SLAP2 Utils: Tools for Processing SLAP2 Data

15 April 2024

# Summary

Two-photon microscopy allows for measuring neural activity *in vivo*, deep within the brain, using fluorescent biosensors. However, neural activity frequently occurs at orders of magnitude faster than what a typical laser-scanning two-photon microscope can record. Several novel two-photon scanning techniques have been developed to overcome this limitation in acquisition rate. However, these sampling methods store neural data in complicated file structures rather than typical image formats. Here, we present our Python library, SLAP2 Utils, to interact with the custom data structure generated from a novelly designed two-photon microscope, SLAP2, capable of recording neural activity at kilohertz sampling rates. 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, which provides 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 [@Kazemipour2019]. 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. This novel microscope design is commercially available as a kit [@mbfSLAP2]. However, 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. Our package provides pure Python implementations of the functions needed to read raw microscope data, facilitating Python-based analysis pipelines. Additional functionalities include visualizing recorded data, core processing functions, and extensions of the primary SLAP2 software used during experiments.

# SLAP2 Datafile Pipeline

SLAP2 Utils provides an open-source version of Matlab functions [@mbfSLAP2] 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 (OOP), creating Python Classes for parsing out the data for analytical pipelines (see Figure 1 ) [@mbfSLAP2]. This is needed because, unlike raster imaging with the SLAP2 microscope, 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 axial and temporal focus of the microscope. This information about the acquisition is needed to index the recording in the binary file correctly. 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 readout of the DAT file’s contents would be insufficient to extract the neural activity correctly. 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 [@h5py]. The digitized signals of the acquisition are not loaded into memory. Rather, the Datafile class generates a memory map (memap) of the 16-bit raw recording in the binary file using Numpyr[@numpy]. This memap is used to index specific data in the raw recording when needed, helping end users work with larger datasets collected at kilohertz acquisition rates.

As with other forms of random access imaging, SLAP2 will only image preassigned regions of interest (ROI). The scan engine of the microscope can jump between these ROIs without any inertia, resulting in high-speed imaging. However, this results in a complicated scan pattern facilitating this acquisition rate, making indexing the raw recording complicated. Furthermore, retrieving a trace becomes more intricate when dealing with multichannel and volumetric recordings. To aid in this retrieval process, we created an additional Python class, Trace, which streamlines this problem. 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 (FOV). The Trace object uses these coordinates to generate indices in the DataFile’s memap 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 in a given ROI; it should be noted that while Trace.TracePixels is a list; it is not ordered spatially until the Trace.orderadjust() method is called.

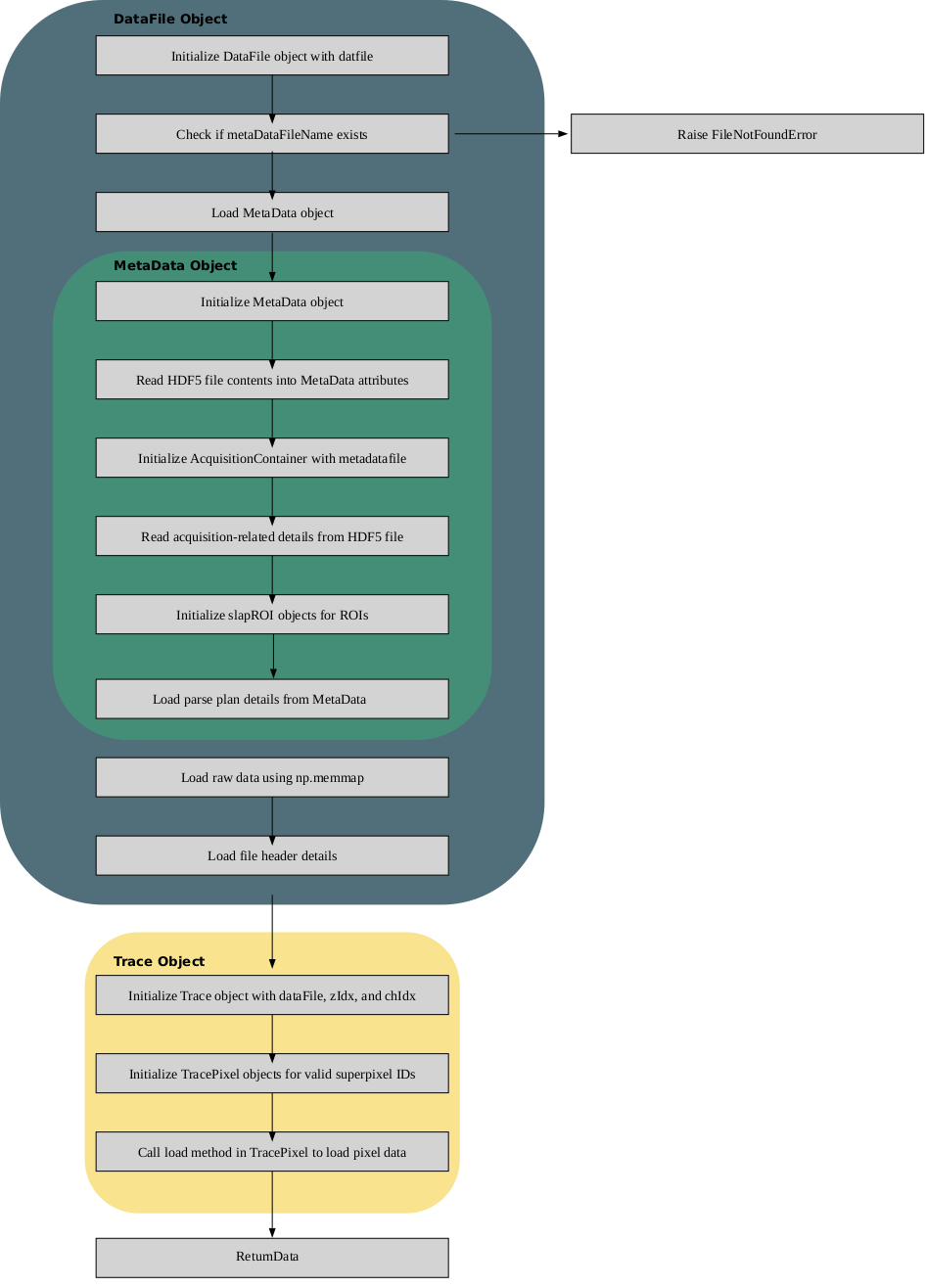


Figure 1: SLAP2 Pipeline. The DataFile class is initialized with the path to a .dat file. The MetaData subclass will be initialized if the .meta file is found. A DataFile object is then used to initialize the Trace Class, which has methods for easy data extraction from the binary file.

# SLAP2 Uility UI that Accompanies SLAP2 Software

In addition to the Python library, extensions to the core SLAP2 software are packaged in the software repository. This code combines MATLAB and Python utilities controlled by a custom user interface (UI). The script directly interfaces with objects in the SLAP2 software, primarily written in MATLAB. The UI includes valuable functions such as generating ROIs automatically from reference images, creating a reference stack, and updating ROI positions due to sample drift [@cupy, @scikit]. Interfacing directly with the SLAP2 software allows these scripts to automate the workflow during an experiment, minimizing human error that can arise during *in vivo* imaging sessions. The calculation of the shift adjustment used to update ROI positions uses phase cross correlation between the live view of the sample and a reference stack. Our implementation of this calculation is a GPU-accelerated rewrite of the algorithm used in the scikit-image library using cupy [@phasecross, @scikit][@cupy]. These GPU-accelerated speed-ups allow the function to perform in real-time.

# 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.

# Acknowledgements

The development of this Python library was supported by funds from the Canadian Institutes of Health Research (CIHR) Foundation Award (FDN-148468).

# References