Preset-based Generation and Exploration of Visualization Designs

Hans-Jörg Schulz[a,*], Steffen Hadlak[a]

[a] *Fraunhofer Institute for Computer Graphics Research IGD, Rostock, Germany*

Abstract

Generating the "right" visual representation for the data and task at hand remains a standing challenge in visualization research and practice. A variety of different approaches to produce visual representations have been proposed in the past, including such noteworthy instances as *visualization by example* and *visualization by analogy*. With this paper, we add a new twist to creating visual representations by proposing a way to construct new visualization designs by blending together a number of existing visual representations, called *presets*. We embed this novel blending approach in suitable visual interfaces, such as a gridded canvas to be used by the casual user in the style of a palette for mixing colors, or a range of sliders to be used by the expert user in the style of a studio mixer for audio tracks. These can be employed for rapid prototyping of a specific visual representation, as well as to explore the overall design space of visual representations captured by our approach. We showcase our preset-based blending and its interfaces with examples of the design of 2D tree visualizations and product plots.

Keywords: visualization design, design spaces, preset-based interaction, product plots, tree visualization

1. Introduction

Visualization design, i.e., the process of creating a suitable visual representations for a dataset and task at hand, is a complex creative procedure [1]. As a means to reduce the complexity that is exposed to the user, output-driven visualization design approaches, such as *visualization by example* [2] or *visualization by analogy* [3], have been developed. They utilize ways to configure visualization designs, which allow for defining a visual representation in an illustrative manner by exemplifying its result. This stands in contrast to the common algorithmic definition (i.e., mapping) that details the procedure in which the visual representation is produced. Approaches like visualization by example offer a shortcut to generate visualizations without the need to specify this procedure, let alone to write lines of code. Due to this straightforward "What You See Is What You Get" (WYSIWYG) form of visualization design, it holds the promise to play a central role in the efforts to develop visualization for the masses and casual visualization. Their importance is underlined by Heer and Shneiderman [4], who remark that "*novel interfaces for visualization specification are still needed, as new tools requiring little to no programming might place custom visualization design in the hands of broader audiences.*" This implies that in order to contribute new creative means to visualization design, we need both – a practical approach beyond programming to describe visualization configurations, as well as interfaces to make this approach interactively accessible to a broader audience.

In this paper, we propose such a new creative way of generating visual representations in an output-driven manner by following the idea of remixing existing visualizations into new ones. This idea sprung from similar recent endeavors in closely related domains, such as the suggestion of a *data DJ* who remixes live data feeds [5], the use of actual mixing boards to explore visualizations [6] or UI designs [7], as well as the mashup of visualization pipelines [8] and layouts [9]. To realize this idea, we contribute a new approach to describe and configure visualizations[1] through numerical parameters and propose two novel interface variants to make this new form of visualization configuration interactively accessible to casual and expert users alike.

Our approach to describe visual representations is based on a set of numerical configuration parameters that are generic enough to capture both, a visualization's fundamental mapping and its further fine-tuning through individual settings. This is an important strength of our approach, as mapping and fine-tuning are usually considered two separate steps in visualization. Of these two steps, often only the fine-tuning is exhibited to the user allowing for marginal changes to an otherwise fixed mapping. Since our approach encapsulates both in a unified set of configuration parameters, it can express a wide range of different visualizations. Due to the numeric nature of the configuration parameters, this form of specifying visualizations enables us to blend visual designs by interpolating between their respective parameter values. Starting with existing visualizations, which we call *presets* [10], we can thus create new visualizations by remixing them. This is exemplified in Figure 1 for the two examples that we use throughout this paper: visual representations for hierarchically structured data – i.e., *tree visualizations* [11], and visual representations for distributions across different categories – so called *product plots* [12].

[*] Corresponding author: hans-joerg.schulz@igd-r.fraunhofer.de

[1] Note that for the sake of readability and brevity, the term *visualization* is used in this paper synonymously to the terms *visual representation* and *visual design*, even though we do not consider a visualization's interactive features.

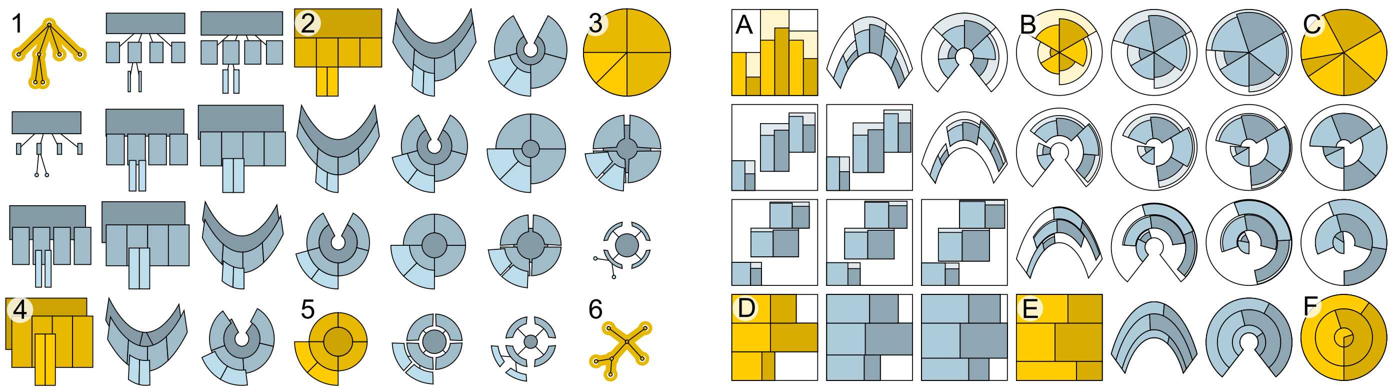


Figure 1: Two sets of intermediate visual designs (blue) generated by blending existing visual representations (yellow presets). The left side shows a set of tree visualizations including the following presets: (1) axis-parallel tree layout, (2) icicle plot [13], (3) pietree [14], (4) cascaded treemap [15], (5) sunburst [16], and (6) radial layout. The right side shows a set of product plots [12] including the following presets: (A) grouped column chart, (B) sector chart, (C) pie chart, (D) stacked bar chart, (E) mosaic plot, and (F) stacked pie chart.

Two interface variants to make this approach accessible are proposed for casual and expert users. The first interface that is mainly thought for the casual user is inspired by the metaphor of an artist's color palette and it aims to make blending visualizations as intuitive as mixing colors. Similar to a number of base colors scattered across the palette and the space in between them being used for mixing them, we place a number of known visualizations (presets) on a gridded canvas and interpolate the visualization designs in the remaining empty grid cells. This allows sampling a visualization from anywhere on the palette, and also rearranging the presets by dragging them across the palette to reposition them and thus to generate different blends. By this means, the user can quickly try possible alternatives to yield a visualization result as an informed choice from the vast range of possible visualizations that would in principal fit the data. The second interface that is mainly thought for the expert user is inspired by the metaphor of an audio mixing device. Similar to the combination of a number of audio channels into a single output signal via sliders, we use one slider per preset to adjust its influence on the blended output design. By means of shifting the sliders, the user can explore the captured visualization design space as a whole to gain a deeper understanding of the effects of individual presets and design parameters.

The following Section 2 gives an overview of the existing approaches for describing visualizations and ties them to the interface classes for visualization design as they were identified in a recent survey [17]. Section 3 introduces our approach for capturing visual representations through a set of numerical parameters and details how to construct such parameter sets for the two examples of tree visualizations and product plots. We then detail our two interfaces for this form of configuring visual representations, the palette and the mixer, in Section 4. Their utility for generating visualization designs by remixing presets is briefly showcased by two examples and a user study in Section 5. Section 6 outlines further extensions, such as how to combine configurations with each other to extend the range of possible visual representations. Finally, Section 7 summarizes our contribution and outlines ideas for future research.

2. Related Work

Over the years, different approaches of capturing and describing various visualizations in a unifying way have been proposed – ranging from early classifications to recent declarative methods. On top of these various ways to describe visualizations, different user interfaces have been built to facilitate their use. In the following, we give an overview of the existing ways of describing and characterizing visualizations, which can be divided into enumerative, constructive, and descriptive approaches, before discussing the gap between choosing and creating visualizations and how our blending approach aims to bridge this gap.

2.1. Enumerative Approaches

Enumerative approaches aim to describe the space of visualization designs through a comprehensive compilation of the existing individual visualizations that constitute it. Common examples for enumerative approaches are *visualization directories* and *visualization references* that list various existing visualization techniques in an order that aids in locating individual techniques – e.g., in an alphabetical order [18] or by using a look-up scheme for data types and visualization tasks [19]. More confined subspaces of visualization designs are described in *visualization taxonomies* (e.g., [20, 11]) and in *visualization typologies* (e.g., [21]). A detailed definition of and distinction between taxonomies and typologies can be found in [22]. In essence they form the two fundamental approaches of classification and thus often employ classes or categories of visualization techniques to structure their listing. This already bears aspects of descriptive top-down approaches (cf. Section 2.3), yet these classes are not (meant to be) specific enough to actually pinpoint and generate individual visualization designs.

The interface type that builds directly on top of these lists of visualizations is the so-called *template editor* [17]. It mirrors the list-style of the underlying enumerative description by providing a fixed list of available visualization techniques (templates) to pick from. While not being strictly output-driven, template editors usually feature iconic representations as examples of what the result will look like. Well known examples are the charting wizards of standard office software packages.

2.2. Constructive Approaches

While enumerative approaches consider a visualization as a whole, constructive approaches capture visualizations in a bottom-up fashion through the individual graphical or functional building blocks that form them. They understand the space of visualization designs as the realm of all possible combinations of these building blocks. In contrast to enumerative approaches, they cannot only capture existing visualizations, but also so far non-existing visualizations that result from novel combinations of these building blocks. Examples for constructive approaches that utilize graphical elements and their visual properties as building blocks are *algebraic notations*, such as the *Grammar of Graphics* [23] or *APT* [24], as well as declarative domain-specific languages, such as *D3* [25, 26] or *VizQL* [27, 28]. Examples for constructive approaches that utilize functional building blocks are *state/operator frameworks*, such as *HiVE* [29] or *GLO-STIX* [30], *generative layouts* [31], or *transformation-based methods* [32] that use high-level operators to form customized visualization procedures.

The most common interface type for constructive approaches is certainly the *textual programming* [17], which is the most direct way to use them. Yet, it is not output-driven, which is why other interfaces have been developed that can be used along the lines of the WYSIWYG-paradigm: the *visual builder* [17] for graphical building blocks and the *visual dataflow programming* [17] for functional building blocks.

The visual builder allows users to construct a desired chart by interactively arranging graphical building blocks on a drawing canvas. Early examples include *Gold* [33] that relies on a rule-based heuristic for linking the graphical elements to the data, and *SageBrush* [34] that builds on the notation of APT. A more recent example is *Lyra* [35] that utilizes *VEGA*, a JSON-based visualization grammar that can be used to specify visual representations and interactions alike [36]. With the advent of modern touch-based input devices, these ideas have been extended, for example, to allow for free-form sketching of charts, as it is realized by *NapkinVis* [37], which uses the D3 predecessor *Protovis* [38, 25] as an underlying description of the sketched visualization. A more restricted interface type is the *shelf configuration* [17], which exposes only parts of the constructive approach in the form of a fixed set of visual variables that can be adjusted through a custom user interface. An example using shelf configuration is Polaris/Tableau [27, 28], which relies on the VizQL language for its realization.

Similar to the visual builder, the visual dataflow programming allows users to interactively arrange and connect functional building blocks into visualization pipelines. Examples include *VANISH* [39], *Gadget/IV* [40], or *iVisDesigner* [41]. While these abolish the need to type lines of code, constructing these pipelines remains a complex and time-consuming task. Thus some incarnations of this interface type provide semi-automatic approaches, such as auto-completion of partial operator pipelines [42] or visualization by analogy [3] to aid the user. In the latter case, the user specifies a desired operator network by choosing a pair of visualizations that denote input and output of the sought pipeline, which is then determined as the difference graph between their respective operator networks.

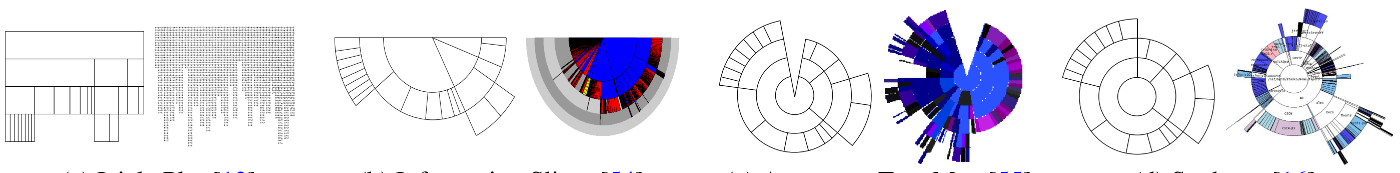
2.3. Descriptive Approaches

Descriptive approaches likewise do not consider a visualization in its entirety, but aim to specify it by detailing a number of independent design characteristics. In contrast to constructive approaches, the space of visualization designs is not an emergent result formed bottom-up by the various possible arrangements of different building blocks. Instead, it is a top-down characterization that already starts with a vision of the diversity it aims to capture through the various design characteristics. Unlike the rough categorical break-down of the taxonomic enumerative approaches, these characteristics are specific enough to fully describe and generate concrete visualizations – existing and novel designs alike. An example of such descriptive approaches are *visualization design spaces* [43, 44, 45] that span the space of visualization designs by establishing independent design dimensions and consider visualizations as tuples of design decisions made along these dimensions. Note that earlier examples of design spaces did not have such a generative notion and were mainly used for classification purposes of existing visualization techniques in an enumerative sense – e.g., [46].

To this point, there exists no standard interface type that fully supports descriptive approaches in the sense that it allows for interactively describing a visualization and thereby generating it in terms of both steps: its fundamental mapping and its detailed parametrization. Yet for the latter step – i.e., the parametrization and fine-tuning of a predetermined principal visualization design or chart type – *visualization spreadsheets* [17] are an established output-driven interface type. They use the design characteristics as independent axes of a parameter space from which to sample different visualization configurations. These samples are then shown as previews in a gridded spreadsheet or on a continuous canvas spanned by the independent axes. Examples are manifold and include spreadsheet-like interfaces for choosing transfer functions [2, 47, 48], selecting viewpoints [49], picking data placements [50], adjusting color maps and isovalues [51], and even for manipulating color distributions [52]. Moreover, such interfaces can act as visual histories that exemplify each interactive adjustment with a respective visualization instance within the configuration space [53].

2.4. The Gap between Choosing and Creating Visualizations

The approaches discussed above allow a user to either choose a visualization as a whole from an enumeration of existing chart types, or to create customized visualizations from scratch – bottom-up by constructing them from a variety of building blocks or top-down by describing their various visual properties and thus pinpointing it in a visualization design space. While choosing from a list of existing visualizations is simple and intuitive, it is also restricted by the choice of visualization techniques provided. Whereas the creation of custom visualizations provides the needed flexibility, but at the cost of being more difficult to



(a) Icicle Plot [13] (b) Information Slices [54] (c) Aggregate Tree Map [55] (d) Sunburst [16]

Figure 2: Between (a) the axis-parallel icicle plot [13] and (d) the radial sunburst [16], a number of intermediate techniques exist. Two such intermediates are, for example, (b) the semicircular information slices [54] and (c) the aggregate tree map [55]. To generate them and other intermediates, we propose weighted blends of the two presets icicle plot and sunburst. A first impression of these blends is shown in the middle column of Figure 1 (left).

use. In between these two principal paths lies the gap of those approaches that one actually wants to use – approaches that are flexible and simple at the same time.

Some approaches aim to bridge this gap from the side of creating custom visualization by reducing its difficulty through the sketching or spreadsheet mechanisms mentioned in the above sections. Even though these approaches are output-driven, they require the user to know in beforehand what they want to create, so that they can combine the appropriate building blocks or configure the relevant visual properties, accordingly. These approaches embody Georg Christoph Lichtenberg's famous aphorism that "*many people, probably most, must, to find something, know in advance that it's there*."

In our work, we aim to go past this restriction by also including solutions, which the user may not yet know they exist. To do so, we pursue an opposite approach by bridging the gap from the side of choosing existing visualization techniques (presets) and allow for blending them to increase the flexibility in terms of the possible outcomes. By sampling the resulting solution space and displaying it, we can furthermore suggest novel visualization blends of which the user may not have previously thought. Thus, this approach makes it possible to build upon the knowledge about the benefits and drawbacks of the existing visualizations, but unlike the pure enumerative approaches, it allows the user to choose a customized compromise between them. The usefulness of this approach is underlined by the fact that a number of such visualization blends already exist as visualization techniques in their own right.

For example, the *icicle plot* for tree-structured data [13] (Figure 2a) has the advantage of an explicitly ordered layout, with the hierarchical order being encoded vertically and any linear order among the siblings being encoded horizontally, making it easy to visually scan and parse it. On the downside, it utilizes the same amount of screen space for each level, whether it holds only a single node, as the root level does, or whether it contains a much larger number of nodes. The opposite to these traits can be found in the *sunburst* [16] (Figure 2d) whose radial alignment yields a more balanced spatial distribution, but at the cost of the explicit vertical and horizontal ordering, as the tree now spreads in all directions.

Existing visualization designs, which compromise between these two techniques and their traits, are on the one hand the *information slices* [54] (Figure 2b) that aim to maintain the vertical and horizontal ordering in part, while nevertheless utilizing the screen space somewhat more effectively through a semicircular design. On the other hand, there are the so-called *aggregate tree maps* [55] (Figure 2c) that do no longer keep the vertical ordering in a top-to-bottom arrangement, but still encode a linear left-to-right order by leaving a small empty wedge in the circular design to denote a clear beginning and end point.

Yet, simply adding these two visualization techniques to an enumerative list of chart types in a visualization system is not enough, as the list will never be exhaustive. The reason is that a compromise between the blended visualizations must be renegotiated each time the underlying data or visualization tasks change. This means that for another input tree with different characteristics and another visualization task to be performed, the user may need to strike another compromise and use yet another visualization, which may not already exist. This is the motivation behind our preset-based visualization approach that is introduced in the following section. Instead of offering only a limited number of such preconfigured compromises – if at all, as most visualization systems do not include any of the above intermediate visualization examples – our approach captures the entire range of possible compromises and offers users to choose from this range through adequate visual interfaces.

3. Defining Parametric Visualization Configurations

Capturing a domain of visualization designs requires to answer the questions of *What to capture?* and *How to capture it?* This section answers these questions – first by introducing our general approach to capture visualizations through numeric design parameters that permit us to blend them, and then by applying this approach to the specifics of our two running examples of product plots and tree visualizations.

3.1. A General Approach to Parametric Visualization Designs

The above questions of *What?* and *How?* target two crucial points of the characterization of visualizations designs:

1. *What?* asks for the *visualization domain*: which kind of visualization designs to capture, how many and which presets to include, as well as which visual characteristics to choose for describing them;

2. *How?* asks for the *visualization specification*: which continuous numerical design parameters can be used in which way to capture the characteristics of the visualization domain including all variants that lie in between.

While the focus and the novelty of this paper lies mainly on the second point, it cannot be used without an understanding of the challenges posed by the first point. Hence we discuss both in the following two sections.

3.1.1. Visualization Domain: What to capture?

In order to actually generate and not merely classify visualizations, their description must be quite specific, which is hard to achieve for a variety of visualization designs that is too broad and thus too diverse to be captured with an overarching set of characteristics. This challenge is not new and its common solution is to confine the description to subsets or types of visualizations, such as only considering tree visualizations [44] or 2-dimensional charts [45]. While this seems restrictive at a first glance, it actually makes a lot of sense, as the data to be visualized will usually not require both of these visualization types at the same time. If the data is hierarchically structured, it is probably best visualized with a tree visualization; and if the data is a set of tuples, the use of a 2-dimensional chart seems to be a logical choice for their depiction.

More complex and heterogeneous data can be visualized by a combination of such simpler visualization approaches. For example, this is the case for time-varying graphs [56] – i.e., data that has network traits requiring a graph visualization and associated dynamics calling for time-oriented visualization. These are often visualized by means of combining the two through a set of common compositing mechanisms, such as juxtaposition, superimposition, or embedding [57]. Therefore, it is reasonable to capture one type of visualization at a time – whereby "type" is used in the usual sense of the general "data type" for which a class of visualization was developed.

While this defines the breadth of the visualization domain, it leaves open its granularity. The granularity is governed by the number of presets one wants to include, as the more presets we aim to capture, the more design characteristics we need to describe them in order to distinguish all the presets from each other. This is evident from looking at an example of tree visualization presets: If we want to include visualizations that are different at a first glance, such as a treemap and a radial node-link-layout, we only need a few visual characteristics to describe them. Yet, if we want to include more similar visualizations, such as slice&dice treemaps [58] and squarified treemaps [59], we need a more detailed description consisting of additional visual properties, such as they were employed in a recent study on tree visualization designs and their interrelations [60].

In general, one can observe an inverse relation between breadth and granularity. If we want to describe a domain at a very fine granularity, we can only do so by reducing its breadth, and vice versa. For example, a visualization characterization that is able to distinguish between different treemap layouts can only do so in a manner that is still practical by reducing its breadth to the subset of sequential, rectangular, space-filling tree visualizations [43]. Whereas on the other end of the spectrum, a visualization characterization that is so wide that it captures the entirety of all visualizations can only do so by reducing its granularity up to a point where the design space is no longer generative [46]. The reason for this is simple: the larger the domain, the more visual characteristics are needed to describe them – but the more visual characteristics we utilize, the harder it becomes to keep them independent of each other. Yet, independence is one of their most important traits, as it allows us to make changes to an individual visual characteristic without implying changes to other visual characteristics.

So finding useful compromises between the breadth and granularity of a visualization description is not simple and many publications have been devoted to this challenge. We acknowledge the complexity of constructing such compromises and advocate to build upon the work that has already been done in this regard. Concretely, this means that we utilize existing design spaces (or parts thereof) as valid and practically proven such compromises between capturing all important visualization presets of a given domain (*completeness*) and maintaining independence between their visual characteristics (*consistency*). We use these design spaces that have been established in their own right as a solid foundation on which to build our parametric visualization configurations.

It is noteworthy that we usually use only binary design dimensions with two either/or design decisions for each dimension in order to ease the process of turning a design space into a parametric configuration, which is described in the following section. Having only two given endpoints that must be met by a parametric transformation makes it much easier to find such a transformation. Yet in principle, design dimensions with more than two design decisions are possible and an example for a parameter that interpolates between three design choices (vertical slice, strip treemap, horizontal slice) is given in Section 6.2.

3.1.2. Visualization Specification: How to capture it?

A design space characterizes a visualization through a number of discrete design choices for all its different visual properties. It is basically a more detailed version of enumeration, only that it does not list pre-defined visualizations from which to choose, but visual properties. So for the example given in Section 2.4, the visual property "alignment" would feature the two mutually exclusive design choices of *radial* and *axis-parallel* [11]. None of the two would capture the intermediate visualizations. Again, we could add them as additional design choices, yet it would never be exhaustive. That is why we aim to turn the discrete design choices into a more continuous spectrum from which we can freely choose. To do so, we utilize three tools: *transformation*, *deformation*, and *combination*.

Transformation is used where we can express the spectrum between two or more discrete design decisions by simple translation, rotation, or scaling. It is probably the most established way of gradually changing the visual appearance, as it is governed by a numerical parameter – i.e., an offset along a translation vector, a rotation angle, or a scaling factor. An example for scaling is the transformation between implicit, space-filling tree visualizations and explicit, node-link tree visualizations, as described in [31]: when the nodes are at their full size, they are space-filling and mask the underlying edges completely; but when the nodes are scaled-down to their minimal size, they are mere dots and the edges connecting them are clearly visible.

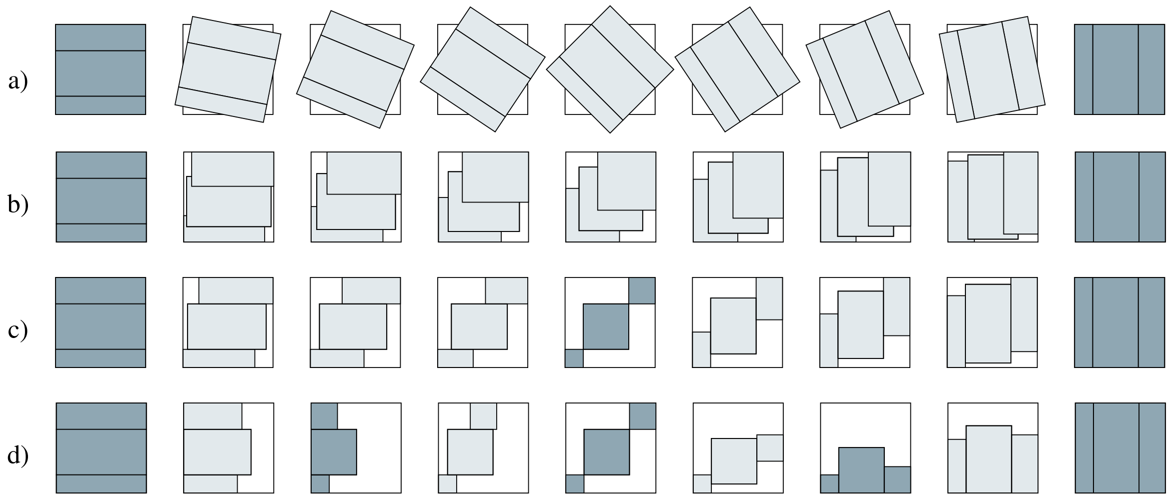


Figure 3: Four possible transitions that interpolate between vertical and horizontal orientation, with the "key frames" shown in a darker blue and the "inbetweens" shown in a lighter blue: (a) rotation, (b) simultaneous scaling and translation in horizontal and vertical direction, (c) staged transition in two steps: 1. simultaneous horizontal scaling and translation, 2. simultaneous vertical scaling and translation, (d) staged transition in four steps: 1. horizontal scaling, 2. horizontal translation, 3. vertical translation, 4. vertical scaling.

Deformation is used for more complex transitions that cannot be expressed by transformation and require a continuous morphing between discrete designs. These are usually pieced together from simpler distortions as functional building blocks. An example that can be realized through deformation is the transformation between straight, axis-parallel visualizations and fully curved, radial visualizations, as it is shown in [61, Fig.13].

Combination is used to couple multiple transitions together – in parallel as a *simultaneous transition*, in sequence as a *staged transition* [62], or any combination of the two. An example is given in Figure 3 for the simple case of transforming a vertically sliced proportional area plot into one that is horizontally sliced. Both of these instances are common subdivisions as they occur in mosaic plots or treemaps, as well. The naïve transition would be to rotate the entire rectangle (Figure 3a). Yet this leads to the situation in which the intermediate stages lie partially outside of the rectangle, which means that they are either not fully visible (if the rectangle is the viewing area) or they occlude other information (if the rectangle is a drawing area that is part of a larger view). Another option is to simultaneously apply scaling and translation, both in horizontal and vertical direction, to the individual subdivisions (Figure 3b). While this ensures that the intermediate stages do not cross the rectangle boundaries, it leads to the problem that the individual subdivisions overlap and thus occlude each other, which is equally disadvantageous. So, we set apart the simultaneously applied transformations and apply them in sequence. We can subdivide the simultaneous transition in either two steps of first scaling and translating horizontally, and then performing the vertical scaling and translation afterwards (Figure 3c); or we can even subdivide it into four steps by further separating the scaling and the translation into transition steps in their own right (Figure 3d). These two transition variants resolve the problems of the earlier transitions, yet they do so at a cost: while the first two variants maintained a correct mapping of a numerical attribute to each partition's area, the latter two variants do not. Hence, the generated intermediate visualizations are only of limited use if one wanted to carry out a task that involved getting correct readings on this attribute from the areas, but they would still be useful if only the number of partitions were necessary for a task.

In general, one would aim to maintain as much of the initial mapping as possible in order to facilitate a diverse set of possible tasks to be carried out on any of the intermediate visualizations. Yet trying to maintain the mapping in full can result in rather complex transitioning schemes. For example, to resolve the problems of the staged transitions in Figures 3c and 3d, we developed a staged transition that requires to incorporate two visual design features at once, as it is discussed in the following section. Hence, if the task to be carried out does not require it, we may choose a simpler transition that does not maintain certain aspects of the mapping. In some cases, we may not even be able to maintain the mapping even with a complex transition, as one of the two design choices between which we want to transition does not offer such a mapping. For example, when transitioning from a space-filling tree visualization to a node-link visualization, the latter does not offer to map a numeric attribute onto the node size.

Finally, once a suitable transition is chosen, we need to map it onto a continuous $[0\ldots1]$ interval. In case of a single transformation, this can easily be done by normalizing the transformation parameter (e.g., the rotation angle or the scaling factor) to $[0\ldots1]$. For simultaneous transitions, the transition parameter in $[0\ldots1]$ can be seen as a high-level compound parameter that simultaneously steers all individual transition parameters, which is a known concept in visualization [63]. For sequential transitions, the transition parameter interval $[0\ldots1]$ has to be subdivided into subintervals for the individual stages of the transition. Note that this mapping onto $[0\ldots1]$ does not necessarily have to be linear and the subdivision into subintervals for sequential transitions does not have to be equidistant. Further examples show that it can be beneficial to use non-linear mappings in order to yield perceptual linearity in the outcome [64] or to increase tracking performance [65].

All three of the above tools – transformation, deformation, and combination – have in common that they operate solely on the geometry of the visualization design. This makes them independent from the algorithm with which the design was created. Thus, it is in theory possible to create presets with their respective native layout algorithms that may function very differently and have different runtimes, and to interpolate between their resulting geometry, similar to the approach of *transmogrification* [61]. In practice though, instead of implementing all the different layout algorithms, it is often simpler to use a custom layout algorithm that produces a generic base layout and applies said transformations to yield the necessary presets. This approach allows the user to define new presets on the fly by adjusting these transformations, as it is exemplified in Section 6.1.

3.2. Parametric Product Plots

Following the two steps outlined above, we build our parametric description of product plots on selected parts of a design space put forward in the seminal work by Wickham and Hofmann [12]. They describe product plots in the form of a mathematical framework for combining rectangular shapes into a variety of visualizations for showing numerical distributions over various categories. As it is outside the scope of this paper to recap the wealth of construction possibilities and plots put forward in their work, the interested reader is referred to the original publication [12] for more details. We use the 1-dimensional atoms for 2-tiered charts (i.e., showing a two-fold categorization, such as age groups on tier 1 and gender on tier 2) to build a design space of product plots with five dimensions:

PI) **Tier 1 – vertical vs. horizontal:** Product plots can be oriented either vertically (column chart) or horizontally (bar chart).

PII) **Tier 1 – bar vs. spine:** Product plots using bars (bar charts or column charts) map a numerical variable onto the height of rectangles of uniform width, whereas those using spines map the numerical variable onto the width of rectangles of uniform height (subdivision, slicing).

PIII) **Tier 2 – vertical vs. horizontal:** Same as PI, but in this case to configure the orientation of the rectangles of the second-level categories nested inside the rectangles of the first-level categories.

PIV) **Tier 2 – bar vs. spine:** Same as PII, but in this case to configure the mapping to height vs. width of the rectangles of the second-level categories nested inside the rectangles of the first-level categories.

PV) **axis-parallel vs. radial:** Axis-parallel product plots subdivide along the x/y-axes of a Cartesian coordinate system to generate rectangles (mosaic plot). Radial product plots subdivide along the radius/angle of a polar coordinate system to generate circle segments (pie chart).

These five design dimensions can be used to describe 32 different design combinations, as they are shown in Figure 4. These 32 design combinations define our *visualization domain* by providing the cornerstones between which the *visualization specification* must interpolate to describe all possible intermediate product plots. Finding and realizing such a scheme is the second step of our approach.

For PI and PIII, we showed in Section 3.1.2 that the straightforward interpolation between horizontal and vertical orientation produce either cropped, overlapping, or unfaithful intermediate representations. This is fine, if the aim of the intermediate stages is to animate a transition between two chart types so that the viewer can maintain his mental map, but it is not sufficient if the intermediates shall be used as visualizations in their own right. To interpolate between orientations while maintaining a faithful mapping of a numerical attribute onto the area, we create a staged transition that takes both into account – the vertical/horizontal design decision (PI and PIII) and the bar/spine design decision (PII and PIV). The input parameters of this interpolation are $0 \leq hv \leq 1$ with $0 =$ *horizontal orientation* and $1 =$ *vertical orientation*, as well as $0 \leq bs \leq 1$ with $0 =$ *bar* and $1 =$ *spine*. We determine an intermediate product plot with given parameter values hv and bs from a number of base diagrams, as it is schematically depicted in Figure 5. The four endpoints of the combined interpolation are:

- *HB*: column chart $hv = 0, bs = 0$
- *VB*: bar chart $hv = 1, bs = 0$
- *HS*: horizontal proportional area chart $hv = 0, bs = 1$
- *VS*: vertical proportional area chart $hv = 1, bs = 1$

On top of these four, we define two additional base diagrams that form intermediate stages ("key frames") between them, but maintain a correct mapping onto the area:

- *HVS* ($hv = 0.5, bs = 1$) can be thought of as an area-preserving variant of the intermediate stages that stand in the center of the transitions depicted in Figure 3c+d.
- *HVB* ($hv = 0.5, bs = 0$) is very similar to *HVS*, but the final mapping scales the differently sized squares into equally sized cells to produce a combination of the columns of equal width in *HB* and the bars of equal height in *VB*.

The detailed procedure of generating these two plots is given in Algorithm 1, which takes the kind of the plot (*HVS* vs. *HVB*) as an input parameter. Its runtime complexity is $\mathcal{O}(|\text{values}|)$ as it considers each value to be mapped onto the base layout only once. Note that for the sake of brevity, Algorithm 1 and all following algorithms do not detail the simple but tedious procedure of normalizing width and height of all rectangles in the resulting chart to a given drawing area by means of stretching or shrinking them after the transformation.

Unless the given parameter values for hv and bs already match any of these six base diagrams, we interpolate in a first step between bar and spine. If $hv \leq 0.5$, then we compute H

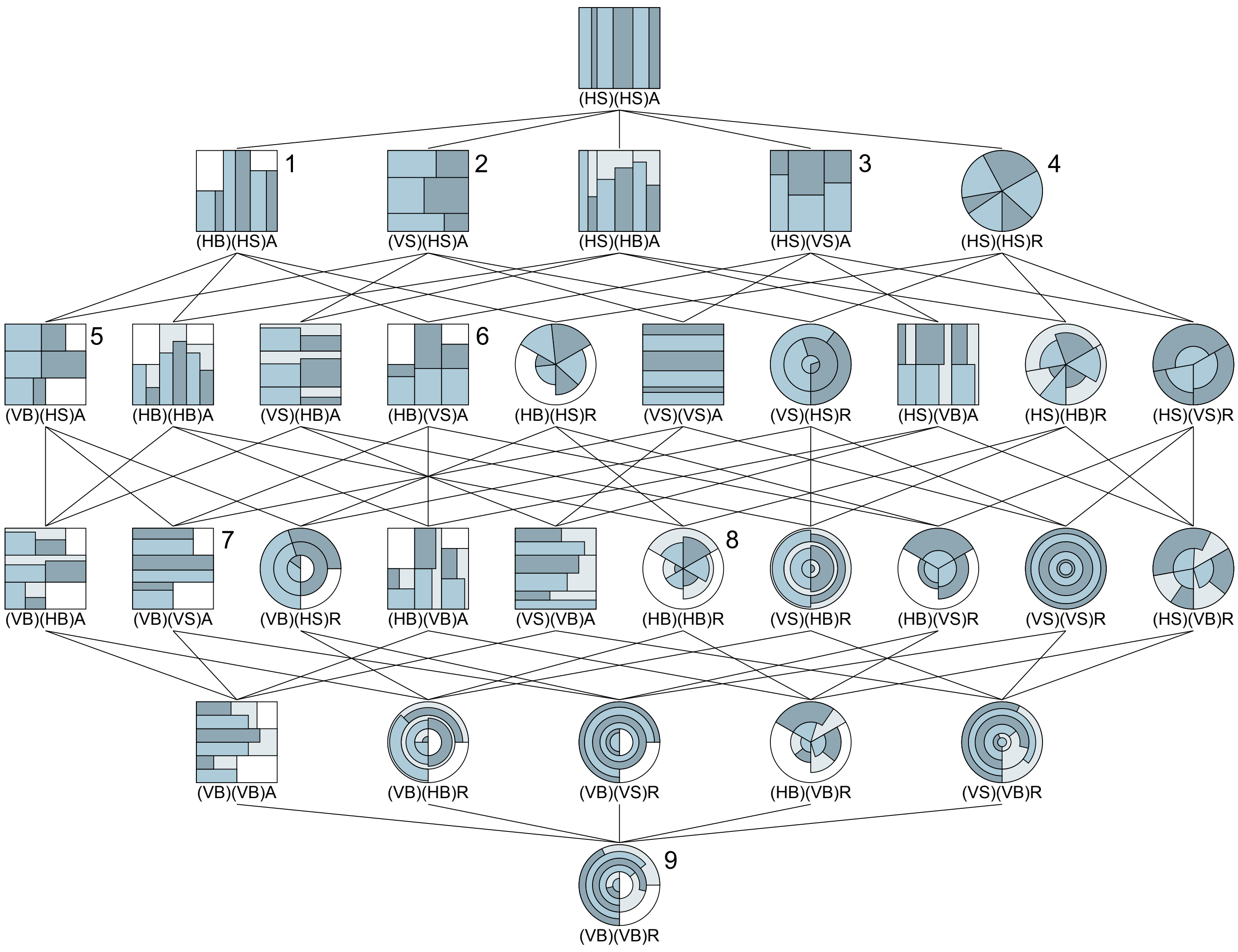
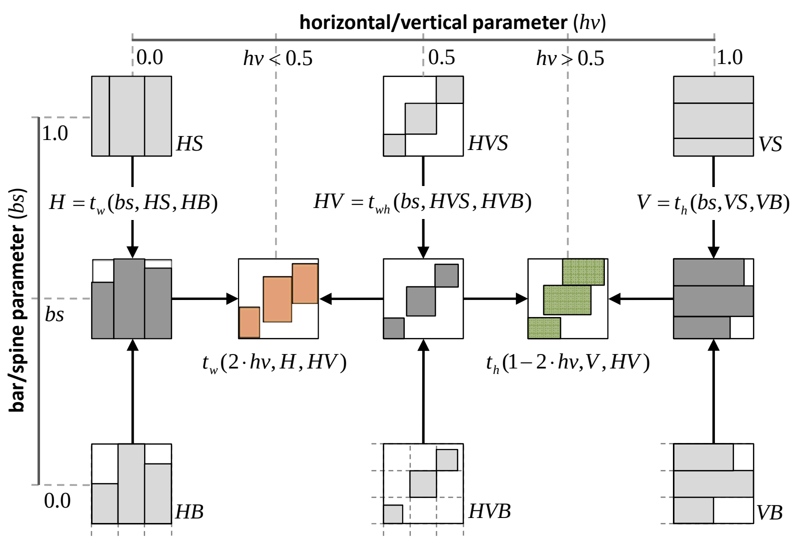


Figure 4: Hasse diagram of our 5-dimensional design space of 2-tiered product plots. The different design decisions are noted underneath the techniques with H/V standing for horizontal/vertical, B/S standing for bar/spine, and A/R standing for axis-parallel/radial. The two brackets in the notation denote the design decisions for tier 1 and for tier 2 in this order. The design space already captures a wide variety of existing product plots, such as (1) grouped column chart, (2)+(3) mosaic plot, (4) pie chart, (5) stacked bar chart, (6) stacked column chart, (7) grouped bar chart, (8) sector graph, and (9) racetrack plot – but also a large number of yet unnamed and potentially novel design combinations.

and HV using transformations t_w and t_{wh}, respectively – otherwise ($hv > 0.5$) we compute HV and V using transformations t_{wh} and t_h, respectively. In a second step, we interpolate between horizontal and vertical: If $hv \leq 0.5$, then we compute the intermediate between H and HV using transformation t_w – otherwise ($hv > 0.5$) we compute the intermediate between HV and V using transformation t_h. Together with a linear interpolation t_{pos} for the positions of the generated rectangles, this two-step procedure is schematically shown in Figure 5. Note that this procedure does not introduce a dependence between these two parameters, as they can still independently be changed and one does not have an effect on the other. Merely the interpolation process needs to take both into account at the same time in order to maintain a correct proportional mapping onto the area.

This leaves design dimension PV – i.e., the interpolation between axis-parallel and radial. For its realization, we use the plot's curvature and assume an underlying polar coordinate system in which we map the plot onto an angular stretch at a specific radius. For an axis-parallel plot ($curvature = 0$), the angular stretch is very narrow and the radius very high, which produces a seemingly straight plot whose curvature is within subpixel range and thus not visible. Whereas for a radial plot ($curvature = 1$), the angular stretch covers the full 360° at a smaller radius. Between these two endpoints, we interpolate by lowering the radius and at the same time increasing the angle in order to increase the curvature from 0 to 1, as it is schematically shown in Figure 6. The full procedure is detailed in Algorithm 2, which describes two functions: the first one for transforming individual points and the second for rectangles, making use of the first one. Note that this algorithm does not produce a "truly" curved shape, but approximates it by sampling its inner and outer boundary with $0 < \#samples \leq 500$ sample points, depending on the length of a rectangle's side. The runtime complexity of this algorithm is $\mathcal{O}(1)$, as it is bound by this sampling



$$t_w(p, A_1[w_1,h_1], A_2[w_2,h_2]) = A_3[w_3,h_3] \text{ with}$$
$$A_3 = (1-p)\cdot A_1 + p\cdot A_2$$
$$w_3 = (1-p)\cdot w_1 + p\cdot w_2$$
$$h_3 = A_3 / w_3$$

$$t_h(p, A_1[w_1,h_1], A_2[w_2,h_2]) = A_3[w_3,h_3] \text{ with}$$
$$A_3 = (1-p)\cdot A_1 + p\cdot A_2$$
$$h_3 = (1-p)\cdot h_1 + p\cdot h_2$$
$$w_3 = A_3 / h_3$$

$$t_{wh}(p, A_1[w_1,h_1], A_2[w_2,h_2]) = A_3[w_3,h_3] \text{ with}$$
$$A_3 = (1-p)\cdot A_1 + p\cdot A_2$$
$$w_3 = h_3 = \sqrt{A_3}$$

$$t_{pos}(p, A_1[x_1,y_1], A_2[x_2,y_2]) = A_3[x_3,y_3] \text{ with}$$
$$x_3 = (1-p)\cdot x_1 + p\cdot x_2$$
$$y_3 = (1-p)\cdot y_1 + p\cdot y_2$$

Figure 5: Schematic depiction of the 2-step interpolation scheme for a given horizontal/vertical-parameter hv and a bar/spine-parameter bs. The first step interpolates the light gray base diagrams according to the bs parameter value to form the dark gray intermediates. The second step interpolates these intermediates according to the hv parameter value into the final representations. The transformation functions interpolate between two rectangular areas A_1 and A_2 with given heights h_1, h_2 and widths w_1, w_2 according to a given parameter value p either along the width (t_w), the height (t_h), or both simultaneously (t_{wh}), if $w = h$. Since the area is interpolated linearly in all three transformations and its width and height are determined accordingly, any mapping of a numeric attribute value onto the area will be preserved. The positions (e.g., the midpoints) of the areas are interpolated linearly using t_{pos} in all steps.

constant of 500, which has proven to be fine-grained enough in practice to ensure a curved appearance in all observed cases. Since this algorithm is invoked once for every value to be displayed, the overall complexity is $\mathcal{O}(|\text{values}|)$, as it was the case for Algorithm 1.

The transition between axis-parallel and radial is an example for a case in which an existing mapping onto the area of an axis-parallel plot cannot be preserved. Radial plots are usually read via the angular proportions of the resulting shapes as for the same angle shapes of different areas can exist depending on the radius at which they lie. Hence, we do not aim for an

Algorithm 1 Generating the Base Layouts HVB and HVS

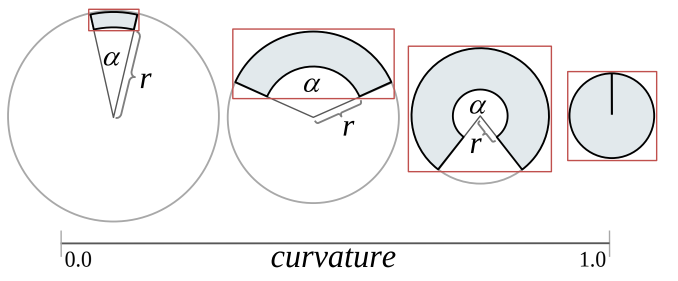
```
function HV($bs$, values[])
                    ▷ $bs \in \{B, S\}$; values[] to map on rectangles
    rects ← $\varnothing$                    ▷ create empty list of results
    $x \leftarrow y \leftarrow 0$                    ▷ start at lower left side
    step ← $\sqrt{\max(\text{values[]})}$      ▷ side of largest square
    for v ∈ values[] do                 ▷ iterate over all values
        $w \leftarrow h \leftarrow \sqrt{v}$            ▷ width = height = $\sqrt{\text{value}}$
        rects ← rects ∪ $\{(x, y, w, h)\}$
        if $bs = B$ then                      ▷ generate HVB
            $x \leftarrow x +$ step
            $y \leftarrow y +$ step
        else                                  ▷ generate HVS
            $x \leftarrow x + w$
            $y \leftarrow y + h$
        end if
    end for
    return rects
end function
```

Algorithm 2 Curvature transformation

```
function CURVPOINT($c, P_{x,y}$)  ▷ $c$ = curvature $(0 \le c \le 1)$
                                  ▷ $P$ = point $(0 \le P_x, P_y \le 1)$
    $r \leftarrow 1/c - 1$                  ▷ radius of circle segment
    $\alpha \leftarrow c * 2 * \pi$            ▷ angular width of circle segment
    $\alpha_x \leftarrow \alpha * P_x$          ▷ angular position $(0 \le \alpha_x \le \alpha)$
    $x \leftarrow \sin(\alpha_x) * (r + P_y)$          ▷ horizontal position
    $y \leftarrow \cos(\alpha_x) * (r + P_y)$            ▷ vertical position
    return $(x, y)$
end function

function CURVRECT($c, R_{x,y,w,h}$)
                                  ▷ $c$ = curvature $(0 \le c \le 1)$
                      ▷ $R$ = rectangle $(0 \le R_x, R_y, R_w, R_h \le 1)$
                            with $R_x + R_w \le 1, R_y + R_h \le 1$
    #samples $= 500 * R_w$     ▷ compute sampling resolution
    polygon[] ← $\varnothing$                     ▷ initialize polygon
    for $i : 0 \le i \le$ #samples do  ▷ iteration over lower side
        $P \leftarrow (R_x + R_w * i/\#\text{samples}, R_y)$  ▷ interpolate point
        $P \leftarrow$ CURVPOINT$(c, P)$                ▷ transform point
        polygon[].add($P$)                ▷ add point to polygon
    end for

    for $i$ : #samples $\ge i \ge 0$ do  ▷ iteration over upper side
        $P \leftarrow (R_x + R_w * i/\#\text{samples}, R_y + R_h)$
                                              ▷ interpolate point
        $P \leftarrow$ CURVPOINT$(c, P)$                ▷ transform point
        polygon[].add($P$)                ▷ add point to polygon
    end for
    return polygon[]
end function
```



$$\alpha = 360° \cdot curvature \qquad r = \frac{1}{curvature} - 1$$

Figure 6: Interpolation between an axis-parallel area with no visible curvature (small angle α, large radius $r \Rightarrow$ rectangle) and a fully curved radial area ($\alpha = 360°$, small radius $r \Rightarrow$ circle).

area-preserving transformation in this case.

To visually summarize our detailed discussions of the interpolations for all five design dimensions, Figure 7 exemplifies the individual effects of each of the five product plot design parameters on a grouped column chart as a base visualization. Yet it is also possible to combine the effects of these interpolations to describe intermediates along multiple dimensions at once by configuring them in the form of a 5-tuple of parameters $(hv_1, bs_1, hv_2, bs_2, curvature)$.

3.3. Parametric Tree Layouts

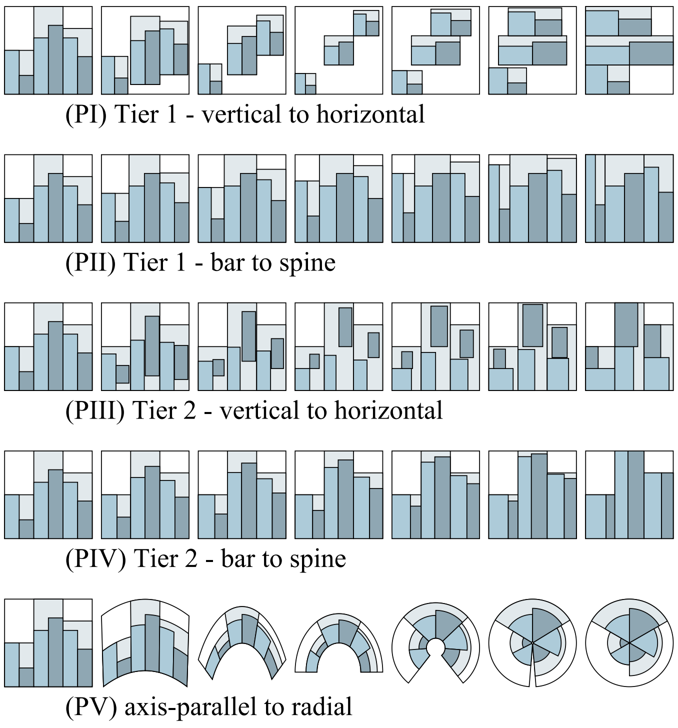
Establishing a parametric description of tree layouts follows again the two steps of first defining a suitable design space and then interpolating between its design choices. As tree visualizations are a well-researched domain, there is plenty of literature from which to choose or build a design space. The one we use here is loosely derived from work on generative tree visualization, including [31, 43, 44]. It features five design dimensions:

TI) **explicit vs. implicit:** Explicit tree visualizations are node-link layouts that rely on connector lines to associate a parent node with its children. Implicit tree visualizations are mainly space-filling layouts that rely on positional relations to encode the parent-child relationship.

TII) **structure vs. attributes:** Structure-centric tree visualizations focus their layout on conveying the hierarchy, whereas attribute-centric tree visualizations focus on a numerical node attribute to be conveyed, which is usually achieved by proportionally scaling the node primitives.

TIII) **aligned vs. cascaded:** Aligned tree visualizations layout a parent's children directly on top or underneath it, while cascaded tree visualizations shift the layout slightly in order to keep at least portions of the parent still visible.

TIV) **inclusion vs. adjacency:** Inclusion-based tree visualizations draw a parent's children on top of it, so that their drawing space is included in the drawing space of the parent. Adjacency-based tree visualizations draw the child nodes right beside the parent node instead of on its top.



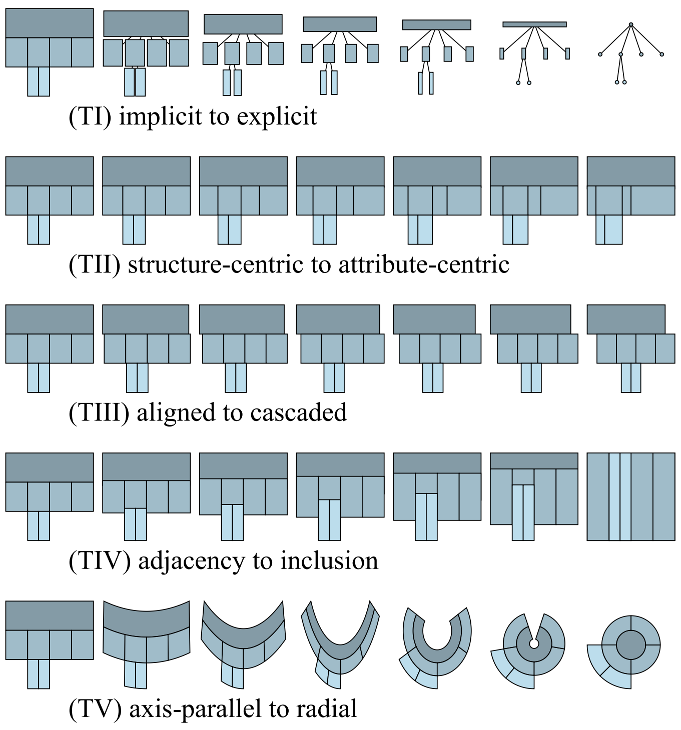
(PI) Tier 1 - vertical to horizontal
(PII) Tier 1 - bar to spine
(PIII) Tier 2 - vertical to horizontal
(PIV) Tier 2 - bar to spine
(PV) axis-parallel to radial

Figure 7: Example of the effect of the five continuous visualization parameters on a grouped column chart. From top to bottom: (PI) Tier 1 – vertical to horizontal, (PII) Tier 1 – bar to spine, (PIII) Tier 2 – vertical to horizontal, (PIV) Tier 2 – bar to spine, (PV) axis-parallel to radial.

TV) **axis-parallel vs. radial:** Axis-parallel tree visualizations map the hierarchy levels onto the y-axis of a Cartesian coordinate system. Radial tree visualizations map the levels onto different radii in a polar coordinate system.

These five design dimensions are sufficient to describe a wide variety of tree visualizations, for example, all those visualizations shown in yellow in Figure 1 (left). This yields a visualization domain of 32 different design combinations between which we want to specify any intermediate visualization designs.

To do so, the second step of interpolating between these either/or design choices uses rather simple transformations, as compared to the complex staged transformation described for the product plots. For the switch between implicit and explicit along dimension TI, we use the duality of the two, as it is described in [31], by basically computing an implicit layout in both cases and merely reducing the size of the node primitives to turn it into an explicit layout. The transformation between structure-centric and attribute-centric visualization, as captured by design dimension TII, is realized as a linear interpolation between having all node attributes set to the number of leaves it contains (effectively yielding equal space distribution for all leaves) and any numerical node attribute of the user's choosing. It is important though, that this attribute follows a hierarchical aggregation scheme, so that for any internal node its value is equal to the sum of the attribute values of its descendent leaves. Design dimensions TIII and TIV are continuously interpolated through a smooth displacement in x- and y-direction, respectively. Finally, design dimension TV uses the curvature inter-



(TI) implicit to explicit
(TII) structure-centric to attribute-centric
(TIII) aligned to cascaded
(TIV) adjacency to inclusion
(TV) axis-parallel to radial

Figure 8: Example of the effect of the five continuous visualization parameters on an icicle plot. From top to bottom: (TI) implicit to explicit, (TII) structure-centric to attribute-centric, (TIII) aligned to cascaded, (TIV) adjacency to inclusion, (TV) axis-parallel to radial.

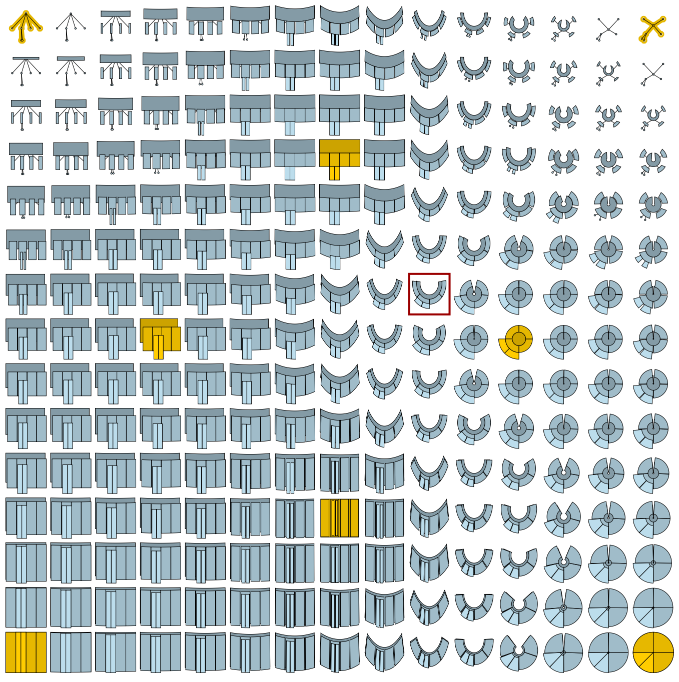


Figure 9: The *visualization palette* smoothly interpolates between the visualization presets placed on a 2D grid (yellow). The presets can be rearranged and the palette can be sampled at any point (red cursor). Note that due to its dimensions, it is usually not suitable to show a large dataset in this form, but only an iconic representation while viewing the actual dataset in a linked view using the visualization design that is currently sampled.

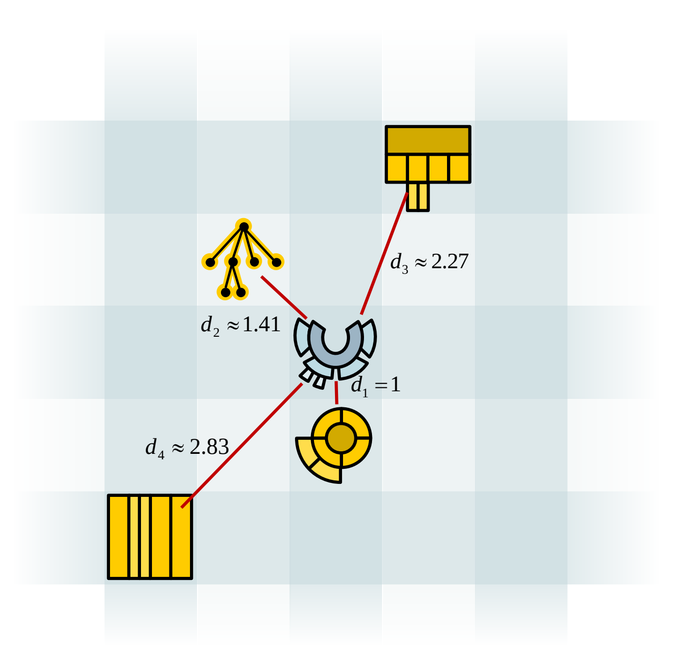
polation that was described for PV in the previous section.

The individual influence of each of the five resulting design parameters on an icicle plot as a base representation is depicted in Figure 8. Again, the effects of these interpolations can be combined to specify any intermediate design as a configuration in the form of a 5-tuple $(size, attributed, \Delta x, \Delta y, curvature)$.

4. Interfaces for Parametric Visualization Configurations

Having established the description of visualization designs in the form of parametric configurations, this section introduces two novel interfaces for preset-based visualization design using these configurations: the *visualization palette* and the *visualization mixer*, as their metaphors are a perfect fit for our blending approach. To showcase the interfaces, we use the tree visualization design as a consistent example throughout this section. The use cases described in the following Section 5 will also show further examples of these interfaces being applied to the design of product plots.

The **visualization palette** shown in Figure 9 is inspired in its handling by an artist's color palette and follows in its technical realization the preset-based parameter space visualization developed by van Wijk and van Overveld [10]. It positions a number of preset visualizations on a grid, which are shown in yellow in Figure 9. The palette then uses the parametric visualization descriptions to smoothly interpolate between the presets and to fill in the remaining blank grid cells, which are shown in blue in Figure 9. This is done by computing for every intermediate grid cell the weighted average of the k nearest presets. These are determined by their Euclidian distance on the 2D grid and their weights are chosen inverse proportionally to this distance, which is normalized with respect to the distance of the $k+1$ nearest preset. This approach is illustrated in Figure 10. While other approaches for filling the non-preset grid cells based on the presets would also be possible, we follow the procedure put forward in [10] and for many more details on the rationale behind it, we refer the interest reader to this publication. As a rule of thumb, we achieved good results with $k = 5$, which takes into account some, but not all of the presets to prevent distant visualizations from affecting a grid cell – e.g., a radial layout anywhere on the palette would otherwise introduce a slight curvature to all non-preset grid cells, no matter how far away they are. The scope of the interpolation, i.e., the value of k, can be adjusted by the user if needed. In order to not overcrowd the palette with the actual visualization of a possibly large dataset, we only use a very small fixed dataset to generate iconic representations of the visualization designs in each grid cell. A user can sample the palette at any point (the red square-shaped cursor in Figure 9) to pick the parametrization of the shown iconic representations and apply it to the full dataset shown alongside the palette in a linked view. Furthermore, the user can drag any of the presets freely across the palette to reposition them and thus to reconfigure the palette's setup in order to generate different blends from those currently shown. For example, dragging two presets further apart increases the number of intermediate designs in between them, which in turn permits



$$= \frac{1}{\sum_{i=1}^{k} w_i} \cdot \sum_{i=1}^{k} w_i \cdot \begin{pmatrix} \text{node size}_i \\ \text{weight}_i \\ \text{x displacement}_i \\ \text{y displacement}_i \\ \text{curvature}_i \end{pmatrix} \quad \text{with} \quad \begin{matrix} w_i = \frac{1}{d_i} - \frac{1}{d_{k+1}} \\ k = 3 \end{matrix}$$

Figure 10: Example of the weighted sum interpolation between presets to compute the visualization design for a non-preset grid cell in the palette. The used interpolation follows the approach introduced by van Wijk and van Overveld [10].

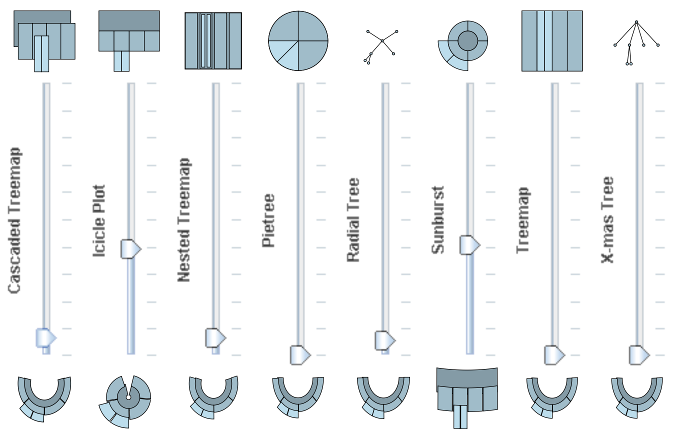


Figure 11: The *visualization mixer* allows users to adjust the influence of each individual visualization preset and combines them via a convex combination. At the top and at the bottom of each slider, a preview shows how the resulting visualization would look like if the slider was moved all the way up or down, respectively. Since there is still hardly enough space to show a large dataset at each slider, only an iconic representation is given while viewing the actual dataset in a linked view.

a user to sample from them in a more fine-grained manner.

While the visualization palette gives a good preview of the various intermediate visualization designs from which the user can pick, it is still limited in the sense that it shows only one possible 2-dimensional arrangement of the high-dimensional space that all possible preset combinations span. This is the reason why we see it first and foremost as a tool for the casual user, whereas the expert user might want a more direct and more comprehensive access to this high-dimensional space.

For these users, the **visualization mixer** that is shown in Figure 11 is the interface of choice. Like a mixing device for audio tracks, it allows for directly adjusting the influence of individual presets onto the output design – yet, it does no longer preview how the result will look like. Note that this missing preview motivated the development of the preset-based parameter space visualization [10] that underlies our palette visualization in a similar scenario – namely the issue of blending different synthesized instruments into new voices. Instead of musical instruments or audio tracks, our mixer allows for blending visualization presets by computing a weighted average as a linear combination of their corresponding parametric descriptions. Their individual weights can be chosen by the user via a slider, yet they have to add-up to 1.0, effectively forming a convex combination of visualizations. Hence, when the user adjusts the weight of one preset, all other weights are adjusted accordingly to ensure this constraint. This means, for example, as the influence of one preset is increased, all the other presets are decreased proportionally to their current influence. We decided to give at least some indication through iconized representations at both end of each slider on how the resulting visualization design would look like, if the slider was moved all the way up or down. At the sliders' top, each of the presets is shown in its pure form, as this would be how the resulting visualization looked like if the slider was moved up all the way to a weight of 1.0 for this visualization and consequently 0.0 for all others. Below each slider, the resulting visualization is shown if the corresponding visualization was to be removed (weight $= 0.0$) from the current combination. While the preview at the top of each slider remains fixed, the outcome of pushing a slider all the way down depends on the settings of all other sliders and thus changes when the sliders are moved. More elaborate preview mechanisms, such as composite parallel coordinate plots [66], can be used instead if one expects more fluctuation of the outcome along each slider.

Note that while it is possible to transition from the "casual" palette interface to the "expert" mixer interface, as any sampled grid cell can be shown as a linear combination of presets in the visualization mixer, this does not work the other way around. The reason is that the mixer permits the user to set a weight combination that does not necessarily have a corresponding position in a currently shown palette setup.

Finally, these interfaces do require a linked view showing the full dataset using the current design configuration that was picked from the palette or set via the sliders of the mixer. This overall setup is shown in the screenshot in Figure 12. In this linked output view, we additionally display two starglyphs [67] in the corners of the view to give an indication of the parametric configuration (i.e., the 5-tuple) that describes and generates the current visualization design, as well as of a number of quality metrics of this design displaying the currently loaded dataset. For the product plots, we measure their quality by computing how well they reflect the given numerical values by their height,

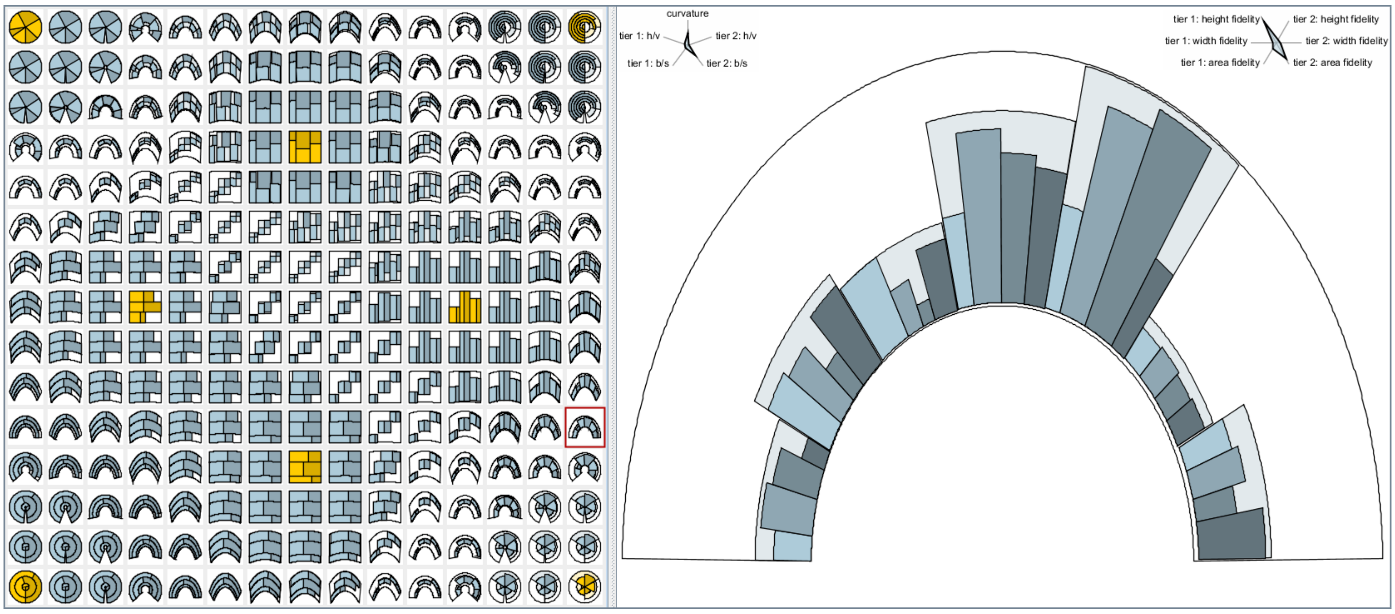


Figure 12: Screenshot showing the palette interface on the left and the full dataset in the picked visualization design (red cursor) on the right. The left starglyph depicts the parameter values of the currently shown visualization design. It shows mainly that the curvature parameter (PV) is set to about 0.5 and the tier 2 bar/spine parameter (PIV) is also set around 0.5, putting it half-way between a column chart and a horizontal subdivision. The right starglyph depicts the quality metrics, showing that the tier 1 values are best determined from the height of the tier 1 geometric primitives, and that the tier-2 values are best determined from the area of the tier-2 geometric primitives.

their width, and their area. We call this measure *fidelity* and compute it for both tiers independently. The computation basically counts the number of different mapping factors between values and geometric property (width, height, area). For radial designs, width is understood as the angle α that a circle segment spans. In the best case (fidelity $= 1$) all mapping factors for a geometric property are the same, which means that the values are encoded by and can be determined from a geometric property. In the worst case (fidelity $= 0$) all mapping factors are different, which means that they are not encoded in a particular property. In Figure 12, one can see from the quality glyph at the top right that the values of the first tier are well represented by the height of the tier-1 geometric objects, whereas the values of the second tier can be determined from the areas of the tier-2 geometric objects.

For the tree visualizations, we use the following quality metrics:

- the current visualization's space utilization (*ink-paper-ratio* [68]),
- the nodes' fidelity of representing a given numerical attribute through their areas (computed as above for product plots),
- the percentage of the drawing space that is occupied by more than one node (*overplotted%* [69]),
- the percentage of nodes that are occluded by sibling nodes (horizontally *overcrowded%* [69]), and
- the percentage of nodes that are occluded by child nodes (vertically *overcrowded%* [69]).

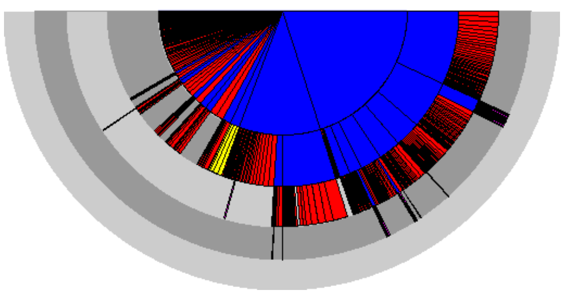
Note that all quality metrics are computed for the dataset at hand and cannot be generalized as properties of the design in general. For example, the particular tree structure of one dataset may produce massive overplotting, while another one is shown fine. This is in line with the common visualization design experience that different datasets require different visualizations. The same is true for the task-dependence of visualization design: while one design supports correct readings of numerical node attributes, but hampers the ease of determining the number of nodes, another one does it the other way around. The starglyphs showing the quality metrics allow the user to judge how well a currently picked visualization design fits the data and task at hand.

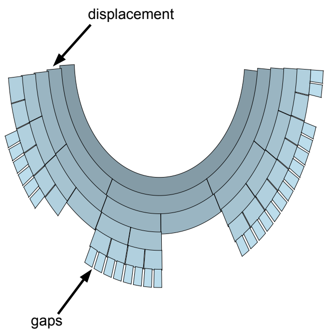
5. Usage Examples and User Study

This section briefly describes two usage examples that are also featured in the video material, which accompanies this paper and which showcases the interactivity of our approach in a way that the static images in this paper cannot. These examples were also used as part of a small qualitative user study on which we report after introducing them.

5.1. Usage Examples

The examples presented here aim to support our claim from Section 2.4 that the blending approach allows us to recreate intermediate visualization designs that exist as visualization techniques in their own right. Thus, the aim of both examples is to recreate visualization prototypes that are not part of the set of the 32 visualization designs that can be generated from the


(a) Information Slice [54]



(b) First Approximation

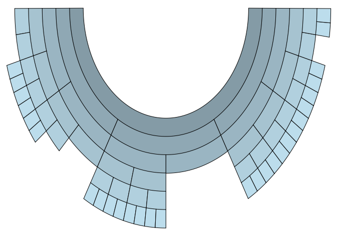
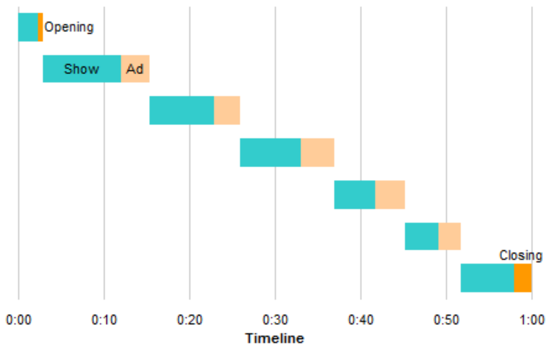
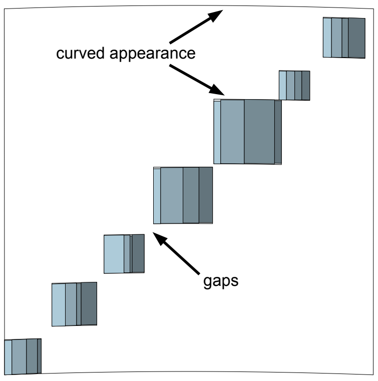

(c) Resulting Design

Figure 13: Example of recreating information slices showing the original (a), a first approximation (b) picked from the palette shown in Figure 9, and the final result (c) after shifting the presets on the palette around to decrease the influence of designs with unwanted visual properties.



(a) Staggered Stacked Bar Chart



(b) First Approximation

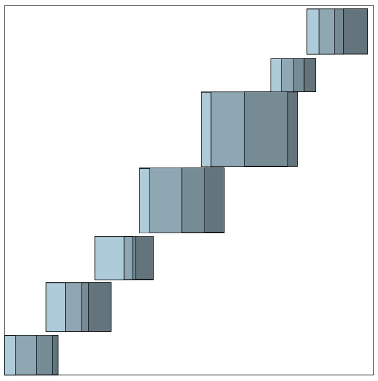

(c) Resulting Design

Figure 14: Example of recreating a staggered stacked bar chart (image source: `http://peltiertech.com`) showing the original (a), a first approximation (b) picked from the palette shown in Figure 9, and the final result (c) after fine-tuning the influence of the presets in the mixer interface.

design spaces alone, but require our more continuous interpolation between them.

The first example aims to create a tree visualization that is similar to the information slice [54], which is shown in Figure 13a. A good starting point is to pick a similar looking visualization from the visualization palette (cf. Figure 9). From our discussion in Section 2.4, it is obvious that such a first starting point would be positioned somewhere between the icicle plot and the sunburst visualization – for example, the position marked by the red cursor in Figure 9. The resulting first approximation of an information slice is shown in Figure 13b. Its appearance still has a few flaws that we need to eliminate: there is a small displacement of the rings and there are some gaps in between nodes. The displacement is due to the influence of the cascaded treemap on the picked palette position and it can easily be removed by dragging the cascaded treemap preset away from the picked location. The gaps are caused by the influence of the two node-link layouts, which prevent the design from being completely space-filling. Yet, these presets cannot be moved further away, as they are already positioned in the corners of the palette. As an alternative, we move the two visualizations that have desired properties (i.e., the icicle plot and the sunburst) closer to the position of the picked design to increase their influence and to marginalize the influence of the node-link layouts. The visualization prototype resulting of these adaptations is shown in Figure 13c. It is quite close to the original, apart from a blank area in the middle of the diagram that is due to our particular curvature interpolation.

The second example aims to create a product plot that is similar to a staggered stacked bar chart, which is shown in Figure 14a. Furthermore, we can see from the Hasse diagram in Figure 4, that such a chart cannot be generated by simply applying the design space with its discrete design choices. To generate a blend of visualizations that is close to the desired result, we pick a similar looking visualization from the visualization palette (cf. Figure 12). This first approximation of such a staggered stacked bar chart picked from the center of the palette is shown in Figure 14b. Yet, it exhibits a slightly curved appearance and also wastes drawing space with overly large gaps in between the tier-1 geometric primitives. There is not much that we can do about these remaining visual glitches in the palette, as for example all the radial presets that are responsible for the curved appearance, are already positioned in the corners and thus as far away as possible. That is why we switch to the mixer interface, in order to adjust the influence of the presets individually and without being constrained by the 2D topology of the palette. In the mixer, we reduce the influence of the radial plots to 0.0 to straighten the plot and we balance the remaining presets so that the gaps in between the primitives become smaller. The result of this fine-tuning is shown in Figure 14c. Its appearance differs from the original mainly by having different heights for each tier-1 primitive, instead of bars of equal height. This is a limitation of our design space, as it does not offer the option of not encoding a value (at least partially) onto the height, as the tree visualization design space does with design dimension (TII). How to include such additional design dimensions to a parametric configuration is shortly sketched in Section 6.2.

5.2. User Study

We conducted a qualitative user study with 8 participants (6 male and 2 female) to validate our approach and the interfaces we built on top of it. The participants were between 25 and 36 years old with an average age of 30. They all had a background in computer science with different levels of exposure to visualization in the past. We conducted a separate study of about 30 minutes with each participant during which they were encouraged to think aloud.

In a *training phase*, the participants were first given a short verbal introduction into the data and the interfaces by using two simple examples – a stacked bar chart for the product plots and an axis-parallel tree layout for the hierarchy visualization. Afterwards, the participants had time to familiarize themselves with the interfaces and the various presets by freely browsing the space of possible visualization designs on their own. During this phase, we paid attention to principle hurdles that might occur when first learning and using our interfaces. In a second phase, the *execution phase*, the participants were asked to fulfill two visualization design tasks using our interfaces. These tasks basically comprised of recreating the two visualization examples given in the previous section (information slice and staggered bar chart) from a printout that was handed to them. During this phase, we mainly observed the strategy the participants followed when using the palette and the mixer to recreate the visualization, as well as how long it took them.

During the training phase, it appeared that the palette interface was the most liked and it seemed to be intuitively understood. Its possibility to transition between visualizations in a smooth way was well perceived by all participants. Yet, this possibility was not apparent to them in the beginning, as the gridded appearance suggested only a picking of the shown intermediate designs. The participants also stated that it helped them in understanding those presets they have not known before, as for example, smoothly transitioning from a stacked bar chart to a mosaic plot and from there to a stacked pie chart helped them to intuitively grasp the idea behind them. This use of our interface for such "self-teaching purposes" was not anticipated by us and certainly points in an interesting research direction, which is underlined by a recent study on teaching visualization by analogy [70].

Only one participant identified right from the start that the palette only allows a limited number of design combinations and favored the mixer for its higher degree of freedom. Though, most participants complained that the mixer with its eight sliders takes considerably more effort in terms of interaction cost to gear towards a desired visual representation. They furthermore remarked that the mixer's lack of previews for intermediate slider positions made it hard to anticipate their effects and lead to quite a bit of trial and error in its use, which made it overall less favorable than the WYSIWYG-style of the palette. Yet this is certainly a point that can be fixed to some degree in future incarnations of this interface by utilizing sliders that provide better previews – for example, scented sliders [71].

By applying the two interfaces to the two tasks in the execution phase, all participants were able to recreate both visualizations. Yet they followed two different strategies. The first strategy used by 5 of the participants included both interfaces. They started with the palette to pick a visualization that looked reasonably similar to the printout and then switched to the mixer for fine-tuning the result. This is basically the same strategy as we have employed it in the previous section. The remaining 3 participants ignored the palette and approached the visual design from a more analytical point of view. They tried to identify design characteristics of the given examples and to rebuild them by directly mixing the presets that embody these characteristics. While they were also able to recreate the visualizations in this way, it took them considerably longer (average completion time around 5 min) than the other group who followed the first strategy (average completion time around 1 min). This finding confirms our initial assessment that the mixer is geared more towards the expert user, but at the same time, it underlines the importance of combining the two interfaces to support blended visualization design.

6. Extending Parametric Visualization Configurations

The previous sections have outlined our basic approach for parametric visualization configurations. To the end of utilizing them for preset-based visualization design, one may soon find the chosen visualization granularity (i.e., the selected presets or the used design dimensions) or even the entire visualization domain (i.e., the type of visualization it captures) to be too restricting to express one's design ambitions. This section gives examples of how to go beyond them by adding new presets, including further design dimensions, and combining configurations for different visualization types.

6.1. Additional Presets

The effectiveness of the preset-based design approach stands in direct relation to a suitable set of presets from which to blend the final visualization design. Unfortunately, suitability lies in the eye of the user and the presets can hardly be predefined without either restricting the user by providing a small set that is guaranteed to lack just the one design he needed, or overwhelming him with too many presets. The solution to this problem is to allow the user to define and add presets himself. To do so, we provide him with a direct interface to the parametric configuration, which we call the *visualization tuner* and which is shown in Figure 15.

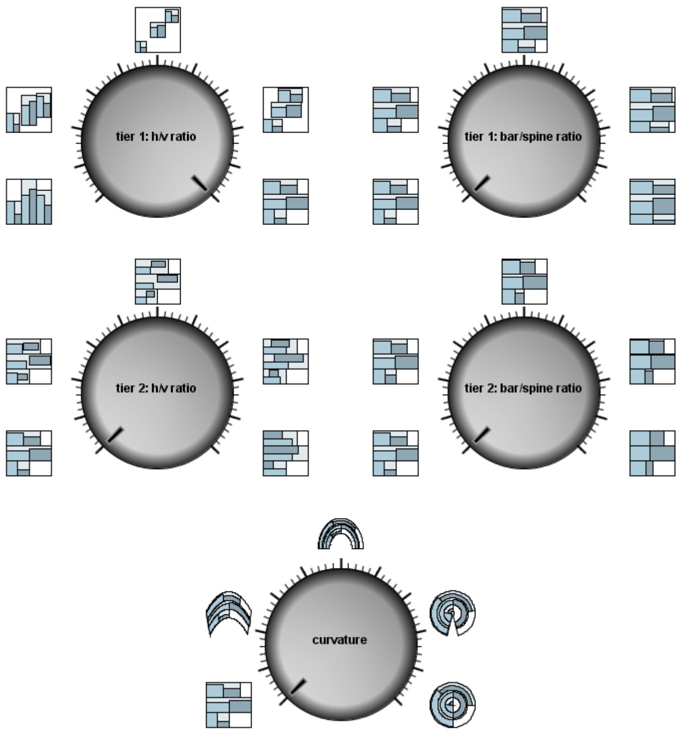


Figure 15: The *visualization tuner* gives direct access to the parameter space of the visualization design. It features one turn-dial per design dimension and shows previews of how the visualization would look like if the dial was turned to 0%, 25%, 50%, 75%, and 100% similar to the previews on the visualization mixer. It allows to construct new presets by adjusting its parameters directly.

This interface shows one turn-dial for each design dimension and permits the user to generate new presets by adjusting the design parameters individually and directly. Previews alongside the dials give an indication how the resulting design will look like if the dial was turned in their directions. As one dial is turned, the previews alongside the other dials adapt accordingly. Besides defining new presets, this interface can also be used to explore the parameter space of the parametric visualization configuration directly, for example, to investigate the parameters' influence on the quality metrics.

6.2. *Further Design Dimensions*

In the examples given in the previous sections, we have chosen five design dimensions to capture a design space of 32 visualization designs, between which we then interpolated. Yet this number is not a constant and there can be design spaces with more or less than five dimensions, as long as they remain independent of each other, which can be challenging to insure. In this section, we show how to extend the tree visualization parameter space by including additional design dimensions for dimensionality and aspect ratio, which broaden the number of possible designs significantly.

Dimensionality is a common design dimension [44] that we fixed to 2-dimensional representations only for keeping the previous discussions brief and the static printed figures easily understandable. If one wanted to add dimensionality as another dimension, one would also have to break up the binary 2D vs. 3D distinction and find a smooth transition between them. We can do this by gradually extruding a 2-dimensional visualization into the 3rd dimension, effectively generating 2.5D visualizations as intermediate designs. An example transforming a 2D treemap into a steptree [72] with 2.5D treemaps in between is shown in Figure 16 (top). Note that such an extrusion into the third dimension may require additional constraints to be met, such as maintaining a possible mapping of a numerical attribute onto the volume of a node – even though such a mapping may not be advisable from a perception point of view.

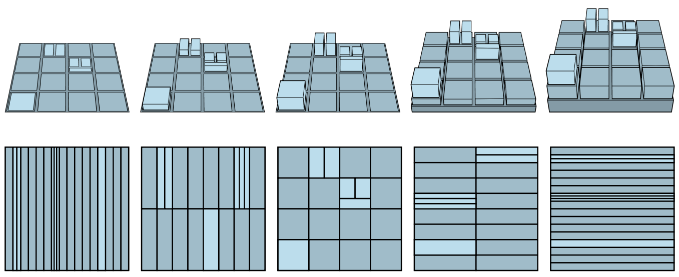


Figure 16: Two examples of additional parameters that can be used to further customize tree visualizations: dimensionality (top) and aspect ratio (bottom).

Aspect ratio has so far been neglected in the treemap layouts generated by our parameter space, which effectively only slice the space horizontally and thus generate areas of unfavorably low aspect ratios approaching 0 as the number of leaves increases. The other extreme would be to only slice the space vertically and thus to generate areas of equally unfavorable aspect ratios approaching $+\infty$ with an increasing number of leaves. Most treemap layouts aim to achieve an aspect ratio of around 1 as an optimum in between these two extremes [73]. By adding the desired aspect ratio as an additional design dimension, we would be able to steer the area subdivision of the layout algorithm. This can be achieved by mapping the different design choices (i.e., aspect ratios) to the so called *chunk score* [43] of the layout, which would then become another parameter of the visualization. An example transforming a vertically sliced treemap via a strip treemap layout [74] with aspect ratios close to 1 into a horizontally sliced treemap is given in Figure 16 (bottom). This effectively varies the number of horizontal strips in the strip treemap layout shown in the figure from 1 to 2, 4, 8, and 16 – the latter then forming a single vertical strip.

6.3. *Parametric Configuration Combinations*

Particular data characteristics require particular visualizations. Specifically when a dataset exhibits traits of two (or more) data classes that are usually visualized separately and differently, it is not uncommon that multiple kinds of visualizations are merged. One way of merging visualizations is by embedding them into each other [9]. For our two examples of distributions and hierarchies, this approach has been published for embedding product plots (column charts) into tree visualizations (treemaps) [75, 76, 77] and vice versa [78]. Figure 17 shows one example of each, as well as our rendition of both that we obtained by combining the two parametric configurations introduced in Section 3.

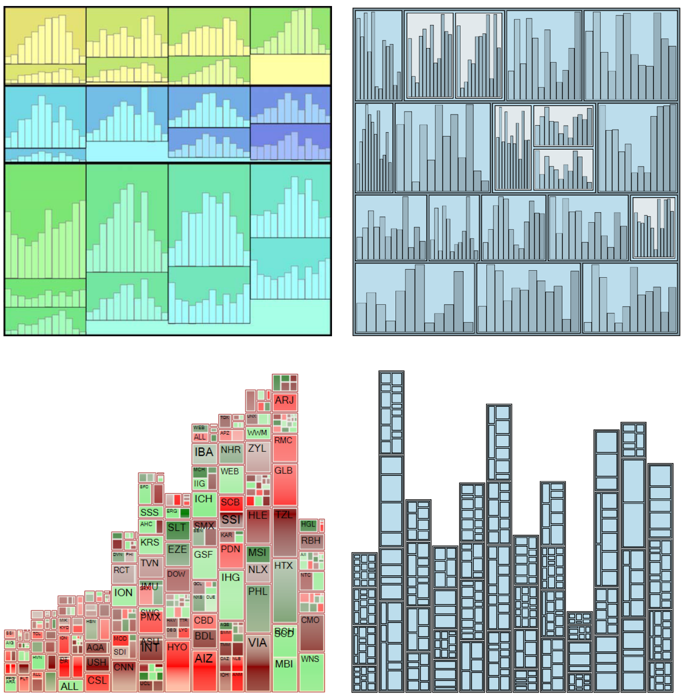


Figure 17: Examples of the two principal ways of combining product plots and tree visualizations via embedding: integrating column charts into a treemap [77] and treemaps into a column chart [78].

7. Conclusion

In conclusion, we can state that the use of continuous parameter spaces to describe entire classes of visualizations is a powerful, extendable, and combinable model to unify existing visualizations under a common hood and to develop new visualization blends by example. Together with appropriate user interfaces, such as the palette and the mixer, it enables rapid prototyping of custom visualization designs. It is this combination of continuously deformable visualization designs and user interfaces that give direct access to them, which gives our approach a unique position among existing approaches. These focus either on creating visualizations through continuous deformation (e.g., transmogrification [61] or visualization morphing [70]) or on interfaces that allow to choose visualizations in a WYSIWYG manner (e.g., visualization spreadsheets [51] or palette-style visualization [49]). By filling the gap in between the two, we gain additional flexibility in the visualization design. This flexibility allows for freely exploring the design space beyond the confines of a few commonly used, distinct design choices. To balance the additional flexibility provided by our approach, we provide feedback on the quality of the created visualization design in the form of starglyphs.

The responses from the participants of the user study were highly encouraging. It motivates us on the one hand to extend our approach to other kinds of visualizations and to further integrate them with each other. Among possible candidates that can be recreated in this way, we see visualizations of flexible linked axes [79] and data-linear visualizations [80], which have already been formally described, so that a design space can be derived and subsequently be utilized for their parametric description. On the other hand, we strive to open up additional application scenarios that can benefit from our approach. Recently published research on using animations between different view configuration for data exploration [81] and deformable visualization designs for teaching purposes [70] point in promising directions for such scenarios.

Acknowledgements

The authors wish to thank the participants of the user study for their time and effort, as well as the anonymous reviewers for their constructive feedback on earlier versions of this manuscript. This research has been funded by the German Research Foundation (DFG) and the federal state of Mecklenburg-Vorpommern and EFRE within the project "Basic and Applied Research in Interactive Document Engineering and Maritime Graphics".

Bibliography

[1] McKenna S, Mazur D, Agutter J, Meyer M. Design activity framework for visualization design. IEEE Transactions on Visualization and Computer Graphics 2014;20(12):2191–200. doi:10.1109/TVCG.2014.2346331.

[2] Liu B, Wünsche B, Ropinski T. Visualization by example – a constructive visual component-based interface for direct volume rendering. In: Richard P, Braz J, Hilton A, editors. GRAPP'10: Proceedings of the International Conference on Computer Graphics Theory and Applications. INSTICC Press. ISBN 9789896740269; 2010, p. 254–9. URL: http://viscg.uni-muenster.de/publications/2010/LWR10/grapp10-tf.pdf.

[3] Scheidegger CE, Vo HT, Koop D, Freire J, Silva CT. Querying and creating visualizations by analogy. IEEE Transactions on Visualization and Computer Graphics 2007;13(6):1560–7. doi:10.1109/TVCG.2007.70584.

[4] Heer J, Shneiderman B. Interactive dynamics for visual analysis. Communications of the ACM 2012;55(4):45–54. doi:10.1145/2133806.2133821.

[5] Groth P, Shamma DA. Spinning data: Remixing live data like a music DJ. In: CHI'13: Extended abstracts of the SIGCHI conference on Human Factors in Computing Systems. ACM Press. ISBN 9781450319522; 2013, p. 3063–6. doi:10.1145/2468356.2479611.

[6] Crider M, Bergner S, Smyth TN, Möller T, Tory MK, Kirkpatrick AE, et al. A mixing board interface for graphics and visualization applications. In: Healey CG, Lank E, editors. GI'07: Proceedings of the Graphics Interface Conference. Canadian Information Processing Society. ISBN 9781568813370; 2007, p. 87–94. doi:10.1145/1268517.1268534.

[7] Hartmann B, Yu L, Allison A, Yang Y, Klemmer SR. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In: UIST'08: Proceedings of the Annual ACM Symposium on User Interface Software and Technology. ACM Press. ISBN 9781595939753; 2008, p. 91–100. doi:10.1145/1449715.1449732.

[8] Santos E, Lins L, Ahrens JP, Freire J, Silva CT. VisMashup: Streamlining the creation of custom visualization applications. IEEE Transactions on Visualization and Computer Graphics 2009;15(6):1539–46. doi:10.1109/tvcg.2009.195.

[9] Hadlak S, Schulz HJ, Schumann H. In situ exploration of large dynamic networks. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):2334–43. doi:10.1109/TVCG.2011.213.

[10] van Wijk JJ, Overveld CWAM. Preset based interaction with high dimensional parameter spaces. In: Post FH, Nielson GM, Bonneau GP, editors. Data Visualization: The State of the Art. Springer; 2003, p. 391–406. doi:10.1007/978-1-4615-1177-9_27.

[11] Schulz HJ. Treevis.net: A tree visualization reference. IEEE Computer Graphics and Applications 2011;31(6):11–5. doi:10.1109/MCG.2011.103.

[12] Wickham H, Hofmann H. Product plots. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):2223–30. doi:10.1109/TVCG.2011.227.
[13] Kruskal JB, Landwehr JM. Icicle plot: Better displays for hierarchical clustering. The American Statistician 1983;37(2):162–8. doi:10.2307/2685881.
[14] O'Donnell R, Dix A, Ball LJ. Exploring the PieTree for representing numerical hierarchical data. In: Bryan-Kinns N, Blandford A, Curzon P, Nigay L, editors. People and Computers XX – Engage: Proceedings of the Human Computer Interaction 2006. Springer. ISBN 9781846285882; 2006, p. 239–54. doi:10.1007/978-1-84628-664-3_18.
[15] Lü HR, Fogarty J. Cascaded Treemaps: Examining the visibility and stability of structure in Treemaps. In: Bartram L, Shaw C, editors. GI'08: Proceedings of the Graphics Interface Conference. Canadian Information Processing Society. ISBN 9781568814230; 2008, p. 259–66. URL: http://portal.acm.org/citation.cfm?doid=1375714.1375758.
[16] Stasko J, Zhang E. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In: Mackinlay JD, Roth SF, Keim DA, editors. InfoVis'00: Proceedings of the IEEE Symposium on Information Visualization. IEEE Computer Society. ISBN 0769508049; 2000, p. 57–65. doi:10.1109/INFVIS.2000.885091.
[17] Grammel L, Bennett C, Tory M, Storey MA. A survey of visualization construction user interfaces. In: Hlawitschka M, Weinkauf T, editors. EuroVis'13: Short Paper Proceedings of the Eurographics/IEEE Symposium on Visualization. Eurographics Association. ISBN 9783905673999; 2013, p. 19–23. doi:10.2312/PE.EuroVisShort.EuroVisShort2013.019-023.
[18] Harris RL. Information Graphics: A Comprehensive Illustrated Reference. Oxford University Press; 1996. ISBN 0964692503.
[19] Keller PR, Keller MM. Visual Cues: Practical Data Visualization. IEEE Computer Society; 1993. ISBN 0818631023.
[20] Aigner W, Miksch S, Schumann H, Tominski C. Visualization of Time-Oriented Data. Springer; 2011. ISBN 0857290789. doi:10.1007/978-0-85729-079-3.
[21] Figueiras A. A typology for data visualization on the web. In: Banissi E, Azzag H, Bannatyne MWM, Bertschi S, Bouali F, Burkhard R, et al., editors. IV'13: Proceedings of the International Conference on Information Visualisation. IEEE Computer Society. ISBN 9780769550497; 2013, p. 351–8. doi:10.1109/IV.2013.45.
[22] Smith KB. Typologies, taxonomies, and the benefits of policy classification. Policy Studies Journal 2002;30(3):379–95. doi:10.1111/j.1541-0072.2002.tb02153.x.
[23] Wilkinson L. The Grammar of Graphics. Statistics and Computing; 2nd ed.; Springer; 2005. ISBN 0387245448. doi:10.1007/0-387-28695-0.
[24] Mackinlay J. Automating the design of graphical presentations of relational information. ACM Transactions on Graphics 1986;5(2):110–41. doi:10.1145/22949.22950.
[25] Heer J, Bostock M. Declarative language design for interactive visualization. IEEE Transactions on Visualization and Computer Graphics 2010;16(6):1149–56. doi:10.1109/TVCG.2010.144.
[26] Bostock M, Ogievetsky V, Heer J. D3: Data-driven documents. IEEE Transactions on Visualization and Computer Graphics 2011;17(12):2301–9. doi:10.1109/TVCG.2011.185.
[27] Stolte C, Tang D, Hanrahan P. Polaris: A system for query, analysis, and visualization of multidimensional databases. Communications of the ACM 2008;51(11):75–84. doi:10.1145/1400214.1400234.
[28] Mackinlay J, Hanrahan P, Stolte C. Show me: Automatic presentation for visual analysis. IEEE Transactions on Visualization and Computer Graphics 2007;13(6):1137–44. doi:10.1109/TVCG.2007.70594.
[29] Slingsby A, Dykes J, Wood J. Configuring hierarchical layouts to address research questions. IEEE Transactions on Visualization and Computer Graphics 2009;15(6):977–84. doi:10.1109/TVCG.2009.128.
[30] Stolper CD, Kahng M, Lin Z, Foerster F, Goel A, Stasko J, et al. GLO-STIX: Graph-Level Operations for Specifying Techniques and Interactive eXploration. IEEE Transactions on Visualization and Computer Graphics 2014;20(12):2320–8. doi:10.1109/TVCG.2014.2346444.
[31] Schulz HJ, Akbar Z, Maurer F. A generative layout approach for rooted tree drawings. In: Carpendale S, Chen W, Hong S, editors. PacificVis'13: Proceedings of the IEEE Pacific Visualization Symposium. IEEE Computer Society. ISBN 9781467347983; 2013, p. 225–32. doi:10.1109/PacificVis.2013.6596149.
[32] Smeltzer K. A language for visualization variation and transformation. In: Fleming SD, Fish A, Scaffidi C, editors. VL/HCC'14: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing. IEEE Computer Society. ISBN 9781479940356; 2014, p. 195–6. doi:10.1109/VLHCC.2014.6883052.
[33] Myers BA, Goldstein J, Goldberg MA. Creating charts by demonstration. In: Adelson B, Dumais ST, Olson JS, editors. CHI'94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press. ISBN 0897916506; 1994, p. 106–11. doi:10.1145/191666.191715.
[34] Roth SF, Kolojejchick J, Mattis J, Goldstein J. Interactive graphic design using automatic presentation knowledge. In: Adelson B, Dumais ST, Olson JS, editors. CHI'94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM Press. ISBN 0897916506; 1994, p. 112–7. doi:10.1145/191666.191719.
[35] Satyanarayan A, Heer J. Lyra: An interactive visualization design environment. Computer Graphics Forum 2014;33(3):351–60. doi:10.1111/cgf.12391.
[36] Satyanarayan A, Wongsuphasawat K, Heer J. Declarative interaction design for data visualization. In: UIST'14: Proceedings of the Annual ACM Symposium on User Interface Software and Technology. ACM Press. ISBN 9781450330695; 2014, p. 669–78. doi:10.1145/2642918.2647360.
[37] Chao WO, Munzner T, van de Panne M. Rapid pen-centric authoring of improvisational visualizations with NapkinVis. In: InfoVis'10: Poster at the IEEE Conference on Information Visualization. 2010,URL: http://www.cs.ubc.ca/~wochao/napkinvis/.
[38] Bostock M, Heer J. Protovis: A graphical toolkit for visualization. IEEE Transactions on Visualization and Computer Graphics 2009;15(6):1121–8. doi:10.1109/TVCG.2009.174.
[39] Kazman R, Carriére J. Rapid prototyping of information visualizations using VANISH. In: Gershon ND, Card S, Eick SG, editors. InfoVis'96: Proceedings of the IEEE Symposium on Information Visualization. IEEE Computer Society. ISBN 081867668X; 1996, p. 21–8. doi:10.1109/INFVIS.1996.559212.
[40] Fujishiro I, Furuhata R, Ichikawa Y, Takeshima Y. GADGET/IV: A taxonomic approach to semi-automatic design of information visualization applications using modular visualization environment. In: Mackinlay JD, Roth SF, Keim DA, editors. InfoVis'00: Proceedings of the IEEE Symposium on Information Visualization. IEEE Computer Society. ISBN 0769508049; 2000, p. 77–83. doi:10.1109/INFVIS.2000.885093.
[41] Ren D, Höllerer T, Yuan X. iVisDesigner: Expressive interactive design of information visualizations. IEEE Transactions on Visualization and Computer Graphics 2014;20(12):2092–101. doi:10.1109/TVCG.2014.2346291.
[42] Koop D, Scheidegger CE, Callahan SP, Vo HT, Freire J, Silva CT. VisComplete: Automating suggestions for visualization pipelines. IEEE Transactions on Visualization and Computer Graphics 2008;14(6):1691–8. doi:10.1109/TVCG.2008.174.
[43] Baudel T, Broeksema B. Capturing the design space of sequential space-filling layouts. IEEE Transactions on Visualization and Computer Graphics 2012;18(12):2593–602. doi:10.1109/TVCG.2012.205.
[44] Schulz HJ, Hadlak S, Schumann H. The design space of implicit hierarchy visualization: A survey. IEEE Transactions on Visualization and Computer Graphics 2011;17(4):393–411. doi:10.1109/TVCG.2010.79.
[45] Viau C, McGuffin MJ. ConnectedCharts: Explicit visualization of relationships between data graphics. Computer Graphics Forum 2012;31(3pt4):1285–94. doi:10.1111/j.1467-8659.2012.03121.x.
[46] Card SK, Mackinlay J. The structure of the information visualization design space. In: Dill J, Gershon ND, editors. InfoVis'97: Proceedings of the IEEE Symposium on Information Visualization. IEEE Computer Society. ISBN 0818681896; 1997, p. 92–9. doi:10.1109/INFVIS.1997.636792.
[47] Marks J, Andalman B, Beardsley P, Freeman W, Gibson S, Hodgins J, et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In: Whitted T, editor. ACM SIGGRAPH'97: Proceedings of the International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley. ISBN 0897918967; 1997, p. 389–400. doi:10.1145/258734.258887.
[48] Andalman B, Ryall K, Ruml W, Marks J, Shieber S. Design gallery

browsers based on 2D and 3D graph drawing. In: DiBattista G, editor. GD'97: Proceedings of the International Symposium on Graph Drawing; vol. 1353 of *Lecture Notes in Computer Science*. Springer. ISBN 3540639381; 1997, p. 322–9. doi:10.1007/3-540-63938-1_76.

[49] Wu Y, Xu A, Chan MY, Qu H, Guo P. Palette-style volume visualization. In: Hege HC, Machiraju R, editors. VG'07: Proceedings of the Conference on Visualization and Data Analysis. Eurographics Association. ISBN 9783905674033; 2007, p. 33–40. doi:10.2312/VG/VG07/033-040.

[50] Biedl T, Marks J, Ryall K, Whitesides S. Graph multidrawing: Finding nice drawings without defining nice. In: Whitesides SH, editor. GD'98: Proceedings of the International Symposium on Graph Drawing; vol. 1547 of *Lecture Notes in Computer Science*. Springer. ISBN 9783540654735; 1998, p. 347–55. doi:10.1007/3-540-37623-2_26.

[51] Jankun-Kelly T, Ma KL. Visualization exploration and encapsulation via a spreadsheet-like interface. IEEE Transactions on Visualization and Computer Graphics 2001;7(3):275–87. doi:10.1109/2945.942695.

[52] Shapira L, Shamir A, Cohen-Or D. Image appearance exploration by model-based navigation. Computer Graphics Forum 2009;28(2):629–38. doi:10.1111/j.1467-8659.2009.01403.x.

[53] Ma KL. Image graphs – a novel approach to visual data exploration. In: Ebert D, Gross M, Hamann B, editors. Visualization'99: Proceedings of the IEEE Conference on Visualization. IEEE Computer Society. ISBN 078035897X; 1999, p. 81–8. doi:10.1109/visual.1999.809871.

[54] Andrews K, Heidegger H. Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs. In: Wills G, Dill J, editors. InfoVis'98: Proceedings of the IEEE Symposium on Information Visualization. IEEE Computer Society. ISBN 0818690933; 1998, p. 9–12. URL: http://www.iicm.tugraz.at/liberation/iicm_papers/ivis98.pdf; late Breaking Hot Topic Paper.

[55] Chuah MC. Dynamic aggregation with circular visual designs. In: Wills G, Dill J, editors. InfoVis'98: Proceedings of the IEEE Symposium on Information Visualization. IEEE Computer Society. ISBN 0818690933; 1998, p. 35–43. doi:10.1109/INFVIS.1998.729557.

[56] Kerracher N, Kennedy J, Chalmers K. The design space of temporal graph visualisation. In: Elmqvist N, Hlawitschka M, Kennedy J, editors. EuroVis'14: Short Paper Proceedings of the Eurographics/IEEE Symposium on Visualization. Eurographics Association; 2014, p. 7–11. doi:10.2312/eurovisshort.20141149.

[57] Javed W, Elmqvist N. Exploring the design space of composite visualization. In: Hauser H, Kobourov S, Qu H, editors. PacificVis'12: Proceedings of the IEEE Pacific Visualization Symposium. IEEE Computer Society. ISBN 9781467308649; 2012, p. 1–8. doi:10.1109/PacificVis.2012.6183556.

[58] Johnson B, Shneiderman B. Tree-Maps: A space-filling approach to the visualization of hierarchical information structures. In: Nielson GM, Rosenblum L, editors. Visualization'91: Proceedings of the IEEE Conference on Visualization. IEEE Computer Society. ISBN 0818622458; 1991, p. 284–91. doi:10.1109/VISUAL.1991.175815.

[59] Bruls M, Huizing K, van Wijk J. Squarified Treemaps. In: de Leeuw W, van Liere R, editors. VisSym'00: Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization. Eurographics Association. ISBN 3211835156; 2000, p. 33–42. URL: http://diglib.eg.org/EG/DL/WS/VisSym/VisSym00/033-042.pdf.

[60] Li S, Crouser RJ, Griffin G, Gramazio C, Schulz HJ, Childs H, et al. Exploring hierarchical visualization designs using phylogenetic trees. In: Kao DL, Hao MC, Livingston MA, Wischgoll T, editors. VDA'15: Proceedings of the Conference on Visualization and Data Analysis. SPIE; 2015, p. 939709:1–14. doi:10.1117/12.2078857.

[61] Brosz J, Nacenta MA, Pusch R, Carpendale S, Hurter C. Transmogrification: Casual manipulation of visualizations. In: UIST'13: Proceedings of the Annual ACM Symposium on User Interface Software and Technology. ACM Press. ISBN 9781450322683; 2013, p. 97–106. doi:10.1145/2501988.2502046.

[62] Heer J, Robertson GG. Animated transitions in statistical data graphics. IEEE Transactions on Visualization and Computer Graphics 2007;13(6):1240–7. doi:10.1109/TVCG.2007.70539.

[63] Bhagavatula S, Rheingans P, des Jardins M. Discovering high-level parameters for visualization design. In: Brodlie K, Duke D, Joy KI, editors. EuroVis'05: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization. Eurographics Association. ISBN 3905673193; 2005, p. 255–62. doi:10.2312/VisSym/EuroVis05/255-262.

[64] Lindow N, Baum D, Hege HC. Perceptually linear parameter variations. Computer Graphics Forum 2012;31(2pt3):535–44. doi:10.1111/j.1467-8659.2012.03054.x.

[65] Dragicevic P, Bezerianos A, Javed W, Elmqvist N, Fekete JD. Temporal distortion for animated transitions. In: CHI'11: Proceedings of the International Conference on Human Factors in Computing Systems. ACM Press. ISBN 9781450302289; 2011, p. 2009–18. doi:10.1145/1978942.1979233.

[66] Beham M, Herzner W, Gröller E, Kehrer J. Cupid: Cluster-based exploration of geometry generators with parallel coordinates and radial trees. IEEE Transactions on Visualization and Computer Graphics 2014;20(12):1693–702. doi:10.1109/TVCG.2014.2346626.

[67] Bruckner S, Möller T. Result-driven exploration of simulation parameter spaces for visual effects design. IEEE Transactions on Visualization and Computer Graphics 2010;16(6):1468–76. doi:10.1109/TVCG.2010.190.

[68] Frank AU, Timpf S. Multiple representations for cartographic objects in a multi-scale tree – an intelligent graphical zoom. Computers & Graphics 1994;18(6):823–9. doi:10.1016/0097-8493(94)90008-6.

[69] Ellis G, Dix A. The plot, the clutter, the sampling and its lens: occlusion measures for automatic clutter reduction. In: Celentano A, Mussio P, editors. AVI'06: Proceedings of the Working Conference on Advanced Visual Interfaces. ACM Press. ISBN 1595933530; 2006, p. 266–9. doi:10.1145/1133265.1133318.

[70] Ruchikachorn P, Mueller K. Learning visualizations by analogy: Promoting visual literacy through visualization morphing. IEEE Transactions on Visualization and Computer Graphics 2015;21(9):1028–44. doi:10.1109/TVCG.2015.2413786.

[71] Lasram A, Lefebvre S, Damez C. Scented sliders for procedural textures. In: Andujar C, Puppo E, editors. EG'12: Short Paper Proceedings of the Annual Conference of the European Association for Computer Graphics. Eurographics Association; 2012, p. 45–8. doi:10.2312/conf/EG2012/short/045-048.

[72] Bladh T, Carr DA, Scholl J. Extending tree-maps to three dimensions: A comparative study. In: Masoodian M, Jones S, Rogers B, editors. APCHI'04: Proceedings of the Asia Pacific Conference on Computer Human Interaction; vol. 3101 of *Lecture Notes in Computer Science*. Springer. ISBN 9783540223122; 2004, p. 50–9. doi:10.1007/978-3-540-27795-8_6.

[73] Kong N, Heer J, Agrawala M. Perceptual guidelines for creating rectangular treemaps. IEEE Transactions on Visualization and Computer Graphics 2010;16(6):990–8. doi:10.1109/TVCG.2010.186.

[74] Bederson BB, Shneiderman B, Wattenberg M. Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. ACM Transactions on Graphics 2002;21(4):833–54. doi:10.1145/571647.571649.

[75] Schreck T, Keim D, Mansmann F. Regular treemap layouts for visual analysis of hierarchical data. In: Slavík P, editor. SCCG'06: Proceedings of the Spring Conference on Computer Graphics. Comenius University, Bratislava. ISBN 8022321753; 2006, p. 184–91. URL: http://www.inf.uni-konstanz.de/gk/pubsys/publishedFiles/ScKeMa06.pdf.

[76] Telea A. Combining extended table lens and treemap techniques for visualizing tabular data. In: Santos BS, Ertl T, Joy KI, editors. EuroVis'06: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization. Eurographics Association. ISBN 3905673312; 2006, p. 51–8. doi:10.2312/VisSym/EuroVis06/051-058.

[77] Kobayashi A, Misue K, Tanaka J. Edge equalized treemaps. In: Banissi E, Bertschi S, Forsell C, Johansson J, Kenderdine S, Marchese FT, et al., editors. IV'12: Proceedings of the International Conference on Information Visualisation. IEEE Computer Society. ISBN 9780769547718; 2012, p. 7–12. doi:10.1109/IV.2012.12.

[78] Huang ML, Huang TH, Zhang J. TreemapBar: Visualizing additional dimensions of data in bar chart. In: Banissi E, Stuart L, Wyeld TG, Jern M, Andrienko G, Memon N, et al., editors. IV'09: Proceedings of the International Conference on Information Visualisation. IEEE Computer Society. ISBN 9780769537337; 2009, p. 98–103. doi:10.1109/IV.2009.22.

[79] Claessen JHT, van Wijk JJ. Flexible linked axes for multivariate data visualization. IEEE Transactions on Visualization and Computer Graphics

2011;17(12):2310–6. doi:10.1109/TVCG.2011.201.

[80] Baudel T. A canonical representation of data-linear visualization algorithms. arXivorg e-print service 2014;1412.4246. URL: http://arxiv.org/abs/1412.4246.

[81] Hurter C, Taylor R, Carpendale S, Telea A. Color tunneling: Interactive exploration and selection in volumetric datasets. In: Brandes U, Hagen H, Takahashi S, editors. PacificVis'14: Proceedings of the IEEE Pacific Visualization Symposium. IEEE Computer Society. ISBN 9781479928736; 2014, p. 225–32. doi:10.1109/PacificVis.2014.61.