slap2-utils: Tools for Processing SLAP2 Data

15 April 2024

Summary

Two-photon microscopy enables the measurements of neural activity in vivo, deep within the brain using fluorescent biosensors. However, capturing neural activity with voltage indicators, fast calcium or neurotransmitter biosensors, or slower indicators across 3D volumes requires sampling speeds that far exceed capabilities of conventional laser-scanning two-photon microscopes. The recently developed Scanned Line Angular Projection-2 (SLAP2) microscope has overcome these limitations by employing novel ultra-fast random-access scanning techniques to record neural activity at kilohertz sampling rates. Unlike traditional raster imaging, where pixels are recorded sequentially by spatial position, SLAP2's random-access acquisition optimizes the sampling order of regions of interest for speed. These recordings are stored in a complex file structure, which tracks the non-sequentially sampled coordinates. This novel data structure is incompatible with post-imaging analysis tools designed for traditional image formats, such as TIF or CZI files. Here, we present our Python library, `slap2-utils`, to interact with the custom data structure generated from SLAP2. `slap2-utils` allows users to extract neuronal activity directly from the SLAP2 binary files in a Python environment. Additionally, we provide extensions to the SLAP2 microscope control and image acquisition software, providing a framework for user-designed microscope functions during in vivo experiments.

Statement of need

To overcome the inherent speed limitations of laser scanning two-photon imaging, the Scanned Line Angular Projection (SLAP) microscope was developed using projection microscopy and deconvolution to sample neuronal activity in kilohertz rates (Kazemipour et al. 2019). The second generation of this technology, called SLAP2, builds upon this technology for random-access sampling of regions of interest (ROIs) throughout 3D volumes without deconvolution and is commercially available as a kit (Podgorski 2021). SLAP2 conducts random-access imaging of preassigned ROIs during high-speed acquisitions by transitioning between ROIs with an inertia-free scan engine. However, this results in a complex scan pattern as the microscope can sample ROI that are not adjacent to each other, complicating the task of indexing the raw recording. Currently, the custom tools needed to read data from the binary files generated during high-speed acquisitions are written using proprietary programming languages, making it cumbersome to design analysis workflows that leverage packages in the Python ecosystem. To overcome this limitation, we developed the Python library `slap2-utils`, intended for neuroscientists, engineers, and imaging specialists who use SLAP2 microscopes and prefer to analyze their data using Python-based tools. This includes both academic researchers and core facility staff managing SLAP2 datasets. Our package provides pure Python implementations of the functions needed to read raw SLAP2 microscope data, facilitating Python-based analysis pipelines. Additional functionalities include visualizing recorded data, plotting components of a SLAP2 file, and core processing functions.

SLAP2 Datafile Pipeline

`slap2-utils` provides an open-source version of Matlab functions (Podgorski 2021) designed to interface with binary files generated from high-speed recordings of neural activity from the SLAP2 two-photon microscope. This implementation uses object-orientated programming, creating Python Classes for parsing out the data for analytical pipelines (see Figure 1) (Podgorski 2021). This is needed because, unlike raster imaging that the SLAP2 microscope can also preform, which generates TIF formatted digital images, high-speed random-access imaging produces two files: a binary file (.dat) with an accompanying metadata (.meta) file that stores acquisition information using the HDF5 format. The binary file consists of a header, which stores core acquisition settings in unsigned 32-bit integers, followed by the digitized recording of the microscope's detectors in unsigned 16-bit integers. The `MetaData` file contains more detailed information about the acquisition, including key parameters, such as the `ParsePlan`, a collection of data describing the timing of axial focus and coordinates being sampled by the microscope. This information about the acquisition is needed to index the recording in the binary file. During random-access imaging, the microscope will sample ROIs at different acquisition rates in an order determined by a planning algorithm, which may not be in any particular spatial order. Given the complications in random-access imaging and the organization of the recording in the binary file, a simple linear readout of the DAT file's contents would be insufficient to extract the neural activity. The core function of our Python library handles this task with our `DataFile` object that parses raw data and meta-data stored in these files directly into a Python environment. This `DataFile` class, upon initialization, loads information from the file header of the binary file and creates a subclass, MetaData, which will load additional acquisition information using the h5py library (Collette 2013). The digitized signals of the acquisition are not loaded into memory. Rather, the Datafile class generates a memory map (memmap) of the 16-bit raw recording in the binary file using Numpy(Harris et al. 2020). This memmap is used to index specific data in the raw recording when needed, helping end users work with larger datasets collected at kilohertz acquisition rates.

To further aid with retrieving traces of neural activity from the random-access acquisitions, we implemented an additional Python class, `Trace`, which streamlines this operation. This class initializes with three variables: a `DataFile` object paired with the z-axis and channel ID indices. Before the `Trace` object can generate an activity trace, users must first call the `Trace.setPixelIdxs()` method. This method uses boolean arrays representing areas in the field of view. The `Trace` object uses these coordinates to generate indices in the `DataFile`'s memmap during trace extraction. The boolean arrays can be generated from SLAP2 ROI objects stored in the `MetaData` class, allowing the `Trace` object to generate activity profiles from individual ROIs. Once these inputs are finalized, the `Trace.process()` method will return an activity trace to the user. Additionally, each ROI is composed of smaller units, superpixels, and information for each superpixel is stored in a subclass `TracePixel` accessible through the `Trace.TracePixels` attribute in the parent class. This allows for higher resolution interrogation of the spatial domains of neuronal activity within a ROI; it should be noted that while `Trace.TracePixels` is a list; it is not ordered spatially until the `Trace.orderadjust()` method is called.

SLAP2 Utility Functions and Visualizations

Several commonly used functions are included for tasks that would be routinely called upon. These utility functions simplify generating the traces from ROIs, cleaning up traces from volumetric recordings, and extracting ROI information in both 2D and 3D. Additional utility functions interact with other file types generated by the microscope, such as extracting stimulus times from stimulus logs stored in H5 files and functions for averaging reference stacks stored as TIF files. Useful plotting functions are also packaged in the library, such as a function that plots ROIs over a reference image. Furthermore, the package comes with a Python-based data viewer that allows for the inspection of activity and ROIs from a given DAT file with a GUI. The data viewer's features include 3D inspection of ROIs over a reference volume and interactive plots of activity traces.

Availability

All the code and scripts mentioned above are publicly available on the GitHub repository ([https://github.com/Peter-Hogg/SLAP2_Utils/tree/main]). The Python library can be installed using pip.

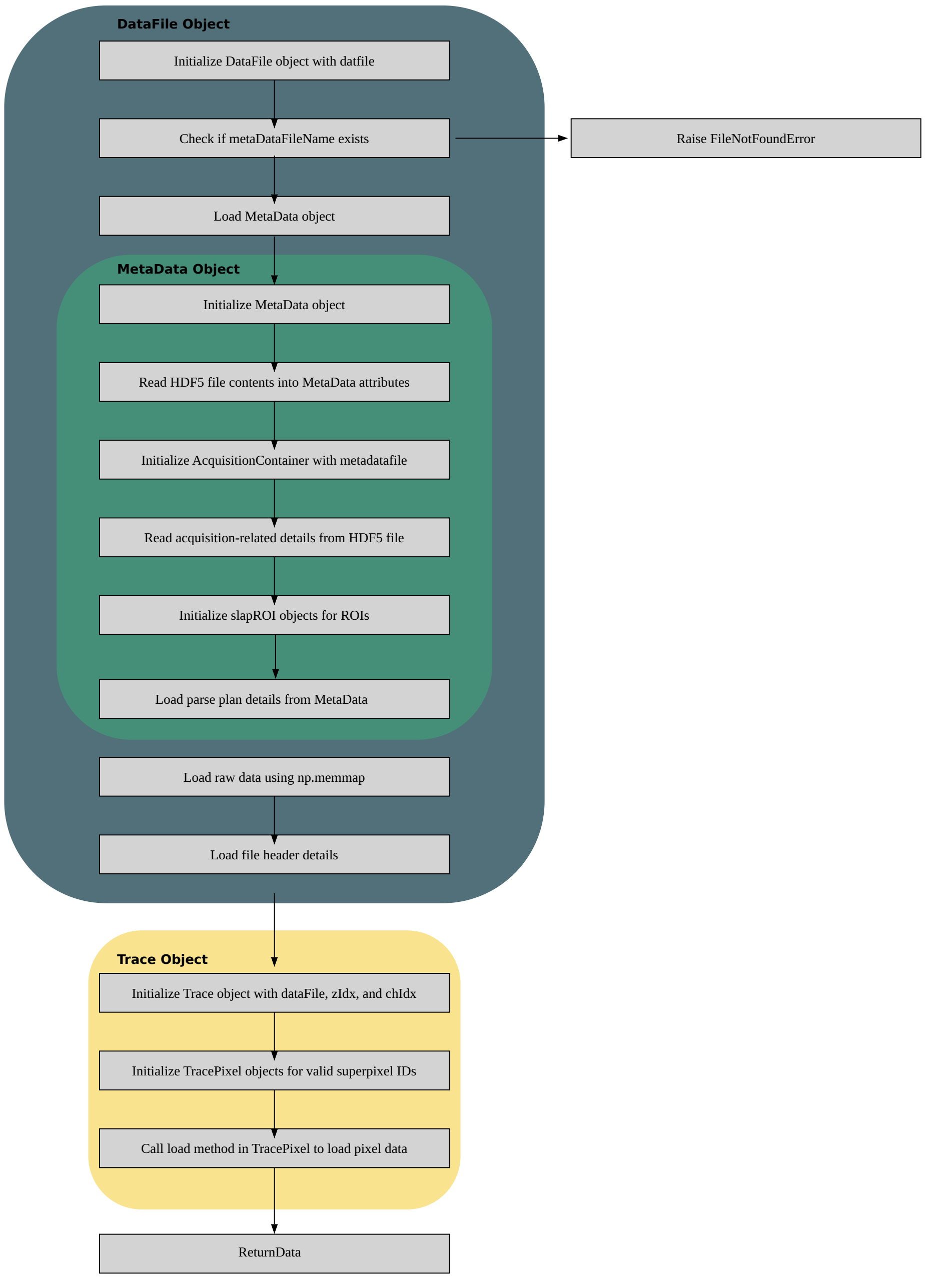
DataFile Object
Initialize DataFile object with datfile
Check if metaDataFileName exists
Raise FileNotFoundError
Load MetaData object
MetaData Object
Initialize MetaData object
Read HDF5 file contents into MetaData attributes
Initialize AcquisitionContainer with metadatafile
Read acquisition-related details from HDF5 file
Initialize slapROI objects for ROIs
Load parse plan details from MetaData
Load raw data using np.memmap
Load file header details
Trace Object
Initialize Trace object with dataFile, zIdx, and chIdx
Initialize TracePixel objects for valid superpixel IDs
Call load method in TracePixel to load pixel data
ReturnData

Figure 1: Figure 1: SLAP2 Pipeline. The DataFile class is initialized with the path to a .dat file. The MetaData subclass will be initialized if the .meta file is found. A DataFile object is then used to initialize the Trace Class, which has methods for easy data extraction from the binary file.

Acknowledgements

The development of this Python library was supported by funds from the Canadian Institutes of Health Research (CIHR) Foundation Award (FDN-148468) and NSF Award 2019976, “AccelNet: International network for brain-inspired computation”.

References

Collette, Andrew. 2013. *Python and HDF5*. O’Reilly.

Harris, Charles R., K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. “Array Programming with NumPy.” *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Kazemipour, Abbas, Ondrej Novak, Daniel Flickinger, Jonathan S. Marvin, Ahmed S. Abdelfattah, Jonathan King, Philip M. Borden, et al. 2019. “Kilohertz Frame-Rate Two-Photon Tomography.” *Nature Methods* 16 (8): 778–86. https://doi.org/10.1038/s41592-019-0493-9.

Podgorski, Kaspar. 2021. “SLAP2 – Two Photon Microscope Kit.” *Product Page*. MBF Bioscience. https://www.mbfbioscience.com/products/slap2.