MetaGlyph: Automatic Generation of Metaphoric Glyph-based Visualization

Lu Ying, Xinhuan Shu, Dazhen Deng, Yuchen Yang, Tan Tang, Lingyun Yu, Yingcai Wu

Abstract— Glyph-based visualization achieves an impressive graphic design when associated with comprehensive visual metaphors, which help audiences effectively grasp the conveyed information through revealing data semantics. However, creating such metaphoric glyph-based visualization (MGV) is not an easy task, as it requires not only a deep understanding of data but also professional design skills. This paper proposes MetaGlyph, an automatic system for generating MGVs from a spreadsheet. To develop MetaGlyph, we first conduct a qualitative analysis to understand the design of current MGVs from the perspectives of metaphor embodiment and glyph design. Based on the results, we introduce a novel framework for generating MGVs by metaphoric image selection and an MGV construction. Specifically, MetaGlyph automatically selects metaphors with corresponding images from online resources based on the input data semantics. We then integrate a Monte Carlo tree search algorithm that explores the design of an MGV by associating visual elements with data dimensions given the data importance, semantic relevance, and glyph non-overlap. The system also provides editing feedback that allows users to customize the MGVs according to their design preferences. We demonstrate the use of MetaGlyph through a set of examples, one usage scenario, and validate its effectiveness through a series of expert interviews.

Index Terms—Glyph-based visualization, metaphor, machine learning, automatic visualization.

1 Introduction

Glyph-based visualization serves as an effective method for encoding multi-dimensional data [39]. However, glyph designs can also be complex due to the increasing number of data dimensions, leading to comprehension problems. Accordingly, visual metaphors are actively used to draw data-driven glyphs with representative and familiar appearances related to the data [35]. We have seen wide adoption of metaphoric glyphs in various domains, such as sports [32,44], urban application [17,36], and blockchain [68]. Studies have also shown that appropriate metaphors can help people understand glyphs quickly and accurately [13,22,35]. However, incorporating metaphors in glyph designs is not an easy task. Designers have to balance various factors, such as the expressiveness of the visual representations and the effectiveness of data mappings.

Many advanced visualization authoring tools have been developed to facilitate the creation of glyph-based visualizations [4,28,48,62,65,66]. However, it is difficult to balance automation and customization in the creation process. For example, GlyphCreator [65] and Diatoms [4] consider basic geometry or limited shapes and do not have explicit support to create metaphoric glyphs with semantic relevance. Other manual authoring tools allow users to craft the graphical elements from scratch through sketching [62] or interactions [28], which are powerful in customization. Such creation is laborious, and the quality of the final glyphs is highly dependent on the user's design experience and expertise. Using online resources may lower the cost of customization and simplify the creative process [66]. However, users need to manually select and upload the image source online without an image library.

- *L. Ying, D. Deng, Y. Yang, Y. Wu are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China. Y. Wu is also with the Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies. E-mail: {yingluu, dengdazhen, yyc_yang, ycwu}@zju.edu.cn.*
- *X. Shu is with Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. E-mail: xinhuan.shu@connect.ust.hk.*
- *T. Tang is with School of Art and Archaeology, Zhejiang University, Hangzhou, China. E-mail: tangtan@zju.edu.cn.*
- *L. Yu is with Department of Computing, Xi'an Jiaotong-Liverpool University, Suzhou, China. Email: Lingyun.Yu@xjtlu.edu.cn.*
- *Yingcai Wu is the corresponding author.*

Manuscript received 31 March 2022; revised 1 July 2022; accepted 8 August 2022. Date of publication 30 September 2022; date of current version 2 December 2022. This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2022.3209447, provided by the authors. Digital Object Identifier no. 10.1109/TVCG.2022.3209447

For better results, we aim to automatically generate metaphoric glyph-based visualization (MGV) using online sources. We attempt to ease the creation of MGV for general users who need to encode multi-variant data. However, two obstacles emerge from the process:

- *It is unclear how metaphors can be embedded into the glyph-based visualization design.* In practice, a great number of visualizations [12,63,64,68] have adopted metaphors to represent data. Existing studies [3,59] have recognized that metaphor is a good design strategy to facilitate glyph understanding. However, a systematic review of these designs to guide the creation of MGV is lacking. Detailed and practical designs for MGV have not been proposed yet.
- *It is difficult to generate an MGV without the involvement of human intelligence.* This generation process involves a series of design decisions, for instance, selecting an appropriate metaphor design and binding data with various elements in the metaphor. It requires comprehensive considerations of the whole process to streamline the production. Although increasing work has been conducted on glyph-based visualization, the automatic method for designing and generating an MGV has received less attention from the community.

To address these challenges, we propose an MGV generation framework according to a qualitative analysis established on a collection of MGV examples. We systematically reviewed 50 examples from publications and websites to explore the MGVs' designs. The analysis results provide guidance for understanding a metaphor within a glyph-based visualization for the first challenge. Informed by the results, we design and implement MetaGlyph, a proof-of-concept system that allows users to generate MGVs automatically by importing a spreadsheet. We find appropriate metaphoric images online for the data and assess how well the images match the input data in the following steps: First, the images are decomposed into a list of visual elements. Given the visual elements and different data dimensions, we then formulate the mapping problem as a tree search question and introduce a Monte Carlo tree search (MCTS) algorithm to explore the mapping space. Finally, we utilize criteria to estimate the quality of MGVs and select the best MGV based on the rewards. MetaGlyph also incorporates an interface for users to refine the MGV. The main contribution can be summarized as follows:

- We conduct a qualitative analysis to understand the design of state-of-the-art MGVs from various stages.
- We propose a novel framework by selecting metaphoric images and constructing MGVs. We also introduce a method to estimate the quality of an MGV in the framework.
- We develop MetaGlyph, a mixed-initiative system for creating MGVs automatically. We demonstrate the usage of MetaGlyph through a usage scenario and validate its usability through expert interviews.

2 Related Work

We summarize prior studies that have covered metaphor-based designs, glyph-based visualizations, and currently available authoring tools.

2.1 Metaphor-based designs

In linguistics, metaphor, analogy, and simile are three basic elements in language [43]. A metaphor is a word or a phrase that compares one thing to another to make a description more intuitive, like "*All the world is a stage*" [24]. Similes create a comparison using "like" or "as" [24]. A well-known example of a simile is "*Life is like a box of chocolates.*" The analogy is a comparison between things that have similar features [24]. An example of an analogy is "*Black is to white as on is to off*". In visualization, researchers do not explicitly differentiate these concepts, and the word "metaphor" is used to depict the case of interpreting complex information via familiar and concrete objects [35]. By allowing users to maximize their experience and knowledge, metaphors make it easier for users to understand the underlying data.

Previous studies have suggested that metaphors promote data comprehension [22, 51]. A well-known example is Chernoff faces [10], which maps one data value to one face character like the eyebrows' angle or the nose's size. Later in two quantitative experiments, Flury et al. [20] and Jacob [26] found that face glyphs outperform other visual designs like polygons and digits. Researchers have proved that, compared with glyphs unrelated to data, some metaphor-based glyphs outperform others in accuracy and efficiency through quantitative experiments, such as car glyphs [55] and clock glyphs [21]. Chau et al. [6] found that a combined design that displays glyphs and numbers together performs better in adopting a flower metaphor. Fuchs et al. [23] recently introduced a leaf glyph based on a natural metaphor and proved its effectiveness in illustrative storytelling. Dasu et al. [14] proposed an organic metaphor to interpret conditional co-occurrences and verified the effectiveness of complex tasks.

The concept of metaphor in visualization has a long history. In the 1920s, Otto Neurath and Gert Arntz [42] invented the 'Vienna Method of Pictorial Statistics', which was renamed 'ISOTYPE (International System Of TYpographic Picture Education)' in the late 1930s. They designed a lot of pictographs using semantically relevant icons.

Metaphors have been widely used to visualize different data. Metaphors like clock [19], wheel [2], and radar [61] are adopted to present radial layouts. Spatial metaphors express the relation "*proximity* $\approx$ *similarity*" [41]. Ropinski et al. [52] recommended 3D metaphoric glyphs to visualize spatial multivariate medical data for the attentive phase. Using Fermat's spirals, Lei and Zhang [33] designed the galaxy visualization to display financial time serials. Matchpad [32] used metaphoric pictograms, which are easy to learn, remember and guess. Setlur and Mackinlay [54] generated scatterplots with semantically-relevant icons to replace traditional data points automatically. Users are kept informed by the semantic information during analysis. TenniVis [44] proposed a novel glyph for individual point outcomes in a tennis match inspired by the needle gauge. SmartAdp [36] designed a novel dashboard-like glyph to represent a solution for billboard placements. Coelho and Mueller [11] created Infomages, which utilized thematic images to develop a data chart. A relevant image help users interpret the data. Recently, Compass [17] introduced a compass glyph to facilitate the in-depth understanding of urban problems.

Although these works have demonstrated the great advantages of metaphors in the visualization from different perspectives, none of them have proposed an automatic way to generate metaphoric glyphs.

2.2 Glyph-based visualization and authoring

Glyph-based visualization has become prevalent in visualization journals [3, 16, 57] and celebrated collections (e.g., Dear Data [38]). It performs well especially for multivariate data [3]. However, it is not easy to design and generate a glyph-based visualization.

Thus, many theoretical researchers have focused on how to design glyphs. Ward [58] discussed the process and issues of glyph generation, including mapping data to graphics attributes and layouts. Borgo et al. [3] drew the link between basic concepts in semiotics and glyph-based visualization and summarized existing design guidelines and techniques. Recently, Fuchs et al. [22] provided an overview of glyph types and design characteristics by reviewing experimental studies.

Furthermore, researchers have proposed many authoring tools to ease the difficulties of creating glyphs. Ribarsky et al. [49] introduced Glyphmaker, which allows non-expert users to customize data glyphs. Kim et al. [28] proposed Data-Driven Guides for Information Graphics, a system that can also create glyphs via interaction. Xia et al. [62] developed DataInk, which aims at author glyphs through freedom sketching. Ren et al. [48] presented Charticular, an authoring tool focused on layouts between glyphs. Chen et al. [9] took the first step in creating glyph-based visualization in Augmented Reality environments using mobile devices. Besides creating from scratch, DataQuilt [66] adopted real images for both inspiration and a resource of visual elements for data binding. On the other hand, while scholars have developed several tools to help users create glyphs, users still need a lot of manual operations in the system. The glyph quality highly depends on the user's design experience and expertise. Therefore, scholars have proposed some automatic systems to simplify the process recently. Ying et al. [65] aimed at circular glyphs and introduced GlyphCreator based on an example-based method. Brehmer et al. [4] developed Diatoms, a technique for inspiring glyph design through a sample-based generative process. However, automatic systems only support basic geometry or limited shapes. Unlike regular shapes, metaphors serve as effective methods to help users understand data. Thus, we opt to generate glyphs with metaphors that are not supported by existing systems.

3 Design of the MetaGlyph System

In this section, we introduce the design of the MetaGlyph system. To gain a better understanding of MGV design, we surveyed previous work and conducted a qualitative analysis. Based on the findings, we proposed the design considerations (DCs) for the MetaGlyph system.

3.1 Data Collection

To better understand how metaphors are embodied in glyph-based visualization, we examined the practice from both academic literature and online design communities.

We collected examples from top visualization conferences and journals (IEEE VIS and TVCG) using the keywords *"metaphor"* and *"glyph"* and found 221 papers. We manually examined all papers to ensure the existence of MGVs in the paper. Specifically, we checked whether both keywords *"metaphor"* and *"glyph"* appeared together to describe a metaphoric glyph. Some papers were excluded. For instance, they might mention *"metaphor"* and *"glyph"* in the related work for two different works, respectively. As a result, we collected 20 examples in the literature as the initial corpus.

To further expand the diversity, we collected more examples from creative websites (e.g.,*Behance* and *Pinterest*) with keywords such as *"information visualization" "metaphor"* and *"glyph"*. We adopted an initial filtering standard of MGV based on our current corpus. Then, we collected the chosen examples as a new part of our corpus. Finally, a total of 50 MGVs were collected as our corpus.

3.2 Qualitative Analysis

To further understand the design of an MGV, we conducted a comprehensive analysis for all examples in our corpus. In general, we analyzed all MGVs in three stages by decomposing them step by step.

S1 We find out how metaphors are embodied in an MGV and analyze the pattern of the glyph placement based on the overall design.

S2 We drill down into a single glyph and consider its layout.

S3 We aim at different visual elements that compose one glyph.

Three authors went through our corpus and independently analyzed all examples following the above stages. All disagreements were resolved through iterative discussions.

S1: At this stage, we focus on MGVs' general design to understand how designers embody the metaphor and place all glyphs in the MGVs.

- **Metaphor Type.** In our corpus, we find that designers use metaphors with respect to data properties and generally divide all examples into two conditions: semantic-related and structure-related. The former indicates the metaphor is related to the topic of data, for example, a coin representing transaction data [68], and a dashboard representing speed data [36]. In this condition, designers commonly use a metaphor within one glyph design. The latter illustrates that the

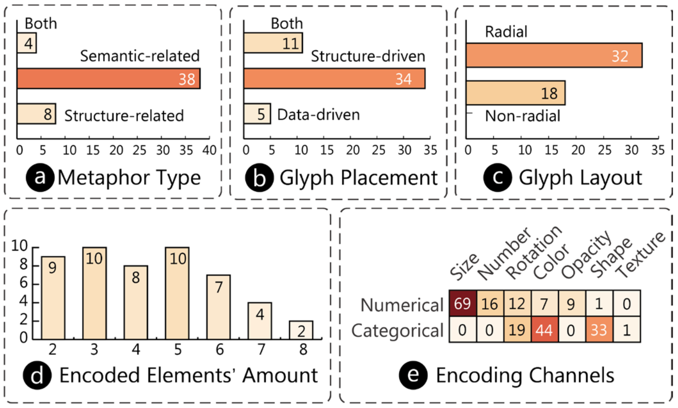

Fig. 1. The number of MGVs (a) in different metaphor types, (b) in different glyph placements, and (c) in different glyph layouts. The frequency of (d) amounts of different encoded elements and (e) different encoding channels for numerical and categorical data types in our corpus.

metaphor is related to the data structure. For example, researchers used blooming flowers in a glyph-based visualization to express hierarchical data [19]. For data about model optimization, researchers may choose the clock metaphor [67] since it is time-related. In this condition, the metaphors are mainly embodied in the visualization layout. Moreover, some designs are both semantic-related and structure-related. The concrete amounts of different metaphor types in our corpus are shown in Fig. 1(a). Finally, the definition of a valid MGV is: **using a visual design that suggests a particular association or similarity with data** [29].

- **Glyph Placement.** In our corpus, we code the glyph placement of MGVs into two groups according to Ward's theory [58]: data-driven and structure-driven. Data-driven placements correspond to glyphs that are placed based on data values. Some data values are directly used as the x- or y-value. Others need computations (e.g., projection space) to derive the position. Glyphs are usually placed in a Cartesian coordinate system in a data-driven group. The structure-driven group assumes that the data have structural characteristics. It is also a method to present metaphors. Some placements are based on a specific object, such as a tree for hierarchical data [27] and a map for geospatial data [45], or a timeline for time series data [50]. Designers adopt metaphors in structure-driven placements, such as a map with landscape and a clock for data ordered by temporal data. Other metaphors use a typical ordering relationship based on categorical information. Glyphs may be arranged evenly between left and right or located with an organization considering non-overlapping. In some cases, designers adopt two placements together for better visualization. Fig. 1(b) indicates the statistics on the corpus.

S2: We then drill down into the design of one glyph. We focus on different glyph layouts in this stage.

- **Glyph Layout.** As glyphs are frequently designed in a radial structure [65], we discuss the glyph layout in two groups: radial and non-radial. Fig. 1(c) displays the used frequency of two groups in our corpus. A radial glyph is a glyph whose elements are organized on a polar coordinate system. Each element shares the same origin. Most elements have a radial shape, like a circle and a sector. Non-radial glyphs can be placed in a Cartesian coordinate system. Notably, some glyphs present a linear structure, that is, elements in such glyphs are arranged vertically or horizontally. Others are arranged relatively freely, such as to compose a specific object like a car [55].

S3: A glyph is composed of different visual elements encoded by different data dimensions. In the last stage, we focus on the visual elements and discuss some findings of data mapping.

- **Visual Element.** Ying et al. [65] divided all visual elements in a circular glyph into four categories: chart, shape, icon, and text. Given the peculiarity of metaphor, we mainly consider two of these categories: shape-level and chart-level. The shape-level element refers to different shapes, including basic geometry (e.g., circles, polygons) and complex shapes (e.g., leaves). A chart-level element is a variant chart within a glyph, which is also a unit of shape-level elements. We integrate these shape-level elements because the unit (e.g., a pie chart) conveys more information than a single component (e.g., several sectors). In our corpus, designers adopt pie charts, donut charts, star plots, heatmap, and boxplots when designing glyphs.
- **Element Number.** The information conveyed by one glyph is limited. A glyph can better represent data in 2 to 4 dimensions [3]. According to the statistics from our corpus, the frequent amount of encoded elements of a metaphoric glyph is between 2 and 6 (Fig. 1(d)).
- **Data Mapping.** The mapping relationship of the data and elements is important for presenting the final visualization. For a given data dimension and a given element, the encoding channel is mainly determined by the data type. Fig. 1(e) presents the frequently used data types and the preferred encoding channel.

  Moreover, we have two interesting findings in our corpus. First, among all elements in an image, some elements may contain additional semantic information, such as the two circles in the car referring to wheels. Designers prefer to encode such elements with correlated data. In a car, MPG (miles per gallon) is more relevant with the wheel than the car body, and designers prefer to use the wheel size to encode MPG. Second, data with similar attributes can be encoded in the same way. For instance, Chau et al. [6] used different leaf elements in a metaphoric flower glyph to encode external and internal links of the webpage.

3.3 Design Considerations

We aim to design a system to generate MGVs automatically from a spreadsheet. We summarize three primary DCs to guide the design.

DC1 **Generate a semantically-resonant MGV.** Using metaphors, successful MGVs can promote the interpretation of the input data. Therefore, it is critical to choose an appropriate metaphoric design with respect to the data semantics. We likewise consider the rationality of the mapping between specific data dimensions and individual visual elements. The system should consider both factors and ensure the quality of the final output MGV.

DC2 **Support automatic and efficient MGV generation.** Abundant online image resources provide design inspirations and create an opportunity for the automatic generation of MGV. However, selecting one appropriate metaphor image from abundant online sources is difficult. People must remember complex data features and consider multiple data mappings. Manual data mapping is labor-intensive since users need to calculate different attributes (i.e., size and angle) for encoding. Thus, we plan to automate the process to ease data exploration through quick generation, including image selection and data mapping. On the one hand, due to the large size of online images, it takes a long time to test various combinations of images to generate an appropriate result. On the other hand, users feel tired of waiting for a result when all online images are required to be transformed and combined. Thus, we set multiple filter conditions in different steps to balance the quality of MGV and the time spent on the creation.

DC3 **Integrate a mixed-initiative workflow.** Although an automatic system provides convenience, the generated visualization may not satisfy users' expectations. Therefore, users should engage in the creation process [53]. We consider a mixed-initiative workflow that integrates the machine's and human being's efforts. Our system provides some initial results for users to choose from and change by the given spreadsheet. Then, after modifying visual elements for specific data dimensions from users, MetaGlyph improves the final output and provides alternatives based on the users' preferences. To follow this practice, our system should provide a collaborative design workflow.

4 MGV Generation Model

This section introduces our two-step model to generate MGVs, including selecting metaphoric images and constructing MGVs.

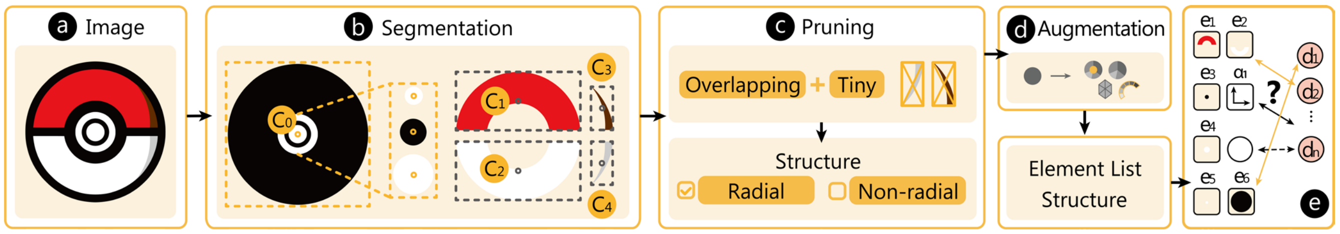

Fig. 2. The entire process of metaphoric image selection for each (a) input online image, including (b) segmentation: dividing the image into visual elements, (c) pruning: deleting redundant elements and determining the structure, and (d) augmentation: checking if some elements can transform into charts. The output is an element list with an image structure. (e) The mapping space for constructing MGVs.

4.1 Selecting Metaphoric Images

To construct valid MGVs for the given input data, we first decide on appropriate metaphors with corresponding images. This subsection elaborates on the process of metaphoric image selection, including segmentation, pruning, and augmentation. After obtaining candidates, we select one image each time for subsequent operations in Sect. 4.2.

Based on the findings from Sect. 3.2, we summarize two criteria to determine the right image:

- **C1:** It is semantically related to the data.
- **C2:** It is a vector image with a relatively simple structure.

First, we can derive **C1** directly given the definition of MGV [29]. For **C2**, vector images are easy to edit and reuse compared to raster images, which correspond to our needs for data binding. As discussed in Sect. 3.2, it is more appropriate to determine the number of the encoded visual elements within the range from two to five. Given that complex images will lead to high understanding costs, the source images of glyphs should be simple and easy to perceive. According to **C1** and **C2**, we search eligible images on the Internet.

Given a spreadsheet, we can obtain general information about the dataset. The spreadsheet name is regarded as the topic of the dataset. We choose *SVG Repo*[1] to search for eligible vector graphics following **C1**. The format of the vector graphics meets the condition, and their structure is simple for subsequent operations (**C2**). In addition, we use the Google Image Search Engine as a supplement to ensure the diversity and abundance of candidate images. We use the keyword "*icon*" and convert the resulting bitmap images into SVG using an open-source package *Portace*. As a result, we get a first-version candidate metaphoric image list in this step. Next, we filter these candidates through subsequent processing steps, as shown in Fig. 2.

Segmentation. We need to segment each image as elements from the previous output list. Given that an SVG file comprises several paths, we convert each path into an individual SVG file to obtain an element list using the SVG format. We derive n files corresponding to the n paths and calculate the center $C_i(i=1,2,...,n)$ of each retained element to determine the overall center of glyph C_0.

Pruning. Given all visual elements, we prune non-essential elements following **C2** and determine the image structure. To map data effectively, we remove tiny overlapped elements temporally, as associating the information with such elements will not improve the understandability. We heuristically found tiny elements were those occupying less than 0.5% of the area of the whole image, which reached a good compromise between the understandability and aesthetics. For instance, elements with center C_3 and C_4 in Fig. 2 are removed in this step. Next, we decide on the image structure based on preserved elements. We follow the same classification as discussed in *Glyph Layout* (Sect. 3.2) since the image is the vital material to construct a metaphoric glyph. The distinction is used for further data mapping. We use the center position $\{C_0,C_1,...,C_n\}$ to confirm the structure. As discussed in Sect. 3.2, we check whether the origin of the polar coordinate system exists by analyzing all elements' center. We first derive a possible rough origin position P_o based on the center of the glyph. If more than one element's center is close to P_o, we define this image as a radial structure and vice versa. For example, in Fig. 2, since four circular elements and the whole glyph have the same center C_0, the image is regarded as a radial structure. For non-radial images, we check if the centers of all essential elements C_i can be connected in a nearly straight line with slope k.

Augmentation. In this step, we transform original basic shapes into charts and integrate them into the glyph design. In qualitative analysis (Sect. 3.2), we identify 5 cases (star plot, donut chart, pie chart, heatmap, and boxplot) in existing research and design. Some charts are directly derived from a circular shape while others (heatmap and boxplot) need extra transformation [36,37]. Therefore, we add an extra tag for circular elements. After the above operations, we get an element list. We define the image as follows:

$$\begin{aligned} Image &= \{e_1,e_2,...,e_n,S\} \\ &= \{\{a_1,p_1\},\{a_2,p_2\},...,\{a_n,p_n\},S\} \end{aligned} \tag{1}$$

where e_i refers to different visual elements, a_i is a boolean variable indicating whether the element can be augmented, p_i describes a path in an SVG file, and S represents the image structure. For a non-radial image, we also record the slope k. For each image in the first-version candidates, we derive an element list for the following step.

4.2 Constructing Metaphoric Glyph-based Visualization

This subsection introduces the method to construct an MGV. We first formulate our problem into a mathematical form and then give a solution overview, including mapping data to different visual elements and placement as well as glyph rendering in MetaGlyph.

4.2.1 Problem Formulation

To generate an MGV, we select several data dimensions from the given spreadsheet, pick some visual elements from the element list, map the selected data attributes to the chosen elements and determine the placement. We also determine the visual encoding channel for each pair of one data attribute and an element. The position attribute for glyph placement (discussed in Sect. 3.2) is also based on data. We adopt two variables, α_1 and α_2, to present both data-driven and structure-driven glyph placements. The two variables are clear for data-driven placement to present numerical data due to a Cartesian coordinate system. We consider several specific objects (map, timeline) and order relationships according to data features for structure-driven placement. One variable α_1 is enough to present this data group. We need one categorical data dimension for the map and order relationship, and one temporal data for the timeline. As the ultimate effect of the MGV is determined by glyphs and placement, we consider them together. Moreover, we take the whole image as an extra visual element e_0 because it often encodes data.

We formulate the problem as: given data $D=\{d_1,d_2,...,d_n\}$, we select n mapping pairs P for each data dimension to solve:

$$\max_{D,L} R_{mgv}(P_{D,L}) \tag{2}$$

where R_{mgv} is the reward function for a solution, $P_{D,L}=\{p_1,p_1,...,p_n\}$ is all mapping pairs, as shown in Fig. 2(e). Each mapping pair can be represented as:

$$p = d_i \leftrightarrow \left\{ \begin{array}{l} e_j \\ \alpha_k \\ \varnothing \end{array} \right. \tag{3}$$

where $i\in[1,n]$, $j\in[0,m]$, $k\in\{1,2\}$ and $\varnothing$ is an empty set. $d_i \leftrightarrow e_j$

[1]https://www.svgrepo.com

means the data dimension is mapped with an element (yellow arrows in Fig. 2(e)). $d_i \leftrightarrow \alpha_k$ shows the data dimension is mapped with one axis of placement (the black arrow in Fig. 2(e)). Since we do not present all data dimensions in an MGV, some dimensions are not displayed. $d_i \leftrightarrow \varnothing$ represents this condition (the black dotted arrow in Fig. 2(e)).

Above all, we should guarantee the quality of the final MGV. The number of axes α_k is one or two to ensure a successful visualization. For Equation 3, the amount of valid pairs ($d_i \leftrightarrow e_j$, $d_i \leftrightarrow \alpha_k$) is an unknown variable, which means we can choose one pair, two pairs, or even all data for data-mapping. With the above considerations, the mapping space can be large for this question, even for a small dataset. Moreover, intermediate results are not worth referring to until all mapping pairs are determined. We cannot enumerate all solutions first and pick some reasonable ones as the output since a long waiting time for users violates **DC2**. We address this problem using an efficient method named MCTS [41], which is proposed to search for the best next move in a game. This algorithm can efficiently and logically explore a large space via a tree structure.

4.2.2 Monte Carlo Tree Search

We explore the mapping space by constructing a searching tree T. Each tree node is a visual element e_i or an axis α_k from the element-placement list. We start from an empty root node, as shown in Fig. 3(b). The mapping pairs (Equation 3) are represented as *Node's Height* $\leftrightarrow$ *Node*, where the *Node's Height* corresponds to a data dimension d_i, and the *Node* is one of three options in Equation 3. In MCTS, nodes close to the root are usually explored more fully. Therefore, the order of data dimensions weighs a lot for the resulting mapping method. We adopt an importance score to estimate the importance of all data dimensions and put the higher score dimension at the top of the tree. The search process repeats four stages starting from an empty node: selection, expansion, simulation, and backpropagation. Then, one mapping method is generated by a path from the root to a leaf node. After identifying one data dimension and an element, we choose an encoding channel according to the data type and element feature as discussed in Sect. 4.2.3. We also design a reward function to estimate the quality of all generated MGVs.

First, we need to order all data dimensions before building a mapping tree. We estimate the importance of one data dimension based on its relevance to the data topic. Semantic text similarity is commonly used in the natural language processing field [25], like machine translation and image caption [34]. Researchers use word embeddings to represent the original text information. Since the description of one data dimension is usually a phrase aside from a word, we choose sentence embeddings. Among existing state-of-the-art models [31, 40, 47], we choose sentence-BERT [47], which is suitable for calculating sentence distance and avoids massive computational overhead. We use cosine distance, which is widely used to measure distance in vector space. Consequently, we rank the data dimensions by calculating the distances between the description of data dimensions and the data topic.

We group numerical data dimensions with similar meanings as G_d, such as math and music scores, and apply the same encoding method to the grouped data dimensions. While the visual elements are being mapped, these groups are treated the same as other data dimensions. They cannot be considered as axes α. We compute the importance of the data dimensions by using the average distance: $dist_G = \overline{dist_d}$, where $dist_d$ is the individual importance score of integrated data dimensions. We also prioritize these groups and put them at the top of the tree ordered by their ranking as shown in Fig. 3(a).

Selection. A mapping tree is initialized with an empty root node, as shown in Fig. 3(b). In the selection stage, we aim to find the most promising element node e_i. This step starts from the root node and selects a child node with the maximum upper confidence bound for trees (UCT) [5] each time. Generally, UCT considers the balance between less-visited and high-valued nodes:

$$UCT = \frac{r_i}{n_i} + c\sqrt{\ln\frac{N_i}{n_i}} \qquad (4)$$

where r_i is the reward value, n_i is the visited times of e_i, N_i is the visited times of the parent node, c is a constant. After multiple experiments,

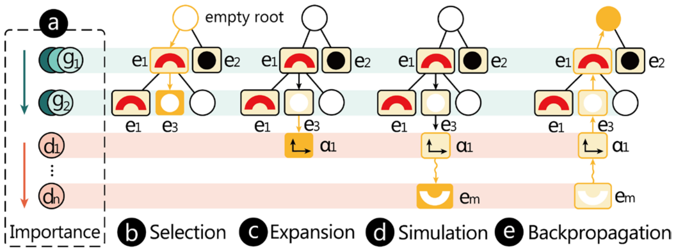

Fig. 3. (a) The data dimensions and groups are ordered by importance independently. An iteration of the Monte Carlo tree search consists of four stages, including (b) selection, (c) expansion, (d) simulation, and (e) backpropagation. The shapes with a yellow frame represent the operation of this stage. Inside the nodes are element thumbnails (e_i) or axes (α_i). Circles with no elements are empty nodes ($\varnothing$).

we used $c = 4$ since it performs better. The selection stage ends until the most urgent expandable node is reached. A node is expandable if it has unvisited (i.e., unexpanded) children.

Expansion. One child node is added to expand the tree in the expansion stage. We add a random visual element or axis node to expand as shown in Fig. 3(c) and initialize the visited time and reward as zero. Remarkably, we can add an empty node to expand because we do not present all data dimensions. For individual data dimensions, all elements in the list $L = \{e_0, e_1, ..., e_m, \alpha_1, \alpha_2, \varnothing\}$ are alternatives. However, for data groups, axes α_i are excluded.

Simulation. A simulation process starts from a new node and uses a rollout policy to produce an outcome, as shown in Fig. 3(d). The rollout policy randomly chooses a node to expand recursively (like the step in *the Expansion* stage) until the node cannot expand, which means all data dimensions have been mapped at that time. To simulate more quickly, we do not pursue a high reward in this stage. Our goal is to simulate more times to get a high reward in a limited time.

Backpropagation. In this stage, the simulation result is backed up to update the selected nodes (Fig. 3(e)). New nodes are also added to the tree T with a reward value and one visited time. Other visited nodes need to update a reward value if it is bigger and add one visited time. Then the search process returns to the selection stage or terminates when the time limits are exceeded, or the search tree is exhausted.

Finally, the path with the highest reward value is identified as the best matching method in this generating iteration.

Reward Function. We propose a reward function to estimate the quality of the final MGV via three criteria: importance (I), semantic relevance (S), and overlapping (O):

$$R_{mgv} = \begin{cases} \frac{1}{n} O_{mgv} \sum^{n}_{d,b\in\{e,\alpha\}} I(d)S(d,b), & \text{if } N_\alpha \in \{0,1\} \\ 0, & \text{otherwise} \end{cases} \qquad (5)$$

where e is one of the selected visual elements, d is one data dimension or one data group, α is one of the axes, and N_α is the number of axis nodes. We should check the number of axes (α) in the derived mapping method to guarantee a valid MGV. We removed all empty nodes at that time due to their zero importance.

- <u>*Importance Score*</u> estimates the importance of one data dimension or group. Since a cosine distance orders the importance mentioned above, some scores will be negative, which violates the reward calculation of MCTS. We normalize the distance as *Importance Score I* to ensure the feasibility of our model.
- <u>*Semantic Score*</u> estimates the semantic relevance between a visual element and one data dimension. As discussed in Sect. 3.2, designers prefer to encode visual elements with correlated data. We adopt the Transformer-MM [7] to bridge a text and image. Chefer et al. [7] used the attention layers of the model to produce relevancy maps for image and text interactions. We choose the model CLIP [46] to derive our semantic relevance score due to its abundance. It learns from 400 million text-image pairs already publicly available on the Internet. Specifically, we first convert the origin SVG into a pixel image. Given the pixel image and a textual description of one data dimension ($d \in D$), we derive a heatmap of pixels corresponding to

the description and normalize its value. Next, for a visual element e, we calculate the average relevance of the area covered by it as the final S. We adopt different scores based on data type separately for axes. When presenting temporal data or geospatial data, the S is assigned one since such placement is semantically-resonant (**DC1**). We use the entire image for other data types to derive the relevance score S because the placement is relevant to all elements.

- *Overlapping Score* estimates the overlapping degree of the final MGV. We calculate the overlapping areas in the MGV rendered in Sect. 4.2.3 based on the bounding box of each glyph:

$$O_{mgv} = \begin{cases} 1 & \text{if } P_{overlap} \leq 30\% \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $P_{overlap}$ is all elements' average overlapping percentage.

4.2.3 Rendering

Given a visual element, the glyph placement, and data dimension, we need to make the encoding channel clear for rendering.

Encoding Channel for Visual Elements. As mentioned above, all data dimensions are divided into d (one data dimension) and g (a data group). Moreover, we divided all visual elements into e_a and $e_{!a}$ for different value of the augmentation tag a_i retained in Sect. 4.1. e_a corresponds to $a = 1$ and vice versa. We consider three conditions separately for different visual elements and data dimensions:

- $\underline{d \leftrightarrow \{e_a, e_{!a}\}}$. Since the data input is only one dimension, we treated $\{e_a, e_{!a}\}$ the same. Although element e_a can be augmented, we preserve its original shape for a better representation. We use a heuristic method based on examples in our corpus (Fig. 1(e)). We determine the encoding channel by analyzing the shape of elements and the data type. Given a data dimension, we select the most frequent encoding channel based on its type. When two or more data dimensions share the same element, we order them based on the reward function (Equation 5). Dimensions with a higher reward can occupy a more frequent encoding channel. When using size channel, we need an additional decision based on the image structure (Equation 1). Height, length, and area are the options for encoding.
- $\underline{g \leftrightarrow e_a}$. For the pair of data groups and elements that can augment, we transform the element into different charts to present all dimensions in the group. We adopt four charts, namely, pie charts, donut charts, star plots, and heatmaps as shown in Fig. 2(d). For numerical data, we consider star plots. Specifically, pie charts, donut charts, and heatmaps are alternatives for proportional data.
- $\underline{g \leftrightarrow e_{!a}}$. We use a typical design in our corpus with visual elements that cannot augment. We choose three encoding channels (rotation, color, and size) of one visual element to represent a data group. Rotations and colors are both used to distinguish different data dimensions in the group. The size encodes the numerical data. An example is displayed in Fig. 6(b). The leaves are rotated to illustrate the forest area in different years. Color is also utilized to distinguish the value year, as shown in the legend (Fig. 6(b1)). The size of the leaves represents the percentage value.

Notably, we check if the removed elements e_r in *Pruning* (Fig. 2(c)) need to be drawn after confirming encoding channels. We call the preserved element that overlaps with the removed one as e_p, and the encoding channel for e_p as c_p. If e_p encodes data and c_p is size, we scale the e_r the same as e_p and add it into the glyph. If e_p is transformed into a chart or does not encode data, we delete the element. For other conditions, we keep it in its original shape for drawing.

Glyph Placement. Owing to the limit in the simulation stage of MCTS, we only need to consider the independent data dimensions d and the axes α. For temporal data, we use a timeline following **DC1**. For geo-spatial data, we utilize a map (Fig. 6(b) and (f)). For other numerical data, if the number of axes is one, we adopt a horizontal axis to place glyphs. A Cartesian coordinate system is suitable for two axes, as shown in Fig. 6(a) and (e). Concerning categorical data, we place glyphs in a specific order as shown in Fig. 6(c) and (g).

Ultimately, we can obtain one semantically-resonant MGV with the highest reward for each image candidate. We pick the greatest as the output based on all candidates' reward values, and others are ranked as alternatives for users to choose from.

5 METAGLYPH

This section introduces the workflow of MetaGlyph that generates MGVs given a spreadsheet input and the interface design.

5.1 Workflow

We design the workflow of MetaGlyph based on the design considerations proposed in Sect. 3.3. Following **DC3**, MetaGlyph integrates the automatic model and user interactions into the authoring process. First, users need to import a spreadsheet of data as the input. An initial MGV is generated based on the automatic model. The quality of MGV is ensured by a reward function that considers different dimensions, including data importance and semantic matching (**DC2**) as discussed in Sect. 4. After that, users can modify the data mappings and the encodings of the initial visualization result. The automatic model will re-calculate and create a new MGV based on users' input. To ensure the efficiency of the computing following **DC2**, we limit the search number of images at one time. Next, users can smoothly switch between different visualizations and refine the satisfactory one as the final output. Given that both the users and the model contribute to the design of the MGV, MetaGlyph can derive more novel and creative designs compared with a solo-authoring workflow.

5.2 User Interface

MetaGlyph (Fig. 4) consists of four views: Menu view, MGV view, Gallery view, and Edit view. The Menu view shows the creation procedures of our system, that is, first upload, then preprocess, and ultimately edit the visualization. Users can upload a spreadsheet, and MetaGlyph will process the data. After processing, we can derive several MGVs with high rewards within a suitable waiting time (**DC2**) using the MGV generation model in Sect. 4. Moving to the edit step, the MGV with the highest reward is presented in the MGV view (Fig. 4(b)), and the other alternatives are shown in the Gallery view (Fig. 4(d)). Moreover, all segmented visual elements of the image are displayed on the left of the MGV view. Here, *Element 0* represents the entire metaphor. Users can select other MGVs in the Gallery view following their preferences. When hovering over the alternatives, we display the original metaphor (Fig. 4(d1)). The Edit view will show all mappings and allow users to modify them (Fig. 4(c)). Each small panel illustrates detailed information about one data dimension or one data group, including the title of the data column, data type, element mapped, and corresponding encoding channel. The ordering of different data dimensions depends on the importance score discussed in Sect. 4.2. We use a small icon to illustrate the data type after its title. For data that are not represented in the current MGV, the mapped element is shown as *None*. Users can alter the data mappings by changing the visual elements in corresponding panels in the Edit view. After clicking the *Update* button in the MGV view, the generation model will re-calculate to obtain a satisfactory result following **DC3**. Finally, by clicking the *Export* button, users can export the MGV shown in the center as an SVG file.

5.3 Usage Scenario

We present one usage scenario in detail in this section. Adam is a junior visualization researcher. His task was to analyze the ingredients and nutrients of different hamburgers in McDonald [1] for customers to make a better choice. He uploaded the processed tabular data in the MetaGlyph system and clicked the "generate" button. After a few seconds, the system generated a list of MGVs, as shown in Fig. 4. The MGV in the center represents three data dimensions: burger lettuce, sugars, and bacon (Fig. 4(e)). After browsing and absorbing the concrete mapping relations displayed in the *Edit* view (Fig. 4(c)), Adam quickly understood the encodings of the glyph. Bread is a kind of carbohydrate. Therefore, the size of the upper bread is appropriate to encode the data attribute *Sugars*. He was also satisfied with the method to show *Burger Lettuce* and *Bacon* using green and red lines (Fig. 4(e)) due to the intuitiveness. He clicked three titles of corresponding data dimensions in the *Edit* view (Fig. 4(c1)) to add three limits for the next update. Next, Adam obtained a new result and continued exploration for other alternatives in the Gallery view (Fig. 4(d)).

Reanalyzing the dataset, he considered grouping some data columns, including iron, calcium, vitamin A, and vitamin C, since these are often considered as nutrients. He returned to the *Preprocess* step (Fig. 4(a))

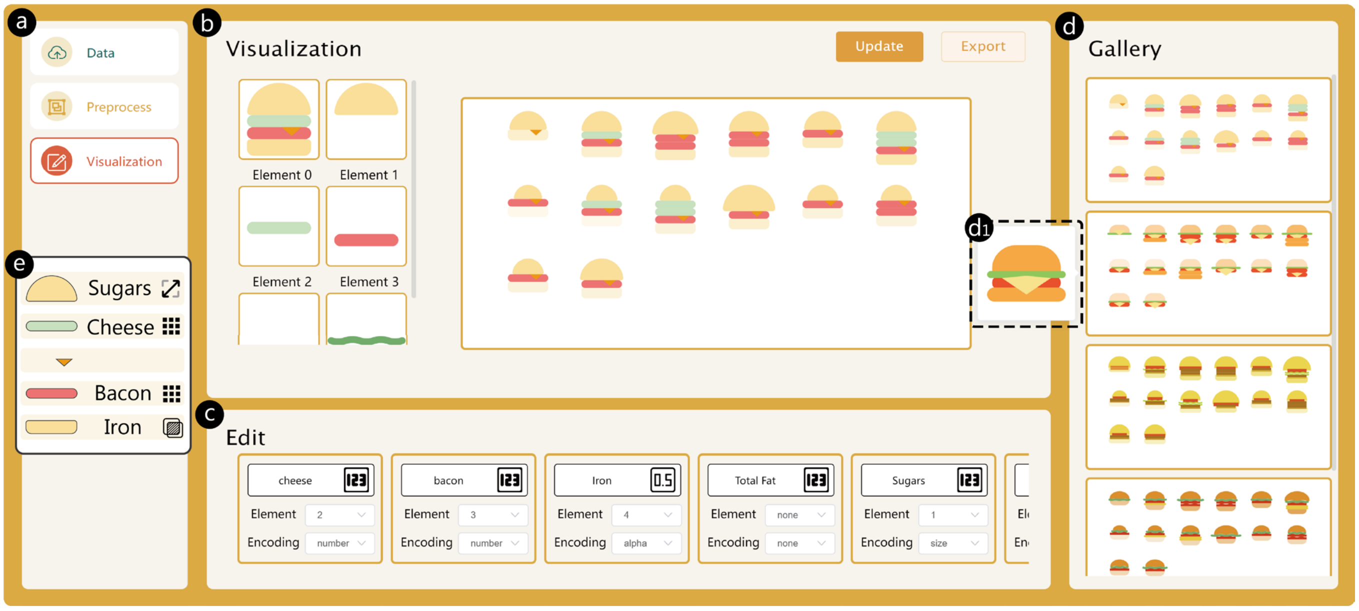

Fig. 4. The interface of MetaGlyph is consisted of : (a) a Menu view for different steps; (b) an MGV view representing visual elements and corresponding MGV, (c) an Edit view, in which the data mappings can be modified based on user's preferences, and (d) a Gallery view with other MGV options. (e) concrete data mappings for the MGV view in detail.

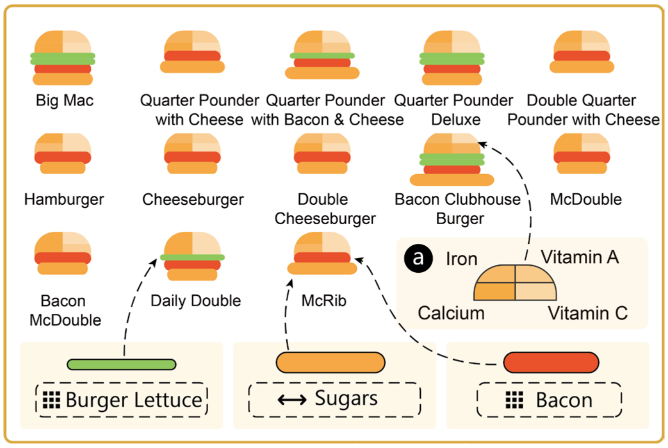

Fig. 5. An example generated with the burger dataset. (a) The legend for the upper bread generated by MetaGlyph. The boxes with yellow backgrounds illustrate the data mappings and encoding channels. Dashed arrows are used to associate visual elements with the MGV.

and added one data group. This time he found an interesting MGV, which adopted a heatmap-like design on the upper bread to represent the new data group. Adam preserved this design and adjusted other mappings following his preferences. After clicking the legend button (Fig. 4(b1)), he understood the encodings for the upper bread (Fig. 5(a)). Using such a design, he could grasp different levels of four components at first sight. Other encodings are displayed in Fig. 5. The numbers of red lines and green lines illustrate the amount of bacon and lettuce layers, respectively. The length of the bread below represents the *Sugars*. Satisfied with this result, he exported this MGV.

Following a similar procedure, we demonstrate the expressiveness of MetaGlyph by creating more MGVs with various datasets (Fig. 6). These examples are hard to create with existing tools. For example, automatic tools (e.g., GlyphCreator [65], Diatoms [4] do not provide explicit support for metaphoric glyph design. Authoring tools (e.g., DDG [28], DataQuilt [66]) provide greater flexibility and allow users to achieve similar results. However, the authoring process is tedious and time-consuming (e.g., requiring users to find or draw suitable metaphor images and encode data), and the quality of MGVs is highly dependent on the user. Moreover, they do not support embedding charts in the glyphs (e.g., star plots in Fig. 6(a) and pie charts in Fig. 6(f)).

6 Expert Interview

To evaluate the MetaGlyph system, we performed a series of interviews with three expert users in different fields. The first expert (E1) is a data journalist who has worked for a digital-news firm for more than three years. The second expert (E2) majored in visual communication and has more than ten years of design experience. The third expert (E3) is a senior researcher who has studied human-computer interaction and data visualization for more than eight years.

Procedure. We conducted the interview via an online meeting system. Each interview began with a 10-minute introduction of MGV following a 10-minute demo of the MetaGlyph system. Next, the experts were encouraged to use MetaGlyph online on their own. After thorough trials of our system's features, we asked experts to generate MGVs using the *Burger* and *Pokemon* datasets introduced in Sect. 5.3. We recorded their creation process in a think-aloud approach. Our interviews were seeded with a set of questions that probed into the effectiveness of MetaGlyph and the quality of the generated MGVs. The one-hour screen capture videos and audios were recorded from these interviews for further analysis.

Feedback. In general, all experts expressed their compliments of MetaGlyph and agreed on its promising usage. We summarized the interview results from three perspectives:

Workflow. Three interviewees appreciated the overall design of the workflow. All of them expressed the effectiveness of integrating metaphors into the design. They believed that the automatic system is an efficient way to speed up the generation. During their previous creation, they preferred design software (e.g., Adobe Illustrator) or programming (E1, E3) after confirming the design ideas. E1 said, *"Some of my colleagues write codes, but it requires long-time learning."* E3 commented, *"When using design software, I need to do time-consuming batch work."* She also expressed the difficulty of dealing with some visual elements, such as adjustments of arc angles. With MetaGlyph, she thought the creation process would be more straightforward.

System. All experts were satisfied with the design of MetaGlyph. They appreciated the aesthetics of the interface and expressed the ease of learning cost when using MetaGlyph. E1 noted, *"The interaction is intuitive, and I can use it proficiently after a simple demonstration."* For the designs of different views, E2 observed the *Gallery view* and commented, *"Due to the difference in users' visual experience, their*

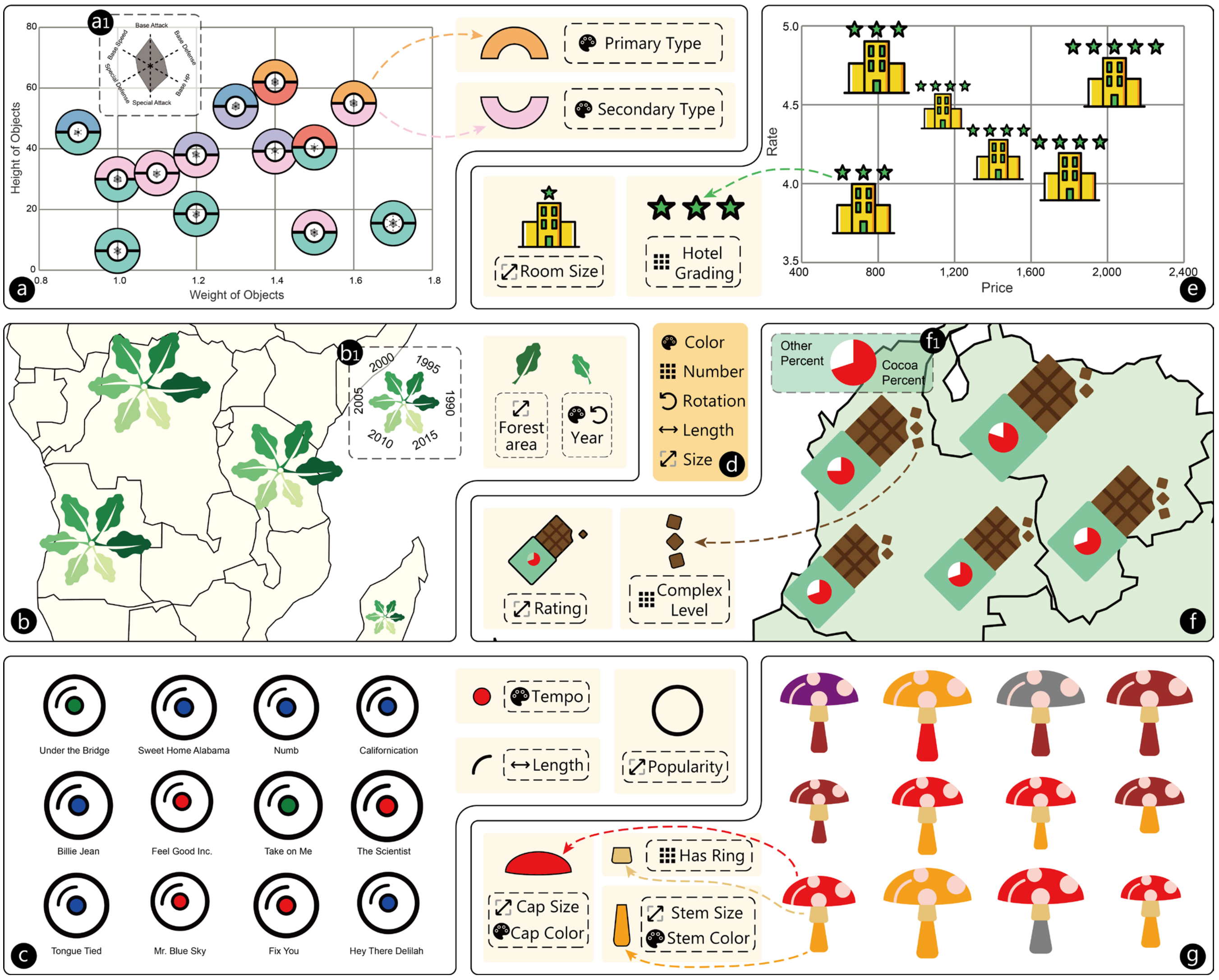

Fig. 6. Example MGVs generated by MetaGlyph: (a) several attributes of different pokemons; (b) forest area changes in different countries from 1995 to 2020; (c) display of various CDs; (e) multiple hotels' information arranged by price (x) and rate (y); (f) chocolates in different ratings placed on a map; (g) disparate mushrooms. The legends in (a1)(b1)(f1) are also generated by MetaGlyph. The center boxes with yellow backgrounds illustrate the data mappings for each glyph. Encoding channels are represented in icons and (d) shows the annotation. Dashed arrows are used to associate visual elements with the resulting visualizations.

understanding ability is disparate. It is considerate to provide various alternatives for them." E3 liked the *Edit* view, *"I prefer to try different results on my own. Although the system's results are nice, I can derive a different version by changing some attributes in the edit view."*

Visualization. In terms of the quality of generated MGVs, all interviewees agreed that the outputs were thoughtful due to the semantic relevance and metaphor embodied. E2 underscored the match pattern for data and visual elements in MetaGlyph, which he also focused on during his design. He said, *"Important data should be mapped with prominent elements."* He also agreed with the limitation of the number of encoded data dimensions, *"It is essential to make trade-offs to data, such as preserving three data dimensions. Too much information will lead to comprehension problems."* E1 and E3 both commented that MGVs could serve as the basis for creative work in their early design. E3 said, *"When designing, the most struggling part is the beginning due to the large design space. The outputs not only give me a direction but also broaden my scope via some unexpected designs."* She agreed to use online sources since such images meet most people's cognition. Therefore, the MGVs can be better understood by audiences.

Suggestions. We also received suggestions from different aspects. As an editor, E1 focused on the stories behind the data and suggested that corresponding conclusions can be generated automatically together with MGVs. E2's advice fell into the improvement of visual comprehension. He was confused about some specific values in the output MGV such as the size of the burger layer represents. E2 suggested adding more annotations and textual descriptions into the MGVs. He noted that color could also be considered in understandability. E3 gave some advice from the perspective of an interaction designer, such as the button position and the highlight effect. She also suggested exporting more files (e.g., legend, files of different elements, mapping space between data and elements) together with the MGV, thereby allowing further refinement using professional design tools.

7 Discussion

This section reflects the implications and limitations of MetaGlyph.

7.1 Implications

In this paper, we contribute an automatic approach to generating glyph-based visualization associated with comprehensive visual metaphors to help people understand data in an intuitive manner [22]. We propose the semantic-based generation framework by selecting resonant metaphorical images and mapping visual elements with data dimensions considering data importance and semantic relevance. Users can customize the resulting MGVs based on their requirements. We derive a set of implications in the design process of our system.

Automating glyph designs in association with semantic-resonant metaphors. Metaphors are widely used in visualization, which associates data semantics with figurative motifs. MetaGlyph aims to automatically generate glyphs whose visual elements are associated with semantically related metaphors. To this end, we first search for appropriate metaphors with images considering the overall topic of the input data. Furthermore, we decompose visual elements of the metaphor and match them with data attributes with respect to the underlying semantic relations by adopting a prediction model. For example, the system searches burger images for the dataset of burger ingredients and maps the lettuce and bacon to the green and red elements, respectively (Fig. 5). In this way, the resulting MGVs are associated with the underlying data semantics cohesively.

Balancing design expressiveness and perceptual effectiveness. The increase in data dimensionality leads to a complicated metaphoric glyph design that is difficult to understand [3]. Our qualitative analysis (Sect. 3.2) and expert interview (Sect. 6) both indicate the constraint of the number of encoded data dimensions. Besides, our expert E2 suggests emphasizing important data with prominent visual elements, *"I will not utilize all elements for presenting data. It would be overwhelming."* Some visual elements in the metaphor are better used to encode data attributes, while others are more suitable for decoration purposes. Accordingly, the encoded data dimensions and visual elements should be balanced with perceptual constraints. Our current approach has filtered MGVs that utilize overmuch elements in the mapping space (Sect. 4.2.2). Further experiments are encouraged to explore the appropriate amount of conveyed information in an MGV that informs better automatic generation criteria with perceptual effectiveness.

Supporting a human-machine teaming approach. We situate MetaGlyph as a human-machine teaming system that integrates the machine with users' feedback. Incorporating metaphors into designs is generally considered a highly creative task, requiring implicit knowledge to determine the right imagery and modify it to fit scenarios. Faced with the large search space of metaphors and data mappings, generating MGVs is time-consuming and laborious. Automating labor-intensive parts with machines is highly required. Moreover, MetaGlyph provides more than tool-level assistance. It explores the search space for users and provides design alternatives. Both novices and experts can benefit from the results as initial designs. Users are likewise supported to modify the results based on their preferences. With the machine's assistance, users can focus on the data mappings without adjusting the elements' attributes manually. Moreover, MGVs can be updated multiple times according to users' requirements and the machine's re-calculation. By assigning the labor-intensive part to the machine and providing the connectors of subjective decisions to human beings, MetaGlyph provides a successful human-machine teaming example and inspires the design of visualization tools in the future.

Providing design inspirations. All experts appreciated the unexpected designs provided by MetaGlyph. Given the large space of metaphor selection and visual mapping, it would be impossible for designers to try all alternatives. In addition, as mentioned by E3, designers are likely to start with familiar designs based on their usual practice. Instead, the machine enumerates all possibilities, which may result serendipitously in promising designs. For example, E3 praised the star plot inside the Pokeball (Fig. 6(a)) and the flower-like leaves (Fig. 6(b)). *"I never thought of using such designs for these datasets. The visualization created by MetaGlyph shows diverse design possibilities,"* E3 said. Designers can further refine the design using MetaGlyph or export the whole design assets for fine-tuning with professional tools.

7.2 Failure Cases and Limitations

Failure Cases. We observed several wrong or bad cases during the development of MetaGlyph. First, one critical source of some failure cases is the diversity of images the system searches from online sources as metaphors. For example, some images may have elements in multiple layers, such as Fig. 7(a) shows a car with a background. When scaling the car to encode data, the system cannot guarantee that the car can always be within the background, resulting in an incomprehensible MGV. Second, as we currently adopt some heuristic methods to decide

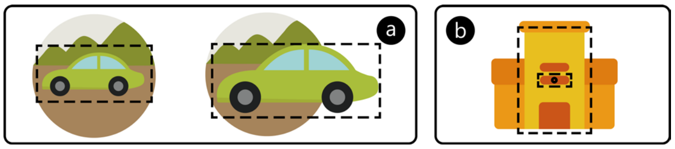

Fig. 7. (a) A car with a driving background. (b) A hotel.

the layout (Sect. 4.2), it cannot always be true. For example, Fig. 7(b) is wrongly identified as a radial glyph as it has two elements whose centers are close to the SVG's center. Future work can use new techniques in computer vision and deep learning to optimize our framework.

Limitations. First, the degree of customization for MetaGlyph is limited. Specially, we follow the default style of the metaphoric source image, and use fixed parameters when calculating (e.g., set the boundary 0.5% in Pruning in Sect. 4.1). Future work can allow users to customize their MGVs, including but not limited to styles [8] and fields of parameters. Second, MetaGlyph only supports tabular data with a single sheet, while complicated data structures, such as graph-related data, are not supported. Because graph-related data can be represented as multiple-sheet data (e.g., adjacency matrices), we plan to extend MetaGlyph with more data processing operators to integrate multiple data sheets in the encodings of a single glyph. In this way, we can improve the diversity of the resulting visualization for MetaGlyph. Third, we mainly consider the importance of data dimensions and their semantic relevance with visual elements in data mapping. Other factors (such as SVG elements' sizes and shapes, mapping different elements with different channels, and users' priori knowledge) also play roles in the mapping. Future research can further explore how to consider all these factors to achieve a better MGV design.

8 Conclusion and Future Work

This paper presents MetaGlyph, an automatic system for generating MGVs. We first conduct qualitative analysis to understand the design patterns of MGVs and then develop an automatic generation framework. We automate the entire process by searching for appropriate metaphors, processing images, and determining multiple data mappings along with visual encodings. Given the ample mapping space between visual elements and data dimensions, we adopt an MCTS algorithm to achieve an efficient and effective search considering data importance, semantic relevance, and glyph separation. By uploading tabular data, our system will generate several MGVs within a few seconds. Moreover, as a human-machine teaming system, MetaGlyph enables users to modify the MGVs following their preferences. We evaluate MetaGlyph through two usage scenarios and a gallery of examples and demonstrate its effectiveness via a series of expert interviews. MetaGlyph shows its potential in various situations, such as education [30] and journalism. Teachers can use MetaGlyph to create metaphoric designs to help teach data literacy or visual encodings, while journalists can use it to create figurative and expressive visualization easily and quickly.

Looking into the future, we hope our work can inspire further research from the perspectives of design expressiveness and perceptual effectiveness with regard to different visual representations (e.g., storyline [56], data comics, and visual analytics systems [60]). First, we plan and also encourage researchers to conduct empirical studies to investigate how different users understand metaphoric visualizations. Cognitive experiments can likewise be designed to derive metrics for the effectiveness of metaphor selection so that more theories about computational metaphors can be constructed. In turn, the theories can be utilized to inspire the development of future visualization tools. Second, research on automatic visualization generation [18] can benefit from large-scale visualization datasets (e.g., VisImages [15]). Future work can focus on constructing glyph datasets to facilitate the use of end-to-end deep learning models for an efficient generation.

Acknowledgments

The work was supported by NSFC (62072400) and the Collaborative Innovation Center of Artificial Intelligence by MOE and Zhejiang Provincial Government (ZJU).

References

[1] McDonald's Menu - Comparative Nutrition Values. https://www.kaggle.com/code/kathakaliseth/mcdonald-s-menu-comparative-nutrition-values. Accessed: 2022-3-21.

[2] B. Alsallakh, W. Aigner, S. Miksch, and M. E. Groller. Reinventing the Contingency Wheel: Scalable Visual Analytics of Large Categorical Data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2849–2858, 2012.

[3] R. Borgo, J. Kehrer, D. H. S. Chung, E. Maguire, R. Laramee, H. Hauser, M. Ward, and M. Chen. Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications. In *Proceedings of the Eurographics*, pp. 39–63, 2013.

[4] M. Brehmer, R. Kosara, and C. Hull. Generative design inspiration for glyphs with diatoms. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):389–399, 2022.

[5] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[6] M. Chau. Visualizing web search results using glyphs: Design and evaluation of a flower metaphor. *ACM Transactions on Management Information Systems*, 2(1):1–27, 2011.

[7] H. Chefer, S. Gur, and L. Wolf. Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers. *arXiv:2103.15679 [cs]*, 2021.

[8] S.-Y. Chen, J.-Q. Zhang, Y.-Y. Zhao, P. L. Rosin, Y.-K. Lai, and L. Gao. A review of image and video colorization: From analogies to deep learning. *Visual Informatics*, 2022.

[9] Z. Chen, Y. Su, Y. Wang, Q. Wang, H. Qu, and Y. Wu. MARVisT: Authoring Glyph-Based Visualization in Mobile Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics*, 26(8):2645–2658, 2020.

[10] H. Chernoff. The Use of Faces to Represent Points in k-Dimensional Space Graphically. *Journal of the American Statistical Association*, 68(342):361–368, 1973.

[11] D. Coelho and K. Mueller. Infomages: Embedding Data into Thematic Images. *Computer Graphics Forum*, 39(3):593–606, 2020.

[12] Y. Cui, C. Li, C. Chen, Y. Liang, Y. Hu, and C. Wang. VineMap: A metaphor visualization method for public opinion hierarchy from text data. *Journal of Visualization*, 24(5), 2021.

[13] J. M. Cunha, E. Polisciuc, P. Martins, and P. Machado. The Many-Faced Plot: Strategy for Automatic Glyph Generation. In *Proceedings of the International Conference Information Visualisation*, pp. 71–77, 2018.

[14] K. Dasu, T. Fujiwara, and K.-L. Ma. An Organic Visual Metaphor for Public Understanding of Conditional Co-occurrences. In *Proceedings of the IEEE Scientific Visualization Conference*, pp. 1–5, 2018.

[15] D. Deng, Y. Wu, X. Shu, M. Xu, J. Wu, S. Fu, and Y. Wu. VisImages: A Large-scale, High-quality Image Corpus in Visualization Publications. *CoRR*, 2020.

[16] Z. Deng, D. Weng, and Y. Wu. You are experienced: Interactive tour planning with crowdsourcing tour data from web. *Journal of Visualization*, 26(to appear), 2023.

[17] Z. Deng, D. Weng, X. Xie, J. Bao, Y. Zheng, M. Xu, W. Chen, and Y. Wu. Compass: Towards Better Causal Analysis of Urban Time Series. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1051–1061, 2022.

[18] Deng, Dazhen and Wu, Aoyu and Qu, Huamin and Wu, Yingcai. Dashbot: Insight-driven dashboard generation based on deep reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[19] M. El-Assady, F. Sperrle, O. Deussen, D. Keim, and C. Collins. Visual Analytics for Topic Model Optimization based on User-Steerable Speculative Execution. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):374–384, 2019.

[20] B. Flury and H. Riedwyl. Graphical representation of multivariate data by means of asymmetrical faces. *Journal of the American Statistical Association*, 76(376):757–765, 1981.

[21] J. Fuchs, F. Fischer, F. Mansmann, E. Bertini, and P. Isenberg. Evaluation of alternative glyph designs for time series data in a small multiple setting. In *Proceedings of the Conference on Human Factors in Computing Systems*, pp. 3237–3246, 2013.

[22] J. Fuchs, P. Isenberg, A. Bezerianos, and D. Keim. A Systematic Review of Experimental Studies on Data Glyphs. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1863–1879, 2017.

[23] J. Fuchs, D. Jäckle, N. Weiler, and T. Schreck. Leaf Glyphs: Story Telling and Data Analysis Using Environmental Data Glyph Metaphors. In *Computer Vision, Imaging and Computer Graphics Theory and Applications*, vol. 598, pp. 123–143. 2016.

[24] S. W. Gilbert. An evaluation of the use of analogy, simile, and metaphor in science texts. *Journal of Research in Science Teaching*, 26(4):315–327, 1989.

[25] D. Hu. An introductory survey on attention mechanisms in nlp problems. In *Proceedings of SAI Intelligent Systems Conference*, pp. 432–448, 2019.

[26] R. J. Jacob. Facial representation of multivariate data. In *Graphical representation of multivariate data*, pp. 143–168. 1978.

[27] S. Jang, N. Elmqvist, and K. Ramani. MotionFlow: Visual Abstraction and Aggregation of Sequential Patterns in Human Motion Tracking Data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):21–30, 2016.

[28] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister. Data-Driven Guides: Supporting Expressive Design for Information Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23:491–500, 2017.

[29] N. Kogan, K. Connor, A. Gross, and D. Fava. Understanding visual metaphor: Developmental and individual differences. *Monographs of the Society for Research in Child Development*, pp. 1–78, 1980.

[30] X. Kui, N. Liu, Q. Liu, J. Liu, X. Zeng, and C. Zhang. A survey of visual analytics techniques for online education. *Visual Informatics*, 2022.

[31] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning*, pp. 1188–1196, 2014.

[32] P. A. Legg, D. H. S. Chung, M. L. Parry, M. W. Jones, R. Long, I. W. Griffiths, and M. Chen. MatchPad: Interactive Glyph-Based Visualization for Real-Time Sports Performance Analysis. *Computer Graphics Forum*, 31(3pt4):1255–1264, 2012.

[33] S. T. Lei and K. Zhang. Visual signatures for financial time series. In *Proceedings of the Visual Information Communication - International Symposium*, pp. 1–10, 2011.

[34] J. Li, N. Xu, W. Nie, and S. Zhang. Image Captioning with multi-level similarity-guided semantic matching. *Visual Informatics*, 5(4), 2021.

[35] Y. Li, D. Li, and K. Zhang. Metaphoric transfer effect in information visualization using glyphs. In *Proceedings of the International Symposium on Visual Information Communication and Interaction*, pp. 121–130, 2015.

[36] D. Liu, D. Weng, Y. Li, J. Bao, Y. Zheng, H. Qu, and Y. Wu. SmartAdP: Visual Analytics of Large-scale Taxi Trajectories for Selecting Billboard Locations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):1–10, 2017.

[37] M. Liu, S. Liu, X. Zhu, Q. Liao, F. Wei, and S. Pan. An Uncertainty-Aware Approach for Exploratory Microblog Retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):250–259, 2016.

[38] G. Lupi and S. Posavec. *Dear data*. Chronicle books, 2016.

[39] H. Mansoor, W. Gerych, A. Alajaji, L. Buquicchio, K. Chandrasekaran, E. Agu, and E. Rundensteiner. ARGUS: Interactive Visual Analysis of Disruptions in Smartphone-detected Bio-Behavioral Rhythms. *Visual Informatics*, 2021.

[40] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations*, 2013.

[41] D. R. Montello, S. I. Fabrikant, M. Ruocco, and R. S. Middleton. Testing the First Law of Cognitive Geography on Point-Display Spatializations. In *Spatial Information Theory. Foundations of Geographic Information Science*, vol. 2825, pp. 316–331. 2003.

[42] O. Neurath. From vienna method to isotype. In *Empiricism and Sociology*, pp. 214–248. Springer, 1973.

[43] P. Norvig. G. lakoff, m. johnson, metaphors we live by. *Artificial Intelligence*, 27(3):357–361, 1985.

[44] T. Polk, J. Yang, Y. Hu, and Y. Zhao. TenniVis: Visualization for Tennis Match Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2339–2348, 2014.

[45] R. A. Proma, M. Sumpter, H. Lugo, E. Friedman, K. T. Huq, and P. Rosen. CleanAirNowKC: Building Community Power by Improving Data Accessibility. In *Proceedings of IEEE Workshop on Visualization for Social Good*, pp. 1–5, 2021.

[46] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision.

In *Proceedings of the International Conference on Machine Learning*, pp. 8748–8763, 2021.

[47] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3980–3990, 2019.

[48] 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.

[49] W. Ribarsky, E. Z. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: Creating Customized Visualizations fo Complex Data. *Computer*, 27(7):57–64, 1994.

[50] A. Rind, T. Lammarsch, W. Aigner, B. Alsallakh, and S. Miksch. TimeBench: A Data Model and Software Library for Visual Analytics of Time-Oriented Data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2247–2256, 2013.

[51] J. S. Risch. On the role of metaphor in information visualization. *arXiv:0809.0884 [cs]*, 2008.

[52] T. Ropinski, S. Oeltze, and B. Preim. Survey of glyph-based visualization techniques for spatial multivariate medical data. *Computers & Graphics*, 35(2):392–401, 2011.

[53] S. Rubab, J. Tang, and Y. Wu. Examining interaction techniques in data visualization authoring tools from the perspective of goals and human cognition: A survey. *Journal of Visualization*, 24(2), 2021.

[54] V. Setlur and J. D. Mackinlay. Automatic generation of semantic icon encodings for visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 541–550, 2014.

[55] H. Surtola. The effect of data-relatedness in interactive glyphs. In *Proceedings of the International Conference on Information Visualisation*, pp. 869–876, 2005.

[56] T. Tang, R. Li, X. Wu, S. Liu, J. Knittel, S. Koch, T. Ertl, L. Yu, P. Ren, and Y. Wu. PlotThread: Creating Expressive Storyline Visualizations using Reinforcement Learning. *arXiv:2009.00249 [cs]*, 2020.

[57] L. Wang, G. Sun, Y. Wang, J. Ma, X. Zhao, and R. Liang. AFExplorer: Visual analysis and interactive selection of audio features. *Visual Informatics*, 6(1), 2022.

[58] M. O. Ward. Multivariate Data Glyphs: Principles and Practice. In *Handbook of Data Visualization*, pp. 179–198. 2008.

[59] J. Wenskovitch, I. Crandell, N. Ramakrishnan, L. House, S. Leman, and C. North. Towards a Systematic Combination of Dimension Reduction and Clustering in Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):131–141, 2018.

[60] A. Wu, D. Deng, F. Cheng, Y. Wu, S. Liu, and H. Qu. In Defence of Visual Analytics Systems: Replies to Critics. *arXiv preprint arXiv:2201.09772*, 2022.

[61] W. Wu, Y. Zheng, K. Chen, X. Wang, and N. Cao. A Visual Analytics Approach for Equipment Condition Monitoring in Smart Factories of Process Industry. In *Proceedings of the IEEE Pacific Visualization Symposium*, pp. 140–149, 2018.

[62] H. Xia, N. H. Riche, F. Chevalier, B. R. D. Araújo, and D. Wigdor. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, p. 223, 2018.

[63] X. Xie, J. Wang, H. Liang, D. Deng, S. Cheng, H. Zhang, W. Chen, and Y. Wu. PassVizor: Toward Better Understanding of the Dynamics of Soccer Passes. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1322–1331, 2021.

[64] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, and H. Qu. CloudDet: Interactive Visual Analysis of Anomalous Performances in Cloud Computing Systems. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019.

[65] L. Ying, T. Tang, Y. Luo, L. Shen, X. Xie, L. Yu, and Y. Wu. GlyphCreator: Towards example-based automatic generation of circular glyphs. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):400–410, 2022.

[66] J. E. Zhang, N. Sultanum, A. Bezerianos, and F. Chevalier. DataQuilt: Extracting visual elements from images to craft pictorial visualizations. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 1–13, 2020.

[67] J. Zhao, N. Cao, Z. Wen, Y. Song, Y.-R. Lin, and C. Collins. #FluxFlow: Visual Analysis of Anomalous Information Spreading on Social Media. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1773–1782, 2014.

[68] Z. Zhong, S. Wei, Y. Xu, Y. Zhao, F. Zhou, F. Luo, and R. Shi. SilkViser: A Visual Explorer of Blockchain-based Cryptocurrency Transaction Data. In *Proceedings of IEEE Conference on Visual Analytics Science and Technology*, pp. 95–106, 2020.