iARVis: Mobile AR Based Declarative Information Visualization Authoring, Exploring and Sharing

Junjie Chen* Chenhui Li† Sicheng Song‡ Changbo Wang§

School of Computer Science and Technology, East China Normal University

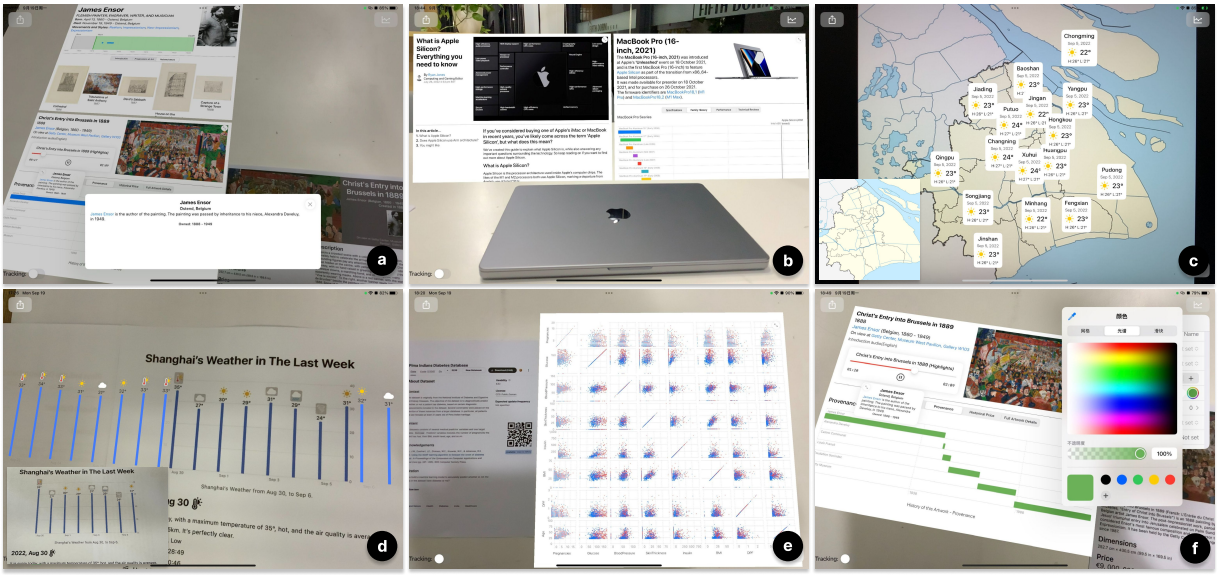

Figure 1: iARVis can easily create augmented reality-based information and data visualization environments in a declarative way for various scenarios while providing interaction ability. **a)** Simulated Scenario: Displaying visualization widgets near an artwork in a museum about its brief introduction, historical provenance and price, and additional information about the author, etc. **b)** Simulated Scenario: Displaying visualization widgets near a MacBook Pro in a store about its brief introduction, price, technical specifications, product family history, and introduction about its new Apple Silicon chip. **c)** Augment a static map visualization of Shanghai with a dynamic visualization to show real-time weather information. **d)** Augment a static bar chart visualization of Shanghai's one-week historical weather information with previous and latest weather information. **e)** Use the SPLOM visualization technique to visualize the relationship between different factors that lead to diabetes. **f)** Dynamically alter the visual encodings of a chart in the widget to match the user's preferences.

Abstract

We present iARVis, a proof-of-concept toolkit for creating, experiencing, and sharing mobile AR-based information visualization environments. Over the past years, AR has emerged as a promising medium for information and data visualization beyond the physical media and the desktop, enabling interactivity and eliminating spatial limits. However, the creation of such environments remains difficult and frequently necessitates low-level programming expertise and lengthy hand encodings. We present a declarative approach for defining the augmented reality (AR) environment, including how information is automatically positioned, laid out, and interacted with, to improve the efficiency and flexibility of constructing AR-based information visualization environments. We provide fundamental layout and visual components such as the grid, rich text, images, and charts for the development of complex visualization widgets, as well as automatic targeting methods based on image and object tracking for the development of the AR environment. To increase design efficiency, we also provide features such as hot-reload and several creation levels for both novice and advanced users. We also investigate how the augmented reality-based visualization environment could persist and be shared through the internet and provide ways for storing, sharing, and restoring the environment to give a continuous and seamless experience. To demonstrate the viability and extensibility, we evaluate iARVis using a variety of use cases along with performance evaluation and expert reviews.

Index Terms: Visualization systems and tools; Augmented Reality

1 Introduction

Over the past few years, with the development of augmented reality (AR) and virtual reality (VR) technologies, research interest in data visualization with AR and VR (e.g., situated analytics and immersive analytics) has increased rapidly. Information and data visualization in AR and VR is a promising method for visualization processes and analysis, providing the marked advantages of unlimited space, rich interactivity, and intuitive 3D graphic representation. The applications of AR and VR-based data visualization are expanding to fields such as scientific visualization [35, 55], medical visualization [26, 46], and information visualization [12, 18], etc.

*e-mail: junjiechen@stu.ecnu.edu.cn
†e-mail: chli@cs.ecnu.edu.cn
‡e-mail: scsong@stu.ecnu.edu.cn
§e-mail: cbwang@cs.ecnu.edu.cn

However, it is still difficult to validate visualization prototypes and develop practical applications for AR or VR-based environments [6,15,50] because it is a complex task that requires knowledge of concepts and technology from data visualization, 3D computer graphics, AR and VR, as well as human-computer interaction. Toolkits for creating AR and VR-based visualization environments have a long tradition [39,53]. Recently, the success and wide adoption of Vega-Lite [48] have inspired the community to develop visualization-specific toolkits for creating visualizations rapidly in a declarative way. For example, both DXR [50] and VRIA [11] provide a declarative way to create 3D data visualizations in an immersive environment using grammar that is similar to Vega-Lite.

Nonetheless, existing toolkits only consider how to create the visualization chart itself rather than the entire visualization environment. For example, displaying other visualization information such as rich text, images, audio, and videos is believed to provide a better understanding of the relationships between perceptual and abstract information [10]. Another issue with AR-based visualization is that we must place the proposed visualization in the physical world, which is a challenging task because the physical world is dynamic and we must perform the placement manually every time we enter the AR environment. For example, every time we want to check out the additional information displayed in the AR environment near an artwork in a gallery, we could place the visualization chart in the physical world and manually move it to an appropriate position.

We believe that it is time to think about how to design an easy, extensible, and user-friendly way to create the entire visualization environment in AR, with automatic positioning, interaction, persistence, and continuity.

In this paper, we present iARVis, a proof-of-concept toolkit that can create AR-based visualization environments in a declarative way. iARVis is based on the mobile augmented reality platform ARKit [4], which is one of the largest AR platforms in the world, as a low-cost alternative to the high-cost and unpopular AR headsets, thus allowing most people to access it. iARVis provides a declarative way to create complex interactable visualization widgets using basic layout components and visual element components. We also discuss the different methods that can allow an entire visualization environment to persist and be restored to achieve persistence and continuity in the AR experience. This process would allow us to save or share the visualization environment, and continue the developed workflow when revisiting it. Finally, we present several example applications using iARVis with performance evaluation and expert reviews to demonstrate iARVis's applicability and scalability.

iARVis is implemented on top of native technologies on iOS, for example, using ARKit as the AR engine and SwiftUI as the 2D content rendering engine. iARVis is responsible for parsing the specification, creating the visualization environment in AR, managing the interaction, persisting the environment, etc. Android could be an alternative platform since all methods we use are general. ARCore (Android's native AR engine) does not support features such as object tracking, so we choose iOS as our platform.

This study makes the following contributions:

- We design an efficient and extensible declarative method for creating an AR-based information and data visualization environment, including automatic positioning, user interface, interaction, etc.
- We provide features such as hot-reload and different methods of creation for both novice and advanced users to improve the efficiency of the creation process.
- We propose approaches for persistent working environments that enable the sharing and restoration of working environments when returning to the same environment.
- We present multiple example applications alongside performance evaluations and expert reviews to demonstrate the applicability and scalability of iARVis.

iARVis is available at `https://github.com/iARVis`.

2 Related Work

We briefly review existing AR and VR-based information and data visualization. Specifically, we compare toolkits for creating visualization environments for AR and VR.

2.1 Applications of Visualization in AR / VR

With the development of augmented reality (AR) and virtual reality (VR) technologies, AR and VR-based information and data visualization have been widely used in various fields, such as information visualization, scientific visualization, and medical visualization.

Immersive analytics (IA) and situated analytics (SA) [13,51,54] use novel display technologies such as AR and VR to provide a more immersive and interactive experience for data analysis. Aided by the recent availability of new and affordable AR and VR headsets, many studies have been performed over the last decade, investigating both AR and VR. Previous AR and VR-based research for data visualization has focused on CAVE environments for scientific visualization [1,19] and abstract information visualization [44]. However, recent work has also shown the potential for perception and interaction [5] and collaboration [18]. AR and VR have been used to visualize tensor-valued volumetric datasets [61], explore networks [37], collaborative immersive worlds that can manage complex data [20], immersive ego and exocentric maps [60], origin-destination flow maps [59], trajectories and brain fibers [33], and data stream from IoT [25].

Portable mobile devices are another form of AR platform that is more accessible and popular to the general public. Mobile AR device-based research has been introduced in various scenarios because ARKit [4] was released on iOS in 2017. For example, MelissAR [22], an AR system based on a tablet, shows on-site drift information to assist beekeepers in understanding bee behaviors. Based on ARKit's real-time face-tracking functionality [23], Zhao et al. [62] developed an ego-centric network AR visualization system supporting on-site analysis of temporal co-authoring patterns of researchers using an iPhone. In addition, using mobile devices as an auxiliary tool for data analysis in AR and VR is also a popular research topic [32,38]. Mobile devices are more common in daily life, and some research focuses on more common and casual scenarios, such as using mobile AR for personal visualization [14], navigation planning, shopping assistance [21], etc.

In this study, we focus on mobile AR-based information and data visualization, to provide a more accessible platform for users. The goal of iARVis is to facilitate the creation of mobile AR-based environments for visualization analysis and to visualize information in daily life. For example, with the help of iARVis, users can create a mobile AR-based environment to visualize data using advanced visualization techniques such as Small Multiples (Sect. 5.1.3) and SPLOM (Sect. 5.3.1) easily and also create visualization environments for more casual and daily life scenarios such as visiting an artwork exhibition (Sect. 5.1.1) and offline shopping (Sect. 5.1.2).

2.2 Authoring Toolkits for Visualization in AR / VR

Authoring toolkits are important in visualization research because they allow researchers and designers to build visualizations with ease of use. The design procedures for data visualization have strongly benefited from the formalization of the grammar of graphics [7]. For example, visualization authoring toolkits, such as InfoVis Toolkit [24], ProtoVis [8], Prefuse [30], d3.js [9], or Vega [49], which are based on such abstractions, have brought data visualization to a broad audience of programmers and designers. Instead of writing low-level codes, these high-level toolkits use GUI or human-readable grammar to define visual mappings and layouts for rapid and automatic design.

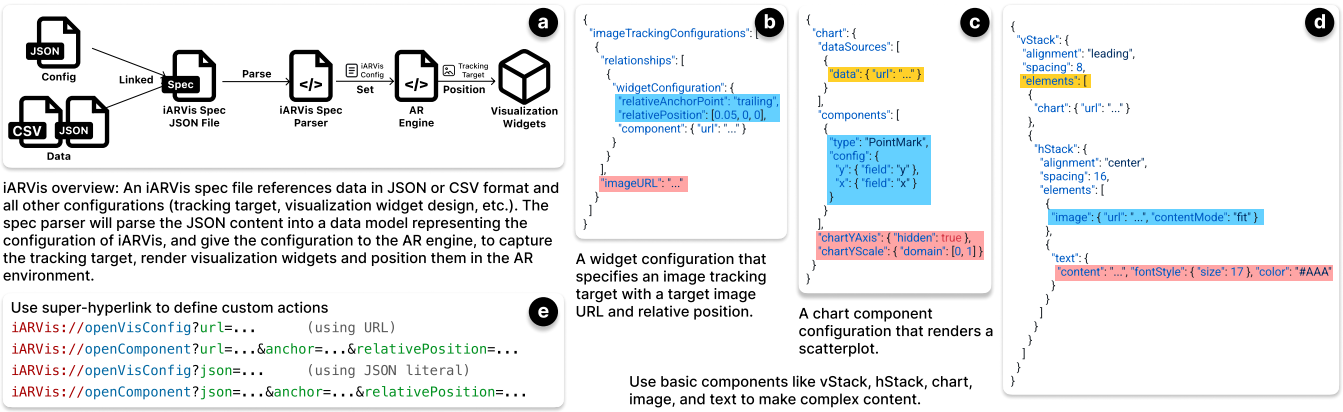

Figure 2: **a)** iARVis overview, **b)** widget configuration example, **c)** chart component example, **d)** `vStack` component example. **e)** super-hyperlink examples

In recent years, inspired by these high-level visualization toolkits, AR and VR frameworks for information and data visualization have been developed to create visualization environments. These toolkits make it easier to create visualizations for AR and VR than using 3D editors because they provide a high-level GUI or grammar for creating visualizations. For example, for immersive analytics, toolkits such as DXR [50] and IATK [17] are built with the Unity engine and provide high-level GUI (Graphical user interface) or declarative grammar to create visualizations in AR or VR. DXR and IATK both provide a GUI panel to set the visual mappings, such as color, size, and shape, to facilitate the creation of visualizations. DXR targets three levels of user expertise (beginner, intermediate, advanced) and uses a similar grammar to Vega-Lite [48]. IATK is built with scalability in mind to visualize and interact with datasets with millions of points while maintaining a usable frame rate. VRIA [11] is another toolkit that can create 3D data visualization in VR using declarative grammar, which is similar to DXR, but is designed specifically for WebXR. PapARVis [15] extends the Vega grammar to augment static visualizations to eliminate the temporal and spatial limitations.

These toolkits make creating visualization charts in 3D easier, but they aren't tailored to some requirements of AR. While DXR and IATK are helpful, they require users to manually position visualizations in the real world, which is problematic for situated analytics. The visualization environment should also include supplementary data such as rich text, images, and videos in addition to visualization charts for better grasping of the interconnections between perceptual and abstract information, but current toolkits such as VRIA and PapARVis can only create visualization charts. Another critical issue that isn't being addressed by existing solutions is the need for the visualization environment to be persistent and shared among users. All of these problems are considered in iARVis, and discussed later in Sect. 3.3 (automatic positioning), Sect. 3.2.1 (widget design), and Sect. 4 (persistence and continuity). The comparison of iARVis and existing declarative authoring toolkits are in Table 1.

Feature	DXR	VRIA	PapARVis	iARVis
Platform	Unity	Web	Unity	iOS
Content	Charts	Charts	Charts	Charts, Multimedia
Dimensionality	2D, 3D	2D, 3D	2D	2D
Positioning	Manually	Manually	QR Code	Tracking*
Interaction	✓	✓	-	✓
Hot-reload	-	-	-	✓
API	-	✓	-	✓

Table 1: Comparison between iARVis and existing declarative authoring toolkits. *iARVis supports using object tracking, image tracking, and QR code for automatic positioning. (Sect. 3.3)

3 Design

This research aims to develop a declarative approach to define the user interface and interaction of visualization widgets in augmented reality with advanced features such as automatic positioning. Because of the need for programming knowledge when dealing with user interface layout and interaction, designers in the past typically only provided some static visualization images. This is a particularly pressing problem when it comes to the visual design of 3D scenes, as game engines like Unity and Unreal Engine are commonly used in the creation of user interface and interaction design.

We thus design a declarative way to define the **automatic positioning method**, **user interface**, and **interaction** of visualization widgets, which is based on the following principles:

- Designed specifications can be stored formally and can thus be shared and reused in the future (Sect. 4).
- Design grammar should be clean, concise, and easy to learn, with no programming knowledge required.
- Provide the ability to design both visualization charts and other visual components, such as rich text, images, audio, videos, etc., to provide a rich user interface.
- Design grammar should be extensible to promise future extensions to provide unsupported visualization types and additional components (Sect. 5.3).

Also, we confine the proposed design scope with the following restrictions: **1)** Our research is based on augmented reality, which is markedly different from virtual reality. All usage scenarios that we consider are based on this difference, in augmented reality, virtual objects placed in augmented reality typically have certain relationships with some physical objects in the real world. **2)** We only consider placing two-dimensional data visualization in the proposed design space because visualizing 3D data using charts in 3D space might be controversial [2, 31, 36, 57]. Many concepts and techniques are general and reusable when applied to 3D data visualization ; thus, we suggest that 3D visualization be investigated in future research.

Due to space limitations, we primarily discuss the functionality of iARVis (see the documentation page for more details).

3.1 Overview

As shown in Fig. 2a, the conceptual model of iARVis is based on the following components: **1)** Data source and separated JSON configuration files. The data source can be JSON files or CSV files, which contain the data to be visualized, and an iARVis specification can contain multiple labeled data sources. JSON configuration files

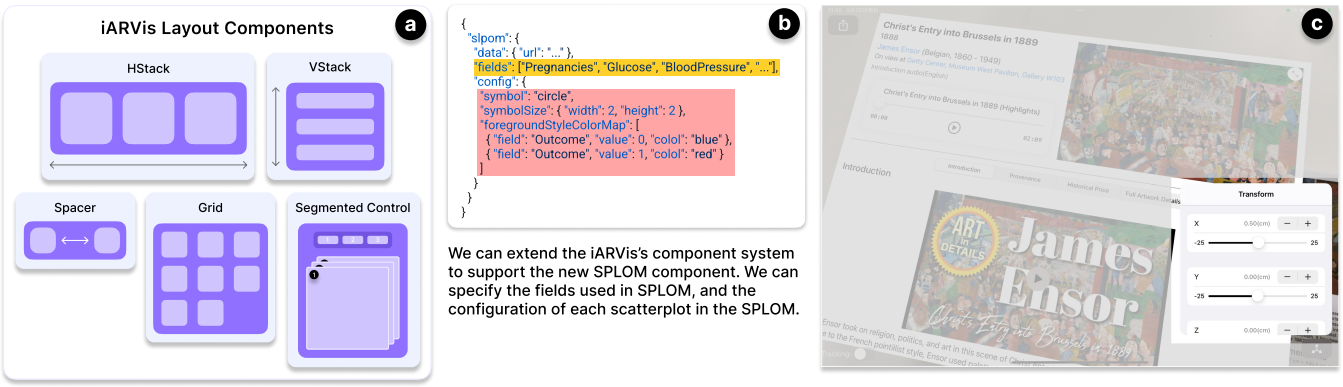

Figure 3: **a)** Explanation of iARVis's layout components. **b)** Extend iARVis to support the `SPLOM` component. **c)** Visualization widget transformation's control panel.

contain all information we need to build the visualization environment such as the visualization widget design, automatic positioning method (Sect. 3.3), etc. **2)** Specification JSON file. The iARVis specification file is a single JSON file that links all data sources and separated JSON configuration files. **3)** Specification parser. The iARVis specification parser is a module that parses the iARVis specification file and builds a data model representing the configuration of iARVis, which could be used by the AR engine to set up the visualization configuration. **4)** AR engine. The AR engine uses the configuration data model to set the tracking target, render visualization widgets, and position them in the AR environment. **5)** Visualization widgets. Visualization widgets contain the interactable visual components that are rendered in the AR environment, including charts, text, images, audio, videos, etc.

We choose JSON-based specifications as the design grammar because JSON is a human-readable format, and researchers are familiar with using JSON to represent the configuration of data visualization (e.g., using Vega-Lite) [40]. Each part of the configuration could be a literal JSON string or separated as a URL pointer to a JSON file which improves the readability of the configuration.

3.2 Visualization Widget Design

Previous researchers proposed an idea called **Information-Rich Virtual Environments** (IRVE) [10,42,43], which is a pattern that describes virtual environments with additional abstract information of text, numbers, images, etc., to provide a better understanding of the relationships between perceptual and abstract information. In the proposed design, we consider the concept of IRVE and decide to provide similar visualization widgets for iARVis.

Visualization widgets are placed in the space of AR, typically near some physical objects in the real world by automatic positioning (Sect. 3.3), showing the designed visualization content, and helping to interact with the visualization. For example, we can attach a visualization widget to an artwork in an exhibition, showing an introduction of the artwork with provenance and historical price charts (Fig. 4).

3.2.1 Visual Element Components

We provide basic visual elements including **rich text**, **image**, **video**, **audio**, **table**, and **chart** etc., and designers can easily compose these elements in a declarative way to build more complex visualization widgets. The proposed design is extensible, allowing more visual elements and visualization types to be added and supported easily (Sect. 5.3).

We use Markdown syntax [16] to generate **rich text** (Fig. 2d), which supports adding heading text, bold text, italic text, hyperlinks, etc., with additional parameters to control the size, weight, color, and multiline alignment of the text. In addition to the basic hyperlink in rich text, we also provide a super-hyperlink(URLs start with `iARVis://`, Fig. 2e), which can be used to provide more powerful actions such as opening a web page (Fig. 5a), presenting a dialog and opening a new visualization widget (Fig. 4e).

The **chart** component allows designers to specify marks with data and configurations to render charts (Fig. 2c). In the chart configuration, we can set the color mapping, mark type, interaction, etc. We can also add basic interactions to the chart, such as hovering or tapping around a data point to show provided detailed information as a tooltip (Fig. 4b, Fig. 4d), and tapping the visualization annotation to open a new visualization widget (Fig. 4e) or a web page (Fig. 5a). The detailed information of a tooltip is built using iARVis's component system, which indicates it has the same ability of interactivity. We also provide the selection interaction when using visualization techniques such as SPLOM (Fig. 5c), which allows users to select a range of data points and display the summary information.

We also provide a set of common properties to control the appearance and layout of each visual element, such as padding size, background color, and corner radius. One important common property is `onTap`, which can bind actions to visual elements, and we provide basic actions such as presenting a dialog, opening a new visualization widget (Fig. 4e), and opening a web page (Fig. 5b).

Additional information about different types of components with their parameters and common properties is available on the documentation page.

3.2.2 Layout Components

Layout components are used to position visual elements. As shown in Fig. 3a, we provide 5 types of layout components: `HStack`, `VStack`, `Grid`, `Segmented Control` and `Spacer`.

With these basic layout components and visual element components, designers can make complex visualization widgets containing a variety of visual elements in different styles and layouts (Fig. 2d). Additional information about all the layout components is available on the documentation page.

3.3 Visualization Widget Automatic Positioning

One thing that we must consider after designing the interface of the visualization widget is positioning [58]. We provide several ways to predefine the 3D transformation of a visualization widget in augmented reality:

- **Object targeting**. A meaningful reason for using augmented reality is that we want to connect data visualizations with real-world objects. Using object detection [41], we can track the transformation of an object with a provided scanned model in the real world.
- **Image targeting**. Image detection [34] is much faster and more accurate than object detection; thus, if the targeting object

is an image, or is difficult to obtain the 3D model of the object, we can use image detection to retrieve the 3D transformation of the target objects.

- **QR code targeting**. If we cannot fulfill the requirements of both object tracking and image tracking, but we still want to place the visualization widget automatically, we can use a QR code to help users.
- **Freedom placing**. If we do not need to bind the visualization widget with any real-world objects, we can freely place the widget on a detected plane, or use gestures and the auxiliary control panel to directly adjust the 3D transformation.

As shown in Fig. 2b, we can simply specify a relative transformation of the visualization widget to the tracking target with an anchor point to determine the 3D transformation in the real world. When we first enter the augmented reality environment, we only need to use the predefined information in the JSON specification to determine the transformation of the visualization widget. The design of the declarative positioning also provides an opportunity to design persistence and continuity, which will be discussed in Sect. 4.

3.4 Digital Presence

Because display techniques such as the layout and size of 3D widgets are complicated to deal with in augmented reality [42, 47, 56] and because various users are comfortable with different levels of digital presence (presence of a digital component) [28], we also provide an option to bring widgets and other information to 2D screen space.

As shown in Fig. 4d, we can bring a visualization widget from the 3D augmented reality space to the 2D screen space by tapping the expand button on the corner of the visualization widget, to explore the widget in a larger size with more information and more accurate interaction. Small tooltips on a visualization chart can also be expanded and shown in the 2D screen space to provide full information (Fig. 4d). Videos, images, and web pages (Fig. 5b) should be displayed in the 2D screen space for a better viewing and interacting experience.

With the support of changing the digital presence of some components, users can have the most comfortable augmented reality-based information visualization experience.

3.5 User Preferences

iARVis supports changing the configuration during the visualization analysis process to fit the user's preferences, and the new configuration can be saved and shared with other users at a later time.

By offering a 2D panel to alter the encoding provided by the iARVis specification, such as changing the color mapping of the data, we offer a way to modify the visual encoding of the visualization chart. As shown in Fig. 1f, we can change the color mapping of the data to match the user's preferences. We can also, change the 3D transformation of the visualization widget, such as changing the anchor point, relative position and size of the visualization widget using gestures or the auxiliary control panel (Fig. 3c).

3.6 Authoring Efficiency

Efficiency is important for augmented reality-based visualization development and prototype validation. We provide features such as different levels of creation and hot-reload to help users create and verify the visualization widget quickly.

3.6.1 Two Levels of Creation for Users in Different Levels

We have designed iARVis to be accessible to novice users, using JSON as the specification format of iARVis, which is straightforward to understand. For more advanced users, we also provide an advanced way to create iARVis specifications programmatically, which is easier, more powerful, and more flexible than directly writing JSON specifications.

Novice Users. The JSON format is human-readable, and we can provide a JSON schema to create an intelligent text editor with the auto-completion feature. Users can refer to the documentation to learn about all the configurations and components, and then create an iARVis specification with the help of auto-completion.

Advanced Users. For advanced users, we provide a way to create iARVis specifications programmatically. Because all configurations are bidirectional JSON codable, we can create the data model of iARVis specifications programmatically, and then encode it to JSON to create the specification. The creation of the data model requires some knowledge of the Swift programming language [52], but it is much easier and faster than creating a JSON specification from scratch with the programming language level autocompletion.

3.6.2 Hot-reload

Creating and verifying an AR-based visualization environment prototype is tedious and involves different tools and devices [15]; thus, we want to optimize the traditional coding, building, deploying, and verifying process to improve efficiency. Because we use JSON format to describe the environment and parse the JSON content to create the environment, we can achieve the hot-reload effect by simply observing the changes in the JSON.

Thus, we can simplify the procedure to, **1)** specify the specification of the environment, **2)** use a mobile device to verify the prototype, **3)** modify the specification to justify the prototype, **4)** the environment updates automatically and we can verify the prototype again.

With the hot-reload feature, we can easily create, modify and verify the visualization environment without any building and installing procedures.

4 Persistence and Continuity

We will now discuss the storage, distribution, persistence, and continuity of augmented reality-based visualization tasks. For web-based data visualization tasks, visualization storage and distribution are easy to implement, for example, we can generate static images for visualizations, and deploy them to a web server. Currently, a better method for web-based data visualization is to use visualization libraries(e.g., d3.js, Vega-Lite) to generate Canvas or SVG-based visualizations with JavaScript codes and render them on the fly.

For augmented reality-based visualization tasks, storage and distribution are more complex because we must recover both the visualization itself with its configuration and its transformation relative to real-world objects. In the field of augmented reality, the term **Persistence** is used to describe the restoration of 3D scenes when revisited [27]. For data visualization tasks, we propose the term **Persistence and Continuity** to describe the seamless restoration experience of the data visualization process and analysis in augmented reality.

4.1 Storage and Distribution

First, the key difference between web-based data visualization and AR-based data visualization is that the latter is located in a virtual 3D space, and virtual objects may have connections to real-world objects, which can lead to problems when creating the visualization environment. Formally storing the connections between the real world and virtual objects and providing an entrance to access them is very important for AR-based data visualization.

We follow these steps to create an AR environment for data visualization to determine what information we must serialize. First, we must **provide an entrance to access the AR environment**. We can provide a QR code with the JSON specification(or a URL) near an object to prompt the user to access the iARVis AR environment. Second, we must **determine the 3D transformation of visualization widgets**. We have discussed several ways to predefine the 3D transformation of a visualization widget in AR (Sect. 3.3). We can

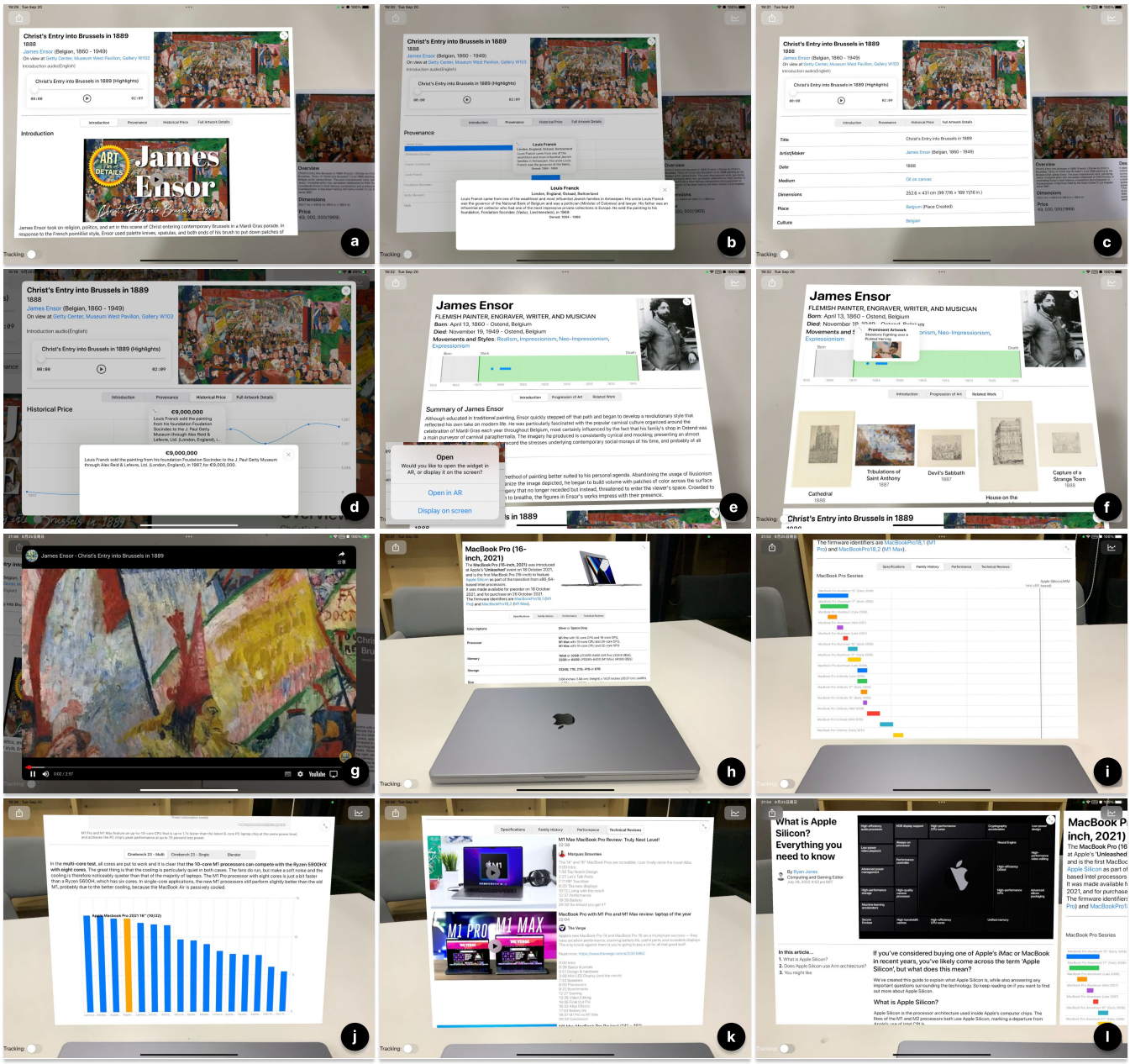

Figure 4: **a)** Display a visualization widget near an artwork. **b)** A timeline chart showing the historical provenance of the artwork. Hovering or tapping near a certain timeline bar can toggle the tooltip that describes more detailed information. **c)** A specification table about the artwork. **d)** Digital presence: A visualization widget can be expanded to the 2D screen space to improve accessibility. **e)** Open a new visualization widget about the author of the artwork after tapping a super-hyperlink. **f)** A grid of all the artworks of the author. **g)** Embed a YouTube video to introduce the author. **h)** Display a visualization widget near a MacBook Pro. **i)** A timeline chart showing the evolution of the MacBook Pro product line. **j)** A bar chart comparing the performance of the MacBook Pro and other laptops. **k)** A list of technical reviews of the MacBook Pro. **l)** Open a new visualization widget about the new Apple Silicon chip after tapping a super-hyperlink.

simply specify a relative transformation of the visualization widget to the target with an anchor point to determine the 3D transformation in the real world and store the information in the JSON specification. Third, we must **declare the user interface and interaction of visualization widgets**, which has been discussed in Sect. 3.2. Because the design grammar itself is based on JSON, we can simply store the declaring detail in a JSON field (Fig. 2c, Fig. 2d).

We can easily distribute the JSON specification to a server and access them through a URL, or generate an `iARVis://` URL with JSON specification encoded. Distributing the configuration to the server can make the data visualization experience dynamic, for example, we can update data over time, correct invalid data, or even change the entire visualization environment.

4.2 Persistence and Continuity

When reviewing the literature on augmented reality with data visualization processing and analysis, we noticed a rarely noted but important problem: every time we enter the augmented reality environment, we need to rearrange all the scene content, and the previous working environment has been lost [11, 50]. For example, we need

to reposition the visualization widget near the object it is related to, alter the visual encoding of the chart, etc.

We define **persistence and continuity** as: if the user leaves the AR environment and revisits it later, the environment will be restored to the previous state, including positioned visualization widgets with their transformation and visual configuration. We've carefully analyzed the situations that may be encountered every time we re-enter the AR environment and propose possible solutions of different granularities to achieve persistence and continuity. By leveraging the storage and persistence abilities, iARVis supports saving the visualization environment and sharing with other users.

4.2.1 World map

ARKit [4] provides a way to serialize all the AR information into a single `ARWorldMap`, which contains all feature points, detected planes, anchors, etc., with a snapshot of the camera view from the time that data were saved. Besides `ARWorldMap`, we also need to persist widgets' transformations, visual configurations, and their binding relationships with anchors. Putting all these together, we can construct a world map, which contains all the necessary information to reconstruct the visualization environment.

4.2.2 Relocation

With the `ARWorldMap`, ARKit could attempt to relocalize to the new world map — that is, to reconcile the received spatial-mapping information with what it senses of the local environment. In general, this relocation procedure is using all feature points information stored in the `ARWorldMap` to match the physical environment and find out the relationships between old and new feature points. Once the relocation is done, the coordinate system is aligned with the coordinate system in the world map so that all previous anchors in the augmented reality space could be re-positioned.

4.2.3 Re-position and re-configure widgets

Once the coordinate system is re-aligned, we can further re-position the anchors saved in the `ARWorldMap`. An anchor (`ARAnchor` [3]) is an object that specifies the position and orientation of a feature point in the augmented reality space. We use a customized anchor to store: **1)** the JSON specification of the content of the widget, **2)** the transformation of the widget, **3)** the visual configuration of the widget. We can restore the widgets from anchors as soon as the world map is reloaded.

4.2.4 Different levels of persistence and continuity

We provide different levels of persistence and continuity to achieve the maximum degree of usability according to the stability of the physical environment. Table 2 shows how widget transformation and configuration can be restored using each level.

Level	Physical Environment	Widget Transformation	Widget Configuration
Level 1	Stable	Directly	Directly
Level 2	Dynamic	Tracking	Directly
Level 3	Dynamic	Dependent	Directly

Table 2: Different levels of persistence and continuity

Level 1, the most rigorous but the easiest to use level, could be used when the physical environment is stable, which means we can directly restore all information from the world map. Level 2, which is enabled by default, provides the most flexibility for most use cases by trying to track all targets again. Level 3 is used when the widget does not have a tracking target, in that case, we can bind the widget to the nearest anchor. All levels can be used concurrently as each widget can use different levels of persistence and continuity for the maximum degree of usability. For example, a portrait on the wall may never move, but the price tag placed nearby could be moved around. We can use the first level to store the widget related to the portrait itself and use the second level to store the widget related to the price tag.

We achieve the maximum possibility of persistence and continuity through these levels to help users gain the seamless and continuous experience of AR-based information and data visualization.

5 Example Applications

To evaluate iARVis and its practicality and extensibility, we build some typical application examples. Our examples focus on **1)** basic usage of iARVis for building the visualization environment using our declarative grammar (Sect. 5.1), **2)** augmenting static visualization using iARVis to enhance the visualization with interactivity and more information (Sect. 5.2), **3)** extending the current declarative language to introduce unsupported visualization techniques, such as SPLOM (Sect. 5.3). A typical iARVis application uses QR Code scanning to open the visualization environment, we can also consider other ways to open the visualization environment, such as image recognition, object recognition, and NFC.

5.1 Construction of the Visualization Environment

The most important advantage of iARVis is that it provides a simple declarative syntax to build augmented reality-based visualization environments with ease. We will now present some example applications built with iARVis.

5.1.1 Build a visualization environment near a static image

Scenario description We consider a typical exhibition scenario where paintings by different authors are displayed in a gallery and visitors can observe the painting they are interested in, learn about the history of the painting, and learn about the artist. Due to site size and cost constraints, not all paintings can show sufficient information. For example, the introduction of the author may not be repeatedly displayed if their paintings are placed in different locations, and there is not sufficient space to display the provenance and historical price information of the painting.

Example Using iARVis, we can easily build a visualization environment that displays additional information near the painting to eliminate spatial limitations and provide interactivity. For example, we can place a widget that contains the introduction to the painting with text, audio, video, and a table near the painting using image targeting, as shown in Fig. 4. We list the basic specification details of the artwork, such as the medium, dimension and culture type, in a table, as shown in Fig. 4c. We can use a timeline chart and a line chart to visualize the historical provenance (Fig. 4b) and price (Fig. 4d) of the painting. We provide some interactions such as hovering and tapping to provide detailed information in a tooltip about the provenance and price of the painting, which can also be displayed in 2D screen space for better readability, as shown in Fig. 4b and Fig. 4d. We can also expand the visualization widget to the 2D screen space to provide better accessibility for the user, as shown in Fig. 4d. Since we fetch all the design information from the JSON specification from a URL, we can easily update the visualization environment, such as the provenance and price of the painting, without changing the client code.

By tapping a hyperlink targeting another visualization widget that the designer provides, we can also open an additional visualization widget that displays information about the painter, such as the introduction to the painter with text and video (Fig. 4e, Fig. 4g), a visualization chart about the painter's life and career (top of Fig. 4e), introduction to the progression of the painter's artwork, and a grid displaying the painter's paintings (Fig. 4f).

5.1.2 Build a visualization environment near an object

Scenario description We now consider a store selling digital products such as phones, tablets, and laptops, where customers can read

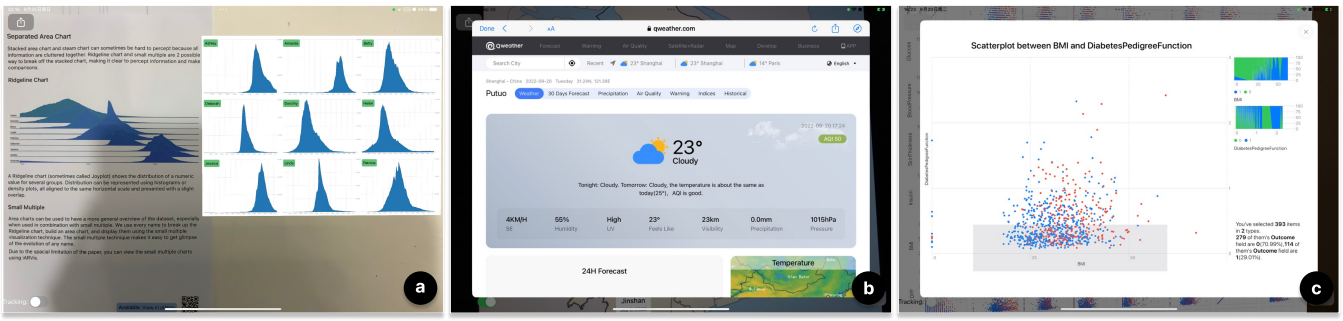

Figure 5: **a)** Use the small multiple technique to visualize the evolution of the baby name in the US. Due to the spatial limitation, we can use iARVis to display the visualization. **b)** We can open a web page through a hyperlink, to show detailed weather information. **c)** We can open a detailed scatter plot view by tapping one of the SPLOM cells. We can select a range of data points to know the data distribution.

a brief description placed near the product. Due to spatial and temporal limitations, information cannot be fully displayed and updated in real time.

Example With the object targeting capability of iARVis, we can easily build a visualization environment that displays additional information near the product object. We should pre-scan the target object using Reality Composer [45] to a `.arobject` file, which is used by `ARKit` to track the object or use a QR Code as the tracking target. As shown in Fig. 4, we build a visualization environment that introduces the latest 16-inch MacBook Pro. In addition, a specification table (Fig. 4h) provides basic information such as the color options, memory and storage options, size and weight, display and camera, etc. We also use a timeline chart to visualize the history of the MacBook Pro family (Fig. 4i), some bar charts and line charts to visualize the CPU and GPU performance of the MacBook Pro (Fig. 4j), and technical review videos from different YouTubers to provide more information about the product (Fig. 4k).

There has been a CPU architecture transition in the MacBook Pro family from Intel to Apple Silicon since 2020, which is shown in the timeline chart (Fig. 4i). If the user wants to know more about what is Apple Silicon, they can tap the highlighted Apple Silicon text, which is a hyperlink, to open a new visualization widget that introduces Apple Silicon, as shown in Fig. 4l.

5.1.3 Build a visualization environment near existing charts

Scenario description Visualizations require a lot of space on paper but we typically do not have sufficient space to display all visualization concurrently [15]. Researchers use techniques such as small multiple [15] to augment existing static visualization using AR to display all the visualizations near the original visualization.

Example With iARVis, we can easily achieve a similar effect by placing the visualization widget near the tracked visualization. We now consider writing an essay that tells a visualization story about the evolution of baby names in the US from 1880 to 2015. The publisher requires the essay to be less or equal to four A4 pages long, which cannot display all relevant visualizations. We thus provide a QR Code that points to an iARVis specification with a hint. When the user scans the QR Code, iARVis will load the specification and build the visualization environment, showing a matrix of area charts using the small multiples technique, as shown in Fig. 5a.

5.2 Augment Static Visualization

Besides building the visualization environment easily, iARVis can also enhance existing static visualization to provide more information and interactivity.

5.2.1 Augment static visualization by overlaying

Scenario description The static limitation of paper visualizations prevents the possibilities of various visualization techniques, a simple example is that when visualizing the weather status of an area on a map, we can only show the weather status of a certain day.

Example With iARVis, we can only display dynamic and real-time weather information on the map and can add interactive elements to the dynamic content. For example, we can deploy an environment specification to a server and keep it up-to-date, so that iARVis can fetch and display the latest weather status while providing interactions such as displaying the weather details of a certain area by tapping. As shown in Fig. 1c, we use iARVis to display the real-time weather status of Shanghai on a district map. The user can tap on the brief weather status card of a district to open a web page displaying the detailed weather status of the district (Fig. 5b).

5.2.2 Augment static visualization by extending

Scenario description The static visualization chart can easily be out of date because it is difficult to update the visualization chart once it has been printed and distributed. For example, a bar chart that shows the historical daily weather status of a city could be out of date right after the weather report is published, and due to spatial limitations, we cannot display all the historical weather statuses.

Example Using iARVis, we can easily extend the existing visualization chart by displaying the continuous part to eliminate the temporal and spatial limitations. As shown in Fig. 1d, we can extend the original chart by putting the latest weather status on the right side, and the previous weather status on the left side.

5.3 Extend the Declarative Grammar

The design of the declarative grammar of iARVis is highly extensible; and we can keep adding more component supports to enhance the visualization ability, which provides various possibilities to support different visualization techniques.

5.3.1 Extend the grammar to support SPLOM

Scenario description The scatter plot matrix, known acronymically as SPLOM [29], uses multiple scatter plots to determine correlations between a series of variables. For example, we can use SPLOM to describe the relationship between diabetes and different predictor variables such as glucose, blood pressure, etc. However, SPLOM takes up a lot of space; thus each scatter plot is small particularly when it is printed on paper and the number of variables is large.

Example We can easily extend the declarative grammar to support the SPLOM technique by adding a new dedicated component with some necessary parameters (Fig. 3b). By merely defining the data source, field names, and the default configuration for each scatter plot, we can quickly construct SPLOM visualizations without any code. The SPLOM technique helps us easily perceive the correlation between two factors at a glance, and we can also tap on a scatter plot to open a new dedicated visualization widget displaying the scatter plot with additional information. For example, the detailed widget

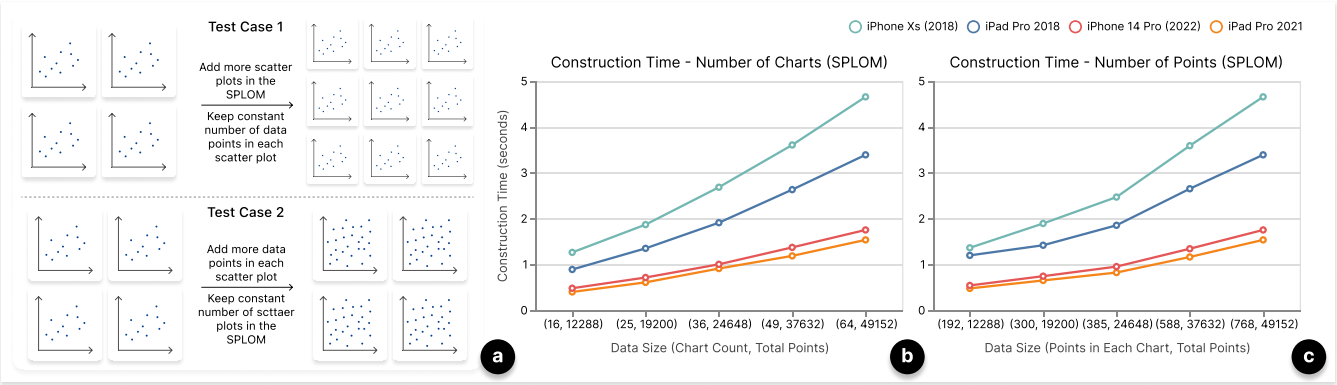

Figure 6: Performance Evaluation: **a)** Performance test scenario illustrations. **b)** Performance test case 1. **c)** Performance test case 2.

displays the data distribution and a brief description of the data in the selected range, as shown in Fig. 5c.

6 Evaluation and Conclusion

Besides example applications we implement, we also evaluate our system with a performance evaluation and expert reviews.

6.1 Performance Evaluation

iARVis aims at quick development and prototype validation, scalability is not an explicit design goal. This part reports on performance measures of the current implementation.

We use iARVis to build environments in different scales of data and view hierarchy, and observe the construction time of the environment with the frame rate of the scene, on devices produced from 2018 to 2022. The result indicates that the construction time of iARVis is highly related to the scale of data in charts, but not influenced much by the complexity of the view hierarchy. The frame rate of iARVis is stable at 60 FPS after the construction process.

Based on the observation, we decide to test the construction performance further using the SPLOM example (Sect. 5.3.1) with different scales of data. We test the performance in two ways: **1)** increase the number of scatter plots in the SPLOM from 16 to 64, and each scatter plot has 768 data points (Fig. 6b, Fig. 6a). **2)** increase the number of data points in each scatter plot from 192 to 768, and the number of scatter plots is 64 (Fig. 6c, Fig. 6a). The total number of data points ranges from 12,288 to 49,152 in both test cases. We run each test case 5 times and calculate the average construction time for each device.

The results in two test cases reveal that the construction time of iARVis is highly related to the total number of points, no matter how many scatter plots are involved. All devices perform well if the total number of points is less than 10,000, which could accommodate most visualization scenarios in daily life. iPad Pro 2021 and iPhone 14 Pro can even construct the scene containing a SPLOM widget of about 50,000 points in less than 2 seconds, which could contain 64 scatter plots with 768 data points in each plot.

6.2 Expert Reviews

We invited 5 experts (visualization researchers with varying levels of experience with visualization toolkits such as Vega-Lite and d3.js from our institute, **E1 - E5**) to evaluate the proposed concepts and implementation. Overall, the experts were highly interested and positive about how to create an AR-based information and data visualization environment with simple JSON specifications. The experts mentioned that the 2 levels of creation (Sect. 3.6.1) can give both novice and expert users the ability to create visualization environments (**E1**, **E2**, **E4**), and the hot-reload (Sect. 3.6.2) can help users to quickly iterate and validate their ideas (**E1 - E5**). All experts agreed that the automatic positioning of visualization widgets is very useful, which allows users directly step into the analysis process (**E1 - E5**). A few of them also positively mentioned that the ability to change the digital presence can make visualization widgets more accessible, and they prefer to bring the widget to the 2D space when interacting with charts (**E3**, **E4**). The sharing experience is appreciated by most experts (**E1, E2, E4, E5**), who mentioned that the sharing ability works well with the persistence and continuity (Sect. 4) feature, giving users the ability to save the current working environment and share with others. And the persistence and continuity feature is important for creating a seamless analysis experience.

In addition to the design and implementation of iARVis, the experts also evaluated the example applications we created. They mentioned that examples of visualizing information about artworks (Sect. 5.1.1) and MacBook Pro (Sect. 5.1.2) could be typical daily life scenarios of AR-based information visualization (**E2**, **E3**, **E5**). The small multiple (Sect. 5.1.3) and SPLOM (Sect. 5.3.1) examples demonstrate the potential of iARVis to support complex visualization techniques and the ability to eliminate spatial constraints (**E1**, **E3**, **E4**). Examples of augmenting static data visualizations indicate another possibility of iARVis, providing dynamic and interactive content to the static visualization in some special scenarios (**E2**, **E4**).

Experts also addressed some critical points. The first point is the limitation of the field of view, which is a limitation of mobile AR devices (**E1**, **E3**). The second one is the fatigue when holding an iPad for a longer period, which is a common problem of both mobile and HMD AR devices (**E3**, **E4**). Also, experts requested the possibility of synchronizing the real-time collaboration for multiple sessions and 3D data visualization chart support (**E2**, **E4**).

6.3 Conclusion

With ARKit, iARVis makes prototyping mobile AR-based information and data visualization easy and quick. By providing declarative grammar and a collection of components, iARVis can generate code-free complex visualization environments with advanced features such as hot-reload, automatic positioning, and persistence and continuity, as well as fluent and seamless user experiences.

We believe that iARVis is an important step towards enabling general users to explore AR-based visualization in their daily lives, and we believe that the proposed concept can be extended in the future to support complex visualization techniques and advanced features, such as real-time collaboration and 3D data visualization.

Acknowledgments

This work was supported by the NSFC under Grant 62072183, the Shanghai Committee of Science and Technology, China (Grant No. 22511104600), and NSSFC under Grant 22ZD05. Changbo Wang and Chenhui Li are the corresponding authors.

REFERENCES

[1] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large data visualization. *The visualization handbook*, 717(8), 2005.

[2] I. J. Akpan and M. Shanker. A comparative evaluation of the effectiveness of virtual reality, 3d visualization and 2d visual interactive simulation: an exploratory meta-analysis. *Simulation*, 95(2):145–170, 2019.

[3] ARAnchor. `https://developer.apple.com/documentation/arkit/aranchor`. Accessed: 2022-07-22.

[4] ARKit. `https://developer.apple.com/augmented-reality/arkit/`. Accessed: 2022-07-17.

[5] B. Bach, R. Sicat, J. Beyer, M. Cordeil, and H. Pfister. The hologram in my hand: How effective is interactive exploration of 3d visualizations in immersive tangible augmented reality? *IEEE transactions on visualization and computer graphics*, 24(1):457–467, 2017.

[6] B. Bach, R. Sicat, H. Pfister, and A. Quigley. Drawing into the ar-canvas: Designing embedded visualizations for augmented reality. In *Workshop on Immersive Analytics, IEEE Vis*, 2017.

[7] J. Bertin. *Semiology of graphics*. University of Wisconsin press, 1983.

[8] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE transactions on visualization and computer graphics*, 15(6):1121–1128, 2009.

[9] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[10] D. Bowman, C. North, J. J. Chen, N. Polys, P. Pyla, and U. Yilmaz. Information-rich virtual environments: theory, tools, and research agenda. In *VRST '03*. doi: 10.1145/1008653.1008669

[11] P. W. Butcher, N. W. John, and P. D. Ritsos. Vria: A web-based framework for creating immersive analytics experiences. *IEEE Transactions on visualization and computer graphics*, 27(7):3213–3225, 2020.

[12] S. Butscher, S. Hubenschmid, J. Müller, J. Fuchs, and H. Reiterer. Clusters, trends, and outliers: How immersive technologies can facilitate the collaborative analysis of multidimensional data. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, p. 1–12. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3173574.3173664

[13] T. Chandler, M. Cordeil, T. Czauderna, T. Dwyer, J. Glowacki, C. Goncu, M. Klapperstueck, K. Klein, K. Marriott, F. Schreiber, and E. Wilson. Immersive analytics. In *2015 Big Data Visual Analytics (BDVA)*, pp. 1–8, 2015. doi: 10.1109/BDVA.2015.7314296

[14] Z. Chen, Y. Su, Y. Wang, Q. Wang, H. Qu, and Y. Wu. Marvist: Authoring glyph-based visualization in mobile augmented reality. *IEEE transactions on visualization and computer graphics*, 26(8):2645–2658, 2019.

[15] Z. Chen, W. Tong, Q. Wang, B. Bach, and H. Qu. Augmenting static visualizations with PapARVis designer. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12. ACM. doi: 10.1145/3313831.3376436

[16] CommonMark specification. `https://commonmark.org`. Accessed: 2022-07-09.

[17] M. Cordeil, A. Cunningham, B. Bach, C. Hurter, B. H. Thomas, K. Marriott, and T. Dwyer. Iatk: An immersive analytics toolkit. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 200–209. IEEE, 2019.

[18] M. Cordeil, A. Cunningham, T. Dwyer, B. H. Thomas, and K. Marriott. Imaxes: Immersive axes as embodied affordances for interactive multivariate data visualisation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, p. 71–83. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3126594.3126613

[19] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 135–142, 1993.

[20] C. Donalek, S. G. Djorgovski, A. Cioc, A. Wang, J. Zhang, E. Lawler, S. Yeh, A. Mahabal, M. Graham, A. Drake, et al. Immersive and collaborative data visualization using virtual reality platforms. In *2014 IEEE International Conference on Big Data (Big Data)*, pp. 609–614. IEEE, 2014.

[21] N. ElSayed, B. Thomas, K. Marriott, J. Piantadosi, and R. Smith. Situated analytics. In *2015 Big Data Visual Analytics (BDVA)*, pp. 1–8. IEEE, 2015.

[22] U. Engelke, H. Hutson, H. Nguyen, and P. de Souza. Melissar: Towards augmented visual analytics of honey bee behaviour. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 2057–2063, 2016.

[23] ARKit Face Tracking. `https://developer.apple.com/documentation/arkit/content_anchors/tracking_and_visualizing_faces`. Accessed: 2022-09-18.

[24] J.-D. Fekete. The infovis toolkit. In *IEEE Symposium on Information Visualization*, pp. 167–174, 2004. doi: 10.1109/INFVIS.2004.64

[25] P. Fleck, A. Sousa Calepso, S. Hubenschmid, M. Sedlmair, and D. Schmalstieg. Ragrug: A toolkit for situated analytics. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2022. doi: 10.1109/TVCG.2022.3157058

[26] L. Gallo, A. P. Placitelli, and M. Ciampi. Controller-free exploration of medical image data: Experiencing the kinect. In *2011 24th international symposium on computer-based medical systems (CBMS)*, pp. 1–6. IEEE, 2011.

[27] A. Guo, I. Canberk, H. Murphy, A. Monroy-Hernández, and R. Vaish. Blocks: Collaborative and persistent augmented reality experiences. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(3), sep 2019. doi: 10.1145/3351241

[28] F. Han, Y. Cheng, M. Strachan, and X. Ma. Hybrid paper-digital interfaces: A systematic literature review. In *Designing Interactive Systems Conference 2021*, pp. 1087–1100. ACM. doi: 10.1145/3461778.3462059

[29] J. Hartigan. *Clustering Algorithms*. A Wiley publication in applied statistics. Wiley, 1975.

[30] J. Heer, S. K. Card, and J. A. Landay. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 421–430, 2005.

[31] Y. Holtz. The issue with 3d in data visualization. `https://www.data-to-viz.com/caveat/3d.html`. Accessed: 2022-09-14.

[32] S. Hubenschmid, J. Zagermann, S. Butscher, and H. Reiterer. Stream: Exploring the combination of spatially-aware tablets with augmented reality head-mounted displays for immersive analytics. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3411764.3445298

[33] C. Hurter, N. H. Riche, S. M. Drucker, M. Cordeil, R. Alligier, and R. Vuillemot. Fiberclay: Sculpting three dimensional trajectories to reveal structural insights. *IEEE transactions on visualization and computer graphics*, 25(1):704–714, 2018.

[34] ARKit Image Tracking. `https://developer.apple.com/documentation/arkit/content_anchors/tracking_and_altering_images`. Accessed: 2022-07-20.

[35] P. Issartel, F. Guéniat, and M. Ammi. A portable interface for tangible exploration of volumetric data. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, pp. 209–210, 2014.

[36] V. Juřík, L. Herman, D. Snopkova, A. J. Galang, Z. Stachoň, J. Chmelík, P. Kubíček, and Č. Šašinka. The 3d hype: Evaluating the potential of real 3d visualization in geo-related applications. *Plos one*, 15(5):e0233353, 2020.

[37] O.-H. Kwon, C. Muelder, K. Lee, and K.-L. Ma. A study of layout, rendering, and interaction methods for immersive graph visualization. *IEEE transactions on visualization and computer graphics*, 22(7):1802–1815, 2016.

[38] R. Langner, M. Satkowski, W. Büschel, and R. Dachselt. MARVIS: Combining mobile devices and augmented reality for visual data analysis. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–17. ACM. doi: 10.1145/3411764.3445593

[39] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter. Dart: A toolkit for rapid design exploration of augmented reality experiences. *ACM Trans. Graph.*, 24(3):932, jul 2005. doi: 10.1145/1073204.1073288

[40] A. McNutt. No grammar to rule them all: A survey of json-style dsls for visualization, 2022. doi: 10.48550/ARXIV.2207.07998

[41] ARKit Object Tracking. https://developer.apple.com/documentation/arkit/content_anchors/scanning_and_detecting_3d_objects. Accessed: 2022-07-20.

[42] N. Polys and D. Bowman. Design and display of enhancing information in desktop information-rich virtual environments: challenges and techniques. 8(1). doi: 10.1007/s10055-004-0134-0

[43] N. F. Polys, S. Kim, and D. A. Bowman. Effects of information layout, screen size, and field of view on user performance in information-rich virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '05, pp. 46–55. Association for Computing Machinery. doi: 10.1145/1101616.1101626

[44] D. Raja, D. Bowman, J. Lucas, and C. North. Exploring the benefits of immersion in abstract information visualization. In *Proc. Immersive Projection Technology Workshop*, vol. 61, p. 69, 2004.

[45] Reality Composer app. https://apps.apple.com/us/app/reality-composer/id1462358802. Accessed: 2022-07-20.

[46] G. C. S. Ruppert, L. O. Reis, P. H. J. Amorim, T. F. de Moraes, and J. V. L. da Silva. Touchless gesture user interface for interactive image visualization in urological surgery. *World journal of urology*, 30(5):687–691, 2012.

[47] K. Satriadi, A. Cunningham, B. Thomas, A. Drogemuller, A. Odi, N. Patel, C. Aston, and R. Smith. Augmented scale models: Presenting multivariate data around physical scale models in augmented reality. 08 2022.

[48] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350, 2016.

[49] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):659–668, 2016. doi: 10.1109/TVCG.2015.2467091

[50] R. Sicat, J. Li, J. Choi, M. Cordeil, W.-K. Jeong, B. Bach, and H. Pfister. Dxr: A toolkit for building immersive data visualizations. *IEEE transactions on visualization and computer graphics*, 25(1):715–725, 2018.

[51] R. Skarbez, N. F. Polys, J. T. Ogle, C. North, and D. A. Bowman. Immersive analytics: Theory and research agenda. *Frontiers in Robotics and AI*, 6:82, 2019.

[52] The Swift Programming Language. https://swift.org/. Accessed: 2022-09-18.

[53] R. M. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. Vrpn: A device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, p. 55–61. Association for Computing Machinery, New York, NY, USA, 2001. doi: 10.1145/505008.505019

[54] B. H. Thomas, G. F. Welch, P. Dragicevic, N. Elmqvist, P. Irani, Y. Jansen, D. Schmalstieg, A. Tabard, N. A. M. ElSayed, R. T. Smith, and W. Willett. *Situated Analytics*, pp. 185–220. Springer International Publishing, Cham, 2018. doi: 10.1007/978-3-030-01388-2_7

[55] W. Usher, P. Klacansky, F. Federer, P.-T. Bremer, A. Knoll, J. Yarch, A. Angelucci, and V. Pascucci. A virtual reality visualization tool for neuron tracing. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):994–1003, 2018. doi: 10.1109/TVCG.2017.2744079

[56] Z. Wen, W. Zeng, L. Weng, Y. Liu, M. Xu, and W. Chen. Effects of view layout on situated analytics for multiple-view representations in immersive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):440–450, 2023. doi: 10.1109/TVCG.2022.3209475

[57] C. Wilke. *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. O'Reilly Media, 2019.

[58] W. Willett, Y. Jansen, and P. Dragicevic. Embedded data representations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):461–470, 2017. doi: 10.1109/TVCG.2016.2598608

[59] Y. Yang, T. Dwyer, B. Jenny, K. Marriott, M. Cordeil, and H. Chen. Origin-destination flow maps in immersive environments. *IEEE transactions on visualization and computer graphics*, 25(1):693–703, 2018.

[60] Y. Yang, B. Jenny, T. Dwyer, K. Marriott, H. Chen, and M. Cordeil. Maps and globes in virtual reality. In *Computer Graphics Forum*, vol. 37, pp. 427–438. Wiley Online Library, 2018.

[61] S. Zhang, C. Demiralp, D. F. Keefe, M. DaSilva, D. H. Laidlaw, B. D. Greenberg, P. J. Basser, C. Pierpaoli, E. A. Chiocca, and T. S. Deisboeck. An immersive virtual environment for dt-mri volume visualization applications: a case study. In *Proceedings Visualization, 2001. VIS'01.*, pp. 437–584. Ieee, 2001.

[62] M. Zhao, Y. Su, J. Zhao, S. Chen, and H. Qu. Mobile situated analytics of ego-centric network data. In *SIGGRAPH Asia 2017 Symposium on Visualization*, pp. 1–8, 2017.