OPETH Documentation

Andras Szell

Mar 22, 2020

OPETH documentation

CHAPTER 1

Getting started

Detailed user guide: *Using OPETH*

Online Peri-Event Time Histogram for Open Ephys.

OPETH visualizes Peri-Event Time Histograms (PETH) of spikes detected in raw Open Ephys data, broadcasted via ZeroMQ. PETH is aligned to triggers from Open Ephys.

1.1 Quickstart

- OPETH requires ZMQInterface plugin. It is part of Open Ephys from version 0.4.6 up.
- Set up Open Ephys with ZMQInterface plugin. The ZMQ plugin is recommended to be put after bandpass filter and/or common average reference filter in the Open Ephys signal chain, while spike detector filter is not required.
- Start with the `opeth` command when using the pip package or start with `python opeth/gui.py` when running from sources (see below).

1.2 Installation

Simplest way is to install the opeth package for Python 2.7 or Python <=3.7 with pip:

```
pip install opeth
```

Then start with:

```
opeth
```

(Python 3.8 support is partially broken until the release of pyqtgraph 0.11.)

1.2.1 Dependencies

Required non-default packages: `pyzmq`, `pyqtgraph` plus one of the qt versions for pyqtgraph, preferably `PyQt5`, and also their dependencies (e.g. `numpy`).

1.3 Running from sources

After cloning the git repository or extracting a source zip file, multiple methods could work.

1.3.1 Setting up python environment with conda

Conda builds are not available yet.

Using conda/miniconda, create an `opeth` environment issuing the following command in the root dir of opeth:

```
conda env create --file environment.yml
```

which will install all necessary prerequisites for Python 3.7.

Activate the new environment with the command

```
conda activate opeth
```

and once activated, you may start OPETH with

```
python opeth/gui.py
```

Using python 3.8 is not recommended (Feb 2020) as some bugs are to be addressed (most probably residing in pyqtgraph), but it is possible with the conda-forge version of pyqtgraph (default environment name will be `opeth_python38`):

```
conda env create --file env38.yml
```

1.3.2 Setting up python environment with pip

Python 3.7 dependencies can be installed with the command

```
pip install -r requirements.txt
```

CHAPTER 2

Using OPETH

This online documentation is associated with pre-print 'OPETH: Open Source Solution for Real-time Peri-event Time Histogram Based on Open Ephys' by András Széll, Sergio Martínez-Bellver, Panna Hegedüs and Balázs Hangya. DOI: https://doi.org/10.1101/783688.

OPETH was tested under Windows only. Feedback is welcomed for other platforms.

2.1 Connecting OPETH to Open Ephys

OPETH relies on data and events recorded and timestamped by Open Ephys. These data are broadcasted by Open Ephys through the ZMQ Interface plugin.

- Signal conditioning/filtering should be performed by Open Ephys, therefore the ZMQ plugin should be placed after appropriate filters. E.g. Band-pass filtering between 600-6000 Hz enables threshold-based action potential detection.
- Spikes are detected by OPETH, so a spike filter plugin in OE is unnecessary for OPETH.

Sample OE signal chain for ZMQ broadcasting:

OPETH connects to the local data and event ports of the ZMQ plugin. It is possible to run multiple OPETH instances with different settings simultaneously. (Connecting over the network could also be possible but not implemented in the GUI currently.)

It is recommended to have Open Ephys set up and started before running OPETH.

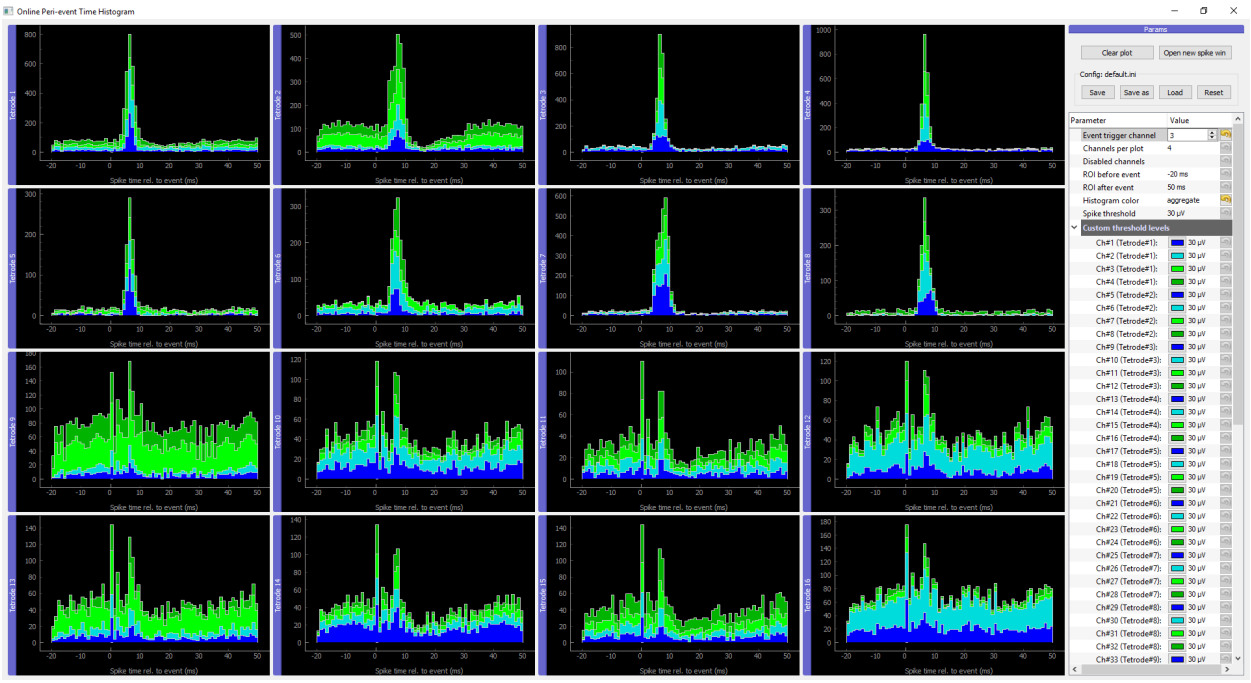
2.2 OPETH graphical interface

The GUI consists of three different type of windows currently:

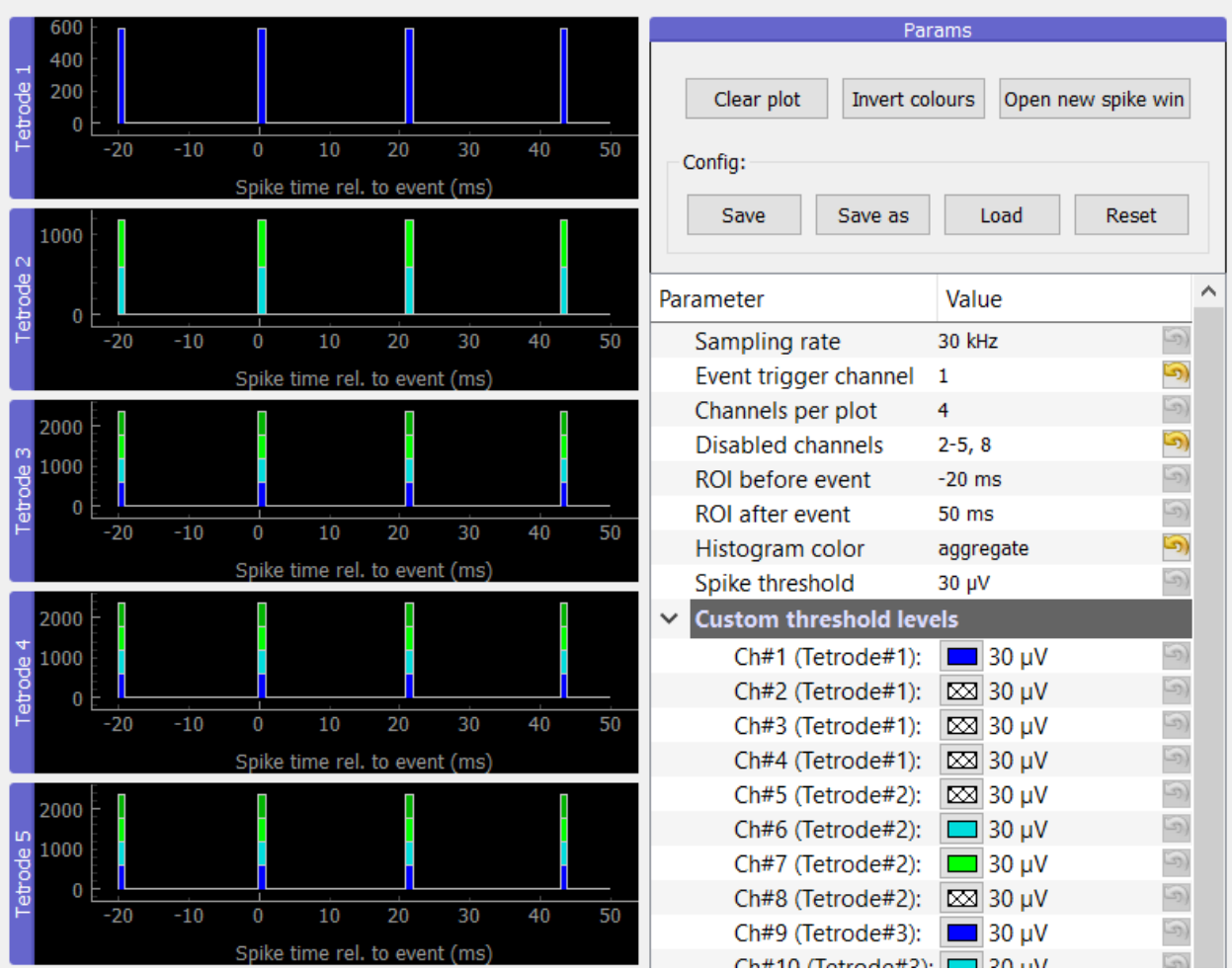
- **Main histogram window:** The main window displays histograms, parameters and buttons for handling the configuration and the different plots.
- **Raw analog data window:** real time data view displays data arriving from Open Ephys, used for debugging (e.g. to determine whether spikes are absent due to triggering issues or because of data content).
- **Spike analysis windows:** Opened from main window. Displays detected spikes for a single channel for visual spike/artefact observations.

2.2.1 Online Peri-event Time Histogram (main window)

A view of an actual recording session performed with OPETH with PETHs on the left side and UI buttons/params on the right:



Synthetic data and parameters close-up with tetrode #1 displaying channel 1, tetrode #2 displaying ch 6 and 7 only:



The histograms are pyqtgraph elements so by dragging their title they can be rearranged or by double clicking even detached from the main window.

Parameter description:

- **Sampling rate:** should match Open Ephys Rhythm FPGA settings. (Will be read-only after fixing a bug preventing accessing this info.)
- **Event trigger channel:** the OE trigger channel. Can be a TTL pulse source e.g. PulsePal or BPOD event triggers. (Technically non-TTL signals can also be reported by OE as timestamped event.)
- **Channels per plot:** channels are collected in groups of four by default as for classical tetrode recordings, but can be set from 1 to 8 (for single electrodes, stereotrodes etc.) Changing it automatically changes the number of histograms displayed.
- **Disabled channels:** disable channels that are not to be spike-filtered, e.g. noisy or inactive channels. See screenshot for accepted formats. OPETH automatically disables and hides the extra 3 gyroscope channels of a 32 or 64 channel setup if 35/70 channels are detected (if not desired, change behaviour with `opeth.gui.HIDE_AUX_CHANNELS`).
- **ROI before/after event:** region of interest around trigger. Only this part of the analog data is spike filtered.
- **Histogram color:**
  - *flat* histogram merges spikes from all channels of the given tetrode
  - *aggregate* (default) view displays channels in distinct colors but in the same histogram bin,

  - *channels* display a line plot for each channel separately.

- **Spike threshold:** threshold level can be applied globally or per channel. By default negative spikes are detected and the threshold levels in GUI are considered absolute value.

If multiple triggers fall within the ROI, the same spikes may be detected for the triggers in the overlapping part.

Plot handling buttons:

- **Clear plot:** clears histogram windows.
- **Invert colours:** switch between black and white background for histogram plots. (Experimental.)
- **Open new spike win:** initiate new spike analysis window.

Parameters can be saved and loaded into ini files, last used file is remembered and reloaded upon startup.

Configuration handling buttons:

- **Save/Save as:** store current configuration into file.
- **Load:** open a different configuration when changing to a different experimental project.
- **Reset:** restore defaults.

2.2.2 Raw data window

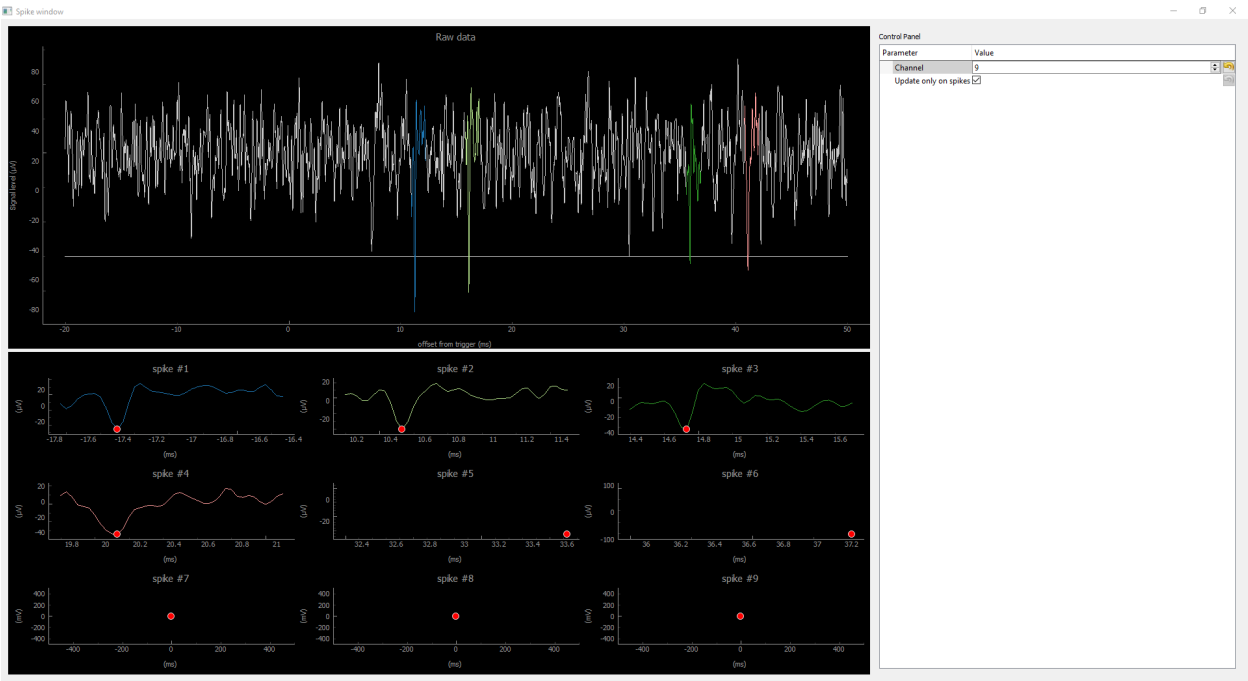
Displays data received directly from Open Ephys, allowing low-level visualization of input for debugging.

- Channels are auto-scaled and do not provide information on actual voltage levels yet.
- The top half of the window is a rolling display that plots all channels simultaneously. To zoom in, hold right mouse button and move the mouse.
- The plot is updated at a low frame rate and the data displayed are downsampled to 1000 Hz.
- The bottom part has a stimulus counter and presents analog data aligned to the trigger stimuli.
- Window boundaries with respect to the trigger are set by the *ROI before event* and *ROI after event* parameters.
- Can be closed if not required for debugging.

When the stimulus counter is not incrementing, no triggers are received and thus no spike detection will be performed (-> histograms not updated).

2.2.3 Spike analysis window

Spike windows are opened pressing the *Open new spike win* button. By opening multiple spike windows it is possible to compare channels side by side, but too many open windows will result in poor performance.



The window consists of two plot parts: the top part shows the raw input around the event, and the bottom part displays the detected spikes within the ROI of the event. The spike position is marked with a red dot in the bottom plots. These spikes are overlaid in the top plot in color and make it easy to differentiate valid spikes from false positives. It is possible to zoom in/out holding down the right mouse button in the spike windows.

Each spike window displays data for a single channel. The *channel number* can be adjusted real-time.

If the *Update only on spike* option is selected, spike windows are updated when new spikes are detected within the ROI of the trigger; otherwise, spike windows are updated 5 times per second even when no spikes are present.

2.3 Spike detection

OPETH performs simple spike detection with threshold crossing detection. No spike sorting or artefact removal is performed.

CHAPTER 3

Architecture overview

This online documentation is associated with pre-print 'OPETH: Open Source Solution for Real-time Peri-event Time Histogram Based on Open Ephys' by András Széll, Sergio Martínez-Bellver, Panna Hegedüs and Balázs Hangya. DOI: https://doi.org/10.1101/783688.

3.1 Data acquisition: Open Ephys ZMQ plugin

Data acquisition and signal conditioning is performed by Open Ephys. OPETH implements spike thresholding itself, therefore a Spike Detector plug-in should not be included before the ZMQ interface. OPETH receives data from OE's ZeroMQ interface plugin. The plugin broadcasts recorded data and events that can be subscribed to by external applications. Timestamps accompanying these data and event packets are the sample index, which will get converted to actual timestamp based on current sampling rate.

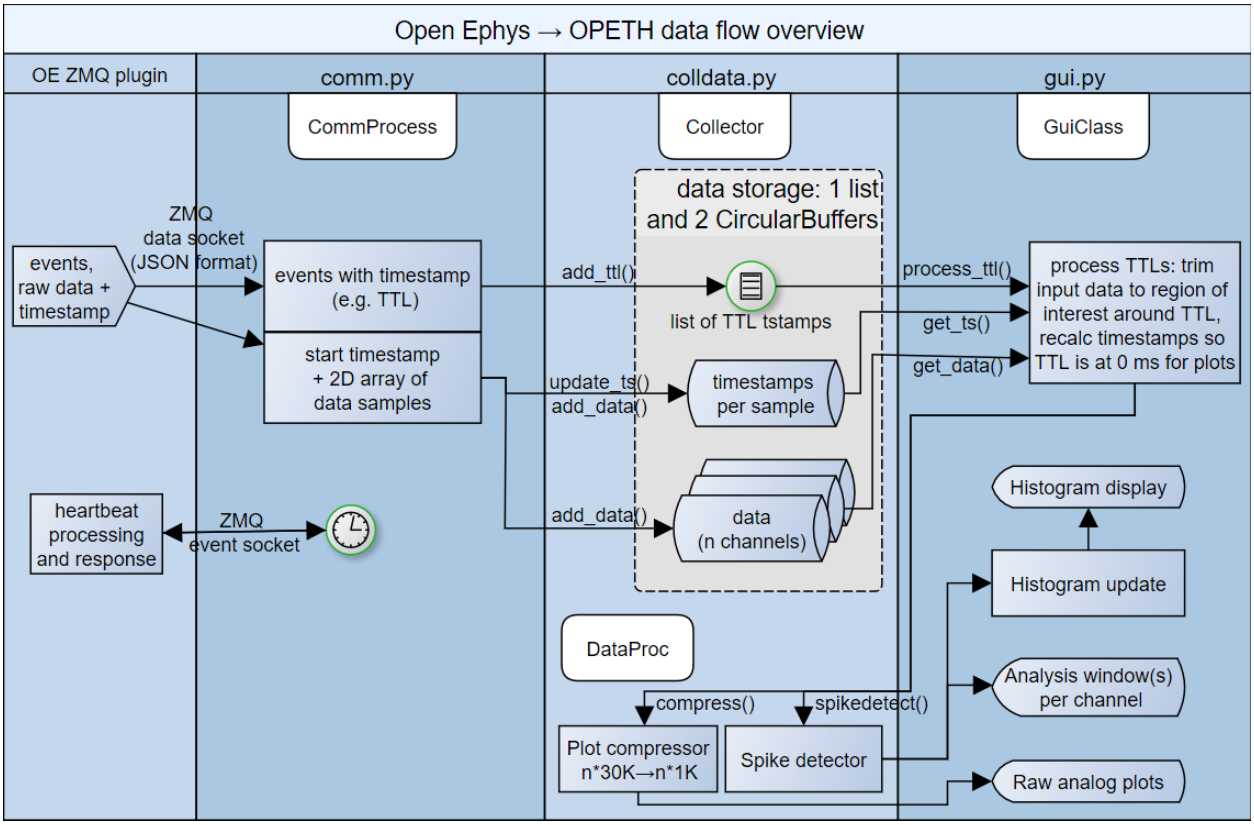
The ZMQ Interface plugin opens a ZMQ publisher socket to allow one or more ZMQ clients to subscribe (connect) locally or over the network. Though the system is typically used with a single client connected locally, it is possible to use multiple OPETH clients on one or more PCs analyzing the same Open Ephys data source simultaneously with different settings.

The ZMQ plugin uses JSON format data packets for the digitized data and event metadata (e.g. timestamp, event channel, number of data channels and sample count). Another socket for event messages and responses is used for heartbeat messages to notify the plugin about the connected clients. (Possibly for listing them on ZMQ plugin interface.)

3.2 Open Ephys - OPETH interface

Input data arriving from Open Ephys is handled by `opeth.comm`, which takes care of parsing the JSON structures containing the measurement samples and trigger events. Depending on the type of the parsed input data, trigger events are stored in OpenEphysEvent objects (defined in openephys.py) and sample data are stored directly in a 2D circular (or rolling) buffer implemented in `opeth.circbuff`; the data flow is managed by the `opeth.colldata.Collector` class in `opeth.colldata.py`. (The `opeth.openephys` and some of the `opeth.comm` interface routines are based on the python samples created by Francesco Battaglia.)

The following figure summarizes the main data flow of OPETH:



3.3 OPETH GUI overview

Display windows currently available:

- Main histogram / parameter setup
- Raw analog data debug
- Spike analysis window (opened by the **Open new spike win** button)

At startup, two windows are opened by default: the main histogram window displaying the online PETH results, and a raw analog data plot for debugging. The spike analysis window is a third view that makes it possible to visually differentiate between spikes and artefacts.

3.3.1 Histogram window

The main GUI window is implemented in `opeth.gui`, which schedules data reading, spike discrimination, performs histogram calculation and enables the adjustment of parameter setup.

3.3.2 Raw analog data window

In the interest of CPU time, the plot is updated at a low frame rate and the data displayed are downsampled to 1000 Hz for this view. Instead of relying on pyqtgraph's downsampling capabilities a different one was used. The raw

debug display was implemented in `opeth.gui.GuiClass.update()`, with min-maxed data downsampling in `opeth.colldata.DataProc.compress()`.

3.3.3 Spike analysis window

Spike windows opened from the main histogram window and are handled by `opeth.spike_gui`. Multiple Spike windows can be displayed simultaneously, but this is CPU intensive.

3.3.4 Logging

Log files are created automatically.

3.3.5 Configuration

Last used configuration file name is stored in the file `lastini.conf`. Configurations are stored in the ini file format, and parsed by the configparser module.

CHAPTER 4

Contributors

Developed by Andras Szell (szell.andris@gmail.com) and other Hangyalab members (http://hangyalab.koki.hu/).

Open Ephys ZMQ plugin connection is based on sample python scripts created by Francesco Battaglia.

CHAPTER 5

License

GNU General Public License v3.0 or later.

See LICENSE for the full text.

CHAPTER 6

For developers

- modindex