INTERACTIVE EXPLORATION AND VISUAL ANALYTICS FOR LARGE SPATIOTEMPORAL DATA USING APPROXIMATE QUERY PROCESSING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Guizhen Wang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. David S. Ebert, Chair

School of Electrical and Computer Engineering

Dr. Niklas Elmqvist

College of Information Studies, University of Maryland

Dr. Alexander J. Quinn

School of Electrical and Computer Engineering

Dr. Edward J. Delp

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of the School Graduate Program

To my parents and siblings

ACKNOWLEDGMENTS

Through the completion of my dissertation, I have received tremendous help. I want to express my deepest appreciation to my advisor, Dr. David Ebert, for his immense support of my study and research. His encouragement, guidance, and patience enabled me to identify cutting-edge research problems and develop this work. It has been a great honor to be his Ph.D. student. Furthermore, I would like to extend my sincere thanks to the rest of my thesis committee: Prof. Niklas Elmqvist, Prof. Alex Quinn, and Prof. Edward Delp, for their insightful comments on my dissertation. I am also grateful to Prof. Walid Aref for his valuable advice and feedback on my research. I would like to thank all members of VACCINE lab, José Florencio de Queiroz Neto, Jieqiong Zhao, Calvin Yau, Jinging Guo, Abish Malik, Jiawei Zhang, Luke Snyder, and Audrey Reinert, for their support in research and life. In addition, I am deeply indebted to my parents and siblings for their unconditional love and supports. Finally, I would like to thank my friends, Xinxin Liu, Xiangning Huang, Leyu Zhang, Ting Zhang, Cancan Cong, Ximing Zhang, Dihong Gong, Wei Deng, Shuqi Zhou, and Junwei Zhang. They were of great support in deliberating over my problems and enriching my life outside of my research.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABSTRACT

Wang, Guizhen Ph.D., Purdue University, August 2020. Interactive exploration and visual analytics for large spatiotemporal data using approximate query processing. Major Professor: David S. Ebert.

Approximate query processing (AQP) provides fewer representative samples to approximate large amounts of data. Processing these smaller data subsets enables visualization systems to provide end-users with real-time responses. However, challenges arise for real-world users in adopting AQP-based visualization systems, e.g., the absence of AQP modules in mainstream commercial databases, erroneous estimations caused by sampling bias, and end-user uncertainty when interpreting approximate query results. In this dissertation, we present an AQP-centered technique for enabling interactive visual analytics for large amounts of spatiotemporal data under the aforementioned challenges. First, we design, implement and evaluate a client-based visual analytics framework that progressively acquires spatiotemporal data from an AQP-absence server-side to client-based visualization systems so that interactive data exploration can be maintained on a client machine with modest computational power. Second, we design, implement, and evaluate an online sampling approach that selects samples of large spatiotemporal data in an unbiased manner and accordingly improves the accuracy of the associated estimates. Last, we design, implement and evaluate a difference assessment approach that compares approximate and exact spatial heatmap visualizations in terms of human perception. As such, information changes perceptible by users are well represented, and users can evaluate the reliability of approximate answers more easily. Our results show the superior performance of our proposed AQP-centered technique in terms of speed, accuracy, and user trust, as compared to a baseline of state-of-the-art solutions.

1. INTRODUCTION

Visual analytics systems enable decision-makers to interactively explore data and then quickly and easily draw insights from digital data. In general, visualizing large amounts of spatiotemporal data requires certain techniques in order to maintain quick responses, e.g., computing in parallel and prefetching data in anticipation of user queries. One widely-used technique is sampling-based Approximate Query Processing (AQP) [1, 2], which allows visual analytics systems to process few data samples while responding to users quickly with approximate but representative answers. End users are willing to accept approximate analytical results for decision-making [3–5]. However, challenges exist for visual analytics systems in employing AQP techniques, including AQP absences caused by IT architectural constraints, sampling bias, and difficulty for users in choosing trustworthy answers. Each challenge requires special treatment for AQP. As such, in this dissertation, we propose AQP-centered techniques to support visual analytics for large-volume spatiotemporal data. We identify three main challenges preventing the implementation of AQP-based visual analytics systems, including (a) enabling AQP under architectural constraints, (b) avoiding bias in the online sampling process, and (c) facilitating users to evaluate AQP results. We address these challenges and report on our progress in this document.

1.1 Sampling Spatiotemporal Data under Architectural Constraints

Approximate queries of large data often require modifications of traditional data management systems. Traditional relational databases [6,7] usually retrieve full query results, while AQP selects a subset of data in order to approximate the entire data in a representative manner. However, as an emerging technique, AQP has not been widely implemented in mainstream commercial databases. Therefore, enabling AQP

operations for visual analytics requires organizations to replace or heavily modify their existing database systems.

Unfortunately, in many businesses and local governmental organizations, security practices or budgetary concerns may prevent the deployment of such AQP-based solutions or the transfer of data to external servers that do provide AQP [8, 9], and therefore, put the burden on the client machines to handle the AQP workload.

To address this challenge, we propose, implement, and evaluate a client-based visual analytics framework for large-scale spatiotemporal data organization and sampling under the aforementioned architectural constraints. The proposed framework designs a novel sampling method that progressively acquires data from external servers; meanwhile, it leverages affordable computational resources in a client machine to visualize data progressively. The presented approach enables end-users to interactively explore large amounts of spatiotemporal data on the client-side.

1.2 Conducting Online Sampling of Large Spatiotemporal Data in an Unbiased manner

Online sampling-supported visual analytics is becoming increasingly important, as it allows users to explore large data sets at interactive rates (e.g., 500ms [10]). However, the state-of-the-art online spatiotemporal sampling technique [1] has primarily focused on reducing computational latencies, and has not fully investigated sampling bias. Biased sampling approaches select data points with unequal probabilities, making their samples erroneously represent the whole queried data [11]. Likewise, statistics derived from biased samples can lead end users to incorrect interpretations.

To address this challenge, we propose, implement, and evaluate a novel online sampling algorithm in order to select large spatiotemporal data in an unbiased manner. Our approach makes sure that each data point satisfying the user-defined multidimensional query specification has the same probability of being selected. Our proposed sampling approach is suitable for analytics tasks that focus on spatiotemporal data

aggregates. Our experiments confirm the superiority of our method over state-of-the-art spatial online sampling techniques, demonstrating that spatial distributions of data samples generated by our approach are at least 50% more accurate at a 5% sample size.

1.3 Creating a Perception-Aware Difference Assessment for Incremental Visualization of Spatial Heatmap

Incremental data visualization uses AQP to sample data progressively. As data samples are continuously loaded into the system, the accuracies of approximate visualizations increase over time. Controlling computational time and finding answers with accuracies that meet the requirements for given analytic tasks relies on users. In general, users use statistical measures (including confidence intervals [3, 12], distribution+precision [13] and bootstrapping strategies [14]) to evaluate the accuracy of approximate results. However, these statistical measures handle visual elements in a way that is inconsistent with the human visual system [15]. Therefore, end-users have difficulty connecting the insight gained from the visualizations with the accuracies indicated by these statistical measures [4].

To address this challenge, we propose, implement, and evaluate a novel approach to assessing differences between approximate and exact spatial heatmaps in terms of human perception. Following the human perception mechanism, our approach identifies visual elements users perceive to be discernible. Using these identified elements to adjust the aforementioned statistical measures can measure the accuracy of approximate answers in a way that is consistent with human perception. Therefore, measures generated with the perception-aware approach are easier for users to interpret in a context where spatial heatmaps incrementally improve.

1.4 Thesis Statement

The thesis statement of this dissertation is as follows:

Sampling-based approximate query processing approaches and approximate answer evaluation methodologies enable end-users to explore large amounts of spatiotemporal data at interactive rates and make effective decisions.

Specifically, this dissertation makes the following contributions:

1. A client-based AQP framework supporting the interactive exploration of large amounts of spatiotemporal data in a computational environment where policies or budgetary concerns restrict a server to be a data provider, and a client machine lacks sufficient computational resources to process all the data. The proposed framework enables an average client machine to progressively sample data from a data server, to execute sampling-based incremental visual analytics, and to respond to users at interactive rates.

2. An online sampling algorithm enabling visualization systems to query multidimensional spatiotemporal data in an unbiased manner. Compared to existing methods, our approach achieves 50% more accuracy to approximate data distributions in the spatial dimension and enables approximate visualizations to present closer visual appearances to their exact counterparts.

3. A perception-aware difference assessment comparing approximate spatial heatmaps with exact ones in terms of human perception. Our approach enhances preattentive elements whose differences can likely be perceived by humans and decreases the role of indiscernible elements in the accuracy quantification. As such, the accuracy derived using our measure reflects the visual differences perceived by users; as a result, users are expected to more easily understand the accuracy of approximate heatmaps and to be less uncertain when choosing heatmaps with satisfactory accuracies for decision-making.

1.5 Outline

This document is organized into six chapters. Chapter 2 discusses the background and related work about interactive exploration of large amounts of data. The next three chapters each address one of the three challenges discussed in this introduction. Specifically, Chapter 3 presents our client-based visual analytics framework for supporting approximate queries. Chapter 4 presents our online sampling algorithm that selects large spatiotemporal data in an unbiased manner. Chapter 5 outlines our perception-aware difference assessment for the incremental visualization of spatial heatmaps. Finally, Chapter 6 summarizes the contributions of this dissertation and future research work.

2. BACKGROUND AND RELATED WORK

Visual analytics (VA) enables users to effectively understand, reason about, and make decisions with data [16]. A key factor of this effective analytic manner is the interactivity that ensures VA systems respond to users in a timely manner so that users can conduct analyses at the same pace with their mental activities [17]. However, huge amounts of data postpone slow down the responsiveness of these systems. As such, interactive exploration and visual analytics become intractable. Focusing on this challenge, this dissertation presents AQP-centered techniques that enable visualization systems to process data in a scalable manner, maintain interactive data exploration, and achieve effective decision-making.

In this chapter, we summarize state-of-the-art general techniques for interactive exploration of large spatiotemporal data, and further expound on AQP-based computational and visualization approaches. Section 2.1 briefs spatiotemporal data visualization and analytics, Section 2.2 overviews methodologies enabling VA systems to reduce data processing time and maintain interactivity, Section 2.3 expands one of the methodologies, sampling-based AQP techniques, Section 2.4 summarizes incremental visualization and progressive visual analytics (PVA) that help users adapt to AQP-leveraged incremental, yet approximate analytic paradigm, and Section 2.5 shows incremental visualization approaches that encode information from the perspective of human perception.

2.1 Spatiotemporal Data Visualization and Visual Analytics

Visualization and visual analytics provide intuitive graphical representations and convenient interaction to ensure effective data exploration and analyses. The adage "One picture is worth a thousand words" points out the importance of visualiza-

tion. Visual representations use graphical layouts to highlight information that is implicit and hidden in sequential sentences, which reduces human mental inference workloads [18]. In addition, interactions allow end users to forage information on demand. For example, VA systems [19,20] widely adopt the *information seeking mantra* of "overview first, zoom, filter, and details on demand" [21] for users to explore information. A broad range of fields, e.g., public safety [22] and social media [23], have adopted visual analytics for their analyses.

Spatiotemporal visualization techniques characterize spatial and temporal data patterns, trends, and anomaly analyses in the geographical domain. Data aggregation in the geospace is a common analytic task [20, 22, 24]. A prevalent visualization approach representing spatial aggregation, the heatmap [25], uses intuitive colors to encode density variations. For example, Hotmap [26] uses heatmaps to visualize spatial locations that are heavily explored by users so that engineers can forecast user access activities in the spatial dimension and load associated map tiles in advance. The contour plot [27] outlines regions whose data volumes are above the normal levels, so that users can easily identify spatial concentration. Spatial analyses are usually conducted on hierarchical spatial aggregates as well, e.g., nations, states, counties, and states. A choropleth map [28] is a common approach displaying information in terms of such spatial aggregates. Topomap [29] preserves the context of textual information in multi-scale spatial aggregates. Treemap [30] visualizes a hierarchical organization of information in a compact way, which is particularly effective on a limited screen. Unlike discrete points on maps, object movements has one more visualization need, movement sequences. A suite of visualization tools (e.g. space-time cubes and voronoi-based trajectory movement summarizations) center on such sequence patterns [31–34].

In addition to graphical information representation, interactivity is crucial to visualization and visual analytics. Users need to explore data at interactive rates (e.g., less than one second) so that their mental activities are not interrupted [17]. A survey shows that if a VA system's response time increases by 500ms, end users significantly

decrease their analytical activities by doing less interactions and reducing dataset coverage [10]. Prolonged data processing prevents users from detecting errors at earlier stages of their analyses [3]. As a result, an analyst's enthusiasm for data exploration is abated by the time cost, which makes them reluctant to conduct further analyses. However, in the big data era, data exponentially increase [35], which significantly prolongs data processing latency. This dissertation focuses on the prolonged latency challenge, and presents AQP-centric approaches for VA systems to ensure exploration of large-volume spatiotemporal data at interactive rates.

2.2 Computational Latency Reduction for Interactive Visualization

This section overviews various methodologies for VA systems to process large data and response to users rapidly.

The Client-Server architecture is the most popular way for visualization systems to handle large data, where remote high-performance servers quickly conduct heavy computation and commodity machines on the client side focus on light-weight computation (e.g., interaction). Scientific Visualization needs to handle large/big textured polygons. Thus, remote high-end servers process these data for intensive rendering tasks and transfer rendered signals to the client side for displays [36–39]. The Client-Server-based remote visualization solutions particularly works for front-end displays conducted on mobile devices [40].

Specific to machines that process large data, parallelizing the execution workflow can dramatically reduce computational latencies. MapReduce [41] is a scalable programming model that maps a large volume of data records to a cluster of commodity machines, processes data chunks in parallel and merges the computed results for the final output. Unlike MapReduce storing intermediate results on disk, Spark [42] further reduces disk I/O latency by exporting results into memory. Following the same parallelism, SpatialHadoop is extended in terms of analytical tasks in the geospatial domain [43], and VisReduce [44] is tailored for visualization computation.

Data cubes aggregate larger dataset into a smaller knowledge graph [45]. Retrieving answers from such a compact graph is efficient for aggregate queries (e.g., imMens [46], Nanocube [47], Hashedcubes [48], TOPKUBE [49], SmartCube [50]). But these approaches are designed for predefined queries, not suitable for new ones.

Prefetching can hide computational latencies by predicting user behaviors and fetching data in advance. A prefetching technique [51] mitigates the discrepancy between a higher CPU computational rate and a lower memory access rate in texture mapping. ATLAS [52] predicts the analytical behaviors of end users in the temporal dimension and uses prefetching to hide data transfer overheads while enabling the smooth exploration of massive time series data. ForeCache [53] models and predicts user behaviors in the geospatial data exploration process.

Scenarios with insufficient computational resources are well handled. Out-of-core precomputation reduces memory usage through dividing data into small blocks and processing blocks sequentially [54–58]. A hybrid spatial index can swap partial in-memory data to external devices (e.g., hard drives) and combine memory/external data together for efficient analysis [1, 59]. *EdiFlow* [60] follows the same paradigm and uses database management systems as intermediate storage medium. These approaches need precomputation and work for only predefined queries.

To further reduce computational workloads, cutting-edge *Data Visualization Management System* (DVMS) proposes latency reduction through optimizing computational workflows integrating both visualization systems and database systems [61] and avoiding duplicated computation in the decoupled two sides.

These techniques above process all the data and generate exact results quickly. Sufficient computational resources are vital in these techniques. However, the increase of these resources cannot surpass the exponential increase of data volumes [5]. Thus, an effective direction is to develop sampling-based approximate query processing approaches (presented in Section 2.3), using a small chunk of data to represent large data and responding to end users with approximate results in a short time instead

of accurate results that need lengthy computation time. Our contribution in this dissertation pertains to AQP-based approaches for spatiotemporal data.

2.3 Approximate Query Processing for General Data and Spatial data

Sampling-based approximate query processing (AQP) techniques use a small ratio of data to quickly generate usable results [11], which reduces computational latency and ensures the interactivity of VA systems.

Data sampling techniques contain a series of approaches, e.g., random simple sampling, stratified random sampling, cluster sampling [11]. Each approach focuses on a particular scenario. For example, a population contains multiple subgroups, and properties per subgroup are diverse. A statistic estimate averaging all of the subgroups cannot approximate each individual subgroup well and likely leads to estimation errors [11]. Stratified random sampling pays attention to this phenomenon and conducts sampling with subgroups. Estimation conducted per subgroup avoids interference from other subgroups and produces representative estimates.

Further approaches significantly improve data sampling performance. BlinkDB [62] used computational parallelism to data sampling in order to speed up approximate answer estimates and generate reliable answers in real time. DAQ [63] and Database learning [64] abstracted knowledge from previous data queries to improve the accuracies of new data sampling results. Online aggregation [12, 13] quickly sampled a few data, evaluated the accuracy of the approximate results, and progressively enlarged the data samples to improve the accuracies of the approximate results. This online method enabled users to see query results quickly and control the sampling execution time in terms of their accuracy goals. These aforementioned data sampling techniques work for general data relations, without making specific improvements in spatial relations. Instead, we focus on spatiotemporal data relations, which adds an additional dimension of complexity.

Efficient data sampling in the spatial dimension requires special treatment. For efficiency, sampling [43, 65, 66] is conducted upon hierarchical spatial indexes (e.g., R-tree [67], Quad-tree [68]) so that sampling time is scalable regarding queried spatial ranges. To satisfy the time-critical scenarios, STORM [1] samples spatial data in the online manner through using well-designed sample buffers. Furthermore, sampling strategies [69–71] center on object movement sequences that incorporate an extra temporal dimension. The approach presented in this dissertation (Chapter 4) supports online aggregation in the spatiotemporal domain as well, addressing the sampling bias issue decreasing answer accuracy. Mozafari [5] discusses challenges and opportunities regarding the interaction of AQP with the real world, pointing out difficulties in persuading organizations to deploy AQP models in their existing databases. Likewise, the AQP approach proposed in Chapter 3 was motivated by computational platforms with AQP functionalities and presents a solution for conducting large spatiotemporal AQP queries between data providers and client-side visual analytics systems.

2.4 Incremental Visualization and Progressive Visual Analytics

As large data processing is time-consuming, VA systems use AQP techniques (Section 2.3) to divide large data into multiple small chunks, process one chunk for quick response, and then progressively process other chunks. Thus, users receive approximate results immediately and can then observe answers progressively improve. This progressive computational paradigm has multiple names [3,72]. This dissertation uses incremental visualization (IV) for approaches presenting new visualizations [73] and progressive visual analytics (PVA) [74] for approaches focusing on data analyses.

Compared to the blocking paradigm [74] that takes time to process all the data and generate exact answers, the progressive paradigm has significant benefits. First, exact answers are not essential for many analytical scenarios. Users can use error-bounded approximate answers to make the best decisions [5]. Second, the progressive method increases user engagement with data exploration activities [3, 74, 75]. The blocking

paradigm makes domain users reluctant to analyze large data, because processing the entire dataset for exact analytic results is time-consuming and waiting for the computation to complete is exhausting. Instead, the incremental workflow helps users see results quickly so that they can detect errors at early stages, e.g., incorrect query specifications or inappropriate data settings. Furthermore, the progressive manner increases analytic data coverage, e.g., trying new analytical hypotheses and exploring more data. Finally, users can steer large data analyses on demand. Users can trade computational time for answer accuracy in terms of analytic task requirements and can adjust computation at interactive rates.

The progressive manner requires users to choose satisfactory answers for their analytic tasks. Approximate answers undergo three phases, starting with early partial results indicating whether the computation is meaningful to analytic goals, proceeding to mature partial results reliably reflecting the final answers within acceptable error bounds, and finally becoming definitive partial results without substantially discernible differences from the final answers [76]. A user study conducting progressive analyses of social media data found that most participants (eight out of ten) chose answers at the mature or definitive phases, because they thought the observed answers were unlikely to change [75].

User uncertainty is a key issue for progressive data analyses [3, 75, 77]. Due to sampling randomness, answers may fluctuate as more data are incorporated into the computation. Consequently, users are unsure whether the answers they receive are stable and trustworthy. To enable users to estimate the accuracy of approximate query results and select trustworthy answers, a series of approaches were developed in different aspects, e.g., accuracy estimation, data management and machine learning, user interaction, and visualization [74]. The following paragraphs categorize these approaches respectively.

First, statistical estimates are widely used to indicate the proximity between approximate and exact answers. A confidence interval [78] is a typical statistical measurement AQP-leveraged analyses [3, 12] to estimate ranges containing exact answers

in a certain probability (usually at least 90%). To enable users to express error guarantees for `Group-by` queries, Sample+Seek [13] introduced a precision metric, *distribution precision*, to express specific accuracy demands and accordingly designed a sampling scheme that generates samples satisfying the expressed guarantees.

Second, data management and computation should accommodate the incremental workflow. Section 2.3 overviews approaches regarding progressive data selection. As for machine learning, A-tSNE [79] progressively projects high-dimension data into lower-dimension spaces. PANENE [80] progressively computes k-Nearest Neighbors.

Third, novel user interface and visualization designs are essential in order to reduce user uncertainty. Badam et al. [75] studies a series of PVA user interface design criteria, e.g., comparisons of partial results calculated through different data selection progresses. These criteria target the improvement of users' understanding of progressive feedback and control of the progressive computational workflow. *Optimistic Visualization* [4, 77] designed a suite of interactive tools allowing users to compare differences between the approximate results already used in their decision-makings and the precise results, so that users can detect and recover from errors. On the visualization side, a series of approaches enhance users' understanding of approximate answers. Statistical measurements of inaccuracies can be represented by *uncertainty visualization*, a research area focusing on visualizing analytical uncertainties. The work [81] used hue to encode uncertainties of an isosurface-based volume rendering. Error bars can intuitively encode confidence intervals [3]; Correll and Gleicher [82] compared four types of error bar designs in terms of their abilities to help end users understand means, confidence intervals, and other measures. They also recommended two error bar options for certain data query goals. Other work [83] also conducted empirical user studies to explore visual design choices for understanding uncertainties. Copula [84] is a visual analytics approach to expressing uncertainties resulting from grid-based statistical analysis in the spatial domain. Bubble Treemaps [85] is a visual design that represents uncertainties in hierarchical structure data. To prevent the randomization of data samples in query answers, Kim et al. [86] came up with a

sampling approach that outputs samples only if the approximate visuals' crucial properties (e.g., ordering) built upon the samples are guaranteed to be consistent with the exact visual properties. INCVISAGE [73] is a sampling-based incremental visualization approach for one-dimensional or two-dimensional Group-by queries. It enables visualization to be smoothly and consistently refined without dramatic changes. As for visualization consistency across different spatial granularities, a spatial *level-of-detail* sampling mechanism ensures that approximate sampling results are consistent and coherent at each spatial granularity the end user explores [87]. For approximate visualizations in the geospace, sampling approaches probably provide inadequate data for displaying low-density areas. Users thus draw incorrect conclusions, e.g, trajectories that discontinue in low-density areas, which do not match the truth. To avoid this issue, one effective sampling strategy is to increase the probability of data being selected from low-density areas [2]. Human perception can guide sampling-based AQP techniques toward providing visualizations that are easier to observe. Details are in Section 2.5. Our perception-aware difference assessment approach (Chapter 5) also focuses on user concerns about the reliability of approximate answers.

2.5 Perception-Aware Data Visualization

Visualizations use graphical elements to encode information, and humans look at these visual displays in order to perceive the information. In light of the vital role of human perception regarding information extraction from visuals, perception-aware visualizations determine their visual design choices and adjust their data processing plans in terms of human perception.

The human visual system (HVS) perceives and processes visual information in a unique way. The human visual system has difficulty precisely decoding graphical information [88], since it cannot perceive minor differences between graphical elements. The Weber-Fechner Law [89, 90] models the relationship between the intensity of a physical stimulus and the smallest change needed for a human to perceived that the

stimulus has changed. In the the Weber's Law component, $\Delta I = K_w I$, where K_w is a constant called the Weber Fraction, I is the stimulus intensity, and ΔI is the discrimination threshold. This component is useful for describing a specific stimuli, e.g., brightness. The equation infers that the brighter a color, the harder it is to discern from colors with similar brightness, since the color's discrimination threshold increases as well. Cleveland et al. [91, 92] conducted user studies to explore the quantities and qualities of numerical values inferred by visual observations, and to provide an preliminary rank of several visual encoding choices for analytical tasks including: position, angle, length, slope, direction, area, volume, curvature, and shading. For scatterplots, empirical rules were constructed to describe the impact of a stimulus (e.g., brightness) on the inferred correlation [93], and further experiments show that shapes marked in the scatterplots strongly influence color and size perception [94]. With time series charts, user-perception experiments showed that visualizing each time series in separate spaces allowed for efficient comparison across a large graphical span, and shared-space charts were better for comparisons over smaller spans [95]. One preliminary experiment regarding animated visualization demonstrated that color is the visual encoding option that makes animated visualization changes easier to observe [96].

For color perception specifically, a series of approaches were developed to measure the discernible distances required in order to differentiate variants of colors. Just-Noticeable-Difference (JND) [97] refers to minimal modifications to a color so that humans can see the difference between the old and new color at a rate better than that of chance. Empirical JND rules are determined using different color models, e.g, luminance whose adaption rules are measured in gray-scale colors [98]. The CIELAB-based color models [99] have evolved their color difference metrics, e.g., CIE76 [99], CIE94 [100], and CIEDE2000 [101]. In addition, the sizes of the objects being observed impact JND thresholds [102]. Stone [103] presented an empirical rule to quantify JND values in terms of color, visual object size, and visual angles; when considering nonuniform color across object appearances, S-CIELAB [104] quantified color differences with this factor. Color difference theory has been widely used in in-

dustry applications [99], e.g., estimating the visibility of printed halftone textures [99] and image quality after image compression [105, 106].

Humans are likely to gain the same amount of information from a visualization with fewer data samples [107]. Therefore, an emerging research topic is to leverage perceptual empirical knowledge to guide AQP-centered incremental visualization and analytics in order to avoid sampling more data than needed. Unlike statistically measured parameters which determine the number of expected data samples, preliminary perceptual functions [15, 107] use human perceptual knowledge to simulate the information users can perceive from visualizations and to further determine an intended sample size, smaller than the size derived from statistical measures.

Our work (in Chapter 5) is related to human perception as well. Unlike the aforementioned works, we use human perception to assess the differences between approximate and exact spatial heatmaps during the incremental visualization process. We detect visual elements with discernible differences between size-by-size heatmaps. Thus, the accuracy of approximate heatmaps quantified based on detected visual elements can reflect the information actually perceived by users, and is expected to help reduce user uncertainty when choosing usable heatmaps for decision-makers.

3. SPATIOTEMPORAL DATA SAMPLING UNDER ARCHITECTURAL CONSTRAINTS

A version of this chapter was previously published by IEEE. Guizhen Wang, Abish Malik, Chittayong Surakitbanharn, José Florencio de Queiroz Neto, Shehzad Afzal, Siqiao Chen, David Wiszowaty, David S. Ebert. A Client-based Visual Analytics Framework for Large Spatiotemporal Data under Architectural Constraints. The IEEE Workshop on Data Systems for Interactive Analysis 2017. doi: 10.1109/DSIA.2017.8339088.

This chapter introduces the client-based data visualization framework designed to support interactive exploration of large spatiotemporal data under architectural constraints. Section 3.1 introduces obstacles of architectural constraints on interactive analysis of large data, Section 3.2 shows system components, Section 3.3 and Section 3.4 detail the client-side data organization and the associated data sampling procedure, Section 3.5 and Section 3.6 present and discuss experimental results, and Section 3.7 concludes this work.

3.1 Background

In the big data era, interactive visual analytic environments often require advanced computing platforms or advanced client-server architectures with sufficient computing abilities to enable interactive analysis. These solutions typically offload the expensive computational tasks to a high-performance server or a distributed computing platform (e.g., Hadoop), while leaving the client-side application to mainly focus on user interactions and visual representations.

Unfortunately, in many businesses and local governmental organizations, policies or budgetary concerns may prevent deployment of such solutions, or the transfer of data to external servers [8, 9], thereby, only allowing the server to provide data, and

requiring the client to take care of all the computational workload. However, for large data, typical client machines probably lack adequate computational resources to process the entire data, let alone providing real-time responsiveness to end users.

In order to address this architectural constraint, we propose an incremental visual analytics framework that enables interactive analysis of large spatiotemporal data under these client-server constraints: (1) a fixed server that only serves as a data provider (e.g., a relational database), and (2) a local client-side system (e.g., desktop, web-based) subject to limited computational and memory resources.

3.2 Framework Overview

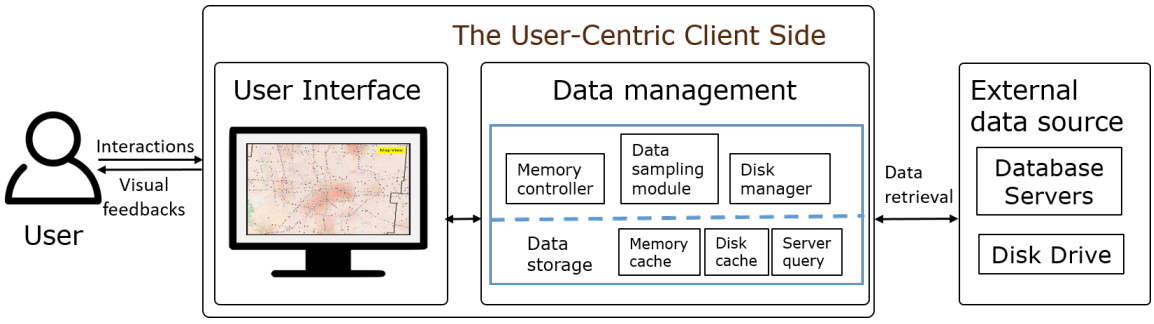

Fig. 3.1. Framework components and workflow within a visual analytics system.

Our interactive client-based visual analytics framework can be decomposed into three main components: (a) the client-side user interface, (b) external data sources, and (c) the client-side data management model, seen in Figure 3.1.

The **user interface** is a multiview visual interface that supports user interactions and provides incremental visual feedback to users. When users issue a new query, the user interface will send the user-specific spatiotemporal range to the data management model, and after receiving data samples sent by the data management model, filter data, update the visualization. The accuracy of approximate visualization is progressively improved, since new samples are continuously received from the model

and added to the visualization. Figure 3.2 demonstrates the progressive updates of a heatmap.

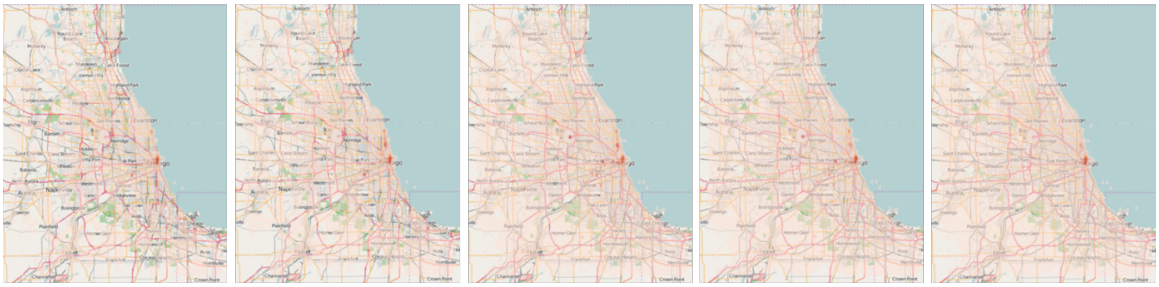

Fig. 3.2. Incremental visual updates of the heatmap using the kernel density estimation technique [20] to estimate the distribution of tweets in Chicago (introduced in Section 3.5). Different data loading percentages are indicated by the numbers below images.

External data sources indicate any data providers that are absent in the client machine's memory, including remote data servers, data in the client hard drive, FTP and so on. Data servers can be any platform that hosts the entire dataset and allows clients to fetch data. Despite the architectural constraints to prevent deploying customized functions, databases on the server side can provide the function to retrieve data based on the spatiotemporal range. For example, popular spatiotemporal indexing techniques (e.g., R-Tree [67] and data cube techniques [45,108]), widely provided in commercial database systems, can efficiently reduce the data query time. Therefore, in our framework, we simply assume that the data server query time is proportional to the number of data records requested by the client, and so is the data transfer time from the server to the client. As long as the client retrieves a smaller number of points from the server, the latency between sending the request and receiving the data can be reduced to an appropriate value.

The **data management model** has three components: the sampling module, the memory controller, and the disk manager. The **data sampling module** (intro-

duced in Section 3.3 and Section 3.4) incrementally fetches data from the server in accordance with the sample size specified by an upper bound for interactive performance. The **memory controller** monitors, predicts, and swaps in-memory data to the disk when necessary. When the application launches, memory will be consumed for initialization of the user interface and the data management model. After initialization, remaining client memory is gradually consumed for loading new data, and the memory cost of the user interface will keep almost constant despite incrementally accumulating data samples for these visualizations. For example, the spatial heatmap in the system initialization stage requires memory allocation for a grid to store the spatial histogram. In the incremental updates, the user interface only updates bin values in the grid, without allocating new memory. To prevent the out-of-memory issue, the controller tracks the memory utilization of each incremental update and estimates the memory usage in next update, since the data sampling module can predict the number of points that is going to be obtained. That information can be applied to coarsely calculate the memory cost of the next update. If the memory is predicted to be insufficient for the next round, one in-memory data node that has not been recently accessed is chosen based on the LRU rule [109] and swapped to the disk cache through the **disk manager**.

To minimize data acquisition from the server side, in addition to reuse data in the memory/disk cache, our framework supports user-driven data acquisition: loading data only matching user interest into the client side, as only data whose spatiotemporal ranges overlap the user query range are sampled, and data that are not requested by user interactions are avoided.

3.3 Client-side data organization

The proposed data management model organizes data with a two-level organization shown in Figure 3.3. The first level is temporal indexing, dividing the entire temporal range into equal bins (e.g., one month). The second level uses the spatial

indexing to organize data within the same temporal bin into a quadtree [110, 111]. Every node in the quad tree covers a rectangular spatial range. Only leaf nodes have data. An upper bound specifies the number of points one leaf node could contain at most - the framework takes care of this limit, equally dividing a node into four children when it is going to have more points than the upper bound. Thus, the spatial space decomposition in the quadtree will follow the spatial data density in its related temporal range: denser areas will have more nodes and vice versa.

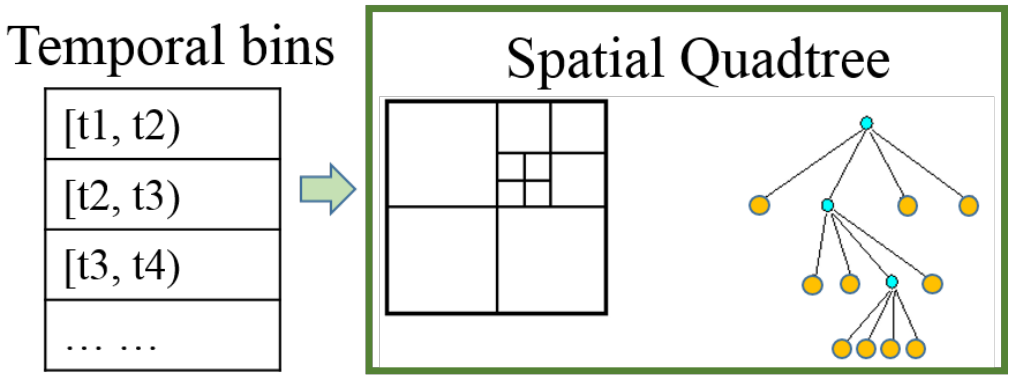

Fig. 3.3. Illustration of the client-side data organization using the two-level indexing.

3.4 Spatiotemporal Data Sampling Model

To build the client-side data organization in an average client machine, we propose a predictive way to estimate data densities through historical sampling results, refine the node organization, and further sample data in accordance with its actual statistical distribution.

The client-side data organization initially assumes that the data is uniformly distributed in the spatiotemporal space, since the client side does not have any information about the data distribution. Hence, the client runs a server query to retrieve the total count of data and the entire spatiotemporal range in the dataset. This information is used to divide the time range equally into temporal bins, and to build,

for each temporal bin, a spatial quadtree. The quadtree is progressively built until the leaf has fewer points than the upper-bound framework parameter.

After receiving a data request with the specified spatiotemporal range, data are incrementally sampled in the following steps:

1. Select all leaf nodes that spatiotemporally overlap with the specified range, and form a node set S
2. Randomly choose a node in the selected node set S
3. If the density of the node is unknown, estimate the data density of the node, apply possible adjustment to the quadtree in case of upper-bound violation or merge the node and its three siblings into a parent node in case of sparseness.
4. Fetch the data of the sampled node N from the server
5. Update the corresponding quadtree leaf with the exact number of points
6. Send the data for visualization, and remove N from S
7. Repeat steps 2-6 until S is empty or canceled by end users

The data density estimation procedure in Step 3 is necessary because sampling a node with unknown data densities actually has an excessively large number of points. Consequently, the visual update can take longer to fetch data from the server and process the data on the client side, making the client application unresponsive. Predicting the number of points in a node $N_{predict}$ involves three steps. The first step is to collect nodes in other temporal bins that have acquired data from the server and have the same spatial range as $N_{predict}$. Then, the number of points in $N_{predict}$ is calculated using Equation 3.1. If predicted to have a larger point number than the upper-bound, the node can be split into four children. If predicted to be smaller than the upper bound, the father node will be estimated to see whether the four children nodes can be merged as long as the predicted data size is no more than the upper bound. The last step is to get the final node for sampling. If the original node is split,

a child is randomly selected; if the node is merged, its father is selected; otherwise, the node itself is finally selected.

$$p_{predict} = \frac{\sum_{i=1}^{n} p_i}{|n|}, \quad \text{where } n \text{ is the number of nodes that have the same spatial range with } N_{predict} \text{ and know their data densities.} \quad p_i \text{ is the number of points in the } i\text{-th node.} \tag{3.1}$$

3.5 Evaluation and Results

In this section, we conducted experiments to verify the effectiveness of our proposed framework. The client is a 32-bit desktop application implemented on the .NET framework, with one thread for the user interface (Figure 3.4) and the other for the data management model. The client connects a MySQL database hosted by a data server in the same network domain. We set the upper bound of a node to be 4096, a multiple of a Windows disk block size, 4KB. The data in the server side is grouped into multiple tables with each table for one temporal bin. Table 3.1 lists the test datasets whose temporal ranges are evenly divided into multiple bins: one month for the osp datasets, and one week for the tweet dataset. The map view was set to overview the entire Ohio state for osp, and the entire Chicago city for tweets. For the osp data, the experiment loaded all the data. For the tweets, a specific temporal range from April 3 to May 9 in 2013, nearly 1.7 million, was selected for experiment. The application was launched with a clean disk cache, and the process was repeated five times for each dataset. All experiments were conducted on a client machine with an Intel(R) Xeon(R) E5-2630 CPU with twelve cores at 2.60GHz, 32GB main memory, and a 256GB solid state drive.

Table 3.2 measured the latency in each update, including the node sampling time, the server side query time, and data transfer time, which averaged less than 500ms

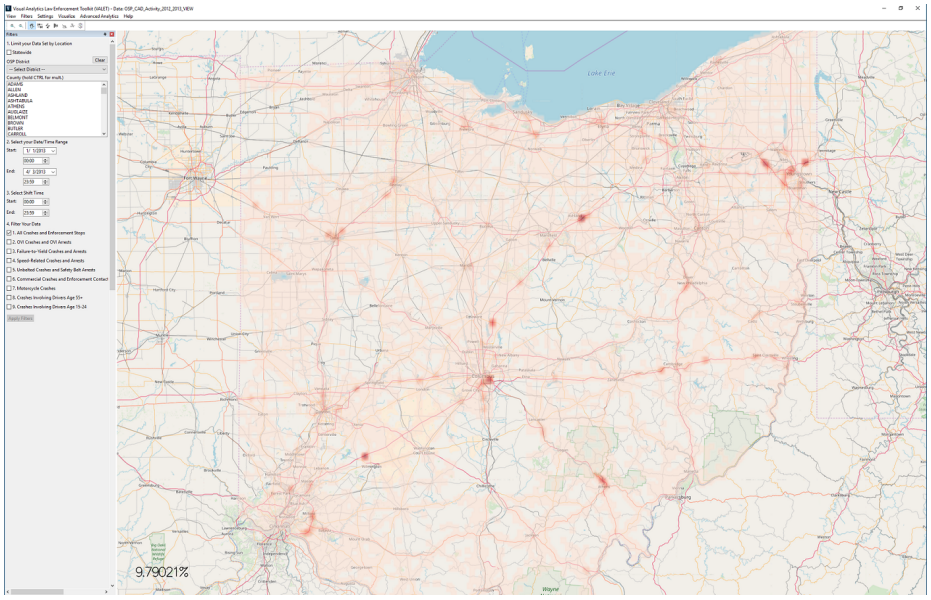

Fig. 3.4. The user interface of an implemented incremental spatiotemporal data visual analytics prototype. The left is the data filter panel, and the right is a map view showing the spatial heatmap and displaying the data loading percentage in the bottom left.

Table 3.1.
Evaluated traffic incident reports and Twitter datasets.

Data	Description	Size
osp	Ohio crime data from January 1, 2012 to December 31, 2013	3.2 million
tweet	Chicago Twitter data from April 1, 2013 to September 30, 2013	9.7 million

for all datasets. The low resulting transfer time supports the interactive analysis of end users.

Table 3.2.
Measurements of incremental updates in our prototype for one incremental visual update, averaged in five trials.

Data	Average time per update(ms)	Total updates
osp	456.8754	1915.8
tweets	407.3	4681

Figure 3.5(a-c) measures the number of points fetched from the server at each incremental update. Figure 3.5(a) shows the histogram of the number of points fetched from the server averaged in the five trials. The distribution of the osp data has a higher density around [1000, 3000], and the tweet data is more concentrated with around [0, 2000]. Figure 3.5(b) shows the number of points fetched per update in one trial of the osp data. We can see that the value in the earlier 8% updates is sometimes significantly larger than the upper bound, 4096, with the peak at 16,962, and in the subsequent updates, going down to around or below the upper bound. Figure 3.5(c) shows the number of points fetched per update in one experiment of the tweet data. In this case, the total points fetched per update is mainly around

or below the upper bound, with only four examples in the first 50% updates being significantly larger than the upper bound.

Figure 3.5(d) uses Root-Mean-Square-Error (RMSE) [112] to measure the accuracy between the approximate spatial data distribution per incremental update and the final exact data distribution. A Kernel Density Estimation (KDE) method [20, 113] measured the spatial distribution, with the spatial resolution of the 2D histogram to be 256 by 256 pixels. To reduce the measure inaccuracy from the geospatial sparsity, we only compare KDE values in denser areas defined as spatial bins with a KDE value no less than 0.05 on a normalized scale of 0 to 1. We find that the RMSE value starts around 0.15 in the first update, gradually decreases with increased data sampling, and is reduced to 0.1 when 10% of the data are sampled.

Figure 3.5(e) measures the total memory usage of our framework under two conditions. The experiment was first conducted with sufficient memory capacity, 3 GB. Then, the system was set with a memory limitation of 1.4GB and to swap memory when the entire system has consumed 1 GB physical memory. The memory required to load all data was 1.6GB for the 3.2 million osp incidents and 2.4GB for the 9.7 million tweets. We did the experiment five times each for the two dataset respectively, loading all the data to the system from the same disk cache copy.

$$RMSE = \sqrt[2]{\sum_{i=1}^{n} \frac{(F_i - I_{k,i})^2}{n}} \qquad (3.2)$$
$$F_i \geq 0.5 \text{ or } I_{k,i} \geq 0.5$$

Here, F_i is the KDE value in the i-th bin in the final converged distribution. $I_{k,i}$ is the KDE value in the i-th bin in the distribution of the k-th incremental cycle. n is the number of dense bins either in the final distribution or in the current incremental cycle. RMSE was measured for only the geospatially dense areas per incremental cycle and the final data convergence. Dense areas were defined as spatial bins with a KDE value that are no less than 0.5 on a normalized scale of 0 to 1.

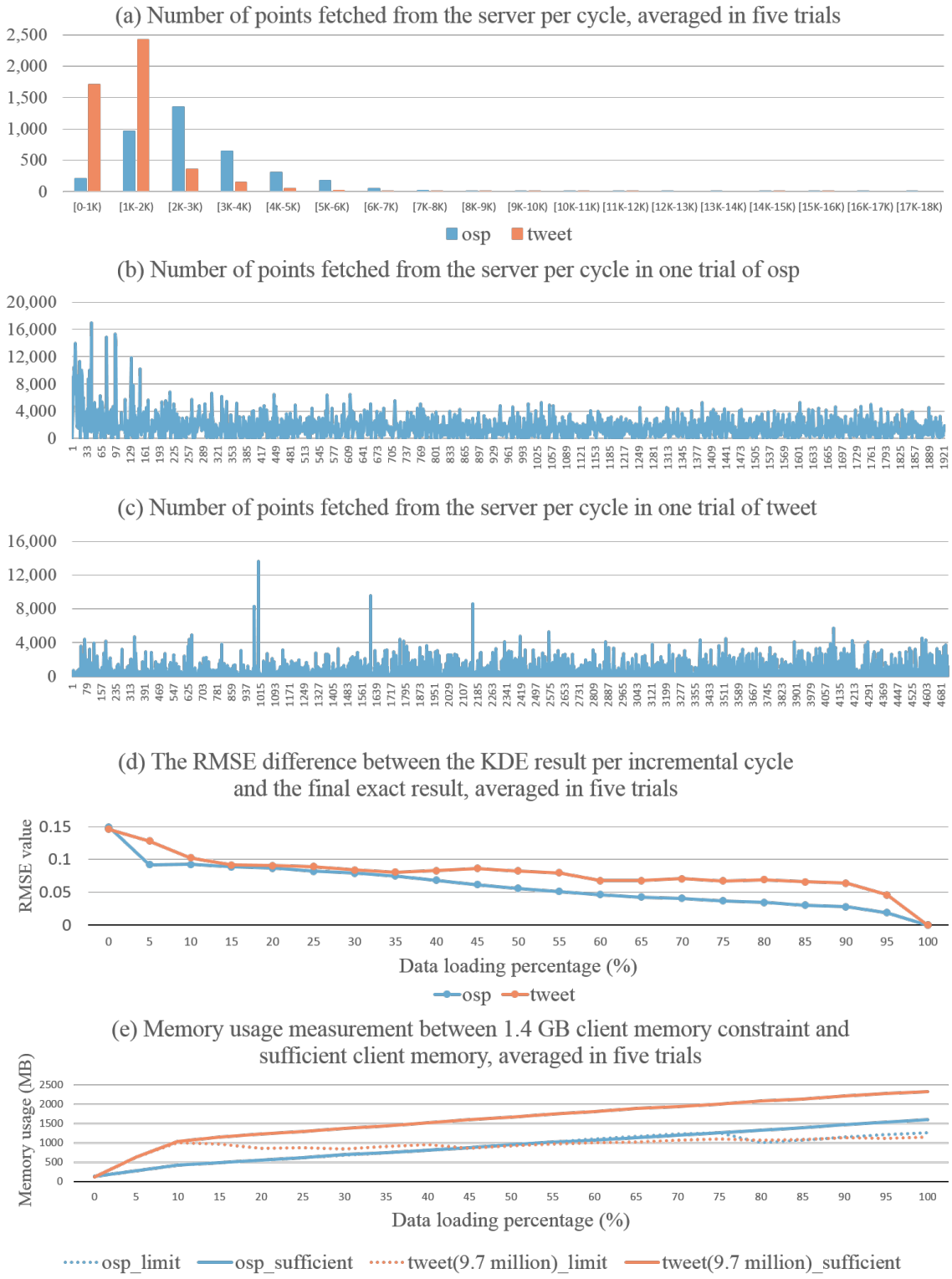

Fig. 3.5. Statistics of the experiment testing our proposed framework in five trials. In (b-c), the x-axis indicates the update sequence.

3.6 Discussion

Our work is motivated by the challenge one law enforcement agency met to use our visual analytics system VALET [20] to analyze a large scale data. Subject to security policies, their servers cannot be allowed to use, except database functions. Officers were concerned that the solution cannot consume a significant portion of the computer resources, since they need the computer to process other tasks as well, and thereby, we developed the framework, and allowed users to specify the amount of computer memory dedicated to the system.

The server query performance is vital to assure each incremental update can be completed in real time. In our experiment, the entire dataset in the server-side database was split into multiple tables, making one table for each temporal bin. We tested that, in the same condition, with the entire data being hosted within a single table, each server query increased by about 400ms in the osp data.

The data dimension choice in the two-level indexing of the client-side data organization can impact the memory usage and performance of the framework. Our method illustrated in Figure 3.3 organizes data temporally first and then spatially. One typical approach first splits data spatially then orders data temporally. Setting the spatial attribute before the temporal attribute can require only one quadtree to organize the entire data, and greatly reduce the quadtree traversal times, whereas our method builds a quadtree for each temporal bin and, to query data, traverses the quadtree in each temporal bin. However, there are important benefits from using the temporal dimension as the first-level indexing. Suppose that the "spatial first" structure uses a sorted array (e.g., Mercury [110]) to do the temporal organization of data. The temporal range will impact the structure of the quadtree. As the upper bound specifies the maximal amount of data a node can hold, a wide temporal range aggregates a large amount of data and accordingly will force nodes to divide into four children. In the case the spatial structure stores data only within the temporal range desired by users, if users extend or shrink the temporal query range, the quadtree

structure will be adjusted significantly. In contrast, in the two-level structure we use, the quadtree will remain unchanged. Another case is that the "spatial first" structure uses multiple temporal bins to split data in the leaf nodes, and makes sure that data in the temporal bins do not exceed the upper bound. In this way, the quadtree will be adjusted if, in any leaf node, the amount of data in some temporal bin exceeds the upper bound. Compared to our method, this method could generate more temporal bins as any temporal bin violating the upper bound constraint will cause the leaf node to be split into four children, which further forces other temporal bins in the same leaf node to be divided as well.

There are two parameters in our sampling method, the upper bound to specify the maximum number of points a leaf node is allowed to have and the length of a temporal bin. The upper bound can impact the interactivity of the framework. If it is set to be inappropriately large, a update can take an excessive time to retrieve and compute a node. Concerning the length of a temporal bin, the shorter the length, the more updates the approximate visualization takes. In our experiment, the total number of updates for loading the 1.7 million tweet data is twice of the 3.2 million osp crime incidents. The reason is that the temporal bin is a week for the tweet and a month for the osp.

Results in Section 3.5 reveals that the spatial distribution can impact the number of points fetched per incremental update. Comparing Figure 3.2 and Figure 3.4, we can see that the tweet data is more spatially concentrated than the osp data, having a relatively higher density only in the downtown region. Therefore, in the data organization initialization stage, with the uniform assumption, the majority of tweet nodes have a smaller number of points than the upper bound, and osp has relatively fewer nodes below the upper bound. That can explain the fact that in Figure 3.5(c), in the first 10% updates, the number of points fetched in the tweet is much smaller than that of osp in Figure 3.5(b), since it is a high chance to select nodes in the sparser area. However, for the osp, nodes in denser areas have a relatively greater chance to pick, and thus, in its first 10% updates, the number of points per update is

sometimes larger than the upper bound. Along the updates proceed, the client can refine the data index through data fetched in previous updates. That can explain the number of points fetched per update in the tweet is quite similar to the osp in the last 60% updates. Overall, the diverse spatial distribution can significantly affect initial or earlier updates, and gradually decease the impact in later updates.

3.7 Conclusion

We presented a client-based visual analytics framework for the interactive exploration of large-scale spatiotemporal data in constrained computer infrastructure settings. Our framework incorporates an incremental data analysis workflow that provides users approximate visual representation in real time within the limited computation capability of a client machine. Experiments have validated that our framework can successfully conduct interactive spatiotemporal data exploration in a typical client machine.

4. UNBIASED ONLINE SAMPLING FOR VISUAL EXPLORATION OF LARGE SPATIOTEMPORAL DATA

In this chapter, we present SpatioTemporal Unbiased onLine sampLing (STULL), a novel unbiased online sampling approach that supports incremental visualization and interactive exploration of large spatiotemporal data. STULL has a carefully designed data index and sample retrieval plan to ensure that each point satisfying the user-specified multi-dimensional data query has an equal probability of being sampled. In particular, unlike state-of-the-art spatial online sampling approach [1], our unbiased guarantee is unaffected by the intrinsic spatial distribution pattern of the data. With our approach, incrementally updated visualizations can not only achieve higher accuracies at the same sample size but also present closer visual appearances to the exact visualizations. In addition to visual quality, STULL retrieves samples as efficiently as state-of-the-art approaches (e.g. [1]), and allows users to control the number of points sampled through incremental updates. Through STULL, VA systems can provide unbiased approximate answers to queries for more accurate spatiotemporal visual analytics without adversely impacting the computational performance of the interactive data exploration. Furthermore, STULL supports sampling both stored data and streaming data, making it suitable for visual analytic environments that leverage both types of data, such as social media analytics tools [114, 115].

Our experiments confirm the effectiveness and efficiency of STULL in producing unbiased samples for large spatiotemporal data queries. Compared to the state-of-the-art online spatial sampling approach [1] on historical data, in the same computational time, STULL improves the approximate spatial accuracy by at least 50% when sampling less than 5% of the original dataset. Using our approach, approximate visualizations reduce visual differences from visualizations encoding the exact answers. For streaming data, STULL takes less than 500ms on average to index incoming streaming data

(1000∼4000+ tweets per second [110]) for answering queries, well below the response time thresholds for interactive visualization [10].

Our contributions include the following:

1. a novel, unbiased online sampling approach for VA systems to incrementally present approximate yet reliable interactive analyses,

2. theoretical guarantees on the unbiased property of our presented approach,

The rest of this paper is structured as follows. We elaborate on the drawbacks of biased sampling for visual analytics in Section 4.2, detail `STULL` in Section 4.3, and provide unbiased guarantee in Section 4.4. In Section 4.5, we explain our experiments and discuss obtained results in Section 4.6. Finally, we conclude the paper and present future directions in Section 4.7.

4.1 Background

Online sampling-supported Visual Analytics (VA) allows users to explore large volumes of data at interactive rates. This is done through continuous retrieval and visualization of data samples that approximate the distribution of the underlying dataset being queried, also known as incremental visualization [3]. As users wait for samples to accumulate over time, the increasing sample size improves the accuracy of the inferred data pattern, allowing users to trade wait time for accuracy [116].

Current state-of-the-art online sampling approach to spatial data [1] focus on the efficiency of data sample retrieval, without fully resolving the sampling bias. Biased sampling methods, by definition, sample data with unequal probabilities [11], generate data patterns that deviate from the original dataset and can lead users to erroneous conclusions. To ensure accurate, trustworthy, and reliable data exploration, selection and filtering during online sampling for visual analytics systems, it is critical to remove sampling bias.

4.2 Visual analytics And Sampling Bias

Unbiased sampling requires that *each record satisfying the query specification has the same probability of being selected* [11]. Conversely, a sampling approach is considered as biased if the probability of each individual record being selected is not equal.

Sampling bias can distort patterns of data and render data exploration ineffective and inaccurate [35]. For instance, a common aggregation task in crime analysis is to identify spatial hotspots where the most incidents occur. Data samples retrieved by unbiased approaches should approximate the hotspot patterns regardless of sample sizes; biased approaches may be skewed towards locations outside of the true hotspots and may create false hotspots.

Furthermore, such erroneous interpretations can accumulate throughout the sense-making process. VA systems are often designed to support the interactive exploration of data, following the information seeking mantra [21]: "Overview first, zoom and filter, then details-on-demand." This practice is common in geospatial analysis, where users often start the exploration by examining data patterns across the overall geographic extent, and then identify locations of interest for further investigation. However, if sampling is biased toward specific geographic regions, the visual display at the overview level could already be misleading. As a result, it would then exacerbate the biased selection of relevant regions for further exploration.

Sampling bias also impairs incremental visualization. Incremental VA systems progressively improve upon approximate answers through three main stages [76]: early, mature and definitive. Answers presented in the early stage can reflect the exact answers, helping users evaluate whether or not their analytic activities are on track. The results of the mature stage can approximate exact answers with acceptable errors and are useful for time-critical tasks. Finally, the definitive stage approximates answers that do not change significantly and can address analytic tasks that require smaller error margins. However, with the same sample size, answers constructed from

biased samples are often further from the exact answers than their unbiased counterparts. Thus, biased sampling hinders the advent of each stage and prolongs wait time for users. Moreover, users' trust in approximate answers is an intrinsic challenge of incremental visualization [77] and sampling bias can exacerbate the trust issue. For example, one effective visualization technique to help users become confident in the analytic results is to compute the exact answers offline so users can compare their selected, approximate answers against exact ones and redo their analyses if needed [4]. Biased approximate answers can increase the number of times a user has to redo analyses, decreasing the rate at which they can complete tasks.

Therefore, for data exploration in VA systems, it is vital for sampling to be unbiased, i.e., for each data record in the original result set to have an equal chance of being sampled. This ensures that the sampled data accurately represent the entire result set.

4.3 STULL

As users issue queries, `STULL` continuously samples data so that the VA system can create rapid visualizations and progressively improve them (Figure 4.1). In single incremental update, `STULL` retrieves sample points per the spatial and temporal specifications of a particular query. After receiving samples, the visualization side generates and updates the visuals. This section presents the computational details of `STULL`.

4.3.1 Some Intuition

Efficient sample retrieval is crucial for online sampling. A sampling plan that randomly selects points from a collection of data points is apparently not efficient because the retrieval accesses all of the points even if users query merely a small part of the data. A scalable plan is indexing data and retrieving samples from the index, as the index can minimize the accessed data to the subsets specified by queries. Specific

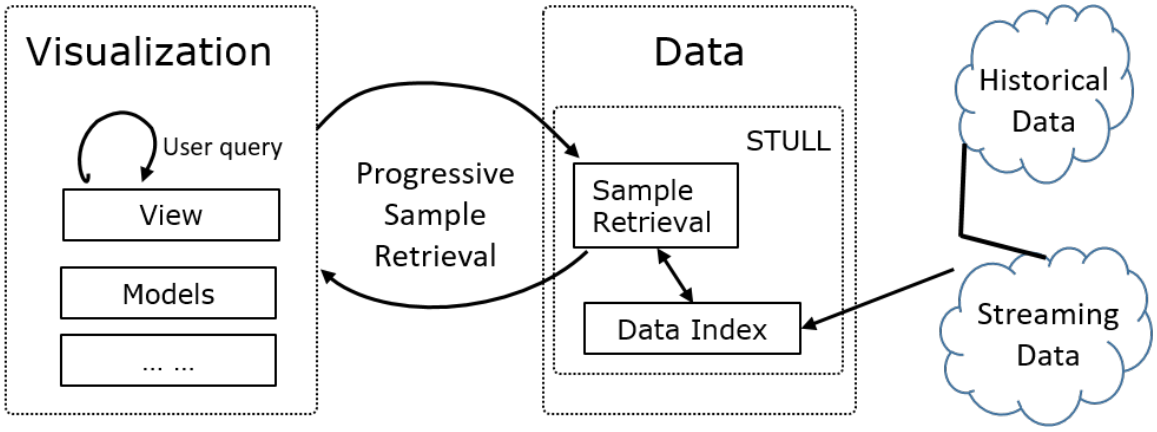

Fig. 4.1. Incremental visualization workflow leveraged by STULL.

to spatial sampling, spatial indexes (e.g., R-tree [67] and Quad-tree [68]) are widely used to organize data in a spatial hierarchy. These trees often store all the points into leaf cells. The sample retrieval procedure starts from the tree root, randomly chooses a child in terms of point volume belonging to each child, and recursively picks a child of the chosen cell until it reaches a leaf from which a point is selected. The same procedure repeats until the desired number of points is collected. This type of approaches produces samples that can unbiasedly represent the queried data, but is not efficient. First, the sample retrieval latency increases in proportion to the tree heights. Second, the retrieval selects samples from most of the leaves belonging to the queried spatial range. In a case where data indexes are stored on hard drives, loading these reached leaves into memory is a lengthy process because each leaf requires at least one disk Input/Ouput (I/O) operation and completing all the I/Os needed for loading these leaves is time-consuming. To reduce the number of I/Os, advanced approaches [1,117] group data points into a series of buffers [117] and retrieve samples from buffers. Since each buffer is a randomly ordering of points, a sequential scan of the buffer can construct a sample set. Therefore, retrieving some samples from buffers may require a single disk I/O, whereas retrieving the same number of samples

from leaves needs much more I/Os. As such, buffers-based approaches can retrieve samples quickly and support VA systems to proceed at interactive rates.

Since buffer-based sampling approaches retrieve samples in the units of buffers, the state-of-the-art **fixed-sized** design [1] in which each buffer has the same number of points raises bias (Figure 4.2). In this example, each non-leaf cell has a 500-point sample buffer. At level 2, collectively, the union dataset of the orange and green cells has the distribution of 43.3% (i.e. $\frac{1000+1600}{2000+4000}$) Q, whereas the samples that is a combination of their sample buffers has the distribution of 45% (i.e., $\frac{250+200}{500+500}$) Q. Therefore, the samples cannot accurately approximate the dataset.

To avoid this issue, we propose a **proportionally-sized** design. In this design, each cell's buffer caches 100α percent of its data. The sample buffer is therefore *proportional* to the cell's specified range. In the case when a query relates to multiple cells (Figure 4.2), the union of these cells' buffers will contain exactly 100α percent of the data being queried. Therefore, the proportionally-sized design can approximate the distribution of the queried data without bias whereas the fixed-sized [1] is contingent on the spatial index itself. `STULL` uses the proportionally-sized design so that it can prevent such issues and ensure that the samples can represent the exact spatial distribution unbiasedly.

4.3.2 Index Design

`STULL` indexes data with an ordered list of pyramids that represents the spatiotemporal segmentation of the data. (Figure 4.3). The temporal range of the data, Δt, is first divided into adjacent, non-overlapping, equal-sized temporal bins, each indexing a subset of the data that falls into its range. Within each temporal bin, data is further indexed with a pyramid (e.g., `Mars` [111]) per its spatial dimensions. Each pyramid follows the *proportionally-sized* design and its non-leaf cells cache 100α percent of points randomly selected from their spatiotemporal ranges. Therefore, each pyramid level has totally 100α percent of points. Therefore, a pyramid has $\frac{100}{100\alpha} = \frac{1}{\alpha}$

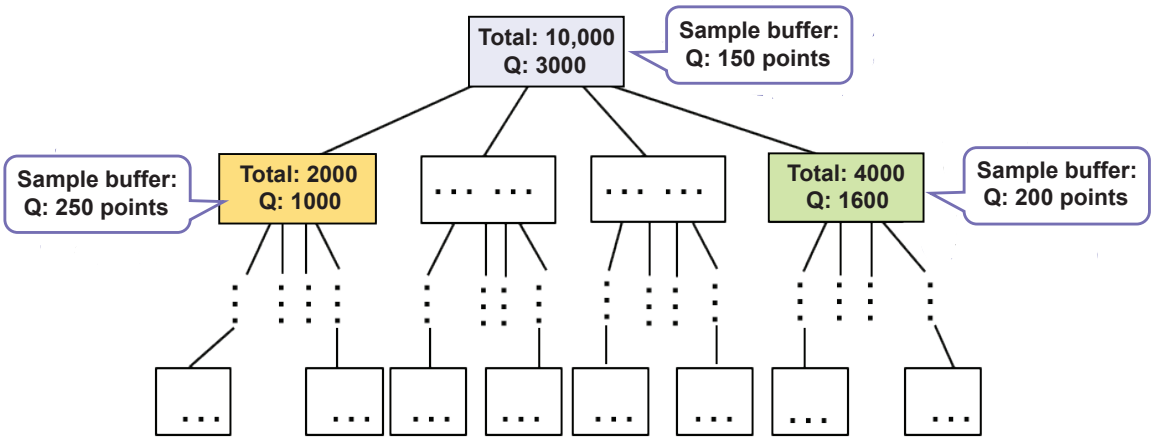

Fig. 4.2. Sampling bias issue in the fixed-sized sample buffer design. Q is a query. Each sample buffer has 500 random data points. Numbers inside each buffer lists the number of points satisfying Q. Numbers inside each cell list the total number of points in the spatial range of the cell and the number of points satisfying Q respectively.

levels in total. Each pyramid recursively and equally divides the data's spatial range into four fixed-sized rectangular sub-ranges until the $\lceil \frac{1}{\alpha} \rceil$-th level.

At the bottom level of a pyramid, leaf cells store all of the data within their range in a **circular array** (Figure 4.3(c)). We divide each circular array into $\frac{1}{\alpha}$ segments in terms of the pyramid height, where each segment contains 100α percent of the data. These segments will be used to add points into non-leaf cells' sample buffers (Section 4.3.3) and participate in sample retrieval (Section 4.3.4). Figure 4.3(c) exemplifies the circular array in the leaf with the id "1122" in Figure 4.3(b). Since $\alpha = 0.25$, its circular queue has four segments.

Cells from the non-bottom levels of a pyramid cache randomly selected samples from their respective ranges in one-dimensional arrays, termed **sample buffers**. Collectively, data in the sample buffers form a sample that approximates the distribution of the original data (Section 4.3.4).

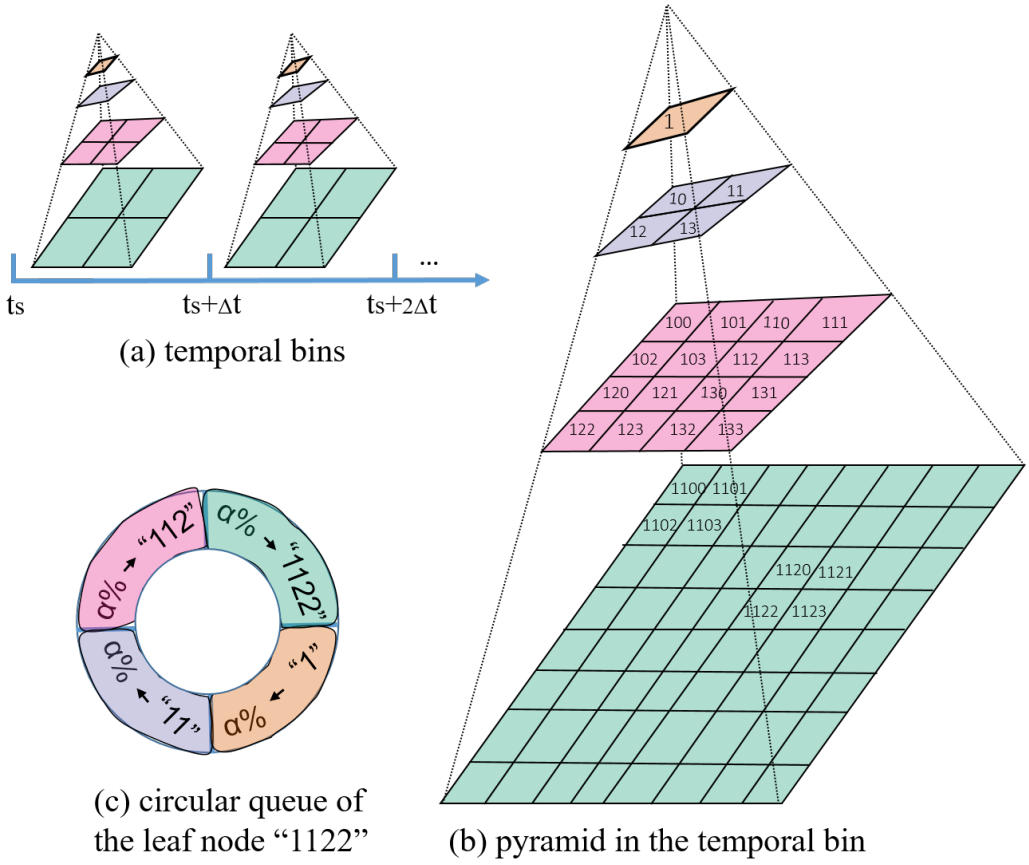

Fig. 4.3. The data index. (a) shows the temporal index beginning at t_s. Each segment in (a) is a temporal bin. Each temporal bin sets $\alpha = 0.25$ and uses a four-level pyramid (in (b)) to spatially organize data. A pyramid leaf uses a four-segment circular array (in (c)) to store data. Each non-leaf cell has a sample buffer to store data.

4.3.3 Index Creation

To build the index, STULL first puts each data point in the appropriate temporal bin. Then, starting from the root level of the pyramid, in a top-down fashion, we proceed to the appropriate spatially-ranged leaf cell and insert this point into its circular array, and repeat this process for all data points. At completion, the circular arrays at the bottom levels of all pyramids will contain the entire data set.

Next, in each pyramid, STULL adds points to sample buffers at the non-bottom levels of pyramids. First, each leaf cell randomly shuffles data in its circular array

(Figure 4.3(c)). Second, a bottom-up procedure copies segments of data from leaf cell circular arrays into sample buffers of their ancestor cells. Take one leaf for example, its circular array has $\frac{1}{\alpha}$ segments. In clockwise order, the data in the first segment is copied into the sample buffer of the root level, data in the second ancestor cell on the second level, and so forth, until the $\frac{1}{\alpha}$-th segment is copied. Algorithm 1 shows the pseudo-code of this procedure.

Algorithm 1: Building sample buffers

input : A list T consisting of temporal bins that need to build sample buffers.

```
1  for each time bin t in T do
2      empty all sample buffers;
3      for each leaf u of t do
4          random shuffle data in u's circular array;
5          n ← the length of u's circular array;
6          index ← 0;
7          for i = 1 : 1/α do
8              c ← the ancestor cell in the i-th level and belonging to the path
                 from u to the root;
9              b ← the whole data in the i-th segment of u's circular queue;
10             Add b to c's sample buffer;
11     random shuffle all sample buffers;
```

4.3.4 Sample Retrieval

`STULL` retrieves sample points that satisfy a given query Q in an incremental way. Suppose a VA system plans to use 100θ percent of data points to generate quick answers and progressively refine answers with the same number of new points.

In order to keep the sampling result unbiased in the temporal dimension, STULL retrieves only a 100θ percent of sample points from each temporal bin requested per Q. The union of samples retrieved from all of the requested bins is the set of samples the visualization side uses for visual computation.

Retrieving 100θ percent of points from a single temporal bin takes the following steps. First, STULL randomly picks a pyramid level l_r to retrieve points. For each of cells spatially overlapping with Q at the l_r level, we retrieve 100θ points from its data, which is equivalent to retrieve θ/α percent of data from its sample buffer. In each incremental update, the retrieval repeatedly retrieves a chunk of θ/α percent of data from sample buffers of eligible cells. The retrieval on a level continues until either users terminate the incremental retrieval procedure or the sample buffer is exhausted after $\frac{1}{\theta}$ rounds, in which case, we move on to the next level $[(l_r + 1)\texttt{mod}\frac{1}{\alpha}]$.

Suppose l_Q is the lowest pyramid level in which a single cell contains the queried spatial range. There is a $(\alpha l_Q - \alpha)$ probability that $l_r < l_Q$ and consequently the l_r level has more points irrelevant with Q. In such a case, to avoid most of the irrelevant points, we will simply retrieve samples from the bottom level since it contains all of the data.

When sampling is on the bottom level, the same steps are conducted on consecutive segments of leaf cells' circular arrays. In the case $l_r \geq l_Q$, since the retrieval has obtained points from Level l_r to Level $(\frac{1}{\alpha} - 1)$, the retrieval begins at the $\frac{1}{\alpha}$-th segment. Otherwise, it starts at the l_r-th segment. Similar to sample buffers, the retrieval procedure accesses the same θ/α portion of point in a segment, continue, and will exhausts all eligible points after $1/\theta$ times. Then, index of the next retrieved segment is $(l_r + 1)\texttt{mod}\frac{1}{\alpha}$. Likewise, the retrieval continues until either users cancel or l_r is reached again.

Algorithm 2 describes the whole retrieval procedure in a time bin.

4.3.5 Index Update

Once new data arrive, STULL updates its data index through finding temporal bins associated with the new points, adding the points into leaf cells and following Algorithm 1 to refresh sample buffers in each associated bin. This updating procedure applies to both existing data and new streams of data. In general, existing data (e.g., historical logs) are well collected and curated before the visual analytics process. Thus, its index update has sufficient time to conduct before queries, unlike streaming data, which must be timed carefully. Incoming data streams and their queries span more recent time ranges (e.g., a monitoring system [111] querying sensor data collected in the last ten minutes); the update procedure likely adjusts only the latest few temporal bins, which is therefore fast.

4.4 Unbiased Sampling Guarantee and Computational Performance

Unbiased sampling in STULL is guaranteed as a result of the index and the aforementioned sample retrieval procedure. We provide the theoretical proof of its unbiased claim in Section 4.4.1. STULL also guarantees interactive rates, making it suitable for online sampling and incremental visualization. A formal computational complexity analysis is detailed in Section 4.4.2 and Section 4.4.3.

4.4.1 Unbiased Sampling Guarantee

We prove that STULL conducts unbiased sampling in two steps. First, we prove that STULL retrieves samples from one temporal bin without bias, and then prove for cases that use multiple bins.

For one temporal bin (e.g., a t-th bin), the sample retrieval procedure follows Algorithm 2 to access its pyramid and obtains 100θ percent of points that satisfy Q. Recalling Algorithm 2, the sample retrieval starts from a random level l_r, if $l_r >= l_Q$, and the bottom level otherwise. The procedure in the case of $l_r >= l_Q$ is equivalent

to the other procedure. Thus, we reduce the proof of unbiased sampling for just the bottom level. Suppose C_Q is a set of leaf cells that spatially overlap with Q. In each leaf of C_Q, the retrieval process on average accesses l segments of its circular queue, where $l = \theta/\alpha$. For each leaf of C_Q, the equivalent procedure first accesses the l_r-th circular queue segment and then continues fetching data from $(l_r + 1)$-th section in a clockwise order until l segments are accessed. Equation 4.1 shows that each segment in the circular queue of a leaf has an equal chance of being selected. Equation 4.2 shows that each point satisfying Q in a leaf is also the same for other points satisfying Q. Therefore, points in each leaf of C_Q are equally likely to be selected.

$$\begin{aligned} &P(\text{Segment } l_i \text{ is chosen}) = P(l_r = l_i) + P(l_r \neq l_i) \times P(l_0 \\ &\quad \text{belongs to } l-1 \text{ segments counterclockwise from } l_i) \\ &= \frac{1}{1/\alpha} + (1 - \frac{1}{1/\alpha})\frac{l-1}{1/\alpha - 1} = l\alpha = \frac{\theta}{\alpha}\alpha = \theta \end{aligned} \tag{4.1}$$

$$\begin{aligned} &P(\text{Point } r \text{ is chosen}) \\ &= \sum_{l_i=1}^{1/\alpha} P(r \text{ is in the } l_i\text{-th segment}) \times P(l_i \text{ is chosen}) \\ &= \sum_{l_j=1}^{1/\alpha} \frac{1}{\alpha} \times \theta = \theta \end{aligned} \tag{4.2}$$

Second, we prove that sample points retrieved from multiple temporal bins are unbiased as well. Suppose Q requires 100θ percent of points from each temporal bin in a set, T_Q. Derived from Equation 4.2, a point satisfying Q in the t-th bin is selected with probability θ. Therefore, `STULL` ensures that each point satisfying Q has the same selection probability θ.

In conclusion, `STULL` is unbiased in selecting points satisfying a multidimensional query specification.

4.4.2 Index Construction and Update Performance

Given a data set of N points, `STULL` first takes $O(N\frac{1}{\alpha})$ to put all points into the right temporal bins and go through a $\frac{1}{\alpha}$-length path to put each point into the right

leaf. Next, building sample buffers (Algorithm 1) includes a linear combination of two steps, 1) at the bottom level: random shuffling the circular array of one leaf having m data points is O(m), hence random shuffling of leaves within all time bins is O(N); 2) at other levels: each cell having n points contains a sample buffer of αn points, and needs O(αn) for random shuffle; since non-leaf cells in the same level have a total of αN points in their sample buffers, thereby, sample buffers in the same level take O(αN) for random shuffle. Therefore, the time complexity to save data into sample buffers is O($N + (\frac{1}{\alpha} - 1)\alpha N$) = O($(2 - \alpha)N$). To conclude, the complexity of index construction is O($N\frac{1}{\alpha}$)+ O($(2 - \alpha)N$) = O($(2 + \frac{1}{\alpha} - \alpha)N$).

In the case of streaming data, suppose the newly arrived data has n points. Let M be the set of temporal bins that will insert new data, and a total of N_M points have been indexed in the bins of M. `STULL` takes O($n\frac{1}{\alpha}$) to put all points into the right bins and leaf cells, and takes to build same buffers is O($(2-\alpha)(n+N_m)$. As such, the time complexity to index new data is O($(2-\alpha)(n+N_m)+n\frac{1}{\alpha}$) = O($(2+\frac{1}{\alpha}-\alpha)n+(2-\alpha)N_M$). From the time complexity, we can see that the performance of indexing newly arrived data relates strongly to N_M. Since continuous data streams often provide data with the up-to-date time values, therefore, temporal bins that contains new points tend to be the ones belonging to the latest time range. If the time interval of temporal bins is relatively smaller, accordingly, the total time span of bins in M becomes smaller in the finer granularity of temporal bins, which further helps reduce N_M. Thus, we recommend that the time interval of temporal bins in the streaming data index should be relatively smaller.

4.4.3 Online sample retrieval performance

According to the sample retrieval algorithm (Algorithm 2), `STULL` retrieves on average 100θ percent of points satisfying Q in each while loop (within lines 14-18). Suppose n_Q denotes the total number of points belonging to a set of leaf cells that

spatially overlap with Q. Thus, each update retrieves $n_Q\theta$ points. Thus, its time complexity is $O(n_Q\theta)$.

4.5 Evaluation

In this section, we present our experiments and results to demonstrate the effectiveness of STULL.

Implementation: STULL and its two baseline approaches are built upon the Microsoft .Net Framework v4.0 and written in Visual C++ [121]. The two baseline approaches are briefly described below.

- `STORM` [1] is a spatial online sampling approach, using a R-tree [67] to index a dataset in terms of spatial coordinates of the points. The R-tree's non-leaf cells have fixed-size sample buffers. In our experiments, `STORM` used the Boost library API [122] to build a quadratic R-tree [67], and each of its sample buffers have 1024 data points.

- `RandomPath` is a variant of a spatial sampling approach [66]. In our implementation, points are grouped into temporal bins, and each bin uses a Quad-tree [68] to index its points spatially. There are no sample buffers, and all of the points are stored merely in leaf cells. In each bin, it follows the spatial approach [66] (in Section 4.3.1) to retrieve samples. `RandomPath` produces an unbiased sampling, but is slower in sample retrieval, especially when its data resides in hard drives. Thus, `RandomPath` is an alternative approach to sampling in place of online sampling.

Data sets. Table 4.1 lists the test datasets. Spatial distributions among the datasets are diverse. Hotspots scattered in the OSP case and concentrate at few locations the in other three.

Environment. We conduct all experiments on a machine with an Intel(R) Core(TM) i7-4770K CPU at 3.5GHz, 8GB main memory, and a 256GB solid state drive.

4.5.1 Numerical Accuracy of Approximate Answers

We quantified one of the advantages of unbiased sampling through the accuracy of approximate answers expressed in numbers. The accuracy was measured by RMSE [112], which calculates differences between exact answers and approximate answers.

Regarding accuracy in the spatial dimension, we queried the Kernel Density Estimation (KDE) [113] results of the whole data in the geospace. RMSE was measured on spatial bins with a KDE value no less than 0.05 on a normalized scale of 0 to 1. Figure 4.4 shows the accuracy of approximate KDE results, compared to the exact KDE results. Overall, RMSE values and sample sizes are inversely correlated. At the same sample size, `STORM` has the most significant RMSE values, and the other two are almost the same or smaller. When the sample size is 5%, `STORM`'s RMSE value is at least twice as much as the others; and the difference decreases along with the increase of sample sizes. Moreover, RMSE values in the OSP case are the largest at the same sample size, and nearly three times that of the other three at a particular 5% size.

For accuracy in the temporal dimension, we queried the hourly distribution of points in the entire data set. The density value in each hour was normalized to the scale of 0 to 1. Figure 4.5 shows RMSE-quantified accuracy, compared to the exact distribution. We see that at the same sample size, `STULL` has the lowest RMSE errors. At 5%, `STULL`'s value is averagely 50% less than that of the others. Furthermore, the RMSE errors in the OSP case are the largest.

4.5.2 Visual Accuracy of Approximate Answers

To show the impacts of unbiased sampling on the accuracy of approximate answers expressed in visualizations, we compared them in the spatial and temporal scenarios respectively.

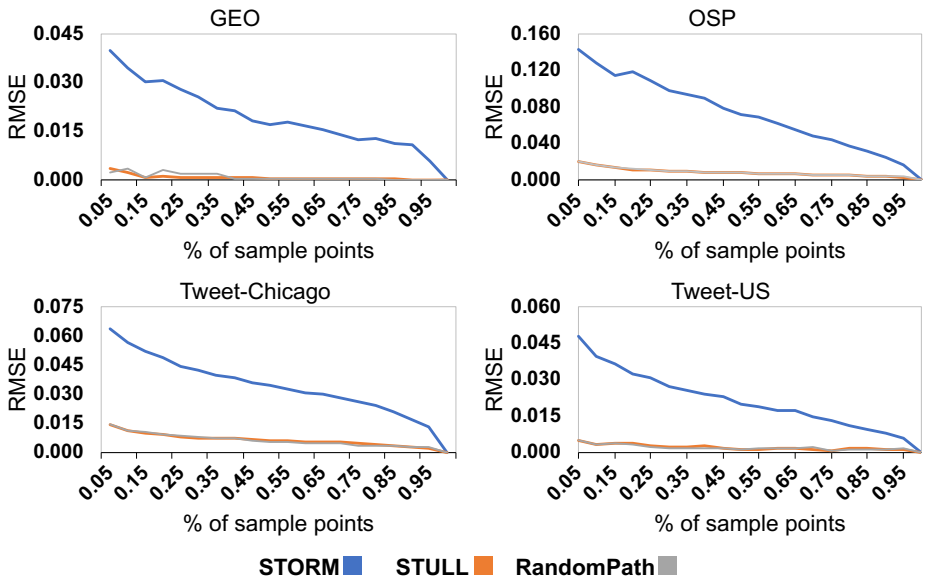

Fig. 4.4. RMSE measurement of approximate Kernel Density Estimation results. The query requested the entire dataset. Results were average over five runs.

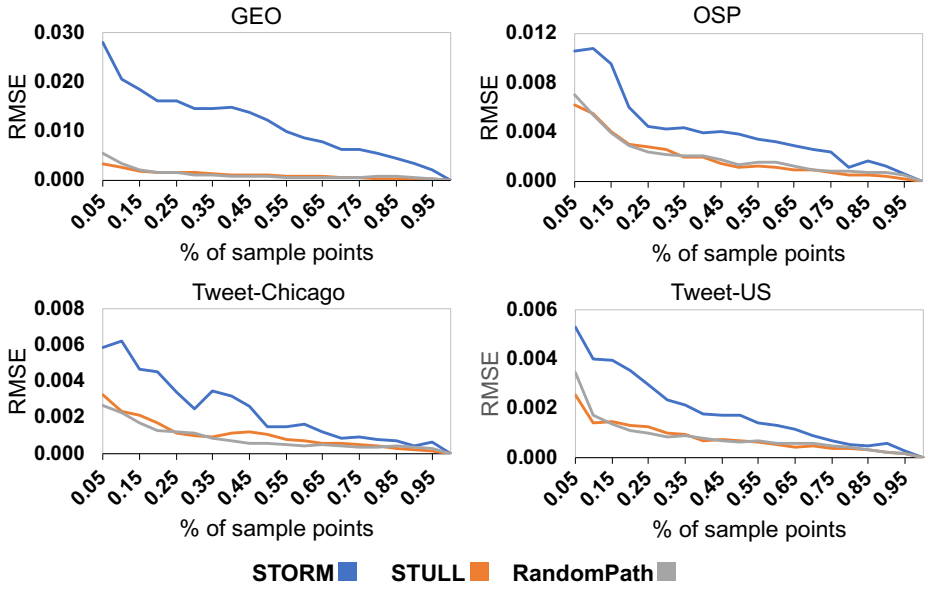

Fig. 4.5. RMSE measurement of approximate hourly distribution results. The query pertained to every point in the entire dataset. Results were average over five runs.

Figure 4.6 shows incremental visualization of approximate spatial heatmaps [24] created by STORM and the unbiased-guaranteed STULL respectively. Overall, heatmaps of both approaches progressively get closer to the visual appearances of the exact ones. At a smaller sample size, both heatmaps have perceptible differences in low-density areas since these areas have fewer points selected; When sample sizes exceeds certain numbers, heatmaps of the both approaches display indiscernible visual appearances. At the same sample sizes, heatmaps generated by STORM are perceived as presenting more visual differences from the exact heatmaps than the other; likewise, STULL uses less samples to generate heatmaps that are indiscernible from the exact ones in terms of human perception. In the incremental updates, hotspot (densities values at least 0.5) distributions in the STULL's heatmaps keep constant without discernible changes, whereas hotspots in the STORM cases have noticeable changes when the sample sizes are smaller, e.g., the heatmap with 1.4% points and the heatmap with 5.1% points in the OSP case.

As for the temporal dimension, Figure 4.7 compares pie charts encoding the hourly distribution of points selected by STULL and STORM. Overall, pie charts associated with STULL have less visual differences to the exact ones than those with STORM. In the OSP case, point densities between 6 AM and 12 PM extrapolated from the STORM-supported chart clearly disagree with that of the exact one. This is also true of the GEO case, where densities between 5 PM and 1 AM extrapolated from STORM's chart have obvious discrepancies. STORM also presents light but discernible color differences between 12 AM and 6 AM in the Tweet-Chicago case and between 7 AM and 11 AM in the Tweet-US case.

4.5.3 Latency of Incremental Updates

We measured incremental sample retrieval latency in multiple scenarios.

First, a series of experiments were conducted when data indexes were in-memory. Figure 4.8 shows the time spent progressively retrieving samples for a query that

queried the entire data. It shows that STULL can retrieve a sample of 5% data in less than 250ms, and the entire dataset is retrieved in 1 to 4 seconds, depending on the data volume. Both STULL and STORM have almost the same retrieval latencies, overall shorter than RandomPath. On average, at the same sample size, STULL saved at least 60% of the time used by RandomPath. Figure 4.9 presents the same time measurement for a query requiring partial temporal ranges. It shows that STULL is faster because it retrieves from partial temporal bins, but STORM indexed points only in the spatial dimension and needs to access the entire dataset to filter out points in a temporal sub-range. Figure 4.10 shows averaged sample retrieval latency in one incremental update for queries requiring various spatial ranges. It shows that RandomPath takes a longer time than the other approaches. At the same sample size, STULL takes less than 35% of the time used by RandomPath. Moreover, when queried spatial extents expand, STULL remains almost the same, whereas RandomPath changes significantly.

Second, as α and θ are essential parameters for STULL, we measured sampling latency under various values of the two. Figure 4.11 shows averaged sample retrieval time per update under different number of points retrieved per update. It shows that the average time per update is almost proportional to the number of points required in each update. Figure 4.12 shows sample retrieval latency regarding α. It shows that STULL saves at least 68% of the time used by RandomPath under the $\alpha = 0.25$ settings and at least 70% of the time under the $\alpha = 0.125$ settings. In addition, the latency of STULL stays the same or increases no more than 35% if α reduces from 0.25 to 0.125, compared to RandomPath, which increases more.

Lastly, we measured sample retrieval latency when an index was stored in hard drives. Figure 4.13 compares retrieval time between STULL and RandomPath. When sampling 5% points for the first-time visual update, at least 62% of the time needed by RandomPath, averagely 3.2-4.7 seconds, is saved by STULL if it starts the retrieval at the pyramid root. But in the GEO case, it takes almost the same time as RandomPath. GEO points are extremely concentrated in a few leaves. As a result, time needed for RandomPath to load points from other leaf cells is negligible.

4.5.4 Latency on Index Creation and Update

Computational complexity analysis (in Appendix A) shows that index creation and update are impacted by α. Thus, we conducted experiments on historical data and streaming data with different α. Table 4.2 shows that the average time to index historical logs is inversely correlated with α. It indicated that latency doubled when α drops from 0.25 to 0.125. For streaming data, Table 4.3 shows the average time to insert new arrivals of 5000 tweets, with the assumed streaming data arrival rate around 1000$\sim$4000+ per second [1, 110]. This is a simulated experiment where we randomly select 5000 points from a temporal bin and measure the time required to add these data into the same bin. The recorded time is a sum of the time to add data to the pyramid and time to build sample buffers. We can see that the insertion takes 75% more time in the Tweet-US case and less than 33% in other cases.

4.6 Discussion

We validated the **importance of the unbiased guarantee for incremental visualization** through designing experiments that measured numerical and visual accuracy between approximate and exact answers. RMSE results (Figure 4.4 and Figure 4.5) show that compared to `STORM`, unbiased sampling reduces both spatial and temporal distribution errors by at least 50%, given a sample set of 5% points. The same accuracies between `STULL` and `RandomPath` confirm that our online-manner approach can ensure the same sampling quality as a regular unbiased sampling approach. Figure 4.6 and Figure 4.7 show that unlike `STORM`, approximate spatial heatmaps and approximate hourly pie charts constructed by unbiased samples have closer visual appearances to exact answers. Consequently, users have a higher chance of inferring high-fidelity answers from approximate visuals. Improved accuracy on the numerical measurements and visual effects are vital for incremental visualization in terms of user uncertainty [4, 77]. Users feel uncertain about choosing trustworthy approximate answers for their decision-making. A common solution to facilitate user evaluation

of the answer reliability is the use of statistical measurements derived from numerical properties of approximate answers (e.g., Confidence Interval) [12, 77]. `STULL` can provide samples that have better performances in such measurements, thereby helping users reduce uncertainty and obtain confidence in choosing the best answers. In addition, improved visual accuracy confirms that unbiased-guarantee incremental visualizations can take fewer sample points to provide visual answers equivalent to the exact answers. This is crucial to incremental visualization, because users probably use the visualizations presented in the first few visual updates to check whether the data selection conditions in a query is correct or not [3]. Visualizations created from biased samples can mislead users that they issued wrong queries and need to do some correction, whereas the query specifications are correct. As a result, users' mental efforts to use incremental visualization for data exploration and decision-making significantly increase.

Specific to **geospatial accuracy**, `STULL` is affected only by sample size [11]. However, `STORM` has one more factor, intrinsic spatial distributions in the data. In our experiments, in spatially clustered distribution cases (e.g., GEO), compared to `STULL`, `STORM` has light or indiscernible visual differences in hotspot areas, but is incompetent in lower-density areas. On the other hand, in a scattered distribution case (e.g., OSP), the absence of an unbiased guarantee causes `STORM` to extend defective visual appearances to hotspots, e.g., incorrect hotspot locations. The RMSE value (Figure 4.4) in the OSP case is almost ten times that of the concentrated cases. So is the visual effect in which `STORM` defectively represented in wider spatial extents in the OSP case but misrepresented merely sparse ones in the concentrated scenes. We believe `STORM`'s reduced accuracy loss in spatially concentrated cases is caused by the fixed-size sample buffer. First, `STORM`'s spatial index, `R-Tree` tends to create more cells in high-density areas, and fewer cells in low-density areas. Accordingly, in a spatially concentrated case, a majority of cells are associated with scarce hotspot regions. Since in the fixed-sized buffer design, the number of points sampled in a region

is proportional to the number of its cells, STORM can retrieve more points in order to characterize hotspot areas but pay less attention to lower-density areas.

As for **temporal unbiased property**, STULL achieves it through determining the number of samples retrieved from each temporal bin proportional to the total data volume in the bin, which is a widely used golden rule [69, 71]. Thus, we do not elaborate on it.

Our range of experiments also validate the efficiency of STULL's sample retrieval. Experiments (Figure 4.8, Figure 4.9, Figure 4.10) indicate that compared to RandomPath, STULL can reduce latency by at least 60% to sample 5% of in-memory data per incremental update despite query specification at various geospatial and temporal scales. As to the case where data indexes are on disk drives, **STULL** and STORM load points from buffers of root cells first and continue retrieval from buffers of its descendants. This significantly reduces the number of STULL's disk I/Os needed for the first visual update, whereas RandomPath needs to retrieve points from most of its leaf cells, consequently forcing almost every cell to be loaded into memory, which results in an extremely slow response. Thus, RandomPath does not satisfy the critical latency.

Our experiments show that users are able to keep **computational latency per incremental update** well under control. Figure 4.11 shows that STULL successfully controls retrieval time proportional to the number of points per visual update. In addition, our experiments sequentially accessed temporal bins to obtain samples, which resulted in higher latency compared to an in-parallel scheme. We leave STULL's adoption of parallel computing techniques to future work.

STULL **reduces the data index creation and update workload**, compared to STORM. STULL indexes data spatially using pyramids for efficiency [110]. We conducted experiments to measure index creation time and confirmed the superiority of our pyramid-based approach, which is approximately 10% faster than the R-tree based STORM. Regarding streaming data, Table 4.3 shows that it takes less than 450ms to index 5000 points. Thus, inserting the new data into the existing data index and re-

trieving samples from the updated data index can be completed in approximately less than 500 ms for the OSP, Tweet-Chicago and Tweet-US datasets and approximately 600ms for the GEO.

STULL is designed for aggregation-based **spatiotemporal analytics** that assist end-users in summarizing trends and patterns of data, e.g., grouping data into hourly-divided time units. Here, we demonstrate aggregation computation with samples retrieved by STULL and use confidence intervals [11, 12] to estimate the proximity of generated approximate answers to exact answers. Suppose a query calculates the average length of tweets posted in the morning. A sample S of n tweets is retrieved by STULL. The average length of tweets is $\bar{v} = \frac{1}{n}\sum_{s_i \in S} v_i$, v_i is the length of the i-th tweet. Let c denotes the standard deviation of the sample estimate. Thus, the interval $[v - 2c, v + 2c]$ contains the exact answer with 95% of the time.

Despite efficient sampling, STULL suffers from **inefficient usage of storage space**. The bottom levels of STULL's pyramids contain all data points, in addition to $(1-\alpha)$ portion of data in non-bottom levels. Thus, STULL maximizes retrieval efficiency at the expense of data storage space. In the case where higher-level data are not duplicated at the bottom level, data retrieval will move from the bottom level to the higher levels, and sample data from these higher levels. Since higher levels consist of cells whose spatiotemporal ranges are quadratically larger than the queried range, we could anticipate a 3-fold increase in retrieval time. Although the duplicated data will take additional storage resources, our design choice has at least two benefits. First, STULL restricts the retrieval to a minimum set of spatially relevant data, per Q. Secondly, when the data index is on disk, retrieving spatially irrelevant data in the alternative option will cause more disk I/Os.

Another limitation of STULL is that it does not yet fully investigate a **disk-based index**. As data volume increases, in-memory data storage becomes scarce. A hybrid index of both in-memory and disk-resident data is essential to overcome the memory shortage [60]. For convenient disk-based data storage and retrieval, STORM [1] uses the fixed-sized design and sets the space usage of a sample buffer as equivalent to the

size of a disk block, resulting in that each cell has the same number of data cached in its sample buffer. Thus, if one sample buffer is needed, `STORM` loads the corresponding disk block into memory, retrieves all data stored in that block, and removes the block from memory after use. But in `STULL`, the proportionally-sized design causes sample buffers to have various sizes. Consequently, it is common for one sample buffer to involve multiple blocks, with one of these blocks only partially filled. These partially filled blocks cause the low disk storage utilization concern and slow down disk I/O as well. We leave this to future work.

4.7 Conclusion and Future Work

This chapter presents an online sampling approach, `STULL`, which samples large spatiotemporal data in an unbiased manner. Extensive evaluations verify that `STULL` is unbiased and computationally superior over comparable online sampling approaches. `STULL` is suitable for a range of online data exploration including visual analytics and incremental visualization. Approximate visualizations leveraged by `STULL` increase their numerical accuracies and reduce their visual differences as compared to the exact visualizations, when compared to approaches without unbiased guarantee.

In the future, we will extend this work by designing a novel scheme to store our data index on hard drives. The current implementation has comparable performance for retrieving a small ratio of points, about 10% in our experiment, from a disk-resident data index, but is slower if more points are wanted. A novel scheme is expected to solve this issue.

Algorithm 2: Retrieving samples in Temporal Bin t

input : Query Q; θ

output: A random sample S with 100θ percent of points

```
1  Determine l_Q according to the spatial query range of Q;
2  Initialize empty lists S and G;
3  l_r ← a level randomly chosen between 1 and 1/α;
4  l ← l_r;
5  u ← 1;
6  while (u <= 1/θ or users didn't terminate) do
7      if G is empty then
8          if l_0 ≥ l_r and l < 1/α then
9              G ← sample buffers of cells that are in the l-th level and spatially
                 overlapping with Q;
10         else
11             G ← the l-th segments of cells that are in the leaf level and spatially
                 overlapping with Q;
12     u_0 ← (u mod α/θ) == 0? α/θ : u mod α/θ;
13     for each element b in G do
14         s ← points satisfying Q and in the [100(u_0 − 1)θ/α%, 100u_0θ/α%]
             portion of b;
15         S ← S ∪ s;
16         Send S to a VA system for visualization;
17     u ← u + 1;
18     if u mod α/θ == 0 then
19         G ← an empty list;
20         l ← 1 + (l mod 1/α) ;
21         if l == l_0 then
22             break;
```

Table 4.1.
Evaluated datasets.

Data	Description	Counts (million)	Memory size (MB)	Spatial range	Temporal bin counts	Temporal bin interval
GEO [118–120]	human movement data from April, 2011 to August, 2013	5.8	3289	Beijing, CHINA	3	year
OSP	Ohio traffic incident data from January 1, 2012 to December 31, 2013	3.2	2279	Ohio, USA	4	6 months
Tweet-Chicago	tweets in Chicago from April 1, 2013 to September 30, 2013	9.4	4452	Chicago, IL, USA	6	month
Tweet-US	tweets across the entire US from January 1, 2018 to March 11, 2018	12.4	4879	USA	11	week

Table 4.2.
Time measurements (in seconds) using `STULL` to index data. Results are averaged over five runs.

	GEO	OSP	Tweet-Chicago	Tweet-US
$\alpha = 0.250$	76.142	58.461	181.493	163.234
$\alpha = 0.125$	172.233	206.146	243.323	227.889

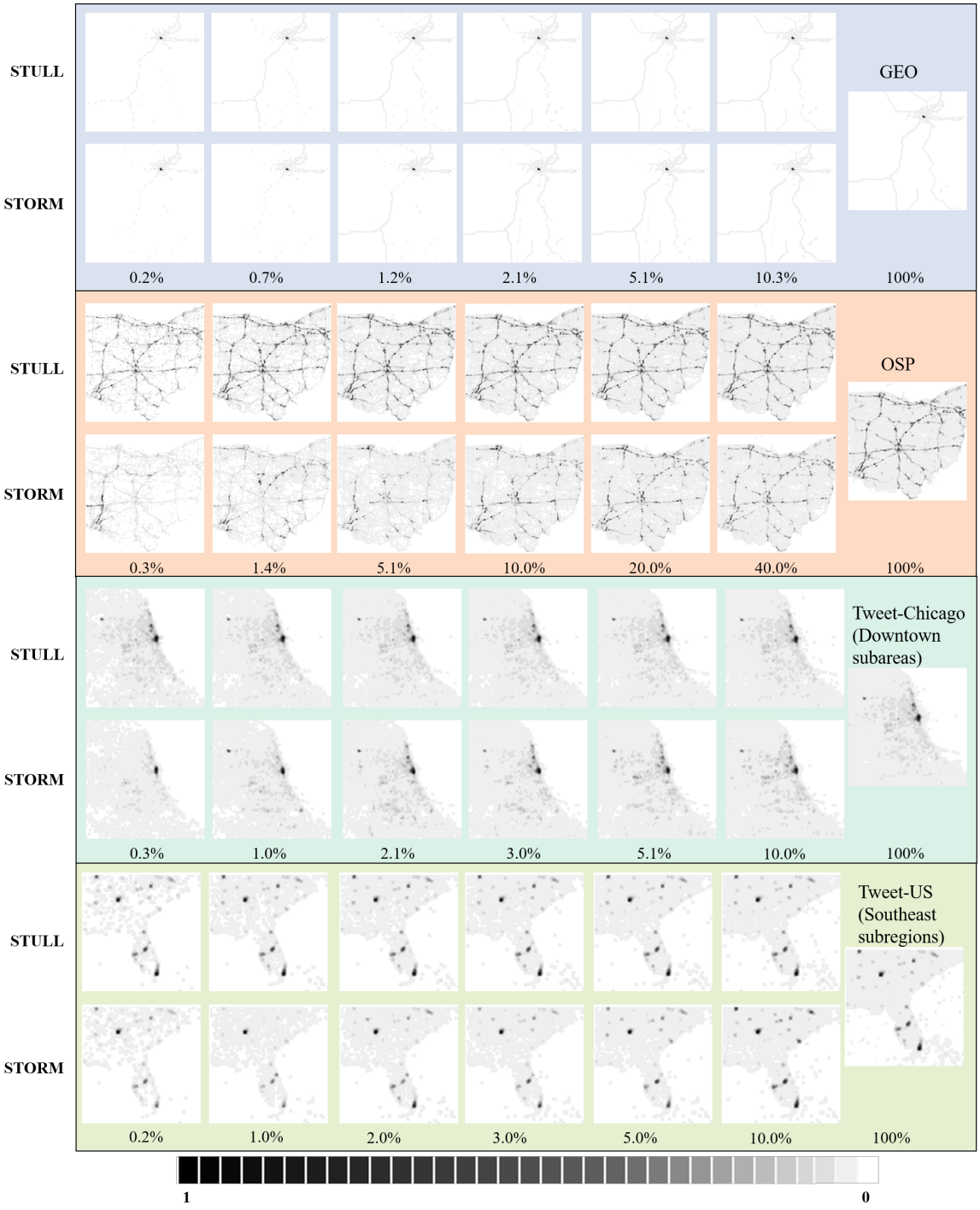

Fig. 4.6. Comparison of spatial heatmaps generated by the two approaches. A number below a heatmap indicates number of sample points selected for approximate distributions. At the bottom is the gray-scale colormap with 32 shades.

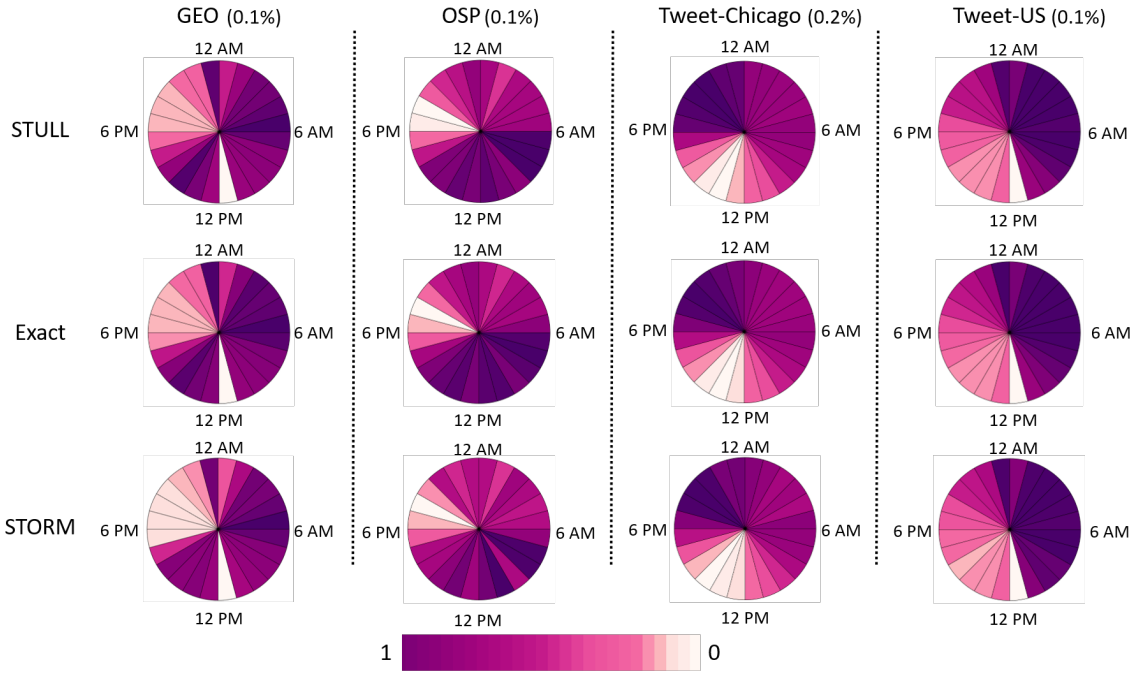

Fig. 4.7. Pie charts showing normalized hourly distribution of the entire data. Each slice denotes a hour. These approximate charts are generated with 0.1% points of being selected. The color legend has 32 color shades. The result is one-time run.

Table 4.3.
Time measurements (in milliseconds) for `STULL` to insert 5000 points into the existing data index. Results are average over five runs.

	OSP	GEO	Tweet-Chicago	Tweet-US
$\alpha = 0.250$	153.103	218.732	268.735	190.614
$\alpha = 0.125$	203.089	260.397	346.852	334.355

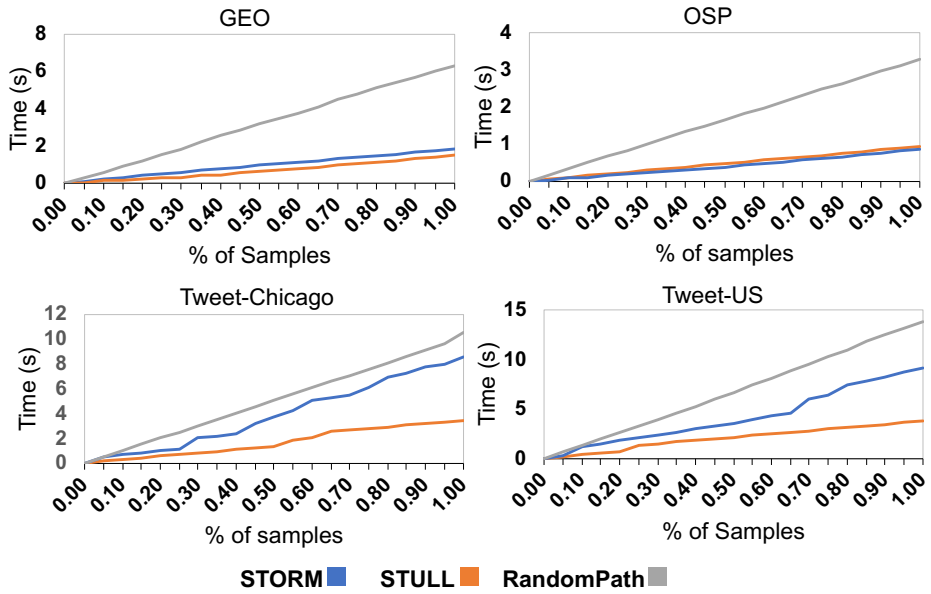

Fig. 4.8. Time measurements (in seconds) to retrieve samples in an in-memory setting. The query requires the entire data. `STULL` has $\alpha = 0.25$, and `RandomPath` has at most 4 levels in each of its Quadtrees. Results are average over 5 runs.

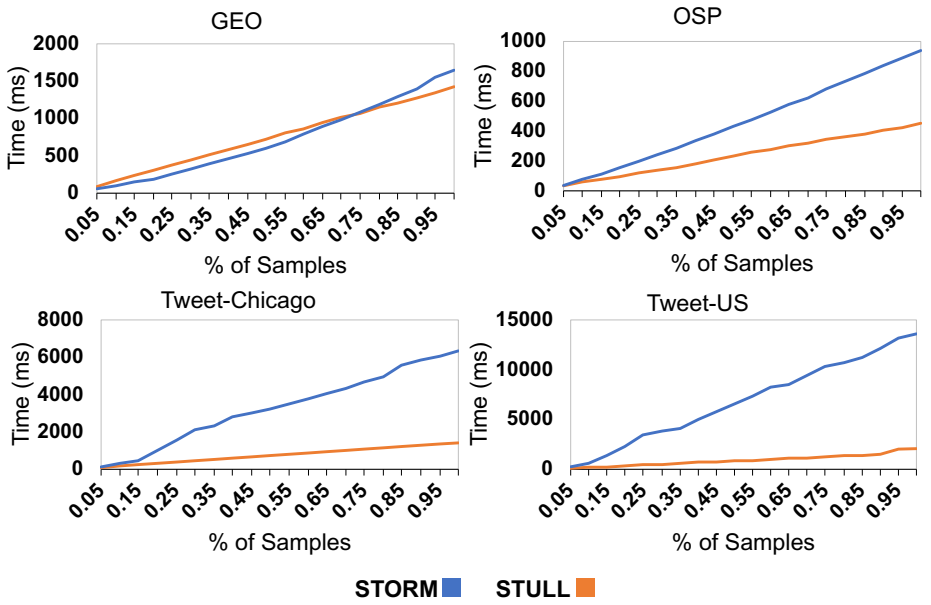

Fig. 4.9. Time measurements (in milliseconds) to retrieve samples from an in-memory data index. Queried temporal ranges are, 2012 for OSP, 2011-2012 for GEO, 2013/04-2013/06 for Tweet-Chicago, and 2018/01/01-2018/02/04 for Tweet-US. `STULL` has $\alpha = 0.25$. Results are average over five runs.

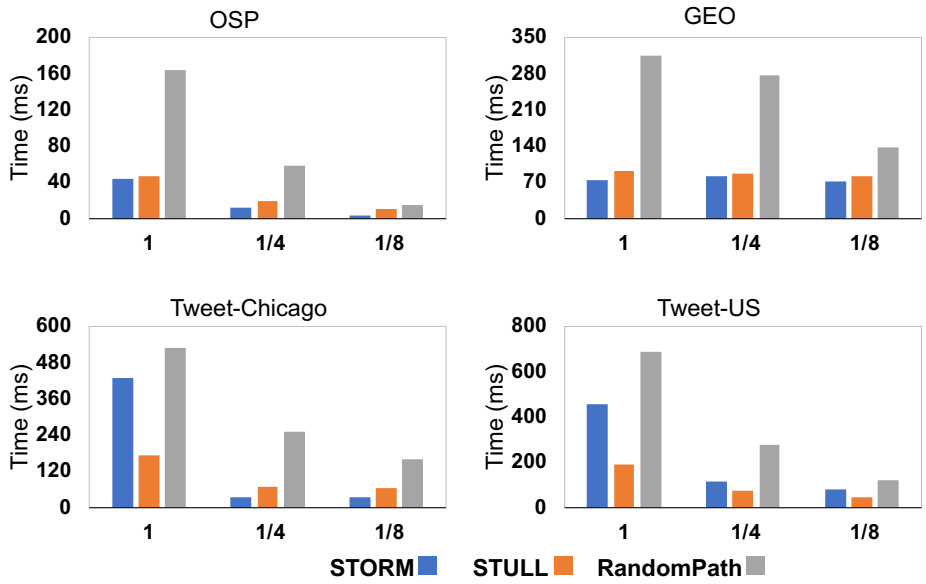

Fig. 4.10. Average time per incremental update. Each incremental update retrieved 5% points. Numbers below a bar indicate queried spatial range, 1 for the whole spatial extent, 1/4 for a quarter of the whole extent, and 1/8 for a one-eighth. For `STULL`, $\alpha = 0.25$. Each of `RandomPath`'s Quad-trees has at most 4 levels.

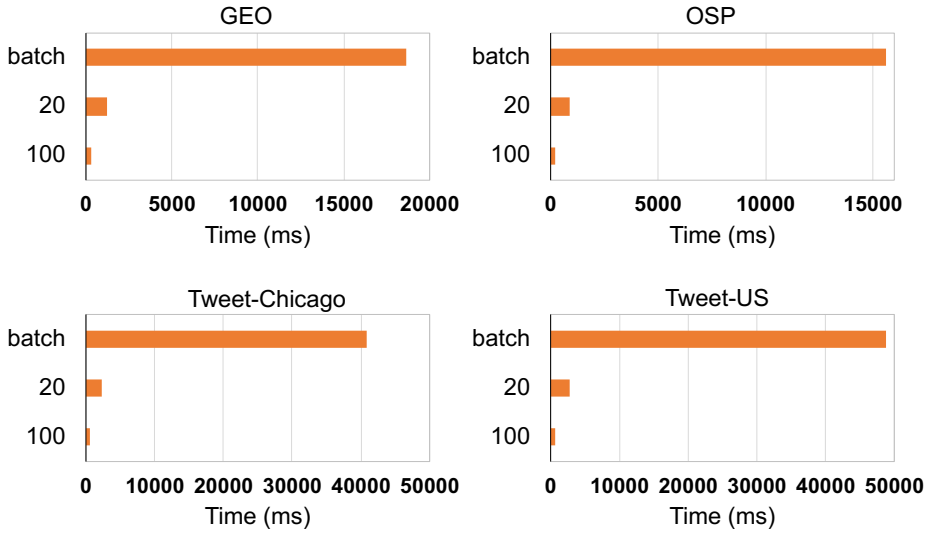

Fig. 4.11. Averaged latency per incremental update with different numbers of points. The query required the entire data. In the y-axis, batch indicates time to retrieve the entire dataset, 20 indicates incremental visualization has 20 updates in total and retrieves 5% point per update; likewise, 100 indicates 1% per update and 100 updates in total. For `STULL`, $\alpha = 0.25$. Results are average over 5 runs.

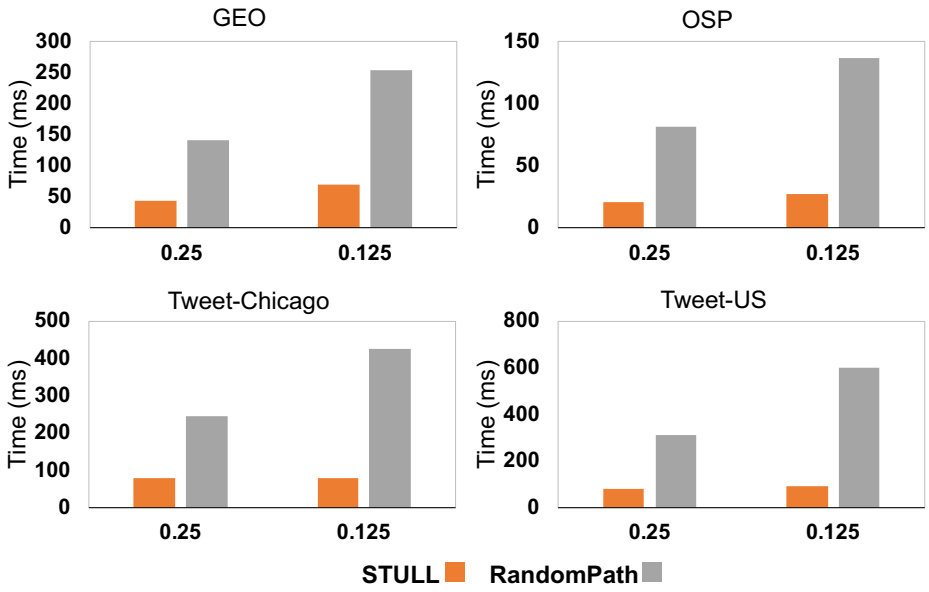

Fig. 4.12. Averaged sample retrieval latency per incremental update. Each incremental update retrieved 2.5% points. 0.25 indicates that α of `STULL` is 0.25, and correspondingly `RandomPath`'s Quad-tree index has no more than 4 levels. Likewise, 0.125 indicates $\alpha = 0.125$, and a Quad-tree index has at most 8 levels. Results are average over 3 runs.

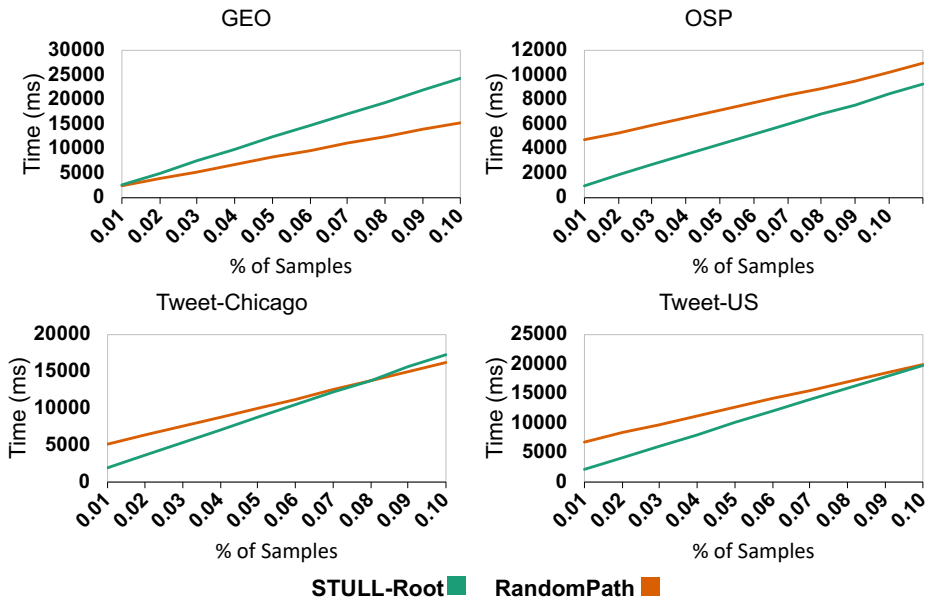

Fig. 4.13. Latency to retrieve samples from disk-resident indexes for a query requiring the entire data. Each incremental update obtains 5% points. STULL-Root refers `STULL` started retrieval from the pyramid root in each temporal bin. For `STULL`, $\alpha = 0.25$. Each of `RandomPath`'s Quad-trees has at most 4 levels. Results are average over 3 runs.

5. PERCEPTION-AWARE DIFFERENCE ASSESSMENT FOR SPATIAL HEATMAPS

Incremental data visualization and analytics provides quick responses to user queries and progressively improves the accuracy of these quick answers. Users need to evaluate these answers and select usable ones for their analytic tasks. This chapter focuses on user uncertainty issues about choosing satisfactory answers to the best decision-makings. We define an appropriate assessment methodology to identify perceptually more accurate spatial heatmap from a sequence of incremental updates. Compared to traditional approaches, e.g., mean squared differences [112], our approach measures differences between approximate and exact heatmaps in terms of human perception. Differences quantified by our approach are regarded to be congruous with human-perceived results [123], so that users can interpret differences more easily and choose heatmaps users observe to be identical to the exact answers. Section 5.1 states the importance of perception-aware difference assessment, Section 5.2 describes human perception effects on spatial heatmap comparison, Section 5.3 shows our perception-aware assessment rationale, Section 5.4 follows the rationale to derive a visual difference detection function, Section 5.5 and Section 5.6 evaluate and discuss our approach respectively, and Section 5.7 summarizes this chapter.

5.1 Background

In the big data era, a proliferation of studies have been conducted on the ways that users can interactively explore data through incremental visualization. Usually, incremental visualization techniques employ sampling-based approximate query processing

(AQP) techniques to provide users immediate but approximate visual feedback. Such visualizations progressively sample more data and refine answers; accordingly, the trade-off between computational time and accuracy relies on end users. Users expect to choose approximate answers satisfying accuracy demands imposed by their analytic tasks. Therefore, assessing the accuracy of approximate visualizations is vital in order for users to choose trustworthy answers.

Existing visual analytics systems typically [3,4] provide users with statistical measures (e.g., confidence intervals [12] and mean squared errors [112]) for estimating whether the presented results satisfy the accuracy requirements of the analytic tasks at hand. These measurements evaluate visual elements equally to all other components; however, visualizations are the primary medium users rely on to evaluate answers, and humans perceive visual elements in a nonuniform way.

The human visual system (HVS) treats visual stimuli (e.g., graphics) unevenly. First, empirical human perception experiments show that humans are insensitive to minor changes of colors [124]. Visual elements with indiscernible appearances but encoding different information can be incorrectly interpreted as identical. Consequently, the process of extracting knowledge from visual information comes at the cost of some information loss [15,107]. Second, humans process visual information at different speeds. For example, humans can pre-attentively identify a red dot among many gray ones, but require focused attention to distinguish targets with similar colors. In the context of incremental visualization, users are mostly attracted by animated preattentive visual elements and perceive peripheral elements the least. Therefore, users have difficulty precisely connecting the perceived visual differences with accuracies measured by traditional measurements which treat all visual elements equally [5].

In order to address this inconsistency between visual differences perceived by humans and the accuracy measurements used for data items, we propose a perception-aware difference assessment that compares spatial heatmaps in terms of human perception.

5.2 Human Perception of Approximate Spatial Heatmaps

A spatial heatmap [25] is a typical visualization used to characterize data density variations across a spatial extent. Usually, such visualizations divide the entire spatial extent into a grid and then use Kernel Density Estimation [113] to estimate densities on the basis of rectangular spatial divisions. Carefully designed colormaps transform density values into colors so that users can intuitively observe hotspots and low-density areas. Figure 5.1 illustrates information encoded by spatial heatmaps.

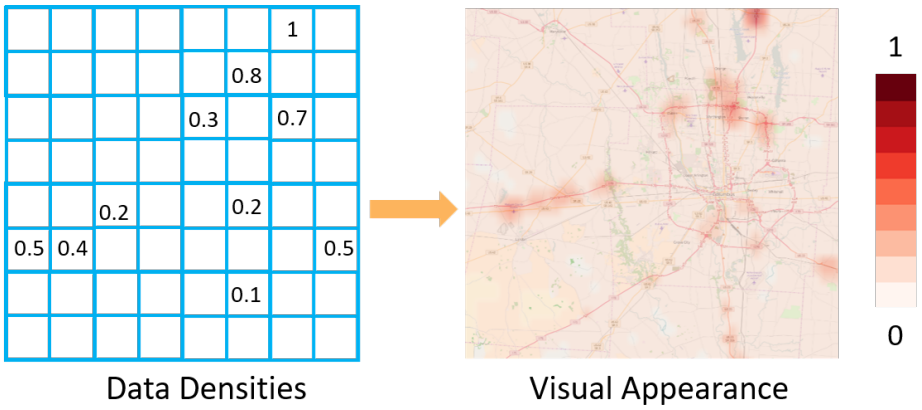

Fig. 5.1. A pictorial description of information represented by a spatial heatmap. Numbers in the bins denote data densities in the same rectangular regions. These numbers are approximate, used for demonstration, and are not the real ones generating the heatmap on the right side.

Spatial data distributions are usually nonuniform and sparse, with data highly concentrated in a few regions and widely scattered across most areas (e.g., criminal incidents often concentrate in specific spatial regions [22]). Consequently, most areas are rendered on a heatmap by colors encoding lower densities, and a few areas are rendered by a sub-range of colors that are easy to observe. Therefore, the heatmap preattentively represents **hotspots** (high-density locations), thereby, enabling observers to notice hotspots immediately (less than 250ms [125]) and requiring focused attention to see sparse areas.

Incremental visualization use online sampling techniques to create approximate heatmaps. Online sampling approaches follow spatial distributions to fetch samples from each region. The number of selected points per region is in proportion to the region's data density. However, subject to the randomness of sampling approaches, data samples can approximate the exact data distribution but fail to visually represent it exactly. As a result, differences always exist between approximate heatmaps and exact ones. Due to the intrinsic sparsity in the geospace, low-density bins contribute to the majority of density differences, whereas hotspots take up a smaller ratio. Since traditional accuracy measurements consider each bin equally, their measures reflect differences in the low-density areas the most. However, when users look at approximate heatmaps and exact ones together, hotspots are the primary elements catching users' attention, and their differences play a key role in their judgments.

5.3 Visual Difference Assessment Rationale

In this section we present a perception-aware rationale for identifying visual elements that humans perceive to be different when simultaneously looking at an approximate heatmap and its exact counterpart. Visual elements in a spatial heatmap include individual pixels, sub-regions, and hotspots. The rationale includes the following four criteria.

R1 Color difference: For two heatmaps placed in juxtaposition, the colors in the same positions are compared for differences (Figure 5.2). Visual elements with color differences that cannot perceived by humans are regarded to have no differences.

R2 Spatial frequency: Heatmaps use colors to encode data densities, assigning distinguishable colors per density level. Accordingly, visual contrast between high-density points and their low-density surroundings should be clearly perceivable. However, such observations become vague when a heatmap has high-frequency density variation in the geospace (Figure 5.3). If the contrast occurs

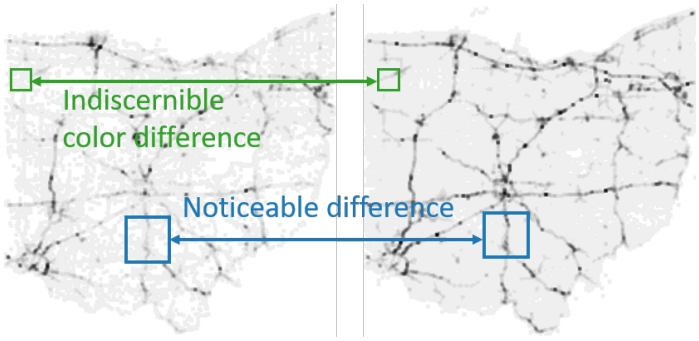

Fig. 5.2. A picture illustrating color differences between an approximate spatial heatmap (leaf) and the exact one (right).

in high frequencies, e.g., multiple hotspots aggregate in close proximity, it is not easy for humans to visually inspect individual hotspots in the moment. On the other hand, contrasts in low-frequency areas are easier to see. Therefore, visual elements located in high-frequency surroundings are difficult to compare.

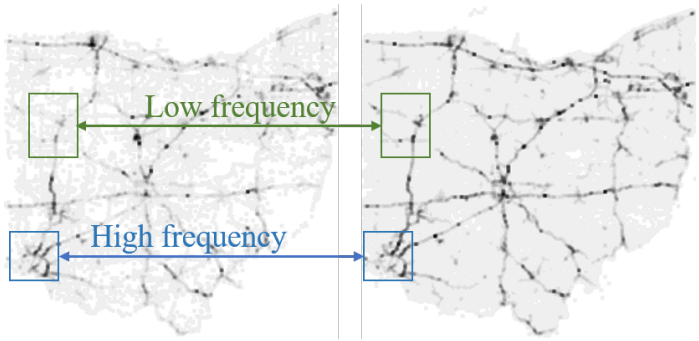

Fig. 5.3. An example demonstrating the impact of spatial frequencies on human perception. The left side is an approximate heatmap, and the rightmost is the exact one.

R3 Visual element size: In situations where approximate heatmaps are created from few data samples, bins in lower-density areas have no or insufficient sample points to depict. These bins are usually connected into components. For visual elements like connected components, their colors are perceived as dramatically different based on their sizes [102,103]. Thus, in two heatmaps showing the same

region (Figure 5.4), color differences become indiscernible if their associated polygons are not large. It is recommended to exclude such areas in perception-aware assessments.

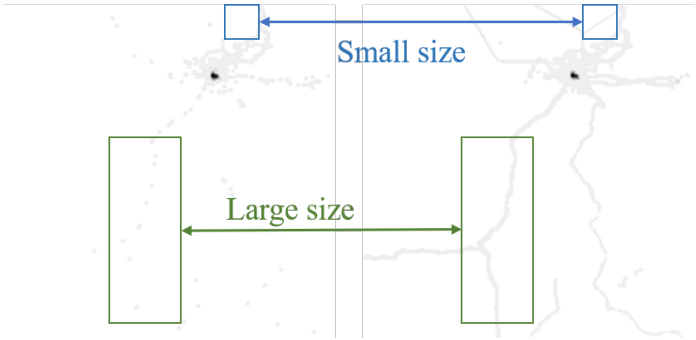

Fig. 5.4. An example demonstrating the size's impact on human perception. Compared to color differences, which are connected into noticeable components, differences in smaller components are indiscernible when users compare the two heatmaps at first glance.

R4 Hotspot importance: Heatmap preattentively present hotspots so that when comparing two heatmaps, hotspot differences will immediately catch a human's attention, whereas differences in lower-density areas require time-consuming focused attention. Therefore, hotspots play a larger role in the overall visual differences between approximate and exact heatmaps, especially in cases when approximate heatmaps update incrementally.

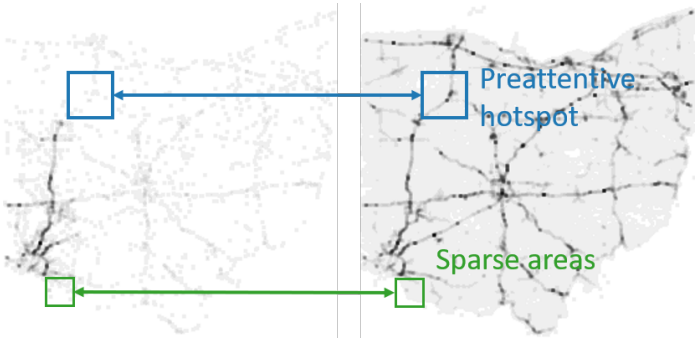

Fig. 5.5. A picture showing preattentive perception of hotspots.

5.4 Visual Difference Detection

Following the perception-aware rationale (Section 5.3), here we derive a perception-aware function in order to identify pixels whose visual differences can be perceived by humans in a side-by-side heatmap image comparison scenario. This function is developed in the RGB color space. Figure 5.6 shows our function's computational workflow. First, we extract the luminance components of the two and generate a luminance difference array. Second, from the luminance differences, we detect visible pixels and their connected components (Section 5.4.1). Next, we exclude individual pixels at high-frequency locations and keep the remaining pixels (Section 5.4.2). Finally, a union of pixels in the both second and third steps is perception-aware pixels. These pixels can serve accuracy assessment metrics (e.g., RMSE [112]) to generate perception-aware measures (Section 5.4.3).

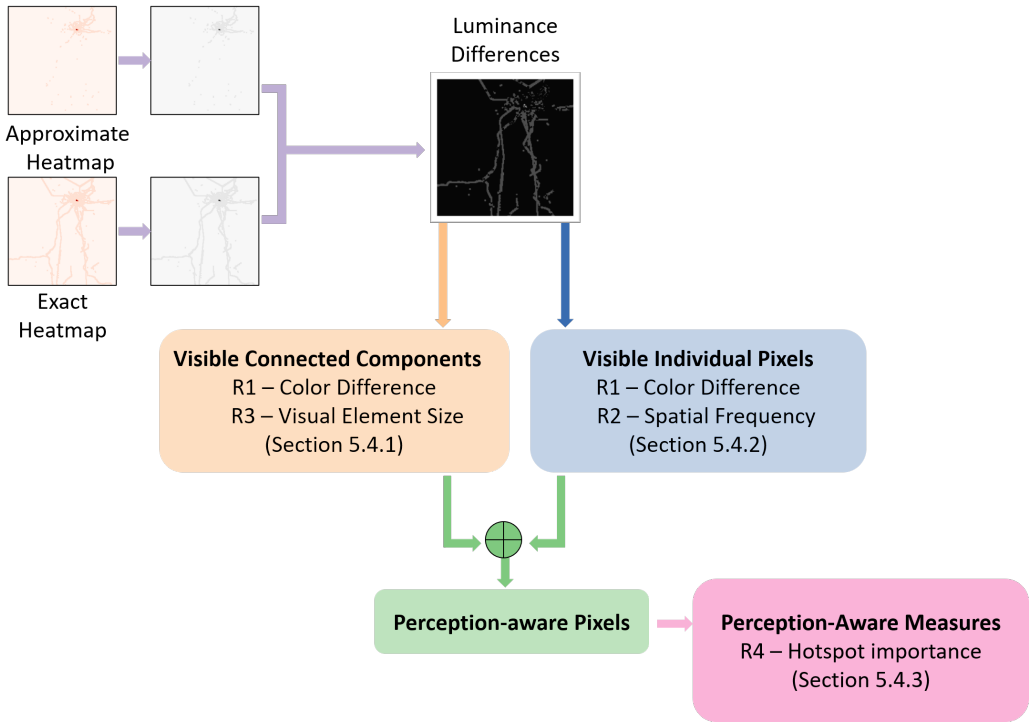

Fig. 5.6. Workflow for perception-aware difference detection and measures.

5.4.1 Detecting Visible Connected Components

Suppose A is an approximate heatmap, and R is the exact heatmap. R and A have the same resolution by $r_x \times r_y$. The following steps find the pixels whose luminance values reach the empirical thresholds of being visible in their luminance backgrounds.

1. Calculate L_R, a grayscale luminance array for R. So is L_A for A.
2. Calculate luminance differences $d = ||L_R - L_A||$.
3. Find A's visible pixels, whose luminance values are no less than d's JND thresholds. Equation 5.1 shows our applied JND formula developed by Chou and Li [98].

$$L_{\text{JND}} = \begin{cases} 17 \times (1 - \sqrt{B/127}) & \text{if } B < 127 \\ 3 \times (B - 127)/128 + 3 & \text{otherwise} \end{cases} \tag{5.1}$$

B is the averaged luminance of a pixel's background pixels

L_{JND} is the JND threshold in a background of B

From the visible pixels, we find the most connected components C and get the bounding boxes of these components. Components whose bounding-box areas have no less than a certain threshold will be regarded as discernible. We determine the threshold as an empirical size, that is, $\frac{r_x}{16} \times \frac{r_x}{16}$.

5.4.2 Finding Visible Individual Pixels

We adapt a JND approach modeled by Wu et al. [105] to our needs in order to determine a visibility score per pixel in terms of the spatial frequency and luminance contrast in its background surroundings. This approach emphasizes the impact of spatial frequency on human perception and regards pixels in the region boundaries with hard-to-perceive differences. We use this approach to process d (the luminance

differences in Section 5.4.1) and calculate the spatial masking per each pixel. Then we apply Equation 5.1 to L_A and obtain the contrast masking per each pixel. Calculations based on spatial masking and contrast mask derive the visibility score [105]. A higher score indicates lower visibility. Based on the obtained scores, we cluster pixels as invisible if their scores exceed a range calculated as follows:

1. Determine r, the ratio of background color pixels in the exact heatmap image
2. Determine p, the percentile of scores regarded as indiscernible in the approximate heatmap, $p = \max\{r + (100 - r)/10, 95\}$
3. Find the visibility threshold t, a score at the p-th percentile
4. Mark a pixel as visible if its score is no more than t.

5.4.3 Computing Perception-Aware Measures

Perception-aware pixels, P, are a union of visible connected components C (Section 5.4.1)) and individual visible pixels I (Section 5.4.2), $P = C \cup I$. Perception-aware measures (e.g., RMSE [112]) are conducted upon P.

Suppose $P = \{p_1, p_2, \cdots, p_n\}$ is the associated properties of n perception-aware pixels between an approximate heatmap and the exact one. $M(p_x)$ is a measurement for evaluating the approximate heatmap, where p_x is the property of the x-th pixel. The measurement result is $M(P) = \frac{1}{n} \sum_{i=1}^{n} M(p_i)$.

Considering the importance of hotspots (R4) for heatmap observations, we adjust $M(P)$ by the differences at hotspots. We define pixels owned by hotspots to be those whose colors encode the higher half of densities. Suppose H is the set of pixels locating at hotspot areas. The final perception-aware measure between an approximate heatmap and the exact one is $M(H)M(P)$.

5.5 Experiments

In this section, we report on experiments designed to verify our proposed perception-aware difference assessment approach. We used two sampling approaches to sample spatial data in the online manner, an unbiased approach [65] and a second approach without unbiased guarantees [1]. Incremental visualization of the continuously generated data samples produced a series of approximate spatial heatmaps with a 256×256 bin resolution (except a 256×512 resolution for the Tweet-US case). These heatmaps were saved as color images with the same resolution. Table 4.1 lists the test datasets. All experiments were implemented in MATLAB [126]. We quantified accuracy between approximate and exact answers through two measures: Root-Mean-Squared-Error (RMSE) [112] and structural similarity (SSIM) measurement [106, 127]. RMSE quantifies the bin-level Euclidean-based distances between approximate data densities and the exact ones. SSIM quantifies the pixel-level similarity between approximate and exact heatmaps, a widely-used image similarity measurement method. For each approximate heatmap, we detected its perception-aware pixels, and applied RMSE and SSIM to these pixels respectively. To easily compare between regular and perception-aware measures in the same figures, we normalized the hotspot differences ($M(H)$ in Section 5.4.3) into a range $[0, 1]$ in order to scale perception-aware results.

5.5.1 Accuracy of Approximate Spatial Heatmaps

Figure 5.7 and Figure 5.8 show approximate heatmaps with samples retrieved by the unbiased and biased sampling approaches respectively. We can see that when sample sizes are smaller, less than 1% in this experiment, the approximate heatmaps are visually discernible from the exact heatmaps in low-density areas. For hotspots, approximate heatmaps in the unbiased case show the same hotspot distributions with the exact, but the biased results are not. In both cases, approximate heatmaps using more than 50% data samples are harder to visually distinguish from exact ones. All of the observations are reflected by the quantified measurements in Figure 5.9.

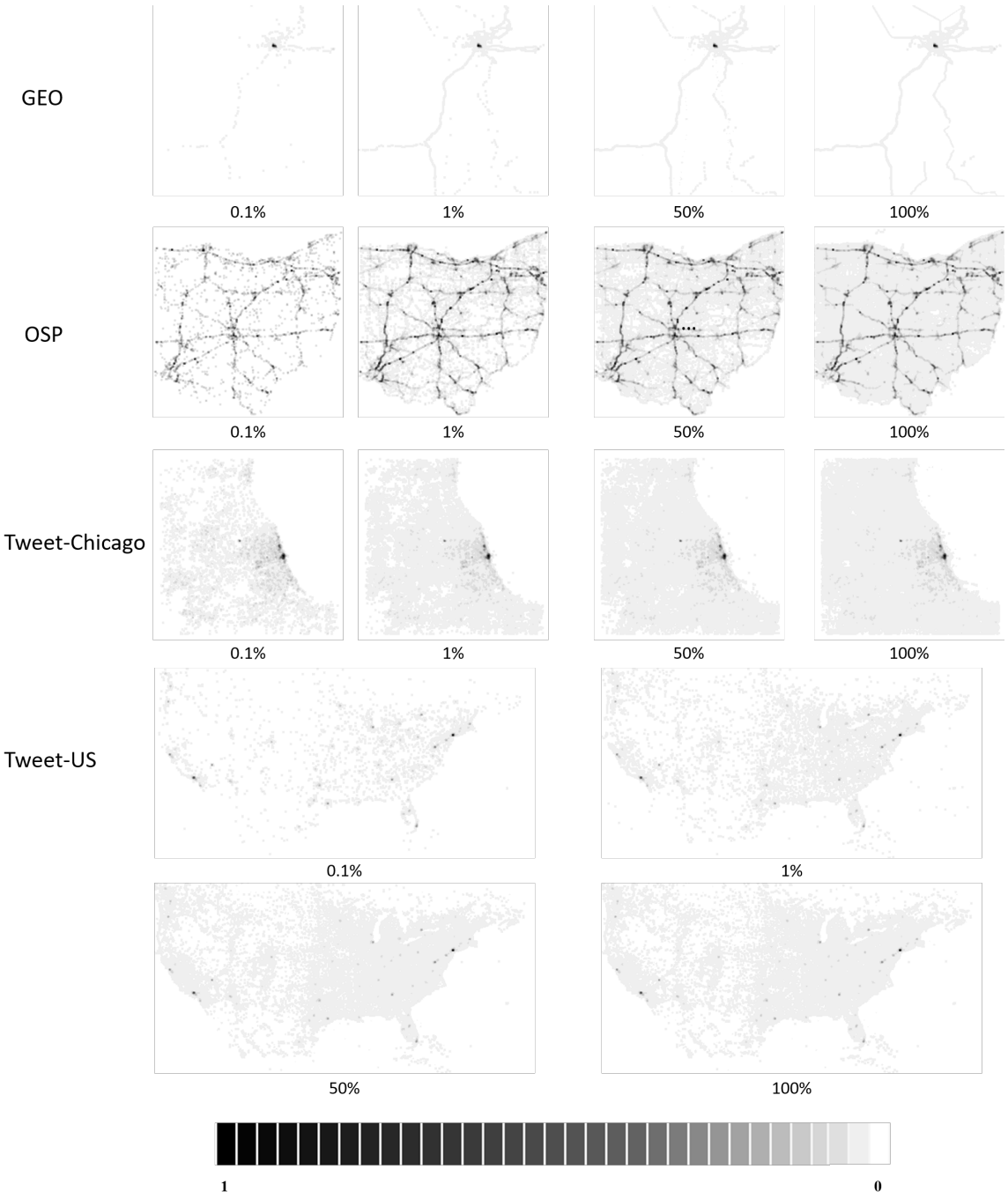

Fig. 5.7. Incremental visualization of spatial heatmaps created with different sizes of sample points retrieved by an unbiased sampling approach [65].

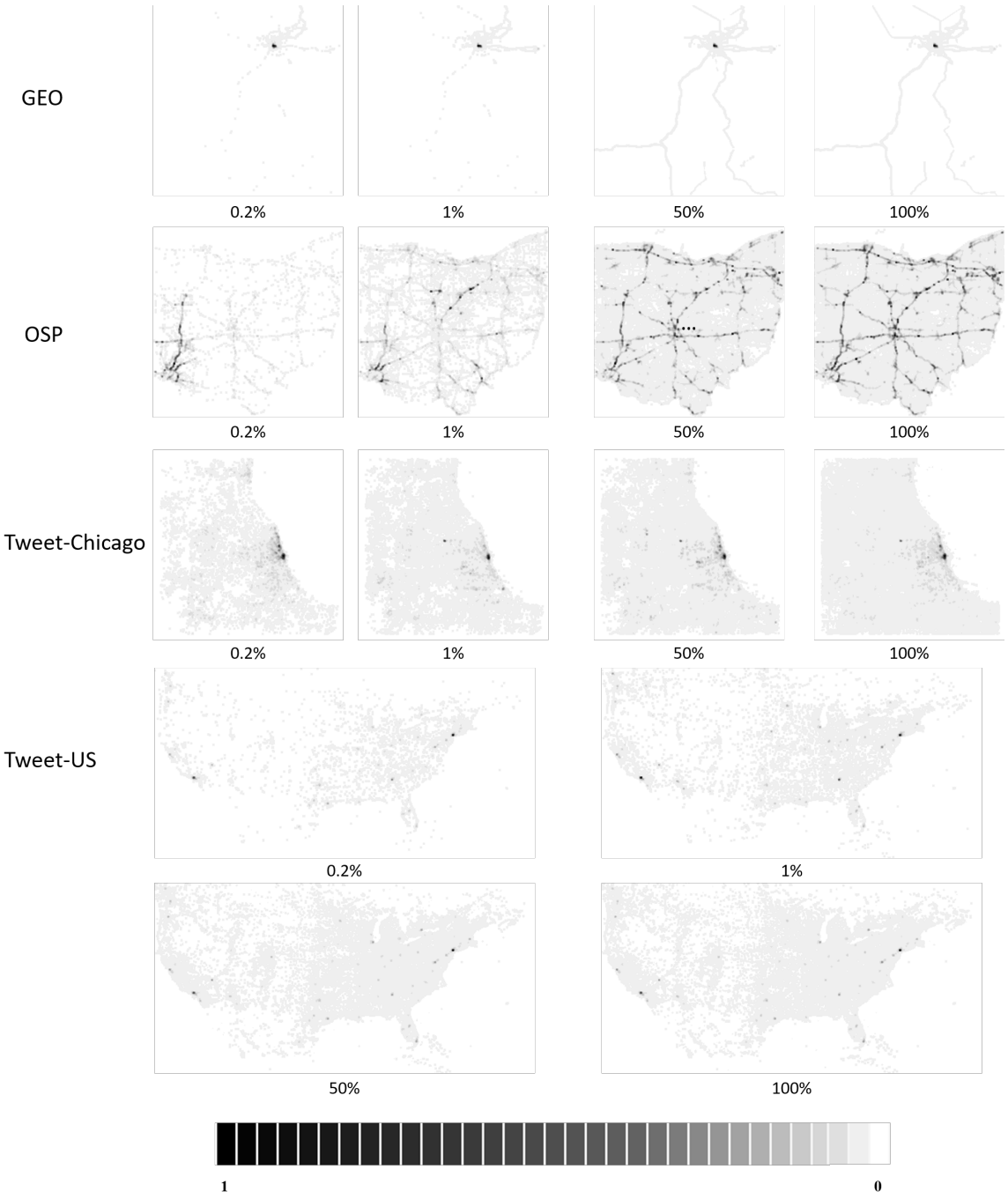

Fig. 5.8. Incremental visualization of spatial heatmaps created with different sizes of sample points retrieved by a sampling approach without unbiased guarantees [1].

First, both regular and perception-aware (PA) results in the RMSE/SSIM measures decrease quickly along with the increase of sample sizes, and the PA results decrease faster. Second, when sample sizes reach a particular number, PA errors reduce into a proximity to zero, but the regular results do not. This shows that after sample sizes reach a particular threshold, hotspots' approximation reaches a relatively high accuracy, and users are visually insensitive to their minor differences. Overall, the particular numbers in the unbiased cases are smaller than their counterparts in the biased cases, which indicates that at the same sample sizes, accuracies in the unbiased cases are higher than their counterparts in the biases cases. Last, the purple PA RMSE/SSIM errors decrease faster than the red measures that do not consider the impact of preattentive hotspots on human observations. In particular, the GEO perception-aware results in both the biased and unbiased cases are zero, which indicates that approximate values in the hotspot areas are almost identical to the exact. This shows that the answers in the hotspots converge to the exact ones more quickly than those in the low-density areas.

5.5.2 CIELAB-based Cross-Validation

Since our perception-aware function (Section 5.4) was conducted in the RGB color space, a cross-validation in the CIELAB space investigated one of our perception-aware rationales, R4 (Section 5.3). Compared to RGB, the color differences derived from CIELAB are closer to differences perceived by humans [99]. We chose S-CIELAB [104] to calculate the differences between a pixel in the approximate images and its counterpart in the exact ones. Our experiment followed S-CIELAB's settings in which a CRT display showed images and the distance between the user and the screen was 18 inches to measure pixel-wise color differences. Figure 5.10 shows the comparison results. At the same sample sizes, biased results' color differences are almost ten times than the unbiased ones. In addition, for the unbiased sampling results, the OSP regular results reduce slower than its perception-aware

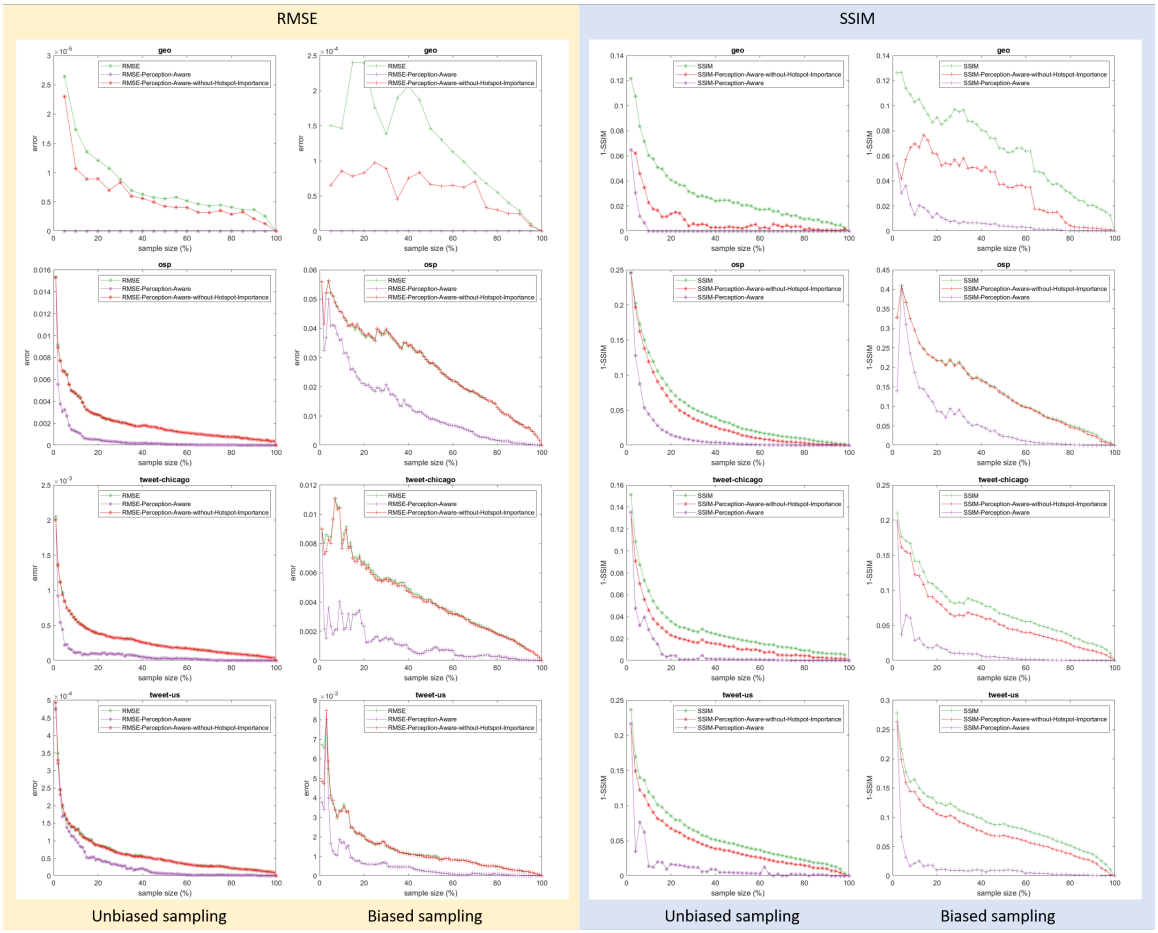

Fig. 5.9. Comparison of accuracies derived from perception-aware metrics and regular metrics. The experiment runs one time.

differences, whereas in the other three datasets the two measures represent exactly the same error trends. This is caused by the diverse spatial distributions in the four datasets. In the OSP case, hotspots scatter across the space; in the other three cases, hotspots are extremely concentrated in a few locations. Thus, during the incremental updates, hotspot distribution in the other three datasets are almost the same as the exact answers. This comparison confirms the importance of using preventative hotspots to assess heatmap differences. Likewise, the unbiased results clearly indicate the particular numbers. Heatmaps created by more samples than the particular numbers almost have no color differences from the exact ones.

5.5.3 Discernible Connected Component Detection

Figure 5.11 shows connected components in approximate spatial heatmaps, whose sizes are large enough for users to perceive differences when compared to the exact heatmaps. We used Matlab's bwconncomp [128] to find components connected by pixels that were visible in terms of Equation 5.1. Then we used Matlab's regionprops [129] to find the bounding-box areas of the detected components. The parameters for the regionprops function include 'Area', 'MajorAxisLength' and 'MinorAxisLength' [129]. We can see that those components locate in lower-density spaces, which are consistent with our observations of the heatmaps. In addition, these components shrink along with the increase in sample sizes, and eventually disappear.

5.6 Discussion

Our perception-aware approach has a presumptive assumption that as sample sizes increase, the visual differences between approximate and exact spatial heatmaps decrease and become indiscernible after a certain sample size is reached. Approximate visualizations using more data than that particular size look the same as the exact ones from the perspective of human perception. Our experiments support this assumption, because PA errors in Figure 5.9 converge to zero after a certain sample

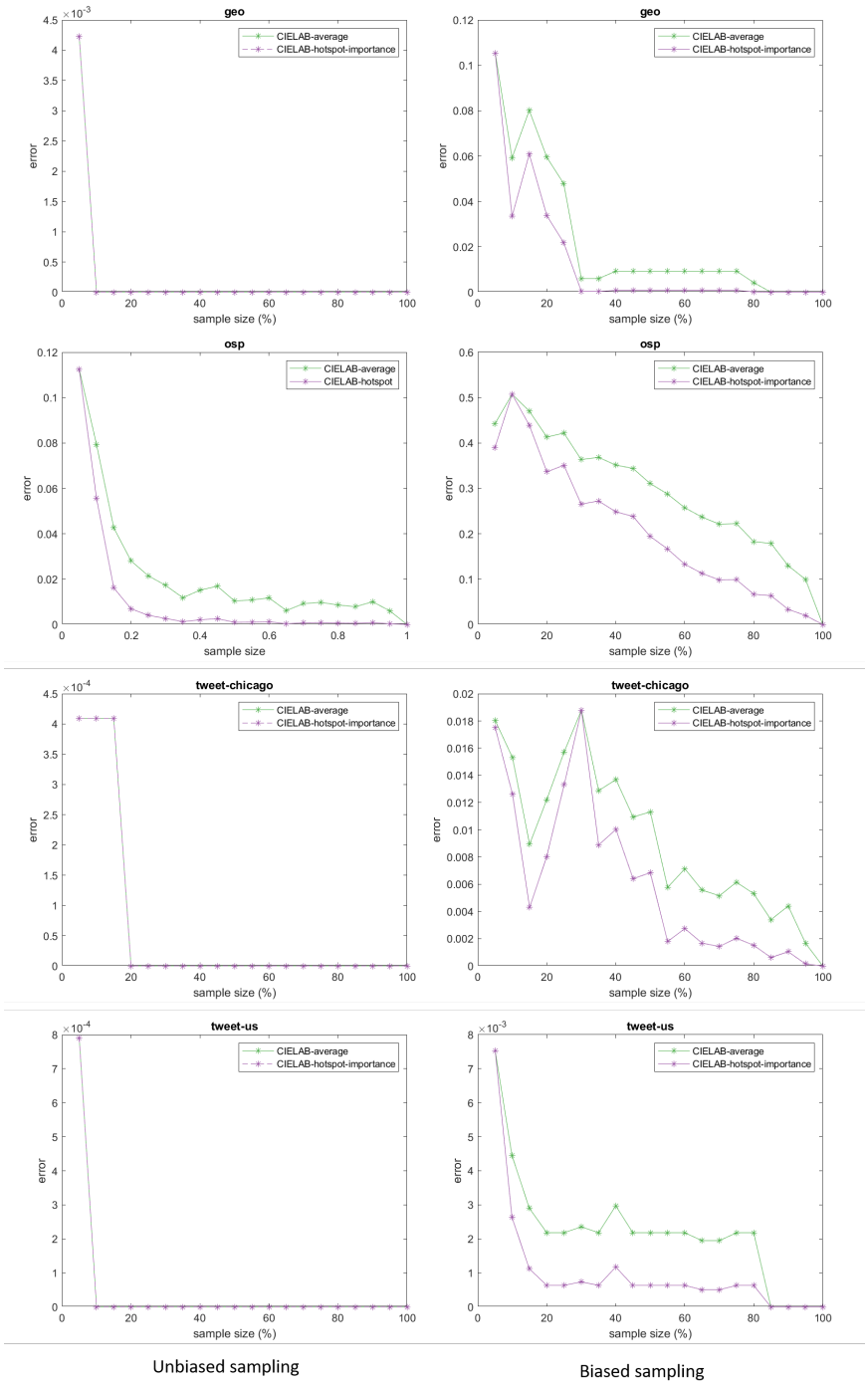

Fig. 5.10. Differences measured in the CIELAB color space between approximate spatial heatmap images and exact ones. Green lines show the average color differences in the CIELAB space per sample size, and purple lines show the average differences scaled by an average of S-CIELAB values at their hotspot regions.

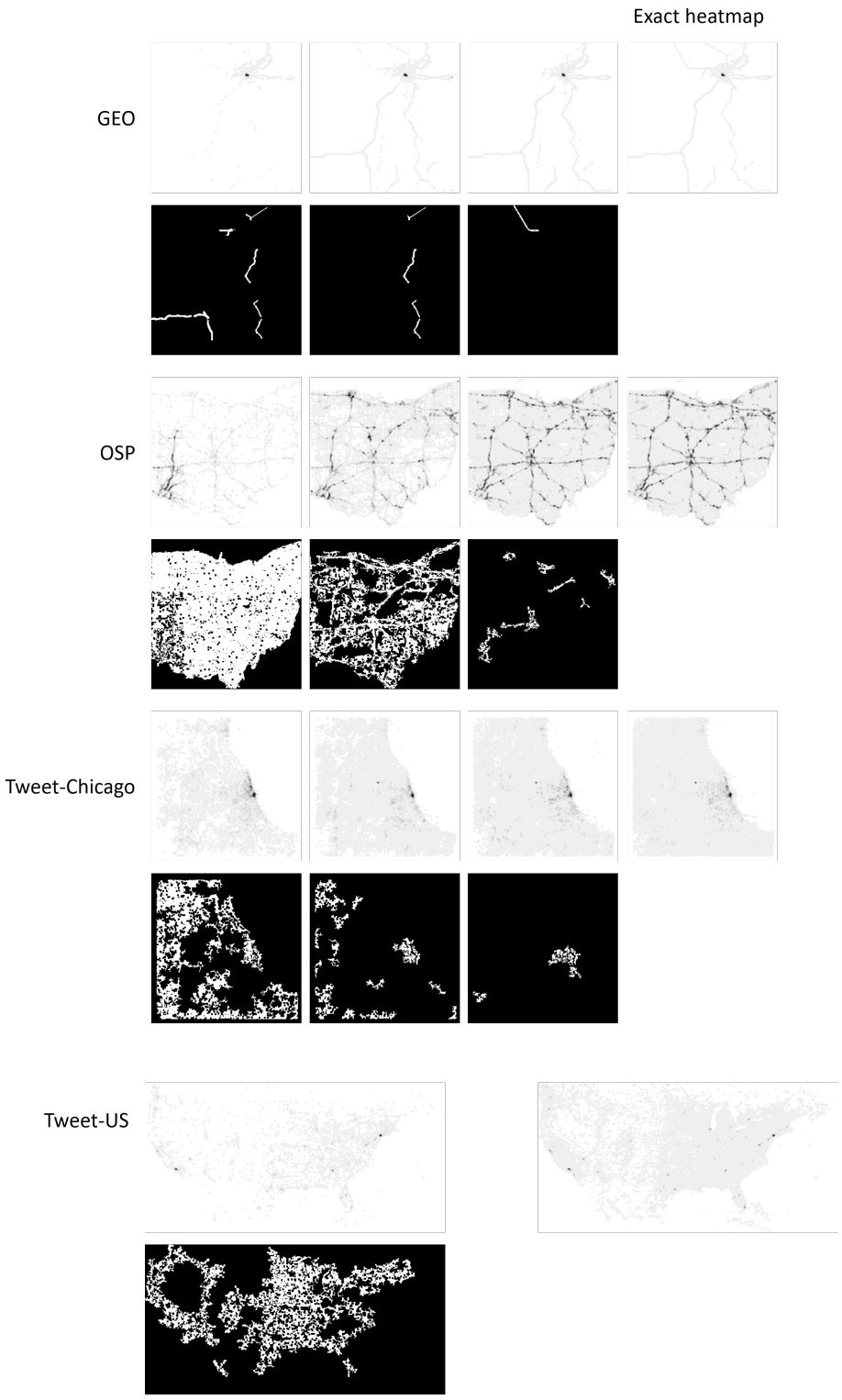

Fig. 5.11. Detection of regions whose color difference area sizes are large to perceive by humans. White indicates detected connected components.

size. We believe for the incrementally updated heatmaps, after a certain sample size reached, users are hardly to notice differences from subsequently displayed heatmaps. If users choose approximate answers whose associated perception-aware accuracies converge to zero, they reduce their uncertainty about the reliability of these answers. This will be investigated by a quantitative user study in the future.

The importance of hotspots (R4) is essential since the characteristics of random simple sampling (RSR) [11] require to emphasize hotspots (R4). RSR approaches randomly select points in a dataset, and therefore, high-density areas would have more points being selected than lower areas. Accordingly, approximate heatmaps have sufficient data to describe hotspots but require more for other areas. Therefore, the differences of hotspots reduce more quickly than those of low-density areas.

The characteristic of heatmaps emphasizes the importance of hotspots (R4) as well. Hotspots are preattentively present when compared to low-density areas. Users can perceive differences at hotspot areas quickly and completely, whereas differences at low-density areas require additional efforts to identify. Thus, from the perspective of human perception, hotspots are the primary visual elements used to assess approximate answers. However, regular measures (e.g., RMSE and SSIM) treat sparse areas and hotspots equally. Since the low-density areas take up a majority of bins or pixels, results gained from regular measurements significantly reduce the magnitude of hotspot differences. Therefore, discrepancies exist between human perception and regular measures. Such a discrepancy was confirmed by our experiments with the GEO, Tweet-Chicago, and Tweet-US datasets. Spatial distributions in the three datasets are highly concentrated, so in the incremental updates of their spatial heatmaps, perceived differences come mainly from low-density areas whose differences tend to be in a smaller scale. Our perception-aware results, shown in Figure 5.9 (the purple lines), converge to zero quicker, proving that visual appearances at the hotspot areas evolve to the exact ones quicker than the low-density areas. However, regular measures (the green lines), which ignore such information, decrease at a steady pace.

Regarding the CIELAB color space, our preliminary experiment verified the effectiveness of two of our four criteria, color difference and hotspot importance. However, the effectiveness of the other two criteria, spatial frequency and visual element size, is not yet investigated in this dissertation, as existing CIELAB-based color difference models [103, 104, 130] and our observations draw different conclusions about the differences among the colors encoding lower densities (KDE values smaller than 0.25) in our experiment. These models estimated the color differences to be indiscernible. For example, we used S-CIELAB to quantify pixel-wise color differences between an approximate heatmap image (e.g., created with 0.2% data) and the exact one. We used the median of these differences to indicate the visibility of the two heatmaps and found out that color differences in lower-density areas were calculated to be indiscernible. Similar results were obtained in another two CIELAB-based color difference models [103, 130]. As a result, we encountered difficulty in deriving formulas for the visual element size criterion, as we perceived the color differences in low-density areas. We think this contradiction comes from the fact that existing CIELAB-based models use assumptions that cannot effectively match the spatial distribution patterns and nuances of spatial heatmaps. For example, Danielle Szafir [130] built one color difference model in terms of color perception experiments conducted upon scatterplots. Nevertheless, the colors chosen to render points on scatterplots tend to have more significant differences to each other than the colors used in the heatmap case. Regarding the spatial frequency criterion, as the existing color difference models [72, 104, 130] evaluated only hotspot-region differences to be discernible, one appropriate way to measure the impact of spatial frequency is grounded in the number of hotspots in a local region of a heatmap. However, to the best of our knowledge, modeling human perception effects on point variations have not been well developed. As such, more investigation is needed before utilizing the spatial frequency and visual element size criteria to develop a CIELAB-based function (e.g., a quantitative user study to evaluate color difference parameters in heatmaps). Thus, in the future, we will extend our perception-aware approach to CIELAB measurements.

The foremost application that benefits from our proposed perception-aware approach is incremental visualization and analytics. Our perception-aware measures quantify visual differences perceived by users and represent the dynamic trend of these differences to users as the incremental workflow proceeds. Combining the trend and perceived visual element changes, users may figure out whether these changes improve approximate answers or not, and the improvement magnitude brought by these elements. Likewise, users can also predict visual elements' changes in the subsequent updates. Also, experiments reveal that heatmaps created by samples more than a certain volume become visually indiscernible. This particular number will be a good indicator for users to choose approximate answers that look accurate.

Our proposed perception-aware approach can serve a series of applications as well. First, the experiment in Figure 5.11 shows that connected components whose color differences are discernible are detected. This detection is able to guide a visualization system to select more samples from the regions that need to improve their visual appearances the most. In fact, adjusting the sampling process in terms of visualization needs is essential since spatial distribution is innately nonuniform [11]. Such a non-uniformity causes SRS approaches to fetch more samples from hotspots and fewer from sparse areas. As a result, the sampling requires more data to describe lower-density areas. Fixing such issues requires advanced sampling techniques, e.g., stratified sampling [11]. Stratified sampling divides the entire spatial range into a series of regions, and selects a proper amount of data from each division. We will extend our approach to facilitate stratified sampling for dividing the spatial range in the future. Another analytical scenario that is limited by scarce data in low-density areas is trajectory visualization [2]. Such visualizations suffer discontinuities in lower-density areas. As a result, users extrapolate no trajectories spread at such areas, which is incorrect. VAS is an approach solving such issues via a mathematical optimization solution. In the future, we will compare our approach with VAS thoroughly. Another application scenario is to compare sampling approaches, since the experiment in Figure 5.9 indicates the ability for our approach to compare unbiased and biased

sampling. In the figure, we can see that in both RMSE and SSIM measures, in a pairwise comparison the results between a perception-aware accuracy score and a perception-aware score without hotspot scaling are quite different. This infers that approximated heatmaps created through the biased approach contain inaccuracies in the hotspot regions, whereas those created using the unbiased approach do not.

5.7 Conclusion

In this chapter, we present a perception-aware approach to assessing visual differences between approximate and exact heatmaps. A perception-aware assessment rationale is introduced, and an RGB-based difference quantification function is presented as well. Our approach follows the characteristics of the human visual system in order to measure visual differences in terms of their influences on human perception. The differences derived from our approach are closer to the differences perceived by humans. In the future, we will extend our approach to the CIELAB color space, and conduct a quantitative user study.

6. CONCLUSION AND FUTURE WORK

In this dissertation, we have presented three approaches for users to incrementally visualize and analyze large spatiotemporal data. Our techniques focus on performance issues arising from the core of incremental analytical solutions, sampling-based approximate query processing (AQP). Three primary challenges include the absence of AQP modules in the computational environments, erroneous estimation due to sampling bias, and user concerns about the reliability of approximate answers [5]. Our techniques address these gaps, enabling users to explore data at interactive rates and reducing uncertainty in selecting best answers for their analytic tasks. We elaborate the contributions in the following three aspects:

- **A client-based visual analytics framework that enables visualization systems to conduct interactive spatiotemporal data exploration in the architectural constraint environments:** We focus on two types of constraints in the server-client architecture, including (1) insufficient computational resources on the client side (e.g., shortages of computer memory), which cannot afford to process the entire data at one time; and (2) organizational policies restricting the server side from deploying associated data processing techniques (e.g., AQP modules). We create a framework for visualization systems hosted on an average client machine in order to maintain interactive data exploration in this constrained environment. The client side assumes that data queried by users are uniformly distributed in the spatiotemporal extent, and accordingly follows this distribution to fetch sample points from the server side. On the other side, retrieved data can help the client side know more about the data distribution being queried, which enables samples fetched in the subsequent sampling procedure to better approximate the queried data. Extensive experiments have

validated the ability of our framework that can support visualization systems to sustain interactive data exploration in the architectural environment.

- **An online sampling approach that queries spatiotemporal data with an unbiased guarantee:** We created an online sampling approach that supports visualization systems in progressively retrieving data samples. Our technique presents a carefully designed data index and an associated sample retrieval plan to guarantee that each point satisfying the query will have the same probability of being selected. As such, samples retrieved by our approach can approximate the subset of the data being queried without bias. We validated that at the same sample size, the accuracy of the approximate answers constructed by our approach is greater than those without unbiased guarantees, improving by at least 50% when 5% points are sampled; the visual appearances of these approximate visualizations are perceived to be closer to the exact ones when compared to visualizations compromised by sampling bias. Therefore, incremental visualizations of spatiotemporal data leveraged by our approach can converge with the exact ones more quickly, which enables users to obtain trustworthy approximate answers in less time.

- **A difference assessment approach that compare approximate spatial heatmaps with exact ones in terms of human perception effects:** Incremental visualization continuously provides users with improved approximate answers and requires users to choose the best answers for their analytic tasks. In this workflow, users need to evaluate the accuracy of approximate answers in order to find trustworthy answers. We created a difference quantification approach that evaluates the accuracy of approximate spatial heatmap according to characteristics of the human visual systems. Our perception-aware approach identifies visual elements whose differences are discernible in a side-by-side comparison of approximate and exact spatial heatmaps. Spatial heatmap accuracy metrics measuring these detected perception-aware elements can reflect infor-

mation observed by users, whereas metrics that (e.g., RMSE [112], SSIM [127]) treat each heatmap elements equally cannot. Preliminary experiments indicate that our perception-aware accuracy measurements can evaluate approximate spatial heatmaps in accordance with empirical human perception rules.

We also discuss future work of this dissertation as follows. This future work is derived based on the deficiencies of our proposed techniques and extensive visual analytics needs suitable for our approach to address.

- **Reducing latency taken by our unbiased online sampling approach to retrieve samples from disk cache:** With an exponential increase of data, visualization systems cannot load the entire data into computer memory and must extend their storage space to hard drives. In such a scenario, samples are retrieved from in a hybrid structure consisting of a memory-based and a disk-based index together. Although we have demonstrated efficient sample retrieval in the memory space, reading and writing data to hard drives (disk I/O) produce a performance bottleneck, since their data transfer rate is significantly slower than memory-based rates. In the future, we will propose a novel sampling strategy that can efficiently retrieve data from disk caches as well.

- **Extending our perception-aware visual difference assessment to the CIELAB color space:** Our perception-aware difference assessment approach contains a perception-aware rationale that evaluates important visual properties of heatmaps from the perspective of human perception. Based on this rationale, we implemented a perception-aware detection function. This function follows empirical luminance adaption rules that were built on the RGB color space [98]. However, unlike the RGB space, color differences quantified in the CIELAB are closer to the actual differences perceived by humans. We anticipate that color differences measured in the CIELAB space will simulate the human visual system more accurately. In the future, we will extend our

perception-aware rationale to the CIELAB space and develop a CIELAB-based difference detection function.

- **Validating our perception-aware difference assessment approach by conducting a quantitative user study:** We followed empirical observations [99, 102, 103] to develop a perception-aware visual difference assessment rationale and its associated visual difference detection function. Our experiments showed that in the incremental update of spatial heatmaps, the accuracy of approximate heatmaps quantified from our detected perception-aware heatmap elements proceeded in a trend coherent with empirical human perception practices. However, these results only partially validate our approach, and a quantitative user study is essential to test whether our approach can be robust in the wild. In the future, we plan to recruit at least 20 participants to analyze data through incremental visualization of spatial heatmaps. A group of people will use our approach to choose the best answer for their questions, and a control group will have no such information. Performance Differences between the two groups will be statistically quantified.

REFERENCES

REFERENCES

[1] L. Wang, R. Christensen, F. Li, and K. Yi, "Spatial online sampling and aggregation," *Proceedings of the VLDB Endowment*, vol. 9, no. 3, pp. 84–95, Nov 2015. doi: 10.14778/2850583.2850584

[2] Y. Park, M. Cafarella, and B. Mozafari, "Visualization-aware sampling for very large databases," in *Proceedings of the IEEE Conference on Data Engineering*, 2016, pp. 755–766. doi: 10.1109/ICDE.2016.7498287

[3] D. Fisher, I. Popov, S. M. Drucker, and mc schraefel, "Trust me, I'm partially right: incremental visualization lets analysts explore large datasets faster," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2012, pp. 1673–1682. doi: 10.1145/2207676.2208294

[4] D. Moritz, D. Fisher, B. Ding, and C. Wang, "Trust, but verify: Optimistic visualizations of approximate queries for exploring big data," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2017, pp. 2904–2915. doi: 10.1145/3025453.3025456

[5] B. Mozafari, "Approximate query engines: Commercial challenges and research opportunities," in *Proceedings of the ACM Conference on Management of Data*, 2017, pp. 521–524. doi: 10.1145/3035918.3056098

[6] "MySQL," https://www.mysql.com/, accessed: 2018-07-09.

[7] "SQL Server," https://www.microsoft.com/en-us/sql-server/default.aspx, accessed: 2018-07-09.

[8] A. J. Lattanze, *Architecting Software Intensive Systems: A Practitioners Guide*, 1st ed. Boston, MA, USA: Auerbach Publications, 2008.

[9] M. Peterson, *Intelligence-led Policing: The New Intelligence Architecture*. Rockville, MD: Bureau of Justice Assistance, 2005.

[10] Z. Liu and J. Heer, "The effects of interactive latency on exploratory visual analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2122–2131, 2014. doi: 10.1109/TVCG.2014.2346452

[11] S. L. Lohr, *Sampling: Design and Analysis*, ser. Second. Boston, MA, USA: Brooks/Cole, 2009.

[12] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," *ACM SIGMOD Record*, vol. 26, no. 2, pp. 171–182, June 1997. doi: 10.1145/253262.253291

[13] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang, "Sample + Seek: Approximating aggregates with distribution precision guarantee," in *Proceedings of the ACM Conference on Management of Data*, 2016, pp. 679–694. doi: 10.1145/2882903.2915249

[14] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari, and I. Stoica, "Knowing when you're wrong: Building fast and reliable approximate query processing systems," in *Proceedings of the ACM conference on Management of Data*, 2014, pp. 481–492. doi: 10.1145/2588555.2593667

[15] D. Alabi and E. Wu, "PFunk-H: Approximate query processing using perceptual models," in *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, 2016, pp. 1–6. doi: 10.1145/2939502.2939512

[16] K. A. Cook and J. J. Thomas, *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE-Press, 5 2005.

[17] E. Zgraggen, A. Galakatos, A. Crotty, J. D. Fekete, and T. Kraska, "How Progressive Visualizations Affect Exploratory Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 8, pp. 1977–1987, 2017. doi: 10.1109/TVCG.2016.2607714

[18] J. H. Larkin and H. A. simon, "Why a diagram is (sometimes) worth ten thousand words," *Cognitive Science*, vol. 11, no. 1, pp. 65–99, 1987. doi: 10.1111/j.1551-6708.1987.tb00863.x

[19] G. Wang, A. Malik, C. Yau, C. Surakitbanharn, and D. S. Ebert, "TraSeer: A visual analytics tool for vessel movements in the coastal areas," in *Proceedings of the IEEE Symposium on Technologies for Homeland Security*, 2017, pp. 1–6. doi: 10.1109/THS.2017.7943473

[20] A. Malik, R. Maciejewski, N. Elmqvist, Y. Jang, D. S. Ebert, and W. Huang, "A correlative analysis process in a visual analytics environment," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, Oct 2012, pp. 33–42. doi: 10.1109/VAST.2012.6400491

[21] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proceedings of the IEEE Symposium on Visual Languages*, 1996, pp. 336–343. doi: 10.1109/VL.1996.545307

[22] G. Wang, A. Akers, J. F. de Queiroz Neto, C. Surakitbanharn, and D. S. Ebert., "Spatiotemporal driven analysis of law enforcement data," in *Proceedings of The IEEE Workshop on Visualization in Practice*, 2017.

[23] J. Zhang, J. Chae, C. Surakitbanharn, and D. S. Ebert, "SMART: Social media analytics and reporting toolkit," in *Proceedings of The IEEE Workshop on Visualization in Practice*, 2017.

[24] A. Malik, R. Maciejewski, S. Towers, S. McCullough, and D. S. Ebert, "Proactive spatiotemporal resource allocation and predictive visual analytics for community policing and law enforcement," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1863–1872, 2014. doi: 10.1109/TVCG.2014.2346926

[25] L. Wilkinson, *The Grammar of Graphics (Statistics and Computing)*. Berlin, Heidelberg: Springer-Verlag, 2005.

[26] D. Fisher, "Hotmap: Looking at geographic attention," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1184–1191, 2007. doi: 10.1109/TVCG.2007.70561

[27] E. Imhof, *Cartographic Relief Presentation*. Redlands, CA: ESRI Press, 2007.

[28] B. D. Dent, J. S. Torguson, and T. W. Hodler, *Cartography: Thematic Map Design*, 6th ed. Boston, MA: McGraw-Hill Higher Education, 1999.

[29] J. Zhang, C. Surakitbanharn, N. Elmqvist, R. Maciejewski, Z. Qian, and D. S. Ebert, "TopoText: Context-preserving text data exploration across multiple spatial scales," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2018, pp. 1–13. doi: 10.1145/3173574.3173611

[30] B. Johnson and B. Shneiderman, "Tree-Maps: A space-filling approach to the visualization of hierarchical information structures," in *Proceedings of the IEEE Conference on Visualization*, 1991, pp. 284–291. [Online]. Available: https://dl.acm.org/doi/10.5555/949607.949654

[31] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel, *Visual Analytics of Movement*. Berlin Heidelberg: Springer, 2013. doi: 10.1007/978-3-642-37583-5

[32] R. Scheepens, N. Willems, H. van de Wetering, and J. J. van Wijk, "Interactive visualization of multivariate trajectory data with density maps," in *Proceedings of the IEEE Pacific Visualization Symposium*, 2011, pp. 147–154. doi: 10.1109/PACIFICVIS.2011.5742384

[33] R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. J. van Wijk, "Composite density maps for multivariate trajectories," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2518–2527, 2011. doi: 10.1109/TVCG.2011.181

[34] R. Scheepens, C. Hurter, H. van de Wetering, and J. J. van Wijk, "Visualization, selection, and analysis of traffic flows," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 379–388, 2016. doi: 10.1109/TVCG.2015.2467112

[35] D. Fisher, "Big data exploration requires collaboration between visualization and data infrastructures," in *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, ser. HILDA '16, 2016, pp. 1–5. doi: 10.1145/2939502.2939518

[36] K. Engel, P. Hastreiter, B. Tomandl, K. Eberhardt, and T. Ertl., "Combining local and remote visualization techniques for interactive volume rendering in medical applications," in *Proceedings of the IEEE Conference on Visualization*, 2000, pp. 449–452. doi: 10.1109/VISUAL.2000.885729

[37] K. Engel, O. Sommer, and T. Ertl, "A framework for interactive hardware accelerated remote 3D-visualization," in *Proceedings of the IEEE VGTC/Eurographic Symposium on Visualization*, 2000, pp. 167–177. doi: 10.1007/978-3-7091-6783-0_17

[38] K.-L. Ma and D. M. Camp, "High performance visualization of time-varying volume data over a wide-area network," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, Nov 2000, pp. 29–29. doi: 10.1109/SC.2000.10000

[39] S. Stegmaier, M. Magallón, and T. Ertl, "A generic solution for hardware-accelerated remote visualization," in *Proceedings of the Symposium on Data Visualisation*, 2002, pp. 87–94. doi: 10.5555/509740.509754

[40] F. Lamberti and A. Sanna, "A streaming-based solution for remote visualization of 3D graphics on mobile devices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 247–260, 2007. doi: 10.1109/TVCG.2007.29

[41] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[42] "Apache spark," https://spark.apache.org/, accessed: 2018-07-09.

[43] A. Eldawy and M. F. Mokbel, "SpatialHadoop: A mapreduce framework for spatial data," in *Proceedings of the IEEE Conference on Data Engineering*, 2015, pp. 1352–1363. doi: 10.1109/ICDE.2015.7113382

[44] J.-F. Im, F. G. Villegas, and M. J. McGuffin, "VisReduce: Fast and responsive incremental information visualization of large datasets," in *Proceedings of the IEEE Conference on Big Data*, Oct 2013, pp. 25–32. doi: 10.1109/BigData.2013.6691710

[45] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, "Data Cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 29–53, Jan. 1997. doi: 10.1023/A:1009726021843

[46] Z. Liu, B. Jiang, and J. Heer, "*imMens*: Real-time visual querying of big data," *Computer Graphics Forum*, vol. 32, no. 3pt4, pp. 421–430, June 2013. doi: 10.1111/cgf.12129

[47] L. Lins, J. T. Klosowski, and C. Scheidegger, "Nanocubes for real-time exploration of spatiotemporal datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2456–2465, Dec 2013. doi: 10.1109/TVCG.2013.179

[48] C. A. L. Pahins, S. A. Stephens, C. Scheidegger, and J. L. D. Comba, "Hashedcubes: Simple, low memory, real-time visual exploration of big data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 671–680, Jan 2017. doi: 10.1109/TVCG.2016.2598624

[49] F. Miranda, L. Lins, J. T. Klosowski, and C. T. Silva, "Topkube: A rank-aware data cube for real-time exploration of spatiotemporal data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 3, pp. 1394–1407, 2018. doi: 10.1109/TVCG.2017.2671341

[50] C. Liu, C. Wu, H. Shao, and X. Yuan, "Smartcube: An adaptive data management architecture for the real-time visualization of spatiotemporal datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 790–799, 2020. doi: 10.1109/TVCG.2019.2934434

[51] H. Igehy, M. Eldridge, and K. Proudfoot, "Prefetching in a texture cache architecture," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, Aug 1998, pp. 133–ff. doi: 10.1145/285305.285321

[52] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan, "Maintaining interactivity while exploring massive time series," in *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, 2008, pp. 59–66. doi: 10.1109/VAST.2008.4677357

[53] L. Battle, R. Chang, and M. Stonebraker, "Dynamic prefetching of data tiles for interactive visualization," in *Proceedings of the ACM Conference on Management of Data*, 2016, pp. 1363–1375. doi: 10.1145/2882903.2882919

[54] R. Bruckschen, F. Kuester, B. Hamann, and K. I. Joy, "Real-time out-of-core visualization of particle traces," in *Proceedings of the IEEE symposium on parallel and large-data visualization and graphics*, 2001, pp. 45–50. doi: 10.1109/PVGS.2001.964403

[55] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno, "Adaptive TetraPuzzles: Efficient out-of-core construction and visualization of gigantic multiresolution polygonal models," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 796–803, Aug 2004. doi: 10.1145/1015706.1015802

[56] M. Cox and D. Ellsworth, "Application-controlled demand paging for out-of-core visualization," in *Proceedings of the IEEE Conference on Visualization*, 1997, pp. 235–ff.

[57] T. A. Funkhouser and C. H. Séquin, "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments," in *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*, 1993, pp. 247–254. doi: 10.1145/166117.166149

[58] C. Silva, Y. jen Chiang, W. Corrêa, J. El-sana, and P. Lindstrom, "Out-of-core algorithms for scientific visualization and computer graphics," in *Proceedings of the IEEE Conference on Visualization*, 2002.

[59] M. Singh, Q. Zhu, and H. Jagadish, "SWST: A disk based index for sliding window spatio-temporal data," in *Proceedings of the IEEE Conference on Data Engineering*. IEEE, 2012, pp. 342–353. doi: 10.1109/ICDE.2012.98

[60] V. Benzaken, J.-D. Fekete, P.-L. Hémery, W. Khemiri, and I. Manolescu, "EdiFlow: Data-intensive interactive workflows for visual analytics," *Proceedings of the IEEE Conference on Data Engineering*, pp. 780–791, 2011. doi: 10.1109/ICDE.2011.5767914

[61] E. Wu, L. Battle, and S. R. Madden, "The case for data visualization management systems: Vision paper," *Proceedings of the VLDB Endowment*, vol. 7, no. 10, pp. 903–906, 2014. doi: 10.14778/2732951.2732964

[62] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "BlinkDB: Queries with bounded errors and bounded response times on very large data," in *Proceedings of the ACM European Conference on Computer Systems*, April 2013, pp. 29–42. doi: 10.1145/2465351.2465355

[63] N. Potti and J. M. Patel, "DAQ: A new paradigm for approximate query processing," *Proceedings of the VLDB Endowment*, vol. 8, no. 9, pp. 898–909, May 2015. doi: 10.14778/2777598.2777599

[64] Y. Park, A. S. Tajik, M. Cafarella, and B. Mozafari, "Database learning: Toward a database that becomes smarter every time," in *Proceedings of the ACM Conference on Management of Data*, 2017, pp. 587–602. doi: 10.1145/3035918.3064013

[65] F. Olken, "Random sampling from databases," Ph.D. dissertation, University of California at Berkeley, 1993.

[66] F. Olken and D. Rotem, "Sampling from spatial databases," *Statistics and Computing*, vol. 5, no. 1, pp. 43–57, Mar 1995. doi: 10.1007/BF00140665

[67] A. Guttman, "R-Trees: A dynamic index structure for spatial searching," *Proceedings of the ACM Conference on Management of Data*, vol. 14, no. 2, pp. 47–57, June 1984. doi: 10.1145/971697.602266

[68] R. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974. doi: 10.1007/BF00288933

[69] Y. Li, C.-Y. Chow, K. Deng, M. Yuan, J. Zeng, J.-D. Zhang, Q. Yang, and Z.-L. Zhang, "Sampling big trajectory data," in *Proceedings of the ACM Conference on Information and Knowledge Management*, ser. CIKM '15, Oct 2015, pp. 941–950. doi: 10.1145/2806416.2806422

[70] N. G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 280–292, Jun 2001. doi: 10.1109/90.929851

[71] P. Cudre-Mauroux, E. Wu, and S. Madden, "TrajStore: An adaptive storage system for very large trajectory data sets," in *Proceedings of the IEEE Conference on Data Engineering*, March 2010, pp. 109–120. doi: 10.1109/ICDE.2010.5447829

[72] C. D. Stolper, A. Perer, and D. Gotz, "Progressive visual analytics: User-driven visual exploration of in-progress analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1653–1662, 2014. doi: 10.1109/TVCG.2014.2346574

[73] S. Rahman, M. Aliakbarpour, H. K. Kong, E. Blais, K. Karahalios, A. Parameswaran, and R. Rubinfield, "I've seen ''enough'': Incrementally improving visualizations to support rapid decision making," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1262–1273, Aug. 2017. doi: 10.14778/3137628.3137637

[74] J.-D. Fekete, D. Fisher, A. Nandi, and M. Sedlmair, "Progressive Data Analysis and Visualization (Dagstuhl Seminar 18411)," *Dagstuhl Reports*, vol. 8, no. 10, pp. 1–40, 2019. doi: 10.4230/DAGREP.8.10.1

[75] S. K. Badam, N. Elmqvist, and J. D. Fekete, "Steering the craft: UI elements and visualizations for supporting progressive visual analytics," *Computer Graphics Forum*, vol. 36, no. 3, pp. 491–502, 2017. doi: 10.1111/cgf.13205

[76] M. Angelini, G. Santucci, H. Schumann, and H. J. Schulz, "A review and characterization of progressive visual analytics," *Informatics*, vol. 5, no. 3, pp. 1–27, 2018. doi: 10.3390/informatics5030031

[77] D. Moritz and D. Fisher, "What users don't expect about exploratory data analysis on approximate query processing systems," in *Proceedings of the ACM Workshop on Human-In-the-Loop Data Analytics*, ser. HILDA'17, 2017, pp. 9:1–9:4. doi: 10.1145/3077257.3077258

[78] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy, "The aqua approximate query answering system," *ACM conference on Management of data*, vol. 28, no. 2, pp. 574–576, 1999. doi: 10.1145/304181.304581

[79] N. Pezzotti, B. P. Lelieveldt, L. Van Der Maaten, T. Höllt, E. Eisemann, and A. Vilanova, "Approximated and user steerable tSNE for progressive visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 7, pp. 1739–1752, 2017. doi: 10.1109/TVCG.2016.2570755

[80] J. Jo, J. Seo, and J. D. Fekete, "PANENE: A progressive algorithm for indexing and querying approximate k-nearest neighbors," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 2, pp. 1347–1360, 2020. doi: 10.1109/TVCG.2018.2869149

[81] P. J. Rhodes, R. S. Laramee, R. D. Bergeron, and T. M. Sparr, "Uncertainty Visualization Methods in Isosurface Rendering," in *Proceedings of the IEEE VGTC/Eurographics Symposium on Visualization*, 2003. doi: 10.2312/egs.20031054

[82] M. Correll and M. Gleicher, "Error bars considered harmful: Exploring alternate encodings for mean and error," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2142–2151, 2014. doi: 10.1109/TVCG.2014.2346298

[83] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan, "Visual semiotics & uncertainty visualization: An empirical study," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2496–2505, 2012. doi: 10.1109/TVCG.2012.279

[84] S. Hazarika, A. Biswas, and H.-W. Shen, "Uncertainty visualization using copula-based analysis in mixed distribution models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 934–943, 2018. doi: 10.1109/TVCG.2017.2744099

[85] J. Görtler, C. Schulz, D. Weiskopf, and O. Deussen, "Bubble treemaps for uncertainty visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 719–728, 2018. doi: 10.1109/TVCG.2017.2743959

[86] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld, "Rapid sampling for visualizations with ordering guarantees," *Proceedings of the VLDB Endowment*, vol. 8, no. 5, pp. 521–532, Jan 2015. doi: 10.14778/2735479.2735485

[87] A. Das Sarma, H. Lee, H. Gonzalez, J. Madhavan, and A. Halevy, "Efficient spatial sampling of large geographical tables," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12, 2012, pp. 193–204. doi: 10.1145/2213836.2213859

[88] E. Goldstein, *Sensation and Perception*, 8th ed. Belmont, CA: Wadsworth, Cengage Learning, 2009.

[89] L. Harrison, F. Yang, S. Franconeri, and R. Chang, "Ranking visualizations of correlation using weber's law," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1943–1952, 2014. doi: 10.1109/TVCG.2014.2346979

[90] M. Kay and J. Heer, "Beyond weber's law: A second look at ranking visualizations of correlation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 469–478, 2016. doi: 10.1109/TVCG.2015.2467671

[91] W. S. Cleveland, *Visualizing Data*. Summit, N.J.: Hobart Press, 1993.

[92] W. S. Cleveland and R. McGill, "Graphical perception: Theory, experimentation, and application to the development of graphical methods," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 531–554, 1984. doi: 10.2307/2288400

[93] R. A. Rensink and G. Baldridge, "The perception of correlation in scatterplots," *Computer Graphics Forum*, vol. 29, no. 3, pp. 1203–1210, Aug 2010. doi: 10.1111/j.1467-8659.2009.01694.x

[94] S. Smart and D. A. Szafir, "Measuring the separability of shape, size, and color in scatterplots," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2019. doi: 10.1145/3290605.3300899

[95] W. Javed, B. McDonnel, and N. Elmqvist, "Graphical perception of multiple time series," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 927–934, 2010. doi: 10.1109/TVCG.2010.162

[96] E. Wu, L. Jiang, L. Xu, and A. Nandi, "Graphical perception in animated bar charts," *ArXiv e-prints*, 2016. [Online]. Available: https://arxiv.org/abs/1604.00080

[97] C. G. Healey and A. P. Sawant, "On the limits of resolution and visual angle in visualization," *ACM Transactions on Applied Perception*, vol. 9, no. 4, 2012. doi: 10.1145/2355598.2355603

[98] Chun-Hsien Chou and Yun-Chin Li, "A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 467–476, 1995. doi: 10.1109/76.475889

[99] G. Sharma, *Digital Color Imaging Handbook*, 1st ed. Boca Raton, FL: CRC Press., 2003.

[100] M. Melgosa, *CIE94, History, Use, and Performance*. Berlin, Heidelberg: Springer, 2014, pp. 1–5. doi: 10.1007/978-3-642-27851-8_13-1

[101] M. R. Luo, G. Cui, and B. Rigg, "The development of the CIE 2000 colour-difference formula: Ciede2000," *Color Research & Application*, vol. 26, no. 5, pp. 340–350, 2001. doi: 10.1002/col.1049

[102] M. Stone, "In color perception, size matters," *IEEE Computer Graphics and Applications*, vol. 32, no. 2, pp. 8–13, 2012. doi: 10.1109/MCG.2012.37

[103] M. Stone, D. A. Szafir, and V. Setlur, "An engineering model for color difference as a function of size," in *Color and Imaging Conference*, vol. 2014, no. 2014. Society for Imaging Science and Technology, 2014, pp. 253–258.

[104] Xuemei Zhang, A. Silverstein, J. Farrell, and B. Wandell, "Color image quality metric S-CIELAB and its application on halftone texture visibility," in *Proceedings of the IEEE COMPCON Digest of Papers*, Feb 1997, pp. 44–48. doi: 10.1109/CMPCON.1997.584669

[105] J. Wu, L. Li, W. Dong, G. Shi, W. Lin, and C. C. Kuo, "Enhanced Just Noticeable Difference Model for Images with Pattern Complexity," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2682–2693, 2017. doi: 10.1109/TIP.2017.2685682

[106] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009. doi: 10.1109/MSP.2008.930649

[107] E. Wu and A. Nandi, "Towards perception-aware interactive data visualization systems," in *Proceedings of The IEEE Workshop on Data Systems for Interactive Analysis*, 2015.

[108] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang, "Indexing spatio-temporal data warehouses," in *Proceedings 18th International Conference on Data Engineering*, 2002, pp. 166–175. doi: 10.1109/ICDE.2002.994706

[109] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 8th ed. Wiley Publishing, 2008, ch. 9, pp. 416–417.

[110] A. Magdy, M. F. Mokbel, S. Elnikety, S. Nath, and Y. He, "Mercury: A memory-constrained spatio-temporal real-time search on microblogs," in *Proceedings of the IEEE Conference on Data Engineering*, 2014, pp. 172–183. doi: 10.1109/ICDE.2014.6816649

[111] A. Magdy, A. M. Aly, M. F. Mokbel, S. Elnikety, Y. He, and S. Nath, "Mars: Real-time spatio-temporal queries on microblogs," in *Proceedings of the IEEE Conference on Data Engineering*, 2014, pp. 1238–1241. doi: 10.1109/ICDE.2014.6816750

[112] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679 – 688, 2006. doi: 10.1016/j.ijforecast.2006.03.001

[113] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, 1969. doi: 10.1137/1114019

[114] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl, "Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, 2012, pp. 143–152.

[115] S. Pezanowski, A. M. MacEachren, A. Savelyev, and A. C. Robinson, "Sense-place3: a geovisual framework to analyze place–time–attribute information in social media," *Cartography and Geographic Information Science*, vol. 45, no. 5, pp. 420–437, 2018. doi: 10.1080/15230406.2017.1370391

[116] B. Mozafari and N. Niu, "A handbook for building an approximate query engine," *IEEE Data Engineering Bulletin*, vol. 38, pp. 3–29, 2015.

[117] L. Arge, "The buffer tree: A technique for designing batched external data structures," *Algorithmica*, vol. 37, no. 1, p. 1–24, 2003. doi: 10.1007/s00453-003-1021-x

[118] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw gps data for geographic applications on the web," in *Proceedings of the 17th International Conference on World Wide Web*, 2008, pp. 247–256. doi: 10.1145/1367497.1367532

[119] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the ACM Conference on Ubiquitous Computing*, 2008, pp. 312–321. doi: 10.1145/1409635.1409677

[120] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, "Understanding transportation modes based on gps data for web applications," *ACM Transactions on the Web*, vol. 4, no. 1, pp. 1–36, 2010. doi: 10.1145/1658373.1658374

[121] *.NET Programming with C++/CLI*, Microsoft Inc., March 2018. [Online]. Available: https://docs.microsoft.com/en-us/cpp/dotnet/dotnet-programming-with-cpp-cli-visual-cpp

[122] *Boost C++ Libraries*, C++ Standards Committee Library Working Group., March 2018. [Online]. Available: http://www.boost.org/

[123] C. Ware, *Information Visualization: Perception for Design*, 2nd ed. Academic Press Burlington, April 2004. doi: 10.1016/B978-1-55860-819-1.X5000-6

[124] B. Brewer, "Perception and its objects," *Philosophical Studies*, vol. 132, no. 1, pp. 87–97, Jan 2007. doi: 10.1007/s11098-006-9051-2

[125] C. Healey and J. Enns, "Attention and visual memory in visualization and computer graphics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 7, pp. 1170–1188, July 2012. doi: 10.1109/TVCG.2011.127

[126] *MATLAB*, The MathWorks, Inc, April 2020. [Online]. Available: https://www.mathworks.com/products/matlab.html

[127] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. doi: 10.1109/TIP.2003.819861

[128] *Find connected components in binary image*, The MathWorks, Inc, April 2020. [Online]. Available: https://www.mathworks.com/help/images/ref/bwconncomp.html

[129] *Measure properties of image regions*, The MathWorks, Inc, April 2020. [Online]. Available: https://www.mathworks.com/help/images/ref/regionprops.html

[130] D. A. Szafir, "Modeling Color Difference for Visualization Design," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 392–401, 2018. doi: 10.1109/TVCG.2017.2744359

VITA

VITA

Guizhen Wang is a Ph.D. student in the School of Electrical and Computer Engineering at Purdue University in West Lafayette, IN, USA. Her research interests include visual analytics, information visualization, human-computer interaction, and database. She received her master's degree in computer science in 2012 from Zhejiang University in Hangzhou, China. She received her bachelor's degree in computer science from Shandong University in Jinan, China.