Eurographics Conference on Visualization (EuroVis) 2013
B. Preim, P. Rheingans, and H. Theisel
(Guest Editors)

Volume 32 (2013), Number 3

imMens: Real-time Visual Querying of Big Data

Zhicheng Liu[⋆], Biye Jiang[‡] and Jeffrey Heer[⋆]

[⋆]Department of Computer Science, Stanford University
[‡]Department of Computer Science and Technology, Tsinghua University

Abstract

Data analysts must make sense of increasingly large data sets, sometimes with billions or more records. We present methods for interactive visualization of big data, following the principle that perceptual and interactive scalability should be limited by the chosen resolution of the visualized data, not the number of records. We first describe a design space of scalable visual summaries that use data reduction methods (such as binned aggregation or sampling) to visualize a variety of data types. We then contribute methods for interactive querying (e.g., brushing & linking) among binned plots through a combination of multivariate data tiles and parallel query processing. We implement our techniques in imMens, a browser-based visual analysis system that uses WebGL for data processing and rendering on the GPU. In benchmarks imMens sustains 50 frames-per-second brushing & linking among dozens of visualizations, with invariant performance on data sizes ranging from thousands to billions of records.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces]: User Interfaces—;

1. Introduction

Traditional data visualization tools are often inadequate to handle big data. While it is debatable what is meant by "big", visualization researchers have regularly used one million or more data cases as a threshold [FP02, UTH06]. More generally, many data sets are too large to fit in memory and may be distributed across a cluster; modern data warehouses often include tables with billions or more records. Most visual analysis tools are not designed to work at this scale, let alone support real-time interaction [KPHH12].

Research on big data visualization must address two major challenges: *perceptual* and *interactive* scalability. Given the resolution of conventional displays (~1-3 million pixels), visualizing every data point can lead to over-plotting and may overwhelm users' perceptual and cognitive capacities. On the other hand, reducing the data through sampling or filtering can elide interesting structures or outliers. Big data also impose challenges for interactive exploration. Querying large data stores can incur high latency, disrupting fluent interaction. Even with data reduction methods like binned aggregation, high dimensionality or fine-grained bins can result in data cubes too large to process in real-time.

In this paper we present techniques to address perceptual and interactive scalability, following the principle that scalability should be limited by the chosen resolution of the visualized data, not the number of records. We realize our techniques in *imMens*[†], a browser-based system for real-time interaction with scalable visual summaries of big data.

To support perceptual scalability, we first review applicable data reduction methods, including filtering [AS94], sampling [DSLG*12, MTS91] and aggregation [CLNL87, JS98]. We select binned aggregation as our primary data reduction strategy and describe a design space of binned plots for numeric, ordinal, temporal and geographic variables.

We then address interactive scalability for panning, zooming and brushing & linking in binned plots. As the number of visualized dimensions increases, the size of the supporting data can explode combinatorially (c.f., [KPP*12]). In response, we develop methods for real-time visual querying:

Precompute Multivariate Data Tiles. Precomputed image tiles, as in Google Maps and Hotmap [Fis07], are a common solution for scalable panning and zooming. Rather than generate image tiles intended for direct display, we instead precompute *multivariate data tiles*: projections corresponding to materialized database views [GM99]. By decomposing a data cube into a set of 3- and 4-dimensional projec-

[†] The name *imMens* stems from our desire to visualize *immense* data in a manner that our minds (Latin: *mens*) can apprehend.

tions, *imMens* flexibly manages data subsets as needed. Using multidimensional projections, *imMens* can compute aggregations across dimensions to support brushing & linking.

Parallelize Data Processing and Rendering. Depending on binning resolution, data tiles may still be quite large, with millions or more values. To speed aggregation, *imMens* uses a dense indexing scheme that simplifies parallel query processing. To realize this approach in contemporary web browsers, *imMens* uses WebGL to leverage parallel processing on the GPU. We present a two-pass approach that uses WebGL shader programs to first compute aggregate queries and then render updated visualizations.

The contributions of our work are two-fold. First, we introduce the use of *multivariate data tiles* for pre-processing and dynamic loading of data to enable scalable interaction. Second, the *imMens* system implements a novel synthesis of binned aggregation, data representation and parallel processing (here using GPU computation) to support interactive visualization of big data. In performance benchmarks *imMens* sustains near 50 frames-per-second brushing and linking among dozens of visualizations. This performance is invariant on data sets ranging from thousands to billions of records. To our knowledge, *imMens* is the first system to enable real-time interactive brushing of data sets this large.

2. Related Work: Visualizing Big Data

A number of prior research projects have focused on improving the scalability of visualization systems.

2.1. Scalability of Visual Encodings

In many visualizations, each data record maps to a visual item, resulting in occlusion and cluttering for high data densities. In response, researchers have proposed a number of approaches. Pixel-oriented visualization techniques plot data points as single pixels to maximize screen utilization [Kei00]. Spatial displacement techniques like jittering [TGC03] and topological distortion [KHD*10] reduce occlusion but do not preserve spatial information. For parallel coordinates and scatterplot matrices, dimension reordering can also reduce clutter [PWR04]. Alpha blending (transparency) is often used to encode density and thus combat over-plotting [JS98, JLJC05, UTH06]. Alpha blending effectively performs aggregation in image space, rather than data space. Still, each of these techniques requires drawing every data record, which imposes inherent scalability limits.

An alternative is to *reduce* big data to smaller, derived data more amenable to visualization [DBC*13]. Data reduction strategies used in information visualization include filtering [AS94], sampling [BS06, DSLG*12, Raf05], binned aggregation [BBSBA08, CLNL87, EF10, Fis07, HDS*10, KMSH12, KPP*12, RWC*08], and model-fitting. We compare these approaches in more detail in the next section and describe a design space for binned plots.

2.2. Scalable Visualization Systems

In addition to work on visualization design, both researchers and companies have developed large scale visualization systems. Commercial products such as Tableau [Tab] and Spotfire [Spo] translate user interactions into database queries, and can push processing of big data to dedicated databases. The query results are typically aggregates, such that the visualizations are perceptually scalable. This approach has two potential issues: query latency can be high for large data sets, and there is no guarantee that the result size has a reasonable upper bound. To reduce long query times, some systems prefetch based on the user's current context [CXGH08, DRW03]. To improve performance, other researchers exploit modern hardware such as multi-core [RWC*08, PTMB09, HB10b] and GPU [FP02, ME09, ZBDS12] computing.

The system most similar to *imMens* is Kandel et al.'s Profiler [KPP*12]. Profiler employs binned plots to enable interaction with over a million data points. However, Profiler uses a single in-memory data cube and sequential query processing to support brushing & linking. As we show in our benchmarks, this approach does not scale to larger data sets.

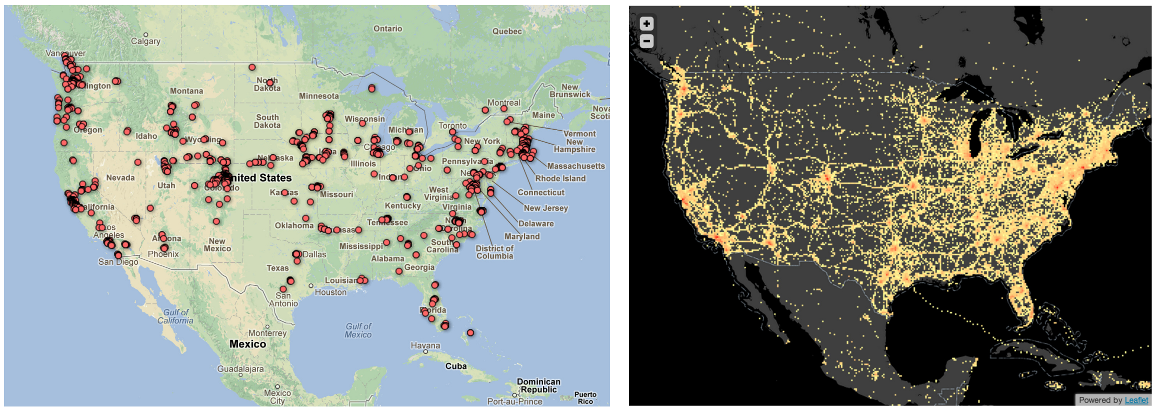
Our work extends and integrates these approaches. We assume a client-server architecture in which scalable databases can be used to precompute multivariate data tiles. *imMens* then requests and manages data tiles to address issues of data size and transfer, and uses parallel processing on the GPU for real-time querying and rendering on the client.

3. Data Reduction Methods

Our approach to visualizing big data follows an overarching principle: *perceptual and interactive scalability should be limited by the chosen resolution of the visualized data, not the number of records*. We now survey data reduction methods that can be used to realize this principle, including filtering, sampling, aggregation and modeling.

3.1. Filtering & Sampling

Filtering and sampling techniques select a subset of data, to which standard visualization techniques can be applied. The selected subset, however, may still be too large to visualize effectively and may omit elements of interest. In simple random sampling [Loh09], every data point has the same probability of being selected. The sample resulting may not be representative and can miss important structures or outliers. Systematic sampling sorts data points in a particular order and selects data points at regular intervals with a random start. Stratified sampling divides a data set into disjoint subgroups or "strata", and applies simple or systematic sampling within each stratum. However, these methods require that specific dimensions be chosen ahead of time, requiring prior knowledge and often costly pre-processing.



(a) (b)

Figure 1: *A symbol map (a) and heatmap (b) visualizing a dataset of Brightkite user checkins. The symbol map visualizes a sample of the data, and the heatmap shows the density of checkins by aggregation. Compared to the heatmap, sampling misses important structures such as inter-state highway travel and Hurricane Ike, while dense regions still suffer from over-plotting.*

3.2. Binned Aggregation

Binning aggregates data and visualizes density by counting the number of data points falling within each predefined bin. For a numeric variable, one can define bins as adjacent intervals over a continuous range. For categorical variables, one can simply treat each value as a bin. Aggregation can also be defined at multiple scales over a hierarchy [EF10], with nested, potentially non-uniform, bins. For example, temporal values can be aggregated by day, week, month, quarter, year, and so on. In terms of visualization, histograms and heatmaps are exemplary 1D and 2D binned plots.

3.3. Model-based Abstraction

Another reduction strategy is to describe data in terms of mathematical models or statistical summaries. For example, one might fit a model and visualize the resulting parameters or theoretical density. For scatter plots one can use regression models to fit trend lines; examples for time series data include moving averages and auto-regressive models.

3.4. Hybrid Reduction Methods

The above data reduction methods can also be combined. For example, a box plot with outliers applies both modeling and filtering. In this vein, Novotný and Hauser [NH06] perform two dimensional binning for parallel coordinates and show specific data outliers along with the bins. To visually summarize large networks, both Bagrow et al. [BBSBA08] and Kairam et al. [KMSH12] combine modeling and aggregation through multi-scale histograms of network statistics.

Database researchers have explored the combination of sampling and aggregation. To provide fast approximate queries, BlinkDB [APM*12] builds multi-dimensional and multi-resolution stratified samples and computes aggregates over this reduced data. However, this approach still suffers from the same problems with sampling discussed above. Online aggregation [HHW97, FPDs12] shows continuously updating aggregates and confidence intervals in response to a stream of samples. Our approach is compatible with these two methods: one could compute approximate data tiles using the BlinkDB approach, or update data tiles in a streaming fashion via online aggregation. Though not explored here, these methods may provide low-latency results when complete data tiles have not yet been precomputed.

4. Designing Binned Plots

In *imMens* we focus on binned aggregation as our primary data reduction strategy. Here we present our rationale for using binned aggregation, and discuss the corresponding visualization design space for binned plots.

4.1. Why Binned Aggregation?

We use binned aggregation because it conveys both global patterns (e.g., densities) and local features (e.g., outliers), while enabling multiple levels of resolution via the choice of bin size. Consider Figure 1, which visualizes a data set of over four million user checkins on Brightkite, a location-based user checkin service, from April 2008 to October 2010 [CML11]. Figure 1(a) shows a symbol map of stratified samples generated by Google Fusion Tables [DSLG*12]. Figure 1(b) shows a binned heatmap in *imMens*, color-coded by the density of checkins at different locations. One can see richer information in the heatmap, including patterns on inter-state highways, events outside the US, and a long trail of checkins spanning the coast of Texas and Gulf of Mexico. These are checkins made by Brightkite user account “Hurricane Ike” that report the location of the hurricane along its path in 2008. Sampling can fail to show such interesting outliers.

4.2. Visualization Design for Binned Plots

We consider binning schemes for four data types commonly found in databases: ordinal, numeric, temporal and geographic. For ordinal and (sorted) categorical data, each distinct value is a bin. We group numeric data into adjacent intervals over a continuous range. Temporal values can be binned at various levels of granularity: year, month, week, day, hour, *etc*. For geographic data, we treat 1D nominal units such as states or countries as unique bins. If locations are specified as latitude and longitude points, we bin their projected spatial coordinates. We primarily use the Mercator projection for consistency with existing map tile providers.

Table 1 provides a summary of relevant visualization designs organized by data type and number of dimensions. One-dimensional plots for ordinal, numeric, temporal and geographical dimensions take the form of bar charts, histograms, line graphs and choropleth maps. Two-dimensional binned plots are heatmap variants; plots with heterogeneous data types (e.g., a temporal and an ordinal variable) are also possible. We focus on binned plots with up to two data dimensions, encoded spatially. Color is used to encode data density and indicate highlights for brushing & linking. We refrain from additional visual variables such as texture or size, as they might interfere with visualization interpretation. Multidimensional displays can be constructed in the form of multiple coordinated views and trellis plots [BCS96].

Figure 2 shows binned scatter plots [CLNL87] of two numeric dimensions. Unwin et al. describe three binning schemes that can tessellate a plot: triangular, rectangular and hexagonal [UTH06]. Carr et al. [CLNL87] argue for hexagonal bins due to reduced bias in density estimation compared to rectangular bins; however, Scott [Sco92] shows that the differences are marginal. In *imMens*, we choose rectangular binning for consistency and efficiency of query processing. Applying consistent binning schemes over 1D (e.g., histograms) and 2D plots ensures compatibility when performing linked selections between plots.

Statisticians have proposed various heuristics to select bin sizes for a numeric range (e.g., Sturges' formula [Stu26] and Scott's reference rule [Sco79]). These heuristics can vary

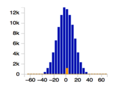
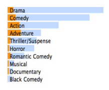
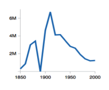
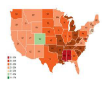
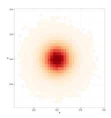
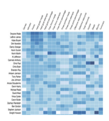
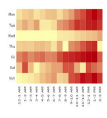
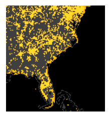
	Numeric	Ordinal	Temporal	Geographic
1D	 Histogram	 Bar Chart	 Line Graph / Area Chart	 Choropleth Map
2D	 Binned Scatter Plot	 Heatmap	 Temporal Heatmap	 Geographic Heatmap

Table 1: *Example visualization designs for binned plots.*

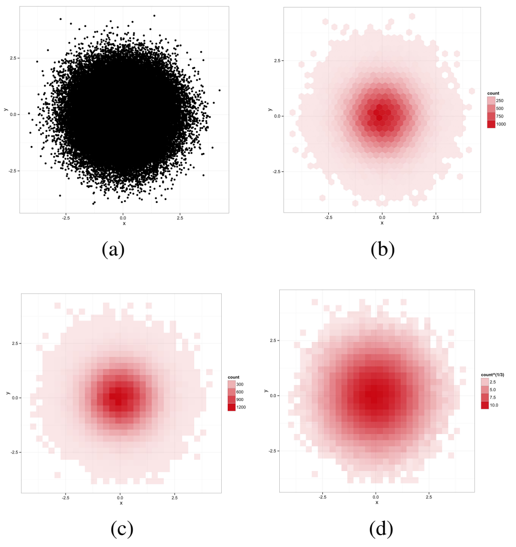


Figure 2: *Scatter plots with 100,000 data points: (a) traditional, (b) hexagonal bins, (c) rectangular bins and (d) rectangular bins with perceptual (cube root) color adjustment.*

significantly and their applicability to big data is unclear. In *imMens*, we treat bin count as an adjustable parameter, bounded by the screen pixels allocated to a plot and available resources. At the limit, we can map one bin to one pixel and include as many bins as memory constraints allow.

4.3. Color Encoding

For color encoding, one can map density values to hues, luminance or opacity. We map a non-zero density value x to a luminance (or opacity) value $Y \in [0,1]$ using the formula:

$$Y = \alpha + \left(\frac{\hat{x} - x_{min}}{x_{max} - x_{min}}\right)^{\gamma}(1-\alpha) \qquad (1)$$

Here $\hat{x}$ denotes the value of x, bounded from above and below by the range parameters x_{max} and x_{min}. These parameters can be determined from the data, or adjusted interactively to explore value ranges at finer resolutions.

Our color encoding employs two techniques to enhance perception. First, linear changes in Y values may not correspond to *perceptually* linear changes. The γ parameter can be used to introduce a non-linearity. By default we set $\gamma = \frac{1}{3}$, as the cube root approximates perceptual linearity, akin to the lightness channel in the CIELAB color space [Sto03] (compare Figures 2(c) and 2(d)). Second, when visualizing big data the maximum density value in a binned plot may be orders of magnitude greater than the minimum non-zero density value. A naïve color ramp can render such bins invisible. To ensure the visibility of outliers, we set a minimum value of $\alpha = 0.15$ for non-zero densities, based on prior experimental results for luminance contrast [SB09, HB10a].

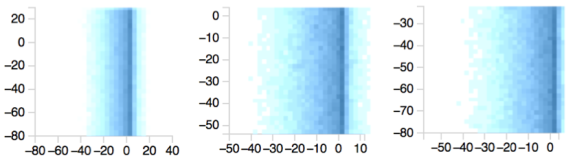


Figure 3: *Panning and zooming in a binned plot: initial view (left), zooming in (middle), panning to the lower-left (right).*

5. Enabling Interaction in Binned Plots

Interaction is essential to exploratory visual analysis [HS12], but big data imposes challenges to real-time response rates. While each binned chart type in the previous section visualizes one or two aggregated dimensions, more data resolution is needed to support interaction. Panning and zooming may require finer grained bins, as in Figure 3.

Brushing & linking, in which selections in one view highlight the corresponding data in other views, requires computing aggregates filtered by an initial data selection. These queries require partially de-aggregated data over which to compute the filtered aggregation (or "roll-up"). Sending these queries to a server incurs latency due to both processing and networking delays, and can easily exceed a 100 millisecond threshold for interactive response [CMN83]. Furthermore, multiple clients might overload the server.

In this section, we present our method for enabling real-time visual querying in *imMens*. We use brushing & linking over the Brightkite data set as a running example. The raw Brightkite data has five dimensions: User, Date, Time, Lat and Lon. Figure 4 shows four linked visualizations depicting binned data from different perspectives. The geographical heatmap (X, Y) is based on Mercator-projected Lon, Lat coordinates; the three histograms show monthly (Month), daily (Day) and hourly (Hour) checkin distributions derived from the Date and Time fields. The Jan bin is selected in the Month histogram. In response, corresponding data are highlighted in orange in the other histograms, and the geographic heatmap shows only checkins in the month of January.

5.1. Data Cube Queries to Support Interaction

Applying binned aggregation to X, Y, Month, Day and Hour, we form a 5-dimensional data cube (Figure 5(a)). The data cube contains the lowest level of data resolution in the linked visualizations. To perform brushing & linking from the Month histogram to the Day histogram, we can filter the data cube to only the rows with bin value 0 in the Month dimension (corresponding to January; highlighted in yellow in Figure 5(a)) and perform a roll-up by summing data along the Hour, X and Y dimensions. To zoom out, we can aggregate adjacent bins to compute a coarser-grained projection. Panning at the most zoomed-in level involves querying the bins visible in the current viewport.

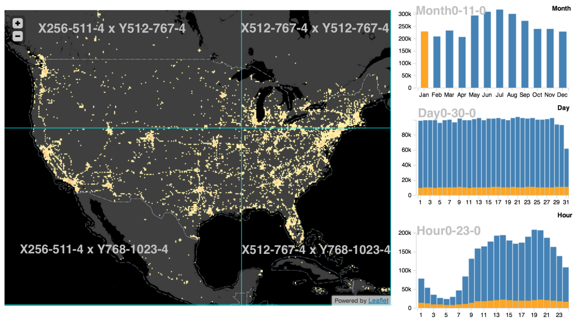


Figure 4: *Multiple coordinated views of Brightkite user checkins in North America. Cyan lines in the heatmap indicate data tile boundaries. Each visualization region is annotated by its backing data dimensions and indices.*

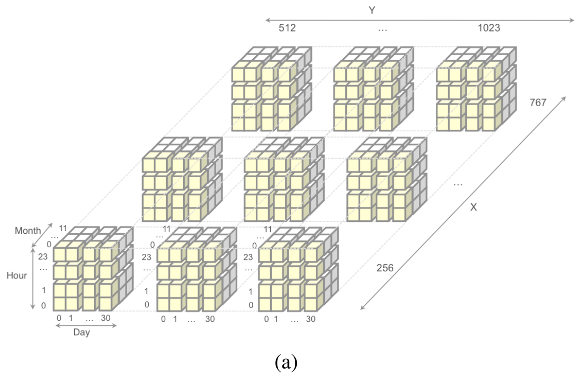
5.2. From Data Cubes to Multivariate Data Tiles

A full data cube is often too big to fit in memory and query in real-time. The size of a cube is $\prod_i b_i$, where b_i is the bin count for dimension i. As the number of dimensions or bins increases, the data cube size may become unwieldy. To address this issue, we decompose the full cube into sub-cubes with at most four dimensions.

The primary contributor to data cube size is the combinatorial explosion of multiple dimensions. However, for any pair of 1D or 2D binned plots, the maximum number of dimensions needed to support brushing & linking is four (e.g., between two binned scatterplots that do not share a dimension). As a result, we can safely decompose the full cube into a collection of smaller 3- or 4-dimensional projections. For example, four 3-dimensional cubes can cover all the possible brushing and linking scenarios shown in Figure 4: (X, Y, Hour), (X, Y, Day), (X, Y, Month), (Hour, Day, Month). If we assume a uniform bin count b, this decomposition reduces the total data record count from b^5 to $4b^3$; when b=50, the reduction is from 312.5M to 0.5M records.

After decomposition, individual sub-cubes may still be prohibitively large if the bin count is high. In some plots, we can treat the bin count as a free parameter, and adjust accordingly. For others – particularly geographic heatmaps – we may wish to zoom in to see fine-grained details, requiring an exponentially increasing number of bins across zoom levels. To handle large bin counts, we segment the bin ranges to form *multivariate data tiles*, as illustrated in Figure 5(b).

Data tiles are inspired by the notion of map tiles used in systems such as Google Maps and Hotmap [Fis07]. However, data tiles differ in two important ways. First, they provide data for dynamic visualization, not pre-rendered images. Second, they contain multidimensional data to support querying as well as rendering. Given a set of data tiles and a query selection (bin range), we can dynamically compute roll-up queries and render projected data. Figure 4 shows geographic tile boundaries highlighted in cyan. We label each



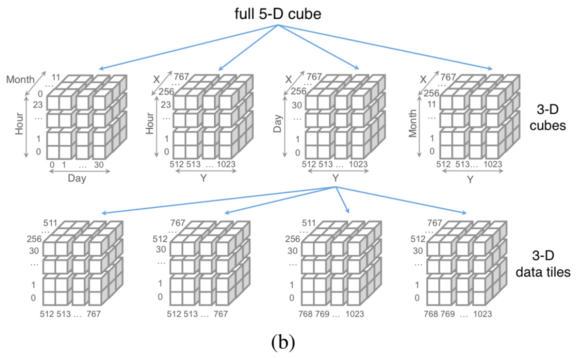


Figure 5: *(a) A 5-dimensional data cube of Brightkite check-ins; (b) Decomposing a full cube into sub-cubes and data tiles.*

tile dimension as $Db_s\text{-}b_e\text{-}z$, where D is the binned data dimension, b_s represents the starting bin index, b_e represents the ending bin index, and z represents the zoom level.

The Brightkite visualization in Figure 4 uses 13 data tiles: one tile representing the 3-dimensional projection of month, day and hour (i.e., Month0-11-0 × Day0-30-0 × Hour0-23-0), and twelve tiles containing 3-dimensional projections for all combinations of the four geographic segments and three histograms (e.g., X256-511-4 × Y512-767-4 × Month0-11-0). Multivariate data tiles are precomputed on a server and then loaded on demand to support client-side visualization.

Brushing & linking involves aggregating these data tiles. For example, when the user selects a region in the geographic heatmap, we need to highlight the corresponding checkin distributions in the three histograms. In the worst case, the selected geographic region covers bins in all four map tiles. To render the highlight in the Day histogram we need to roll-up the four data tiles containing the X × Y × Day dimensions and sum the results. Figure 6 shows this process. For interactions like panning and zooming, we dynamically fetch data tiles precomputed at different levels of binning resolution, similar to existing mapping services.

5.3. Dense vs. Sparse Data Tile Storage

Data tiles can use either sparse or dense packing schemes. A sparse representation stores indices and values only for non-zero bins (Figure 7(b)). A dense representation includes zero values, but can store all the data as a simple array if the bin counts for all dimensions are known (Figure 7(c)).[‡] If a data tile has many empty bins, a sparse representation can reduce storage costs. For example, a sparse packing is used in Profiler [KPP*12] for full data cubes of up to 5 dimensions.

[‡] We treat row indices as numbers in a mixed-radix number system [Knu06]. The row index in a k-dimensional data tile can be expressed as $V(k-1)_{R(k-1)}|V(K-2)_{R(K-2)}|\ldots|V(0)_{R(0)}$, where $V(k)$ is the value of the kth digit, and $R(k)$ is the radix of the kth digit.

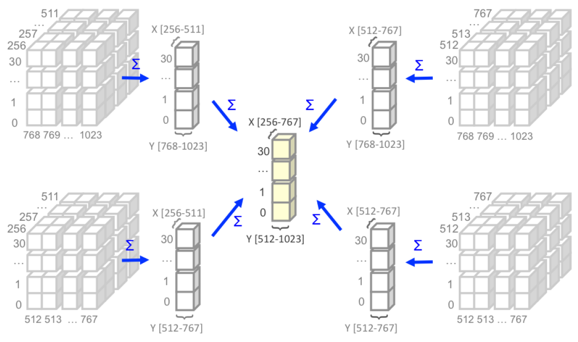
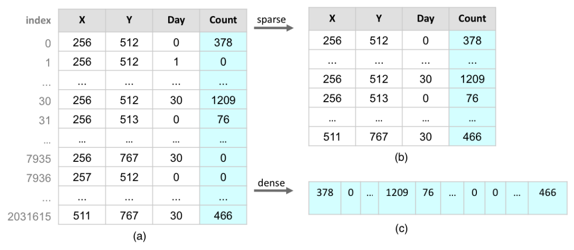


Figure 6: *Brushing & linking from the geographic heatmap to the* Day *histogram. We aggregate four data tiles along the X and Y dimensions and sum up the projections.*



index	X	Y	Day	Count
0	256	512	0	378
1	256	512	1	0
...	...	...	...	...
30	256	512	30	1209
31	256	513	0	76
...	...	...	...	...
7935	256	767	30	0
7936	257	512	0	0
...	...	...	...	...
2031615	511	767	30	466

(a)

sparse

X	Y	Day	Count
256	512	0	378
...	...	...	...
256	512	30	1209
256	513	0	76
...	...	...	...
511	767	30	466

(b)

dense

378	0	...	1209	76	...	0	0	...	466

(c)

Figure 7: *Sparse and dense representations of a data tile.*

However, as the number of data records increases, the density of the data typically also increases. Once the proportion of non-zero bins exceeds a threshold (20% for 4D tiles, 25% for 3D tiles), a dense representation is more efficient because we can omit bin indices. In *imMens* we use dense tiles to exploit these space savings, safeguard worst-case performance, and simplify parallel query processing.

5.4. Parallel Query Processing

A dense representation scheme supports simple, efficient parallel processing when aggregating data tiles. Dense tiles provide a consistent indexing scheme that enables direct lookup of any multidimensional bin using a predictable integer index. As a result, aggregation queries can be parallelized easily. For each output bin (summed value), we can use a simple loop that accesses only the bins needed for that summation. These computations can be run in parallel in a shared memory environment without any conflicts, resulting in faster query performance than a sequential scan.

```
Input: $c_2$, $c_3$, $R_2$, $R_3$, $T$
Output: sum $v$ at index $i$ of $d_1$'s projection
$v = 0$;
foreach $j \in R_2$ do
    foreach $k \in R_3$ do
        $v += T[i * c_2 * c_3 + j * c_3 + k]$;
    end
end
```
Algorithm 1: Data tile roll-up for a projection index.

Consider a 3D data tile T with dimensions (d_1, d_2, d_3), with respective bin counts (c_1, c_2, c_3). If users brush a 2D binned plot of d_2 and d_3 to select ranges R_2 and R_3, we can compute the summed value v at index i of the d_1 projection using Algorithm 1. With this simple roll-up procedure, we can run the algorithm in parallel for all c_1 indices.

We note that our summation scheme can be further optimized by the use of summed area tables. If data tiles instead store *cumulative* densities over index ranges, the summation for an output bin can be computed with a constant number of lookups. However, in practice we find that our simpler scheme provides good performance, while the use of cumulative densities exacerbates issues of arithmetic overflow.

6. Implementation Details

We implement our visualization and interaction techniques in *imMens*, a browser-based system for interactive visual exploration of big data. Given a visualization definition, *imMens* loads data tiles from a server and provides an interactive multiple view display of binned plots within standards-compliant web browsers. We use WebGL, a JavaScript variant of the OpenGL ES 2.0 specification, to perform GPU-based computation and render visualizations to HTML5 canvas elements. We also use Leaflet [Lea] to display map image tiles and D3 [BOH11] to render axes and labels. We now describe our scheme for storing data tiles as image textures and present our querying and rendering pipeline.

6.1. Storing Data Tiles as Image Files

By packing data tiles in an image format, we can directly bind them to the WebGL context as textures and take advantage of existing browser caching facilities. Packing data tiles into images facilitates efficient storage and transfer of tiles and makes the data accessible for parallel processing on the GPU. For each data tile integer value v, we apply the scheme in Equation 2 to pack it into the 8-bit RGBA channels of a pixel. If the value is smaller than 2^{31}, we can preserve complete information with no precision loss. We set the highest-order bit in the alpha channel to 1 due to image format constraints; we use the PNG file format for its lossless compression and network portability. The PNG specification instructs that fully transparent pixels be assigned the same RGB values (zero) for better compression. Storing zero values in alpha channels thus results in data loss.

$$\begin{cases} A = 0x80 \mid ((0xFF000000 \ \& \ v) >> 24) \\ R = (0x00FF0000 \ \& \ v) >> 16 \\ G = (0x0000FF00 \ \& \ v) >> 8 \\ B = (0x000000FF \ \& \ v) \end{cases} \tag{2}$$

The maximum supported texture size imposes a constraint on the amount of data we can fit into a data tile. The maximum texture size can vary by graphics card from $1{,}024^2$ to $16{,}384^2$ pixels. A texture of size $4{,}096^2$ (a commonly supported size) has 16 million pixels. With this size, the finest possible resolution is 256 bins per dimension for a 3D data tile and 64 bins per dimension for a 4D tile. Assuming 50 bins per dimension, the average file size is 30KB for a 3D PNG tile and 150KB for a 4D tile. The 13 PNG-encoded data tiles used in the Brightkite example require 352KB.

Applying more aggressive packing schemes by quantizing data tile values into fewer pixel channels is more space-efficient, but comes at a cost of precision loss. We have experimented with packing each value into one of the RGBA channels, quantizing the value to a single byte. Benchmarks show per-bin root mean square errors ranging from 0.003 to 0.18, where an error of 0.01 maps to a perceptual difference of one pixel in a 100-pixel-tall histogram. We leave more efficient packing schemes that limit data loss to future work.

6.2. Parallel Query and Render via Shader Programs

WebGL uses *shader programs* that run on the GPU: *vertex shaders* transform 3D geometry, *fragment shaders* compute pixel colors. In *imMens*, we use a default vertex shader and implement query processing and rendering as fragment shaders. We use a two-stage process, illustrated in Figure 8.

The query fragment shader reads from data tiles bound as textures and performs a roll-up (§5.4), providing the data to visualize. The shader program computes the sum for a single output bin, and writes the result to an offscreen frame buffer object (FBO). This program is run in parallel for all bins in the resulting plot. With multiple plots, we need to compute roll-ups for each. To do so, we dynamically change data tiles during the execution of the shader, and store all the aggregates in the same FBO. For queries spanning multiple data tiles (as in Figure 6), we bind the required data tiles and perform multiple roll-ups in a loop.

The render fragment shader binds the FBO as a texture and renders the plots. For each pixel in a plot, the shader de-

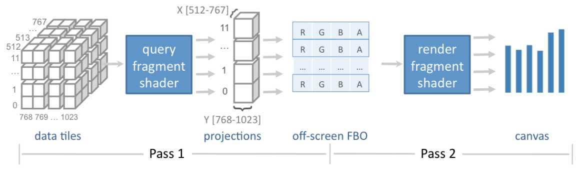


Figure 8: *A two-pass approach for parallel data querying and rendering using WebGL fragment shaders.*

termines which bin that pixel belongs to and assigns it an appropriate color. Consider the case of rendering a histogram. The query fragment shader first computes the necessary data and writes it as a 1D array in the FBO. To draw a histogram with an on-screen bin width of c pixels, the render fragment shader computes the corresponding bin index i for each pixel coordinate (x, y). If the histogram starts at x-coordinate 0, then $i = x/c$. By reading the data value from the FBO, the shader determines the height h of the bar at bin index i. Finally the shader makes a binary choice to assign the pixel a color: if $y > h$, the color is left as background; otherwise the pixel is colored as part of the histogram bar.

Our two-pass approach carries two advantages. First, using the FBO we can eliminate redundant computation for pixels corresponding to the same data bin. Second, by examining the FBO we can identify the maximum bin value. The FBO enables us to render bar heights and cell color values accurately by normalizing with this maximum value.

7. Performance Benchmarks

To evaluate the scalability of *imMens*, we measure the system frame rate during brushing & linking. For a given visualization configuration (consisting of both 1D and 2D plots), we programmatically brush the bins in each 1D histogram. Brushing only the 1D histograms and not the 2D plots provides a conservative estimate of performance, as brushing 2D bins results in more selective queries and hence faster processing. For each brushed bin, we record the time taken to both compute a roll-up query and re-render the display. We average these update times over multiple runs.

We first conducted benchmarks using two real-world data sets: 4 million user checkins on Brightkite [CML11] and 118 million records regarding on-time performance of U.S. domestic flights [BTS]. We visualize the Brightkite data set with a geographic heatmap and three temporal histograms as shown in Figure 4. For the flight delay data set, we use a similar visualization configuration, including a binned scatter plot of departure delay versus arrival delay and three histograms showing the distributions of delayed flights by airline, year and day of week.

To compare *imMens* to the existing state-of-the-art, we replicate the benchmarks used for Profiler [KPP*12] and compare the two systems. We generate a total of 60 test data sets by varying three parameters: the per-dimension bin

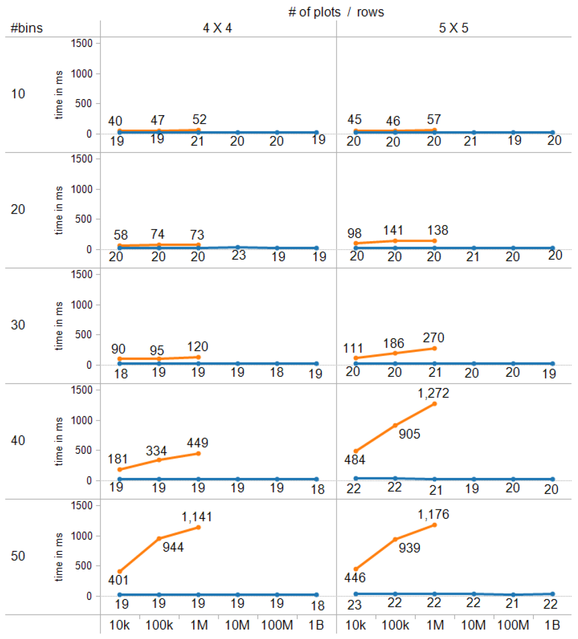


Figure 9: *Average time (ms) for imMens (blue) and Profiler (orange) to update frames during brushing & linking.*

Data Set	Brightkite	Flight Delays	SPLOM
Size	4M	118M	1B
Bins	Month (12)	Carrier (28)	Dim. A (50)
	Day (31)	Year (20)	Dim. B (50)
	Hour (24)	Day of Week (7)	Dim. C (50)
	X (256)	Dep. Delay (174)	Dim. D (50)
	Y (256)	Arr. Delay (174)	Dim. E (50)
Data Tiles	13	4	10
Time	*17.76 ms*	*16.56 ms*	*20.47 ms*

Table 2: *Benchmark results for three data sets.*

count (10, 20, 30, 40, 50), the number of dimensions (4, 5), and the number of input records (10K, 100K, 1M, 10M, 100M, 1B). Each generated data set contains five dimensions consisting of random samples drawn from normal distributions. Three of these distributions are independent; the other two distributions are linearly derived from the first. We visualize the data as a scatter plot matrix (SPLOM) with univariate histograms along the diagonal, and programmatically brush the bins in each of these histograms.

We ran the benchmarks in Google Chrome v.23.0.1271.95 on a quad-core 2.3 GHz MacBook Pro (OS X 10.8.2) with per-core 256K L2 caches, shared 6MB L3 cache and 8GB RAM. The test machine has a PCI Express NVIDIA GeForce GT 650M graphics card with 1024MB video RAM. Table 2 summarizes the benchmark results, and Figure 9 plots the detailed results for the comparative benchmark across all 60 test data sets. Due to memory limits, Profiler can not visualize data with 10M or more records. *imMens* sustains an average 50 fps across all conditions. The performance is invariant to the original data set size.

8. Conclusion and Future Work

In this paper, we contribute methods for real-time visual querying of big data. We first provide a review of data reduction methods to support scalable visual summaries, and examine the design space of binned plots for large multidimensional data. To enable real-time interaction, we integrate *multivariate data tiles* and parallel processing.

We implement these methods in *imMens*, a browser-based system using WebGL. We use WebGL because it is currently the only standardized way to access GPU processing in a web browser. Future browser-based GPGPU or parallel computing (multi-core) features will permit alternative approaches. Our general approach of generating multivariate data tiles and leveraging parallel query processing is also directly applicable in standard (non-web) desktop contexts.

With a sustained performance of 50 frames-per-second brushing and linking, *imMens* enables data analysts to interactively examine summaries of billions of data records in real-time. To our knowledge, it is the first system to enable real-time brushing with data sets this large. *imMens* is available as open source software at `https://github.com/StanfordHCI/imMens`.

One limitation of our approach is the lack of support for ad-hoc compound brushing of more than four dimensions. For example, in the Brightkite visualizations in Figure 4, users may want to explore the geographic distribution of checkins across 24 hours in a particular day. To do so, they may select both a specific month and day, then perform brushing and linking between the Hour histogram and the geographic heatmap. Currently such compound queries require computing 5-dimensional data tiles – an untenable increase in tile size. However, once queries become highly selective, the decrease in data size due to filtering may permit more responsive server-side querying and dynamic tile generation.

In future work, we plan to automatically optimize client-side visualization specifications (including requested dimensions and bin counts) based on available resources. For example, the maximum size and number of textures supported by WebGL varies across machines and graphics cards. Given a client's resource constraints, reducing the number of bins or dimensions may preserve quality of service.

A critical issue in providing a seamless user experience is supporting natural transitions between levels of detail and visualization configurations. Ideally all possible data tiles are pre-computed and stored on a server. The client might anticipate possible user interactions and prefetch data tiles to reduce latency [DBC*13]. This strategy may not be realistic for high-dimensional data sets as preprocessing all possible data tiles can be computationally expensive. Graceful degradation is thus important when data tiles are yet to be computed. One possibility is to adopt approaches such as BlinkDB [APM*12] and online aggregation [HHW97] to rapidly provide approximate tiles on the fly based on data samples. The resulting visualizations will have a coarser resolution and can be enhanced with more details after the full data tiles are computed.

Another future step is to provide an interface for visualization construction. Currently *imMens* uses hand-coded specifications. In contrast, Tableau's drag-and-drop interface [Tab] supports dynamic construction of multiscale visualizations. We plan to combine a similar design with the visualization and interaction facilities of *imMens*.

Finally, by focusing on 2-dimensional binned plots of multi-dimensional data, we do not address scalability issues in higher-dimensional plots such as parallel coordinates or for complex data types such as networks. These remain as interesting challenges for future research.

Acknowledgments

This work was supported by the Intel Science and Technology Center for Big Data and the DARPA XDATA program. We thank Leilani Battle, Justin DeBrabant and Michael Stonebraker for their helpful insights and conversations.

References

[APM*12] AGARWAL S., PANDA A., MOZAFARI B., MADDEN S., STOICA I.: BlinkDB: queries with bounded errors and bounded response times on very large data. *arXiv:1203.5485* (Mar. 2012). 3, 9

[AS94] AHLBERG C., SHNEIDERMAN B.: Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of CHI* (1994), pp. 313–317. 1, 2

[BBSBA08] BAGROW J. P., BOLLT E. M., SKUFCA J. D., BEN-AVRAHAM D.: Portraits of complex networks. *EPL (Europhysics Letters) 81*, 6 (2008), 68004. 2, 3

[BCS96] BECKER R. A., CLEVELAND W. S., SHYU M. J.: The visual design and control of trellis display. *Journal of Computational and Graphical Statistics 5*, 2 (1996), 123–155. 4

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3: Data-driven documents. *IEEE TVCG 17*, 12 (2011), 2301–2309. 7

[BS06] BERTINI E., SANTUCCI G.: Give chance a chance: modeling density to enhance scatter plot quality through random data sampling. *Information Visualization 5*, 2 (2006), 95–110. 2

[BTS] Bureau of transportation statistics. http://www.transtats.bts.gov/. 8

[CLNL87] CARR D. B., LITTLEFIELD R. J., NICHOLSON W. L., LITTLEFIELD J. S.: Scatterplot matrix techniques for large n. *Journal of the American Statistical Association* (1987), 424–436. 1, 2, 4

[CML11] CHO E., MYERS S. A., LESKOVEC J.: Friendship and mobility: user movement in location-based social networks. In *Proceedings of SIGKDD* (2011), pp. 1082–1090. 3, 8

[CMN83] CARD S. K., MORAN T. P., NEWELL A.: *The Psychology of Human-Computer Interaction*. Psychology Press, 1983. 5

[CXGH08] CHAN S.-M., XIAO L., GERTH J., HANRAHAN P.: Maintaining interactivity while exploring massive time series. In *Proceedings of VAST* (Oct. 2008), pp. 59 –66. 2

[DBC*13] DEBRABANT J., BATTLE L., CETINTEMEL U., ZDONIK S., STONEBRAKER M.: Techniques for visualizing massive data sets. In *New England Database Summit* (Feb 2013). 2, 9

[DRW03] DOSHI P. R., RUNDENSTEINER E. A., WARD M. O.: Prefetching for visual data exploration. In *Proceedings of the Eighth International Conference on Database Systems for Advanced Applications* (2003), DASFAA '03, pp. 195–203. 2

[DSLG*12] DAS SARMA A., LEE H., GONZALEZ H., MADHAVAN J., HALEVY A.: Efficient spatial sampling of large geographical tables. In *Proceedings of SIGMOD* (2012), ACM, pp. 193–204. 1, 2, 3

[EF10] ELMQVIST N., FEKETE J. D.: Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE TVCG 16*, 3 (2010), 439–454. 2, 3

[Fis07] FISHER D.: Hotmap: Looking at geographic attention. *IEEE TVCG 13*, 6 (2007), 1184–1191. 1, 2, 5

[FP02] FEKETE J.-D., PLAISANT C.: Interactive information visualization of a million items. In *Proceedings of InfoVis* (2002), pp. 117–124. 1, 2

[FPDs12] FISHER D., POPOV I., DRUCKER S., SCHRAEFEL M.: Trust me, i'm partially right: Incremental visualization lets analysts explore large datasets faster. In *Proceedings of CHI* (2012), pp. 1673–1682. 3

[GM99] GUPTA A., MUMICK I. S. (Eds.): *Materialized views: techniques, implementations, and applications*. MIT Press, Cambridge, MA, USA, 1999. 1

[HB10a] HEER J., BOSTOCK M.: Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of CHI* (2010), pp. 203–212. 4

[HB10b] HEER J., BOSTOCK M.: Declarative language design for interactive visualization. *IEEE TVCG 16*, 6 (2010), 1149–1156. 2

[HDS*10] HAO M. C., DAYAL U., SHARMA R. K., KEIM D. A., JANETZKO H.: Visual analytics of large multidimensional data using variable binned scatter plots. In *Proceedings of SPIE* (2010), Bibliothek der Universitat Konstanz. 2

[HHW97] HELLERSTEIN J. M., HAAS P. J., WANG H. J.: Online aggregation. *ACM SIGMOD Record 26*, 2 (1997), 171–182. 3, 9

[HS12] HEER J., SHNEIDERMAN B.: Interactive dynamics for visual analysis. *Queue 10*, 2 (2012), 30. 5

[JLJC05] JOHANSSON J., LJUNG P., JERN M., COOPER M.: Revealing structure within clustered parallel coordinates displays. In *Proceedings of InfoVis* (Oct. 2005), pp. 125 – 132. 2

[JS98] JERDING D. F., STASKO J. T.: The information mural: A technique for displaying and navigating large information spaces. *IEEE TVCG 4*, 3 (1998), 257–271. 1, 2

[Kei00] KEIM D. A.: Designing pixel-oriented visualization techniques: Theory and applications. *IEEE TVCG 6*, 1 (2000), 59–78. 2

[KHD*10] KEIM D. A., HAO M. C., DAYAL U., JANETZKO H., BAK P.: Generalized scatter plots. *Information Visualization 9*, 4 (2010), 301–311. 2

[KMSH12] KAIRAM S., MACLEAN D., SAVVA M., HEER J.: GraphPrism: compact visualization of network structure. In *Proceedings of AVI* (2012), pp. 498–505. 2, 3

[Knu06] KNUTH D. E.: *The art of computer programming*, vol. 2. Addison-Wesley, 2006. 6

[KPHH12] KANDEL S., PAEPCKE A., HELLERSTEIN J. M., HEER J.: Enterprise data analysis and visualization: An interview study. *IEEE TVCG 18*, 12 (2012), 2917–2926. 1

[KPP*12] KANDEL S., PARIKH R., PAEPCKE A., HELLERSTEIN J. M., HEER J.: Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of AVI* (2012), pp. 547–554. 1, 2, 6, 8

[Lea] Leaflet. http://leafletjs.com/. 7

[Loh09] LOHR S. L.: *Sampling: Design and Analysis*, 2 ed. Duxbury Press, Dec. 2009. 2

[ME09] MCDONNEL B., ELMQVIST N.: Towards utilizing GPUs in information visualization: A model and implementation of image-space operations. *IEEE TVCG 15*, 6 (Nov. 2009), 1105—1112. 2

[MTS91] MIHALISIN T., TIMLIN J., SCHWEGLER J.: Visualizing multivariate functions, data, and distributions. *IEEE Comput. Graph. Appl. 11*, 3 (May 1991), 28–35. 1

[NH06] NOVOTNY M., HAUSER H.: Outlier-preserving focus+context visualization in parallel coordinates. *IEEE TVCG 12*, 5 (2006), 893–900. 3

[PTMB09] PIRINGER H., TOMINSKI C., MUIGG P., BERGER W.: A multi-threading architecture to support interactive visual exploration. *IEEE TVCG 15*, 6 (Nov. 2009), 1113–1120. 2

[PWR04] PENG W., WARD M. O., RUNDENSTEINER E. A.: Clutter reduction in multi-dimensional data visualization using dimension reordering. In *Proceedings of InfoVis* (2004), pp. 89–96. 2

[Raf05] RAFIEI D.: Effectively visualizing large networks through sampling. In *Proceedings of VIS* (2005), pp. 375–382. 2

[RWC*08] RÜBEL O., WU K., CHILDS H., MEREDITH J., GEDDES C. G., CORMIER-MICHEL E., AHERN S., WEBER G. H., MESSMER P., HAGEN H., HAMANN B., BETHEL E. W.: High performance multivariate visual data exploration for extremely large data. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing* (2008), IEEE Press, pp. 51–62. 2

[SB09] STONE M., BARTRAM L.: Alpha, contrast and the perception of visual metadata. In *Color Imaging Conf* (2009). 4

[Sco79] SCOTT D. W.: On optimal and data-based histograms. *Biometrika 66*, 3 (1979), 605–610. 4

[Sco92] SCOTT D. W.: *Multivariate Density Estimation: Theory, Practice, and Visualization*, 1 ed. Wiley, Aug. 1992. 4

[Spo] TIBCO spotfire. http://spotfire.tibco.com/. 2

[Sto03] STONE M.: *A Field Guide to Digital Color*, 1st ed. A K Peters/CRC Press, Aug. 2003. 4

[Stu26] STURGES H. A.: The choice of a class interval. *Journal of the American Statistical Association 21*, 153 (1926), 65–66. 4

[Tab] Tableau software. http://www.tableausoftware.com. 2, 9

[TGC03] TRUTSCHL M., GRINSTEIN G., CVEK U.: Intelligently resolving point occlusion. In *Proceedings of InfoVis* (Oct. 2003), pp. 131–136. 2

[UTH06] UNWIN A., THEUS M., HOFMANN H.: *Graphics of Large Datasets: Visualizing a Million*, 1 ed. Springer, July 2006. 1, 2, 4

[ZBDS12] ZINSMAIER M., BRANDES U., DEUSSEN O., STROBELT H.: Interactive level-of-detail rendering of large graphs. *IEEE TVCG 18*, 12 (Dec. 2012), 2486 –2495. 2