Augmenting Static Visualizations with PapARVis Designer

Zhutian Chen[1], Wai Tong[1], Qianwen Wang[1], Benjamin Bach[2], Huamin Qu[1]

[1] Hong Kong University of Science and Technology, Hong Kong, [2] University of Edinburgh, UK
{zhutian.chen, wtong, qwangbb}@connect.ust.uk, bbach@ed.ac.uk, huamin@ust.hk

Figure 1. Augmenting static visualizations can leverage the best of both physical and digital worlds: a) a data journalism uses an augmented static visualization to extend the space of the newspaper that is limited by the banner; b) a designer uses AR to update the outdated wall-sized timeline without recreating it; c) a tourist overlays the trajectories data on a public map in AR to see his/her moving pattern.

ABSTRACT

This paper presents an authoring environment for augmenting static visualizations with virtual content in augmented reality. Augmenting static visualizations can leverage the best of both physical and digital worlds, but its creation currently involves different tools and devices, without any means to explicitly design and debug both static and virtual content simultaneously. To address these issues, we design an environment that seamlessly integrates all steps of a design and deployment workflow through its main features: *i)* an extension to Vega, *ii)* a preview, and *iii)* debug hints that facilitate valid combinations of static and augmented content. We inform our design through a design space with four ways to augment static visualizations. We demonstrate the expressiveness of our tool through examples, including books, posters, projections, wall-sized visualizations. A user study shows high user satisfaction of our environment and confirms that participants can create augmented visualizations in an average of 4.63 minutes.

Author Keywords

Visualization in Augmented Reality; Augmented Static Visualization; Data Visualization Authoring.

CCS Concepts

•Human-centered computing → Visualization systems and tools;

CHI'20, April 25–30, 2020, Honolulu, HI, USA

ISBN 978-1-4503-6708-0/20/04…$15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

INTRODUCTION

Data visualizations—in the form of posters, newspapers, scientific reports, on public displays as well as slideshows and mobile screens—become increasingly more widespread. In most of these cases, visualizations are printed on paper, which, due to its extremely low-tech and tangibility, makes it very simple to create, distribute, view, and engage with the visualization content: they can be viewed in-situ without requiring any specific hardware; viewers can freely engage with them through touch, pen and annotation, and sharing thoughts and discussions in collaborative settings. However, a drawback of paper visualizations is that these visualizations are *static*, *i.e.*, the displayed information is limited in both *space* and *time*.

Augmented Reality (AR) allows for dynamic and interactive content, by adding an extra dynamic display layer onto an existing visualization. This combination can complement the static visualizations through extra data (Figure 1a), update outdated data (Figure 1b), highlight data (Figure 1c), show details, protect privacy, provide 3D content and interactivity [22], *etc*. We envision such kind of *augmented static visualizations* being used for public display (*e.g.*, information board, artworks in exhibitions), education (*e.g.*, textbook), and creative products (*e.g.*, giftcards), *etc*.

At the same time, it is demanding to create combinations and hybrids of static and virtual AR-visualizations since a couple of criteria must be met to provide for a seamless and consistent integration. We can define these *criteria of consistency* (C1-C3) as follows:

- **C1: Graphical consistency** refers to equal graphical styles between static and virtual content, *e.g.*, fonts, colors.
- **C2: Readability** requires to provide for correct interpretation of visualizations across both media, *e.g.*, aligning

related (equal) graphical elements and labels, axes, and layouts; avoid visual clutter and overlap of graphical elements.
- **C3: Validity** of visual encodings, *e.g.*, values of the same visual variables mean the same data.

Support for maintaining these crucial characteristics is specifically more problematic when design iterations are required, *e.g.*, trying different layouts, exploring the available space in AR. Without an integrated authoring approach, the design process is not just technically tedious, *i.e.*, switching between authoring tools and environments, but can lead to inconsistent visualizations if a designer does not manage to manually account for consistency (C1-C3).

To address C1-C3, this paper introduces ***PapARVis Designer***, an authoring environment to create augmented static visualizations. The design of PapARVis Designer (Sec.5) is based on a design space (Sec.3) that defines possible and valid ways of combining static and virtual content. Similar to DXR [21], PapARVis Designer delivers an extension to the Vega grammar [18], leveraging its simplicity and expressiveness to specify 2D visualizations. PapARVis Designer enables designers to create static and 2D virtual visualizations based on the same specification (C1,C3) and to deploy everything with one click. To facilitate design and avoid invalid combinations, we included two bespoke features into PapARVis Designer: first, ***AR-preview*** gives a preview of the augmented static visualization that allows designers to view their designs on the desktop environment, thus assuring *Readability* and reducing switching between devices (C2); second, ***Validator*** automatically validates a design based on our design space and provides guidelines for correction to ensure consistency of the visual encodings between the static and virtual visualizations (C3).

To demonstrate the expressiveness of PapARVis Designer, we provide an exemplary gallery of 13 augmented static visualizations (Figure 1 and Figure 7), each varying in data type, chart type, media, and purpose. A controlled user study with 12 visualization designers but without further experience in AR development suggests the overall usability of PapARVis Designer and the two main features *AR-preview* and *Validator*. PapARVis Designer is available at `https://github.com/PapARVis`.

RELATED WORK

This section overviews prior research on AR visualization, augmenting physical documents, and visualization tools.

AR Visualization

Willett et al. [27] introduced embedded data representations, a conceptual framework to unify the research on visualization systems that connect to the physical world. As an important method to blend digital data and the physical world, AR has attracted the attention of the visualization community. Benefits of visualizing data in AR have been reported by previous research. First, for example, AR can visualize data in the physical space to facilitate certain visual explorations and collaborative analysis. Butscher et al. [6] presented ART, an AR collaborative analysis tool, and reported that ART could facilitate communication and coordination between collaborators, since the interaction is grounded in the physical world. Besides, AR can augment physical objects with rich digital information, enabling situated analytics. Zhao et al. [30] developed a mobile AR system to support on-site analysis of co-authoring patterns of researchers. SiteLens [19] visualizes relevant data in the 3D physical context for urban site visits. ElSayed et al. [10] developed an AR system to help customers filtering, finding, and ranking products during shopping. Moreover, AR has been used to present visualizations for communication purposes [7] as it has the potential to engage audiences better. In summary, previous work shows that AR can connect virtual data to physical spaces, augment real objects with rich information, and even engage users. We draw on this line of work and attempt to augment static visualizations with virtual visualizations in AR, harnessing the best of both physical and digital media.

Augmenting Physical Documents

A variety of AR systems (*e.g.*, projector-based [26], handled-based [3], and HMD-based [13]) have been proposed to augment physical documents by providing additional functionality and content. For example, HoloDoc [13], a mixed reality system based on HoloLens, augments physical artifacts to allow users to take notes and look up words. Although these systems augment physical documents with rich virtual content, they are not designed for augmenting static visualizations, which requires a high level of information integrity, visual style consistency, and spatial alignment between the static and virtual content. Recently, initial explorations have been made to augment static visualizations: Kim et al. [22] presented VisAR, an AR prototype system that provides interactive functions to static visualizations. Differently, we aim to use AR to extend static visualizations with additional data.

Visualization Authoring Tools

A number of authoring tools have been proposed to facilitate the creation of visualizations in desktop and AR environments. Prior work on creating desktop visualization ranges from highly expressive programming toolkits to easy-to-use WIMP UI tools. Programming toolkits (*e.g.*, D3 [5], ProtoVis [4]) provide strong expressiveness and support flexible visualization designs but require a high level of programming skills. Instead, WIMP UI systems (*e.g.*, Lyra [16], iVisDesigner [15], and Voyager [28]) lower the barrier of creating visualizations by providing interactive functions to allow users to design visualizations via drag and drop, but compromising the flexibility. To strike a balance between expressivity and simplicity, Vega [18] and Vega-Lite [17] enable users to define visualizations with a concise declarative specification in JSON format. Overall, these systems focus on 2D visualizations on desktop platforms but cannot create visualizations in AR.

On the other hand, tools have recently been proposed for creating visualizations in AR environments. For example, MARVisT [7] is a touch-based creativity support tool to assist general users in creating AR-based glyph visualizations. Programmatic toolkits have presented to support more flexible visualizations; DXR [21] and IATK [9] are fast prototyping toolkits, that provide both programmatic and graphical user interfaces to help designers create immersive visualizations.

These systems mainly focus on 3D visualizations, thus requiring knowledge of 3D computer graphics (*e.g.*, shader, meshes, and 3D camera).

Most important, these tools do not envision extending any existing static visualization. First, none of these tools can create both static and virtual visualizations simultaneously, thus requiring the designer to switch between tools frequently. Second, none of these tools can help designer to ensure the consistency of data, visual styles, and positions between the static and virtual visualization. Our PapARVis Designer presents an authoring environment to create visualizations crossing between reality and virtuality.

DESIGN SPACE

This section discusses the concept of validity for augmented static visualizations, explores the design space for spatially aligning static and virtual content, and derives a set of design goals for our authoring environment.

What kind of augmented static visualizations is valid?

In this work, we distinguish between these three terms: *i) static visualizations* (V_s), which are static but can be in different media (*e.g.*, be printed, projected, and displayed in digital screens), *ii) virtual visualizations* (V_v), which are displayed in AR (*e.g.*, the virtual timeline in Figure 1b), and *iii) augmented static visualizations* (V_{ar}), which combine both static and virtual visualizations in AR.

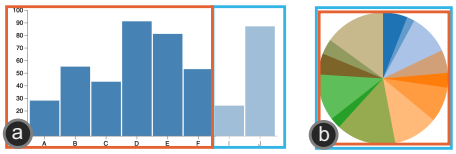


Figure 2. The validity of augmented static visualizations: a) A valid augmented bar chart; b) An invalid augmented pie chart.

To maintain perceptual effectiveness, we propose that the visual encodings in V_s and V_v should be consistent with respect to C3, *i.e.*, the same visual values mean the same data. Otherwise, the visualization design is invalid. For example, Figure 2 shows that when augmenting a V_s with additional data items, a bar chart visualization is *valid*; a pie chart visualization is *in*valid since the arc length of the virtual pie chart leads to inconsistent mappings (C3).

How can a static visualization be augmented by AR?

As mentioned in the previous section, The design space of augmented static visualizations can be determined by two dimensions: *i)* the **visual encodings** of V_s and V_v, which can be **different** or the **same**; and *ii)* the **composition** of V_s and V_v, which can be an **integrated** view or two **separate** views. We use these two dimensions to construct a design space (Figure 3), which outlines four ways to augment V_s:

- **Extended View *(Same visual encoding + Integrated composition)*.** Figure 3a presents the scenario that V_v has the same visual encodings with V_s and is integrated with V_s in a single view. *Augmentation Point*: In this augmentation method, V_s and V_v are actually the same visualization (*i.e.*, use the same visual encodings in the same view) but different in data. The designer can extend V_s with additional data, allowing a static visualization to present continually updated data or to provide details on demand. *Validity*: A particular validation point of this method is that the designer should consider the data dependency between V_s and V_v. Not all V_s can be augmented in this way. For example, a pie chart V_s^{pie} cannot be augmented with more data items through V_v^{pie}. Specifically, V_v^{pie} can only match the "updated" V_s^{pie} but not V_s^{pie} itself. Thus, V_v^{pie} will hinder the perception of V_s^{pie}. In other words, using this method, the designer must ensure V_s^{pie} will not change when the new data is appended.
- **Composite View *(Different visual encoding + Integrated composition)*.** Figure 3b depicts the condition when V_v has different visual encodings with V_s and is integrated with V_s. *Augmentation Point*: When augmenting V_s in this way, the designer can visualize another dataset using V_v that complements V_s, or can present extra data attributes of the dataset of V_s. *Validity*: Given V_v has different visual encodings from V_s, the designer does not need to concern the data dependency between them and should focus on general design issues, *e.g.*, occlusions between V_v and V_s.
- **Small Multiple *(Same visual encoding + Separate composition)*.** Figure 3c shows the scenario where V_v has the same visual encodings with V_s and is displayed as a separate view. *Augmentation Point*: V_v can be used to present other datasets in the same visual encodings of V_s. Generally, the result of this method is a small multiple. *Validity*: The augmentation will always be valid as V_v is displayed separately from V_s. However, the designers should consider whether the visual encodings fits for different datasets, *i.e.*, the scalability issue.
- **Multiple Views *(Different visual encoding + Separate composition)*.** Figure 3d demonstrates the case that V_v has different visual encodings with V_s and is displayed separately from V_s. *Augmentation Point*: Choosing this augmentation allows the designer to use AR to extend V_s to a multiple view (*e.g.*, a dashboard), presenting different perspective of the data behind V_s, or use V_v to visualize new datasets along with V_s that yields deeper insights into the data. *Validity*: This method can always ensure valid AR

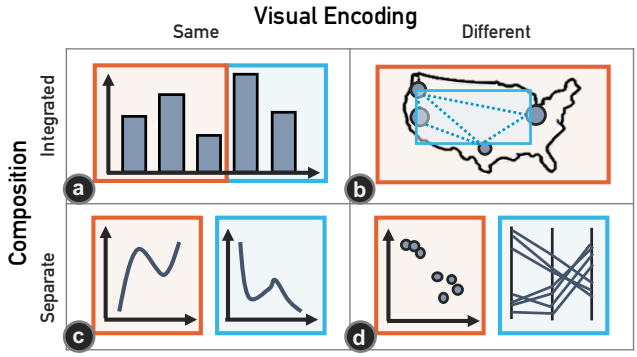


Figure 3. Four ways to augment static visualizations using AR: a) Extended View; b) Composite View; c) Small Multiple; d) Multiple View.

visualizations and has the most flexibility as V_s and V_v are completely independent (both in data and visually).

In summary, in our design space, V_s can be augmented in four different ways with different augmentation points. Among these four augmentation methods, *Extended View* is the most strict one, as in which V_s and V_v are highly correlated; *Multiple View* is the most flexible one, as in which V_s and V_v are completely independent of each other.

Design Goals

To create augmented static visualizations, we conceive our authoring environment to accomplish three main goals:

G1: Integrate the visualization design in one tool—To assure *graphical consistency* (C1) and *validity* (C3) between V_s and V_v, we aim to integrate V_s and V_v into one single specification and allow designers without AR expertise to create both V_s and V_v in a single tool simultaneously: specify V_s and V_v, test and debug, and deploy.

G2: Preview the visualization design in one platform—To assure *readability* (C2) of an augmented static visualization, the designer may need to work back-and-forth between the desktop platform and AR devices (*e.g.*, head-mounted displays, mobile handheld). This kind of cross-device workflow is tedious and time-consuming. The authoring environment should alleviate this burden as much as possible. Given that V_s will be part of the reality, we can preview the V_v together with the V_s on the desktop platform to simulate the AR scenario. Moreover, some augmentations (*e.g.*, *Composite View*) may display data that is unknown during the process of visualization design. Therefore, an authoring environment must allow previewing the augmented static visualization.

G3: Provide automatic design support—The designer must assure *validity* (C3) of an augmented static visualization. As indicated by our design space, designing valid *Extended View* is the most challenging since in which there are data dependencies exist between the V_s and V_v. When new data is appended and visualized by the V_v, a mismatch between the V_s and V_v can easily happen due to inappropriate visual encodings, leading to an invalid augmented static visualization. When the data is large, or the visual encodings are complex, it will be difficult for the designer to verify and debug the inappropriate design manually. Thus, the authoring environment should automatically verify the design and provide hints for debugging inappropriate visual encodings.

USAGE SCENARIO

To illustrate how PapARVis Designer accomplishes the three design goals and introduce the full workflow, we describe how Bob, a hypothetical visualization designer, creates the node-link diagram in Figure 7a.

Bob is a teaching assistant coordinate of the Computer Science and Engineering department (CSE). He is asked to create a poster for introducing the CSE to the prospective post-graduate (PG) students. He plans to use a hierarchical visualization to present the organization structure from the university to the department as an overview, as well as the faculties in the CSE for details. Since presenting the complete hierarchical structure requires a large area, Bob has to hide some details in the poster. Besides, some new faculties might join the CSE in the future, making the current poster outdated. Considering these issues, Bob decides to use PapARVis Designer to create an augmented hierarchical visualization in the poster.

Bob opens PapARVis Designer and decides to choose a node-link tree example to initialize his design. Bob loads the data in the Vega code and defines the data to be displayed in AR in the `ar` block (Figure 4a). For the uncertain child nodes (*e.g.*, the new faculties), he defines a placeholder for them in the `ar` block using wildcard characters. PapARVis Designer previews how the tree visualization looks like in AR (Figure 4c) by showing both the real and virtual parts which are indicated by the orange and blue border boxes, respectively.

In the preview (Figure 4c), Bob notices a terrible mismatch between the real and virtual visualizations. But he does not know why this happens. His attention is then attracted by a warning hint showing in the Vega code (Figure 6a). Reading this hint, Bob notices that this mismatch is caused by the "cluster" layout, which places leaf nodes of the tree at the same depth. Thus, when new virtual nodes with different depths are appended to the leaf nodes of the real tree, all the nodes of the real tree need to be repositioned to make sure that the added virtual leaf nodes can be placed at the same depth with the real leaf nodes, leading to the mismatch. Following the hint, he changes the layout from "cluster" to "tidy", which appends new nodes without changing the existing nodes. The mismatch problem is then solved (Figure 4b).

Satisfied with the design, Bob clicks the `publish` button to export his design. PapARVis Designer automatically separates the real and virtual part and generates the tree for presenting in reality with a QRCode that identifies the tree diagram (Figure 4d). For the virtual part, PapARVis Designer pushes the Vega specification to a backend server for further processing, including generating an AR reference image of the real tree so that it can be recognized by AR viewers, converting the virtual part into AR environments, and hosting it in a repository that can be fetched by AR viewers. These details are handled by a "black box" that Bob does not need to concern. Finally, Bob uses an AR viewer on his mobile phone to scan the real visualization and observe the virtual visualization (Figure 4e).

PAPARVIS DESIGNER

This section describes the design and implementation of PapARVis Designer, our authoring environment for augmented static visualizations.

Workflow

PapARVis Designer creates both static and virtual visualizations. Instead of developing a tool that combines both the functionalities of visualization tools for desktop and for AR environments, we abstract the common part of them and propose a decoupled workflow (Figure 5). By separating the specification (*i.e.*, designing visualizations) from the deployment (*i.e.*, presenting visualizations), our workflow allows designers without AR expertise to create both V_s and V_v (G1) in a single specification and leave the deployment to AR experts.

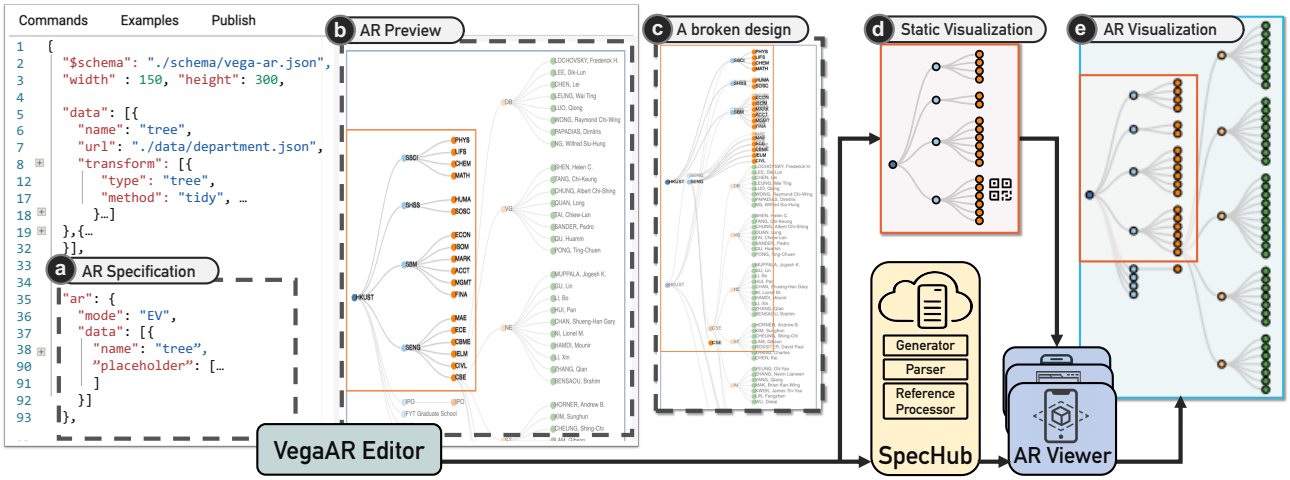


Figure 4. The main user interface and pipeline: 1) The designer specifies the static and virtual (in the `ar` block) visualization in the VegaAR Editor, which preview the result on the desktop platform. 2) Then he publishes the specification to the SpecHub and exports the static part for printing. The SpecHub will further process the specification and prepare all prerequisites for the augmented static visualization. 3) Finally, the audiences can use an AR viewer to scan the static visualization, fetch the virtual part from the SpecHub, and combine them to display the augmented static visualization.

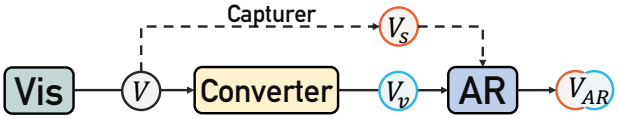


Figure 5. A workflow creates both static and virtual visualizations: a "stem" visualization (V) can be rendered as a static visualization (V_s) and a virtual visualization (V_v), which are then combined as an augmented static visualization (V_{AR}).

Specifically, our workflow adopts a browser/server architecture. In the workflow, the designer uses a visualization authoring tool to create a "stem" visualization which can then be captured as a static visualization V_s. The designer can continue updating the "stem" visualization, similar to update a web page, and use a converter to convert it into the virtual visualization V_v. An AR viewer can be used to combine the V_s and V_v to obtain the augmented static visualization. By this mean, both V_s and V_v are created based on the same specification, thus maintaining a high level of consistency.

We implement the workflow in PapARVis Designer (Figure 4), which consists of three main components:

- **VegaAR Editor** is built based on a UI editor of Vega [24]. In VegaAR Editor, the designer can create a V_s following the same practice in Vega and specify the V_v in an `ar` block, which will be further introduced in the next section (Sec.5.2). When finishing the creation, the designer can `publish` the design to export the V_s (Figure 4d) and push the whole specification, including the `ar` block, to *SpecHub*. VegaAR Editor will generate a QRCode that links the V_s to the specification on SpecHub.
- **SpecHub** is deployed on a cloud server to receive and host the specifications from VegaAR Editor (*i.e.*, like a piece of code hosted on Github). Besides, it prepares all prerequisites for the AR visualization, such as processing the specification of V_s to generate the AR reference image so that V_s can be recognized by AR viewers, parsing the `ar` block to render the V_v with the new data whenever a user views it using an AR viewer.
- **AR Viewer** is the endpoint (*i.e.*, browser) for audiences to view the augmented static visualization. When scanning a V_s, an AR viewer will identify the V_v based on its QRCode, fetch its corresponding V_v from the SpecHub, and register V_v onto V_s to present the augmented static visualization (Figure 4e). The AR viewer is not platform-specific. In this work, we have implemented three AR viewers on the iOS, Android, and web-based platform, respectively.

With this workflow, designers can create augmented static visualizations in one tool without digging into the AR details.

AR-Preview

We design *AR-preview* to avoid the frequently switching between devices (G2). Specifically, to preview V_v together with V_s, the designer must explicitly specify the V_v during the creation, which is not supported by the current Vega grammar. Thus, we extend the Vega grammar by adding an additional configuration block `ar` (Figure 4a) to it. In the `ar` block, the designer can explicitly specify the augmentation method, the visual encodings, and the data of a V_v, no matter the data is certain or uncertain:

- **Certain**. When the data is known (*e.g.*, some detail information cannot be presented due to the space limitation), the designer can specify the V_v in the `ar` block, starting from defining its augmentation method: 1) in *Extended View*, the designer only needs to specify new data and which datasets of the V_s to be appended, and can reuse other visual encodings of the V_s; 2) in *Composite View*, the designer can specify a new visualization in the `ar` block with new datasets or with the existing datasets of the V_s; 3) in *Small Multiple*, the designer only need to specify the new datasets and the layout of the virtual small multiples in relation to the V_s; 4) in *Multiple View*, the designer is allowed to create a complete new visualization in the `ar` block and define its placement with respect to the V_s.
- **Uncertain**. The data of the V_v could be uncertain during the creation (*e.g.*, the data can only be obtained in the future).

In this case, it will be arduous for the designer to imagine the V_v. We provide a `placeholder` mechanism to allow designers to generate mockup data using wildcard. In the `ar` block, the designer can create placeholder datasets and generate their data by specifying the data type (*e.g.*, categorical, temporal, and quantitative), number of datum, and ranges or potential options of the values of each column.

The physical space of the V_v is determined by the Vega specification (*e.g.*, set the canvas width to 500 unit), relative to the V_s. By explicitly defining the V_v in the `ar` block, VegaAR Editor will provide visual preview of the virtual content together with the V_s, thus reducing the switching between devices.

Validator

An *Extended View* (see design space) can easily be invalid given the data dependency between V_s and V_v (*e.g.*, the pie chart in Sec.3.1 and the tree diagram in Sec.4). When choosing this augmentation method, PapARVis Designer automatically verifies the dataflow of the visual design and provides hints for debugging invalid visual encodings (G3).

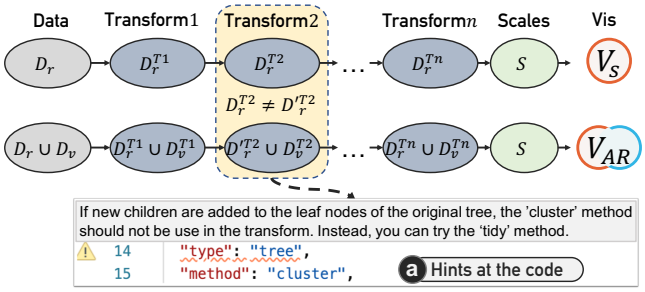


Figure 6. Validate the visual design by comparing the dataflows, thus enabling providing debug messages specific to a transform.

Instead of using a rule-based method, we propose to validate the visualization by comparing the dataflow between a V_s and an augmented static visualization. A general visualization process of a dataset D_r can be summarized in Figure 6: D_r will first be processed by a serials of data transforms and then mapped to visual channels to construct the V_s. As demonstrated in Sec.3.2, in a valid augmented static visualization, the V_s should remain unchanged after appending the new data D_v, which means its all intermediate states D_r^T before the visual mapping should also be unchanged. Thus, for each data transform, we can compare the intermediate states of the V_s and the augmented static visualization to see whether the intermediate states derived from D_r changes. If an intermediate state changes, for example $D'^{T2}_r \neq D_r^{T2}$, it means the transform (*i.e.*, Transform2) alters the V_r after appending new data, thus leading to an invalid augmented static visualization. Therefore, we can give the designer a hint that is specific to a transform and further provide messages for fixing the invalid visualization (*e.g.*, Figure 6a) regardless of the complexity of the visualization. For example, when trying to augment a treemap, the system warns `avoid 'treemap' when new nodes are added to the internal nodes` as this would update the layout of the underlying treemap.

Implementation

The implementation of PapARVis Designer mainly consists of three parts, namely, VegaAR Editor, SpecHub, and AR viewers. VegaAR Editor is a web-based application that extends the Vega Editor [24], Vega Schema [25] (*i.e.*, for the `ar` block), and Vega Compiler [23] (*i.e.*, for the *AR-preview* and *Validator*). SpecHub is a NodeJs-based application that runs on a cloud server. SpecHub also reuses the extended Vega Compiler in VegaAR Editor to parse and generate V_v. We have implemented AR viewers based on Vuforia [11] on three different platforms, including iOS (iPhone8 Plus), Android (Huawei P10), and web-based platform. All AR viewers only provide minimal functionalities, including QRCode decoder, AR image recognition, and 3D registration.

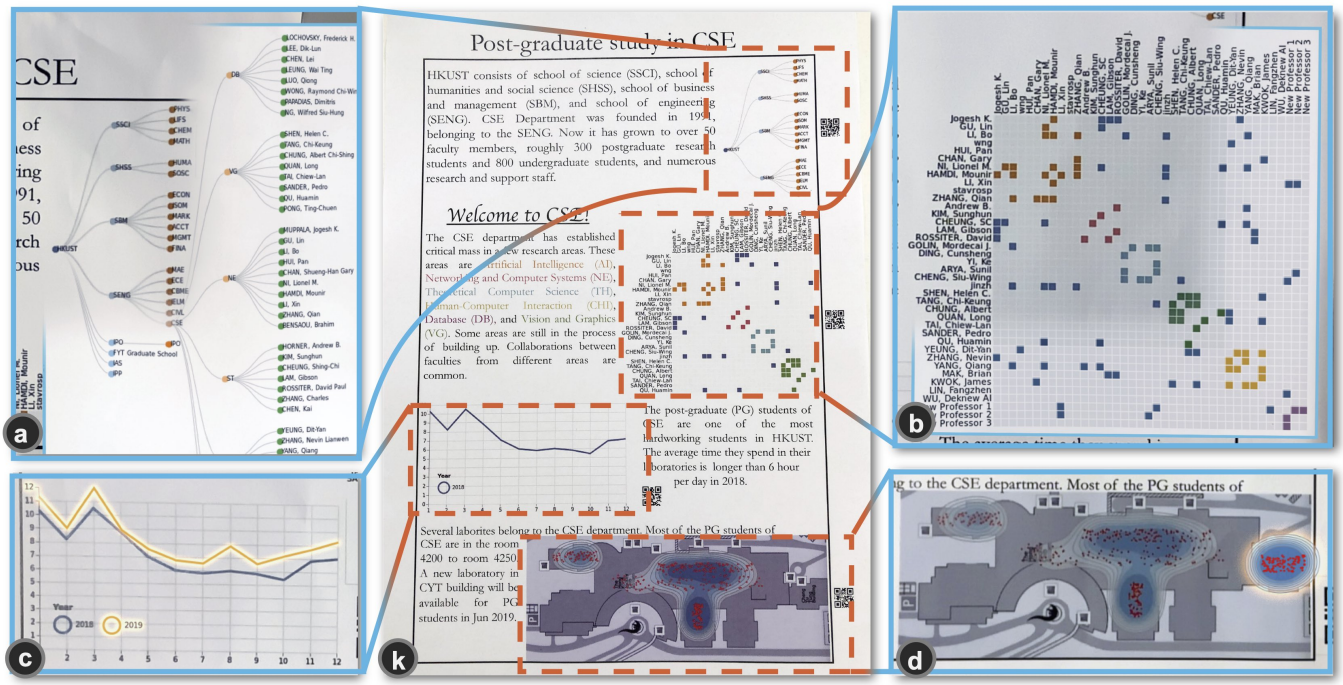
EXAMPLES AND SCENARIOS

Figure 1 and Figure 7 present examples varying in data type, chart type, augmentation method, media, and purpose. We summarize scenarios that can benefit from augmenting static visualizations.

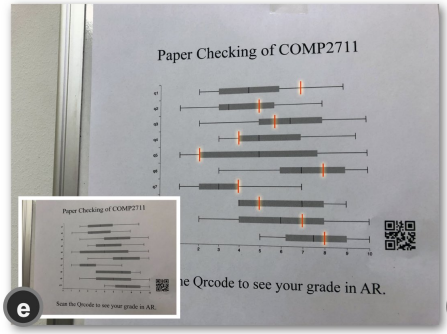
Overcoming space limitations—AR can extend a static visualization where the space is scarce. For example, in Figure 1a, given that much space of the newspaper is occupied by the banner, the data journalism uses an augmented static visualization to extend the canvas for visualization, thereby presenting additional data. A similar usage can be found in the example of poster (Figure 7a), leaflet (Figure 7f), name card (Figure 7i), and resume (Figure 7j). Moreover, in the wall-sized timeline in Figure 1b, it is difficult to physically extend the wall for presenting updated data. By using augmented static visualizations, the designer can easily break through the limitation of physical world and present more information.

Displaying new data—AR can update the static visualizations, thus providing the latest information and saving the cost to recreate them. In real world, information can be updated frequently while the materials to present the information might not. Augmented static visualization is a promising way to relieve the mismatch between the update cycle of information and materials. Figure 1b presents a representative example of using AR to update a wall-sized timeline, which has been outdated for more than three years. Figure 7g demonstrates a stock market book uses an augmented static visualization to provide the latest S&P index data to help the reader better understand the historical context. Generally users can also use augmented static visualizations to keep their information in static visualizations online. For example, Figure 7i is an academic name card with google scholar citations that will not be outdated after distributing.

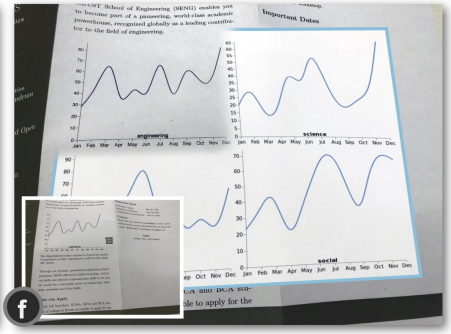
Showing details—AR can empower static visualizations with the capability to show details on demand. "Overview first, zoom and filter, then details on demand" [20] is a well-known information visualization guideline yet can hardly be followed in static visualizations. Augmented static visualizations can achieve this classic technique on static visualizations. For example, Figure 7a presents the overview hierarchical architecture of a university and collapses the detail nodes into AR, thus avoiding information overload. As shown in Figure 7h and j, many tree diagrams can benefit from this feature.



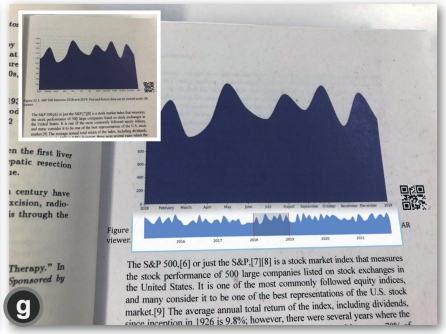
k) A poster with four augmented visualizations that introduces an university department, including a) a node-link tree that visualizes the hierarchy structure of the school and the detail information of faculties, b) a matrix that presents the coauthor network between faculties and the new faculties and collaborations, c) a multiple line chart that visualizes the mean in-lab time per day of the post-graduate (PG) students of the department in 2018 and 2019, d) a contour geo map that visualizes the check-in data of the PG students of the department in 2018 and the first season in 2019



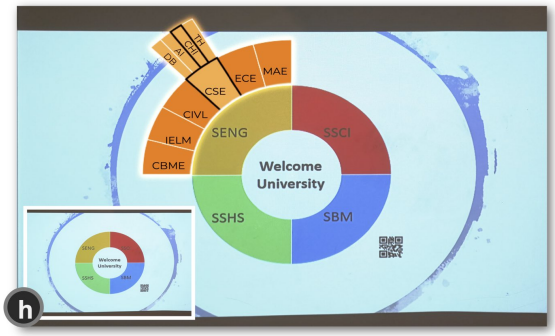
e) A box plot presents in exam paper checking session shows the statistic of all students and the individual scores to protect privacy



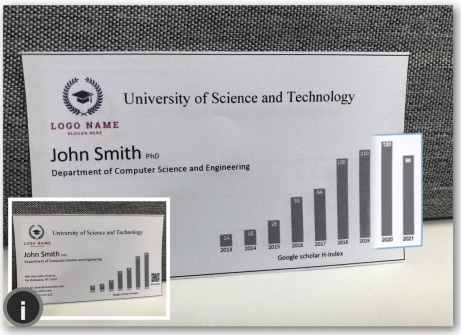
f) An advertisement leaflet that present the departmental salary of CSE and those of other departments



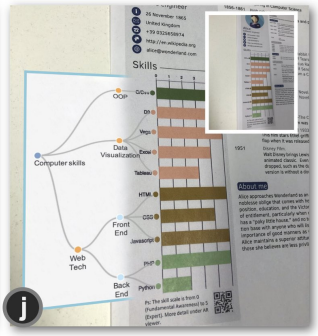
g) A book about stock market presents the S&P 500 index in 2018 as a focus and two years before and after as an updated context



h) A projected donut chart in the registration center of the university presents the four schools and details of each school based on the audience



i) An academic name card with a google scholar citations allows directly checking the updated information instead of typing the URL to open the web page



j) A customized resume visualizes the supplemental information apart from the most critical one

Figure 7. Examples created by PapARVis Designer. a) - d) show the augmenting results of k). In e) - j), the sub-figures show the static visualizations.

Complementing additional data—For visualizations in public spaces, augmented static visualizations can be used to overlay personal data. For example, in Figure 1c, by overlaying the trajectories data onto a public map, the augmented static visualization complements the geographical visualization with the user's own data, thus helping the user to understand his/her movement pattern. Another example of complementing visualizations is to provide supplemental information. The leaflet advertisement in Figure 7f shows the salaries of other departments in AR for comparison purpose, since this extra information is not the focus of the advertisement but can help readers obtain the context information.

Protecting privacy—Augmented static visualizations can protect individuals privacy. In Figure 7e, the teacher printed the class' summary statistics for an exam, overlaying a student's personal ranking in the augmented static visualization. Similarly, Figure 7h shows a projected chart for the new students to check their private information.

USER STUDY

We conducted a controlled user study to assess the usability and the utility of PapARVis Designer. The study aimed to evaluate whether visualization designers without AR expertise could create augmented static visualizations using PapARVis Designer and whether the two advanced features (*AR-preview* and *Validator*) facilitated the creating process.

Baseline technique: As no comparable tool exists for a fair baseline comparison, we provided two versions of PapARVis Designer: a *base mode* (BASE) and a *pro mode* (PRO). PRO provided full features of PapARVis Designer while the BASE did not provide the *AR-preview* and *Validator*. To ensure a fair comparison, BASE also provided the `ar` block (without the visual preview) for explicitly defining the virtual visualizations.

Tasks: The study simulated creating both static and virtual visualizations. We took the poster example in Figure 7k (without the visualizations) and asked participants to complete the poster by creating the four visualizations (referred to as tasks T1-T4). T1-T4 covered a wide range of data types and were required to extend the visualization using an *Extended View*, as we considered this extension is the most challenging and hence expected strongest evidence for our tool. For each task, participants were provided with the background information, relevant Vega documents, an instruction to load the dataset, and one or two Vega examples to initialize the design.

- **T1** ***Validity-tree*** required participants to visualize a hierarchical dataset about the university structure. The provided examples were a node-link tree and a treemap. The treemap was ensured to be invalid for the dataset while the node-link tree was invalid in default but can be fixed to be valid by modifying *one* visual encoding. The purpose of this setting was to assess whether the participant can choose the proper example and debug the design in different modes.
- **T2** ***Validity-matrix*** had the same purpose as **T1** but required participants to visualize a network dataset about the collaboration among faculties in the college. The examples provided a matrix diagram and a radial node-link graph. The radial node-link graph was ensured to be invalid for the dataset while the matrix diagrams was invalid in default but can be fixed by modifying *one* visual encoding.
- **T3** ***Occlusion*** required visualizing a temporal dataset about the mean in-lab time of students. The provided examples included a single line chart and a multiple line chart. Both examples were valid without any corrections. We deliberately put the text description of **T3** on the right to see how the participants deal with the potential occlusions due to the AR extension in different modes.
- **T4** ***Unnoticeable*** required the participants to visualize a geographic dataset. The provided example was a contour map. The example was invalid in default for the dataset but can be corrected by modifying *one* visual encoding. However, the default invalid design only led to an unnoticeable mismatch, *i.e.*, the mismatch of **contour**s, between the V_s and V_v. The mismatch is expected to be detected by *Validator* and we want to see how the participants handle the conflict between their observation and our hints.

The four tasks were divided into two groups (*Validity-tree* & *Occlusion vs*. *Validity-matrix* & *Unnoticeable*) to balance the workload in different modes. The materials we provided to the participants can be found in the supplemental materials.

Participants and Apparatus: We recruited 12 participants (8 male; age: 22-30, average 25.6), who had at least two-year experience with Vega or D3 (since Vega is built based on D3) and no expertise in AR programming (*e.g.*, Unity). According to the pre-study survey, all participants had more than two years experience on data visualization ($\mu = 3.54, \sigma = 1.44$). Each participant received a gift card worth $14 at the beginning of the session, independent of their performance. The study was run in the lab, using a 15-inch laptop, or their own. A 5.5-inch iPhone8 Plus was provided to view the augmented static visualization.

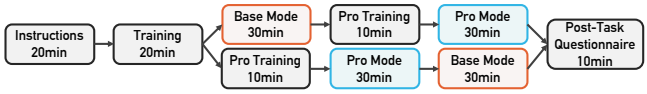


Figure 8. Participants completed four tasks, two in each of two conditions (BASEor PRO). The conditions were counter-balanced across participants, thus each participant completed only one path in this figure.

Procedure: The procedure of our study is summarised in Figure 8. We first introduced the purpose of the study, explained the concept of *Validity* via examples (*i.e.*, Figure 2), introduced Vega, as well as how to create a visualization with PapARVis Designer (*introduction*, 20min). Participants then were given a step-by-step *training* (20min) instruction to reproduce the two examples in Figure 2. Participants were encouraged to ask questions and explore examples.

We performed a within-subject study on the four tasks, two in each condition: BASE (30min) and PRO (30min). We counterbalanced the order of the conditions across participants: the PRO-group started with the PRO and then performed on BASE, the BASE-group did start with BASE. Since a task cannot be done in both conditions, we also counterbalanced the order of the groups of tasks across participants. Before performing with PRO, participants were given an instruction sheet of the two advanced features (*AR-preview* and *Validator*) and were

encouraged to explore the PRO and ask questions (PRO *training*, 10min). Before each task, participants were provided with the materials and encouraged to ask question about and get familiar with the materials. A task was started when a participant was confident and confirmed to begin and ended when the participant confirmed finishing. Participants finished with a *post-study questionnaire* (10min, adapted from [14]) in which they rated their self-perceived efficiency, effectiveness, and mental demand on a scale from 1 ("better in BASE") to 7 ("better in PRO"). To collect subjective feedback for PRO, participants rated its usability, usefulness, and satisfaction and answered questions in a semi-structured interview. Each session lasted approximately 1.5-2 hours.

Task Performance Measures: We recorded task completion *time* and *correctness* for each task. A correct design a) does not occlude its text description and b) is valid. All measures were explicitly explained to the participants before the tasks.

Quantitative Results

Figure 9 shows the results with 95% confidence intervals (CIs). Significance values are reported for $p < .05(*)$, $p < .01(**)$, and $p < .001(***)$, abbreviated by the number of stars.

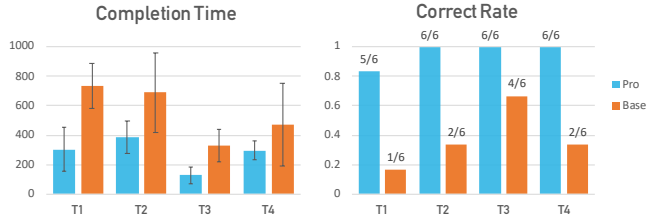


Figure 9. Left: Average task completion time and 95% CIs across tasks. Right: The correct rate across tasks.

Completion Time: Participants finished the tasks faster with PRO ($\mu = 278s$, $95\%CI = [206, 351]$) than with BASE ($\mu = 557s$, $95\%CI = [413, 701]$). Before more detailed examinations, we first confirmed that all results of completion times in each condition follow normal distribution using Anderson-Darling test. Using an independent-samples t-test with a null hypothesis that the participants perform equally fast in each mode, we found that participants performed significantly faster in PRO on *Validity-tree* $(**)$, *Validity-matrix* $(*)$, and *Occlusion* $(**)$. No significant difference was observed for completion time with *Unnoticeable* ($p = 0.14$).

Correctness: Almost all the augmented static visualizations (23/24) created in PRO were correct while far less were correct with BASE (9/24). Correctness follows a Bernoulli distribution (*i.e.*, 1 *vs.* 0), and we used Binomial test with a null hypothesis that an augmented static visualization has a equal chance to be correct or wrong, by which significant positive effects were observed on *Validity-matrix* $(*)$, *Occlusion* $(*)$, *Unnoticeable* $(*)$ in PRO, and no significant effect was observed in others.

Post-Study Feedback: Anderson-Darling test reveals that the post-study self-reported results (Figure 10) did not follow normal distribution. Thus, we used a one-sample non-parametric Wilcoxon signed rank test with a null hypothesis that the result is middle Likert scale value. We found significant positive

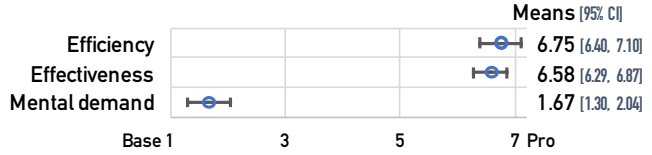


Figure 10. Means and 95% CIs of post-study self-reported measures on a scale from 1 (better in BASE) to 7 (better in PRO). Mental demand accesses how much mental activity was required.

effects for participants' reported efficiency $(***)$ and effectiveness $(**)$ in PRO. For BASE, a negative effect was observed for the participants' self-reported mental demand $(**)$.

Qualitative Results

Usability: Generally, participants lauded the usability of our system as *easy to learn* ($\mu = 6.50, 95\%CI = [6.11, 6.88]$) and *easy to use* ($\mu = 6.33, 95\%CI = [5.96, 6.70]$) (Figure 11). P8 commented that *"I have never tried Unity and AR things"* but PapARVis Designer *"can help me quickly produce AR extensible visualizations."* When we asked the participants whether they want more control of the AR details (*e.g.*, visual effects), participants preferred that the tool would be cost-effective and remained simple.

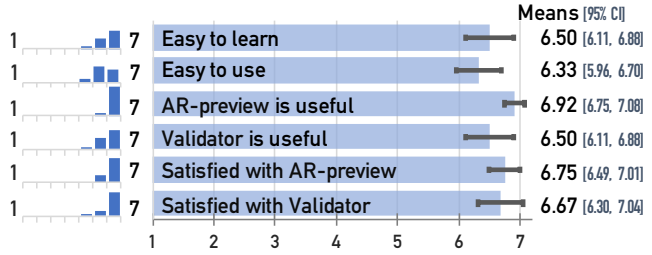


Figure 11. Subjective feedback on the usability, showing means and 95% CIs. Distributions are shown on the left.

Usefulness: As for the two features of our authoring environment, the participants commented positively and confirmed the usefulness of *AR-preview* ($\mu = 6.92$, 95% $CI = [6.75, 7.08]$) and *Validator* ($\mu = 6.50$, 95% $CI = [6.11, 6.88]$). Although we provided an AR viewer for participants to check their design in the study, participants didn't use it frequently since *"AR-preview is enough thus no need to switch to the AR device"* (P2) and *"it provides immediate feedback."* (P9). Participants also agreed the usefulness of *Validator*. As pointed out by P4, who had 6 years experiences on data visualizations, *"the Validator is really important for the debug"*. Indeed, the participants were able to finish the creation more quickly and effectively with these two features, just like P10 noted that *"creation process is tedious without Validator and AR-preview."*

Satisfaction: The participants responded with a high user satisfaction for *AR-preview* ($\mu = 6.75$, 95% $CI = [6.49, 7.01]$) and *Validator* ($\mu = 6.67$, 95% $CI = [6.30, 7.04]$). They said *AR-preview* was *"intuitive"*. Other comments mentioned improvements for *Validator*, including *"better highlight the problematic code"* (P2) and *"improve the error descriptions"* (P7).

FINDINGS SUMMARY

Our study shows that all participants were able to create augmented static visualizations using PapARVis Designer. We discuss further observations and insights here.

Whose faults? AR or my design?—As discussed in Sec. 3.1, *in*appropriate visual encoding may lead to a mismatch between the static and virtual visualizations, *e.g.*, the color, sizes, and positions. During the study, we observed that some participants suspected that the AR viewer cannot correctly align the V_v with V_s at the first time when there was a misalignment happened between the V_s and V_v. Even after confirming with us that there was nothing wrong with the AR part, some participants still doubted that the AR device worked properly, especially when designing without the help of *Validator*. In practice, given that the AR details are similar to a "black box" in our workflow, it is important to help designers distinguish the error from AR and their designs.

Where am I? Reality or virtuality?—Creating augmented visualizations requires designers to maintain two designs for V_s and V_v in mind. Although *AR-preview* can alleviate this burden, some participants could not consider both V_s and V_v simultaneously in the tasks. For example, some participants chose the single line chart to present the temporal data in *Occlusion*. When they discovered that the virtual part would occlude the text description, they attempted to flip the V_s to a right-to-left orientation and then struggled in this counter-intuitive design. An interesting fact was that all the participants who chose the single line chart and struggled in this issue had more than 3 years experiences in visualization design while those with less than 3 years experiences directly chose the multiple line chart. We suspected this was due to more experienced participants having more clear routines. How to effectively help designers closing the gap between reality and virtuality during creation is one of the important future improvements. Allowing the designer to create the visualization design in-situ rather than on a desktop can be a potential solution.

What is the hint? Ignore or follow?—In PRO, our authoring environment provides useful hints for debugging augmented static visualizations. In the study, we observed that one participant, the only one who failed in *Validity-tree* in PRO, ignored any hints. In the interview, he said *"I thought the hints are useless so I didn't read it [sic]."* Yet, the majority of participants followed the hints but admitted that they actually did not know what was wrong with their designs. We also observed that the hints occasionally confused the participants. For instance, in *Unnoticeable*, the *contour*s between the V_s and V_v were not consistent, which was too subtle to be noticed by the participants but can be detected by *Validator*. Thus, participants got confused by the hints and spent relative long time on *Unnoticeable*, which we believe was the reason of no significant difference between BASE and PRO was observed on *Unnoticeable*. We discussed this issue with the participants; 11 out of 12 agreed that the hints from *Validator* seemed as a "strict mode warning" instead of an error.

What scenarios can augmented static visualizations be used for?—We were interested in what kind of scenarios the participants could envision augmented static visualizations. In the interviews, all participants suggested that such kind of visualizations can be used for public display, such as information boards or park maps, and interactive artworks (P4-P7, P11) in exhibitions or as gifts (P2, P5).

FUTURE WORK AND LIMITATIONS

Generalization to 3D and dynamic AR visualizations—PapARVis Designer currently focuses on 2D static AR visualizations but it can be generalized to 3D and dynamic AR visualizations. As has been proven successful in DXR [21], Vega grammar can be extended from 2D to 3D visualizations. It is possible to adopt the extension and design from DXR, thus allowing designers to create 3D V_v in PapARVis Designer (examples can be found in https://github.com/PapARVis). Meanwhile, the Vega grammar inherently supports interactive visualizations, providing opportunities to bring interactivity and animation to static visualizations.

Multiple augmentations for collaborations—Our design so far concentrates on augmenting one V_s on one AR device. A promising but more challenging question is how to design AR visualizations distributed across multiple devices to support collaborations. Challenges, for example, include how to design cross-device interactions (*e.g.*, HMDs prefer mid-air gestures and handheld devices use touch), or how to support collaborations without sacrificing privacy protection.

Design with scene understandings—In many real-world examples, the immediate environment influences readability and how a visualization is read. For example, designers can create V_v that adapts to the ambient light, visualizes in-situ temperature data, or adapt more deeply to the AR-Canvas [2]. Recent progress in computer vision on visualizations [8] offers possibilities for environment adaptive visualizations.

Study Limitations—Similar to other studies of authoring tools [1, 12, 29], the sample size of our user study is small, given that access to experts is naturally limited. Further evaluation is thus suggested. PapARVis Designer currently generates a QRCode along with a V_s when publishing, which size and position can be configured in the `ar` block. While the QRCode has a good usability and is a flag to attract audiences to scan, it adds a superfluous component to the visualization. Future study is required to optimize the QRCode by striking a balance between functionality and aesthetics.

CONCLUSION

In this paper we presented and evaluated PapARVis Designer, an authoring environment for augmenting static visualization with virtual content. It integrates the workflow of creating both static and virtual visualizations by extending Vega, a popular visualization grammar. Two features are provided to facilitate the creation process: *AR-preview* reduces the switching between platforms and *Validator* provides debugging hints. We provided an exemplary gallery to demonstrate the expressiveness of PapARVis Designer as well as the broad application prospect of augmented static visualizations. A user study showed that PapARVis Designer enabled visualization designers with no AR development experiences to create augmented static visualizations. We have also shared and discussed the insights from our study, which implies future research.

ACKNOWLEDGEMENTS

This project is partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. AoE/E-603/18).

REFERENCES

[1] Fereshteh Amini, Nathalie Henry Riche, Bongshin Lee, Andres Monroy-Hernandez, and Pourang Irani. 2017. Authoring Data-Driven Videos with DataClips. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 501–510. DOI: http://dx.doi.org/10.1109/TVCG.2016.2598647

[2] Benjamin Bach, Ronell Sicat, Hanspeter Pfister, and Aaron Quigley. 2017. Drawing into the AR-CANVAS: Designing Embedded Visualizations for Augmented Reality. In *Proceedings of the Workshop on Immersive Analytics of IEEE VIS*.

[3] Mark Billinghurst, Hirokazu Kato, and Ivan Poupyrev. 2001. The MagicBook - Moving Seamlessly between Reality and Virtuality. *IEEE Computer Graphics and Applications* 21, 3 (2001), 6–8. DOI: http://dx.doi.org/10.1109/38.920621

[4] Michael Bostock and Jeffrey Heer. 2009. Protovis: A Graphical Toolkit for Visualization. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1121–1128. DOI: http://dx.doi.org/10.1109/TVCG.2009.174

[5] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309. DOI: http://dx.doi.org/10.1109/TVCG.2011.185

[6] Simon Butscher, Sebastian Hubenschmid, Jens Müller, Johannes Fuchs, and Harald Reiterer. 2018. Clusters, Trends, and Outliers: How Immersive Technologies Can Facilitate the Collaborative Analysis of Multidimensional Data. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, 90. DOI: http://dx.doi.org/10.1145/3173574.3173664

[7] Zhutian Chen, Yijia Su, Yifang Wang, Qianwen Wang, Huamin Qu, and Yingcai Wu. 2019. MARVisT: Authoring Glyph-based Visualization in Mobile Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics* 14, 8 (2019). DOI: http://dx.doi.org/10.1109/TVCG.2019.2892415

[8] Zhutian Chen, Yun Wang, Qianwen Wang, Yong Wang, and Huamin Qu. 2020. Towards Automated Infographic Design: Deep Learning-based Auto-Extraction of Extensible Timeline. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 917–926. DOI: http://dx.doi.org/10.1109/TVCG.2019.2934810

[9] Maxime Cordeil, Andrew Cunningham, Benjamin Bach, Christophe Hurter, Bruce H. Thomas, Kim Marriott, and Tim Dwyer. 2019. IATK: An Immersive Analytics Toolkit. In *Proceedings of the Conference on Virtual Reality and 3D User Interfaces (VR)*. 200–209. DOI: http://dx.doi.org/10.1109/VR.2019.8797978

[10] Neven A. M. ElSayed, Bruce H. Thomas, Kim Marriott, Julia Piantadosi, and Ross T. Smith. 2015. Situated Analytics. In *Proceedings of the Symposium on Big Data Visual Analytics (BDVA)*. 96–103. DOI: http://dx.doi.org/10.1109/BDVA.2015.7314302

[11] Vuforia Engine. 2019. Vuforia. https://developer.vuforia.com/. (2019).

[12] Nam Wook Kim, Nathalie Henry Riche, Benjamin Bach, Guanpeng Xu, Matthew Brehmer, Ken Hinckley, Michel Pahud, Haijun Xia, Michael J Mcguffin, Hanspeter Pfister, and Mathew Brehmer. 2019. DataToon: Drawing Data Comics About Dynamic Networks with Pen + Touch Interaction. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, 12. DOI:http://dx.doi.org/10.1145/3290605.3300335

[13] Zhen Li, Michelle Annett, Ken Hinckley, Karan Singh, and Daniel Wigdor. 2019. HoloDoc: Enabling Mixed Reality Workspaces that Harness Physical and Digital Content. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, 1–14. DOI: http://dx.doi.org/10.1145/3290605.3300917

[14] Arnold M Lund. 2001. Measuring Usability with the Use Questionnaire12. *Usability interface* 8, 2 (2001), 3–6.

[15] Donghao Ren, Tobias Höllerer, and Xiaoru Yuan. 2014. iVisDesigner: Expressive Interactive Design of Information Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2092–2101. DOI: http://dx.doi.org/10.1109/TVCG.2014.2346291

[16] Arvind Satyanarayan and Jeffrey Heer. 2014. Lyra: An Interactive Visualization Design Environment. *Computer Graphics Forum* 33, 3 (2014), 351–360. DOI: http://dx.doi.org/10.1111/cgf.12391

[17] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 341–350. DOI: http://dx.doi.org/10.1109/TVCG.2016.2599030

[18] Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. 2016. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 659–668. DOI: http://dx.doi.org/10.1109/TVCG.2015.2467091

[19] White Sean and Feiner Steven. 2009. SiteLens: Situated Visualization Techniques for Urban Site Visits. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, 1117. DOI: http://dx.doi.org/10.1145/1518701.1518871

[20] B. Shneiderman. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of IEEE Symposium on Visual Languages*. 336–343. DOI: http://dx.doi.org/10.1109/VL.1996.545307

[21] Ronell Sicat, Jiabao Li, Junyoung Choi, Maxime Cordeil, Won Ki Jeong, Benjamin Bach, and Hanspeter Pfister. 2019. DXR: A Toolkit for Building Immersive Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 715–725. DOI: http://dx.doi.org/10.1109/TVCG.2018.2865152

[22] Taeheon Kim and Bahador Saket and Alex Endert and Blair MacIntyre. 2017. VisAR: Bringing Interactivity to Static Data Visualizations through Augmented Reality. In *Proceedings of the Workshop on Immersive Analytics of IEEE VIS*.

[23] Vega Team. 2019a. Vega Compiler. https://github.com/vega/vega/tree/master/packages/vega-parser. (2019).

[24] Vega Team. 2019b. Vega Editor. https://github.com/vega/editor. (2019).

[25] Vega Team. 2019c. Vega Schema. https://github.com/vega/vega/tree/master/packages/vega-schema. (2019).

[26] Pierre Wellner. 1991. The DigitalDesk Calculator: Tangible Manipulation on A Desk Top Display. In *Proceedings of the Symposium on User Interface Software and Technology (UIST)*. 27–33. DOI: http://dx.doi.org/10.1145/120782.120785

[27] Wesley Willett, Yvonne Jansen, and Pierre Dragicevic. 2017. Embedded Data Representations. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 461–470. DOI: http://dx.doi.org/10.1109/TVCG.2016.2598608

[28] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 649–658. DOI: http://dx.doi.org/10.1109/TVCG.2015.2467191

[29] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. 2018. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, 223. DOI: http://dx.doi.org/10.1145/3173574.3173797

[30] Mingqian Zhao, Yijia Su, Jian Zhao, Shaoyu Chen, and Huamin Qu. 2017. Mobile Situated Analytics of Ego-centric Network Data. In *Proceedings of the Symposium on Visualization of SIGGRAPH ASIA*. ACM, 14:1–14:8. DOI: http://dx.doi.org/10.1145/3139295.3139309