Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups

Tom Horak
Interactive Media Lab
Technische Universität Dresden
Dresden, Germany
horakt@acm.org

Andreas Mathisen
Department of Computer Science
Aarhus University
Aarhus, Denmark
am@cs.au.dk

Clemens N. Klokmose
Digital Design & Information Studies
Aarhus University
Aarhus, Denmark
clemens@cavi.au.dk

Raimund Dachselt
Interactive Media Lab
Technische Universität Dresden
Dresden, Germany
dachselt@acm.org

Niklas Elmqvist
College of Information Studies
University of Maryland
College Park, MD, USA
elm@umd.edu

Figure 1: The Vistribute system: Based on a design space we derived six heuristics that can guide an automatic distribution of visualizations in changing device setups, e.g., (a) dual desktop, (b) laptop and large display, or (c) mobile device ensemble.

ABSTRACT

We present *Vistribute*, a framework for the automatic distribution of visualizations and UI components across multiple heterogeneous devices. Our framework consists of three parts: (i) a design space considering properties and relationships of interactive visualizations, devices, and user preferences in multi-display environments; (ii) specific heuristics incorporating these dimensions for guiding the distribution for a given interface and device ensemble; and (iii) a web-based implementation instantiating these heuristics to automatically generate a distribution as well as providing interaction mechanisms for user-defined adaptations. In contrast to existing UI distribution systems, we are able to infer all required information by analyzing the visualizations and devices without relying on additional input provided by users or programmers. In a qualitative study, we let experts create their own distributions and rate both other manual distributions and our automatic ones. We found that all distributions provided comparable quality, hence validating our framework.

CCS CONCEPTS

• **Human-centered computing** → **User interface management systems**; *Visualization systems and tools*; Visualization theory, concepts and paradigms.

KEYWORDS

Distributed user interfaces, infovis, cross-device visualization, cross-device interaction, multi-display environments.

ACM Reference Format:
Tom Horak, Andreas Mathisen, Clemens N. Klokmose, Raimund Dachselt, and Niklas Elmqvist. 2019. Vistribute: Distributing Interactive Visualizations in Dynamic Multi-Device Setups. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland, UK*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3290605.3300846

CHI 2019, May 4–9, 2019, Glasgow, Scotland, UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300846

1 INTRODUCTION

Advances in mobile computing have spawned a ubiquity of networked digital devices in our everyday lives [9]. Such devices are increasingly liberating office workers from the

bonds of their desks, allowing tasks to be distributed across the day, continued in different contexts with different device setups, and performed with an ever-changing constellation of participants [18, 28, 54]. Data analysis and sensemaking is no different, but current practice rarely exploits the full potential of cross-device interaction, instead merely using additional devices to increase screen real estate [2, 3] or improve visibility in multi-user settings [12]. Fully utilizing these ad-hoc multi-device environments would enable analysts to seamlessly continue their data exploration throughout an entire day across a plethora of devices, settings, and people [19]. For example, consider an oncologist in a hospital using patient tumor data to inform her practice (analyzing, e.g., tumor growth rate, blood levels). The doctor may spend some time on her morning commute to get up to speed (smartphone), in her office to plan treatment (desktop and tablet), continuing during a coffee break (laptop and phone) with a colleague spontaneously joining after a while (adding a tablet), then at a tumor board[1] with other doctors (large displays, laptops, and mobile devices), and finally in a treatment room consulting the patient (tablet and large TV)—all without spending time on manually setting up the interface. To our knowledge, no existing data analysis framework exists that is capable of dynamically and seamlessly adapting to such a multitude of devices, settings, and collaborators.

To address this gap, we propose the *Vistribute* framework, an automatic approach for distributing visualization interfaces across dynamic multi-device environments based on view, data, and user properties. Unlike existing automatic distribution mechanisms for general user interfaces, such as AdaM [46], Vistribute uses in-depth information about views, the data they visualize, and the tasks users want to perform on them to optimize the layout. The framework consists of a design space, a set of heuristics, and an example implementation. Our design space for cross-device visualization draws on the literature as well as an analysis of existing visualization interfaces, and explicitly considers dynamic factors such as view properties and relationships, device properties and the current device ensemble, as well as user preferences. Using this design space, we propose several heuristics as high-level constraints for distributing visualization views. Finally, our web-based implementation automatically collects information about the devices, the dataset, and the visualizations to derive a suitable distribution. In addition to the distribution itself, we enable users to adapt the interface distribution according to their needs and preferences.

In summary, our paper presents the conceptual Vistribute framework with the following contributions: (1) a *design space* identifying important properties and relations for distributed visualization interfaces; (2) six *heuristics* guiding the distribution process; (3) a web-based *implementation* as one possible instance of the heuristics; and (4) a *qualitative study* where experts manually created distributions and rated both other manual distributions and our automatic distribution.

[1]Tumor board meetings convene doctors with different specialties to discuss cancer cases, share knowledge, and plan treatment.

2 RELATED WORK

Our work is located in the intersection of human-computer interaction (HCI) and visualization research, straddling two topics specifically: (i) multi-display setups in visual data analysis and (ii) general HCI research on view distribution.

Visual Data Analysis in Multi-Display Environments

In recent years, the visualization community has intensified their efforts in investigating analysis systems and visualizations that go "beyond the desktop" [38, 53]. While incorporating many different aspects (e.g., utilized modalities, display technologies, or interaction styles), using multiple displays in parallel is often one prominent characteristic of these systems. Often, mobile devices are used as movable containers for content, settings, or preferences [16, 26, 42]. This can allow for switching between working alone and in concert [42], or having user-specific tools available on hand [26, 61]. In this context, HCI research has also suggested general cross-device interaction techniques for data transfer [14, 36, 41, 59, 61].

Other literature focuses on specific device combinations. Spindler et al. [60] as well as Kister et al. [32] used handheld displays in relation to a bigger context display (tabletop and display wall, respectively) in order to show details and alternative representations for a large visualization, while Langner et al. [35] used smartphones to enable co-located remote interaction inside a *coordinated & multiple views* (CMV, [52]) application running on a display wall. With Thaddeus, Wozniak et al. [62] applied these principles to a smaller scale by using a spatially-aware smartphone for probing data on a tablet. In VisTiles, Langner et al. [34] focussed on physical ad-hoc layouts with mobile devices (one visualization per device) for data exploration.

While the presented concepts provide useful device combinations for specific visualizations or interaction styles, it remains unclear how these can be generalized for any situation and device constellation. However, as a study by Plank et al. [47] showed, this is a crucial aspect for multi-device setups: they found that participants did not take advantage of having multiple tablets during sensemaking tasks, even with optimized tasks. As their setup had multiple restrictions (e.g., one view per device, fixed device pairs), it appears that having a proactive and flexible interface is a gatekeeper for unleashing the full potential of multi-device setups.

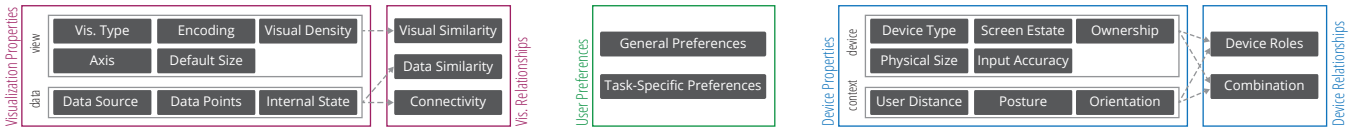

Figure 2: The design space comprises aspects coming from the visualizations, users, and devices. When considered pairwise, relationships emerge for both devices and visualizations.

A broader analysis was presented by Chung et al. [15], discussing multiple considerations for data analysis in multi-device setups. For instance, the combination of displays can support different view arrangements (e.g., continuous, CMV, separated instances). Similarly, the way how updates and states are synchronized across devices can promote different interface functionality during the analysis. However, the authors do not touch on how to exactly realize a system supporting these different aspects, and only point out the technical challenges, e.g., system-imposed constraints coming from the device's hardware or software platform.

Multi-Device Frameworks & View Distribution

Utilizing heterogeneous devices in parallel introduces multiple technical challenges how to coordinate and synchronize the devices. The HCI community has proposed several frameworks to tackle these challenges, often using web technologies as a foundation. Synchronization can happen on multiple levels: For instance, Webstrates [33] operates on the level of the Document Object Model (DOM), effectively maintaining exact copies on different devices. Other frameworks focus on the graphical aspects of an interface from a developer's perspective, e.g., when spanning one canvas across multiple devices [50, 58]. Also, functionality supporting cross-device interaction techniques can be provided [27]. Designed specifically for visualization, Badam and Elmqvist [5] and Badam et al. [6] presented technical frameworks for synchronizing user interactions as well as application states across devices. While all these frameworks are designed to ease the development of new applications, they require that programmers or users manually arrange interface components.

This gap is partly addressed by research proposing specific distribution algorithms or frameworks, most of which automatically derive a candidate distribution based on interface semantics provided by the developer, and then let the user adjust the result. Panelrama [63] introduced a lightweight specification that allows programmers to provide additional semantics for HTML elements, which are then consumed by an interface optimizer. Park et al. [46] proposed an optimizer called AdaM, which is based on a constraint solver; however, AdaM requires users or developers to provide additional semantics for each interface component, too. The XDBrowser [44, 45] segments web pages and distributes the parts across devices. In all examples, the layout is not guaranteed to be optimal, and serves rather as a starting point.

More specialized applications may allow for automatically determining dependencies between interface components and how to organize them. As a case in point, recent work by Husmann et al. [28] presented a similar system in the context of an integrated development environment, but applied automatic assignments only for a few selected view constellations. To our knowledge, such an approach has not been proposed for visualization and data analysis yet.

3 DESIGN SPACE: INTERACTIVE VISUALIZATIONS IN MULTI-DEVICE ENVIRONMENTS

The distribution and layout of views in a visualization interface are not arbitrary, but often follow certain patterns. Based on related work, considerations of existing interfaces, and our own experience in cross-device research, we aim to provide a conceptual framework that is able to reproduce these patterns when distributing and arranging views across multiple devices. The framework consists of a design space, distribution heuristics, and a prototype implementation.

In creating our framework, we were guided by multiple considerations. First of all, individual visualizations encode richer semantics compared to other user interface components [48], such as the data being visualized, the visual representation chosen, and the typical tasks supported. By considering these aspects, it is possible to automatically derive properties required for a distribution that otherwise would have to be provided by analysts, designers, or developers.

Second, these semantics also reveal relationships between multiple visualizations [48], which allows for further refining the distribution. In existing interfaces or dashboards (see, e.g., Tableau dashboards[1] or examples analyzed by Sarikaya et al. [55]) it is possible to observe such relationships, e.g., two bar charts are aligned for comparison. Similar aspects can be observed in research focusing explicitly on large displays or multi-device ensembles [34, 35], as well as for the involved devices, where their properties and relationships imply their strengths or possible roles in a distributed interface.

The design space aims to give an overview of interactive visualizations in multi-device setups, considering all relevant properties and relationships occurring (Figure 2), which we group and discuss as five dimensions in the following. At

[1] "Customer Survey Result" and "Sales Summary" from http://tableau.com.

the end, this design space will eventually provide a fundamental understanding of the incorporated dimensions. By molding this knowledge into easy-to-apply heuristics, we aim to provide a guidance for new distribution approaches (i.e., specific implementations) for interactive visualizations.

Visualization Properties

In comparison to traditional UI components, visualization views feature a rich body of properties that depends on their configuration, visual representation, or encoded data. These properties can be used to construct visualizations (as in, e.g., D3 [11], Idyll [17], Vega [56, 57]) as well as to analyze them (essentially the inverse of construction), as in our case.

First, visualizations can be characterized through properties related to their visual appearance: the actual *visualization type* (i.e., used visual marks), the applied *encoding* and mapping (i.e., visualized data dimensions), the *axis* configuration (e.g., orientation, scale, sorting), as well as the *default size* (and also implicitly the aspect ratio). Although these properties are often defined in the context of the considered data, they do not fully depend on the actual data: two views can have the same visual configuration but show disjoint data subsets. We also consider a *visual density* property, resulting from the mark size, potentially occurring overlaps, and existing additional elements (e.g., guides). This density can affect the comprehension and supported interaction; e.g., the selection of small marks is more difficult and requires a certain minimum precision (cf. Park et al. [46]).

For data-related properties, we consider the used data source, the data points themselves, as well as the internal state. The *data source* can describe only the source or the complete data flow prior to the view, i.e., from the dataset through filters or aggregation components. Depending on the visualization system, certain functionality (e.g., aggregation) can be part of the view (e.g., Vega-lite [56]) or a separate component (e.g., Vistrates [7]). Nevertheless, we consider them as pre-processing and not part of the visualization itself. The *data points* allow comparing the data of two views or analyzing the view regarding the number of visualized marks, e.g., to estimate how dense the visualization is. Finally, visualizations often maintain an *internal state*, e.g., selected marks or ranges, which can be accessed by other views.

Visualization Relationships

Typical visualization interfaces consist of multiple visualizations (often known as dashboards [55]) where the views complement each other by showing different aspects of the data and, in combination, help the user gain insights. We characterize the interplay between views as one of three relationship types: visual similarity, data similarity, and connectivity (Figure 3). These relationships yield patterns for grouping and aligning views common in existing interfaces [55].

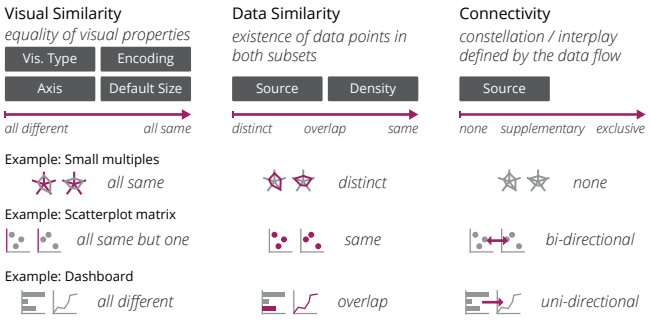

Figure 3: Three dimensions of visualization relationships based on view properties; many combinations can be useful.

Visual similarity considers how similar the two views appear, regardless of the actual encoded data points. We use the visual-appearance properties described above (e.g., type, encoding, axis configuration) to rate the consistency of two views [48]; by comparing the properties, the similarity can range from *all different* to *all same*. Similar views can support visual comparison when placed in juxtaposition [21, 22, 30, 48]. For instance, two views with the very same visual configuration is an example of *small multiples*, where the single instances differ only in the shown data. Slightly weaker relationships can be found in scatterplot matrices, where two plots differ in one dimension. In contrast, dashboards may feature multiple views that are not or partly consistent and, thus, have only a weak visual similarity [55].

The second relationship type is *data similarity*, and expresses how big the overlap between the visualized data points of two charts is. When the data is exactly the *same*, this indicates that the two views show different representations for the same data subset. A weaker similarity is a data *overlap*, and no similarity means the data is *distinct*. These constellations can indicate certain exploration pattern, e.g., overview+detail (overlap). However, in many situations, data similarity must be considered with respect to visual similarity. For instance, some combinations of the two measures are not practical, e.g., a perfect visual similarity and a perfect data similarity describes the same visualization. In conclusion, data similarity provides an indication which views are related data-wise and, thus, can provide additional insights.

Finally, views can also have a relationship with respect to the data flow, which we define as *connectivity*. This involves mechanisms such as linked brushing in multiple coordinated views [52], or incorporating a selection in one chart as a filter condition in another. We distinguish between different connectivity levels; the strongest is an *exclusive* connectivity, where a view receives its data purely from another (e.g., a filter component). Linked brushing, instead, is an example of an additional, *supplementary* connectivity; however, both

views would still be able to display data without this connectivity. Similar rankings of the connectivity can also be found in the literature; for instance, VisTiles [34] encoded this ranking by distinguishing between connections triggered by side-by-side combinations (i.e., stronger ones) and general connections (e.g., selections). Notably, the connectivity extends also to non-visualization components, e.g., UI elements for defining filters or aggregations.

User Preferences

Visualization interfaces are typically flexible and can be adapted to user preferences. We distinguish here between two types of preferences: general and task-specific preferences. *General preferences* are independent of a specific situation and derive mostly from how a user prefers to arrange things or what overall strategy for device organization he follows [23]. For instance, a user may want to keep a filter component on the right device border, or prefers to have one specific visualization on a specific device. *Task-specific preferences* emerge during the data exploration [1, 13, 64], and also affect the distribution. This can involve, e.g., aligning views for visual comparison, temporarily enlarging a visualization, or moving a view to another device to simplify interaction.

While multiple distributions of the same quality exist, they may fit analyst's preferences differently. Thus, considering these user preferences helps to improve the system's usability. However, retrieving such information automatically is challenging; instead, interfaces should provide adequate functionalities that allow users to express their preferences.

Device Properties

Devices today have a very wide spectrum of distinct characteristics, many of which have already been considered in a multitude of existing research [23, 26, 34, 45, 46, 63]. Likely the most important property is the available *screen estate*, determining how many visualizations can be displayed at what size. Since pixel density differs between devices, screen resolution should not be a sole measure as the resulting physical size is also important. Further, devices differ in the available input modalities, i.e., no input, touch, pen, mouse, or keyboard, and the resulting *input accuracy* [46] of these. The *device type* can also indicate useful information with regards to mobility or computation power. In combination with the *ownership*, this allows to distinguish between personal smartphones (mostly used by one person) or public large display (shared with multiple users) [26, 32, 42].

Besides these basic properties, further characteristics can be considered. Contextual information about the device's *posture*, *orientation*, and *user distance* (i.e., user-to-device proximity) provide insights on how the device is used by analysts. For instance, hand-held devices are more likely to be used for input. Similarly, a distant device may require scaling up views for readability reasons. Further, advanced display specifications could be considered (e.g., viewing angles, color accuracy, brightness). However, such properties are hard to access and require external sensors or knowledge.

Device Relationships

Depending on the actual device ensemble, devices can step into different relationships during the interaction. While the theoretically possible combinations are manifold, we focus here on realistic device combinations. The simplest *combination* is a two-display desktop setup, where the displays are aligned and form one big surface. However, in a scenario where a laptop is connected to a projector, these two screens act as separate units with different properties. The second case can also be applied to mobile devices (i.e., smartphones and tablets): they can be used in combination with a larger display or a desktop [26, 32, 42], as well as with multiple other mobiles [34, 47, 50, 62]. In these situations, devices differ regarding their type, size, input modality, posture, and distance, which makes it possible to assign certain *device rules* to them. For instance, smaller devices in addition to a larger device are most often suitable to host additional details and UI elements [32, 34, 42], or devices closer to the user can act as remote controls for a more distant device [26, 35, 37, 61, 62].

4 HEURISTICS FOR DERIVING A VIEW-SENSITIVE DISTRIBUTION AND LAYOUT

Our design space and its dimensions can be used to both describe and generate layout strategies for cross-device visualization. In our work, we use these dimensions to derive six heuristics for distributing components of a visualization interface across multiple devices. With these heuristics, we aim to provide comprehensible and replicable high-level constraints. We found that formal specification, such as in the AdaM framework [46], is often costly with little practical gain, and—most importantly—results in definitions that are hard to relate to. In contrast, our heuristics are prescriptive also to human designers and can be used to guide the design of manual distribution, algorithms, or even optimizers.

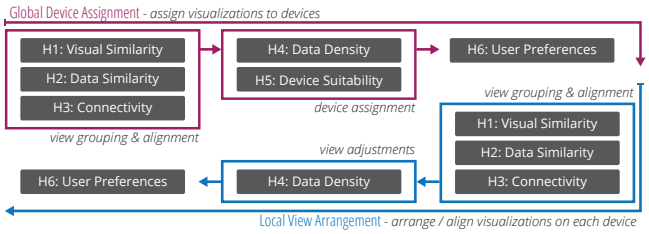

Figure 4: The heuristics are incorporated for both the global device assignment and the local view arrangement on devices; in the process, heuristics can contradict each other.

Each heuristic contributes to different aspects of a distribution, such as view grouping or device assignment, while they also allow for promoting common analysis tasks (e.g., visual similarity supports comparison tasks). Specifically, we consider the heuristics to be applied in a step-wise process (Figure 4), where a later heuristic can contradict earlier assignments. In this process, the heuristics can be detailed, weighted, and transformed into a specific quantification; our Vistribute implementation serves only as one example.

Grouping & Alignment Based on View Relationships

The relationships between visualizations can serve as indicators for which views should be grouped or aligned [48]. Therefore, we introduce three corresponding heuristics.

As pointed out above, views with a high visual similarity promote visual comparison. Based on common practice, such as in small multiple displays and scatterplot matrices, it is beneficial to place these views next to each other. Reducing the screen distance facilitates the user alternating their focus between the two views and, thus, to actually compare them. Aligning the views along a shared axis will further support comparison. Here, we utilize the visual similarity as an indicator if and how well two views are comparable. We consider a high visual similarity as the strongest type of relationship between views that motivates juxtaposing them. However, a lower visual similarity is often not of interest. We define the heuristic as follows:

Heuristic 1 (Visual Similarity)**.** *If two views are visually very similar, they should be both juxtaposed and aligned.*

The second driver for grouping is data similarity. Placing the views with a high data similarity close to each other, i.e., forming view groups, can support the search-related tasks of users [13] as well as focussing on related aspects (cf. the semantic substrate concept by Chung et al. [15]). For instance, if multiple views encode the exact same data subset and are placed next to each other, they will provide different visual representations of the same subset. Similarly, this applies to other constellations, such as overview+detail patterns (i.e., one view shows a subset of the other view). However, this relationship is not as strong as the visual-similarity-based one, and typically does not require an alignment of the views. Further, it may also depend on the type of visual similarity: for example, a subset relationship eventually represents a useful overview+detail pattern if the two views are also of the same type. As a result, this heuristic focuses on data similarity, but also incorporates visual similarity:

Heuristic 2 (Data Similarity)**.** *If two views have a high degree of data similarity and a corresponding visual similarity, they should be placed close to each other.*

As described before, views can consume data from another view and either rely on it exclusively (e.g., filter), or use it as a supplementary input (e.g., linked brushing). In the first case, the component providing the input must be accessible so that the other view can be used. Therefore, it is beneficial to place it close to the affected view, in order to emphasize their dependency. Also, and similar to visual similarity, proximity helps to reduce the cost of attention switches between the input component and the affected components. This is also true when the connection provides supplementary input. In all cases, a close proximity of the views is desirable:

Heuristic 3 (Input Connectivity)**.** *If an interface component serves as data input for others, it should be placed close to the affected components.*

As a result of these heuristics, we expect two types of view groups: (i) strong groups that result in guaranteed alignment, and (ii) weak groups that lead to view proximity, but also can be split up in case of insufficient space.

View Adjustments and Device Assignments

The next step towards the distribution is considering the single views with respect to the current device ensemble.

First, it should be identified how much space a view requires: although exceptions may exist [29], generally, the more data points a visualization encodes, the more it benefits from being scaled up [39]. For instance, a bar chart showing three bars requires less space than one with 50 bars. Similarly, a scatterplot encoding hundreds of data points should be allocated more space than one with 10 marks. While the optimal size in relation to the number of data points always depends on the visualization type, it is still a good estimation of relative space requirements. Finally, many visualizations are sensitive to changes in their aspect ratio. Therefore, scaling should be uniform or only slightly alter the aspect ratio to avoid tampering with the original perception.

Heuristic 4 (Data Density)**.** *A view should be allocated space proportional to the number of data points it encodes.*

Second, we consider the device suitability, which expresses how well a certain device can fulfill the requirements derived from a view or a group of views. These requirements mainly comprise the space requirement, input accuracy, and relations arising from the connectivity. For instance, views with a high space requirement are likely to be placed on a larger display. However, the suitability has not always an impact, i.e., when all devices are very similar, and, thus, interchangeable. For instance, when only tablets are available, it does not matter which part of the interface is distributed to which device. In contrast, with high diversity in the device ensemble, device suitability can be used for assigning different device roles (see device relationships described in design space). This can lead to exceptions of the grouping, e.g., components

serving as an input can be moved to a mobile device and act as a remote control for the larger displays. In summary, device suitability is a main constraint in diverse ensembles:

Heuristic 5 (Device Suitability)**.** *If devices are diverse, view assignments should be guided by device suitability.*

User Preferences

No matter how advanced a view distribution system is, users should be able to change the layout based on their preferences or current situation. These preferences can involve, e.g., a fixed placement of some views, an altered alignment, or even the exclusion of certain devices or components. These constraints should always be reflected in the distribution and overwrite the definitions coming from the other heuristics. Furthermore, these preferences should be stored and reapplied automatically, but must be editable by the user.

Heuristic 6 (User Preferences)**.** *If user preferences are applicable, they outweigh all other heuristics.*

In the context of analysis tasks [1, 13, 64], i.e., temporary user interests, it could be theoretically possible to infer these automatically based on user interactions. For instance, if a user makes alternating selections in two views, this can express the need to bring the views closer together. As we explicitly left room for weighting the heuristics, this allows for optimizing the distribution for the current task, e.g., emphasizing data similarity (H2) and connectivity (H3) to support investigating related items (*connect* [64]). However, too many (unexpected) interface changes must be avoided.

5 THE VISTRIBUTE SYSTEM

We implemented a web-based system[2] that is able to (i) extract required properties from visualization/UI components and connected devices, (ii) derive and apply a distribution, and (iii) allow user adaptions via a control panel. The implementation is one of many possible instances of our heuristics; for each feature, we will reference the related heuristic. Stated quantifications/values were determined empirically.

Underlying Systems and Dependencies

Our implementation builds upon three existing system layers: Webstrates, Codestrates, and Vistrates. Webstrates [33] provides the underlying synchronization (of, e.g., states, selections, device information) across devices. Besides an in-browser computing environment, Codestrates [51] provides a package management system based on Webstrates. Vistrates [7] is a visualization layer for Codestrates offering specific visualization components and a data-flow-based execution model. This combination provides common visualizations and the possibility to connect them to a data source or with each other, hence, providing all tools to create an adaptable and full-fledged visualization interface.

Our distribution layer is implemented as a Vistrates meta-package and makes use of the offered functionality of the before-mentioned layers, e.g., when accessing view properties (including states and data flow configurations). The distribution algorithms are run on one client; the resulting distribution is synchronized with all clients as a JSON object. Then, the clients move their assigned views to the given position on an interface layer. The creation of the visualizations and their connections is, however, left to the user.

Deriving Properties

The first step for the distribution is to derive all required information, i.e., visualization and device properties.

View Properties and Relationships. To extract these properties, we directly access the standardized state of the Vistrates components, e.g., template, size, data source(s), and accessed data properties. Based on the rendered view, we can distinguish between visualization and UI components. We also identify the incoming data as a basis for following steps.

The visual similarity is calculated by comparing selected properties and assigning points for matches; specifically, we consider the component template (3 pts; comprises type and encoding), dimensions (i.e., consumed data properties; 2 pts), number of data points (1 pt), and size (1 pt). By traversing the components' data source, we extract the connectivity (H3, *exclusive* or *supplementary*) and the data similarity (H2, *none* or *same*). For performance reasons, data points were not compared directly; instead, we determine the closest common source and check if the data structure changes on the way (by, e.g., aggregation). While this does not allow detecting data overlap, it provides an indication if the data structure is the same.

Device Properties. Current browsers provide access to a set of device specific properties, allowing us to characterize as well as (re)identify them. Besides common properties such as resolution, language, platform, and user agent, in many cases also hardware-specific properties (e.g., parallel threads, memory size, CPU, GPU) are available. However, some device information is missing, e.g., advanced display properties, physical size, or attached input devices (e.g., keyboard, mouse). As a result, we cannot distinguish larger displays (e.g., digital whiteboard, projector) from desktop displays, as their resolution is identical. Similarly, contextual information (e.g., user proximity, ownership) would require external sensors.

Notably, one physical device can host multiple clients (e.g., laptop with projector), where each client should be considered independently. At the same time, in some setups multiple clients must be perceived as one unit (e.g., display wall consisting of multiple displays), even if they are not hosted

[2] github.com/tomhorak21/vistribute

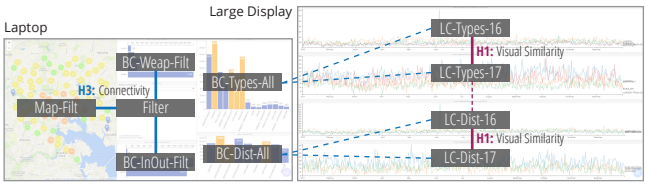

Figure 5: Example distribution illustrating H1 and H3: On the laptop, views form a block based on their connectivity to the filter (H3, *exclusive*); the line charts form two pairs based on their visual similarity (H1, *all-same*, 7 of 7 pts).

on the same device. Therefore, we introduce an abstracted representation of a device called *surface*. Each surface represents one or more clients and maps its resolution to them. For the distribution, only these surfaces are considered; except for resolution, the device's properties are inherited.

Distribution: Grouping, Assignment, and Adjustment

As described before, we consider the distribution to be a multi-step process. The first step is to identify the view groups and their types (strong and weak). To qualify as a strong group, views must have an exact visual similarity (= 7 pts; H1), while weak groups are formed based on data similarity (H2) and connectivity (H3). An example distribution is given in Figure 5. In addition to these groups, we also calculate a relative space requirement V_{SR} for each view based on the number of visualized data points (damped via log_2) and normalized so that $\sum V_{SR} = 1$ (H4). Similarly, based on the available area, we calculate a relative screen estate S_{SE} for each surface, again with $\sum S_{SE} = 1$.

Next, we identify special view-device pairs, e.g., offloading input components to smaller mobile devices, and assign the views directly to the surface (H5). Then, we proceed with the default assignment of views to surfaces based on the space requirement (H4,5). We consider strong view groups first, then weak view groups, and finally all other views. If no surface is big enough to exclusively host a group, we either accept to scale down the views (strong groups), or to split them up across multiple devices (weak groups).

The last step is arranging the views on each surface. Here, we applied an approach similar to bin packing [20]: basically, we create columns and fill them up until the available surface height is no longer sufficient. The initial size of views is based on their space requirement in relation to the surface's screen estate (i.e., $V_{Area} = V_{SR} \times S_{Area}$; H4). Because of different aspect ratios and sizes, some rows may not fill up the whole column width; in these situations we try to fill up the spots with smaller views. While adding views to columns/rows, we allow for a flexibility in view size and aspect ratio (up to 25%). As constellations can exist, where views cannot be fit into the available screen space (e.g., because of contrary aspect ratios), we scale the whole layout down to fit into the surface. Finally, we again adjust view height and width up to 50% to eliminate any free space. Although our implementation does not explicitly align views yet, this approach typically maintains the alignment/grouping implicitly as the views are processed in order of their group membership.

Control Panel for User Adaptations

Our implementation provides a control panel allowing users to fine-tune the distribution (H6). The panel shows the surfaces and distributed views in both a preview and lists. The lists provide indicators for group membership and space requirement and allows ignoring surfaces and views, making them ineligible for automatic layout. Views can also be manually assigned to surfaces by drag and drop. The system reacts differently to these changes: while ignoring views or surfaces triggers a recalculation of the complete distribution, the manual assignment only re-runs the local layout. Here, we expect users to have the mental model of reassigning one specific view, regardless of its relations to other views. Therefore, we skip the view assignment to avoid side effects. The user can also switch to a completely manual process, where they can place and scale views freely.

Currently, distribution updates are only triggered on major changes, such as a changed device configuration or when new views are added to the interface. In these situations, we fade in a miniature overview map of the surface configuration highlighting moved views and/or new surfaces. However, smaller view-specific changes, e.g., caused by filter conditions, are ignored to avoid interrupting the user.

6 STUDY: USER-CREATED DISTRIBUTIONS

In order to back up our heuristics, we compare distribution and layout generated by our system to multiple user-created ones as well as report on user ratings of the distributions.

Participants. We recruited six paid participants (age M=36.8, SD=12.59 yrs; 1 female, 5 male) at the University of Maryland. We required that all of them have both theoretical and practical background in data analysis and/or visualization theory, i.e., actively conducting research in this area or work with these interfaces on a regular basis. All participants have been active in the field for at least 3 years (M=9.8, SD=10.26 yrs).

Apparatus and Dataset. We used the *Vistribute* system as described before on a crime dataset from the City of Baltimore[3]. The example interface consisted of 10 views (Figure 5). Two bar charts showed the overall crime distribution for districts and crime types (BC-Dist-All, BC-Types-All). Selections in these were used as a filter for two connected line charts each, showing the distribution over time for 2016 and 2017

[3]https://data.baltimorecity.gov

(LC-Dist-16/17, LC-Types-16/17). A filter component allowed for filtering the data to explore subsets (Filter). The filtered output was consumed by two bar charts (weapons, BC-Weap-Filt; inside/outside location, BC-InOut-Filt) and a map (Map-Filt). We extended the prototype with a manual layout mode, allowing a free view assignment and arrangement using the control panel. Once placed on a surface, views could also be moved and resized directly in the interface.

Physical Setup. We included three device ensembles:

- S1 A *traditional dual-display desktop* (each 24 ″, full-HD, 1 landscape, 1 portrait);
- S2 A *novel desktop setup* with a laptop (13 ″, 1600 × 900 px) on a standing desk and a large display (55 ″, full-HD) within arm's reach; and
- S3 A *mobile device ensemble* consisting of a tablet (HTC Nexus 9, 9 ″, 2048 × 1536 px, landscape), a smartphone (Samsung Galaxy S8, 5.8 ″, 2690 × 1440 px, landscape), and the laptop from before.

We chose these setups as they represent realistic combinations that are already in use or are likely to be commonly used in the near future. Figure 1a-c shows similar setups.

Procedure. Participants first received a short introduction on view distribution as well as the experimental dataset. We provided them with an initial understanding for the requirements of a distribution by explaining typical scenarios and tasks in the context of the crime dataset. We also explained the abilities and connections of the existing views as well as provided a printout showing these connections.

In **Phase I**, participants were asked to distribute all views across the available surfaces for all three setups (within-participants, counter-balanced order). None of Vistribute's automatic layout functionality was active during this phase. We asked participants to think-aloud while distributing views and logged the created distributions. As the interface offered no support for alignment, we carefully adjusted them afterwards to remove smaller and unintended overlaps or offsets. These adjusted distributions were used for Phase II.

In **Phase II**, participants were shown three existing distributions for each setup. For all distributions they were asked to rate its quality on a 5-point Likert-scale and provide free-form comments. Since we included three physical setups, each participant rated nine distributions. The setup order was the same as in Phase I. From the three distributions, two were created by prior participants (randomly selected), while one was generated by *Vistribute*. Their order was also randomized per participant. We did not indicate to participants how these distribution were created. For the first two participants, we used distributions created during earlier pilot runs. In total, sessions lasted approximately one hour.

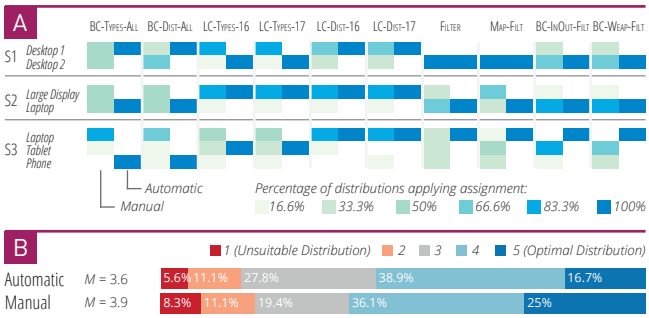

Figure 6: (a) heatmap showing view-to-surface assignment for each setup (per column: left, aggregated manual distributions; right, automatic one), e.g., 50% of participants placed BC-Types-All on the Laptop in S2; (b) the manual distributions were rated slightly better than the automatic version.

User Feedback and Findings

We found three main results: when considering a distribution, (1) participants make decisions based on very similar aspects as embodied in our heuristics, but (2) personal preferences have a strong influence leading to diverse distributions across participants (Figure 6a), and (3) the manual distributions were rated slightly better than the automatic ones (Figure 6b).

When stating their thoughts during the distribution, participants touched on similar principles as covered in our heuristics. For instance, they explicitly stated that views with more data points should be placed bigger (P1–6), connectivity must be valued (P1–4, P6), or that similar views should be aligned for comparison (P3–P4, P6). Figure 6a also shows some of these patterns, e.g., the line charts (LC-Types-16/17 and LC-Dist-16/17 form clear pairs, and, especially for S2, are often assigned to the same device (e.g., Figure 5). We also observed participants considering the influence of device size (P1, P3–4, P6) or input capabilities (P2–4, P6).

However, multiple aspects were considered differently across participants. While most participants valued smaller devices as appropriate for input purposes, P2 used the mobile devices explicitly for visualizations, as these *"can be easily passed around."* For connectivity, we observed that some participants strongly favored placing connected views adjacent to each other (P1, P6), while others found it useful to split them between devices. We also found that some aspect are not covered in our framework yet: multiple participants had a higher-level definition of data similarity by considering their semantics. As an example, the views encoding districts (LC-Dist-16/17, BC-Dist-All), the map, and the Inside-Outside bar chart were classified as geographical data, and therefore combined by three participants (P2, P4, P6). Participants (P1–2, P6) also mentioned the importance of surface adjacency and its influence on the perceived proximity between views.

As a result, we could observe a high diversity across the created distributions. In Figure 6a, this can especially be observed for the bar charts in S1 and S2, as well as for most of the views in S3. Further, no two distributions were similar. Three distributions for S1 and two for S2 used the same view-to-surfaces assignment; however, they had different local layouts. This diversity in user preferences can also be observed in the ratings in the form of high standard derivations. On average, participants rated the manual distribution (M=3.9, SD=0.99) slightly better than the automatic ones (M=3.6, SD=1.21; see Figure 6b). However, the ratings must be considered carefully: our study included only a small number of participants and they all worked only for a limited time on the distributions without performing specific analysis tasks.

Interestingly, multiple participants found the distribution *"exhausting"*, and one participant explicitly stated that *"the computer should suggest where to put things; there should be some optimization for this"* (P5), also stressing that a manual placement is considered a burden (P1, P5). On average, participants spent 8 minutes on the second and third distribution (M=19.6 minutes for the first one). Although a certain part of this time is caused by the think-aloud design and lacking interface support for aligning, even in a real-world system users would eventually have to spend a couple of minutes for the distribution. Any shortcut offered by an automatic distribution would therefore be an improvement. Finally, P1 also noted that *"semantically beautiful is much more important than aesthetically beautiful."* Hence, even if an automatic approach is not able to reach the visual quality of a manual one, it may still be able to provide a valuable layout. All created distributions are listed in the supplementary material.

7 DISCUSSION & FUTURE WORK

We believe that our framework can serve as a foundation for future research on distributed visualization. Although limitations remain, we hope to stimulate follow-up work on distribution approaches as well as aspects even beyond that. Our long-term goal is to simplify the usage of multi-device environments so that their full potential can be realized.

Limitations, Framework Extensions, and Evaluations

Participant feedback indicates that some of our heuristics or the implementation could be refined; for instance, a semantic data similarity (e.g., all location-related views) or contextual device aspects (e.g., physical device arrangements) are currently not represented, as they are hard to capture. For example, device proxemics [8, 40] can currently only be sensed with external tracking systems, which are hardly applicable outside of research prototypes [34, 49, 62]. However, this might change as internal device sensors improve [31], allowing to better facilitate cross-device dependencies.

Our current Vistribute implementation dynamically responds to changes in the device and view ensemble by automatically recomputing the distribution. Unfortunately, such events may trigger a radical rearrangement of the distribution, particularly if the surface in question is large. Beyond the overview minimap, we currently provide no mechanism to help a user reorient themselves when this happens. In the future, we may want to incorporate specific technologies to visualize changes [4, 25], e.g., by using animated transitions showing how views are rearranged from one distribution to another, or by using transient color highlights [10]. Also, distribution layout changes may require explicit user confirmation. Finally, a history of the latest applied distributions could allow switching between different variations.

In extension to our current study, more thorough evaluations should be conducted. An extensive observation study on how users manage visualizations in MDEs during an analysis session could provide further insights, e.g., how often they want to adapt the interface and for which tasks. In a quantitative manner, it would be interesting to measure performance indicators (e.g., task completion time, error rate) in comparison to non-optimized or random distributions.

From Heuristics Towards Formalism

While the Vistribute framework does not stipulate a specific distribution algorithm, our example implementation is a rather simple algorithm realizing our heuristics, rather than a formal user interface specification such as AdaM [46]. For this paper, we explicitly eschewed such a formal approach, since we felt that current practice in arranging visualization views is mostly qualitative in nature. Instead we relied on heuristics that could be balanced for each specific implementation. The results from our evaluation bore this decision out; our automatic layouts were similar to layouts hand-crafted by experts. Nevertheless, extending our current algorithm towards an optimizer can help to improve the distribution quality, especially when cases exist that cause sub-optimal layouts. This could be done by running multiple variations with different parameters and identifying the best one.

Beyond that, nothing is preventing us from implementing a formal version, akin to AdaM, based on our heuristics in the future. This could also be further extended by applying machine learning approaches for deriving weights for the heuristics. However, as machine classifiers require a large training dataset of successful distributions, this can only be a second step after introducing distributed visualization interfaces to a broader audience. Finally, even when following this vision towards a distribution purely based on formalism, we believe that allowing users to modify the result is central. Notably, it should be possible to apply these adaptions in a natural way, e.g., by drag-and-drop, and not through abstract parameters, as it is often the case for current optimizers [46].

From Distribution Towards Visualization Generation

In the process of developing our framework, we noted several times that being able to automatically generate and modify the views (instead of working with existing views) would make our approach more powerful. For instance, when scaling a view, this would make it possible to optimize the aspect ratio for improved perception [24]. Instead of just aligning two views in order to promote visual comparison, an even more sophisticated approach would be to rebuild the views to use the same chart type and normalize both of their scales to further increase consistency [48]. This step, to either generate views to complement existing ones, or even to generate a complete dashboard from scratch [43], is not far.

In other words, to truly realize the potential of multi-device environments for visual analytics, it may be necessary to entirely relinquish the task of visualization specification to the distribution middleware, merely specifying the datasets and tasks involved. Unlike the human designer, who can only enumerate so many variant visualizations for a finite set of possible device ensembles, a fully automated visualization generation engine would be able to construct precisely the visual representations that are best suited to the available hardware, physical context, and overarching analysis task.

8 CONCLUSION

We have presented *Vistribute*, a combined design space, set of heuristics, and prototype implementation for cross-device distribution of visualizations across a dynamically changing ensemble of displays and devices. Informed by current visualization practice, we have validated our heuristics and their implementation in a qualitative evaluation where visualization experts manually constructed distributed layouts. Our findings suggest that there is little qualitative difference between manual and automatic layouts, and that the automatic layout can save significant time and effort.

ACKNOWLEDGMENTS

We thank Karthik Badam for his valuable feedback as well as Sigfried Gold for providing his voice for our video. This work was supported by the Deutsche Forschungsgemeinschaft (DA 1319/3-3, DA 1319/11-1), the Danish Center for Big Data Analytics Driven Innovation (IFD-5153-00004B), the Aarhus University Research Foundation, and the U.S. National Science Foundation (IIS-1539534). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Robert Amar, James Eagan, and John Stasko. 2005. Low-level Components of Analytic Activity in Information Visualization. In *Proceedings of the IEEE Symposium on Information Visualization*. IEEE, Piscataway, NJ, USA, 111–117. https://doi.org/10.1109/infvis.2005.1532136

[2] Christopher Andrews, Alex Endert, and Chris North. 2010. Space to think: large high-resolution displays for sensemaking. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 55–64. https://doi.org/10.1145/1753326.1753336

[3] Christopher Andrews, Alex Endert, Beth Yost, and Chris North. 2011. Information visualization on large, high-resolution displays: Issues, challenges, and opportunities. *Information Visualization* 10, 4 (Aug. 2011), 341–355. https://doi.org/10.1177/1473871611415997

[4] Daniel Archambault, Helen Purchase, and Bruno Pinaud. 2011. Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *IEEE Transactions on Visualization and Computer Graphics* 17, 4 (April 2011), 539–552. https://doi.org/10.1109/tvcg.2010.78

[5] Sriram Karthik Badam and Niklas Elmqvist. 2014. PolyChrome: A Cross-Device Framework for Collaborative Web Visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 109–118. https://doi.org/10.1145/2669485.2669518

[6] Sriram Karthik Badam, Eli Fisher, and Niklas Elmqvist. 2015. Munin: A Peer-to-Peer Middleware for Ubiquitous Analytics and Visualization Spaces. *IEEE Transactions on Visualization and Computer Graphics* 21, 2 (Feb. 2015), 215–228. https://doi.org/10.1109/tvcg.2014.2337337

[7] Sriram Karthik Badam, Andreas Mathisen, Roman Rädle, Clemens N. Klokmose, and Niklas Elmqvist. 2019. Vistrates: A Component Model for Ubiquitous Analytics. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 586–596. https://doi.org/10.1109/TVCG.2018.2865144

[8] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 121–130. https://doi.org/10.1145/1936652.1936676

[9] Patrick Baudisch and Christian Holz. 2010. My new PC is a mobile phone. *ACM XRDS* 16, 4 (June 2010), 36–41. https://doi.org/10.1145/1764848.1764857

[10] Patrick Baudisch, Desney S. Tan, Maxime Collomb, Daniel C. Robbins, Ken Hinckley, Maneesh Agrawala, Shengdong Zhao, and Gonzalo Ramos. 2006. Phosphor: explaining transitions in the user interface using afterglow effects. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 169–178. https://doi.org/10.1145/1166253.1166280

[11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

[12] Lauren Bradel, Alex Endert, Kristen Koch, Christopher Andrews, and Chris North. 2013. Large high resolution displays for co-located collaborative sensemaking: Display usage and territoriality. *International Journal of Human-Computer Studies* 71, 11 (Nov. 2013), 1078–1088. https://doi.org/10.1016/j.ijhcs.2013.07.004

[13] Matthew Brehmer and Tamara Munzner. 2013. A Multi-Level Typology of Abstract Visualization Tasks. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (Dec. 2013), 2376–2385. https://doi.org/10.1109/tvcg.2013.124

[14] Olivier Chapuis, Anastasia Bezerianos, and Stelios Frantzeskakis. 2014. Smarties: an input system for wall display development. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2763–2772. https://doi.org/10.1145/2556288.2556956

[15] Haeyong Chung, Chris North, Sarang Joshi, and Jian Chen. 2015. Four considerations for supporting visual analysis in display ecologies. In

Proceedings of the IEEE Conference on Visual Analytics Science and Technology. IEEE, Piscataway, NJ, USA, 33–40. https://doi.org/10.1109/vast.2015.7347628

[16] Haeyong Chung, Chris North, Jessica Zeitz Self, Sharon Lynn Chu, and Francis K. H. Quek. 2014. VisPorter: facilitating information sharing for collaborative sensemaking on multiple displays. *Personal and Ubiquitous Computing* 18, 5 (June 2014), 1169–1186. https://doi.org/10.1007/s00779-013-0727-2

[17] Matt Conlen and Jeffrey Heer. 2018. Idyll: A Markup Language for Authoring and Publishing Interactive Articles on the Web. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 977–989. https://doi.org/10.1145/3242587.3242600

[18] David Dearman and Jeffery S. Pierce. 2008. It's on My Other Computer!: Computing with Multiple Devices. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 767–776. https://doi.org/10.1145/1357054.1357177

[19] Niklas Elmqvist and Pourang Irani. 2013. Ubiquitous Analytics: Interacting with Big Data Anywhere, Anytime. *IEEE Computer* 46, 4 (April 2013), 86–89. https://doi.org/10.1109/mc.2013.147

[20] M. R. Garey and D. S. Johnson. 1981. Approximation Algorithms for Bin Packing Problems: A Survey. In *Analysis and Design of Algorithms in Combinatorial Optimization*. Springer Vienna, Vienna, 147–172. https://doi.org/10.1007/978-3-7091-2748-3_8

[21] Michael Gleicher. 2018. Considerations for Visualizing Comparison. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 413–423. https://doi.org/10.1109/tvcg.2017.2744199

[22] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D. Hansen, and Jonathan C. Roberts. 2011. Visual comparison for information visualization. *Information Visualization* 10, 4 (Sept. 2011), 289–309. https://doi.org/10.1177/1473871611416549

[23] Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: enabling and understanding cross-device interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2773–2782. https://doi.org/10.1145/2556288.2557170

[24] Jeffrey Heer and Maneesh Agrawala. 2006. Multi-Scale Banking to 45 Degrees. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (Sept. 2006), 701–708. https://doi.org/10.1109/tvcg.2006.163

[25] Jeffrey Heer and George Robertson. 2007. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (Nov. 2007), 1240–1247. https://doi.org/10.1109/tvcg.2007.70539

[26] Tom Horak, Sriram Karthik Badam, Niklas Elmqvist, and Raimund Dachselt. 2018. When David Meets Goliath: Combining Smartwatches with a Large Vertical Display for Visual Data Exploration. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 19:1–19:13. https://doi.org/10.1145/3173574.3173593

[27] Steven Houben and Nicolai Marquardt. 2015. WATCHCONNECT: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1247–1256. https://doi.org/10.1145/2702123.2702215

[28] Maria Husmann, Alfonso Murolo, Nicolas Kick, Linda Di Geronimo, and Moira C. Norrie. 2018. Supporting out of office software development using personal devices. In *Proceedings of the ACM Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, New York, NY, USA, 27:1–27:11. https://doi.org/10.1145/3229434.3229454

[29] Mikkel R. Jakobsen and Kasper Hornbæk. 2013. Interactive Visualizations on Large and Small Displays: The Interrelation of Display Size, Information Space, and Scale. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (Dec. 2013), 2336–2345. https://doi.org/10.1109/tvcg.2013.170

[30] Waqas Javed and Niklas Elmqvist. 2012. Exploring the Design Space of Composite Visualization. In *Proceedings of the IEEE Pacific Symposium on Visualization*. IEEE, Piscataway, NJ, USA, 1–8. https://doi.org/10.1109/pacificvis.2012.6183556

[31] Haojian Jin, Christian Holz, and Kasper Hornbæk. 2015. Tracko: Ad-hoc Mobile 3D Tracking Using Bluetooth Low Energy and Inaudible Signals for Cross-Device Interaction. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 147–156. https://doi.org/10.1145/2807442.2807475

[32] Ulrike Kister, Konstantin Klamka, Christian Tominski, and Raimund Dachselt. 2017. GraSp: Combining Spatially-aware Mobile Devices and a Display Wall for Graph Visualization and Interaction. *Computer Graphics Forum* 36, 3 (June 2017), 503–514. https://doi.org/10.1111/cgf.13206

[33] Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 280–290. https://doi.org/10.1145/2807442.2807446

[34] Ricardo Langner, Tom Horak, and Raimund Dachselt. 2018. VisTiles: Coordinating and Combining Co-located Mobile Devices for Visual Data Exploration. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 626–636. https://doi.org/10.1109/tvcg.2017.2744019

[35] Ricardo Langner, Ulrike Kister, and Raimund Dachselt. 2019. Multiple Coordinated Views at Large Displays for Multiple Users: Empirical Findings on User Behavior, Movements, and Distances. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 608–618. https://doi.org/10.1109/TVCG.2018.2865235

[36] Ricardo Langner, Ulrich von Zadow, Tom Horak, Annett Mitschick, and Raimund Dachselt. 2016. Content Sharing Between Spatially-Aware Mobile Phones and Large Vertical Displays Supporting Collaborative Work. In *Collaboration Meets Interactive Spaces*. Springer International Publishing, 75–96. https://doi.org/10.1007/978-3-319-45853-3_5

[37] David Ledo, Saul Greenberg, Nicolai Marquardt, and Sebastian Boring. 2015. Proxemic-Aware Controls: Designing Remote Controls for Ubiquitous Computing Ecologies. In *Proceedings of the ACM Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, New York, NY, USA, 187–198. https://doi.org/10.1145/2785830.2785871

[38] Bongshin Lee, Petra Isenberg, Nathalie Henry Riche, and Sheelagh Carpendale. 2012. Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec. 2012), 2689–2698. https://doi.org/10.1109/tvcg.2012.204

[39] Can Liu, Olivier Chapuis, Michel Beaudouin-Lafon, Eric Lecolinet, and Wendy E. Mackay. 2014. Effects of display size and navigation type on a classification task. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 4147–4156. https://doi.org/10.1145/2556288.2557020

[40] Nicolai Marquardt and Saul Greenberg. 2012. Informing the Design of Proxemic Interactions. *IEEE Pervasive Computing* 11, 2 (Feb. 2012), 14–23. https://doi.org/10.1109/mprv.2012.15

[41] Nicolai Marquardt, Ken Hinckley, and Saul Greenberg. 2012. Cross-device interaction via micro-mobility and f-formations. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 13–22. https://doi.org/10.1145/2380116.2380121

[42] Will McGrath, Brian Bowman, David McCallum, Juan David Hincapié-Ramos, Niklas Elmqvist, and Pourang Irani. 2012. Branch-explore-merge: Facilitating Real-time Revision Control in Collaborative Visual Exploration. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 235–244.

https://doi.org/10.1145/2396636.2396673

[43] Dominik Moritz, Chenglong Wang, Greg L. Nelson, Halden Lin, Adam M. Smith, Bill Howe, and Jeffrey Heer. 2019. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 438–448. https://doi.org/10.1109/tvcg.2018.2865240

[44] Michael Nebeling. 2017. XDBrowser 2.0: Semi-Automatic Generation of Cross-Device Interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 4574–4584. https://doi.org/10.1145/3025453.3025547

[45] Michael Nebeling and Anind K. Dey. 2016. XDBrowser: User-Defined Cross-Device Web Page Designs. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 5494–5505. https://doi.org/10.1145/2858036.2858048

[46] Seonwook Park, Antti Oulasvirta, Otmar Hilliges, Christoph Gebhardt, Roman Rädle, Anna Maria Feit, Hana Vrzakova, Niraj Ramesh Dayama, Hui-Shyong Yeo, Clemens N. Klokmose, and Aaron Quigley. 2018. AdaM: Adapting Multi-User Interfaces for Collaborative Environments in Real-Time. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 184:1–184:14. https://doi.org/10.1145/3173574.3173758

[47] Thomas Plank, Hans-Christian Jetter, Roman Rädle, Clemens N. Klokmose, Thomas Luger, and Harald Reiterer. 2017. Is Two Enough?! Studying Benefits, Barriers, and Biases of Multi-Tablet Use for Collaborative Visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 4548–4560. https://doi.org/10.1145/3025453.3025537

[48] Zening Qu and Jessica Hullman. 2018. Keeping Multiple Views Consistent: Constraints, Validations, and Exceptions in Visualization Authoring. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 468–477. https://doi.org/10.1109/tvcg.2017.2744198

[49] Roman Rädle, Hans-Christian Jetter, Jonathan Fischer, Inti Gabriel, Clemens N. Klokmose, Harald Reiterer, and Christian Holz. 2018. PolarTrack: Optical Outside-In Device Tracking that Exploits Display Polarization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 497:1–497:9. https://doi.org/10.1145/3173574.3174071

[50] Roman Rädle, Hans-Christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. 2014. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 45–54. https://doi.org/10.1145/2669485.2669500

[51] Roman Rädle, Midas Nouwens, Kristian Antonsen, James R. Eagan, and Clemens N. Klokmose. 2017. Codestrates: Literate Computing with Webstrates. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 715–725. https://doi.org/10.1145/3126594.3126642

[52] Jonathan C. Roberts. 2007. State of the Art: Coordinated & Multiple Views in Exploratory Visualization. In *Proccedings of the IEEE Conference on Coordinated and Multiple Views in Exploratory Visualization*. IEEE, Piscataway, NJ, USA, 61–71. https://doi.org/10.1109/cmv.2007.20

[53] Jonathan C. Roberts, Panagiotis D. Ritsos, Sriram Karthik Badam, Dominique Brodbeck, Jessie Kennedy, and Niklas Elmqvist. 2014. Visualization beyond the Desktop—the Next Big Thing. *IEEE Computer Graphics and Applications* 34, 6 (Nov. 2014), 26–34. https://doi.org/10.1109/mcg.2014.82

[54] Stephanie Santosa and Daniel Wigdor. 2013. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing*. ACM, New York, NY, USA, 63–72. https://doi.org/10.1145/2493432.2493476

[55] Alper Sarikaya, Michael Correll, Lyn Bartram, Melanie Tory, and Danyel Fisher. 2019. What Do We Talk About When We Talk About Dashboards? *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 682–692. https://doi.org/10.1109/TVCG.2018.2864903

[56] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan. 2017), 341–350. https://doi.org/10.1109/tvcg.2016.2599030

[57] Arvind Satyanarayan, Kanit Wongsuphasawat, and Jeffrey Heer. 2014. Declarative interaction design for data visualization. In *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 669–678. https://doi.org/10.1145/2642918.2647360

[58] Mario Schreiner, Roman Rädle, Hans-Christian Jetter, and Harald Reiterer. 2015. Connichiwa: A Framework for Cross-Device Web Applications. In *Proceedings of the ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2163–2168. https://doi.org/10.1145/2702613.2732909

[59] Shaishav Siddhpuria, Sylvain Malacria, Mathieu Nancel, and Edward Lank. 2018. Pointing at a Distance with Everyday Smart Devices. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 173:1–173:11. https://doi.org/10.1145/3173574.3173747

[60] Martin Spindler, Christian Tominski, Heidrun Schumann, and Raimund Dachselt. 2010. Tangible views for information visualization. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 157–166. https://doi.org/10.1145/1936652.1936684

[61] Ulrich von Zadow, Wolfgang Büschel, Ricardo Langner, and Raimund Dachselt. 2014. SleeD: Using a Sleeve Display to Interact with Touch-sensitive Display Walls. In *Proceedings of the ACM Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 129–138. https://doi.org/10.1145/2669485.2669507

[62] Paweł Woźniak, Lars Lischke, Benjamin Schmidt, Shengdong Zhao, and Morten Fjeld. 2014. Thaddeus: a dual device interaction space for exploring information visualisation. In *Proceedings of the ACM Nordic Conference on Human-Computer Interaction*. ACM, New York, NY, USA, 41–50. https://doi.org/10.1145/2639189.2639237

[63] Jishuo Yang and Daniel Wigdor. 2014. Panelrama: enabling easy specification of cross-device web applications. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2783–2792. https://doi.org/10.1145/2556288.2557199

[64] Ji Soo Yi, Youn ah Kang, and John Stasko. 2007. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (Nov. 2007), 1224–1231. https://doi.org/10.1109/tvcg.2007.70515