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

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

Abstract

Visualization grammars are gaining popularity as they allow visualization specialists and experienced users to quickly create static and interactive views. Existing grammars, however, mostly focus on abstract views, ignoring three-dimensional (3D) views, which are very important in fields such as natural sciences. We propose a generalized interaction grammar for the problem of coordinating heterogeneous view types, such as standard charts (e.g., based on Vega-Lite) and 3D anatomical views. An important aspect of our web-based framework is that user interactions with data items at various levels of detail can be systematically integrated and used to control the overall layout of the application workspace. With the help of a concise JSON-based specification of the intended workflow, we can handle complex interactive visual analysis scenarios. This enables rapid prototyping and iterative refinement of the visual analysis tool in collaboration with domain experts. We illustrate the usefulness of our framework in two real-world case studies from the field of neuroscience. Since the logic of the presented grammar-based approach for handling interactions between heterogeneous web-based views is free of any application specifics, it can also serve as a template for applications beyond biological research.

CCS Concepts

• ***Human-centered computing*** → ***Visual analytics;*** • ***Software and its engineering*** → ***Specification languages;*** • ***Applied computing*** → ***Biological networks;***

1. Introduction

Visualization frameworks have greatly simplified the process of supporting interactive visual analysis workflows in different domains [RCM*20]. Desktop-bound applications and libraries, such as Amira [SWH05], ParaView [par], and Inviwo [JSS*20] provide powerful toolkits to create interactive visualizations without the need for reimplementing basic functionality, such as volume rendering. Parallel to this, a rich ecosystem of visualization frameworks and libraries has developed in the web-based world, for example, D3.js [BOH11], Vega(-Lite) [SWH14, SRHH16, SMWH17], Babylon.js [bab], and Plotly.js [Plo]. In fact, low-level visualization frameworks, such as D3.js [BOH11], and high-level frameworks, such as Vega-Lite [SMWH17], have become de facto standards and have gained widespread adoption in the visualization community as a result.

An emerging trend for specifying visualizations in such frameworks, particularly in web-based scenarios, are JSON-based visualization grammars, which can formally be seen as instances of domain-specific languages (DSLs) [McN22]. The prime example in this category of visualization frameworks is Vega-Lite [SMWH17], where both visual encodings of the displayed data and interaction behavior [SRHH16] are defined in a JSON-based specification language. Because the grammar-based specification and resulting visualization can be easily changed, this is an attractive approach for prototyping and iterative refinement of visual analysis tools.

However, there are three main shortcomings of current visualization grammars: (1) They are typically tied to a specific framework and cannot easily be mixed and matched at will. (2) They do not support 3D views and brushing & linking of combined 2D and 3D views. (3) They are not flexible enough to support entire analysis workflows where the user explores multiple scales.

When creating visual analysis tools for domain experts from neurobiological research, we face all of the aforementioned challenges. Technological advances now make it possible to scan neural tissue samples at nanoscopic resolution [KD18], allowing the study of brain networks in unprecedented detail. A recurring structural feature of neural networks found in such experimental studies is their

DOI: 10.2312/vcbm.20231218

multilevel organization from larger brain regions to individual neuronal cells with complex branching structures, down to nanoscopic synaptic contacts between neurons [FZB16]. Integrated visual analysis of such data [BTB*22] requires the coordination of different types of views, including both information views and 3D anatomical views, in configurations that are adjusted to the level of detail required for the respective analysis task.

In addition to the multilevel organization of brain networks, a further challenge is the visual analysis of simulated neural activity data [BO22], which adds a temporal dimension to the neuroanatomical data under study. Neural activity simulations performed by our domain experts are controlled by more than 30 biophysical parameters and generate hundreds of spatio-temporal features that describe the neural activity response of the simulated neurons [BO22]. Thus, we are dealing with high-dimensional data that domain experts would like to analyze visually – both with abstract representations suitable for high-dimensional parameter analysis [SHB*14] (such as parallel coordinates or 2D embeddings using dimensionality reduction techniques [SZS*17]) and 3D representations of neuroanatomy.

Contributions. The novelty of the proposed interaction grammar is that it allows the integration of existing visualization frameworks, namely Vega(-Lite) [SWH14, SRHH16, SMWH17], which are themselves internally configured through a JSON-based specification, with regular visualization libraries, such as Plotly.js [Plo] or Babylon.js [bab]. Importantly, this integration also includes the combination of standard information views with 3D (anatomical) views. To the best of our knowledge, doing so with a grammar-based approach (as opposed to manually programmed event and interaction handling) has not been addressed so far. Unlike previous grammar-based approaches, the presented interaction grammar also controls the overall layout of the application workspace. Finally, grammar-based approaches have not been employed for visual analysis in neuroscience.

2. Related Work

Multiple coordinated views. Multiple coordinated views are a common tool to offer different perspectives on data. For example, brushing and linking [DGH03, Dol04] can be used to create associations between multiple visualizations, representing different attributes of multi-faceted data [KH13]. Roberts [Rob07] surveyed visualization approaches using multiple coordinated views, which have since then become a standard technique in visualization systems and dashboard design across all domains. Streamlit [Str] and Dash [Das] are examples of existing frameworks for building data analytic dashboards in which different types of views can be combined; however, they do not use a JSON-based specification.

Visualization grammars. In a recent survey, McNutt [McN22] formalizes the design space of JSON-style domain-specific languages (DSLs) for visualization definition. DSLs have been applied in the biomedical domain [LWLG22], including visual analysis of neurobiological data [TCG*22]. Troidl et al. [TCG*22] focused primarily on spatial neighborhood analysis and comparison of spatial brain structures but did not explicitly link multiple views through individually defined interactions. Domain-agnostic visualization grammars [SWH14, SRHH16, SMWH17, ZPWS22] often model interactions by modeling data selections as a subset of the data that users want to manipulate. More recently, visualization grammars specifically for 3D data have been introduced [Sch21], which do not require users to be familiar with the intricate implementation details of 3D rendering systems, but rather allow specifying certain rendering and interaction parameters through JSON. Our grammar specification unifies these previous approaches by combining (1) visualization specification, (2) view layout definition, and (3) interaction definition for both information and 3D visualizations.

Multi-scale visualization. When analyzing measured and computed high-resolution data from neurobiology, experts must be able to examine detailed features of individual simulation runs, such as neuron morphology, synapse types, and activation potentials. On the other hand, larger-scale distributions and statistical patterns are of great importance to put small-scale findings into context. Therefore, multi-scale visualization is a necessary tool to navigate between the different scales of such highly complex datasets. Cakmak et al. [CJS*22] surveyed the state of the art in multi-scale visualization. A typology of abstract visualization tasks that also applies to multi-scale analysis scenarios is provided by Brehmer and Munzner [BM13]. An important aspect of multi-scale visualizations are data summarization techniques, which have been reviewed in [SGS18]. For example, previous interactive tools [AABS*14, MAAB*17, TCG*22] have used multiple levels of detail to analyze neuronal tissue. By increasing the information density and leveraging data abstraction, visual representations can be continuously transformed from analyzing a single object to effectively comparing larger groups of objects. A recent application example for multilevel analysis/semantic zoom in the domain of genomics is the grammar-based toolkit called Gosling [LWLG22].

Visualization for neuroscience. Beyer et al. [BTB*22] survey the landscape of visualization approaches for high-resolution image-based neuroscience. Neuron simulation models can be augmented by visual analysis to better understand the effect of intricate parameter configurations on simulation outcomes. VIOLA [SCH*18] is an interactive visualization tool that uses multiple coordinated views for exploratory analysis and quality assessment of spatiotemporal features of a neuron simulation model. In contrast, Nowke et al. [NDPW*18] propose an interactive visual approach to study parameter configurations of connectivity generation models. VisNEST [NSA*13] allows visual exploration of simulations that consider both the low-level neuronal activity of individual neurons and the general interplay between multiple brain regions. NeuroVIISAS enables the integration of different neuro-ontologies, imaging data, and neural activity simulation data in one framework [SE12]; it also facilitates comparative analyses of connectome data [SJES19]. Open Source Brain [GCM*19] is a web-based repository for neural network models with comprehensive visualization capabilities. Other approaches focus entirely on neuronal connectivity analysis without considering activity simulations (e.g., [TWC*22, HVU*22, BSG*09]).

```
{
  "views" : [{
        "name" : <view_name>,
        "type" : <type_name>,
        "dataTable" : <table_name>,
        "dataColumn" : [<column_name>, ...],
        "configuration" : {...},
        "minNumDataColumns" : int,
        "maxNumDataColumns" : int
  }],
  "grid" : { // partitions workspace
    "cols" : int, "rowHeight": int, "width" : int
  },
  "layouts" : {
    <layout_name> : {
        "view" : <view_name>,
        "x": int, "y": int, "w": int, "h": int
    }
  }
  "initialLayout" : <layout_name>
  "interactions" : {
    <source_view_name> : [{ // source of interaction event
        "filter" : {
            "currentLayout" : <layout_name>,
            "interactionType" : <interaction_type>
            "selectedEntityType" : <type_name>
            "numSelectedRange" : int | [int, int],
        }
        "action" : {
            "assignData" : [<target_view_name>, ...],
            "clearData" : [<target_view_name>, ...],
            "assignSelectionAsData" : [<target_view_name>, ...],
            "assignSelection" : [<target_view_name>, ...],
            "intersectSelection" : [<target_view_name>, ...],
            "unionSelection" : [<target_view_name>, ...],
            "clearSelection" : [<target_view_name>, ...],
            "changeLayout" : <layout_name>
        },
    }]
  }
}

<interaction_type> = "select" | "deselect" | <custom_event_name>
```

Listing 1: JSON structure of the proposed interaction grammar for coordinating heterogeneous views. All <*_name>-fields are string-valued. In the default scenario, the *selectedEntityType* is *"row_index"*, and the selection represents a set of rows in the underlying *dataTable* (see Supplementary Information).

3. Requirements and Design Goals

In this paper, we deal with multi-scale neuroanatomical data and high-dimensional neural activity simulation data (i.e., biophysical model parameters and features describing the neural activity response of the simulated cells). In order to support the analysis questions of our domain experts, we developed an interaction grammar (Sect. 4) that supports easy and fast prototyping for coordinated views of multi-scale and abstract data. For a detailed description of the associated data and analysis tasks, we refer to the respective case studies in Sects. 6.1 and 6.2. To clarify the requirements, we had regular virtual meetings with our collaborators over several weeks in which we discussed their needs and performed interviews. Through these meetings, we iteratively defined and refined the requirements and design goals of the interaction grammar. This resulted in the following requirements:

R1: The analysis workflows of our domain experts require multiple coordinated views [Rob07] that include information views (e.g., scatter plots or line charts) and 3D anatomical views.
R2: To be of practical use in neuroscientific research and many other application fields, the visual analysis tools should be available on short notice and be easily changeable as the domain expert's insights and research questions evolve.
R3: The multi-scale organization of neuroanatomical entities requires that the application supports multiple and highly heterogeneous combinations of views, depending on the scale of analysis and the specific research questions.

These requirements translate into the following design goals that guided both the specification of the interaction grammar (Sect. 4) and the implementation of the tool (Sect. 5):

G1: Support the combination of different view types (**R1, R3**).
G2: Integrate existing visualization frameworks and libraries to facilitate code reuse and reduce development time (**R2**).
G3: Enable interactive and iterative refinement (i.e., rapid prototyping) of views, workspace layout, and interaction logic (**R2**).
G4: Avoid hard-coded or domain-specific ontologies tied to a priori known analysis questions or workflows (**R2, R3**).
G5: Support dynamic transitions between different workspace configurations (**R3**).

4. Interaction Grammar

To enable rapid prototyping and iterative refinement of the tool (**G3**), we chose a grammar-based approach to control the overall workspace layout and interaction behavior between views (Fig. 1**A,B**). The syntax specification of the resulting JSON-based grammar is given in Listing 1. We describe the different components of the grammar in the following sections.

4.1. Specifying Views

The *views* property in the grammar allows defining a set of named views that can have different types. Views have the following attributes in the grammar:

- *name*: User-selected name for the view.
- *type*: Type specifier describing what the view represents and with which library it is implemented. For example, one view may be of type *vega* [SMWH17] representing a line chart. Another one may be of type *babylon* [bab] representing a 3D view (**G1**, **G2**).
- *dataTable*: Each view is associated with a data source that maps to a csv file provided to the data server in the backend (see Sect. 5).
- *dataColumn*: Views can reference selected columns in the data table.
- *configuration* (optional): Additional parameters that can be supplied to control the exact representations of the views (e.g., color or mark type); which parameters can be specified depends on the type/visualization library of the view.
- *min/maxNumDataColumns*: Number of data columns that are required/can be specified for the view type (needed for the UI editor and consistency checks).

An example of the view specification section of the grammar is shown in Fig. 1D.

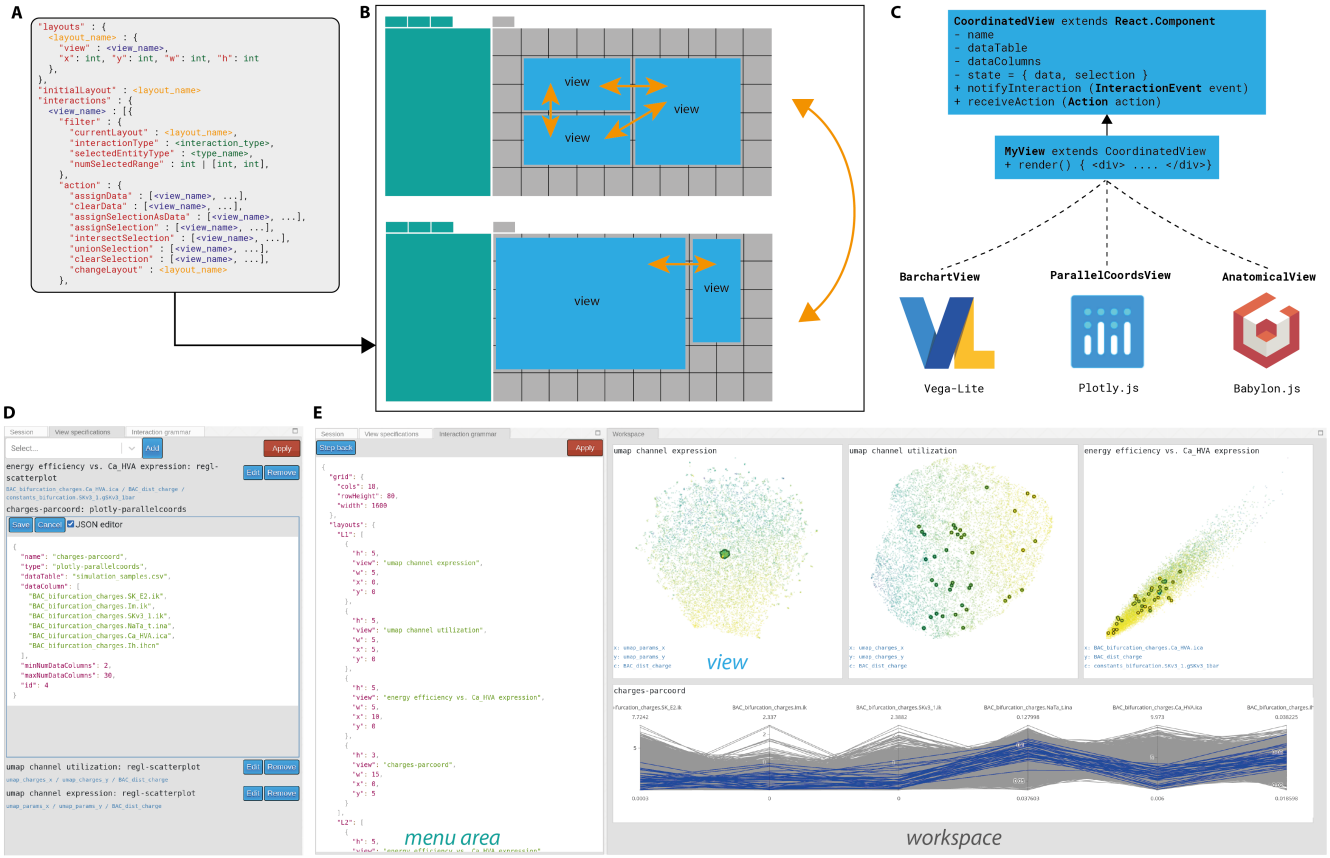

Figure 1: *Technical outline and components of the user interface.* ***A:*** *JSON specification of interaction grammar.* ***B:*** *Interaction grammar controls overall workspace layout and event propagation between views.* ***C:*** *Views from different existing libraries can be integrated by implementing a thin wrapper class that interfaces with our framework (see Sect. 5.3).* ***D:*** *View specification panel in the menu area.* ***E:*** *Interaction grammar panel (left) and workspace (right).*

4.2. Specifying Multi-Scale Analysis Layouts

The layout of the workspace area is also controlled by the grammar and can dynamically change (Fig. 1B) to support multi-scale/multi-level analysis workflows that require different types and arrangements of views (**G5**). The named layouts are specified using the following properties of the grammar:

- *grid*: Specifies how the workspace area is partitioned into equally sized grid cells, according to React-Grid-Layout [rea].
- *layouts*: A set of named layouts. Every layout references one or multiple views (*view*) by their name and specifies which grid cells the view should occupy on the workspace (x, y, w, h). For example, {x: 0, y: 0, w: 5, h: 4} positions the view on the top left corner of the workspace and makes it occupy 5 grid cells horizontally and 4 vertically. For details of the layouting scheme (e.g., how overlaps are resolved), refer to React-Grid-Layout [rea].
- *initialLayout*: Name of the workspace layout that is shown when the application is started.

4.3. Specifying Interactions and Linking between Views

The *interactions* property in the grammar defines how interaction events in one named source view are propagated to the specified target views and how they may alter the workspace layout. Interaction event-triggered actions can depend on the following optional filter conditions:

- *currentLayout*: Only propagate event when the application is currently in the specified workspace layout state.
- *interactionType*: "select", "deselect", and any named event provided by one of the views.
- *selectedEntityType*: Type of data item(s) that was/were selected by the user. We chose to keep these entity types generic and do not introduce neuroscience terminology into the grammar (**G4**).
- *numSelectedRange*: How many data items were selected by the user? For example, this condition can be used to show detail views or change the layout of the workspace when one or a small number of entities in a large collection are selected (**G5**).

The operations under the *action* property define what happens if the *filter* condition is met. In our concept, every view v is associated with a set of data items D and a subset of data items S that are

(locally) selected in the view.

$$view = (S, D) \qquad S \subseteq D$$

Note that D and S may be empty. On specified interaction events, D_{source} and S_{source} from the view in which the event originated are propagated to the named target views as follows:

specified action	operation in named target views
assignData	$D_{target} \leftarrow D_{source}$
clearData	$D_{target} \leftarrow \emptyset$
assignSelectionAsData	$D_{target} \leftarrow S_{source}$
assignSelection	$S_{target} \leftarrow S_{source}$
intersectSelection	$S_{target} \leftarrow S_{target} \cap S_{source}$
unionSelection	$S_{target} \leftarrow S_{target} \cup S_{source}$
clearSelection	$S_{target} \leftarrow \emptyset$

If the *changeLayout* action is specified, the application workspace will be dynamically changed as a result of the interaction (**G5**).

In conclusion, our grammar depends on data item selections, how and under what conditions they are propagated between views, and how they affect the workspace layout.

5. Implementation

To offer platform-agnostic access to the tool and to take advantage of the robust capabilities of already-existing web-based visualization frameworks (**G2, G3**), we decided to develop an online tool. We describe details of the implementation in the following sections.

5.1. Components of User Interface

The main UI components of the framework are depicted in Fig. 1B; they consist of the menu area on the left with multiple control panels and the main application workspace on the right. The configuration of views in the workspace is entirely controlled by the interaction grammar, which modifies its configuration based on the user-specified interactions. The **menu area** includes the following panels:

- **Session panel.** It provides standard project management settings to load and save the state of the analysis session.
- **View specification panel** (shown in Fig. 1D). It defines the views that can be referenced from the grammar and specifies to which data sources they refer. The panel serves as GUI editor to modify the *"views"* section of the interaction grammar. The UI can be disabled to edit the JSON specification directly. An example of the underlying JSON specification is shown in Fig. 1D.
- **Interaction grammar panel** (shown in Fig. 1E). It provides a code editor to modify the interaction grammar directly from within the application (**G3**).

The **workspace** is partitioned into equally sized grid cells as specified in the *"grid"* section of the grammar (Listing 1). Views are then assigned to occupy one or multiple of these grid cells, as specified in the *"layout"* section of the grammar (Sect. 4), which allows defining multiple named layout configurations. The structure and contents of the workspace, therefore, change dynamically (Fig. 1B), depending on user interactions and the layout configuration that is automatically activated.

5.2. Front- and Backend

We wanted to integrate the functionalities of Vega-Lite [SMWH17] for information views into our tool. To realize the case studies in Sects. 6.1 and 6.2, we additionally used the following visualization libraries: Babylon.js [bab] for WebGL-enabled 3D views, Plotly.js [Plo] for parallel coordinates views, and Regl-Scatterplot [Lek23] for WebGL-enabled scatterplots with large numbers of points. Our application was built using the Meteor [met] web-development framework in combination with React. The grid-based layout of the application workspace is based on React-Grid-Layout [rea].

In the backend, we run a Python server, which reads all csv files in a designated folder. Mapping of these table-based data records to the respective views can be specified by the user of the tool in the view specification panel (see Sect. 1D). Other resources in JSON format (e.g., the neuron morphology shown in the anatomical view in Fig. 3A are read from a designated resource folder and supplied to the client by the backend server. Time-resolved neural activity simulations for selected biophysical parameters (as shown in Fig. 5) are computed on-demand on a separate data server (Python) that runs a custom-implemented neural activity simulator [BO22].

5.3. Extensibility and Code Availability

To add additional views in the front end, a wrapper class that is derived from *CoordinatedView* needs to be created by the developer (Fig. 1C). The wrapper class connects the events from our framework to the JavaScript library in which the view is implemented. Several example implementations of the wrapper class are provided in the github repository (see below). For data requests that refer not to table-based (i.e., csv) or JSON-based records, the backend server has to be modified, or a separate data server needs to be created (as we did for the on-demand computations of neural activity simulations).

The code is available at `https://github.com/zibneuro/coordinated-views`.

6. Evaluation

We demonstrate the utility of the proposed interaction grammar for the rapid prototyping of coordinated views using two case studies from ongoing neuroscientific research. The domain scientists are two Ph.D. students (first-year physics and fifth-year neuroscience/medicine) and their group leader/principal investigator from the *MPI for Neurobiology of Behavior – caesar*; all three are coauthors of this paper. The domain experts used the tool under our guidance in four virtual sessions (using screen sharing and remote control of mouse and keyboard) in which we asked them to comment on their interactions and share their thoughts (think-aloud method [LBI*12]). The case studies described below are based on the protocols of these sessions.

6.1. Case Study 1: Validating Anatomical Models

The domain experts in our group created detailed anatomical models of specific brain regions to perform biophysical simulations of

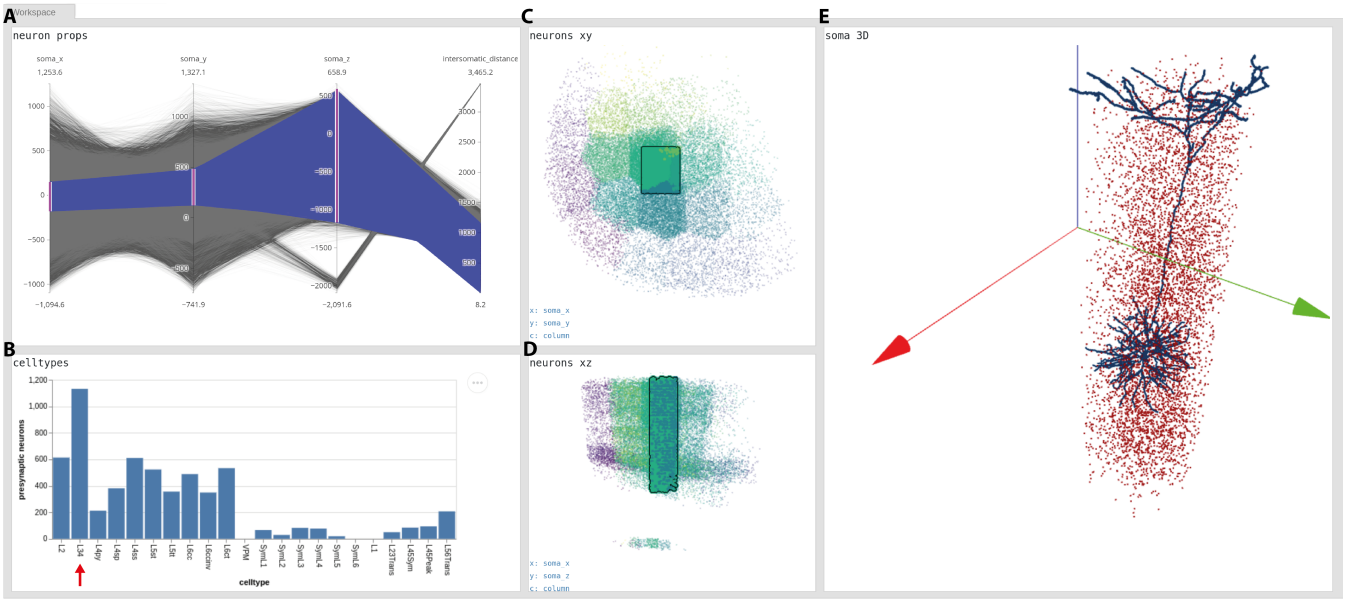

Figure 2: *Workspace as used in the first case study (Sect. 6.1). **A:** Parallel coordinates view of presynaptic neuron properties using Plotly.js. **B:** Breakdown of selected neurons by cell type using Vega-Lite; most presynaptic cells in the current selection are layer-3/4 neurons (red arrow). **C:** xy-projection of neuron soma locations (colored by cortical column) using a WebGL-enabled scatter plot. **D:** xz-projection of soma locations. **E:** 3D view of presynaptic neurons (red) and dendrite morphology of postsynaptic neuron (blue) using Babylon.js. Workspace layout and brushing & linking of views are controlled by the interaction grammar.*

individual neurons and their neighbors [BO22, ENG*20]. A brain region of particular interest to them is the somatosensory cortex in rat, which processes sensory inputs from the whiskers in the snout of the animal [UHM*22]. In one realization of this anatomical model, the following anatomical structures are found (from large to small):

1. 24 cortical columns: subregions that partition the modeled brain region,
2. 32,587 presynaptic neurons (treated as point-like objects) whose cell bodies are located in these columns,
3. the detailed 3D dendrite morphology of a selected postsynaptic neuron, and
4. 42,853 synaptic contacts from the presynaptic neurons to dendrites of the postsynaptic neuron.

A recurring nontrivial task that cannot be easily automated is the validation of the rich structural features in these digitally generated anatomical models and the identification of potential limitations of the model. This validation step, which is performed visually and includes the tasks enumerated below, is a prerequisite for ensuring that subsequent biophysical simulations described in the second case study (Sect. 6.2) are realistic and interpretable.

T1: Characterize cell populations of presynaptic neurons that connect to one postsynaptic neuron, which is to be simulated as a full-compartmental model (i.e., with spatially resolved morphology at subcellular resolution). This requires understanding where the presynaptic neurons (which are modeled as point-like objects) are located and how they are divided by cell type.
T2: Inspect the average connectivity statistics from the presynaptic populations of point-like neurons to the postsynaptic full-compartmental neuron and examine to which degree these model parameters comply with known empirical knowledge.
T3: Group synapses by the cell type of the incoming connections. This involves inspecting synaptic connections in 3D on the dendrite morphology of the simulated postsynaptic neuron.

The domain experts in our group suggested showing the coordinates of the cell bodies (soma_x, soma_y, soma_z) in the model reference frame and their distance to the soma of the postsynaptic neuron (intersomatic_distance), for which the users created a parallel coordinates view (Fig. 2A). A subpopulation of presynaptic neurons became immediately apparent, which is located deeper than all other cells in the model (negative z-coordinate) and, in consequence, has a higher intersomatic distance to the postsynaptic neuron. We added a bar chart to the workspace (Fig. 2B) that groups all neurons by their cell type (**T1**). The interaction grammar was modified accordingly to sync the selection in the parallel coordinates view to the bar chart. It then became clear that this subpopulation of cells comprises of VPM (ventral posteromedial nucleus) neurons located in the Thalamus, while the other group of neurons resides in the cortex. This was confirmed when adding an xz-projection view of the soma coordinates (Fig. 2D).

The density of presynaptic cells in the xz-projection view is highly heterogeneous and differs greatly between cortical layers (i.e., along the z-coordinate). After selecting the upper and lower layers in the projection view using lasso-selection and inspecting the respective breakdown of the selection by cell type, the experts commented that these non-uniform distributions of presynap-

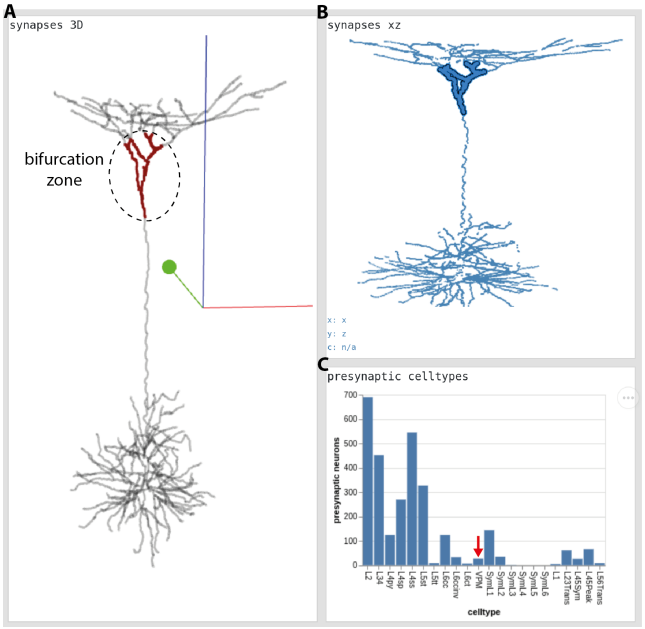

Figure 3: *Workspace for detailed view of synapses on the dendrite. **A:** Selected synapses near the bifurcation zone of a dendrite. **B:** xz-projection of synapses with selection. **C:** Breakdown of selected synapses by cell type of presynaptic neuron. Connections from VPM neurons are likely underestimated in this model (red arrow).*

tic cells reflect the horizontally extended morphologies of layer-2 and layer-6 neurons [NEJ*15]. The model therefore adequately captures the complexity of these empirically established connectivity patterns (**T2**). The experts also observed that in the cortical column, in which the postsynaptic neuron is located, the density of presynaptic cells is much higher than in adjacent areas (along the x-direction). To gain a better spatial understanding of the density distribution of presynaptic neurons (**T1**), we rearranged the workspace layout using the interaction grammar editor and added an xy-projection view (Fig. 2C) and a 3D view that also show the morphology of the postsynaptic neuron (Fig. 2E). The experts lasso-selected the central column in the xy-projection view, confirming that the postsynaptic neuron is mainly targeted by layer-3/4 neurons from the same cortical column (Fig. 2B) and that the proportion of layer-6 neurons [NEJ*15] increases when more remote columns are selected (Fig. S1; **T2**).

The experts were interested in the locations on the dendrite where the neurons establish synaptic connections. For this purpose, we created a second workspace layout showing a detailed 3D anatomical view of synaptic contacts on the dendrite (**T3**), together with an xz-projection view of the synapse locations and breakdown of the incoming synaptic connections by cell type (Fig. 3). The experts selected the synapses near the bifurcation zone of the dendrite (Fig. 3A, B), noting that targeted synaptic contacts from VPM neurons that had been observed in latest experimental findings [GBNO21] are likely underrepresented (Fig. 3C) in the current version of the model (**T2, T3**). When synapses located near the soma were selected, the cell types view (Fig. 3C) showed that they originate mostly from layer-6 neurons (as would be expected due to their depth along the z-axis), but remarkably also from layer-3/4 neurons (Fig. S2), which can be attributed to their vertically extended morphologies (**T2, T3**).

In conclusion, visual examination of this anatomical model with our tool verified that morphological restrictions and cellular connectivity metrics are in line with empirical knowledge about structural connectivity in this brain region [UHM*22]. Visual analysis also uncovered the remaining limitations of the model in terms of intricate subcellular wiring preferences, such as the targeted synapse formation of VPM neurons at the bifurcation zone of the dendrite [GBNO21].

To create an integrated analysis workflow, we can combine both workspaces, for example, by specifying that the detail workspace with synaptic connectivity information opens when the user clicks on the dendrite morphology in the overview workspace (see Supplementary Information, Listing S3). We could also specify a single workspace in which both levels of detail are combined (see Supplementary Information, Fig. S3).

6.2. Case study 2: Analyzing Biophysical Simulations

The goal of the domain experts in their biophysical simulations is to understand the relationship between the expression of ion channels, which control the membrane potential on the surface of the dendrite, and the energy efficiency of the computations that the neuron performs [BO22].

The simulations result in high-dimensional data sets. In this case study, we consider a simulation experiment in which $n = 15,000,000$ simulation runs were performed, of which we used $n = 10,000$ as an example in the evaluation, where each run was associated with

1. sampled simulation parameters ($dim = 35$) that result in observable functional features of the cell ($dim = 345$). The simulation parameters control *ion channel expressions* and the functional features describe how much current flows through the ion channels (i.e., *ion channel utilization)* as well as characteristics of the membrane potentials; collectively, we refer to these quantities as scalar *features* ($dim = 380$).
2. spatiotemporal data: membrane potentials and ion channel activity mapped to the surface of the dendrite morphology for every time step in the simulation.

Interpreting these simulation data requires performing the following analysis tasks:

T1: Understand the relationship between ion channel expressions and ion channel utilizations.
T2: Identify features that have descriptive value for energy-efficient computation from the large set of available features.
T3: Inspect the spatially and time-resolved neural activity on the dendrite morphology and compare it for different simulation parameters.

We first aimed to investigate the relationship **(T1)** between ion channel expression and the response of the cell to current stimuli, which is a functional property of the cell that arises from the dynamic activation of these ion channels. To this end, the experts

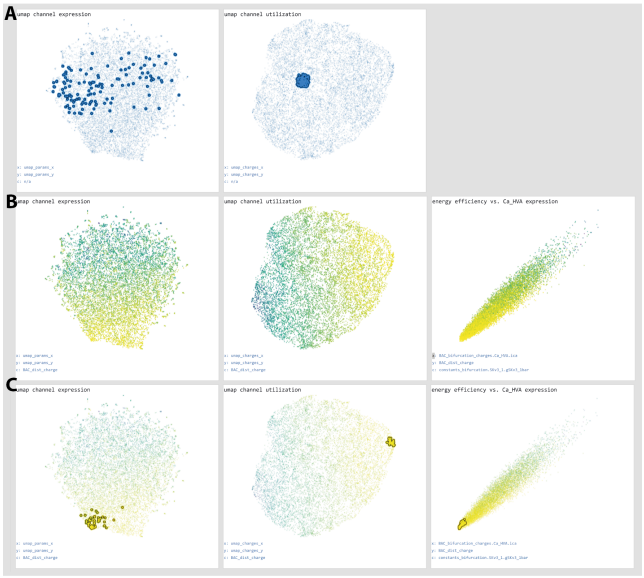

Figure 4: ***A-C:*** *Subsequent steps in the analysis of simulation parameters (refer to Sect. 6.2 for details) using linked views of UMAP embeddings [MHM20] of ion channel expressions (left column), ion channel utilizations (center column), and two correlated features (right column). The coloring encodes the energy efficiency of dendritic computations where bright yellow colors indicate high energy efficiency.*

suggested to create 2D embeddings using UMAP [MHM20] for the channel expression parameters (Fig. 4, left) and seven channel utilization features (Fig. 4, middle) that describe ion channel utilization near the bifurcation zone of the dendrite (the region selected in Fig. 3A, B). Additionally, we selected these features in a parallel coordinates view, which we link to the 2D embeddings of both feature sets for brushing and linking. To test if there is a close correspondence between simulation parameters and functional features **(T1)**, we lasso-selected a small region in the channel utilization space. We found this to result in a wide spread in the UMAP embedding [MHM20] of channel expression parameters (Fig. 5A). Similarly, lasso-selection of a small region in the channel expression embedding leads to a wide spread in the channel utilization embedding. This indicates that channel expression and channel utilization are generally weakly linked. This weak link was further illustrated by selecting narrow ranges of features in the parallel coordinates view that did not constrain the ranges of other features (**T1**).

We found a notable exception to this: The expression of the calcium channel (feature *.Ca_HVA) in the apical dendrites appeared to restrict the amount of total charge exchanged (feature BAC_dist_charge) during dendritic calcium action potentials (Fig. S4). It also limited the expression of fast, non-inactivating potassium conductances in the apical dendrite (feature *.SKv3_1.gSKv3_1bar). Therefore, we plotted the first two features against each other in a scatter plot and color-coded the third (Fig. 4, right). Indeed, we found a close correlation between the total charge exchanged and the expression of Ca_HVA (**T2**). Additionally, the energy-efficient models also had a low expression of Kv3.1. Thus, in contrast to the initial result indicating a weak link between functional and structural features, we predicted a tighter relationship for energy-efficient models (**T1,T2**). We, therefore, color-coded the UMAP [MHM20] embedding by our energy metric and found that indeed both the embeddings of the structural and the functional features are organized by this energy efficiency feature (Fig. 4B). We then repeated the lasso selection, this time specifically selecting those models that were the most energy-efficient. Contrary to the first result, selecting energy-efficient models in the channel utilization embedding led to a well-constrained selection of models in the channel expression embedding (Fig. 4C). This indicates a tight link between channel expression and utilization for this subset of models (**T1, T2**).

To compare energy-efficient and inefficient samples, we examined the spatially and time-resolved neuronal activity of such models on the dendritic morphology (**T3**). We specify in the interaction grammar that the simulation panel workspace layout is opened (Fig. 5) when single samples are selected in the scatter plot. Comparing the time-resolved membrane potential of energy efficient and energy inefficient models at different dendrite locations (Fig. 5C) revealed significant differences in the shape of the apical action potential decay (Fig. 5D, blue line), which can guide further investigation and definition of more descriptive scalar features (**T3**).

7. Discussion & Conclusion

Our grammar-based framework for rapid prototyping of coordinated views seamlessly integrates heterogeneous types of web-based views from different visualization frameworks and libraries. A key design goal of our study was to link regular information views with 3D (anatomical) views (**G1**) using a grammar-based specification of the interaction behavior. Previously, implementing the interaction between views in this particular scenario would have required complex event and callback structures in the code. Our approach eliminates this error-prone programming task by providing the interaction grammar as an additional layer of abstraction.

The views used in the presented case studies (Sects. 6.1 and 6.2) consist of information views based on Vega-Lite [SMWH17], an interactive parallel coordinate view from Plotly.js [Plo] for analysis of high-dimensional simulation data [BO22], WebGL-enabled scatter plots supporting display of large numbers of points [Lek23], and 3D anatomical views based on Babylon.js [bab]. However, our framework is not limited to the aforementioned visualization libraries and view types. To integrate a new view from another library, developers need to implement a thin wrapper class that connects the attributes and events of the new view with the messaging interfaces of our framework (see Sect. 5.3, Fig. 1C). This facilitates the reuse of code and the integration of the best available visualization modules from different libraries for the respective tasks (**G2**).

Our work follows a current trend of using JSON-based visualization grammars [McN22]. A key advantage of such grammar-based specifications is that they allow for rapid prototyping of visual analysis tools (**G3**). We fully leverage this advantage by providing a

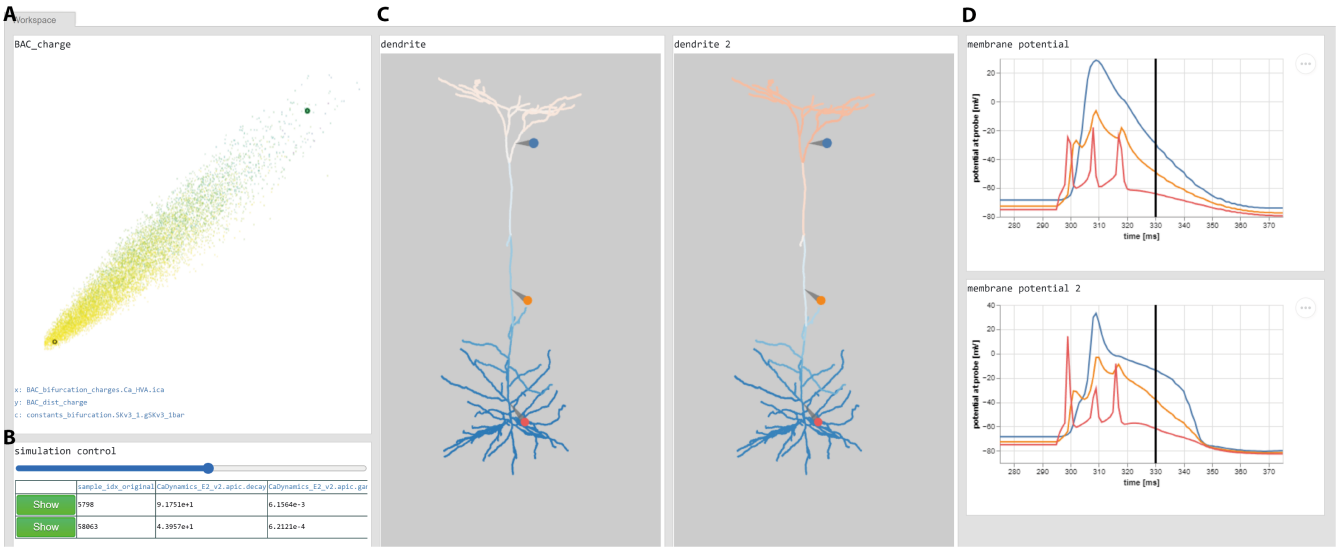

Figure 5: *Configuration of the workspace for comparing biophysical simulations.* ***A:*** *Two selected parameter samples (associated with high and low energy efficiency for dendritic computation).* ***B:*** *Simulation control panel (slider changes time step).* ***C:*** *Simulated membrane potential on the dendrites for both parameter samples; virtual probes (colored pins) have been placed on the dendrite to record the membrane potential at specific locations.* ***D:*** *Time-resolved membrane potentials at the probe sites. The probe near the bifurcation zone (blue) shows a difference between both parameter samples in the decay of the membrane potential.*

grammar editor (Fig. 1E) that allows changing the view configurations and interaction behavior live in the application.

The specification of our grammar (Listing 1) is agnostic to specific ontologies from the application domain (**G4**). This makes it usable for all conceivable use cases where coordinated views (even from different libraries) are to be used. Furthermore, no adaptations are necessary if the application-related semantics change. The JSON-based specifications are stored in a file on the backend, allowing to version-control the respective workspaces.

Our grammar allows us to specify dynamic layout changes of the whole application workspace (**G5**), enabling, e.g., complex analysis workflows from overview to detail [Shn96] that require different arrangements of views. In the second case study (Sect. 6.2), e.g., the tool was used to analyze neuroanatomy jointly with simulated neural activity. This demonstrates how the dynamic grammar-based layout of views and propagation of user selections facilitate a multi-staged filter process to explore the high-dimensional space of simulation parameters in a drill-down manner. Hence, our tool supports aggregating, projecting, and filtering data [SGS18]. However, support for non-expert users in specifying the respective views and interactions still needs to be improved.

The decision to use a web-based tool enabled the rapid development of the interaction grammar and the accompanying tool. On the other hand, web-based techniques still face a problem when handling large amounts of data. For example, it is not possible to display all 15 million simulation samples from case study 2 (instead, we used a representative subset of 10,000 simulation runs). However, we believe that this limitation will become less severe with further advances in web-based technology. To address these scalability issues from the backend side, we are currently integrating a Python library (vaex [BV18]) for big data handling that allows fast computation of aggregate representations of the data, such as 2D density plots of selected attributes.

Another limitation is that the grammar-based application might not be accessible to all domain experts since it requires some technical knowledge. To lower this burden, it would be desirable to have consistent local specification grammars in the "configuration" attribute of the views. A first step in this direction could be to provide a more generic 3D viewer that can be configured using a syntax that is similar to that of Vega(-Lite) [SWH14, SRHH16, SMWH17].

Future work will also include provenance tracking of the application with options to jump back to previous interaction states. There also exist highly specialized web-based tools for structural connectivity analysis (e.g., Vimo for connectivity motif analysis [TWC*22]) that could be integrated as views into our framework. It would also be desirable to be able to load graph data in addition to tabular datasets. A key aspect for improving the usability of the tool would be to provide debugging help for the JSON specification when it fails during runtime. Additional usability improvements include supporting regular expressions (e.g., "scatterplot*" instead of "scatterplot1" and "scatterplot2") to reference the views in the grammar, which would make the specification more concise.

In summary, we demonstrated in two case studies from neurobiology that the proposed grammar-based approach is feasible and useful in real-world visual analysis scenarios that require the integration of heterogeneous web-based views and iterative refinement of the visual analysis tool together with domain experts. However, since the logic of the proposed interaction grammar is generic (independent of neuroscience specifics), our tool can serve as a tem-

plate for developing visual analysis applications in other domains that require the coordination of information views and 3D views and/or interaction-driven changes to the overall workspace layout.

8. Acknowledgements

This work was supported by Deutsche Forschungsgemeinschaft grant SPP 2041 Computational Connectomics (D. B. and M. O.), the Max Planck Institute for Neurobiology of Behavior – caesar (M. O.), European Research Council grant 101069192 (M. O.), Neuroscience Network North Rhine-Westphalia grant iBehave (M. O.), and by the NSF grants NCS-FO-2124179, and IIS-1901030 (H. P.).

References

[AABS*14] AL-AWAMI A. K., BEYER J., STROBELT H., KASTHURI N., LICHTMAN J. W., PFISTER H., HADWIGER M.: Neurolines: a subway map metaphor for visualizing nanoscale neuronal connectivity. *IEEE Trans. Vis. Comput. Graph. 20*, 12 (2014), 2369–2378. doi:10.1109/TVCG.2014.2346312. 2

[bab] Babylon.js. URL: https://www.babylonjs.com/. 1, 2, 3, 5, 8

[BM13] BREHMER M., MUNZNER T.: A multi-level typology of abstract visualization tasks. *IEEE Trans. Vis. Comput. Graph. 19* (12 2013), 2376–2385. doi:10.1109/TVCG.2013.124. 2

[BO22] BAST A., OBERLAENDER M.: Ion channel distributions in cortical neurons are optimized for energy-efficient active dendritic computations. *bioRxiv* (1 2022), 2021.12.11.472235. doi:10.1101/2021.12.11.472235. 2, 5, 6, 7, 8

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3 data-driven documents. *IEEE Trans. Vis. Comput. Graph. 17* (2011), 2301–2309. doi:10.1109/TVCG.2011.185. 1

[BSG*09] BRUCKNER S., SOLTESZOVA V., GRÖLLER M. E., HLADŮVKA J., BÜHLER K., YU J. Y., DICKSON B. J.: Braingazer - visual queries for neurobiology research. *IEEE Trans. Vis. Comput. Graph.* (2009). doi:10.1109/TVCG.2009.121. 2

[BTB*22] BEYER J., TROIDL J., BOORBOOR S., HADWIGER M., KAUFMAN A., PFISTER H.: A survey of visualization and analysis in high-resolution connectomics. *Comp. Graph. Forum 41*, 3 (2022), 573–607. doi:10.1111/cgf.14574. 2

[BV18] BREDDELS M. A., VELJANOSKI J.: Vaex: big data exploration in the era of gaia. *Astronomy & Astrophysics 618* (10 2018), A13. doi:10.1051/0004-6361/201732493. 9

[CJS*22] CAKMAK E., JACKLE D., SCHRECK T., KEIM D. A., FUCHS J.: Multiscale visualization: A structured literature analysis. *IEEE Trans. Vis. Comput. Graph. 28* (12 2022), 4918–4929. doi:10.1109/TVCG.2021.3109387. 2

[Das] Dash. URL: https://dash.plotly.com/. 2

[DGH03] DOLEISCH H., GASSER M., HAUSER H.: Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the 5th Joint IEEE TCVG – EUROGRAPHICS Symposium on Visualization (VisSym)* (2003), Bonneau G.-P., Hahmann S., Hansen C. D., (Eds.), The Eurographics Association, pp. 239–248. doi:10.2312/VisSym/VisSym03/239-248. 2

[Dol04] DOLEISCH H.: *Visual Analysis of Complex Simulation Data using Multiple Heterogenous Views*. PhD thesis, TU Wien, Vienna, Austria, 2004. URL: https://www.cg.tuwien.ac.at/research/publications/2004/doleisch-thesis/doleisch-thesis-pdf.pdf. 2

[ENG*20] EGGER R., NARAYANAN R. T., GUEST J. M., BAST A., UDVARY D., MESSORE L. F., DAS S., DE KOCK C. P., OBERLAENDER M.: Cortical output is gated by horizontally projecting neurons in the deep layers. *Neuron 105* (1 2020), 122–137.e8. doi:10.1016/j.neuron.2019.10.011. 6

[FZB16] FORNITO A., ZALESKY A., BULLMORE E. T.: *Fundamentals of Brain Network Analysis*. Elsevier Academic Press, 2016. doi:10.1016/C2012-0-06036-X. 2

[GBNO21] GUEST J. M., BAST A., NARAYANAN R. T., OBERLAENDER M.: Thalamus gates active dendritic computations in cortex during sensory processing. *bioRxiv* (2021). doi:10.1101/2021.10.21.465325. 7

[GCM*19] GLEESON P., CANTARELLI M., MARIN B., ET AL.: Open source brain: A collaborative resource for visualizing, analyzing, simulating, and developing standardized models of neurons and circuits. *Neuron 103* (8 2019), 395–411.e5. doi:10.1016/j.neuron.2019.05.019. 2

[HVU*22] HARTH P., VOHRA S., UDVARY D., OBERLAENDER M., HEGE H.-C., BAUM D.: A stratification matrix viewer for analysis of neural network data. In *Eurographics Workshop on Visual Computing for Biology and Medicine* (2022), Raidou R. G., Sommer B. et al., (Eds.), The Eurographics Association. doi:10.2312/vcbm.20221194. 2

[JSS*20] JONSSON D., STENETEG P., SUNDEN E., ENGLUND R., KOTTRAVEL S., FALK M., YNNERMAN A., HOTZ I., ROPINSKI T.: Inviwo — a visualization system with usage abstraction levels. *IEEE Trans. Vis. Comput. Graph. 26* (11 2020), 3241–3254. doi:10.1109/TVCG.2019.2920639. 1

[KD18] KORNFELD J., DENK W.: Progress and remaining challenges in high-throughput volume electron microscopy. *Curr. Opin. Neurobiol. 50* (June 2018), 261–267. 1

[KH13] KEHRER J., HAUSER H.: Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Trans. Vis. Comput. Graph. 19*, 3 (2013), 495–513. doi:10.1109/TVCG.2012.110. 2

[LBI*12] LAM H., BERTINI E., ISENBERG P., PLAISANT C., EMPIRICAL S. C., CARPENDALE H. L. E. B. P. I. C. P. S.: Empirical studies in information visualization: Seven scenarios. *IEEE Trans. Vis. Comput. Graph. 18* (2012), 1520–1536. doi:10.1109/TVCG.2011.279. 5

[Lek23] LEKSCHAS F.: Regl-scatterplot: A scalable interactive javascript-based scatter plot library. *J. Open Source Softw. 8*, 84 (4 2023), 5275. doi:10.21105/joss.05275. 5, 8

[LWLG22] LYI S., WANG Q., LEKSCHAS F., GEHLENBORG N.: Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Trans. Vis. Comput. Graph. 28* (1 2022), 140–150. doi:10.1109/TVCG.2021.3114876. 2

[MAAB*17] MOHAMMED H., AL-AWAMI A. K., BEYER J., CALI C., MAGISTRETTI P., PFISTER H., HADWIGER M.: Abstractocyte: A visual tool for exploring nanoscale astroglial cells. *IEEE Trans. Vis. Comput. Graph. 24*, 1 (2017), 853–861. doi:10.1109/TVCG.2017.2744278. 2

[McN22] MCNUTT A. M.: No grammar to rule them all: A survey of json-style dsls for visualization. *IEEE Trans. Vis. Comput. Graph.* (2022), 1–11. doi:10.1109/TVCG.2022.3209460. 1, 2, 8

[met] MeteorJs: The platform for shipping JavaScript applications. URL: https://www.meteor.com. 5

[MHM20] MCINNES L., HEALY J., MELVILLE J.: Umap: Uniform manifold approximation and projection for dimension reduction, 2020. arXiv:1802.03426. 8

[NDPW*18] NOWKE C., DIAZ-PIER S., WEYERS B., HENTSCHEL B., MORRISON A., KUHLEN T. W., PEYSER A.: Toward rigorous parameterization of underconstrained neural network models through interactive visualization and steering of connectivity generation. *Front. Neuroinform. 12* (2018), 32. doi:10.3389/fninf.2018.00032. 2

[NEJ*15] NARAYANAN R. T., EGGER R., JOHNSON A. S.,

MANSVELDER H. D., SAKMANN B., DE KOCK C. P., OBERLAENDER M.: Beyond columnar organization: Cell type- and target layer-specific principles of horizontal axon projection patterns in rat vibrissal cortex. *Cereb. Cortex 25* (11 2015), 4450–4468. doi:10.1093/cercor/bhv053. 7

[NSA*13] NOWKE C., SCHMIDT M., ALBADA S. J. V., EPPLER J. M., BAKKER R., DIESRNANN M., HENTSCHEL B., KUHLEN T.: VisNEST - interactive analysis of neural activity data. In *IEEE Symposium on Biological Data Visualization, BioVis 2013, Atlanta, GA, USA, October 13-14* (2013), IEEE Computer Society, pp. 65–72. doi:10.1109/BioVis.2013.6664348. 2

[par] ParaView. URL: https://www.paraview.org/. 1

[Plo] Plotly.js. URL: https://plotly.com/javascript/. 1, 2, 5, 8

[RCM*20] REINA G., CHILDS H., MATKOVIĆ K., BÜHLER K., WALDNER M., PUGMIRE D., KOZLÍKOVÁ B., ROPINSKI T., LJUNG P., ITOH T., GRÖLLER E., KRONE M.: The moving target of visualization software for an increasingly complex world. *Comp. Graph. 87* (4 2020), 12–29. doi:10.1016/j.cag.2020.01.005. 1

[rea] React-Grid-Layout. URL: https://github.com/react-grid-layout/react-grid-layout. 4, 5

[Rob07] ROBERTS J. C.: State of the art: Coordinated & multiple views in exploratory visualization. In *Proceedings - Fifth International Conference on Coordinated & Multiple Views in Exploratory Visualization 2007 (CMV 2007), Zürich, Switzerland, 2 July* (2007), Andrienko G., Roberts J. C., Weaver C., (Eds.), IEEE Computer Society, pp. 61–71. doi:10.1109/CMV.2007.20. 2, 3

[SCH*18] SENK J., CARDE C., HAGEN E., KUHLEN T. W., DIESMANN M., WEYERS B.: VIOLA - A multi-purpose and web-based visualization tool for neuronal-network simulation output. *Front. Neuroinform. 12* (11 2018), 75. doi:10.3389/fninf.2018.00075. 2

[Sch21] SCHOLZ D.: A modular domain-specific language for interactive 3d visualization. Diploma Thesis, 2021. doi:10.34726/hss.2021.80484. 2

[SE12] SCHMITT O., EIPERT P.: NeuroVIISAS: Approaching multiscale simulation of the rat connectome. *Neuroinformatics 10* (7 2012), 243–267. doi:10.1007/s12021-012-9141-6. 2

[SGS18] SARIKAYA A., GLEICHER M., SZAFIR D. A.: Design factors for summary visualization in visual analytics. *Comp. Graph. Forum 37* (6 2018), 145–156. doi:10.1111/cgf.13408. 2, 9

[SHB*14] SEDLMAIR M., HEINZL C., BRUCKNER S., PIRINGER H., MOLLER T.: Visual parameter space analysis: A conceptual framework. *IEEE Trans. Vis. Comput. Graph. 20* (12 2014), 2161–2170. doi:10.1109/TVCG.2014.2346321. 2

[Shn96] SHNEIDERMAN B.: The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages* (1996), pp. 336–343. doi:10.1109/VL.1996.545307. 9

[SJES19] SCHWANKE S., JENSSEN J., EIPERT P., SCHMITT O.: Towards Differential Connectomics with NeuroVIISAS. *Neuroinformatics 17* (1 2019), 163–179. doi:10.1007/s12021-018-9389-6. 2

[SMWH17] SATYANARAYAN A., MORITZ D., WONGSUPHASAWAT K., HEER J.: Vega-Lite: A grammar of interactive graphics. *IEEE Trans. Vis. Comput. Graph. 23* (2017), 341–350. doi:10.1109/TVCG.2016.2599030. 1, 2, 3, 5, 8, 9

[SRHH16] SATYANARAYAN A., RUSSELL R., HOFFSWELL J., HEER J.: Reactive Vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Trans. Vis. Comput. Graph. 22* (2016), 659–668. doi:10.1109/TVCG.2015.2467091. 1, 2, 9

[Str] Streamlit. URL: https://streamlit.io/. 2

[SWH05] STALLING D., WESTERHOFF M., HEGE H.-C.: amira: A highly interactive system for visual data analysis. In *Visualization Handbook*, Charles D. Hansen C. R. J., (Ed.). Academic Press / Elsevier, 2005, pp. 749–767. doi:10.1016/b978-012387582-2/50040-x. 1

[SWH14] SATYANARAYAN A., WONGSUPHASAWAT K., HEER J.: Declarative interaction design for data visualization. In *The 27th Annual ACM Symposium on User Interface Software and Technology UIST '14, Honolulu, HI, USA, October 5-8* (2014), Benko H. et al., (Eds.), ACM, pp. 669–678. doi:10.1145/2642918.2647360. 1, 2, 9

[SZS*17] SACHA D., ZHANG L., SEDLMAIR M., LEE J. A., PELTONEN J., WEISKOPF D., NORTH S. C., KEIM D. A.: Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Trans. Vis. Comput. Graph. 23* (1 2017), 241–250. doi:10.1109/TVCG.2016.2598495. 2

[TCG*22] TROIDL J., CALI C., GRÖLLER E., PFISTER H., HADWIGER M., BEYER J.: Barrio: Customizable spatial neighborhood analysis and comparison for nanoscale brain structures. *Comp. Graph. Forum 41* (2022). doi:10.1111/cgf.14532. 2

[TWC*22] TROIDL J., WARCHOL S., CHOI J., MATELSKY J., DHANYSAI N., WANG X., WESTER B., WEI D., LICHTMAN J. W., PFISTER H., BEYER J.: Vimo: Visual analysis of neuronal connectivity motifs. *bioRxiv* (2022), 2022–12. doi:/10.1101/2022.12.09.519772. 2, 9

[UHM*22] UDVARY D., HARTH P., MACKE J. H., HEGE H.-C., DE KOCK C. P. J., SAKMANN B., OBERLAENDER M.: The impact of neuron morphology on cortical network architecture. *Cell Rep. 39* (4 2022). doi:10.1016/j.celrep.2022.110677. 6, 7

[ZPWS22] ZONG J., POLLOCK J., WOOTTON D., SATYANARAYAN A.: Animated Vega-Lite: Unifying animation with a grammar of interactive graphics. *IEEE Trans. Vis. Comput. Graph.* (2022), 1–11. doi:10.1109/TVCG.2022.3209369. 2