

VizPilot: Automated Onboarding for SVG-based Composite Visualizations using Multimodal LLMs

Nishaanthini Gnanavel  and Yong Wang 

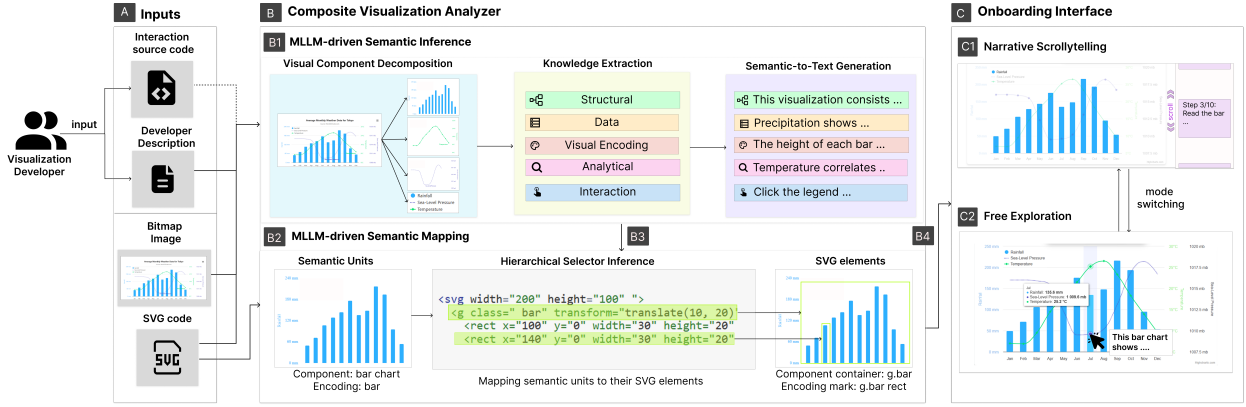


Fig. 1: **VizPilot Overview.** (A) *VizPilot* consumes multimodal inputs including SVG code, bitmap images, developer-provided description and optional interaction source code (example chart from Highcharts [19]). (B) The Composite Visualization Analyzer processes these via two stages: (B1) Semantic Inference decomposes the chart to extract knowledge and generate explanation text, producing (B3) intermediate semantic representation. (B2) Semantic Mapping then performs Hierarchical Selector Inference to map these units to relevant SVG elements. (C) The resulting (B4) mapped explanations power a dual-mode Onboarding Interface featuring (C1) Narrative Scrollytelling and (C2) Free Exploration.

Abstract—Composite visualizations integrate multiple visualizations in a meaningful manner and show strong expressiveness in representing complex dataset. They have been widely proposed by visualization developers in different applications. However, they often impose a high initial cognitive load on novice users, due to their intrinsic composite designs. Existing visualization onboarding approaches have attempted to mitigate this issue, but they are typically platform-dependent, require substantial manual authoring effort, and struggle with the structural complexity of composite visualizations, making them unable to be used for general composite visualizations. In this paper, we present *VizPilot*, a novel automated visualization onboarding approach that reverse-engineers composite visualization structure to enable an interactive onboarding experience directly from raw visualization artifacts. The proposed approach consists of two primary modules: the *Composite Visualization Analyzer* and the *Onboarding Interface*. Leveraging Multimodal Large Language Models (MLLMs), the Analyzer employs a two-stage pipeline: The *Semantic Inference* stage decomposes the visualization into individual visual components and extracts structured knowledge using a five category taxonomy to generate atomic explanation units, and the *Semantic Mapping* stage anchors these explanations to precise SVG elements via hierarchical DOM traversal, enabling reliable highlighting and interactions. We implement *VizPilot* as a browser extension. To initiate the workflow, a visualization developer provides a brief description and optional interaction source code and *VizPilot* then automatically extracts the SVG and generates bitmap image to drive the multimodal reasoning. Then, the interactive Onboarding Interface will show the structured outputs from the Composite Visualization Analyzer, which supports both *Narrative Scrollytelling*, providing guided, step-by-step explanations with synchronized highlighting, and *Free Exploration* for on-demand inquiry by users. We extensively evaluate *VizPilot* through a comparative analysis of different input modalities to assess the pipeline robustness, a usage scenario to illustrate reduced authoring effort, and a user study to confirm its effect in reducing users’ cognitive load. The overall results demonstrate the effectiveness and usability of *VizPilot* in facilitating an easy onboarding of composite visualizations.

Index Terms—Composite Visualization, Visualization Onboarding, Multimodal Large Language Models, SVG

1 INTRODUCTION

Composite visualizations have become a popular visualization design strategy in the past decades [9, 22, 60]. With the growing size and complexity of datasets, common basic charts like bar charts, pie charts and line charts are overwhelmed and unable to tackle complex analytics tasks in different applications. Accordingly, visualization researchers and developers have proposed various novel composite visualizations by integrating multiple visual representations in a coherent and meaningful

manner. Such composite visualizations can facilitate a comprehensive exploration of different data facets and their relationships, enabling advanced data analysis. A large number of composite visualizations have been presented in the field of data visualization and more are expected to be developed for different application domains in the future [9].

However, despite their expressiveness and popularity, novice users often encounter a significant cognitive barrier when starting to comprehend and use composite visualizations. Common questions, like what each component represents and how interactions between components work, will arise. Multiple visualizations can be combined in different ways such as juxtaposition, overloading, superimposition and nesting [22], and different visual encoding channels are used to represent complex and multivariate datasets as well as their relationships. Also, interactions are often enabled for composite visualizations, making it even more challenging for novice users to understand and interact

• Nishaanthini Gnanavel and Yong Wang are with Nanyang Technological University. E-mail: GN0001NI@e.ntu.edu.sg, yong-wang@ntu.edu.sg.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxx/TVCG.201x.xxxxxxx

with a composite visualization for the first time. For example, MatrixWave [57] (Fig. 3) is a composite visualization integrating multiple matrix visual representations to display event sequences, which requires users to simultaneously interpret matrix-based transitions, spatial layouts, and interaction-driven highlighting. PrettiSmart [49] (Fig. 4) is another composite visualization that juxtaposes multiple barcode-based designs to overview smart contract simulation results, and links each barcode-based design to a flow-based design to display transaction details. When first encountering customized composite visualizations, users must infer component encodings, relationships, and linked interactions, which is cognitively demanding. These challenges are particularly acute for visualization consumers (e.g., researchers, analysts, and domain practitioners) encountering customized composite visualizations without accompanying guidance. They typically rely on manual inspection or consulting the original authors. To better support these consumers, visualization developers need tools to generate onboarding materials.

Different visualization onboarding approaches [12, 41] have been proposed to help visualization users easily understand and use an unfamiliar visualization. Specifically, prior research on visualization onboarding has explored the manual creation of interactive tours via authoring [21], semi-automatic generation of guided tours [10], and a multimodal assistant to answer users’ questions [11]. Despite being effective in specific contexts, these approaches share three key limitations that hinder them from widespread adoption. First, these approaches require *substantial authoring effort* to enable visualization onboarding, such as manual specifications [59], predefined templates [10] and recorded interactions [21]. Second, they are typically *platform-dependent* and depend on proprietary APIs or structured metadata exposed by visualization tools such as Tableau or Power BI [11] [21]. Third, they predominantly focus on *basic charts or dashboards* consisting of basic charts. These three key limitations restrict their applicability to real-world visualization practice and make them unable to facilitate effective onboarding of sophisticated composite visualizations.

Visualization developers often create composite visualizations as customized SVG-based visualizations by using programming libraries like D3.js [5] and Vega-Lite [36]. SVG has become the most common format of composite visualizations. Since SVG is directly accessible in the browser DOM, it provides the foundation for possible platform-agnostic onboarding. Inspired by this, we propose *VizPilot*, an automated visualization onboarding approach implemented as a browser extension that requires no manual authoring effort. *VizPilot* comprises two main modules: the *Composite Visualization Analyzer*, a two-stage pipeline designed to bridge the semantic gap between low-level geometry and high-level intent, and the *Onboarding Interface*, which presents the resulting onboarding steps to users (Fig. 1). Within the *Composite Visualization Analyzer*, *Semantic Inference* decomposes the composite visualization, extracts structured knowledge, and generates atomic explanation units, while *Semantic Mapping* automatically maps these units to precise SVG elements. To initiate the process, a developer provides a brief description about the visualization and optional interaction source code and *VizPilot* then extracts the raw SVG code and bitmap image to perform MLLM-based reasoning. The Onboarding interface delivers both *Narrative Scrollytelling* for guided, step-by-step explanations with relevant element highlighting and *Free Exploration* for user-driven interrogation through hover-based inspectors and an AI-driven exploration assistant.

Given a SVG-based composite visualization, *VizPilot* automates visualization onboarding while still keeping developers’ agency. Developers can review and refine generated explanations directly on the interface without altering the underlying code, and export the finalized onboarding which can be seamlessly integrated into their existing project. At runtime on the end-user side, *VizPilot* dynamically overlays interactive onboarding interface onto existing visualizations, enabling users to interpret composite visualizations without any modifications to the underlying code.

Our main contributions are as follows:

- We propose *Composite Visualization Analyzer*, an effective tech-

nique to reconstruct composite visualizations from raw artifacts and produce semantically-meaningful explanation units for the corresponding visual elements of a composite visualization, i.e., a two-stage pipeline comprising *Semantic Inference* and *Semantic Mapping*.

- We develop *VizPilot*, a browser extension that leverages this Analyzer to drive an *Onboarding Interface*, where it automatically provides synchronized *Narrative Scrollytelling* and context-aware *Free Exploration*, allowing developers to deploy interactive onboarding without any manual authoring effort.
- We conduct extensive evaluations to demonstrate the effectiveness and usability of *VizPilot* in visualization onboarding for both visualization developers and users, which includes a comparative analysis of input modalities on inference accuracy, a usage scenario, a gallery of diverse examples and an in-depth user study with 16 visualization users. We have published our example gallery online: <https://vizpilot-gallery.onrender.com>.

2 RELATED WORK

This work is related to prior research on visualization onboarding, semantic understanding of visualization, and LLM-based visualization reasoning.

2.1 Visualization Onboarding

Visualization onboarding supports users in interpreting and extracting information from visual data representations [41]. Effective onboarding addresses knowledge gaps across domain, data, visual encoding, and interaction through structured storytelling [40–42]. Research suggests that users prefer in-situ scrollytelling over static videos, as it reduces cognitive load and the split-attention effect [44]. *VizPilot* adopts these proven principles to bridge those core knowledge gaps.

To operationalize these guidelines, recent work has explored semi-automated onboarding and narrative authoring tools. Dashboard systems such as Hey Dashboard! [11], DashGuide [21], and D-Tour [10] leverage dashboard metadata to generate guides. Other approaches support slideshow, scene-based, comic-style, scroll-driven, annotated and animated visual narratives [25, 30, 32, 35, 45, 46, 58]. LEVA [59] and the Authoring Tool for Data Journalists [43] instead rely on declarative grammars (e.g., Vega-Lite) to generate template-based explanations.

These approaches remain platform-dependent, authoring intensive, and limited in composite visualization support. *VizPilot* addresses these gaps by operating directly on SVG artifacts and leveraging MLLMs for multimodal semantic analysis, enabling a platform-agnostic pipeline that interprets composite visualizations without manual authoring or access to underlying specifications.

2.2 Semantic Understanding of Visualization

Semantic understanding recovers the logic, data mappings, and functional roles of graphical primitives, often using a multi-level taxonomy that decomposes a chart into structural components (e.g., axes, legends), visual encodings (e.g., marks and channels), and analytical insights. Prior work has approached this via heuristic-based structural analysis and, more recently, deep learning and generative models.

To recover these semantics from low-level primitives, several systems leverage the hierarchical structure of Scalable Vector Graphics (SVG). Mystique [7] deconstructs SVG charts into a structured representation (GREC) for layout reuse, while others use SVG tree structures as proxies for visual similarity [29]. Similarly, SeeChart [1] analyzes D3-based charts to generate accessible audio summaries. However, these rule-based approaches are generally limited to predefined chart types and struggle to generalize to structurally complex composite visualizations that integrate multiple visualizations within a single interface.

Recent work has incorporated large-scale semantic datasets and generative models. VISANATOMY [6] introduced an SVG corpus and taxonomy for chart semantics, while subsequent work leveraged GNNs and LLMs for video narration [53], semantic extraction from infographic timelines [61], and reverse-engineering SVG visualizations into semantic representations [51].

A key gap remains as existing methods focus on isolated charts, missing the coordinated semantics and cross-component relationships of composite visualizations, and lack deterministic grounding for interactive highlighting. We address this with a multimodal pipeline that fuses hierarchical SVG structures and bitmap images to reconstruct component-level organization, mapping structured knowledge directly to SVG elements for precise, interactive onboarding.

2.3 LLM-Based Visualization Reasoning

Recent research has investigated the capacity of Large Language Models (LLMs) and Multimodal LLMs (MLLMs) to interpret and reason about visual data. Instruction-tuned systems like ChartLlama [17] excel at chart-based QA and summarization [13, 54], while systematic benchmarks across GPT-4, Claude, and Gemini on VLAT [26] and CALVI [15] show that MLLMs can interpret common encodings and identify high-level trends [4, 20, 23, 33].

Despite these advances, MLLMs often struggle with precise numerical extraction, subtle encoding distinctions, and identifying misleading visualizations [23, 33]. While structured prompting strategies like Charts-of-Thought [8] partially mitigate these issues through intermediate reasoning, models still face difficulties with precise quantitative reasoning and cross-view coordination when processing raw SVG representations [52]. These limitations highlight the fragility of current multimodal reasoning approaches without explicit structural support.

Existing methods focus mainly on single-chart understanding via static benchmarks, producing free-form text outputs with no structural alignment to graphical elements. *VizPilot* instead frames visualization interpretation as a constrained multimodal pipeline, decomposing composite visualizations and aligning semantic roles directly with low-level SVG elements to support interactive onboarding.

3 VizPilot: OVERVIEW

Our goal is to automatically generate onboarding guidance for composite visualizations without requiring manual annotations or access to the visualization specifications. Rather than relying on declarative grammars or platform-specific APIs, our approach reconstructs visualization semantics directly from raw visualization artifacts.

Figure 1 illustrates the overall workflow of *VizPilot*. Given a web-rendered visualization, *VizPilot* extracts the SVG code, generates the bitmap image. To further refine the semantic inference, visualization developers can provide a brief high-level description about the visualization and optionally upload the interaction source code. These multimodal inputs are processed through a pipeline that reconstructs the visualization’s semantic structure and generates mapped explanation units for onboarding.

The methodology consists of two main components:

- **Composite Visualization Analyzer:** A two-stage pipeline consisting of *Semantic Inference* to decompose composite visualizations and extract structured knowledge, and *Semantic Mapping* to map these inferred explanations directly to SVG elements for precise interactive highlighting.
- **Onboarding Interface:** A dual-mode onboarding that transforms this structured representation into an actionable user experience, supporting both guided *Narrative Scrollytelling* and context-aware *Free Exploration*.

Together, these components enable automated onboarding generation directly for SVG-based composite visualizations, while maintaining precise alignment between textual explanations and the underlying visual elements.

4 COMPOSITE VISUALIZATION ANALYZER

We formalize Composite Visualization Analyzer as a reasoning process that reconstructs the semantic structure of composite visualizations from raw artifacts to support onboarding. Prior work such as DataWink [51] shows that multimodal models can derive semantically enriched representations from SVG-based visualizations, but primarily focuses on reuse and template generation rather than supporting user

onboarding. However, interpreting web-based composite visualizations directly from raw artifacts remain challenging: SVG representations encode low-level graphical primitives without exposing high-level constructs such as views or analytical relationships, existing approaches often depend on platform-specific APIs which limits generalizability, interactive behaviours (e.g., filtering or brushing) are defined in scripts and are not evident from static artifacts, and generative methods often struggle to align explanations with visual elements.

To address these challenges, we adopt a knowledge-driven approach that decomposes a visualization into atomic explanation units. Rather than generating free-form descriptions directly, Composite Visualization Analyzer first constructs an intermediate semantic representation capturing visual components, knowledge types, and explanation order, which is then mapped to SVG elements for interaction and highlighting. Generating accurate onboarding cannot be achieved in a single stage because semantic interpretation and SVG element alignment require fundamentally different reasoning. We therefore decompose the analyzer into two stages (Fig. 1B): **Semantic Inference**, which reconstructs high-level concepts such as components, encodings, and interactions from multimodal inputs to produce structured explanation units, and **Semantic Mapping**, which maps these units to specific visual elements to enable precise highlighting and interaction during onboarding.

To support this two-stage reasoning process, the pipeline operates over multimodal inputs, including SVG code for hierarchical structure and element localization, rendered bitmap images to capture spatial layout and context missing from SVG code, author-provided descriptions to guide semantic interpretation and narrative focus, and interaction source code (e.g. D3.js) to help disambiguate interactions. Together, these inputs enable our approach to progressively transform raw visualization artifacts into mapped, structured explanation units for onboarding.

4.1 Semantic Inference

The semantic inference stage performs the core reasoning required to reconstruct the semantic structure of the visualization from the collected inputs. Instead of directly producing the free-form textual descriptions, this stage derives a structured intermediate representation that organizes the visualization into components and associated explanation units that can later be mapped to specific visual elements. To reduce hallucinations, reasoning follows a constrained multi-step prompting strategy in which each step’s output bounds the next. This stage is decomposed into three steps to reflect the hierarchical nature of visualization understanding: it must first identify what constitutes meaningful units in the visualization, then determine what each unit represents, and finally express this information in a form suitable for onboarding. Accordingly, it consists of (1) **Visual Component Decomposition**, which segments the visualization into logical visual components using spatial and structural cues; (2) **Knowledge Extraction**, which derives structured semantic information for each visual component based on a predefined taxonomy; and (3) **Semantic-to-Text Generation**, which converts this structured knowledge into concise, atomic explanation units (Fig. 1B1). These three steps are executed within a single structured prompt to enable joint reasoning.

Visual Component Decomposition. Composite visualizations often contain multiple integrated visualizations, each with distinct semantic and functional roles, such as charts, legends, axes, or overview panels. SVG code provides low-level graphical primitives and hierarchical grouping, but it does not explicitly encode high-level component boundaries or inter-component relationships. To address this, the Component Decomposition step identifies and organizes these visual structures into meaningful *visual components* by combining information from the rendered bitmap image and the SVG structure to perform decomposition. Spatial layout cues from the image, such as the relative positions, adjacency and alignment, are integrated with the SVG’s hierarchical groupings to cluster low-level elements into coherent visual components.

We define a visual component as a coherent visual structure that serves as the fundamental unit of decomposition for subsequent knowl-

Table 1: Types of knowledge and their inference objectives.

Knowledge Type	Inference Objective
Structural Knowledge	Captures the overall layout, hierarchical relationships, and shared structural components such as axes or legends.
Data Knowledge	Extracts the underlying domain semantics, encompassing the specific variables, units, and categorical contexts represented within the visualization.
Visual Encoding Knowledge	Explicates the specific mappings between data attributes and graphical primitives (e.g., position, colour, size) within the decomposed components.
Analytical Knowledge	Synthesizes data trends, comparisons and inter-component relationships to articulate how multiple components coordinate to generate insights.
Interaction Knowledge	Deduces the supported interactions (e.g., hover, click, brush, zoom), explaining the analytical purpose of these user actions.

edge extraction and explanation generation. A visual structure qualifies as a visual component only if it independently supports the extraction of at least one distinct form of *visualization knowledge* [41] that will be introduced in Table 1 and the subsequent paragraph *Knowledge Extraction*. For example, an individual axis tick is not considered a visual component because it does not contribute independently extractable knowledge beyond that provided by the axis. Also, we do not consider basic visual marks (e.g., a bar in a bar chart) as visual components, because they can share redundant, near-identical visualization knowledge. For example, all the bars in a bar chart will share the same visual encoding knowledge. It is possible that one visual component can contain multiple other visual components. For example, in the PrettisMart visualization (Fig. 4), the simulation panel (Fig. 4A1) constitutes a higher-level visual component that contains multiple simulation thumbnails (Fig. 4A2), each of which can also be treated as an individual visual component. Visual components are categorized into two types. *Data-driven visual components* are coherent visual structures that represent analytical representations of the underlying data that require an independent explanation. Separate data-driven components sharing the same coordinate system are treated independently when they encode different information that requires distinct explanations. *Auxiliary visual components* such as legends, axes, and overview panels provide contextual or reference information that supports interpretation and independently contribute extractable knowledge. For instance, the state legend of the PrettisMart visualization (Fig. 4A3) can be treated as an auxiliary visual component.

This decomposition enables *VizPilot* to reason about each visual component independently, facilitating targeted knowledge extraction and the generation of atomic explanation units. By mapping subsequent analysis in well-defined components, *VizPilot* reduces ambiguity and supports multi-component co-ordination. Further justification for the visual component decomposition strategy is provided in Appendix A.5.

Knowledge Extraction. Without an explicit visualization grammar, it is difficult to programmatically determine what data is shown, how it is encoded, and what analytical meaning it conveys. To prevent hallucinations or generic descriptions, this step enforces a strict five-category knowledge taxonomy derived from visualization onboarding literature [41]: Structural, Visual Encoding, Data, Analytical and Interaction knowledge. As summarized in Table 1, the model evaluates each decomposed visual component and extracts relevant entities, classifying them into the predefined knowledge categories.

Furthermore, interactive behaviors (e.g., brushing and linking) are invisible in the static SVG files, which makes it impossible to infer the interaction knowledge of a composite visualization from the SVG files and unable to provide onboarding explanations for interactions. To address this, we selectively incorporate interaction source code into the context only to infer the existence and intent of interactions. MLLM is instructed to abstract away the implementation details to ensure that the

interaction knowledge is expressed solely from the user’s perspective.

This process produces a set of structured visual component-level knowledge units, where each unit describes the visual component type, associated data, visual encodings, and inferred interactions. These units produce a semantically rich representation that serves as input for the subsequent Semantic-to-Text Generation step, ensuring explanations are accurate, user-centric, and aligned with the visualization.

Semantic-to-Text Generation. Once structured knowledge has been extracted for each visual component, this step converts this information into concise explanation units suitable for onboarding. This step addresses the challenge that standard MLLM outputs tend to be overly long, mix multiple knowledge categories, and lack suitability for stepwise, progressive disclosure in composite visualizations.

To mitigate this issue, this step generates *atomic explanation units*, each describing a single concept associated with one view and one knowledge category. By constraining each unit to exactly one sentence and one knowledge type, the pipeline ensures clarity and cognitive accessibility for first-time users. This rendering process also enforces a hierarchical presentation order to guide users logically from understanding the layout and components of the visualization to interpreting data, encodings, analytical insights, and interactions, supporting a structured learning experience.

The resulting output is an ordered sequence of atomic explanation units with each corresponding to a visual component, and a concise explanation text. This intermediate semantic representation serves as the input for the semantic mapping stage, ensuring that textual explanations can be reliably mapped to visual elements for interactive onboarding.

4.2 Semantic Mapping

While Semantic Inference produces structured explanation units, these units remain abstract until they are visually anchored to the corresponding elements in the visualization. It is non-trivial to build the mapping between structured explanations and visualization elements, as raw SVG code often contains generic groupings, inconsistent naming, and deeply nested structures, making the direct alignment difficult.

We address this with *Semantic Mapping*, a mapping stage that links explanation units to SVG elements using the intermediate representation and SVG code (Fig. 2A). Rather than relying on zero-shot generation, we designed a specialized prompting strategy that explicitly instructs the MLLM to act as an expert DOM parser. Our key contribution is a *Hierarchical Selector Inference* mechanism where instead of directly outputting element IDs, the MLLM is prompted to construct selector paths from parent containers to target nodes, enabling deterministic and interpretable grounding. The semantic mapping here operates at two levels of granularity:

- **Component-Level Mapping:** The MLLM first identifies the macro-structure, mapping structural descriptions to the correct bounding container (e.g., the top-level <g> defining a specific chart panel).
- **Encoding-Level Mapping:** Constrained within the resolved container, the MLLM then links fine-grained explanation units to specific visual elements (e.g., data marks, legends, or axis ticks), supporting accurate highlighting and interaction.

During this process, the MLLM resolves ambiguities in mapping by leveraging contextual information such as component hierarchy, relative positioning, and layout structure to disambiguate candidate elements. For the nodes without unique identifiers, our approach generates hierarchical selectors to enable deterministic targeting, and favors selectors that uniquely identify elements to avoid ambiguous mappings.

Since MLLMs are probabilistic, relying solely on their output for UI targeting is unsafe. All inferred selectors are therefore passed through a programmatic multi-level verification step (Fig. 2C). First, selectors are resolved against the SVG DOM to ensure **DOM Validity**. Next, retrieved elements are checked for **Component Containment**, verifying consistency with the predicted component decomposition. Finally, **Layout Consistency** is enforced by comparing the spatial extent of mapped elements with the bounding regions of candidate

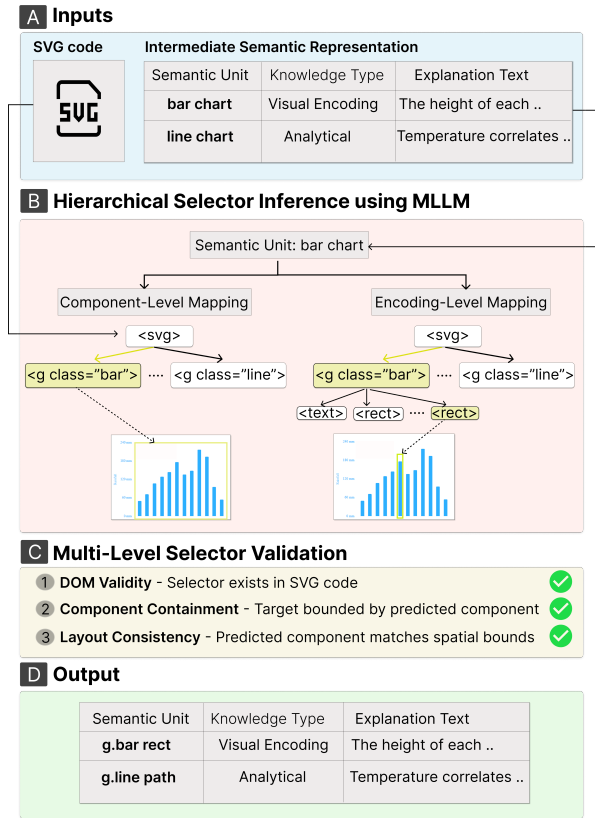


Fig. 2: **The Semantic Mapping Stage.** *VizPilot* maps each semantic unit from the (A) Intermediate Semantic Representation into precise SVG elements. (B) An MLLM proposes candidate DOM selectors through a hierarchical selector inference for both component-level and encoding-level targets, which are then (C) verified against a multi-level selector validation. The resulting (D) mapped explanation units enable synchronized visual highlighting for the onboarding interface.

containers derived from the SVG layout. Selectors that fail any of these checks are rejected and regenerated.

By combining MLLM-based hierarchical inference with programmatic validation, this stage produces precise and verifiable mappings between abstract explanation units and SVG elements (Fig. 2D), enabling interactive onboarding features such as stepwise highlighting and hover-based exploration.

5 ONBOARDING INTERFACE

The Onboarding Interface operationalizes the structured output of the Composite Visualization Analyzer, transforming mapped explanation units into an interactive onboarding experience. While the Analyzer resolves what the visualization represents, the interface must determine how this knowledge is delivered to support effective onboarding. This introduces a tension between providing scaffolded guidance to reduce cognitive load and preserving user agency for flexible exploration, as purely guided or unguided approaches are each insufficient.

To address this, we adopt a dual-mode design that separates guided explanation from exploratory interaction. The **Narrative Scrollytelling Mode** presents explanation units sequentially with synchronized highlighting to build an initial mental model, while the **Free Exploration Mode** enables independent interaction supported by an Exploration Assistant that provides contextual, on-demand guidance. This design supports both structured onboarding and flexible analysis while maintaining alignment between explanations and visual elements.

Narrative Scrollytelling Mode. The Narrative Scrollytelling mode (Fig. 3A) serves as an initial guided introduction to composite visualization. Its primary cognitive objective is to reduce the feeling of overwhelm by establishing the global structure and giving the user

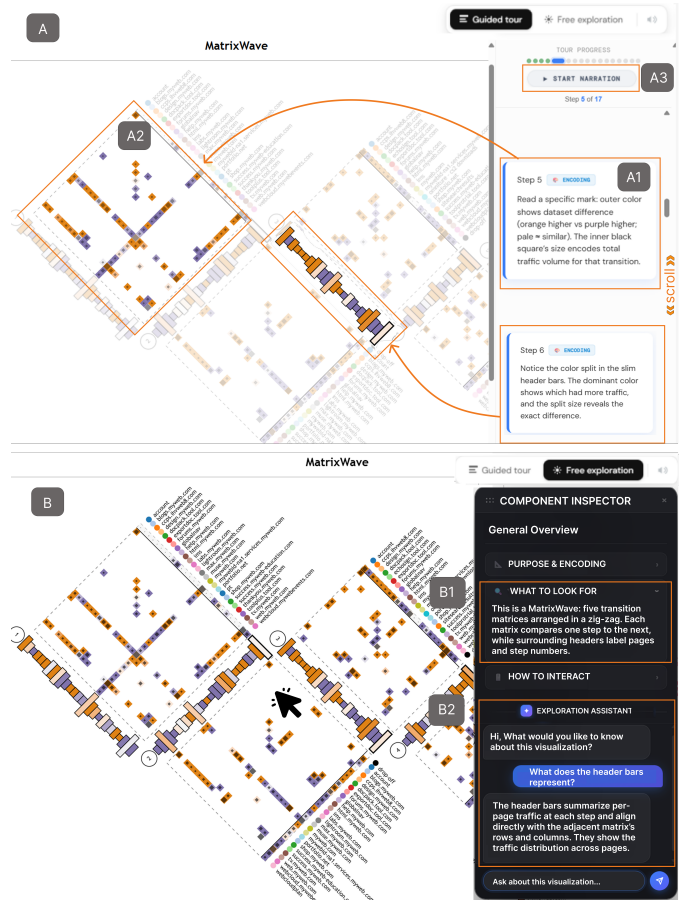


Fig. 3: **Onboarding Interface in MatrixWave [57].** (A) Narrative Scrollytelling provides a structured, linear guide where each narrative step (A1) automatically highlights the relevant visual element (A2), accompanied by text-to-speech narration (A3). (B) Free Exploration supports user-driven inquiry where hovering or selecting a component triggers an immediate semantic overview (B1), which can be further investigated via the conversational exploration assistant (B2).

a start on where and how to start reading the composite visualization. This mode provides a progressive walkthrough using atomic explanation units presented as scroll-triggered steps where each step focuses on a single knowledge type (Fig. 3A1). To ensure high-fidelity grounding, the interface utilizes the mapped SVG selectors derived during the Semantic Mapping stage to dynamically highlight target elements while dimming unrelated visual components (Fig. 3A2). This precise spatial anchoring ensures that every narrative explanation is visually verified in the visualization. Onboarding steps are ordered hierarchically, where it first introduces visual component roles and data encodings before surfacing analytical trends or complex interactions. This automated scrollytelling delivers a low-friction, scaffolded entry into composite visualizations.

Free Exploration Mode. While Narrative Scrollytelling provides a structured introduction, users inevitably require non-linear inquiries to test hypotheses, revisit specific components, or seek deeper semantic clarification. To support this, this Free Exploration mode (Fig. 3B) allows users to interrogate the composite visualization autonomously. Hovering over or selecting a visual component retrieves its mapped semantic entity from the pre-computed representation, displaying a concise overview (Fig. 3B1). To support open-ended questions, a context-constrained chatbot (Fig. 3B2) is initialized with the component's role, knowledge type, interaction semantics, and rendered image. This multimodal framing anchors responses to pre-inferred semantics, preventing the model from hallucinating or reinterpreting the visualization.

6 IMPLEMENTATION

We implemented *VizPilot* as a browser extension, which operates on web-based SVG visualizations that can be accessed through the browser DOM. The browser extension automatically extracts the underlying SVG code and generates the bitmap rendering of the visualization to capture spatial layout context. The developer of a composite visualization is also allowed to provide optional interaction source code, which will be further analyzed by *VizPilot* to improve the accuracy of interaction extraction. While interaction source code is beneficial for recovering interaction semantics, *VizPilot* can still generate onboarding content from the SVG and bitmap inputs alone when no interaction source code is provided. This deployment model allows *VizPilot* to analyze the rendered composite visualizations in-situ, without requiring any modifications to the source code. To support practical application and workflow separation, our approach distinguishes between onboarding creation and usage through two workflows: the *Authoring Workflow* and the *Consumption Workflow*.

For the **Authoring Workflow**, visualization developers provide a brief description and optionally the interaction source code. *VizPilot* automatically extracts SVG code, generates bitmap image and leverages GPT-5 to generate the onboarding structure. Developers can review and refine the explanation units before exporting as a decoupled package, which consists of 1) a structured data file in json format with narrative steps and SVG selectors, and 2) a lightweight JavaScript file for the UI elements. A detailed usage scenario illustrating the authoring workflow and onboarding deployment is provided in Section 7.2. For the **Consumption Workflow**, the visualization is presented in a read-only setting with onboarding activated in the interface. Visualization users can either follow guided scrollytelling or explore the visualization freely. Optional Text-to-Speech (TTS) narration further enhances the accessibility of the visual explanations generated by *VizPilot*.

This design enables onboarding for composite visualizations without modifying their original implementation, while supporting a clear separation between authoring and consumption through a reusable, decoupled deployment model.

7 EVALUATION

We evaluate *VizPilot* through an input modality analysis to assess pipeline robustness, a usage scenario and gallery to demonstrate reduced authoring effort and generalizability, and a user study assessing comprehension and cognitive load.

7.1 Comparative Analysis of Input Modalities

We conduct a comparative analysis to isolate the specific contributions of SVG code, bitmap image, and interaction source code to the onboarding pipeline. To validate this, we manually inspect the outputs across four representative visualizations spanning distinct composite design patterns: PrettSmart, RuleMatrix [31], PonziLens [48], and Shanghai Index [3]. Table 2 summarizes the observed capabilities and limitations. To illustrate these trade-offs concretely, the detailed inference results for PrettSmart (PrettSmart overviews smart contract simulation results via barcode-based summaries, each linked to a flow-based diagram of transaction details) (Fig. 4) are presented below. The detailed results for the remaining three visualizations are provided in Appendix A.8.

SVG code only: While the raw SVG code captures structural boundaries and provides precise selectors, it lacks visual context, reducing output to geometric description. For example, without visual rendering, the model identifies the tan-colored blocks in the visualization as “tan blocks” rather than as “Payable Functions,” since the color to category mapping is only conveyed visually.

Image only: This supports high-level semantic reasoning and pattern recognition but lacks structural grounding. The descriptions rely on ambiguous visual anchors like “wide pink band” that cannot be mapped to the DOM elements, making the onboarding output unactionable for scrollytelling.

SVG code + Image: Combining these modalities enables coordinated layout identification and valid selector linking. Without interaction logic, *VizPilot* infers interaction behaviors from typical UI appearance rather than verified code. (e.g., assuming hover behaviors).

Table 2: Capabilities and limitations of different input configurations.

Input	Capabilities	Limitations
SVG code Only	Isolates individual components and extracts precise DOM selectors.	Fails to infer visual hierarchy or spatial relationships; focuses on low-level geometry.
Image Only	Demonstrates visual reasoning and identifies layout patterns.	Lacks syntactic precision; explanations remain ungrounded and unactionable.
SVG code + Image	Recognizes tiered layouts and grounds high-level explanations.	Does not fully verify interactive behavior.
SVG code+ Image + Interaction source code	Verifies event handlers and deduces action-consequence.	Minimal; Primarily dependent on input completeness and code availability.

SVG code + Image + Interaction source code: Incorporating interaction source code enables precise action-consequence reasoning. By interpreting event handlers directly, *VizPilot* generates guidance grounded in implementation logic, such as identifying that clicking a simulation thumbnail triggers the detailed Functions and Contract/Address view, shifting onboarding from static description to actionable assistance.

In summary, each modality is essential, where removing images sacrifices semantics, removing SVG negatively affects the DOM mapping accuracy, and removing interaction source code reduces interactions to guesswork.

7.2 Usage Scenario: Authoring Workflow for a Composite Visualization

To illustrate the authoring workflow, we present a scenario involving Alex, a visualization developer tasked with creating an automated onboarding experience for a complex composite visualization, PrettSmart (Fig. 4). While the visualization contains coordinated charts with rich analytical insights, its intricate structure and hidden interactions present a significant learning curve for first-time users.

Automated Generation and Mapping: Instead of manually coding step-by-step explanations and UI bindings, Alex simply initiates the *VizPilot* browser extension. By providing a brief high-level description and the interaction source code, Alex triggers the Composite Visualization Analyzer. *VizPilot* then automatically extracts the SVG code and generates the bitmap image to perform its two-stage pipeline: it first generates atomic explanation units via Semantic Inference, and then maps those units to precise SVG elements via Semantic Mapping.

Review and Refinement: Using the generated output, *VizPilot* produces onboarding content in narrative scrollytelling and free exploration modes. As Alex reviews the generated onboarding sequence, he observes that each explanation step highlights the relevant visual elements directly within the visualization (Fig. 4B). Alex then refines one explanation step using the edit option in the interface (Fig. 4C) without manually adjusting the underlying UI bindings.

Decoupled Deployment: After finalizing the sequence, Alex exports the onboarding specifications. *VizPilot* separates content from rendering logic via a decoupled architecture: a structured json file stores narrative steps and SVG selectors, and a runtime script file handles the UI elements. By including these files in his project, the runtime securely overlays the interactive onboarding onto the visualization, without requiring any modifications to the original visualization code.

7.3 Gallery

To further demonstrate the usefulness of *VizPilot*, we apply it to a diverse set of composite visualization examples¹. We curated 18 vi-

¹<https://vizpilot-gallery.onrender.com>

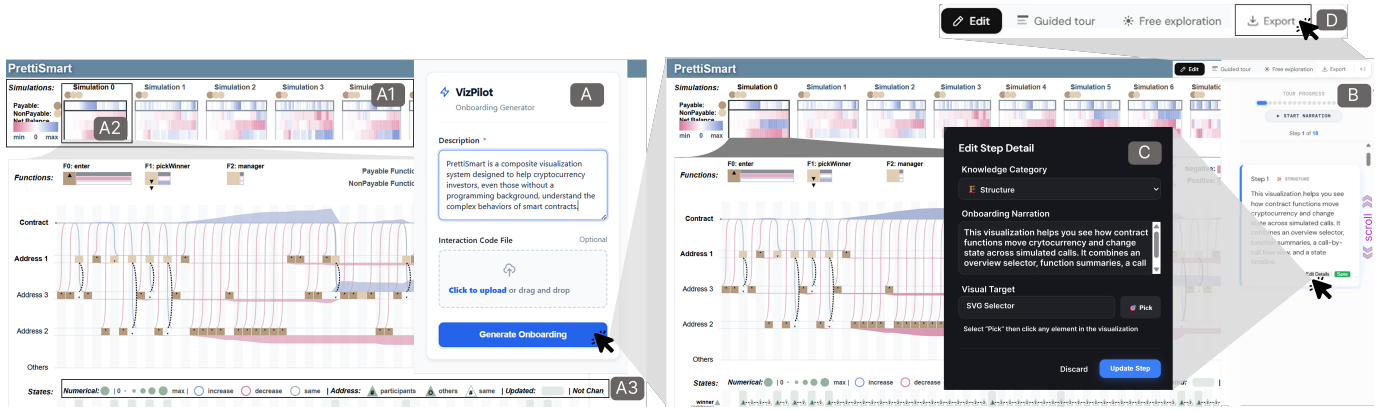


Fig. 4: **Usage Scenario: Authoring Workflow for PrettiSmart Composite Visualization** [49]. (A) The developer, Alex, provides the interaction source code and a brief description of the visualization. (B) The system automatically generates a mapped onboarding sequence, which Alex reviews for accuracy. (C) Alex refines a specific explanation unit via the edit option. (D) Finally, Alex exports the onboarding content to deploy the onboarding on the visualization.

Table 3: Pipeline robustness evaluation aggregated by composite design pattern across 18 gallery visualizations. Decomposition (Precision, Recall, F1); Semantic Mapping (Exact, Partial accuracy); Explanation Coherence (Type alignment, Component alignment, Atomicity); E2E: End-to-End Success Rate.

Composite Pattern	Decomposition			Semantic Mapping		Coherence			E2E
	Prec.	Rec.	F1	Exact	Partial	Type	Comp	Atom	
Juxtaposition	0.95	0.96	0.96	0.93	0.98	0.96	0.99	0.94	0.93
Overloading	0.96	0.88	0.92	0.89	0.95	0.99	0.97	0.86	0.83
Nesting	0.96	0.90	0.93	0.92	0.98	0.96	0.96	0.89	0.91
Superimposition	1.00	0.94	0.97	0.92	0.96	0.95	1.00	0.94	0.94

visualizations encompassing all composite design patterns identified in the prior work [22]: Juxtaposition (e.g., RuleMatrix [31]), Superimposition (e.g., ShangHai Index [3]), Nesting (e.g., PEARL [56]), and Overloading (e.g., VISTory [14]) (Fig. 5A). To quantitatively assess the robustness across this diverse gallery, we evaluated all 18 visualizations across three dimensions: component decomposition accuracy (Precision, Recall, F1), semantic mapping accuracy (exact mapping and partially-correct mapping), and explanation coherence (type alignment, component alignment, atomicity). Evaluation was conducted using an LLM-as-judge protocol [16] with three judge models, GPT-5.5, Gemini 3.1 Pro Preview, and Claude 3.5 Sonnet. Each of the model received the visualization bitmap, SVG code, and full pipeline output and assessed them against a structured rubric (Appendix A.6). Each judge reports a confidence score in [0,1] per criterion and the cases with a confidence score lower than 0.7 in any of the criteria were escalated to manual review (5 of 18) by the authors. Also, we reported the end-to-end (E2E) success rate that requires correct decomposition, mapping, and coherence jointly per ground-truth component. Table 3 reports the results aggregated by each composite design pattern. VizPilot achieves a strong performance consistently across all four patterns. The detailed results for each composite visualization can be found in Appendix A.6.

As a representative case, we illustrate the onboarding for EgoLines [55], a composite visualization that integrates continuous actor lines with discrete adjacency blocks for egocentric dynamic network analysis (Fig. 5B–D). VizPilot automatically generates a 13-step scrollytelling sequence: an initial structural step establishes the hybrid layout and temporal axis (Fig. 5B); the encoding steps selectively emphasize specific components such as actor lines while dimming surrounding context to mitigate visual clutter (Fig. 5C); and the analytical steps highlight complex visual patterns and suggest exploration strategies (Fig. 5D). A visualization user can either follow the guided tour using the Narrative Scrollytelling mode, or switch to Free Exploration mode for on-demand inquiries.

7.4 User Study

To extensively evaluate the effectiveness of VizPilot from an end-user perspective, we conducted a controlled user study. Specifically, we aimed to assess whether VizPilot improves visualization comprehension and reduces cognitive load when compared to a text-based baseline.

Participants. We recruited 16 participants (5 females, 11 males) aged 20 to 32 years ($M = 25.94$, $SD = 3.91$) from analytical disciplines including Computer Science, Data Science, and Mathematics. Participants reported frequent engagement with data visualizations, but only moderate prior familiarity with composite visualizations (7-point Likert scale: $M=4.75$, $SD=1.29$). Each one-on-one session was conducted in-person or via Zoom, with participants compensated 10 USD/hour. This study was approved by our institutional IRB, and informed consent was obtained from all participants prior to the experiment. Participants are anonymized as P1–P16 in subsequent analyses.

Design and Tasks. We adopted a within-subjects design with Latin Square counterbalancing to the mitigate ordering effects. Each participant answered four analytical questions on two composite visualizations, MatrixWave (Fig. 3) and PrettiSmart (Fig. 4), requiring understanding of structural hierarchy, visual encodings, data trends, and interactions. We selected MatrixWave and PrettiSmart because they represent distinct composite visualization structures and onboarding challenges. MatrixWave requires users to integrate information across coordinated matrix components, whereas PrettiSmart combines heterogeneous visual encodings and interactions across linked panels. The Baseline condition provided static text descriptions extracted from the corresponding papers for each composite visualization. The Experimental condition provided dual-mode explanations for each composite visualization.

Procedure. Each session comprised three phases and lasted about 60 minutes. During the initial 10-minute briefing, participants completed consent and demographic forms, and performed a practice walk-through of both conditions. In the 35-minute task execution phase, participants explored two composite visualizations under alternating conditions, answering four analytical questions and completing a post-condition Likert questionnaire after each condition. Finally, the session concluded with a 15-minute semi-structured interview to capture qualitative insights regarding participants’ sensemaking strategies and overall feedback.

Measurements and Analysis. To evaluate the effectiveness of VizPilot, we measured task accuracy and total task completion time (in seconds) during the question-answering phase, and utilized the Post-Study System Usability Questionnaire (PSSUQ) [27] to evaluate system usability and information quality, alongside the NASA Task Load Index (NASA-TLX) [18] to assess multi-dimensional cognitive load. Quantitative survey data was analyzed for statistical significance using the Wilcoxon signed-rank test [38]. We conducted semi-structured interviews centered around four open-ended questions: (1) Which condition

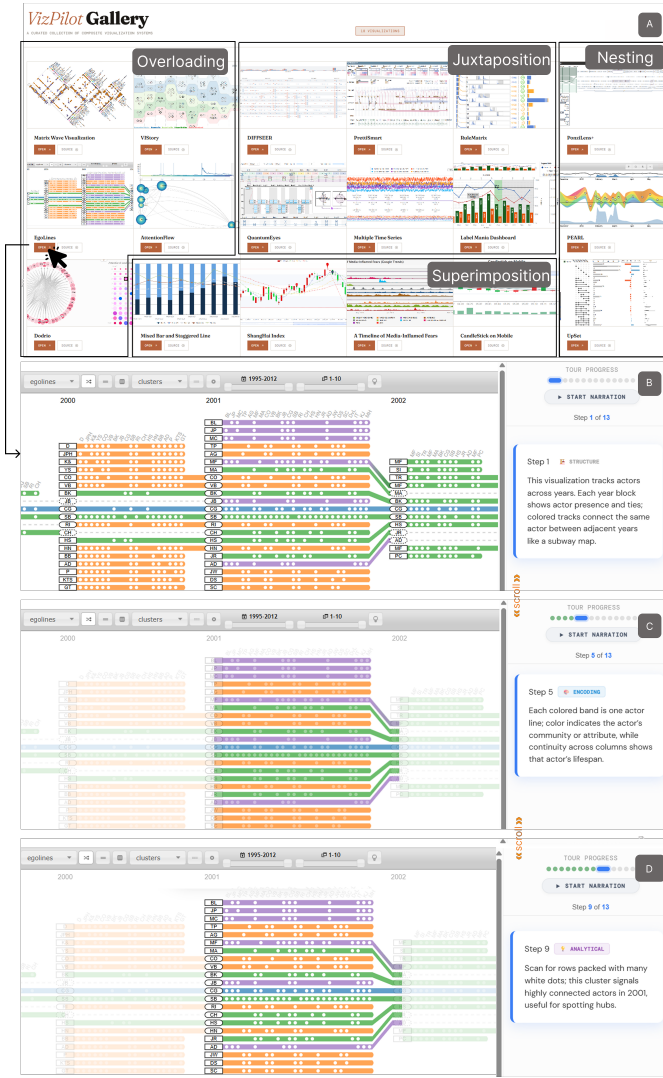


Fig. 5: **Gallery** (A) The *VizPilot* gallery demonstrates generalizability across diverse composite visualizations. A representative onboarding sequence illustrates our narrative scrollytelling mechanism: as the user progresses through (B–D) ordered explanation units, *VizPilot* dynamically highlights the focal component while dimming irrelevant views to mitigate visual clutter and reduce cognitive load.

better supported your understanding of the composite visualization, and why? (2) Which approach was more cognitively demanding, and what specific factors caused this? (3) Which specific features of *VizPilot* were the most helpful? and (4) What aspects of *VizPilot*'s interface or explanations were confusing or needed improvement? Responses were transcribed and analyzed via thematic coding.

7.4.1 Quantitative Results

Given the non-parametric nature of our ordinal survey data and paired task times, all statistical comparisons between the two conditions were conducted using the Wilcoxon signed-rank test.

Task Performance and Analytical Efficiency. We first examined whether the medium affected the users' ability to successfully extract insights from the visualizations. We observed a ceiling effect for task accuracy across both conditions. Participants successfully answered the analytical questions regardless of whether they used the text baseline ($M_{\text{base}} = 3.62, SD = 0.62$) or *VizPilot* ($M_{\text{viz}} = 3.75, SD = 0.45$). This indicates that detailed text descriptions are sufficient for task completion if users invest the necessary time. However, the temporal cost differed drastically. Participants using *VizPilot* ($M_{\text{viz}} = 194.38, SD = 92.42$) were significantly faster than those relying on the text baseline

($M_{\text{base}} = 262.86, SD = 150.79$), saving an average of 68 seconds, a statistically significant improvement ($p = 0.039$) confirming that our *VizPilot* accelerates insight extraction.

System Usefulness and Information Quality. We also examined participants' subjective perceptions of the onboarding experience using the Information Quality and System Usefulness dimensions of the PSSUQ. Participants reported significantly more favorable ratings for *VizPilot* across all selected items. The largest differences were observed for locating referenced visual elements (Q10: $M_{\text{viz}} = 6.44, SD = 0.81$ vs. $M_{\text{base}} = 3.06, SD = 1.57$; $p = 0.0004$), organization of information (Q12: $M_{\text{viz}} = 6.81, SD = 0.40$ vs. $M_{\text{base}} = 3.50, SD = 2.10$; $p = 0.0005$), and perceived usefulness and learnability (Q3: $M_{\text{viz}} = 6.00, SD = 0.89$ vs. $M_{\text{base}} = 3.44, SD = 1.67$; $p = 0.0005$; Q5: $M_{\text{viz}} = 6.81, SD = 0.54$ vs. $M_{\text{base}} = 3.38, SD = 1.93$; $p = 0.0005$). These results suggest a strong user acceptance of *VizPilot* and are consistent with the observed reduction in the task completion time. However, prior work has shown that subjective evaluations of AI-assisted systems can be influenced by expectation and novelty effects [24]. We therefore interpret these findings primarily as evidence of perceived utility and acceptance, while relying on the objective performance measures (task accuracy and task completion time) reported above as stronger evidence of effectiveness.

Cognitive Load. To better understand the mechanisms underlying the observed efficiency gains, we analyzed participants' NASA-TLX workload ratings (Fig. 7). Participants reported significantly lower cognitive burden when using *VizPilot* across several workload dimensions. The largest differences were observed in Mental Demand ($M_{\text{viz}} = 2.94, SD = 1.00$ vs. $M_{\text{base}} = 5.62, SD = 1.00$; $p = 0.001$) and Effort ($M_{\text{viz}} = 2.94, SD = 1.65$ vs. $M_{\text{base}} = 5.19, SD = 1.72$; $p = 0.007$). This result is consistent with the interpretation that the synchronized text and visual guidance reduced the need for participants to repeatedly switch between textual descriptions and visual elements, thereby mitigating split-attention effects during onboarding. Participants also reported substantially lower Frustration ($M_{\text{viz}} = 1.88, SD = 1.45$ vs. $M_{\text{base}} = 4.31, SD = 1.99$; $p = 0.001$) and higher perceived task Performance ($M_{\text{viz}} = 5.62, SD = 0.81$ vs. $M_{\text{base}} = 3.94, SD = 1.53$; $p = 0.003$) when using the onboarding by *VizPilot*. These subjective workload assessments are consistent with the objective finding (task completion time) that participants completed the tasks more quickly with *VizPilot*.

7.4.2 Qualitative Results

To contextualize our quantitative findings, we conducted a thematic analysis of the post-study interviews. Rather than focusing on overall preference for the onboarding approaches, we analyzed participant feedback to identify specific interface mechanisms that influenced how users interpreted and navigated composite visualizations. The analysis revealed two advantages of the onboarding of *VizPilot* in terms of cognitive support and onboarding structure, as well possible areas for future improvements..

Significant Mitigation of the Split-Attention Effect. Participants consistently described the baseline condition as requiring frequent shifts between textual descriptions and visual elements. A recurring challenge was maintaining textual information in working memory while searching for the corresponding visual components. As P7 noted, “*In the process of reading the text, I need to constantly wander through the pictures and text... which makes me very tired.*” Participants attributed the reduced effort in the onboarding of *VizPilot* primarily to the visual mapping provided by synchronized highlighting. These observations are consistent with the reductions in task completion time and perceived mental demand reported in the quantitative results.

Useful Dual-Mode Onboarding. Almost all participants highlighted the dual-model onboarding provided by *VizPilot*. Participants noted that the step-by-step presentation helped them focus on individual concepts clearly. P12 described the experience as an “*ordered step-by-step*” process, while P5 explained that fading irrelevant components “*prevented me from getting overwhelmed*” by reducing visual clutter. Participants also appreciated the transition from guided onboarding to free exploration mode, which enabled them to revisit concepts and

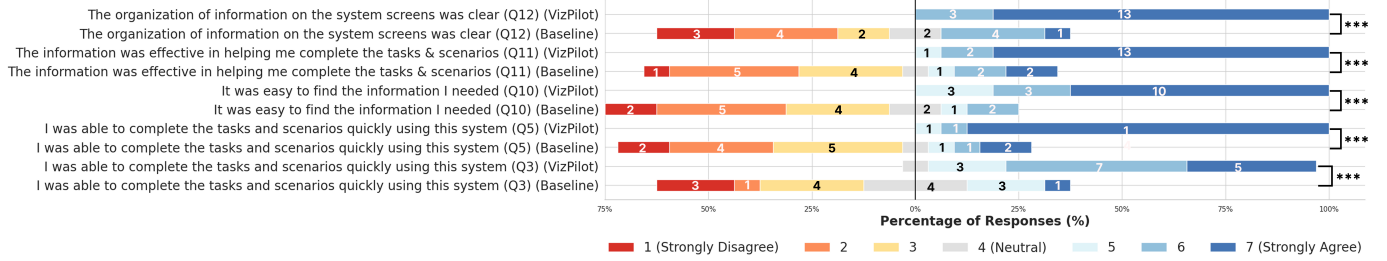


Fig. 6: **PSSUQ Results.** Results for selected PSSUQ items comparing the text baseline and *VizPilot*. *VizPilot* demonstrates a significant positive shift across both Information Quality Q10–Q12 and System Usefulness (Q3, Q5). Brackets with stars denote statistical significance based on the Wilcoxon signed-rank test (***: $p < 0.001$).

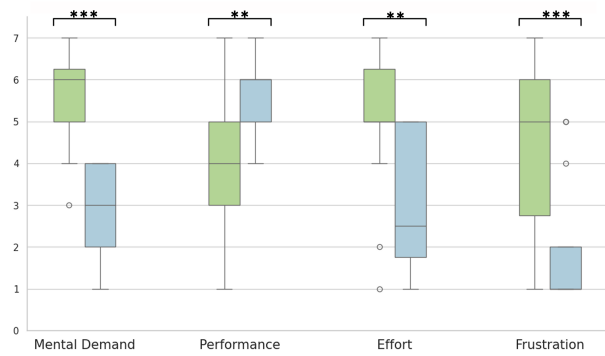


Fig. 7: **NASA-TLX Workload Assessment.** Box plots comparing the perceived workload across selected dimensions for the baseline (shown in green color) and *VizPilot* (shown in blue color). Except for Performance, lower scores indicate a more favorable, lower-friction experience. *VizPilot* significantly reduced mental demand, effort and frustration while improving perceived performance. Brackets with stars denote statistical significance based on the Wilcoxon signed-rank test (***: $p < 0.001$, **: $p < 0.01$).

inspect details at their own pace. As P8 explained, they could “*use the chat to check my understanding.*” These responses suggest that the combination of structured guidance and flexible exploration helped users navigate complex visualizations more systematically.

Areas for Improvement. Participants also provided constructive feedback regarding our onboarding approach. A recurring theme was the desire for greater structural control. For example, P4 suggested making linear onboarding “*hierarchical*”, while P9 requested a “*search feature for the guided tour*” to locate specific explanations. Users also suggested minor visual refinements, such as adding a “*blink*” before highlighting an element (P13) to catch subtle opacity changes. Participants also suggested for the chatbot to actively trigger highlights on components when answering.

8 DISCUSSION

To contextualize our findings, we reflect on *VizPilot*’s deployment, design lessons, limitations, and future directions.

8.1 Deployment and Generalizability

VizPilot is deployed as a browser extension that automatically extracts SVG structure and generates bitmap images to produce onboarding overlays, without the need of manual instrumentation. Our focus is on customized composite visualizations that still lack effective onboarding support, rather than commercial dashboards with built-in guidance or editorial visualizations designed for broad accessibility, which already provide comprehension affordances which are usually missing for customized composite visualizations. *VizPilot* is a platform-agnostic approach and operates directly on rendered SVGs, making it compatible with all composite visualizations implemented by popular visualization packages like D3.js and Vega-Lite, and broadly applicable across diverse web-based visualization ecosystems. *VizPilot* assumes

structurally grouped, DOM-accessible SVG elements. Canvas rendered visualizations and heavily flattened SVGs are outside the current scope and may result in a reduced component extraction accuracy. Interaction source code is optional but improves the interaction mapping when available.

8.2 Design Lessons

A core challenge is reconciling the probabilistic nature of MLLMs with the deterministic demands of interactive interfaces. Directly binding explanations to visual elements risks hallucinated or imprecise mappings. *VizPilot* addresses this via a hybrid design: MLLMs propose semantic mappings, which are then verified programmatically to ensure correctness. Beyond technical correctness, our findings suggest that effective onboarding also depends on how users build understanding over time. Drawing on narrative visualization principles [37], a staged interaction design where by combining guided scrollytelling and free exploration, supports progressive mental model construction while maintaining user anatomy. Taken together, these findings position generative inference within constrained system-guided interaction workflows rather than as a standalone mechanism.

8.3 Limitations and Future Directions

Despite the good performance of the onboarding by *VizPilot*, practical limitations remain. Our controlled study evaluated two visualizations selected for structural diversity, while the gallery evaluation provided broader coverage across all four design patterns. Future work should extend controlled studies to more visualization types and real-world sources. Furthermore, the subjective ratings of AI-enabled systems may be influenced by novelty and expectation effects [24]. We have mitigated this issue by including objective measures such as task accuracy and task completion time, but a more in-depth analysis of the potential bias introduced by expectation and novelty effects remains an avenue for future work. *VizPilot* currently assumes a single SVG container, limiting support for multi-view dashboards. Large SVGs may also exceed MLLM context limits, motivating targeted element selection. In addition, MLLM latency can hinder real-time interaction, making smaller or locally hosted models a promising direction. Future work also includes adaptive onboarding that adjusts pacing and detail based on users’ expertise and interaction patterns.

9 CONCLUSION

As composite visualizations grow in complexity, the gap between developer intent and user comprehension widens. We present *VizPilot*, an automated onboarding approach implemented as a browser extension comprising a Composite Visualization Analyzer, a two-stage MLLM-driven pipeline performing Semantic Inference and Semantic Mapping over SVG-based composite visualizations, and an Onboarding Interface supporting Narrative Scrollytelling and Free Exploration. Our evaluation demonstrates improvements in analytical efficiency, reduced cognitive load, and strong pipeline robustness across diverse visualization structures. This work illustrates how multi-stage MLLM outputs can be effectively integrated to deliver scalable, generalizable onboarding for web-based composite visualizations, lowering barriers for both developers and end-users.

REFERENCES

- [1] M. Z. I. Alam, S. Islam, and E. Hoque. Seechart: Enabling accessible visualizations through interactive natural language interface for people with visual impairments. In *Proceedings of the 28th International Conference on Intelligent User Interfaces, IUI '23*, 19 pages, p. 46–64. Association for Computing Machinery, New York, NY, USA, 2023. doi: [10.1145/3581641.3584099](https://doi.org/10.1145/3581641.3584099) 2
- [2] Apache Software Foundation. Apache ECharts Examples: Candlestick on Mobile. <https://echarts.apache.org/examples/en/editor.html?c=candlestick-touch>. [Online]. Accessed: 2026-06-24. 16
- [3] Apache Software Foundation. Apache ECharts Examples: Shanghai Index. <https://echarts.apache.org/examples/en/editor.html?c=candlestick-sh>. [Online]. Accessed: 2026-04-01. 6, 7, 15, 16
- [4] A. Bendeck and J. Stasko. An empirical evaluation of the gpt-4 multimodal language model on visualization literacy tasks. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1105–1115, 2025. doi: [10.1109/TVCG.2024.3456155](https://doi.org/10.1109/TVCG.2024.3456155) 3
- [5] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 9 pages, Dec. 2011. doi: [10.1109/TVCG.2011.185](https://doi.org/10.1109/TVCG.2011.185) 2
- [6] C. Chen, H. K. Bako, P. Yu, J. Hooker, J. Joyal, S. C. Wang, S. Kim, J. Wu, A. Ding, L. Sandeep, A. Chen, C. Sinha, and Z. Liu. Visanatomy: An svg chart corpus with fine-grained semantic labels. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):560–570, 2026. doi: [10.1109/TVCG.2025.3634263](https://doi.org/10.1109/TVCG.2025.3634263) 2
- [7] C. Chen, B. Lee, Y. Wang, Y. Chang, and Z. Liu. Mystique: Deconstructing svg charts for layout reuse. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):447–457, 2024. doi: [10.1109/TVCG.2023.3327354](https://doi.org/10.1109/TVCG.2023.3327354) 2
- [8] A. K. Das, M. Tarun, and K. Mueller. Charts-of-thought: Enhancing llm visualization literacy through structured data extraction. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):1427–1437, 2026. doi: [10.1109/TVCG.2025.3634813](https://doi.org/10.1109/TVCG.2025.3634813) 3
- [9] D. Deng, W. Cui, X. Meng, M. Xu, Y. Liao, H. Zhang, and Y. Wu. Revisiting the design patterns of composite visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5406–5421, 2023. doi: [10.1109/TVCG.2022.3213565](https://doi.org/10.1109/TVCG.2022.3213565) 1
- [10] V. Dhanoa, A. Hinterreiter, V. Fediuk, N. Elmqvist, E. Gröller, and M. Streit. D-tour: Semi-automatic generation of interactive guided tours for visualization dashboard onboarding. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):721–731, 2025. doi: [10.1109/TVCG.2024.3456347](https://doi.org/10.1109/TVCG.2024.3456347) 2
- [11] V. Dhanoa, G. M. León, E. Hoggan, E. Gröller, M. Streit, and N. Elmqvist. Hey dashboard!: Supporting voice, text, and pointing modalities in dashboard onboarding. *arXiv preprint arXiv:2510.12386*, 2025. 2
- [12] V. Dhanoa, C. Walchshofer, A. Hinterreiter, H. Stitz, E. Groeller, and M. Streit. A process model for dashboard onboarding. *Computer Graphics Forum*, 41(3):501–513, 2022. doi: [10.1111/cgf.14558](https://doi.org/10.1111/cgf.14558) 2
- [13] X. L. Do, M. Hassanpour, A. Masry, P. Kavehzadeh, E. Hoque, and S. Joty. Do llms work on charts? designing few-shot prompts for chart question answering and summarization. *arXiv preprint arXiv:2312.10610*, 2023. 3
- [14] A. Dong, W. Zeng, X. Chen, and Z. Cheng. Vistory: Interactive storyboard for exploring visual information in scientific publications. In *Proceedings of the 12th International Symposium on Visual Information Communication and Interaction, VINCI '19*, article no. 12, 8 pages. Association for Computing Machinery, New York, NY, USA, 2019. doi: [10.1145/3356422.3356430](https://doi.org/10.1145/3356422.3356430) 7, 16
- [15] L. W. Ge, Y. Cui, and M. Kay. Calvi: Critical thinking assessment for literacy in visualizations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23*, article no. 815, 18 pages. Association for Computing Machinery, New York, NY, USA, 2023. doi: [10.1145/3544548.3581406](https://doi.org/10.1145/3544548.3581406) 3
- [16] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Z. Lin, B. Zhang, L. Ni, W. Gao, Y. Wang, and J. Guo. A survey on llm-as-a-judge. *The Innovation*, 7(6):101253, 2026. doi: [10.1016/j.xinn.2025.101253](https://doi.org/10.1016/j.xinn.2025.101253) 7
- [17] Y. Han, C. Zhang, X. Chen, X. Yang, Z. Wang, G. Yu, B. Fu, and H. Zhang. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*, 2023. 3
- [18] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, eds., *Human Mental Workload*, vol. 52 of *Advances in Psychology*, pp. 139–183. North-Holland, 1988. doi: [10.1016/S0166-4115\(08\)62386-9](https://doi.org/10.1016/S0166-4115(08)62386-9) 7
- [19] Highsoft AS. Highcharts demo: Combo multi-axes chart. <https://www.highcharts.com/demo/highcharts/combo-multi-axes>. [Online]. Accessed: 2026-03-30. 1
- [20] J. Hong, C. Seto, A. Fan, and R. Maciejewski. Do llms have visualization literacy? an evaluation on modified visualizations to test generalization in data interpretation. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):7004–7018, 2025. doi: [10.1109/TVCG.2025.3536358](https://doi.org/10.1109/TVCG.2025.3536358) 3
- [21] N. Hoque and N. Sultanum. Dashguide: Authoring interactive dashboard tours for guiding dashboard users. *Computer Graphics Forum*, 44, 05 2025. doi: [10.1111/cgf.70107](https://doi.org/10.1111/cgf.70107) 2
- [22] W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *2012 IEEE Pacific Visualization Symposium*, pp. 1–8, 2012. doi: [10.1109/PacificVis.2012.6183556](https://doi.org/10.1109/PacificVis.2012.6183556) 1, 7
- [23] S. R. Khan, V. Chandak, and S. Mukherjee. Evaluating llms for visualization generation and understanding. *Discover Data*, 3(1):15, 2025. doi: [10.1007/s44248-025-00036-4](https://doi.org/10.1007/s44248-025-00036-4) 3
- [24] T. Kosch, R. Welsch, L. Chuang, and A. Schmidt. The placebo effect of artificial intelligence in human–computer interaction. *ACM Trans. Comput.-Hum. Interact.*, 29(6), article no. 56, 32 pages, Jan. 2023. doi: [10.1145/3529225](https://doi.org/10.1145/3529225) 8, 9
- [25] C. Lai, Z. Lin, R. Jiang, Y. Han, C. Liu, and X. Yuan. Automatic annotation synchronizing with textual description for visualization. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI '20*, 13 pages, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2020. doi: [10.1145/3313831.3376443](https://doi.org/10.1145/3313831.3376443) 2
- [26] S. Lee, S.-H. Kim, and B. C. Kwon. Vlat: Development of a visualization literacy assessment test. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):551–560, 2017. doi: [10.1109/TVCG.2016.2598920](https://doi.org/10.1109/TVCG.2016.2598920) 3
- [27] J. R. Lewis. Ibm computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78, 1995. doi: [10.1080/10447319509526110](https://doi.org/10.1080/10447319509526110) 7
- [28] A. Lex, N. Gehlenborg, H. Strobel, R. Vuilleumot, and H. Pfister. Upset: Visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1983–1992, 2014. doi: [10.1109/TVCG.2014.2346248](https://doi.org/10.1109/TVCG.2014.2346248) 16
- [29] H. Li, Y. Wang, A. Wu, H. Wei, and H. Qu. Structure-aware visualization retrieval. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, CHI '22*, article no. 409, 14 pages. Association for Computing Machinery, New York, NY, USA, 2022. doi: [10.1145/3491102.3502048](https://doi.org/10.1145/3491102.3502048) 2
- [30] W. Li, Z. Wang, Y. Wang, D. Weng, L. Xie, S. Chen, H. Zhang, and H. Qu. Geocamera: Telling stories in geographic visualizations with camera movements. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23*, article no. 170, 15 pages. Association for Computing Machinery, New York, NY, USA, 2023. doi: [10.1145/3544548.3581470](https://doi.org/10.1145/3544548.3581470) 2
- [31] Y. Ming, H. Qu, and E. Bertini. Rulematrix: Visualizing and understanding classifiers with rules. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):342–352, 2019. doi: [10.1109/TVCG.2018.2864812](https://doi.org/10.1109/TVCG.2018.2864812) 6, 7, 15, 16
- [32] E. Mörth, S. Bruckner, and N. N. Smit. Scrollyvis: Interactive visual authoring of guided dynamic narratives for scientific scrollytelling. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5165–5177, 2023. doi: [10.1109/TVCG.2022.3205769](https://doi.org/10.1109/TVCG.2022.3205769) 2
- [33] S. Pandey and A. Ottley. Benchmarking visual language models on standardized visualization literacy tests. *Computer Graphics Forum*, 44(3):e70137, 2025. doi: [10.1111/cgf.70137](https://doi.org/10.1111/cgf.70137) 3
- [34] S. Ruan, Q. Guan, P. Griffin, Y. Mao, and Y. Wang. Quantumeyes: Towards better interpretability of quantum circuits. *IEEE Transactions on Visualization and Computer Graphics*, 30(9):6321–6333, 2024. doi: [10.1109/TVCG.2023.3332999](https://doi.org/10.1109/TVCG.2023.3332999) 16
- [35] A. Satyanarayan and J. Heer. Authoring narrative visualizations with ellipsis. *Computer Graphics Forum*, 33(3):361–370, 2014. doi: [10.1111/cgf.12392](https://doi.org/10.1111/cgf.12392) 2
- [36] 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, 2017. doi: [10.1109/TVCG.2016.2599030](https://doi.org/10.1109/TVCG.2016.2599030) 2
- [37] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–

- 1148, 2010. doi: [10.1109/TVCG.2010.179](https://doi.org/10.1109/TVCG.2010.179) 9
- [38] R. Shier. Statistics: 2.2 the wilcoxon signed rank sum test. *Mathematics Learning Support Centre*. Retrieved from <http://www.statstutor.ac.uk/resources/uploaded/wilcoxonsignedranktest.pdf>, 7, 2004. 7
- [39] M. Shin, A. Tran, S. Wu, A. Mathews, R. Wang, G. Lyall, and L. Xie. Attentionflow: Visualising influence in networks of time series. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, 4 pages, p. 1085–1088. Association for Computing Machinery, New York, NY, USA, 2021. doi: [10.1145/3437963.3441703](https://doi.org/10.1145/3437963.3441703) 16
- [40] C. Stoiber, D. Ceneda, M. Wagner, V. Schetinger, T. Gschwandtner, M. Streit, S. Miksch, and W. Aigner. Perspectives of visualization onboarding and guidance in va. *Visual Informatics*, 6(1):68–83, 2022. doi: [10.1016/j.visinf.2022.02.005](https://doi.org/10.1016/j.visinf.2022.02.005) 2
- [41] C. Stoiber, F. Grassinger, M. Pohl, H. Stitz, M. Streit, and W. Aigner. Visualization onboarding: Learning how to read and use visualizations. In *IEEE Workshop on Visualization for Communication*. IEEE, 08 2019. doi: [10.31219/osf.io/c38ab](https://doi.org/10.31219/osf.io/c38ab) 2, 4
- [42] C. Stoiber, M. Pohl, and W. Aigner. Design actions for the design of visualization onboarding methods. In *2023 IEEE VIS Workshop on Visualization Education, Literacy, and Activities (EduVis)*, pp. 1–10, 2023. doi: [10.1109/EduVis60792.2023.00007](https://doi.org/10.1109/EduVis60792.2023.00007) 2
- [43] C. Stoiber, S. Radkohl, F. Grassinger, D. Moitz, H. Stitz, E. Goldgruber, D. Girardi, and W. Aigner. Authoring tool for data journalists integrating self-explanatory visualization onboarding concept for a treemap visualization. In *Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter*, pp. 1–14, 2023. doi: [10.1145/3605390.3605394](https://doi.org/10.1145/3605390.3605394) 2
- [44] C. Stoiber, C. Walchshofer, M. Pohl, B. Potzmann, F. Grassinger, H. Stitz, M. Streit, and W. Aigner. Comparative evaluations of visualization onboarding methods. *Visual Informatics*, 6(4):34–50, 2022. doi: [10.1016/j.visinf.2022.07.001](https://doi.org/10.1016/j.visinf.2022.07.001) 2
- [45] Q. Wang, Z. Li, S. Fu, W. Cui, and H. Qu. Narvis: Authoring narrative slideshows for introducing data visualization designs. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 08 2018. doi: [10.1109/TVCG.2018.2865232](https://doi.org/10.1109/TVCG.2018.2865232) 2
- [46] Y. Wang, L. Shen, Z. You, X. Shu, B. Lee, J. Thompson, H. Zhang, and D. Zhang. Wonderflow: Narration-centric design of animated data videos. *IEEE Transactions on Visualization and Computer Graphics*, 31(9):4638–4654, 2025. doi: [10.1109/TVCG.2024.3411575](https://doi.org/10.1109/TVCG.2024.3411575) 2
- [47] Z. J. Wang, R. Turko, and D. H. Chau. Dodrio: Exploring transformer models with interactive visualization. In H. Ji, J. C. Park, and R. Xia, eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pp. 132–141. Association for Computational Linguistics, Online, Aug. 2021. doi: [10.18653/v1/2021.acl-demo.16](https://doi.org/10.18653/v1/2021.acl-demo.16) 16
- [48] X. Wen, T. D. Nguyen, S. Ruan, Q. Shen, J. Sun, F. Zhu, and Y. Wang. PonziLens+: Visualizing bytecode actions for smart ponzi scheme identification. *IEEE Transactions on Visualization and Computer Graphics*, 31(9):6451–6465, 2025. doi: [10.1109/TVCG.2024.3516379](https://doi.org/10.1109/TVCG.2024.3516379) 6, 15, 16
- [49] X. Wen, T. D. Nguyen, L. Zhang, J. Sun, and Y. Wang. Prettismart: Visual interpretation of smart contracts via simulation. *IEEE Transactions on Visualization and Computer Graphics*, 31(6):3822–3835, 2025. doi: [10.1109/TVCG.2025.3567130](https://doi.org/10.1109/TVCG.2025.3567130) 2, 7, 15, 16
- [50] X. Wen, Y. Wang, M. Wu, F. Wang, X. Yue, Q. Shen, Y. Ma, and M. Zhu. <italic>diffseer</italic>: Difference-based dynamic weighted graph visualization. *IEEE Comput. Graph. Appl.*, 43(3):12–23, 12 pages, May 2023. doi: [10.1109/MCG.2023.3248289](https://doi.org/10.1109/MCG.2023.3248289) 16
- [51] L. Xie, Y. Lin, C. Liu, H. Qu, and X. Shu. Datawink: Reusing and adapting svg-based visualization examples with large multimodal models. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):824–834, 2026. doi: [10.1109/TVCG.2025.3634635](https://doi.org/10.1109/TVCG.2025.3634635) 2, 3
- [52] Z. Xu and E. Wall. Exploring the capability of llms in performing low-level visual analytic tasks on svg data visualizations. In *2024 IEEE Visualization and Visual Analytics (VIS)*, pp. 126–130, 2024. doi: [10.1109/VIS55277.2024.00033](https://doi.org/10.1109/VIS55277.2024.00033) 3
- [53] L. Ying, Y. Wang, H. Li, S. Dou, H. Zhang, X. Jiang, H. Qu, and Y. Wu. Reviving static charts into live charts. *IEEE Transactions on Visualization and Computer Graphics*, 31(8):4314–4328, 2025. doi: [10.1109/TVCG.2024.3397004](https://doi.org/10.1109/TVCG.2024.3397004) 2
- [54] X. Zeng, H. Lin, Y. Ye, and W. Zeng. Advancing multimodal large language models in chart question answering with visualization-referenced instruction tuning. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):525–535, 2025. doi: [10.1109/TVCG.2024.3456159](https://doi.org/10.1109/TVCG.2024.3456159) 3
- [55] J. Zhao, M. Glueck, F. Chevalier, Y. Wu, and A. Khan. Egocentric analysis of dynamic networks with egolines. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, 12 pages, p. 5003–5014. Association for Computing Machinery, New York, NY, USA, 2016. doi: [10.1145/2858036.2858488](https://doi.org/10.1145/2858036.2858488) 7, 16
- [56] J. Zhao, L. Gou, F. Wang, and M. Zhou. Pearl: An interactive visual analytic tool for understanding personal emotion style derived from social media. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 203–212, 2014. doi: [10.1109/VAST.2014.7042496](https://doi.org/10.1109/VAST.2014.7042496) 7, 16
- [57] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, 10 pages, p. 259–268. Association for Computing Machinery, New York, NY, USA, 2015. doi: [10.1145/2702123.2702419](https://doi.org/10.1145/2702123.2702419) 2, 5, 16
- [58] J. Zhao, S. Xu, S. Chandrasegaran, C. Bryan, F. Du, A. Mishra, X. Qian, Y. Li, and K.-L. Ma. ChartStory: Automated Partitioning, Layout, and Captioning of Charts into Comic-Style Narratives. *IEEE Transactions on Visualization & Computer Graphics*, 29(02):1384–1399, Feb. 2023. doi: [10.1109/TVCG.2021.3114211](https://doi.org/10.1109/TVCG.2021.3114211) 2
- [59] Y. Zhao, Y. Zhang, Y. Zhang, X. Zhao, J. Wang, Z. Shao, C. Turkay, and S. Chen. Leva: Using large language models to enhance visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1830–1847, 2025. doi: [10.1109/TVCG.2024.3368060](https://doi.org/10.1109/TVCG.2024.3368060) 2
- [60] Q. Zhu, T. Lu, S. Guo, X. Ma, and Y. Yang. Compositingvis: Exploring interactions for creating composite visualizations in immersive environments. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):591–601, 2025. doi: [10.1109/TVCG.2024.3456210](https://doi.org/10.1109/TVCG.2024.3456210) 1
- [61] C. Zhu-Tian, Y. Wang, Q. Wang, Y. Wang, and H. Qu. Towards automated infographic design: Deep learning-based auto-extraction of extensible timeline. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):917–926, 2020. doi: [10.1109/TVCG.2019.2934810](https://doi.org/10.1109/TVCG.2019.2934810) 2
- [62] ZingChart. Label mania dashboard. <https://www.zingchart.com/gallery/label-mania-dashboard>. [Online].Accessed: 2026-06-24. 14, 16
- [63] ZingChart. Mixed bar and staggered line. <https://www.zingchart.com/gallery/mixed-bar-line-staggered>. [Online].Accessed: 2026-06-24. 16
- [64] ZingChart. Multiple area charts with shared crosshairs. <https://www.zingchart.com/gallery/multiple-area-charts-with-shared-crosshair>. [Online].Accessed: 2026-06-24. 16
- [65] ZingChart. Multiple time series scales. <https://www.zingchart.com/gallery/multiple-time-series-scales>. [Online].Accessed: 2026-06-24. 16

A APPENDICES

Section A.1 details the Semantic Inference prompt, which reconstructs structured explanation units from multimodal inputs. Section A.2 presents an example of the Intermediate Semantic Representation produced by the Semantic inference stage. Section A.3 describes the Semantic Mapping prompt, which maps explanation units to SVG DOM elements. Section A.4 illustrates a complete Mapped Semantic Representation. Section A.5 provides the justification for the Visual Component Decomposition method. Section A.6 reports the metric definitions and the detailed results for each composite visualization for the quantitative evaluation of gallery visualizations. Section A.7 describes about the expert interviews conducted with two experienced visualization researchers to assess *VizPilot*’s utility in real-world authoring workflows, and Section A.8 reports the results of the comparative analysis of input modality conditions.

A.1 Semantic Inference

The following prompt implements the semantic inference stage, where the model reconstructs a structured intermediate representation from multimodal inputs. It performs visual component decomposition, knowledge extraction, and semantic-to-text generation within a single constrained prompt to produce atomic explanation units.

You are an expert in data visualization analysis, visualization onboarding design, and composite visualization reasoning. Your task is to analyze a composite visualization provided as: (1) an SVG specification, (2) a rendered bitmap image, (3) a user-authored description, and (4) the interaction source code, and produce a structured onboarding representation.

Structured Multi-Stage Reasoning (Strict)

You **must** follow a constrained three-stage reasoning pipeline. Each stage produces a structured intermediate representation that strictly bounds the next stage. You are **not** allowed to: skip stages, introduce new components after decomposition, or generate explanations not grounded in extracted knowledge.

Stage 1 - Visual Component Decomposition

Identify meaningful visual components using **both**: spatial layout cues from the image (position, alignment, adjacency) and structural grouping from the SVG. A base visual unit is the smallest visual structure that serves a distinct analytical role, which is distinguished by its own encoding, coordinate system, or data series, without decomposing to the level of individual marks.

Rules:

- A visual structure qualifies as a visual component **only** if it independently supports the extraction of at least one distinct form of visualization knowledge.
- Visual components can be hierarchical, where higher-level components contain multiple sub-components, provided each level independently yields extractable knowledge.
- Basic visual marks are **not** separate visual components, as they share redundant visual encodings and do not convey independent knowledge on their own.
- Visual components are categorized into:
 - **Data-driven components**: Coherent visual structures representing analytical data representations that require distinct, independent explanations.
 - **Auxiliary components**: Contextual or reference elements (e.g., legends, axes, overview panels) that independently contribute extractable onboarding knowledge.

Rules:

Output: component_id, role (overview | detail | auxiliary | shared), chart_type, approximate_position, description.

Stage 2 - Knowledge Extraction

For **each** visual component, extract structured knowledge using this taxonomy:

- 1) **Structural** - Describe the overall composition and layout pattern (e.g., small multiples, coordinated views, overlays); explain shared elements (shared axes, legends) and how they connect different visual components.
- 2) **Visual Encoding** - Explain how data attributes are mapped to visual channels (position, color, size, shape, or mark type); focus on how to read each individual mark within a visual component.
- 3) **Data** - Identify what data is shown in each visual component: variables, units, categories, and time ranges.
- 4) **Analytical** - Describe observable patterns, trends, comparisons, or relationships.
- 5) **Interaction** - Infer supported interactions (hover, filter, brush, zoom) and explain what analytical tasks they enable.

Constraints:

- Knowledge **must** be grounded in visible evidence
- Analytical knowledge **must** describe observable patterns
- Interaction knowledge **must** only be included if supported by the interaction source code or visual evidence
- No new components may be introduced

Stage 3 - Semantic-to-Text Generation

Transform structured knowledge into atomic explanation units. Each unit **must**:

- correspond to **one** visual component
- correspond to **one** knowledge category
- express **one** atomic concept
- derive directly from Stage 2 knowledge

Use of Interaction Source Code

The Interaction source code may be used **only** to:

- Infer the existence and intent of user interactions (e.g., hover, click, brush, zoom, filter, linked highlighting)
- Understand interaction scope across multiple visual components (e.g., coordinated or linked components)
- Disambiguate interactions that are not visually obvious from the static SVG or rendered image alone

Traceability Requirement (Critical)

Each explanation_step **must** be traceable to a visual component (target_component_id), a knowledge category (step_type), and a specific extracted knowledge unit. If it cannot be traced, **do not** generate it.

Strategic Narrative Rules

- *Relational Logic*: Explain how visual components relate (e.g., filtering, shared axes)
- *Dual-Encoding Clarity*: Explain combined encodings clearly
- *Action-Oriented Language*: Guide user attention (e.g., “Look at the colored dots”)
- Include at least one analytical “Aha” insight
- Use mark-level terms: “colored dot”, “bar”, “line”, “box”

Conceptual Framing Rule (Critical): Start by establishing (1) overall goal of the visualization, (2) main visual components, (3) role of each component. Do **not** begin with mark-level explanations.

Cross-Component Coordination Rule: Include at least **two** steps explaining relationships between visual components (e.g.,

overview → detail navigation, filtering, highlighting, shared axes, causal relationships).

Atomicity & Cognitive Load: Each step must be 1–2 sentences (≈15–35 words) expressing only **one** idea. Recommended progression: structure → data → encodings → analytical → interactions.

Output Format (JSON only):

```
{
  "explanation_steps": [
    {
      "step_type": "structure | encoding | data | analytical | interaction",
      "target_component_id": "...",
      "priority": 1,
      "explanation_text": "..."
    }
  ]
}
```

Quality Requirements

- Include at least one step for: structure, data, encoding, analytical
- Include interaction steps only if supported by interaction source or visual evidence
- Include at least one workflow guidance step
- Include at least **two** cross-component coordination steps
- `explanation_text` must be concise and user-facing
- No hallucinated elements or interactions

Important:

- Each explanation step must be traceable to a visual component and knowledge type
- Do **not** output intermediate reasoning
- Return **JSON only**

Input

The following will be provided: (1) user-authored description, (2) SVG code, (3) rendered bitmap image, (4) interaction source code.

A.2 Intermediate Semantic Representation

To illustrate the output of the Semantic Inference stage, we present a single representative explanation unit from the structured explanation steps array generated for the MatrixWave visualization (Fig. 3) (Listing 1). Each entry encodes a single atomic concept tied to one visual component and one knowledge category, ready to be consumed by the Semantic Mapping stage. The `step_type` field classifies the knowledge category. The `target_component_id` field identifies the logical component within the composite visualization to which the explanation unit belongs. The `priority` field governs presentation order.

Listing 1: Example JSON output for a single explanation unit produced by Semantic Inference stage.

```
{
  "step_type": "structure",
  "target_component_id": "step_bars_shared",
  "priority": 2,
  "explanation_text": "Use the slim header bars as the overview. They summarize per-page traffic at each step and align with the rows"
}
```

A.3 Semantic Mapping

The following prompt implements the semantic mapping stage, which maps abstract explanation units to SVG elements. It employs hierarchical selector inference to construct deterministic selector paths

from component-level containers to encoding-level elements, enabling precise and verifiable mappings.

You are an expert in SVG structure analysis, DOM reasoning, and visualization semantic mapping. Your task is to map each explanation unit to exact SVG elements using precise, hierarchical, and verifiable SVG selectors.

Hierarchical Selector Inference (Core Method)

You **must** construct selector paths that reflect the DOM hierarchy:

parent-container > intermediate-group > target-element

Rules:

- Always begin from the resolved component container
- Progressively refine through nested `<g>` groups toward target marks
- Flat selectors (e.g., `rect` alone) are **not** acceptable unless uniquely identifying
- Do **not** invent IDs, classes, or attributes — use only those present in the SVG

Two-Stage Mapping Pipeline

Stage 1 - Component-Level Mapping (Container Resolution)

Identify the SVG container (`<g>`, `<svg>`, or grouping node) that bounds each visual component. Use spatial layout cues (left-/right/top/bottom), `<g>` nesting depth, and structural grouping to resolve the correct container.

Stage 2 - Encoding-Level Mapping (Mark Resolution)

Within the resolved container **only**, identify the SVG elements that correspond to the semantic unit of each explanation step. These include marks such as bars (`rect`), lines (`path/line`), dots (`circle`), areas, or glyphs. Encoding selectors **must** be scoped within their corresponding component container.

Selector Construction Rules

- Prefer **IDs** > **classes** > attribute selectors > structural selectors
- If multiple attributes exist, include **both** for precision
- Return **multiple candidate selectors** per category when ambiguity exists

Ambiguity Resolution

If multiple candidates exist, return all with confidence scores:

- **high** - strong structural and visual evidence
- **medium** - partial evidence
- **low** - weak or ambiguous match

Use component hierarchy, relative positioning, and repetition patterns to disambiguate.

Output Format (JSON only):

```
{
  "mappings": [
    {
      "step_priority": 1,
      "target_component_id": "...",
      "step_type": "structure | encoding | data | analytical | interaction",
      "component": {
        "candidates": [
          {
            "selector": "...",
            "confidence": "high | medium | low",
            "reason": "..."
          }
        ]
      }
    }
  ]
}
```

```

    ]
  },
  "encoding": {
    "candidates": [
      {
        "selector": "...",
        "confidence": "high | medium | low",
        "reason": "..."
      }
    ]
  }
}
}
}
}

```

Important:

- Each mapping entry corresponds to **exactly one** explanation unit, identified by `step_priority`, `target_component_id`, and `step_type`
- Encoding selectors **must** be scoped within their resolved component container
- Do **not** output executable code or explanations outside the JSON
- Selectors must be deterministic and verifiable against the SVG DOM
- Return **JSON only**

Input

The following will be provided: (1) Explanation units from semantic inference, (2) SVG code

A.4 Mapped Semantic Representation

To demonstrate *VizPilot*'s output after the Semantic Mapping stage, we provide a representative Mapped Explanation Unit generated for the MatrixWave visualization (Fig. 3) (Listing 2). The output follows a structured format where the `step_type` dictates the knowledge category. The `target_component` field identifies the logical parent visual component within the composite visualization, providing the spatial context necessary during the Semantic Mapping stage. The `resolvedSelector` represents the final, validated DOM path produced after the MLLM output undergoes the programmatic validation step. The actions array tells the *VizPilot* frontend how to visually emphasize the corresponding SVG elements. This structured output ensures that the onboarding remains interactive and synchronized with the live visualization state.

Listing 2: Example JSON output for a single mapped explanation unit produced by the Semantic Mapping stage.

```

{
  "step_id": "step_2",
  "step_type": "structure",
  "text": "Use the slim header bars as the overview..",
  "target_component": "step_bars_shared",
  "resolvedSelector": "g.header g.mathead rect.val",
  "actions": ["highlight"]
}

```

A.5 Visual Component Decomposition Justification

VizPilot performs decomposition at the level where the visual structure should independently support the extraction of at least one distinct form of visualization knowledge rather than at the level of individual visual marks. We justify this design choice as follows.

Alignment with the onboarding task. *VizPilot* is designed to help first time users build a structural understanding of an unfamiliar composite visualization as what components exist, how they relate, and what each contributes. This requires identifying visual components not enumerating every mark within them. A user does not need an onboarding step explaining each individual bar in a bar chart as they

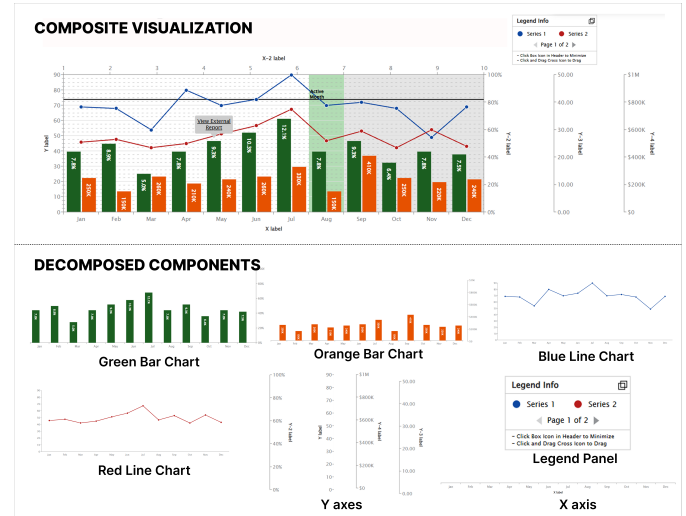


Fig. 8: Component decomposition of a composite visualization from ZingChart [62].

need to understand that the bar chart exists, what it encodes, and how it relates to adjacent components. Mark level detail is conveyed within a component's explanation, not by further decomposing the chart into per mark components.

Avoiding combinatorial explosion. Composite visualizations in our gallery contain charts with tens to hundreds of individual marks. Treating each mark as a separate component would produce an unusable number of onboarding steps, which would be against the purpose of structured onboarding and overwhelming the user with redundant, near identical explanations.

Illustrative Example

Figure 8 illustrates how our knowledge driven decomposition criterion operates in practice on a composite visualization. This example demonstrates the important boundary decisions that follow directly from our definition of a visual component:

Data-driven components within a shared coordinate space. Although the “green bar chart” and “orange bar chart” are rendered within the same main plot area and share an X-axis, they are treated as two distinct data-driven visual components. They encode different data dimensions using distinct mark types and colors, each contributing independent visualization knowledge that requires separate explanation. Similarly, the two overlaying line charts are treated as separate data-driven components. Merging all four into a single “main plot” visual component would be against the fact that they encode distinct variables and require separate onboarding explanations.

Exclusion of individual visual marks. Despite the chart containing individual green bars, orange bars, and line points, none of these individual marks constitute separate visual components because they share redundant visual encoding knowledge. A first time user needs to understand what the green bar series represents as a whole across months, rather than receiving near identical explanations per bar.

Auxiliary visual components. X-axis, Y-axes, and legend panel are each treated as separate visual components because they serve meaningfully distinct onboarding purposes, where axes provide scale context and the legend provides series identification. Each warrants its own explanation step rather than being subsumed into a generic “main chart” component.

A.6 Quantitative Evaluation of Gallery Visualizations: Metrics and Detailed Results for Each Composite Visualization

This section provides full definitions of the evaluation metrics, along with the detailed results of each composite visualization.

Metric Definitions

Decomposition (Precision, Recall, F1). Precision measures the proportion of system-identified visual components that correspond to a ground-truth visual component, Recall measures the proportion of ground-truth visual components our pipeline correctly identified and F1 is their harmonic mean. Ground-truth visual component lists were established by the authors prior to pipeline execution using the Visual Component Decomposition definition.

Semantic Mapping (Exact, Partial). For each generated onboarding step, this dimension evaluates whether the generated SVG selector correctly maps the visual region described in the onboarding step. Exact accuracy counts only fully correct selectors and Partial accuracy additionally counts near misses such as a selector targeting a parent container one level too broad.

Explanation Coherence (Type alignment, Component alignment, Atomicity). Each step is assessed on three independent sub-criteria: Type alignment, which evaluates whether the explanation matches its declared step type (structure, data, encoding, analytical, or interaction); Component alignment, which evaluates whether the text describes content belonging to its target visual component; and Atomicity, which checks whether the step expresses exactly one knowledge category without mixing structural, encoding, data, analytical, or interaction knowledge. Each sub-criterion is reported as a pass rate across all steps for a visualization.

End-to-End Success Rate (E2E). A ground-truth visual component counts as a full success only if it was correctly decomposed, every step generated for it received a correct or partial semantic mapping judgment, and every step passed all three explanation coherence criteria. The E2E rate is the proportion of ground-truth visual components satisfying this joint criterion, making missed components explicitly visible in the score.

Detailed Results for Each Composite Visualization

Tables 4 and 5 report detailed results for each composite visualization across all 18 gallery visualizations for each evaluation dimension and the end-to-end success rate, respectively.

Across all 18 visualizations, *VizPilot* achieves strong and consistent performance on all three dimensions. Decomposition precision consistently exceeds recall, indicating our approach more often misses a minor visual component than invents a phantom one, with lower recall cases concentrated in visualizations with densely overlapping components or ambiguous spatial layouts. Semantic mapping partial accuracy is uniformly high, with lower exact accuracy reflecting cases where *VizPilot* selects a slightly broader parent container rather than the precise target element. Among coherence sub-criteria, component alignment is the strongest across all visualizations while atomicity is the weakest, reflecting occasional steps that blend encoding and analytical knowledge within a single explanation unit. The end-to-end success rate, which jointly requires correct decomposition, grounding, and explanation quality, further confirms that errors are not dominated by a single failure mode but are instead distributed sparsely across cases. Overall, results suggest that *VizPilot* generalizes effectively across heterogeneous composite design patterns.

A.7 Expert Interview

To assess *VizPilot*'s utility in real-world authoring workflows, we conducted an evaluation with two visualization researchers with more than 7 years of experience in visualization development. Both experts noted that manually building onboarding materials typically requires labor-intensive documentation or custom video rendering. After testing *VizPilot*, they confirmed the automated pipeline accurately captured the structural intent, visual encodings, and high-level patterns. They highlighted that *VizPilot* drastically reduces developer effort by eliminating the need to manually code event listeners, while delivering a significantly more engaging end-user experience than static text. Both expressed a strong willingness to integrate the tool into their regular workflows, suggesting future enhancements such as animated state transitions for complex interaction chains and improved accuracy for the QA assistant on high-level analytical queries.

A.8 Results of Comparative Analysis of Input Modalities

We present inference outputs across four input modality conditions for four composite visualizations: PrettSmart [49], RuleMatrix [31], PonziLens [48], and Shanghai Index [3], spanning distinct composite design patterns. Because interaction source code is explicitly restricted to interaction inference in our pipeline design, structural and encoding outputs are identical between SVG + Image and the full pipeline. Therefore, Tables 6 and 7 compare three conditions (SVG Only, Image Only, SVG + Image), while Table 8 compares all four to isolate the specific contribution of interaction source code.

Key Findings (Table 6). SVG-only consistently produces DOM-valid selectors with structural precision but shallow spatial semantics and its naming is geometry driven and its structural narratives describe geometric relationships rather than spatial reading strategies. Image-only produces richer spatial language and coarser but more semantically meaningful component groupings, but critically relies on `target_description` fields rather than `component_ids`, making all outputs unresolvable by the semantic mapping stage. SVG + Image combines both, where spatially grounded naming with selector ready identifiers. This pattern holds consistently across all four visualizations, with PonziLens showing the most severe image-only degradation as its specialized domain vocabulary (Ponzi features, execution modules) cannot be anchored to generic SVG primitives without structural context, suggesting that domain complexity amplifies the image-only limitation.

Key Findings (Table 7). Color semantics and mark-type meanings are recovered consistently across SVG-only and SVG + Image for all four visualizations. The critical difference is mapping, where image-only descriptions rely on visual anchors (e.g., “tan squares”, “colored dots”, “arc shows accuracy”) that cannot be resolved to any DOM node, making them unactionable for selector inference regardless of how semantically accurate the description is. SVG-only and SVG + Image both produce component-mapped selectors, with SVG + Image additionally confirming visual appearance and achieving higher-confidence grounding. This pattern is consistent across all four visualizations. SVG + Image correctly identifies and confirms the encoding while image-only produces generic descriptions without DOM anchors.

Key Findings (Table 8). The interaction results are consistent across all four visualizations. SVG-only infers interactions from DOM structure (e.g., hover overlays, nested click targets) but systematically misses action-consequence relationships that are only encoded in event handlers. For example, it correctly identifies the simulation card hover overlay in PrettSmart but cannot infer that clicking loads a new simulation and updates the timeline. Image-only produces the least reliable output, as it correctly identifies interactive affordances from visual cues (buttons, draggable regions, hover-styled elements) but confuses interaction types (e.g., click vs. hover in PrettSmart; unverified filter behaviors in RuleMatrix and PonziLens) and produces no DOM valid selectors for any interaction step across any visualization. SVG + Image partially bridges the gap and it confirms which elements are interactive visually but still cannot infer cross-component consequences without source code. Only the full pipeline (SVG + Image + Interaction Source Code) produces all interaction steps with verified selectors and cross-component workflow guidance in all four visualizations, replacing convention-based guesswork with grounded action-consequence reasoning derived directly from event handlers.

In summary, each modality provides a distinct and non-redundant capability across all four tested visualizations: SVG structure offers precise visual component decomposition and DOM-valid selectors; bitmap images resolve spatial ambiguities and confirm visual appearance; and interaction source code replaces convention-based guesswork with verified event handler logic. All three inputs are essential for accurate, grounded, and actionable onboarding, and the necessity of each modality is consistent regardless of composite design pattern or domain complexity.

Table 4: Detailed results for each visualization across decomposition, semantic mapping, and coherence dimensions.

Composite Pattern	Visualization	Decomposition			Semantic Mapping		Coherence		
		Prec.	Rec.	F1	Exact	Partial	Type	Comp	Atom
Juxtaposition	PrettiSmart [49]	1.000	1.000	1.000	0.940	1.000	0.889	1.000	1.000
	Diffseer [50]	1.000	1.000	1.000	0.843	0.929	1.000	1.000	0.930
	RuleMatrix [31]	0.889	1.000	0.941	0.929	0.929	1.000	0.929	1.000
	QuantumEyes [34]	0.909	0.909	0.909	0.950	1.000	0.950	1.000	0.800
	Multiple Time Series [65]	1.000	1.000	1.000	0.917	1.000	1.000	1.000	0.934
	Label Mania [62]	0.917	0.846	0.880	1.000	1.000	0.944	1.000	1.000
Overloading	MatrixWave [57]	1.000	0.850	0.919	0.882	1.000	0.941	0.941	0.824
	VIStory [14]	1.000	0.800	0.889	0.900	1.000	1.000	1.000	0.900
	AttentionFlow [39]	1.000	1.000	1.000	0.917	1.000	1.000	0.917	0.933
	EgoLines [55]	0.818	0.900	0.857	0.923	0.923	1.000	1.000	0.792
	Dodrio [47]	1.000	0.850	0.919	0.850	0.850	1.000	1.000	0.875
Nesting	PEARL [56]	0.875	0.875	0.875	0.938	0.938	1.000	0.875	1.000
	PonziLens+ [48]	1.000	1.000	1.000	0.936	1.000	0.882	1.000	0.765
	UpSet [28]	1.000	0.833	0.909	0.900	1.000	1.000	1.000	0.900
Superimposition	CandleStick on Mobile [2]	1.000	0.889	0.941	0.886	0.929	0.929	1.000	0.929
	Shanghai Index [3]	1.000	1.000	1.000	0.857	0.929	0.929	1.000	1.000
	A Timeline of Media-Inflamed Fears [64]	1.000	1.000	1.000	0.929	1.000	0.929	1.000	0.929
	Mixed Bar and Staggered Line [63]	1.000	0.878	0.935	1.000	1.000	1.000	1.000	0.917
<i>Mean</i>		0.967	0.918	0.940	0.912	0.968	0.966	0.981	0.913

Table 5: End-to-End Success Rate for each visualization across all 18 gallery visualizations.

Visualization	End-to-End Success Rate
PrettiSmart [49]	1.000
Diffseer [50]	1.000
RuleMatrix [31]	0.800
QuantumEyes [34]	0.909
Multiple Time SeriesMultiple Time Series [65]	1.000
Label Mania [62]	0.846
MatrixWave [57]	0.850
VIStory [14]	0.800
AttentionFlow [39]	0.875
EgoLines [55]	0.800
Dodrio [47]	0.800
PEARL [56]	0.889
PonziLens+ [48]	1.000
UpSet [28]	0.833
CandleStick on Mobile [2]	0.889
Shanghai Index [3]	1.000
A Timeline of Media-Inflamed Fears [64]	1.000
Mixed Bar and Staggered Line [63]	0.878
<i>Mean</i>	0.898

Table 6: Visual Component decomposition across four visualizations and three modality conditions.

Visualization	SVG Only	Image Only	SVG + Image
PrettiSmart	11 geometry-driven layers; layer-driven naming (e.g., <code>flow_links_layer</code> , <code>simulations_marks_layer</code>); structural step describes geometric relationships without spatial grounding: “Start at the top with Simulation strips . . . read central views where rectangles, curved links, and colored areas align to the Contract line . . .”	5 coarser components; appearance-driven naming (e.g., <code>flow_diagram</code> , <code>simulation_thumbnails</code>); structural step uses spatial language: “The large central blue-pink flow between Contract and addresses is the primary view. Small heatmaps above and the state strip below provide overview and temporal context.”	Spatially grounded naming with selector-ready identifiers (e.g., <code>sim0_heatmap-sim7_heatmap</code> , <code>flow_paths_layer</code>); structural step combines spatial and structural language: “Start at the top row of eight small-multiple cards . . . move down to the large middle view where colored curved paths link the Contract row to addresses at the same steps . . .”
RuleMatrix	15 steps; precise layer-driven component names (e.g., <code>g.rules</code> , <code>g.mo-fidelity</code>); selectors resolve to specific DOM elements including transform-indexed histogram columns.	Component-level mapping only (no step list); selectors reasonable (e.g., <code>g.rm-labels</code> , <code>g.func</code>) but unanchored to step text; interaction inference absent.	14 steps; spatially grounded names (e.g., <code>output_probability</code> , <code>fidelity_gauge</code>); selectors confirmed against both structure and appearance.
PonziLens	12 steps; all grounded to <code>g.overview_G</code> ; specific encoding candidates (loop bracket paths, action dot colors); all three modules (Path Feature, Path Grouping, Execution Detail) correctly identified.	14 steps but all selectors low-confidence generic (<code>svg</code> , <code>g</code> , <code>circle</code>); specialized domain vocabulary (Ponzi features, blockchain execution) cannot be anchored to SVG primitives without structural context; zero DOM actionability.	17 steps; most specific outputs across all four visualizations (e.g., <code>g.legend</code> <code>g.WRITE circle</code> , <code>g.overview_G path[stroke="#9b8ea9" i]</code>); domain-specific component roles recovered via joint SVG + Image reasoning.
Shanghai Index	11 steps; candlestick color-attribute selectors are strong; some selectors anchored to exact d attribute values; MA lines correctly identified by stroke color.	18 steps but almost entirely low-confidence generic selectors (<code>g[clip-path]</code> , <code>g[cursor="pointer"]</code>); strong visual semantic reasoning but zero DOM actionability; most extreme semantic-grounding gap across all tested visualizations.	14 steps; cleanest and most specific selector outputs (e.g., <code>g[clip-path="url(#zr0-c0)]" > path[stroke="rgb(84,112,198)" i]</code>); image confirms component boundaries that SVG-only over-specifies; fragile SVG-only selectors resolved.

Table 7: Selected encoding steps across four visualizations and three modality conditions.

Visualization	Mark	SVG Only	Image Only	SVG + Image
PrettiSmart	Function call rectangles	Brown = Payable; beige = NonPayable. <i>Selector: component-level</i>	“Tan squares mark function call events.” <i>No DOM anchor</i>	Brown square = Payable call; beige = NonPayable call, placed at the exact step on the corresponding row. <i>Selector: component-grounded, appearance confirmed</i>
	Flow curves	Pink = Money Out; blue = Money In; thicker = larger amount. <i>Selector: component-level</i>	“Curved strokes with arrowheads indicate money in or out.” <i>No DOM anchor</i>	Pink path = negative outflow; blue path = positive inflow; origin and landing point identify sender and receiver. <i>Selector: direction semantics visually confirmed</i>
RuleMatrix	Rule-output cells	Cell color encodes output class probability; cell size encodes support count. <i>Selector: g.rules container</i>	“Colored grid cells show rule strength.” <i>No DOM anchor; no size encoding mentioned</i>	Color encodes class probability; size encodes support; both confirmed against rendered cell appearance. <i>Selector: g.rules rect — appearance confirmed</i>
	Fidelity gauge	Arc length encodes fidelity score. <i>Selector: g.mo-fidelity</i>	“A gauge-like arc shows model accuracy.” <i>No DOM anchor</i>	Arc length and fill color both encode fidelity; full arc confirmed visually. <i>Selector: g.mo-fidelity path — confirmed</i>
PonziLens	Loop bracket paths	Bracket shape encodes loop structure; color encodes action type. <i>Selector: g.overview_G path</i>	“Arcs connect related events.” <i>Generic anchor; action-type color not mentioned</i>	Bracket shape = loop; color = action type (READ, WRITE, CALL); confirmed against rendered paths. <i>Selector: g.overview_G path[stroke="#9b8ea9" i]</i>
	Action dots	Dot color encodes action type (READ, WRITE, CALL). <i>Selector: g.legend g.WRITE circle</i>	“Colored dots indicate event types.” <i>No anchor; WRITE/READ/CALL distinction absent</i>	Dot color encodes action type; legend confirms READ/WRITE/CALL mapping visually. <i>Selector: g.legend g.WRITE circle — confirmed</i>
Shanghai Index	Candlestick marks	Color (red/green) encodes price direction; body height encodes open-close range. <i>Selector: color-attribute selector — strong</i>	“Red and green bars show price movement.” <i>No DOM anchor; wick encoding absent</i>	Red = price decline; green = price rise; wick height encodes high-low range; both confirmed. <i>Selector: path[stroke="rgb(84,112,198)" i] — confirmed</i>
	MA lines	Stroke color distinguishes MA5, MA10, MA20. <i>Selector: anchored by stroke color — correct</i>	“Colored lines show moving averages.” <i>MA5/MA10/MA20 distinction absent; no anchor</i>	Stroke color confirmed per MA series; clip-path container correctly disambiguates series. <i>Selector: g[clip-path] > path — confirmed</i>

Table 8: Interaction step comparison across four visualizations and all four modality conditions.

Visualization	Interaction	SVG Only	Image Only	SVG + Image	SVG + Image + Interaction Source Code
PrettiSmart	Simulation card	Hover reveals guide lines and outlined box; align to the same window in the flow view. <i>Basis: DOM structure</i>	“Click a thumbnail to switch the central flow.” <i>Incorrect: hover confused with click</i>	Hover reveals guide lines and outlined box. <i>Partially verified; click-to-load consequence unknown</i>	Click a card to load that simulation; timeline and state marks update. <i>Selector: <code>g.simu rect.hover.h_rect</code> — D3 verified</i>
	Function card hover	<i>Not generated</i>	“Use F0, F1, F2 to filter function events.” <i>Unverified; visual resemblance only</i>	<i>Not generated</i>	Hover a function card to highlight its calls across the timeline. <i>Selector: <code>g.func</code> — D3 mouseover verified</i>
	Workflow guidance	<i>Not generated</i>	<i>Not generated</i>	<i>Not generated</i>	Pick a simulation → inspect function cards → hover to spotlight calls → trace flows → confirm state changes. <i>Cross-component event chain — D3 verified</i>
RuleMatrix	Rule row hover	Hover highlights row; cross-highlight inferred from DOM class. <i>Basis: DOM structure</i>	“Hover a row to highlight it.” <i>Correct but no selector anchor</i>	Hover highlights row and cross-highlights linked columns. <i>Partially verified; filter consequence unknown</i>	Hover a rule row to highlight it and filter the linked output probability column. <i>Selector: <code>g.rules rect</code> — D3 mouseover verified</i>
	Sort interaction	<i>Not generated</i>	“Click column headers to sort.” <i>Visual affordance — unverified</i>	<i>Not generated</i>	Click a column header to re-sort rules by that criterion; row order updates. <i>Selector: <code>g.header text</code> — D3 click verified</i>
PonziLens	Path selection	Click path to expand detail view; inferred from DOM nesting. <i>Basis: DOM structure</i>	“Click a path to see details.” <i>Visual affordance — partially correct</i>	Click path expands detail; related action dots highlighted. <i>Partially verified; cross-component update unknown</i>	Click a path to expand the Execution Detail panel; action dots update to reflect selected path’s event sequence. <i>Selector: <code>g.overview_G path</code> — D3 click verified</i>
	Legend filter	<i>Not generated</i>	“Click legend items to filter.” <i>Visual resemblance — unverified</i>	<i>Not generated</i>	Click a legend item (READ-/WRITE/CALL) to toggle visibility of corresponding action dots across all modules. <i>Selector: <code>g.legend g.WRITE</code> — D3 click verified</i>
Shanghai Index	Brush zoom	Brush on overview panel zooms main chart; inferred from DOM rect overlap. <i>Basis: DOM structure — partially correct</i>	“Drag the overview bar to zoom.” <i>Visual affordance — correct action, no anchor</i>	Brush zooms main candlestick view; MA lines update accordingly. <i>Partially verified; MA line update unknown</i>	Drag the brush on the overview panel to zoom the main chart; all MA lines and candlesticks update to the selected range. <i>Selector: <code>g[cursor=“pointer”]</code> — D3 brush verified</i>
	Tooltip hover	<i>Not generated</i>	“Hover to see price details.” <i>Visual affordance — unverified</i>	<i>Not generated</i>	Hover a candlestick to reveal OHLC values and volume in a tooltip. <i>Selector: <code>g[clip-path] > path</code> — D3 mouseover verified</i>