SHYBRID Documentation

Release 0.4.0

Jasper Wouters

Mar 31, 2020

Contents

1 Installation Procedure 3

2 Getting Started with SHYBRID 5

3 Use your own data 11

4 Sorting and comparing sorting results 15

5 Extra features 17

SHYBRID is a graphical user interface for the assisted transformation of extracellular brain recordings into so-called hybrid ground truth recordings which are characterized by the availability of partial ground truth spike times. As such, SHYBRID empowers spike sorting users to gain more insight into spike sorting performance and answer other spike sorting related questions for their specific recording setting.

This documentation will focus on installing and using the provided software. If you are interested in a more formal description of the hybrid ground truth model, we refer you to the preprint of our scientific SHYBRID paper. Feel free to contact me if you have any questions/doubts related to this documentation or usage of the software. In case you experience issues using SHYBRID or have ideas for new features don't hesitate to open an issue on GitHub or contact me by e-mail.

CHAPTER 1

Installation Procedure

Installing SHYBRID is as simple as installing a python package. On this page we will guide you through the different installation steps.

1. Install Miniconda or another Python distribution (Python 3.x version) for your operating system. Miniconda installation instructions can be found on the official Conda website.

The remaining SHYBRID installation commands have to executed in the (Anaconda) terminal.

2. Optional, yet recommended, create a virtual environment for SHYBRID:

```
>> conda create -n shybrid_env python=3
```

   and activate the newly created environment:

```
>> conda activate shybrid_env
```

3. Install SHYBRID using pip:

```
>> pip install shybrid
```

4. To launch SHYBRID execute the following command:

```
>> shybrid
```

If you decided to install SHYBRID in a virtual environment, keep in mind that the program is only accessible from within the shybrid conda environment. This implies that you have to reactivate this environment, e.g., after a PC reboot.

CHAPTER 2

Getting Started with SHYBRID

In this Section, we will walk you through the different steps of the data hybridization process.

The main idea behind the generation of hybrid data is to move a single unit, that has been identified during an initial spike sorting, to a different set of recording channels within that same recording. All single units that have been relocated make up the hybrid ground truth. Moving a unit is key to the hybridization process. Because of the relocation of a unit, potential false negatives in the initial spike sorting are prevented from being falsy characterized as false positives, compared to the case where one uses the initial spike sorting as ground truth. Note that next to the relocated units, other units might exist for which no ground truth is available. As such, hybrid recordings are only partial ground truth recordings. Also, note that the hybridization process requires multi-channel extracellular recordings. Preferably from probe-like devices, where the distance between recording channels is regular, such that the spatial structure of spike templates is also meaningful after relocation of a unit.

If you want to experience this walkthrough in a hands-on fashion, we recommend you to download the SHYBRID example data. After downloading, extract the content of the zip file on your machine. Make sure to adjust that paths in the parameter file (*.yml) to reflect the correct location of the extracted files.

2.1 Input data

Turning a binary **extracellular recording** into a hybrid ground-truth recording requires some additional information besides the recording itself.

Note: Every binary recording that has to be processed by the SHYBRID has to be kept in its own subfolder, the reason for this is that SHYBRID keeps track of the hybridization process (by outputting the ground truth labels and a binary file that supports cross-session undo functionality). Keeping multiple recordings in the same subfolder, will overwrite those files that keep track of the ground truth and will render the generated hybrid data useless.

Initial spike sorting of the provided extracellular recording, that has been manually curated, is an important ingredient for creating a hybrid ground truth. During the manual curation, a subset of automatically identified spike trains are selected/annotated because they are believed to contain only spikes from a single neuron/unit. The initial spike sorting

and manual curation can be obtained from your spike sorting workflow of choice, e.g., through a combination of SpyKING CIRCUS (for sorting) and phy (for curation).

The **probe file** of the recording device is also required by SHYBRID. This so-called probe file is required because it contains the geometric arrangement of the recording channels. This geometric knowledge is required during the relocation of a unit.

To ease the process of loading those different files into SHYBRID and providing a way for inputting recording related meta information in SHYBRID, a **parameter file** is required.

Natively, shybrid supports the following file formats:

extracellular recording	.bin, .raw, .dat	only binary recordings are supported
initial spike sorting	.csv, phy-format	both CSV and phy-format initial spike sorting is supported
probe file	.prb	phy probe files are supported
parameter file	.yml	parameter file is a structured YAML file

Because many more file formats are in use, we integrated SHYBRID with SpikeInterface. Due to this integration, many of the common formats can be converted to SHYBRID compatible formats very efficiently. More details related to the input data and code examples for converting your own data can be found in *Use your own data*.

To actually load the input data into SHYBRID, you'll have to first open the graphical user interface (GUI) by typing the following command in your (anaconda) terminal:

```
>> shybrid
```

Once the GUI is opened, click the top-left button *select data*, after which a file browser is shown. Browse to the folder where the binary recording for hybridization is located and double click the recording to open. When the file browser closes without error, your data has been succesfully loaded. After loading the data, the *select cluster* dropdown is populated with the provided initial spike sorting clusters.

2.2 Visualizing the template

The first step of the hybridization process is the visualization of the spike template for an initial sorting cluster of choice. To visualize the spike template, first select a cluster from the *select cluster* dropdown (Fig. 1 (1)), then fill out the desired template temporal *window* length (Fig. 1 (2)) and push the *draw template* button (Fig. 1 (3)). The spike template is calculated from the provided data in combination with the initial spike sorting. After the template has been calculated, it is shown in the plotting area on the right side of the GUI (Fig. 1 (4)).

The spike template visualization is organized according to the geometrical probe information that is given. Every connected line segment that is plotted represents a recording channel/electrode and it is plotted according to the same spatial organization as the channels of the recording device. Channels in blue contribute to the template, whereas channels in gray are channels containing only noise. The plot can be manipulated using the plot control buttons (Fig. 1 (5)) on the top left of the plotting canvas.

A first visual check enables the user to verify whether or not the spike template is sensible. Noise-only channels are explicitly set to zero, but depending on the signal-to-noise ratio of the spike template, the zero-force fraction (Fig. 1 (6)) might have to be adjusted. Increasing the zero-force fraction will result in more channels that are explicitly forced to zero. Besides checking whether or not the spike template itself is sensible, it is also important to verify whether or not the template window size is chosen sufficiently large. Sufficiently large means that the spike template reaches zero on both sides of the temporal window for all channels. The gray lines are the zero lines for each channel, and are to be used as helper lines for checking whether or not the window size is adequate.

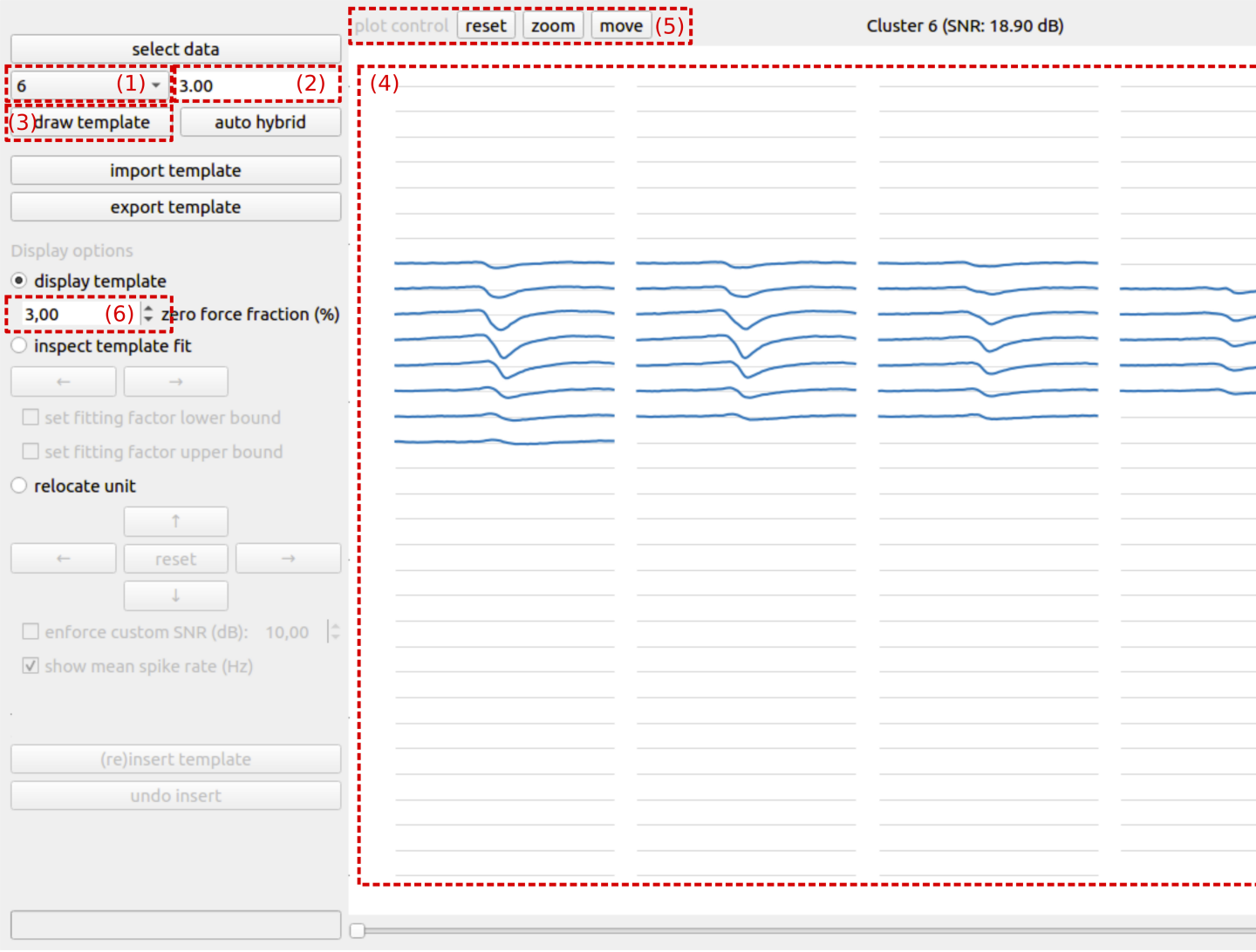

Fig. 1: Fig. 1: Visualizing the template

2.3 Inspecting template fit

Once that we have verified that the template is suitable for hybridization, we can check how well the template fits the underlying data from which it was estimated, by switching to the *inspect template fit* display option. For every spike time, an optimal template fit, i.e., template scaling factor, is calculated. Due to noise, overlapping spikes or errors in the initial spike sorting, the template fit for some spike times can become unrealistic. The goal of this step is to eliminate such unrealistic template fits from the hybridization process.

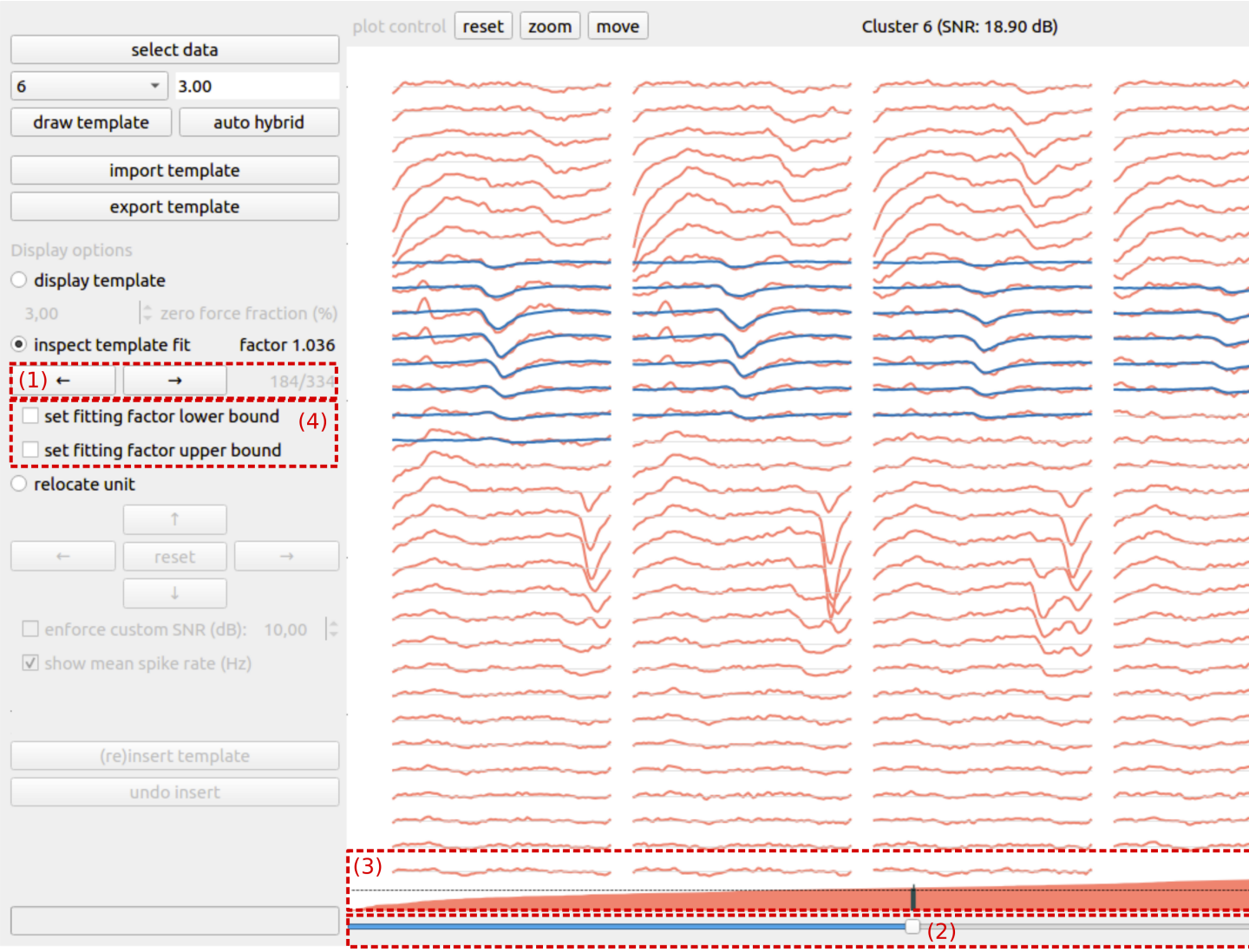

Fig. 2: Fig. 2: Inspecting template fit

In the inspect template fit mode, one can scroll through all signal chunks that were used during the estimation of the template. You can either scroll through by clicking the arrow buttons (Fig. 2 (1)) or by using the bottom slider (Fig. 2 (2)). The different chunks are ordered based on their fitting factor, and the fitting factor is graphically represented (Fig. 2 (3)) above the bottom slider. We recommend scrolling through the signal chunks, starting from the lowest fitting factor, until an isolated signal chunk and realistic fitting factor is identified. This fitting factor can be set as a lower bound by checking the *set lower bound* checkbox (Fig. 2 (4)). For identifying an upper bound, one can repeat this procedure, but scrolling down from the largest fitting factor until an isolated, reasonable upper bound is identified.

2.4 Relocating a unit

After assessing the template and its fitting factors, the final step in the hybridization process cosists of relocating the unit to a different set of recording channels. As explained earlier, this key step in the hybridization process prevents false negatives in the initial spike sorting from being falsy interpreted as false positives in the case where the initial sorting would be used for ground truth. The new location of the unit can be obtained by moving its template over the probe with the arrow buttons (Fig. 3 (1)). To guide the user in the selection of a new location for the unit, the channels are color coded according to the channel's mean spike rate (Fig. 3 (2)). Moving a unit to a zone on the probe with a low spike rate will typically result in easier to sort recordings, whereas relocating a unit to a more active zone will typically result in more difficult to sort recording, e.g., due to spike overlap. When all hybridization parameters have been set (i.e, window size, zero-force fraction, fitting bounds, new location), a hybrid unit can be created by pushing the *(re)insert template* button (Fig. 3 (3)). Besides changing the underlying recording by relocating the unit's spikes to the new location, a CSV-file with ground truth spike times is automatically generated in the directory that contains the recording (*hybrid_GT.csv*).

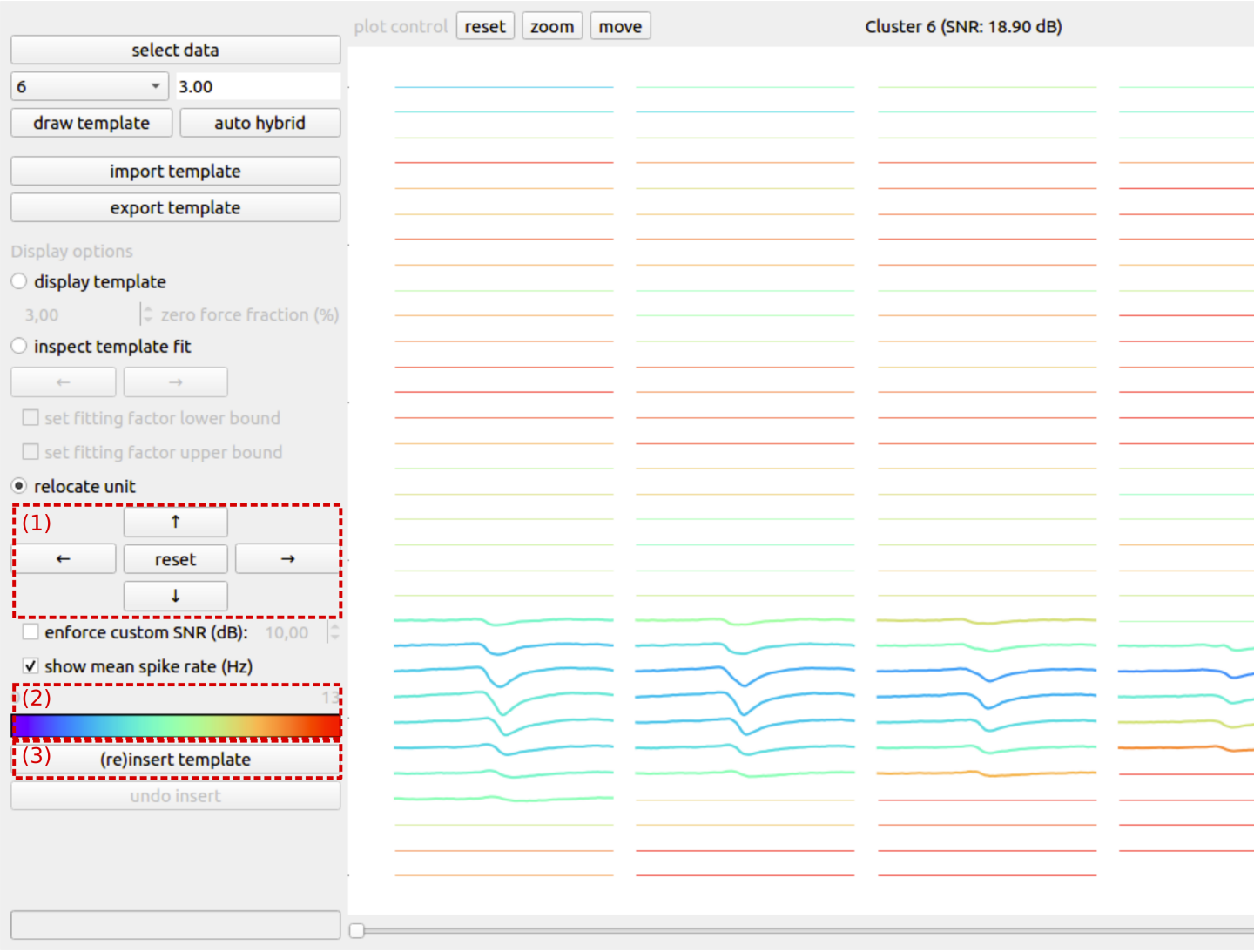

Fig. 3: Fig. 3: Relocating a unit

2.5 Auto hybridization

The above procedure can be repeated for all clusters in the initial sorting. However, when many clusters are present, this procedure can become time consuming. Therefore, we provide an auto hybrid function that automatically sets conservative bounds on the fitting factors and moves the unit to a random location on probe. The window size and zero-force fraction have to be determined by the user prior to the automatic hybridization. When the *auto hybrid* button is pushed, a pop-up is shown (Fig. 4) where the user can select a set of clusters for automatic hybridization. By default, all clusters are selected.

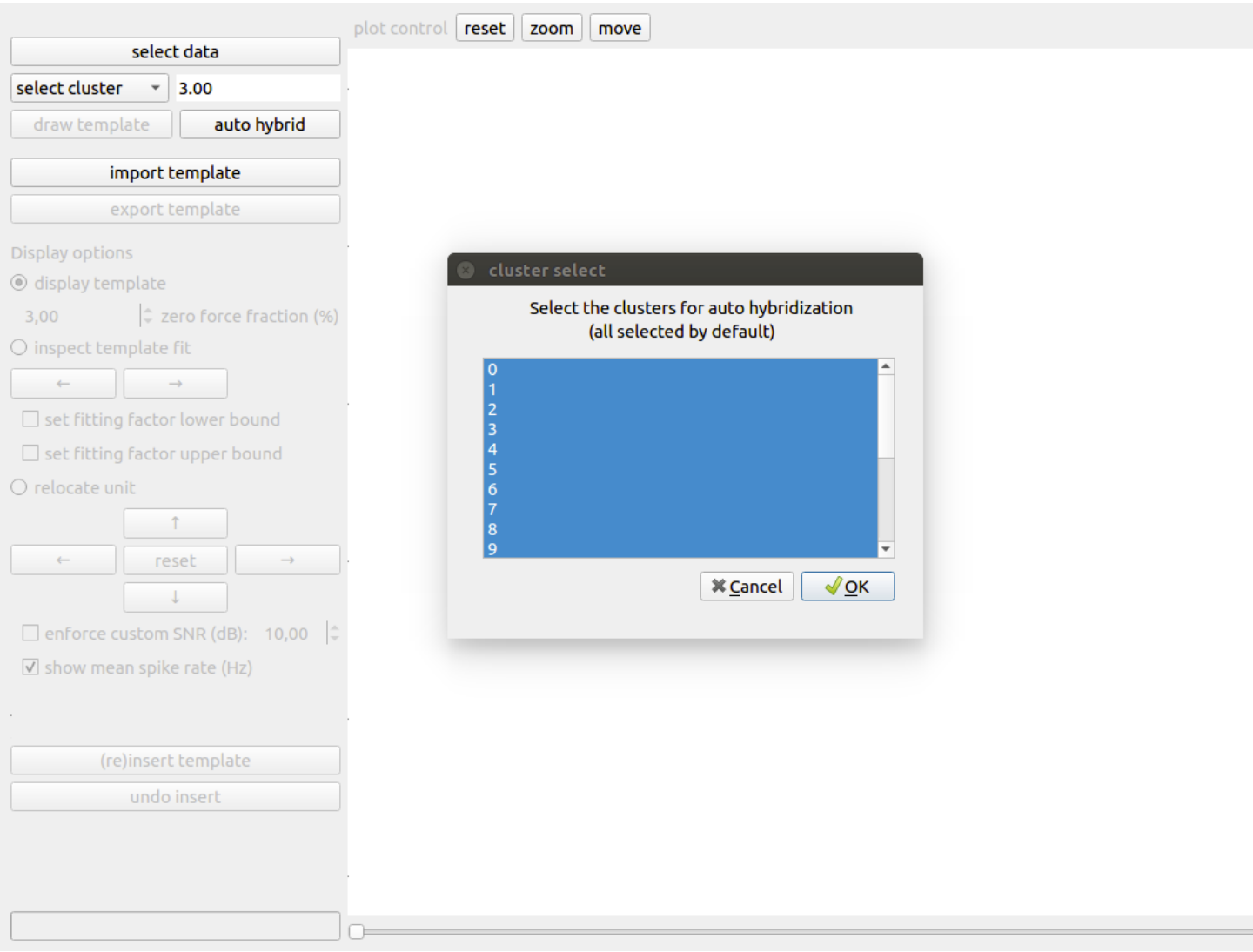

Fig. 4: Fig. 4: Automatic hybridization

CHAPTER 3

Use your own data

The main idea behind SHYBRID is to provide a tool that can be used to transform your own extracellular recordings into ground truth recordings that can be used in the context of spike sorting. In this section we will show you how to use your own data (extracellular recording and initial sorting) with SHYBRID.

3.1 Native support

SHYBRID provides native support for a limited number of file formats only. We will first go over these formats and show you further down how you can easily convert between other file formats and the native SHYBRID formats using SpikeInterface.

As already discussed in *Getting Started with SHYBRID*, a SHYBRID *project* consist of four files as summarized in the table below:

SHYBRID file	native format
extracellular recording	.bin, .raw, .dat
initial spike sorting	.csv, phy-format
probe file	.prb
parameter file	.yml

3.1.1 Extracellular recording

Only binary extracellular recordings are supported natively. SHYBRID assumes that the recording has been high-pass filtered (as is a typical prerequisite in the context of spike sorting). Please note that only binary recordings that encode their samples in a signed data type are supported.

3.1.2 Initial spike sorting

The initial spike sorting is natively supported in two formats:

1. As a CSV-file where the first column contains the neuron ID of the spike and where the second column contains the spike times in samples. For example, consider the following initial sorting example that contains two clusters (neuron 0 and neuron 1), where cluster 0 contains six spike times and cluster 1 contains eight spikes:

Listing 1: *initial-sorting.csv*

```
0,6088484
0,6417026
0,6839332
0,1605072
0,6893813
0,2860165
1,7599880
1,6430392
1,3161103
1,1214103
1,3112478
1,7447323
1,6454460
1,7652192
```

2. In the phy sorting format. This format is commonly used by various spike sorting packages such as KiloSort and SpyKING CIRCUS. Essentially, the phy format consists of a folder with multiple saved numpy arrays that contain information related to the spike sorting results.

Note: When using the phy format for the initial spike sorting, only clusters that have been labeled as *good* (e.g., during manual curation) are loaded by SHYBRID.

3.1.3 Probe file

The probe file that is required by SHYBRID, has to be supplied in the phy format. An example probe file for a neural probe with 16 recording channels is shown below.

Listing 2: *probe.prb*

```
total_nb_channels = 16
radius            = 100

channel_groups = {
  1: {
      'channels': list(range(16)),
      'geometry': {
        0: [0, 0],
        1: [0, 50],
        2: [0, 100],
        3: [0, 150],
        4: [0, 200],
        5: [0, 250],
        6: [0, 300],
        7: [0, 350],
        8: [0, 400],
        9: [0, 450],
        10: [0, 500],
        11: [0, 550],
```

(continues on next page)

(continued from previous page)

```
        12: [0, 600],
        13: [0, 650],
        14: [0, 700],
        15: [0, 750]
      },
      'graph' : []
      }
  }
```

3.1.4 Parameter file

Finally a YAML parameter file has to be supplied that contains the required recording meta information and links together all of the above files. Below, an example parameters file is shown:

Note: The parameter file has to have the same file name as the binary recording, i.e, *recording-name.yml*.

Listing 3: *recording-name.yml* (with CSV initial sorting)

```
---
data:
  fs: 25000
  dtype: float32
  order: C
  probe: /path/to/probe/probe.prb

clusters:
  csv: /path/to/clusters/initial-sorting.csv
...
```

As can be seen from this example parameter file, the following parameters related to the data have to be provided:

- *fs*: represents the sampling frequency in Hz
- *dtype*: represents the datatype that was used to encode the signal samples of the binary recording files
- *order*: contains the order in which the data matrix has been serialized (F: by stacking matrix columns, or C: by stacking matrix rows (more info))
- *probe*: contains the path to the probe file

When the initial spike sorting is provided in the CSV format, the path to the CSV file has to be passed as shown in the above example. However, when the initial sorting is given in the phy format, consider the example below:

Listing 4: *recording-name.yml* (with phy format initial sorting)

```
---
data:
  fs: 25000
  dtype: float32
  order: C
  probe: /path/to/probe/probe.prb

clusters:
  phy: /path/to/phy-initial-sorting
...
```

3.2 Data conversion

Although the binary recording and phy format can be considered as *de facto* standards in spike sorting, many other formats exist. To improve the compatibility of SHYBRID in terms of input data, we implemented SHYBRID extractors for the SpikeInterface project. Besides improved input data compatibility with all technologies that are supported by SpikeInterface, the SpikeInterface integration also enables a straightforward spike sorting of the SHYBRID data and ground truth validation of those sorting results, as we will show in *Sorting and comparing sorting results*.

This section is not meant to be a thorough introduction for SpikeInterface. We will only provide a minimal code example, to give you an idea of how easy it is to convert your data into SHYBRID compatible data. If you want to learn more about SpikeInterface, we would like to point you to their extensive tutorials section.

Listing 5: *data conversion python code example*

```
import spikeinterface.extractors as se

# create a recording and initial spike sorting extractor
# for example when SpyKING CIRCUS was used for the initial sorting
sc_path = '/path/to/sc-initial-sorting'
sorting_ex = se.SpykingCircusSortingExtractor(sc_path)
recording_ex = se.SpykingCircusRecordingExtractor(sc_path)

# define a SHYBRID project folder
project_folder = '/path/to/my-shybrid-project'

# create SHYBRID compatible data from a recording and sorting extractor
se.SHYBRIDSortingExtractor.write_sorting(sorting_ex, project_folder)
initial_sorting_fn = os.path.join(project_folder, 'initial_sorting.csv')
se.SHYBRIDRecordingExtractor.write_recording(recording_ex, project_folder,
                                             initial_sorting_fn)
```

The result of executing the above code example is that all files required by SHYBRID have been generated, based on the SpyKING CIRCUS project. After executing this script, you can immediately load this data into SHYBRID. Although this example is minimal, often nothing more is needed to convert the data from your workflow into a SHYBRID project. Note that SpikeInterface can also be used to perform the high-pass filtering (by adding one line of code) and can also be used to perform a curation of the initial sorting. If you want to know more about these features, we recommend you to consult the SpikeInterface manual.

CHAPTER 4

Sorting and comparing sorting results

4.1 Option 1: SpikeInterface

Eventually, after you've generated hybrid ground truth data, you will want to sort this data and compare the sorting results against the ground truth labels. Luckily, these tasks are easy because of the SHYBRID SpikeInterface integration. Please consider the code example below, which implements an entire spike sorting and ground truth validation in just a couple of lines of code.

```
import spikeinterface.extractors as se
import spikeinterface.sorters as ss
import spikeinterface.comparison as sc

# full filenames to both the hybrid recording and ground truth
recording_fn = '/path/to/recording.bin'
gt_fn = '/path/to/hybrid_GT.csv'

# create extractor object for both the recording data and ground truth labels
recording_ex = se.SHYBRIDRecordingExtractor(recording_fn)
sorting_ex = se.SHYBRIDSortingExtractor(gt_fn)

# perform spike sorting (e.g., using SpyKING CIRCUS)
sc_params = ss.SpykingcircusSorter.default_params()
sorting_sc = ss.run_spykingcircus(recording=recording_ex, **sc_params,
                                  output_folder='tmp_sc')

# calculate spike sorting performance
# note: exhaustive_gt is set to False, because the hybrid approach generates
#       partial ground truth only
comparison = sc.compare_sorter_to_ground_truth(sorting_ex, sorting_sc, exhaustive_
↪gt=False)
print(comparison.get_performance())
```

4.2 Option 2: Using the shybrid validation api

After you have sorted the hybrid recording, either by using your sorting pipeline of choice or SpikeInterface, the sorting results can also be compared to the ground truth through the SHYBRID validation API. Compared to SpikeInterface, our API implements an automatic cluster merging, which might give more realistic sorting results for spike sorting software that has been tuned toward overclustering (i.e., splitting a single unit cluster into multiple clusters). Please consider the following code example which shows you how to use our validation API.

```
import os

from hybridizer.validation import validate_from_phy, validate_from_csv

root = '/path/to/hybrid'
phy_folder = 'phy_results_folder'
hybrid_gt = 'hybrid_GT.csv'

comparison_window = 10

# non-curated spike sorting results
phy_results = os.path.join(root, phy_folder)
# ground truth
hybrid_gt = os.path.join(root, hybrid_gt)

print('compare from phy')
validate_from_phy(hybrid_gt, phy_results,
                  comparison_window=comparison_window)

print('\ncompare from csv (to itself in this example)')
validate_from_csv(hybrid_gt, hybrid_gt,
                  comparison_window=comparison_window)
```

Note: You can easily convert sorting results to the CSV format by using SpikeInterface's *write_sorting* method, as was shown in *Use your own data*.

CHAPTER 5

Extra features

5.1 Exporting template

A template can be exported as a CSV-file. Every channel is exported as a row in the CSV dump. The order in which the channels are exported is depending on the order of the channels in the probe file. For proper reconstruction, the channels in the probe file / recording are should be ideally ordered based on the actual geometry. More concretely, channels are assumed to be ordered by increasing x- and increasing y-coordinates, with the x-coordinate being the fastest changing variable.

A subset of channels can be exported by using the zoom functionality. All channels which have their leftmost plotted point in the zoom window are considered for exporting.

5.2 Importing template

Ground truth data can also be generated in the absence of initial spike sorting results for a certain recording. This can be obtained by importing an external template in CSV-format, where every row in the CSV-file represents the waveform on a channel. The window size is automatically determined. The template spatial width parameter will control the width (i.e., the number of channels in the x-direction) of the template on the probe. The horizontal starting index parameter allows more control over which channel is used as a starting point. The template can also be shaped by adding additional zero-channels to the CSV.

When working with an imported template, the inspect template fit feature will be disabled until a spatial location is chosen and the template is actually inserted in the recording.

5.3 Making figures

The content of the visualization canvas can be saved to an image at any time by pushing the *export plot* button. Remember that the plot control buttons can be used to manipulate the visualization canvas.