Towards Perception-aware Interactive Data Visualization Systems

Eugene Wu, *Member, IEEE*, and Arnab Nandi, *Member, IEEE*

Abstract— An often overlooked element of the interactive data visualization stack is the human in the loop. While computational and data processing capabilities have increased over time, human limits have remained constant. In this light, we describe extensions to client-server database-driven visualization systems that are both customized to interactive workloads, and support *perceptual models* that approximate the human's ability to decode visually encoded information. We recognize and accommodate human perceptual limitations as a way to minimize computation, network and rendering costs, and support high frame-rate interactions. Based on these models, we propose to answer a critical question: how can these models inform approximation decisions that improve end-to-end visualization performance? In this short paper, we describe research efforts towards using these limits to automatically approximate data transformations that are perceptually indistinguishable while applying database optimization techniques to minimize latency.

Index Terms—visualization, interaction, perception

Data visualizations are an information-dense and intuitive method to represent and communicate information. The ideal visualization tool should made it easier to both perform ad-hoc explorations of a new dataset, and to create highly interactive dashboards that consumers can use to explore a curated subset of the data. In contrast to existing visualization tools that largely present a small handful of analyses in the form of static visualizations, the increasing client processing power, and the advent of novel input interfaces such as touch, pen, and gestures [22] present the opportunity to support an increasingly popular class of direct manipulation visualization interfaces. This form of visualization not only gives users the ability to freely explore facets of the data they are interested in, and follow their own hypotheses, but speeds the responsiveness of the visualization so that responding to user inputs is eliminated as a bottleneck in data analysis.

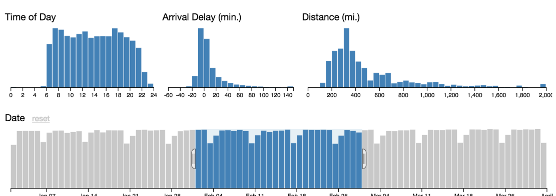

Fig. 1: Multi-dimensional interactive visualization of checkin data.

As an example, Figure 1 illustrates a multidimensional brushing and linking visualization [19] that renders the count of airline flights[1] along different dataset attributes. Selecting over any part of the bar charts will filter the dataset by the selected attribute value and update the counts in the other views – each movement of the user's mouse translates into executing and rendering a new set of database queries with different grouping or filtering parameters. By ensuring that the visualization reacts to user feedback within tens of milliseconds, the individual visualizations (frames) become perceptually fused [2] into a single, smooth animation. This form of immediate feedback enables "what-if" questions that are otherwise cumbersome to ask using point and click interfaces [6]. For example, users may wonder if there is a trend across a connected stretch of mid-western states and simply drag their mouse across a swath of the U.S. to see statistics change in the visualization in real time. This allows the user to use the *motion* in the interactive visualization as an additional source of insight. Looking towards the future, visualization interfaces, virtual reality, and motion tracking-based systems all demand such high frame rate interactions.

- *Eugene Wu is an Assistant Professor at Columbia University. E-mail: ewu@cs.columbia.edu.*
- *Arnab Nandi is an Assistant Professor at The Ohio State University. E-mail: arnab@cse.osu.edu.*

[1]This data is part of the ASA Data Expo dataset. Screenshot from `http://square.github.io/crossfilter/`

In order for direct manipulation to feel natural, it is imperative that interaction latency is minimized. Empirical evaluations demonstrate that increased latency adversely influences data exploration behavior [18], heavily motivating low latency (e.g. sub-second) performance windows for end-to-end feedback.

Unfortunately, today's client-server visualization architectures make creating highly interactive visualizations challenging. Such architectures combine a visualization front-end that sends queries to a DBMS backend through a generic database API, and renders the query results. Even as DBMS technology improves (e.g., columnar-stores, in-memory databases, approximate query processing), the roundtrip costs of network de/serialization and transfer, query processing, and rendering are unacceptably high for the hundreds of queries generated by user manipulations each second. In addition, as dataset sizes increase, and the demand for more powerful exploration, annotation, and analysis features [10, 28] continues to grow, supporting these interactions will only become more challenging. We believe the key issue is that the interface between the client and server continues to expect *generic, independent queries* that produce *exact answers*, which obscures the reality in which highly interactive visualizations generate bursts of strongly correlated queries whose results are ultimately perceived by humans.

Our proposed system, InterVis, approaches the client-server architecture from both sides . We find that many high-frame rate interactions follow a common query pattern and abstract this pattern into a set of SQL query extensions that we call *exploration specifications*. In addition, we note that the end product (visualization) is consumed by users through a lossy perceptual process, and explicitly model these inaccuracies as *perceptual functions*. Together, InterVis leverages both of these extensions to perform more aggressive approximations, precomputation, caching, and other optimizations that enable the ability to trade-off end-to-end interaction latency with the visualization's perceptual accuracy.

1 The InterVis System

InterVis is designed to be an end-to-end data exploration system that considers the overall interaction of a user with the visualization. By leveraging user-level aspects such as user perception and session-based interactions, the system is able to perform optimizations that were not possible with an interaction-agnostic approach.

Prior work in visualization systems often trade-off between expressiveness and performance. Expressive toolkits [3, 12] often require low-level programming that impedes the ability to quickly iterate, while declarative grammar-based languages [1, 31] allow for iteration during visualization design but are restricted within their host environments (SPSS, R). Similarly, tools such as Lyra [25] have focused on expressivity and interaction for *building* visualizations rather than interactivity of the final visualization itself. Recent systems address data scalability limitations by either adopting specific data management techniques such as pre-computation [19], indexing [17], sampling, speculation, and aggregation [16] or developing two-tiered architectures where the visualization client composes and sends queries to a data management backend [13, 23, 28]. The former approaches are optimized towards properties of specific applications or visualization types and may not be broadly applicable. The latter approach isolates each tier in the architecture and gives up numerous optimization opportunities. As shown in Figure 2, given the requirements of ad-hoc data exploration over large datasets, users are often left with a choice *between* **High frame-rate interactivity** and **Approximation**, instead of having both: this motivates the design of InterVis.

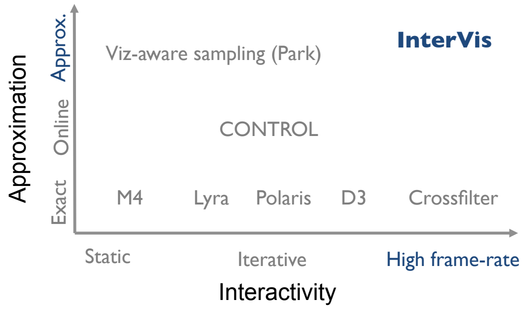

Fig. 2: Visualization projects by approximation type (y-axis) and rate of interactivity (x-axis).

1.1 Data and Execution Model

We use the relational data model for InterVis. For simplicity, we assume that a relation's attributes $\mathbb{A}$ can be partitioned into dimension $A_D \subseteq \mathbb{A}$, measure $A_M \subseteq \mathbb{A}$, and general $A_G \subseteq \mathbb{A}$ attributes.

We model visualization views as two-tier SQL aggregation queries with parameterized filter clauses. As a starting example, consider the *Time of Day* histogram in Figure 1. A simplified version can be represented by aggregating on `hour`, parameterized on the values of `arrival_delay`, `date` and `distance`:

```
SELECT hour, COUNT(*) FROM T
WHERE arrival_delay = ?  AND distance = ?  AND date = ?
GROUP BY hour
```

The parameterized values in the predicate clauses (e.g., the ? in `day=?`) are defined by the user's interactions when selecting bars in the histograms adjacent to the *Time of Day* histogram and can be any subset of the values in its corresponding attribute domain, or a special value `<all>` that matches all values. Similarly, each of the three histograms are themselves parameterized by the user's selection in the other visualization views. Under this model, an interaction such as moving the mouse across the date view represents a sequence of queries that vary the *date* parameter.

1.2 Exploration Specifications

We build upon this notion of parameterized queries as a fundamental construct of defining interactive visualizations. In order to articulate the separation of data exploration at the frontend and query processing at the frontend, we consider a *two-tier* approach: the data that is to be explored is considered a query, and then visualizations are modeled as queries over the result of the former query. Thus, our query model takes interactions into account by modeling visualizations as nested parameterized queries of the form:

```
SELECT gb0, ..., gbm, agg0(v0),...
  FROM (
    SELECT gb'0, ..., gb'p, agg'0(v'0),...
    FROM T1 (JOIN T2 ON ax)?
    WHERE gb'0 = ?  and ... a'0 = ?  ...
    GROUP BY gb'0, ..., gb'p
  ) as exploration_data
WHERE gb0 = ?  and ... a0 = ?, ... and an = ?
GROUP BY gb0, ..., gm
RENDERED BY <chart>, E1, ...,
PERCEIVED BY P1,...
```

where `gb` denotes `GROUP-BY`-eligible dimensions, e.g., `lat, lon`. Filter predicates are denoted by `a`, e.g., `hour`. `agg` denotes analytic measures, e.g., `COUNT`. The presence of `?` denotes scope for *interactive exploration* along that element – allowing a user to explore not only filter predicates, but also grouping conditions and joins.

`RENDERED BY` and `PERCEIVED BY` are **visualization specific language extensions** to support models for visualization rendering and human perception, respectively. The former clause specifies a chart type that the query should be rendered in, followed by a series of invertible *encoding functions* E_i that map attributes in the `SELECT` clause to properties of the visualization elements e.g., height of a bar. For example, if the visualization linearly maps $agg_0(v_0)$ from a domain of $[0, 10^6]$ to a canvas height of 100 pixels, the function effectively discretizes the input domain using the function $E_{height}(v) = floor(\frac{v}{10^4})$. The `PERCEIVED BY` clause defines a set of *perceptual functions* P_i that model the user's inaccuracy when perceiving visually encoded information e.g., decoding numerical values encoded in color. These new clauses capture the visualization semantics that are necessary for the optimizations in the next sections. To summarize the end-to-end process, the value that a user perceives is first computed by the standard SQL components of the nested query, then encoded into pixels using the encoding functions, and finally, the human eye's decoding process is modeled by applying the perceptual functions.

We call this two-tier parameterization an **exploration specification** for an interactive visualization. It allows developers to articulate the demands that an interactive visualization will be putting forth towards the database. There are several benefits to this model: by decoupling the visualization from the query, both the visualization and the underlying data can be interchangeably swapped out with a replacement during data exploration. Another benefit of our template-based query model is that we can distinguish between different classes of parameterization depending on which query clause is parameterized: `JOIN` clause (e.g., $\mathtt{a_x} =?$), `GROUP BY` clause (e.g., $\mathtt{gb_i} =?$), filter on aggregated attribute (e.g., $\mathtt{v_i} =?$), or filter on a non-aggregated attribute (e.g., $\mathtt{a_i} =?$). Each of these conditions can be catered to by a variety of optimizations to enable fluid and interactive visualizations. Thus, exploration specifications are a convenient and concise representation of visualization stacks, providing algorithmic hooks for several independent research contributions.

1.3 InterVis Architecture

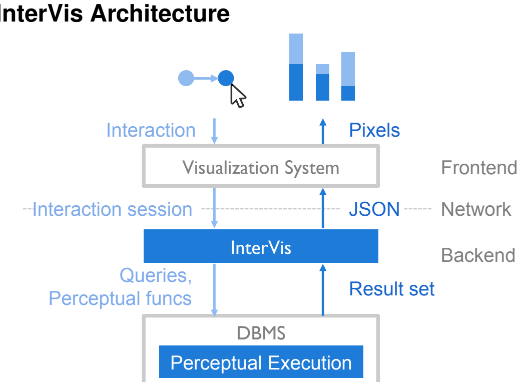

Fig. 3: System architecture

InterVis is designed as a client-server system where a visualization frontend translates user interactions into a sequence of annotated SQL queries that are executed in a backend database system. The key idea of the system is that **every layer of the architecture is interaction-aware**: requests throughout the system are *session-based* and interaction hints such as models of *perceptual inaccuracy* are factored into both the frontend and the backend execution engine.

Current interactions in visualizations, such as sliders and scroll bars, will generate a new query for each user action (e.g., dragging the bar by one pixel). Without additional information, InterVis would simply trigger a re-computation of the entire query and rendering pipeline. Since requerying for each frame with the backend would involve roundtrips through multiple levels of abstraction and network latencies, serving such queries at scale can be slow, wasteful, and moreover untenable, which can prove detrimental to the data exploration process [18]. Thus, we model a series of such queries as part of a *session*, where there is little difference between successive queries within a session, e.g., queries differing only on the value of one of their `WHERE` predicates. We call each query in the session a *frame*, to underscore the importance of the interaction's animation.

In the context of this architecture, we focus on a perceptually motivated challenge: Instead of simply considering the task of generating a visual representation of the query result, we consider the *insight perceived* by the user from the interaction with the visualization. Given large datasets and human limitations in perceiving fine-grained details, we detail modifications to the database execution engine that facilitate the use of visualization-aware approximation. In the remaining sections of this short paper, we focus on our approach towards perceptually accurate visualizations through the use for *perceptual functions*.

2 Perceptual Functions for Invisible Approximate Interactions

Practical limits due to both the finite pixel density of the output viewport and human limitations in perceiving small differences in visually encoded values [5] (e.g., color, position) make approximation a natural fit for quickly computing and rendering data visualizations without large perceivable differences. Although query approximation has been well-studied [24], they neither take the interaction sessions into account, nor are designed for supporting bursts of queries during an interaction. The latter is problematic, because the faster the user interacts with a visualization, the higher the rate of queries yet the less time available to service each query. Simply increasing the approximation error until each query can be computed quickly enough is undesirable, because the error bounds may be impractically large.

One approach is to develop specialized approximation algorithms that preserve specific features in a visualization e.g., pairwise relative differences in a bar chart. However, we would need to enumerate every visualization feature and custom tailor an algorithm for each one. In contrast, we observe that the visualization, psychology, and HCI communities have established and are actively developing mathematical models of graphical perception [5, 7, 9, 15, 26, 27, 30] that may be used in a general optimization framework demarcating the human limits of perceiving visual changes. These models hold promise for improving query performance for interactions.

An important class of graphical perceptual models describe how accurately humans can perceive (i.e. decode) values that are visually embedded as e.g., the height of a bar in a bar chart. For example, the power law of psychophysics models the perceived magnitude of a visually encoded value using a power law relationship with the actual value [27]. A related class of models [5, 9, 30] measure the perceived error when making proportional comparisons between visual encodings, for example, comparing the heights of two adjacent bars in a bar chart. Although these results have been used to make qualitative judgements of visualizations and help rank visual encodings by effectiveness (e.g., encoding numerical values using length is preferable to area), usage of these models for query execution has not been explored.

In light of this area of research, we are studying the use of *perceptual functions* in the `PERCEIVED BY` clause in the query execution engine for making approximation and filtering-based optimization decisions. A perceptual function is defined for a specific combination of visual properties vp (e.g, height in a bar chart) and returns the perceived error (e.g., perceiving a 100 pixel height by ± 5 pixels). We currently focus on two general forms of functions: univariate functions $P_e^u : \mathbb{R} \to \mathbb{R}$ that map a visually encoded value to the perceived error, and bivariate functions $P_e^b : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that map a pair of encoded values to the error in the perceived proportional differences. In both cases, the computed error $\pm\varepsilon$ describes the value range that the encoded value may be perceived within. This general formulation of the perceptual functions lets InterVis support a range of models found in the literature — from models that compute a single error value for an encoding (e.g., 2% when comparing adjacent bars [5]), to those whose error varies as a function of the true proportional difference [5], to those that depend on the magnitude of the true values [8]. In addition, InterVis does not rely on any specific perceptual functions and can thus adapt to new models as they are developed and refined, as well as support personalized perceptual models [4].

InterVis analyzes and annotates new interaction sessions with the applicable perceptual functions that match the encodings used by the visualization. These perceptual functions are composed with similar functions defined by the encodings in the `RENDERED BY` clause. For example, the universally applicable perceptual function[2] $P_*^u(v)$ is composed with its matching encoding function E to create the function $(E \circ P_*^u)(v)$ that is sent to the query executor. During execution, InterVis uses *perceptual approximation* to prioritize and manage approximation decisions for the query results, and *perceptual filtering* to avoid computing and returning results. The techniques we describe below integrate ideas from prior database sampling and online aggregation work [11] with perceptual functions to build towards automated optimization approaches based on these functions. By informing the execution engine of human perception, we are able to reduce computational requirements to serve each exploration specification.

Perceptual Approximation: A naive approach to using perceptual functions for approximation is to use the perceived error as the error bound for picking the input sample size. For example, suppose the query result is a single value v, the encoding and perceptual functions are E and P, respectively, and the perceived error is $\varepsilon = (E \circ P)(v)$. Then the user perceives the visual encoding between $\underline{\varepsilon} = v - \varepsilon$ and $\bar{\varepsilon} = v + \varepsilon$. Inverting E and P results in the perceivable margin of error for the query result $(E \circ P)^{-1}(\bar{\varepsilon}) - (E \circ P)^{-1}(\underline{\varepsilon})$. Finally, InterVis can derive an estimate of the sample size needed to compute v within the margins at a given confidence interval by using closed form equations [20].

As a proof of concept to evaluate the potential benefits, we ran a preliminary experiment using a synthetic dataset containing 1 million single-attribute records and measured the effect on sample size when the aggregated result (using `AVG`) maps to opacity. We generated each attribute value from a normal distribution with mean μ and standard deviation $\sigma = 10$, and varied μ from 1 to 10^5. Furthermore, let $\bar{v}, \underline{v}$ be the minimum and maximum attribute values in the dataset. We evaluated all combinations of four `PERCEIVED BY` clauses: each with a constant function that returns $10^{-5}, 50^{-4}, 10^{-4}$, or 50^{-3} – and two encoding functions for opacity: a simple linear mapping $[\underline{v}, \bar{v}] \to [0,1]$ into the opacity space $E_1(v) = \frac{v - \underline{v}}{\bar{v} - \underline{v}}$, and a mapping that is approximately *perceptually linear*[3] $E_2(v) = 0.15 + \frac{v - \underline{v}}{\bar{v} - \underline{v}}^{\frac{1}{3}} \times (1 - 0.15)$. Figure 4 reports the average determined sample size (log scale) as a function of μ over 20 runs. We find that for both encoding functions, even a small perceptual error of 10^{-5} can reduce the the determined sample size by $10^4\times$, whereas larger perceived errors can reduce the size by orders of magnitude. In addition, the trends for the perceptually linear encoding function show that the sampling size can depend significantly on the aggregated value.

Although these results are promising, the key limitation of the naive approach is that it assumes v has been fully computed ($P(v)$ depends on v), however, that assumption defeats the purpose of sampling during query execution. One direction is to push the evaluation of perceptual functions into the query execution process, so that the margins of error are refined in concert with the estimation of the aggregation result values. Further, we plan to study various stopping criteria based on how quickly the estimated aggregation results converge, both for each query result individually, and for the query result set as a whole.

Perceptual Filtering: Whereas the previous approach performs optimization for individual queries, perceptual filtering compares query re-

[2]InterVis uses the error from decoding a visually encoded value to optimize query execution, e.g., $P_*^u = 0$ is the most conservative possible function.

[3]This mapping is borrowed from prior work [19], which in turn used results from CIELAB [29] and prior results for luminance contrast [9, 21]

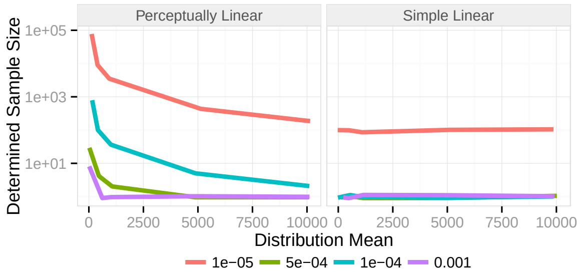

Fig. 4: Determined sample size as a function of true distribution mean for different perceptual functions.

sults between adjacent interactive frames (e.g., Q_{i-1} and Q_i) and filters away results in Q_i that are perceptually similar to those in Q_{i-1} (where Q_{i-1} has been executed already and Q_i is the current query). This filtering uses the bivariate perceptual functions P^b to compare records with the same `GROUP BY` values in Q_{i-1} and Q_i, and filter records in Q_i whose estimated difference to the corresponding record in Q_{i-1} is within the error computed by P^b. The approach serves to reduce the network latency for both transferring results to the client and updating the rendered visualization (similar to M4 [14]), and to decrease the required cache size in the client cache. The key challenge is to push this filtering process into the query plan so that intermediate results that will not be perceived are pruned early on. In addition to techniques similar to perceptual approximation, univariate perceptual functions present opportunities to more aggressively prune intermediate results by analyzing how the univariate function's error bound overlaps with that of the bivariate functions. Window functions provide a suitable mechanism for expressing perceptual filtering, and similar ideas may be considered when frames are not adjacent (e.g., Q_i and Q_{i+n} where $n > 1$) or when comparing more than two frames.

Function Selection: Although an exploration specification may define multiple functions in the `PERCEIVED BY` clause, it is unclear how multiple perceptual functions will be used together for the above, nor is it clear if combining them is safe. Thus, an additional challenge is to select the optimal perceptual function to use. For the former, given a set of functions $\mathbb{P} = \{P_1, \ldots, P_n\}$, how should the optimal $P* \in \mathbb{P}$ be selected for a given query? This policy depends on the curve of each function, as well as the values of each result record. For example, the univariate functions $P_1^u(v) = 0.5 \times v$ is preferable to $P_2^u(v) = 1$ when $v < 2$, however the latter provides more optimizations when $v > 2$. Thus, efficiently picking the optimal $P*$ must be performed during query execution.

Perceptual Experiments: Interaction results in animation, however the perception of animated data visualization is poorly understood despite numerous perceptual studies for static visualizations and time-varying encoding such as video and audio. In short, perceptual functions for animated data visualizations are an uncharted area of work. We have been running two perceptual judgment experiments in the context of animated bar charts (e.g., a bar chart changes in response to user interactions with a scroll bar). Our studies vary data properties such as when the target bar reaches a maximum or simulated forms of approximation, as well as animation properties such as the frame rate or how a target bar is marked. In the value reading [32] task, users are asked to estimate the maximum value of a target bar during the animation. Preliminary results have found that when the target bar does not exhibit sudden changes (e.g., an impulse), user perception is largely invariant of the frame rate, even in the presence of noise. In contrast, perceptual accuracy drops significantly if the target achieves a maximum value early (in the first 10%) in the animation.

These findings can help us better understand which components of an animation are good candidates for approximation, at what levels of approximation, and under which settings.

Acknowledgements: We acknowledge the generous support of the National Science Foundation under awards IIS-1422977, IIS-1527765 / 1527779, and CAREER-1453582.

References

[1] ggplot2. `ggplot2.org`.
[2] P. Bertelson and M. Radeau. Cross-modal bias and perceptual fusion with auditory-visual spatial discordance. *Perception And Psychophysics*, 1981.
[3] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *InfoVis*, 2011.
[4] W. S. Cleveland et al. Human factors and behavioral science: Experiments on quantitative judgments of graphs and maps. *Bell System Technical Journal*, 1983.
[5] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 1984.
[6] M. C. F. De Oliveira and H. Levkowitz. From visual data exploration to visual data mining: A survey. *TVCG*, 2003.
[7] J. Falmagne. The generalized fechner problem and discrimination. *Journal of Mathematical Psychology*, 1971.
[8] L. Harrison, F. Yang, S. Franconeri, and R. Chang. Ranking visualizations of correlation using weber's law. *InfoVis*, 2014.
[9] J. Heer and M. Bostock. Crowdsourcing graphical perception: Using mechanical turk to assess visualization design. 2010.
[10] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *CACM*, 2012.
[11] J. M. Hellerstein et al. Online aggregation. *SIGMOD*, 1997.
[12] J. D. Hunter. Matplotlib: A 2d graphics environment. *CISE*, 2007.
[13] J.-F. Im, F. G. Villegas, and M. J. McGuffin. Visreduce: Fast and responsive incremental information visualization of large datasets. In *BigData Conference*, 2013.
[14] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: A visualization-oriented time series data aggregation. *VLDB*, 2014.
[15] C. Kaernbach and C. Bering. Exploring the temporal mechanism involved in the pitch of unresolved harmonics. *Acoustical Society of America*, 2001.
[16] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and Interactive Cube Exploration. In *ICDE*, 2014.
[17] L. D. Lins, J. T. Klosowski, and C. E. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *TVCG*, 2013.
[18] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *TVCG*, 2014.
[19] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, 2013.
[20] S. L. Lohr. *Sampling: Design and Analysis*. Cengage Learning, 2010.
[21] L. B. Maureen Stone. Alpha, contrast and the perception of visual metadata. In *Color Imaging Conf*, 2009.
[22] A. Nandi. Querying without keyboards. In *CIDR*, 2013.
[23] A. Parameswaran, N. Polyzotis, and H. Garcia-Molina. Seedb: Visualizing database queries efficiently. *PVLDB*, 2014.
[24] S. A. Phillip, P. B. Gibbons, and V. Poosala. Aqua: A fast decision support system using approximate query answers. 1999.
[25] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Computer Graphics Forum*, 2014.
[26] H. Schiffman. Chapter twenty psychophysics. *Handbook of Research Methods in Experimental Psychology*, 2008.
[27] S. Stevens. *Psychophysics*. Transaction Publishers, 1975.
[28] C. Stolte and P. Hanrahan. Polaris: A system for query, analysis and visualization of multi-dimensional relational databases. *InfoVis*, 2002.
[29] M. C. Stone. *A field guide to digital color.* A K Peters, 2003.
[30] J. Talbot, V. Setlur, and A. Anand. Four experiments on the perception of bar charts. *TVCG*, 2014.
[31] L. Wilkinson. *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag New York, Inc., 2005.
[32] H. Yang, Y. Li, and M. X. Zhou. Understand users: Comprehension and preferences for composing information visualizations. *CHI*, 2014.