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 make 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.

As an example, Figure 1 illustrates a multidimensional brushing and linking visualization [19] that renders the count of airline flights1 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 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 each second. In addition, as dataset sizes increase, and the demand for more powerful exploration, annotation, and analysis features 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 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, pre-computation, caching, and other optimizations that enable the ability to trade-off end-to-end interaction latency with the visualization’s perceptual accuracy.

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 flights1

---

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

1.1 Data and Execution Model

We use the relational data model for InterVis. For simplicity, we assume that a relation’s attributes \( A_D \) can be partitioned into dimension \( A_D \subseteq A \), measure \( A_M \subseteq A \), and general \( A_G \subseteq A \) 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 \( \text{hour} \), parameterized on the values of \( \text{arrival}_{\text{delay}} \), \( \text{date} \), and \( \text{distance} \):

\[
\text{SELECT hour, COUNT(*) FROM T1}
\text{WHERE arrival}_{\text{delay}} = ? \quad \text{AND} \quad \text{distance} = ? \quad \text{AND} \quad \text{date} = ?
\]

The parameterized values in the predicate clauses (e.g., the ? in \( \text{day} = ? \)) 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 \(<\text{all}>\) 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 \( \text{date} \) 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 that map attributes in 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: \( \text{JOIN} \) clause (e.g., \( a_k = ? \)), \( \text{GROUP BY} \) clause (e.g., \( gb_2 = ? \)), filter on aggregated attribute (e.g., \( v_2 = ? \)), or filter on a non-aggregated attribute (e.g., \( a_3 = ? \)). 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

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 querying for each frame with the backend would render 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 \( \text{SELECT} \) clause to properties of the visualization elements e.g., height of a bar. For example, if the visualization linearly maps \( agg_{\text{group}}(v_i) \) from a domain of \([0, 10^6]\) to a canvas height of 100 pixels, the function effectively discretizes the input domain using the function \( E_{\text{group}}(v_i) = \text{floor}\left(\frac{v_i}{10^6}\right) \). 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 put 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: \( \text{JOIN} \) clause (e.g., \( a_k = ? \)), \( \text{GROUP BY} \) clause (e.g., \( gb_2 = ? \)), filter on aggregated attribute (e.g., \( v_2 = ? \)), or filter on a non-aggregated attribute (e.g., \( a_3 = ? \)). 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.
2 PERCEPTUAL FUNCTIONS FOR INVISIBLE APPROXIMATE INTERACTIONS

Practical limits due to both the finite pixel density of the output view-  
port and human limitations in perceiving small differences in visually  
encoded values [5] (e.g., color, position) make approximation a nat-  
ural fit for quickly computing and rendering data visualizations with-  
out large perceivable differences. Although query approximation has  
been well-studied [24], they neither take the interaction session into  
account, nor are designed for supporting bursts of queries during an  
interaction. The latter is problematic, because the faster the user inter-  
acts 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 ev-  
ery 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 mathemat- 
ical 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 ac- 
tual value [27]. A related class of models [5, 7, 9, 30] measure the per- 
ceived 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 quali- 
ative judgements of visualizations and help rank visual encodings by  
effectiveness (e.g., encoding numerical values using length is prefer- 
able 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 percep- 
tual functions in the PERCEIVED BY clause in the query execution  
engine for making approximation and filtering-based optimization de- 
cisions. A perceptual function is defined for a specific combination  
of visual properties \( v \) (e.g., height in a bar chart) and returns the per-  
ceived error (e.g., perceiving a 100 pixel height by \( \pm 5 \) pixels). We  
currently focus on two general forms of functions: univariate func- 
tions \( P^v : \mathbb{R} \to \mathbb{R} \) that map a visually encoded value to the per-  
cieved error, and bivariate functions \( P^v : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \) that map a pair of en- 
coded values to the error in the perceived proportional differences. In  
both cases, the computed error \( \pm \epsilon \) 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 ad- 
dition, 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 simi- 
lar functions defined by the encodings in the RENDERED BY clause.  
For example, the universally applicable perceptual function \( P^v E^v(v) \)  
is composed with its matching encoding function \( E \) to create the func- 
tion \( (E \circ P^v)(v) \) that is sent to the query executor. During execution,  
InterVis uses perceptual approximation to prioritize and manage ap- 
proximation 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 aggre- 
gation work [11] with perceptual functions to build towards automated  
opimization approaches based on these functions. By informing the  
execution engine of human perception, we are able to reduce computa- 
tional 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 \( \epsilon = (E \circ P)(v) \).  
Then the user perceives the visual encoding between \( \bar{v} = v - \epsilon \) and  
\( \bar{v} = v + \epsilon \). Inverting \( E \) and \( P \) results in the perceivable margin of  
error for the query result \( (E \circ P)^{-1}(\bar{v}) - (E \circ P)^{-1}(v) \). 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 equa- 
tions [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 \( \mu \) and standard  
deviation \( \sigma = 10 \), and varied \( \mu \) from 1 to 10. Furthermore, let \( \bar{v} \) be  
the minimum and maximum attribute values in the dataset. We eval- 
uated all combinations of four PERCEIVED BY clauses: each with a  
constant function that returns \( 10^7 \times 10^{5} \times 10^{-3} \) and \( 50 \) and two  
encoding functions for opacity: a simple linear mapping \( [0, 1] \) into  
the opacity space \( E_1(v) = \frac{v - \bar{v}}{\bar{v} - \bar{v}} \), and a mapping that is approxi- 
mately perceptually linear\(^2\) \( E_2(v) = 0.15 + \frac{v - \bar{v}}{\bar{v} - \bar{v}} \times (1 - 0.15) \).  
Figure 4 reports the average determined sample size (log scale) as a  
function of \( \mu \) 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 \( 10^7 \times \), 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 sig- 
ificantly 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 dur- 
ning query execution. One direction is to push the evaluation of per-  
ceptual 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 opti- 
mization 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., \( P^v = 0 \) 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]
Function Selection: Although an exploration specification may define multiple functions in the PERCEIVED BY clause, it is unclear how multiple perceptual functions can 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 \( P = \{ P_1, \ldots, P_n \} \), how should the optimal \( P^* \in P \) 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 \( P^i_1(v) = 0.5 \times v \) is preferable to \( P^i_2(v) = 1 \) 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

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