UnitoPia: Interactive Construction of Pictorial Unit Visualization using a Contain-and-Fill Model

Xinghui Fu, Zhida Sun, Yoojin Jeon, Guozheng Li, Yu Zhang, Bongshin Lee, Min Lu

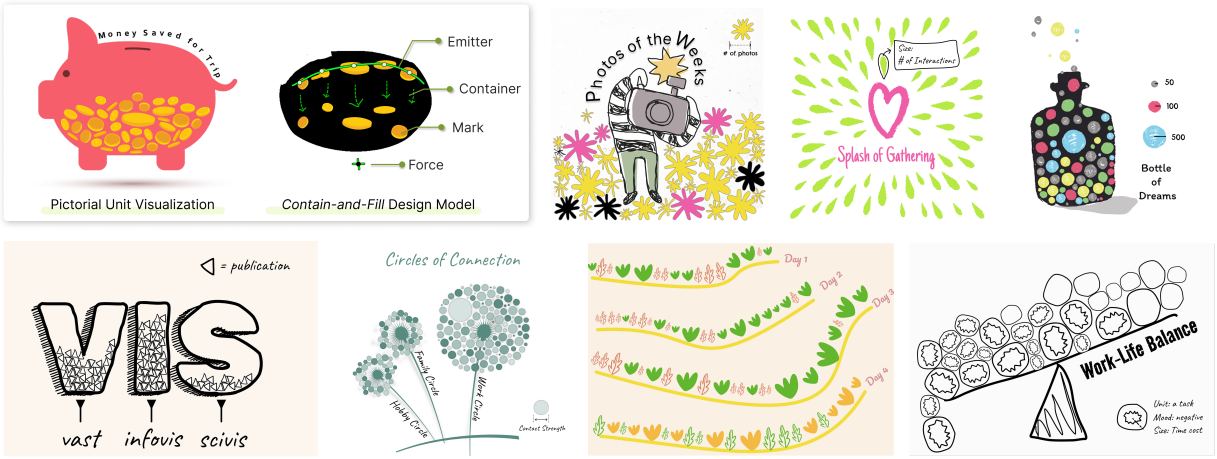


Figure 1: Examples created with *UnitoPia*: by directly manipulating the four concepts of *marks*, *container*, *emitter*, and *force* in the 'contain-and-fill' design model shown on the top left, users can create diverse pictorial unit visualizations with expressive elements and flexible layouts.

Abstract—Unit visualization represents each single data item as a visual element, facilitating intuitive understanding while preserving individual data identity. However, existing tools for creating unit visualizations are constrained by rigid, rule-based layouts and regular geometric shapes, limiting the flexibility and creativity in their visual expression. Informed by a study of 602 existing examples, we introduce UnitoPia, an interactive authoring system for *pictorial unit visualization*, enabling the creation of unit visualizations with irregular visual elements in expressive organic layouts. Built upon an intuitive *contain-and-fill* design model, UnitoPia allows users to construct pictorial unit visualizations through direct manipulation of four tangible concepts: *marks*, *container*, *emitter*, and *force*. To realize this design model, we introduce image-space based layout optimization via differentiable rendering, which efficiently contains and fills arbitrary visual elements within containers of any shape. A wide range of examples demonstrates UnitoPia's versatility and expressiveness, while a user study confirms its usability and effectiveness. The system and source code are available at https://unitopia-web.github.io/.

Index Terms—Unit visualization, pictorial chart, authoring toolkit, differentiable rendering

1 Introduction

Unit visualization is a family of visualization techniques that employ one-to-one visual encoding, where each data item is associated with one visual element [33]. Unlike aggregated visualization that maps multiple data items to a single visual element [12], this one-to-one mapping provides an effective and intuitive representation, particularly beneficial for emphasizing data granularity and conveying dataset composition. The use of unit visualization dates back to the Paleolithic era, when visual shapes were used as tallies for counting and record-keeping [29]. Over the years, unit visualizations have been developed in various forms [33, 48] and integrated with diverse interactions for data representation and storytelling [11, 40].

Data drawing projects like *Dear Data* [28] demonstrate that the potential of unit visualizations goes beyond strict analytical rigor, embracing artistic and illustrative expression. Similar to other forms of artistic visual designs [19, 46, 57], incorporating personalized and artistic expression into unit visualizations can enhance memorability [2], evoke emotional responses [21, 50], and promote data humanization [27].

However, existing unit visualization authoring toolkits fall short in supporting this creative potential. Current toolkits, such as ATOM [33], are constrained to basic geometric shapes (e.g., circles, squares) and rigid layout rules, making them ill-suited for generating creative and artistic unit visualizations. Two key challenges arise in this context. First, visual elements in projects, such as *Dear Data* [28], typically employ visually rich, irregular shapes rather than regular geometric primitives. Second, arranging these expressive elements demands advanced optimization techniques [37] and sophisticated geometric descriptors [20], both of which are often difficult to generalize across diverse visualization designs.

- *Xinghui Fu and Zhida Sun are with Shenzhen University, Shenzhen, China. E-mail: {fuxinghui7, zhdsun}@gmail.com.*
- *Yoojin Jeon is with Yonsei University, Seoul, Repulbic of Korea. E-mail: tmrwu@yonsei.ac.kr.*
- *Guozheng Li is with Beijing Institute of Technology, Beijing, China. E-mail: guozheng.li@bit.edu.cn.*
- *Yu Zhang is with the University of Oxford. E-mail: yuzhang94@outlook.com.*
- *Bongshin Lee is with the Department of Computational Science and Engineering and the Yonsei Institute of Digital Health, Yonsei University, Seoul, Republic of Korea. E-mail: b.lee@yonsei.ac.kr.*
- *Min Lu is with Hong Kong University of Science and Technology (Guangzhou) and the corresponding author of this paper. E-mail: lumin.vis@gmail.com.*

Manuscript received xx xxx. 202x; accepted xx xxx. 202x. Date of Publication xx xxx. 202x; date of current version xx xxx. 202x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.202x.xxxxxxx

In this work, we introduce *UnitoPia*, an interactive authoring system that addresses the challenges of creating artistically expressed unit visualizations. We formalize the concept of *pictorial unit visualizations* to denote unit visualizations characterized by irregular graphical elements and expressive layouts. Based on a study of 602 existing designs, we propose a design model grounded in the *contain-and-fill* metaphor, which comprises four tangible concepts: *marks*, *container*, *emitter*, and *force*. Unlike conventional design models for unit visualizations that rely on coordinate systems (e.g., Cartesian [11, 33] or radial [48]), our model does not explicitly compute coordinates. Instead, it employs an intuitive containing-and-filling mechanism to dynamically arrange units through their interactions with the container, emitter, and force. As shown in Figure 1, this contain-and-fill model affords a diverse range of expressive layouts with visual elements in various forms.

To realize the contain-and-fill model, we introduce an image-space layout optimization method based on differentiable rendering [22]. Rather than solving the layout problem in object space, which would involve complex geometric computations, our approach calculates image-space losses between rasterized images of unit shapes and containers, bypassing intricate geometric descriptions of shapes and spatial relationships. These image-space losses are shape-agnostic and can be easily back-propagated to optimize transformation parameters (i.e., scale, rotation, translation) in the object space via differentiable rendering. Meanwhile, working in image space naturally aligns with sketch-based direct manipulation [44], where a container of arbitrary shape can be simply defined as regions of pixels. Therefore, UnitoPia is developed as a sketch-based interface, allowing users to easily sketch *marks*, *containers*, *emitters*, and *forces*.

A user study with nine designers and non-experts demonstrates that UnitoPia supports direct and easy authoring of pictorial unit visualizations. A gallery of diverse examples illustrates UnitoPia's expressivity. The key contributions of this work include:

- A formative study analyzing 602 existing pictorial unit visualizations to characterize their design space and patterns.
- The *contain-and-fill* design model for pictorial unit visualization, supported by an image-space layout optimization method based on differentiable rendering.
- UnitoPia, a sketch-based interactive authoring system, validated through a user study and showcased via diverse example applications.

2 Related Work

In this section, we discuss the research work on two closely related topics of unit visualization and visualization authoring frameworks.

2.1 Unit Visualization

Unit visualization has been explored in diverse unit forms and layouts. For **unit representation**, early work by Neurath [31] used repetitive icons, while data dots in scatterplots [35] remain widely adopted, with subsequent variations [26, 55]. Beyond geometric shapes, pixel visualization [17] and image-based approaches such as Histomages [7], PhotoMesa [1], ShapeCollage [41] encode individual pixels or images as units. Text-based units appear in Wordle [13] and ShapeWordle [49]. Within artistic data visualization, *Dear Data* [28] exemplifies hand-crafted unit narratives that motivate this work. For **unit layout**, most work relies on Cartesian or radial coordinates. ATOM [33] uses an XY axes system. Dang et al. [8] proposed stacking to avoid overlap. Wang et al. [48] introduced a chain-linked circular packing. Elmqvist and Fekete [12] covered the hierarchical aggregation of scatterplot visualization. Another line of work adopts physical metaphors [14]: Sedimentation bar chart [16], Dust & Magnet [60], Kinetica [36], and SandDance [11] all leverage physics-inspired affordances for unit arrangement.

Existing work primarily focuses on rule-based layouts and regular shapes. Our work extends unit visualization into the pictorial domain, supporting irregular marks and flexible layouts beyond coordinate systems.

2.2 Visualization Authoring Frameworks

Visualization authoring tools fall into three paradigms. (1) **Declarative Specification:** The Grammar of Graphics [56] established a foundational data-to-visual mapping workflow, inspiring subsequent systems like ggplot2 [54], D3 [4], and Vega [39]. While expressive, these systems require programming expertise and offer limited support for custom graphical elements. ATOM [33], the most relevant declarative framework for unit visualization, restricts marks to regular geometric shapes and layouts to Cartesian coordinates. (2) **Shelf-based Configuration:** Tools such as Tableau [45], Lyra [38], and Charticulator [34] provide drag-and-drop interfaces for data to visual mapping. Recent systems have introduced pictorial elements [51, 59], but layout remain constrained by shelf-based templates. (3) **Constructive Assembly:** Another group of authoring tools uses the design workflow of constructive visualization [15]. Vector-based editing tools like Data Illustrator [25] and Dataink [58] adopt this bottom-up constructive approach, supporting freeform drawing [18], lazy data binding [25, 58], and sketch formalization [32]. Across these systems, layout is typically manual or rule-based.

Our work advances the constructive assembly paradigm with a new design model using the contain-and-fill metaphor. Unlike prior systems that rely on manual positioning or grid-based rules, UnitoPia supports creating flexible, organic unit visualization by directly manipulating the four tangible concepts.

3 Formative Study

In this section, we lay the foundation for UnitoPia through a formative study of existing pictorial unit visualizations. This corpus analysis allows us to establish a consensus definition for this emerging genre, while explicitly informing the subsequent design goals and interface interaction choices that guide our system's development.

3.1 Scope of Pictorial Unit Visualization

Pictorial unit visualization lies at the intersection of unit visualization [33] and pictorial charts [9, 31, 43]. As no widely accepted formal definition currently exists for pictorial unit visualization, we define the following scope used throughout this paper to ground both our corpus analysis and the design of UnitoPia: *pictorial unit visualization* is a class of unit visualization that preserves one-to-one data mapping while enabling creative, personal, and expressive visual design. To operationalize this definition, we consider a unit visualization as pictorial if it satisfies **either** criterion:

C1 **Irregular Graphical Elements.** The visualization uses semantically or metaphorically meaningful icons, pictures, or symbols beyond regular geometric primitives (e.g., circles, squares), reflecting *creative expression through mark design*.

C2 **Non-standard Visual Layout.** The visualization adopts an idiosyncratic spatial organization that deliberately departs from regular coordinate-based layouts, reflecting *creative expression through arrangement*.

3.2 Corpus Construction and Analysis

To build a representative corpus of pictorial unit visualizations, we collected examples from three sources: (1) **Open-source pictorial datasets:** 387 examples from Vistylist [43] and 78 from PICCL [42]. (2) **Data journalism outlets:** 73 examples from the New York Times (11), The Tardigrade (15), Information is Beautiful (9), Flowing Data (14), and UCLAB (24). (3) **Published works:** 64 examples from *Dear Data* [28], a seminal collection of hand-crafted pictorial unit visualizations. We screened all examples against C1 and C2 and retained those satisfying at least one criterion. No temporal restrictions were imposed. This process yielded a final corpus of 602 pictorial unit visualizations. Guided by our definition of a pictorial unit visualization, we designed an initial coding scheme with two perspectives:

- **Element design:** the visual representation of individual data units.
- **Visual layout:** the spatial arrangement of data elements.

To develop a preliminary coding scheme and assess reliability, we conducted iterative coding with a team of four coders: two of the authors and two professional graphic designers with experience in data visualization (each with over five years of industry practice). All four coders independently coded a set of 100 random samples from the corpus using the initial scheme. This independent coding revealed several ambiguities. The coders then discussed discrepancies and refined the coding criteria. This process was repeated until full consensus was reached on the 100 samples, which lasted for three rounds of discussions. The final scheme reached high inter-coder agreement (Cohen's $\kappa = 0.84$). Using this finalized scheme, two of the authors independently coded the full corpus of 602 examples, with ambiguous cases flagged and resolved through discussion with the designer-coders.

Figure 2: Examples from the pictorial unit visualization corpus. Each example satisfies at least one of the two criteria: (a) irregular elements (e.g., icons, hand-drawn marks), and (b) non-standard layout strategies (e.g., metaphor-based compositions such as "grapes" collage).

3.3 Findings

Our analysis of 602 pictorial unit visualizations reveals consistent patterns along two dimensions: *element design* and *visual layout*. For element design, we coded *shape regularity* (regular or irregular) and *pattern repetition* (identical, transformed, or proportion-encoding). For visual layout, we coded *layout regularity* (grid, linear, radial, circular, or organic) and *representational strategy* (abstract enclosure or metaphor-based composition). Below, we report the distributions and findings for each dimension.

3.3.1 Element Design: Shape and Pattern

Shape Regularity. Most pictorial unit visualizations use *irregular* forms (509 examples, 84.6%), while 93 examples (15.4%) use *regular* shapes in non-standard layouts. Among regular-shape examples, circles dominate (74 examples, 79.6%), followed by rectangles (14, 15.1%), polygons (2, 2.1%), and raster image patches (3, 3.2%). Among irregular-shape examples, representational icons of semantically meaningful pictures and symbols are predominant (441, 86.6% of irregular examples), while hand-drawn marks form a notable subset of irregular shapes (67, 13.2%). One example uses text as the primary graphical unit.

Pattern Repetition. Repetitive patterns are common (433 examples, 71.9%). For visualizations using *repetitive* patterns, identical repetition is most frequent (318 examples, 73.4% of repetitive cases), followed by rigid transformations (e.g., rotation or scaling; 61 examples, 14.1%) and repetition for proportional encoding (54 examples, 12.5%). *Non-repetitive* designs, where elements vary across categories or are entirely distinct, account for 169 examples (28.1%). Notably, nearly all non-repetitive designs use irregular marks (167, 98.8%), and many of the irregular marks are hand-drawn (67, 39.6%).

3.3.2 Visual Layout: Regularity and Representation

Layout Regularity. *Geometric-based* layouts dominate the corpus (481 examples, 79.9%), characterized by adherence to predetermined structural patterns. Within this group, grid layouts are most frequent (241 examples, 50.1% of geometric-based), followed by linear layouts (212, 44.0%), circular layouts (9, 1.9%), and radial layouts (5, 1.0%). A smaller subset (19, 3.9%) uses data-driven layouts, where units are positioned by external coordinates, such as timelines or maps. *Organic* layouts prioritize naturalistic appearance over geometric precision, and better approximate how objects gather naturally in real-world contexts [5, 16]. Organic layouts account for 121 examples (20.1%). Within this category, packing dominates (111 examples, 91.7%), while there is a small group of layouts emphasizing natural movement and fluidity is less common (10 examples, 8.3%).

Representational Strategy. Layouts also differ in meaning-making. *Abstract enclosure* layouts, where units are arranged within non-metaphorical boundaries, are most common (425 examples, 70.6%). Among these, shaped enclosures dominate (399, 93.9%), while a minority use canvas (26, 6.1%). *Metaphor-based compositions* account for 177 examples (29.4%), where the overall arrangement represents a symbolic referent (e.g., "grapes" collage in Fig. 2), leveraging viewers' prior knowledge for interpretation [23].

3.4 Design Goals

Our analysis of the corpus above reveals two central observations that directly inform the design of UnitoPia:

- **Visual Expressiveness Desired:** Pictorial unit visualizations achieve expressive power through both rich mark design (84.6% irregular marks) and diverse layout strategies, including organic arrangements (20.1%) and metaphor-based compositions (29.4%).
- **Manual Effort Required:** While these expressive forms are prevalent, they remain disproportionately difficult to create. The two designers on our coding team described current workflows as "labor-intensive" and "painfully manual", requiring hours of manual adjustments in tools such as Illustrator or Figma with no guarantees of consistency. In contrast, grid-based layouts are well-supported by existing tools [33] but represent only one end of the expressive spectrum.

This gap between *what people want to create* and *what tools easily support* motivates the **D**esign **G**oals of UnitoPia:

DG1: Embrace Visual Richness of Graphical Elements Without Compromise. The corpus shows that pictorial unit visualizations predominantly use irregular, representational icons (84.6% of irregular examples) and hand-drawn elements (13.2%). UnitoPia should support a wide range of visual marks, e.g., vector graphics, raster images, and hand-drawn elements, and preserve their internal design, including textures, complex geometric compounds, and other stylistic details, regardless of their source (e.g., Adobe Illustrator, Figma, or hand sketches). Also, UnitoPia should support flexible repetition patterns. This goal directly operationalizes *C1* (*Irregular Graphical Elements*) by ensuring that the creative richness of elements is never compromised during layout.

DG2: Support for Expressive Visual Layouts. The corpus reveals a rich diversity in layout, with organic layouts (20.1% of the corpus) remaining underserved by existing tools. Those organic layouts are dominated by packing (91.7% of organic), the rest are fluid configurations that evoke sedimentation and natural gathering. While grid layouts (50.1% of rule-based) are well-supported by current systems, these non-grid alternatives typically require extensive manual effort. UnitoPia should focus its computational power on these non-standard layouts that are difficult to specify through coordinates but can emerge naturally from metaphors. This goal directly operationalizes *C2 (Non-standard Visual Layout)*.

DG3: Enable Easy Construction Through Familiar Concepts. Our corpus analysis reveals that metaphor-based compositions appear in 29.4% of existing designs, indicating that designers instinctively employ semantic, relatable concepts (e.g., 'balance scales', 'footprint' collages) to organize visual elements. Despite this prevalence, creating such layouts still has to construct them manually from rigid, regular grid templates [33]. UnitoPia should bridge this gap by enabling specification through intuitive, everyday concepts rather than coordinates or optimization parameters. This design goal aligns with established HCI principles of leveraging user pre-existing mental models to lower learning barriers and support creative exploration [36].

DG4: Preserve Data Encoding Fidelity While Supporting Expression. Unlike purely artistic arrangements, pictorial unit visualizations should maintain rigorous data encoding fidelity, adhering to the core principles of conventional unit visualizations [33], i.e., each mark corresponds to a data item, and visual channels (size, color, orientation) encode attributes. UnitoPia shall guarantee that visual encoding and data-determined properties are preserved throughout the layout optimization process. This goal requires that expressive pictorial unit visualizations in UnitoPia does not mean arbitrary, and the data foundation must remain intact.

4 UnitoPia

UnitoPia is designed as a lightweight yet expressive authoring tool for creating pictorial unit visualizations, addressing the extensive manual effort currently required. Developed as a sketch-based interface, UnitoPia supports users in creating pictorial unit visualization via a contain-and-fill design model. In this section, we introduce the core design model (Section 4.1) and the sketch-based authoring interface (Section 4.2) that enables intuitive authoring through familiar metaphoric concepts (**DG3**). We then present the image-space layout method (Section 4.3) that powers expressive visual layouts (**DG2**) for arbitrarily designed graphical elements (**DG1**), while maintaining high data encoding fidelity (**DG4**).

4.1 Design Model

UnitoPia aligns with the basic design workflow of *constructive visualization* [14], mapping data units to individual tokens and assembling them. Unlike unit visualization frameworks that assemble units through coordinate systems, such as ATOM's Cartesian coordinates [33] or Wang et al.'s radial system [48], UnitoPia introduces a conceptually new design model using the metaphor of *contain-and-fill* to support the expressive visual arrangements in pictorial unit visualization.

4.1.1 *Contain-and-Fill* Design Metaphor

The effectiveness of metaphors for rapidly conveying new concepts through familiar ones is well-established [10], and demonstrated in prior metaphor-based interaction techniques [24, 36, 60]. Metaphors are particularly powerful in visualization authoring, where they allow users to reason about complex layout behaviors through intuitive, everyday experiences rather than abstract coordinates or optimization parameters. In UnitoPia, we draw on the universal physical experience of objects naturally packing and falling within a bounded space, and introduce a metaphoric design model, *contain-and-fill*, which models the expressive arrangement of visual elements in pictorial unit visualization as a filling problem in a container, specified with four tangible concepts, *Mark*, *Container*, *Emitter*, and *Force*.

Figure 3 illustrates a pictorial unit visualization and its instantiation of the four concepts. In this example, each data item is represented as a 'star' mark, with its size encoding a data value. A 'jar' container defines the area holding the marks, while an emitter, lining all stars on the top, efficiently determines where the marks are initialized. A force point is placed at the bottom of the jar, creating an attractive force field. During the layout process, the star marks are drawn toward this force point, gradually filling the container from the bottom up. The final result is a fully formed pictorial unit visualization that resembles stars settled down on the bottom of the jar (shown in Figure 3(a)).

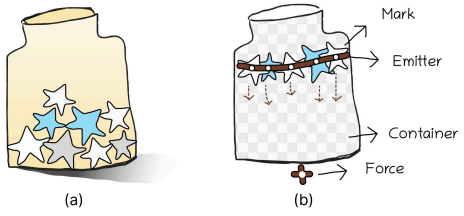
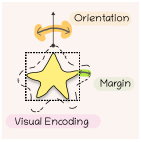


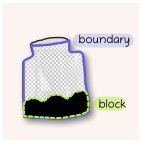
Figure 3: Illustration of four components in *contain-and-fill* design model: (a) a pictorial unit visualization example and (b) the corresponding setting of marks, container, emitter, and force.

In the above, we introduce the four core components through an intuitive example designed to align with a natural mental model of containment and filling. It is important to note that the expressiveness of the contain-and-fill model is presented beyond the force-like layout shown in this example. A versatile range of diverse layouts can be achieved by varying the properties of the four components in the model. Figure 1 presents a set of examples created using this model. Several scenarios are demonstrated in Section 4.2, showing the use of the contain and fill model. Following, we elaborate on each of the four components and their properties in detail.

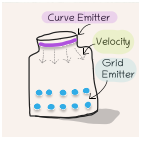
4.1.2 Components and Properties



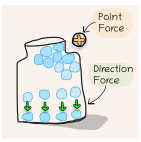
A **mark** is a graphical element used to encode a single data unit [3], in which data attributes are mapped to visual channels, such as size and color. In pictorial unit visualization, we support marks in a wide range of designs, mainly with two sources, i.e., *vector* graphics and *raster* images. For vector graphics (i.e., SVG files), they can be regular geometric objects (e.g., circles, rectangles), as well as compound vector primitives of glyphs and icons. Each mark can be assigned a *margin* property that controls its spacing relative to the surrounding elements. This margin can be either a constant value (e.g., 2% of the mark's size) or a data-driven value. For example, in the 'bottle of dreams' example shown in the first row of Figure 1, the bubble marks demonstrate increasing margins as they move upward. The *orientation* of each mark can be customized, such as being oriented towards the center of the container.



A **container** is the visual region on the canvas that defines where the unit marks are laid out [33]. It features a closed boundary that strictly constrains marks within the container. The entire canvas can be viewed as a special container. In pictorial unit visualization, there can be designs where the layout of one sub-unit visualization depends on the other sub-unit visualization. For example, in the sedimentation example in Figure 10(b), each layer of marks in the same color is a sub-unit visualization. The whole unit visualization is created by layering one sub-unit visualization on top of the other. To manage the layout dependencies among sub-unit visualizations, the *block* defines exclusion areas where marks are not allowed. In Figure 10(b), all of the sub-unit visualizations are set with the same container but with different blocks accumulated from their previous sub-unit visualizations. That is, given a sub-unit visualization, the identities of all sub-unit visualizations beneath it will be added as blocks.



An **emitter** is the component that quickly initializes mark placement in the visualization. Our framework supports two primary types, *grid emitter* (i.e., 2D) and *curve emitter* (i.e., 1D). The grid emitter distributes the marks uniformly across the container in a grid arrangement. For curve emission, marks are placed along a cubic Bézier curve. Mark ordering in emitters can be specified. The *velocity* of the emitter defines the *speed* and *direction* that marks leave the emitter. If the speed is zero, marks stay where they are initialized, creating a rigid grid or linear layout. The velocity direction defines where the marks leave the emitter. Specifically, for the 1D layout shown in Figure 1, the direction is set along the emitter curve.



A **force** acts like a magnet or gravity that attracts marks. The concept of force aligns well with previous work on organic layouts, such as sedimentation bar chart [16] and Dust & Magnet (DnM) [60]. Our framework introduces two types of forces, *point force* and *directional force*. A point force originates from a point specified by force position and influences marks based on their proximity to that point, while a directional force acts uniformly in a specific direction, influencing marks to move along that direction.

Figure 4: UnitoPia's interface and its sketch-based interactions: the interface consists of (A) a dataset panel, (B) a mark panel, (C) a resource library, (D) a main canvas, and (E) a pop-out mark design panel. Via sketching and direct manipulation, users can easily define the four components to create a pictorial unit visualization.

4.2 User Interface and Interactions

Inspired by prior work on interactive vector graphics and personalized visualization [18, 32], we implemented UnitoPia as a sketch-based interface with drag-and-drop interactions. Figure 4 shows the system interface, which comprises five main components: (A) the dataset panel for uploading and managing data; (B) the mark panel for customizing marks; (C) the resource library containing reusable images and vectors; (D) the main canvas for sketching and direct manipulation; and (E) a pop-up panel for mark design and visual encoding configuration. UnitoPia can run as a web-based application that supports pen-and-touch enabled tablets. We illustrate the use of UnitoPia through three example scenarios. A demonstration video and an interactive demo are available on the project website.

4.2.1 Interactions

Data-Mark Bind. After uploading the dataset file, data fields are extracted and groups are automatically detected for categorical fields in dataset panel (A). Users can select the data to visualize by dragging and dropping the corresponding data field tag into the mark panel (B), to create groups of marks. In (B), users can click a mark group to further customize it in panel (E). By a drag-and-drop, users can bind a data field to the visual channels of the mark, e.g., its size, width.

Resource Upload. Users can upload visual design resources to the library in Panel (C). For mark resource, UnitoPia supports vector resources (in SVG format) and bitmap images (e.g., PNG format). For containers, users can upload a binary bitmap image to the library panel (C). Users can directly drag the uploaded container image to the canvas panel for use.

Free-form Sketch. Besides to use the uploaded resources, users can also do free sketching to design marks on the canvas in Panel (E). Similarly, users can do free-form sketch to create a container on the main canvas panel (D). The sketch is supported with a toolkit with essential functions, including eraser, color style, stroke width, etc.

Drag-and-drop Direct Manipulate. In the canvas panel (D), users can switch among 'container', 'emitter', and 'force' by clicking the buttons on the left toolbar, and direct manipulate them on the canvas. Once set, users can drag and drop marks into the container or emitter if it is defined for initialization. Then by clicking the 'Run' button at the top right, marks will be laid out, generating the pictorial unit visualization.

Canvas Layering and Block. UnitoPia supports multiple containers organized in layers, similar to Photoshop's layer-based editing. Users can create new layers or duplicate existing ones. A key feature is the *block* mechanism, where a layer can designate the result of another layer as an exclusion zone, where marks in later layers avoid overlapping with those in earlier layers. This enables effects like sedimentation, where each layer accumulates atop the previous without interference, as demonstrated in the use example (Section 4.2.4).

4.2.2 Scenario 1: Reproduction of a Piece of Dear Data

As shown in Figure 5, we reproduce a piece of pictorial unit visualization from the Dear Data book [28] for a new dataset with fields `Task` and `Duration`. By dragging the data field tag *Task* into the mark panel, we create a group of marks. We then upload a screenshot image of the sketchy-style graphical element from the original design and bind it to the marks. Next, we customize the marks by directly dragging and dropping the `Duration` attribute onto the size visual encoding. In the main canvas panel, we upload the data-removed image of this Dear Data example as the background image and sketch an area to serve as a container. After dragging the marks onto the container, we click the 'Run' button and obtain a fully packed layout. Then we adjust the marks' margin and run again, which reproduces the final result with the loosely packed layout in a closed container.

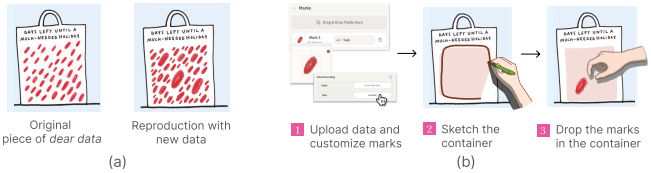


Figure 5: An example of reproducing a piece of pictorial unit visualization in *Dear Data* [28] for new data: (a) the original and reproduced one, (b) key interaction steps. The background image copyright *Dear Data* by Giorgia Lupi and Stefanie Posavec [28].

4.2.3 Scenario 2: Organic Packing Trees of Yearly Production

As shown in Figure 6, this example visualizes two unit groups juxtaposed left and right, each arranged in a centric packing layout with open organic boundaries. We first load a dataset of weekly production amounts, with attributes `Year` (2025, 2026) and `Production Amount`. Two groups of marks are created based on the `Year` attribute, and each mark within a group is bound to a custom-designed SVG file, with its size encoded to the `Production Amount`. A background image featuring two tree branches is then loaded. For each tree, we define a rectangular container to constrain the marks. A point force is placed at the center of each container as an attraction point. The two groups of marks, each configured with a specified margin, are dropped into

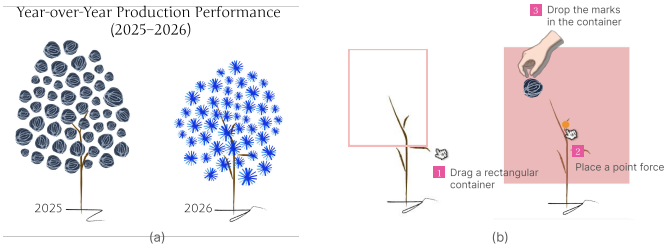


Figure 6: An example of pictorial unit visualization with organic packing using soft, open boundaries: (a) visualization of `production amounts` for two years; (b) key interaction steps.

their respective containers. Upon clicking the "Run" button, the system generates a pictorial unit visualization with organic packing without hard defined boundaries.

4.2.4 Scenario 3: Sedimentary Layers of Flower Blooms

Figure 7 shows an example of a sedimentary pictorial unit visualization, created using forces, emitters, and the layered block mechanism. The dataset captures *flower blooms throughout the season*, with each data unit representing a bloom event and numerical fields for `Bloom Count` and `Flower Type`. We sketch flower shapes for each `Flower Type`, and bind them to the group. Then we upload a container image and define a curve emitter on the top, a downward force. We then duplicate this setup to create two other layers. For the three layers, the three types of flowers are dropped onto the emitter line respectively. The blocks are then configured to create layered accumulation, allowing marks to settle on top of previous layers without overlapping.

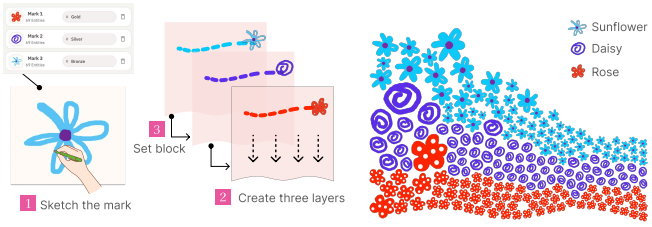


Figure 7: An example of sedimentation unit visualization of three types of flowers blooming over seasons, and the key interaction steps.

4.3 Image-space Layout Optimization

The power of UnitoPia to accommodate expressive layouts with units with visual richness is enabled by an image-space-based layout optimization method.

Layout Problem in Pictorial Unit Visualization. In unit visualization, the layout problem can be modeled as an optimization problem that rigidly transforms marks to satisfy certain design constraints, such as ensuring non-overlapping elements, achieving full packing, and enforcing spatial alignment. Given the list of marks $M = (m_1, m_2, \ldots, m_n)$, each mark m_i is associated with a parameter tuple for rigid transformation, translation, rotation and scaling ($\langle t_i, r_i, s_i \rangle$). Then the layout problem can be formalized as follows:

$$(\mathbf{t}^*, \mathbf{r}^*, s^*_{global}) = \mathbf{t},\mathbf{r},s_{global} \; \mathscr{L}(M, \mathbf{t}, \mathbf{r}, s_{global}),$$

where $\mathscr{L}$ is the layout loss to be minimized, $\mathbf{t} = (t_i)_{i=1}^n$ and $\mathbf{r} = (r_i)_{i=1}^n$ are the lists of translation and rotation parameters for all marks. In unit visualization, the size (or width or height) of marks often encodes data, therefore it is essential to apply a uniform s_{global} to scale marks by the boundary box to ensure the fidelity of the data encoding in size (or width or height). The goal of a layout algorithm is to find the optimal solution $(\mathbf{t}^*, \mathbf{r}^*, s^*_{global})$ which minimizes the loss. Different designs of the loss functions drive different layouts.

Object Space versus Image Space. While defining layout loss in object space is manageable for regular geometries (e.g., circle packing [48]), it fails to generalize to irregular shapes and flexible layouts due to the need for sophisticated geometric descriptors [20]. For example (see the insect), evaluating intersections in object space requires complex mathematical descriptions of shape boundaries (∂A and ∂B). In contrast, it is more intuitive to measure the intersection between A and B in the image space when A and B are rasterized into pixel grids. We can simply count the number of pixels in their joint region, regardless of what shapes they are. This idea of optimizing layout in the image space is enabled by differentiable rendering [22] and has been justified in related collage and packing tasks [52]. UnitoPia leverages this paradigm to power our contain-and-fill model. Compared to traditional object-space benchmarks that rely on complex geometric collision checking, our pixel-level optimization generalizes seamlessly to arbitrary custom silhouettes without custom engineering.

4.3.1 Overview

Figure 8 shows the pipeline of our method, which complies with the classic InfoVis pipeline [6], i.e., mapping data into visual content and arranging them by a layout while maintaining faithful data encoding.

Given a dataset D with n data items as input, marks $M = (m_1, m_2, \ldots, m_n)$ visually encode the items. Each mark m_i then undergoes a geometric transformation parameterized by $\langle t_i, r_i, s_{global} \rangle$. All transformed marks are subsequently rendered into a raster image $\hat{I}$ through differentiable rendering. The image-space loss function $\mathscr{L}$ is then computed between $\hat{I}$ and the container image I_c, to measure the current layout according to the configuration set via our layout model of marks, container, emitter, and forces. Then, this loss is backpropagated to optimize the transformation parameters of marks until convergence.

The backpropagation from image-space loss to transformation parameters in object space is enabled by integrating a differentiable renderer in our pipeline. Differentiable rendering is a group of rasterization techniques that convert geometric objects (marks in our case) into raster images while maintaining gradient flow. This process can be formulated as follows.

$$\hat{I} = f(M, \mathbf{z}), where \quad \mathbf{z} = \langle \mathbf{t}, \mathbf{r}, s_{global} \rangle \tag{1}$$

The key feature of differentiable rendering is that the renderer f is differentiable from output pixels back to input parameters $\mathbf{z}$. There are different implementations of differentiable rendering, such as implementing the renderer as a neural network [30]. We adopted the differentiable rendering implementation for vectors by Li et al. [22]. Therefore, the geometric transformation parameters $\mathbf{z}$ can be optimized following the derivative chains:

$$\mathbf{z} := \mathbf{z} - \frac{\partial \mathscr{L}(\hat{I}, I_c)}{\partial \hat{I}} \cdot \frac{\partial \hat{I}}{\partial \mathbf{z}}. \tag{2}$$

4.3.2 Image-based Loss

To realize the expressiveness afforded by the contain-and-fill design model, we introduce a set of image-based loss functions. These loss functions are selectively activated based on user interactions in the UnitoPia interface, e.g., forces, container boundaries.

Basic Constraint I: Non-overlapping Marks. Overlapping damages the identity perception of unit visualization. Therefore, we prioritize non-overlapping among marks as a basic requirement for pictorial unit visualization. Following collage work [52], marks are differentiable rendered into an image $\hat{I}_{opacity}$, with certain opacity τ (set to 0.3 in this work), in which the overlapped areas have opacity larger than τ, defined as following where α_p is the opacity of the pixel p:

$$\mathscr{L}_{overlap} = \frac{1}{w \cdot h} \sum_{p \in \hat{I}_{opacity}} ReLU(\alpha_p - \tau). \tag{3}$$

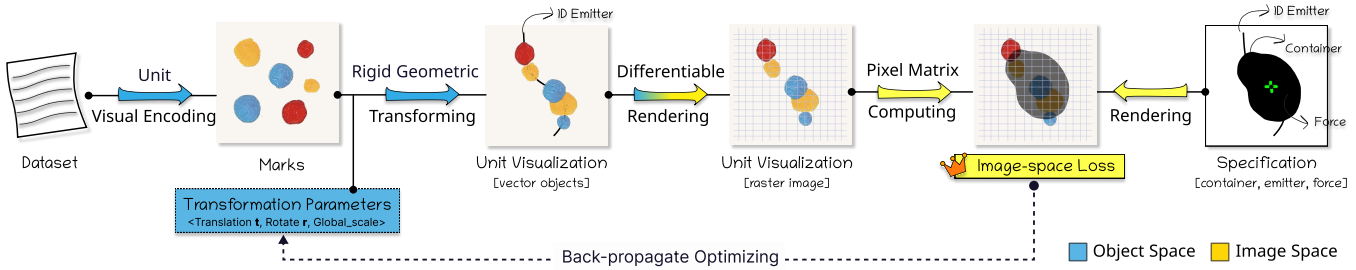


Figure 8: Pipeline of image-space based layout method: The input dataset is visually encoded into marks, which undergo geometric transformation (translation, rotation, and global scaling) and are rasterized to an mark image via differentiable rendering. Then the image-space loss is calculated and back-propagated to optimize the geometric transformation parameters, which marks are moved, rotated, scaled until the process converged.

Basic Constraint II: Container Containment. Given the container customized by users, marks are required to stay within its boundary without overflowing. The container containment loss is computed as a weighted MSE (mean square error) between two images, the image of masks $\hat{I}_{marks}$ and the container image $\hat{I}_{container}$. Both images are binary images with objects (masks or containers) rendered as black areas:

$$\mathscr{L}_{contain} = \frac{1}{w \cdot h} W \odot \left\| \hat{I}_{b\&w} - I_C \right\|^2, \tag{4}$$

where the penalty mask $W \in R^{w \times h}$ assigns weights to exclusive pixel in the difference image. A mild penalty weight ($w_{ij} = 1$) is assigned for unoccupied pixels inside the target container to encourage mark coverage. A severe penalty ($w_{ij} = 100$) is assigned to pixels outside the container to strongly discourage mark overflow.

Optional Constraint I: Force Attraction. In UnitoPia, marks can be attracted towards a *point force* or *directional force* if specialized. Given a unit vector of direction (v_x, v_y), the attraction effect can be quantified by calculating the sum of projections of the set of centroid points (c_{xi}, c_{yi}) of each visual primitive onto the given attracting vector v:

$$\mathscr{L}_{force} = \sum_i (c_{xi} v_x + c_{yi} v_y), \tag{5}$$

where the centroid (c_{xi}, c_{yi}) of a primitive can be computed by offsetting its initial centroid point with the translation $(t_x, t_y)_i$. Note that this loss serves as an additional term in the optimization process to simulate the attraction effect. Consequently, we can effectively implement force attraction effects in visualization tasks, thereby enriching the visualization results and enhancing visual impacts.

Optional Constraint II: Mark Orientation. Marks can be specified with orientation preference, such as *to_center_pos*, constraining marks to orient towards the center point. Therefore, given a mark m_i, we compute two vectors, the current orientation of the mark $\mathbf{u_i}$ and the desired orientation of the mark $\mathbf{v_i}$. The *orientation alignment* loss is defined as the dissimilarity between$\mathbf{u_i}$ and $\mathbf{v_i}$, based on the cosine similarity, as following:

$$\mathscr{L}_{orient} = \sum_i \left(1 - \frac{\mathbf{u_i} \cdot \mathbf{v_i}}{\|\mathbf{u_i}\| \|\mathbf{v_i}\|}\right). \tag{6}$$

Overall Loss. The overall loss function combines the losses described above. The *overlap* loss is always applied, while the remaining terms receive conditional parameters from the UnitoPia interface (e.g., force vector (v_x, v_y)) and some are activated only when the corresponding features are enabled. The loss is defined as:

$$\mathscr{L} = \alpha \mathscr{L}_{overlap} + \underbrace{\beta \mathscr{L}_{contain} + \gamma \mathscr{L}_{force} + \delta \mathscr{L}_{orient}}_{Customized in UnitoPia's Interface}, \tag{7}$$

where weights from α to δ are set to control the impact of factors during the layout optimization process. In our experiment, the weights are 1, 7e4, 1, and 1e2, respectively.

5 Evaluation

5.1 Use Cases

UnitoPia supports a wide range of pictorial unit visualizations (Fig.1). More examples are available on the project homepage. Below, we highlight connections between pictorial unit visualization and related topics in visualization research.

Unit Data Storytelling. Our method supports generating a series of pictorial unit visualizations for data storytelling. Figure 9 demonstrates this capability through a narrative depicting *researchers and publications in the visualization field from 1986 to 2024* [47]. Across all three slides, each of the top 200 researchers is represented as a mark, with its size encoding the `publication number`. This consistent visual encoding maintains data identify as well as perceptual continuity across slides, enabling viewers to follow the narrative without reinterpreting the marks.

Engaging Animation Effects. The gradient descent of our method intrinsically creates smooth animation during the optimization. An animation effect that mimics physical phenomena such as gravity and attraction diffusion can be easily created using force. For example, Figure 10(a) illustrates the animation effect of attracting towards a point created with a point force, and Figure 10(b) shows the animation effect of 'falling down' created with a downward directional force.

5.2 User Study

We conducted a user study to evaluate UnitoPia's effectiveness in supporting the creation of pictorial unit visualizations. Our study aimed to assess how the system supports its four design goals (Section 3.4): visual-rich marks (DG1), expressive layouts (DG2), intuitive authoring (DG3), and data fidelity (DG4). We targeted both designers and non-experts to understand the system's accessibility and expressive power across different skill levels. Details on the task design, questionnaire, and study data are provided in the supplementary material and project homepage.

Figure 9: Storytelling with three slides of pictorial unit visualizations, where researcher are represented as marks, with size encoding their publication number.

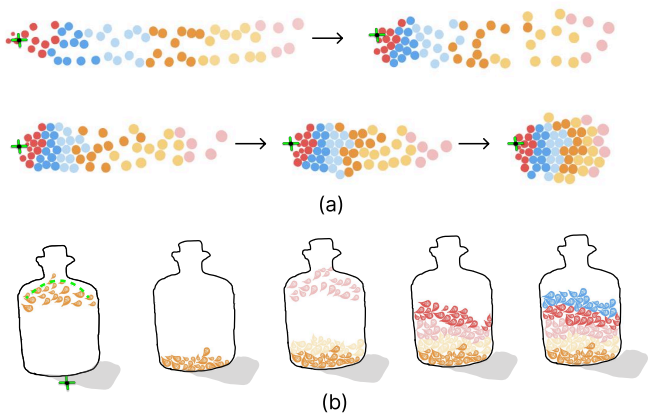


Figure 10: Smooth animation during the layout optimization: (a) attracting visual effect by a point force, (b) falling down visual effect created via a downward directional force.

5.2.1 Study Design

Apparatus. UnitoPia was implemented as a web-based application a standard web browser. For the study, we deployed UnitoPia on a 13.3-inch Wacom Pen Tablet, where pen and touch input can be simultaneously detected and differentiated, enabling users to sketch containers, place forces, and draw emitter curves directly on the canvas.

Participants. We recruited nine participants (3 female, 6 male; ages 22–25, $M = 23.8$, $SD = 0.87$) from university-affiliated design programs and local design communities. Five participants (D1–D5) were professional designers with 2–6 years of experience in graphic design, UI/UX, or information visualization. The other four (P1–P4) were non-experts who reported no formal design training but had basic familiarity with digital tools and data visualization concepts. All participants reported prior experience with data visualization (e.g., creating charts in Excel or Tableau), but none had used a sketch-based visualization authoring tool before. All participants provided informed consent for this routine software evaluation, which our institution exempts from formal IRB review as it involves no invasive equipment or sensitive data.

Procedure and Tasks. Participants first completed a demographics questionnaire, followed by a training session. Two task sessions were then conducted in sequence: reproduction and a free-form exploration. Afterward, participants provided feedback through a questionnaire and an interview.

Demonstration and Training. The experimenter introduced the contain-and-fill design model and demonstrated the four concepts of UnitoPia, including customizing marks, sketching containers, placing forces, defining emitters, and binding data attributes, via a walkthrough example. As the exercise, participants were instructed to replicate the exact same example, during which they were encouraged to seek guidance and ask questions. They were also invited to freely explore the interface and try all available functions.

Task I: Reproduction. In this session, participants were asked to complete three design reproduction tasks. The three replication examples were designed for a fair coverage of key concepts and functions: one example with one container, one with multiple containers, and another one with a layering mechanism. For each task, the target pictorial unit visualization was shown to participants as a static image, together with an explanation of visual encoding. Then, participants were asked to reproduce a target pictorial unit visualization for a new dataset. The new dataset was provided to participants, as well as corresponding design resources, including SVG or PNG images for marks, background image, etc. Participants were encouraged to think aloud. The experimenter observed but did not intervene unless participants became stuck.

Task II: Free-form Exploration. After the reproduction session, participants took a 10-minute break and then began the free-form exploration session. In this session, participants were asked to freely create pictorial unit visualizations using UnitoPia. They could use either the datasets from the previous session or any new datasets of their choice. The experimenter observed their entire creation process.

Questionnaire and Interview. After completing both tasks, participants filled out a questionnaire assessing their experience with UnitoPia using a 7-point Likert scale (1 = strongly disagree, 7 = strongly agree). The questionnaire consisted of 8 custom items directly mapping to the four design goals (2 items per goal).

5.2.2 Results

All of the participants successfully reproduced the target charts for the first two tasks (one container and multiple containers). For the third task, which required the use of a layering mechanism, five participants completed it without any hints, and two completed it with only a few hints. Two participants failed to create layers. In free-form exploration, all participants succeeded in creating designs as they desired. Figure 11 shows some examples that the participants created.

Figure 11: Design examples created by participants in the free-form exploration stage.

Subjective Ratings. Strong ratings were received for the satisfaction of all four design goals. For *DG1: Visual-rich Elements* (M = 6.33/7, SD = 0.61), participants appreciated that marks retained their integrity. P1 observed that imported SVGs appeared "perfect" with "no distortion." D3 noted hand-drawn icons stayed "exactly as I drew them." For *DG2: Expressive Layouts* (M = 6.22/7, SD = 0.83), organic layouts were highly rated: D2 remarked that "the dynamic implementation feels very 'comfortable,'" and P4 described the force and emit progress as "natural and satisfying to watch." *DG3: Intuitive Authoring* received the highest ratings (M = 6.44/7, SD = 0.63). D1 praised that the combination of automatic initialization, forces, and filling effectively provides design convenience and enhances efficiency, compared to the "dozen operations" required in traditional XY coordinate systems. P2 noted the features are very rich, yet the operational logic remains simple, so users don't have to overthink while operating, because it "made immediate sense." For *DG4: Data Fidelity* (M = 6.39/7, SD = 0.55), participants trusted the system to preserve encodings: D1 valued that the progressive animation reinforced his confidence in the visual encodings.

Table 1: Performance metrics across varying mark counts. Values represent means and standard deviations across 10 runs.

Marks	Total Time (s)	Frame Rate (FPS)	Time/Iter. (s)
200	69.7 ± 7.75	95.1 ± 3.8	0.49
400	78.27 ± 13.37	42.7 ± 3.7	0.57
800	94.69 ± 8.63	19.3 ± 0.6	0.71
1,000	103.75 ± 13.19	17.3 ± 0.6	0.78
2,000	147.04 ± 24.91	7.4 ± 0.4	1.09
5,000	250.17 ± 36.92	2.8 ± 1.9	1.99

Qualitative Findings. Below we report the qualitative comments from the semi-structured interviews, revealing several recurring themes: *Separation of Concerns.* Multiple participants noted that UnitoPia separated creative vision from technical execution. D3 explained: "I focused on what I wanted: using bubbles to represent tweets, with their size indicating the number of likes, filled within a bubble wand. The system handled the how and where exactly each bubble goes, and how they settle without overlapping. That's a powerful division of labor." P3 echoed: "The time from concept to implementation is quite short, because I didn't have to think about coordinates. I just thought about what I wanted it to look like."

Sketching as Natural Expression. Participants found that sketching interactions aligned seamlessly with the contain-and-fill design model, enabling creative yet efficient authoring power. P4 noted: "Drawing patterns on the canvas and sketching boundaries on the background image feels like playing." D2 added: "Sketching allows for a broader creative space."

Arbitrary Shapes in Arbitrary Containers. Participants valued the system's ability to accommodate marks from diverse sources, whether hand-drawn sketches, raster images, or vector icons from design tools. D2 noted: "It is really impressive and powerful; I can easily load and use icons or drawings that I like directly." P2 remarked: "The sketching feature and the material import function effectively accommodate the user experience for both professional designers and general users."

5.3 Performance Evaluation

We conducted a performance evaluation to assess UnitoPia's scalability and responsiveness under varying mark counts using two key metrics: (1) *optimization time*, the total time required for the layout to converge, measured as the average time per optimization iteration to indicate computational efficiency and potential for real-time interaction; and (2) *frame rate*, the rendering smoothness during optimization, measured in frames per second (FPS). This second metric is particularly important for animation effects that users observe during layout evolution.

5.3.1 Experimental Setup

We tested six mark counts (200, 400, 800, 1,000, 2,000, and 5,000), each averagely over 10 runs of layout optimization in a fixed circular container with bottom point force, up to 1000 iterations per test, with early stopping if the loss did not change for 12 consecutive iterations. We report the average measurements of 10 runs.

We used a fixed image resolution in differentiable rendering (1440×830), aligned with the resolution of the main canvas in UnitoPia on a 24-inch screen. The higher resolutions would increase per-iteration time, while lower resolutions might degrade visual quality. The reported results should be interpreted as representative of typical use rather than exhaustive across all possible configurations. We evaluated UnitoPia on a workstation running Ubuntu 22.04 LTS with an AMD Ryzen 9 7950X (4.5 GHz), 128 GB RAM, and an NVIDIA GeForce RTX 4090 GPU (24 GB VRAM).

5.3.2 Results

Optimization Time. Table 1 summarizes the performance measurements across mark counts. Optimization time exhibited a near-linear growth with mark count up to 2,000 marks. Beyond 2,000 marks, however, the growth in time relative to the increase in mark count became less pronounced. When the marks increased from 2,000 to 5,000, the total optimization time increased to 250.17±36.92 s, a rise of 103.13 s from the case of 2,000 marks. This represents a substantially lower relative growth in time (about 1.7×) compared to the relative growth in mark count (2.5×), indicating that the scaling of optimization time remains better than linear even at higher mark densities.

Frame Rate. Frame rate decreased as mark count increased. For layouts with 200 to 800 marks, frame rates remained around 20 FPS or above, meeting standard thresholds for smooth animation. For 1,000 marks, the average frame rate dropped to 17 FPS, still acceptable for viewing layout evolution but less fluid. At 2,000 marks, frame rate fell below 10 FPS, and at 5,000 marks, below 5 FPS, resulting in visibly choppy animation.

Time per Iteration. Time per iteration grows approximately linearly with the number of marks, increasing from 0.49 seconds at 200 marks to 1.99 seconds at 5,000 marks. This suggests the algorithm's computational complexity is roughly O(n), corresponding to a near-constant per-mark processing cost.

Scalability Implications. As the number of marks increases, the total runtime and per-iteration time of the backend algorithm scale near-linearly, maintaining predictable performance. On the front end, however, the refresh rate (frame rate) drops significantly as the number of marks grows, especially beyond 1,000 marks. For workloads with up to 1,000 marks, the frame rate remains around 20 FPS, which is sufficient to support interactive creation of expressive pictorial-unit visualizations with mark counts up to 1000.

6 Limitation Discussion

In this section, we discuss several limitations identified from the participant feedback and observed behaviors in the user study.

Unit visualization with nested layouts. Currently, UnitoPia supports multiple containers and a stack layering mechanism. However, as noted by Park et al. [33], unit visualizations may employ nested layouts, where data points are positioned according to their attributes and may occlude one another. Superposition layout can also occur in visual decorations or multi-layered data representations [53]. Our current work does not address pictorial unit visualizations with superimposed layouts well, remaining as an interesting direction for future exploration.

Compound visual perception and encoding of marks. Currently, UnitoPia supports several visual encodings of irregular marks, including width, height, and size. However, even with uniform scaling, irregular geometries can induce perceptual biases. Quantifying size perception for such arbitrary shapes is essential but out of this paper's scope. Also, it does not yet incorporate compound visual encoding within marks. Enabling multiple visual encodings for data attributes would be useful for glyph-based design. An exciting future direction is to integrate the unit design module into our framework, offering functionality similar to data-graphics binding techniques [18, 25].

Loose v.s. Hard Layout Control. UnitoPia leverages a contain-and-fill metaphor, offering a loose layout method that bypasses precise individual positioning. Unlike conventional tools like Charticulator [34] or Data Illustrator [25] that enforce hard geometric layouts, UnitoPia prioritizes a high-level physics metaphor. While our framework excels at generating organic layouts, achieving hard control (e.g., axis alignment or grid snapping) remains difficult under global optimization. Integrating local manual overrides into this loose optimization framework represents an interesting future direction.

7 Conclusion

In this work, we introduce UnitoPia, an interactive authoring tool for creating pictorial unit visualizations. Unlike conventional coordinate-based design paradigms, UnitoPia adopts a novel design model leveraging a contain-and-fill metaphor. Driven by an image-space layout optimization method via differentiable rendering, this model aligns seamlessly with intuitive real-world physical experiences, as validated through our user study. While this metaphor-based automation lowers technical barriers, it inherently prompts a reflection on the delicate balance between algorithmic assistance and individual creative autonomy. Ultimately, we believe UnitoPia opens new possibilities at the intersection of artistic illustration and data-driven storytelling, inspiring future research in more expressive and personalized unit visualizations.

Acknowledgments

Thanks to Yanhui Guo and Pu Wang for the help in corpus labeling and analysis. This work was supported in part by the National Natural Science Foundation of China (62472288) and the Guangdong Natural Science Foundation (2026A1515010423). Yoojin Jeon and Bongshin Lee were supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT), Artificial Intelligence Graduate School Program, Yonsei University, under Grant RS-2020-II201361.

References

[1] B. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pp. 71–80. Association for Computing Machinery, New York, 2001. doi: 10.1145/502348.502359

[2] M. A. Borkin, A. A. Vo, Z. Bylinskii, P. Isola, S. Sunkavalli, A. Oliva et al. What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2306–2315, 2013. doi: 10.1109/TVCG.2013.234

[3] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1121–1128, 2009. doi: 10.1109/TVCG.2009.174

[4] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011. doi: 10.1109/TVCG.2011.185

[5] J. Boy, A. V. Pandey, J. Emerson, M. Satterthwaite, O. Nov, and E. Bertini. Showing people behind data: Does anthropomorphizing visualizations elicit more empathy for human rights data? In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*, pp. 5462–5474. ACM, New York, 2017. doi: 10.1145/3025453.3025512

[6] S. K. Card, J. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[7] F. Chevalier, P. Dragicevic, and C. Hurter. Histomages: fully synchronized views for image editing. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pp. 281–286. Association for Computing Machinery, New York, 2012. doi: 10.1145/2380116.2380152

[8] T. N. Dang, L. Wilkinson, and A. Anand. Stacking graphic elements to avoid over-plotting. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1044–1052, 2010. doi: 10.1109/TVCG.2010.197

[9] DataViz Project. Pictorial unit chart. `https://datavizproject.com/data-type/pictorial-unit-chart/`. Accessed: Mar. 31, 2025.

[10] A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction*. Prentice-Hall, Englewood Cliffs, NJ, 2nd ed., 1997.

[11] S. M. Drucker and R. Fernandez. A unifying framework for animated and interactive unit visualizations. Technical Report MSR-TR-2015-65, Microsoft Research, Redmond, WA, USA, 2015.

[12] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, May 2010. doi: 10.1109/TVCG.2009.84

[13] J. Feinberg. Wordle: Beautiful word clouds, 2009.

[14] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer. Constructive visualization. In *Proceedings of the 2014 conference on Designing interactive systems*, pp. 433–442, 2014. doi: 10.1145/2598510.2598566

[15] S. Huron, Y. Jansen, and S. Carpendale. Constructing visual representations: Investigating the use of tangible tokens. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2102–2111, 2014. doi: 10.1109/TVCG.2014.2346292

[16] S. Huron, R. Vuillemot, and J.-D. Fekete. Visual sedimentation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2446–2455, 2013. doi: 10.1109/TVCG.2013.227

[17] D. A. Keim, M. C. Hao, U. Dayal, and M. Hsu. Pixel bar charts: a visualization technique for very large multi-attribute data sets. *Information Visualization*, 1(1):20–34, 2002. doi: 10.1109/INFVIS.2001.963288

[18] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic et al. Data-driven guides: Supporting expressive design for information graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):491–500, Jan. 2017. doi: 10.1109/TVCG.2016.2598620

[19] R. Kosara. Visualization criticism-the missing link between information visualization and art. In *2007 11th International Conference Information Visualization (IV'07)*, pp. 631–636. IEEE, 2007. doi: 10.1109/IV.2007.130

[20] K. C. Kwan, L. T. Sinn, C. Han, T.-T. Wong, and C.-W. Fu. Pyramid of arclength descriptor for generating collage of shapes. *ACM Trans. Graph.*, 35(6):229–1, 2016. doi: 10.1145/2980179.2980234

[21] X. Lan, Y. Wu, and N. Cao. Affective visualization design: Leveraging the emotional impact of data. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1–11, 2024. doi: 10.1109/TVCG.2023.3327385

[22] T.-M. Li, M. Lukáč, M. Gharbi, and J. Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. doi: 10.1145/3414685.3417871

[23] Y.-N. Li, D.-J. Li, and K. Zhang. The impact of metaphors on information visualization. *J. Vis.*, 20(3):487–504, Aug. 2017. doi: 10.1007/s12650-016-0371-9

[24] C. Liu, Y. Zhang, C. Wu, C. Li, and X. Yuan. A spatial constraint model for manipulating static visualizations. *ACM Trans. Interact. Intell. Syst.*, 14(2), art. no. 12, 29 pp., June 2024. doi: 10.1145/3657642

[25] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg et al. Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 1–13. Association for Computing Machinery, New York, 2018. doi: 10.1145/3173574.3173697

[26] M. Lu, S. Wang, J. Lanir, N. Fish, Y. Yue, D. Cohen-Or et al. Winglets: Visualizing association with uncertainty in multi-class scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):770–779, 2020. doi: 10.1109/TVCG.2019.2934811

[27] G. Lupi. Data humanism: the revolutionary future of data visualization. *Print Magazine*, 30(3), 2017.

[28] G. Lupi and S. Posavec. *Dear Data*. Princeton Architectural Press, 2016.

[29] A. Marshack. *The Roots of Civilization: The Cognitive Beginnings of Man's First Art, Symbol and Notation*. McGraw-Hill, New York, 1972.

[30] R. Nakano. Neural painters: A learned differentiable constraint for generating brushstroke paintings. Technical Report arXiv:1904.08410, arXiv, 2019. [cs.CV]. doi: 10.48550/arXiv.1904.08410

[31] O. Neurath. *International Picture Language: The First Rules of Isotype; with Isotype Pictures*. Paul, Trench, Trubner, 1936.

[32] A. Offenwanger, T. Tsandilas, and F. Chevalier. Datagarden: Formalizing personal sketches into structured visualization templates. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1268–1278, 2025. doi: 10.1109/TVCG.2024.3456336

[33] D. Park, S. M. Drucker, R. Fernandez, and N. Elmqvist. Atom: A grammar for unit visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3032–3043, 2018. doi: 10.1109/TVCG.2017.2785807

[34] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):789–799, 2019. doi: 10.1109/TVCG.2018.2865158

[35] H. Rosling's. 200 countries, 200 years, 4 minutes—the joy of stats—bbc four, 2010.

[36] J. M. Rzeszotarski and A. Kittur. Kinetica: naturalistic multi-touch data visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pp. 897–906. Association for Computing Machinery, New York, 2014. doi: 10.1145/2556288.2557231

[37] R. A. Saputra, C. S. Kaplan, and P. Asente. Improved deformation-driven element packing with repulsionpak. *IEEE transactions on visualization and computer graphics*, 27(4):2396–2408, 2019. doi: 10.1109/TVCG.2019.2950235

[38] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Computer Graphics Forum*, vol. 33(3), pp. 351–360. Wiley Online Library, 2014. doi: 10.1111/cgf.12391

[39] 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

[40] D. Seyser and M. Zeiller. Scrollytelling–an analysis of visual storytelling in online journalism. In *2018 22nd international conference information visualisation (IV)*, pp. 401–406. IEEE, 2018. doi: 10.1109/iV.2018.00075

[41] Shape Collage. Shape collage – automatic photo collage maker. [Online] `http://www.shapecollage.com/`. Accessed: Jul. 8, 2024.

[42] H. Shi, Y. Wang, J. Chen, C. Wang, and B. Lee. Piccl: Data-driven composition of bespoke pictorial charts. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):714–724, 2026. (Early Access). doi: 10.1109/TVCG.2025.3634264

[43] Y. Shi, P. Liu, S. Chen, M. Sun, and N. Cao. Supporting expressive and faithful pictorial visualization design with visual style transfer. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):236–246, 2023. doi: 10.1109/TVCG.2022.3209486

[44] B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, 1983. doi: 10.1145/800276.810991

[45] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on visualization and computer graphics*, 8(1):52–65, 2002. doi: 10.

1109/2945.981851

[46] F. Viegas, E. Perry, E. Howe, and J. Donath. Artifacts of the presence era: Using information visualization to create an evocative souvenir. In *IEEE Symposium on Information Visualization*, pp. 105–111, 2004. doi: 10.1109/INFVIS.2004.8

[47] Vispubs – visualization publications database. [Online]. Available: https://vispubs.com/. Accessed: Jun. 30, 2026.

[48] W. Wang, H. Wang, G. Dai, and H. Wang. Visualization of large hierarchical data by circle packing. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 517–520, 2006. doi: 10.1145/1124772.1124851

[49] Y. Wang, X. Chu, K. Zhang, C. Bao, X. Li, J. Zhang et al. Shapewordle: Tailoring wordles using shape-aware archimedean spirals. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):991–1000, 2020. doi: 10.1109/TVCG.2019.2934783

[50] Y. Wang, A. Segal, R. Klatzky, D. F. Keefe, P. Isenberg, J. Hurtienne et al. An emotional response to the value of visualization. *IEEE Computer Graphics and Applications*, 39(5):8–17, 2019. doi: 10.1109/MCG.2019.2923483

[51] Y. Wang, H. Zhang, H. Huang, X. Chen, Q. Yin, Z. Hou et al. Infonice: Easy creation of information graphics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2018. doi: 10.1145/3173574.3173909

[52] Z. Wang and M. Lu. Image-space collage and packing with differentiable rendering. In *Proceedings of the ACM SIGGRAPH 2025 Conference Papers*, SIGGRAPH Conference Papers '25, art. no. 172, 11 pp. ACM, New York, 2025. doi: 10.1145/3721238.3730690

[53] H. Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010. doi: 10.1198/jcgs.2009.07098

[54] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, New York, 2011.

[55] L. Wilkinson. Dot plots. *The American Statistician*, 53(3):276–281, 1999.

[56] L. Wilkinson. *The Grammar of Graphics*. Springer-Verlag, Berlin, Heidelberg, August 1999.

[57] J. Wood, P. Isenberg, T. Isenberg, J. Dykes, N. Boukhelifa, and A. Slingsby. Sketchy rendering for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2749–2758, 2012. doi: 10.1109/TVCG.2012.262

[58] H. Xia, N. Henry Riche, F. Chevalier, B. De Araujo, and D. Wigdor. Dataink: Direct and creative data-oriented drawing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2018. doi: 10.1145/3173574.3173797

[59] S. Xiao, S. Huang, Y. Lin, Y. Ye, and W. Zeng. Let the chart spark: Embedding semantic context into chart with text-to-image generative model. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):284–294, Jan. 2024. doi: 10.1109/TVCG.2023.3326913

[60] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information visualization*, 4(4):239–256, 2005. doi: 10.1057/palgrave.ivs.9500099