CheetahTraj: Efficient Visualization for Large Trajectory Dataset With Quality Guarantee

Qiaomu Shen, Chaozu Zhang, Xiao Yan, Chuan Yang, Dan Zeng, Wei Zeng, and Bo Tang

***Abstract*—Visualizing large-scale trajectory dataset is a core subroutine for many applications. However, rendering all trajectories could result in severe visual clutter and incur long visualization delays due to large data volume. Naively sampling the trajectories reduces visualization time but usually harms visual quality, i.e., the generated visualizations may look substantially different from the exact ones without sampling. In this paper, we propose CheetahTraj, a principled sampling framework that achieves both high visualization quality and low visualization latency. We first define the *visual quality function* measuring the similarity between two visualizations, based on which we formulate the quality optimal sampling problem (QOSP). To solve QOSP, we design the *V*isual *Q*uality *G*uaranteed *S*ampling algorithms, which reduce visual clutter while guaranteeing visual quality by considering both trajectory data distribution and human perception properties. We also develop a quad-tree-based index (IvQuad) that allows using trajectory samples computed offline for interactive online visualization. Extensive experiments including case-, user-, and quantitative-studies are conducted on three real-world trajectory datasets, and the results show that CheetahTraj consistently provides higher visual quality and better efficiency than baseline methods. Compared with visualizing all trajectories, CheetahTraj reduces the visualization latency by up to 3 orders of magnitude while avoiding visual clutter.**

***Index Terms*—Trajectory visualization, interactive data exploration, sampling.**

I. Introduction

THE ubiquity of location acquisition devices leads to an explosive growth of trajectory data such as vehicle traces, shared bike trails, and pedestrian movements. Visualizing these large-scale trajectory data is crucial for many smart city applications such as urban planning [35], traffic management [65], [71], tourism management [29], disease transmission [68] and location recommendation [27], [43], [64]. Among various visualization methods, line-based trajectory visualization, which connects consecutive locations of a moving object with polylines, is widely adopted for spatial-temporal data analytics [9], [17]. Line-based visualization is the most straightforward method for depicting trajectories, offering two key benefits. First, it intuitively displays the path of moving objects, closely aligning with the human mental impression of trajectories. Second, it can clearly highlight outliers when compared to density- [27] and aggregation-based visualization strategies [60]. As applications usually require interactive exploration of trajectory data, it is crucial to generate line-based trajectory visualizations with high quality and low latency.

Problems of visualizing the full dataset: To visualize the trajectories in a target region $\mathcal{Q}$, a straightforward solution is to find the trajectories in $\mathcal{Q}$ and visualize all these trajectories (Full for short). However, Full may take a long time as the dataset can be extremely large. For example, Shenzhen has 24,237 taxis which collectively generate more than 298 thousand trajectories including 7.72 million GPS locations each day [1]. Our profiling results in Table I show that visualizing 1,000,000 taxi trajectories needs 16.154 seconds, which is far more than the 2-seconds maximum delay recommended for interactive exploration [57]. Moreover, Full suffers from causing visual clutter for large-scale trajectory dataset because there are too many lines occluding each other. The clutter makes it difficult for users to gain insights from the visualization. One such example is provided in Fig. 1(A), from which it is difficult to recognize the road networks in the dense center region.

Data sampling for trajectory visualization: Sampling is widely used to accelerate large scale data visualization, particularly when the required precision is limited to the screen and human perceptual resolutions [18], [21], [42], [46]. Previous studies have focused on using sampling techniques to create aggregated visualizations like bar charts [21], [36], binned scatter plots [58], and cell-based heatmaps [60]. These techniques aim to preserve certain features during the sampling process, such as the order [36], value distribution [21] or visual distribution [46]. However, applying these techniques directly to trajectory data is challenging because trajectory data has arbitrary spatial shapes and complex characteristics. The most commonly used sampling method in trajectory dataset is random sampling (denoted as

Manuscript received 31 December 2021; revised 8 January 2024; accepted 28 March 2024. Date of publication 26 April 2024; date of current version 27 September 2024. This work was partially supported in part by Shenzhen Fundamental Research Program under Grant 20220815112848002, in part by the Guangdong Provincial Key Laboratory under Grant 2020B121201001, and a research gift from Huawei Gauss department. Recommended for acceptance by J. Tang. *(Corresponding author: Bo Tang.)*

Qiaomu Shen is with the Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: shenqm@sustech.edu.cn).

Chaozu Zhang, Xiao Yan, and Dan Zeng are with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: zhangcz2021@mail.sustech.edu.cn; yanx@sustech.edu.cn; zengd@sustech.edu.cn).

Chuan Yang is with the Alibaba Group, Hangzhou 311121, China (e-mail: chuck.yc@alibaba-inc.com).

Wei Zeng is with the Hong Kong University of Science and Technology, Guangzhou 510540, China (e-mail: weizeng@ust.hk).

Bo Tang is with the Department of Computer Science and Engineering, Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: tangb3@sustech.edu.cn).

Digital Object Identifier 10.1109/TKDE.2024.3387480

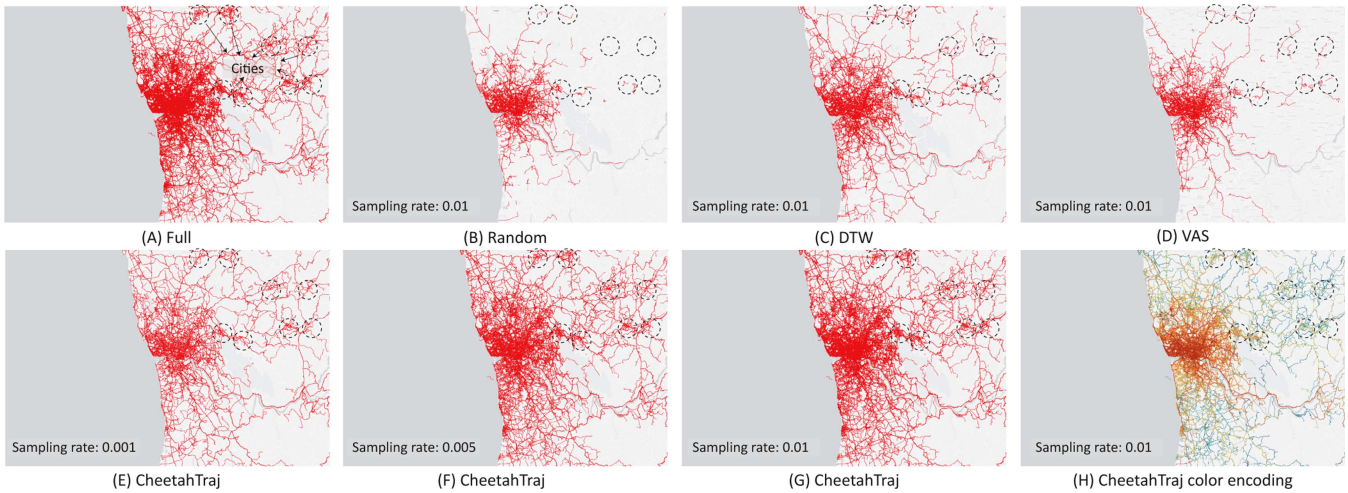


Fig. 1. Comparative visualization results. (A) presents the visualization of the complete Porto taxi trajectory dataset at zoom level 11. (B), (C), and (D) illustrate visualizations generated using uniform random sampling, DTW, and VAS techniques, respectively. (E), (F), (G) and (H) showcase visualizations produced by CheetahTraj employing different parameters, with (H) utilizing color encoding to denote representativeness.

TABLE I
VISUALIZATION TIME ON THE Porto DATASET (IN SECONDS)

No. of trajectories	No. of GPS positions	Mapping time	Rendering time	Total time
1,000	31,300	0.027	0.003	0.03
10,000	316,531	0.169	0.005	0.174
100,000	3,167,120	1.701	0.057	1.758
1,000,000	31,646,379	15.562	0.592	16.154

Mapping time refers to the time for mapping GPS positions to the screen coordinates, while rendering time is the time for rendering the lines on screen.

Random) because it is easy to implement and without the need of trajectory characterization. However, Random has poor visual quality as the generated visualization can be significantly different from the ground-truth, especially in areas where trajectories are sparse. As shown in Fig. 1(B), Random fails to show the cities marked by circles in the remote areas. Moreover, we have developed two baseline methods based on DTW [13] and VAS [46], respectively. The results are shown as Fig. 1(C) and (D). But there are still obvious differences between these two approaches and the ground-truth in Fig. 1(A).

To generate high-quality visualizations with low latency, we introduce CheetahTraj, a framework that encapsulates a novel trajectory sampling algorithm (VQGS) and an efficient indexing method (InvQuad). We first define a *visual quality* function to measure the pixel-based similarity between two visualization images – one rendered from sampled trajectories (denoted as *sampled*) and the other rendered from the full trajectory dataset (denoted as *ground-truth*). We prove that it is an NP-hard problem to select an optimal set of trajectories that maximizes the visual similarity between the *sampled* and the *ground-truth*. Next, we devise a *visual quality guaranteed sampling* algorithm named VQGS, which provides theoretical visual quality guarantee for the sampled trajectories. Then, we *tackle the visual clutter problem* by taking data distribution and human perception into consideration in an advanced algorithm named VQGS$^+$. Furthermore, to enable low latency interactions (e.g., drag, zoom in and out) for the interactive exploration of trajectories, we design an InvQuad-tree index based on quad-tree, which allows using sampling results computed offline for online visualization.

To evaluate the effectiveness and efficiency of CheetahTraj, we conduct two case studies, a qualitative user study, and a quantitative performance comparison on three real-world trajectory datasets. The case studies show that CheetahTraj generates high-quality visualizations for both large and small target regions. The user study with 55 participants confirms that CheetahTraj effectively reduces visual clutter and produces visualizations that are plausible to human inspectors. The quantitative performance comparison shows that CheetahTraj provides good visual quality by sampling only a small proportion of the trajectories, and thus achieves significantly shorter visualization delay than Full. In particular, the visualization delays of CheetahTraj are below 1 s for all target regions on the three trajectory datasets while Full can take more than 10 seconds.

We illustrate the merits of our CheetahTraj framework in Fig. 1. Fig. 1(E), (F) and (G) are the visualizations produced by CheetahTraj on the Porto dataset with sampling rates of 0.1%, 0.5% and 1%, respectively. Compared with baseline approaches in Fig. 1(B), (C) and (D), (E), (F) and (G) are more similar to the full dataset visualization in Fig. 1(A). Fig. 1(H) is produced by CheetahTraj using the same parameters as Fig. 1(G) but the trajectories are colored according to their representativeness generated by our VQGS$^+$ (warmer color means more representative). Compared with Fig. 1(A), the main routes in the dense region can be identified much more easily, which shows that CheetahTraj effectively reduces visual clutter. Last but not least, it takes CheetahTraj only 0.116 seconds and 0.339 seconds to generate Fig. 1(E) and (G), respectively, while the full visualization in Fig. 1(A) takes 16.154 seconds.

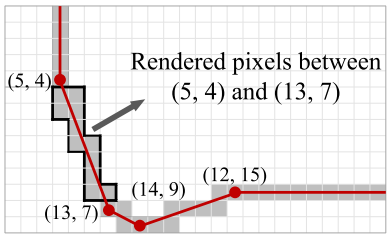


Fig. 2. Illustration of line-based trajectory visualization.

To summarize, the key contributions of this work are outlined as follows:

- *Problem definition:* We have defined a visual quality function for large-scale line-based trajectory visualization. Subsequently, we formulated the problem and conducted a detailed analysis of its complexity, ultimately classifying it as an NP-hard problem.
- *Solution with performance guarantee:* We have developed greedy-based algorithms to address the problem with performance guarantee.
- *Framework for interactive exploration:* We have designed and implemented a framework CheetahTraj based on the algorithms and a quadtree-based index. CheetahTraj is able to support the interactive exploration of large trajectory datasets, enhancing user experience and data engagement.
- *Comprehensive evaluation:* We first conducted two user studies, concentrating on the performance in analytical tasks as well as the empirical evaluation of visualization effects. Following this, we engaged in a quantitative evaluation, analyzing key metrics and time efficiency associated with our methodologies. The collective results from these experiments clearly demonstrate that our approaches outperform the baseline methods.

The remainder of this paper is organized as follows. Section II discusses the background of trajectory visualization and the problem definition. Section III formulates our problem and analyzes its hardness. Section IV provides approximate solutions for it, together with a suite of optimization techniques. Section V proposes a framework supporting the quick visualization of arbitrary regions given by users. Section VI elaborates our extensive experimental studies and our findings in detail. Section VII discusses the related work. Finally, Section VIII concludes this work.

II. Preliminaries

A trajectory $t = \{p_1, p_2, \ldots, p_L\}$ is an ordered sequence of spatial locations (e.g., GPS positions) and we assume that each location p_l (with $1 \leq l \leq L$) is a point on two dimensional plane. Existing trajectory visualization techniques can be classified into three categories according to the form of visualization [17], i.e., *point-based*, *region-based*, and *line-based*. Among them, line-based visualization uses polylines to connect consecutive positions in each trajectory, and one example is provided in Fig. 2. As it preserves the moving process of objects and is the closest to human perception of trajectories, line-based visualization is widely used in applications [10], [37] and we also adapt it to visualize individual trajectories.

In trajectory related applications, users usually handle a large dataset $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ containing many trajectories (e.g., taxi trajectories for a city). Users usually select a region of interest by brushing or zooming-in/out the views to observe the trajectories in the region, for example, to inspect the traffic for certain areas of the city. We denote the user-selected region as $\mathcal{Q}$ (also called query region), and assume that $\mathcal{Q}$ is a rectangle specified by its lower-left and upper-right points on the two dimensional plane. Note that for a trajectory t, $\mathcal{Q}$ may contain only some of its locations. In this case, we only visualize the segment of t that is inside $\mathcal{Q}$. For trajectory visualization, users usually require high quality to support tasks such as identifying road networks, finding congestion and comparing traffic flow. On the other hand, a low visualization latency is also critical such that users can quickly switch between different regions for interactive exploration. We formally define the trajectory visualization problem as follows.

Problem 1 (Trajectory Visualization): Given a trajectory dataset $\mathcal{T}$ and a query region $\mathcal{Q}$, show the trajectories in $\mathcal{Q}$ using line-based visualization.

Given the trajectories in a query region $\mathcal{Q}$, it takes two steps to generate visualization, i.e., *location mapping* and *screen rendering*. Location mapping maps locations in the trajectories to points on the screen while screen rendering renders the mapped points on the screen using graphic devices such as GPU. For both steps, processing complexity is proportional to the number of locations. As we have shown in Table I, large datasets contain many trajectories and thus locations, which lead to long visualization delay. Moreover, visualizing all trajectories in dense areas also results in severe visual clutter as shown in Fig. 1(A). Therefore, we design methods to sample some trajectories in the query region $\mathcal{Q}$ to visualize, with the goal of improving visualization quality and reducing visualization latency at the same time. Note that we assume that the query region $\mathcal{Q}$ is the minimum rectangle that contains all trajectories in Sections III and IV, and will discuss how to handle arbitrary query regions in Section V.

III. The Quality Optimal Sampling Problem

As shown in Fig. 2, 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 marks a pixel if the pixel is passed by at least one trajectory. Thus, the result of line-based trajectory visualization can be regarded as a 2-dimensional array of boolean variables with 1 indicating that a pixel has been marked. Thus, we can treat a visualization result as a set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{m}$ that contains all marked pixels. This observation leads to the following definition of the visualization quality function

$$\mathtt{Q}(\mathcal{S}, \mathcal{S}') = \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S}|}, \tag{1}$$

in which $|.|$ measures the cardinality of a set, $\mathcal{S}$ is the visualization result of the entire trajectory dataset $\mathcal{T}$ while $\mathcal{S}'$ is the visualization result of some trajectories sampled from $\mathcal{T}$. As $\mathcal{S}' \subseteq \mathcal{S}$, $\mathtt{Q}(\mathcal{S}, \mathcal{S}')$ essentially measures the ratio of the pixels in the ground-truth visualization $\mathcal{S}$ that are marked in the approximate visualization $\mathcal{S}'$. This definition matches human visual perception and the approximate visualization $\mathcal{S}'$ will look similar to $\mathcal{S}$ if $\mathtt{Q}(\mathcal{S}, \mathcal{S}')$ is large. With the quality function, we define the quality optimal sampling problem as follows.

Problem 2 (Quality Optimal Sampling Problem,. QOSP): Let the entire trajectory dataset be $\mathcal{T}$ and a sample set that contains some trajectories from $\mathcal{T}$ be $\mathcal{R}$. Using $\mathtt{V}(\mathcal{U})$ to denote the visualization result derived from a trajectory set $\mathcal{U}$, with a sampling rate α, the quality optimal sampling problem finds a set $\mathcal{R}$ that satisfies

$$\max_{\mathcal{R}\subseteq\mathcal{T}, |\mathcal{R}|=\lceil\alpha|\mathcal{T}|\rceil} \mathtt{Q}(\mathtt{V}(\mathcal{T}), \mathtt{V}(\mathcal{R})) = \frac{|\mathtt{V}(\mathcal{T}) \cap \mathtt{V}(\mathcal{R})|}{|\mathtt{V}(\mathcal{T})|}. \quad (2)$$

where, $\lceil\cdot\rceil$ is a ceiling function which is utilized to ensure that the number of trajectories sampled is an integer. Note that we are sampling *complete trajectories* instead of *individual locations* in QOSP such that the lines and orientations in the trajectories are preserved. Given a visualization quality threshold τ, the QOSP problem can be transformed into finding the sampled trajectory set $\mathcal{R}$ with the smallest α that meets the quality requirement, i.e, $\mathtt{Q}(\mathtt{V}(\mathcal{T}), \mathtt{V}(\mathcal{R})) \geq \tau$. For sampling-based methods, the visualization quality is determined by the sample set $\mathcal{R}$, and thus we use $\mathtt{Q}(\mathcal{R})$ to denote $\mathtt{Q}(\mathtt{V}(\mathcal{T}), \mathtt{V}(\mathcal{R}))$ for conciseness.

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 $\mathtt{V}(\mathcal{T})$ and we also use t_i to denote this set of pixels. Thus, we have $\mathtt{V}(\mathcal{T}) = \cup_{t_i\in\mathcal{T}} t_i$ and $\mathtt{V}(\mathcal{R}) = \cup_{t_i\in\mathcal{R}} t_i$. We can transform Problem (2) as follows:

$$\begin{aligned} &\max_{\mathcal{R}\subseteq\mathcal{T}, |\mathcal{R}|=\lceil\alpha|\mathcal{T}|\rceil} \frac{|\mathtt{V}(\mathcal{T}) \cap \mathtt{V}(\mathcal{R})|}{|\mathtt{V}(\mathcal{T})|} \\ &\Leftrightarrow \max_{\mathcal{R}\subseteq\mathcal{T}, |\mathcal{R}|=\lceil\alpha|\mathcal{T}|\rceil} |\mathtt{V}(\mathcal{R})| \Leftrightarrow \max_{\mathcal{R}\subseteq\mathcal{T}, |\mathcal{R}|=\lceil\alpha|\mathcal{T}|\rceil} |\cup_{t_i\in\mathcal{R}} t_i|. \end{aligned} \quad (3)$$

The transformations use the fact that $\mathtt{V}(\mathcal{R}) \subseteq \mathtt{V}(\mathcal{T})$ as $\mathcal{R} \subseteq \mathcal{T}$, and the ground-truth marked point set $\mathtt{V}(\mathcal{T})$ has constant cardinality. The last line shows that QOSP is equivalent to the famous set cover maximization problem.[1] Specifically, given an integer k, and a collection of sets $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$, set cover maximization finds a subset $\mathcal{R} \subset \mathcal{T}$ such that $|\mathcal{R}| = k$ and the number of covered elements $|\cup_{t_i\in\mathcal{R}} t_i|$ is maximized. The set cover maximization problem is well-known to be NP-hard [19].

IV. Solutions for QOSP

In this section, we first present the VQGS algorithm as a solution to QOSP and propose techniques to boost its efficiency. Then we improve VQGS with an advanced algorithm VQGS$^+$ by considering trajectory data distribution and human perception capability.

[1] https://en.wikipedia.org/wiki/Maximum_coverage_problem

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

1: Initialize the result set $\mathcal{R} \leftarrow \emptyset$
2: **while** $|\mathcal{R}| < k$ **do**
3: $\quad \mathtt{tmp} \leftarrow \arg\max_{t_i\in\mathcal{T}} |t_i \cup \mathtt{V}(\mathcal{R})|$
4: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathtt{tmp}\}$
5: $\quad$ Remove trajectory tmp from $\mathcal{T}$
6: Return $\mathcal{R}$

A. Visual Quality Guaranteed Sampling VQGS

Our visual quality guaranteed sampling method (VQGS) is presented in Algorithm 1, which takes the trajectory dataset $\mathcal{T}$ and a sampling rate α as input (i.e., $k = \lceil\alpha|\mathcal{T}|\rceil$). VQGS employs a greedy paradigm and finds the trajectory tmp in $\mathcal{T}$ that maximizes $|\mathtt{tmp} \cup \mathtt{V}(\mathcal{R})|$ at each iteration, as shown in Line 3 of Algorithm 1. It terminates after $k = \lceil\alpha|\mathcal{T}|\rceil$ iterations and returns $\mathcal{R}$ as the result set. As the visualization quality $\mathtt{Q}(\mathcal{R})$ can be computed after each iteration in Algorithm 1 with pre-computed ground-truth $\mathtt{V}(\mathcal{T})$, we can also terminate the algorithm when $\mathtt{V}(\mathcal{R}) \geq \tau$, in which τ is the quality threshold. Algorithm 1 provides provable visual quality guarantee for the result set $\mathcal{R}$, as stated in Theorem 1.

Theorem 1: Given a sample rate α, and let the optimal solution to QOSP defined in (2) be $\mathcal{R}^\star$ and the solution provided by Algorithm 1 be $\mathcal{R}$, we have $\mathtt{Q}(\mathcal{R}) \geq (1 - \frac{1}{e}) * \mathtt{Q}(\mathcal{R}^\star)$.

Theorem 1 follows directly from the submodularity of the visualization quality function $\mathtt{Q}(\mathcal{R})$, and it is well known that greedy solution provides a $(1 - \frac{1}{e})$ approximation of the optimal solution for submodular cost functions [28]. As $\mathtt{Q}(\mathcal{R})$ is a linear scaling of $|\mathtt{V}(\mathcal{R})|$ as shown in (3), we prove that $|\mathtt{V}(\mathcal{R})|$ is submodular as follows.

Lemma 1 (Submodularity): Define the contribution value of a trajectory t to a sample set $\mathcal{R}$ as $\Delta(\mathcal{R}, t) = |\mathtt{V}(\mathcal{R} \cup t)| - |\mathtt{V}(\mathcal{R})|$. Given a trajectory t and two sample 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., $\Delta(\mathcal{R}, t) = |\mathtt{V}(\mathcal{R} \cup t)| - |\mathtt{V}(\mathcal{R})|$) is the number of pixels covered by t but not the trajectory set $\mathcal{R}$, which can be expressed as $|\mathtt{V}(t)| - |\mathtt{V}(\mathcal{R}) \cap \mathtt{V}(t))|$. We have $\mathtt{V}(t) \cap \mathtt{V}(\mathcal{R}) \subseteq \mathtt{V}(t) \cap \mathtt{V}(\mathcal{R}')$ because $\mathcal{R}'$ is a superset of $\mathcal{R}$, which implies $|\mathtt{V}(t)| - |\mathtt{V}(\mathcal{R}) \cap \mathtt{V}(t))| \geq |\mathtt{V}(t)| - |\mathtt{V}(\mathcal{R}') \cap \mathtt{V}(t))|$. Thus, it holds that $\Delta(\mathcal{R}, t) = |\mathtt{V}(\mathcal{R} \cup t)| - |\mathtt{V}(\mathcal{R})| \geq |\mathtt{V}(\mathcal{R}' \cup t)| - |\mathtt{V}(\mathcal{R}')| = \Delta(\mathcal{R}', t)$.

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

Lemma 2 (Time Complexity): For trajectory dataset $\mathcal{T}$ and integer $k = \lceil\alpha|\mathcal{T}|\rceil$, the time complexity of Algorithm 1 is $O(\alpha \cdot m \cdot |\mathcal{T}|^2)$, where m is the length of the longest trajectory in $\mathcal{T}$.

Proof: In each iteration, Algorithm 1 computes the trajectory with the largest number of uncovered pixels in $\mathcal{T}$. It takes $O(m)$ cost to compute the number of uncovered pixels for a trajectory, and Algorithm 1 runs for $k = \lceil\alpha|\mathcal{T}|\rceil$ iterations. Hence, 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

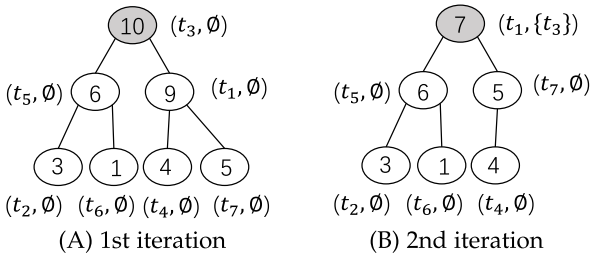


Fig. 3. Heap-based lazy computation.

contains 2.39 million taxi trajectories, Algorithm 1 takes 413.6 seconds to obtain the result set $\mathcal{R}$ with a sampling rate of 0.1%.

Heap-based lazy Computation: Algorithm 1 essentially adds the trajectory that maximizes $\Delta(\mathcal{R}, t) = |\mathtt{V}(\mathcal{R} \cup t)| - |\mathtt{V}(\mathcal{R})|$ 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 $\mathcal{R} \subset \mathcal{R}'$. For example, the contribution of t_1 is 9 at the first iteration (i.e., $\mathcal{R} = \emptyset$), see Fig. 3(A). Its contribution turns 7 with $\mathcal{R} = \{t_3\}$ at the second iteration, as shown in Fig. 3(B). Based on this property, we can use $\Delta(\mathcal{R}, t)$ calculated in the previous iterations to prune it from contribution computation. Specifically, 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 as $\Delta(\mathcal{R}', t) \leq \Delta(\mathcal{R}', t')$ and t' is a better choice. As shown in Fig. 3(B), even if we do not know the exact value of $\Delta(\mathcal{R}' = \{t_3\}, t_7)$, we can conclude that t_7 will not be added to the sample set in the second iteration as $\Delta(\mathcal{R} = \emptyset, t_7) = 5 < \Delta(\mathcal{R}' = \{t_3\}, t_1) = 7$.

To implement this idea, we maintain a max-heap for the number of uncovered pixels in each trajectory and update the contribution of a trajectory only when necessary, i.e., computing in a lazy manner [41]. Specifically, the pixel-based contribution of a trajectory is only updated when the node of this trajectory becomes the root node. Consider a tiny example with 7 trajectories, i.e., t_1 to t_7. Fig. 3(a) shows the initial max-heap and the contributions of trajectories t_1 to t_7 w.r.t. result set $\mathcal{R} = \emptyset$. At the first iteration, the root node of the max-heap, t_3 in Fig. 3(A), is selected. At the second iteration, the number of uncovered pixels of the new root node t_1 is updated to 7 w.r.t. result set $\mathcal{R} = \{t_3\}$ (the gray node in Fig. 3(B)). Then t_1 is selected at the second iteration without computing the contributions of other trajectories w.r.t. $\mathcal{R} = \{t_3\}$. This is because the contributions of these trajectories are smaller than 7 when $\mathcal{R} = \emptyset$, according to Lemma 1, their contributions must be smaller than 7 when $\mathcal{R} = \{t_3\}$. The efficiency of Algorithm 1 improves significantly with heap-based lazy computation. Recall that Algorithm 1 takes 413.6 seconds with $\alpha = 0.1\%$ on Porto while our heap-optimized VQGS needs only 1.2 seconds.

B. *Visualization Aware Solution* VQGS$^+$

In this part, we improve VQGS by considering (i) trajectory data distribution, and (ii) human perception capability. We elaborate (i) and (ii) by the examples in Fig. 4.

Algorithm 2: VQGS$^+(\mathcal{T}, k = \lceil \alpha \mathcal{T} \rceil, \delta)$.

1: Initialize set of sampled trajectories $\mathcal{R} \leftarrow \emptyset$
2: Initialize set of augmented visualized points $\mathtt{V}(\mathcal{R})^+ \leftarrow \emptyset$
3: **while** $|\mathcal{R}| < k$ **do**
4: $\quad \mathsf{tmp} \leftarrow \arg\max_{t_i \in \mathcal{T}} |t_i \cup \mathtt{V}(\mathcal{R})^+|$
5: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathsf{tmp}\}$
6: $\quad \mathtt{V}(\mathcal{R})^+ \leftarrow \mathtt{V}(\mathcal{R})^+ \cup \mathsf{augment}(\mathsf{tmp}, \delta)$
7: $\quad$ Remove tmp from $\mathcal{T}$
8: **for** each trajectory t in $\mathcal{T}$ **do** ▷Representative encoding
9: $\quad\quad tr \leftarrow \arg\min_{t_i \in \mathcal{R}} |t - \mathsf{augment}(t_i, \delta)|$
10: $\quad tr.\mathsf{cnt}++$
11: Return $\mathcal{R}$

Trajectory data distribution: Considering the Porto trajectory dataset, Fig. 4(A) and (B) are the visualization results of Full and VQGS(sampling rate is 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 two rectangles in Fig. 4(A) and (B). There are many points in the dense region, which creates visual clutter and makes it difficult to identify the main roads.

Human perception capability: Comparing Fig. 4(A) and (B), it is easier to tell their differences in the sparse regions than in the dense regions. This is because human perception has limited capability, and hence two visualizations look indistinguishable if both of them contain a large number of points in the same area. The two dense regions look similar although Fig. 4(B) contains fewer trajectories in this region than Fig. 4(A). However, for the sparse region, VQGS loses some trajectories and it is easy to tell the differences between Fig. 4(A) and 4(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. VQGS$^+$ in Algorithm 2 achieves both objectives using a perception tolerance parameter δ, which models the perception capability of human. Specifically, δ is provided by the user and is a constant value that denotes the count of pixels. If pixel (x, y) in the canvas is marked by the result set $\mathcal{R}$, the pixels around (x, y), i.e., from $(x - \delta, y - \delta)$ to $(x + \delta, y + \delta)$, do not need to be marked as they are close to (x, y) and human perception cannot tell nearby pixels apart. We modify VQGS in Algorithm 1 to incorporate the perception tolerance parameter δ in Algorithm 2. VQGS$^+$ measures the contribution of each trajectory t_i w.r.t. the augmented visualized point set $\mathtt{V}(\mathcal{R})^+$ in Line 4, where $\mathtt{V}(\mathcal{R})^+$ includes both pixels on the selected trajectories and their tolerance pixels (in Line 6). We also use the heap-based lazy computation to speed up VQGS$^+$.

VQGS$^+$ in Algorithm 2 selects trajectories with good representativeness and some trajectories will not be included in 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). Compared with VQGS, VQGS$^+$ is more likely to sample trajectories in the sparse regions as their pixels

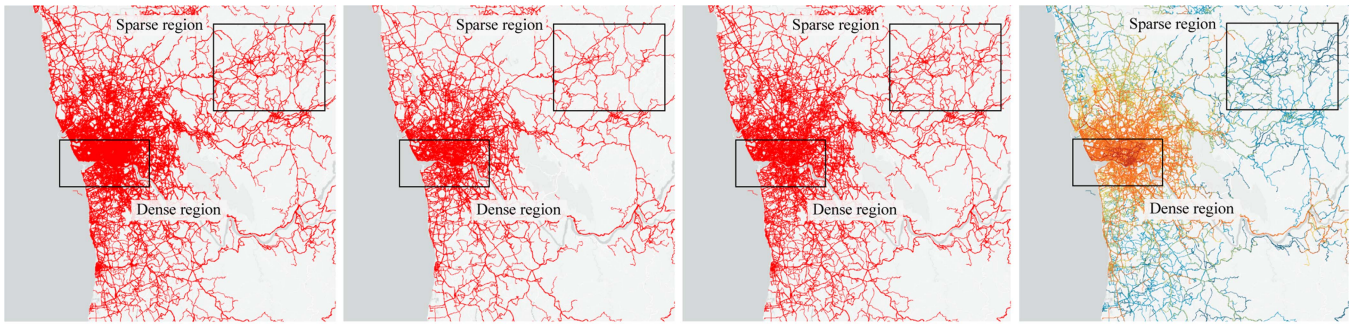


Fig. 4. Visualizations produced by VQGS and VQGS$^+$ on Porto ($\alpha = 0.5\%, \delta = 64$), VQGS$^+$CE encodes more representative trajectories with warmer colors.

are less likely to be covered by other trajectories as shown in Fig. 4(C). This helps preserve the road networks in the spare areas. Moreover, reducing the number of trajectories sampled from the dense regions helps to reduce visual clutter.

In addition, both VQGS and VQGS$^+$ employ a strategy where the next trajectory added is determined by the maximization of the number of covered pixels. Given the submodular nature of the visualization quality function, this greedy strategy ensures the applicability of Theorem 1 to VQGS$^+$. This suggests that VQGS$^+$ provides equivalent performance guarantees as VQGS.

Color encoding scheme: To further augment the expressiveness of our visualization, we have leveraged the gradient color to encode the representativeness of the trajectories in $\mathcal{R}$, which can help to show the spatial crowdedness of trajectories more intuitively. We define the representativeness of a trajectory t_i as the size of its *reverse nearest neighbor set*, which contains the trajectories in $\mathcal{T}$ 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 $\mathcal{R}$ in Lines 8-10 in Algorithm 2. Fig. 4(D) shows the visualization result by encoding trajectories with larger representativeness with warmer colors, for which the main roads in the dense region are clearer than Fig. 4(C) with very warm colors.

V. The CheetahTraj Framework

Recall that our goal is to provide high quality trajectory visualization for any user selected query region $\mathcal{Q}$ with low latency. In this section, we first introduce the motivation behind the CheetahTraj framework, then present its three key procedures, i.e., *index building*, *query processing* and *index updating*.

Motivation of CheetahTraj*:* Given a region query $\mathcal{Q}$, a naive visualization procedure with our sampling algorithms works as follows: it first retrieves all trajectories (or trajectory segments) that are in this region (a.k.a, WayPoint query [37]), then it invokes VQGS$^+$ (or VQGS) to obtain a set $\mathcal{R}$ of sample trajectories, and finally the trajectories in $\mathcal{R}$ are rendered to the canvas (e.g., displaying device). VQGS$^+$ has a short *visualization time* as it effectively reduces the number of processed locations by sampling. However, VQGS$^+$ has a long *sampling time* (e.g., several seconds to tens of seconds) even with our performance optimization techniques. Hence, the naive procedure can not achieve low latency for large-scale trajectory visualization. To tackle this problem, we propose the CheetahTraj framework (illustrated in Fig. 6) which efficiently generates trajectory visualization with user-specified visual quality.

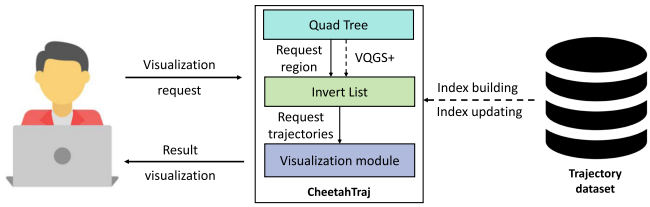


Fig. 5. The pipeline of query processing and CheetahTraj construction.

Fig. 5 illustrates the construction of CheetahTraj and the query pipeline based on CheetahTraj. The construction and updating of CheetahTraj, which is based on the entire trajectory dataset, are performed offline and are represented in Fig. 5 by dashed arrows. The query processing, on the other hand, is performed online: when a user issues a visualization request, CheetahTraj efficiently queries the trajectories through the precomputed quadtree and inverted list, and then delivers the visualization to the users. These online procedures are represented in Fig. 5 by solid arrows.

A. Index Building

The key idea of CheetahTraj is to conduct VQGS$^+$ sampling in the offline *index building* phase such that the sampling results can be used for online visualization. To handle arbitrary query regions, we develop an inverted list augmented quad-tree index (InvQuad-tree).

As shown in the example InvQuad-tree index $\mathcal{I}$ at the bottom of Fig. 6, we exploit a quad-tree to recursively partition the entire area (spanned by the trajectory dataset) into smaller areas and manage each area with a tree node. For each tree node, we run VQGS$^+$ using the trajectories (or trajectory segments) in its associated area as input to compute the *visualization quality inverted lists* for this area. $\mathcal{L}_0$ and $\mathcal{L}_{64}$ in Fig. 6 are two example visualization quality inverted lists, in which the subscripts are the values of perception tolerance parameter δ used for this list.

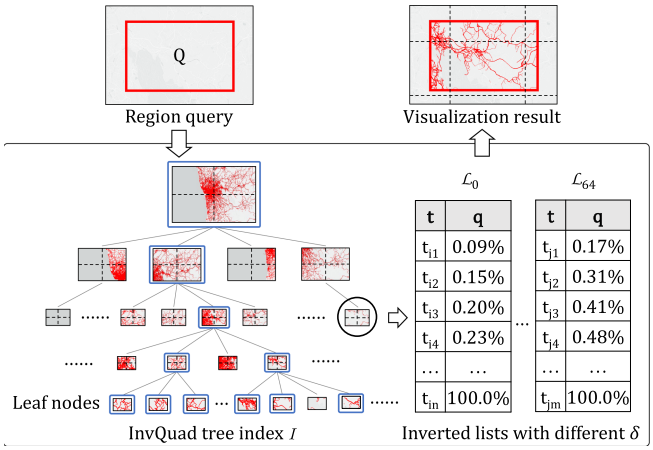


Fig. 6. The CheetahTraj framework.

Specifically, we compute several inverted lists with different δ values[2] to support the efficient quality guaranteed result visualization at various zoom levels. For each inverted list, (i) VQGS$^+$ terminates until the visual quality of the sampled trajectory set is 100%, i.e., the visualization result of the sample set is the same as the full dataset; (ii) the trajectory selected at each iteration of VQGS$^+$ is stored in the inverted list with its *cumulative visual quality* in ascending order. Take inverted list $\mathcal{L}_0$ in Fig. 6 for example, t_{i4} is the trajectory selected at the 4th iteration, cumulative quality of t_{i4} is 0.23%, which means that the quality achieved by $\{t_{i1}, t_{i2}, t_{i3}, t_{i4}\}$ as a whole is 0.23%.

The implementation of the inverted list offers several advantages: (1) For a given region, the inverted list can enhance query efficiency by pre-computing and ranking the trajectories based on their contributions to visual quality. (2) It enables users to define their own acceptable visual quality threshold and efficiently search for the most concise subset of trajectories that meet these criteria.

B. Query Processing

For a region visualization query $\mathcal{Q}$ with a quality threshold τ, Algorithm 3 summarizes the Query subroutine, which finds a quality guaranteed trajectory sample set $\mathcal{R}$. The algorithm starts by invoking Query$(\mathcal{Q}, \tau, \mathcal{I}.root, \mathcal{R} = \emptyset)$, i.e., from the root of InvQuad-tree index $\mathcal{I}$ with an empty result set $\mathcal{R}$, and then transverses the tree nodes recursively. If node $\mathcal{N}$ is a leaf node or its associated area is entirely contained in the query region, we retrieve a quality guaranteed trajectory set by calling subroutine findRet(), which conducts binary search on the proper visualization quality inverted list in $\mathcal{N}$ (Line 2). Otherwise, we call Query() on the four children nodes of $\mathcal{N}$ (Line 3-6). Note that some trajectories in $\mathcal{R}$ (i.e., the result returned by Query() for region $\mathcal{Q}$) may have segments outside

[2]We set δ as 0 (i.e., VQGS), 4, 8, 16, 32, 64. We need quality inverted lists with different δ for one area as the area may be covered by query regions of different sizes, and we use lists with larger δ for larger query region as discussed in Section IV-B.

Algorithm 3: Query($\mathcal{Q}$, τ, InvQuad node $\mathcal{N}$, result $\mathcal{R}$).

```
1: if N is leaf node or N is entirely contained in Q then
2:   R ← R ∪ findRet(N, τ)
3: else if Q ∩ N ≠ ∅ then
4:   for i from 0 to 3 do
5:     tmpQ ← Q ∩ N.child[i]
6:     Query(tmpQ, τ, N.child[i], R)
```

$\mathcal{Q}$, we conduct a way point query WayPoint$(\mathcal{Q}, \mathcal{R})$ to filter these segments before visualization.

Correctness analysis: We first show that CheetahTraj meets the visualization quality requirement in Theorem 2.

Theorem 2: If all selected nodes in the InvQuad-tree index $\mathcal{I}$ are entirely contained in the query region $\mathcal{Q}$, then the result set $\mathcal{R}$ returned by Algorithm 3 satisfies that $\mathtt{Q}(\mathcal{R}) \geq \tau$.

Proof: Suppose query region $\mathcal{Q}$ selects areas $\mathcal{A}_1$,$\mathcal{A}_2$,$\cdots$,$\mathcal{A}_K$, these areas satisfy $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for $i \neq j$, and $\cup_{k=1}^{K} \mathcal{A}_k = \mathcal{Q}$. For each area $\mathcal{A}_k$, denote the number of points marked in the ground truth visualization as n_k, and the number of points marked by the trajectories in $\mathcal{R}$ as m_k, we have $\frac{m_k}{n_k} \geq \tau$ as we use the visualization quality inverted index for trajectory selection. Thus, for query region $\mathcal{Q}$ with result set $\mathcal{R}$, we have $\mathtt{Q}(\mathcal{R}) = \frac{\sum_{k=1}^{K} m_k}{\sum_{k=1}^{K} n_k} \geq \tau$.

In more general cases, we also select some areas that intersect with the query region $\mathcal{Q}$ and the sample set $\mathcal{R}_k$ may not satisfy $\frac{m_k}{n_k} \geq \tau$ for these areas. This does not significantly affect visualization quality for two reasons. (i) These areas are the leaf nodes of the InvQuad-tree index and thus reside on the border of the query region. For visual exploration, human tends to move the region of interest to the screen center, where is more "close" to eyes [2]. (ii) The areas of the border regions are small w.r.t. the query region if the InvQuad-tree has a sufficient height (i.e., the leaf nodes have a small area).

C. Handling Updates

In practice, new trajectories may be continuously added to the dataset $\mathcal{T}$ when they are collected. CheetahTraj uses three different strategies to handle updates according to the volume of updates.

- If a small number of trajectories are added (e.g., several thousand), CheetahTraj first generates trajectory samples for the original dataset $\mathcal{T}$ using the InvQuad-tree index as we have discussed. Then, for each added trajectory t that passes the query region $\mathcal{Q}$, CheetahTraj calculates the percentage of pixels in t covered by the trajectory samples (denoted as t_p). If t_p is below the quality threshold τ, t is added to the trajectory samples.
- When a considerable number of trajectories are added, CheetahTraj collects them into a new dataset $\mathcal{T}'$ and builds a InvQuad-tree index $\mathcal{I}'$ for $\mathcal{T}'$. CheetahTraj accesses $\mathcal{I}'$ and the InvQuad-tree index $\mathcal{I}$ of the original dataset $\mathcal{T}$ to obtain two sets of sample trajectories, and merges them as the final result. We can show that the visualization result

TABLE II
STATISTICS OF THE DATASETS USED IN THE EXPERIMENTS

Dataset	No. of Trajectories	No. of GPS points	Maximum length
Porto	2,389,863	75,667,503	3,490
Shenzhen	3,066,861	53,527,890	2,268
Chengdu	2,400,000	80,040,361	6,468

satisfies the quality threshold τ following the proof of Theorem 2.

- When the number of added trajectories is comparable to the original dataset (e.g., 10%), CheetahTraj rebuilds the InvQuad-tree index for the updated dataset. As we will show in Section VI-C, building InvQuad-tree index only takes several minutes for large trajectory datasets.

VI. EXPERIMENTAL EVALUATION

We first present a case study of the visualizations provided by CheetahTraj in Section VI-A to demonstrate its good visualization quality. In Section VI-B, we conduct two comprehensive user studies to compare the visualization of different methods. In Section VI-C, we quantitatively evaluate the visual quality and efficiency of CheetahTraj.

Experiment Settings: We use 3 real-world trajectory datasets, i.e., Porto, Shenzhen and Chengdu, and their statistics are summarized in Table II. The 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. UnfoldingMap 0.9.92 [5] is used to provide interactive map and GPS mapping, and the Processing 3 library [3] is used for rendering. All timing results are measured in single-thread mode. The datasets and source codes to reproduce our results are available at [4].

Baselines: We evaluate CheetahTraj in comparison with four baselines, namely, Full, Random, DTW, and VAS. The Full method visualizes all trajectories within a user-selected region. Conversely, the Random approach uniformly and randomly samples trajectories within the same region. The DTW method, based on the Dynamic Time Warping (DTW) distance between trajectories, is outlined in Borcan et al. [13]. Lastly, the VAS method, proposed by Park et al. [46], is the only method thus far designed to accelerate visualization using sampling techniques with a guaranteed visual quality. In particular, we have developed our own implementation of the algorithm described in [13], as the authors did not provide the corresponding source code. Our approach involves a process where, in each iteration, we identify and eliminate the trajectory with the smallest average DTW distance in relation to all other trajectories within the dataset. It's important to mention that running the DTW algorithm on the experimental datasets took several days due to its computational intensity. This is because the algorithm computes the DTW distance, which has quadratic complexity in terms of the length of the trajectories, for every possible pair of trajectories. As for the VAS approach, the original technique is intended for scatter plot-based visualization. We have adapted this by employing the same computational framework but modifying the distance calculation among objects to use the discrete FrÃ©chet distance [24], a method that is widely used in movement analysis [40]. For a fair comparison, we ensure that Random, DTW and VAS use the same number of trajectories as CheetahTraj.

A. Case Study

We conduct the case studies on the Porto dataset to demonstrate the visualization quality of CheetahTraj. The observations are similar for the other datasets and we omit their results for conciseness.

1) Overview Visualization: We illustrate the visualization results of different methods for the entire Porto dataset in Fig. 1.

Good visual quality for overview: Fig. 1(A) is the visualization result of Full on the Porto dataset. With a sampling rate $\alpha = 1\%$, Fig. 1(B), (C), (D) and (G) are the visualizations produced by Random, DTW, VAS, and CheetahTraj, respectively. Compared with Fig. 1(B), (C) and (D), it is obvious that Fig. 1(G) is more similar to Fig. 1(A). In particular, Fig. 1(G) 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). However, the details of these cities are lost in Fig. 1(B) as Random mostly selects trajectories in the dense regions. VAS is better than Random by maintaining very few trajectories in the sparse regions. DTW in Fig. 1(C) preserves more details than Random and VAS in the sparse regions as it considers diversity in trajectories but its visualization quality is still inferior compared with CheetahTraj in Fig. 1(G).

Good visual quality under different sampling rates: Fig. 1(E), (F) and (G) are the visualizations produced by CheetahTraj with a sampling rate of 0.1%, 0.5% and 1%, respectively. We can make two observations: (i) the larger the sampling rate, the better the visual quality, i.e., Fig. 1(G) is more similar to Fig. 1(A) than Fig. 1(E) and (F); (ii) the visualization of CheetahTraj with a sampling rate of 0.1% (i.e., Fig. 1(E)) looks more appealing than the visualizations of Random, DTW and VAS with a sampling rate of 1% (i.e., Fig. 1(B), (C) and (D)) as Fig. 1(E) better preserves the structures in Fig. 1(A).

Color encoding effectively enhances the expressiveness: Fig. 1(G) and (H) are the visualizations produced by our CheetahTraj and CheetahTrajCE (i.e., enabling color encoding), respectively. Visual clutter is severe for Full (i.e., Fig. 1(A)) and CheetahTraj (i.e., Fig. 1(G)) as many pixels are visualized for the dense region in the center, which makes it difficult to identify the main routes. The visualization of CheetahTrajCE in Fig. 1(H) enhances the expressiveness of visualization by encoding more representative trajectories with warmer color. This enhancement facilitates the identification of main routes, providing a more insightful visualization compared to Fig. 1(A) and (G).

2) Detail Visualization: We analyze the visualizations produced by different methods for small areas with details by investigating two regions of interest in the Porto dataset in Fig. 7.

Reduce visual clutter and preserve micro structures for dense region: Region B in Fig. 7(A) is the center of Porto and has

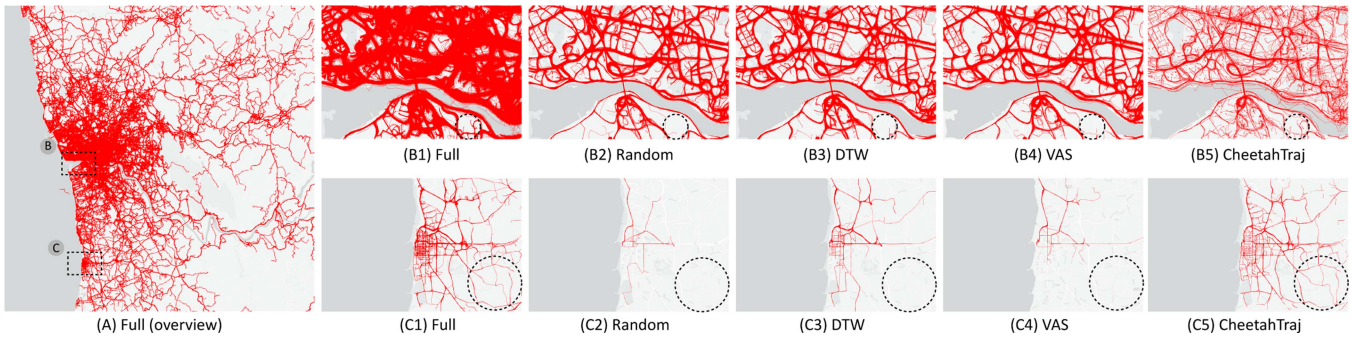


Fig. 7. Case study of the visualization quality of CheetahTraj for two detail regions.

the highest concentration of trajectories. Therefore, Full suffers from severe visual clutter and it is difficult to identify the road networks in Fig. 7(B1). Random DTW and VAS in Fig. 7(B2), (B3) and (B4) reduce the visual clutter to some extent by sampling some trajectories. CheetahTraj in Fig. 7(B5) is more successful in reducing the visual clutter and allows to identify a much larger number of routes. In addition, CheetahTraj preserves more micro structures of the trajectories than Random, DTW and VAS, e.g., the routes in the dashed circular region.

Preserve overall layout for sparse region: Region C in Fig. 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 overall layout of the trajectories to provide good visualization quality. Compared with Full in Fig. 7(C1), Random, DTW and VAS in Fig. 7(C2), (C3) and (C4) fail to meet this requirement as they do not show any trajectories in areas far from the city, e.g., in the dashed circle. This makes their visualization layout very different from Full. In contrast, CheetahTraj in Fig. 7(C5) preserves the trajectories in areas far from the city and has an overall layout similar to Full.

To sum up, the case study shows that CheetahTraj effectively mitigates visual clutter for dense regions. In addition, CheetahTraj also provides better visualization quality than Random, DTW and VAS by preserving the micro structures and overall layout of full visualization.

B. User Study

In this section, we present two user studies to evaluate CheetahTraj by assessing user performance and gathering feedback during trajectory analysis tasks. In the first study, participants were tasked to utilize trajectory visualizations to complete tasks designed based on three analytical goals. The second study is an empirical evaluation, focusing on participants' perceptual feedback of different visualizations.

The user study system is a web-based platform, in which all visualizations are displayed with a resolution of 450×300. The studies were conducted on the Porto dataset and Shenzhen. For a comprehensive comparison, our methods CheetahTraj and CheetahTrajCE were evaluated against three baselines (i.e., Random, DTW, VAS) and the visualization of full dataset Full.

We have recruited 55 participants from research lab in our university and a corporate research institution for these two studies. Out of these, 35 were assigned to the first study and another 35 to the second, with some participants overlapping in both studies. They have research experience about database system, spatial temporal data analysis. None of the participants had specific prior knowledge of our research project. Notably, 15 participants took part in both studies, yet their performance was comparable to those who participated in only one study.

1) Analytical Tasks Targeted at Visualization Goals: In the first study, we selected three widely recognized visualization goals of trajectory visualization from existing research: *region center identification* [33], [55], *reachable route inspection* [37], [44], and *traffic flow comparison* [37]. Participants were tasked with performing tasks related to these goals using visualizations created by both our methods and baseline approaches. Their performance across these tasks was then systematically compared to evaluate effectiveness.

Settings: Of the 35 participants who performed the first study, there were 7 females and 28 males, with ages ranging from 20 to 25 and an average age of 23.5. The user study was conducted using the Porto dataset. We invited domain experts to manually identify city centers. From there, we randomly selected 50 regions at various zoom levels and removed those with insufficient city centers (fewer than 3). This process resulted in 32 regions being selected for further study. For each region, we generated 6 views using 6 methods (namely, CheetahTraj, CheetahTrajCE, Full, Random, DTW, and VAS), leading to a total of 192 views. Each participant was asked to complete three analytical tasks (ATs) related to the three goals.

- *AT1. Region center identification:* The center of a city or commercial region is crucial for traffic management. This significance is reflected in taxi trajectories, where the centers typically exhibit a higher frequency of taxis compared to surrounding regions, forming a star-shaped cluster in visualizations. For this task, each participant was presented with 5 randomly selected views. Their task was to identify the correct city or region center(s) in each view, distinguishing them from randomly generated incorrect centers. As illustrated in Fig. 8 (AT1), participants were required to identify 3 accurate centers out of 6 circles by clicking the corresponding circles.
- *AT2. Reachable route inspection:* The visualized trajectories effectively represent reachable routes connecting different regions. In this task, participants were provided

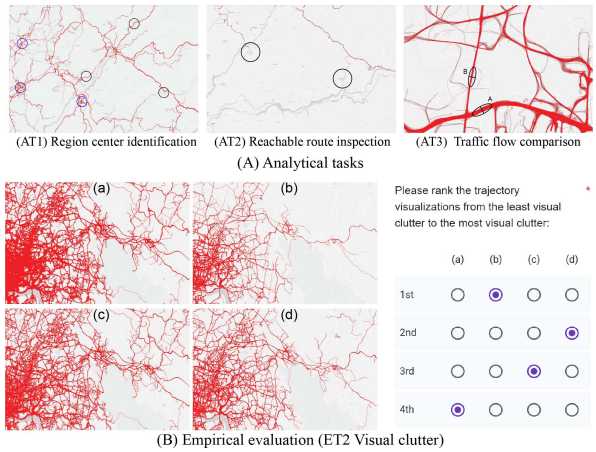


Fig. 8. Illustration of tasks in user study.

with a visualization view displaying two circular areas representing city centers, as depicted in Fig. 8 (AT2). The task involved inspecting the most representative reachable routes between these two areas. It was assumed that routes with a higher number of trajectories were more representative. For each visualization, the total number of reachable routes was specified. Each participant was given 5 randomly selected views, each containing two or more city or commercial centers. Within each view, two identical circular areas were randomly chosen, and participants were asked to examine the connecting routes.

- *AT3. Traffic flow comparison:* In practice, a road with a heavy traffic flow is typically characterized by a larger number of passing trajectories, resulting in denser and broader trajectory clusters in visualizations. When color encoding is applied in trajectory visualization, such dense traffic patterns can be further accentuated by a concentration of darker-colored trajectories. In this task, as illustrated in Fig. 8 (AT3), participants were required to compare traffic flows on two roads based on the visualization results. Specifically, they were asked to select the road with heavier traffic flow by clicking the corresponding radio button. An option to select "I am not sure" was also available in cases of uncertainty. This task comprised 5 randomly selected regions. The number of passing trajectories on each road was counted to determine the actual traffic flow.

User study procedure: In the user study, we first provided a brief introduction about the motivation and visual encoding scheme, followed by three tasks. In each task, we included a tutorial (with the correct result) to help the participants familiarize themselves with the interface, interaction, and tasks. We then randomly chose different views with different questions for each task and for each participant. After they completed all the questions, their answers were collected and saved in the database for the result analysis.

Result analysis: Fig. 9 dipicts the average accuracy achieved by six distinct approaches across three analytical tasks. In both the *region center identification* (AT1) and *reachable route inspection* (AT2) tasks, the average accuracy of our proposed methods (CheetahTraj and CheetahTrajCE) is significantly

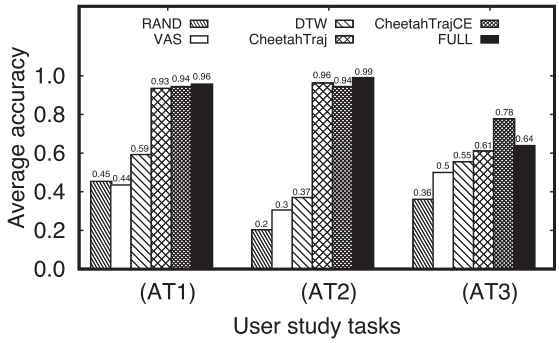


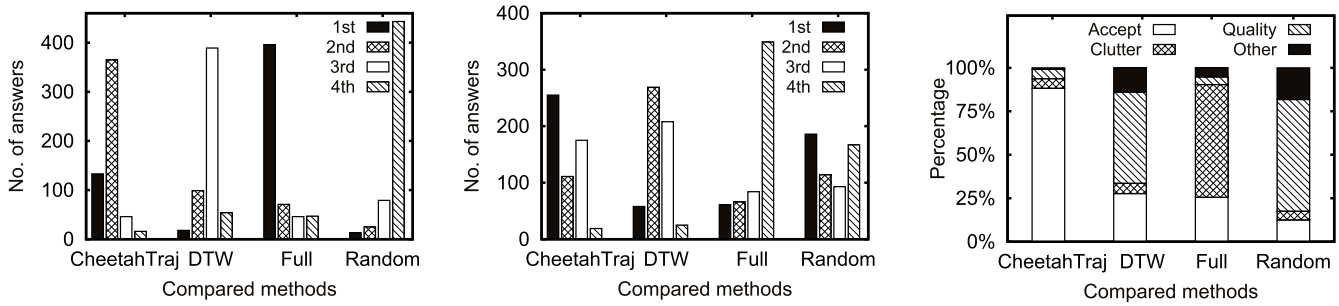
Fig. 9. User study results of analytical tasks.

higher compared to the baseline approaches (Random, DTW, and VAS). This improvement is largely due to the fact that the baseline methods are lack of the ability to maintain detailed trajectory information, especially at more refined levels of visualization. Interestingly, our approaches exhibit performance closely matching that of visualizing the full dataset (Full). This suggests that the visualization generated by our methods is as effective as those obtained from full data visualization in these analytical tasks. However, in task AT2, we noted a slightly lower performance for CheetahTrajCE compared to CheetahTraj. In the post-interviews, several participants noted that the color scheme used in the trajectories could potentially divert a user's focus, making it more challenging to identify the most representative bunches of trajectories.

Visually, the task of *traffic flow comparison* (AT3) presents a greater challenge than the other two tasks, leading to a generally lower average accuracy across all methods. Notably, the average accuracy achieved with the full dataset visualization (Full) is lower than that of CheetahTrajCE. In the post-interviews, participants noted that CheetahTrajCE effectively highlighted crowded road segments through color coding, contributing to its higher average accuracy in this task.

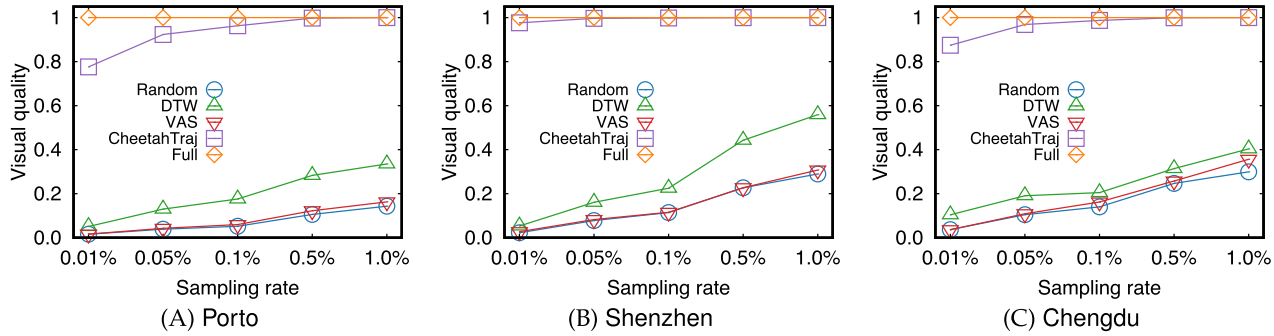
2) Empirical Evaluation of Visualization Effects. Settings: The 35 participants comprised 10 females and 25 males, ranging in age from 19 to 31, with an average age of 24.78. The user study is conducted on the Porto and Shenzhen datasets, and four visualization methods are investigated, i.e., Full, Random, DTW and CheetahTraj. We define a *comparable group* as a group of visualizations generated by different methods for the same query region. 80 comparable groups are generated automatically in random, and 14 groups that contain no trajectories are manually removed. Thus, we collect 66 comparable groups and 264 visualization results (66 groups $\times$ 4 visualization methods). We are interested in the visual quality and visual clutter of the visualizations, and thus designed three tasks for a comparable group.

- *ET1* rank the visualizations in a group from the highest visual quality to the lowest visual quality.
- *ET2* rank the visualizations in a group from the least visual clutter to the most visual clutter.
- *ET3* select the visualizations considered acceptable (multiple choices allowed) and choose the reason (including "severe visual clutter", "poor visual quality" and "others") for the visualizations considered unacceptable.



(A) ET1: visual quality study (B) ET2: visual clutter study (C) ET3: acceptance study

Fig. 10. Results of empirical evaluation.



(A) Porto (B) Shenzhen (C) Chengdu

Fig. 11. Effect of sampling rate on visual quality.

User study procedure: When the participants enter the user study system, they are given a tutorial to introduce the background of our research including the key concepts in our research such as "visual quality" and "visual clutter". Additionally, they are instructed on how to carry out the tasks, which helps them to get familiar with the interface and tasks at hand. For each participant, we randomly select 16 comparable groups. For each comparable group, the 4 visualizations (without specifying generated by which method) in it are displayed on the same page and a participant is required to perform tasks ET1, ET2 and ET3 by inspecting them. An example of ET2 is shown in Fig. 8(B).

Result analysis: Fig. 10(A) reports the visual quality ranking of the 4 methods in ET1. The results show that Full ranks 1st in most cases while CheetahTraj usually ranks 1st or 2nd. In contrast, DTW and Random rank 3rd and 4th at most times. This suggests that the visualizations generated by CheetahTraj are more appealing to the participants than DTW and Random. Fig. 10(B) reports the anti-visual clutter ranking of the 4 methods in ET2. The results show that visual clutter is most severe for Full, ranking 4th in most cases. While CheetahTraj is the most successful in reducing visual clutter, ranking 1st in 255 out of the 560 cases and ranking 4th for only 19 cases. We report the acceptance frequency of the 4 methods in T3 using bar chart in Fig. 10(C). Each column corresponds to a method, and from bottom to top, the lengths of the bars indicate the percentage of participants choosing "*acceptable*", "*not acceptable due to visual clutter*", "*not acceptable due to poor visual quality*" and "*not acceptable for other reasons*". The results show that CheetahTraj is regarded as acceptable for about 88.2% of the cases, and the other methods have a much lower acceptance rate than CheetahTraj.

C. Quantitative Evaluation

In this part, we quantitatively evaluate the visual quality and efficiency of CheetahTraj on the three real-world trajectory datasets in Table II.

Visual quality: Fig. 11 reports the visualization quality (defined in (1)) of the trajectory sampling methods under different sampling rates. We consider the entire region in each dataset for this experiment. The results show that CheetahTraj achieves significantly higher quality than Random, DTW, and VAS under the same sampling rate. This is because the sampling algorithms in the CheetahTraj framework are designed with explicit considerations for visual quality. Specifically, the quality of CheetahTraj approaches 1 when the sampling rate is still less than 1% for all 3 datasets. DTW and VAS have higher quality than Random because they consider the diversity of trajectories.

Furthermore, we employ another widely used metric, NMI (Normalized Mutual Information), to assess the effectiveness of our approach. We chose NMI because it is designed to better capture the perceived changes of the visual content [45], [71], which is essential for evaluating the quality of trajectory visualizations. A higher NMI value signifies a closer similarity to the full dataset. As shown in Fig. 12, the results demonstrate

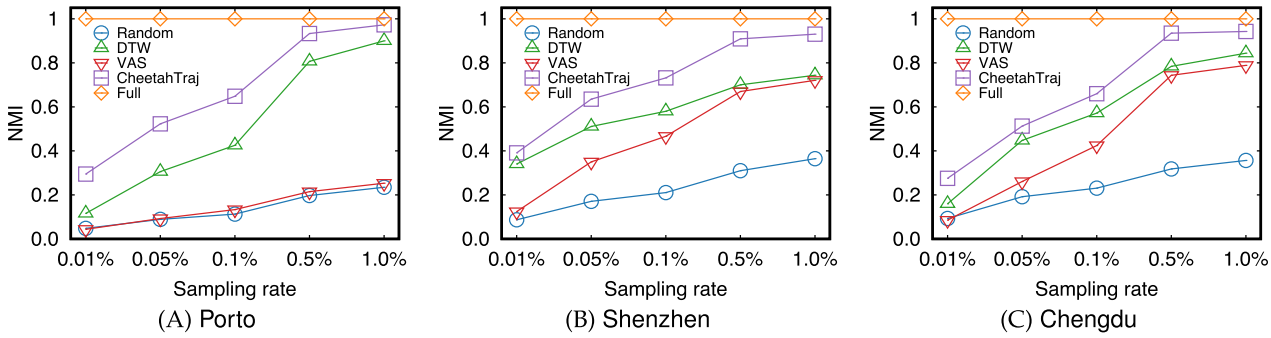


Fig. 12. Effect of sampling rate on NMI with Full.

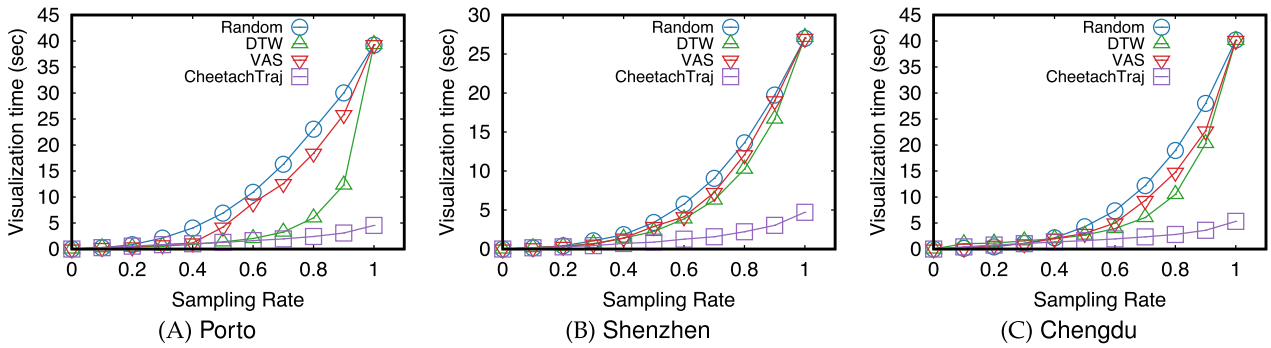


Fig. 13. Effect of visualization quality on visualization time.

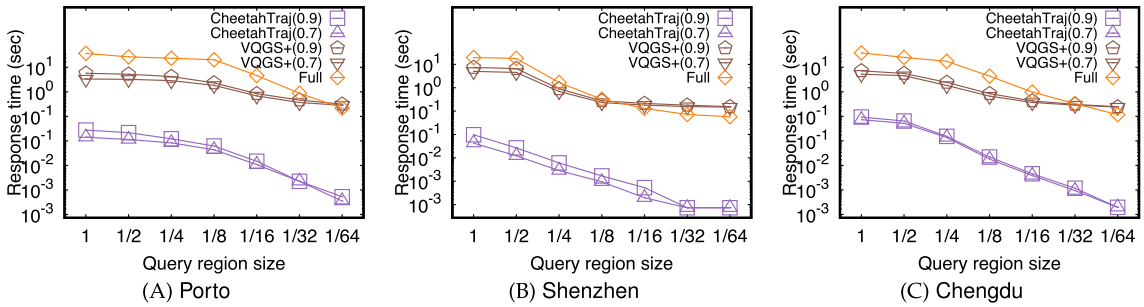


Fig. 14. Effect of region size on end-to-end response time.

that our method, CheetahTraj, outperforms the other techniques across all sampling rates in all three datasets.

In Fig. 13, we report the visualization cost (i.e., the wall clock time to generate visualization result using the sampled trajectories) for the methods under different quality requirements. We still consider the entire region in this experiment and the results show that CheetahTraj has significantly shorter visualization time than Random, DTW and VAS under the same quality requirement. This is because CheetahTraj is designed to sample trajectories that optimize visual quality and thus requires a smaller number of trajectories to achieve the same quality.

Efficiency of CheetahTraj*:* We evaluate the *response time* of our CheetahTraj framework under different quality guarantees and region sizes in Fig. 14. The response time of CheetahTraj is the end-to-end time for generating visualization for a selected region, which includes querying the CheetahTraj index and computing the visualization. For comparison, we also plot the response time of VQGS$^+$ (with $\delta = 8$), which uses on-line sampling instead of querying the index in CheetahTraj framework. We constrain the regions to be rectangles with a constant height/width ratio and measure the size of a region by the ratio of its height over the height of the entire region. For 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 in Fig. 14 show that CheetahTraj achieves a short response time (less than 1 s in all cases and 0.2 seconds for most cases) for different region sizes and quality guarantees. VQGS$^+$ is 1 to 2 orders of magnitude slower than CheetahTraj

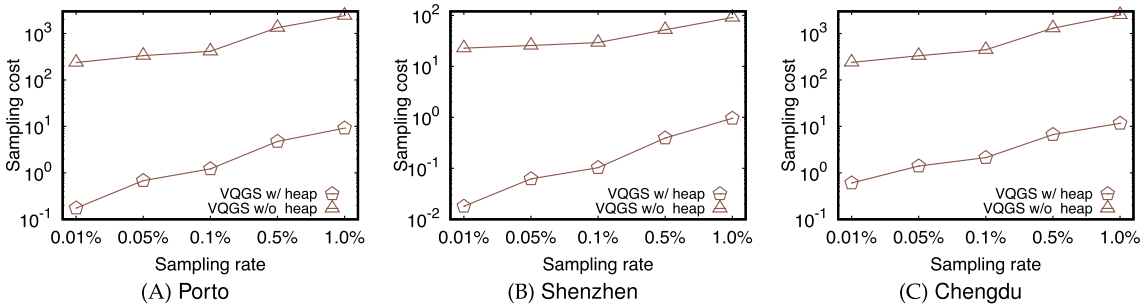


Fig. 15. Effect of the heap-based optimization on the running time of VQGS$^+$.

TABLE III
THE COSTS OF THE InvQuad-TREE INDEX

Dataset (size)	Height	Building time	Memory size
Porto (1.44G)	3	74s	0.88GB
	8	205s	1.96GB
	13	428s	3.65GB
Shenzhen (1.02G)	3	119s	0.76GB
	8	236s	1.68GB
	13	547s	3.12GB
Chengdu (1.49G)	3	98s	0.90GB
	8	227s	2.01GB
	13	454s	3.71GB

and takes at most 14.802 seconds in all cases. These results show that VQGS$^+$ cannot support interactive visual exploration and the InvQuad-tree index in CheetahTraj is effective in improving efficiency. Fig. 14 also shows that the response time of CheetahTraj can be shorter than Full by more than 3 orders of magnitude. The response time of all methods decreases when the region size shrinks as there are fewer trajectories in a smaller region. However, the response time (for both VQGS$^+$ and CheetahTraj) are not significantly longer when the quality threshold increases from 0.7 to 0.9 as quality improves quickly with the number of sampled trajectories as shown in Fig. 11.

Optimizations and costs: In Fig. 15, we report the running time of VQGS$^+$ with and without the heap-based lazy computation. This experiment is conducted on the entire spanned by each dataset and VQGS$^+$ uses $\delta = 4$. The results show that the heap-based optimization reduces the running time of VQGS$^+$ by around 2 orders of magnitude. We also report the building time and memory cost of the InvQuad-tree index in Table III. For all three datasets, it takes less than 10 minutes to build the InvQuad index with a height of 13, with which each leaf node is sufficiently small (covering 1.49×10^{-8} of the entire area). The memory costs of the InvQuad index in the last column of Table III are also comparable with the sizes of the raw dataset in the first column.

InvQuad-*tree index cost evaluation:* We report the building time and memory cost of the InvQuad-tree index in Table III. For all three datasets, it takes less than 10 minutes to build the InvQuad index with a height of 13. The memory cost of the InvQuad index in the last column is also comparable with the size of the raw data shown in the first column.

VII. RELATED WORK

As we focus on improving visualization latency and reducing visual clutter for trajectory data in this paper, we discuss related work in the two directions.

A. Improve Visualization Efficiency

Some approaches utilize *advanced computing facilities* for high visualization efficiency. For example, the massive parallel computation capabilities of GPUs are leveraged for fast graphics rendering [23], [39] and data compression [32]. Instead of GPU, ATLAS [15] exploits a powerful multi-core CPU server to accelerate visual analysis tasks. Piringer et al. [48] propose an architecture for interactive visual exploration, which utilizes multi-core devices and avoids the common pitfalls of multi-threading for quick visual feedback. Big data systems such as Hadoop and Spark are adopted to generate high-resolution visualizations in an offline fashion [14], [25], [70]. Data compression techniques can be also used to improve visualization efficiency. For instance, *trajectory simplification* extracts a subset of the locations in a trajectory to approximate the original trajectory, and thus reduces the data scale and the workloads of downstream tasks such as storage, analysis and visualization [13], [59], [61], [72]. These works are orthogonal to our CheetahTraj, and CheetahTraj can also benefit from advanced computing facilities (e.g., GPUs and multi-core CPUs) and trajectory simplification techniques.

Sampling has been used to improve visualization efficiency by reducing the data volume [11], [21], [30], [36], [47], [54], [62] but existing methods all target scatter plot, which shows individual locations (e.g., the distribution of restaurants in a city). For instance, BlinkDB [7] maintains a set of stratified location samples and proposes a strategy to sample locations according to user requirements such as visualization accuracy and maximum response time. Kwon et al. [38] discuss the relation between visualization efficiency and information loss when using sampling techniques for scatter plot. Some methods sample a set of locations that satisfy special characteristics, e.g., blue noise property [52] and multi-class property [16]. Yu et al. propose a framework [69] that allows users to sample locations that minimize a user-defined loss function for various visualization tasks. Wang et al. [63] progressively generate random samples that satisfy a given condition in an online fashion.

Rahman et al. [51] jointly utilize online sampling and incremental visualization to support interactive exploration. VAS [46] reduces the number of points in a scatter plot while preserving the spatial distribution of the points in the original dataset. These techniques cannot be directly applied to trajectory visualization as a trajectory contains a sequence of locations and thus is more complex than individual locations in scatter plot. As a result, sampling individual locations in the trajectories will lose the orientation information, which is crucial for tasks such as route identification and traffic flow analysis. To our knowledge, CheetahTraj is the first framework that samples trajectories with guarantee on visualization quality.

B. Reduce Visual Clutter

It is well-known that trajectory visualization suffers from severe visual clutter in dense areas due to a large number of trajectories, and many techniques are proposed to solve this problem. *Spatial aggregation techniques* divide the entire area into units (e.g., uniform grids [66] and Voronoi cells [6]) and visualize trajectories based on these units. For example, glyphs are used to show aggregate features such as destination distribution, moving directions and displacements [6], [8] in each unit. Movements among different units are visualized as flows connecting the units [8] or nested grids [66]. Instead of conducting aggregation by areas, *bundling techniques* merge (segments of) trajectories sharing the same pattern into bundles and thus reduce the number of trajectories. According to their way of creating bundles, these techniques can be classified into image-based [34], [71], force-based [31], [56], geometry-based [20].

While large datasets often contribute to visual clutter in data visualization, sampling techniques can intuitively mitigate this issue by reducing data size [22]. Sampling is commonly applied in point-based visualizations or particle systems, where the quantification of distance or distribution is more straightforward due to the nature of the data objects (spatial points). For example, Dix et al. [22] argued that random sampling can make large dataset visualization more computationally efficient and perceptually effective. Stochastic sampling has been used to enhance performance and diminish visual clutter in visual debugging of smoothed particle hydrodynamics (SPH) simulations [53]. The Void-and-Cluster approach was proposed to reduce visual clutter while preserving the blue noise property of spatial points [52]. Beyond point-based visualizations, sampling techniques have also been employed in graph visualizations [49], [50], [67] and parallel coordinates [26], [73] to curb visual clutter and maintain desirable patterns. However, the application of sampling techniques to trajectory visualization remains relatively unexplored. Direct adoption of existing methods is challenging due to the complexity and unique features of trajectory datasets. The work by [52] can be applied to simulation trajectory datasets, which differs from our focus as simulation trajectories can be time-aligned, and at each timestamp, location positions can be analyzed as a scatter plot. In contrast, our application does not consider time as a crucial feature of the dataset. This distinct focus necessitates the development of a novel approach, which we present in the form of CheetahTraj.

VIII. Conclusion and Future Work

Conclusion: This paper presents the CheetahTraj framework, which achieves high visual quality and low visualization latency for large-scale trajectory datasets by sampling the trajectories. CheetahTraj provides guaranteed visual quality in trajectory sampling by formulating a quality optimal sampling problem and developing efficient solutions including VQGS and VQGS$^+$. Low visualization latency is achieved with the InvQuad-tree index, which combines quad-tree with an inverted list, and allows using the sampling results computed offline for online visualization. Experiment results show that CheetahTraj consistently provides high quality visualizations in different cases and its visualization delay is orders of magnitude shorter than full visualization. Beyond trajectory visualization, the concept of visual quality and the computational framework can be applied to other line-based visualizations like parallel coordinates and line charts, as well as other spatial visualization forms such as scatter plots.

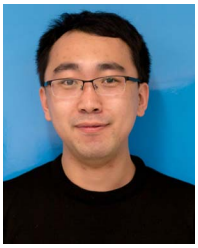
Future work: There are three promising directions. (1) Enhancing sampling performance: Trajectory simplification and modification techniques [12], [72] could be considered to further enhance the efficiency of visualization. This aims to reduce the number of locations within each trajectory, thereby decreasing the data size and rendering load. Moreover, additional features of trajectories, such as direction and length, could also be explored to enhance the overall performance. (2) Another promising direction involves employing trajectory representation learning to generate embedding vectors, facilitating the measurement of visual similarity among trajectories. This approach will enable us to sample the trajectory dataset with greater efficiency. (3) Supporting different scenarios: The framework of CheetahTraj can be improved to support the insertion and deletion of trajectories, as well as the streaming processing of trajectories. This will broaden the applicability of our approach, making it suitable for a wider range of real-world scenarios.

References

[1] Shenzhen dataset, 2021. Accessed: Sep. 17, 2021. [Online]. Available: http://jtys.sz.gov.cn/
[2] Fitts' law: Tracking users' clicks, 2020. [Online]. Available: https://www.interaction-design.org/literature/article/fitts-law-tracking-users-clicks
[3] The open-source graphical library, 2020. [Online]. Available: https://processing.org
[4] Source code, 2020. [Online]. Available: https://github.com/ChrisZcu/CheetahTraj
[5] Unfolding maps, 2020. [Online]. Available: http://unfoldingmaps.org/
[6] N. Adrienko and G. Adrienko, "Spatial generalization and aggregation of massive movement data," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 2, pp. 205–219, Feb. 2011.
[7] 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 *Proc. 8th ACM Eur. Conf. Comput. Syst.*, 2013, pp. 29–42.
[8] G. Andrienko and N. Andrienko, "Spatio-temporal aggregation for visual analysis of movements," in *Proc. IEEE Symp. Vis. Analytics Sci. Technol.*, 2008, pp. 51–58.
[9] 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," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 8, pp. 2232–2249, Aug. 2017.
[10] N. Andrienko, G. Andrienko, J. M. C. Garcia, and D. Scarlatti, "Analysis of flight variability: A systematic approach," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 54–64, Jan. 2019.

[11] L. Battle, M. Stonebraker, and R. Chang, "Dynamic reduction of query result sets for interactive visualizaton," in *Proc. IEEE Int. Conf. Big Data*, 2013, pp. 1–8.
[12] L. Baur, "Over-the-web retrieval and visualization of massive trajectory sets," Master's thesis, M.S. thesis, Inst. Formal Methods Comput. Sci., Univ. Stuttgartm, Stuttgart, Germany, 2021. [Online]. Available: https://elib.uni-stuttgart.de/handle/11682/11966
[13] O. Borcan, "Improving visualization of trajectories by dataset reduction and line simplification," Master's thesis, Utrecht Univ., Utrecht, The Netherlands, 2012. [Online]. Available: https://studenttheses.uu.nl/handle/20.500.12932/11638
[14] M. Budiu, P. Gopalan, L. Suresh, U. Wieder, H. Kruiger, and M. K. Aguilera, "Hillview: A trillion-cell spreadsheet for Big Data," 2019, *arXiv: 1907.04827*.
[15] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan, "Maintaining interactivity while exploring massive time series," in *Proc. IEEE Symp. Vis. Analytics Sci. Technol.*, 2008, pp. 59–66.
[16] H. Chen et al., "Visual abstraction and exploration of multi-class scatterplots," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 1683–1692, Dec. 2014.
[17] W. Chen, F. Guo, and F.-Y. Wang, "A survey of traffic data visualization," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 2970–2984, Dec. 2015.
[18] X. Chen, J. Zhang, C.-W. Fu, J.-D. Fekete, and Y. Wang, "Pyramid-based scatterplots sampling for progressive and streaming data visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 28, no. 1, pp. 593–603, Jan. 2022.
[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.
[20] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li, "Geometry-based edge clustering for graph visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 6, pp. 1277–1284, Nov./Dec. 2008.
[21] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang, "Sample seek: Approximating aggregates with distribution precision guarantee," in *Proc. Int. Conf. Manage. Data*, 2016, pp. 679–694.
[22] A. Dix and G. Ellis, "By chance enhancing interaction with large data sets through statistical sampling," in *Proc. Work. Conf. Adv. Vis. Interfaces*, 2002, pp. 167–176.
[23] T. Eaglin, I. Cho, and W. Ribarsky, "Space-time kernel density estimation for real-time interactive visual analytics," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, pp. 1381–1390.
[24] T. Eiter and H. Mannila, "Computing discrete fréchet distance," CD-Laboratory Expert Syst., TU Vienna, Austria, Tech. Rep. CD-TR 94/64, 1994. [Online]. Available: http://www.kr.tuwien.ac.at/staff/eiter/et-archive/cdtr9464.pdf
[25] A. Eldawy, M. F. Mokbel, and C. Jonathan, "HadoopViz: A MapReduce framework for extensible visualization of big spatial data," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 601–612.
[26] G. Ellis and A. Dix, "Enabling automatic clutter reduction in parallel coordinate plots," *IEEE Trans. Vis. Comput. Graphics*, vol. 12, no. 5, pp. 717–724, Sep./Oct. 2006.
[27] Z. Feng, H. Li, W. Zeng, S.-H. Yang, and H. Qu, "Topology density map for urban data visualization and analysis," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 2, pp. 828–838, Feb. 2021.
[28] S. Fujishige, *Submodular Functions and Optimization*. New York, NY, USA: Elsevier, 2005.
[29] Y. Gao, D. Wang, Y. Liao, and Y. Zou, "Relationship between urban tourism traffic and tourism land use," *J. Transport Land Use*, vol. 14, no. 1, pp. 761–776, 2021.
[30] T. Guo, K. Feng, G. Cong, and Z. Bao, "Efficient selection of geospatial data on maps for interactive and visualized exploration," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 567–582.
[31] D. Holten and J. J. Van Wijk, "Force-directed edge bundling for graph visualization," *Comput. Graph. Forum*, vol. 28, pp. 983–990, 2009.
[32] Y. Huang, Y. Li, Z. Zhang, and R. W. Liu, "GPU-accelerated compression and visualization of large-scale vessel trajectories in maritime IoT industries," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10794–10812, Nov. 2020.
[33] Z. Huang et al., "A natural-language-based visual query approach of uncertain human trajectories," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 1256–1266, Jan. 2020.
[34] C. Hurter, O. Ersoy, and A. Telea, "Graph Bundling by Kernel Density Estimation," *Comput. Graph. Forum*, vol. 31, pp. 865–874, 2012.
[35] F. Kamw et al., "Urban structure accessibility modeling and visualization for joint spatiotemporal constraints," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 104–116, Jan. 2020.
[36] A. Kim, E. Blais, A. G. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld, "Rapid sampling for visualizations with ordering guarantees," in *Proc. VLDB Endowment*, vol. 8, no. 5, pp. 521–532, 2015.
[37] R. Krüger, D. Thom, M. Wörner, H. Bosch, and T. Ertl, "TrajectoryLenses–A set-based filtering and exploration technique for long-term trajectory data," *Comput. Graph. Forum*, vol. 32, pp. 451–460, 2013.
[38] B. C. Kwon, J. Verma, P. J. Haas, and C. Demiralp, "Sampling for scalable visual analytics," *IEEE Comput. Graph. Appl.*, vol. 37, no. 1, pp. 100–108, Jan./Feb. 2017.
[39] O. D. Lampe and H. Hauser, "Interactive visualization of streaming data with kernel density estimation," in *Proc. IEEE Pacific Visual. Symp.*, 2011, pp. 171–178.
[40] P. Laube, *Computational Movement Analysis*. Berlin, Germany: Springer, 2014.
[41] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2007, pp. 420–429.
[42] X. Liang, S. Sintos, Z. Shang, and S. Krishnan, "Combining aggregation and sampling (nearly) optimally for approximate query processing," in *Proc. Int. Conf. Manage. Data*, 2021, pp. 1129–1141.
[43] D. Liu et al., "SmartADP: Visual analytics of large-scale taxi trajectories for selecting billboard locations," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 1–10, Jan. 2017.
[44] H. Liu, Y. Gao, L. Lu, S. Liu, H. Qu, and L. M. Ni, "Visual analysis of route diversity," in *Proc. IEEE Conf. Vis. Analytics Sci. Technol.*, 2011, pp. 171–180.
[45] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Trans. Med. Imag.*, vol. 16, no. 2, pp. 187–198, Apr. 1997.
[46] Y. Park, M. Cafarella, and B. Mozafari, "Visualization-aware sampling for very large databases," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 755–766.
[47] J. Peng, D. Zhang, J. Wang, and J. Pei, "AQP connecting approximate query processing with aggregate precomputation for interactive analytics," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 1477–1492.
[48] H. Piringer, C. Tominski, P. Muigg, and W. Berger, "A multi-threading architecture to support interactive visual exploration," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 6, pp. 1113–1120, Nov./Dec. 2009.
[49] J. R. Ponciano, C. D. Linhares, L. E. Rocha, E. R. Faria, and B. A. Travençolo, "Combining clutter reduction methods for temporal network visualization," in *Proc. 37th ACM/SIGAPP Symp. Appl. Comput.*, 2022, pp. 1748–1755.
[50] D. Rafiei, "Effectively visualizing large networks through sampling," in *Proc. IEEE Vis.*, 2005, pp. 375–382.
[51] S. Rahman et al., "I've seen "enough," incrementally improving visualizations to support rapid decision making," in *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1262–1273, 2017.
[52] T. Rapp, C. Peters, and C. Dachsbacher, "Void-and-cluster sampling of large scattered data and trajectories," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 780–789, Jan. 2020.
[53] S. Reinhardt, M. Huber, O. Dumitrescu, M. Krone, B. Eberhardt, and D. Weiskopf, "Visual debugging of SPH simulations," in *Proc. 21st Int. Conf. Inf. Vis.*, 2017, pp. 117–126.
[54] E. A. Rundensteiner et al., "Xmdvtool Q: Quality-aware interactive data exploration," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2007, pp. 1109–1112.
[55] Y. Sadahiro, R. Lay, and T. Kobayashi, "Trajectories of moving objects on a network: Detection of similarities, visualization of relations, and classification of trajectories," *Trans. GIS*, vol. 17, no. 1, pp. 18–40, 2013.
[56] D. Selassie, B. Heller, and J. Heer, "Divided edge bundling for directional network data," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 12, pp. 2354–2363, Dec. 2011.
[57] B. Shneiderman, "Response time and display rate in human performance with computers," *ACM Comput. Surv.*, vol. 16, no. 3, pp. 265–285, 1984.
[58] E. Starr and B. Goldfarb, "Binned scatterplots: A simple tool to make research easier and better," *Strategic Manage. J.*, vol. 41, no. 12, pp. 2261–2274, 2020.
[59] M. van Kreveld, M. Löffler, and L. Wiratma, "On optimal polyline simplification using the hausdorff and Fréchet distance," Mar. 2018, *arXiv: 1803.03550*.
[60] 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," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 1, pp. 11–20, Jan. 2016.

[61] K. Vrotsou et al., "SimpliFly: A methodology for simplification and thematic enhancement of trajectories," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 1, pp. 107–121, Jan. 2015.
[62] G. Wang et al., "STULL: Unbiased online sampling for visual exploration of large spatiotemporal data," in *Proc. IEEE Conf. Vis. Analytics Sci. Technol.*, 2020, pp. 72–83.
[63] L. Wang, R. Christensen, F. Li, and K. Yi, "Spatial online sampling and aggregation," in *Proc. VLDB Endowment*, vol. 9, no. 3, pp. 84–95, 2015.
[64] D. Weng, R. Chen, Z. Deng, F. Wu, J. Chen, and Y. Wu, "SRVis: Towards better spatial integration in ranking visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 459–469, Jan. 2019.
[65] D. Weng et al., "Towards better bus networks: A visual analytics approach," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 2, pp. 817–827, Feb. 2021.
[66] J. Wood, J. Dykes, and A. Slingsby, "Visualisation of origins, destinations and flows with OD maps," *Cartographic J.*, vol. 47, no. 2, pp. 117–129, 2010.
[67] Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu, and W. Cui, "Evaluation of graph sampling: A visualization perspective," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 401–410, Jan. 2017.
[68] C. Yang et al., "EpiMob: Interactive visual analytics of citywide human mobility restrictions for epidemic control," *IEEE Trans. Vis. Comput. Graphics*, vol. 29, no. 8, pp. 3586–3601, Aug. 2023.
[69] J. Yu and M. Sarwat, "Turbocharging geospatial visualization dashboards via a materialized sampling cube approach," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 1165–1176.
[70] J. Yu, Z. Zhang, and M. Sarwat, "GeoSparkViz: A scalable geospatial data visualization framework in the apache spark ecosystem," in *Proc. 30th Int. Conf. Sci. Statist. Database Manage.*, 2018, pp. 1–12.
[71] W. Zeng, Q. Shen, Y. Jiang, and A. Telea, "Route-aware edge bundling for visualizing origin-destination trails in urban traffic," *Comput. Graph. Forum*, vol. 38, pp. 581–593, 2019.
[72] D. Zhang, M. Ding, D. Yang, Y. Liu, J. Fan, and H. T. Shen, "Trajectory simplification: An experimental study and quality analysis," in *Proc. VLDB Endowment*, vol. 11, no. 9, pp. 934–946, 2018.
[73] Z. Zhou et al., "Context-aware visual abstraction of crowded parallel coordinates," *Neurocomputing*, vol. 459, pp. 23–34, 2021.



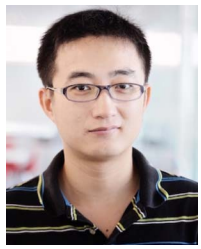
Qiaomu Shen received the PhD degree in computer science from The Hong Kong University of Science and Technology, in 2020. He is currently an research Assistant Professor with the Southern University of Science and Technology (SUSTech). Before joining SUSTech, he worked as a senior developer with Noah's Ark Lab for Huawei Technologies. His current research interests include spatial-temporal visualization, urban computing, and visual analytics of complex system.

Chaozu Zhang is currently working toward the master's degree with the Department of Computer Science and Engineering, Southern University of Science and Technology. His research interests include spatiotemporal data visualization and Big Data systems visualization.

Xiao Yan is currently a research assistant professor with the Department of Computer Science and Engineering, Southern University of Science and Technology. His research interest includes systems and algorithms for database, and large-scale machine learning systems.

Chuan Yang received the BE degree in computer science from the Southern University of Science and Technology. He is currently a software development engineer with Alibaba group.

Dan Zeng received the BE and PhD degrees in computer science and technology from Sichuan University, in 2013 and 2018. She is currently a research assistant professor with the Department of Computer Science and Engineering, Southern University of Science and Technology. She was a post-doc research fellow with the Data Management and Biometrics Group, University of Twente, The Netherlands. Her main research topics is machine learning for bimetrics and super-resolution.



Wei Zeng received the PhD degree in computer science from Nanyang Technological University, and worked as a senior research with Future Cities Laboratory, ETH Zurich, and an associate researcher with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He is an Assistant Professor with the Hong Kong University of Science and Technology, Guangzhou campus. His recent research interests include visualization and visual analytics, computer graphics, AR/VR, and HCI.

Bo Tang received the PhD degree in computer science from The Hong Kong Polytechnic University, in 2017. He is currently an assistant professor with the Southern University of Science and Technology. He was an visiting researcher with Centrum Wiskunde & Informatica and Microsoft Research Asia, respectively. His research interests include database system and Big Data processing techniques.