CheetahTraj: Quality and Efficiency in Large-scale Trajectory Data Visual Exploration

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Visualizing large-scale trajectory data is a core subroutine for many smart city applications, e.g., traffic management, urban planning, and route recommendation. The task is challenging due to high scalability requirement and severe visual clutter. Sampling can effectively mitigate both problems but may harm visual quality, i.e., generating visualizations that look different from the exact one. In this work, we propose sampling techniques that provide *theoretically guaranteed visual quality* for large-scale trajectory data visualization. We first define a natural pixel-based *quality loss function* to measure the difference between two visualizations. However, our analysis shows that it is NP-hard to sample a fixed-size subset of trajectories to minimize the loss function. Therefore, we then devise an approximate algorithm with a suite of optimization techniques, which runs efficiently but still provides guaranteed quality loss. By taking data distribution and human perception characteristics into consideration, we further improve it to an advanced algorithm to tackle the visual clutter problem. Extensive experiments (i.e., case-, user-, and quantitative- studies) are also conducted on real-world trajectory datasets to verify the effectiveness and efficiency of our proposals.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Nowadays, the ubiquity of location-acquisition devices leads to an explosive growth of the volume of movement data (i.e., trajectories), e.g., for vehicles, shared bikes and pedestrians. Visualizing these large-scale trajectory data is crucial for many smart city applications [1]–[3] and location-based services [4], [5]. Among various visualization methods, line-based trajectory visualization, which connects the locations of a moving object by polylines, is widely adopted for spatial-temporal data analytics [6]–[8]. However, large-scale line-based trajectory visualization suffers from two challenges–(i) *exploration at interactive rates (scalable analytics)* and (ii) *visual clutter*, which we elaborate as follows [9].

Exploration at interactive rates: The scalable interactivity is crucial in ad-hoc data analysis which requires the visualizations to be generate in milliseconds to seconds [10]. However, trajectory datasets can be extremely large and thus take a long data to visualization time. For example, Shenzhen has 24,237 taxis which collectively generate more than 82.8 million GPS locations each day [11]. Our benchmark on the Porto taxi trajectory dataset shows that it takes 13.95 seconds to render 1 million real-world trajectories using an NVIDIA GeForce GTX 1080 GPU with 8GB video memory. The long delay caused by large dataset cardinality makes interactive visual exploration difficult. Thus, we want visualization to scale to very large datasets while maintaining a short delay.

Visual clutter: It is a common problem for large-scale data visualization, which we illustrate with an example in Figure 1(A). Because of visual clutter, it is almost impossible to recognize the road network in the embedded figure of Figure 1(A), making it difficult for human-users to gain insights from the visualization. Hence, we want the trajectory visualization to be visual-clutter-free for good user experience.

Sampling techniques are widely used for large-scale data analysis in both database and visualization communities to reduce the visual clutter and improve the scalability [12]–[15]. By sampling a subset of records from the raw large-scale dataset, it helps to reduce the data to visualization time. One such example is ScalaR [16], which employs a reduction layer between the visualization layer and the data management layer. The reduction layer samples records *uniformly at random* (denoted as RAND) when the query results are too large.

There are three challenges to design appropriate sampling methods for interactive visualizations: C1) guarantee the visual quality, C2) can be generated efficiently to support the interactive exploration, C3) support different-level of details. However, RAND does not work well for large-scale trajectory data visualization as it cannot provide quality guarantee. Take Figure 1(B) for example, they are the visualization results generated by RAND on the Porto taxi trajectory dataset with sampling rate 1%. Obviously, it is very different from the visualization of the full Porto dataset in Figure 1(A), since the random sampling method is easily ignore the data patterns in the spare areas. Another basic idea of the data reduction is to iteratively remove the trajectories which have the least impact of the whole visualization [17]. Thus impact of a trajectory can be evaluated by sum of the distance(DTW or Fréchet distance) to all other trajectories. As shown by Figure. 1, the visualization generated baseline is much better than $Rand$ by preserve more trajectories in the sparse region, but the general structure is still missing because it cannot theoretically guarantee the visual quality. Moreover the pairwise distance calculation for large trajectory dataset is very time consuming,

which always needs days of preprocessing for millions of trajectories.

In this work, we set out to design efficient sampling algorithms that provides visual quality guarantee for large-scale line-based trajectory visualization. This goal leads to three research problems: (i) *how to measure the visual quality of one visualization result?*(ii) *how to tackle the visual clutter problem in large trajectory visualization?* (iii) *how to devise an efficient sampling algorithm that balances visual quality and clutter from multi-level of details?* To address these problems, we first propose a novel pixel-based *visual quality loss function* to formally measure the difference between two visualizations and show that it is NP-hard to select a sized-k sample of the trajectories to minimize the visual quality loss function. Next, we devise an *visual quality guaranteed algorithm*named VQGS, which provides theoretical visual quality guarantee for the sampled results. Then, we *explicitly tackle the visual clutter problem* by taking data distribution and human perception characteristics into consideration in an advance algorithm named CheetahTraj. Case studies shows CheetahTraj performs very well in the visualization of overview. Last, we maintain a visual quality tree which indexes the trajectories for multi level of regions thus to provide the quality guaranteed trajectory subset for the detail views.

We illustrates the merits of our proposals in Figure 1. Figure 1(D) and (E) are the results of our CheetahTraj algorithm on the Porto dataset with sampling rate 0.1% and 1%, respectively. Compared with the uniform random sampling (i.e., RAND) and distance based sampling(i.e., baseline) counterparts Figure 1(B) and (C), they are obviously more similar to the full dataset visualization in Figure 1(A). Further more, we colored the trajectories shown as Figure 1(F) which is produced by our CheetahTraj algorithm using the same parameters as Figure 1(E) but the trajectories are colored according to their algorithm-generated representativeness (warmer color means more representative), thus the trajectory distribution information in the dense region can be easily observed.

To sum up, our contributions in this paper include:

- We formulate the visual quality-guaranteed sampling problem for large-scale trajectory data visualization, and prove that it is NP-hard in Section III.
- We devise an approximate algorithm VQGS for the visual quality-guaranteed sampling problem with a suite of efficiency optimizations including submodularity and lazy computation in Section IV.
- We propose an advance algorithm CheetahTraj to tackle the visual clutter problem by introducing a perception tolerance parameter, and encoding the representativeness of trajectories using different colors in Section V.
- We conduct extensive experiments on real-world trajectory datasets to demonstrate the superiority of our proposals in Section VI. In particular, nearly 200 real users are recruited to test the effectiveness of our methods on three practical applications.

II. Related work

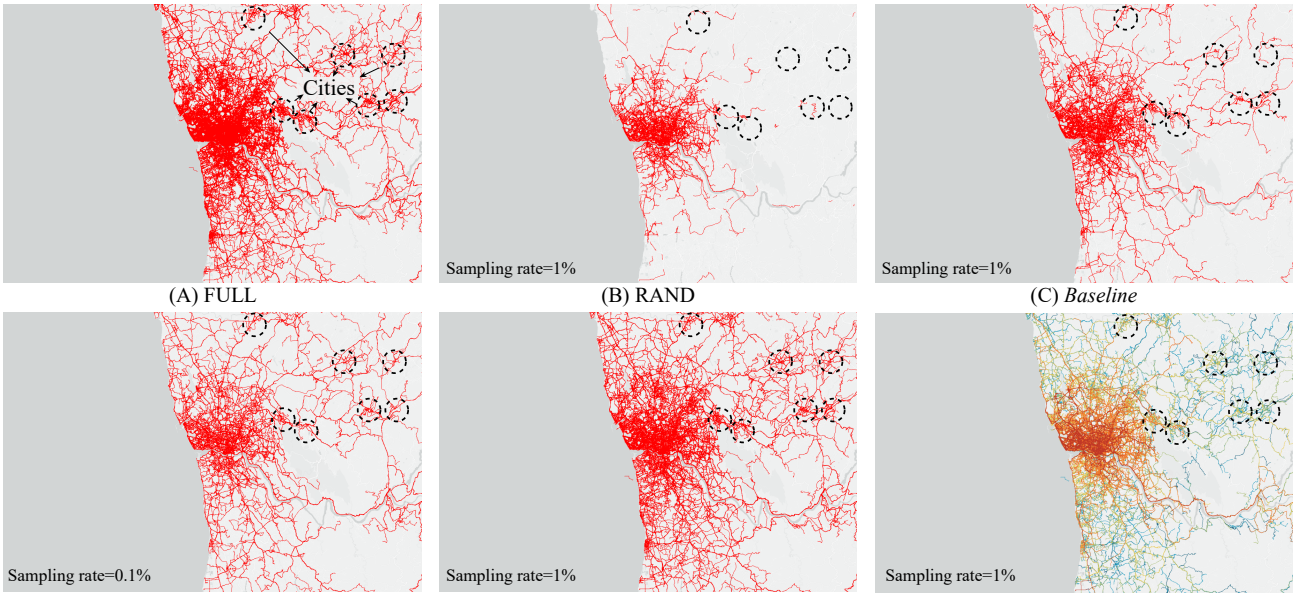
In this part, we survey related works on *trajectory visualization methods* in Section II-A and *interactive data visualization for large datasets* in Section II-B, respectively.

A. Trajectory Visualization Methods

A trajectory is a sequence of spatial locations (e.g., GPS positioning results) and trajectory is the most common representation of object movements. Existing trajectory visualization methods can be classified into three categories according to the form of visualization [6], i.e., *point-based visualization*, *region-based visualization*, and *line-based visualization*. We give a brief introduction to these methods and refer the interested readers to [6] for more detailed discussions. Point-based visualization plots the locations in each trajectory independently and captures the overall spatial distribution of the moving objects. Many density-based methods, e.g., kernel density estimation, are applied in point-based visualization [18]–[22] to preserve the spatial distribution. Region-based visualization slices the entire region into sub-regions and visualizes the aggregated information in each sub-region [23]–[25]. As region-based visualization focuses on aggregated statistics, it is effective in capturing macro-patterns. In this work, we focus on line-based visualization, which uses polylines to connect the locations in each trajectory and shows the trace of object movements (see examples in Figure 3). As it preserves the continuous movement information of objects [26], [27], line-based visualization is widely used in many visual analysis applications such as traffic management, urban planning, and route recommendation. However, line-based visualization is known to suffer from serve visual clutter, especially when the dataset is large. Several techniques have been proposed to alleviate visual clutter, such as clustering-based techniques [25], [28], [29] and advanced interaction techniques [30], [31].

B. Interactive Visualization for Large Datasets

Figure 2 illustrates the general architecture of interactive visualization systems, e.g., Spotfire [32], Tableau [33], ATLAS [34], and Viate [35]. There are typically three layers: user interface in the front-end layer, optimization techniques in the middle-layer, and database management system (usually cloud-based) in the back-end layer. Researchers in the visualization community usually focus on improving the effectiveness of data visualization at the front-end, e.g., designing novel visualization methods/toolkits such as D3 [36] to enable data analysts to gain insights from data effectively. For researchers in the database community, they usually aim to improve query efficiency, e.g., devising big data processing systems such as Spark [37] for efficient data processing at the back-end. With recent developments of location-acquisition technologies, the sizes of available trajectory datasets have become extremely large. For example, the operating taxis in Shenzhen generate $\sim$9.3GB trajectory data per day. However, large datasets increase visualization generation latency due to heavy data processing/graphic rendering, which harms the responsiveness of interactive visualization. Therefore, both visualization and



(A) FULL (B) RAND (C) *Baseline*

(D) CheetahTraj (E) CheetahTraj (F) CheetahTraj (color encoding)

Fig. 1. Effectiveness of CheetahTraj at overview visualization in Porto.

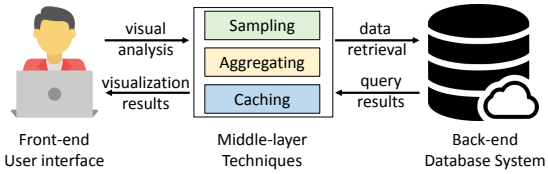


Fig. 2. System architecture for interactive visualization.

database communities began to advance techniques in the middle-layer to reduce the visualization latency for large datasets. We briefly elaborate these techniques as follows.

Aggregating-based techniques: These works [23]–[25] divide the spatial space into basic units and visualize the aggregated information of the trajectories for each unit. For more details on aggregating-based techniques, we refer the reader to the related works [38], [39]. Our problem and solutions are different from these works as we focus on visualizing the raw input data, instead of the aggregated results.

Sampling-based techniques: Sampling techniques are widely used in both visualization and database communities [12]–[16], [40]–[42]. These works try to reduce the dataset to a subset with some special characters: such as the blue noise property **??**, the multi-class property [41] or maximize other optional user-defined quality [41]. The work most relevant to ours is [15], which is designed for scatter plots (a form of point-based visualization). It solves the problem of point overdrawing and preserves the spatial point distribution of the original dataset at the same time. The techniques in [15] cannot be adapted to our trajectory visualization problem as trajectory is more complex than scatter points (e.g., varying lengths, contiguous GPS points). Moreover, [17] discusses the incremental reduction method to sample trajectory dataset: in each iteration, a trajectory with the smallest average dynamic time warping(DTW) distance to every other trajectory is removed. However, the this method has a high time complexity and it is hard to evaluate the "distance" from the perspective of visual similarity.

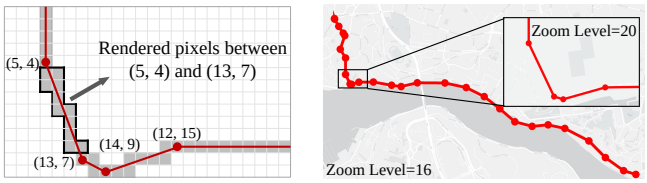
It's worth to mention the area of trajectory simplification [] which is a kind of sampling inside each trajectory.

Caching-based and other techniques: Chan et al. present ATLAS [34], which utilizes caching for efficient data communication between server and client. In addition, ATLAS also exploits a powerful multi-core server to accelerate visual analysis task processing from the middle-layer to the back-end. Piringer et al. [43] propose a multi-threading architecture for interactive visual exploration, which utilizes multi-core devices and avoids the pitfalls of multi-threading to provide quick visual feedback. Our techniques in this work are orthogonal to researches in this category.

Novelties of our work. To the best of our knowledge, we are the first to observe that random sampling harms visual quality for large-scale trajectory visualization and formulate the problem of quality-guaranteed sampling. To tackle this problem, we design a complete framework with quality loss function, theoretical quality loss analysis, and algorithm efficiency optimizations. The well-know visual clutter problem of trajectory visualization is also addressed naturally in our sampling framework. Extensive experimental results show that our proposals effectively maintain visual quality and reduces visualization latency at the same time.

III. Problem Formulation

In this part, we first formally define the *quality-optimal sampling problem* in Section III-A and then show that it is NP-hard to solve the problem exactly in Section III-B.



(A) Line-based visualization (B) Zoom levels

Fig. 3. Illustration of line-based trajectory visualization.

A. Problem Definition

We motivate our definition of the *quality loss function* by introducing how line-based trajectory visualization works. As introduced earlier, a trajectory contains a sequence of 2-dimensional locations. Given an empty canvas (i.e., the screen of a displaying device) with pixels indexed by horizontal and vertical coordinates (i.e., x and y), line-based trajectory visualization connects consecutive locations in each trajectory with polylines and marks the pixels passed by these polylines (with a color different from the background). As shown in Figure 3(A), the result of line-based trajectory visualization can be regarded as a 2-dimensional array of boolean variables with 1 indicting a pixel has been marked. Alternatively, we can treat a visualization result $\mathcal{V}$ as a set that contains all marked pixels. This observation leads to the following definition of the quality loss function

$$loss(\mathcal{V}, \mathcal{V}') = \frac{|\mathcal{V} - \mathcal{V}'|}{|\mathcal{V}|}, \tag{1}$$

in which $|.|$ measures the cardinality of a set, and set $\mathcal{V}^\star = \mathcal{V} - \mathcal{V}'$ contains all distinct elements between $\mathcal{V}$ and $\mathcal{V}'$. We use $loss(\mathcal{V}, \mathcal{V}')$ to measure the quality loss of a visualization result $\mathcal{V}'$ compared to the ground-truth visualization $\mathcal{V}$. This definition matches human visual perception as it is essentially the ratio of different pixels in two visualization results. Thus, the approximate visualization $\mathcal{V}'$ will look similar to $\mathcal{V}$ if $loss(\mathcal{V}, \mathcal{V}')$ is small.

Given the quality loss function, we are ready to define the quality-optimal sampling problem. Denote the set of all trajectories in a dataset as $\mathcal{T}$ and a subset of $\mathcal{T}$ (which contains some sampled trajectories) as $\mathcal{R}$. With a slight abuse of the notation, we use $V(\mathcal{S})$ to denote the visualization results derived from a set $\mathcal{S}$ of trajectories.

Problem 1 (Quality-optimal sampling problem)**.** *Given a sampling rate α, find a set $\mathcal{R}$ that satisfies*

$$\min_{\mathcal{R} \subseteq \mathcal{T}, |\mathcal{R}| = \alpha \mathcal{T}} loss(V(\mathcal{T}), V(\mathcal{R})). \tag{2}$$

If there is a solution for the above quality-optimal sampling problem, to provide a guaranteed quality $loss(V(\mathcal{T}), V(\mathcal{R})) \leq \tau$, we can simply use a binary search to find the smallest α under which the visual quality requirement holds.

One subtlety is that trajectory visualization needs to work at different *zoom levels* upon user request. For example, Google map [44] provides a zoom levels from 0 to 20, with level 0 providing the largest visualization range (i.e., the whole world) but the lowest resolution, and level 20 providing the smallest visualization range (e.g., individual building, if available) but the highest resolution. We also provided an illustration of zoom level in Figure 3(B). Ideally, we want a sample to be *zoom-level-independent*, providing a consistent quality guarantee at different zoom levels. This turns out to be straightforward as trajectory visualization merges several pixels in a high-level result (by pixel-wise OR) to obtain a pixel in a lower-level visualization result. The following theorem shows that it suffices to satisfy the quality guarantee at the highest zoom level.

Theorem 1. *Use $loss(V(\mathcal{T}), V(\mathcal{R}), l)$ to denote the quality loss induced by a sample set $\mathcal{R}$ at zoom level l, we have $loss(V(\mathcal{T}), V(\mathcal{R}), l)$ $\leq loss(V(\mathcal{T}), V(\mathcal{R}), l')$ if $l \leq l'$, with larger l indicating higher resolution.*

Proof. Denote the value of the pixel at location (x, y) for the visualization result of zoom level l as $p^l(x, y)$, which can be either 0 or 1 in line-based trajectory visualization. Recall that pixels of lower zoom levels are obtained by merging pixels from lower zoom levels with pixel-wise OR. Thus, we have $p^l(x, y) = \vee_{(a,b) \in g(x,y,l,l')} p^{l'}(a, b)$, in which $l \leq l'$ and $g(x, y, l, l')$ is the set of pixels in the visualization of zoom level l' that are used to determine $p^l(x, y)$ at zoom level l. Use $p^l_{V(\mathcal{S})}(x, y)$ to denote the $p^l(x, y)$ induced by a set $\mathcal{S}$ of sample trajectories, we have $p^l_{V(\mathcal{T})}(x, y) = p^l_{V(\mathcal{R})}(x, y)$ if $\exists (a, b) \in g(x, y)$ such that $p^{l'}_{V(\mathcal{T})}(x, y) = p^{l'}_{V(\mathcal{R})}(x, y)$. It follows that

$$loss(V(\mathcal{T}), V(\mathcal{R}), l) = \frac{|V^l(\mathcal{T}) - V^l(\mathcal{T})|}{|V^l(\mathcal{T})|} \leq \frac{|V^{l'}(\mathcal{T}) - V^{l'}(\mathcal{T})|}{|V^{l'}(\mathcal{T})|} = loss(V(\mathcal{T}), V(\mathcal{R}), l'). \tag{3}$$

□

We are aware that lower zoom levels need more coarse-grained visualization and thus the sampling rate at the highest level may be larger than necessary. We left more sophisticated designs, such as re-sampling a sample set or generating multiple sample sets for future work, and consider only sampling for the highest zoom level. As our quality loss function considers the entire visible region (i.e., it is a *global quality measure*), the visual difference between our sample and the ground-truth visualization may be large for some specific regions, especially when the marked points are sparse in these regions. *Local visual quality* can be important for some tasks(e.g., outlier discovery [45], [46]) and we leave it for future work. We want to note that a quality-guaranteed sample $\mathcal{R}$ is also *query independent* from the definition of the quality loss function, which means $\mathcal{R}$ only needs to be constructed once to answer all queries.

B. Hardness Analysis

We use $t_i \in \mathcal{T}$ to denote a trajectory in the dataset. According to the working mechanism of line-based trajectory visualization, t_i corresponds to a set of marked pixels on the canvas in the ground-truth visualization and we also use t_i to

denote this set of pixels. Thus, we have $V(\mathcal{T}) = \cup_{t_i \in \mathcal{T}} t_i$ and $V(\mathcal{R}) = \cup_{t_i \in \mathcal{R}} t_i$. We can transform Problem 1 as follows

$$\min_{\mathcal{R} \subseteq \mathcal{T}, |\mathcal{R}| = \alpha |\mathcal{T}|} \frac{|\mathsf{V}(\mathcal{T}) - \mathsf{V}(\mathcal{R})|}{|\mathsf{V}(\mathcal{T})|} \Leftrightarrow \min_{\mathcal{R} \subseteq \mathcal{T}, |\mathcal{R}| = \alpha |\mathcal{T}|} -|\mathsf{V}(\mathcal{R})|$$
$$\Leftrightarrow \max_{\mathcal{R} \subseteq \mathcal{T}, |\mathcal{R}| = \alpha |\mathcal{T}|} |\mathsf{V}(\mathcal{R})| \Leftrightarrow \max_{\mathcal{R} \subseteq \mathcal{T}, |\mathcal{R}| = \alpha |\mathcal{T}|} |\cup_{t_i \in \mathcal{R}} t_i|$$

The above transformations use the fact that $V(\mathcal{R}) \subset V(\mathcal{T})$ as $\mathcal{R} \subset \mathcal{T}$, and the ground-truth set $V(\mathcal{T})$ is constant. The last line shows that quality-optimal sampling is equivalent to the well-known set cover maximization problem. Specifically, given an integer $k = \alpha|\mathcal{T}|$, and a collection trajectory pixel set $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$, set cover maximization finds a subset $\mathcal{R} \subset \mathcal{T}$ such that $|\mathcal{R}| \leq k$ and $|\cup_{t_i \in \mathcal{R}} t_i|$ is maximized. The set cover maximization problem is well-known to be NP-hard [**?**].

IV. Our Solution: VQGS

In this part, we address the second technical challenge: *how to devise an efficient sampling algorithm that provides guaranteed visual quality?* Specifically, we first propose a visual quality-guaranteed sampling algorithm VQGS in Section IV-A. Next, we devise optimizations to improve the efficiency of VQGS in Section IV-B.

A. Visual quality-Guaranteed Sampling

Our visual quality-guaranteed sampling method VQGS is presented in Algorithm 1, which employs a greedy paradigm. In particular, it finds the trajectory tmp in $\mathcal{T}$ that maximizes $|tmp \cup \mathcal{R}|$ at each iteration, as shown in Line 3 of Algorithm 1. It terminates after $k = \alpha|\mathcal{T}|$ iterations and returns $\mathcal{R}$ as the result set for rendering.

Algorithm 1 VQGS$(\mathcal{T}, k = \alpha|\mathcal{T}|)$

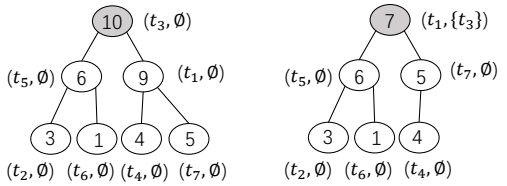
1: Initialize result set $\mathcal{R} \leftarrow \emptyset$
2: **while** $|\mathcal{R}| < k$ **do**
3: $\quad tmp \leftarrow argmax_{t_i \in \mathcal{T}} |t_i \cup \mathcal{R}|$
4: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{tmp\}$
5: **end while**
6: Return $\mathcal{R}$

Interestingly, Algorithm 1 provides provable visual quality guarantee for the returned result $\mathcal{R}$, as stated in Theorem 2.

Theorem 2. *Define the visual quality of a sample set $\mathcal{S}$ as $f(\mathcal{S}) = 1 - loss(\mathsf{V}(\mathcal{T}), \mathsf{V}(\mathcal{S}))$. Let the sized-k result produced by Algorithm 1 be $\mathcal{R}$ and the sized-k solution of Equation* (2) *be $\mathcal{R}^\star$, we have $f(\mathcal{R}) \geq 0.632 * f(\mathcal{R}^\star)$.*

Theorem 2 follows directly from the submodularity of the visual quality function $f(\mathcal{S})$ as it is well known that the greedy solution provides a 0.632 approximation of the optimal solution for a submodular function. We show that submodularity of $f(\mathcal{S})$ as follows.

Lemma 1 (Submodularity)**.** *Define the contribution of a trajectory t to the result set $\mathcal{R}$ as $\Delta(\mathcal{R}, t) = |V(\mathcal{R} \cup t)| - |V(\mathcal{R})|$.*



(A) 1st iteration (B) 2nd iteration

Fig. 4. Lazy computing manner.

Given a trajectory t and two result sets $\mathcal{R}, \mathcal{R}^{'}$, if $\mathcal{R} \subset \mathcal{R}^{'}$, then $\Delta(\mathcal{R}, t) \geq \Delta(\mathcal{R}^{'}, t)$.

Proof. The contribution value of trajectory t w.r.t. a given result set $\mathcal{R}$ (i.e., $\Delta(\mathcal{R}, t) = |V(\mathcal{R} \cup t)| - |V(\mathcal{R})|$) is the newly covered pixels, which can be expressed as $|V(t)| - |V(\mathcal{R} \cap t)|$. We have $t \cap \mathcal{R} \subseteq t \cap \mathcal{R}^{'}$ as $\mathcal{R}^{'}$ as a superset of $\mathcal{R}$, which implies $|V(t)| - |V(t \cap \mathcal{R})| \geq |V(t)| - |V(t \cap \mathcal{R}^{'})|$. Thus, it holds that $\Delta(\mathcal{R}, t) = |V(\mathcal{R} \cup t)| - |V(\mathcal{R})| \geq |V(\mathcal{R}^{'} \cup t)| - |V(\mathcal{R}^{'})| = \Delta(\mathcal{R}^{'}, t)$. □

Although Algorithm 1 provides quality guarantee for the result set $\mathcal{R}$, it has a high time complexity, which we show in the following analysis.

Lemma 2 (Time Complexity)**.** *Given trajectory dataset $\mathcal{T}$ and an integer $k = \alpha|\mathcal{T}|$, the time complexity of Algorithm 1 is $O(\alpha \cdot m \cdot |\mathcal{T}|^2)$, where m is the maximum length of all trajectories in dataset $\mathcal{T}$.*

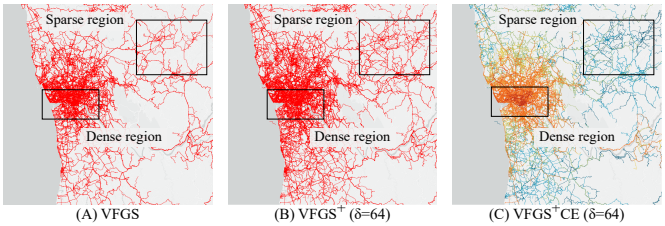
Proof. In each iteration, Algorithm 1 computes the uncovered pixels of each trajectory in dataset $\mathcal{T}$ with $O(m)$ cost. As the dataset $\mathcal{T}$ has $O(|\mathcal{T}|)$ trajectories and Algorithm 1 runs for $k = \alpha|\mathcal{T}|$ iterations, the total cost is $O(k \cdot m \cdot |\mathcal{T}|) = O(\alpha \cdot m \cdot |\mathcal{T}|^2)$. □

The high complexity of Algorithm 1 hurts its scalability for large-scale trajectory datasets. For example, the Porto dataset contains 2.39 millions taxi trajectories, and the longest trajectory has 3,490 GPS points. It takes 413.6 seconds for Algorithm 1 to obtain a result set $\mathcal{R}$ with sampling rate 0.1%. Obviously, the running time is too long for interactive trajectory exploration.

B. Heap-based Lazy Computation

Algorithm 1 essentially adds the trajectory that maximizes $\Delta(\mathcal{R}, t) = |V(\mathcal{R} \cup t)| - |V(\mathcal{R})|$ to $\mathcal{R}$ in each iteration. Lemma 1 shows that the contribution of a trajectory (i.e., $\Delta(\mathcal{R}, t)$) cannot increase when Algorithm 1 runs for more iterations because $\Delta(\mathcal{R}^{'}, t) \leq \Delta(\mathcal{R}, t)$ for $\mathcal{R} \subset \mathcal{R}^{'}$. Based on this property, we can use $\Delta(\mathcal{R}, t)$ calculated in the previous iterations to prune a trajectory t from computation. If we have $\Delta(\mathcal{R}, t) \leq \Delta(\mathcal{R}^{'}, t')$, in which $\mathcal{R}$ and $\mathcal{R}^{'}$ are a previous and the current sample set, respectively, we know that t can not be added to the sample set in the current iteration.

To implement this idea, we maintain a max-heap for the number of uncovered pixels of each trajectory and update the max-heap in a lazy fashion. That is, the contribution of a



(A) VFGS (B) VFGS$^+$ (δ=64) (C) VFGS$^+$CE (δ=64)

Fig. 5. The advanced approach CheetahTraj on Porto ($\alpha = 0.5\%$).

trajectory is computed only when necessary. Figure 4(a) shows a tiny max-heap example for the numbers of uncovered pixels of trajectories from t_1 to t_7 with result set $\mathcal{R} = \emptyset$. At the first iteration, the root node of the max-heap, t_3 in Figure 4(A), is selected. At the second iteration, the number of uncovered pixels of the root node t_1 is updated to 7 w.r.t. result set $\mathcal{R} = \{t_3\}$ (see the gray node in Figure 4(B)). Then t_1 is selected at the second iteration without computing the number of uncovered pixels for other trajectories, i.e., all white nodes in Figure 4(B). The reason is that the contributions of these trajectories are all less than 7 according to the submodularity in Lemma 1.

The efficiency of Algorithm 1 is significantly improved with heap-based lazy computation. To exemplify, Algorithm 1 takes 413.6 seconds to return the results with sampling rate 0.1% on the Porto taxi trajectory dataset. In contrast, our performance-optimized VQGS needs only 1.2 seconds.

V. Advanced Approach: CheetahTraj

In the previous section, we presented the VQGS algorithm, which produces quality-guaranteed samples and runs efficiently. In this section, we focus on the third technical challenge: *how to tackle the visual clutter problem in large trajectory visualization*? In particular, we devise an advanced approach CheetahTraj by considering (i) trajectory data distribution, and (ii) human perception capability. We elaborate (i) and (ii) by the examples in Figure 5.

Trajectory data distribution: Considering the Porto trajectory dataset, Figure 5(A) is the visualization result of VQGS with sampling ratio 0.5%. It is obvious that the trajectories follow a non-uniform distribution, and there are some dense regions and sparse regions as illustrated by the rectangles in Figure 5(A).

Human perception capability: Comparing Figures 5(A) and (B), it is easier to tell their differences in the sparse regions than in the dense regions. This is because human beings have limited perception capability, and hence two visualizations look indistinguishable if both of them contain a large number of points in the same area. This is exactly the case for the two dense regions in Figures 5(A) and (B).

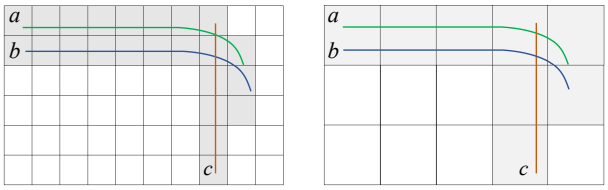
Based on the two observations above, we can improve VQGS by delivering richer information in the sparse regions and reducing visual clutter in the dense regions. CheetahTraj in Algorithm 2 achieves both objectives using a perception tolerance parameter δ, which models the perception capability of humans at the highest level of details. Specifically, if pixel (x, y) in canvas is covered by the result set $\mathcal{R}$ at the highest level, the pixels around (x, y), i.e., from $(x - \delta, y - \delta)$ to $(x + \delta, y + \delta)$, does not need to be covered as they are in the perception tolerance of human beings. We can easily modify VQGS in Algorithm 1 to incorporate the perception tolerance parameter δ as shown in Algorithm 2. CheetahTraj measures the contribution of each trajectory t_i w.r.t the augmented set $\mathcal{R}^+$ in Line 4, where $\mathcal{R}^+$ includes both the selected trajectories and their tolerance pixels (in Line 6).

Algorithm 2 CheetahTraj$(\mathcal{T}, k = \alpha|\mathcal{T}|, \delta)$

1: Initialize result set $\mathcal{R} \leftarrow \emptyset$
2: Initialize augmented result set $\mathcal{R}^+ \leftarrow \emptyset$
3: **while** $|\mathcal{R}| < k$ **do**
4: $\quad tmp \leftarrow argmax_{t_i \in \mathcal{T}} |t_i \cup \mathcal{R}^+|$
5: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{tmp\}$
6: $\quad \mathcal{R}^+ \leftarrow \mathcal{R}^+ \cup \mathsf{augment}(tmp, \delta)$
7: **end while**
8: **for** each t in $\mathcal{T}$ **do** ▷ Representative encoding
9: $\quad tr \leftarrow argmin_{t_i \in \mathcal{R}} |t - \mathsf{augment}(t_i, \delta)|$
10: $\quad tr.\mathsf{cnt} + +$
11: **end for**
12: Return $\mathcal{R}$

VQGS in Algorithm 1 selects trajectories with good representativeness and some trajectories will not be included into the result set $\mathcal{R}$ even though they have more uncovered pixels w.r.t. $\mathcal{R}$. The reason is that their uncovered pixels are too close to the pixels in the selected trajectories (i.e., within the tolerance area of selected pixels). Take Figure 6(A) for example, suppose $\delta = 1$ and trajectory a was selected at the first iteration, the trajectory to select in the second iteration is c instead of b because almost all pixels in b is in the tolerance area of a's. CheetahTraj also provides excellent visual quality at arbitrary zooming resolutions. This is because it considers the perception tolerance parameter δ at the highest zoom level. For example, the zoom level in Figure 6(A) is higher than that in Figure 6(B). According to our elaboration, CheetahTraj selects trajectory a and c for Figure 6(A). When the area is zoomed out, as shown in Figure 6(b), trajectory a and c still captures the main sketch of the underlying dataset (as gray cells shown). We will further elaborate this phenomenon by the experimental study in Section VI.

The visual clutter problem for large-scale trajectory visualization can be further alleviated by encoding the representativeness of the trajectories in $\mathcal{R}$ with colors. We define the representativeness of a trajectory t_i in $\mathcal{R}$ as the number of trajectories in the dataset $\mathcal{T}$ that has t_i as its nearest neighbor in $\mathcal{R}$. The distance between trajectory t and t_i is defined as the number of pixels in t that can not be covered by the augmented pixels of t_i. We compute the representativeness of each trajectory in $\mathcal{R}$ from Line 8 to Line 10 in Algorithm 2. Figure 5(C) shows the visualization result by encoding trajectories with larger representativeness with warmer colors. There are more details in the sparse regions compared with the VQGS result in



(A) High zoom level (B) Low zoom level

Fig. 6. CheetahTraj at different zoom levels.

TABLE I
STATISTICS OF THE DATASETS USED IN THE EXPERIMENTS

Dataset	Trajectories	GPS Points	Maximum Length
Porto	2,390,000	75,670,000	3,490
Shenzhen	3,070,000	53,530,000	2,268
Chengdu	280,000	6,690,000	4,025

Figure 5(A), and we can identify the main roads in the dense region with very warm colors.

VI. Experimental Evaluation

In this part, we first present several case studies of the results provided by CheetahTraj in Section VI-A to demonstrate its good visualization quality. In Section VI-B, we conduct a comprehensive user study to test the effectiveness of CheetahTraj on practical tasks including *region center identification*, *reachable route inspection* and *traffic flow comparison*. In Section VI-C, we quantitatively evaluate the visual quality and efficiency of CheetahTraj and compare with well-designed baselines.

Experiment Settings. We conduct the experiments using three real-world trajectory datasets: Porto, Shenzhen and Chengdu. Porto [47] contains a total of 2.39 million taxi trajectories and 75.67 million of GPS points, and the longest trajectory has 3,490 GPS points. Shenzhen [11] consists of 3.07 million taxi trajectories with 53.53 million GPS points, and the longest trajectory has 2,268 GPS points. Chengdu [48] has 0.28 million taxi trajectories and 6.69 million GPS points, and the longest trajectory consists of 4,025 GPS points. The statistics of the datasets are summarized in Table I. All experiments are conducted on a machine with an Intel i7-8700 CPU, 24 GB memory and an NVIDIA GeForce GTX1080 GPU with 8 GB on-chip memory, running on Windows 10. All methods are implemented using Java 1.8, and the Processing 3 library [49] is used for rendering. All timing results are measured in sing-thread mode. All datasets and source codes required to reproduce our results are available at [50].

Baselines. We compare CheetahTraj with three main baselines, i.e., Full, Random and DTW. Full visualizes all trajectories in the user selected region while Random selects trajectories in the user selected region at random for visualization. DTW is based on the DTW distance between trajectories [17] and designed by us to select trajectories with good diversity. Specifically, DTW samples the trajectory that maximizes the aggregate DTW distance to all remaining trajectories in each step. DTW takes days to run on our datasets because it needs to compute the DTW distance between all trajectory pairs and DTW distance computation has a quadratic complexity w.r.t. trajectory length. In all comparisons, we ensure that Random and DTW use the same number of trajectories as CheetahTraj.

A. Case Study

We conduct case study on the Porto dataset to demonstrate the visualization quality of CheetahTraj from the following aspects.

1) Overview visualization: We analyze the visualization result for overview by considering the entire Porto dataset.

Consistently good visual quality at overview: At zoom level 11, Figure 1(A) is the visualization result of Full on the Porto dataset. With a sampling rate $\alpha = 1\%$, Figures 1(B), (C) and (E) are the visualizations produced by Random, DTW, and CheetahTraj, respectively. Comparing with Figure 1(B) and (C), it is obvious that Figure 1(E) is more similar to Figure 1(A). In particular, Figure 1(E) not only preserves the overall visual structure of the entire region but also keeps the details of cities that are far from the center (marked by the dashed cycles in the figure). However, the details of these cities are lost in Figure 1(B) as Random is more likely to select trajectories in the dense region. DTW in Figure 1(C) preserves more details than Random in the sparse regions but still cannot match the quality of CheetahTraj in Figure 1(E).

Consistently good visual quality under different sampling rates: Figures 1(D) and (E) are the visualizations produced by CheetahTraj with a sampling rate of 0.1% and 1%. We can make two observations: (i) the larger the sampling rate, the better the visual quality, i.e., Figures 1(E) is more similar to Figure 1(A) compared with Figure 1 (D); (ii) the visualization of CheetahTraj with a sampling rate of 0.1% (i.e., Figure 1(D)) looks even more appealing than the visualizations of Random and DTW with a sampling rate of 1% (i.e., Figure 1(B) and (C)) as Figure 1(D) better captures the overall visual structure of Figure 1(A).

Color encoding effectively mitigates visual clutter: At zoom level 11 and with a sampling rate of 1%, Figures 1(E) and (F) are the visualizations produced our CheetahTraj and CheetahTrajCE (i.e., CheetahTraj with color encoding), respectively. Visual clutter is severe for Full (i.e., Figure 1(A)) and CheetahTraj (i.e., Figure 1(E)) as many pixels are colored for the dense region in the center, which makes it difficult to identify the main routes. The visualization of CheetahTrajCE in Figure 1(F) migrates this problem by encoding the trajectories with color, and it is easier to identify some prominent trajectories and busy routes.

2) Detail visualization: We analyze the visualization result for details by investigating two regions of interest in the Porto dataset in Figure 7.

Dense region: At zoom level 15, region B in Figure 7(A) is the center of Porto and has the highest concentration of trajectories. Therefore, Full suffers from severe visual clutter and it is difficult to identify the road networks in in

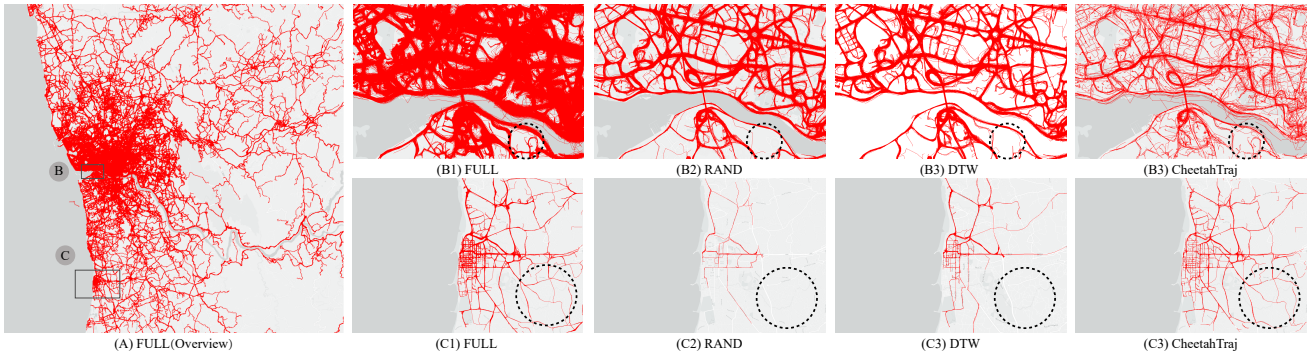


Fig. 7. Case study of the visualization quality of CheetahTraj for two detail regions.

Figure 7(B1). Random and DTW in Figure 7(B2) and (B3) reduce the visual clutter to some extent by sampling some trajectories. CheetahTraj is more successful in reducing the visual clutter of Full and the main road networks are very clear in Figure 7(B4). In addition, CheetahTraj preserves more micro structures of the trajectories than Random and DTW, e.g., the circular route in the dashed circular region.

Sparse region: At zoom level 14, region C in Figure 7(A) contains the city of Casino Espinho and has fewer trajectories than the dense region in the center. In this case, the sampling methods need to keep the structures in the trajectories of Full in Figure 7(C1) to provide good visualization quality. However, Random and DTW in Figure 7(C1) and (C2) fail to meet this requirement, e.g., they show no trajectory in the dashed circle. In contrast, CheetahTraj in Figure 7(C4) preserves these structures.

To sum up, the case study shows that CheetahTraj effectively mitigates visual clutter with sampling and color encoding. Due to the quality-aware VQGS sampling algorithm, CheetahTraj also provides better visualization quality than Random and DTW by preserving more micro structures in the trajectories.

B. User Study

In this part, we conduct a comprehensive user study to evaluate the quality of visualizations generated by different methods.

1) Settings: We recruited ** participants with ** females, ** males, aged **-** with a mean of ** for the user study. The user study is conducted on the two larger datasets, i.e., Porto and Shenzhen, and four methods are investigated, i.e., FULL, RAND, DTW and CheetahTraj. We manually select 22 *center points* in the two datasets and define 3 *visualization scales* including: large-scale region (zoom level less than 13), middle-scale region (zoom level between 13 and 15), small-scale region (zoom level more than 15). For each center point and visualization scale, we generate a *comparable visualization group*, which includes one visualization generated by each of the 4 methods. This results in 66 comparable groups (22 center points $\times$ 3 scales) and 264 visualization results (66 comparable groups $\times$ 4 visualizations).

We are interested in the visual quality and visual clutter of the visualization results, and hence designed three tasks for a comparable group: *T1*) rank the visualizations in the group from the highest visual quality to the lowest visual quality by 1-4; *T2*) rank the visualizations in the group from the least visual clutter to the most severe visual clutter by 1-4; *T3*) select the acceptable visualizations (multiple choices allowed) for analysis and choose the reason for those that are not selected, and we provide three reasons including "severe visual clutter", "poor visual quality" and "others". The user study system is a web-based platform, in which all visualizations are displayed with a resolution of 450*300.

2) User study procedure: When the participants enter the user study system, they are given a brief introduction and a tutorial on how to conduct the tasks to get familiar with the interface and tasks. For each participant, we randomly select 16 comparable groups and generate 48 tasks. For each comparable group, the 4 visualizations (*without specifying generated by which method*) in one comparable group are shown on the same web-page and a participant is required to perform task T1, T2 and T3 by inspecting them.

3) Result analysis: The left plot of Figure 8 reports the quality ranking of 4 methods in T1. The results show that FULL ranks the 1st in most cases while CheetahTraj usually ranks 1st or 2nd. In contrast, DTW and RAND rank 3rd and 4th in most cases. This suggests that CheetahTraj outperforms DTW and RAND in visual quality. We also found that CheetahTraj ranks 1st mainly for large-scale regions in which there are many trajectories.

The right plot of Figure 8 reports the anti-visual clutter ranking in T2. The results show that FULL has the most severe visual clutter, ranking 4th in most cases. RAND and DTW reduce visual clutter via sampling, and thus usually rank 2nd and 3rd. CheetahTraj is the most successful in reducing visual clutter, ranking 1st in 155 out of the xxx cases.

We report the frequency each method is selected as acceptable and why a method is not selected for T3 using bar chart in Figure 9. Each column corresponds to a method and from left to right, the lengths of the bars means "favorable", "not favorable due to visual clutter", "not favorable due to poor visual quality" and "not favorable for other reasons". The results show that CheetahTraj is acceptable in 85% of

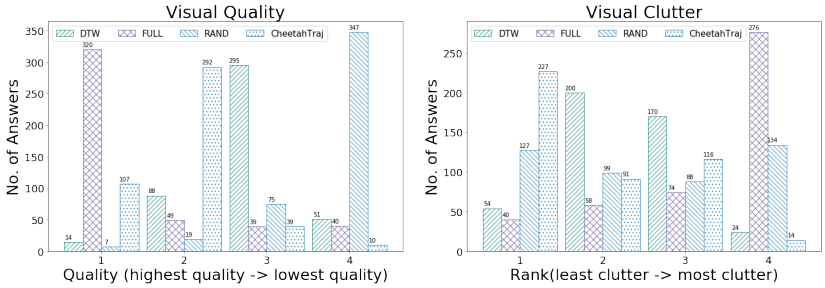


Fig. 8. User study, rank distribution.

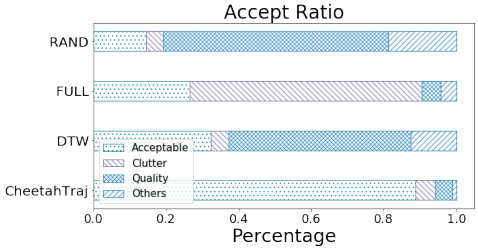


Fig. 9. User study, accept rate.

the cases, and the other methods have significantly lower acceptance rate than CheetahTraj.

C. Quantitative Evaluation

In this part, we quantitatively evaluate our proposals on the three trajectory datasets in terms of both visual quality and efficiency.

Visual quality: Figure 10 reports the visualization quality (as defined in Equation (1)) of the methods under different sampling rate. We consider the entire region in each dataset for this experiment. The results show that both VQGS and VQGS$^+$ achieve significantly high quality than RAND and DTW under the same sampling rate. This is because VQGS and VQGS$^+$ explicitly considers visual quality in the sampling process. Specifically, VQGS$^+$ has a quality close to 1 with a sampling rate less than 1%. DTW has a higher quality than RAND because it considers the diversity of trajectories.

In Figure 10, we report the visualization time (i.e., the time to generate visualization using the sampled trajectories) for the methods under different sampling rates. We still consider the entire region in this experiment and the visualization time of FULL is included at the top of each figure for reference. The results show that all sampling methods achieve significantly shorter visualization time than FULL, which confirms our observation that sampling is effective in improving visualization efficiency. Under the same sampling rate, VQGS and VQGS$^+$ take longer visualization time than RAND and DTW because VQGS and VQGS$^+$ are more likely to select long trajectories for quality maximization. Combining Figure 10 and 10, we can conclude that VQGS$^+$ can achieve high visualization quality with short visualization time.

TABLE II
VISUALIZATION RENDERING COST

No. Trajs	GPS points	Mapping (s)	Rendering (s)	Total (s)
1,000	31,300	0.027	0.003	0.03
10,000	31,6531	0.169	0.005	0.174
100,000	316,7120	1.701	0.057	1.758
1,000,000	31,646,379	15.562	0.592	16.154

We also eventuate the *response time* of the CheetahTraj framework under different quality guarantee and region size in Figure 12. The response time is defined as the time taken to process the user query region to generate the visualization result. We contain the regions to be rectangles with a constant height/width ratio and measures the size of a region by dividing its height over the height of the entire region. Under each region size, we report the average response time of three typical regions, i.e., a dense region, a sparse region and a medium region. The results show that CheetahTraj achieves a short response time (less than 1 second in all cases) for different region size and quality guarantees. The response time decreases rapidly when the region size shrinks as there are fewer trajectories in a smaller region. The response time required to achieve a high quality (e.g., 0.9) is not significantly longer than a low quality (e.g., 0.6) as quality improves quickly with the number of sampled trajectories.

In Figure 13, we report the running time of VQGS$^+$ with and without heap-based lazy computation under different sampling rate. The results show that the heap-based optimizations reduces the running time of VQGS$^+$ by 2-3 orders of magnitude. For the sampling rates we considered, VQGS$^+$ runs efficiently and can finish within 1 second for the entire dataset.

VII. Conclusions and Future Directions

This paper presents a novel sampling technique, CheetahTraj, that guarantees the visual quality of line-based trajectory visualization and alleviates the visual clutter problem. The effectiveness and efficiency of the proposed method are validated with real world visual analysis tasks and quantitatively performance measurements. Possible future directions include (i) improving visual quality by sampling trajectory segments instead of complete trajectories and (ii) developing advanced color encoding schemes to better describe the spatial distribution of the trajectories.

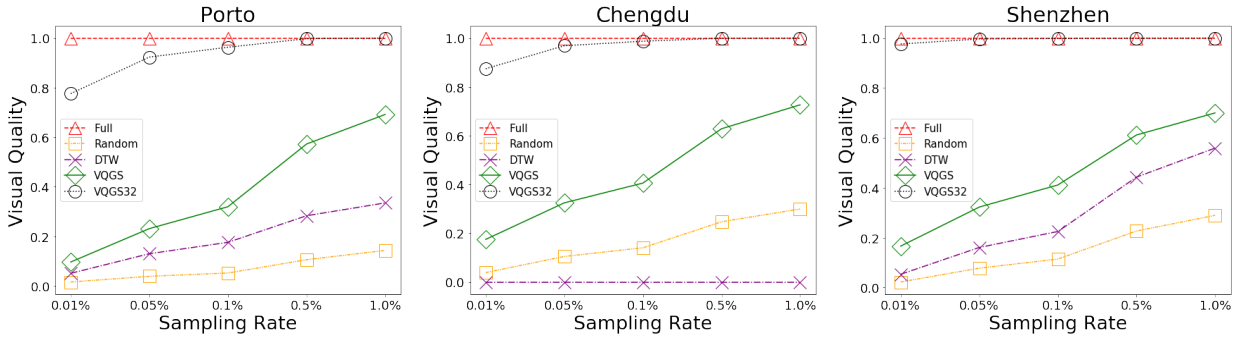


Fig. 10. Visual quality vs. sampling rates(T1).

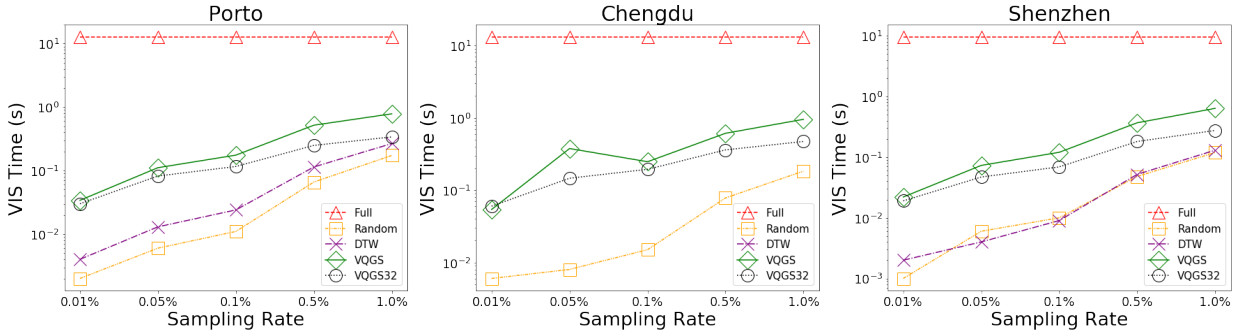


Fig. 11. Processing time vs. sampling rates.

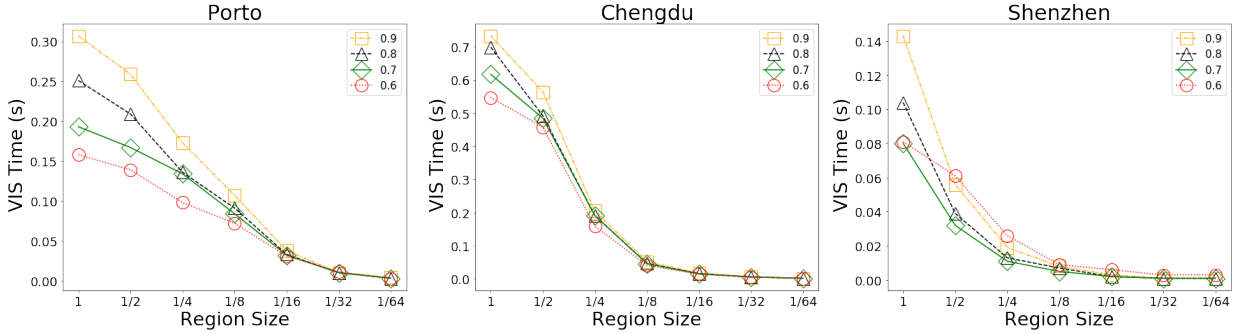


Fig. 12. Response time vs. region size.

REFERENCES

[1] Z. Wang, T. Ye, M. Lu, X. Yuan, H. Qu, J. Yuan, and Q. Wu, "Visual exploration of sparse traffic trajectory data," *TVCG*, vol. 20, no. 12, pp. 1813–1822, 2014.

[2] B. Tang, M. L. Yiu, K. Mouratidis, and K. Wang, "Efficient motif discovery in spatial trajectories using discrete fréchet distance," in *EDBT*, 2017.

[3] Y. Zheng and X. Xie, "Learning travel recommendations from user-generated gps traces," *TIST*, vol. 2, no. 1, pp. 1–29, 2011.

[4] D. Liu, D. Weng, Y. Li, J. Bao, Y. Zheng, H. Qu, and Y. Wu, "Smartadp: Visual analytics of large-scale taxi trajectories for selecting billboard locations," *TVCG*, vol. 23, no. 1, pp. 1–10, 2016.

[5] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with gps history data," in *WWW*, 2010, pp. 1029–1038.

[6] W. Chen, F. Guo, and F.-Y. Wang, "A survey of traffic data visualization," *TITS*, vol. 16, no. 6, pp. 2970–2984, 2015.

[7] G. L. Andrienko, N. V. Andrienko, W. Chen, R. Maciejewski, and Y. Zhao, "Visual analytics of mobility and transportation: State of the art and further research directions," *TITS*, vol. 18, no. 8, pp. 2232–2249, 2017.

[8] G. L. Andrienko, N. V. Andrienko, S. M. Drucker, J. Fekete, D. Fisher, S. Idreos, T. Kraska, G. Li, K. Ma, J. D. Mackinlay, A. Oulasvirta, T. Schreck, H. Schumann, M. Stonebraker, D. Auber, N. Bikakis, P. K. Chrysanthis, G. Papastefanatos, and M. A. Sharaf, "Big data visualization and analytics: Future research challenges and emerging applications," in *EDBT/ICDT joint conference*, ser. CEUR Workshop Proceedings, vol. 2578. CEUR-WS.org, 2020.

[9] B. C. Kwon, J. Verma, P. J. Haas, and C. Demiralp, "Sampling for scalable visual analytics," *IEEE Computer Graphics and Applications*, vol. 37, no. 1, pp. 100–108, 2017.

[10] B. Shneiderman, "Response time and display rate in human performance with computers," *ACM Computing Surveys (CSUR)*, vol. 16, no. 3, pp. 265–285, 1984.

[11] "Shenzhen taxi trajectory dataset," http://jtys.sz.gov.cn/, 2020.

[12] X. Qin, Y. Luo, N. Tang, and G. Li, "Making data visualization more efficient and effective: A survey," *The VLDB Journal*, vol. 29, no. 1, pp. 93–117, 2020.

[13] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang, "Sample

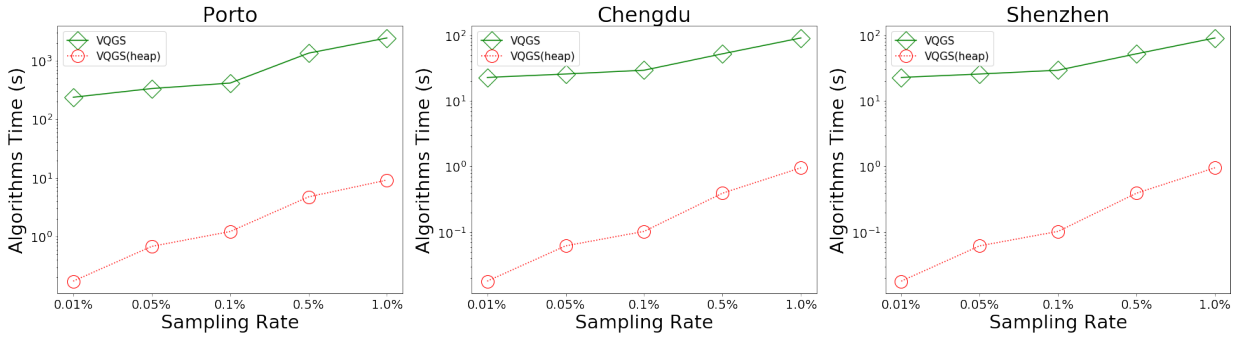


Fig. 13. Algorithm running time vs. sampling rate.

+ seek: Approximating aggregates with distribution precision guarantee," in *SIGMOD*. ACM, 2016, pp. 679–694.

[14] A. Kim, E. Blais, A. G. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld, "Rapid sampling for visualizations with ordering guarantees," *PVLDB*, vol. 8, no. 5, pp. 521–532, 2015.

[15] Y. Park, M. Cafarella, and B. Mozafari, "Visualization-aware sampling for very large databases," in *ICDE*. IEEE, 2016, pp. 755–766.

[16] L. Battle, M. Stonebraker, and R. Chang, "Dynamic reduction of query result sets for interactive visualizaton," in *IEEE International Conference on Big Data*. IEEE, 2013, pp. 1–8.

[17] O. Borcan, "Improving visualization of trajectories by dataset reduction and line simplification," Master's thesis, 2012.

[18] S. Liu, J. Pu, Q. Luo, H. Qu, L. M. Ni, and R. Krishnan, "Vait: A visual analytics system for metropolitan transportation," *TITS*, vol. 14, no. 4, pp. 1586–1596, 2013.

[19] X. Yang, Z. Zhao, and S. Lu, "Exploring spatial-temporal patterns of urban human mobility hotspots," *Sustainability*, vol. 8, no. 7, p. 674, 2016.

[20] J. Chae, D. Thom, Y. Jang, S. Kim, T. Ertl, and D. S. Ebert, "Public behavior response analysis in disaster events utilizing visual analytics of microblog data," *Computers & Graphics*, vol. 38, pp. 51–60, 2014.

[21] Z. Xie and J. Yan, "Kernel density estimation of traffic accidents in a network space," *Computers, environment and urban systems*, vol. 32, no. 5, pp. 396–406, 2008.

[22] G. Borruso, "Network density estimation: a gis approach for analysing point patterns in a network space," *Transactions in GIS*, vol. 12, no. 3, pp. 377–402, 2008.

[23] D. Guo, "Flow mapping and multivariate visualization of large spatial interaction data," *TVCG*, vol. 15, no. 6, pp. 1041–1048, 2009.

[24] J. Wood, J. Dykes, and A. Slingsby, "Visualisation of origins, destinations and flows with od maps," *The Cartographic Journal*, vol. 47, no. 2, pp. 117–129, 2010.

[25] T. Von Landesberger, F. Brodkorb, P. Roskosch, N. Andrienko, G. Andrienko, and A. Kerren, "Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering," *TVCG*, vol. 22, no. 1, pp. 11–20, 2015.

[26] H. Guo, Z. Wang, B. Yu, H. Zhao, and X. Yuan, "Tripvista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection," in *IEEE Pacific Visualization Symposium*. IEEE, 2011, pp. 163–170.

[27] C. Hurter, B. Tissoires, and S. Conversy, "Fromdady: Spreading aircraft trajectories across views to support iterative queries," *TVCG*, vol. 15, no. 6, pp. 1017–1024, 2009.

[28] N. Ferreira, J. T. Klosowski, C. E. Scheidegger, and C. T. Silva, "Vector field k-means: Clustering trajectories by fitting multiple vector fields," in *Computer Graphics Forum*, vol. 32, no. 3pt2. Wiley Online Library, 2013, pp. 201–210.

[29] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko, "Visually driven analysis of movement data by progressive clustering," *Information Visualization*, vol. 7, no. 3-4, pp. 225–239, 2008.

[30] R. Krüger, D. Thom, M. Wörner, H. Bosch, and T. Ertl, "Trajectorylenses–a set-based filtering and exploration technique for long-term trajectory data," in *Computer Graphics Forum*, vol. 32, no. 3pt4. Wiley Online Library, 2013, pp. 451–460.

[31] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva, "Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips," *TVCG*, vol. 19, no. 12, pp. 2149–2158, 2013.

[32] "Spotfire," https://www.tibco.com/products/tibco-spotfire, 2020.

[33] "Tableau," https://www.tableau.com/, 2020.

[34] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan, "Maintaining interactivity while exploring massive time series," in *IEEE Symposium on Visual Analytics Science and Technology*. IEEE, 2008, pp. 59–66.

[35] C. Yang, Y. Zhang, B. Tang, and M. Zhu, "Vaite: A visualization-assisted interactive big urban trajectory data exploration system," in *ICDE*. IEEE, 2019, pp. 2036–2039.

[36] "D3," https://d3js.org/, 2020.

[37] "Apache spark," https://spark.apache.org/, 2020.

[38] G. Andrienko and N. Andrienko, "Spatio-temporal aggregation for visual analysis of movements," in *IEEE symposium on visual analytics science and technology*. IEEE, 2008, pp. 51–58.

[39] N. Adrienko and G. Adrienko, "Spatial generalization and aggregation of massive movement data," *TVCG*, vol. 17, no. 2, pp. 205–219, 2010.

[40] T. Rapp, C. Peters, and C. Dachsbacher, "Void-and-cluster sampling of large scattered data and trajectories," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 780–789, 2019.

[41] 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," *TVCG*, vol. 20, no. 12, pp. 1683–1692, 2014.

[42] J. Yu and M. Sarwat, "Turbocharging geospatial visualization dashboards via a materialized sampling cube approach," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1165–1176.

[43] H. Piringer, C. Tominski, P. Muigg, and W. Berger, "A multi-threading architecture to support interactive visual exploration," *TVCG*, vol. 15, no. 6, pp. 1113–1120, 2009.

[44] "Google map," https://www.google.com/maps/preview, 2020.

[45] D. Feng, L. Kwock, Y. Lee, and R. Taylor, "Matching visual saliency to confidence in plots of uncertain data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 980–989, 2010.

[46] A. Mayorga and M. Gleicher, "Splatterplots: Overcoming overdraw in scatter plots," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 9, pp. 1526–1538, 2013.

[47] "Porto taxi trajectory dataset," http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html, 2020.

[48] "Chengdu didi trajectory dataset," https://outreach.didichuxing.com/app-vue/dataList, 2020.

[49] "The open-source graphical library," https://processing.org, 2020.

[50] "Source code and datasets," https://github.com/ChrisZcu/VFGS, 2020.