Nebula: A Coordinating Grammar of Graphics

Ran Chen, Xinhuan Shu, Jiahui Chen, Di Weng, Junxiu Tang, Siwei Fu, Yingcai Wu

Abstract—In multiple coordinated views (MCVs), visualizations across views update their content in response to users' interactions in other views. Interactive systems provide direct manipulation to create coordination between views, but are restricted to limited types of predefined templates. By contrast, textual specification languages enable flexible coordination but expose technical burden. To bridge the gap, we contribute Nebula, a grammar based on natural language for coordinating visualizations in MCVs. The grammar design is informed by a novel framework based on a systematic review of 176 coordinations from existing theories and applications, which describes coordination by demonstration, i.e., how coordination is performed by users. With the framework, Nebula specification formalizes coordination as a composition of user- and coordination-triggered interactions in origin and destination views, respectively, along with potential data transformation between the interactions. We evaluate Nebula by demonstrating its expressiveness with a gallery of diverse examples and analyzing its usability on cognitive dimensions.

Index Terms—Coordination, multiple coordinated views, interactive visualization, grammar of graphics

1 Introduction

Multiple coordinated views (MCVs), wherein visualizations across views update their content in response to users' interactions in other views, are prevalent in modern visualization systems [1]. An example of MCVs is the scatterplot matrix, wherein users select points in one scatterplot, and other scatterplots will highlight the corresponding points (Fig. 1). By coordinating multiple views and synchronizing visualizations automatically, MCVs facilitate a comprehensive and multifaceted analysis, presenting unforeseen data relations and improving user performance [1].

Despite the widespread use of MCVs, constructing effective and expressive coordination between views remains challenging. Interactive systems, such as Tableau, only provide limited types of coordination based on predefined templates (low expressiveness). Textual specification languages (e.g., D^3 [2], Vega [3], and Vega-Lite [4]) allow users to author flexible coordination, but it can be difficult because these languages lack a proper abstraction of coordination (low usability). Specifically, the difficulties are manifested in two aspects. First, these languages model coordination as a complex composition of basic visualization elements (e.g., marks, transforms, encodings) and interaction configurations (e.g., initialization, events) rather than explicit interaction responses (e.g., selecting, highlighting). Such a poor *closeness of mapping* [5] may increase the *gulf of execution* [6] in the construction of coordination. Second, users usually coordinate visualizations from different libraries to construct various MCVs. However, the incompatible mechanisms of libraries exacerbate the difficulty of coordinating visualizations. For example, users must write excessive code to synchronize the *selection* of Vega-Lite with the *d3-event* of D^3 and establish coordination between the two libraries.

The gap between the expressiveness and usability in coordinating visualizations motivates us to develop Nebula, a grammar for the rapid construction of flexible coordination based on a novel demonstration-driven coordination framework. Our approach is threefold. First, we capture and analyze the current views of coordination through an extensive review of the existing theories and applications to abstract coordination properly. We observe that although researchers model coordination as data flows and state changes between multiple views [7], they incline to describe coordinations *by demonstration*, i.e., how coordination is performed by users. Second, we propose a novel demonstration-driven coordination framework based on the observation. This framework formalizes coordination as a composition of *user-* and *coordination-triggered interactions*, along with the potential *data transformations* between the two types of interactions. Such a framework complements the state-of-the-art research on the concise representation of coordination. Third, we design Nebula, a grammar that allows users to construct coordination based on the framework intuitively with the structured templates written in natural language. For example, the coordination in a scatterplot matrix (Fig. 1) can be constructed in Nebula simply with *"select items in any scatterplot, then highlight items in other scatterplots."* Moreover, the architecture of Nebula decouples coordination specifications from visualization design, thereby improving compatibility with different visualization libraries.

We evaluate the expressiveness and usability of Nebula with two different approaches [8], respectively. For expressiveness, we create a gallery of the diverse examples that reproduce the frequently-occurring coordinations in the reviewed literature. We also demonstrate how the coordination constructed by Nebula could support complicated visual analysis tasks with a real-world application of MCVs. For usability, we conduct a qualitative analysis based on the Cognitive Dimensions of Notation framework [5]. The result indicates that Nebula shows some advantages in most

- *Ran Chen, Jiahui Chen, Di Weng, Junxiu Tang, and Yingcai Wu are with the State Key Lab of CAD&CG, Zhejiang University and Zhejiang Lab, Hangzhou, China. Yingcai Wu is the corresponding author. Email: {chenran928, jhchen6, dweng, tangjunxiu, ycwu}@zju.edu.cn.*
- *Xinhuan Shu is with the Hong Kong University of Science and Technology, Hong Kong, China. Email: xinhuan.shu@connect.ust.hk.*
- *Siwei Fu is with the Zhejiang Lab, Hangzhou, China. Email: fusiwei339@gmail.com.*

Manuscript received Oct 30, 2020; revised Feb 3, 2021.

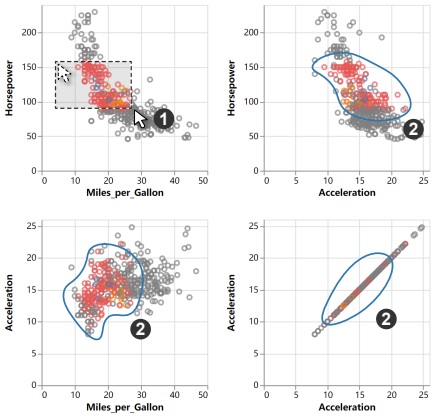


Fig. 1. Top: Example of MCVs showing an interactive scatterplot matrix, where the points selected by users in any scatterplot will highlight the corresponding points in the other scatterplots. Bottom: A Nebula specification which describes the above coordination by demonstration in a natural language sentence.

dimensions, suggesting that it may offer potential usability benefits to users.

2 Related Work

This section presents works regarding foundations of coordination and coordination support in visualization tools.

2.1 Foundations of Coordination

A variety of theories, models, and frameworks have been proposed since the late 1990s to characterize MCVs [9]. Most of these works focus on the design of multiple views [10], [11], [12], [13]. By contrast, we focus on existing works on coordination. North and Shneiderman [1] defined coordination as a link between visualizations and classified coordinations into six categories based on user actions and data relationships at both ends of the link. Following a similar definition, Baldonado et al. [11] generalized user actions as interactions. North et al. [14] extended the line of research and proposed a conceptual model named "Snap-Together Visualization" based on the relational data model. In the model, multiple views are coordinated by *joining* data from different data tables through *primary-* and *foreign-key* actions. North et al. [15] further discussed visualization schemas based on the Snap model and represented MCVs as graphs whose nodes and edges respectively represent visualizations and coordinations. Pattison et al. [16] presented a component-based framework for the implementation of generic visualizations and coordinations in the Model-View-Controller (MVC) pattern. Visualization components in this framework are abstracted into data models, while coordination components are used to observe changes in data and trigger changes of dependent data. Boukhelifa et al. [17] enhanced the framework by shared abstract objects. Weaver [7] combined the shared-object direct coordination mechanism [17] and an indirect coordination mechanism based on an expression-based visual abstraction language to enable precise control of coordination, which visualized the structures of MCVs as dataflow graphs [18]. Koytek et al. [19] focused on brushing and linking coordination and developed a specific design space that divides coordination into the *source*, *link*, and *target* to refer to data points and relationships between views.

To conclude, researchers commonly regard coordination as a dataflow graph in MCVs. However, such a dataflow graph is often invisible and cannot directly reflect the interaction responses in coordination. This work extends the dataflow graph and introduces a novel framework that abstracts coordination by demonstration. In particular, the framework formalizes coordination as a composition of interactions and potential data transformations.

2.2 Coordination Support in Visualization Tools

A multitude of visualization authoring tools has emerged in recent years to assist in constructing MCVs. This section reviews the realization of coordination in these tools to support MCVs, which is classified into two categories: interactive systems and programming-based tools [20].

Grammel et al. [21] and Mei et al. [22] surveyed a variety of interactive systems for authoring visualization in a non-programming environment. Nevertheless, only a small part of these systems support coordinating multiple visualizations. For example, Snap [14] enables users to coordinate views by *joining* data tables of these views based on relational data models and coupling *primary-* and *foreign-key* actions on these data tables. Improvise [7] provides *live properties*, a direct shared-object coordination mechanism, along with an indirect coordination mechanism (coordinated queries), using a visual abstraction language. However, creating coordinations in these systems, which require high learning overhead and verbose operations, can be tedious for users. To address the issue, advanced visual models or paradigms out of programming are applied to help construct coordinations. For example, business intelligence (BI) software, such as Tableau (formerly Polaris [23]), enable users to create common coordinations based on predefined templates. MyBrush [19] supports customizing various brushing and linking coordinations by configuring the details (e.g., visual attributes, temporality) of *source*, *link*, and *target*. Lyra2 [24] allows users to rapidly create interactions in multiple views by demonstration and a set of heuristics. Dataflow systems (e.g., VisFlow [25]) provide dataflow operators that can construct coordination along the dataflow, in which interactions in upstream views could change the visual representation of downstream views. The construction can be further simplified With the assistance of natural language interfaces (e.g., FlowSense [26]). Despite their capability to reduce technical burdens, these methods suffer from limited expressiveness, i.e., only supporting limited types of coordinations.

By contrast, programming-based tools allow users to author custom coordination. Visualization programming toolkits and libraries, such as ProtoVis [27] and D^3 [2], can achieve extremely high expressiveness. However, they also face similar problems to Snap and Improvise, namely, considerable programming expertise and tedious execution codes. To alleviate the issue, declarative grammars grounded in *The Grammar of Graphics* [28] are introduced

to enable rapidly generating visualizations through concise JSON syntax. These grammars mask the underlying execution details and focus on design decision-making by composing visualization elements, thus increasing efficiency and reducing the learning cost. Recently, various declarative grammars have emerged [3], [4], [29], [30], [31]; however, these grammars are not targeted at coordination and lack specific design. For example, Vega-Lite requires users to first configure interaction with its interaction primitive *selection* and then apply *selection* to basic visualization building blocks (e.g., *transform*, *encoding*) in different views to specify coordination. Such a grammar highly couples visualization design and coordination specifications, requiring users to frequently switch context between individual visualization elements and coordination between visualizations, which introduces obstacles to the construction of coordination.

In summary, authoring toolkits balancing expressiveness and usability for creating coordination remain absent. Thus, we propose a novel coordination grammar, Nebula, which allows users to specify coordinations by demonstration in structured natural language templates. The Nebula architecture decouples coordination specifications from visualization design, thus improving compatibility to integrating external visualization libraries.

3 Terminology

To clarify the scope of our research, we introduce the related terms used in our paper and present the model in Fig. 2.

MCVs. MCVs are developed to aid exploratory visualization. MCVs generally comprise a set of views and coordinations between these views [1]. The coordinated views are updated accordingly upon the users' interactions in some of the views. A typical example is the interactive scatterplot matrix (Fig. 1).

View. A view comprises a visualization or a composition of several visualizations, along with potential widgets (e.g., button, slider, and text), to control visual properties [4], such as size, title, and label. For example, each scatterplot in a scatterplot matrix can be regarded as a view, while the scatterplot matrix can also be regarded as a view when cooperating with other visualizations. Moreover, a view can accept interactions and manipulate its visual representations accordingly. In particular, interactions will result in the modification of the backing data and states in the view [32].

Coordination. Coordination creates links among the views. These links propagate the responses of user interactions (i.e., visual changes) from one view to others. Specifically, these links propagate the data and state changes between the views. For example, after a selection interaction is performed by users in one view to specify a subset of data, the coordination may trigger other views to update their subsets accordingly (e.g., highlight the data). *Data transformation*, such as subsetting, merging, filtering, and intersection, is often required to resolve data inconsistencies among the views as the data changes are propagated. Moreover, advanced algorithms, such as clustering and reduction, are needed for remarkably complex data transformation scenarios. Coordination can also be modeled as a dataflow graph [7], where the nodes denote the data and states in the views, while the links represent data transmission and transformation between the views.

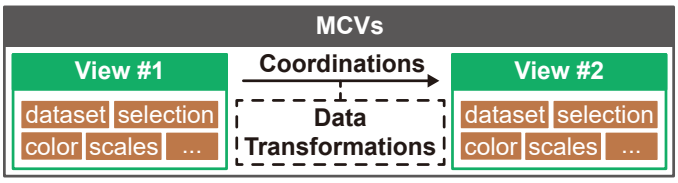


Fig. 2. In traditional models, MCVs comprise views and coordinations. Views control the visual representations by their backing data and states. Coordinations propagate interaction responses by building links to transmit and transform these data and states, which can be modeled as a coordination dataflow graph.

4 Literature Review and Analysis

To abstract the concept of coordination properly, we conduct an extensive literature review and capture the view of coordination in visualization. We choose this methodology because it covers a wide range of real practices of creating coordination. Consequently, the survey collects a corpus of 176 coordination techniques and informs the design of the coordination framework by demonstration.

4.1 Survey on Coordination

Toward a deep and comprehensive understanding of coordination, we surveyed coordination techniques from papers and visualization systems to cover as many scenarios as possible. First, we began by collecting papers on coordination theories and surveys. Next, we expanded the collection by studying commercial visualization systems and academic visualization authoring tools, which typically implement a wide range of coordinations to facilitate the construction of MCVs. Finally, to enrich the diversity, we selected a collection of design study papers involving different data types (e.g., high-dimensional, time-series, geospatial, and hierarchical) published in the past decade, along with the corresponding visual analytics systems that integrate rich coordination applications for various visualization tasks.

We then manually extracted coordinations from these papers and systems for the subsequent analysis. Specifically, we first captured each coordination by reading these papers or documents, watching their videos, and using the online demos. We then recorded authors' descriptions for these coordinations, because these descriptions are carefully crafted and explained to the audience, thus preserving authors' intent and intuitively presenting coordinations with a rich trove of details, e.g., purposes, tasks, and realizations.

Consequently, the survey involves 45 papers, 1 commercial visualization system, 6 visualization tools, and 31 visual analytics systems, arriving at a corpus of 176 coordination techniques. All these materials are available at https://nebula-vis.github.io/survey.

4.2 Data Analysis

To gauge an appropriate abstraction level, we first observe the description of coordinations by visualization researchers in the survey and the reasonable abstraction of these descriptions. Importantly, we find researchers tend to describe the coordination *by demonstration*, i.e., depicting the users' interactions in some views and the corresponding updates

of other views. For example, Satyanarayan et al. [4] described coordinations in Vega-Lite using the sentences, such as *"selections in one view can also be used to filter input data to another view"* and *"panning/zooming the bound interval selection in the first view also updates the second view"*. In addition, although not directly characterizing the coordination dataflow graphs, these descriptions maintain the rationality of these dataflow graphs. Therefore, based on the analysis of the authors' descriptions by demonstration, we model the abstraction of *coordination* into the following three parts (Fig. 4(A)):

- ◇ **User-triggered interactions**—interactions that are directly performed by users.
- ◇ **Coordination-triggered interactions**—"virtual" interactions performed in coordinated views that cause content updates in response to user-triggered interactions.
- ◇ **Data transformations**—transformations that manage the potential inconsistency of the backing data between the user- and coordination-triggered interactions.

In the above examples, *"selection in one view"* and *"panning/zooming the first view"* are user-triggered interactions, while *"filter input data to another view"* and *"updates the second view"* are coordination-triggered interactions.

To effectively investigate and summarize the user- and the coordination-triggered interactions, we divide them into seven categories (Fig. 4(B)), which were initially drawn from the interaction taxonomy of Yi et al. [33] and refined in the context of coordination during the survey. The seven categories are described as follows.

- ◇ *Select* specifies some elements of interest (e.g., data items and intervals) by actions such as clicking and brushing.
- ◇ *Filter* alters the exclusion criteria for visual elements.
- ◇ *Navigate* alters the viewpoint and scales of visualization by actions such as panning, zooming, and rotating.
- ◇ *Encode* pertains to changing the fundamental visual representation (e.g., color, size, and shape) of the data.
- ◇ *Reconfigure* changes the spatial arrangement (e.g., sorting, adjusting baselines) of visualizations.
- ◇ *Set* is frequently used in coordination to operate view configuration (e.g., replace the dataset, operate the control widgets, such as slider, and modify the text of title/axis).
- ◇ *Append* appends new data to the visualization without overwriting the old one. Such interaction is especially useful in coordinations related to streaming data.

Compared with the original taxonomy, *Select*, *Filter*, *Encode*, and *Reconfigure* are retained. In addition, *Navigate* is derived by merging *Explore* and *Abstract/Elaborate* due to their similarity in navigating to different data perspectives. Next, *Connect* is removed because it is usually used in multiple views and can be replaced by the composition of other interactions. Finally, two unique interactions in coordination (i.e., *Set* and *Append*) are observed and added to the categories. It is noted that these categories are summarized following the literature survey and may be adjusted to better fit the coordination context in the future. For example, *Append* currently focuses on streaming data and may be extended to general interactive scenarios.

Two of the authors individually coded all the collected coordinations and resolved the disagreements by discussing and refining the coordination abstraction iteratively.

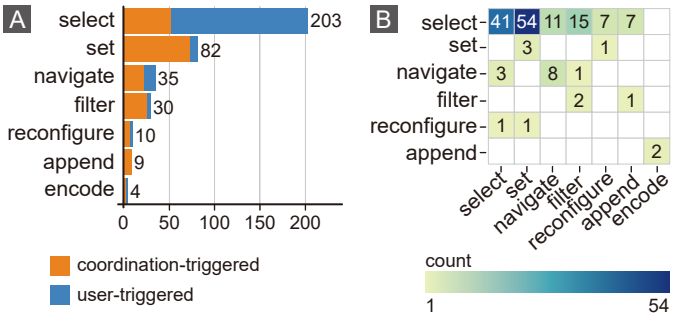


Fig. 3. Statistics of the corpus of 176 coordination techniques. (A) The frequency distribution of interactions in coordination. (B) The frequency distribution of one-to-one interaction compositions in coordination. The y- and x-axes respectively represent the user- and the coordination-triggered sides.

4.3 Findings

Based on our survey, we compute a series of summary statistics to gain an overview of the coordination usage. Fig. 3(A) shows the frequency of each interaction in coordination. The 176 coordinations contain a total of 373 interactions. Among these interactions, *Select* is the most frequently used interaction technique in coordination (203/373). Another commonly used interaction is *Set* (82/373), while others are less than 10%. The distribution is reasonable, as coordinations are realized based on dataflow graphs while *Select* and *Set* provide direct means to manipulate data. In addition, *Select* is mainly used on the user-triggered side (151/203), while the others mainly occur on the coordination-triggered side. After investigating the examples, we find *Select* is mostly used by users for searching a data subset of interest, followed by the further exploration on the subset in other views with various interactions.

Furthermore, we study the composition of these interactions considering pairing the user- and coordination-triggered interactions. In total, 26 composition types are identified from the 176 coordinations, including 16 one-to-one (one user- and another coordination-triggered interaction) and 10 complex (at least two interactions on either user- or coordination-triggered side) types. Specifically, 16 one-to-one types appear 158 times (89.8%), while 10 complex types only appeared 18 times (10.2%). As expected, simple compositions are more common in coordination, while complex compositions may be designed for specific scenarios. Fig. 3(B) presents the frequency distribution of these one-to-one compositions, where *Select* $\rightarrow$ *Set* (54/158) and *Select* $\rightarrow$ *Select* (41/158) are the top two popular compositions, followed by *Select* $\rightarrow$ *Filter* (15/158), *Select* $\rightarrow$ *Navigate* (11/158), *Navigate* $\rightarrow$ *Navigate* (8/158), *Select* $\rightarrow$ *Reconfigure* (7/158), and *Select* $\rightarrow$ *Append* (7/158). This distribution is consistent with the statistics on individual interactions, where *Select* and *Set* are dominant in coordination.

Then, we examine the backing data transformations between the pairs of user- and coordination-triggered interactions. Although data transformations may not be directly elaborated, we infer the implementation from the descriptions of coordination and the experience of authoring visualization. For example, a coordination in a scatterplot matrix aims to highlight the intersection of points selected in two

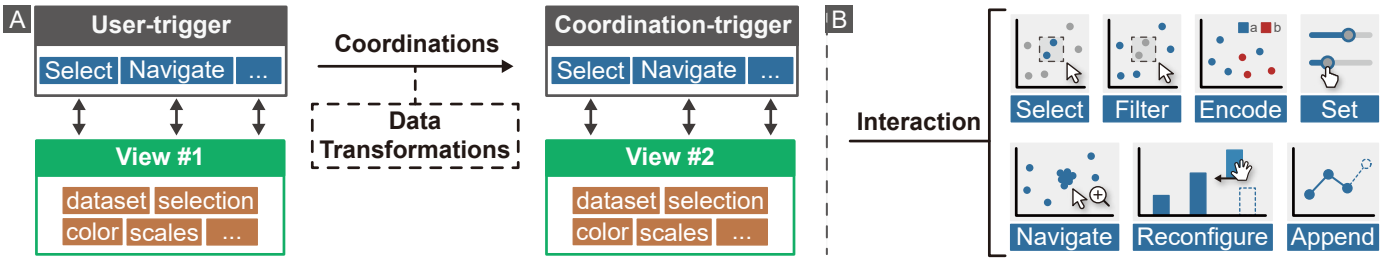


Fig. 4. Overview of our novel coordination framework. (A) The coordination is abstracted by demonstration and formally modeled as a composition of user- and coordination-triggered interactions above the views in the traditional model, along with potential data transformations. (B) Interactions are further divided into seven categories and control the visual representations of views based on the data and states.

views (Fig. 5(D). Realizing the coordination requires a data transformation that *computes the intersection* of the two sets of points selected by their respective views. Consequently, we find that 82 out of the 176 coordinations involve at least one data transformation (46.6%). These data transformations are diverse for various purposes (e.g., mining features of a dataset, joining two data tables, and maintaining the consistency between two data structures).

Finally, we investigate the coordination structures, i.e., the composition of these interactions and data transformations. We found that most of these coordinations are asymmetric (163/176). That is, the coordination arising from the user-triggered interactions in view A to the coordination-triggered interactions in another view B is different from that from view B to A if it exists. The result is consistent with most multiple-view systems, wherein different views carry out different tasks, and the coordination links these views based on a specific workflow of the system. Meanwhile, we found a rare but special coordination structure from the remaining 13 coordinations, which are symmetric. Typical examples include scatterplot matrix and small multiples, in which multiple views present the same type of visualizations. The symmetric structure motivates a method to rapidly specify coordination between views.

5 Coordination Framework

Informed by our survey and the analysis in Section 4.2, we propose a novel demonstration-driven coordination framework illustrated in Fig. 4(A). The framework abstracts coordination *by demonstration*, namely, how coordination is performed by users. Thus, the coordination in MCVs can be formalized as a composition of user- and coordination-triggered interactions, along with potential data transformations. To maintain the rationality of the underlying structure, the abstraction is based on the coordination dataflow graph [7], which encapsulates the data and states of views into interactions (node) and manages data transmission and transformation between interactions (link).

Based on the summary statistics of the corpus in Section 4.3, we first elaborate two basic components in coordination, i.e., seven categories of interactions (Fig. 4(B)) and data transformations in Section 5.1. We then discuss the composition of these components in Section 5.2. Finally, we generalize the coordination structures in terms of various compositions of interactions and data transformations in Section 5.3.

5.1 Coordination Components

5.1.1 Interactions in Coordination

To construct the coordination dataflow graph, interactions are usually associated with the backing data and states of views. Thus, interacting with views will modify these data and states. Coordination connects these data and states to propagate interaction responses between views. These data and states are called *targets* of interactions, which are elaborated following the taxonomy of interactions in Section 4.2 (Fig. 4(B)).

Select. The targets of *Select* are the selected elements, which include two types: items (Fig. 6(A)) and intervals (Fig. 6(C)). The selected items refer to single or multiple data records in views (e.g., points in scatterplots, nodes and links in graphs) and are usually stored in arrays. The selected intervals are similar to the intervals of Vega-Lite [4], which specify a range of data (e.g., brushing a period of time or lassoing a geospatial area).

Filter. Unlike *Select*, the targets of *Filter* are the elements that users intend to discard (Fig. 6(B)) but possess the same types: items and intervals.

Navigate. *Navigate* alters the viewpoint and the scales of visualization. The targets can be one-dimensional or multi-dimensional scales. For example, scrolling a document up and down is a one-dimensional *Navigate* interaction because it only alters the page number. Typical two-dimensional *Navigate* interactions include panning and zooming in maps (Fig. 7(B)). Both interactions modify the scales of the x and the y dimensions of the map, which are usually stored in two variables (e.g., intervals of the longitude and latitude, positions of the upper-left and the lower-right points, the position of the focal points, and the zoom level).

Encode. The targets of *Encode* indicate the visual channels (e.g., position, color, and size). *Encode* is usually performed to modify visual channels or their properties via control widgets. For example, users can interact with a control slider whose value encodes the size of the points to change the size (Fig. 8). These targets are usually stored as a value of an object.

Reconfigure. The targets of *Reconfigure* describe the spatial arrangement of data in the canvas. For example, LineUp [34], a multi-attribute ranking visualization tailored from stacked bar charts, enables users to adjust the column (attribute) order by dragging and dropping the column headers (Fig. 9). Such interactions modify the spatial positions of attributes to reconfigure the visualization.

Set. The targets of *Set* are view configurations, which are usually modified through control widgets (e.g., select, input, and text). For example, users can select to import new data and change the parameters of algorithms via the select and input widgets (Fig. 10).

Append. *Append* focuses on streaming data and updates the visualization with new data based on the existing encoding mechanism. The target of *Append* is the newly-generated data to be added to the visualization (Fig. 10).

5.1.2 Data Transformations

Data transformation is required in coordination in the presence of inconsistency between the backing data of user- and coordination-triggered interactions in views. The inconsistency may originate from the following various reasons: different interactions (e.g., *Select* and *Navigate*), types of views (e.g., *Select* in scatterplots and node-link graphs), or underlying implementations (e.g., *Select* in D^3 [2] and Vega-Lite [4]) at both ends of links in the coordination dataflow graph. Moreover, the inconsistency can be caused by visualizing different datasets in these views. For example, two views may visualize two related data tables (e.g., information of employees and their companies) and are coordinated in an MCV system for association analysis. Moreover, users can specify data in one view and visualize their patterns in another view after applying mining algorithms, thus requiring the data transformation between the two views for extracting features. To support these scenarios, we leverage data transformations to connect interactions. Specifically, these transformations handle the inconsistency between interactions and transmit data from the user-trigger interactions to the coordinated views; examples include mapping the selected points in scatterplots to the nodes in node-link graphs, formatting different data structures, joining two related data tables, and leveraging mining algorithms.

5.2 Component Compositions

The basic compositions comprise one user-triggered interaction in one view and one coordination-triggered interaction in another view, along with potential data transformations between the two interactions. We denote this one-to-one type of composition as $interaction \rightarrow (transformation) \rightarrow interaction$. Therefore, the universal set covering all compositions can be a 7×7 matrix. Since our goal is to reveal the mechanism of the component composition and inform the understanding of coordination, we discuss the representative one-to-one compositions with high frequency in the survey (Fig. 3(B)). The discussion is organized from the interactions triggered by users.

Select $\rightarrow *$**.** *Select* is the most widely used user-triggered interaction, which allows specifying a subset of data as a step toward further analysis. In our survey, *Select* can trigger six categories of interactions, except *Encode* in the coordinated views. Specifically, the selection can be used to highlight the related data (*Select* $\rightarrow$ *Select* in Fig. 1) or filter the unrelated data (*Select* $\rightarrow$ *Filter* in Fig. 6(B)) in another view. These coordinations may involve data transformation to handle data transmission. For example, visualizing two datasets in two views may require a *join* operation on the two datasets to find associations. Moreover, the selected data can be set as the input dataset in another view (*Select* $\rightarrow$ *Set* in Fig. 6(A)). This process can involve mining algorithms in data transformation to discover features of selection. For example, users select data in one view which are then computed through a clustering algorithm, and the clustered results are set as the input data in coordinated views for further analysis. Another common coordination is *Select* $\rightarrow$ *Navigate*, in which the selection is used to guide the navigation. A typical scenario is hierarchical browsing, wherein overview provides a global map, and selection in overview will navigate to the corresponding area for details (Fig. 7(A)). This composition usually requires a data transformation to extract the navigation information from the selection. One exception is that the target of *Select* is interval, which may directly serve as the input of *Navigate* (Fig. 6(C)). *Select* $\rightarrow$ *Reconfigure* is usually used to rearrange the data in a view based on the selection in another view through a specific data transformation to extract the spatial arrangement from the selection. An example is the user-guided matrix reordering [35], where the selected data in the scatterplot will be arranged in adjacent orders in the matrix based on a reordering algorithm. *Select* $\rightarrow$ *Append* appends the new selection instead of overwriting the old for further exploration and comparison, which is used in the decision-making of alternatives [36].

Set $\rightarrow *$**.** Since *Set* is mainly used to configure views, it rarely triggers coordination. In our survey, *Set* commonly works with itself (*Set* $\rightarrow$ *Set*) to update the input dataset in the coordinated view in correspondence to the dataset set by users in the origin view. Another case is *Set* $\rightarrow$ *Reconfigure*. For example, a bar chart visualizes attribute weights of a data table, where changing the length of bars modifies the attribute weights and thus sorts the table [37].

Navigate $\rightarrow *$**.** The target of *Navigate* is the scales of visualizations, which means that *Navigate*-triggered coordinations mainly focus on the scales or internal data in coordinated views. For the former, one common example is *Navigate* $\rightarrow$ *Navigate*, where the scales are shared in coordinated views to maintain synchronicity with the view in which users pan and zoom (Fig. 7(B)). This composition is commonly used in multifaceted analysis, which visualizes the same dataset in multiple views, such as the scatterplot matrix and small multiples. For the latter, one case is *Navigate* $\rightarrow$ *Select*/*Filter*, where the data in the navigated area in a view will be highlighted/filtered in another view (Fig. 7(C)). This composition is widely used in the bottom-up analysis. For example, in a large network, the nodes and links in the navigated area showing the details of its structure are highlighted/filtered in the overview heatmap [38].

Filter $\rightarrow *$**.** *Filter*-triggered coordinations are usually used to discard uninteresting elements in multiple views to maintain their consistency, or discover patterns of these elements. For the former scenario, the filtered elements trigger the corresponding filtering of the corresponding elements in other views (*Filter* $\rightarrow$ *Filter*). The latter one involves the *Filter* $\rightarrow$ *Append* coordination, where the elements filtered by users are continuously appended to a view [39] to reveal users' implicit preference.

Encode $\rightarrow *$**.** *Encode* focuses on visual channels. Therefore, it is mainly composed with itself (*Encode* $\rightarrow$ *Encode*) to keep multiple views consistent in the visual encoding (e.g.,

the same point size of scatterplots in Fig. 8).

Reconfigure $\rightarrow *$**.** *Reconfigure* specifies the spatial arrangements of interest, which can be used in the subsequent analysis to discover valuable information masked behind them. For example, reordering usually contains implied user preference, which can be applied to infer and highlight the data of interest (*Reconfigure* $\rightarrow$ *Select*) [35] and compute and visualize the weights of dimensions (*Reconfigure* $\rightarrow$ *Set*) [37] supported by data transformations. Moreover, despite the absence of examples of *Reconfigure* $\rightarrow$ *Reconfigure* in our survey, the coordination can be useful to keep consistency in spatial arrangements between multiple views (e.g., the attribute order between stacked bar chart and LineUp [34] in Fig. 9).

Append $\rightarrow *$**.** The survey does not find instances that use *Append* as the user-triggered interaction. However, we assume that *Append* can at least trigger itself (*Append* $\rightarrow$ *Append*) to input streaming data in multiple views.

5.3 Coordination Structures

Based on the basic components and their compositions, we elaborate two types of coordination structures found in our literature review: asymmetric and symmetric structures.

Asymmetric Structure is a unidirectional coordination structure, in which all links between interactions and data transformations are unidirectional. This structure is a superset of one-to-one interaction composition and is extended to a larger scope including one-to-many, many-to-one, and many-to-many structures. The one-to-many structure can be defined as $interaction \rightarrow (transformation) \rightarrow interactions$, which means that a user-triggered interaction can cause multiple coordination-triggered interactions. A typical example is that an interaction performed in the overview view of a visual analytics system can elicit multiple coordination-triggered interactions in detailed views and cause updates simultaneously. The many-to-one structure can be defined as $interactions \rightarrow transformation \rightarrow interaction$, which means users should perform multiple interactions to cause a coordination-triggered interaction. The process must involve data transformations that receive multiple inputs from user-triggered interactions and transform into one output to the coordinated view (e.g., Fig. 10). The many-to-many structure is defined as $interactions \rightarrow transformation \rightarrow interactions$, which can be a composition of one-to-many and many-to-one structures. In addition, all these aforementioned unidirectional asymmetric structures can be further nested and composed to generate complex structures.

Symmetric Structure is a special directed complete graph structure in coordination. All interactions are identical in this structure, and each pair of interactions is connected by a bidirectional link. Thus, this structure can be regarded as symmetric because interactions can trigger or be triggered by others. Such coordination is widely used in the scatterplot matrix and small multiples that visualize one dataset in multiple views in the same visualization type and allow comparison and discovery of associations between data dimensions or subsets. For example, the scatterplot matrix in Fig. 1 encodes each pair of dimensions in x and y channels. The selecting and highlighting coordination are symmetric, in which selecting points in any scatterplot can highlight the corresponding selections in other scatterplots.

TABLE 1
Synonym list for interactions in Nebula.

Type	Synonyms
select	highlight
navigate	pan, zoom, scroll
reconfigure	rearrange, arrange, organize, sort, align
set	modify, change, replace
append	add

6 NEBULA

In this section, we introduce Nebula, a coordinating grammar with a generalized MCV architecture. The Nebula grammar is motivated by two goals: (1) *expressiveness*: to enable expressive specification of various coordinations and (2) *usability*: to do so with concise and accessible primitives to facilitate efficient coordination specification. To achieve the goals, Nebula is grounded in the demonstration-driven coordination framework and provides a natural language grammar based on structured templates. Moreover, the Nebula architecture reduces the technical burden to integrate external toolkits; thus users can coordinate visualizations from different toolkits using the Nebula grammar. Fig. 5 illustrates the grammar, architecture, and workflow.

6.1 Grammar

A Nebula specification is a natural language (NL) sentence based on structured templates, which describe coordination by demonstration. Informed by our coordination framework, the specification is a three-tuple:

$$coordination := (origin, transformation, destination)$$

The *origin* and *destination* components describe the user- and coordination-triggered interactions, respectively, while the *transformation* component describes the potential data transformations between *origin* and *destination*. We explain each component in detail.

6.1.1 Origin

The *origin* component specifies the users' interactions in the view and is defined as a three-tuple structured template:

$$\langle origin \rangle \rightarrow \langle type \rangle \, \langle target \rangle \, in \, \langle view \rangle$$

The *type* field identifies the interaction type from the seven interaction categories, and the *target* field specifies the target manipulated by the interaction (Section 5.1.1). The *view* field defines which view is interacted with. Examples of these specifications include *"select items in scatterplot1"* (Fig. 6(A)) and *"navigate scales in map"* (Fig. 7(B)).

To support the flexible creation of coordination, Nebula provides synonyms for users to specify the interaction *type*. Table 1 shows a synonym list that Nebula supports for each interaction category. For example, using *Pan*, *Zoom*, or *Scroll* to replace *Navigate* further specifies the interaction representations and presents an intuitive statement to users. These synonyms are summarized based on common colloquial descriptions of coordination in our literature review. It is noted

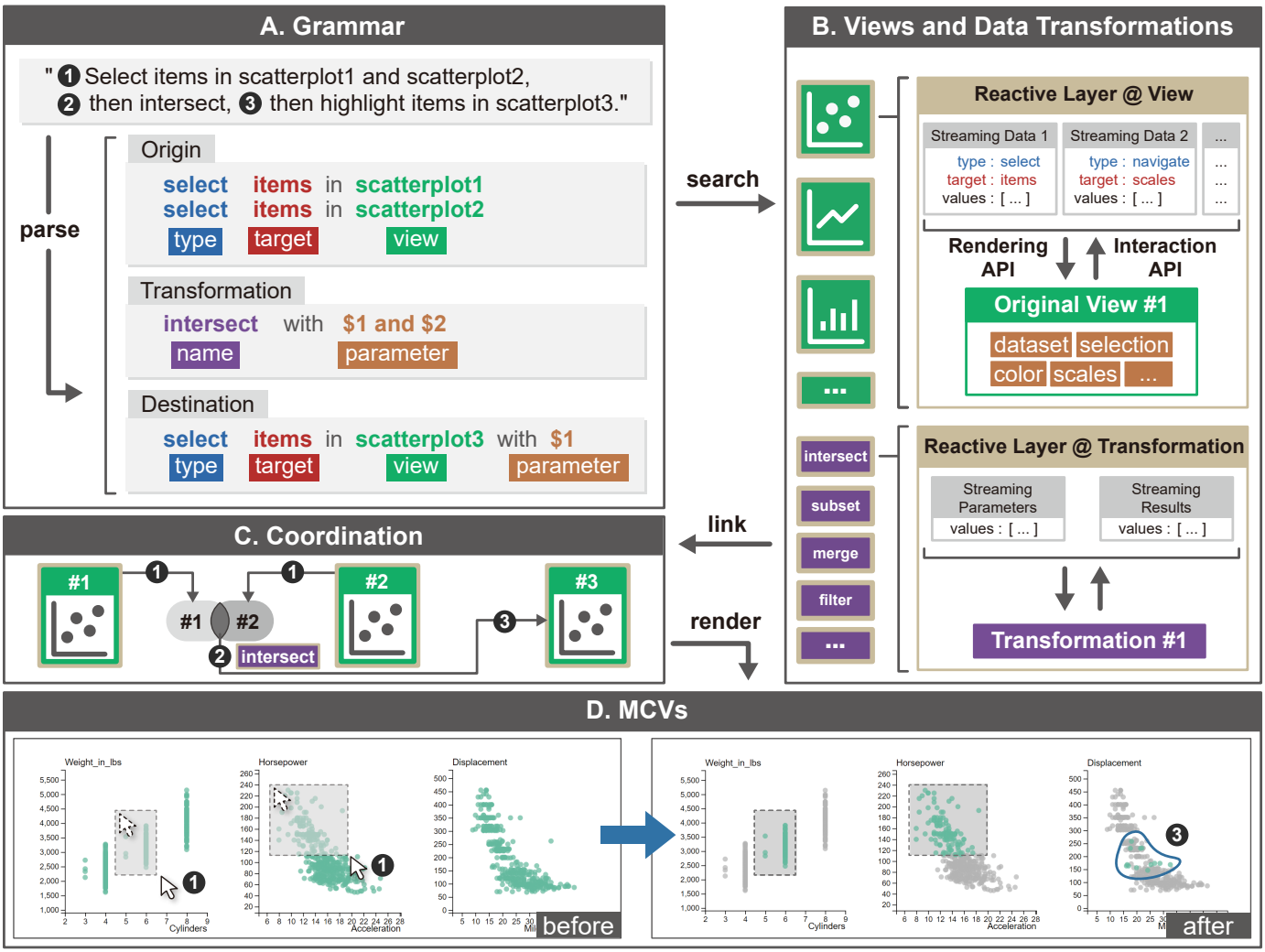


Fig. 5. Nebula architecture and workflow. (A) Original Nebula specifications are segmented and parsed, where tokens are labeled in color, missing fields are completed, and synonyms are replaced. (B) Views and data transformations are searched, and streaming data are located according to the tokens in the specifications. (C) Streaming data are linked between views and data transformations to construct coordinations. (D) MCVs are finally rendered and can receive users' interactions.

that these synonyms are only for an intuitive manner to specify coordination and have no substantial difference. All these synonyms will be parsed into the original interaction category at runtime, which also means that extending the list is easy and inexpensive.

6.1.2 Transformation

The *transformation* specification consists of a three-tuple structured template to handle the condition if data from *origin* and *destination* discord:

$$\langle transform \rangle \rightarrow when\ \langle trigger \rangle,\ \langle name \rangle\ with\ \langle parameter \rangle$$

The *trigger* field is optional and defines a conditional statement when the data transformation would be executed (e.g., click a button). The *name* field registers the name of the data transformation in Nebula, such as intersection and k-means. The *parameters* field specifies the data sources of the transformation based on an ordered assignment rule, which are mainly from pre-loaded datasets, literal constants, and interactions in *origin*. For example, in Fig. 5(A), *"intersect with $1 and $2"* calls a built-in data transformation *intersect(set1, set2)* with the parameters *set1* = *$1* and *set2* = *$2*. Particularly, Nebula incorporates the *"$"* keyword to map the data from *origin* interactions via their order number. The idea is borrowed from Shell, a concise scripting language in Unix/Linux systems that codes parameters with default rules. Therefore, in this example, *"$1"* corresponds to the first interaction in *origin*, namely *"select items in scatterplot1"*. Similarly, *"$2"* denotes the second interaction, namely *"select items in scatterplot2"*. Moreover, to alleviate the technical burden and produce a readable sentence, users need not specify *parameters* in Nebula (Fig. 5(A-2)), and the Nebula parser will auto-complete the parameter specification (e.g., *"$1 and $2"* in the above example).

6.1.3 Destination

The *destination* component describes the interactions triggered by coordination and is defined in a four-tuple form:

$$\langle destination \rangle \rightarrow \langle type \rangle\ \langle target \rangle\ in\ \langle view \rangle\ with\ \langle parameter \rangle$$

The *type*, *target*, and *view* fields are the same as those in *origin*. The *parameter* field is similar to *transformation*, which specifies the parameter to trigger this interaction. Particularly, users can use *"$"* plus order number to specify the results of the previous component in parameters. The rules are that if data transformation exists, then *"$"* corresponds to the output of the data transformation; otherwise, it maps the data from *origin* interactions. For example, in Fig. 5(A), *"$1"* in *destination* refers to the output of the data transformation *intersect*, which means that the intersected items will be highlighted in scatterplot3. For convenience, *"$1"* is set as the parameter by default.

6.1.4 Walkthrough

To demonstrate the cooperation of the three components (i.e., origin, transformation, and destination), Fig. 5 presents an example of a walkthrough to the entire process from the grammar specification to the coordination workflow. In this example, the points in scatterplot3 will be highlighted if they fall within both selections of scatterplot1 and scatterplot2. First, to compose a complete coordination specification, Nebula provides the keyword "then" to connect the three components (Fig. 5(A)). Then, Nebula segments and parses this specification. Each token is labeled in color, missing fields are auto-completed, and synonyms are replaced following Table 1. Specifically, the specification *"select items in scatterplot1 and scatterplot2"* in *origin* is parsed as two individual interaction specifications. The parameters of the data transformation *intersect* are completed following the default rule. The parameter in *destination* is also auto-completed, while the synonym *Highlight* is replaced by *Select*. As a result, an asymmetric two-to-one coordination structure with a data transformation is constructed, which will be rendered in subsequent steps (Fig. 5(B–D)).

In addition, to support rapid creation of symmetric coordination (Section 5.3), Nebula provides pronouns *any* and *other* as syntactic sugar referring to a set of views. For example, in Fig. 1, *any* refers to any scatterplot in the scatterplot matrix, while *other* refers to the remaining scatterplots. Such coordination specification allows users to select points in any scatterplot, and the corresponding points in other scatterplots will be highlighted.

6.2 Architecture

To coordinate the visualizations from external toolkits, Nebula combines the *Model-View-ViewModel* (MVVM) software architectural pattern and the streaming dataflow architecture [3], decoupling coordination construction from interactive visualization design with *reactive layers*.

A reactive layer (Fig. 5(B)) serves as a tailored view model connecting a view or a data transformation with the *streaming data* (as the models in MVVM), which can be communicated with other views and data transformations. In the reactive layer that encapsulates a view, each interaction is modeled as a streaming datum identified by its type chosen from seven interaction categories and its target data field. When an interaction is received, the value of the corresponding streaming datum will be modified. Moreover, changing the values of a streaming datum is equivalent to triggering the corresponding interaction and updating the encapsulated view. For example, streaming data 1 in Fig. 5(B) represents the *Select* interaction. The items selected by users will be stored in the value of streaming data 1. Conversely, modifying the value of streaming data 1 will highlight the corresponding items in view as if they were selected. Such architecture allows the integration of Nebula with other libraries easily by reusing their application programming interfaces (APIs). For example, `view.addDataListener` in Vega-Lite can be used to modify the streaming data upon interactions, and `view.data` and `view.runAsync` help update the views upon changes in the streaming data. Reusing the existing APIs facilitates efficient adaption and provides compatibility with the views from external toolkits. Similarly, the parameters and results of data transformations can also be modeled as streaming data to establish reactivity.

Thereafter, coordinations can be described as the streaming dataflow graphs constructed by parsing Nebula specifications and connecting the streaming data in the reactive layers of different views and data transformations (Fig. 5(C)). The links between the streaming data propagate data mutations, maintaining the synchronization of interaction responses between the views to establish coordination. Moreover, these links enable data transformations to directly process interaction data and control view rendering.

Nebula users can conveniently create MCVs based on such architecture. First, users can author visualizations with different libraries (e.g., D^3, Vega, and Vega-Lite) and arrange these visualizations in a desired layout. Next, the reactive layers that encapsulate the visualizations and data transformations are created conveniently with the APIs of Nebula. For the data transformations that are not programmed in JavaScript, Nebula provides a browser/server API to allow these data transformations to receive and send streaming data, including parameters and results, asynchronously based on HTTP. Finally, users can specify coordinations with the Nebula grammar and implement MCVs based on the encapsulated visualizations and data transformations.

7 Proof-of-concept implementation

Nebula is an open-source toolkit implemented in JavaScript and is available at https://nebula-vis.github.io/. The toolkit comprises three major parts: parser, integrator, and generator. The parser implements Fig. 5(A) and can parse the Nebula specifications. The integrator implements Fig. 5(B) and provides APIs to encapsulate reactive layers for views and data transformations, leveraging their existing APIs. Finally, the generator links streaming data in reactive layers and renders MCVs (Fig. 5(C) and (D)) based on the results of the parser and the integrator.

8 Evaluation

Nebula aims to balance expressiveness and usability to coordinate visualizations. This section demonstrates the realization of the goal. To evaluate expressiveness, we showcase a variety of examples that cover our coordination framework. To assess usability, we analyze Nebula using the Cognitive Dimensions of Notation framework [5].

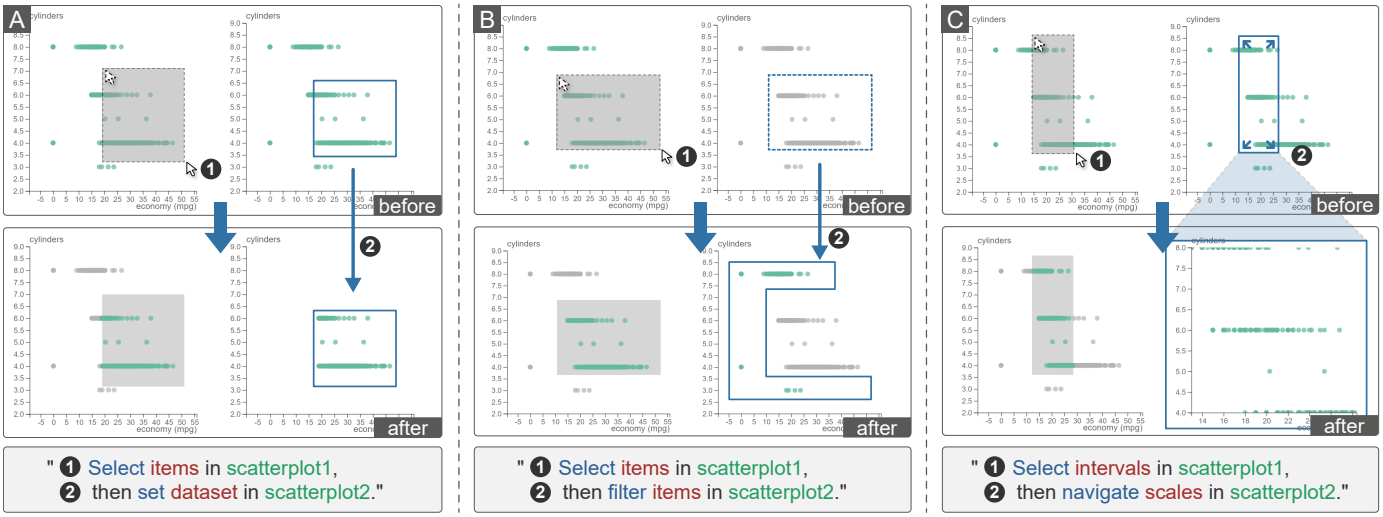


Fig. 6. Examples of *Select* → ∗ coordinations in two scatterplots, where the selection of the first scatterplot will (A) modify the dataset, (B) be filtered out from the dataset, and (C) cause the navigation to the area in the second scatterplot.

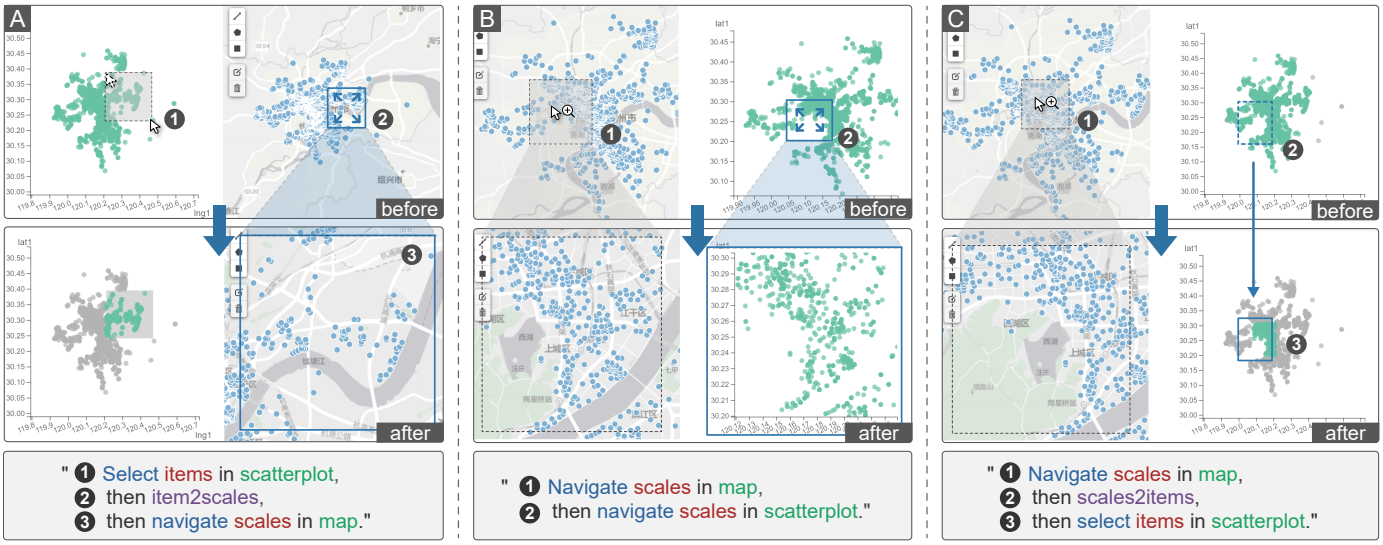


Fig. 7. Examples of *Navigate*-related coordinations in a scatterplot and a map. (A) An example of *Select* → *Navigate* coordination, where the map will be navigated to the area according to the selected points in the scatterplot. (B) An example of *Navigate* → *Navigate* coordination, where panning and zooming in the map will cause the corresponding panning and zooming in the scatterplot. (C) An example of *Navigate* → *Select* coordination, where the points in the scatterplot will be highlighted if they fall in the area navigated in map.

8.1 Examples

To demonstrate the expressiveness of Nebula, we present a selected gallery of common and typical coordination examples based on our literature review.

Figs. 6–10 cover all seven categories of interactions in Nebula. Specifically, Fig. 6 presents the composition of *Select* with different interactions (*Set*, *Filter*, and *Navigate*) to handle various coordinating tasks. Fig. 7 illustrates a set of *Navigated*-related coordinations. Figs. 8 and 9 showcase examples with *Encode* and *Reconfigure*, respectively. Moreover, Fig. 10 presents a many-to-one coordination structure, where the data of four *Set* interactions serve as the parameters of a data transformation, and the results of the data transformation are *appended* to a list view for further comparison and analysis. Fig. 1 shows a symmetric coordination example.

To demonstrate how multiple coordinations cooperate to handle complicated visual analytics tasks, we reproduce the core module of SRVis [40], a real-world MCVs for multidimensional geospatial data (Fig. 11). We select SRVis for its complexity (most of the coordinations involve at least three visualizations) and familiarity with the authors of this paper (three authors participated in its design and development). The prototype comprises six visualizations: a map showing the geospatial details, a LineUp [34] ranking multidimensional data, and four histograms around the map showing the multidimensional statistical distribution along the latitude and longitude. We specify five coordinations to mine patterns between geospatial and multidimensional features. Specifically, the points of interest selected in the map are fed to the datasets of the four histograms to discover their distributions and also highlighted in the LineUp to show their details of dimensions (Fig. 11(A)). In addition, the x scales of the map, encoding the longitude of the data,

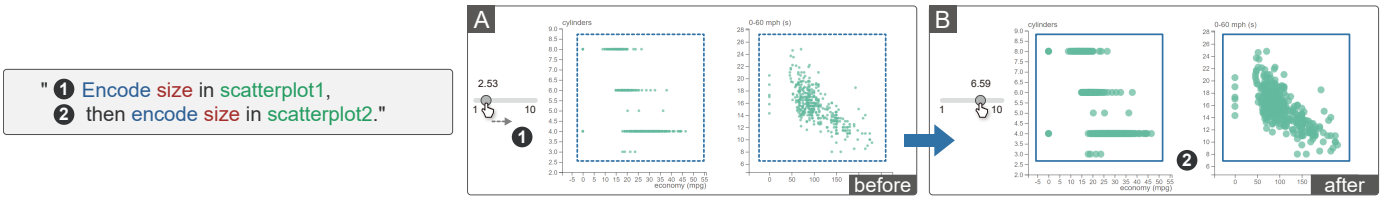


Fig. 8. Example of *Encode* → *Encode* coordination, where modifying the value of the slider will change the point size in both scatterplots.

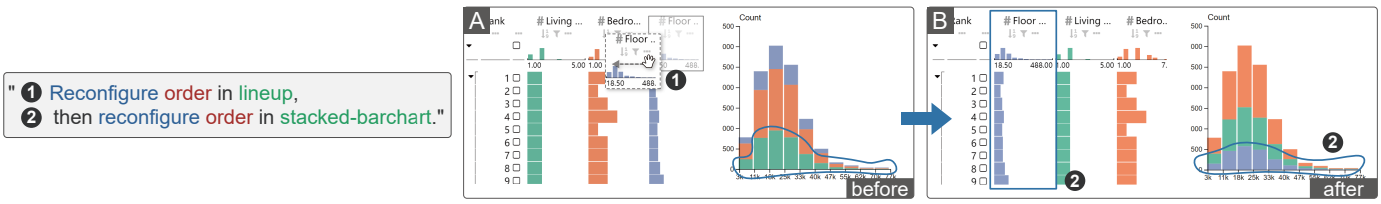


Fig. 9. Example of *Reconfigure* → *Reconfigure* coordination, where modifying the attribute order of the dataset in LineUp [34] by dragging and dropping will also modify the stacking order of attributes in the stacked bar chart.

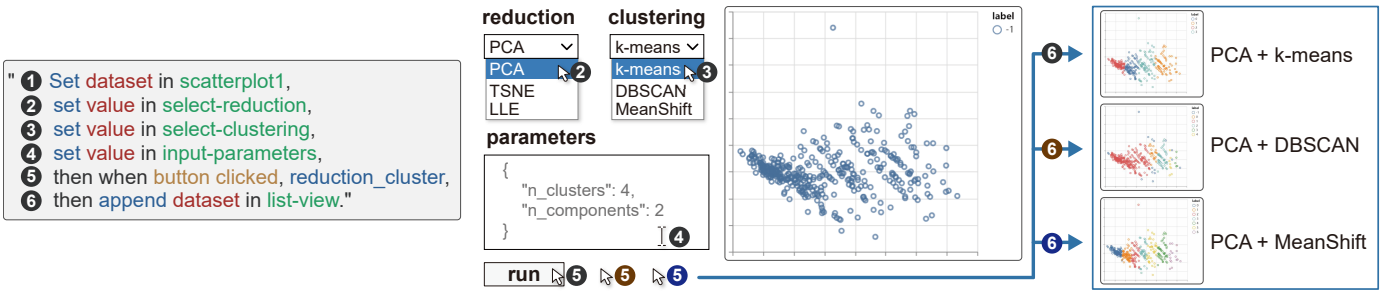


Fig. 10. Example of *Set* → *Append* coordination, where the dataset visualized in the scatterplot will be clustered using different reduction and clustering algorithms and various parameters to compare their performances. The results are appended in a list view, where each result is visualized in a scatterplot. A button is set as the trigger to control the execution of the coordination.

are bound up with the top and the bottom histograms (Fig. 11(B)), similar for the y scales (latitude) of the map with the left and the right histograms (Fig. 11(C)). Moreover, the visible points in this navigated area serve as the dataset of the LineUp to present their details (Fig. 11(D)). Therefore, navigating the viewpoint in the map will update all other views and synchronize their data accordingly (Fig. 11(B), (C), and (D)). Finally, to support a consistent analysis on data dimensions, reconfiguring the orders in the LineUp will update those in the stacked histograms (Fig. 11(E)).

8.2 Cognitive Dimensions of Notation

Based on our survey and framework, we design Nebula to integrate coordination notations and specifications, considering users' mental model. To assess the design intuition and usability of Nebula, we conduct an analysis using the Cognitive Dimensions of Notation framework [5], which is a widely adopted method in the visualization community to evaluate toolkits and systems, such as Protovis [27], Vega [32], PGoG [30], and Lyra 2 [24]. The framework provides a heuristic inspection method with 14 cognitive dimensions to assess the effectiveness of notation systems (e.g., programming languages and visual interfaces) from different perspectives. These dimensions describe the generic properties of notation by names, such as *closeness of mapping (closeness of representation to domain)* and *hidden dependencies (the visibility of relationships)*, inspiring designers to consider the nature of notation from the perspective of cognitive psychology without evaluating the entire system.

Below, we briefly discuss the performance of Nebula using a relevant subset (9/14) of the above cognitive dimensions [5]. Due to the lack of visualization tools dedicated to coordination, we do not directly compare Nebula with other tools. Instead, we elaborate the distinctions between the traditional mechanism of existing tools for constructing coordination (i.e., configuring interactions and composing visualization building blocks) and the mechanism of Nebula (i.e., composing seven categories of interactions). The rest of the dimensions is beyond the goal of Nebula and will be considered in future work.

Closeness of mapping (closeness of representation to domain). Nebula is motivated by the need for a close mapping between coordination and its specification. We analyze how authors describe the coordination and design the notations correspondingly. As such, the design allows users to easily construct coordination in a similar way they describe coordination. Using this grammar, users can focus on the coordination design, including choosing interactions or adjusting data transformations, rather than getting lost in tedious configurations of basic visualization and interaction details.

Hidden dependencies (important links between entities are not visible). Nebula provides the *parameter* fields in *transformation* and *destination* specifications that explicitly reveal

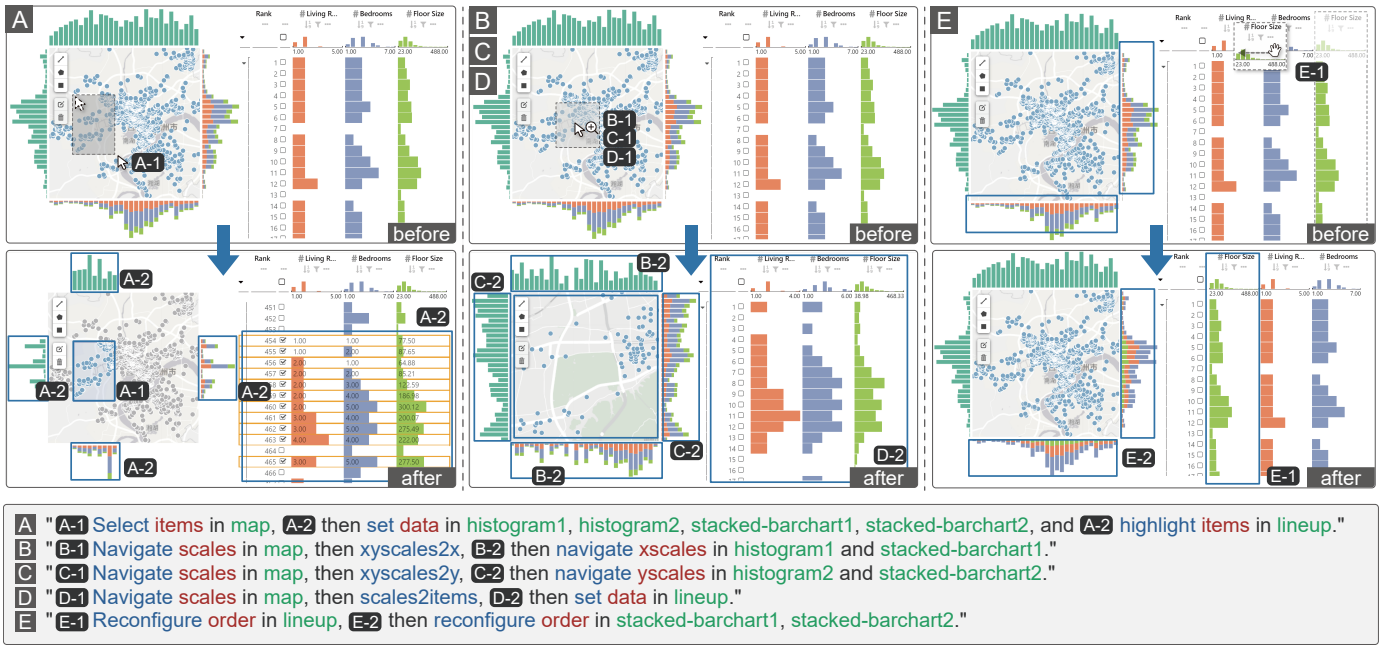


Fig. 11. Reproduction of the core module of SRVis [40] with six visualizations, i.e., a map, four histograms, and a LineUp [34]. The core module consists of five coordinations. Specifically, (A) selecting data of interest in the map will modify the dataset of the four histograms to present the statistical distributions of the selection along the longitude and latitude. The selection is also highlighted in the LineUp to show the details of dimensions. Panning and zooming in the map will modify (B) the scales of x (longitude) in the top and bottom histograms and (C) the scales of y (latitude) in the left and right histograms with a data transformation to transform 2D scales to 1D. (D) When panning and zooming, the visible data in the map are extracted by a data transformation and fed into the LineUp to maintain consistency of their datasets. (E) Rearranging the order of dimensions in the LineUp will update that in the stacked histograms to maintain consistency of their orders.

the dependencies between interactions and data transformations. By contrast, the existing approach to composing visualization and interaction configurations does not model coordination as a first-class primitive, thus requiring users to manually search through other primitives (e.g., *encoding*, *selection*, and *transform*) to build such dependencies.

Viscosity (resistance to change) and consistency (similar semantics are expressed in similar syntactic forms). The *viscosity* indicates the amount of efforts required to change a coordination to another. Nebula avoids high *viscosity* via structured templates, which enable the independent changes in interactions and data transformations to construct different coordinations. These templates also shorten the edit distance of coordination variations by reusing the semantics of similar coordination, thus facilitating consistency.

Role-expressiveness (the purpose of a component is readily inferred) and visibility (ability to view components easily). Nebula provides concise and consistent structured templates to enable precise identification of each component of interaction and data transformation specifications. Moreover, uniform interaction types assist users to infer interaction semantics with a synonym list. Finally, all these components are collected and aggregated in a natural language sentence to facilitate the viewing.

Hard mental operations (high demand on cognitive resources). Nebula may require less mental effort to adapt to their interaction primitives, which directly surface interaction semantics and goals. Data transformation is an exception. Specifying data transformation in Nebula is as hard a mental operation as that in traditional coordination mechanisms. A simplified method is desired in the future.

Diffuseness (verbosity of language). Nebula is concise due to its simple primitives and concise templates. In addition, the Nebula architecture provides a unified layer to reuse the existing APIs of the toolkits to reduce the technical burden.

Abstraction (types and availability of abstraction mechanism). The abstraction of Nebula is discussed in Section 4.2, and the coordination is abstracted by demonstration. Specifically, the basic interaction configurations and visualization blocks are abstracted into seven categories of interactions. The dependencies between these configurations and blocks are abstracted into the interaction compositions. Such abstraction is familiar to visualization practitioners and lower the threshold to formalize coordinations.

In summary, Nebula performs well in most of these dimensions. Note, however, the usability still needs to be assessed with users in future work. The analysis also indicates future directions to improve Nebula.

9 Discussion

Despite the expressiveness of Nebula for authoring coordination as demonstrated in the evaluation results, the design process faces many trade-offs. We discuss the benefits and limitations of Nebula and present our reflections on developing coordination tools.

Grammar. The grammar design of Nebula based on natural language (NL) structured templates is admittedly modest. The design still requires users' prior knowledge and cannot compete with free-form NL sentences in usability. Nevertheless, we highly recognize the strength of structured templates in systematic enumeration and exploration. Finally, we decide to focus on coordination variation

exploration and intentionally fix the Nebula grammar to interpretable structured templates. A potential improvement is to strike a balance between free-form grammar and systematic enumeration.

This paper mainly aims to contribute an overall approach to abstract the coordination dataflow graph and formalize coordination at a high level. Nebula inspect the coordination from the perspective of demonstration, based on our observation to users' general description of coordination. Apart from demonstration, we believe there exist other abstractions for coordination, which can be potential research directions to coordinate multiple views.

Data transformations are essential components in coordinations to resolve data inconsistencies, find data relationships, and extract data patterns. However, directly accessing and evaluating the details of data transformations is impossible because the existing papers and systems in our survey provide limited discussion on their internal data transformations. Therefore, Nebula provides a set of common data transformations and allows users to manually integrate their customized transformations based on the APIs. Moreover, Nebula does not consider the latency in data transformations, and human involvement in the transformations is limited to configuring execution parameters. Therefore, we hope to further investigate and improve data transformations in coordination in the future work.

Architecture. The Nebula architecture aims to improve compatibility with various external toolkits. The architecture that decouples visualization design and coordination specifications is promising in terms of facilitating the construction of MCVs by combining Nebula with powerful interactive visualization authoring tools (e.g., Vega-Lite [4], MyBrush [19], Lyra2 [24]). However, the architecture presents high requirements for developers to construct reactive layers for visualizations and may hinder its development in related ecosystems. Specifically, the architecture assumes that users are familiar with the components they want to integrate and requires users to encapsulate the reactive layers manually. An effective approach is to automate this process by checking and inferring the usage of code.

Beyond Nebula. Up to this point, we have positioned Nebula as a tool for coordinating views in MCVs. Apart from this, we hope that Nebula can facilitate the research in MCVs as a coordination paradigm. First, Nebula can help explore the design space of coordination. By surfacing coordination variations between visualizations, users can compare and choose among different alternatives. In addition, the formal paradigm enables users to cluster a large number of coordinations. The results may promote the development of advanced design guidelines, recommendation models, or other coordination practices. Moreover, Nebula can be applied to not only coordination but also multi-view design in MCVs. For instance, *Encode*-related coordinations can be used to keep visual encoding consistent between views [13].

10 Conclusion and Future Work

This study introduces a novel demonstration-driven framework of coordination and instantiates the framework in a natural-language-based grammar named Nebula. Through a diverse example gallery and an analysis on cognitive dimensions, we show that Nebula enables expressive specifications of coordinations with potential usability benefits.

Nebula is only the first attempt to efficiently author coordinations in MCVs, and we will turn to several potential directions. First, future work should collect and explore feedback from real-world users and usages. The feedback can provide insights and guide the improvement of Nebula, as well as related ecosystems. Besides, enriching coordination types in Nebula is another next step. Finally, integrating Nebula and flexible visualization authoring tools is also a promising direction. It can help users efficiently construct MCVs by combining individual visualization design and coordination between them.

Acknowledgments

We thank all the reviewers for their constructive suggestions and comments. The work was supported by NSFC (62072400), NSFC (62002331), Zhejiang Provincial Natural Science Foundation (LR18F020001), and the 100 Talents Program of Zhejiang University. The work was also partially funded by Zhejiang Lab.

References

[1] C. North and B. Shneiderman, "A Taxonomy of Multiple Window Coordinations," University of Maryland, Department of Computer Science, Tech. Rep., 1997.

[2] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[3] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer, "Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 659–668, 2016.

[4] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer, "Vega-Lite: A Grammar of Interactive Graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 341–350, 2017.

[5] A. F. Blackwell, C. Britton, A. L. Cox, T. R. G. Green, C. A. Gurr, G. F. Kadoda, M. Kutar, M. Loomes, C. L. Nehaniv, M. Petre, C. Roast, C. Roe, A. Wong, and R. M. Young, "Cognitive Dimensions of Notations: Design Tools for Cognitive Technology," in *Proceedings of International Conference on Cognitive Technology: Instruments of Mind*, 2001, pp. 325–341.

[6] E. L. Hutchins, J. D. Hollan, and D. A. Norman, "Direct Manipulation Interfaces," *Human-Computer Interaction*, vol. 1, no. 4, pp. 311–338, 1985.

[7] C. Weaver, "Building Highly-Coordinated Visualizations in Improvise," in *Proceedings of IEEE Symposium on Information Visualization*, 2004, pp. 159–166.

[8] D. Ren, B. Lee, M. Brehmer, and N. H. Riche, "Reflecting on the Evaluation of Visualization Authoring Systems: Position Paper," in *IEEE Evaluation and Beyond - Methodological Approaches for Visualization*, 2018, pp. 86–92.

[9] J. C. Roberts, "State of the Art: Coordinated & Multiple Views in Exploratory Visualization," in *Proceedings of International Conference on Coordinated and Multiple Views in Exploratory Visualization*, 2007, pp. 61–71.

[10] ——, "Multiple-View and Multiform Visualization," in *Proceedings of Visual Data Exploration and Analysis VII*, 2000, pp. 176–185.

[11] M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky, "Guidelines for Using Multiple Views in Information Visualization," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2000, pp. 110–119.

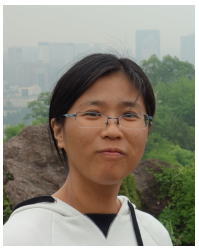
[12] R. Sadana and J. T. Stasko, "Designing Multiple Coordinated Visualizations for Tablets," *Computer Graphics Forum*, vol. 35, no. 3, pp. 261–270, 2016.

[13] Z. Qu and J. Hullman, "Keeping Multiple Views Consistent: Constraints, Validations, and Exceptions in Visualization Authoring," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 468–477, 2018.

[14] C. North and B. Shneiderman, "Snap-Together Visualization: A User Interface for Coodinating Visualizations via Relational Schemata," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2000, pp. 128–135.
[15] C. North, N. Conklin, and V. Saini, "Visualization Schemas for Flexible Information Visualization," in *Proceedings of IEEE Symposium on Information Visualization*, 2002, pp. 15–22.
[16] T. Pattison and M. Phillips, "View Coordination Architecture for Information Visualisation," in *Proceedings of Asia-Pacific Symposium on Information Visualisation*, 2001, pp. 165–169.
[17] N. Boukhelifa, J. C. Roberts, and P. J. Rodgers, "A Coordination Model for Exploratory Multi-View Visualization," in *Proceedings of International Conference on Coordinated and Multiple Views in Exploratory Visualization*, 2003, pp. 76–85.
[18] C. Weaver, "Visualizing Coordination In Situ," in *Proceedings of IEEE Symposium on Information Visualization*, 2005, pp. 165–172.
[19] P. Koytek, C. Perin, J. Vermeulen, E. André, and S. Carpendale, "MyBrush: Brushing and Linking with Personal Agency," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 605–615, 2018.
[20] A. Satyanarayan, B. Lee, D. Ren, J. Heer, J. T. Stasko, J. Thompson, M. Brehmer, and Z. Liu, "Critical Reflections on Visualization Authoring Systems," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 461–471, 2020.
[21] L. Grammel, C. Bennett, M. Tory, and M. Storey, "A Survey of Visualization Construction User Interfaces," in *Proceedings of Eurographics Conference on Visualization - Short Papers*, 2013.
[22] H. Mei, Y. Ma, Y. Wei, and W. Chen, "The design space of construction tools for information visualization: A survey," *Journal of Visual Languages & Computing*, vol. 44, pp. 120–132, 2018.
[23] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 52–65, 2002.
[24] J. Zong, D. Barnwal, R. Neogy, and A. Satyanarayan, "Lyra 2: Designing Interactive Visualizations by Demonstration," *IEEE Transactions on Visualization and Computer Graphics*, 2021.
[25] B. Yu and C. T. Silva, "VisFlow - Web-based Visualization Framework for Tabular Data with a Subset Flow Model," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 251–260, 2017.
[26] ——, "FlowSense: A Natural Language Interface for Visual Data Exploration within a Dataflow System," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1–11, 2020.
[27] M. Bostock and J. Heer, "Protovis: A Graphical Toolkit for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1121–1128, 2009.
[28] L. Wilkinson, *The Grammar of Graphics, Second Edition*. Springer, 2005.
[29] G. Li, M. Tian, Q. Xu, M. J. McGuffin, and X. Yuan, "GoTree: A Grammar of Tree Visualizations," in *Proceedings of ACM Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.
[30] X. Pu and M. Kay, "A Probabilistic Grammar of Graphics," in *Proceedings of ACM Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.
[31] Y. Kim and J. Heer, "Gemini: A Grammar and Recommender System for Animated Transitions in Statistical Graphics," *IEEE Transactions on Visualization and Computer Graphics*, 2021.
[32] A. Satyanarayan, K. Wongsuphasawat, and J. Heer, "Declarative Interaction Design for Data Visualization," in *Proceedings of ACM Symposium on User Interface Software and Technology*, 2014, pp. 669–678.
[33] J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko, "Toward a Deeper Understanding of the Role of Interaction in Information Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1224–1231, 2007.
[34] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit, "LineUp: Visual Analysis of Multi-Attribute Rankings," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2277–2286, 2013.
[35] M. Behrisch, T. Schreck, and H. Pfister, "GUIRO: User-Guided Matrix Reordering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 184–194, 2020.
[36] D. Liu, D. Weng, Y. Li, J. Bao, Y. Zheng, H. Qu, and Y. Wu, "SmartAdP: Visual Analytics of Large-scale Taxi Trajectories for Selecting Billboard Locations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 1–10, 2017.
[37] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert, "Podium: Ranking Data Using Mixed-Initiative Visual Analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 288–297, 2018.
[38] W. Chen, F. Guo, D. Han, J. Pan, X. Nie, J. Xia, and X. Zhang, "Structure-Based Suggestive Exploration: A New Approach for Effective Exploration of Large Networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 555–565, 2019.
[39] C. Xie, W. Zhong, and K. Mueller, "A Visual Analytics Approach for Categorical Joint Distribution Reconstruction from Marginal Projections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 51–60, 2017.
[40] D. Weng, R. Chen, Z. Deng, F. Wu, J. Chen, and Y. Wu, "SRVis: Towards Better Spatial Integration in Ranking Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 459–469, 2019.

Ran Chen is currently a Ph.D. student in the State Key Lab of CAD&CG, Zhejiang University. He received the B.E. degree from Zhejiang University in 2018. His research interests mainly lie in tools and systems to author visualizations.

Xinhuan Shu is currently a Ph.D. candidate in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). She received her B.E. degree in Computer Science and Technology from Zhejiang University, China in 2017. Her research interests include data-driven storytelling, animated visualization, and visual analytics.



Jiahui Chen received her B.Eng. degree from Zhejiang University in 2016. She is now pursuing an M.Eng. in computer science and technology in the State Key Lab of CAD&CG, Zhejiang University. Her research interests include visualization and visual analytics.

Di Weng received his B.S. degree in Computer Science from Taishan Honored College, Shandong University in 2016. He is currently pursuing the doctoral degree with the State Key Lab of CAD&CG, Zhejiang University. His research interests mainly include data mining, visualization, and visual analytics of large-scale urban data. For more information, please visit https://dweng.org.

Junxiu Tang received a B.S. degree in Digital Media Technology from Zhejiang University in 2019, where he is currently pursuing a doctoral degree with the State Key Lab of CAD&CG, Zhejiang University. His research interests mainly include information visualization tools and data-driven storytelling.

Dr. Siwei Fu is an associate research scientist in Zhejiang Lab. His main research interests include visual analytics, intelligent user interface, and natural language interface. He received his Ph.D. degree in Computer Science and Engineering from the Hong Kong University of Science and Technology. For more information, please visit https://fusiwei339.bitbucket.io/

Dr. Yingcai Wu is a Professor at the State Key Lab of CAD&CG, Zhejiang University. His main research interests are information visualization and visual analytics, with focuses on urban computing, sports science, immersive visualization, and social media analysis. He received his Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology. Prior to his current position, Dr. Wu was a postdoctoral researcher in the University of California, Davis from 2010 to 2012, and a researcher in Microsoft Research Asia from 2012 to 2015. For more information, please visit http://www.ycwu.org.