Interactive Visual Analytics of Big Data
A Web-Based Approach

Matthias Nielsen

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark
Interactive Visual Analytics of Big Data
A Web-Based Approach

A Dissertation presented to the Faculty of Science and Technology of Aarhus University in partial fulfillment of the requirements for the PhD degree.

By
Matthias Nielsen
January 31st, 2016
Abstract

The generation and availability of enormous amounts of data – Big Data – poses a momentous challenge for exploratory visual analytics. Especially when interactive exploratory visual analytics becomes available in web-browsers, which are ubiquitous and accessible to all types of users in companies and the public, most of whom are non-programmer users. This means that there is a great opportunity to enable exploratory visual analytics for big data to non-programmer users such as domain experts or private users with low experience with visual data analysis.

However, enabling exploratory visual analytics of big data for non-programmer users is a compound challenge in which visualizations needs to be apprehensible and available for non-programmer domain experts and private users in formats and on platforms that are usable by them. The key challenge is therefore, how to develop a new breed of efficient interactive visual analytics tools that enable non-programmers to perform iterative analysis and hypotheses generation and evaluation related based on Big Data.

A web based approach to exploratory visual data analysis for non-programmers is advantageous because the internet has shown to have a hitherto unparalleled capability to converge technologies, tools, workflows, mediums, applications, etc. This means that users are both accustomed to the platform and have come to expect to able to use it for a wide range of uses. However, making responsive and efficient interactive visual analytics on the web is a non-trivial challenge, which need to be addressed.

In my PhD project I demonstrate how a web-based approach for enabling non-programmer domain experts and private users to perform exploratory visual analysis of Big Data is both a technically viable and propitious solution. I do this in a three-pronged approach: (1) I develop and apply visualization techniques with high apprehensibility to facilitate exploratory visual analytics for non-programmer users. (2) I rigorously investigate and document performance impact of rendering SVG visualizations of large datasets in browsers and provide and evaluate techniques for optimizing rendering performance. (3) I develop novel direct-touch interaction with a low barrier of entry for fluid, yet sophisticated, data querying on multivariate interactive data visualizations.
Resumé

Generering og tilgængelighed af enorme mængder data – Big Data – udgør en betydningsfuld udfordring for værktøjer for interaktiv eksplorativ visuel dataanalyse. Især når interaktiv eksplorativ visuel dataanalyse tilgængeliggøres i webbrowsere, som er tilgængelige på adskillige platforme for alle typer af brugergrupper (f.eks. domæneeksperten og privatpersoner). Dog er de fleste brugergrupper kendetegnet ved at have lille teknisk ekspertise og kan karakteriseres som ikke-programmør. Det betyder, at der opstår en stor mulighed for at muliggøre eksplorativ visuel dataanalyse for ikke-programmør brugere, såsom domæne ekspert eller private brugere med lille erfaring med visuel dataanalyse.


En web-baseret tilgang til eksplorativ visuel dataanalyse for ikke-programmører er fordelagtig, fordi internettet har vist sig at have en hidtil uovertruffen evne til at konvergere teknologier, værktøjer, arbejdsgange, medier, applikationer osv. Det betyder, at brugerne er både vant til platformen og er kommet til at forvente at kunne bruge det til en bred vifte af anvendelser. Men en web-baseret tilgang repræsenterer også en udfordring i form af at gøre interaktiv visuel dataanalyse responsive og effektiv.

1 INTRODUCTION

Data is being generated and collected by companies, organizations, public sectors, etc. in unprecedented pace and quantities – and the pace and quantity is expected to only increase in coming years (Manyika, Chui et al. 2011). However, in order to facilitate exploratory analysis these immense amounts of data, often characterized as Big Data, the data must be made available and apprehensible for domain experts, as well as private users, who commonly are not programmers. Information visualization and visual analytics has the potential for making data available and apprehensible for users in many different application domains (Card, Mackinlay et al. 1999). Extending on this, the web has shown an unprecedented ability to converge media and technologies into a single, open-ended platform. With the emergence of standardized web technologies for creating rich visualization with sophisticated interactions, the web also has great potential to integrate exploratory visual analytics of big data in browsers. Facilitating exploratory visual analytics of big data in browsers is a momentous challenge because it has the potential to endow a large group of users with access to apprehensible data visual analytics tools. Users such as domain experts who can benefit from being able to visually analyze data to do data driven decision making, or private users who have a personal interest in e.g. open data for institutional or governmental transparency. However, enabling exploratory visual analytics of big data in browsers is a grand challenge because browsers exist on a multitude of device types with different performance and interaction capabilities.

1.1 Research Objectives

Recent years has seen advancements in the abilities for developing advanced interactive visual analytics tools in browsers. Standardized web technologies such as Scalable Vector Graphics (SVG) (Dahlstrøm, Dengler et al. 2011) have been
utilized to create advanced JavaScript libraries for creating interactive visualizations in browsers, including D3 (Bostock, Ogievetsky et al. 2011) and Reactive Vega (Satyanarayan, Russell et al. 2016). Furthermore, web-browser capabilities are developing rapidly by third party JavaScript libraries in general, such as JQMultiTouch (Nebeling and Norrie), which facilitates creating sophisticated multitouch interactions in browsers on touch-enabled devices.

My research objectives take advantage of existing technologies and capabilities available in web browsers to provide novel exploratory big data visual analytics in browsers for non-programmer domain experts and private users. In my work I have adopted a three-pronged approach:

1. To enable domain experts with low experience with visual analysis to do data driven decision making, I engage with end users to develop visualization techniques with high apprehensibility to facilitate domain experts to leverage their contextual expertise when performing visual analysis. Furthermore, dissemination of web-based visual analytics tools to domain experts and private users alike also requires the availability of applying them to datasets that users wishes to analyze.

2. Creating and rendering interactive visualizations of large datasets in web-browsers requires detailed understanding of performance—especially true for declarative SVG visualizations. Therefore, techniques for retaining responsiveness is essential for facilitating web-based exploratory visual analytics of big data.

3. Developing web-based visual analytics tools for domain experts and private users means that these tool will be used on devices familiar to such users. Particularly direct-touch enabled multitouch devices, which have high commercial penetration (eMarketer) and has been of increasing research interest in recent years (Schmidt 2015). This means that fluid touch interaction techniques with a low barrier of entry are needed for domain experts and private users untrained in visual data analysis.
How I unfold this three-pronged approach within the EcoSense project (http://ecosense.au.dk) will be the subject of the following section.

1.2 EcoSense Project
My research takes place within the EcoSense project (http://ecosense.au.dk), a joint research project between Aarhus University, private companies, municipalities in Denmark, and international universities. EcoSense aims to infer climate impacts from novel sensing, visualization, and analysis methods.

My contributions with the AffinityViz visualization technique (Nielsen and Grønbæk 2015) and applying AffinityViz to an application area (Nielsen, Brewer et al. 2016) are aimed at addressing resource consumption and eco-awareness in particular. This also includes my contribution in defining and describing the Computational Environmental Ethnography (CEE) methodology (Blunck, Bouvin et al. 2013). Furthermore, my work on PivotViz (Nielsen and Grønbæk 2015) and direct-touch interaction with multidimensional data visualizations (Nielsen, Kjærgaard et al. 2013) (Nielsen, Elmqvist et al. Submitted) is generally applicable work that applies to EcoSense subprojects as well as other application areas. This also goes for my work on performance optimization when creating and rendering SVG visualizations in web-browsers (Nielsen, Badam et al. 2016), which is a contribution that is elementary in retaining responsiveness in interactive web-based SVG visualization.

1.3 Hypotheses
My research objectives can be summed into the following three hypotheses. For my hypotheses I assume that relevant data is available in infrastructures capable of supplying it through data APIs, batch downloads, or other means if relevant.

**Hypothesis 1:** It is possible to provide interactive visual analytics with a high affinity between the object of research (e.g. a building or a road network) and a visual representation of data collected from the same object facilitates a contextual relatability, which makes visualizations apprehensible for domain experts.

**Hypothesis 2:** We can develop techniques for data aggregation, data sampling, and progressive rendering for SVG visualizations in browsers that can yield significant performance gains, which are pivotal to provide responsiveness in interactive SVG visualizations.

**Hypothesis 3:** Novel direct-touch interaction techniques can enable sophisticated data querying with simplified, fluid interactions for visual analytics characterized by having a low barrier of entry for domain experts and causal data analysts.

1.4 Research Method
In my work I have adopted an experimental computer science approach, depicted in Figure 3. Within this research approach, my research has taken offset in real world challenges, commonly in scope of the EcoSense project (section 1.2), but
sometimes I have expatiated my research to other application areas when needed e.g. for deployment of prototypes or evaluation.

I have explored my three research hypotheses (section 1.3) by applying an experimental computer science approach (Figure 3) to different real world challenges. Some challenges have resulted in multiple iterations of the approach generating several results, e.g. my contributions first with the AffinityViz technique (Nielsen and Grønbæk 2015) and since the application of the technique in a visual analytics tool (Nielsen, Brewer et al. 2016). In other cases, I have participated in iterations headed by collaborators, such as my contribution to the Computational Environmental Ethnography methodology (Blunck, Bouvin et al. 2013), where I have since ceased my contributions to pursue other projects.

In the remainder of this section I will elaborate on the iterative fashion I have conducted my research

1.4.1 Analysis and Design

The analysis and design steps in experimental computer science is a matter of understanding the application area by engaging with users who face these real world challenges. Such an understanding is hard won, and commonly requires multiple sub-iterations of continued analysis and design, which means that analysis and design are closely connected. For example, my contributions with AffinityViz (Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016) was preceded by numerous iterations of static, physical, and interactive visualizations that was demonstrated for domain experts. Most resulted in renewed analysis and design because the domain experts found the abstract visualizations too overwhelming and could not work them into their interdisciplinary workflow. Eventually,
however, a printed visualization folded to mimic a building’s layout received positive feedback, and it was further developed into a functional prototype and evaluated.

1.4.2 Implementation and Prototypes
Prototype development is pivotal in my research as instantiations of preceding analysis and design and because it enables me to subsequently evaluate my contributions. In my work I have, with few exceptions, implemented interactive SVG visualizations created and rendered in browsers using JavaScript. Sometimes my implementations and prototypes has utilized existing third-party JavaScript to create rich interactive SVG visualizations in browser. Third party libraries such as D3 (Bostock, Ogievetsky et al. 2011) creating SVG visualizations and JQMultiTouch (Nebeling and Norrie) for developing novel multitouch interaction techniques. However, as I in my research also have developed novel visualization techniques, I have also found it necessary to create visualizations from scratch by writing JavaScript that retrieves, parses, and interprets data and manually declares SVG elements in a browser’s DOM instance.

1.4.3 Evaluation
Evaluation is important to understand not only whether a design functions as intended but also whether designs and implementations are an improvement on current state of the art. Furthermore, evaluations are fundamental in providing insights for renewed analysis for subsequent iterations. Overall I have applied two different evaluation strategies depending on the type of my research contributions: (1) Commonly I have conducted user evaluations with domain expert end users, whom I targeted with my designs. Specifically, I have conducted with-in subject comparative studies, questionnaire surveys, and interviews with evaluation participants. Note that I in evaluations conducted with domain experts included domain experts that were not introduced to my designs in the analysis and design phase described in section 1.4.1. (2) In my paper on performance and optimization of creating and rendering SVG visualizations in browsers (Nielsen, Badam et al. 2016) I conducted simulation in order to conduct automated technical evaluations. By conducting simulations, I was able to isolate and detailedly document the impact of different factors on performance when creating and rendering SVG visualizations in browsers, as well as document the impact of applying techniques for improving performance.

1.4.4 Iterations
As depicted in Figure 3, experimental computer science is iterative, meaning that multiple full iterations can be conducted with subsequent iterations building upon results from preceding iterations. As mentioned previously, my work on AffinityViz (Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016) builds on iterations of this approach. Much of my other work also relies on multiple iterations of experimental computer science. My first publication, a poster and two-page published submission, addressed direct-touch brush interactions on parallel
coordinates visualizations (Nielsen, Kjærgaard et al. 2013). Despite its humble origin, this work has led to multiple subsequent iterations, which have resulted in a facet of contributions: PivotViz (Nielsen and Grønbæk 2015) refines and formalizes the visualization technique and my work on Scribble Query (Nielsen, Elmqvist et al. Submitted) is a novel direct-touch interaction technique for brushing.
2 APPREHENSIBLE INTERACTIVE VISUAL ANALYTICS FOR NON-PROGRAMMERS

In this chapter I examine how to make visual analytics apprehensible for a wide range of non-programmer users, including transient and occasional users. This is an important challenge because as visual analytics becomes ubiquitously available via web-based platforms, a diverse array of users become potential users. Furthermore, developing tools for apprehensible visual analytics is both a challenge of apprehensibility of visualization techniques as well as a challenge of availability of tools for visual analytics (e.g. as cloud service). To motivate this, I first describe a continuum of non-programmer data analysts, by distinguishing between three archetypes of non-programmer users of visual analytics applications. These three archetypes are end-user data analysts who rely on others to prepare data for visual analysis (e.g. business intelligence professionals), domain experts whose main job function is not data analysis (e.g. facility managers and librarians) and private users who have a casual personal interest in analyzing data. Common for these three types of non-programmer users is that they benefit from having the possibility to visually analyze data, but rely on others (e.g. colleagues or data journalists), because data is not available to them in an apprehensible format. Second I argue that apprehensible visualization techniques first and foremost should be easily relatable in order to enable domain experts to do visual analytics. This I exemplify with a novel visualization technique I have developed – AffinityViz – which uses a simplified 3D model of high-rise buildings to visualize detailed data collected with e.g. apartment or office granularity. Third, I propose and argue for reorienting visual analytics tools away from an application mindset and towards visual analytics as a service, which is important to provide widespread availability. I demonstrate this with the PivotViz tool, a generalization and extension of parallel coordinates visualization technique, which enables visual analysis of large multidimensional datasets. The PivotViz technique is generalizable and can be applied to a wide range of multidimensional data.

2.1 Visual Analytics for Non-Programmer Users

In this section I will motivate the need for apprehensible interactive visual analytics tools by describing a continuum of non-programmer users of visual analytics tools. I define non-programmers as users who do not have the technical abilities to prepare a dataset for visual analysis. I distinguish between non-programmers of visual analytics on two parameters: apprehensibility of visual representations and availability of visual analytics tools to analyze a given dataset. Apprehensibility consider whether a user understand a visual representation as relatable to an underlying dataset and the dataset’s context, which is pivotal in order to make the underlying data usable. This is important because if a user does not understand a visual representation(s) in a visual analytics tool, the tool ultimately falls short.

Availability considers whether users of a visual analytics tool have to rely on programmers to prepare data for analysis, e.g. by interfacing with databases,
extracting data from data APIs, merging or massaging unstructured data. This is important to keep in mind, because it means that data that is not readily available in visual analytics tools is off-limits for non-programmers.

2.1.1 Archetypes of Non-Programmer Users of Visual Analytics Tools

My argument is therefore, that in order to make visual analytics tools ubiquitously available, the visual analytics tools need to be apprehensible and available for non-programmer users. Apprehensibility is a holistic parameter that varies with users’ prior experience with visual analytics tools and understanding of a dataset’s context. Availability is a technical parameter that varies with users’ ability to use a visual analytics tool for a given dataset. As mentioned previously, I distinguish between programmers and non-programmers on whether a user has the ability to make data available to be visualized using a visual analytics tool. E.g. if a dataset requires parsing or massaging before it can be visualized using a visual analytics tool. In my work, I have focused on developing visual analytics tools for non-programmers, and in this sub-section I will elaborate three archetypes of non-programmer users of visual analytics tools – data analysts, domain experts, and private users. These three user archetypes are depicted in Figure 4. I consider non-programmers as users who require high availability because they are not able to prepare a dataset for visual analysis.

<table>
<thead>
<tr>
<th>Data Analysts</th>
<th>Domain Experts</th>
<th>Private Users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Background:</strong> Professional or trained data analysts.</td>
<td><strong>Background:</strong> Professional for whom data analysis is secondary to primary job function.</td>
<td><strong>Background:</strong> Individuals with little or no data analysis background.</td>
</tr>
<tr>
<td><strong>Profile:</strong> Business Intelligence professional in house or consultant.</td>
<td><strong>Profile:</strong> Many professions, e.g. facility managers, librarians, and anthropologists.</td>
<td><strong>Profile:</strong> Transient interest in analyzing data within a particular topic.</td>
</tr>
<tr>
<td><strong>Applications:</strong> Tableau, QlikView, PowerBI, Excel, etc.</td>
<td><strong>Applications:</strong> Prepared dashboards, domain tailored visual analytics tools.</td>
<td><strong>Applications:</strong> Dashboards, narrated visualizations (e.g. data journalism).</td>
</tr>
</tbody>
</table>

Figure 4. Three archetypes of non-programmer users of visual analytics tools.

Data analyst: Data analysts are users who work with data analysis professionally, e.g. in a business intelligence role as a consultant or in-house in a company. Many visualization techniques will be apprehensible to professional or trained data analysts. They are generally able to work with data import and preparation features offered by data analysis software applications, such as Tableau. However, if a data source is not available in a format recognized, e.g. unstructured JSON, they have low ability to change availability of data for visual analysis and therefore rely on others. I do not target these professionals in my work, but I include them in categorization because they are power-users of commonly available applications for visual analysis, such as Tableau (Hanrahan, Stolte et al. 2007), QlikView, PowerBI, Excel, etc. Using these applications, they either conduct data analysis themselves, or they prepare e.g. dashboards to enable other data
stakeholders to analyze data. Such stakeholders could be domain experts or private users.

**Domain expert**: Domain experts is a broad archetype encompassing users who work professionally in a given domain, using visual analytics tools as a contributory part of their job function. Many professions fit into this archetype – librarians, facility managers, anthropologists, public servants, business managers, etc. Common for domain experts is that they have little experience with visual analysis meaning they commonly require visual representations with high apprehensibility. Apprehensibility of a visual representation for a domain expert depends on the visual representation’s relatability to the context of use as well as the consistency of the visual representation across analysis tasks. Contextual relatability considers navigating a space between abstracted visualization techniques (e.g. bar charts and scatter plots) and context retaining visualization techniques (e.g. map based geovisualizations (Gao, Hullman et al. 2014, Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016)). This is important because domain experts are likely to be occasional users of a visual analytics tool and therefore the visual representation should relate to the context of the data. Consistency refers to whether a visual representation can be applied to multiple data contexts without significantly changing the data dimensions. For example, a bar chart is widely applicable for visualizing many datasets but loses consistency when data dimensions are changed because the perceiver needs to re-interpret the bar chart and distinguish it from previous views. Apprehensibility in terms of contextual relatability and consistency of visual representations is a theme I explore in depth for energy consumption analysis in my work on AffinityViz (Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016). Besides apprehensibility, availability is an important issue in visual analytics tools for domain experts, because domain experts have low ability to change a visual analytics tool’s availability for analyzing a dataset without assistance. However, domain experts can be in a good position to commission external assistance if their professional work with visual analysis necessitates analysis of a given dataset.

**Private users**: Private users encompasses users that out of personal, possibly transient, interest wishes to analyze a dataset. Private users commonly have low experience with visual analysis and, like domain experts, require visual representations with high apprehensibility. However, unlike domain experts, apprehensibility in visual representations for private users can only utilize data context for contextualization, because no assumptions can be made regarding private users background knowledge of the context. This means that visual analytics tools for private users cannot expect to rely on single visual representations and instead should enable users to shuffle between representations. Regarding availability, private users seldom have any ability to change the availability of a visual analytics tool to analyze a dataset, nor do they have resources to commission external assistance to change availability. This
means that private users are often are dependent on prepared dashboards or narrated visualizations such as data journalism. However, such scenarios require authored visualizations and exempts private users from exploring new datasets, such as the abundance of datasets that are becoming available through open data platforms.

2.2 Developing Visualization Techniques for Apprehensible Visual Analytics

Developing apprehensible visualization techniques is a matter of developing visualization techniques that are relatable for users in the context they wish to conduct visual analysis of data. In this section I elaborate and discuss making data relatable to a diverse group of users by using the data’s origin’s spatial context as structure in visual representations. I exemplify this with my work on the AffinityViz technique (Nielsen and Grønbæk 2015), which visualizes detailed building consumption data using a simplified rendition of the building as structure for the visual representation, not unlike Bertin’s topographical maps (Bertin 2010). Extending upon the AffinityViz visualization technique, I also include my work on a tool for exploratory visual analytics of energy behavior in buildings.

2.2.1 Highly Affine Visualization for Apprehensibility

AffinityViz (Nielsen and Grønbæk 2015) is a visualization method that utilizes a building’s spatial layout when visualizing detailed data collected from a building (e.g. electricity consumption per apartment or office). The basic idea behind AffinityViz is shown in Figure 5. When visualizing detailed data collected from a building, the AffinityViz method retains high affinity with the building by visualizing the data using an abstracted visualization technique (heat map, area map, or bar chart) that wraps around a simplified core structure mimicking the real building.

![Figure 5. Three visualization techniques – heat map, area map, and bar chart – visualizing detailed building data and extended to utilize a building’s spatial layout to increase relatability. In heat map color intensity signifies energy consumption, while in the area map and bar chart color solely signifies orientation to assist in correlating the three-dimensional and flat versions. Note that dimension descriptions and legends have been left out for brevity.](image-url)
This way AffinityViz provides a simple yet powerful contextual relationship between the visualization and the real building.

2.2.1.1 AffinityViz: AffinityHeat, AffinityArea, and AffinityBar

AffinityViz relies on dividing a building to its lowest common denominator – units – in terms of area of apartments, offices, rooms in the building. If the units are dissimilar, a further subdivision of units can be calculated and used as a lowest common denominator. These units are then converted into visual structures in visualizations, and positioned in a layout mimicking the spatial layout of the building in terms of floor level and room number. In Figure 5, units and data from is depicted with three different types of visual structures or visual dimensions. From left to right in the bottom row in Figure 5 data and units are encoded as color intensity of equally sized squares in a heat map, varying area size of squares in an area map, and extrusion of bars in a three dimensional bar chart. In the top row in Figure 5, the visualizations are wrapped around a cuboid core that functions as a simplified representation of the building’s layout. The simplified building model can be rotated continuously horizontally to reveal visual structures on all surfaces of the building model. In AffinityHeat and AffinityArea, visual structures are simply positioned on the simplified building model in three dimensions, corresponding to the unit’s position. AffinityBar extends upon this by also depicting data in three dimensions by extruding bars from a surface to represent data. It is important to note that although bars are represented in three dimensions, the individual bar only extrudes in a single dimension making comparison easier. This way, AffinityBar, effectively distorts the simplified building model to reflect patterns in data collected from units.

The case used in Figure 5, units are rooms in an apartment building and the visual structures encode consumption data from the apartments. This is related to the application case using AffinityBar I describe in section 2.2.2, units encode consumption electricity, cold water, hot water, and district heating. In that case using AffinityBar, the extrusion of bars is calculated in relation to the average consumption of all units in the building. The average consumption is always a regular cube and thereby units with higher than average consumption extrudes far out from the building structure’s core and units with lower than average extrudes only a little outwards.

2.2.1.2 Contextual relatability

The basic idea behind AffinityViz is to establish an explicit visual relationship between the data visualization and the real building from which the data is collected. AffinityViz does this by retaining the building’s structural hierarchy in three dimensions with floors ordered ordinal and units sequenced within floors around the perimeter of the core. Each unit is then represented as a visual structure, which encodes data collected from the unit. Thereby, AffinityViz fuses an abstract data visual data representation with a simplified representation of a real-world spatial layout of a building.
While wrapping visual structures around a cuboid might seem straightforward, the fusing of abstract visual data representation with a simplified real-world spatial layout creates high affinity between the data visualization as a whole and the original building. Likewise, AffinityViz establishes a relationship between the context of the visual representation (the real-world building) and the visual representation of data (the units encoded as visual structures). By establishing this contextual relationship, the visual representation increases relatability for domain experts who are familiar with the context (e.g., building managers of a particular building) and who wish to analyze yet are not trained data analysts.

The high relatability is crucial because AffinityViz is designed for enabling domain experts to perform exploratory hypothesis generation and evaluation of building consumption data. In section 2.2.2 I demonstrate how the AffinityBar method from AffinityViz is applied to a setting where an interdisciplinary working group have used it for analyzing and hypothesizing about consumption data from an apartment building. First, however, I will discuss the drawbacks of the AffinityViz techniques.

2.2.1.3 Limits of the AffinityViz Method
The high affinity and contextual relatability in AffinityViz does, however, come at a cost of overview in the visualization. This is caused by occlusion, which is a natural effect when objects are represented in three dimensions (Ware 2012). In AffinityHeat and AffinityArea (Figure 5), occlusion happens as a result of that maximum two surfaces are visible at any given rotation angle. Rotation of the building model can reveal visual structures on hidden surfaces, however, they user still needs to keep occluded visual structures in memory for comparison. Occlusion in AffinityBar (Figure 5), is a bit different because it not only positions visual structures in three dimensions but also represents data by extruding bars three-dimensionally. This makes high outlier visual structures very easy to identify, however, low outlier visual structures can be occluded by neighboring visual structures.

As elaborated previously, AffinityViz relies on dividing a building into lowest common denominator units. In buildings with a generally uniform floor plan, dividing a building into units is trivial. However, disparate units, special floorplans, or unorthodox buildings can be challenging to adapt to an AffinityViz representation. For example, it can be non-straightforward to represent corner units (e.g., apartments that face two surfaces of a building). In the examples in Figure 5 and in the application scenario in section 2.2.2, corner units are positioned on the face of the simplified building model with which they share the largest surface. Besides corner units, internal units in a building are not possible to visualize in the current versions of AffinityViz. One could imagine interactions that enables slicing of a building to reveal consumption patterns related to internal structures, which current be a future research direction.
Besides units in buildings, AffinityViz is not appropriate for visualizing consumption data for buildings with complex structures. For example, the Sydney Opera House would be virtually impossible to adapt a simplified building layout used in AffinityViz. Still, however, many multistory buildings such as apartment buildings and office buildings have simple floorplans adaptable to an AffinityViz representation.

From a technical perspective, applying AffinityViz to visualize data collected from a building necessitates a balanced and fine-grained data collection from that building. E.g. for resource consumption apartment buildings, measurements should be conducted for each apartment. Further subdivision of measurements (e.g. each electricity outlet in an apartment), however, is not applicable to AffinityViz.

In the following section I will elaborate on a deployment of AffinityViz with an interdisciplinary team working from a data driven ethnography perspective to analyze and understand energy consumption patterns in an apartment building.

2.2.2 AffinityViz Applied
In this section I will briefly elaborate on deploying AffinityViz (here the AffinityBar version) with an interdisciplinary team conduction data driven ethnography. Furthermore, I discuss how the contextual relatability of AffinityViz has assisted the team analyzing and hypothesizing energy consumption patterns in a real-world apartment building.

2.2.2.1 Deploying AffinityViz with an Interdisciplinary Team of Experts Analyzing Energy Behavior in an Apartment Building
AffinityViz was deployed with an interdisciplinary team consisting of anthropologists, a sociologist, engineers, and computer scientist who studied energy consumption behavior among residents in an apartment building. The interdisciplinary team collaborated in analyzing and understanding resident consumption behavior patterns based on quantitative consumption data and qualitative resident data in a fashion related to CEE (Blunck, Bouvin et al. 2013) and Grounded Visualization (Knigge and Cope 2006). The apartment building consists of 156 apartments from which temporally fine-grained sensor-data is collected detailing resource consumption of electricity, hot water, cold water, and district heating for each apartment individually. Furthermore, the interdisciplinary team collected qualitative data describing residents’ behavior and attitude towards resource consumption. In order enable the interdisciplinary team to conduct exploratory visual analysis of resource consumption data and qualitative data describing residents’ behaviors and attitudes I deployed an interactive version AffinityViz with the interdisciplinary team. The design and development process preceding the deployment consisted of multiple minor design iterations to investigate and determine the needs for the interdisciplinary team, in a fashion related to Goodwin’s et al. work (Goodwin, Dykes et al. 2013). A screenshot with
elaboration is shown in Figure 6. For further details of the application case I refer to my paper on applying the AffinityViz method (Nielsen, Brewer et al. 2016). The deployment of AffinityViz was intended to support the domain experts in understanding energy causes consumption, thus relating the work to Chetty et al. (Chetty, Tran et al. 2008) and Irwin et al. (Irwin, Banerjee et al. 2014).

2.2.2.2 AffinityViz as a Contextual Relatable Visual Analytics Tool for Domain Experts

The aforementioned interdisciplinary team was faced with the challenge of analyzing data interdisciplinarily. This was a challenge because all members of the team were domain experts (section 2.1.1) in their own right. The anthropologist understood resident behavior, the sociologist understood general resident patterns, the engineers understood quantitative data analysis, and the computer scientists understood large scale data management. Commonly the different domain experts would apply tools or techniques familiar within their profession, however, they found it hard to disseminate findings as well as include perspectives from other professions in their analyses.

I developed and deployed a visual analytics tool based on AffinityViz with the interdisciplinary team because all domain experts in the team were familiar the context of the case (aforementioned apartment building) because they already worked with it. By deploying AffinityViz with the team, the domain experts were able to apply both their experience with the context of the case and their professional expertise when collaborating with the other experts in the interdisciplinary team. Their experience with the context was utilized because
AffinityViz established a contextual relationship to the apartment building by using the spatial layout of the building in the visualization Figure 6. Furthermore, the AffinityViz visual analytics tool enabled the team to analyze data interdisciplinarily because the tool facilitated exploratory visual analysis of quantitative consumption data cross-referred with qualitative data describing the residents’ behavior and attitudes towards energy consumption.

The AffinityViz visual analytics tool did not substitute existing tools familiar to the domain experts, nor was this the purpose of deploying the tool with the interdisciplinary team. It did, however, enable the interdisciplinary team to conduct data analysis and hypothesis generation and evaluation in an interdisciplinary fashion. In the following subsection I will briefly demonstrate this with two examples of analyses and hypothesis conducted using the tool. These examples are reproduced from my paper on applying AffinityViz as a visual analytics tool (Nielsen, Brewer et al. 2016).

2.2.2.3 Examples of Analyses with the AffinityViz Visual Analytics Tool

Figure 7 shows an example where the AffinityViz visual analytics tool has been used to analyze higher than average electricity consumption in apartments. The time-period for electricity consumption has been narrowed down to three days between Christmas and new year to protrude baseline electricity consumption in apartments. By cross-referring electricity consumption data with qualitative data collected from the residents, the resident in the marked apartment in Figure 7 has reported high usage of a desktop computer. This is an interesting finding because it means that the resident is likely aware that he or she has a high consumption of electricity because it is a conscious choice.

Figure 8 shows consumption of district heating for all apartments. Two things in this figure are very conspicuous. First, there is very high variation in the

![Figure 7. Electricity consumption shown for three-day period between Christmas and new year 2014. Color intensity depicts how long time per week the resident uses a desktop computer. Only apartments whose residents have participated in a qualitative study are shown.](image-url)
consumption of district heating with some apartments consuming many times the average consumption and other apartments consuming very little or nothing. Second, all Apartments but one on the A-C surface of the building has very low district heating consumption.

2.3 Visual Analytics as a Service

In this section I propose the concept of visual analytics as a service, by which I mean to promote a focus on developing visual analytics tools for plug and play visual analysis of data. Visual analytics as a service is related to Elmqvist’s et al. (Elmqvist and Irani 2013), which proposes extreme ubiquitous availability of data analytics. Supporting visual analytics as a service requires generally declaring how mappings between data and visualizations can be done. I demonstrate this with further development I have made on my work with PivotViz (Nielsen and Grønbæk 2015). There I have adapted the concept of reflection, as known from object-oriented programming, to datasets and I use dataset reflection for visualization specification. This means that a visualization should require as little specification as possible and instead rely on reflecting visualization properties from the dataset. Thereafter, I motivate the need for visual analytics as a service by elaborating how it can be applied to open data. Open data is an interesting case, because many countries have launched open data initiatives (Janssen, Charalabidis et al. 2012), which aims to make all kinds of data publicly available. However, making open data available is not just a matter of technical availability, as discussed by Berners-Lee (Berners-Lee 2009), it is also a matter of matter making the data available for programmers and non-programmers alike.
2.3.1 Data Reflection for Visualization Specification

Data reflection for visualization specification is inspired from the reflection concept known from object-oriented programming languages (Ferber 1989). By data reflection I mean relying on runtime examination to derive a specification for a visualization, rather than requiring an explicitly declared of a visualization. In its extremity data reflection for visualization specification proposes specification-less visualization, meaning that all properties for a visualization are derived from examining a dataset that is to be visualized. However, in some cases a minimum of visualization specification is necessary, e.g. in AffinityViz as I will elaborate momentarily.

By using data reflection for visualization specification, onus is placed the data structures that are to be visualized. This means that despite that many data structures are easily reflected technically, commonly datasets should be published in a reflectable variant.

2.3.1.1 Data Reflection Application Cases

I have adapted my work with PivotViz (Nielsen and Grønbæk 2015) to function as a fully reflected visualization tool. In practice for PivotViz, this is done by parsing a raw CSV file, using column headers as dimension titles and iterating through the dataset to determine dimensions’ extents and subdivisions. The current implementation of PivotViz relies on pre-computed discretization of continuous data dimensions in order to visualize large datasets. This is because performing on-demand or client-side discretization is undesirable as it can require considerable computation power and network resources. Furthermore, some data dimensions might require deliberate discretization, for example if a continuous age dimension is discretized into age brackets it would be desirable if these age brackets matches conventionally used age brackets rather than e.g. automatically computed evenly distributed age brackets.

Figure 9. My adaption of PivotViz (Nielsen and Grønbæk 2015), which performs visualization specification from data reflection.
Because the visual representation in AffinityViz (Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016) is tightly coupled with a real buildings layout, AffinityViz requires a rudimentary visualization specification. In the case presented in 2.2.2 four parameters have specified for the visualization: the number floors of the building, the number of units on each sides of the building, the starting unit for continuous unit numbering, and a keying scheme for mapping data entries to apartments. For the latter I have simply used each unit’s floor and apartment number combination. For more complex building layouts more parameters are likely to be required, e.g. if apartments are not positioned consecutively.

2.3.2 Visual Analytics as a Service for Availability of Open Data

Many countries have launched open data initiatives (Janssen, Charalabidis et al. 2012) for a variety of reasons – to instigate entrepreneurship, for institutional transparency, etc. Commonly, however, focus is on making open data technical availability of data, meaning that only data in itself is available with little or no means for making data available or apprehensible for users with no technical expertise. For example, Berners-Lee’s 5-star classification of open data focuses solely on technical availability (Berners-Lee 2009).

Making open data available and apprehensible is relevant for domain experts and private users alike. Both are user archetypes who do not have the ability to influence whether a dataset is available for visualization in a visual analytics tool, as discussed in section 2.1.1. Open data is highly relevant for discussing availability and apprehensibility for non-programmer domain experts and private users alike. This is because that a lot of open data comes from various governmental or other public institutions. This means that such open data is effectively owned by the public in the sense that a democratic government is employed by its electorate. Therefore, it is not sufficient to merely make open data technically available, it should also be available and apprehensible for the general public.

Enabling visual analytics as a service for making open data available and apprehensible is an issue of visual analytics tools that are available for being applied to data for analysis. My work on AffinityViz (Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016) together with my work on PivotViz (Nielsen and Grønbæk 2015) demonstrates that highly apprehensible visualizations and data reflection for visualization specification is both technically feasible as well as enabling for non-programmer users to perform exploratory visual analysis.

2.4 Conclusion

In this chapter I have discussed the challenges in addressing non-programmer users of tools for visual analytics. I have distinguished between two types of transient users – professional domain experts for whom data analysis only plays a minor part and private users who have a casual personal interest in analyzing data. To support these two types of transient users, I argue that tools for visual analytics should strive for high apprehensibility and availability. I demonstrate this with two
tools for visual analytics – AffinityViz and PivotViz. AffinityViz demonstrates high apprehensibility by visualizing data collected from high-rise buildings using a simplified 3D model of the real building, thus retaining a high affinity between data and the building where the data is collected. Lastly, I propose to reorient visual analytics tools towards visual analytics as service in order to promote availability of visual analytics. This I have argued for through the concept of data reflection for visualization specification, and motivated by arguing for that open data should be available and apprehensible all users, including non-programmers.

Making web-based tools for exploratory visual analytics apprehensible is, however, also a technical challenge, when web-based interactive visualizations should be responsive and scalable with large datasets. This I will investigate in the following chapter.
3 SCALABLE WEB-BASED INTERACTIVE VISUALIZATIONS

In this chapter I investigate how to make web-based interactive data visualizations of large datasets scalable with large datasets. I focus on SVG among multiple mediums for creating interactive visualizations in browsers. Web-based information visualization was first proposed by Rohrer et al. (Rohrer and Swing 1997), and has since been popularized by toolkits such as Protovis (Bostock and Heer 2009), D3 (Bostock, Ogievetsky et al. 2011), and recently Reactive Vega (Satyanarayan, Russell et al. 2016).

I first propose a categorization of mediums for creating interactive data visualizations in browsers. In this categorization I see SVG (vector graphics) as a medium distinct from four other mediums – basic HTML elements (e.g. img and div), HTML canvas element (bitmap graphics), JavaScript libraries that render to either SVG or Canvas (e.g. X3DOM), and proprietary plugins (e.g. Adobe Flash). Based on my categorization I, secondly, motivate that research is needed to make SVG fully scalable web-based interactive visualizations, because SVG visual structures are represented as DOM elements, which enable powerful selection-mechanisms and manipulations through CSS and JavaScript in modern browsers. However, because SVG visual structures are represented as DOM elements in a browser’s DOM instance, a browser’s interactive responsiveness will decrease as the number of manipulations of DOM elements increases. Third, because responsiveness is pivotal in exploratory visual analytics, I conduct an in-depth investigation of how browser rendering performance of SVG visualizations is influenced by the number and type of visual structures, and the pixel are size of the SVG visualization. Fourth, based on my investigation of browser rendering performance, I propose techniques for aggregating, sampling, and progressively rendering visual structures of a visualization in order to improve responsiveness of SVG visualizations in browsers.

3.1 Categorizing Mediums for Creating Web-based Interactive Data Visualizations

In this section I present a DOM-centric practitioner’s categorization of mediums for creating web-based interactive data visualizations, Figure 10. By DOM-centric I mean that the categorization takes offset in how the visualization elements are represented in a browser’s DOM instance. This is important because how a visualization appears in the DOM can have a large impact on a user’s experience.
E.g. in terms of backwards compatibility with old browser versions or whether the user need to install third party plugins for rendering a visualization. By practitioner I mean how developers of an interactive visualization is able to create a visualization for rendering in a browser. This is important because the choice of medium greatly impacts developers’ ability to implement and express visual structures and interactions. E.g. because SVG elements are declarative means that visual structures are represented as DOM elements, which enables powerful selections and manipulations using CSS and JavaScript.

**Basic HTML elements.** This category covers interactive visualizations created using HTML tags that precede elements designed for manipulatable web-based graphics. For example, a bar chart can be constructed using HTML div elements, a heat map can be constructed using a HTML table, etc. Furthermore, combining static images (png, jpeg, etc.) with a HTML image map element and alternative texts can be used to create interactive visualizations, albeit with a finite predefined number of pre-rendered possible states. However, commonly the more recent SVG or Canvas HTML elements are more appropriate than rudimentary HTML elements for creating interactive visualization, except e.g. if extreme backwards compatibility is necessary.

**SVG.** Scalable Vector Graphics (SVG) is a recent W3C recommendation for rendering 2D vector graphics in browsers (Dahlstrøm, Dengler et al. 2011). The SVG element is declarative, meaning that all elements, or visual structures (paths, rectangles, circles, etc.), in a SVG visualization appear in a browser’s DOM instance and therefore are rendered by client-browsers and the state of a SVG visualization is represented explicitly in the browser’s DOM instance. With the advent of SVG, JavaScript libraries, such as D3 (Bostock, Ogievetsky et al. 2011), have emerged that can assist developers in developing interactive SVG visualization that are created by manipulating client browsers. That individual SVG elements are represented as DOM elements means that SVG elements can be constructed and manipulated in client-browsers using JavaScript or be styled and modified using CSS. However, it also means that SVG visualization with many elements can result in slow rendering and unresponsiveness to interaction (although performance will differ across client devices and browsers). This especially becomes an issue with atomic visualizations of large datasets (Shneiderman 2008). To make SVG visualizations scale with large datasets, techniques are required for retaining a sublinear relationship between the number of elements in a dataset and the number elements in a SVG visualization. This I will cover later in this chapter.

**Canvas.** The canvas HTML element is also a recent W3C recommendation for rendering 2D bitmap graphics in browsers (Hickson, Berjon et al. 2014). The canvas element is imperative, meaning that graphics are created and represented as pixels in the canvas element and any state beyond the visual pixel state is maintained externally to the canvas element. It is, however, possible to extend the
canvas element to enable sophisticated visualization, e.g. using the widely supported WebGL API (Leung and Salga 2010).

**Custom DOM elements.** Custom DOM elements can be used for creating interactive visualizations by declaring elements in a Browser’s DOM instance that are not part of a W3C recommendation or specifications and therefore require external resources to render visualizations. For example, X3DOM (Behr, Eschler et al. 2009) is a JavaScript library that enables visualization developers to create declarative 3D visualizations using the X3D standard. Because X3D elements are not natively recognized by browsers, the X3DOM JavaScript library interprets the X3D format and renders the declared visualization to the HTML canvas element.

**Proprietary plugins.** The last category covers technologies that enable authoring and deployment of visualizations that renders an encapsulated element by using a third party plugin. Examples include Adobe Flash and Java, which both require a user to download third party software in order to display applications authored for these mediums. Earlier, such plugins have been momentous for authoring and deploying sophisticated interactive applications, however, they have fallen somewhat from grace in recent years.

The categorization in Figure 10 is DOM-centric meaning it focuses on how mediums that enable creating interactive data visualizations and thereby the categorization also traverses applications and libraries for creating interactive web-based visualizations. Libraries such as D3 (Bostock, Ogievetsky et al. 2011), which commonly is used for creating SVG elements, and Reactive Vega (Satyanarayan, Russell et al. 2016), which uses either SVG or Canvas for rendering visualizations.

In the following subsection I will explain the workflow when creating interactive visualization using SVG as well as motivate why creating scalable SVG visualization is an important issue.

### 3.2 Creating and Rendering Interactive Visualizations in Web-Browsers Using SVG

As depicted in Figure 10, SVG and canvas are the two W3C recommended standards for creating graphics in web-browsers. Both are technologies are capable of enabling rendering sophisticated interactive visualizations in browsers. However, the imperative nature of canvas and the declarative nature of SVG makes them very different in practice. In this subsection I motivate why I in my work I have focused on creating scalable web-based interactive visualizations using SVG.

### 3.2.1 Background for Scalable Web-based Visualizations

Scalable web-based interactive data visualization is an important issue to address because as data volume increases so does the need to visualize this data in accessible platforms. In his paper on Extreme Visualization (Shneiderman 2008)
Shneiderman distinguishes between atomic and aggregated visualizations when visualizing millions or billions of elements. In atomic visualizations each visual structure (dot, path, shape, etc.) represents a single data element while in aggregate visualizations a visual structure can represent many data elements. This means that naively visualizing large datasets using an atomic SVG visualization in a web browser means inserting equally many visual structures as DOM elements into a browser’s DOM instance. Taking another approach Fekete et al. (Fekete and Plaisant 2002) explored using techniques for hardware acceleration, stereovision, and animation for creating interactive visualizations of a million elements.

3.2.2 Basic Browser Workflow when Creating and Rendering SVG Visualizations

In this section I will briefly clarify the main steps involved when creating data-driven SVG visualizations in browsers. The main steps, as depicted in Figure 11, are (A) retrieving HTML documents, stylesheets, JavaScript files, data files, and other resources which are loaded by the browser into a DOM instance, (B) manipulating the DOM with JavaScript by inserting, updating or removing DOM elements, (C) the browser computes the style of visible DOM elements from stylesheets and inline element styling after which it computes the layout of the webpage, (D) finally, the pixel representation of DOM elements are painted by the browser. (E) Interaction (or e.g. a timed interruption) can occur after styling and layout computation is complete or during or after pixel painting, and can result in new manipulation of the DOM instance, possibly after retrieving new data. In some browsers it is possible to execute JavaScript or react on interactions while the browser is painting pixels. However, in the case of interactive SVG visualizations, this is likely undesirable because it could lead to discrepancy between a user’s perception of a visualization’s state and the visualization’s actual state, which potentially could lead to perceived non-deterministic interactions.

An important point to note in Figure 11 is the unidirectional procedure computing style and layout (C). This is important because when creating SVG visualizations in a client-browser by manipulating the browser’s DOM instance, the browser is unresponsive when computing style and layout of a DOM instance. This becomes especially important when creating visualizations with many visual structures, which I will cover with performance measurements in section 3.3. Furthermore,
the unidirectional procedure of (C) means that any optimizations for improving rendering performance must be performed before (C) at either a (A) or (B).

Manipulating a browser’s DOM by inserting elements triggers the browser to compute style of the elements, compute the layout of the webpage, and finally paint pixels. How the procedure of the steps occur differs between browser implementations, but these steps applies to all elements in the DOM that the browser renders. The number of DOM elements a browser can manage differs between browser implementations and hardware resources available to the browser. However, while some browsers under certain circumstances might be able to manage a few hundreds of thousands of DOM elements, most browsers can manage much fewer elements.

3.3 Challenges of Creating Rendering SVG Visualizations in Web-Browsers

In this section I examine performance measurements of rendering SVG visualizations in browsers, reported in depth in my paper with Badam and Elmqvist (Nielsen, Badam et al. 2016). I use these performance measurements to identify a distinct linear relationship between the number of visual structures (dots, paths, rectangles, etc.) in a visualization and the time taken by a browser to render the visualization. The linear relationship is due to that SVG is declarative and each visual structure therefore is rendered independently in the browser. That there is a linear relationship between the number of visual structures in an SVG visualization and the time taken to create and render it is significant because it means that atomic visualizations of large datasets can become infeasible.

Therefore, in perspective of the browser workflow described in section 3.2.2, I investigate how DOM manipulation time, styling and layouting time, and painting time is impacted when creating and rendering SVG visualization of large datasets in browsers. I first document the detailed linear relationship between the number and type of visual structures in a SVG visualization and the time taken to create and render the visualization.

I instrumentalize the results of my performance in two ways: First, because there is a distinct linear relationship between the number of visual structures in a visualization and the time taken to create and render the visualization, it is possible to construct a detailed linear equation for determining device-specific SVG creation and rendering performance. This can be used by developers and designers to pro-actively make informed decisions on upper limits for the number of visual structures in interactive SVG visualizations in browsers. Second, in the subsequent section I define and evaluate three techniques – sampling, aggregation, and progressive rendering – for retaining responsiveness when creating and rendering SVG visualizations of large datasets.
3.3.1 Test Setup

In this section I briefly describe test setup considerations for determining the detailed linear relationship between the number and type of visual structures in SVG visualization and the time taken to create and render the visualization.

The performance measurements are conducted with scatterplot and parallel coordinates visualization techniques. I have chosen these two visualization techniques because their visual structures (dots/circles for scatterplots and paths for parallel coordinates) differ fundamentally and as such will yield different results. Furthermore, in their basic instantiation scatterplots and parallel coordinates are both atomic visualization techniques, meaning that they retain a one-to-one relationship between the number of elements in a dataset and the number of visual structures (dots/circles or lines/paths) in a visualization. All scatterplots and parallel coordinates visualizations are of the same size (except when otherwise stated). Examples of the scatterplot and parallel coordinates visualizations used in my performance measurements are show in Figure 14. Each visualization is created and rendered with a random sample of data points from a multidimensional dataset with around 200,000 elements. The scatterplots visualizations are created and rendered with two dimensions from this dataset and parallel coordinates visualizations are created and rendered with five dimensions from this dataset. This means that a dot in a scatterplot only covers a small area while a line in a parallel coordinates visualization covers the full width of the visualization.

For further details on the test setup and data collection method, I refer to my paper written together with Badam and Elmqvist (Nielsen, Badam et al. 2016). All performance measurements reported in this section are from that same paper.
3.3.2 Rendering Performance of SVG Visualizations

In Figure 13, Figure 16, Figure 15, and Figure 12 the rendering time for 10,000 to 100,000 elements in scatterplot (dots) and parallel coordinates (paths) visualizations visualized atomically using our test setup. The rendering time is shown as the total time in Figure 12 as well as subdivided into time taken to manipulate the DOM in Figure 13, compute style and layout in Figure 16, and to paint pixels in Figure 15. From these figures we can derive a number of insights into performance considerations when rendering SVG visualizations in client browsers:

![DOM Manipulation Time](image)

Figure 13. DOM manipulation time in seconds (vertically) for scatterplots and parallel coordinates for 10k to 100k elements (horizontally).

![Example Visualizations](image)

Figure 14. Examples of a scatterplot (left) and parallel coordinates (right) visualizations created and rendered in my performance measurements.
Creation and rendering time of SVG visualizations increase linearly with the number of visual structures. That there is a linear relationship between the number of elements and the performance measurements in Figure 13, Figure 16, and Figure 15 is the most rudimentary observation as well as the most anticipated. Nevertheless, it is an important and operationalizable observation – the more elements encoded as visual structures in an SVG visualization the longer time it will take to render. The consequence of this relationship is that responsiveness when creating and rendering SVG visualizations in browsers will decrease when visualizations with many visual structures are created and rendering. As elaborated in section 3.2 and documented in Figure 13, Figure 16, and Figure 15, the time taken to create to create (Figure 13) and render (Figure 16 and Figure 15) SVG visualizations in browsers creating and rendering SVG visualization can make a browser irresponsive. These results means that a primary mechanism for improving rendering performance of SVG visualizations in browsers is to lower the number of elements being visualized. This is because the performance impact of visualizing a large number of elements can be measured already when manipulating the DOM (Figure 13) and followed through styling and layouting (Figure 16) to painting (Figure 15). In section 3.4 I will describe and evaluate aggregation and sampling techniques for increasing responsiveness by lowering the number of visual structures being rendered.
technique for retaining responsiveness while rendered full, sampled, or aggregated datasets. Besides the number of elements being visualized, the type of visual structures used in a visualization also impacts performance, which I will discuss in the next paragraph.

Creating and rendering time is higher for complex visual structures that cover more pixels. As previously mentioned, the reported performance measurements are conducted on visualization techniques with disparate visual structures – dots in scatterplot and paths in parallel coordinates. In our test setup, a dot in a scatterplot is represented as a circle and a line in a parallel coordinates visualization is represented as a path. In SVG, a circle is declared by a center coordinate set and a radius and a path is declared by a series (at least two) of coordinate sets. This means three things: First, a path is more complex to describe than a circle, and therefore takes longer to create in a browser’s DOM instance, as can be seen in Figure 13. Second, because a path has a more complex geometry than a circle, the layout computation time will take longer, which can be seen Figure 16. The performance measurements reported in Figure 16 also includes styling computation time, however, in our performance measurements the style for both circles in scatterplots and paths in parallel coordinates are the same – their color is steelblue. Third, because a path in a parallel coordinates visualization covers more pixels than a circle in a scatterplot, parallel coordinate visualizations takes much longer time to render than scatterplots Figure 15. It should be noted that availability of GPU hardware acceleration when rendering SVG visualizations can greatly reduce the painting time and thereby also reduce the difference reported in Figure 15. However, DOM manipulation, styling, and layout time will not be changed by hardware accelerated rendering.

The takeaway is that visualization designers and developers should carefully consider the type of visual structures used when creating and rendering SVG visualizations in browsers. However, since always choosing simple visual structures is infeasible, techniques are needed that facilitate retaining responsiveness when creating and rendering interactive visualizations with complex. I address this in section 3.4 by outlining and evaluating techniques for data aggregation, data
sampling, and progressive rendering when visualizing large dataset in browsers using SVG.

**Increasing the size of visualizations with visual structures that covers many pixels increases the rendering time.** Lastly, and related to the above takeaway, increasing the size of a parallel coordinates visualization increases the time taken to render the visualization, as can be seen in Figure 17. This is because, as mentioned previously, the paths in the parallel coordinates visualization covers the full width of the visualization, which leads to an increase in number of pixels being painted. Contrary, the rendering time of scatterplots is constant as size of the visualization is increased because an increase in size of a scatterplot only leads to larger spacing between dots and not more pixels to paint. This further motivates the abovementioned takeaway that techniques are needed that retain responsiveness when visualizing large datasets in browsers using SVG, as I will cover in section 3.4.

3.4 Techniques for Retaining Responsiveness when Visualizing Large Datasets in Browsers using SVG

In this section I propose and evaluate three techniques for retaining responsiveness when creating and rendering SVG visualizations of large datasets in browsers. The three techniques are data aggregation, data sampling, and progressive rendering and all three introduce a tradeoff between visualization detail and creating and rendering performance. Aggregation and sampling increases creation and rendering performance to increases responsiveness while trading in accuracy (aggregation) or detail (sampling). Progressive rendering attempts to retain accuracy, detail, and responsiveness by creating and rendering visual structures iteratively, which comes at a cost of increased total creation and rendering time.
I describe the data aggregation, data sampling, and progressive rendering techniques independently, after which I discuss the general applicability of the techniques. In my recent version of PivotViz (Nielsen and Grønbæk 2015), which has been utilized for my work on Scribble Query (Nielsen, Elmqvist et al. Submitted), I have implemented aggregation, sampling, and progressive rendering techniques for retaining responsive rendering and interaction. Therefore, I use this version as example when explaining the aggregation, sampling, and progressive rendering techniques. The dataset used in the particular instance in Figure 18 is a dataset describing material loans from municipal libraries in Aarhus, Denmark. The dataset includes information about loan location, loan time, material type, material language, loaner age, and loaner residence area. The raw dataset contains over 12 million entries and fills more than 500 megabytes.

3.4.1 Aggregation
Aggregation is the process of combining similar data elements into a single aggregated element and thereby lowering the number of visual structures needed to visualize a dataset. As elaborated in the previous section, fewer visual structures is highly desirable when creating and rendering SVG visualizations. My work on aggregation in SVG visualization is related to Novotný and Hauser’s work data discretization in parallel coordinates visualizations (Novotný and Hauser 2006).
When multiple data elements are aggregated into a single visual structure, the visual structure should reflect this by e.g. changing size, color, or perhaps shape. This way aggregation can assist in protruding patterns in data that might otherwise be concealed by overplotting. Aggregation does come at a cost of detail because some data elements will not be distinguishable by a user as the data elements constitute only a part of a visual structure.

The general trends in performance for visualizing aggregated datasets scatterplots and parallel coordinates visualization with 50,000 elements are depicted in Figure 20. Horizontally in both bar charts in Figure 20, the aggregation aggressiveness is reported as number of bins/categories on each data dimension. For these measurements, the bins have been dynamically computed as interval bins meaning that the bin have approximately an equal amount of entries each. These particular number of bins are not a relevant result by itself, however, the general trend, which shows that as aggregation becomes more lenient, the effect of aggregation is diminished. This can be seen by juxtaposing Figure 20 with Figure 12, which shows the time taken to create and render the aggregated scatterplot and parallel coordinates visualizations approximates time taken to create and render corresponding atomic versions.

Besides the general trends, Figure 20 shows that aggregation needs to be much less aggressive to improve rendering performance compared to atomic versions. This is because that the multidimensional data for the parallel coordinates visualizations has multiple (five) dimensions and as the number of bins on each
dimension increases the maximum number of possible combinations equals the number of bins on each data dimension to the power of the number of dimensions. This does not mean that this maximum will be reached when it is theoretically possible, however, it is still a severe exponential growth in possible combinations.

In my work with PivotViz (Nielsen and Grønbæk 2015) I have applied aggregation in two steps. The first step is pre-discretization of data before it is loaded by a client-browsers. This is necessary when datasets become very large — e.g. in the dataset used in Figure 18, the raw dataset would be infeasible to load and massage in client-browser. In the first step, the data dimensions that are to be included in the visualization are extracted and continuous dimensions are discretized — e.g. age is discretized in age brackets and time is discretized into year. This dataset is loaded into the client-browser when the visual analytics application is loaded. The second step takes place on-demand in the client-browser and happens when only a subset of data dimensions in the dataset needs to be visualized. When this happens, a copy of the loaded data is reduced by disregarding all non-relevant data dimensions and identical consequent simplified data entries are aggregated. When fewer than the maximum number of data dimensions are visualized, this results in a smaller dataset, which results in faster rendering performance, as elaborated and discussed in section 3.3.2. However, sampling and progressive rendering is also required for retaining responsive. In the following section I will discuss sampling.

3.4.2 Sampling
The idea behind sampling is similar to aggregation because it seeks to reduces the number of visual structures to create and render in an SVG visualization. Contrary to aggregation, which modifies a dataset, sampling reduces the number of visual structures by visualizing a subset of a dataset. Therefore, the performance impact of creating and rendering SVG visualizations from sampled data follows the linear relationship elaborated and discussed in section 3.3.2. Besides improving creation and rendering performance, sampling can reduce overplotting and clutter of visual structures, as discussed by Bertini and Santucci (Bertini and Santucci 2004) for
scatterplot visualizations, and Heinrich and Weiskopf (Heinrich and Weiskopf) and Ellis and Dix (Ellis and Dix 2006) for parallel coordinates visualizations.

3.4.2.1 Visual Budget for Sub-Sampling
Implementing sampling is not simply a matter of selecting a random subset of a dataset. Instead implementing sampling requires a carefully considered trade-off between the loss of detail when visualizing a subset of a dataset and the desired number of visual structures for retaining responsiveness. I propose the concept of a visual budget for dynamically calculating this tradeoff, which I have implemented in my recent work on PivotViz (Nielsen and Grønbæk 2015) depicted in Figure 18. A visual budget is an approximate number of visual structures desired in a visualization, which is sought evenly distributed on the dataset being visualized while preserving minimum and maximum extremes. In my recent version of PivotViz, which visualizes discrete or discretized data as elaborated in section 3.4.1, I define the left-most axis in the visualization as the dominant axis. The visual budget is then evenly distributed across bins on the left-most data dimension. Within each bin on the lest-most axis, the bins part of the visual budget is divided into three thirds: one third of the minimum extremes in the bin are picked, one third of the maximum extremes in the bin are picked, and one third of random entries in between the extremes are picked. When filters or selection in the visualization are changed, the visual budget is applied to the filtered dataset and a new subset of the dataset is sampled.

Although, a visual budget retains minimum and maximum extremes, applying it for retrieving a sample of a dataset risks omitting significant entries in a dataset. This is sought alleviated by recomputing a sample when filters or selections are changed, because an equal size sample of a filtered dataset will constitute a larger proportion of the dataset. However, if interactions are necessary to reveal significant elements, the risk is that these elements will not be discoverable, if the user never performs the necessary interactions.

3.4.3 Progressive Rendering
Progressive rendering is a technique that can be used for creating and rendering all data elements as visual structures without loss of responsiveness. This is done by completing multiple iterations of steps (B), (C), and (D) in Figure 11 when visualizing a dataset rather than performing the steps sequentially for an entire dataset at once. Creating and rendering an SVG visualization progressively therefore means that the process can be interrupted while a dataset is being visualized. This is especially useful if the visualization takes a long time to visualize if there are many visual structures, as discussed in detail in section 3.3.2. Progressive rendering is related to Fekete’s work on ProgressiVis (Fekete) and Stolper’s et al. work on progressive analytics (Stolper, Perer et al. 2014), which enables visualizing partial results from time intensive data queries. However, here I focus solely on performing progressive rendering to improve responsiveness of SVG visualization created and rendered client-side.
However, progressive rendering can introduce a longer total time taken to visualize a dataset because additional operations are performed at each visualization iteration. E.g. even 0.1 extra milliseconds per iteration means on extra total second for visualizing 10,000 elements, if each element is visualized independently. Commonly it is desirable to visualize elements in small batches because it will lessen extra overhead caused by progressively rendering an SVG visualization. However, such batches should not be too large because the rendering process is only interruptible at each iteration step.

In my recent version of PivotViz (Nielsen and Grønbæk 2015), depicted in Figure 18, I have implemented progressive rendering in conjunction with aggregation (discussed in section 3.4.1) and sampling (discussed in section 3.4.2). Technically progressive rendering is implemented by manually iterating a list of elements that are to be visualized. Each interruptible iteration (whether for each element or for a small batch of elements) is instantiated by setting a timeout which instantiates next iteration with a delay of 0 milliseconds. Because a function triggered by a timeout will be delayed until by possible ongoing tasks (Hickson, Berjon et al. 2014), the next iteration will be performed after steps (B), (C), and possible (D) in Figure 11 have been performed for the current iteration.

3.4.4 Multiple Techniques for Creating and Rendering SVG Visualizations in Browsers

The three techniques I have discussed in this subsection – data aggregation, data sampling, and progressive rendering – are techniques that aim at improving rendering performance and interaction responsiveness in SVG visualizations in browsers. Yet they do so in distinct ways and with different tradeoffs. Data aggregation trades in accuracy for by combining multiple data elements into aggregated data elements, which results in fewer visual structures, which again leads to improved SVG creation and rendering performance. Data sampling also reduces the number of visual structures, but trades in detail instead by picking a subset of data elements to visualize. Lastly, progressive rendering of visual structures retains interactive responsiveness by visualizing visual structures individually or in small batches, which introduces a slower complete creation and rendering time. In my recent version of PivotViz, depicted in Figure 18, I have implemented all three techniques to enable responsive interactive visual analysis of large datasets.

Using my recent version of PivotViz as example, I have explained and discussed implementation deliberations when applying the three techniques. Implementing all three techniques does also induce their respective down-sides (loss of accuracy and detail, and increase in total visualization creation and rendering time). However, together the three techniques enable client-side creation and rendering interactive SVG visualizations of large datasets.
3.5 Conclusion

In this section I have given an overview of categories of mediums for creating web-based interactive visualizations consisting of four categories – SVG for vector graphics, Canvas for bitmap graphics, JavaScript libraries that render to SVG or Canvas, and proprietary plugins. Based on this categorization, I have motivated that my research has focused on creating interactive visualization using SVG because SVG visual structures are represented in modern browsers as DOM elements, which enable powerful selection mechanisms and manipulations using CSS and JavaScript. In order to investigate performance when creating SVG visualizations from datasets in client-browsers I have conducted detailed time measurements of time taken to manipulate a browser’s DOM instance, time taken for the browser to compute style and layout, and time taken for the browser to paint pixels. Based on my detailed time measurements of rendering performance of SVG visualizations in client-browsers, I propose techniques for aggregating, sampling, and progressively rendering visual structures in SVG visualization in order to improve responsiveness. I have implemented and evaluated the methods for scatterplots, parallel coordinates, and node-link diagrams, and I discuss the techniques applicability for other visualization techniques as well. Hereby, I have demonstrated that SVG is an appropriate medium for creating scalable responsive interactive visualizations of large datasets in client-browsers.

The next chapter will elaborate on the different techniques and methods that I have developed to utilize this scalability to provide novel types of interactivity into Visual Analytics.
4 TOUCH INTERACTION WITH WEB-BASED VISUAL
ANALYTICS

As tools for visual analytics become available on web-based platforms, exploratory
visual analysis becomes possible on a wide range of device types, which support
new input and output technologies. This means that tools for visual analytics
should support these new interaction technologies and integrate new interaction
techniques. In my work I have focused on interaction techniques with touch
interaction technology on touch displays. I first examine input modalities for
interaction with display-based visual analytics applications. Based on this
overview, I motivate that I have investigated direct-touch interaction because it
supports low indirection and has a high penetration in consumer products and
therefore is widely available. Second, I demonstrate and discuss touch interaction
techniques for information visualization by first adapting the brushing interaction
technique to direct-touch interaction and since designing Scribble Query. Scribble
Query is a simplified touch-first interaction technique for creating sophisticated
data queries with the stroke of a finger.

4.1 Input Modalities for Display-Based Interactive Visual
Analytics Applications

In this section I will briefly examine the range of input modalities for display-based
interactive visual analytics applications (Figure 21). I focus on explicit input
modalities (and not technologies or techniques) to avoid discussing particularities
of implementations of a technology (e.g. capacitive vs. resistive touch displays) or
application specific techniques because a modality can be mapped to many
interaction techniques. Instead I use input modalities order to focus on the process
of mapping explicit human action to input in a virtual application. Furthermore, I
focus exclusively on input modalities for display-based output because high
resolution monitors facilitate rendering data visualizations with high detail. The
high precision of screens are possible because the human eye is capable of
perceiving great detail on multiple visual channels (Ware 2010). Thereby, other
output modalities used for creating data representations, such as data
physicalization (Jansen, Dragicevic et al. 2015, Taher, Hardy et al. 2015), is not
covered by this categorization. I use this categorization of input modalities to
subsequently situate my own work, where I have worked with direct touch input
for display-based interactive visualizations, as well as discuss the strengths and
weaknesses of using direct touch as an input modality (for display-based
interactive visualizations).

4.1.1 Categorizing Input Modalities for Display-based Input

The input modalities in Figure 21 are categorized according to the indirection
(horizontal) and precision (vertical) that the particular input modality commonly
induces. I position input modalities in quadrant categories rather than on
continuous dimensions in order to simplify my categorization. Input modalities are categorized in quadrants where they commonly are actualized. E.g. direct touch is categorized as having low indirection and low precision. However, particular interaction techniques can alter the precision and indirection of the input modality, such as the precision-handle interaction technique (Albinsson and Zhai 2003).

I have adapted the indirection dimension in Figure 21 from Instrumental Interaction interaction model (Beaudouin-Lafon). In Instrumental Interaction, indirection denotes both spatial (physical offset between input and output) and temporal (delay between input and output) indirection. However, in my categorization I only refer to indirection as spatial indirection because I want to encapsulate how an input modality situates the user in relation to the display. The indirection dimension is also related to Lee’s et al. Interaction Distance sub-dimension in their design considerations for information visualization interaction (Bongshin, Isenberg et al. 2012). The indirectness dimension is not an implicit argument that low indirectness is always desirable. E.g. direct touch input on a wall-size display could be an inconvenient input modality if it hinders overview, although this is not necessarily the case as direct-touch enabled wall displays definitely can support collaborative work (Jakobsen and Hornbæk 2014).

The vertical dimension in Figure 21, precision, is directly related to my delimitation to only focus on display-based output. Modern displays easily have millions of pixel, which in turn enables rendering high granularity data visualizations. However, there is a large variation in how precisely input modalities can map a user’s explicit input to an interaction with an interactive visualization. This is not to say that an input modality with high precision is always better, because the visual structures in a visualization technique (e.g. rectangles in a bar chart or dots in a scatterplot)
influences to which extent high precision is required. The precision dimension is related to Lee’s et al. Input Resolution subdimension in their design considerations for information visualization interaction (Bongshin, Isenberg et al. 2012).

4.1.1.1 Quadrant 1: High Indirection and High Precision
Input modalities with high indirection and high precision are input modalities, whose input takes place away from the output display, yet facilitates precise interaction. In this quadrant we find conventional input technologies (mice, keyboards, and controllers) designed for fine grained input that match high resolution computer displays. These input modalities have high precision meaning that they have near pixel-precision (in case of mice or analogue sticks) or execute specific commands in response to keystrokes. However, they also have high indirection because physical interaction instrument in the interaction (mouse, keyboard, controller), is situated spatially separated from the display. Mouse and keyboard based input combined with display output in a classic desktop setup is a widespread interaction mode in information visualization systems (Bongshin, Isenberg et al. 2012).

4.1.1.2 Quadrant 2: Low Indirection and High Precision
Input modalities with low indirection and high precision are input modalities, whose input takes place on the output on output displays while also facilitating precise interaction. The only input modality I have categorized as having low indirection and high precision is pen-based touch input inputted directly on a display. However, other interaction technologies that use a physical instrument for expressing input to a display with similar precision would also fit in this quadrant. In recent years, pen-based direct touch input has been investigated in information visualization research in recent years – e.g. NapkinVis (Chao, Munzner et al. 2009), SketchInsight (Lee, Smith et al.), and PanoramicData (Zgraggen, Zeleznik et al. 2014).

4.1.1.3 Quadrant 3: Low Indirection and Low Precision
Input modalities with low indirection and low precision are input modalities whose input takes place on output on the output display, however, with low precision in mapping input to the output display. In this category, I have only positioned direct touch input as an explicit input modality for interacting with display based out. Experimental input modalities, such as eye tracking and brain wave tracking could fit into this category as well. However, because they commonly applied for implicit interaction (e.g. gazing or concentration), they are excluded from this model. Direct touch input, on the other hand, has received a considerable amount of attention in information visualization interaction research in recent years. Examples include TouchWave (Baur, Lee et al.), TouchViz (Drucker, Fisher et al.), DimpVis (Kondo and Collins 2014), as well as my own work on direct touch selection (Nielsen, Kjærgaard et al. 2013), and Scribble Query (Nielsen, Elmqvist et al. Submitted). I have also positioned speech input in this quadrant. It could be argued that speech input has high precision when it is used to trigger specific
commands, such as visualization creation or manipulation in Articulate (Sun, Leigh et al. 2010). However, because such speech input relies on parsing verbally articulated commands, such interfaces have a limited body of possible interactions. Furthermore, because speech input can be registered both in close proximity to, and far away from, an output display, it could be placed in both quadrant 3 and 4. However, I have positioned speech in quadrant 3 because it has the ability to facilitate interaction in close spatial relation to an output display.

4.1.1.4 Quadrant 4: High Indirection and Low Precision
Input modalities with high indirection and low precision are input modalities whose input takes place physically separated from output on the output display, yet also having low precision in mapping input to the output display. In this category we find indirect touch (similar to direct touch, except the touch sensitive surface is separate from the display), body tracking (camera or wearable tracking of body or body-part movement), and speech input. In-direct touch interaction has the advantage that because the touch input surface is separate from the output display, fingers or interaction instruments does not obstruct the view of the visualization interface. An example is Kosara’s work where a touchpad is used to facilitate multitouch input on a parallel coordinates visualization (Kosara, Kosara). Regarding body-tracking, examples include Jakobsen et al. who track a user’s spatial distance to a wall-size display to interact with visualizations (Jakobsen, Sahlemariam Haile et al. 2013).

4.1.2 The Schism Between Top and Bottom Quadrants
As delimited previously I have only included input modalities for interaction with screen-based output. This means that the output modality has a very high precision of hundreds of thousands, if not millions, of pixels even on display sizes of a few diagonal inches. This precision of the output modality has a momentous impact on the relation between the output modality and the input modality.

In the two top quadrants (quadrants 1 and 2) in Figure 21 are input modalities, which have a high precision. Although they do not necessarily match high resolution display output precision pixel for pixel, they have a precision that comes close. Common for the input modalities in quadrants 1 and 2 are that they are centered around providing the user with a physical instrument, which is essential in achieving a high input precision.

In the two bottom quadrants (quadrants 3 and 4) in Figure 21 are input modalities that have a lower precision. Common for these input modalities are that they register, track, or sense human movement or expressions without the use of instruments. However, the precision of human movement or expressions is difficult to match the high precision of high resolution displays. Yet, as new interaction technologies that enable using instrument-less human movement and expressions as input has become available, research interest into leveraging such interaction technologies has increased. E.g. as devices with multitouch capable
displays have emerged commercially (eMarketer) research in multitouch interaction has increased (Schmidt 2015). Interfaces implementing interaction technologies that use instrument-less human movement and expressions are often described as Natural User Interfaces (Wigdor and Wixon 2011), although the designation of NUI as “natural” has been debated (Norman 2010).

In some cases, the distinction between high and low indirectness can become less significant when the input modality matches the human action that is tracked or sensed for input. For example, the high indirectness of Jakoben’s et al. use of proxemics for input (Jakobsen, Sahlemariam Haile et al. 2013) tracks human body movement, which necessitates a high indirectness because body movement is coarse-grained interaction input.

My work on direct-touch interaction (Nielsen, Kjærgaard et al. 2013, Nielsen, Elmqvist et al. Submitted) explores the challenge of using instrument-less human action as input for interaction. Specifically, I have researched direct-touch interaction as input for display based exploratory visual analytics tools. Therefore, my work is positioned in the bottom left quadrant in Figure 21, and, as elaborated in section 4.1.1.3, this quadrant is characterized by having low indirection and low precision.

The main challenge I have sought in my research is the discrepancy between the low precision of using coarse-grained finger input to interact with exploratory visual analytics tools on touch-enabled high-resolution displays. The issue is conceptualized in Figure 22, which illustrates the incompatibility between finger-precision and pixel-precision. Figure 22 arguably oversimplifies the issue between finger-precision and pixel-precision, because few visualization techniques use visual structures the size of single pixel size. The relationship between scale of visualizations, information space, and display size has been investigated by Jakobsen et al. (Jakobsen and Hornbaek 2013). However, studies show that virtual objects in interfaces for direct-touch input should have a size of 9.2 or 9.6 mm

Figure 22. Discrepancy between pixel-precision and finger-precision in direct-touch screen based interaction
(Parhi, Karlson et al. 2006). This means that accommodating direct-touch for input in interactive information visualization is a matter of both developing new interaction techniques as well as adapting visualization techniques. Or developing touch interaction techniques that enable high precision touch input such as precision handle (Albinsson and Zhai 2003).

In the following section I will discuss by distinguishing between adapting existing interaction technologies and developing new interaction techniques for direct-touch input.

4.2 Adapting vs. Developing Direct-Touch Interaction Techniques

In this section I will elaborate on my work on direct-touch adapted brushes (Nielsen, Kjærgaard et al. 2013) and Scribble Query (Nielsen, Elmqvist et al. Submitted). I will distinguish between adapting interaction techniques for direct-touch input and developing new interaction techniques for direct-touch input. I exemplify this distinction thusly: First with my work on adapting brushes for direct-touch (Nielsen, Kjærgaard et al. 2013), which I originally developed for an early version of PivotViz (Nielsen and Grønbæk 2015), and since have iterated for my recent version of PivotViz (Figure 23). Second with my work on Scribble Query (Nielsen, Elmqvist et al. Submitted), which rethinks brushes as a touch-first interaction technique with a simple interaction model that facilitates a low barrier of entry for usage. Figure 24 shows Scribble Query applied to my recent version of PivotViz.

4.2.1 Adapting Existing Interaction Techniques for Direct-Touch Input

As new interaction technologies emerge, one approach to adapt interaction technologies is to map registerable input to known input types on existing interaction techniques.
In its simplest form, adapting a new interaction technology to an existing interaction technique can be a matter of attempting to convert registered input to a known and recognizable metric. Such as parsing a finger-touch input point as a mouse input point in an interface that does not accommodate touch input directly. However, besides facilitating backwards compatibility such an approach can be adverse, especially if finger-precision input is interpreted as pixel-precision input where pixel-precision is expected. This could be extending the range of a brush by dragging an edge, which requires the capacity to precisely select the edge.

Instead, a more desirable approach is to rethink how registered input is converted and mapped to interaction techniques. It is more desirable because enables it allows to arrive at some middle road by altering what interactions can be performed with an interaction technique while also taking advantage of possibilities of a new interaction technology. For example, in my early work on direct-touch brushes (Nielsen, Kjærgaard et al. 2013), I explored how to adapt brushing to direct-touch by converting brushes to transient instruments that was only visible during interaction. Furthermore, for my recent version of PivotViz I have developed multitouch brushes (Figure 23). These multitouch brushes can be defined by using two finger-touch input points concurrently, something which would not be possible with conventional mouse input. Extending and decreasing already created brushes works differently because edges of a brush do not match the comparatively imprecise input of finger-touch input, as discussed in section 4.1.2. Extending brushes is performed by overlapping two or more brushes, which leads to them being merged. Contracting brushes is, however, not possible and can only be indirectly performed by removing a brush and reading a smaller brush in its place. This implementation of multitouch brushes is related to Kosara’s work on in-direct brushing on parallel coordinates visualizations (Kosara 2010, Kosara 2011). In his work Kosara uses multiple concurrent finger-touch input points on a trackpad for defining advanced brushes.
4.2.2 Developing new Interaction Techniques for Direct-Touch Input

Instead of adapting existing interaction techniques to work with direct-touch input, another approach is to rethink how interactions are performed by leveraging the interaction style afforded by direct-touch. This way, interaction techniques can be better merged with the interaction technology, which potentially can lead to a low barrier of entry for novice users.

By implementing a simplified fluid interaction for sophisticated data queries, my work on Scribble Query (Figure 24) is an example of this (Nielsen, Elmqvist et al. Submitted). Yet it is an approach that has in multiple recent research contributions that have develop touch-gestures for instantiating interactions with interactive visualizations. Here I name a few recent exemplars. In TouchViz, Drucker et al. develop touch-gestures for interacting with an interactive visualization based around a bar chart (Drucker, Fisher et al.). With Touchwave, Baur et al. develop gestures for interacting with stacked graphs (Baur, Lee et al.). With Kinetica, Rzeszotarski et al. develop gestures for interacting with stacked graphs (Rzeszotarski and Kittur). My work on Scribble Query (Nielsen, Elmqvist et al. Submitted) relates to these contributions because all these contributions seek leverage direct-touch to re-think how interactions are performed with interactive visualizations. However, because of the fluid interaction in Scribble Query, it also relates to other contributions, such as DimpViz (Kondo and Collins 2014), which utilizes continuous touch input to explore temporal dimensions in data.

4.2.2.1 Scribble Query: A Fluid and Consistent Direct-Touch Interaction Technique

In this section I will elaborate and discuss how to facilitate fluid and consistent direct-touch interaction using my work on Scribble Query as example (Nielsen, Elmqvist et al. Submitted). In Figure 24 Scribble Query is applied to a recent version of my work on PivotViz, which has been used for developing and evaluating Scribble Query. Furthermore, Figure 24 demonstrates how Scribble Query can...
apply to interactive visualizations of large datasets – the dataset used in the example contains data on more than 12 million material borrows from municipal libraries in Aarhus, Denmark.

Scribble Query work by detecting Scribble Query’s intersections with data points on visualized data dimensions. When a Scribble Query intersects with a data point (see Figure 26, Figure 25, or Figure 27) the Scribble Query filters all data entries that intersect with the same data point(s). Crossing data dimensions is possible with or without intersecting any data points, which relates Scribble Query to the crossing interaction paradigm (Apitz, Guimbretière et al. 2008) as well as applications of such as Perin’s et al. Crossets (Perin, Dragicevic et al.). As a Scribble Query is executed, the trajectory scribbled by the user is represented as a path in the visualization, thus effectively persisting a visual representation of the user’s kinesthetic interaction.
The aim of Scribble Query has been to develop a direct-touch interaction technique that has a simplified and consistent conceptual model for its interaction. A simplified and consistent conceptual model is important because it can facilitate a low barrier of entry for novice users. Scribble Query strives for this by condensing its interactive functionality into a single class of interactions: paths that are scribbled directly on top of the data points that a user wishes to filter. This is in Figure 26 where a Scribble Query is used to select a single data points, in Figure 25 where Scribble Queries are used to select consecutive and non-consecutive ranges of data points, and in Figure 27 where a Scribble Query is used for creating a sophisticated high-dimensional filtering. Furthermore, the simple and consistent conceptual interaction model of Scribble Query can help users bridge the gulf of execution and evaluation (Norman and Draper 1986). Scribble Query helps bridging the gulf of execution because the consistency of interaction and helps bridging the gulf of evaluation because a Scribble Query persists a user’s kinesthetic interaction.

However, applying a simplified conceptual model of interaction also runs the risks of lowering the expressiveness of the interaction technique. This means that fluid interaction techniques, such as Scribble Query, likely are not suitable for all users. E.g. experienced data analysts who have a refined workflow for analyzing data might not be willing to adopt new interaction techniques if such users find them incompatible with their workflow. Still, research into simplified interaction techniques, such as Scribble Query, for powerful are necessary if interactive visual analytics are to be disseminated to non-programmer users.

4.3 Conclusion
In this chapter I have given an overview of the breadth of recent research into applying new interaction technologies to information visualization. Based on this overview, I have motivated that I in my work I have focused on direct-touch interaction with touch-enabled devices because it is an interaction technology that affords low indirection between input and resulting output of an interaction. I have
demonstrated this by first adapting conventional brushes to direct-touch, and second by developing Scribble Query as a simplified touch-first interaction technique that assists bridging the gulf of execution and evaluation. Lastly I have demonstrated the applicability of novel touch interaction techniques, such as Scribble Query, by applying it to a fully functional visual analytics tool for exploratory visual analyses of large multidimensional datasets in browsers using touch interaction.

In the final chapter of this dissertation I conclude on my work by summing up this and previous chapters and identify future research directions.
5 CONCLUSION AND FUTURE WORK

In this chapter I conclude the summary part of my thesis. First I summarize my contributions which I have elaborated and discussed in previous chapters on Apprehensible visual analytics, scalable web-based interactive visualizations, and direct-touch interaction techniques. Second I discuss my research method in terms rigor and relevance, as discussed by Fallman et al. (Fallman and Stolterman 2010). Rigor and relevance is always a challenge, and in my contributions developed for end-user applications I have prioritized and evaluated relevance, whereas in my investigation of performance and optimization of rendering SVG visualizations I have prioritized and evaluated rigor. Third and finally, I point towards future research directions, including enhancing direct-touch interaction techniques with the recently available 3D touch technology, and providing visual analytics as a service for open and proprietary data alike.

5.1 Contributions

My contributions, as covered in the previous chapters, can be categorized in three categories: (1) Apprehensible interactive visual analytics, (2) Scalable interactive web-based visualization, and (3) Novel direct-touch interaction techniques. This categorization follows the three hypotheses described in Chapter 1.

1. **Apprehensible interactive visual analytics**: To support non-programmer visual analytics users, I have developed and applied the AffinityViz visualization technique (Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016), which achieves high apprehensibility by retaining a high contextual relatability that leverages domain experts preexisting knowledge and experience with a domain. Furthermore, I have proposed the concept of visual analytics as a service that aims ensure high availability of visual analytics for non-programmers (domain experts and private users alike) to perform data analysis, which I have demonstrated with my work on PivotViz (Nielsen and Grønbæk 2015). To support high availability of visual analytics for non-programmers, I have proposed the concept of data reflection for visualization specification.

2. **Scalable interactive web-based visualizations**: In order to facilitate rich web-based visualization I have rigorously documented performance issues and optimizations techniques for rendering SVG visualizations in browsers. This is an important issue because the declarative nature of SVG scales poorly with atomic visualization of large datasets. Yet the very same declarative nature of SVG makes SVG a desirable technology for rich because it enables powerful manipulation and interactions through CSS and third party JavaScript libraries. I have demonstrated and evaluated data aggregation, data sampling, and progressive rendering of SVG visualizations in browsers in order to enable responsive rendering and interaction (Nielsen, Badam et al. 2016). This is a crucial contribution in
understanding how to utilize the SVG technology for creating responsive visualizations of large datasets in browsers.

3. **Novel direct-touch interaction techniques:** With my work on direct-touch brushing (Nielsen, Kjærgaard et al. 2013), and Scribble Query (Nielsen, Elmqvist et al. Submitted) in particular, I have demonstrated that novel direct-touch interaction techniques can enable simplified, yet sophisticated, data querying with a low barrier of entry. Using my recent version of PivotViz as an evaluation vehicle for the Scribble Query interaction technique, I have demonstrated its applicability for exploratory web-based visual analytics of large multidimensional datasets.

Together these contributions have allowed me to achieve the goals of my research project – to enable exploratory visual analytics of big data in browsers. This is underlined by the demand that the products I have developed have experiences during my research project: My recent version of PivotViz is currently applied with orderly managers at hospitals to help them explore, analyze, and understand patterns in the large number of tasks performed by orderlies. Furthermore, PivotViz is also deployed with a large Danish internet retailer where it is used for analyzing patterns in customer satisfaction scores, and it is deployed in a publicly available version visualizing borrows from municipal libraries in Aarhus, Denmark.

5.2 Research Methods: Rigor and Relevance

As discussed in Chapter 1, I have adopted an experimental computer science approach in my research project. I have adopted this approach because I have strived for developing generalizable results from research based on addressing real world challenges. The distinction, or balancing, of rigor and relevance is a recurring issue when conducting research in a field that is practitioner oriented (Fallman and Stolterman 2010).

In my research, addressing real world challenges commonly has entailed working with potential end-users. In such cases, I have prioritized relevance in my implementations and evaluations in the form of focusing my work towards real world challenges. For example, my work on AffinityViz (Nielsen and Grønbæk 2015, Nielsen, Brewer et al. 2016), which is designed to address the issue of contextual relatability in exploratory visualizations for disparate users, and therefore evaluated with potential domain experts users in order to investigate its relevance. This does not mean that I have disregarded rigor in such cases, however, as I have also strived for producing generalizable results. Instead it means that as a result of my application oriented approach I have chosen to prioritize relevance.

In my work on performance and optimization of rendering SVG visualizations (Nielsen, Badam et al. 2016), on the other hand, I have prioritized rigor in the sense that I have implemented and evaluated rendering rigorously evaluating by simulation. The relevance in this case is therefore of a technical nature because
the relevance is very particular to a technical matter (rendering performance and optimization).

Conclusively, I have struck a balance between rigor and relevance in my work (Fallman and Stolterman 2010). I have worked with users from different fields in my design to prioritize the relevance of the visualization methods. I have strived to also achieve some rigor in the evaluations of my work by deploying or evaluating my work with a variety of domain experts and ordinary citizens. The evaluations have not been conducted with large user groups, yet the qualitative evaluations have supported that my visual analytics techniques and applications work well in real world cases, thus supporting the high relevance of the research.

5.3 Future Research Opportunities
In this section I will describe future research opportunities that either have been made possible through my contributions or has been out of scope in my project.

5.3.1 Advanced Direct-Touch Interaction Technologies
Recently, 3D-touch have been become commercially available in Apple iPhone 6s. This is an interesting interaction technology for enhancing fluid direct-touch interaction techniques such as Scribble Query because it enhances the binary touch/no-touch distinction in current commercial direct-touch technologies with an analogous pressure dimension that can enable new fluid ways of formulating sophisticated data queries. For example, direct-touch has low precision when applied to display-based visualizations, as discussed in chapter 4. To alleviate this low precision, 3D touch interaction could enhance interaction techniques such as Scribble Query by enabling zooming when with hard touch pressure to facilitate high precision in the interaction.

5.3.2 Visual Analytics as a Service
Visual analytics as service is a momentous challenge that requires a multi-faceted approach to explore. However, it can have profound impact on the way data, open as well as proprietary, is available for non-programmer users. It will require great efforts in multiple areas: Datasets to be used in a visual analytics as a service platform will need preparation, further investigation and development of apprehensible visualization techniques for visualizing data, and long term in the wild deployments. Likely, visual analytics as a service would benefit from being developed with a perpetual beta perspective, where dynamic visual analytics platforms are updated frequently based on input from close collaboration with end-users.

This is not meant as an exhaustive list of research opportunities that springs from my research project. Instead it is meant as source of inspiration for researchers who wishes to pursue exploratory visual analytics of big data in browsers further.
5.4 Conclusion

To summarize, the research contributions in my thesis are generalizable results that have advanced the state of the art in exploratory visual analytics of big data for non-programmers in web browsers. I have done this by producing and demonstrating apprehensible visualization, investigating performance optimization of rendering SVG visualizations, and developing novel direct-touch interaction. I have struck a balance between rigor and relevance in my evaluations although my application oriented contributions imply that I have prioritizing relevance. Lastly, I have pointed towards future research opportunities, thus establishing a progressed state of the art on which future work can be based.
6 BIBLIOGRAPHY


Nielsen, M., N. Elmqvist and K. Grønbæk (Submitted). "Scribble Query: Fluid Touch Brushing for Multivariate Data Visualization."


Publications
Publication 1

Exploring Interaction Techniques and Task Types for Direct-Touch as Input Modality

Main author on a poster including two peer-reviewed pages presented at the 2013 IEEE VIS conference.
Exploring Interaction Techniques and Task Types for Direct-Touch as Input Modality

Matthias Nielsen¹, Mikkel Baun Kjærgaard², Kaj Grønbæk³
Department of Computer Science
Aarhus University

ABSTRACT
Recent InfoVis interaction research has called for the inclusion of novel interaction technologies for InfoVis tools. However, it remains an underexplored question what impact novel interaction technologies will have on the existing body of work addressing interaction techniques and task types. This study presents results based on a prototype of an InfoVis tool developed to substitute conventional mouse input with direct-touch as input modality. Two versions of the prototype will be presented – one geared towards mouse input and one geared towards direct-touch input – and the implications entailed by introducing direct-touch as an input modality will be discussed.

Keywords: Interaction, information visualization, direct-touch.

Index Terms: J.5.2.4; J.1.3.6; J.1.5.3;

1 INTRODUCTION
Novel interaction technologies have recently been proposed as a solution capable of meeting an increased significance of interaction in InfoVis [3]. Touch-enabled displays, as a novel interaction technology, are of particular interest as it is an interaction technology that currently has high penetration in terms of smartphones [5], with 45 %, 51 %, and 44 % in Denmark, UK, and US respectively. Furthermore, it is an interaction technology expected to experience extensive growth in annual shipments in the coming years, rising to 2.8 billion displays in 2016 [1], and it is expected to achieve high penetration across a range of products including laptop computers.

Nevertheless, while there in CHI research has been developed guidelines for the design of direct-touch, such as that the size of interface elements for direct-touch should be at least 11.52 mm on a side or diameter [6], existing work is incomplete regarding a holistic view of the impact of touch-enabled displays in InfoVis.

The work presented in this poster abstract will highlight exemplifying differences in terms of interaction techniques, interaction technologies, and task types between two versions of the same prototype of an InfoVis tool. The tool facilitates analysis of library transaction data from 19 public libraries in the Danish municipality of Aarhus, which serves over 300.000 citizens. Both versions, originate in the parallel coordinates diagram made by Jason Davies with the D3 JavaScript library [4]. A reference screenshot of one of the versions is shown in Figure 1.

2 INFOVIS PROTOTYPES
As both mouse input and direct-touch input has their respective merits neither should be considered a definitive input modality. Therefore this exploration is concerned with unraveling what the merits of the respective input modalities are in terms of interaction techniques and task types. In this exemplifying case, the task type is selection, which is supported primarily by brushing as interaction technique. Brushing as an interaction technique to support selection is broached in [2]. Furthermore, this exploration is concerned with how the choice of input modality propagate into the design of the visualization and the implementation of interaction techniques.

Due to limitations of the data provided for these visualizations, all axes have been implemented with discrete values, rather than continuous scales.

2.1 Mouse Input
The interaction in the version geared towards mouse input is based on multiple single-dimensional brushes supporting selection – one for each axis. This version of the prototype has four modes that modify selection using a brush. First, the creation of a brush selects all elements within the extent of the brush. Second, resizing a brush by dragging either the lower or the upper extent of the brush. Third, a brush can be moved along its corresponding axis. Finally, a brush can be removed by clicking outside the range of the brush. At all times the selected elements are the ones that fall within the extent of a brush, or all elements if no brushes are active.

Two of the four selection manipulations are of particular interest here – the resizing of a brush and the removal of a brush. The reason for this is that they respectively show the benefit of the mouse’s high input precision and the limitations in the sequential selection of a brush. These are shown in Figure 2.
Resizing a brush is an interaction that benefits from the high precision of using a mouse as input modality, as the selection of a brush can be updated with high granularity. However, this also showcases that rather than being performed directly on the visualization itself, interaction is performed on the interaction technique (i.e. the brush), thus adding indirectness in the interaction.

While the functionality to remove a brush by clicking within its corresponding axis, but outside its range, is sufficient in itself, it is also an indicator of a limitation – only a single brush can exists on a particular axis at a time. This denotes that a selection only can be a sequential range of values, and limits the modularity in selecting and therefore also limits dynamic selection. While this could be mended, e.g. by using a keyboard key as a modifier, it would also introduce indirectness in the interaction.

In conclusion, selection and representation of selection is contained and evidenced by the presence of a brush. However, even though the brush is moveable and resizeable it is still somewhat inflexible as it can only select a continuous range.

2.2 Direct-Touch Input

In this version interaction is geared towards direct-touch input. Here selection and selection manipulation is supported through four interactions for selecting – selecting a range, selecting a single element, deselecting a range, and deselecting a single element. However, there are only two distinct types – selecting a range and selecting a single element (shown in Figure 3) – as both cases of deselecting is the inverse operation of their respective selecting counterpart.

Brushing is utilized in interaction in this version for selecting a range of elements, marking them with larger font-size and a dark font-color, but the brush is only active for as long as the interaction lasts. This denotes that while selection is being done by means of brushing, the marking of what elements are selected is contained in the visualization and thus separate from the brush. This means that a brush is coupled with a particular interaction instance and not with the visualization, which induces that selection is more dynamic as sequential brushing interactions can update the selection independently of each other.

Selecting a single element is supported by tapping single elements. All elements are eligible for selection by tapping independent of any previous selections. Thus interaction is an expansion upon the abovementioned dynamic brushes.

The central point that is reflected by the implementation of these two interaction techniques is that interaction has been moved from being centered on the interaction technique (i.e. brushing) to being centered on the selectable elements in the visualization. The major redesign is found in the rethinking of brushing as an interaction technique, that is only present when a selection by brush is being performed by a user. This entails that the interaction technique is pushed to the background and indirectness between the interaction and the visualization is lowered. Furthermore, combined with the possibility for updating the selection by marking single elements, the flexibility of selection has been enhanced by using direct-touch.

The rethinking of selection by brushing also showcases that it is non-trivial to introduce novel interaction technologies to InfoVis tools, as a novel interaction technology necessitates rethinking interaction techniques and revising visualizations and their underlying data structure.

3 CONCLUSION

The work presented in this summary has demonstrated that adopting a novel interaction technology in an InfoVis tool – i.e. direct-touch as input modality – cannot be considered plug and play. This has been shown by demonstrating the necessity of rethinking the design of the InfoVis tool both in terms of the visual presentation and the interaction techniques that the visualization offers.

However, these prototypes should only be considered exemplifying, as future research is needed to explore what the impact of novel interaction technologies will have on the design of visualizations and the interaction techniques they offer. This goes for both depth in terms of the impact of single interaction technologies and width in terms of impact across different novel interaction technologies.

REFERENCES

Publication 2

PivotViz: Interactive Visual Analysis of Multidimensional Library Transaction Data

Main author on a four-page peer reviewed paper presented at the 2015 JCDL conference
PivotViz: Interactive Visual Analysis of Multidimensional Library Transaction Data

Matthias Nielsen
Department of Computer Science
Aarhus University
Denmark
matthiasnielsen@cs.au.dk

Kaj Grønbæk
Department of Computer Science
Aarhus University
Denmark
kgronbak@cs.au.dk

ABSTRACT
As public libraries become increasingly digitalized they become producers of Big Data. Furthermore, public libraries are often obliged to make their data openly available as part of national open data policies, which have gained momentum in many countries including USA, UK, and Denmark. However, in order to utilize such data and make it intelligible for citizens, decision makers, or other stakeholders, raw data APIs are insufficient. Therefore, we have developed PivotViz that is a comprehensible visualization technique, which combines parallel coordinates and pivot tables. It provides, a multidimensional visual interactive pivot table for analysis of library transactions – loans, renewals, and returns of books and other materials across location and time. The paper presents the PivotViz technique and discusses its prospects based on implementations in two publicly available versions using open data from the two largest municipalities in Denmark. Examples of analysis results from these data illustrate the power of PivotViz.

ACM Classification Keywords
H.5.2 User Interfaces (D.2.2, H.1.2, I.3.6)

Author Keywords
Library Data; Information Visualization; Visual Analytics; Big Data; Open Data

1. INTRODUCTION
Libraries of today are hybrid libraries, where both physical and digital material can be loaned. But a common feature of physical or digital material loans is that the administration of loans is fully digitalized and thus generate huge amounts of data, which due to its sheer volume, velocity, and variety bears the characteristics of Big Data [3]. This provides new opportunities for making this data useful for a wide range of stakeholders, such as users of libraries, librarians, managers, politicians etc. Concurrently, many countries have launched Open Data initiatives aiming at making all sorts of governmental data publicly available for anybody to use, such as the UK’s open data initiative [4]. This makes public libraries a special case because they, like other libraries, will generate large amounts of data, and additionally they can be obliged to make such data publicly available if they fall within their governments open data policy. However, making data available is very different from making it discernable especially for public libraries, because their stakeholders range from politicians and decision makers to the common citizen, who are not per se trained in business intelligence type data analysis.

We have developed a visualization technique called PivotViz to support efficient and comprehensible analysis of big open data sets, here open loan transaction data from public libraries. In our work with PivotViz, we have taken offset in publicly available data describing transactions of materials at libraries in the two largest municipalities Denmark, Aarhus and Copenhagen, both maintaining around 20 physical branches respectively. We have developed a novel visualization technique, based on parallel coordinates and pivot tables, to facilitate interactive visual analysis of loaner transactions. Furthermore, we have implemented two publicly available versions, available at URLs [1, 2], which, although targeted for managers or decision makers, allows anyone interested to analyze library transactions across location (library branch), time (weekday and hour of day), and transaction type (loan, renewal, or return).

In the remainder of this paper we first cover related work in data mining and interactive visualization of library data. Second, we elaborate and discuss the PivotViz visualization technique developed by combining parallel coordinates and pivot tables. Third, we present use cases of two PivotViz visualizations of library transaction data in Aarhus and Copenhagen. Fourth, we discuss future work for evolving the PivotViz visualization technique into generally applicable tool for visualizing and interactively analyzing multidimensional data.

2. RELATED WORK
In this section, we cover a selection of previous research addressing mining, analyzing and visualizing data in the context of libraries. Furthermore, we cover specific related work from the field of information visualization.

Bibliomining combines bibliometrics and data mining [5, 6] for the purpose of mining data collected by libraries for knowledge that can help users, librarians, managers, or other relevant parties. It is a multifaceted approach covering a wide scope from data handling to communication, and related to our work because we visualize large datasets of library transaction data to enable interactive analysis and hence knowledge generation.

Research in visualization of data from libraries has taken many forms. Strategies and tools for visual querying or exploring (digital and/or physical) library collections have been investigated in [7, 8], and Gelernter [9] present a selection of information...
visualization interfaces to collections of digital libraries and propose the foundation for a classification of information visualization for digital libraries. Furthermore, Radburn et al. [10, 11] have researched how visualizations of library records can assist local authorities in Leicestershire in understanding the needs of its citizens and providing better services for them. These researchers all tackle challenges of how to handle digital aspects of libraries by visualization, and therefore relates to our work.

In relation to the field of information visualization, our work relates specifically to the exhaustive work with the parallel coordinates visualization technique by Inselberg [12] and Dasgupta et al. [13] among others, as well interaction techniques with parallel coordinates as explored in [14, 15]. Furthermore, our work relates specifically to the pivot table functionality, which is common in spreadsheet software applications [16].

3. PIVOTVIZ

The core principle in the PivotViz visualization technique is the combination of the structural layout of the parallel coordinates with the summed, but easily comprehensible, numbers of pivot tables, to create an interactive visualization of large datasets. The visualization technique was originally developed to enable municipal decision makers analyze transaction data from public libraries, because the millions of material transactions being collected had become too extensive for them to analyze using conventional methods.

3.1 Multidimensional Visual Pivot Table

PivotVis borrows the structural layout of parallel coordinates [12, 13], which represents data dimensions as parallel axes and plots data entries as individual lines according to coordinates on these axes. Instead of plotting data tuples as individual lines, as conventionally done in a parallel coordinates visualization, PivotVis plots a line, represents the summed number of all transactions that have identical values. I.e. each line in the visualizations in Figure 1 and Figure 2, available online at URLs [1] and [2], represents all transactions of the same type that have taken place at a specific location, weekday, and hour of day, with the summated number represented as thickness and opacity of the line. The strategy of summing all identical tuples and only representing aggregated numbers is similar to the functionality of pivot tables [16], which represents precise summed numbers of a dataset across select dimensions.

This combination of parallel coordinates and pivot can best be described as an interactive multidimensional visual pivot table. PivotViz is illustrated in use in Figure 1 and Figure 2.

Incorporating the summarized numbers of pivot tables [16], means that, for the same layout, the maximum number of lines will stay same (the number of properties on each dimension multiplied). Even if the underlying dataset keeps growing each line will just represent an increasingly larger number – like a cell in a pivot table. This means that the visualization technique scales well as data grows in size and therefore it is highly suitable for visualizing Big Data. Like a pivot table, however, this is limited to datasets with entries that can be treated as discrete data and be mapped onto ordinal or nominal scales. Visualizing datasets with continuous data dimensions would mean that the number of lines needed in the visualization would equal the number of entries in the dataset. That is, unless the continuous data dimension can be converted to a non-continuous scale, as is exemplified in the usage of time in the online versions at URLs [1, 2], which has been reduced to two ordinal dimension (weekday and hour of day) although it is continuous in the underlying dataset. Making a pivot table visual means that outliers and patterns becomes easier to detect because graphical structures are effectively processed by human vision [17]. On the other hand, overlapping lines can clutter the visual representation, making interactive selection and filtering a necessity rather than just a useful feature.

PivotViz incorporates the multiple dimensions of parallel coordinates visualization technique [12, 13], which means that it is suitable for visualizing complex multidimensional datasets and still maintaining an overview of the data. This is a significant improvement over the tabular layout of pivot tables because pivot tables on the other hand loses interpretability if more than two data dimensions are included. However, the scalability of PivotViz only extends to a certain limit as hundreds or even tenths of concurrently visualized dimensions would counter overview of the visualization both in terms of scattering of axes as well as in terms of increase in the number of individual lines.

3.2 Interactive Visual Analysis

Interaction is crucial in PivotViz because the visual representation does not yield comparatively precise visual queries
as e.g. a conventional bar chart does (i.e. bar A is approximately 2/3 the height of bar B). However, with interaction, it traverses large scales from municipality wide transactions to a single library’s transactions in a narrow timeframe for specific types of transactions. Furthermore, clustering of lines reveals clusters in the data, but dominant lines can hide subtle clusters, making interactive filtering important. PivotViz supports flexible filtering in the form range selection and selection of single elements on one or more axes, as shown in Figure 1 (deselction is implemented as inverted selection and is equally supported). Range selections are performed by means of temporary brushing over a range of ticks on an axis and single elements are selected by clicking on a tick. The selection functionality facilitates flexible filtering meaning that PivotViz follows Shneiderman’s overview first, zoom and filter, then details-on-demand mantra [18]. Although the details available on-demand is only an increasingly precise number and not contextual information on actual materials that were objects of transactions.

Even though PivotViz combines properties of parallel coordinates and pivot tables it does not, however, substitute either as some of strengths normally associated with the two is not carried over. For instance, the immediately available high detailed numbers of the pivot table can only be reproduced in PivotViz through interaction. Likewise, advanced visual queries supported in conventional parallel coordinates, like correlation analysis between axes if a select set of lines have a similar slope, is not supported either.

3.3 Architecture and Implementation

PivotViz is developed as web-browser based visualization using JavaScript, including D3 [19], to create SVG elements in the browser DOM, making the web-browser render the visualization without the need for third party plugins, thus making it easily accessible for interested parties.

![Figure 3: Architecture in PivotViz.](image)

The current implementation visualizes batch-processed static datasets that are loaded by a single-page web application, which creates the visualization following a Model View ViewModel pattern. Albeit with a simple model in the current implementation because the data in the model is static and the bindings between the ViewModel and the View are not updated. However, it is trivial to either update the model or replace with a new data source, making it a flexible architecture, shown in Figure 3. Furthermore, no persistence of performed analyses or functionality for sharing or cooperation is currently built-in (needed for process and provenance in Heer and Shneiderman’s taxonomy [20]), meaning that currently the only way to persist, share, or corporate analysis results is conventional screenshots. However, making deep links to make persistent analysis results is possible to implement, by e.g. implementing server based save functionality or by creating custom URLs using a hashtag fragment identifier that contains information of a specific state of the visualization, which should be created on load.

The visualizations in [1, 2] have been tested, and they render and work interactively, in the four major desktop browser – Mozilla Firefox, Google Chrome, Internet Explorer, and Apple Safari. However, due to the implementations’ heavy use of CSS selections of large sets of elements (for interaction) it performs best in Mozilla Firefox with Google Chrome a close second best.

4. TWO MUNICIPALITY USE CASES

The two versions of PivotViz have been publicly available for approximately 1½ year for Aarhus municipal libraries, and for approximately ½ year in Copenhagen municipal libraries. Both versions have been used by library management in their respective municipalities. In this section, we present the two use cases and examples of types of analysis results.

4.1 Analysis: Usage of Self-Service Libraries

In response to comparatively little usage, some of the smaller public libraries residential areas or suburbs in Aarhus have had their opening hours extended into the evening while not being staffed. The intention with this initiative was to keep libraries open when users are not at work and have time to loan or return books, while at the same time not increasing the cost of running these libraries by staffing them in the afternoon and evening. This initiative has resulted in that almost 6 % of the total transactions of materials in public libraries in Aarhus municipality takes in unstaffed libraries. A screenshot of this analysis is available Figure 4 and can be recreated via the URL in [1].

![Figure 4: Usage of unstaffed public libraries in Aarhus](image)

4.2 Analysis: Online Renewal of Materials

Besides loans and returns, the data from Copenhagen municipal libraries also includes renewals. Furthermore, the data also includes transactions that have taken place through the municipal libraries’ common web-portal. A screenshot showing the distribution of all renewals is shown in Figure 2, which shows a distinctive clustering at a specific location on the first axis, which, admittedly intelligible in Figure 2, is the Web. The number of renewals performed through the municipal web-portal equates to almost 82 % of the total number of renewals across all libraries. This indicates that the users of the municipal libraries in Copenhagen also use the web-portal extensively, meaning that it is a highly integrated part of the service that the municipal libraries provide.
5. FUTURE WORK

In our efforts to develop the PivotViz visualization technique, used in The Books of Aarhus (Figure 1), available at URL [1], and The Books of Copenhagen (Figure 2), available at [2], a generally applicable visualization technique has emerged that we intend to expand upon in terms of modularity and flexibility.

The current implementations at URLs [1, 2] only allows end-users, i.e. non-programmers, to interact with a predefined static dataset, as discussed in section 3.3. According to the reference model for visualization [17], this means that human interaction is currently limited to view transformations and visual mappings and data transformations requires programmers to perform. We plan to demonstrate the general applicability of the PivotViz visualization technique by developing a tool that allows end-users to perform both visual mappings and view transformations. As discussed in section 3.1, the visualization technique applies to certain types of data and, generally, we do not consider it feasible to implement functionality for end-users to perform data transformations. However, for data commonly collected at libraries, such as electronic visitor counts, ILS data, etc., data parsers or connectors could also be implemented to allow end-users to import data.

Creating such a tool means increasing modularity in terms of allowing end-users to change, update, remove, and add data sources. Furthermore, because data sources increasingly are providing dynamic data (e.g. streams) [21], timed or event-based updates of data should also be supported. Besides modularity, flexibility should also be increased, meaning that end-users should be able to specify what parts of datasets should be visualized and to some extent how – related to Heer and Shneiderman’s Data & View Specification [20].

Increasing modularity and flexibility of a visualization is also highly relevant for the data visualized at URLs [1, 2]. In the presented visualizations, time, location, and type of transaction is included, but it could also be desirable to correlate transaction patterns with other data commonly collected at libraries, e.g. visitor numbers, events, ILS data, etc.

For the future version of PivotViz, we plan to conduct evaluations of the visualization technique and its applicability for visual analysis by end-users. Because we envision the future version of PivotViz as a generally applicable tool, evaluations will be conducted with both extended datasets of library transactions, as described above, as well as datasets from other domains.

6. CONCLUSION

This paper has introduced the PivotViz visualization technique for library transaction data. The main contributions of the paper are 1) PivotViz, a generalizable visualization technique for visual analysis of library transaction data, and potentially other data. 2) Two publicly available interactive visualizations – The Books of Aarhus and The Books of Copenhagen – exhibiting the PivotViz visualization technique for analyzing library transaction data in practice. 3) Discussion of prospects for the PivotViz visualization technique and how to evolve it into a modular and flexible visualization tool.

Based on the successful implementations of PivotViz in the library domain, we see promise for PivotViz in other domains where big data on location and time tagged events needs to be analyzed. We are already looking into evaluating PivotViz in such new domains.

7. ACKNOWLEDGEMENTS

The work was supported by Danish DSF grant no. 11-115331.

8. REFERENCES

[18] Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. IEEE, City, 1996.
Publication 3

Scribble Query: Fluid Touch Brushing for Multivariate Data Visualization

Main author on a ten-page peer paper submitted to the 2016 EuroVis conference
Scribble Query: Fluid Touch Brushing for Multivariate Data Visualization

Submission ID: 297

Abstract

The wide availability of touch-enabled devices is a unique opportunity for visualization research to invent novel techniques to fluently explore, analyze, and understand complex and large-scale data. In this paper, we introduce Scribble Query, a novel interaction technique for fluid freehand scribbling (casual drawing) on touch-enabled devices to support interactive querying in data visualization. Inspired by the low-entry yet rich interaction of touch drawing applications, Scribble Queries can be created with a single touch stroke yet have the expressiveness of multiple conventional brushes. We have applied Scribble Query to multivariate visualizations of data from five different domains, deployed it with domain experts, and conducted deployment studies with these domain experts on their utilization of multivariate visualization with Scribble Query. The two studies suggest that the Scribble Query interaction technique has a low entry barrier facilitating easy adoption, casual and infrequent usage, and, in one case, enabled live dissemination of findings by the domain expert to managers in the organization.

Categories and Subject Descriptors (according to ACM CCS):
H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input devices and strategies H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction Styles

1. Introduction

Touch interaction has been shown to be intuitive, familiar, and easy to learn [HXSH14]. As a result, touch displays have been deployed in more than 2 billion smartphones, tablets, computers, and large displays all over the world [eMa], and fluid touch interaction such as scribbling, swiping, and shape writing has found high preference with users everywhere [NB12]. However, this also represents a potent challenge for data visualization designers because users have come to expect to be able to just as easily use their touch-screen devices to interact with such visualizations. However, fingers lack the unambiguity of keystrokes on keyboards and the pixel-precision of mice, thus requiring robust and fault-tolerant interaction that leverages the strengths of touch input and mitigates its weaknesses. Recent work in the information visualization (InfoVis) field has increasingly started to utilize touch interaction to explore and develop novel ways of interaction with interactive data visualizations. Applying conventional touch gestures (tap, drag, swipe, pinch, etc.), to trigger contextualized commands specific to a single visualization (e.g. a swipe on a barchart orders the barchart numerically [DFSH13]) is a common approach [BLC,DFSH13,RK,SNDC10]. Mapping touch input to brushing on parallel coordinates has been explored through indirect multitouch [Kos10,Kos] and direct singletouch [NKG13]. Other work [EZ15, HF, Wat01] inves-

![Figure 1: Using Scribble Query to create an advanced, multi-dimensional data query. For review, all place names have been substituted with Estonian city names.](image-url)
Figure 2: A use case for using Scribble Query to create touch brushes on a multivariate data visualization of over 12 million library loans with data dimensions describing loans by their year, location, material type, material language, loaner postal area, and loaner age group. (1) The visualization’s initial state. (2) A Scribble Query selects year range 2010-2012. (3) A Scribble Query further selects Movie loans. (4) A multidimensional Scribble Query further filters, in a single gesture, the Tallinn postal code and the non-consecutive “English” and “Multiple” Languages, while crossing the Material and Library dimensions without making selections. This Scribble Query sequence answers that in years 2010-2012, loaners in Tallinn loaned 136,662 movies with English or Multiple languages.

The contributions of this work are the following: (1) the Scribble Query interaction technique for touch-based freehand brushing; (2) the application of Scribble Query to parallel coordinate plots for multivariate data; and (3) results from our case studies of domains where we have deployed Scribble Query with multivariate data visualizations with domain experts.

The remainder of this paper is structured as follows: First, we review the literature in relation to Scribble Query. Second, we introduce and discuss the Scribble Query interaction technique and the broader design space of brushing for touch interaction. This includes the design rationale for multivariate and time-series visualizations using touch interaction. Third, we report on our findings from our two deployment studies investigating Scribble Query in use with a few domain experts from two domains. Fourth, we discuss applying Scribble Query to other types of visualizations as well as its limitations. Finally, we outline future work and conclude the paper with a summary of our contributions.

2. Background

In this section, we position Scribble Query in relation to direct manipulation and post-WIMP interaction, recent work in touch interaction with interactive data visualizations, as well as general multivariate visualization.

2.1. Direct Manipulation and Post-WIMP Interaction

Our work in this paper is based on freehand input performed on visualized data elements, yielding a high affinity between the user’s input and the resulting data query. This is an approach inspired by direct manipulation [Shn83], which is based on the idea of allowing users to perform direct, iterative interactions on continuously-updated data items rather than through complex and abstract syntax. Furthermore, we are inspired by instrumental interaction [BL00], which uses three properties—indirection, in-
tegration, and compatibility—to operationalize design and analysis of so-called interaction instruments in post-WIMP interfaces. In our work, we seek to lower the indirectness and increase the integration and compatibility for formulating queries. Finally, we also seek inspiration from fluid interaction [EMJ11] by designing an interaction technique for interactive data visualization that promotes flow, supports direct manipulation of domain objects, and minimizes Norman’s gulf of interaction [ND86].

Our work takes place within the context of increased focus on new interaction technologies (i.e. without mouse and keyboard) embraced by the InfoVis community in recent years. This is exemplified by recent contributions on InfoVis-specific interaction design considerations [LIRC12], as well as a data visualization-specific interaction model [JD13], extending upon instrumental interaction, for visualizations utilizing new interaction technologies for data visualization.

2.2. Touch Interaction for Visualization

The increasing availability of touch mobile devices has made touch interaction for visualization a hot topic in recent years. However, creating touch-based visualizations is often challenging. The simplest approach converts mouse interaction to touch, such as in Rizzo [VCH10], a multi-touch interaction technique that allows users to perform mouse-precision input on multi-touch devices. This enables applications that are dependent on mouse-precision input—visualizations especially—to work on touch devices.

However, fully harnessing touch requires customizing common touch input gestures to each specific visualization. Schmidt et al. [SNDC10] propose touch gestures for node-link diagrams, TouchWave [BLC] provides single and multi-touch gestures for interacting with stacked graphs, TouchViz [DFSH13] adapts touch gestures to barcharts, Kinetica [RK] uses multi-touch interaction to create virtual instruments for interacting with scatterplots and histograms, and Sadana et al. develop multi-touch gestures for scatterplots [SS14]. Specifically for parallel coordinates, recent work includes performing indirect multi-touch brushing [Kos10, Kos], using a computer touchpad for input, and direct single-touch for filtering [NKG13]. Common for these gestures, techniques, and interfaces are that they rely on touch gestures that quickly have become standardized (swipe, pinch, dwell). These touch gestures are then mapped to visualization-specific commands related to the objects that are interacted with; e.g. dragging the vertical axis in TouchViz [DFSH13] orders the barchart. There are, however, exceptions, like DimpVis [KC14], which utilizes fluent, continuous touch input to explore a temporal dimension in a visualization.

Related to scribbling is using sketched input on touch-enabled devices, which has been explored by Wattenberg [Wat01]. Ryall et al. [RLL05], and Holz et al. [HF]; the latter uses a sketched path for matching trajectories of lines in time-series visualizations. Line trajectories that match the sketched path within some margin are filtered and users can then modify or enhance their query. Eichmann et al. [EZ15] expand upon this work and investigates users’ perception of tolerance of matching line trajectories to sketched input. Furthermore, scribbling input for information visualization has also been utilized in Napkin-Vis [CMP09] and SketchInsight [LSR11], which both explore the use of sketched gestures to create manipulate visualizations. Brownee et al. [BLC11] and SketchStory [LKS13] further explores using sketched input on interactive whiteboards for creating visualizations by drawing components of visualizations (e.g. bars in a barchart) to visualize data. SketchSliders [TB15], on the other hand, explores scribbling ad hoc widgets on tablet devices in order to explore data visualizations on large wall-mounted displays.

Scribble Query relates to the crossing interaction paradigm [AGZ08] that, although designed for pen input on touch screens, rethinks the conventional mouse-move, mouse-over, and mouse-click interaction instantiations to continuous touch-down and touch-down that instantiate actions by crossing goals. So far, crossing has mainly been explored in the form of interface menu interactions, with Apitz et al. [AGZ08] using crossing to continuously select nested tools in a drawing application, and Perin et al. [PDF15], who uses crossing to manipulate multiple sliders. We expand on this work because a Scribble Query instantiates queries based whether and where a Scribble Query intersects, or crosses, a dimension, as can be seen in Figure 2 (4).

In Scribble Query, we seek to look beyond conventional touch gestures and let naturalistic fluid scribbling interaction take precedence over point and click instruments for interaction by maintaining a high affinity between a user’s interaction and the resulting query. Furthermore, we seek to utilize freehand naturalistic scribble input for directly querying data visualizations.

2.3. Multivariate Data Visualization

Visualization of multivariate data has received extensive attention over many years, and while it is out of scope for this paper to include an exhaustive review, below we provide a brief overview with selected examples of common visualization techniques.

In general, multivariate data visualization is challenging because most traditional statistical graphics cannot handle more than a handful of concurrent data dimensions. For example, scatterplots are one of the canonical types of statistical graphics, but can only reliably visualize two or three data dimensions at the same time. For datasets with more dimensions, a common approach is to generate several small visualizations and organize them side by side, a technique often called small multiples [TF91]. For example, multiple scatterplots can be grouped into so-called scatterplot matrices
Scribble Query: Fluid Touch Brushing for Multivariate Data Visualization

(SPLOMs) [BC87]. Elmqvist et al. propose using SPLOMs as an overview for navigating and exploring multivariate data [EDF08]. However, multiple views imply that it can become challenging to correlate data in one view with another. Coordinated and multiple views [Rob07] and brushing and linking [BC87] are general approaches for updating and highlighting selected items in other views to make understanding small multiples easier.

The idea of brushing can be extended to virtually any type of data and visualization. Timeboxes [HS04] is an interaction technique for querying time-series data by constructing rectangular boxes similar to brushes. The technique improves upon normal brushing by allowing for flexible construction of composite timeboxes that move together and facilitate constrained data on multiple dimensions.

In contrast to multi-view visualizations, some visualization techniques are designed specifically to represent multiple dimensions in the same view. As a case in point, parallel coordinate plots [Ins97] visualize high-dimensional data by representing data dimensions as axes organized in parallel, transforming data points into polylines connecting the axes. Parallel sets [KBH06] extends parallel coordinates to accommodate very large datasets of categorical data or categorized continuous data using aggregated bands. Both of these techniques are relevant to our work in this paper, but none of them have yet been redesigned for touch input.

Finally, current visual analytics systems for multivariate data connect multiple visualization views placed on an infinite canvas with data connections between the views. Examples of such so-called data flow systems include DataMeadow [EST], GraphTrail [DRL’12], and Explorers [JE13]. Common between all of them is that they allow users to create advanced multidimensional data queries that update as one view along the query path is brushed and filtered.

All of the techniques discussed above are based on the notion of brushing: selected items being highlighted across correlated views in a scatterplot matrix [BC87], parallel coordinates plot [The06], or data flow systems [EST]. Although more advanced than brushes, timeboxes [HS04] are a related interaction technique and they exemplify a desirable flexibility in query creation and manipulation.


Scribble Query is designed for fluid touch brushing on multivariate, applying a simplified and consistent interaction paradigm that is also easily discoverable. Technically, the central principle in Scribble Query is that a user scribbles a path on a touch-screen, which creates and persists the path traversed by the user. While the Scribble Query is being created, the visualization is updated as the Scribble Query intersects with categorical or discretized items on visualized data dimensions. In Figure 2, a use case of a sequence of Scribble Query interactions are performed to analyze library loan data. In this use case, three Scribble Query are created sequentially by a user to rapidly perform a sophisticated query of a multivariate dataset. The central point in this use case is the flexibility: a Scribble Query filters exactly what a user’s finger intersects by performing Scribble Query gestures on the visualization. Furthermore, due its consistency, the Scribble Query interaction technique is easily discoverable due to its direct interaction paradigm, and grows with a user’s desire for increasingly sophisticated queries (e.g. Figure 2 (4)) due to its consistency.

The simplified and consistent interaction paradigm of Scribble Query differentiates our contribution previous work using touch-gestures to adapt data visualization interaction to touch-screen interaction [BLC, DFSH13, RK, SNDC10, SS14]. For example, performing a two-finger pinch gesture in TouchWave [BLC] scales stacked graphs horizontally or vertically, while a similar gesture in Kinetica [RK] either creates a scatterplot using the input points as scale or filters plots in the scatterplot. Put simply, the simple approach of mapping touch gestures to commands is somewhat arbitrary and not easily discoverable.

In this section, we first present the Scribble Query interaction technique, including the principles of designing a crossing-inspired interaction technique for fluently scribbling queries of multivariate data. Second, we elaborate on how adapting the conventional brushing interaction technique to touch-screen interaction can enhance the interaction technique. Third, we discuss design considerations for adapting parallel coordinates [Ins97] and parallel sets [KBH06] to touch-screen interaction.

3.1. Design

Scribble Query is the result of our exploration into how the fluent interaction common in touch gestures (swipe, pinch to zoom, etc.) can be for brushing in multivariate data visualizations. We based our exploration on the brushing technique, because, as presented in the related work section, it is a common selection technique in general and multivariate data visualization [BC87, EDF08, HS04, Rob07, The06]. Existing work in touch interaction with brushes has so far only addressed adapting conventional brushes for touch interaction [Kos, NKG13].

In order to explore the design space of touch brushing on multivariate data visualization, we first developed a touch-friendly adaption of parallel coordinates [Ins97] and parallel sets [KBH06] for visualizing large datasets. The visualization (Figure 2), visualizes aggregated multivariate data by plotting dimensions in the data as vertical parallel axes, with a circle for each category on the individual axis, and uses paths to visual inter-dimensional relations in the data.

submitted to Eurographics Conference on Visualization (EuroVis) (2016)
3.1.1. Interactions

Whenever a Scribble Query intersects a category on any axis in the multivariate data visualization (Figure 2), the visualization is filtered to only show lines intersecting this category. Any number and combinations of Scribble Query can be added to the visualization and all Scribble Query can be drawn in any direction (up, down, left, right, or any combination of two). The visualization is updated continuously as a query is being scribbled; as new categories are covered by a Scribble Query, lines intersecting with this category are added to the visualization instantaneously.

Selecting a single category (Figure 3) is performed by scribbling a short stroke covering a single element on an axis in the multivariate visualization. This is the simplest query that can be performed with Scribble Query and all interactions covered in remainder of this section can be constructed as a series of selections of single categories.

Removing a Scribble Query (Figure 3) is performed by tapping a single time on a Scribble Query. When a Scribble Query is removed, all lines intersecting the category are removed from the visualization. Unless no Scribble Query brushes are left, then all lines are shown in the visualization.

Selecting a range of categories (Figure 4) is performed by scribbling a path intersecting multiple categories on an axis. A Scribble Query can cover any number of categories, including all categories on an axis.

Selecting a non-consecutive set of categories (Figure 4) is performed by simply scribbling around a category that should not be included in the selection.

High-dimensional selection (Figure 5) is performed by initiating a Scribble Query on any axis, selecting one or more categories, and extending the Scribble Query onto other axes to select more categories. A Scribble Query can intersect all axes multiple times in any order. If the user does not intend to select categories on a specific axis, the Scribble Query can be drawn above, under, or between (and not intersecting) two categories.

Modifying a Scribble Query (Figure 6) is performed by dragging on a Scribble Query. Categories neighboring the point where the dragging was initiated locks the Scribble Query. This can be used to unselect a category from a high-dimensional Scribble Query if the dragging is initiated on a selected category. We have not implemented this interaction in our current version.

3.1.2. Interaction Paradigm

Scribble Query allows users to use freehand touch input to create queries on data visualizations. This interaction based on the principle of direct manipulation [Shn83]: when a user scribbles a query, the interaction is directly persisted and the user’s interaction becomes the query, thus retaining a high affinity between the user’s interaction and the resulting Scribble Query. This way, Scribble Query leverages fluid scribble-based interaction with computer interfaces and supports visual thinking, as argued by Gross [Gro09].

Furthermore, in the design of Scribble Query, we sought inspiration from the casual, yet engaging, interaction in scribbling/doodling applications commonly available on tablet devices. The functionality of such applications—creating drawings or doodles using fingers as input on a touch-screen—is characterized by a low entry barrier.

3.1.3. Comprehensible Conceptual Model

A Scribble Query imposes a conceptual model of interaction on users that is at the same time simple while expressing a advanced query. This is because the same basic interaction—scribbling—allows users to perform queries ranging from selection of single categories (Figure 3) over non-consecutive selections (Figure 4) to high-dimensional queries (Figure 5). This means that a user’s interaction with Scribble Query can evolve with users as the user becomes either better capable of, or more interested in, performing increasingly advanced queries.

An essential part of the conceptual model of Scribble Query is that it avoids constructing dispensable virtual instruments for interaction, which can introduce indirection in
interaction [BL00]. This way, Scribble Query does not impose a complex set of instructions, nor does it require that the user alternate between one set of interactions when constructing a query and another set of interactions when modifying a query, as is commonly the case with brushes. The only time a Scribble Query becomes an instrument for interaction is when the user deletes it by single-tapping it (Figure 3), or when the Scribble Query is modified by dragging (Figure 6).

3.1.4. Crossing-based Interaction

Scribble Query integrates the crossing interaction paradigm [AGZ08], which is designed to rethink conventional mouse-based individual interactions with fluent sequential interactions crossing multiple interface elements. This is demonstrated in the interaction sequence in Figure 2. First, in Figure 2 (2) and (3), the Scribble Query exhibits basic crossing when it used filter a range (2) and a single element (3). The Scribble Query in Figure 2 (4) uses sophisticated crossing when it both crosses dimensions without making selections, and performs non-consecutive filterings when it crosses a dimension on non-neighboring points. Because the visualization is updated continuously as a query is scribbled, the technique enables continuous data exploration through a single gesture.

3.1.5. Limitations of a Scribble-based Approach to Querying

The main limitations of using finger-based touch-Scribbling for querying data visualizations are the relatively low accuracy of finger touch-input (compared to mouse-input) and the potential occlusion of parts of the visualization by the user’s hand or fingers.

Through their in-depth investigation of finger-input on touch-screens, Wang et al. [WR09] conclude that interface elements meant for touch input should have a physical size of at least 11.52 mm to facilitate users to accurately target an element. Because Scribble Query brushes are Scribbed on top of a visualization, we utilize this guideline in the design of our visualization of multivariate by discretizing data, which enables us to maintain adequate spacing between categories on axes. We further elaborate on the visualization design in the following section.

Occlusion caused by a user’s fingers or hands, also discussed by Wang et al. [WR09], is a stronger concern because it is caused by a fundamental property of finger-touch interaction. In Scribble Query, we attempt to alleviate issues caused by occlusion through the simple conceptual model of interaction. Specifically, because all queries can be performed by series of single selections (Figure 3), the user can construct the query step-by-step. This is also a property that can assist a user in learning the interaction technique, because the interaction for the simplest selection is essentially the same as for an advanced high-dimensional query.

Using a pen for touch interaction could be considered an alternative to finger-touch input. However, using a pen causes indirection in the interaction and not all users will have a pen available meaning the interaction technique would be usable on less devices. Scribble Query could be performed using a pen nonetheless if the pens touch input is mapped to finger touch input.

3.2. Multivariate Data Visualization for Touch Interaction

The visualization we initially developed for exploring touch-based brushing for data visualization is an adaption of the parallel coordinates visualization technique [Ins97] to visualize large datasets in a fashion related to parallel sets [KBH06]. In this section, we will briefly elaborate on how discretizing data helps accommodating finger-touch input as well as enables visualization of large amounts of data. Discretization, or data binning, is a the process of counting frequency of occurrences in a number of bins, and it is a technique that is commonly applied when creating his-

submitted to Eurographics Conference on Visualization (EuroVis) (2016)
tograms of continuous data (e.g. age groups, months and years in time-series data, and profit margins in intervals).

The design of our visualization of multivariate revolves around either using discrete (nominal, ordinal, or ranges of the two) data or converting continuous data to discrete values. As observed by Kosara et al. [KBH06], much data is categorical. We further propose that discretizing continuous—e.g. converting time and date of a data entry to year, month, day of month, hour, etc—is an approach that yields desirable results for data visualization for finger-touch interaction on data visualizations of large datasets.

Discretized data can be easier to interact with than continuous, when using finger-touch input, because aggregated categories are easier to distinct and deliberately select, than continuous data entries. Using Wang’s et al. [WR09] guideline for size of elements for finger-touch interaction, the spacing of categories, the maximum number of categories on a dimension should be decided in consideration of the intended screen size. Meaning that in order to facilitate accurate selection using finger touch, categories on a dimension should be spaced with at least 11.56 mm of space between them.

3.3. Implementation Notes

Discussing all technical details of the implementation is outside the scope of this paper; here we summarize the issues most relevant to the Scribble Query technique.

Scribble Query and our touch-friendly visualization for large datasets were implemented in the JavaScript programming language and developed for use in common web browsers. Our implementation uses the D3 [BOH11], Jquery, Crossfilter JavaScript libraries to handle, visualize, and interact with data in SVG format. Due to potentially very large datasets, we have implemented subsampling of data entries to be rendered, because the visualization is implemented as SVG, which are DOM elements and too many DOM elements can stall even modern browsers on powerful computers.

Our implementation is organized into three modules—a data management module, a visualization module, and an interaction module—that are decoupled and coordinated through a simple protocol. When a user scribbles a query, the interaction module translates user input into virtual elements (i.e. DOM elements) and informs the data management module of the current state of the virtual elements. Furthermore, the virtual elements, maintained by the interaction module, becomes elements ready for interaction, and the user can remove (Figure 3).

The data management module uses the current state of interaction elements to create a corresponding filtered dataset. This filtered dataset is passed onto the visualization module, which updates the SVG visualization according to the filtered data. This modularized structure is important in order to easily develop and experiment with exchanging interaction techniques as well as rapidly exchanging the input data. These properties are essential to evaluate Scribble Query with domain-experts, as we will describe in the next section.

4. Evaluation

We have deployed Scribble Query with versions of the multivariate data visualization shown in Figure 2 with domain experts from two organizations. The first is an orderly manager from a large hospital who needs to analyze hundreds of thousands of orderly tasks across organization, urgency, time, type of task, etc. The second is four customer relations employees from a large online retailer who need analyse tens of thousands of sales records across customer satisfaction score, delivery company, region, time, price type, etc. All participants worked with analysis of data in their job function with varying levels of non-programmer’s data analysis expertise.

We introduced Scribble Query and the multivariate data visualization to one orderly manager at the hospital and four customer relations employees from the online retailer. All domain experts were used to analyze data as part of their job function; a customer relations employee would e.g., based on customer contact, like to investigate causes for unusual high customer dissatisfaction amongst customers in a specific region, and the orderly manager would e.g. like to investigate whether certain departments at the hospital would request unnecessarily many urgent tasks. Two of the employees from the online retailer had experience with retrieving data needed for analysis from databases etc, while the two other employees and the orderly manager relied on colleagues for retrieving data and preparing data for analysis in software applications (commonly Microsoft Excel); e.g. creating pivot tables or simple dashboards.

In this section we briefly report on excerpts from the deployments of Scribble Query and the accompanying multivariate visualization. Both interfaces were similar to the one depicted in Figure 2; they differed only due to their domain specific datasets.

4.1. Excerpts from Deployments

At the introduction all domain experts expressed enthusiasm towards the Scribble Query and the accompanying multivariate visualization. However, although all live domain experts expressed ambition to use the tool, in the long term only the three domain experts who relied on colleagues have continued to use Scribble Query and the accompanying multivariate visualization to analyze data. This was especially true for the orderly manager who quickly adopted the tool, and Scribble Query in particular, into organization-political responsibilities. He brings live versions of the tool to meet-
ings with the hospital manager and employees from hospital departments to argue for man-power allocations performing live analyses of the orderly task dataset. This is supported by interviews we conducted early in the deployment process with the domain experts. In this two participants explicitly stated that they found the Scribble Query easy to decode and understand and that this supported them in performing iterated interaction cycles. This suggests that because the user’s scribbling interaction is persisted as a Scribble Query with high affinity to the user’s input, the Scribble Query technique also functions as an annotation that helps users to bridge the gulf of evaluation and execution [ND86]. Both for the data analysts as well as for observers.

The last lesson we will report on is the surprising observation that none of the participants used more than one finger to query the visualization with Scribble Query, even though the possibility was carefully introduced to the domain experts. For Scribble Query, although multiple simultaneous scribbling interactions are possible, we think this is due to the fact that advanced queries can be scribbled with a single finger touch-input. Put differently, multiple concurrent touches are not needed to query data with Scribble Query, at least not for these casual users.

However, to be able to evaluate or conclude on our speculation would require a larger evaluation with more domain experts, which is outside the scope of this paper.

5. Discussion

The excerpts from our deployment of Scribble Query with domain experts from two organizations indicates that free-hand scribbling of queries imposes a straightforward conceptual model of interaction on the user, which can assist in adoption of the interaction technique. This gives Scribble Query the advantage that it is able to leverage the intuitive and familiar interaction of touch-screen interaction. Furthermore, statements from participants suggest that Scribble Query’s high affinity with the user’s interaction assists users in understanding the result of their interactions and further react on them.

Deploying Scribble query with domain experts in organizations has limited the scope of our study of Scribble query. We still decided to study our deployment with domain experts because we wanted to investigate the feasibility of the Scribble Query interaction technique for conducting data analysis by professional data analysts. Another approach, which could have enabled us to reach firm quantitative conclusions, would be to perform a strictly controlled study with a significant number of participants on a more generic dataset that can be analyzed by a wider range of participants.

Nonetheless, we are confident that the Scribble Query interaction technique is indeed capable of facilitating easy access to complex visual analysis of multivariate data. We also believe that Scribble Query is highly promising for use by non-professionals and that it is also usable in non-professional settings, enabling a wider range of users—beyond data scientists—to perform data analysis. Furthermore, we are confident that Scribble Query is applicable to a wider range of data types than tested here. In Figure 7 we have applied it to a mock-up of a visualization of time-series data. More advanced time queries, on par with the queries made available by Timeboxes [HS04], would be easy to add.

Naturally, our Scribble Query technique has several limitations and weaknesses. One of the challenges of touch interaction is that many of the more advanced interactions are based on composite gestures—such as specific finger postures, kinetic movement, or multiple consecutive actions—that are not easily discoverable and require training. For example, many smartphones support zooming by double-tapping on text, an operation that a user may only hope to stumble upon if they are not explicitly taught to use it. The Scribble Query method represents another form of specialized touch gesture that is not easily discoverable and may require (possibly recurring) training. In fact, this is also an argument in favor of using standardized touch gestures, even for visualization. Similarly, additional limitations include low touch precision as well as finger occlusion, both exacerbated by the high-density visual displays typically used in visualization. Future work will have to explore the tradeoff between the extra strength provided by our customized gestures and their inherent weaknesses.

6. Conclusion and Future Work

We have presented Scribble Query, a new family of direct query mechanisms for interactive data visualization that was designed based on users’ familiar pinching and swiping touch behavior for multi-touch displays. Unlike existing interaction techniques for touch-enabled visualization, Scribble Query is not constrained to standardized touch gestures, but allow the user to fluently “scribble” their data queries using a single or multiple touch points. Our deployment study indicates propensity towards the new technique for our participants. We also provide implementations of the technique for both multivariate data in parallel coordinate plots as well as time-series data in a line graph.

While much work on touch-enabled visualization has
focused on mitigating the drawbacks of touch surfaces—such as occlusion, parallax, and low accuracy—there is comparably little work on taking full advantage of the unique strengths of the input medium for data visualization. We claim that novel Scribble Query technique has shown promise in utilizing touch-devices to provide intuitive and advanced data analysis for non-programmer domain experts. We believe that this is only a first step towards creating a large toolbox of novel analysis tools, harnessing touch in the future: to touch data, as it were. Finally, we have shown that such tools can be provided as cloud services utilizing standard web technologies to provide the interaction. This will allow a much larger user population of domain experts to enter the scene of big data analytics.

Acknowledgments

[Anonymized for double-blind review.]

References


submitted to Eurographics Conference on Visualization (EuroVis) (2016)


Publication 4

Structured Vector Graphics for Large-Scale Visualizations:
Performance, Optimization, and Practical Guidelines

I am main author on a ten-page paper
submitted to the 2016 EuroVis conference.
Structured Vector Graphics for Large-Scale Visualizations: Performance, Optimization, and Practical Guidelines

Abstract
Structured vector graphics using a scene graph representation has recently become very popular for creating interactive visualizations, especially propelled by extensive native support for Scalable Vector Graphics (SVG) in modern web browsers. However, when visualizing large datasets with many thousands of elements using structured vector graphics, rendering time and interaction latency deteriorates. In this paper, we enumerate three general techniques for optimizing structured vector graphics for visualization: aggregation, sampling, and progressive rendering. Our work is informed by careful performance measurements on rendering time and latency of vector graphics visualizations rendered in web browsers. Our performance experiments partition rendering time into three phases: (1) DOM manipulation, (2) style and layout, and (3) pixel rasterization. By distinguishing between these phases, we are able to document a detailed linear relationship between the number and types of elements being visualized and the time taken to create and render visualizations. These detailed measurements allow us to better understand how to optimize visualizations using structured vector graphics. Furthermore, we implement our three optimization techniques—aggregation, sampling, and progressive rendering—in three representative visualization techniques—scatterplots, parallel coordinates, and node-link diagrams—and measure their impact on performance using the aforementioned method. This yields insight into practical approaches for optimizing large-scale visualization in a web browser. Finally, our evaluation web service has been made publicly available to promote this type of performance evaluation in the future.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

1. Introduction
Vector graphics has become increasingly popular in recent years, particularly in the form of the Scalable Vector Graphics (SVG) format [Fer, Dah]. As support for SVG in modern web browsers has expanded, third-party JavaScript libraries, such as D3 [BOH11] and Vega [SRHH16], that leverage vector graphics to create interactive visualizations have emerged. However, web-based visualizations tend to scale poorly when designers encode large datasets into vector graphics in a straightforward manner, often resulting in complex SVG graphics with a large number of nodes. This may yield high rendering time and high latency, causing unresponsive interaction and poor user experience. In order to help visualization designers make informed decisions on how to handle such large datasets while retaining responsiveness in structured vector-based visualization, a firm understanding of the influencing factors as well as possible techniques for addressing them is required. This makes managing large datasets in modern web browsers a potent, and unaddressed, challenge for developers, who often rely on rule of thumb guidelines for how many data elements can be rendered while retaining interactive responsiveness.

Fortunately, there exists several general techniques for improving rendering performance and reducing latency of structured vector graphics. More specifically, we study how to apply aggregation, sampling, and progressive rendering techniques for this purpose. These techniques are based on insights gathered from an in-depth investigation of performance in vector graphics generation and rendering of basic visualization techniques, using two-dimensional, multidimensional, and network data, which we visualize with scatterplots, parallel coordinates, and node-link diagrams respectively. We implement and evaluate rendering performance of these visualization techniques in a modern web browser (Google Chrome) using the D3 JavaScript library [BOH11], and we describe our results in terms of the number of visual structures, the type of visual structures (e.g. dots and lines), and the pixel area size of the visualization.
From these results we are able to outline a generally applicable formula that describes the linear relationship between the number of elements in a dataset and the time taken to render it. We instrumentalize on this formula by providing it as a publicly available web service where developers automatically can calculate the detailed linear relationship particular specific to a particular device, browser, and a generic visualization technique.

The contributions of this paper are the following: (1) An in-depth investigation and measurements of how the number and type of elements the size of a visualization impacts the performance of creating and rendering vector graphics in a modern web browser; (2) A set of general techniques for managing and visualizing large data sets in web browsers by aggregation, sampling, and progressive rendering; and (3) A publicly available web-service enabling users to measure the detailed linear relationship between the number of elements and time taken to render them as a SVG visualization for particular devices, browsers, and generic visualizations.

2. Related Work

Visualization and visual analytics fields have been rapidly adapting to the advances in other computing domains including parallel, high-performance, and distributed computing, as well as graphics processing. This has enabled visualization systems to tackle the challenges of big data analytics [AKMHP09]. However, achieving ubiquity to support interaction with data anywhere and anytime [EI13] requires platforms such as the web to handle large scales of data. Here, we review the current technologies in web visualization, scalable visualization techniques for large data, and responsive visualization systems used in practical scenarios.

2.1. Web Visualization and Evaluation

Rohrer and Swing [RS97] formalized the idea of web-based visualization as using the web as an information source and as a delivery mechanism for visualization. Over the past decade, the web visualization frameworks have caught up with the new technological advances in web platforms including the standards of vector graphics. ProtoVis [BH09], a JavaScript toolkit, bridged the gap between designing high-level visualizations and mapping them to low-level vector drawings through an embedded domain-specific language. Later, D3 [BOH11] was developed to create web visualizations by directly manipulating the document object model (DOM)—the hierarchical structure of a web page content maintained by modern web browsers. This close coupling with the browser’s DOM allows maintaining the scene graph within the DOM itself, eliminating the indirection of an intermediate scene graph, and thus improving efficiency, compatibility, and performance, while supporting easy debugging. D3 has gained wide popularity over the past few years with many visualization systems built using this framework for structured interactive vector graphics over the web (e.g. [BE14,CSGB14,HVF13,YEB16,WMA*16]).

Targeting the web platform also implies dealing with the restricted computing and rendering abilities of modern web browsers. To combat this, current web visualization frameworks such as D3 [BOH11] and Reactive Vega [SRHH16] have optimized the web visualization pipelines, leading to better performance than ProtoVis and Adobe Flash technology. In an evaluation of initialization times and frame rates [BOH11], it was found that D3 had significantly faster initialization, but that Flash provided higher frame rates for animations. This study, however, only uses small dataset sizes (< 20,000 data points) compared to modern big datasets, and did not partition performance in terms of the time taken by the browser to carry out its internal stages of rendering. Recently, Reactive Vega [SRHH16], a system architecture for declarative specification of visualizations, has been released and shown to provide further performance gains over D3.

2.2. Scalability for Information Visualization

Visualizing big datasets on conventional displays can lead to overplotting, which overwhelms the user’s perceptual and cognitive capacities [LJH13]. Data reduction methods such as sampling [BS06,DSLG*12], aggregation [BB,SBA08,HDS*10,KPP*12], and filtering [AS94] have therefore been proposed to support perceptual scalability. More sophisticated versions of these techniques have also been proposed including kernel density estimation methods for specific visualization types [MG13] and hierarchical aggregation [EF10] to transform any visualization into a multi-scale visual structure. Beyond this, incremental query mechanisms have been explored for using databases to handle big datasets in visualization [FPD*12], especially in scenarios where rapid, less-accurate results are more valuable than slow, more-accurate results. ProgressiVis [Fek15] formalized the idea of progressive visualization, in which the computations driving a visual representation are split into small chunks and handled with an execution strategy to provide visual information incrementally as each chunk is processed. A major advantage of progressive visualization is the ability to interrupt the progression to allow user interaction at any stage without significant delays. Other techniques for dealing with complex graphical representations include billboarding—using pre-rendered 2D graphics such as raster images whenever possible to reduce graphics processing.

There have been many visualization systems proposed over the past years to handle big datasets using the aforementioned techniques and more. Fekete and Plaisant’s attempt at visualizing a million items [FP02] explored techniques based on hardware acceleration, non-standard visual attributes (stereovision), and using time multiplexing techniques (animation). Shneiderman [Shn08] reviewed attempts at handling even larger datasets (scaling to a billion
records) using rapid aggregation, multiple views, and summary graphics. Stolte et al. introduced a multi-scale visualization system with data cubes—a unique combination of values for the data dimensions—for providing meaningful data abstractions [STH03]. Sundara et al. [SAK*10] tackled the challenge of handling large-scale Resource Description Framework (RDF) data using subsets, summaries, and sampling implemented into Oracle. Apart from this, commercial tools such as Tableau [Tab] can translate user interactions into data queries that are processed by dedicated servers.

More recently, Kandel et al.’s Prolifer [KPP*12] used data cubes to interact with a million data points, while imMens [LJH13] targeted interactive scalability for panning, zooming, and brushing and linking in aggregated plots through WebGL for data processing and rendering at the GPU. Benchmarks revealed that imMens could handle interaction on billions of data records in real-time, with a sustained performance of 50 fps for brushing and linking.

3. Framework: Web Visualization

Vector graphics, specifically Scalable Vector Graphics (SVG), have recently begun to be utilized to create interactive data visualizations in modern web browsers. This can be largely attributed to SVG being a standardized technology [Dah], that is platform-independent across many modern web browsers for Windows, MacOS, Unix, and also mobile platforms. We investigate techniques for handling and creating visualizations of large datasets, and base our conclusions on thorough empirical studies of browser rendering performance of SVG visualizations.

3.1. Visualization Pipeline

Many variations of the visualization pipeline—the steps involved in transforming data into interactive visuals—have been proposed so far. Card et al.’s version of the pipeline [CMS99] containing data analysis, filtering, mapping, and rendering steps has been very popular in the field of interactive visualization. This, and many other pipelines abstractions [VW05,KAF*08], are more suited for native visualizations systems developed for specific target platforms and user groups. In contrast, the inherent advantage of web visualization is ubiquity and universal access, and this creates a reliance on the web browser to automatically handle the final steps of the visualization pipeline (rendering).

Web visualization frameworks such as D3 [BOH11] connect visual representations, created by combining low-level graphical primitives into a scene graph, to the browser’s DOM. Once the necessary transformations are performed from data to visual abstractions (e.g., shapes) and presentation attributes (e.g., position and color) [TIC09], the browser’s internal thread updates the DOM, computes layout and style, and paints the pixels. The pipeline for web visualization from a browser’s perspective is described in Fig. 1 and explained in detail in the next section.

3.2. Data Models

The process of transformation of raw data into visual elements expects a structured form of data storage at some level to associate items in a dataset with graphical primitives. However, many modern domains deal with raw data (e.g., text), which requires significant pre-processing to convert into relational data tables with hierarchies. Beyond this, the visualization pipeline itself expects data transformations between subsequent stages [Ch00]. Examples include (1) visualization transformation using algorithms such as Multi-Dimensional Scaling (MDS), and (2) mapping transformation for layout computation (e.g., force-directed layout). In web visualization, systems such as imMens [LJH13] have processed large amounts on the web browser using parallel architectures. Furthermore, the choice of the transformation algorithms is dependent on the visualization design, and can often be performed offline. Therefore, we assume availability of processed structured data on the browser cache for the JavaScript code to create the visual representations, and evaluate the rendering performance for SVG visualizations.

4. Performance Evaluation

In this section, we outline our test methodology and report in detail on the performance of visualizing datasets using SVG in a browser. In our experiment, we measure these time periods for conventional implementations of scatterplots, parallel coordinates, and node-link diagrams, implemented using D3 [BOH11]. Based on these measurements, we are able to demonstrate a detailed linear relationship between the number of elements in a dataset and the time taken to render the dataset as a visual structure. We then apply scalable techniques including sampling, aggregation, and progressive rendering for balancing responsiveness, rendering performance, and data granularity in browser-based visualization of large datasets. Our results provide an in-depth understanding of what factors are at play when rendering SVG visualizations in web browsers, and thus enable visualization developers to make informed decisions of how to achieve a desired responsiveness.

4.1. Test Methodology

We focus on controlled experiments where we perform automated tests by changing multiple factors (one at a time). In Section 4.2, we independently vary the number of data elements being visualized, the type of visualization (thereby also the visual structures), and the size of the visualization. In Section 4.3, we independently vary the aggressiveness of our aggregation technique and the number of elements for our progressive rendering technique. We do not report on performance of the sampling technique because, as we will
discuss in Section 4.2, rendering performance is linearly proportional to the number of elements being visualized.

To test responsiveness on interaction with SVG visualizations in a browser, we make a worst-case assumption based on the deliberation that the worst-case result of an interaction is that the entire visualization must be redrawn. Therefore, we rigorously measure the time it takes to render a SVG visualization from data over DOM manipulation and styling to rendered visualization, but understand this as the worst-case consequences of interactions. This means that we do not measure particular interactions or interaction techniques. Furthermore, we perform measurements of up to 100k elements, which might be considered unrealistic or impractical to visualize with SVG in a browser. The data is, nevertheless, from publicly available real-world datasets: for scatterplots and parallel coordinates we use a dataset of over 75,000 Amazon user communities.

All measurements are performed under the same conditions on a consumer-grade all-in-one computer with an Intel i5-4200U CPU with a base frequency of 1.6 GHz and turbo frequency of 2.6 GHz, a 20" 1600×900 resolution monitor, and running the Microsoft Windows 10 operating system. The computer performed no other tasks than rendering performance measurements reported here. All renderings of visualizations were made in a standard version of Google Chrome version 46 Stable with GPU acceleration disabled. The browser windows always had focus, and all pixels of rendered visualizations were visible inside the browser’s viewport without scrolling.

### 4.1.1. Browser Primer

When creating a SVG visualization, the browser performs the following operations (see also Figure 1): (A) The DOM is manipulated by inserting or manipulating SVG nodes, (B) the nodes in the DOM are styled and the webpage’s layout is computed (henceforth style and layout is referred to only as style/styling for brevity), and (C) the visual elements are painted. The exact implementation, and hence performance, will differ from browser to browser but all browsers need to perform these operations. We considered evaluating our implementations in the four major desktop web browsers: Apple Safari, Internet Explorer, Mozilla Firefox, and Google Chrome. However, we ruled out Apple Safari and Internet Explorer because they are not fully open source, meaning the inner workings of the browser are not publicly known. Consequently, we were forced to disregard Mozilla Firefox as well because it proved unstable when handling the large number of elements over multiple iterations. Therefore, the technique for measuring time taken by a browser to manipulate the DOM, style elements, and paint pixels outlined here (Listing 1) is specific to Google Chrome. However, similar measurements can be made in other major browsers (Mozilla Firefox, Internet Explorer, etc.) with small modifications, which are beyond the scope of this paper.

![Simplified pipeline of browser workflow.](image)

We measure DOM manipulation, styling, and painting time concisely using JavaScript timeouts and requesting animation frames at four selected points (also indicated in Listing 1): DOM manipulation is measured by noting the timestamp before (1) and after (2) a loop that manipulates the DOM. Styling time is measured by requested a function to be executed after a 0 millisecond delay (3) immediately after the DOM manipulation loop is done. This works because, in JavaScript, the function passed to a timeout is only requested to be executed after the specified delay, but it will at earliest be executed after the browser’s main thread becomes available, which is after the DOM manipulation is finished. Finally, we measure the time taken to paint the pixels by requesting an animation frame that is passed a function that will execute before the next paint cycle (4), which happens earliest when the current painting is complete.

### 4.2. Number of Elements, Visual Structures, and Size

Our first series of performance tests involved measuring the effects of number of elements being visualized, the visual structure, and the size of the visualization, on the SVG rendering. To this end, we used standard implementations of scatterplots, node-link diagrams and parallel coordinates made with D3. This choice is based on the fact that these atomic visualizations [Shn08] are both capable of visualizing large number of data items and are prone to overplotting, which is undesirable as it means that a pixel can be painted multiple times. Furthermore, they differ in their visual structures, and therefore the number of pixels that the web browser needs to paint.

```
var startTime, domManipulationFinish, stylingComplete, paintingComplete;

(1) startTime = time.now();
    d3.selectAll("elementName")
        .data(dataArray).enter()
        .append("elementName")
    ...;

(2) domManipulationFinish = time.now();
    setTimeout(function () {
        (3) stylingComplete = time.now();
            requestAnimationFrame(function () {
                (4) paintingComplete = time.now();
                });
            }, 0);
    }
```

Listing 1: Simplified JavaScript script for measuring DOM manipulation time, styling time, and paint time.

We measure DOM manipulation, styling, and painting time concisely using JavaScript timeouts and requesting animation frames at four selected points (also indicated in Listing 1): DOM manipulation is measured by noting the timestamps before (1) and after (2) a loop that manipulates the DOM. Styling time is measured by requested a function to be executed after a 0 millisecond delay (3) immediately after the DOM manipulation loop is done. This works because, in JavaScript, the function passed to a timeout is only requested to be executed after the specified delay, but it will at earliest be executed after the browser’s main thread becomes available, which is after the DOM manipulation is finished. Finally, we measure the time taken to paint the pixels by requesting an animation frame that is passed a function that will execute before the next paint cycle (4), which happens earliest when the current painting is complete.
4.2.1. Number of Elements

Figures 2, 4, and 3 show rendering times of scatterplots, node-link diagrams, and parallel coordinates respectively. The general observation, besides that scatterplots are faster to render than node-link diagrams and much faster to render than parallel coordinates, is that the time taken for rendering is linearly proportional to number of elements. This result is partially a consequence of using vector graphics as overplotting where elements and thus individual pixels can be rendered multiple times. This stands in contrast to bitmap graphics, which would render each pixels once and therefore likely would see rendering flatten in a logarithmic fashion as the number of elements increase.

The takeaway here is that the basic premise for responsive rendering is to render or update as few SVG nodes as possible. We exploit this later, when we outline techniques for visualizing large datasets using aggregation and sampling.

4.2.2. Visual Structures

In Figures 2, 4, and 3, the time periods of the renditions of the standard scatterplot, parallel coordinates, and node-link visualizations are also divided into the DOM manipulation time, styling time, and painting time. These figures reveal that the cause of the high difference in rendering is mostly due to a much longer painting time for parallel coordinates, and a slightly longer paint time for node-link diagrams. This is because Chrome (and Firefox alike) updates the viewport following a commonly applied “dirty rectangle” principle where the viewport is divided into rectangles and each element that is to be painted or repainted marks rectangles that the element intersects as dirty, triggering the browser to repaint the rectangle. Therefore, since a dot in a scatterplot will intersect fewer of the viewport’s rectangles than a path in a parallel coordinates visualization, it requires fewer rectangles to be painted. Because node-link diagrams has a combination of nodes and edges, its painting time is likely to be somewhere between that of scatterplots and parallel coordinates. However, because a node-link diagram’s paths are generally shorter than those of a parallel coordinates of the same size, the painting time for node-link diagrams is closer to scatterplots. Painting time can be substantially lowered using GPU acceleration. However, performance improvement due to GPU-acceleration will vary from device to device, and will not alleviate DOM manipulation and styling, which is influenced by the complexity of the visual structures.

Looking only at scatterplots and parallel coordinates, we see that, besides the difference in painting time, a small, but still notable, increase in DOM manipulation time and styling time. This is because a circle in scatterplots is simpler to describe (it requires one x, y coordinate set and a radius) than a path in parallel coordinates, which requires a series of x, y
coordinate sets. Therefore, the time taken to insert a path into the DOM will take longer than a circle node.

The takeaway is that SVG visualization developers should consider the complexity of visual structures—using simpler visual structures can enable the visualization to cope with a larger number of nodes and still remain responsive.

4.2.3. Size

In Figure 5, the rendering time of scatterplots and parallel coordinate sets renderings of 50k elements and node-link diagrams renderings of 47,547 elements is shown as the physical area of the visualization is changed. The area sizes are increased linearly, with area of 1x being the size used for all other tests reported in this paper. The complexity of the visualization is similar to the previously reported—the only difference here is that the nodes are spread on a larger area.

As seen in Figure 5, the time taken to render a scatterplot and node-link diagram increases slightly while the time to render the parallel coordinates increases sub-linearly to an increase in area size. These trends can be connected back to the “dirty” rendering protocol of the web browser. In case of parallel coordinates, the path will intersect with more “dirty” rectangles as size increases, but the total number of intersected rectangles does not increase at the same rate as the area of the visualized.

The takeaway is the relationship between size and visual structure. Complex visualizations that encode paths can have sub-linear relationships with size—small size changes in a bounding box may not drastically affect the rendering time.

4.2.4. Linearity Between Time and Number of Elements

Summarizing the results from the performance tests (Figures 2, 3, and 4), we observe that even with the large difference between parallel coordinates vs. scatterplots and node-link diagrams, there is a linear relationship between the number of elements being visualized in a SVG and the time taken to render them. This includes the individual stages of DOM manipulation, style computations, and painting. The following equation details this observation: \[ t = (d + s + p) * e + b \]

Time \( t \) can then be calculated by adding a DOM manipulation factor \( d \), a styling factor \( s \), and a painting factor \( p \), scaled by the number of elements \( e \). Finally, externally determined factors, such as data transmission, data processing, and data transformation times can be added as a constant \( b \). Since these variables change with the target device, web browser, and the visual structures of the visualization (seen in Figures 2, 4, and 3), we have developed a web service [blinded for review] for determining \( d, s, \) and \( p \) factors for a particular device. We will elaborate on this webservice in Section 5.2.

4.3. Techniques: SVG Visualization of Large Datasets

In the previous Section 4.2, we saw that for visualizations of the same size rendering time increases linearly with the number of elements being visualized. In this section, we describe techniques for managing large datasets and empirically evaluate the performance or responsiveness benefits on scatterplots, parallel coordinates, and node-link diagrams. The techniques we describe are sampling, data aggregation, and progressive rendering. These techniques differ fundamentally in how they are applied to datasets—sampling visualizes a subset of the data, aggregation visualizes all data but with less detail, and progressive rendering visualizes all data slowly but with high responsiveness.

4.3.1. Sampling

Data sampling to improve responsiveness is a seemingly simple process of selecting a subset of a dataset and only visualizing the subset. Technically, the motivation and result is straightforward—rendering fewer elements results in faster rendering time, as seen in Figures 2 and 3. However, sampling requires careful deliberation because rendering a subset of a dataset means that the performance gains may come at the cost of losing important detail in the visualization. As there is a linear relationship between the number of elements and rendering time, we do report performance measurements because a sample of a dataset is just a smaller version of a larger dataset.

One way to mitigate the loss of detail from sampling is to investigate the distributions of data variables and select representative samples following similar trends including minima, maxima, and statistics of other orders. This approach can be relatively painless in a scatterplot, where dots signify the pairwise relationship between just two data dimensions. However, sampling of paths in parallel coordinates is more complex, as investigated in-depth by Dasgupta and Kosara [DK10], who introduced Pargnostics for layout management based on screen-space metrics. Furthermore, sampling can alleviate overplotting and clutter as discussed in detail by Heinrich [HW13], Ellis [ED06] for parallel coordinates, and by Bertini for scatterplots [Ber04].
To instrumentize sampling techniques for SVG visualizations, it is important to consider a visual budget denoting a maximum number of elements to visualize or the delay time that a target user can withstand. As discussed previously in Section 4.2.4, this will differ across device, browser, and type of visualization. However, using our web service, described in Section 5.2, it is possible to determine a visual budget for a visualization that can be approximated in an actual implementation. We say “approximate” as it is impossible to predetermine criteria for sampling (e.g., retaining minimums, maximums, and distributions) that will guarantee a sample size, especially for high dimensional datasets.

4.3.2. Aggregation

Like sampling, aggregation improves performance by rendering fewer elements. Contrary to sampling, however, aggregation seeks to visualize all data in a dataset, albeit with a loss of detail as similar entries are grouped or discretized. Frequency in a group can then be indicated by size, color, and opacity of a visual structure. Discretization, or binning, of a dataset can be achieved while retaining some other statistics, such as averages and extrema, as investigated for parallel coordinates by Novotny and Hausner [NH06].

We have implemented generic aggregation algorithms for scatterplots, parallel coordinates, and node-link diagrams—see Figure 6 for example—and evaluated their performance. Note that because aggregation techniques will differ across visualization techniques, the numbers are not directly comparable. Nevertheless, we can observe clear trends in the performance implications of aggregation.

As the two-dimensional data for a scatterplot is binned, the uniform dots are transformed into simple circular glyphs whose size denotes the number of entries in each bin. Because binning for a scatterplot is performed on two dimensions, it is possible to perform a relatively fine-grained aggregation while retaining the rendering performance. This is seen in Figure 7, where we can see an initial rapid increase in time taken to render aggregated scatterplots until it flattens out as the bin count rises and number of aggregated plots nears a maximum similar similar to the rendering time of non-aggregated data.

Aggregation of data for parallel coordinates needs to be much more aggressive than for scatterplots. This is simply because of the, commonly, many more data dimensions in parallel coordinates leading to significantly higher combinations of bins. As we can see in Figure 8, this means that the time taken to render an aggregated multidimensional dataset quickly approaches, like the aggregated scatterplot, a maximum similar to the rendering time of non-aggregated data. Rendering time will, however, vary greatly with the number of dimensions in the data, and the numbers reported in Figure 8 should only be used to understand the general trend.

Aggregation in graph (node-link) visualizations typically happens by collapsing community structures—connected nodes in the graph—based on a connectivity or proximity measure. To conduct a rigorous analysis of the performance in relation to the number of neighbors collapsed, we introduced an aggregation parameter $\alpha (\alpha \in [0, 1])$ that dictates how many connected nodes are combined during aggregation. This means that if, say, $\alpha = 0.5$, then 50% of the connected nodes in each community in the dataset are collapsed into one single node. The analysis of the total time taken (Figure 9) for rendering aggregated node-link diagrams shows a linear change across aggregation factors, with the painting time dominating others for low aggregation levels (e.g., $\alpha = 0.1$). This is due to the presence of more edges connecting nodes for low aggregation levels, leading to higher number of pixels that have to be painted on the screen (following a similar argument as visual structures in Section 4.2).

4.3.3. Progressive Rendering

Progressive rendering ensures responsiveness in rendering data without loss of detail by rendering parts of a visual structure at a time. This means that the browser completes a full circle of DOM manipulation, styling, and painting for each visual structure. In practice, progressive rendering is implemented using a function for manually iterating through an array of data elements to visualize. However, instead of self-instantiating the function, the function sets a timeout
with a 0 ms. delay, as depicted simplified in Listing 2. The timeout facilitates that the visualization iteration can be terminated at each iteration, e.g. due to user interaction, thus ensuring responsiveness. The idea to progressively render visualization to show partial visualizations while allowing for user interruption is related to ProgressiVis [Fek15]. However, the progressive rendering technique presented here (in Listing 2) differs from ProgressiVis because our technique is implemented exclusively on the client.

However, this responsiveness to user interaction comes at the cost of rendering time, as seen in Figure 10. We can observe that all three visualizations take approximately the same time to render, despite having a very different rendering performance when rendered naively (Figures 2, 4, and 3). This is because the HTML5 specification specifies that a nested timer instantiation must enforce a minimum delay of 4 ms [Hic], meaning that, at most, 250 elements can be rendered per second using progressive rendering. A way to mitigate this restriction is to render elements in small batches and only use a timeout triggered iteration for every e.g. 20th or 50th iteration, which, however, could introduce perceivable stuttered rendering.

The take-away from progressive rendering is that while it can be utilized to ensure responsiveness to user interaction, it is not appropriate for large, unless long waiting times for rendering to conclude are acceptable.

![Diagram](image)

**Figure 8:** Detailed time periods in seconds (y-axis) for rendering of aggregated parallel coordinates with 50k elements and a bin count on each dimension ranging from 5 to 50.

![Diagram](image)

**Figure 9:** Detailed time periods in seconds (y-axis) for rendering of aggregated node-link diagrams with 50k nodes and edges typically between 5k and 50k. The aggressiveness of the aggregation is depicted horizontally from 0.9 (heavy aggregation) to 0 (no aggregation).

![Diagram](image)

**Figure 10:** Time in seconds for progressively rendering scatterplots, parallel coordinates, and node-link diagrams of 1k to 10k elements.

```javascript
var svgParent = d3.select(body).append("svg");
var timeoutID;

function appendElement(data, index) {
  if (index === data.length) {
    return;
  }
  svgParent.append("elementName")
    .datum(data[index])
    ...;
  timeoutID = setTimeout(function () {
    appendElement(data, index + 1);
  }, 0);
}
appendElement(data, 0);

cancelTimeout() {
  clearTimeout(timeoutID);
}

Listing 2: Simplified JavaScript implementation of progressive rendering.
```

5. Discussion

First of all, because we investigate client-side renderings of datasets we are limited to rendering less than 100,000 data points, which is beyond the SVG rendering capabilities of most current browsers. In contrast, some web visualization systems have tackled billions of data items [LJH13] using databases and server-side technologies. However, even in such systems, the browser is left to deal with orders of thousands of processed data items to render.

When we started our investigation of rendering browser performance of SVG visualizations, we expected to just see that rendering time changes linearly with number of data items used for the visualizations seen in this paper. We were,
however, surprised to discover that there is a much more nuanced relationship between browser internals and the type of visual structures being visualized, as revealed during our evaluations. This underlines the need for techniques that can visualize large datasets with a sub-linear relationship between the number of elements and the time taken to render them, because naïvely visualizing large datasets in browsers using SVG can cause serious performance implications.

5.1. Limitations

The detailed performance measurements reported in this paper may not be generalizable across the very diverse landscape of devices and browsers all capable of rendering SVG visualizations. However, it is an impossible task to undertake in-depth investigations of all possible combinations of devices and browsers available. Therefore, we have limited our in-depth studies to one device and a single browser to show the general trends in the relationship between the number of elements and time taken to render these elements. To generalize and instrumentalize our results, we have developed a web-service [blinded for review] for determining a detailed linear relationship for a particular device.

Furthermore, hardware advancements in devices, such as increases in processing power, memory availability, and GPU-acceleration, will decrease the slope of a linear relationship between the number of elements being visualized and the time taken to render them. Nevertheless, hardware advancements will only shift the starting point of this poor performance in large-scale SVG visualization, because it will still occur for an even higher number of elements. Therefore, techniques that facilitate a sublinear relationship between the number of elements and time to visualize them are more ideal (as quantified by our performance results).

However, applying sampling, aggregation, and progressive rendering techniques for visualizing large datasets in a browser comes at the cost of losing details or adding latencies. Our performance measurements bring in a systems perspective in handling big data to a visualization designer. We believe that this work therefore complements the research in understanding the cognitive effects of interaction latencies [LH14] and optimization of visualization processes [CG15] in general.

5.2. A Benchmark Webservice for SVG Rendering

As noted in Section 4.2.4, a detailed linear equation for SVG rendering performance will be specific to a particular device, a particular browser, and the visual structures of the visualization. Therefore, we have developed a publicly available web service [blinded for review], which users can visit using their own device and choose a visualization technique in order compute the DOM manipulation factor, styling factor, and paint factor for specific use cases. Currently our web service is only available for desktop versions of Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, and Microsoft Edge browsers. However, we are working to expand its functionality to also cover Apple Safari and mobile browsers. We will continue to maintain and expand our web service in order to collect data on different devices’ capabilities for rendering SVG and we will publish result data results performed on specific devices to help visualization designers determine an appropriate number of elements to visualise.

6. Conclusion

While other approaches to visualizing large datasets in browsers are capable of much larger datasets [LHJ13] than the datasets used in our evaluations, there currently exist little work that details and instrumentalizes the nuanced relationship between dataset size and rendering time when creating SVG visualization in a browser.

Based on in-depth investigations of browser rendering performance of SVG visualizations, we have detailedly describe the nuanced relationship between browser internals, dataset size, and type of visualization. Because such a detailed investigation with a single browser on a particular device can be non-straightforward to apply prospectively, we provide and evaluate a set of sampling, aggregation, and progressive rendering techniques to instrumentalize our investigations. We provide these techniques, together with a webservice (see section 5.2) for determining the detailed relationship between particular devices, browser, and generic visualization techniques, to instrumentalize and generalize the results of our investigations.

References


[Ber04] In Smart Graphics, vol. 3031 of Lecture Notes in Computer Science. 2004. 6


Submitted to Eurographics Conference on Visualization (EuroVis) (2016)
Publication 5

Towards Highly Affine Visualizations of Consumption Data from Buildings

I am main author on an eight-page peer reviewed paper presented at the 2015 IVAPP conference.
TOWARDS HIGHLY AFFINE VISUALIZATIONS OF CONSUMPTION DATA FROM BUILDINGS

Matthias Nielsen¹, Kaj Grønbæk¹

¹Department of Computer Science, Aarhus University, Aabogade, Aarhus, Denmark
{matthiasnielsen, kgronbak}@cs.au.dk

Keywords: Interactive Visual Analytics, Spatial Visualizations, Energy Management Support.

Abstract: This paper presents a novel approach AffinityViz to visualize live and aggregated consumption data from multistory buildings. The objective of the approach is to provide a generic but high affinity relation between real buildings' spatial layouts and the consumption data visualizations. Current approaches come short on maintaining such affinity. This implies an avoidable cognitive load on users such as energy managers and facility managers who need to monitor consumption and make decisions from consumption data. To alleviate this we have transformed three conventional types of visualizations into highly affine visualizations lowering the cognitive load for users. The contributions are: 1) Development of the AffinityViz techniques featuring three generic designs of highly affine visualizations of consumption data. 2) Comparison of the affine visualizations with the conventional visualizations. 3) Initial evaluation of the AffinityViz designs by expert users on real world data. Finally, the design challenges of AffinityViz are discussed, including prospects for AffinityViz as a future tool for visual analysis of data from buildings.

1. INTRODUCTION

The research behind this paper took place in the EcoSense project (EcoSense, 2014), where we study human energy related behavior in a dorm living lab equipped with multiple sensors continually monitoring consumption data from the dorm apartments (Blunck, H., et. al., 2013).

As energy and resource consumption data in large buildings is collected at an increasingly granulated level from sensors in modern buildings, it is necessary to rethink how such data is visualized. Although existing types of visualizations are technically capable of visualizing high granularity consumption data from multistory buildings, novel visualization techniques are needed to create visualization that cater to a broader spectrum of professionals wanting to analyze such data. This applies to use cases where building administrators need to analyze and understand patterns in consumption to better understand requirements for infrastructure revisions or building upgrades. Another use case is interventionists (researchers, administrators, or others) who want to launch initiatives to lower consumption and therefore need to understand which parts of the building, or which tenants, are evident targets. This means that professionals from a wide range of disciplines could need to analyze buildings consumption data, and that they need to analyze varying types of data, such as consumption of water, electricity, district heating, etc.

In the design of AffinityViz, we exploit that many multistory buildings such as office buildings and apartment buildings have a simple recurring physical layout across office/apartment size and floor plans, by rendering a simplified 3D layout plotted with data points representing single units (apartments or offices) in the building. This results in a novel visualization technique that leverages user understanding of visualized data by retaining a building’s spatial layout.

The paper is structured as follows. First, we describe related work on conventional visualization techniques as well as current state of the art in visualizations of consumption data from buildings. Second, we discuss how conventional visualization techniques can be adapted to become affine visualizations as well as a more radical highly affine visualization. Third, we elaborate on the design of AffinityViz –in terms of current implementation as well as envisioned enhancements. Fourth, we elaborate on the implementation of the current prototype and lessons learned from testing it with facility managers. Finally, we discuss AffinityViz
and future work on tools for visual analysis of consumption data.

2. RELATED WORK

This review covers conventional visualization techniques appropriate for visualizing consumption data from multistory buildings, examples of usages of 3D data representation in information visualization, examples of state of the art in spatial and volume-based visualization, examples of academic work in energy consumption visualization, and a related architecture concept.

Cluster based heat maps (Wilkinson, L., et. al., 2009) visualizes data in a matrix using color to represent data values. They have wide-ranging applicability and excel in visualizing ordinal data while retaining hierarchies in the data. Jacques Bertin (Bertin, J., 1969) discusses the use of 3D topographic reliefs to visualize data from nations or regions in effect creating 3D cartograms. Reliefs are extruded to represent data values of topographic areas and the reliefs themselves serve as contextualization.

Perspective visualization have been explored as general information visualization interface technique (Carpendale, M. S. T., et. al., 1995). They discuss in-depth different ways of handling distortion of graphs when visualizing data in three dimensions. Wright (Wright, W., 1995) has pioneered 3D information visualization for applications in capital markets. Wright constructs 3D scenes, plotted with abstract 3D geometrical objects, which users can navigate and explore. Wright’s work builds upon the 3D user interface design paradigm, Information Visualizer, developed by Robertson et. al. (Robertson, 1993).

Power BI for Office 365 (Microsoft, 2014) is a plugin visualization tool for Microsoft Excel that supports overlays on 2D maps, viewing maps from tilted angles, creating a 3D view, and plotting data with a geospatial reference onto the map as 3D histograms. Other general tools for data visualization include Tableau (Tableau Software, 2014), a BI tool creation of interactive visualizations and dashboards. Data-Driven Documents (Bostock, M., et. al., 2011) is a multipurpose JavaScript library for transforming datasets into web browser DOM elements.

The New York City Energy Usage Map (Howard, B., et. al., 2012) is an interactive map that plots energy usage on block and lot level in New York City, creating a high detail cartogram. Data is visualized as polygons that are colored according to energy usage in terms of kWh per m². A contemporary practice of consumption data is to create an interactive visualization dashboard that visualizes resource consumption data in faceted views. Examples include Lucid’s BuildingOS and Building Dashboard (Lucid, 2014) and BuildingsAlive (Buildings Alive, 2014), all products using composition of visualizations to support visual analysis of consumption data for various settings. Cube Lease (Cube Cities, 2014) visualizes entire floor plans or single leases superimposed onto renderings of the shape of large multistory buildings. South Korean studio randomwalks has proposed a futuristic architectural concept, Data Formation (randomwalks, 2009), which interconnects the resource consumption of inhabitants in a tall rise building with their physical habitat in order to create a symbiotic relationship.

3. DATA ABOUT BUILDINGS.

In section 2, we saw examples of data with a geospatial reference to locations. But, we did not find any that relate data to the spatial layout of buildings. Visualizing data with a geospatial reference in a layout adhering to the geospatial reference, such as the topographic reliefs discussed in (Bertin, J., 1967), is a commonly used technique. It creates a direct relation between the data and the location of its origin and uses a familiar spatial layout of territories rather than abstract textual descriptors and graphs. The same argument can be applied to visualizations of consumption data from large multistory buildings – by retaining the spatial layout of a building in a visualization of data from the building, we use a familiar layout and lessen the cognitive load on the user. It is, however, not a straightforward to retain the spatial layout of a building when visualizing data from a building.

One approach is to model true to a building or its shell and visualize data using overlays. However, this would limit the visualization to the particular building and limited its generalizability. Instead, we propose the AffinityViz techniques (Figure 2, Figure 4, Figure 6) to adhere to a simple model that retains the spatial layout of a building and is generalizable across multistory buildings with a simple recurring layout as well as it is implementable in programming environments that can render visual elements. Using simple geometric objects and shapes in a 3D scene is similar to the Wright’s approach (Wright, 1995), but in AffinityViz the spatial layout of the scene is a familiar reference, like the relief topographic (Bertin, J., 1967) only to a building instead of a territory.

We have created three AffinityViz designs that retain the spatial layout of a building through a
simplified model of the building. Each design represents data differently, but derived from or inspired by conventional visualization techniques. Two are derived directly from cluster based heat maps (Wilkinson, L., 2009) and area maps (Tableau Software, 2014), and the third is inspired by bar charts, but makes a radical leap beyond these. Below we elaborate on the underlying assumptions of a building and its consumption measurements before we discuss and compare our designs to similar conventional visualization techniques. The visual representation in AffinityViz relies on that the real building being analyzed has a comparatively simple layout. This excludes certain types of large buildings that have complex shapes such as the Sydney Opera House and the Gherkin in London.

3.1 Data from Multistory Buildings

Consumption data in large buildings can be gauged for a number of resources. For AffinityViz, we assume consumption data is a continuously measurable resource, such as electricity consumption, gas usage, district heating, etc. The resource consumption itself is assumed to take place in a particular unit out many similar units – e.g. an apartment or office. These units will have a spatially significant location in the building in the form of a [floor, room] symbolic coordinate. Floors, we can assume, are ordinal, meaning that they are categories of data that have an interrelationship that can be ordered – i.e. floor 12 is a higher floor than floor 11. Rooms, on the other hand, can only be assumed to be discrete, meaning that may or may not follow a spatially sequential order.

3.2 Foundations of AffinityViz

We have developed three AffinityViz designs – AffinityHeat, AffinityArea, and AffinityBar – by exploring strengths and shortcomings in conventional visualization techniques applicable for visualizing building consumption data. Here we discuss our three AffinityViz designs in relation to the founding conventional visualization techniques. Legends are omitted to emphasize the visual representations and all examples visualize the same data ordered in the same way.

3.2.1 From Heat Map to AffinityHeat

The cluster based heat map (Wilkinson, L., et. al., 2009) is used for visualization three dimensional data in a uniformly distributed 2D matrix of fixed size rectangles with color or color intensity for conveying data values. Data points in a heat map can vary greatly in granularity from high granularity visualizations showing gradual transitions to low granularity categorical steps between boxed data points. Either way the layout of a heat map is commonly a meaningful ordinal layout of e.g. geo coordinate based data of high or low granularity.
coordinate system, the affinity between the visualization and the real building is lowered considerably. Although a complete overview might be desired in some circumstances its abstraction away from the real building’s layout introduces a mental indirection as the user is required to mentally map a data point in the heat map to an actual apartment. This is depicted as AffinityHeat in Figure 2, with the same data and ordinal layout as Figure 1.

For infrequent users an abstract layout can imply a recurrent comprehension cycle. For users who are familiar with the actual building and its spatial layout resource consumption data can be visualized with considerably higher affinity by complying with the ordinal layout of floors and rooms in three dimensions instead of just two dimensions. This means that an important property of the real building is retained in the visualization, namely that data points wrap the same way apartments do in the real building. This means that just like one would expect, on, e.g. a floor with 14 apartments, that apartment 2 and 3 are next to each other, so will apartment 14 and 1 be neighbors. In a cuboid building this will conceal three of the six surfaces, but by making it rotatable all surfaces can be viewed, though not at the same time. The issue of lacking overview is lessened by the heat map’s usage of color intensity to visualize data, because outliers and patterns will still be conspicuous. Only now, outliers or patterns that are a product of their surface will become easier to identify, such as whether surfaces with high solar radiation has lower heat consumption.

By visualizing data on a rectangular cuboid building structure, patterns in data points grouped by surfaces of the building become more apparent, and data points wrap the visualization in a manner true to the real building. Thus, by sacrificing complete overview, it is possible to create a direction relation to the spatial layout of the real building, while also retaining the visual properties of the heat map and lower the cognitive load on the user.

3.2.2 From Area Map to AffinityArea

The area map utilizes its other visual dimension – the 2D area of its data. This is appropriate in high granularity heat maps, but for low granularity heat maps, with relatively few data points, substituting color or color intensity with area size frees up color as visual dimension to encode other properties of a dataset. This is done in area map. An area map version of a heat map is shown in Figure 3.

As in a heat map, outliers are easy to detect in an area map because a considerably large or small areas are conspicuous compared to similarly sized areas. Comparison of two resembling areas, however, becomes more difficult because a data value is encoded as area, which is the product of two lengths multiplied, meaning that two spatial dimensions must be compared concurrently. Nevertheless, freeing up color means that this visual dimension can be used to encode surfaces of a large building by grouping data using a distinct tone for each surface.

Figure 4: AffinityArea visualization of building consumption data.

Although individual surfaces becomes distinguishable a 2D representation of the area map otherwise share similar drawbacks and advantages as the 2D heat map; it adheres to the ordinal layout of floors and rooms but introduces an abstract layout in order to facilitate a complete overview of all data points. Also like the heat map, the area map can be visualized on a rectangular cuboid achieving high affinity with the real building in terms of layout as well as a layout of data points that wraps in a
manner true to the real building. This is visualized as an AffinityArea visualization shown in Figure 4. Here the color encoding from the 2D area has been retained for consistency, but color can be used to encode other data as area encodes consumption.

### 3.2.3 From Bar Charts to AffinityBar

Opposed to the previously discussed area map, which uses area of data points to encode data values, the bar chart uses area as a supplementary visual encoding to its primary encoding – extend of bars.

It does this by fixing one dimension of the area of all bars changing only one dimension in order to facilitate easy comparison of two bars. The bar chart, however, adapts poorly to a multidimensional layout because its layout only expands in a single direction necessitating either multiple bar charts with a similar layout or a recurring bar chart to represent the [floor, room] layout of a large building.

The 3D bar chart in Figure 5 attempts to facilitate an ordinal 2D layout, similar to the heat map and the area map but uses height of bars for encoding data, which both sacrifices a complete overview of data points as well as potentially hiding outliers in the lower range of data. Although the 3D bar chart seems inferior, the heat map and the area map in terms of its ability to represent layout and encode data values, the principle of a volume based bar chart is very useful when combined with a high affinity spatial layout of a real building. By applying the principle of a volume based bar chart to the spatial layout of a real building encoded as a rectangular cuboid, by fixing two dimension of each apartment data point and extruding each apartment in a single direction dependent on its orientation relative to its position on the building. The result is the AffinityBar design in Figure 6. The volume of a data point is used to encode consumption and color of units is retained for consistency. The volume of the core structure can be used to encode resource consumption that is not attributable to an apartment, and thus serve as a common reference for the extent of the extrusion of individual apartments. Because the core structure differ in three dimensions, it can be difficult to compare it to individual units because they only expand in one dimension.

### 4. QUALITIES OF AFFINITYVIZ

The basic construction is a simplified isometric 3D model that utilizing a real building’s spatial layout as layout in order to create a direct relation to that building. In this section we discuss the key features of AffinityViz – both common and unique features for the designs (Figure 2, Figure 4, Figure 6).

#### 4.1 Simplified 3D Model

AffinityViz’s usage of volume in a visual representation is new in that it uses volume to achieve physical affinity with the building whose data is visualized. All three AffinityViz designs uses
the three dimensions of units of a multistory building as the key layout feature. In the AffinityBar (Figure 6) visualization, the volume created by the three dimensions of units of a multistory building are used as a relative measure for extent of the single units. This is done by using common consumption (e.g. elevator electricity usage) as reference for calculating the volume (extent) of a single unit. If no common consumption is available or it is not appropriate to use, then the volume of a cubic unit is set to the average consumption of all units.

AffinityViz is designed to achieve physical affinity by mimicking the spatial layout of a building, boiled down to its simplest rendition retaining a common unit (apartment, office, etc.) used for measurement. This enables AffinityViz to retain an important spatial relationship between units – namely that sequence of units is both sequential and it wraps from highest to lowest. Meaning that e.g. apartment 14 and 1 on a given floor are situated next to each following the same rules that situate e.g. apartment 3 and 4 next to each other.

For all three AffinityViz designs an issue arises with corner units, which are located on a single surface. It requires consideration from implementation to application which surface to place it, and thus potentially hinders immediate generalizability. Units that are only somewhat similar such as offices combined to create a single larger office, can be handled to some extent by aggregating multiple units into a single larger composite units. The orientation of the individual unit, i.e. the surface on which the unit is situated, reflect the orientation of the corresponding apartment or office. Furthermore, the orientation of a unit on the spatial layout helps to group units directly related to surfaces of a real building as well as distinguish between such groups because, a unit appear distinctively different due to the isometric perspective. The orientation is most pivotal in the AffinityBar (Figure 6) design as a unit bar expands and contracts along a single dimension only. In the AffinityBar design, both the units’ data and the common data is encoded with volume, but in different ways. Where the spatial layout expands into three dimensions, the volume of a single unit always fixes two dimensions, and data expanding in a single dimension. This can make units stand out, but it can also potentially hide units with low extrusion, necessitating rotation to detect such units.

4.2 Low Cognitive Load

AffinityViz exhibits its true strength in the low cognitive load it introduces to the user when compared to the abstract layout of generic types of visualizations. This is to a large degree owing to properties elaborated in the previous three subsections – the 3D layout, the simplified model, and the 1D volume growth. Together, these three properties establish a visualization representation that has a high affinity with the real building, thus using the real building as a direct frame of reference because the visualization shares the same basic structure as the real building.

The overload of three visual dimensions for both layout and data representation lessens the need for legends or labels describing the location of single apartments or offices as often needed in generic types of visualizations. This means that the user does not need keep an ongoing reference to an abstract coordinate system in order to place a unit in its spatial context. Furthermore, as described previously, the spatial layout of AffinityViz wraps around the building in the same way as the apartments or offices do around the real building. This means that adjacent units in the real building also are adjacent in AffinityViz’s layout. This makes AffinityViz suitable for users with different prerequisites on the analysis.

What AffinityViz does not support effectively, as discussed in 3.2, is a complete overview of units because units on two faces of the AffinityViz will be hidden from view. This means that some analyses, such as comparing units on opposite surfaces, will introduce a higher cognitive load because a user will need to remember non-visible units.

Although AffinityViz is already contextualized through its design as a simplified model of a real building, more contextual information can be added to create an even stronger relation to a real building. For instance a compass can indicate the building’s orientation relative to the corners of the world. Other contextual enhancements could be to show solar radiation to assist in analyzing differences in heat and electricity consumption between surfaces with differing solar radiation. Another enhancement could be to add simple landmarks or infrastructure elements such as adjacent roads or structures.

4.3 Visual Analysis

The 3D layout of AffinityViz provide a natural segmentation of units into groups that adhere to the real buildings structure thus the overview of the location of units in the building is a part of AffinityViz. This assists in analyzing patterns in consumption either on entire sides of a building, between different sides. Distilling apartments or offices into uniform units, is essential in AffinityViz to compare apartments or offices. The low-fidelity of AffinityViz features single units to convey their corresponding data because of the underlying
uniformity of units, as distortion of similar units expresses variations in the underlying powerfully.

For the AffinityBar (Figure 6) design, it is only a single dimension of a unit that is used as a measure for visually comparing units. This makes it easy to spot high outliers, or the lack thereof, than if units were transformed in two or three dimensions dependent based on the data. The differing orientation, and dimension of growth, of a single unit can cause visual indirection because units adjacent to each other can grow in different directions.

Currently AffinityViz is implemented with horizontal rotation and mouseover tooltips as the only interaction, but all three designs will benefit greatly from rich interaction to support users to create and rapidly test and rethink hypotheses. All three AffinityViz designs can be utilized to visualize flow of live data of replay of historic (user controlled or not) by animating the data points over time, though it would be most distinct in the AffinityArea (Figure 4) and AffinityBar designs. This is easily done in our current AffinityViz implementation because employed SVG elements are all animatable.

5. FROM DATA TO AFFINITYVIZ

AffinityViz was developed experimentally for a 12 story apartment building in Aarhus, Denmark.

5.1 Data Management

All data from the building is transmitted to and stored using the Karibu architecture (Christensen, et al., 2014), from where it is retrieved and massaged into a format appropriate for loading in a client web browser and rendering with SVG elements. Although the current implementation of AffinityViz only visualizes electricity data from the building, it is interchangeable with other data sources extracted from the Karibu architecture, such as consumption of district heating, hot/cold water, CO₂, etc. This will be subject to future work on AffinityViz implementations as it matures into a more complete tool for visual analysis.

5.2 Browser-based Visualization

AffinityViz is implemented using JavaScript to create SVG elements in a browser DOM, which in return renders the elements. As SVG elements are 2D, and therefore has no real concept of depth, there are obstacles in creating a 3D visualization, as angles and lengths of all shapes need to be calculated manually. Although this therefore might seem like a counterintuitive choice, opposed to e.g. WebGL or standalone 3D modelling software, rendering AffinityViz using SVGs enables us to take advantage of the rich set of interactivity supported by browsers. E.g. transitioning the extent of a single unit becomes trivial, as many SVG elements are animatable. Furthermore, the vast amount of existing JavaScript libraries that operate on DOM elements can be applied, and therefore this implementation of AffinityViz becomes open for further development.

5.3 AffinityViz in Use

We have collaborated with facility managers and conducted tests with the three versions of AffinityViz from section 3.2. This has lead us to identify central elements of the AffinityViz designs. Overview of data represented in AffinityViz is generally lower when compared to 2D counterparts as two sides always are hidden from sight. But AffinityBar handles this better than the AffinityHeat and AffinityArea designs because extreme high outliers are visible even if their corresponding surface is not front-facing, as they extent greatly from the building’s core. However, bars with little extent can potentially be hidden from view by neighboring bars with larger extent. We can alleviate this by implementing full horizontal rotation, because it enabled users to eventually view all bars. But this still is a drawback compared to the 2D visualizations discussed in section 3.2. Because the layout of the three AffinityViz designs provides spatial reference to the building, it supports queries based on spatial position of units. This is due to units are both grouped onto surfaces and has a significant spatial location, reflecting their real location.

6. PROSPECTS OF AFFINITYVIZ

AffinityViz has both advantages and shortcomings, i.e. the visualization technique that do not suit all potentially use scenarios equally well. Currently, AffinityViz relies on a simple recurring floor plan of a rectangular circumference of units multiplied by a number of floors. Although many apartments and office buildings have such a layout, AffinityViz’s generalizability is conditioned since buildings with a more complex layouts may not be suited for having consumption data visualized using AffinityViz techniques. Furthermore, because AffinityViz is 3D it is only suitable for visualizing
buildings where units are in the circumference of the building.

Because the current implantation of AffinityViz the 3D model is isometric, meaning that it is a construction of parallelograms and thus has no vanishing points, it can be argued to violate Tufte’s Lie Factor (Tufte, 1983) because similar sized units will be perceived as not similarly sized due the perceived perspective of the visualization. Although this can obfuscate precise comparison of far apart units it does not hinder holistic exploratory analysis.

In section 3.2, we discussed differences in layout properties the AffinityViz designs and the conventional visualization techniques. Together with the inferior overview of data in the AffinityViz designs, as documented in the evaluation, this illustrates that AffinityViz will not fully replace related conventional visualization techniques in all cases. Rather, it is a novel concept for visualizing consumption data from buildings while retaining a building’s spatial layout, thus lowering users’ cognitive load.

Next steps will be to mature the AffinityViz visualization technique with a more advances set of interactions, e.g., filtering of data and open access to data sets. Such features can make it a tool usable a wider range of professions. Also including users with non-technical backgrounds, who have a desire to analyze data, but not necessarily has prerequisites for using conventional visualization tools. This will be developed through continued professional consultation with facility managers and experts from other professions who are relevant to include.

7. CONCLUSION

This paper has introduced AffinityViz techniques for making generalizable and highly affine visualizations of consumption data from multistory buildings. Three AffinityViz designs were implemented and evaluated with expert users from the facility management domain. The evaluations showed that the AffinityBar technique is slightly better than the AffinityHeat and AffinityArea techniques with respect to minimizing the cognitive load when users have to deal with different visual analytics tasks that requires mapping of results to locations in buildings. The implementation of the AffinityViz data supply chain has been described for tall multistory buildings. However, the techniques can be tailored to work for most archetype building layouts of office buildings, schools, and factories. The techniques are under continual development with the goal of generalizing to cover more building types and supporting AffinityViz visualizations to integrate a wide range of real world data.

ACKNOWLEDGEMENTS

The work was supported by Danish DSF grant no. 11-115331. We wish to thank our colleagues in the EcoSense project for their contributions.

REFERENCES

Publication 6

The AffinityViz Method: Exploratory Visual Analytics for Energy Behavior in Buildings

I am main author on a 23-page paper submitted to the ACM Transactions on Knowledge Discovery from Data special issue on Interactive Data Exploration and Analytics.
The AffinityViz Method: Exploratory Visual Analytics for Energy Behavior in Buildings

MATTHIAS NIELSEN, Aarhus University
ROBERT S. BREWER, Aarhus University
KAJ GRØNBÆK, Aarhus University

Many domain experts today face the need to develop an understanding of big data sets collected via sensors or by humans in their domain. Domain experts are typically not familiar with advanced and more abstract data mining tools. In this paper, we introduce a method AffinityViz that empowers domain experts within energy management with the ability to visually analyze energy behavior based on consumption meters, sensors and user reported survey data. The AffinityViz method and tool for mixed visual analysis of resource consumption data and occupant survey data (e.g., questionnaire data) from large multi-story buildings using a highly affine visualization technique, that directly maps the visualization to real world buildings. AffinityViz is an overview+detail visual analysis tool supporting the rapid generation and evaluation of hypotheses concerning patterns and anomalies in resource consumption data mixed with survey data. Furthermore, by applying a simplified and consistent, yet highly affine, visual layout that spatially contextualizes data, AffinityViz is highly appropriate for use by experts from diverse professions working individually or interdisciplinarily. AffinityViz has been deployed with five domain experts within energy management actively using it, and it has been formally evaluated with 10 more experts. The contributions of the paper are: 1) the design goals for interdisciplinary mixed visual analysis of cross-referenced resource consumption and occupant survey data, 2) design rationale and architecture for the AffinityViz tool, 3) a detailed use case demonstrating AffinityViz’s applicability for resource consumption analysis, 4) results of an evaluation with 10 energy management experts. Finally, we discuss future development and applications of AffinityViz.

1. INTRODUCTION

Information visualization and visual analytics possess the potential for creating new data insights in many different application domains (Card, Mackinlay, & Shneiderman, 1999) through the notion of interactive exploration of the data. Visual analytics provides domain experts with little or no programming and data base skill to interactively explore big data sets to find patterns and types of outlier cases not being revealed yet. In order to support domain experts with efficient and
In this paper, we investigate the domain of energy and resource consumption in buildings, and it emerges that in order for energy managers and users to understand or shift consumption we need to include characteristics of the occupants of buildings in analysis (Chetty, Tran, & Grinter, 2008; Irwin, Banerjee, Hurst, & Rollins, 2014), which in turn requires new specialized visual analytics methods for this domain.

Residential and commercial buildings consume a large portion of the world’s energy (as much as 40% in the United States). Reducing building resource usage (water, electricity, heating) can reduce greenhouse gas emissions and the cost of building operations. Increasingly, buildings have advanced sensors for measuring resource consumption data with high granularity. One use for such data is data-driven interventions aimed at motivating occupants to reduce their resource consumption.

However, in order for energy analysts, building managers, and others wishing to understand or change individuals’ resource consumption, we need to include characteristics of the occupants of buildings in the analysis (Chetty et al., 2008; Irwin et al., 2014). Many interdisciplinary studies address this challenge and use ethnographical, psychological, or other sociological methods (Blunck et al., 2013; Darby, 2006; De Young, 1993; Kollmuss & Agyeman, 2002; Mills & Rosenfeld, 1996; Pelletier, Tuson, Green-Demers, Noels, & Beaton, 1998) to understand everyday practices and how those practices affect energy consumption. However, analyzing cross-referred resource consumption data and occupant survey data is a complex issue, especially for experts with diverse backgrounds who have diverging technical proficiencies.

This is where novel methods like the AffinityViz tool proposed in this paper (See Figure 1) are needed. AffinityViz combines resource consumption data from sensors and data from inquiries such as interviews or surveys, in order to achieve a rich understanding of the state of consumption and the potential to reduce consumption.
AffinityViz is aimed to support cross-referred analysis of quantitative sensor data and user survey data as proposed in the Computational Environmental Ethnography (CEE) methodology (Blunck et al., 2013), and by Knigge and Cope (Knigge & Cope, 2006) in their mixed visual analysis of ethnographic data and GIS data. AffinityViz builds upon the visualization technique discussed and evaluated in (Nielsen & Grønbæk, 2015), which creates a highly affine spatial relation between its visual layout and a real building by using a simplified model of the building. The visualization technique applied in AffinityViz achieves a high affinity between a real-world building and the visualization representation by using a simplified rendition of the building’s layout in terms of number of floors, number of apartments on each floor, and the positions of apartments on floors. The AffinityViz is well-suited for many buildings that have a cuboid layout where apartments or offices are positioned in the circumference of the building, such as many tall-rise buildings, office buildings, and apartments buildings.

The remainder of this paper is structured as follows: First, we examine and discuss related academic and commercial work in this area. Second, we introduce and motivate the cross-referred analysis challenges that we address. Third, we describe the design and software architecture for the AffinityViz tool. Fourth, we demonstrate how experts use AffinityViz to analyze data from a large apartment building. Fifth, we discuss an evaluation of AffinityViz with 10 experts in building resource usage. Finally, we end with a discussion of future work and conclusions.

2. RELATED WORK

In this section, we motivate our work based on a short review of recent work on the visualization of energy consumption data and more specifically, the special case of applications for large buildings. Furthermore, we outline how our work goes beyond existing examples of combined analysis of occupant and consumption data, as well as literature regarding how to conduct interdisciplinary data analysis.

Visualization of energy consumption data is an area that has seen considerable attention in recent years (Froehlich, Findlater, & Landay, 2010; Pierce, Odom, & Blevis, 2008; Pierce & Paulos, 2012), covering a multitude of approaches for physically or virtually visualizing energy consumption. Specifically, our work fits into Pierce et al.’s “tools for analysis and insight” strategy (Pierce et al., 2008). Therefore, while others explore ways of, e.g. creating ambient awareness of energy consumption (Rodgers & Bartram, 2011) or using gamification to impact on energy consumption (Reeves, Cummings, Scarborough, Flora, & Anderson, 2012), our work relates closer to examples visual analysis of energy consumption like (Erickson et al., 2013) and (Goodwin et al., 2013), although the latter paper’s main contribution is a design process, based on creativity, for the design of visualization for analysts.

Furthermore, our work is inspired from Pierce et al.’s proposed research opportunity to engage with energy infrastructure, service, and policy beyond the individual consumer (Pierce & Paulos, 2012). A research opportunity that echoes Froehlich et al.’s (Froehlich et al., 2010), which identifies the residential context as the one most often targeted in eco-feedback technology. Specifically, our work utilizes the spatial layout of buildings as a reference in visualizations for analysis of data from these buildings. This includes work by Kim et. al. (Kim, Shin, Choe, Seibert, & Walz, 2012) who use real-world spatial references for visualizing consumption data, including overlays onto large buildings depicting consumption e.g. on a particular floor, as well as Yang et. al. (Yang & Ergan, 2014) who conclude, based on a comparative evaluation, that tools for visual analysis of data from buildings benefit
from using corresponding building's layout as reference in the visualization. Their results also show that while experts in general perform better with both 2D and 3D layouts, experts familiar with the building in question perform better with 3D layouts. Other examples using the layout of large buildings as layout in visual representation of consumption data include (Bellala, Marwah, Arlitt, Lyon, & Bash, 2012; Kusy, Rana, Valencia, Kurdak, & Wall, 2014; Martani, Lee, Robinson, Britter, & Ratti, 2012; Orestis, Dimitrios, Dimitrios, & Ioannis, 2013).

Another promising direction in energy consumption research is to include data describing the occupants and their activities, which results in the consumption, in order to understand the consumption itself (Chetty et al., 2008; Irwin et al., 2014). This is addressed explicitly in the Computational Environmental Ethnography (CEE) methodology (Blunck et al., 2013), which relates to our work because it argues for iterated collection and analysis of both consumption data and survey data in order to understand the values, motivations, and behaviors that result in energy consumption. While CEE only mentions visualization of data as a single stage in its methodology, Knigge and Cope (Knigge & Cope, 2006) argue for a tight integration of visualization for recursive, mixed visual analysis of qualitative ethnographic data and quantitative GIS data. Beckel et al. have used smart meter data to infer household characteristics, such as employment status, using occupant inquiry data as the ground truth to evaluate the accuracy of their algorithms.(Beckel, Sadamori, Staake, & Santini, 2014; Kleiminger, Beckel, Staake, & Santini, 2013)

A final case is made in the CEE methodology is the importance of interdisciplinary analysis when analyzing consumption data and survey data (Blunck et al., 2013). However, tools for interdisciplinary visual analysis of data are comparatively rare. In geoscience, Jacobs et. al. (Jacobs, Kilb, & Kent, 2008) argue for the applicability 3D interactive visualizations, as opposed to conventional 2D charts and plots, of geographical features and complementary data because they afford greater comprehensibility for the interdisciplinary analysis that is necessary for disparate datasets.

To sum-up, we have identified a need – not covered by existing work – to develop interactive visual analytics for energy consumption stemming from large buildings, which utilizes the building's spatial layout for layout in the visual representation. Furthermore, we see the need to address inclusion of data describing the occupants whose activities cause consumption, as well as methods for mixed and interdisciplinary visual analysis of consumption and survey data.

3. CHALLENGES OF MIXED VISUAL ANALYSIS

In this section, we describe the background and motivation for interweaved analysis of consumption and survey data and representative examples of interweaved analyses.

3.1 Consumption Data is Insufficient to Understand Potential Saving

Multistory apartment buildings represent a special case in energy and resource consumption analysis because they commonly are a composite of many single households that are either all similar in size and layout, or can be grouped into a few categories. Therefore, such buildings can be considered living laboratories for researchers and energy analysts interested in introducing and evaluating interventions aimed at lowering consumption targeted at multiple occupants with similar physical living conditions. However, because residential resource consumption is a product of occupants’ activities, occupants should not be viewed as
micro-resource managers consciously weighing up cost and benefits of an activity (Strengers, 2011). It is therefore crucial for successful attempts to lower consumption through social or technological interventions to understand the underlying values, behaviors, and motivations of occupants that result in consumption.

Data used in mixed visual analysis of consumption data and occupant survey can be organized into two major categories – quantitatively measured consumption data and survey data collected from occupants collected through e.g., questionnaires or interviews. Although not all analyses will rely on both types of data, they are both important because, roughly speaking, consumption data can help to answer questions of factual nature (“what” questions) and survey data can help answer questions of more in-depth investigative nature (“why” questions).

3.2 Design Goals for Shared Understanding

Since 2012, a group of experts, consisting of computer scientists, engineers, anthropologists, a sociologist, and designers, which we have collaborated with when developing the AffinityViz tool, have conducted work and research on resource consumption and survey data collected from an apartment building, when analyzing causes of consumption on par with (Blunck et al., 2013; Chetty et al., 2008; Irwin et al., 2014). The ambition of this group is ultimately to lower the environmental impact of the apartment building by adopting a data-driven, interdisciplinary approach to introducing technological or social interventions. The group collects and analyzes sensor and survey data from a multistory apartment building with 159 apartments, each with individual high-frequency meters for electricity, district heating, and hot and cold water. We collaborated with these experts to define and address the needs that they have for analyzing data from apartment buildings.

1. **Spontaneous Exploratory Analysis:** While exploratory analysis might seem rudimentary, we found that the group of experts lacked a tool that supported the rapid iterative process of hypothesis generation, evaluation, and refining that arise when enthusiastic people collaborate. Commonly, one expert would use a tool familiar to him or her – e.g., spreadsheet software, Tableau (Hanrahan, Stolte, & Mackinlay, 2007), or Matplotlib (Hunter, 2007) – but equally commonly this would sideline other experts from the group resulting in skewed analyses. Therefore, a tool should have an easily amendable conceptual that mitigates the concentration cost of frequent focus shifts.

2. **Mixed, yet Consistent, Representation:** As argued previously, solely relying on consumption data is insufficient for understanding activities underlying consumption and therefore data describing. Besides large quantities of sensor data, the group of experts conducts questionnaires and interviews describing occupants’ self-reported values and consumption appraisals and habits. This means that a tool should facilitate analysis of quantitative sensor data mixed with occupant survey data. Furthermore, because data collection is ongoing, the tool should provide a view that is adaptable to new data sources (i.e. new types of sensor data, or an updated questionnaire), yet has a consistent representation as data sources are updated or substituted.

3. **Dissemination of Findings to Stakeholders:** Besides analyzing data for themselves, we ascertained that the group of experts have obligations to disseminate their findings to multiple external stakeholders. These stakeholders include the company that sponsors the building and uses it a laboratory for testing new resource consumption sensing systems, building managers operating...
and servicing the building, as well as transient collaborators. This means that a tool should be able to communicate findings, or substantiate hypotheses, in a way that is apprehensible to people familiar with the building but who are external to group of experts.

To overcome this task, we sought a boundary object (Susan Leigh Star, 2010; Susan L Star & Griesemer, 1989) that could be used by experts from each profession and support them in performing mixed visual analysis. We did not seek to substitute or converge existing tools used by the experts individually, but instead develop a tool that facilitated collaboration across disciplines in the group and responsibilities of the group. For this purpose we utilized the highly affine visualization technique for visualization of building data presented in (Nielsen & Grønbæk, 2015), which simplifies a building’s layout to simplest form by removing details unnecessary for the data to be represented. We have extended this technique with support for integrating survey data, and we have combined the technique with selection, filtering, and temporal scrolling and zooming functionality.

4. DESIGN AND ARCHITECTURE OF AFFINITYVIZ TOOLS

In this section, we describe the software architecture of our tool for interwove analysis of occupants’ resource consumption data and data describing occupants’ characteristics. Furthermore, we describe the technical architecture underlying our tool.

4.1 Visualization Design

Our tool is designed to provide an overview+detail (Shneiderman, 1996) interwoven view of consumption data and survey data. The central component is a building model visualizing consumption data and survey data questions (Figure 2.

![Figure 2: A selection of previous design iterations. Top-left (A): Manual case-by-case scatterplot. Top-right (B): Interactive and highly manipulable scatterplot. Bottom-left (C): Area map using floor (vertical) and room (horizontal) numbers for layout with color encoding orientation and size consumption. Bottom-right (D): Heat-map using same layout. Color encodes consumption.](Image)
The building model is surrounded by selection functionality for types of consumption (Figure 2 (1)) and survey data (Figure 2 (3)) as well as a temporal overview of the selected consumption data supporting dynamic filtering of consumption data (Figure 2 (4)). The interface is implemented with thorough cross-filtering, meaning that a selection or filtering in one part of the view triggers corresponding updates in relevant others parts of the view.

4.1.1 Previous Design Iterations.

Before arriving at our final design (Figure 1) we went through numerous iterations, which were pivotal in the process of refining and spelling out the requirements for a tool for mixed visual analysis of sensor data and occupant survey data. Initially we created case-by-case scatterplots with simple glyphs of varying sizes (Figure 2 (A)), which were good at communicating a specific finding. However, they were time consuming to great because they commonly took a couple of attempts to get right. Therefore, we thought that a generic interactive scatterplot (Figure 2 (B)) would suit the group of experts well. This design allowed users to select, and weigh consumption data sources and survey questions and encode in any combination of color and horizontal and vertical position. However, as the data encoded for the horizontal and spatial layout became composite, the conceptual model became too complex for most of the group members. The result was that the group of experts rejected this design.

Because it was the generic and abstract layout of the interactive scatterplot that, we decided to experiment with visualization techniques that were capable of retaining the common denominator in all data collected by the group of experts – the spatial layout of the apartment building. We quickly arrived at area and heat maps with floors plotted as rows and apartments plotted as columns (Figure 2 (C) and (D)). However, although these visualizations provided an overview, members of group found that the visualizations lacked contextual information from the apartment building. Therefore, we decided to apply the highly affine visualization introduced in (Nielsen & Grønbæk, 2015), which introduces the principle of simplifying developed interactive elements that supported the needs of the group of experts, and deployed the tool with the group, which we will be the subject of the remainder of this section and the following section.

4.1.2 Building View.

The building model (Figure 1 (2)) in our tool is based on the highly affine visualization technique (Nielsen & Grønbæk, 2015), which is designed around reducing the structure of a block building to simple components – units – that still can be individually metered for consumption data. In an apartment building, this could be individual apartments and in an office building, it could be individual offices. The visualization technique is applicable to the apartment building used in this project because all apartments are located in the circumference of the building. By doing this, the visual representation retains a simple, yet explicit and easily recognizable, reference to the apartment building. The building view provides a partial spatial overview relating consumption data and survey data to a specific location in a real building. Furthermore, it provides a detail view of the current selection and filtering of data by means of a tooltip on mouse over on a column. We have opted for the column design of the highly affine visualization technique because even though the visual representation is laid out in three dimensions each individual bar only grows in a single dimension making extreme outliers prominent. The extent
of individual columns’ extrusion is used to encode consumption data, i.e. electricity, water, and district heating. The actual extrusion a single column is calculated as a function of the average consumption of all apartments. The average consumption of all apartments always corresponds to a cubic column. Therefore, an individual apartment’s extrusion extend corresponds to how much higher or lower the apartment’s consumption is relative to the average consumption – e.g. if an apartment consumes half of the average, this apartment’s extrusion would correspond to a halved cubic column. The column design is similar to the parallel bar charts in (Chittaro, Combi, & Trapasso, 2003) through its use of 3D columns extruding in a single dimension, but also to the topographic reliefs in (Bertin, 2010), because a direct spatial relation to the real building is retained. In the case of an apartment located in a corner of the building, they are positioned on the surface of the building, with which the apartment shares its largest surface.

Using a three dimensional visual representation of data introduces occlusion of elements, as discussed by Ware (Ware, 2008). Occlusion provides a natural order of units in the building in the sense that units are grouped on the surfaces of the building that they belong to and units closer to the perceiver will be positioned fully or partially in front of units further away. Occlusion, however, also means that (roughly) half of the apartments are occluded from at any given state of the visualization and that neighboring units can hide low outliers. Ware suggests selectable transparency to reveal occluded objects in three-dimensional visual representations. However, we found through testing that full horizontal rotation of the building model around its center was best suited for allowing users to bring units into view. This is done to retain a high spatial affinity to the physical building, which is important because it introduces a recognizable, and therefore easily attainable, conceptual model of the visual layout in the tool.

4.1.3 Consumption Data View.

The Consumption pane, Figure 1 (1), supports selection of any of four different types of consumption data – electricity, hot water, cold water, and district heating – which we have collected from the dormitory. This selection is implemented as select/deselect buttons (similar to radio buttons). To the right of the consumption pane is a cube, which shows the average consumption of all apartments for the currently selected resource and thereby functions as a legend for the selected consumption. Consumption data can be unselected to just show tiles on the building (if a user is only interested in survey data), and apartments that have no consumption can be excluded. Furthermore, the consumption data view supports filtering away any apartments that does not have any registered consumption of the currently selected type. Interaction with the consumption data view triggers transitioned update of the building view and the temporal selection view.

4.1.4 Survey Data View.
Similar to the consumption data view, the survey data view (Figure 1 (3)) is constructed as a series of select/deselect buttons. However, because there are a total 39 different answer metrics in the current survey data, we have constructed a nested selection mechanism for the survey data options. Therefore, answer metrics have been grouped, and each group will unfold to reveal sub-answers when clicked. Selecting an answer metric triggers an update of the building view, which colors relevant apartments according to a legend to the left of the survey data view. As with the consumption pane, apartments for which there is no survey data can be excluded from the visualization.

4.1.5 Temporal Selection View.

The last part of the view is a timeline of the currently selected consumption data (Figure 2 (4)) beneath the building view. This is a bar chart were the height of each bar depicts the total daily consumption for all apartments for the selected consumption type. The bar chart itself performs two roles: 1) it shows the overall temporal development of consumption in the apartment building, and 2) it depicts days for which data is missing by negative space / missing bars. The bar chart supports sequential temporal selection by dynamic brushing (Becker & Cleveland, 1987) on a range of bars. The brush also facilitates single-dimensional temporal panning and zooming by moving the brush and by extending or shortening the brush respectively. The temporal selection view also supports automated playback of by either incrementing the position of the brush or by extending the range of the brush by one day at a time. All interactions, manual or automatic, trigger an update of the building view, including the cubic column consumption legend.

4.2 Architecture

The architecture for the AffinityViz tool, presented in Figure 3, consists of multiple coordinated MVC instantiations that implement and handle interactions and updates.
of the visualization. For the sake of clarity, session-state persistence implemented as supplementary view-models for all controllers (except the rotation controller) has been left out of Figure 3. Read from the bottom, when the visualization is first initialized it loads and parses survey data and consumption data. This data is then passed onto corresponding controllers, which populate interactive elements in the visualization, and registers callbacks before finally drawing the visualizations.

### 4.2.1 Controller Layer.

Controllers 1 through 4 are outlined in Figure 3, and their corresponding views are shown in Figure 2. Controllers 1 allows users to select which type of consumption data, if any, should be visualized in the building visualization and the bar chart. Furthermore, it supports selecting whether apartments with no consumption should be rendered in the building visualization. Controller 2 handles rotation of the building visualization, which is implemented as full horizontal scrolling using either mouse or touch input. Controller 3 allows users to select which data, if any, from the survey should be included in the visualization. Furthermore, it allows users to select whether all apartments should be included in the building visualization, or only apartments whose occupants have responded to surveys. Controller 4 implements temporal zooming and panning in resource consumption data by means of brushing on the bar chart visualization. Controller 4 also supports automatic temporal playback of consumption data by either shifting an existing span of dates or by adding consecutive dates to an existing span dates.

Whenever a controller registers an update, e.g. a user selects a new type of consumption, it notifies a function common for all controllers, which then queries all controllers, calculates an updated layout of the building visualization and the bar chart visualization, before it selects and updates appropriate elements.

### 4.2.2 Current implementation.

Our implementation of AffinityViz is web-browser based using JavaScript to create and enter SVG-elements into the DOM of the web-browser, which renders the SVG-elements as graphics. This means that it uses technologies that are standardly available in modern browsers, making it very easy to deploy with users, which is important because AffinityViz is seeing active use by experts analyzing consumption and survey data in a current deployment. The bar chart in the bottom is a fairly standard implementation using D3 (Bostock, Ogievetsky, & Heer, 2011), while the building model is custom made using JavaScript and math (trigonometry and isometric projection). In our implementation, we rely on a Karibu system (Christensen, Blunck, Bouvin, Brewer, & Wüstenberg, 2014) for collecting raw sensor data and an accompanying middle layer for extracting consumption data. As can be seen in Figure 3, these are easily interchanged with other systems or layers if needed.

5. **AFFINITYVIZ APPLIED TO AN APARTMENT BUILDING**
We have developed AffinityViz iteratively through a deployment with an ongoing research project started in 2012 revolving around a newly built apartment building for students in Aarhus, Denmark. Most often, we have collaborated with an interdisciplinary team, consisting of computer scientists, anthropologists, engineers, designers, and a sociologist, conducting CEE (Blunck et al., 2013). The apartment building consists of 159 apartments and was constructed as a living laboratory for a sponsoring company to test new infrastructure technologies and for researchers to study how to lower the resource consumption of occupants. This case will feature as example throughout this section and the analysis scenarios elaborated in this section have taken place within the CEE team. In order to facilitate these endeavors, the apartment building includes a multitude of sensors measuring consumption (water, district heating, and electricity), as well as indoor climate metrics such as temperature, humidity, CO₂ concentration. This data is collected for each apartment individually, as well as overall consumption from all common areas, elevators, etc. Data from each sensor is commonly collected at one-minute intervals, and even more frequently for some sensors.

Survey data describing the occupants has been gathered by means of questionnaires and interviews. As the apartment building houses students, there is a high occupant turnover and survey data quickly becomes outdated, and the survey data used in the examples in this paper stems from a questionnaire conducted in the second half of 2014, to which approximately half of the occupants responded.

For AffinityViz analysis, we have had preliminary access to electricity consumption data for around eight consecutive months in 2012 and 2013, and since high granularity electricity, hot and cold water, and district heating data from August/September 2014 onwards. Furthermore, inquiry data has been collected by conducting two rounds of questionnaires, both with a response rate of approximately 50%, and with a round of semi-structured interviews with approximately 20 participants.
5.1 Lacking Awareness of District Heating Consumption

An early and iterated hypothesis in the apartment building analysis involved how consumption of district heating in individual apartments would vary depending on their location in the apartment building. District heating is the most common source of heating in Denmark supplying either all or some heating to 63% of all households. The district heating is produced in large plants and distributed through extensive networks of pipes. This is also the case for the apartment building in this case where the district heating in all apartments is controlled individually by the occupants and consumption is metered for each apartment. Before the CEE team got access to

Figure 5. Consumption of District Heating vs. Disagreeing that Apartment is Not Too Warm (Dark Blue)

Figure 6. Unusually low consumption of district heating on surface AC
quantitative consumption data, they initially hypothesized that the higher the floor of an apartment, the higher the consumption of district heating because apartments on higher floors would be more exposed to wind.

This was investigated by anthropologists in the CEE team by conducting interviews with a group of occupants from the apartment building, but the majority of the interviewees consistently said that they did not turn on their radiators and therefore did not consume any district heating. When the anthropologists and computer scientists correlated this with collected consumption data, it showed that the interviewees did in fact use district heating. This lead the team to hypothesize that the users would set-and-forget their thermostats. This would imply that if a user does not actively and continuously control and adjust the thermostat he or she would perceive it as if he did not consume any district heating – a latent consumption.

To evaluate the above hypothesis, the team included questions regarding users’ experience of living in their apartment with regards to temperature in a subsequent questionnaire. In order to analyze this, along with other data from the same questionnaire, we designed our tool to support the CEE team in the exploratory analysis of quantitative consumption data and qualitative questionnaire data. Some of the respondents with continuously high consumption of district heating answered that they agreed their apartment was “not too cold” (white or light blue apartments in Figure 4) but they disagreed that their apartment was “not too warm” (dark blue apartments in Figure 5).

This confirms the hypothesis that district heating can be a latent consumption is indeed the case for some occupants and that it is likely to be worthwhile area to investigate further to find a potential for saving energy. Furthermore, these results are not statistically significant as they only concern 4 to 6 out of 71 respondents, out
of approximately twice as many total occupants. Thus, there appeared no significant correlation by linear regression analysis on the same data.

Another interesting insight the team gained from our tool visualization in relation to district heating consumption is that apartments on surface AC consumes remarkably little district heating compared to other surfaces, as shown in Figure 6. Out of 43 apartments on a face of the building, that faces a neighboring building and has little solar radiation, only a single apartment consumes marginally more district heating than the average apartment. Currently the CEE team hypothesizes that this is due to that particular surface of the apartment building being sheltered by a neighboring building. The team intends to further investigate this by obtaining access to temperature data and including that in our tool, as well target this in future surveys.

5.2 Exploratory Investigation of Electricity Consumption in a Period of Low Occupancy

Using consumption data and questionnaire data, the CEE team has used AffinityViz for an exploratory investigation of habits that result in passive resource consumption. Passive resource consumption means that it occurs when an occupant is not at home. Such consumption can be both intentional and unintentional. As occupancy data for apartments is not available, the team decided to look for shorter periods of time where there are low levels of cold water consumption, as (leaky water installations apart) it is a reasonable indicator of whether an occupant has been home. Though this is coarse-grained filter for deciding this, they deemed it appropriate as long as only entire days or periods of days was investigated, because it is unlikely that a person being at home would use exactly zero water for an entire day. This investigation revolved around the three-day period 24-26 of December, a period where less than 20% of the apartments have registered any cold water consumption.

5.2.1 Out-of-Apartment Electricity Consumption.

The CEE team originally hypothesized that passive unintentional electricity consumption, e.g. from devices on standby, would amount to a noticeable differences.
in electricity consumption. It was furthermore hypothesized that this would be dependent on the types of devices and occupants' knowledge about, and opinion towards, electricity consumption. In the aforementioned questionnaire, the team therefore included questions about respondents weekly usage of a range of electric devices, as well as questions concerning their energy (un)awareness.

To evaluate this hypothesis, our tool was used to highlight apartments that consumed electricity without consuming any water in the aforementioned period. These apartments are highlighted with red in Figure 8. Although there are some variations between assumed non-occupied apartments, most of them have similar electricity consumption with relatively few apartments as outliers. As the apartment building is built recently, the team believe that a comparatively large part of this electricity consumption can be attributed to domestic appliances that would of the same brand and model in most, if not all, apartments. This leads to a new hypothesis, namely that the majority of the occupants in the apartment building are comparatively aware of their electricity consumption. Therefore, the team does not think that a wide-ranging energy awareness campaign in the apartment building will lead to significant reductions in consumption, but instead need to look at particularities in electricity consumption.

Out of the apartments that are assumed unoccupied in the period and responded to the questionnaire, one particular outlier is particularly conspicuous (highlighted in Figure ?). Even though this apartment is assumed unoccupied, it consumes more than nine kWh electricity over the three-day period. Not only is this level for this apartment below any other three-day period in the dataset, except for week 42 (commonly a holiday week), the fluctuation in total daily electricity consumption for this specific apartment is less than 0.1 kWh. Furthermore, this occupant's daily electricity consumption is never below three kWh, which is about 50 % higher than the average apartment consumption, and it fluctuates comparatively little. In the questionnaire, this respondent indicated that a desktop computer is on more than 21 hours per week – likely a server-like computer that is always on. Furthermore, this occupant responded that he predominantly disagrees that he does not know much energy devices in general use. The CEE team considered this interesting because, rather than regarding this occupant as a heavy consumer of electricity, and therefore a clear-cut candidate for initiatives that aim at lowering electricity consumption, he is likely an occupant whose intentional activities cause heavy electricity consumption.

6. EVALUATION WITH ENERGY MANAGEMENT PROFESSIONALS

Our tool has been designed iteratively with involvement of an interdisciplinary group of experts, deploying, redesigning, and evaluating before we arrived at our current version. In addition to the ongoing testing during the iterated development process, we have conducted an evaluation of the current version of the tool with 10 experts, who have professional expertise in the analysis of resource consumption data from buildings, but had no prior knowledge of the AffinityViz tool.

6.1 Evaluation Setup and Objectives
The evaluation was conducted as a formative evaluation consisting of the following elements: 1) a brief introduction to the AffinityViz tool, 2) a structured task-based exercise, 3) time to fill in a questionnaire, 4) an open-ended exploratory analysis, and 5) a short semi-structured interview. The evaluations were conducted with participants individually and lasted between 45 and 60 minutes. Data was collected by screen capture of the participants’ interactions, audio recording, and notes written by the evaluation facilitator. All participants were energy management.

![Figure 10](image)

Figure 10. Averaged responses to questionnaire. Blue are answers regarding the simplified building model. Orange are answers regarding the daily-summated view of consumption data. Gray are answers regarding the consumption and survey selection panes.

The evaluation was conducted as a formative evaluation consisting of the following elements: 1) a brief introduction to the AffinityViz tool, 2) a structured task-based exercise, 3) time to fill in a questionnaire, 4) an open-ended exploratory analysis, and 5) a short semi-structured interview. The evaluations were conducted with participants individually and lasted between 45 and 60 minutes. Data was collected by screen capture of the participants’ interactions, audio recording, and notes written by the evaluation facilitator. All participants were energy management.

![Figure 9](image)

Figure 9. High dispersion in consumption of district heating. More high outliers in the top part of the building.
professionals who all had professional expertise in analysis of resource consumption in buildings.

With our evaluation setup, we evaluate how well our tool supported two of the three design goals from Section 3.2 – Spontaneous Exploratory Analysis and Mixed, yet Consistent, Representation. We did not evaluate whether the tool supported Dissemination of Findings to Stakeholders, because it was not feasible to include stakeholders in the evaluations. Both parts of the evaluation was intended to evaluate whether the conceptual model of our tool was easily attainable and supported experts in quickly appropriating the tool. This we measured by asking the participants how well the tool supported them in reaching conclusions in the task-based part, and in the open-ended analysis part we gauged whether the tool supported them independently reaching findings similar to those described in Section 5.

6.2 Evaluation Results

First, we report on the results from the task-based exercise of the evaluation, second we report on the open-ended analysis part.

6.2.1 Task-based Results.

All participants were asked to complete four tasks each and afterwards complete a 10 question questionnaire gauging whether our tool supported them in reaching conclusions to the task. The results of the questionnaire is shown in Figure 10, with each bar depicting the average response to each question. These results leaves us with three main takeaways: 1) the participants consistently rated the simplified building very high, in terms of overview (Q1), interweaved view (Q2), extruded columns (Q3), and coloring of columns (Q4). 2) the bar chart depicting daily-summatated consumption data providing a good overview (Q5), but it lacked details, e.g. it was hard to select individual days (Q6) and its interaction could be improved (Q7). 3) the perceived usefulness of the consumption data and survey data selection

Figure 11. Occupants disagreeing that there is sufficient natural light in their room.
panes were very different, which we believe can be attributed to the fact that the four different kinds consumption data were easy grasp and to select amongst (Q8), in contrast to a total of 39 nested questions in a hierarchy showing the survey data to select from (Q9). Furthermore, the ability to filter columns with no data for a set of selections was rated high (Q10).

6.2.2 Open-ended Analysis Results.

The second part of each evaluation consisted of two open ended scenarios where participants were asked to assume a generally described role and was concluded with a semi-structured interview. The participants were asked to assume the role of an energy management analyst respectively analyzing occupants’ satisfaction with living in the apartment building and consumption of district heating. The scenarios were open-ended and were provided to the participants only to establish a rough setting for them to explore. There were no correct or incorrect answers for this part, which resulted in several interesting findings by the participants. During the open-ended analysis part, which lasted between 15 and 20 minutes, the 10 participants identified and reported a combined total of 29 findings, including findings similar to the findings reported in Figure 4, Figure 5, and Figure 6. The number of findings per participant ranged from one to six with an average of three. We will report on two example findings identified by participants, selected because they were novel (i.e. had not been identified by experts familiar with the tool) and were refined to a degree similar to the ones reported in Section 5.

The first finding is concerned with natural light satisfaction, and is illustrated in Figure 11. It was identified by two participants independently, and show results for occupants’ satisfaction with the amount of natural light in their apartment. In Figure 11, the color scale goes from white (strongly agree) to dark blue (strongly disagree). The two participants concluded that there seemed to be a problem with the amount of natural light for apartments on surfaces AC and BC on the lower half of the building, because many occupants do not strongly agree that there is sufficient natural light there. Furthermore, they both found that this was not an issue for occupants with apartments on surfaces AD and BD (not shown), where all but eight responding occupants strongly agreed that there was sufficient natural light in their apartment. The real apartment building has neighboring buildings on exactly surfaces AC and BC, but is taller than those buildings, so this finding corresponds well with the physical context of the real apartment building.

The second finding concerns the dispersion of district heating consumption, and is shown in Figure 9. The participants identifying the finding were investigating temporal development of district heating consumption. The three participants were quite surprised to find that there were high dispersion between apartments in the amount of district heating they consumed, with many apartments having little or no district heating consumption and several apartments having very high consumption. The participants found the apartment marked with a thick border in Figure 9 particularly interesting. This apartment had for several weeks had a consumption of district heating that was around 10 times the average consumption of all apartments and around double that of the apartment with the second highest consumption. One participant’s presumption was that something was amiss with the radiator, the thermostat, or the sensor measuring the consumption, because he considered it unlikely that such an outlier would exist. Furthermore, all three participants found that the very high outliers seemed to be located nearer to the top of the building.
The first part of the evaluation shows that the participants thought that the design of the AffinityViz tool supported them in performing mixed visual analysis of cross-referenced sensor and survey data. The second part shows that the conceptual model of the tool were easily attainable for the participants, because they were able to identify refined findings similar to the ones reported in Section 5, even though they did not have prior knowledge of, or experience with the tool.

7. FUTURE WORK

The AffinityViz method and tool for interweaved analysis of consumption data and survey data is currently seeing active use in the case of the apartment building described in Section 5. As it is an ongoing project, we are working together with the domain experts in the project to further refine our tool. Specifically, we are investigating how AffinityViz can be used to communicate consumption data to the occupants of the apartment building. This is a non-trivial issue, because the consumption data relating to individuals is private data and therefore this raises privacy concerns. We consider simple aggregation techniques (e.g. averaging all apartment’s consumption except for a current user’s) too simple because a lot of the analytical power of the tool is easily lost that way. Currently we are investigating applying the SITA principle (Andersen, Kjargaard, & Gronbaek, 2013), which, although it is primarily designed for location privacy, provides a vocabulary for occupants to define which data they are willing to share with other occupants.

Furthermore, we have discussed with housing associations who wish to use our AffinityViz tool to analyze resource usage and indoor climate parameters in apartment blocks that they manage. The request from the housing associations have put focus on the scalability and flexibility of AffinityViz tools, while maintaining the ease of use for the energy managers. We are thus working on two extensions to AffinityViz tools: On the one hand we need to be able to handle many individual buildings placed geographically spread over a larger urban area, thus the simplified building models need to be related to a map overview. On the other hand we need to be able to make affine visual analytics for a much richer set of building types. We are thus investigating ways to model other archetypical building structures in a similar affine manner to maintain the power of AffinityViz in other building layouts (see Figure 12).

Figure 12. Alternative layouts using the AffinityViz Technique
8. CONCLUSION

We have developed an interactive visual analytics method AffinityViz as well as a web-based tool to support the method. This tool is specific example of how interactive visual analytics can be developed to support domain experts in exploring and understanding of big data sets collected via sensors or by humans in a specific domain. Since domain typically experts are not familiar with advanced and abstract data mining, we have shown how a visual analytics method and tool can be developed to support non-programmer domain experts in understanding complex data sets. We have focused on empowering domain experts within energy management with the ability to visually analyze energy behavior based on consumption meters, sensors and user reported survey data. The development the AffinityViz method to support interdisciplinary analysis of interweaved consumption data and survey data is a novel way of visualizing and analyzing data relating to large buildings, while retaining a direct affine relationship to the buildings. It fills a gap in existing applications for visual analysis of resource consumption data by implementing the simple, yet highly affine, AffinityViz method for interweaved analysis. Specifically, applying the AffinityViz method has given our tool a relatability that enables it to apply to experts from diverse professions, thus democratizing the analysis of complex and interrelated data that are conventionally reserved for experts with high technical proficiencies making complicated calculations. Through real world usage examples, we have documented how AffinityViz fits into and supports interdisciplinary analysis in a comprehensive use case covering interweaved analysis of different types of consumption data and comprehensive survey data representing inhabitant characteristics. Furthermore, the AffinityViz tool has been validated through evaluations with 10 energy management professionals, who did not have prior experience with the tool and yet were able to identify fairly elaborate and advanced findings about resource consumption. Therefore, we conclude that applying the AffinityViz technique to support interweaved analysis of consumption and survey data is a highly promising approach, both in interdisciplinary settings, as well as for energy management professionals, and even for individual users. The detailed insight about resource consumption can be extended to many more parameters about indoor climate, occupant satisfaction, etc. However, it is important to take proper privacy measures when engaging in the analysis and revealing the results outside the professional groups who have access to the data being analyzed. We currently obfuscate the pictures of the tool to make it impossible to trace consumption measures back to individuals living in the building.

As a final remark we wish to emphasize that the AffinityViz method is applicable beyond the application to building and energy management. The ideas of coupling analysis results to a familiar shape of objects and buildings being handled in a domain, are promising beyond the buildings case. For instance, manufacturing plants and data from machinery may be analyzed in a similar manner with a slightly modified results model. We thus see promise in undertaking experiments on the same conceptual and tool platform in new domains in order provide more easy to grasp visual analytics tools for multiple domains.

9. ACKNOWLEDGMENTS

The work was supported by Danish DSF grant no. 11-115331. We wish to thank our colleagues in the EcoSense project for their contributions.
REFERENCES


