Visual Informatics 4 (2020) 12–23

Contents lists available at ScienceDirect

Visual Informatics

journal homepage: www.elsevier.com/locate/visinf

DataV: Data Visualization on large high-resolution displays

Honghui Mei [a], Huihua Guan [a], Chengye Xin [a], Xiao Wen [a], Wei Chen [b,c,*]

[a] Alibaba Group, China
[b] State Key Lab of CAD&CG, Zhejiang University, China
[c] Joint Institute of Frontier Technologies, Alibaba-Zhejiang University, China

ARTICLE INFO

Article history:
Received 18 March 2020
Received in revised form 30 June 2020
Accepted 17 July 2020
Available online 29 July 2020

MSC:
00-01
99-00

Keywords:
Information visualization
Large high-resolution displays
Interactive design
Software-as-a-service

ABSTRACT

In recent years, the technology and applications of visualizations on large high-resolution displays (LHDs) have received widespread attention because of its perceptual benefits and improved productivity. However, existing work on LHD visualization lacks both comprehensive guidance for design requirements and tools developed for its specific usage scenarios. In this paper, we present the scenarios, design, and implementation of DataV, a Software-as-a-Service (SaaS) visual deployment tool that enables rapid construction and cross-platform publishing of interactive visualization on LHDs. Our framework can support rich components for the high-performance rendering of multi-source heterogeneous data. DataV provides a full-fledged toolchain to help the user efficiently specify layout and interactions. We present its accessibility and impressive visual effects with examples and comparison with Tableau, Power BI, VisComposer, and iVisDesigner. We also report the performance of using DataV for 3D map rendering by comparing it with deck.gl.

1. Introduction

With the increasing use of large high-resolution displays (LHDs), the perceptual and cognitive benefits to visualization and visual analysis on such displays have been widely recognized. A well-designed LHD visualization interface can promote collaborative visual analytics (Bradel et al., 2013), stimulate exploratory behaviors (Reda et al., 2015), and extend perceptual scalability (Yost and North, 2006). However, the design of efficient LHD applications requires various non-trial steps. Though visualization community and commercial developers have provided a variety of choices for visual construction tools (Mei et al., 2018b), there still lacks tools designed explicitly for LHD visualization. In this paper, we analyze the requirements of visualization on LHDs and explore appropriate construction methods to fill this gap.

Driven by the large display size and promoted physical navigation (Ball et al., 2007), LHD visualization shows unique design requirements in many aspects compared to traditional information visualization or business intelligence (BI) dashboards. When designing LHD visualization, designers have to take into account massive factors, including multi-source heterogeneous data, the connection between data and context, as well as the balance between task-driven design and aesthetics. A large number of research papers studying the cognitive process and sensemaking on LHDs have proposed useful design guidelines (Yost and North, 2006; Ball et al., 2007; Bradel et al., 2013; Reda et al., 2015). However, due to the lack of tools that convert such concepts into reliable solutions, it is still difficult for the user to construct efficient LHD visualization in practice. At this point, we believe that such devices have enormous but untapped potential for data fusion, collaborative design, and communicative visualization.

In this paper, we summarize the design guidelines for LHD visualization and present DataV, an efficient tool that produces Software-as-a-Service (SaaS) visualization on LHDs. Our main contribution is a template-based framework, in which we propose a variety of design paradigms that take advantage of LHD applications. The DataV framework can be characterized by the following main features:

- Rich components collected for multi-source heterogeneous data and business-driven representation;
- High-performance rendering designed for maps and 3D views, in which analysis is embedded into spatial context;
- A full-fledged toolchain that enables flexible layout and impressive display for aesthetical pleasure while maintaining accessibility;
- Visualization built with DataV can be easily published as a SaaS application, providing convenient cross-platform access and cross-device visual analysis.

The remaining sections are organized as follows. We first summarize the related literature. We then conduct our design

* Corresponding author.
E-mail addresses: honghui.mhh@alibaba-inc.com (H. Mei), huihua.ghh@alibaba-inc.com (H. Guan), mier.cxy@taobao.com (C. Xin), ninglang.wx@taobao.com (X. Wen), chenwei@cad.zju.edu.cn (W. Chen).

https://doi.org/10.1016/j.visinf.2020.07.001

decisions from prior guidelines for LHD visualization and in-practice usage scenarios. Based on these decisions, we introduce the framework and interface of DataV. To evaluate the effectiveness of DataV paradigms, we report some cases in real-world scenarios. We also discuss the tradeoff of design decisions as well as the limitations of DataV and our future work.

2. Related work

2.1. Visualization on LHDs

Recently, LHDs are becoming increasingly popular and widely deployed for research, data analysis, decision making, presentation, and education. Its commercial success has further promoted the development of display technologies and makes it accessible to a more significant number of users. Compared to typical desktop displays, the increased display area and pixels can improve user engagement and the scalability of visualizations on LHDs.

Many researchers have conducted studies on the cognitive process and sensemaking for visual analysis on LHDs. The larger display size allows audiences to apply physical navigation instead of virtual navigation that must be synchronized (Ball et al., 2007). Thus, different reading paces and analysis strategies can be implemented in collaboration (Bradel et al., 2013). Numerous studies have shown that these displays provide perceptual and cognitive benefits when performing visual analysis tasks by stimulating exploratory behaviors (Reda et al., 2015) and extending perceptual scalability (Yost and North, 2006). Existing studies make suggestions for creating visualizations on LHDs in terms of chart style (Yost and North, 2006), interface layout (Bradel et al., 2013), visual encoding (Bezerianos and Isenberg, 2012), etc.

However, the LHD applications studied in the existing work are quite different from the common LHD visualization in the industry. Compared to tiled single-type charts, small multiples on maps, or arranged documents, which are studied in the experiments, the LHD visualization in actual use consists of more charts with different data and types. These visualization applications are similar to the dashboards commonly created in business intelligence (BI) applications, such as Tableau[1] and PowerBI.[2] Both LHD visualization and dashboards show multiple data sources simultaneously and highlight their associations to support decision-making and real-time monitoring (Sarikaya et al., 2018). These data sources are strongly industry- or domain-specific (Stodder, 2013). However, although they have similar organizational forms and functions, the unique perceptual characteristics make the visualization on LHD follow the design principles that are different from BI dashboards. For example, larger viewing angles require viewers to move their eyes to see different charts. When performing brushing and linking operations on LHD, the highlighted areas are often ignored so that the correct insight cannot be captured (Andrews et al., 2011).

Due to the difference in usage scenarios and properties of display devices, visualization on LHDs "deviates" from conventional visualization principles in many aspects, especially in aesthetic and attention management. In this work, we study the unique design requirements of LHD visualization by combining existing research and practical experience.

[1] Tableau: Business Intelligence and Analytics Software, https://www.tableau.com/.

[2] Power BI: Microsoft Power Platform, https://powerbi.microsoft.com/.

2.2. Visualization construction tools

With the wide application of information visualization, researchers and commercial developers have designed a variety of visualization construction tools to reduce the expertise and workload required to design visualization diagrams (Mei et al., 2018b).

Compared to graphical libraries that require programming skills, or declarative grammars that are easier to learn but still require textual specifications, most users still prefer to use interactive authoring tools (Satyanarayan et al., 2019). Among them, template-based ones developed on chart typology have been widely used because they are the easiest to use. These tools provide the user with control over data, visual mapping, and color encoding through direct manipulation on control panels. Microsoft Excel is the most common example, which allows the user to create charts by choose a chart type and assign data columns. Tableau (formerly Polaris (Stolte et al., 2002)) and Microsoft PowerBI support a more flexible configuration of visual attributes mappings, but are still based on selecting predefined chart types.

A new generation of interactive authoring tools has appeared in recent years of research, and they can provide the more flexible designation of visual effects (Satyanarayan et al., 2019). However, the price of flexibility is higher learning costs and the unique usage paradigm, which has hindered the promotion of these tools. Another idea is to extend the original template-based paradigm to provide more flexible visual design capabilities while maintaining the original usability. VisComposer (Mei et al., 2018a) allows the user to modify the dataflow underlying beneath the charts for more complex data processing and visual binding. Data-Driven Guides (Kim et al., 2017) enable importing SVG graphics as a basis for generating infographics. Saket et al. (Saket et al., 2016) developed an interaction paradigm that can recommend transformations on chart types or visual mappings after the user arranges a few points in a scatterplot.

However, though these tools have made successful attempts in the creation of information visualization charts, they are designed for desktop display devices, and most of them are only suitable for building a single chart. A large number of non-professional users face multiple difficulties in using existing tools to build LHD visualization, including the presentation of multi-source heterogeneous data, flexible layout, and complex deployment environments. Therefore, we developed DataV, which provides a variety of charts that support different types of data, editors that allow flexible specification of layout and interactions, and rapid SaaS publishing.

3. Design philosophy

3.1. Usage scenarios

We collected LHD visualization projects built using DataV, and the practical cases of how users presenting and communicating with these visualizations. From the collected projects and cases, we summarize the following typical usage scenarios, which can reflect the key points of visual design on LHDs.

S1. **Data fusion.** Data scientists like to display a variety of data on the LHDs for discussion. Their tasks include data fusion, data governance, and collaborative iteration of the digital product.
For the presented visualizations, they are more concerned with the source of the data than the visual results. Sometimes they also discuss the data collecting methods and the data transfer interfaces. They often gain new insights based

on their understanding of the data they are familiar with and how it correlates with other displayed data.
For example, during the design iteration process of a data-driven LHD presentation, frequent discussions include "Is the data updated in real-time?", "How should the temperature data be collected and uploaded?", "We must subdivide power usage data into precise regions to support decision-making"..

S2. **Storytelling.** The key of a successful presentation through LHD visualization is to portray stories and show the state behind the numbers.
During our discussions with several designers who are in charge of different LHD projects, we found that they usually first set the story line to be told. Based on the main idea, they gradually add the required data, design the visual effect to display the data, and finally carry out the overall layout.
Taking the LHD presentation that shows an overview of the conference venue as an example, designers start with determining multiple topics to be presented. The topics involved include attendee information, the status of the main venue, the status of the exhibition area, entrance and parking lot monitoring, etc. Then designers consider the story to be told for each topic. For example, the story of attendee information is mainly based on the check-in status of attendees and popularity in different exhibition areas, supplemented by information about WiFi connection, traffic, power consumption and other content. These contents are listed in a table to guide the subsequent data collection and visual design.
Such a story-based visual design helps draw the audience's attention to a variety of data and insights.

S3. **Two-way communication.** Although LHD visualization is often designed for presentation, which is a one-way process, viewers still come up with their own insights.
Such two-way communication with viewers' engagement is very beneficial for both presenters and viewers. This often occurs when the viewers see impressive visual effects or data related to their professional domains.
Among all, the presentation based on geospatial views can best attract the interest of the audiences and stimulate their passion for discussion. The reason is that geographic information is the shared knowledge between the presenter and all audiences. When displaying geographic information, viewers can often connect with their own life and work experience, which leads to heated discussions or stimulates more insights.
For example, after seeing a three-dimensional map showing the terrain, a visitor from a plateau region shared the unique customs of the region she came from, and then triggered discussions about urban planning, transportation, and education.

S4. **Cross-platform deployment.**
Product managers from DataV shared their customers' needs in actual production. We found an important feature that distinguishes LHD visualization from traditional analysis applications; that is, the development and deployment of the productions are on different platforms. Due to considerations of cost and convenience, LHD visualization is often viewed on desktop displays during designing and will only be deployed and tested on actual devices when it is nearly complete. In addition, because many LHDs are used as long-term displays, they need to be updated quickly and seamlessly without restarting when new versions are released.

Another requirement of LHD visualization is the support of cross-device interaction, as presenters often need portable devices (e.g., a tablet) to assist in interaction due to the large physical sizes. For example, an event organizer hopes that the dashboards initially designed to be presented on LHDs can also be accessed on mobile phones, and have the function of reporting and handling emergencies.

3.2. Design guidelines for LHD visualization

Based on the in-practice scenarios and former researches, we have formulated a set of guidelines that promote LHD advantages. We explore the main advantages of LHD visualizations and the requirements of users from actual cases and existing research. In collecting prior research to provide guidance, we focused more on the possible interpretation of the phenomena that appear in the actual scenarios to have a clearer understanding of the principles embodied. In this way, we combine theory with practice to guide the design of our construction tool for visualization on LHDs.

G1. **Organize more information on the display.** On LHD, data from different sources should be presented simultaneously, detailedly, and orderly.
The advantages of large displays are noticeable: more data, more details, and more heterogeneity. The larger display area makes it possible to display more content simultaneously, thereby some of the aggregation and interaction that are commonly used in traditional information visualization are no longer necessary, such as details-on-demand, scrollable or zoomable canvas, and multi-level progressive display. Designers are suggested to spread data and information out on the display (Bradel et al., 2013). In this way, the user can better utilize physical navigation instead of virtual navigation for parallel information acquisition, establish inter-relationships between diverse data sources, and quickly review previously viewed information when needed (S1). This makes LHDs ideal for visualizing multi-source, unstructured, and loosely connected datasets that meet the recent trend toward visual analytics (Heer et al., 2008).
The major issue to be considered is how to arrange a large amount of information on the screen in an efficient organization. On the one hand, different data types need to be presented in different manners. This relies on the support for various data sources and rich selections of chart types. Sometimes the user is required to use certain visualization types that are related to the industry or specific data sources (Stodder, 2013). On the other hand, a reasonable design also contains varied requirements for the layout of different charts on display to support comparison, communication, and cooperation (S2). The spatial organization can be a form of metadata that affects sensemaking. By grouping or forming a timeline, the semantic association between the data can be effectively highlighted, thereby enhancing the cognition of data details (Andrews et al., 2011; Bradel et al., 2013). Conversely, as longer distances affect the recognition and comparison of visual encodings (Bezerianos and Isenberg, 2012), improper layout may impede the analysis of associated data.

G2. **Trap audiences' attention with impressive visual effects.**
A challenge for designing visualization on LHDs is to capture, maintain, and timely divert the audiences' attention according to the analysis process with impressive visual effects.
LHD visualizations can display more data at the same time and eliminate the cost of context switches, thereby providing an immersive visual presentation. Reda et al. show

that large physical size and high resolution can raise the audiences' visual analysis engagement and maintain their attention for a longer period of time (Reda et al., 2015). However, the huge viewing angle makes audiences hard to mention all changes on the large display. For instance, when designing the brushing and linking operation, designers have to avoid the need for the audiences to scan the whole complex visualization when looking for the highlighted parts (Andrews et al., 2011). In addition, because it is often used for reporting and presentation, the design of such visualization takes into account more social concerns, such as privacy and aesthetics (Ni et al., 2006).
These factors make visualization on LHDs more focused on visual impact. Gradient colors, 3D views, and complex animations are rarely used in traditional information visualization but are indispensable in LHD visualization. Impressive visual effects can help capture the audiences' attention, thereby promoting their engagement and directing their analysis to the specific views at the appropriate time (S3).

G3. **Reference to geospatial information.** Embedding data inside geospatially-referenced visualizations is efficient for extending perceptual scalability.

When presenting visualizations on a larger display and with higher resolution, the embedded design is proven to be more efficient than small multiples (Yost and North, 2006). Embedded visualizations use the increased display area and pixels for embedding more data in a single larger view instead of displaying multiple views. This can achieve better scalability because it can present more attributes with the same display size (Andrews et al., 2011).

The most common and important embedded design is based on geospatial information. We focus on geospatial information for the following reasons. First, geography is domain knowledge shared by all designers and audiences. Thus, spatial metaphors can provide the audiences with immersive sensemaking and a higher level of knowledge share (Robinson, 2008). This enables the information embedded in the geospatial display to have the best semantic organization (G1). Second, the applications based on geographic analysis have a large overlap with the use scenarios of large-screen visualization. Successful examples include meteorological data visualization (Mei et al., 2016), urban planning (Chen et al., 2017), network anomaly detection (Zhang et al., 2017), and sales data analysis (Liu et al., 2018). Third, as mentioned above (S3), data displayed in the immersive urban scenes can be related to the life and work experience of most audiences, thereby obtaining better engagement. In addition, geospatial displays based on 3D rendering have the most impactful visual effects (G2).

3.3. Design decisions

With these guidelines, we draw a clear picture of what is essential or not in designing LHD visualization, thus guide reasonable design attempts of our approach. We construct our approach based on the identified key points, following five design decisions.

D1. **Support rich types of data and charts.** LHD visualization often consists of a variety of multi-source heterogeneous data (G1). DataV must be able to handle different data types and data interfaces, as well as present them using suitable visual mappings. In addition, due to different business scenarios, even the same type of data may have different design requirements to better stimulate the analytical engagement of people in industries (G2). To this end, DataV is designed to support different data sources and provides a business-driven visual components library.

D2. **Enable flexible layout.** Traditional BI interfaces usually use a grid-based layout, which cannot fulfill the needs of LHD visualization (G1). DataV is designed to enable flexible layouts that the user can control freely.

D3. **Emphasize geospatial context and connections.** DataV lays emphasis on the display of geospatial scenes such as maps and 3D spaces (G3). In order to better present the connection between data and context, the support for interaction and linked views is also a key point to be considered.

D4. **Provide impressive presentation.** DataV provides the user with rich and well-designed chart templates to create impactful visual effects (G2). This requires a balance between task-driven design and aesthetics. In addition, DataV also provides a high-performance 3D rendering engine to support the construction of geospatial scenes (G3).

D5. **Allow easy-to-use construction and publication with SaaS.** Accessibility is essential for a visualization construction tool. To this end, DataV proposes a template-based paradigm that follows chart typology (Wilkinson, 2006), supplemented by a full-fledged toolchain.

Moreover, to better support cross-platform deployment and cross-device interaction (S4), DataV allows the user to publish the designed visualization as a SaaS application. SaaS publication is also a common practice of BI tools, as it can provide convenient access and transmission through Cloud services.

4. DataV

In this section, we introduce the design details of DataV and our LHD construction paradigms.

4.1. Framework

We first present a brief overview of the DataV framework, which is a template-based construction tool. As illustrated in Fig. 1, DataV consists of four parts, including data importing, visual components, editor toolchain, and application publishing. When using DataV for LHD visual creation, the user first loads the data into the *data center*, and then adds *components* for visual mapping. The added components can be edited in the editor toolchain, which includes three different editors. *View editor* controls the properties of components and their layout. *Blueprint editor* enables the design of dataflow and interaction. *3D editor* allows the user to trim the map and 3D scenes. Finally, DataV provides one-click *SaaS publishing* of created visualization.

4.2. Data center

LHD visualizations are used as platforms for data display and communication, which often integrate multi-source and heterogeneous data (D1). However, the format of unprocessed raw data always cannot match the charts to draw. For unified access, cleaning and transformation of different data sources, DataV has established a data center for each user. In the data center, the user can import and manage their data sources by uploading local files or connecting to online databases. Components in different LHD projects can access these data sources; thus, data can be reused. DataV uses filters (a piece of code written in a JavaScript function) to transform and process data. The raw data can be processed by a series of filters and transformed into the required data format of different components. Among them, some reusable filters can be saved as code snippets and managed uniformly in the data center for future reuse.

DataV supports multiple types of data sources, including databases, files, web APIs, and others. DataV allows the user to connect to these data sources with a single click, and provides the user with familiar data acquisition methods such as SQL queries (D5).

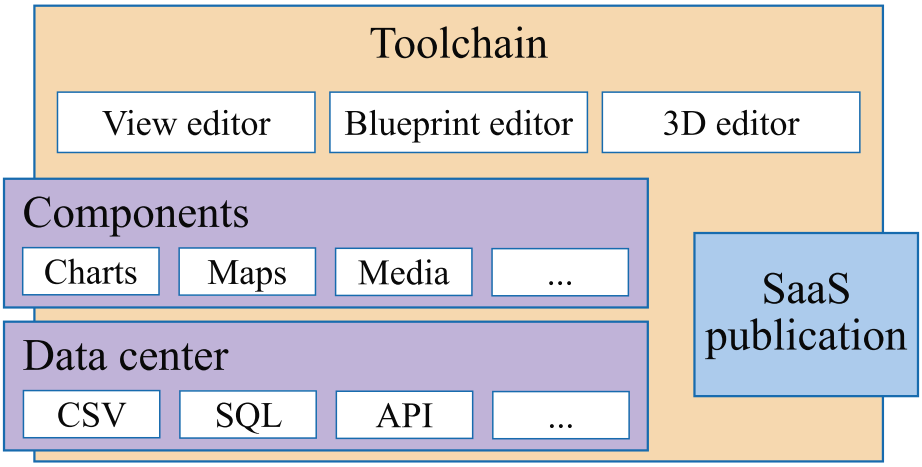

Fig. 1. The DataV framework.

4.3. Components

Component is the basic unit of DataV projects. A component refers to an independent view, such as a bar chart or a line chart. The user selects the appropriate components and modifies the style, data, and interaction of the components to compose the LHD visualizations. Some examples of components are shown in Fig. 2.

One of the main characteristics of LHD visualization is its diverse applications. In different scenarios or different industries, the data, format, and interaction requirements of components are different. To expand the audience, DataV has developed various components that can be found in the component center (D1). The component center is an online shared market. The user can browse and select the components that best meet their needs and add them to their own visualizations. In addition to default components, DataV also cooperates with developers of other visualization libraries to utilize their component packages, such as the Echarts (Fig. 3(d)). The user can also develop their own component packages and upload them to the component market for sale.

DataV has deployed automated testing tools to ensure the quality of uploaded components. All possible errors and exceptions are captured before the components are put on the shelves. Moreover, DataV also has established a run-time monitoring mechanism to provide the user with high-quality services. The component center can integrate existing chart resources well, forming a good component ecology and community environment (D5).

4.4. Toolchain

DataV provides a complete toolchain to manage the created components to achieve flexible layout and impressive display for aesthetical pleasure while maintaining accessibility. The toolchain contains three editors: view editor, blueprint editor, and 3D editor.

4.4.1. View editor

The view editor is an interactive editor for visual specifications, which is mainly used to modify the layout and visual effects of the components. The view editor enables flexible layout of LHD visualizations (D2). With view editor, the user is allowed to add appropriate components, adjust their sizes, and modify the layout through clicks and drags. A control panel on the side provides direct manipulation to adjust overlays, access data, and specify color mappings (Fig. 7).

4.4.2. Blueprint editor

The blueprint editor enables flexible weaving of interactions between components with visual programming presented as a node-link structure (Fig. 4). The blueprint editor abstracts components and logical operations into two different types of nodes, namely “component node” and “logical node”. A component node (Fig. 4(a)(d)) contains events (Fig. 4(b)) that are triggered by changes (e.g., clicked or animation finished) in the component and actions (Fig. 4(e)) that control changes to the component. Dataflows are represented by wires from events to actions, which specify interactions by controlling the response actions of components when different events occur. Logical nodes (Fig. 4(c)) specifies the data mapping and transformation of dataflows, enabling process control, data processing, and device input, etc.

The user is allowed to specify complex interactions by creating different nodes and connecting nodes via wires. In this way, DataV gets rid of the need for text programming and reduces the user’s learning curve (D5).

4.4.3. 3D editor

The 3D editor is a design tool for 3D components in 3D scenes on LHDs. 3D scenes are usually related to spatiotemporal data and can help build context and enhance immersion (D3). However, the construction of 3D scenes is very complicated and involves multiple transformations that requires the expertise of graphics. To reduce user learning costs, DataV provides the 3D editor. In the 3D editor, the user can create multiple scenes, add visual effects, and control the animation frames. As shown in Fig. 5, the 3D editor also supports quick build of 3D region maps by simple selection on 2D maps, visual elements overlay (e.g., lines, glyphs, heatmap colors.), and camera paths set. Components created through the 3D editor also support events and actions that allow interactions to be specified in the blueprint editor. The easy-to-use editor allows the user to focus more on the expression of scenes and data without having to think about the realization, which is achieved by predefined impressive effects designed by professional designers (D4).

4.5. Publishing

DataV provides one-click publishing of designed LHD visualization as a SaaS application (D5) to allow the user to share and communicate with data easily. The user can access the generated visualization through a Web page. This makes the visualization available on any devices, including LHDs, personal computers, and

Fig. 2. Some examples of components grouped by data types, including (a) regular charts for tabular data, (b) text, (c) maps, and (d) networks.

Fig. 3. Examples of different styled bar charts, including (a) basic bar chart, (b) rounded bar chart, (c) striped bar chart, and (d) bar chart with custom shapes.

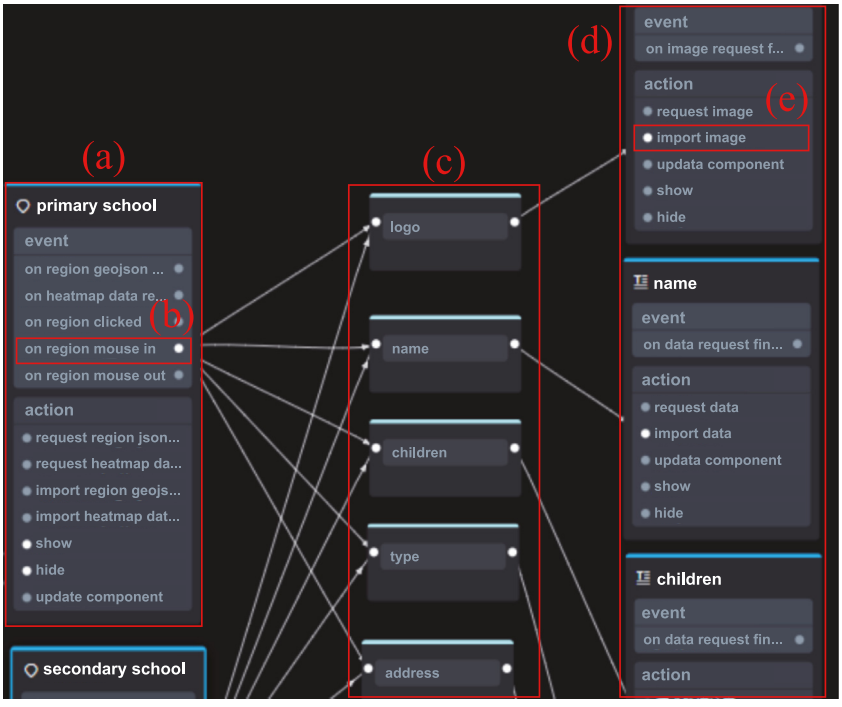

Fig. 4. An example of the node-link structure in the blueprint editor. The "blueprint" consist of (a) the component node where (b) the event is processed to start the dataflow, (c) logical nodes, and (d) the component nodes where the dataflow ends and trigger (e) the action. This blueprint specifies the interactions of the visualization presented in Section 6.1.2.

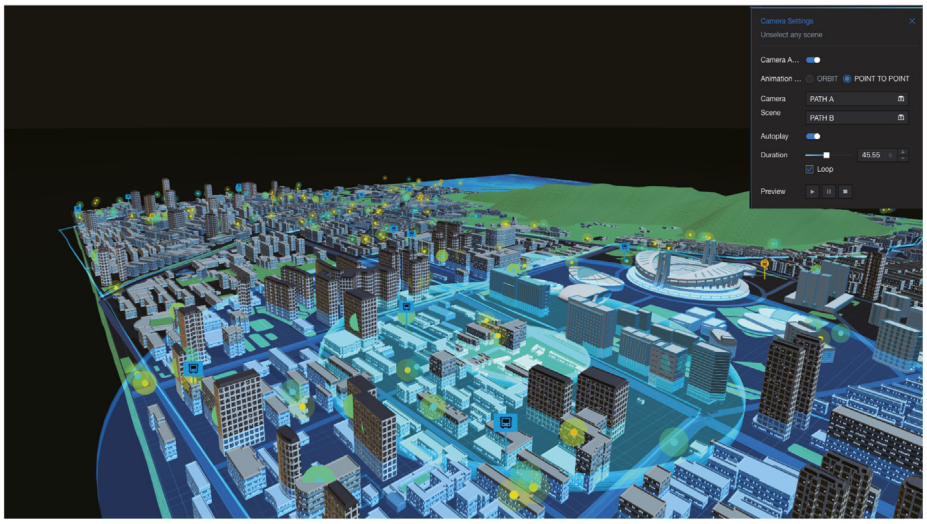

Fig. 5. The 3D editor. This scene is created by selecting a rectangular region on the map of Hangzhou, China. Points to identify the stores, lines on the roads, regions of the business circles, and glyphs showing the position of bus stops are added. The control panel at the top-right allows the user to set camera paths quickly.

smartphones. DataV also provides the option to force authenticate through a password or the token before the visualization can be seen to keep the user's data security.

The SaaS publishing of DataV is based on snapshots; thus, online changes do not affect the published content. In this way, the new versions of the visualization on LHDs for long-term displays can be updated quickly and seamlessly.

5. Implementation

In this section, we discuss the implementation and user interface of DataV. DataV is implemented in HTML5 with Cloud computing techniques. The users access DataV through web pages, and the visualizations they created are stored and published through Cloud services.

5.1. Overall process

First, we discuss the overall process of creating and presenting LHD visualization using DataV, which is illustrated in Fig. 6. There are several participants in this process, including developers who develop custom components, authors who create LHD visualizations through the interactive interface, and presenters who present the final products.

Developers can program custom widgets or encapsulate the visual effects from an existing library (e.g., ECharts (Li et al., 2018)) for use in DataV as components. The custom components they developed are managed by the unified component center. Automatic tests provided by DataV ensure the usability and stability of these custom components.

The authors are the primary users of the toolchain provided by DataV. They import data, add components, and design LHD visualizations using the editors provided by DataV. The specifications of the visualizations are then stored and published by cloud services to provide SaaS access. DataV provides Cloud services with two main modules: Object Storage Service (OSS) for storing data and visual specifications; and web hosting for cross-platform access.

Presenters can access the created visualization through web pages that can access different data sources, display specified visual effects, and support user interactions. DataV also supports the authentication of presenters before accessing the corresponding web pages.

5.2. Interface and interaction

Except for custom widgets development and library encapsulation, which are processed through textural programming, DataV provides a full-fledged interface and interaction to enable other steps throughout the process.

5.2.1. Console desk

All the user starts with a console desk that integrates the data center, component center, and the access to all projects stored on OSS. After logging into the DataV console, the user can check the tutorial, manage data sources or custom components, and add or edit LHD projects. The project workspace supports grouping, sorting, and searching for projects.

5.2.2. Editor canvas

The user can create or select a project and enters the editor canvas, which starts as the view editor by default (Fig. 7).

In the view editor, the user can specify the layout and color palette, adjust the position and size of each component, and the preview the LHD visualization created through the canvas. DataV provides a thumbnail view in the bottom right corner of the canvas that the user can preview or adjust the layout. After selecting single or multiple components, the user can perform the following operations:

- *Drag*: Drag to move.
- *Resize*: Move the mouse around the components' boundaries. Once the resize icon appears, drag to resize.
- *Rotate*: Move the mouse around the components' boundaries. Once the rotate icon appears, drag to rotate.
- *Align*: Click the alignment or distribution icon on the right control panel.
- *Group components*: Group through the right-click menu.
- *Add to the blueprint editor*: Add through the right-click menu.

When a component is added to the blueprint editor, the corresponding component node with all available events and actions is generated. On the basis of component nodes, the user wires or adds logical nodes to specify interactions (Fig. 4). The 3D editor is embedded in the editor canvas. When 3D components are selected, the control panel on the right will automatically switch to options such as model import, shader selection, and animation manage.

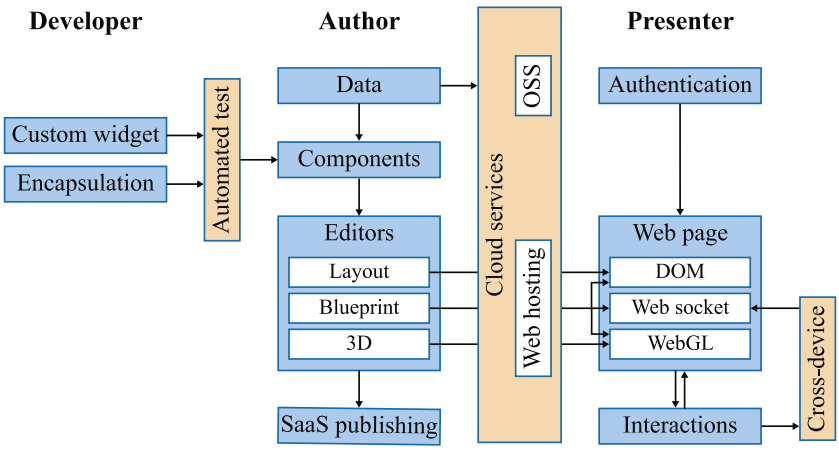

Fig. 6. The overall process of creating and presenting LHD visualization using DataV.

Fig. 7. The interface of the view editor.

5.3. SaaS accessing

All static data and visual specifications that the user adds in the DataV are stored on OSS. When the user clicks to publish, DataV generates a link to an accessible Web page through the Web hosting service of the Cloud. Users can open this link through a Web browser to display the resulting visualization. This Web page is implemented by HTML5 technology, displaying components via SVG and utilizes WebGL to accelerate the rendering of 3D scenes.

In addition, the generated Web pages can communicate with Cloud servers through Web sockets, enabling requests to online remote data and cross-device interactions.

6. Cases and comparisons

6.1. Cases

6.1.1. Predefined templates

DataV provides a variety of predefined templates for the user to easily and rapidly construct LHD visualization. As illustrated in Fig. 8, DataV support rich type of components, flexible layout, and impressive geospatial representation. Examples include grid-aligned dashboards, map-centric displays, and visual effects embedded in the 3D scene.

6.1.2. Interactive visualizations

Next, we present an example showing how to specify interactions using the blueprint editor. The geographical and other information about primary and secondary schools in Hangzhou is displayed in this example (Fig. 9).

By creating nodes and wires in the blueprint editor, the created LHD visualization can provide multiple ways to change the data to display interactively. The tabs on the top are used to switch between primary and secondary schools. The user can set up the visibility of different types of schools on the map by clicking radio boxes. When hovering the regions on the map, the school information corresponding to each region is displayed. In addition, the detailed information of each school is displayed when its name in the list is clicked. Taking the hovering as an example, Fig. 4 shows how this interaction is implemented in the blueprint editor. When the cursor enters a region on the map, the event “on region mouse in” (Fig. 4(b)) is triggered. This event passed the data related to the hovered region, of which the attributes are extracted (Fig. 4(c)) and passed to other components (Fig. 4(d)) to update the detail views.

6.1.3. Geospatial data representation

The 3D editor can provide easy construction of realistic 3D scenes, in which data representations can be embedded. Fig. 10 shows an example built upon the traffic data of an airport. In the construction of this scene, the 3D editor allows the user to split different floors of the airport after importing the original model to

Fig. 8. Some examples of the predefined templates provided by DataV, including (a) grid-aligned dashboards, (b) map-centric displays, and (c) visual effects embedded in the 3D scene.

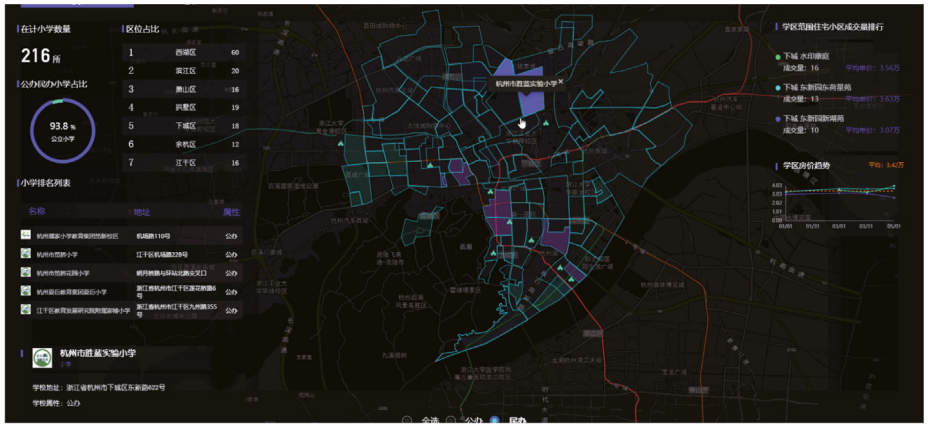

Fig. 9. An example of interactive visualization specified by the blueprint editor. The detail views on both sides are updated with user interaction. Some part of the corresponding blueprint editor interface is shown in Fig. 4.

display the data inside it better. Then, particles that show crowd flows are attached to each floor.

6.2. Comparisons

In this section, we compare DataV with other existing BI tools and research works, including Tableau, Power BI, VisComposer, and iVisDesigner (Ren et al., 2014). All these approaches support authoring multiple charts on the screen. We compare these approaches based on the identified design guidelines for LHD visualization (Table 1).

First are the data types and layouts supported by different tools (G1). Both DataV and BI tools support multiple types of data sources, while the prototype systems in research works usually import data from files. These approaches all support floating layout for free control of chart positions. In addition, DataV and Tableau also support a tiled layout, which can automatically align and resize charts based on the overall canvas. DataV and Power BI also support auto-resize options such as fit to page and fit to width, which improves flexibility to display charts on different devices with varying screen sizes and aspect ratios. Flexible

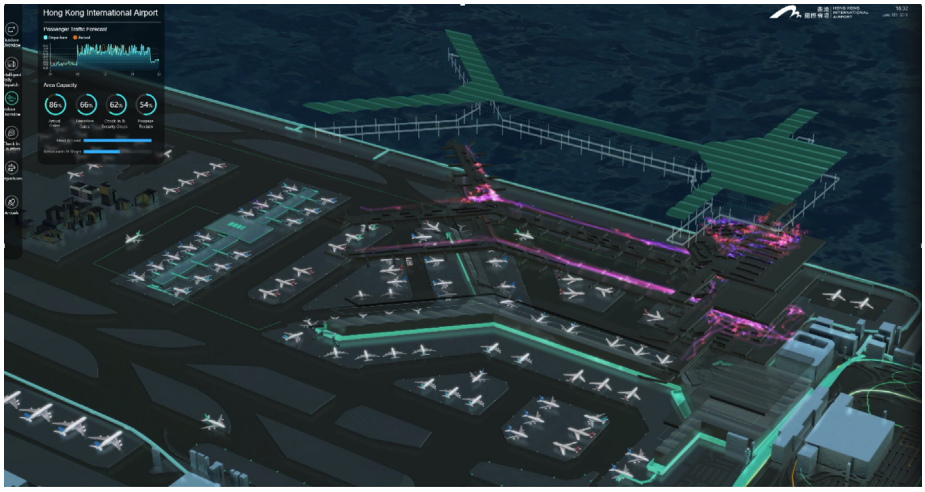

Fig. 10. An example of a 3D scene constructed by the 3D editor. The imported model is splitted into multiple floors to better display the data inside the model.

Table 1
Comparisons of different approaches to authoring multiple charts.

	Tableau	PowerBI	VisComposer	iVisDesigner	DataV
Data source	All[a]	All	File	File	All
Tiled layout	✓	✗	✗	✗	✓
Floating layout	✓	✓	✓	✓	✓
Auto fit	✗	✗	✓	✗	✓
Styled charts	✗	✗	✗	✗	✓
Animated charts	✗	✗	✗	✗	✓
Interaction	Fixed	Fixed	✗	Fixed	Flexible
3d map	Extension[b]	Extension[c]	✗	✗	✓

[a]All common data source, including databases, text files (e.g., csv files), and Web APIs.
[b]Mapbox Geospatial Analytics Extension for Tableau.
[c]Mapbox Visual for Power BI.

Table 2
Comparisons of the mean frame rates of deck.gl and DataV when rendering the scene in Fig. 11 and when interacting.

	DataV	deck.gl
Render (overview)	55.1 fps	23.8 fps
Pan (overview)	29.5 fps	4.8 fps
Zoom	16.4 fps	2.1 fps
Render (zoomed view)	44.2 fps	18.7 fps
Pan (zoomed view)	36.4 fps	7.5 fps

supports for data source and layout make DataV easy to display different data (G1).

Second, in terms of visual effects, DataV provides more styled charts for selection than traditional BI and visual analysis tools. For example, the bar charts have various styles such as rounded corners, bar with custom shapes (Fig. 3). In addition, DataV also supports animated charts such as the carousel list. Depending on the blueprint editor, DataV can achieve flexible interaction specification. In contrast, most interactive tools only provide fixed interactions such as brushing and linking. These differences allow DataV to display data in a more flexible and interesting way (G2), making it more suitable for presentations.

Third, in terms of displaying geographic information (G3), DataV has native support for 3D scenes, while Tableau and Power BI require extensions to provide the ability to draw 3D maps. We also tested the 3D rendering performance of DataV. We construct a scene that contains 430,000 buildings in Shanghai (Fig. 11) and tested the mean frame rates when use deck.gl (based on Mapbox) and DataV to render this scene or interact(Table 2). All the tests run in Chrome on a MacBook Pro with 2.7 GHz 4-core Intel i7 (Gen 8) CPU, and the resolution is set to 1920 × 1080 pixels. Compared with deck.gl, we can find that DataV has better performance, reflected on the mean frame rates.

7. Discussions

7.1. Design alternatives

From the usage scenarios and design requirements, we can conclude that the primary goal of the tools for LHD visualizations is to provide support for diverse audiences with simple operations. However, there is more than one design that can achieve similar goals. Here we have some brief discussions based on the successful practices of other similar software and researches, analyzing the existing design alternatives and the reasons for our decisions.

7.1.1. Template-based construction

We considered a number of different patterns of visualization construction tools when developing DataV, balancing between flexibility and accessibility. Although declarative languages and those interactive tools with high customizability (Satyanarayan et al., 2019) can provide greater design flexibility, they are not applicable to all populations. For the user who lacks visualization expertise and experience with these tools, they can only imitate the samples rather than actual design. In contrast, it is more practical to let the user choose from the templates designed by professional designers and then customize them. Therefore, we adopt a template-based design pattern and then extend it based on the predefined components.

In particular, we found template-based construction beneficial for a cold startup in practice. As mentioned above, LHD visualization can greatly motivate the user to explore and produce insights with their domain knowledge. Therefore, allowing the user to utilize templates for the rapid construction of preliminary results to see and modify can improve efficiency.

7.1.2. Blueprint editor

The blueprint editor is designed to provide flexible interaction specifying. Before developing the blueprint editor, DataV utilizes callback functions to specify the response to interactions. Based on the flexibility of JavaScript, callbacks are efficient and are widely used on Web-based visualizations (e.g., D3). However, for LHD visualization, which displays multiple charts synchronously, the relationship between events and responses becomes more

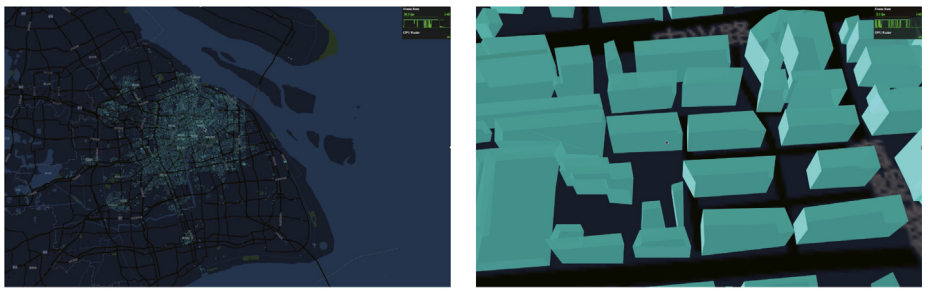

Fig. 11. The scene build for comparing the performance of deck.gl and DataV when rendering 3D maps. This scene contains 430,000 buildings in Shanghai. (a) The overview. (b) The zoomed view.

complex. As the number of charts and interactions increased, the triggering and connection of callbacks become increasingly massive. To this end, we designed the blueprint editor, which can also be seen as a visualization of the original callback specification. Through the visible "blueprint", the user can better understand and manage the added interactions.

7.1.3. SaaS

One of the main differences between DataV and common visualization construction tools is SaaS publishing. This design choice is based on the usage scenarios of LHD visualizations. In contrast to standalone visualization, the SaaS visualization published by DataV enables cross-platform access, remote publishing, and seamless runtime updates. This well suits the common uses of LHD visualization, such as large-screen panels and real-time monitoring. The automated SaaS publishing of DataV encapsulates the underlying resolution adaptation, database access, and multi-device linkage, thus the user can focus on the visual design itself.

7.2. Limitations and future work

For higher accessibility, DataV is designed to construct visualization through template-based components and drag-and-drop interactive specifications. However, this also limits the user's creativity to some extent. In practice, we found that a large percentage of user-created visualizations often share a few designs, which are the original components without changes. To improve the user's design creativity, we plan to better combine the predefined components with the user's design intentions by recognizing hand-drawn sketches or recommendations based on imported data in the future.

With the continuous development of mobile devices, there has been an increasing need to view data and visualization anytime and anywhere. Portable display devices such as tablets or smartphones are becoming important visual display platforms. LHD visualization is not directly applicable to smaller screens, leaving many developers with additional adaptation costs. How to adapt LHD visualization to different display sizes quickly and automatically is another direction of our future work.

8. Conclusion

We formulated a set of guidelines for designing LHD visualization and present DataV, an efficient SaaS framework for constructing LHD visualization. The main characteristics of DataV are threefold. First, DataV provides high accessibility by template-based components and interactive specification of layout and interactions. Second, Web rendering based on HTML5 and WebGL allows impressive visual effects. Third, SaaS publishing powered by Cloud services supports sophisticated cross-platform access and cross-device interaction. Examples in practical use demonstrate the usability of DataV.

CRediT authorship contribution statement

Honghui Mei: Methodology, Writing - original draft. **Huihua Guan:** Software, Writing - review & editing. **Chengye Xin:** Software, Project administration. **Xiao Wen:** Software, Supervision. **Wei Chen:** Resources, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Wei Chen is supported by National Natural Science Foundation of China (61772456, 61761136020). This project is also partially funded by Alibaba-Zhejiang University Joint Institute of Frontier Technologies (AZFT).

References

Andrews, C., Endert, A., Yost, B., North, C., 2011. Information visualization on large, high-resolution displays: Issues, challenges, and opportunities. Inf. Vis. 10 (4), 341–355.

Ball, R., North, C., North, C., Bowman, D.A., 2007. Move to improve: Promoting physical navigation to increase user performance with large displays. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 191–200.

Bezerianos, A., Isenberg, P., 2012. Perception of visual variables on tiled wall-sized displays for information visualization applications. IEEE Trans. Vis. Comput. Graphics 18 (12), 2516–2525.

Bradel, L., Endert, A., Koch, K., Andrews, C., North, C., 2013. Large high resolution displays for co-located collaborative sensemaking: Display usage and territoriality. Int. J. Human-Comput. Stud. 71 (11), 1078–1088.

Chen, W., Huang, Z., Wu, F., Zhu, M., Guan, H., Maciejewski, R., 2017. VAUD: A visual analysis approach for exploring spatio-temporal urban data. IEEE Trans. Vis. Comput. Graphics 24 (9), 2636–2648.

Heer, J., Van Ham, F., Carpendale, S., Weaver, C., Isenberg, P., 2008. Creation and collaboration: Engaging new audiences for information visualization. In: Information Visualization. Springer, pp. 92–133.

Kim, N.W., Schweickart, E., Liu, Z., Dontcheva, M., Li, W., Popovic, J., Pfister, H., 2017. Data-driven guides: Supporting expressive design for information graphics. IEEE Trans. Vis. Comput. Graphics 23 (1), 491–500.

Li, D., Mei, H., Shen, Y., Su, S., Zhang, W., Wang, J., Zu, M., Chen, W., 2018. Echarts: A declarative framework for rapid construction of web-based visualization. Vis. Inform. 2 (2), 136–146.

Liu, D., Xu, P., Ren, L., 2018. Tpflow: Progressive partition and multidimensional pattern extraction for large-scale spatio-temporal data analysis. IEEE Trans. Vis. Comput. Graph. 25 (1), 1–11.

Mei, H., Chen, W., Ma, Y., Guan, H., Hu, W., 2018a. Viscomposer: A visual programmable composition environment for information visualization. Vis. Inform. 2 (1), 71–81.

Mei, H., Chen, H., Zhao, X., Liu, H., Zhu, B., Chen, W., 2016. Visualization system of 3d global scale meteorological data. J. Softw..

Mei, H., Ma, Y., Wei, Y., Chen, W., 2018b. The design space of construction tools for information visualization: A survey. J. Vis. Lang. Comput. 44, 120–132.

Ni, T., Schmidt, G.S., Staadt, O.G., Livingston, M.A., Ball, R., May, R., 2006. A survey of large high-resolution display technologies, techniques, and applications. In: IEEE Virtual Reality Conference (VR 2006). IEEE, pp. 223–236.

Reda, K., Johnson, A.E., Papka, M.E., Leigh, J., 2015. Effects of display size and resolution on user behavior and insight acquisition in visual exploration. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 2759–2768.

Ren, D., Höllerer, T., Yuan, X., 2014. Ivisdesigner: Expressive interactive design of information visualizations. IEEE Trans. Vis. Comput. Graphics 20 (12), 2092–2101.

Robinson, A.C., 2008. Design for synthesis in geovisualization.

Saket, B., Kim, H., Brown, E.T., Endert, A., 2016. Visualization by demonstration: An interaction paradigm for visual data exploration. IEEE Trans. Vis. Comput. Graph. 23 (1), 331–340.

Sarikaya, A., Correll, M., Bartram, L., Tory, M., Fisher, D., 2018. What do we talk about when we talk about dashboards? IEEE Trans. Visual. Comput. Graph. 25 (1), 682–692.

Satyanarayan, A., Lee, B., Ren, D., Heer, J., Stasko, J., Thompson, J., Brehmer, M., Liu, Z., 2019. Critical reflections on visualization authoring systems. IEEE Trans. Visual. Comput. Graph. 26 (1), 461–471.

Stodder, D., 2013. Data Visualization and discovery for better business decisions. In: TDWI Research.

Stolte, C., Tang, D., Hanrahan, P., 2002. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. IEEE Trans. Vis. Comput. Graphics 8 (1), 52–65.

Wilkinson, L., 2006. The Grammar of Graphics. Springer Science & Business Media.

Yost, B., North, C., 2006. The perceptual scalability of visualization. IEEE Trans. Vis. Comput. Graphics 12 (5), 837–844.

Zhang, T., Wang, X., Li, Z., Guo, F., Ma, Y., Chen, W., 2017. A survey of network anomaly visualization. Sci. China Inf. Sci. 60 (12), 121101.