine learning in the context of MNG.

Declarations

Ethical vote: Medizinische Ethikkommission der RWTH Aachen, Vo-Nr. EK141-19, Chair: Prof. Dr. Schmalzing.
Conflict of Interest: The authors declare no conflict of interest.
Contributions of the authors: BN, EK conducted this project; FS, FK developed the system; EK, BN and RR supervised the development, FS drafted the paper and all authors substantially revised. All authors approved the final version and agreed to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved.

References

[1] Namer B, Handwerker HO. Translational nociceptor research as guide to human pain perceptions and pathophysiology. Exp Brain Res. 2009 Jun;196(1):163–72.
[2] Wilkinson MD, Dumontier M, Aalbersberg IjJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016 Mar 15;3(1):160018.
[3] Garcia S, Guarino D, Jaillet F, Jennings TR, Pröpper R, Rautenberg PL, et al. Neo: an object model for handling electrophysiology data in multiple formats. Front Neuroinform .2014;8.
[4] Pröpper R, Obermayer K. Spyke Viewer: a flexible and extensible platform for electrophysiological data analysis. Front Neuroinform 2013;7.
[5] Denker M, Yegenoglu A, Grün S (2018) Collaborative HPC-enabled workflows on the HBP Collaboratory using the Elephant framework. Neuroinformatics 2018, P19.
[6] DAPSYS Data Acquisition Processor System [Internet]. Available from: http://www.dapsys.net.
[7] Turnquist B, RichardWebster B, Namer B. Automated detection of latency tracks in microneurography recordings using track correlation. J Neurosci Methods. 2016 Mar 15;262:133–41.
[8] Troglio A, De Col R, Namer B, Kutafina E, Modeling of activity-induced changes in signal propagation speed of mechano-electrically stimulated neural fiber. Studies in Health technology and Informatics, accepted for publication.
[9] Zehl L, Jaillet F, Stoewer A, Grewe J, Sobolev A, Wachtler T, et al. Handling Metadata in a Neurophysiology Laboratory. Front Neuroinform. 2016;10.