STULL: Unbiased Online Sampling for Visual Exploration of Large Spatiotemporal Data

Guizhen Wang* Jingjing Guo* Mingjie Tang† José Florencio de Queiroz Neto‡
Calvin Yau* Anas Daghistani§ Morteza Karimzadeh¶ Walid G. Aref* David S. Ebert‖

Purdue University* Chinese Academy of Science† Federal University of Ceará‡
Umm Al-Qura University§ University of Colorado Boulder¶ University of Oklahoma‖

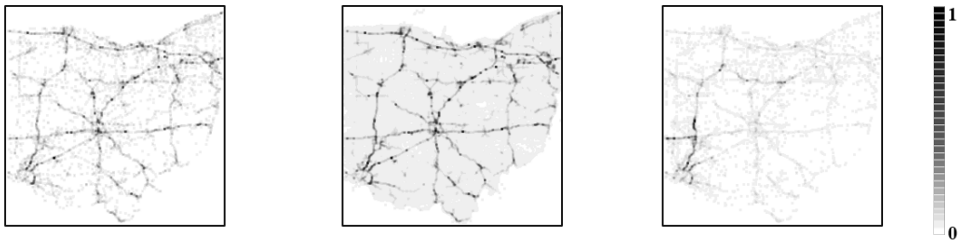

Figure 1: Visual comparison of Ohio highway traffic incident distributions approximated by 0.3% data samples retrieved by STULL (left) and STORM (right) in 40 milliseconds or less, against the exact map (middle) at 100% data, with 32 shades of gray (colorbar). Both using 0.3% sample data, the STORM heatmap indicates hotspots are mainly located on the west side of Ohio's highway network, whereas the STULL heatmap shows hotspots across the state and better resembles the exact map.

Abstract

Online sampling-supported visual analytics is increasingly important, as it allows users to explore large datasets with acceptable approximate answers at interactive rates. However, existing online spatiotemporal sampling techniques are often biased, as most researchers have primarily focused on reducing computational latency. Biased sampling approaches select data with unequal probabilities and produce results that do not match the exact data distribution, leading end users to incorrect interpretations. In this paper, we propose a novel approach to perform unbiased online sampling of large spatiotemporal data. The proposed approach ensures the same probability of selection to every point that qualifies the specifications of a user's multidimensional query. To achieve unbiased sampling for accurate representative interactive visualizations, we design a novel data index and an associated sample retrieval plan. Our proposed sampling approach is suitable for a wide variety of visual analytics tasks, e.g., tasks that run aggregate queries of spatiotemporal data. Extensive experiments confirm the superiority of our approach over a state-of-the-art spatial online sampling technique, demonstrating that within the same computational time, data samples generated in our approach are at least 50% more accurate in representing the actual spatial distribution of the data and enable approximate visualizations to present closer visual appearances to the exact ones.

Keywords: Geospatial data, large-scale data techniques, data management, visual analytics

*e-mail: {wang1908, guo49, yauc, aref}@purdue.edu
†e-mail: tangrock@gmail.com
‡e-mail: florencio@lia.ufc.br
§e-mail: ahdaghistani@uqu.edu.sa
¶e-mail: karimzadeh@colorado.edu
‖e-mail: ebert@ou.edu

1 Introduction

Online sampling-supported Visual Analytics (VA) allows users to explore large volumes of data at interactive rates when it is not feasible to retrieve and render the whole dataset interactively. This is done through continuous retrieval and visualization of retrieved samples that approximate the distribution of the underlying dataset being queried, also known as incremental visualization [21]. As users wait for samples to accumulate over time, the sample size increases, which improves the accuracy of the inferred data pattern; this allows users to trade wait time for accuracy [46]. Therefore, to ensure effective incremental analyses, it is crucial that the progressively retrieved sample is representative of the entire dataset and is not biased. Biased sampling approaches, by definition, sample data with unequal probabilities [37], generate data patterns that deviate from the original dataset, and can lead users to erroneous conclusions (See Figure 1 for an example). To ensure trustworthy and reliable data exploration for VA systems, unbiased sampling is critical.

Incremental visualization requires low retrieval latency to support progressive sampling. In cases using incremental visualization of spatiotemporal data, prevalent spatial sampling approaches [12, 47] have slow retrieval times as they use tree-based spatial indexes (e.g., R-Tree [24]) and iteratively traverse trees from the root to leaf, which often leads to unacceptably high retrieval latency, especially in cases where partial trees reside on disk. A state-of-the-art spatial online sampling approach, `STORM` [61], applies sample buffers to tree-based indexes and uses these buffers to substitute high-latency tree traversals. However, this approach focuses on the efficiency of sample retrieval, without fully resolving the sample bias problem.

In this paper, we present SpatioTemporal Unbiased onLine sampLing (`STULL`), a novel unbiased online sampling approach that supports incremental visualization and interactive exploration of large spatiotemporal data. Motivated by the advantages of sample buffers, `STULL` proposes a carefully designed sample buffer-based data indexing and sample retrieval plan to ensure that each data point satisfying the user-specified multi-dimensional query has an equal probability of being sampled. In particular, unlike state-of-the-art spatial online sampling approach [61], 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. [61]), 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 [9, 54].

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 [61] 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 [39]) for answering queries, well below the response time thresholds for interactive visualization [35].

Our contributions include the following:

- a novel, unbiased online sampling approach for VA systems to incrementally present approximate yet reliable interactive analyses,
- theoretical guarantees on the unbiased property of our presented approach.

2 Visual Analytics And Sampling Bias

Unbiased sampling requires that *each record satisfying the query specification has the same probability of being selected* [37]. Conversely, a sampling approach is considered 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 [20]. 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 often support the interactive exploration of data, following the information seeking mantra [58]: "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 [3]: 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 [43] 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 [44]. 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. This ensures that approximate visualizations can reliably represent the exact answers.

3 Related Work

We organize the state-of-the-art work related to STULL by the following four topics:

Interactive exploration of large data: In a VA system, an additional 500-ms computational latency significantly decreased users' enthusiasm for exploring data [35]. Here we categorize popular techniques that enable VA systems to rapidly process data. First, well-designed data indexes can avoid selecting most query-irrelevant data, which significantly reduces latency [38, 59]. Second, data-cube oriented approaches aggregate the original dataset into a hierarchical knowledge graph, and retrieving answers from such a compact graph is efficient for aggregate queries (e.g., imMens [36], Nanocube [33], Hashedcubes [50], TOPKUBE [42], SmartCube [34]). Third, computational latencies can be reduced by computational parallelism [17, 28] or hidden by pre-fetching [5, 10]. Finally, unlike the above techniques, which process the whole dataset to produce exact results but are slow to return results if the data volume grows exponentially, sampling-based approximate query processing (AQP) techniques [1, 53, 55] use less data to approximate the original dataset but are able to process large volumes of data without performance degradation. This allows VA systems to quickly process small samples of data and produce error-bounded answers, regardless of the total data volume [20, 45].

Sampling-based AQP and sampling bias: A broad range of applications (e.g., Geosciences [57], Ecology [22] and population census [37]) employ sub-samples of large data to extrapolate characteristics of the whole dataset. These extrapolations often assume that their input data can equally represent a large geographical distribution, as biases systematically favor partial data and cause overestimation or underestimation of certain data [30]. Research shows that in logistic regression-based classification problems, inconsistency between the whole dataset and sub-sampled data regarding the multi-class data distribution reduced the predictability of trained models [49]. In ecological research, species distribution models are prone to overfitting, as sample data is biased in favor of regions where it is easy to collect data [7, 30]. Thus, unbiased sampling is essential in many domains to better reflect the true state of the world. To date, unbiased sampling approaches have been widely investigated [37, 60]. For example, unbiased graph sampling [31, 64] considers graph properties (e.g., degree of nodes in social networks) and designs special sampling strategies to ensure the properties extracted from the samples are representative of the whole. Our work focuses on unbiased sampling of spatial data in the online manner, which requires the low latency of sampling large volumes of data.

Visualization can affect sampling strategies as well. For example, some spatial visualization tasks might include more data from low-density areas so that visualizations do not appear to ignore these regions [52]. As for scatterplots, in order to keep desired information (e.g., a data outlier) sampled or to avoid overdraw in high-density areas, some approaches [11, 12, 26, 62] assign each point an uneven

probability of being selected for rendering. This also occurs in some spatial data analyses [14] where each point has intrinsic priority in the sampling procedure (e.g., advertisement ranking). In essence, the aforementioned sampling approaches use intentional bias to preserve desired visual properties, whereas our approach focuses on visualization scenarios that retain the distribution of the underlying data and support interactive exploration of large data.

Online sampling and incremental visual analytics: Online sampling approaches [25, 51] select data in a continuous way so that VA systems can produce immediate outputs and incrementally refine them. Here we review three common types of methods supporting users to conduct progressive analyses. First, users' demands for analytical accuracy were typically expressed as certain statistical measurements to configure the number of needed samples (e.g., Confidence Interval [15, 21, 25, 43]). Second, randomness is crucial to data sampling, which inevitably causes analytical results generated in previous executions to have some degree of numerical difference from those in subsequent executions. Consequently, users have difficulty choosing trustworthy answers. A series of visual analytics approaches [29, 44, 56] were developed to help users reduce uncertainty and determine the best answers. Finally, in addition to mathematical measurements, online sampling processes can also factor in users' perception and measure approximate answer accuracy in terms of perceived information [2, 63]. In this paper, we focus on sampling bias, which hinders visual analytic activities. Our approach samples spatiotemporal data in an unbiased way and will therefore reduce the inaccuracy of approximate answers.

Online sampling of spatial data: Particular sampling techniques build on the special characteristics of spatial data. Olken et al. [47, 48] presented a suite of spatial sampling methods (e.g., `RandomPath` [48, 61]) that conduct back-and-forth traversals over typical hierarchical spatial structures (e.g., R-tree [24], Quad-tree [19]) to retrieve samples. Likewise, similar sampling strategies have been used for object movement trajectories [13, 16, 32] and scatterplots [12]. These approaches traverse their index from root to leaf to obtain one or a few points. The sampling procedure repeats to retrieve more samples. As a result, the sampling time of these methods scales as the number of tree traversals increases. This problem is compounded in cases where the available memory cannot store trees completely and must save partial trees to disks. Disk I/O is much more expensive than in-memory access [23]. Consequently, retrieving samples from disks cannot satisfy time-critical performance requirements that are mandatory in online scenarios [35]. However, in the big data era, it is common to use hard drives as secondary storage to alleviate memory shortage [61]. To adapt to the hybrid data storage system that allocates data to both memory and hard drives, `STORM` [61] proposed a novel data index that uses sample buffers to substitute expensive tree traversals. Sample buffers pack well-selected sample points into disk blocks [4]. Batched disk I/O operations can quickly load a significant number of disk blocks into memory. As such, loading these buffers from disks is more efficient than traditional tree-based approaches. To the best of our knowledge, `STORM` is the first approach to employ online sampling of spatial data [61]. Section 4.1 elaborates details regarding the two sampling genres. In this paper, we focus on sampling bias arising from the sampling procedure equipped with sample buffers. Sampling bias can be avoided either by ensuring equal sampling probability for each point or by involving remedies to correct the bias. [60]. Our approach avoids bias by ensuring each point has the same probability of being selected, and can conduct unbiased sampling of discrete spatiotemporal data records, while satisfying the latency requirement for interactivity.

4 STULL

As users issue queries, `STULL` continuously samples data so that the VA system can create rapid visualizations and progressively improve them. During a 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 introduces the computational details of `STULL`. Section 4.1 gives some intuition on the advantage of our sampling strategy. Sections 4.2, 4.3 and 4.5 detail the scalable data index design, creation and updating. Section 4.4 provides details of the sample retrieval procedure with guaranteed unbiasedness.

4.1 Intuition

Efficient sample retrieval is crucial for online sampling. A sampling plan that randomly selects a subset of 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 involves indexing data and retrieving samples from the index, as the index can minimize the accessed data to the subsets specified by queries. Specific to spatial sampling, spatial indexes (e.g., R-tree [24] and Quad-tree [19]) are widely used to organize data in a spatial hierarchy. These trees often store all the points into leaf cells. The sample retrieval procedure (e.g., `RandomPath` [48, 61]) starts from the tree root, randomly chooses a child in terms of some metrics (e.g., the 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 approach produces samples that represent the queried data in an unbiased manner, but the retrieval process is not efficient. First, the sample retrieval latency increases in proportion to the tree traversal cost. Second, the cost of traversing trees and selecting data from leaves can exceed the latency bound specified by interactive data exploration [35] if the available memory cannot store the whole index and partial of indexes reside on a disk. In cases where data indexes are stored on hard drives, loading non-leaf cells and reached leaves into memory is a lengthy process because each access requires at least one disk Input/Ouput (I/O) operation and completing all the I/Os is time-consuming and at least one to two orders of magnitude slower than in-memory access [23]. To reduce the number of I/Os, an advanced approach, `STORM` [61], proposes storing samples satisfying a query specification in a continuous region on the disk. A batch I/O operation can sequentially scan the disk region to retrieve samples, which is faster. In terms of tree-based spatial hierarchy, `STORM` allocates a continuous region (known as a buffer [4]) on a disk for each of its non-leaf cells. Each buffer stores data samples that are retrieved in advance from the spatial range represented by its linked cells. Thus, the points stored in a sub-tree root's sample buffer can approximate all the data indexed by the sub-tree and act as a sample set. Likewise, the union of points in the sample buffers belonging to the root's children can approximate the data distribution. As such, progressively merging more relevant buffers can form an online sampling manner. In summary, the recent buffer-based approach can retrieve samples rapidly and support VA systems at interactive rates.

Since buffer-based sampling approaches retrieve samples in the units of buffers, the state-of-the-art **fixed-sized** design proposed by `STORM` [61] in which each buffer has the same number of points raises bias (Figure 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 43.3% points (i.e. $\frac{1000+1600}{2000+4000}$) satisfying Q, whereas the sample set that is a combination of their sample buffers has 45% points (i.e., $\frac{250+200}{500+500}$) satisfying 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 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 [61] 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.

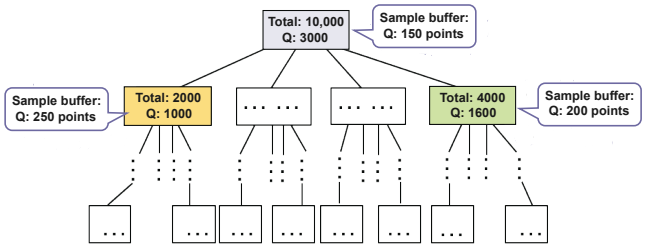

Figure 2: Sampling bias issue in the fixed-sized sample buffer design. Q is a query. Each sample buffer has 500 random data points. Numbers inside each buffer lists the number of points satisfying Q. Numbers inside each cell list the total number of points in the spatial range of the cell and the number of points satisfying Q respectively.

4.2 Index Design

STULL indexes data with an ordered list of pyramids that represents the spatio-temporal segmentation of the data. (Figure 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 [38]) per its spatial dimensions. 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. α is the reciprocal of a pyramid's height. Unlike a Quad-Tree, each pyramid's non-leaf cells have sample buffers. These sample buffers follow the *proportionally-sized* design to cache 100α percent of points randomly selected from their spatiotemporal ranges. Therefore, each pyramid level has in total 100α percent of points. Accordingly, a pyramid has $\frac{100}{100\alpha} = \frac{1}{\alpha}$ levels in total.

At the bottom level of a pyramid, leaf cells store all of the data within their range in a circular array (Figure 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) and participate in sample retrieval (Section 4.4). Figure 3(c) exemplifies the circular array in the leaf with the id "1122" in Figure 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 4.4).

4.3 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 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

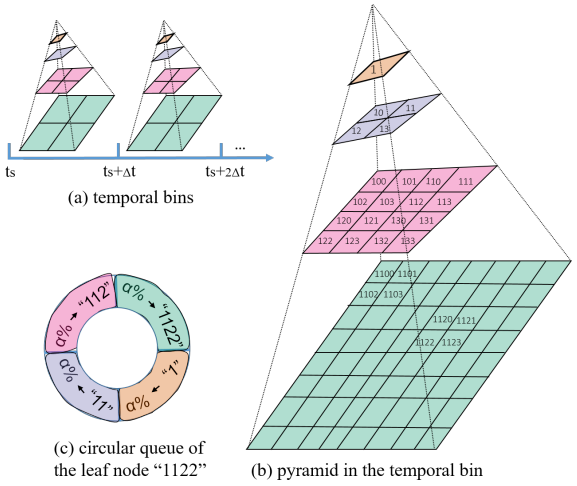

Figure 3: The data index. (a) shows the temporal index beginning at t_s. Each segment in (a) is a temporal bin. Each temporal bin sets $\alpha = 0.25$ and uses a four-level pyramid (in (b)) to spatially organize data. A pyramid leaf uses a four-segment circular array (in (c)) to store data. Each non-leaf cell has a sample buffer to store data.

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

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

4.4 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 100θ percent of sample points from each temporal bin requested per Q, where θ denotes the ratio of points desired by users. 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 $[(l_r+1)\bmod\frac{1}{\alpha}]$.

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 $l_r \geq l_Q$, after 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. Similarly to sample buffers, the retrieval procedure accesses the same θ/α portion of points in a segment, continue, and will exhaust all eligible points after $1/\theta$ times. Then, the index of the next retrieved segment is $(l_r+1)\bmod\frac{1}{\alpha}$. 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.

Algorithm 2: Retrieving samples in Temporal Bin t

input : Query Q; θ
output : A random sample S with 100θ percent of points

Determine l_Q according to the spatial query range of Q;
Initialize empty lists S and G;
$l_r \leftarrow$ a level randomly chosen between 1 and $\frac{1}{\alpha}$;
$l \leftarrow l_r$; $u \leftarrow 1$;
while *($u <= \frac{1}{\theta}$ or users didn't terminate)* **do**
  if *G is empty* **then**
    if $l_Q \leq l_r$ *and* $l < \frac{1}{\alpha}$ **then**
      $G \leftarrow$ sample buffers of cells that are in the l-th level and spatially overlapping with Q;
    else
      $G \leftarrow$ the l-th segments of cells that are in the leaf level and spatially overlapping with Q;
    end
  end
  $u_0 \leftarrow (u \bmod \frac{\alpha}{\theta}) == 0 ? \frac{\alpha}{\theta} : u \bmod \frac{\alpha}{\theta}$;
  for *each element b in G* **do**
    $s \leftarrow$ points satisfying Q and in the $[100(u_0-1)\theta/\alpha\%, 100u_0\theta/\alpha\%]$ portion of b;
    $S \leftarrow S \cup s$;
    Send S to a VA system for visualization;
  end
  $u \leftarrow u+1$;
  if *u mod $\frac{\alpha}{\theta} == 0$* **then**
    $G \leftarrow$ an empty list;
    $l \leftarrow 1 + (l \bmod \frac{1}{\alpha})$;
    if $l == l_0$ **then**
      break;
    end
  end
end

4.5 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 [38] querying sensor data collected in the last ten minutes); the update procedure likely adjusts only the latest few temporal bins, which is therefore fast.

5 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 as follows. STULL also guarantees interactive rates, making it suitable for online sampling and incremental visualization. A formal computational complexity analysis is detailed in Appendix A.

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 per visual update. 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 1 shows that each segment in the circular queue of a leaf has an equal chance of being selected. Equation 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.

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

$$\begin{aligned} &P(\text{Point } r \text{ is chosen}) \\ &= \sum_{l_i=1}^{1/\alpha} P(r \text{ is in the } l_i\text{-th segment}) \times P(l_i \text{ is chosen}) \\ &= \sum_{l_j=1}^{1/\alpha} \frac{1}{\alpha} \times \theta = \theta \end{aligned} \tag{2}$$

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 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.

Table 1: Evaluated datasets.

Data	Description	Counts (million)	Memory size (MB)	Spatial range	Temporal bin counts	Temporal bin interval
GEO [65–67]	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

6 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 with the Microsoft .Net Framework and Visual C++ [41]. Section 3 and Section 4.1 elaborate on our baseline choice.

- STORM [61] is a spatial online sampling approach, using a sample-buffer equipped R-tree [24] to index data. It uses the fixed-sized sample buffer design, making non-leaf cells have the same number of points in their buffers. In our experiments, STORM used the Boost library API [8] to build a quadratic R-tree [24], and each of its sample buffers have 1024 points.
- RandomPath is a variant of a spatial sampling approach [48] that traverses a tree-based spatial index to sample data. In our implementation, points are grouped into temporal bins, and each bin uses a Quad-tree [19] to index its points spatially. Unlike STORM and STULL, there are no sample buffers, and all of the points are stored merely in leaf cells. In each bin, it follows the tree-traversal based manner [48] to retrieve samples. RandomPath produces an unbiased sampling, but is slower in sample retrieval. Thus, RandomPath is an approach to offline sampling instead of online sampling.

Data sets. Table 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.

6.1 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 Root Mean Square Error (RMSE) [27], which calculates differences between exact answers and approximate answers.

Regarding accuracy in the spatial dimension, we queried the Kernel Density Estimation (KDE) [18] 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 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 the entire dataset. The density value in each hour was normalized to the scale of 0 to 1. Figure 5 shows RMSE-quantified accuracy, compared to the exact distribution. We see that at the same sample size, STORM has the most significant RMSE errors. At 5%, STULL's value is averagely 50% less than that of STORM. Furthermore, the RMSE errors in the OSP case are the largest.

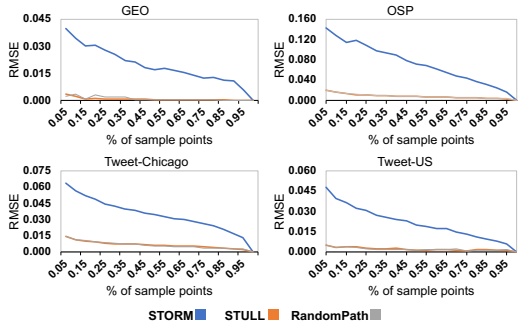

Figure 4: RMSE measurement of approximate Kernel Density Estimation results. The query requested the entire dataset. Results were averaged over five runs.

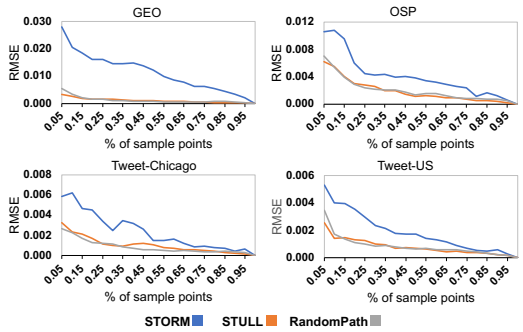

Figure 5: RMSE measurement of approximate hourly distribution results. The query pertained to every point in the entire dataset. Results were averaged over five runs.

6.2 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.

Figure 6 shows incremental visualization of approximate spatial heatmaps [40] 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

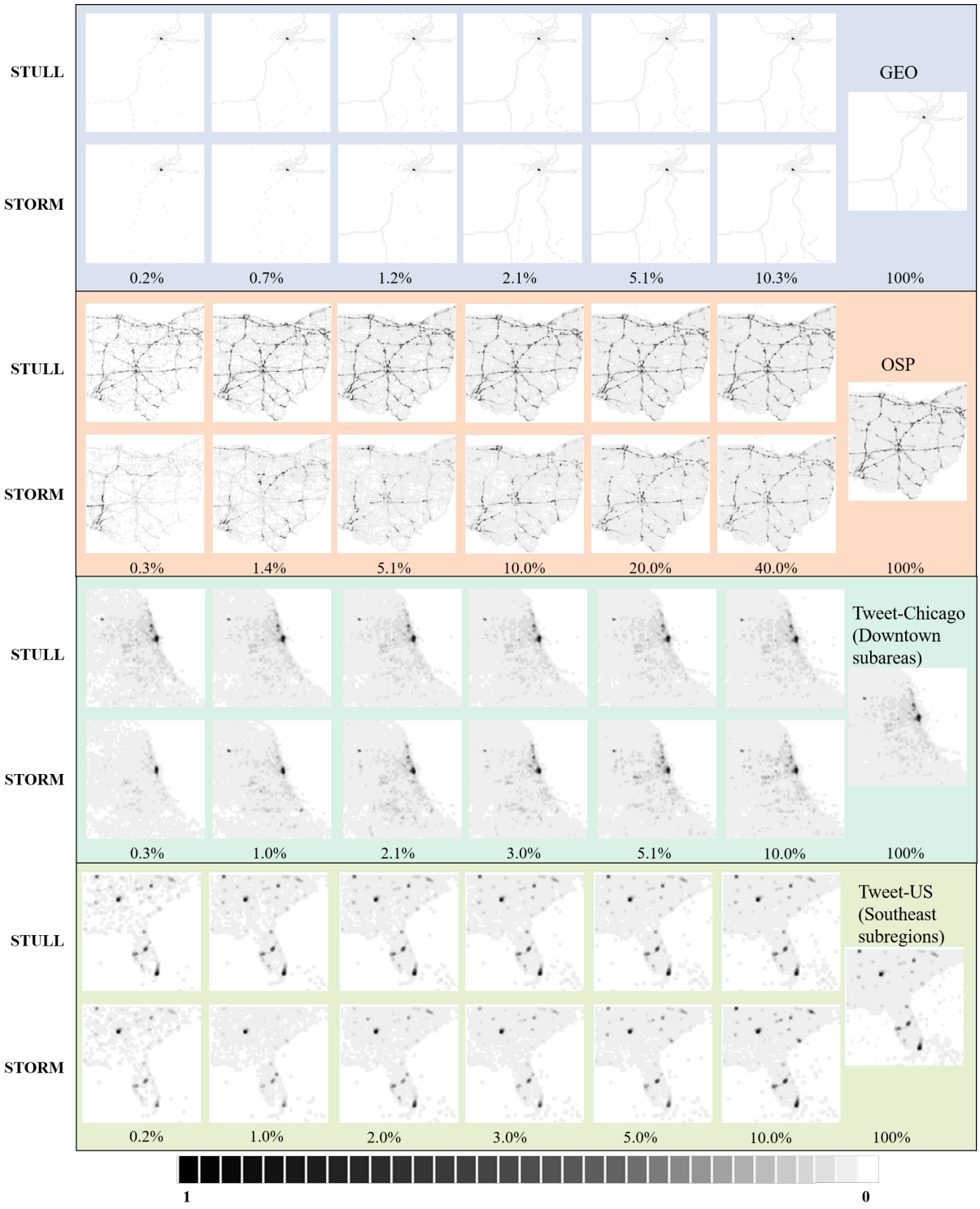

Figure 6: Comparison of spatial heatmaps generated by the two approaches. A number below a heatmap indicates number of sample points selected for approximate distributions. At the bottom is the gray-scale colormap with 32 shades.

areas since these areas have fewer points selected; When sample sizes exceed 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 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 from the exact ones than those with STORM. In the OSP case, point densities between 12 PM and 6 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 11 PM and 6 AM extrapolated from STORM's chart have obvious discrepancies. STORM also presents light but discernible color differences between 6 PM and 12 AM in the Tweet-Chicago case and between 1 PM and 5 PM in the Tweet-US case.

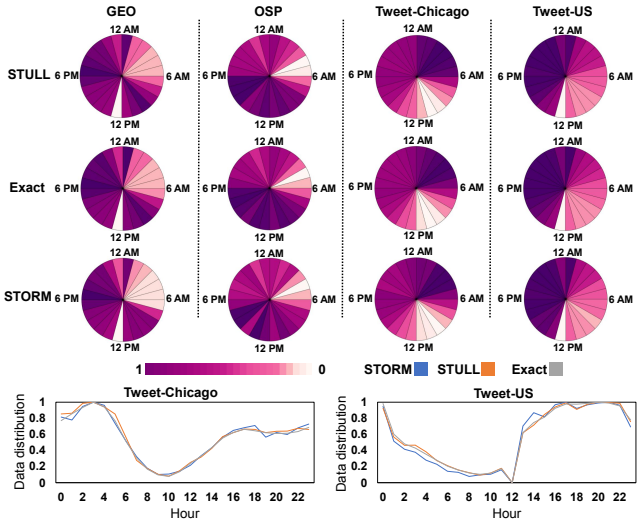

Figure 7: Pie charts showing normalized hourly distribution of the entire data. Each slice denotes a hour. These approximate charts are generated with 0.1% points of being selected. The color legend has 32 color shades. Line charts show the two datasets having closer visual appearances in the pie chart views. The result is one-time run.

6.3 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 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, which are overall shorter than RandomPath. On average, at the same sample size, STULL saved at least 60% of the time used by RandomPath. Figure 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 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 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

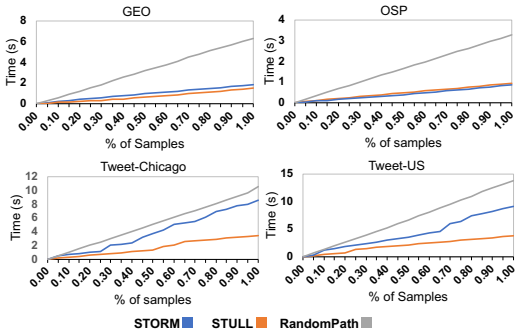

Figure 8: Time measurements (in seconds) to retrieve samples in an in-memory setting. The query requires the entire data. STULL has $\alpha = 0.25$, and RandomPath has at most 4 levels in each of its Quad-trees. Results are averaged over 5 runs.

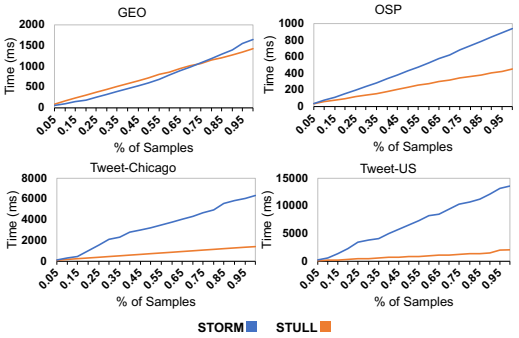

Figure 9: Time measurements (in milliseconds) to retrieve samples from an in-memory data index. Queried temporal ranges are, 2012 for OSP, 2011-2012 for GEO, 2013/04-2013/06 for Tweet-Chicago, and 2018/01/01-2018/02/04 for Tweet-US. STULL has $\alpha = 0.25$. Results are averaged over five runs.

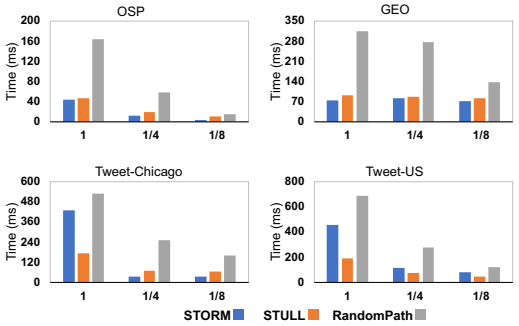

Figure 10: Average time per incremental update. Each incremental update retrieved 5% points. Numbers below a bar indicate queried spatial range, 1 for the whole spatial extent, $1/4$ for a quarter of the whole extent, and $1/8$ for a one-eighth. For STULL, $\alpha = 0.25$. Each of RandomPath's Quad-trees has at most 4 levels.

in each update. Figure 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 13 compares retrieval time between STULL and RandomPath. When sampling 5% points for the initial

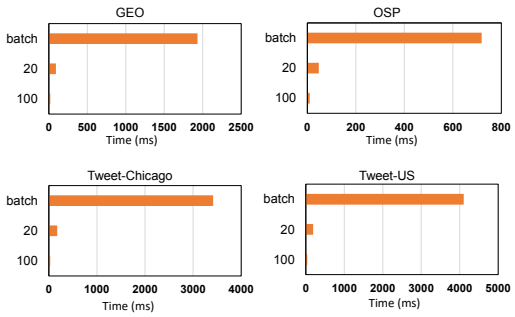

Figure 11: Averaged sample retrieval latency per incremental update. The query required the entire data. In the y-axis, batch indicates time to retrieve the entire dataset, 20 indicates incremental visualization has 20 updates in total and retrieves 5% point per update; likewise, 100 indicates 1% per update and 100 updates in total. For STULL, $\alpha = 0.25$. Results are averaged over five runs.

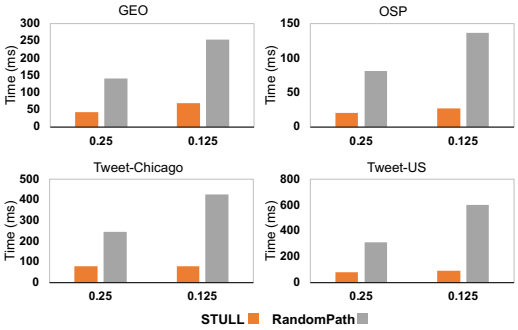

Figure 12: Averaged sample retrieval latency per incremental update. Each incremental update retrieved 2.5% points. 0.25 indicates that α of STULL is 0.25, and RandomPath's Quad-tree index has no more than 4 levels. Likewise, 0.125 indicates α = 0.125, and a Quad-tree index has at most 8 levels. Results are averaged over three runs.

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, the time needed for RandomPath to load points from other leaf cells is negligible.

6.4 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 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 3 shows the average time to insert new arrivals of 5000 tweets, with the assumed streaming data arrival rate around 1000∼4000+ per second [39, 61]. 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.

7 Discussion

We validated the **importance of the unbiased guarantee for incremental visualization** through designing experiments that measured

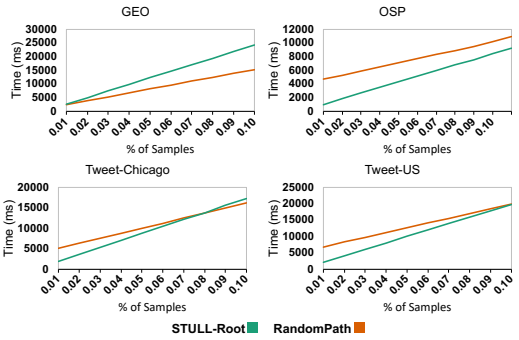

Figure 13: Latency to retrieve samples from disk-resident indexes for a query requiring the entire data. Each incremental update obtains 5% points. STULL-Root refers STULL started retrieval from the pyramid root in each temporal bin. For STULL, $\alpha = 0.25$. For RandomPath, each Quad-tree has at most 4 levels. Results are averaged over three runs.

Table 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

Table 3: Time measurements (in milliseconds) for STULL to insert 5000 points into the existing index. Results are averaged over five runs.

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

numerical and visual accuracy between approximate and exact answers. RMSE results (Figure 4 and Figure 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 6 and Figure 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 [43, 44]. 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) [25, 43]. 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 are

correct or not [21]. 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 mainly by sample size [37]. However, STORM has one more factor, intrinsic spatial distributions in the data. 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) 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.

STULL achieves the **temporal unbiased property** 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 [13, 32]. Thus, we do not elaborate on it.

Our range of experiments also validate the efficiency of STULL's sample retrieval. Experiments (Figure 8, Figure 9, Figure 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 for the case in 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 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 techniques to future work.

STULL **reduces the data index creation and update workload**, compared to STORM. STULL indexes data spatially using pyramids for efficiency [39]. 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 3 shows that it takes less than 450ms to index 5000 points. Thus, inserting the new data into the existing data index and retrieving 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., estimating data count per hour or evaluating averaged statistics of income per neighborhood. Here, we demonstrate aggregation computation with samples retrieved by STULL and use confidence intervals [25, 37] 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**. STULL's pyramids contain all points in the bottom levels, in addition to $(1-\alpha)$ portion of data in non-bottom levels, whereas Quad-Tree does not have such extra costs. Thus, STULL maximizes retrieval efficiency at the expense of data storage space. If data stored at non-bottom levels are not duplicated at the bottom level, STULL will move from the bottom level to the higher levels, and retrieve samples from these higher levels. Since higher levels consist of cells whose spatial 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 [6]. For convenient disk-based data storage and retrieval, STORM [61] 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 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 and slow down disk I/O as well. We leave this to future work.

8 Conclusion and Future Work

This paper 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 needed. A novel scheme is expected to resolve this issue.

Acknowledgments

This work was supported in part by the National Science Foundation Grant CA-FW-HTF 1937036. Walid G. Aref acknowledges the support of the National Science Foundation under Grant Numbers III-1815796 and IIS-1910216. Mingjie Tang acknowledges the support of the National Natural Science Foundation of China (Grant No. 61802364). The authors wish to thank Jieqiong Zhao, Audrey Reinert, and Luke Snyder for their editorial comments and helps.

REFERENCES

[1] 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*, EuroSys '13, pp. 29–42, April 2013. doi: 10.1145/2465351.2465355

[2] 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*, HILDA '16, pp. 10:1–10:6, 2016. doi: 10.1145/2939502.2939512

[3] M. Angelini, G. Santucci, H. Schumann, and H. J. Schulz. A review and characterization of progressive visual analytics. *Informatics*, 5(3):1–27, 2018. doi: 10.3390/informatics5030031

[4] L. Arge. The buffer tree: A technique for designing batched external data structures. *Algorithmica*, 37(1):124, sep 2003. doi: 10.1007/s00453-003-1021-x

[5] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *Proceedings of the International Conference on Management of Data*, SIGMOD '16, pp. 1363–1375, 2016. doi: 10.1145/2882903.2882919

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

[7] R. A. Boria, L. E. Olson, S. M. Goodman, and R. P. Anderson. Spatial filtering to reduce sampling bias can improve the performance of ecological niche models. *Ecological Modelling*, 275:73 – 77, 2014. doi: 10.1016/j.ecolmodel.2013.12.012

[8] C++ Standards Committee Library Working Group. *Boost C++ Libraries*, March 2018.

[9] 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*, pp. 143–152, May 2012.

[10] 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*, pp. 59–66, Oct 2008. doi: 10.1109/VAST.2008.4677357

[11] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K. L. Ma. Visual abstraction and exploration of multi-class scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1683–1692, 2014. doi: 10.1109/TVCG.2014.2346594

[12] X. Chen, T. Ge, J. Zhang, B. Chen, C.-w. Fu, O. Deussen, and Y. Wang. A recursive subdivision technique for sampling multi-class scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):729–738, Jan 2020. doi: 10.1109/TVCG.2019.2934541

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

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

[15] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample + Seek: Approximating aggregates with distribution precision guarantee. In *Proceedings of the International Conference on Management of Data*, SIGMOD '16, pp. 679–694, 2016. doi: 10.1145/2882903.2915249

[16] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Transactions on Networking*, 9(3):280–292, Jun 2001. doi: 10.1109/90.929851

[17] A. Eldawy and M. F. Mokbel. SpatialHadoop: A mapreduce framework for spatial data. In *Proceedings of the IEEE 31st International Conference on Data Engineering*, pp. 1352–1363, April 2015. doi: 10.1109/ICDE.2015.7113382

[18] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969. doi: 10.1137/1114019

[19] J. L. Finkel, R. A.and Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, March 1974. doi: 10.1007/BF00288933

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

[21] 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*, CHI 12, pp. 1673–1682, May 2012. doi: 10.1145/2207676.2208294

[22] Y. Fourcade, J. O. Engler, D. Rdder, and J. Secondi. Mapping species distributions with MAXENT using a geographically biased sample of presence data: A performance assessment of methods for correcting sampling bias. *PLoS ONE*, 9(5):1–13, May 2014. doi: 10.1371/journal.pone.0097122

[23] Fujitsu Technology Solutions. *Basics of disk I/O performance*, 2011. accessed July 31, 2020.

[24] A. Guttman. R-Trees: A dynamic index structure for spatial searching. *ACM SIGMOD Record*, 14(2):47–57, June 1984. doi: 10.1145/971697.602266

[25] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. *SIGMOD Rec.*, 26(2):171–182, June 1997. doi: 10.1145/253262.253291

[26] R. Hu, T. Sha, O. Van Kaick, O. Deussen, and H. Huang. Data sampling in multi-view and multi-class scatterplots via set cover optimization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):739–748, 2020. doi: 10.1109/TVCG.2019.2934799

[27] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pp. 679–688, 2006.

[28] 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 International Conference on Big Data*, pp. 25–32, Oct 2013. doi: 10.1109/BigData.2013.6691710

[29] 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*, 8(5):521–532, Jan 2015. doi: 10.14778/2735479.2735485

[30] S. Kramer-Schadt, J. Niedballa, J. D. Pilgrim, B. Schrder, J. Lindenborn, V. Reinfelder, M. Stillfried, I. Heckmann, A. K. Scharf, D. M. Augeri, S. M. Cheyne, A. J. Hearn, J. Ross, D. W. Macdonald, J. Mathai, J. Eaton, A. J. Marshall, G. Semiadi, R. Rustam, H. Bernard, R. Alfred, H. Samejima, J. W. Duckworth, C. Breitenmoser-Wuersten, J. L. Belant, H. Hofer, and A. Wilting. The importance of correcting for sampling bias in maxent species distribution models. *Diversity and Distributions*, 19(11):1366–1379, 2013. doi: 10.1111/ddi.12096

[31] J. Leskovec and C. Faloutsos. Sampling from large graphs. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006:631–636, 2006. doi: 10.1145/1150402.1150479

[32] 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 International Conference on Information and Knowledge Management*, CIKM 15, pp. 941–950, Oct 2015. doi: 10.1145/2806416.2806422

[33] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, Dec 2013. doi: 10.1109/TVCG.2013.179

[34] 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*, 26(1):790–799, Jan 2020. doi: 10.1109/TVCG.2019.2934434

[35] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2122–2131, 2014.

[36] Z. Liu, B. Jiang, and J. Heer. *imMens*: Real-time visual querying of big data. *Computer Graphics Forum*, 32(3):421–430, 2013. doi: 10.1111/cgf.12129

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

[38] 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 International Conference on Data Engineering*, ICDE 2014, pp. 1238–1241, March 2014. doi: 10.1109/ICDE.2014.6816750

[39] 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 International Conference on Data Engineering*, ICDE 2014, pp. 172–183, March 2014. doi: 10.1109/ICDE.2014.6816649

[40] 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*, 20(12):1863–1872, Dec 2014. doi: 10.1109/TVCG.2014.2346926

[41] Microsoft Inc. *.NET Programming with C++/CLI*, March 2018.

[42] 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*, 24(3):1394–1407, Mar. 2018. doi: 10.1109/TVCG.2017.2671341

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

[44] 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*, CHI '17, pp. 2904–2915, 2017. doi: 10.1145/3025453.3025456

[45] B. Mozafari. Approximate query engines: Commercial challenges and research opportunities. In *Proceedings of the ACM International Conference on Management of Data*, SIGMOD '17, pp. 521–524, 2017. doi: 10.1145/3035918.3056098

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

[47] F. Olken. *Random sampling from databases*. PhD thesis, University of California at Berkeley, 1993.

[48] F. Olken and D. Rotem. Sampling from spatial databases. *Statistics and Computing*, 5(1):43–57, Mar 1995. doi: 10.1007/BF00140665

[49] T. Oommen, L. G. Baise, and R. M. Vogel. Sampling bias and class imbalance in maximum-likelihood logistic regression. *Mathematical Geosciences*, 43:99–120, 2011. doi: 10.1007/s11004-010-9311-8

[50] 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*, 23(1):671–680, Jan 2017. doi: 10.1109/TVCG.2016.2598624

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

[52] Y. Park, M. Cafarella, and B. Mozafari. Visualization-aware sampling for very large databases. In *Proceedings of the IEEE 32nd International Conference on Data Engineering*, ICDE 2016, pp. 755–766, June 2016. doi: 10.1109/ICDE.2016.7498287

[53] 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 International Conference on Management of Data*, SIGMOD '17, pp. 587–602, 2017. doi: 10.1145/3035918.3064013

[54] S. Pezanowski, A. M. MacEachren, A. Savelyev, and A. C. Robinson. SensePlace3: A geovisual framework to analyze placetimeattribute information in social media. *Cartography and Geographic Information Science*, 45(5):420–437, 2018. doi: 10.1080/15230406.2017.1370391

[55] N. Potti and J. M. Patel. DAQ: A new paradigm for approximate query processing. *Proceedings of the VLDB Endowment*, 8(9):898–909, May 2015. doi: 10.14778/2777598.2777599

[56] 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*, 10(11):1262–1273, Aug. 2017. doi: 10.14778/3137628.3137637

[57] S. Reddy and L. M. Dvalos. Geographical sampling bias and its implications for conservation priorities in africa. *Journal of Biogeography*, 30(11):1719–1727, 2003. doi: 10.1046/j.1365-2699.2003.00946.x

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

[59] Y. Tanahashi, C.-H. Hsueh, and K.-L. Ma. An efficient framework for generating storyline visualizations from streaming data. *IEEE Transactions on Visualization and Computer Graphics*, 21(6):730–742, Apr. 2015.

[60] J.-F. Wang, A. Stein, B.-B. Gao, and Y. Ge. A review of spatial sampling. *Spatial Statistics*, 2:1–14, 2012. doi: 10.1016/j.spasta.2012.08.001

[61] L. Wang, R. Christensen, F. Li, and K. Yi. Spatial online sampling and aggregation. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 9(3):84–95, Nov 2015. doi: 10.14778/2850583.2850584

[62] Y. Wei, H. Mei, Y. Zhao, S. Zhou, B. Lin, H. Jiang, and W. Chen. Evaluating perceptual bias during geometric scaling of scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):321–331, 2020. doi: 10.1109/TVCG.2019.2934208

[63] 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.

[64] Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu, and W. Cui. Evaluation of Graph Sampling: A Visualization Perspective. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):401–410, 2017. doi: 10.1109/TVCG.2016.2598867

[65] 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*, 4(1):1–36, Jan. 2010. doi: 10.1145/1658373.1658374

[66] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma. Understanding mobility based on gps data. In *Proceedings of the International Conference on Ubiquitous Computing*, UbiComp '08, pp. 312–321, 2008. doi: 10.1145/1409635.1409677

[67] 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 International Conference on World Wide Web*, WWW '08, pp. 247–256. ACM, New York, NY, USA, 2008. doi: 10.1145/1367497.1367532