Eurographics Workshop on Visual Computing for Biology and Medicine (2023)
C. Hansen, J. Procter, and R. G. Raidou (Editors)

Supplementary Information
Rapid Prototyping for Coordinated Views of Multi-scale Spatial and Abstract Data: A Grammar-based Approach

Philipp Harth[1], Arco Bast[2,3], Jakob Troidl[5], Bjorge Meulemeester[2,3], Hanspeter Pfister[5], Johanna Beyer[5], Marcel Oberlaender[2,4], Hans-Christian Hege[1], and Daniel Baum[1]

[1]Department of Visual and Data-Centric Computing, Zuse Institute Berlin (ZIB), Germany
[2]Max Planck Institute for Neurobiology of Behavior – caesar, Bonn, Germany
[3]International Max Planck Research School for Brain and Behavior, Bonn, Germany
[4]Department of Integrative Neurophysiology, Center for Neurogenomics and Cognitive Research, VU Amsterdam, the Netherlands
[5]School of Engineering & Applied Sciences, Harvard University, USA

S1. Technical Aspects of Entity Selections

Selections of entities that are propagated between views are a central concept in our grammar. Here we give a more technical description how these entities relate to tabular and non-tabular data and how selections referring to different tabular datasets are matched. When starting the Python data server (representing the backend), the user specifies the path to a data folder which contains:

- **.csv* files with tabular data
- a *config.json* file with settings (see below)
- a resources folder with data files in **.json* format

The data server loads the *.csv files into memory and either sends them in their entirety to the client or handles data requests (of certain columns or rows) as needed. This can be controlled by adding *.csv files to the *cached_tables* property in the config file. The user can also specify sampling parameters for large csv files (Listing S1). JSON-based resources are loaded on demand when a view requests them (e.g., a neuron morphology). Such required resources are specified in the *configuration* attribute of the view.

The config file also specifies which views are available in the front end and to which data tables they refer. In the default case, user selections in a view can be mapped to a set of row indices of the underlying data table (i.e., *selectedEntityType="row_index"*). When propagating selections between views that have the same data table, the specified operations (listed in Sect. 4.3 in the main manuscript) are therefore set operations on row indices.

If selections are propagated between views that do not share the same data table, the selected row indices must be mapped from one table to another. How to perform this mapping must be defined in the config file, where the user specifies which columns should be used to match rows (Listing S1). This allows specifying workspaces that combine data from different tables, which may represent different scales (e.g., *neurons.csv* and *synapses.csv* as in case study 1). An example for such a workspace is shown in Fig. S3; all views in it are linked as specified in Listing S5. Note

```
"sampling_settings" : {
    "synapses.csv" : {
        "number" : 15000,
        "seed" : 3000
    },
    "neurons.csv" : {
        "number" : 15000,
        "seed" : 3000
    }
}
```

Listing S1: Sampling settings for tabular datasets in the config-file.

that *intersectSelection* is the default choice here because it allows iteratively refining the initial selection in other views.

While *selectedEntityType* is usually *"row_index"* and the selection represents a set of integer values (i.e., the row numbers), *selectedEntityType* is a string-valued property and can represent any custom data type associated with one particular view. We use this in case study 1 to switch the workspace layout, when the user clicks on the dendrite in the 3D viewer by testing for *selectedEntityType="dendrite"* (Listing S4). In case study 2 (Fig. 5 in main manuscript), we use custom data types to propagate information about the selected simulation samples and the current time step to the morphology views and to the voltage trace views when changes in the *simulation control* view occur (Listing S4). This, of course, requires that the receiving views can handle the respective data types.

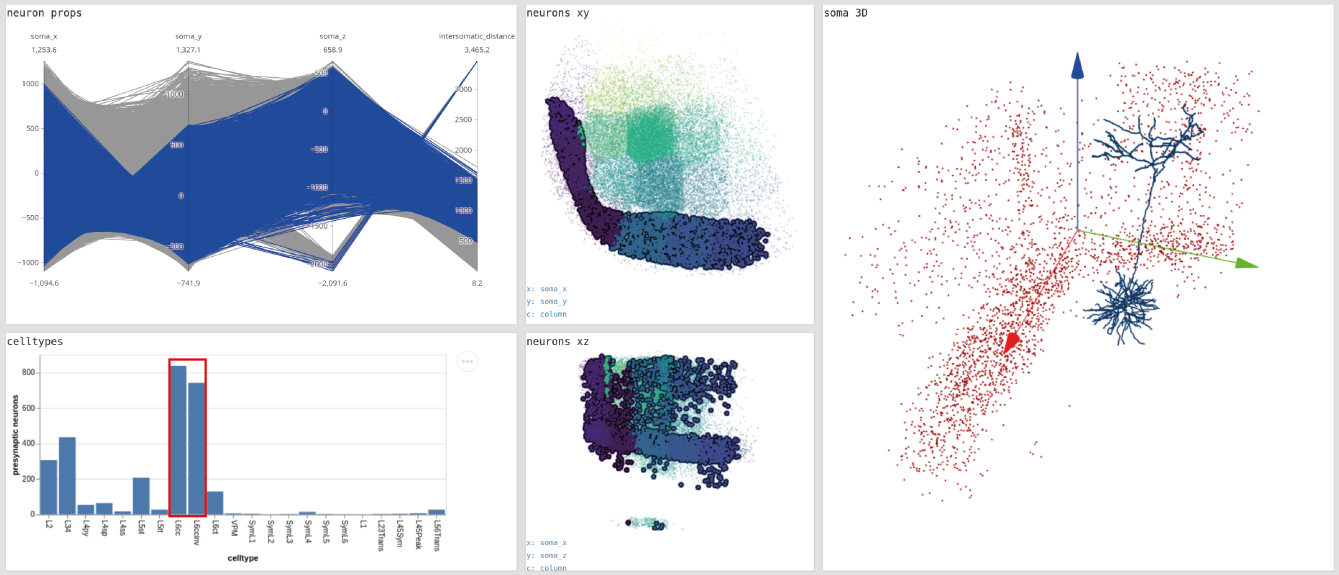

Figure S1: ***(related to Fig. 2; case study 1)*** *Lasso-selecting remote cortical columns in the xy-projection view shows that the presynaptic neurons from these columns are mostly layer-6 neurons (highlighted in the cell types view).*

```
"table_mapping" : [{
        "left_table" : "neurons.csv",
        "left_key" : "neuron_id",
        "right_table" : "synapses.csv",
        "right_key" : "neuron_id"
    },{
        "left_table" : "synapses.csv",
        "left_key" : "neuron_id",
        "right_table" : "neurons.csv",
        "right_key" : "neuron_id"
    }],
```

Listing S2: Specification of relations between tables in config file. *left_key* and *right_key* define the corresponding columns that should be used when propagating selections of rows from *left_table* to *right_table*.

```
"soma 3D": [{
    "action": {
      "changeLayout": "L2"
    },
    "filter": {
      "currentLayout": "L1",
      "selectedEntityType": "dendrite"
    }
}]
```

Listing S3: ***(related to case study 1)*** Switching workspace layout when the user clicks at the dendrite in the *soma 3D* view.

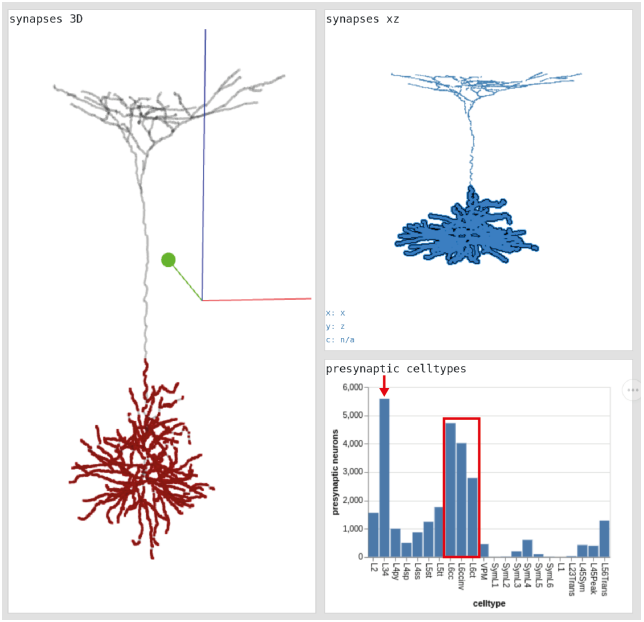

Figure S2: ***(related to Fig. 3; case study 1)*** *Synaptic inputs near the soma originate mostly from layer-3/4 neurons (red arrow) and layer-6 neurons (red rectangle).*

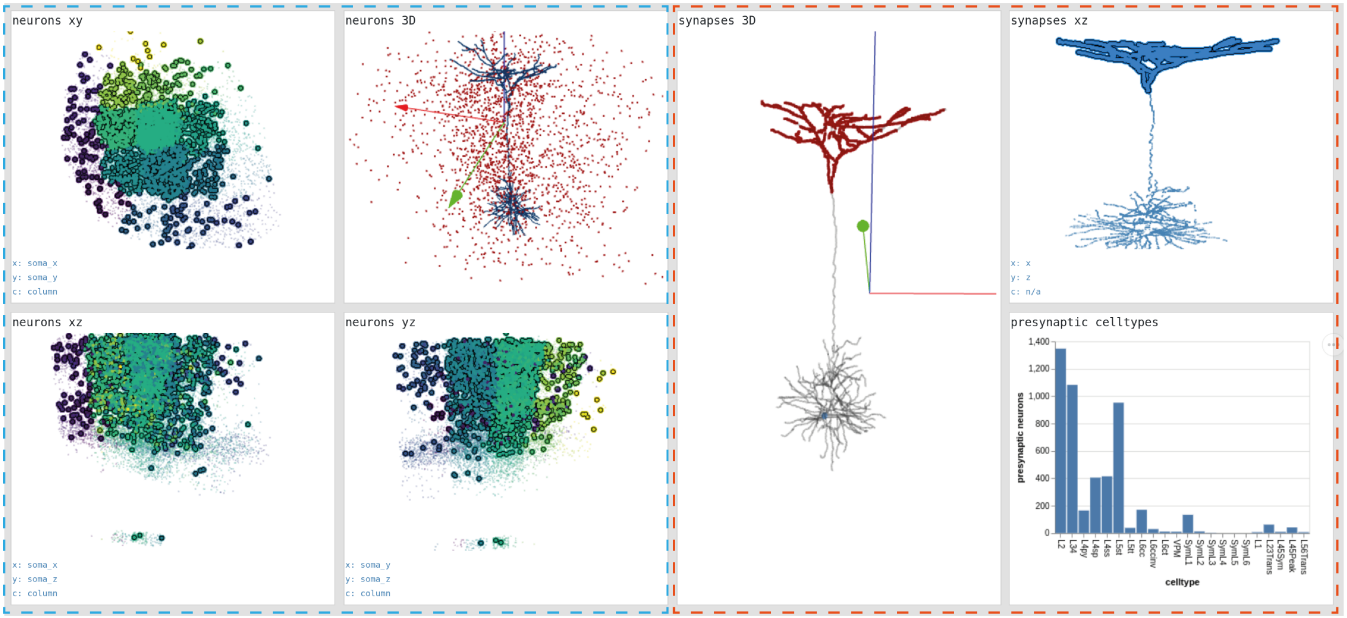

Figure S3: ***(related to case study 1)*** *Example of a single multi-scale workspace in which two levels of detail are combined and linked. Locations of presynaptic neurons (views highlighted with blue dashed line) and locations/properties of synapses on the dendrite (views highlighted with orange dashed line). Neurons and synapses were downsampled to n=15,000.*

```
"simulation control": [{
    "action": {
        "assignData": [
            "dendrite",
            "dendrite 2",
            "membrane potential",
            "membrane potential 2"
        ]
    }
}]
```

Listing S4: ***(related to Fig. 5; case study 2)*** Propagating simulation data selected in *simulation control* to morphology views (*dendrite*, *dendrite 2*) and voltage trace views (*membrane potential*, *membrane potential 2*).

```
"interactions": {
    "neurons xy": [
      {
        "action": {
          "intersectSelection": [
            "neurons xy",
            "neurons xz",
            "neurons yz",
            "neurons 3D",
            "synapses xz"
          ]
        }
      }
    ],
    "neurons xz": [
    ...
}
```

Listing S5: Specification of interactions between views in the workspace shown in Fig. S3, using the view *neurons xy* as an example.

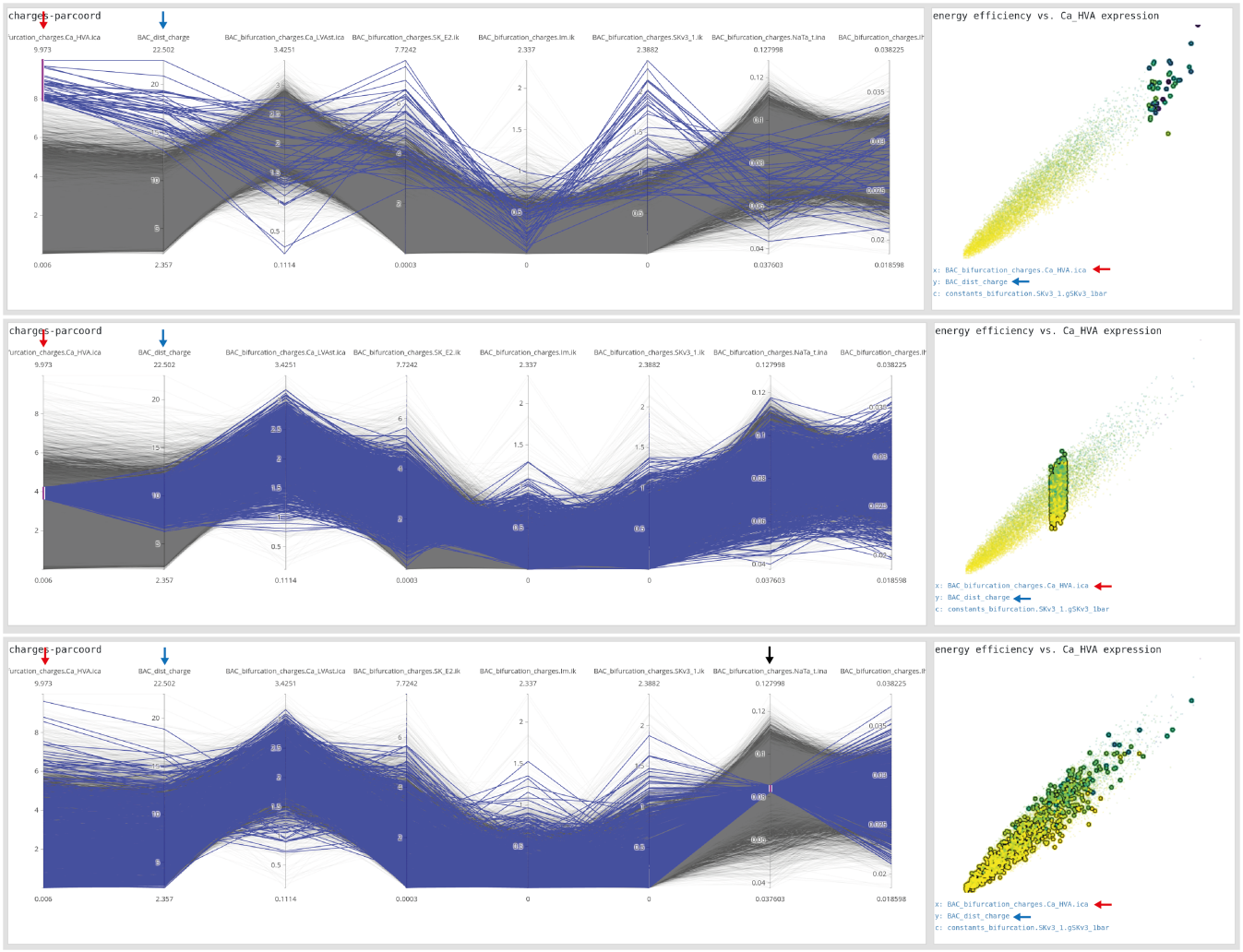

Figure S4: ***(related to case study 2)*** *Top, middle: Expression of the calcium channel near the bifurcation zone in the apical dendrite (feature BAC_bifurcation_charges.Ca_HVA.ica; red arrow) restricts the amount of total charge exchanged (feature BAC_dist_charge, blue arrow). This strong correlation is also observed in the linked scatter plot (right), where these features represent the axes. The coloring in the scatter plot encodes the energy efficiency of dendritic computations. Bottom: Selecting a small value range in the parallel coordinates plot for another feature (black arrow) yields no obvious correlations to other features.*