This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

PanoramicData: Data Analysis through Pen & Touch

Emanuel Zgraggen, *Student Member, IEEE*, Robert Zeleznik, and Steven M. Drucker, *Member, IEEE*

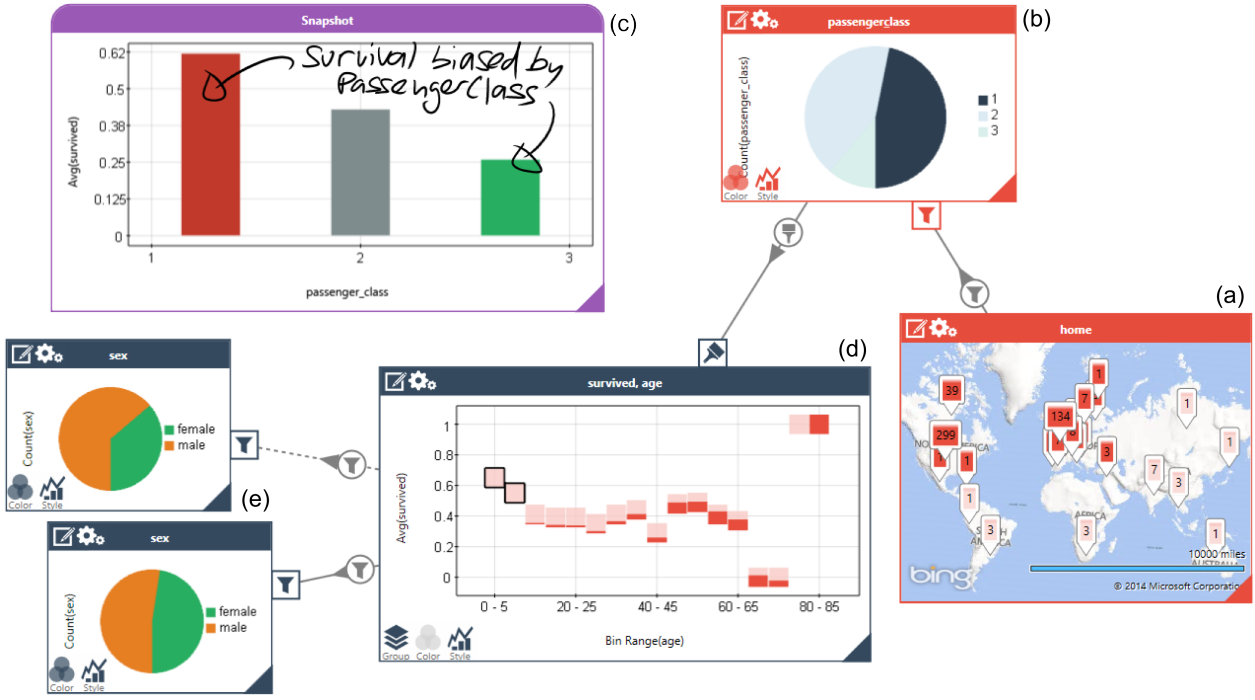

Fig. 1. Data panorama of the Titanic passenger data-set. (a) Map-view of passenger home towns. (b) Pie-chart of passenger distribution from North-America and Europe (filtered by selection in (a)) across passenger classes. (c) Annotated snapshot of average survival rate by passenger class. (d) Average survival rate for passenger age bins. Brushed by selections in (a) and (b). (e, bottom) Gender distribution for passengers selected in (d). (e, Top) Gender distribution for passengers not selected in (d). Dashed line indicates inversion of selection.

Abstract— Interactively exploring multidimensional datasets requires frequent switching among a range of distinct but inter-related tasks (e.g., producing different visuals based on different column sets, calculating new variables, and observing the interactions between sets of data). Existing approaches either target specific different problem domains (e.g., data-transformation or data-presentation) or expose only limited aspects of the general exploratory process; in either case, users are forced to adopt coping strategies (e.g., arranging windows or using undo as a mechanism for comparison instead of using side-by-side displays) to compensate for the lack of an integrated suite of exploratory tools. PanoramicData (PD) addresses these problems by unifying a comprehensive set of tools for visual data exploration into a hybrid pen and touch system designed to exploit the visualization advantages of large interactive displays. PD goes beyond just familiar visualizations by including direct UI support for data transformation and aggregation, filtering and brushing. Leveraging an unbounded whiteboard metaphor, users can combine these tools like building blocks to create detailed interactive visual display networks in which each visualization can act as a filter for others. Further, by operating directly on relational-databases, PD provides an approachable visual language that exposes a broad set of the expressive power of SQL, including functionally complete logic filtering, computation of aggregates and natural table joins. To understand the implications of this novel approach, we conducted a formative user study with both data and visualization experts. The results indicated that the system provided a fluid and natural user experience for probing multi-dimensional data and was able to cover the full range of queries that the users wanted to pose.

Index Terms—Visual analytics, pen and touch, user interfaces, interaction design, coordinated and multiple views

- *Emanuel Zgraggen is with Brown University. E-mail: ez@cs.brown.edu.*
- *Robert Zeleznik is with Brown University. E-mail: bcz@cs.brown.edu.*
- *Steven M. Drucker is with Microsoft Research. E-mail: sdrucker@microsoft.com.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Introduction

Visual data analysis – gaining insights out of a dataset through visualizations – is an interactive and iterative process where users need to switch frequently among a range of distinct but interrelated tasks. The set of tasks that recur in visual data analysis, as well as the tools that support them, is well understood [2, 14]. However, designing a system that supports this diversity of tasks in a unified, understandable and approachable way is a non-trivial challenge itself.

Design approaches for visual data analysis typically fall into either the rich-general or strong-specific categories. Rich general approaches are manifest in the form of programming languages, such as SQL or Python, possibly in combination with a general purpose

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

tool like Excel. The syntactic, programming nature of these rich general systems creates learning and performance barriers for all but the most dedicated users. Alternatively, strong specific approaches provide more accessible tools, albeit for some narrower problem domain, such as data-transformations [17], data presentations [19], or limited tasks within data exploration [27]. Unfortunately, this approach requires users to adopt cumbersome workarounds, such as exporting data to other programs, to complete many common tasks that fall outside the system's target domain.

Inspired by the metaphor of narrative panoramas, our research attempts an interactive rich-general approach by providing a small set of simple visualization primitives which can be linked in very direct, concrete ways through Boolean operations on an unbounded canvas to create sophisticated visualizations. These dynamic panoramas are expressive enough to represent interactive visualizations for the scope of tasks required for detailed exploratory analysis of multi-dimensional structured data. Although this approach requires that users understand how to piecewise compose a complex visualization, we believe even beginning users can develop this mastery with minimal training because it corresponds to the incremental way in which questions generally form. For example, a complex query might germinate from a simple question about a dataset, such as "how many women died on the titanic?" Followed by, "show their age distribution," and then by "compare those distributions by berthing class," and so on. The totality of such a visualization chain can be a rather complex acyclic graph. However, the panoramic approach to visually representing this working set of composed queries is powerful [4, 9]. Not only is the step-by-step creation process approachable, but it also affords interactive "handles" at each query stage to explore tangential queries, such as switching from women to men in the example, or to simply verify that the question had been posed correctly.

During our iterative design of PanoramicData (PD), our embodiment of this rich-general approach, we referred to Heer & Shneiderman's [14] *taxonomy of interactive dynamics for visual analytics* as well as the *analytical task taxonomy* of Amar, Eagan & Stasko [2] as a guide to ensure that PD broadly covers exploration and analysis tasks. We observed that in order to support those tasks in a comprehensible and unified way our most effective design choices clustered around a set of four concepts (Derivable Visualizations, Exposing Expressive Data-Operations, Unbounded-Space, and Boolean Composition). These concepts, which all have been discussed to some extent in the literature, both guided and encapsulated the critical reasoning behind our design decisions. Although we found that each of these concepts typically allowed for multiple different equivalent designs, omitting any concepts had a significant negative impact on either usability or visualization power. The strength of PD lies within the effective combination of those four concepts. We discuss these concepts in detail and describe how these concepts, and compounds of them, provide a framework that supports a wide range of data-exploration and analysis tasks.

A large part of our design efforts were targeted towards offering a *fluid* interaction model [10] which serves as the overarching structure of PD and unifies those four concepts in a comprehensible way. Inspired by recent work suggesting the benefits of pen and / or touch for analysis work [6, 8, 19, 34], we specifically designed PD for interactive whiteboards and pen-enabled tablets. By considering what can easily be accomplished through the use of pen and touch, we were able to combine the aforementioned concepts in a way that avoids explicit mode switching, minimizes UI cluttering, reduces indirection in the interface and provides effortless switching between the variety of interrelated tasks that data exploration and analysis require. Even though we emphasize the modalities of pen & touch, the entire UI could still be conveniently operated through standard mouse and keyboard interaction.

In this paper we present PD a novel pen & touch system for data exploration and analysis which is based on four core concepts, *Derivable Visualizations*, *Exposing Expressive Data-Operations*, *Unbounded Space* and *Boolean Composition*. The combination and interaction between those concepts presents an approachable interface that allows for incremental and piecewise query specification where intermediate visualizations serve as feedback as well as interactive handles to adjust query parameters. PD supports a wide range of common tasks.

We evaluated PD through a formative user study with both data and visualization experts. The results support our belief that PD, and the combination of the design- concepts it embodies, provides a fluid and intuitive user experience while still being expressive enough to answer the range of queries that users are likely to pose.

1 Related Work

We relate and contrast our work to research efforts in the areas *of Pen & Touch Visualizations*, *Linked and Coordinated Views* and *Database Interaction*.

1.1 Pen & Touch Visualization

Our work has been inspired by the research of Elmqvist et al. [10] and Lee et al. [18], which emphasize the importance of interaction to Information Visualization and propose to investigate new interaction models that go beyond traditional WIMP (windows, icons, menus, pointers) interfaces. More specifically, [6, 19, 34] have transposed insights about the usefulness of whiteboards in supporting thinking, collaborating and general problem solving processes [21, 33] to interactive whiteboards; an emerging class of hardware. SketchStory [19] focuses on the insight dissemination aspect of data-analysis. It supports presentation of pre-recorded data-related insights through a gestural pen & touch UI. Another pen & touch based whiteboard UI is presented in SketchVis [6]. It allows users to draw and label charts which the system fills in correspondingly and offers gestures to filter objects. Furthermore it offers a set of mnemonic gestures to specify data-transformations and mapping of visual elements. While similar to our work, we focus less on gestures for chart creation and more on gestural approaches to coordinate multiple visualizations.

Another direction of recent research investigates the usefulness of touch interaction for data exploration and analysis tasks. The TouchViz paper [8] presents FLUID, a touch interface for manipulating data-visualizations. The techniques presented focus on a single visualization rather than on a network of linked and coordinated views and filters.

1.2 Linked and Coordinated Views

The notion of coordinating visualizations manually in order to construct custom exploration interfaces has been introduced by [22]. SnapTogether Visualization allows users to coordinate views in order to support a set of common tasks, such as *Brushing-and-linking*, *Overview and detail view, Drill-down, Synchronized scrolling and Details on demand*. While this idea serves as a fundamental building block in PD, we expanded this concept through exposing finer-grained control in view coordination by allowing views to have multiple inputs that are combinable through Boolean operators and by propagating filtering operations across multiple hops. By rethinking the concept of snappable visualizations in terms of a gestural UI and by abstracting the underlying database-schema we are able to reduce the mental overhead that the heavy-weight view-coordination dialogs of SnapTogehter Visualization impose on users.

GraphTrail [9] is a system that allows exploration of large network datasets while preserving exploration history. The system is optimized to work with network oriented data such as social networks and scientific collaborations. One of the core-concepts of *Derivable Visualizations*, being able to create visualizations out of existing ones in order to create query-chains, has been explored by GraphTrail. However, GraphTrail displays these chains statically whereas PD offers more flexibility through interactive modification of these

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

chains and their elements. DataMeadow [11] presents an interactive visual analytics system based on *DataRoses;* a parallel coordinate starplot that exposes filtering along its axes and linking. We expand on their notion of linking by providing different types of links and base our implementation on familiar charts that allow for a higher degree of customizability and data-transformations.

Yuan et al [36] introduce a system that allows users to build up a visualization tree through a divide and conquer strategy. While this approach is similar to our piecewise query specification, it does not allow for manual rewiring of linked visualizations nor does it support data-transformations such as grouping or aggregation functions. Lark [29] links visual elements through a meta-visualizations which supports the creation of multiple variations of a view. However, linking between views from different data-attributes is not possible and it also supports limited data-types and operations. The "Stack'n'flip" UI presented in [28] uses links between visualization to show exploration histories and to guide users through an analysis based on a pre-defined setup model. VisLink [7] presents a visualization technique where 2D visualizations can be organized in 3D planes and relationships between views are displayed through edges. VisLink however does not address view-creation or data-transformation.

PD's filter-chains are comparable to dataflow networks. It is therefore similar to [1, 31], but offers specialized building-blocks which are targeted towards data-centered tasks.

1.3 Database Interaction

Tableau Software and its research predecessor Polaris [27] are systems to visually analyze large datasets. They offer support to create visually appealing and print-ready visualization of large datasets with a high degree of customizability, but have limited functionality to link multiple visualizations together.

dbTouch [16] presents a UI that *enables users to touch and manipulate data intuitively* and suggests the need for new database systems which are optimized for fluid interaction with data. While its UI is limited in its support for common analysis tasks and does not expose familiar visualizations, it coins the term *Schema-less Querying*. PD supports this notion of abstracting the complexity of the underlying database schema by offering implicit table-joins.

Approaches to create visual query languages have been an active research topic [1, 35]. Those systems have limited ability to incorporate interactive visualizations or coordinate visualizations and often require users to understand the underlying database-schema in detail.

2 Core Concepts

PD's design emerged from a series of implementation iterations in which we explored the relative value of exposing different levels of data functionality, and of different interaction styles and techniques for creating customized, interactive and coordinated visualizations. Our intent was to create a system that was approachable and predictable for untrained users yet comprehensive enough to support the complex queries of advanced users. Over time, we observed that our more effective design choices clustered around a handful of concepts, all of which have been discussed to some degree in the literature, and that by being aware of these patterns, we were better able to identify and prune design alternatives. Ultimately, we resolved upon a set of four core concepts which we feel depict the fundamentally important characteristics of our approach.

2.1 Derivable Visualizations (C1)

Since visualizations are the focal components of any visual analysis system, users spend significant effort creating views of underlying data attributes. While several methods address this view-specification step [14], each with their own set of benefits and trade-offs, we believe, similar to approaches exposed in GraphTrail [9], that visualizations must additionally be derivable. Derivable visualizations are new, tangential visualizations made by directly referencing or reusing part of existing visualizations. By effortlessly deriving visualizations from existing ones, users can fluidly explore what-if scenarios of related data without disrupting the relevant context of their existing visualizations. In addition, derivable visualizations can reduce the complexity of creating an initial visualization, since modifying is typically easier than creating. In some ways, they become a form of "suggested visualization" [12]. An important consideration for derivable visualizations is to maintain visual consistency (e.g., same color or size) between data attributes that are shared in different visualizations.

2.2 Exposing Expressive Data-Operations (C2)

Manipulating data into a format capable of visualization requires data transformation (e.g., aggregation, grouping-operations) and derivation tools (e.g., calculating new field values from existing data) tools [27]. Further, these tools need to be available directly in the context where they are needed to avoid the cognitive disruption of searching. In the limit, every possible transformation could be needed at any point, requiring the full power of a complete scripting or programming language. Although such completeness should be a goal, providing a sufficiently large subset of possible data-operations may be sufficient, particularly if achieving completeness comes at the cost of obscuring or confounding the more frequent simple tasks.

2.3 Unbounded Space (C3)

Managing display space is a requirement of any interactive system. However, for cognitively heavyweight tasks, the distraction of having to switch between views to see the components of a working set of information can significantly impact task performance [4]. By providing unbounded space including simple techniques for managing that space, several critical analytic tasks can be simplified to implicit, familiar activities. Multiple visualizations can simply be juxtaposed for comparison; reasoning chains can be reviewed to validate results; and maintaining prior queries in the periphery provides temporal context [3, 9].

2.4 Boolean Composition (C4)

They key concept for creating visualization networks is that each visualization's output must be combinable with the output of other visualizations [22]. We believe that having simple operations between visualizations affords complex results while imposing only the cognitive burden of understanding basic Boolean logic [25].

This requirement is not equivalent to typical data flow because what flows between visualizations is not data records themselves, but rather the data selection specifications used by the visualizations. That is, one visualization showing a filtered selection of data column "x", could be linked to another visualization showing data column "y". With data flow, this wouldn't make sense, but with our notion of composition, the second visualization would do the equivalent of a database join of column "y" with column "x" and then apply the "x" column filter before projecting to just column "y". This approach allows for multiple visualizations to be linked to a single new visualization, where Boolean logic operators determine which data records the new visualization receives. For generality and in particular to support the notion of visualization brushing, we extend this composition requirement to include an operator for preserving all the input data selection specifications as an array instead of always combining them with Boolean logic into single data selection specification.

3 The PanoramicData Prototype System

We developed PD, the embodiment of a rich-general approach to data analysis which unifies the aforementioned four design concepts in a comprehensible way, through an iterative process in which we steadily added and refined features in order to support a wide range of exploration and analysis tasks. We will motivate our approach

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

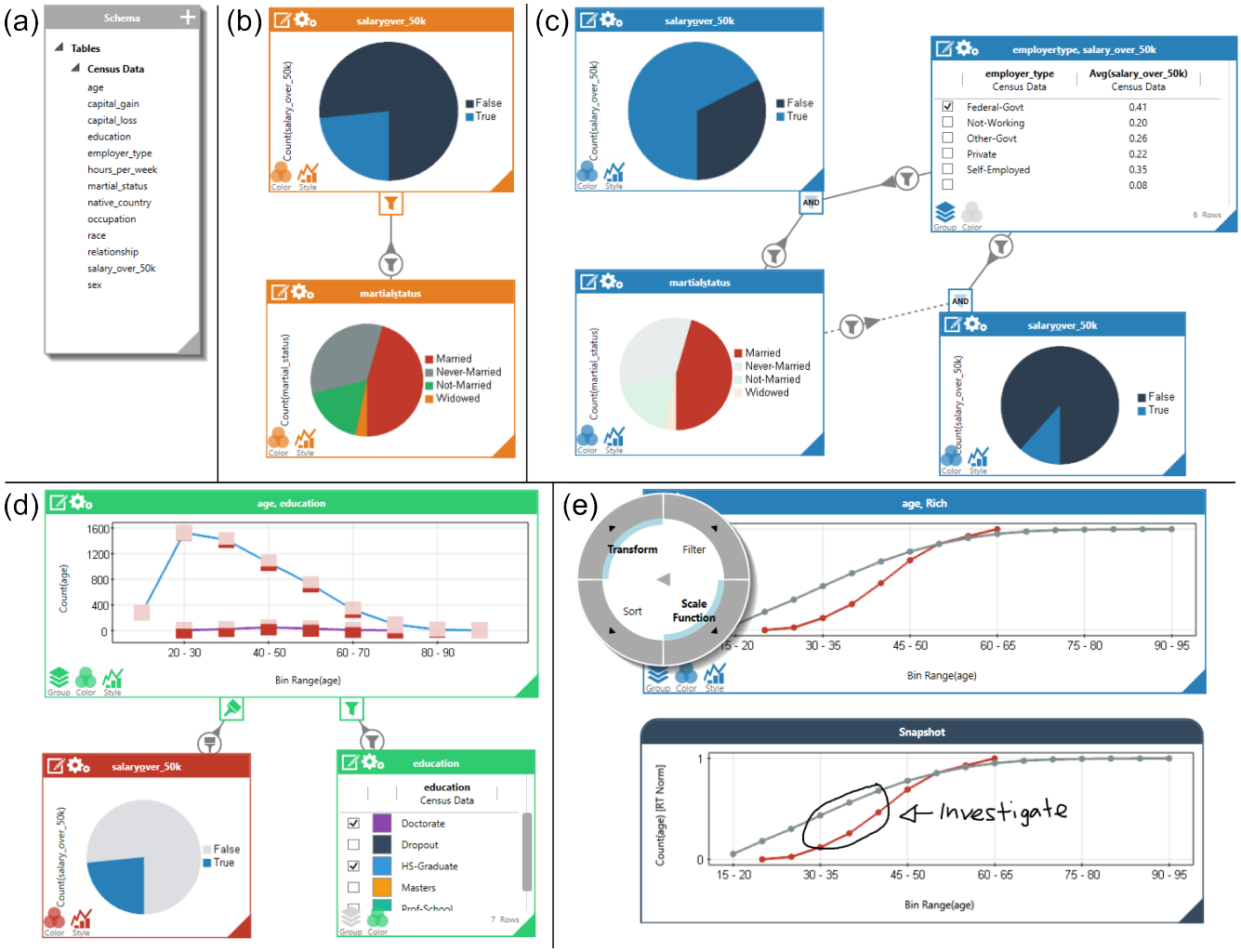

Fig. 2. Different parts of a data panorama create by a user exploring Census data. The different parts are described in Section 3.1.

through an introductory use-case and will then highlight PD's key features in the context of their relation to the four concepts and how they are used to perform common exploration and analysis tasks. We will point directly to specific items of Heer & Shneiderman's (**HS**) *taxonomy of interactive dynamics for visual analysis* [14] (Table 1).

3.1 Introductory Use-Case

Figure 2 shows different parts of a rich, dynamic visualization panorama that a user constructed with PD while exploring and analyzing a random sample of census data (10,000 records). By double-tapping on the background the user opens up the schema-viewer which displays all the attributes of the data-set (Figure 2 (a)). The user drags two attributes, marital-status and salary-over-50K, out of the schema-viewer and drops them anywhere on the 2D canvas (C3). To analyze the relationship between the two attributes the user uses the pen to connect the two (C4) (Figure 2 (b)). Tapping on a slice in the marital-status pie-chart filters the second pie-chart to only show those records that satisfy the marital-status selected in the first chart. He toggles through the different marital statuses to observe if any of those have a higher chance of earning more than $50K annually than the rest. By dragging and dropping he derives two new visualizations and connects them again by using the pen (C1) (Figure 2 (c)). He has now arranged (C3, C4) a custom exploration interface that lets him analyze the correlation between employer-type and marital-status and how those reflect on people's annual salaries. He gains the insight that people who are married and work for the federal government have a significantly higher chance of earning more than $50K than other groups. By flipping one of the visualizations around and using handwritten input he tags people in that group with the keyword "rich" (C2). This will facilitate later referral to this insight.

The user wants to explore some more attributes of this data-set; age and education-level (Figure 2 (d)). He draws an "L" type-shape (indicating X and Y axes) on the canvas and drops attributes from the schema-viewer to label the axes of his graph. By writing a "C" onto the label of the Y-axis he transforms the data to display the aggregate-count of the attribute instead of the raw value (C2). He changes the grouping context of this aggregate function to use a binning strategy instead of distinct values. He now sees the distribution of people's ages. By dropping another attribute onto the graph's color drop-target he splits the rendering into separated series. Now the chart visualizes age histograms for each education-level in a different color. To understand the meaning of the colors and to be able to filter on specific series he derives a legend-visualization by dragging off the color icon. Once again, he connects the visualizations with the pen so that he brushes the chart to see the probability of earning more than $50K individually for each data-point (C4). Doctorates are above this threshold even early-on in their career, while most High-school graduates are below it.

Our user creates another similar chart by first using a multi-touch copy gesture and then modifying it through drag and drop operations. He has colored this second chart by dropping his previously created "rich" tag onto the color-drop target. He now sees two age distributions, as a running total after applying a data-transformation through a radial menu accessible on the Y-axis (C2) (Figure 2 (e) top), one for people that match his tag and the second one for people who do not match his tag. By additionally selecting people who are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

self-employed in the tagged query he updates his definition of the "rich" tag and the age-histogram chart updates accordingly (C2, C4). To preserve an interesting insight seen in this chart he creates a static snapshot of it (C1) and annotates it with handwriting ((Figure 2 (e) bottom).

3.2 Pen & Touch and Gestural Interaction

Many analysis systems [27] offer a rich set of functionality that is exposed trough traditional WIMP (Windows, Icons, Menus, Pointer) interfaces and are prone to come with the drawbacks of this UI-metaphor [32]. The interaction design of a data analysis system should reduce the mental overhead and should not distract from the fundamental task. As others have argued for Information Visualization in general [10, 18], we believe that especially visual data analysis systems could heavily benefit from advances made in interaction technologies. Being able to offload user interface reasoning (e.g., when searching for a tool) to sub-conscious natural interactions promises a significant benefit for data analysis work which is already cognitively overloaded. Gestural interactions for managing space with touch gestures are well known and effective. However, touch alone is not expressive enough to disambiguate certain interactions, such as performing a selection on data vs. moving the container of the data. By combining both pen and touch in the UI, we believe expressive power can be enhanced without increasing cognitive load since users can learn to subconsciously associate certain interaction, such as manipulation with touch, and others such as region selection with the pen. In addition, pen gestures offer the possibility of affording very efficient, easily remembered modeless shortcuts for abstract operations, such as writing a sigma symbol over a column of data to view its sum (C2). To achieve the cognitive offloading benefits of gesturing, interactions must be consistent across the entire system so that users will feel comfortable performing interactions with just muscle memory.

By considering what can easily be accomplished through the use of pen and touch, we were able to focus on an uninterrupted experience. Having two input modes eliminates the need for many mode changes and provides a consistent selection metaphor. Recognizing pen-input also obviates keyboard-input and reinforces PD's whiteboard metaphor. Using a pen to handwrite annotations on the background or on a snapshot of a visualization feels natural (**HS** "Annotate"). The pen is also used when precise control is needed, for example when selecting only a few data-points within a large scatterplot or when pointing to a narrow slice in a pie-chart (**HS** "Select", "Filter") We also use the pen to enable fluid interactions through shortcuts of commonly used actions within the system. Two visualizations can be linked together by drawing a line between them (C4, **HS** "Coordinate") or sorting within a table-view is performed by performing an up- or down-flick gesture (**HS** "Sort") similar to [23]. Additionally, selections within a visualization can be inverted by using a flick-gesture or transformations to data-attributes can be performed through a set of symbolic gestures (C4, **HS** "Filter", "Select", "Derive"). Similar to [6] PD offers pen-gestures to create new views. An "L"-shape drawing on the background is used to create scatter-, line- or bar-charts, a circle gesture for pie-charts and a rectangular gesture for tables. Finally, PD uses a scribble-erase gesture as a way to support deletion of unwanted content or links [38].

Touch allows for a fluid interaction with all visual elements within PD. For example all the drag & drop operations that PD expose can be performed through touch-interactions. We make extensive use of drag & drop operations in order to create, derive or modify visualizations (**HS** "Visualize"). Additional touch gestures include, a double-tap gesture anywhere on the 2D canvas go gain access to all attributes of the dataset, a press-and-hold gesture to get quick-previews of visualizations within a table-viewer or a two-finger gesture to create copies of visualizations. We also expose well-known direct-manipulation touch gestures to pan and zoom the 2D canvas or to change the viewports of visualizations (C3, **HS** "Navigate", "Organize").

Table 1: Taxonomy of interactive dynamics for visual analysis [14]

Data & View Specification	**Visualize** data by choosing visual encodings. **Filter** out data to focus on relevant items. **Sort** items to expose patterns. **Derive** values or models from source data.
View Manipulation	**Select** items to highlight, filter, or manipulate them. **Navigate** to examine high-level patterns and low-level detail. **Coordinate** views for linked, multi-dimensional exploration. **Organize** multiple windows and workspaces.
Process & Provenance	**Record** analysis histories for revisitation, review and sharing. **Annotate** patterns to document findings. **Share** views and annotations to enable collaboration. **Guide** users through analysis tasks or stories.

3.3 SQL Mapping

PD operates directly on relational-databases. To make a dataset accessible within PD it needs to be annotated with a small set of meta-data. This includes explaining relationships between different tables (cardinality of relationship, primary and foreign keys), providing some information about the columns of the tables (data-types, preferred visualizer, aliases and human-understandable labels if needed) and meta-data about the tables themselves (aliases). All the data-related operations within PD are mapped to their corresponding SQL function. PD is therefore a visual language to SQL that exposes a broad set of its expressive power (C2, **HS**: "Derive"). Database-joins are done implicitly when needed and rely on the meta-data information (i.e., primary-foreign key relationships between tables). Users do not need to struggle with the complexity of SQL-joins, but this also limits PD to only expose part of the full expressive power of SQL (i.e., only natural joins). Different join types or joins on non-key-columns of tables are not supported yet. We plan to expose that functionality as part of our future modifications.

Within the system each visualization is represented as an abstracted model. This model includes the data-attributes needed to construct the visualization, the transformation applied to them, the incoming filtering or brushing relations, the items currently selected in the visualization, as well as a few visual properties that do not affect the SQL layer (e.g., rendering style or color mappings). It is important to note that no data flows between linked visualizations; instead they share information contained in their abstract visualization models. Triggering operations that affect the abstract model forces a particular visualization to refresh by generating and executing SQL queries. PD does not perform any data-related computations in memory; all computations are delegated to the underlying database system.

3.4 Schema-Viewer

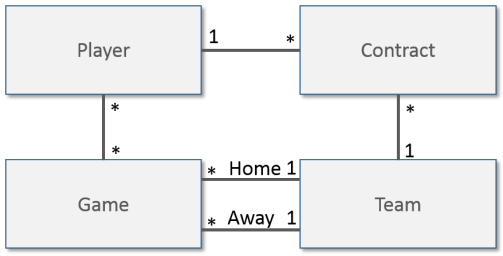

Fig: 3. Simplified database-schema for sports statistics

Users need to be able to access attributes of the data-set. While being widely used and powerful, relational databases come with the potential caveat of having a complex schema that are hard to grasp for untrained users. Finding an intuitive and approachable way to expose the structure of the underlying database-schema has been a challenging aspect of the PD's design. Since database-schemas are essentially graphs, there could be multiple ways to "connect" two attributes from different tables. This ambiguity needs to be solved in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

order for the system to automatically assemble SQL joins. Consider the database-schema in Figure 3. Imagine a user wants to create a visualization that displays an attribute from the "Player" table and an attribute from the "Team" table. Without specifying a path that encapsulates how the two tables should be connected the system does not know which of the three possible options the user intended (Player – Contract –Team, Player – Game – Team (Home) or Player – Game – Team (Away)). Each of the paths has a semantically significant different meaning. The same problem arises and is further complicated when linking two visualizations together. In the case that a user is looking at two visualizations - one that displays a list of players and one that displays a list of teams - the user must connect the two together to perform a filter or brushing operation. Without further specification the meaning of this connection is again ambiguous.

In our current version we address this problem by presenting the database-schema in a tree-view that is always rooted at the same table. If a user now drags an attribute out of this tree-view the path of how to connect the root table to the table from which the attribute comes is unambiguous. The same applies for linked visualizations. Because they are initially created by dragging attributes from the schema tree-view we can always use the root-table as the lowest common denominator when performing SQL joins. This approach offloads the problem to choosing a root table. Databases that are laid out in star-schema form, with a single fact-table, are therefore a naturally good fit for PanoramicData. It intuitively makes sense to use such a fact-table as the starting point and therefore as the root element in our schema-viewer. In other cases, we let the user choose a root-table at the beginning of an exploration or analysis session. Our current prototype does not support re-rooting during a session and therefore diminishes some of the expressiveness of SQL in order to reduce the cognitive burden on users.

The schema-viewer in PD (Figure 2 (a)) is brought up by a double-tap gesture anywhere on the 2D canvas. This allows users to quickly gain access to the underlying dataset in whatever context they currently are. Dragging and dropping data-attributes from the schema-viewer is used throughout the system to create or modify visualizations.

3.5 Data-Transformers

Data-transformers, such as group-by aggregates (e.g., sum, count, max, min, avg) or sorting operators, are applied directly to every data-attribute in PD and can immediately update the rendering of a visualization (C2, **HS**: "Derive"). PD exposes data attributes wherever it makes intuitive sense for a given visualization. Scatter-plots for example place visual handles to their data-attributes on the X- and Y-axis, whereas table-viewers expose them as column-headers. Common transformers can be triggered through a set of gestures (see section 3.2) but are also accessible through a more traditional radial-menu for discoverability.

Additionally each visualization exposes two drop targets: a group target and a color target. Dropping attributes on the group target specifies the context of aggregate calculations (e.g., what are we summing over?), whereas the color target is used to specify coloring of data-points within the visualization.

3.6 Calculated Fields

PD offers support to use well-known mathematical notations [37] to create calculated fields from any attribute (C2, **HS**: "Derive"). This functionality is exposed through the schema-viewer. A calculated field is no different than other attributes and can be used in exactly the same way as data-attributes to create or modify visualizations or to calculate group-by aggregates. PD further allows the user to create new attributes by transforming a chain of visualizations into a Boolean attribute that indicates if a data-record is part of this sub-set. This technique, described as tagging in the introduction use-case, is useful to create custom groupings within a visualization or to condense a complicated filtering operation into a single attribute. Again, such set-attributes can be used as any other data-attribute, for example for coloring a graph or for filtering or brushing a visualization.

3.7 Zoomable Canvas

PD features an unbounded pan- and zoom 2D Canvas in which visual elements can be arranged in a free-form fashion (C3, **HS**: "Coordinate", "Organize"). Such a 2D canvas offers a couple of advantages. Firstly, users can manifest their elements in a way that matches their mental model. Secondly, logically corresponding elements can be arranged spatially near each-other without forcing them into a limited area. Furthermore, open space to fluidly explore what-if scenarios or tangentially related data-attributes is available within a set of simple pan and zoom gestures. Finally, this whiteboard metaphor offers an intuitive way to label findings or important parts of the exploration or analysis process by using handwritten annotations on the background (**HS**: "Annotate"). Figure 1 (c) provides examples of such annotations. This approach also offers a way to record the user's exploration history. The filter-chains that are constructed during the exploration process can be conveniently revisited by locating them on the 2D canvas (**HS**: "Record"). While this is not as structured or automated as [13, 26] it enables users to decide which part of the exploration history they want to keep and allows them to layout and annotate exploration histories in a free-form fashion.

Even though PD works well on pen-enabled tablets, it leverages this whiteboard metaphor best when used on an interactive whiteboard. In a single-user scenario the extra screen-real-estate is useful for exploring visualizations in full detail. Furthermore, in collaborative-scenarios a whiteboard offers a natural way of compiling or discussing data insights (**HS**: "Share") and it turns PD into a tool to disseminate knowledge in presentation scenarios while benefiting from its interactive nature to quickly answer questions from the audience similar to [19].

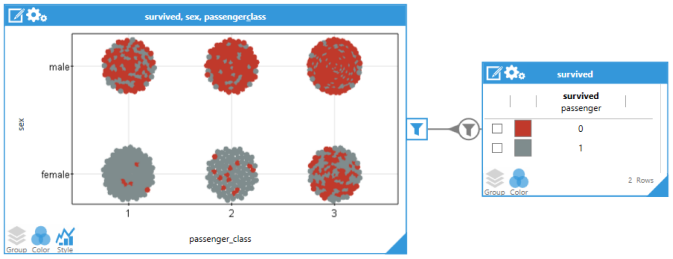

Fig: 4. Scatter-plot with interactive legend directly derived from plot.

3.8 Creating Visualizations

PD's central visual elements are visualizations and creating them is one of the most common tasks within the visual data analysis process (**HS**: "Visualize"). We encourage fast and easy creation of visualization through drag & drop interaction (C1). Every visual element that represents a data-attribute can be dropped anywhere on the 2D canvas to create a default visualization of its values. Similar to [20], the default visualizations are based on simple heuristics such as the data-type of the attribute and the number of unique elements in the data and can be changed by adding hints to the pre-defined meta-data information. A geographic attribute, for example, is displayed in a map-visualization whereas a categorical or binary attribute gets rendered as a histogram showing the distribution of its unique values. PD supports a manageable set of familiar visualizations such as pie-, bar-, scatter- and line-charts as well as table- and map-viewers, but other visualizations could be included. Each visualization can be customized through dropping data-attributes on pre-defined drop targets. 2D plots for example expose drop targets on their X- and Y-axis or on a special color shelf. On the other hand, a table -viewer

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

allows dropping of attributes anywhere on its column-headers to extend the columns they show.

Following our design concept of "Derivable Visualizations" (C1) led to some interesting design choices. For example we completely eliminate any sort of built-in legends within a visualization. Legends instead are visualizations themselves and can be derived from an existing visualization by dragging from its color drop-target. Legends are therefore fully interactive [24] and can be further customized if needed. This function works in part because PD offers consistent coloring of data-points across the system even for visualizations that are not linked. Figure 4 exemplifies such a legend.

Since Grammel et al. [12] showed that users of visualization systems often thought about data without any "processed" visual structures or visual attributes, we also include a gesture for deriving a table-view from any visualization (C1). Exposing the raw-data can serve as a valuable validation mechanism to avoid misinterpreting complex visualizations, and can be a more effective way to identify certain patterns or to perform computations.

3.9 Linking

Being able to coordinate views is a powerful tool that addresses a lot of different common exploration and analysis tasks (C4, **HS**: "Coordinate") In PD two visualizations can be linked through a one-way directed connection. Links indicate that the target visualization is influenced in some form by the source visualizations. A link can be in one of two states: filtering or brushing. When using filter-links the target visualization is filtered to the set of items that is selected in the source visualization. A visualization can have multiple incoming filter-links and the user can choose how to combine filters (AND or OR). Additionally the output of any filter is invertible (NOT). Thus PD offers functionally complete visual filtering logic. In Figure 2 (c), the bottom right pie-chart displays the distribution of salaries for people that are NOT married (selection in the bottom left pie-chart and inverted filter-link visible by the dashed line) AND are working for the federal government. Brushing-links are used to highlight the same data within a different context. In simple cases, for example when two scatter-plots showing the same data points but on different axes are linked together, this enables standard *brushing and linking*, where a selection in one chart highlights the corresponding data points in the other. Figure 2 (d) shows a more complex example. The top plot is brushed by the red pie-chart on the bottom. Notice that because the plot displays aggregated data-points (people are binned by age ranges) brushing is more nuanced. It reveals what percentage of the people within the aggregation would fit the query specified by the red filter. It is also important to mention that the red visualization is fully interactive and that a selection change would automatically update the dependent visualization. A user can switch between those two link states (filter or brushing) through simple touch or pen gestures.

This feature of PD provides a powerful tool for fluid data-analysis. Even just two simple visualizations of two attributes can be examined in a variety of ways. Consider a user trying to find an unknown relationship between two attributes, age and passenger-class, of passengers on the Titanic (Figure 5). The user starts with two visualizations (Figure 5 (a)), one showing the age histogram of all passengers and the other depicting their distribution according to passenger-classes. These are the default-visualizations for those attributes, which were attained by dragging the corresponding attributes from the schema-viewer. A pen gesture connects the two visualizations. Touching the slices of the pie-chart selects them and filters the age-histogram. After swiftly toggling through all the different passenger-classes the user thinks they noticed a slight shift in distributions for the first class (Figure 5 (b)). The user changes the link to a brushing-link and one now sees the age histogram of first class passengers relative to all passengers (Figure 5 (c)),. Indeed, it seems that there may be a shift in the distribution. With another pen-gesture the link flips in the opposite direction and the user pen lasso-select the passengers over 40. The little bars on the side of the pie-slices show the number of passengers within the class that the selection in the age histogram (Figure 5 (d)). It appears that older passengers were most likely in the first class.

Notice that PD allows configuration of these transformations within this scenario with one or two touch and pen gestures.

Fig. 5. Four ways to compose two visualizations to show different relationships between attributes. (a) No relationship. (b) Top filters bottom. (c) Bottom brushes top. (d) Top brushes bottom.

3.10 Copying and Snapshotting

PD offers two ways to create copies of visualizations. The first-one derives an exact live copy of the visualization that is fully interactive. The second-one takes a static snapshot of the visualization that can then be annotated (**HS:** "Annotate", "Record"). We included this feature based on feedback we received through our user-study. In some cases user wanted to "keep" or "save" a visualization and disable automatic updating through filtering or selection changes.

3.11 Selections

All visualizations manage selection states. Selections in PD are represented as queries over the data (**HS**: "Select"). For 2D plots those queries are represented as ranges over the space of the data-attributes. This allows the system to keep selections even if the underlying data has changed (e.g., through a filtering operation). To speed up the calculation of those selection queries, especially for multi-dimensional-charts (X, Y and color) with a large number of data-points, we use spatial data-structures (i.e., Octrees). 2D chart types, such as scatter-plots or maps, allow free-form lasso selection with the pen to allow for fine-grained selection control

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

3.12 Scalability

Our system currently works best for databases up to 30,000 records. Performance of rendering and database querying, especially for large visualization networks across multiple database-tables, drops to a non-interactive level for data-sets with over 80,000 rows.

4 Evaluation

We evaluated PD through a formative user study with both data and visualization experts. The goal of this study was to understand PD's utility and its approachability, to gain some insight about how its features are used and to explore the type of queries user pose. The study involved five participants, three PhD students (conducting research in the fields of Databases or Visualization) and two advanced Undergraduates (Teaching Assistants for a Data Science class). None of the participants had any prior knowledge of our tool.
The results of this study support our belief that PD, and the design-concepts it embodies, provides a fluid and intuitive user experience while still being expressive enough to answer the range of queries that users are likely to want to pose. The study also suggests that users are able to use the system with a minimal training amount.

4.1 Procedure

Our participants were given a 10 minute introduction to PD. In this introduction we used a small example data-set and gave the participants a brief overview of all the features and gestures in the system. After that we briefly allowed the participants to familiarize themselves with the tool and the UI, before we exposed them to the Titanic dataset. This dataset contains 14 attributes about the passengers of the Titanic [15]. We instructed the participants to freely explore the dataset and to use a "think-aloud" protocol. This open-ended exploration lasted 40 minutes on average. The examiner provided suggestion in cases where the users did not know what aspects of the data to look at. The participants were also instructed to explain and present any interesting findings to the examiner. At the end we asked them to reflect on their experience with PD and to answer a set of questions, either targeted towards specific features in PD (e.g., How did you like the pen and touch interface?) or more free-form questions about their experience with data-analysis tasks (e.g., Do you have any data, work or private, that you analyze or explore? What tools do you currently use to do so?).

4.2 Results

During the opened-ended exploration phase of our study all the participants were able answer the questions they posed with the help of PD. They created sophisticated queries even within the limited amount of time (Figure 6, 7). All users mentioned that they particularly liked the fact that visualizations can be linked together to create filters. However, only two of our participants made use of brushing-links. Two of the users initially mentioned that they have problems distinguishing when to use the pen and when touch. This confusion was cleared up by telling them that touch is used to "move" objects while the pen allows for shortcut gestures or fine-grained selections. All of the users valued the fluid nature of the UI. More specifically, they thought that the UI was not distracting them from the question they wanted to answer and that it was easy to switch between different tasks (e.g., creating a visualization and linking / filtering). The mnemonic pen gestures for fast access of data-transformation operations were rarely used, while others, such as scribble-delete, the linking of visualization gesture or lasso-selection, were adopted instantaneously by all users. Three of the participants mentioned that they forgot how to perform those mnemonic gestures but would have liked to use them. Discoverability of gestures is a point that we would like to address in future versions of the system [5]. Four out of five participants indicated that they would use PD for their own analysis tasks. One user pointed out that the lack of statistical hypothesis-testing and machine learning methods lessens the value of PD for his own work.

We observed that none of the users had any issues with the touch-gestures. Even without prior demonstration, most users would walk up to the whiteboard and naturally expected that they could drag visualizations around with their fingers or perform pinch-zoom gestures within graphs or maps. Users were able to decompose compound queries into sequences of simple linked visualization, which was a strategy we never explicitly explained. Users rarely created visualizations form scratch and instead modified visualizations that they derived from existing ones. The notion of deriving a visualization in order to display its legend was initially not clear to participants (e.g., "What do the colors in this graph mean?", "How can I display a legend for this graph?") but especially the reusability of those visualization-legends was appreciated after comprehending the concept ("Ah, now I can just use my legend to filter this other visualization.").

4.3 Anecdotal Insights

We would like to point to two anecdotal instances where participants used a table-view as a fallback solution. In one example our user tried to validate his hypothesis that people from less wealthy countries were more likely to be staying in the third class. He approached this by creating a map-visualization showing the passengers home countries, selected countries that he conceived as being less wealthy and linked it to a pie-chart showing the passenger distribution according to classes. The pie-chart showed that most of those passenger were staying in either the first or second class and therefore our user discarded his hypothesis. He noted that in the general population of all passengers is heavily biased towards the third class. Our participant wanted to find out where those third-class passengers came from. Different selections of countries still all offered a similar passenger class distribution (mostly first and second class). After quibbling with this for a while and expressing distrust in what the visualization was showing him, he decided to use a table-view to calculate the exact count of passengers per passenger-class and country. He then realized there was no record of where a majority of the passengers came from and that most of those unknowns were actually staying in the third class. We are planning to address this flaw (not dealing with unknown values properly) in a future version.

In a second example, this time from our pilot-study, a user wanted to display average survival rates for passengers in different age-bins. Furthermore he wanted to see two series that were colored differently for male and female passengers. He expressed concerns that he did not know how to create this plot. However, he said that he knew exactly what data he would like to have plotted. He started to create a table-view with the two columns and applied the appropriate operations to transform the data and color the rows. After finishing, his table-view depicted the data he was interested in and he realized that it should be simple to plot it now ("Maybe I can actually just plot those two columns against each-other…"). He used the axis-gesture, labeled the X- and Y-axis by dragging and dropping columns from his table and got the visualization he initially intended (Figure 7).
Those two anecdotes provide some interesting insight. They hint at a concept discussed by [12] and that we summarized as "Data Duality". Users frequently think about data in its raw form and tend to separate data from visual elements. In cases where they either do not trust a visualizations or have problem formulating a visualizations they use raw-data views as a fallback approach.

5 Conclusions and Future Work

PD supports a limited set of chart types (bar-, pie-, scatter-, line-charts, table- and map-views), but it could be extend to include others. The current version PD also only exposes a subset of data-operations. Calculated fields in PD are based on mathematical expressions but there is no way to specify more involved operations

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2346293, IEEE Transactions on Visualization and Computer Graphics

(e.g., if, else statements). Additionally, users within PD are limited to compute aggregates over either distinct groups or evenly-spaced bins. More complex grouping schemes such as user-defined or data-type specific groups (e.g., grouping by state or county in geographic attributes or grouping by day, month or year in dates) are not supported yet. There are some exploration interface patterns (e.g., small-multiples of charts [30]) that are used commonly in the set of tasks that PD tries to support. While users in PD could technically layout such interfaces manually through a set of view-creation, linking and filtering operations, it would be a cumbersome task. We are planning to include shortcuts to such patterns in future version so PD.

While some more complete or commercial systems, like Tableau, do address some of the limitations mentioned above, they often lack support for simultaneous viewing and creation of custom exploration interfaces through flexible view-coordination. Furthermore they do not expose a model to incrementally build up complex and modifiable queries where intermediate views serve as interactive handles.

In this paper we introduced PanoramicData (PD), a novel pen & touch interface for exploring and analyzing multi-dimensional data-sets. PD is based on a set of four core-concepts: *Derivable Visualizations*, *Exposing Expressive Data-Operations*, *Unbounded Space* and *Boolean Composition*. By unifying these concepts through a gestural pen & touch UI we are able support a wide range of common exploration and analysis tasks in a way that avoids explicit mode switching, minimizes UI cluttering, reduces indirection in the interface and provides effortless switching between the variety of interrelated tasks that data exploration and analysis require. PD allows for incremental and piecewise query specification where intermediate visualizations serve as feedback as well as interactive handles to adjust query parameters. A formative user study indicates that our approach provides a fluid and intuitive user experience while exposing a comprehensive set of tools to answer a wide range of data-related questions.

ACKNOWLEDGMENTS

The authors wish to thank Andries van Dam, David Laidlaw, Stan Zdonik and Joseph J. LaViola Jr. as well as the anonymous reviewers for their comments and suggestions. This work was funded by Microsoft Research.

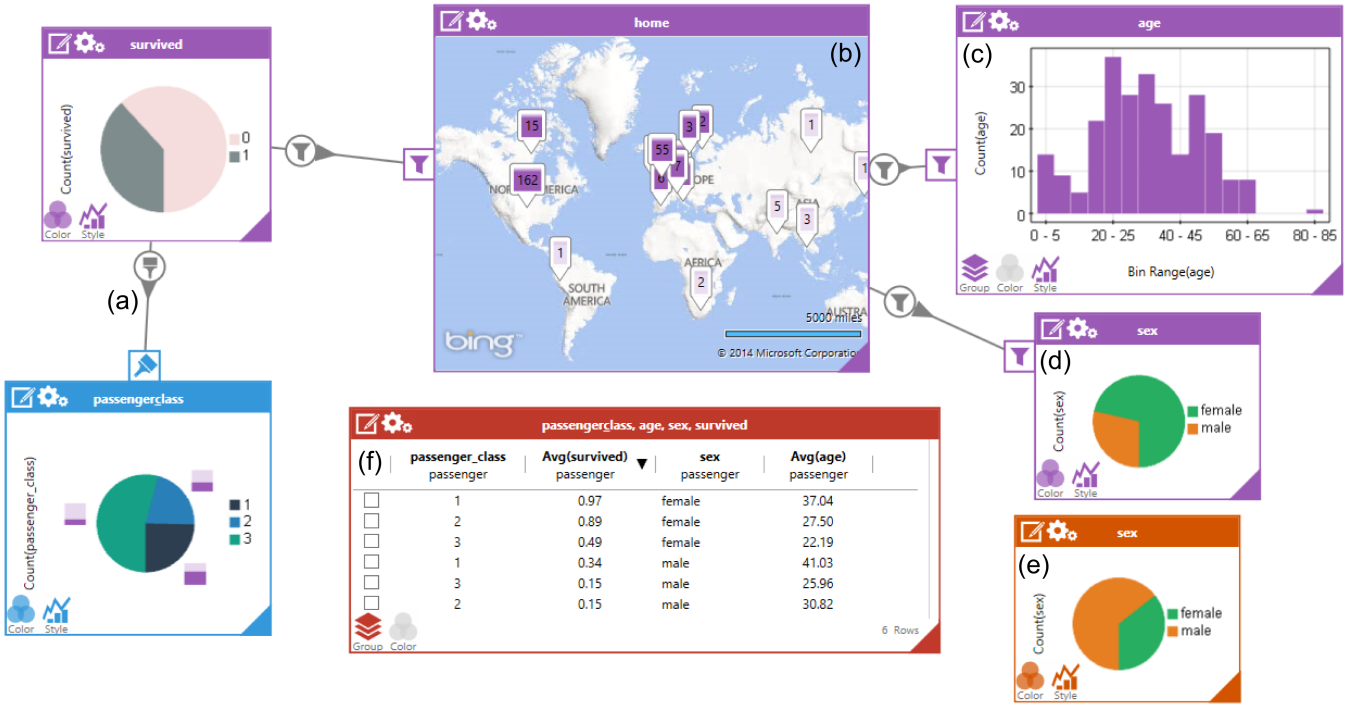

Fig: 6. Data panorama that a user built up during our evaluation to investigate different aspects of the Titanic data-set. (a) The user started of exploring the relationship between passengers that survived and their passenger class. (b) He then tested different hypothesis (i.e., are there correlations between survival and home towns of passengers (b), their ages (c) and their gender (d). He kept an unfiltered distribution of gender (d) to compare the ratios. (f) In order to obtain more accurate numbers and to confirm what he visually inferred, he built a table containing average survival rates for each passenger class and gender combination.

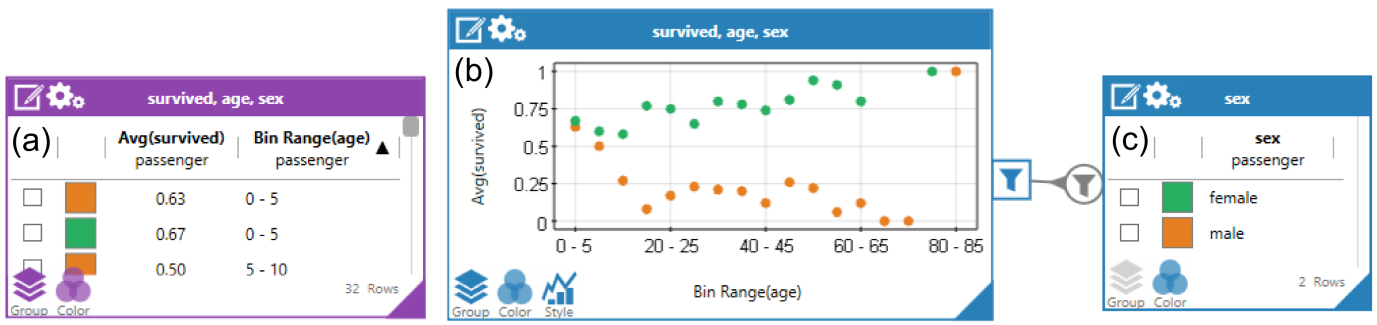

Fig: 7. An example from our evaluation where a user first described the data he wanted to see in tabular form (a) and then created a chart by dragging and dropping the column headers to the x and y axis (b). To understand the colors he derived a legend (c) from his char

REFERENCES

[1] Abouzied, A., J. Hellerstein, and A. Silberschatz, *DataPlay: interactive tweaking and example-driven correction of graphical database queries*. Proceedings of the 25th annual ACM symposium on User interface software and technology, 2012: p. 207-218.

[2] Amar, R., J. Eagan, and J. Stasko, *Low-Level Components of Analytic Activity in Information Visualization*. Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization, 2005: p. 15.

[3] Bederson, B.B. and J.D. Hollan, *Pad++: a zooming graphical interface for exploring alternate interface physics*. Proceedings of the 7th annual ACM symposium on User interface software and technology, 1994: p. 17-26.

[4] Bragdon, A., R. Zeleznik, S.P. Reiss, S. Karumuri, W. Cheung, J. Kaplan, C. Coleman, F. Adeputra, and J. Joseph J. LaViola, *Code bubbles: a working set-based interface for code understanding and maintenance*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2010: p. 2503-2512.

[5] Bragdon, A., R. Zeleznik, B. Williamson, T. Miller, and J. Joseph J. LaViola, *GestureBar: improving the approachability of gesture-based interfaces*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2009: p. 2269-2278.

[6] Browne, J., B. Lee, S. Carpendale, N. Riche, and T. Sherwood, *Data analysis on interactive whiteboards through sketch-based interaction*. Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, 2011: p. 154-157.

[7] Collins, C. and S. Carpendale, *VisLink: Revealing Relationships Amongst Visualizations*. Visualization and Computer Graphics, IEEE Transactions on, 2007. **13**(6): p. 1192-1199.

[8] Drucker, S.M., D. Fisher, R. Sadana, J. Herron, and m.c. schraefel. *TouchViz: A Case Study Comparing Two Interfaces for Data Analytics on Tablets*. 2013.

[9] Dunne, C., N.H. Riche, B. Lee, R. Metoyer, and G. Robertson, *GraphTrail: analyzing large multivariate, heterogeneous networks while supporting exploration history*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2012: p. 1663-1672.

[10] Elmqvist, N., A.V. Moere, H.-C. Jetter, D. Cernea, H. Reiterer, and T. Jankun-Kelly, *Fluid interaction for information visualization*. Information Visualization, 2011. **10**(4): p. 327-340.

[11] Elmqvist, N., J. Stasko, and P. Tsigas, *DataMeadow: a visual canvas for analysis of large-scale multivariate data*. Information Visualization, 2008. **7**(1): p. 18-33.

[12] Grammel, L., M. Tory, and M.-A.D. Storey, *How Information Visualization Novices Construct Visualizations*. IEEE Transactions on Visualization and Computer Graphics, 2010. **16**(6): p. 943-952.

[13] Heer, J., J. Mackinlay, C. Stolte, and M. Agrawala, *Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation*. Visualization and Computer Graphics, IEEE Transactions on, 2008. **14**(6): p. 1189-1196.

[14] Heer, J. and B. Shneiderman, *Interactive Dynamics for Visual Analysis*. Queue, 2012. **10**(2): p. 30-55.

[15] Hind, P. *Encyclopedia Titanica*. Available from: http://www.encyclopedia-titanica.org/.

[16] Idreos, S. and E. Liarou, *dbTouch: Analytics at your Fingertips*. Proceedings of the 7th International Conference on Innovative Data Systems Research (CIDR), 2013.

[17] Kandel, S., A. Paepcke, J. Hellerstein, and J. Heer, *Wrangler: interactive visual specification of data transformation scripts*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2011: p. 3363-3372.

[18] Lee, B., P. Isenberg, N.H. Riche, and S. Carpendale, *Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions*. Visualization and Computer Graphics, IEEE Transactions on, 2012. **18**(12): p. 2689-2698.

[19] Lee, B., R.H. Kazi, and G. Smith, *SketchStory: Telling More Engaging Stories with Data through Freeform Sketching*. Visualization and Computer Graphics, IEEE Transactions on, 2013. **19**(12): p. 2416-2425.

[20] Mackinlay, J., P. Hanrahan, and C. Stolte, *Show Me: Automatic Presentation for Visual Analysis*. Visualization and Computer Graphics, IEEE Transactions on, 2007. **13**(6): p. 1137-1144.

[21] Mynatt, E.D., *The Writing on the Wall*. IFIP Conference on Human-Computer Interaction, 1999.

[22] North, C. and B. Shneiderman, *Snap-together visualization: a user interface for coordinating visualizations via relational schemata*. Proceedings of the working conference on Advanced visual interfaces, 2000: p. 128-135.

[23] Rao, R. and S.K. Card, *The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1994: p. 318-322.

[24] Riche, N.H., B. Lee, and C. Plaisant, *Understanding Interactive Legends: a Comparative Evaluation with Standard Widgets*. Computer Graphics Forum, 2010. **29**(3): p. 1193-1202.

[25] Shneiderman, B., *Visual User Interfaces for Information Exploration*. 54th Annual Meeting, Washington, DC, October 27-31, 1991., 1991.

[26] Shrinivasan, Y.B. and J.J.v. Wijk, *Supporting the analytical reasoning process in information visualization*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2008: p. 1237-1246.

[27] Stolte, C., D. Tang, and P. Hanrahan, *Polaris: a system for query, analysis, and visualization of multidimensional relational databases*. Visualization and Computer Graphics, IEEE Transactions on, 2002. **8**(1): p. 52-65.

[28] Streit, M., H. Schulz, A. Lex, D. Schmalstieg, and H. Schumann, *Model-Driven Design for the Visual Analysis of Heterogeneous Data*. Visualization and Computer Graphics, IEEE Transactions on, 2012. **18**(6): p. 998-1010.

[29] Tobiasz, M., P. Isenberg, and S. Carpendale, *Lark: Coordinating Co-located Collaboration with Information Visualization*. Visualization and Computer Graphics, IEEE Transactions on, 2009. **15**(6): p. 1065-1072.

[30] Tufte, E.R. and P. Graves-Morris, *The visual display of quantitative information*. Vol. 2. 1983: Graphics press Cheshire, CT.

[31] Upson, C., T.A. Faulhaber, Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. Van Dam, *The application visualization system: a computational environment for scientific visualization*. Computer Graphics and Applications, IEEE, 1989. **9**(4): p. 30-42.

[32] van Dam, A., *Post-WIMP user interfaces*. Commun. ACM, 1997. **40**(2): p. 63-67.

[33] Walny, J., S. Carpendale, N.H. Riche, G. Venolia, and P. Fawcett, *Visual Thinking In Action: Visualizations As Used On Whiteboards*. Visualization and Computer Graphics, IEEE Transactions on, 2011. **17**(12): p. 2508-2517.

[34] Walny, J., B. Lee, P. Johns, N.H. Riche, and S. Carpendale, *Understanding Pen and Touch Interaction for Data Exploration on Interactive Whiteboards*. Visualization and Computer Graphics, IEEE Transactions on, 2012. **18**(12): p. 2779-2788.

[35] Young, D. and B. Shneiderman, *A graphical filter/flow representation of Boolean queries: A prototype implementation and evaluation*. Journal of the American Society for Information Science, 1993. **44**(6): p. 327-339.

[36] Yuan, X., D. Ren, Z. Wang, and C. Guo, *Dimension Projection Matrix/Tree: Interactive Subspace Visual Exploration and Analysis of High Dimensional Data*. Visualization and Computer Graphics, IEEE Transactions on, 2013. **19**(12): p. 2625-2633.

[37] Zeleznik, R., T. Miller, C. Li, and J.J. Laviola Jr. *Mathpaper: Mathematical sketching with fluid support for interactive computation*. in *Smart Graphics*. 2008. Springer.

[38] Zeleznik, R.C., A. Bragdon, C.-C. Liu, and A. Forsberg. *Lineogrammer: creating diagrams by drawing*. in *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 2008. ACM.