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

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

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

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

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

I. Introduction

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

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

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

Introduce the data to visualization time: database/file - memory - data transformation(GPS to screen position)- visualization.

Sampling techniques are widely used for large-scale data analysis in both database and visualization communities to reduce the visual clutter and [12]–[15]. By sampling a subset of records from the raw large-scale dataset, it helps to reduce the data to visualization time. One such example is ScalaR [16], which employs a reduction layer between the visualization layer and the data management layer. The reduction layer samples records *uniformly at random* (denoted as RAND) when the query results are too large. There are three challenges to design appropriate sampling methods for interactive visualizations: C1) guarantee the visual quality , C2) can be generated efficiently to support the interactive exploration, C3) support different-level of details. However, RAND does not work well for large-scale trajectory data visualization as it cannot provide quality guarantee. Take Figure 1(B) and (C) for example, they are the visualization results generated by RAND on the Porto taxi trajectory dataset with sampling rate 0.1% and 1%, respectively. Obviously, both of them are very different from the visualization of the full Porto dataset in Figure 1(A), since the random sampling method is easily ignore the data patterns in the spare areas. Another basic idea of the data reduction is to iteratively remove the trajectories which have the least impact of the whole visualization [17]. Thus impact of a trajectory can be evaluated by sum of the distance(DTW or Fréchet distance) to all other trajectories. However the pairwise distance calculation for large trajectory dataset can be very time consuming.

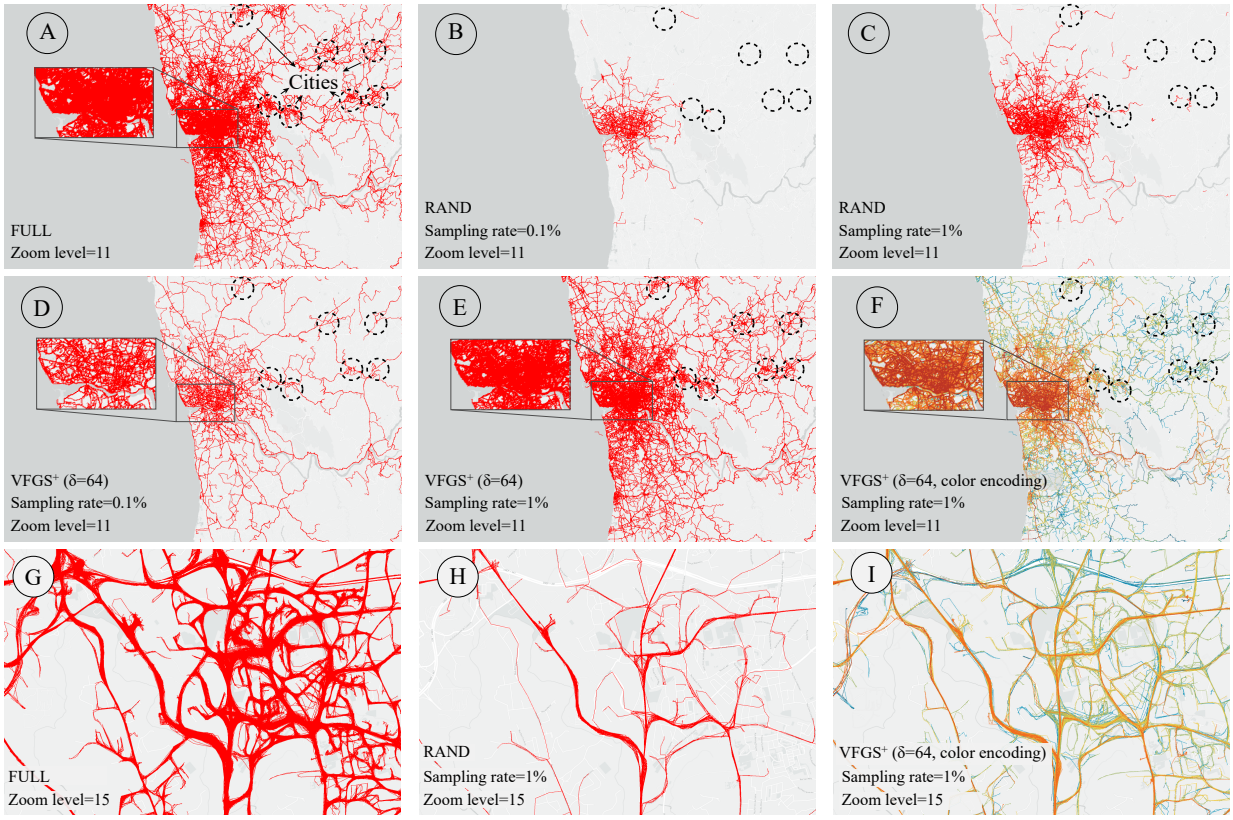


Fig. 1. A comparison of visualization results. (I) A is the visualization of the full Porto taxi trajectory dataset at zoom level 11. At the same zoom level, B and C are produced by uniform random sampling, while D, E, F are produced by our CheetahTraj algorithm. (II) G is the visualization of the full Porto dataset at zoom level 15, while H and I are the corresponding results of uniform random sampling and CheetahTraj, respectively. (III) F and I are generated by CheetahTraj with color encoding for representativeness to combat visual clutter. Best viewed in color.

In this work, we set out to design efficient sampling algorithms that provides visual quality guarantee for large-scale line-based trajectory visualization. This goal leads to four research problems: (i) *how to measure the visual fidelity of one visualization result?* (ii) *how to devise an efficient sampling algorithm that provides guaranteed visual fidelity from multi-level of details?* (iii) *how to tackle the visual clutter problem in large trajectory visualization?* To address these problems, we first propose a novel pixel-based *visual fidelity loss function* to formally measure the difference between two visualizations. We then show that it is NP-hard to select a sized-k sample of the trajectories to minimize the visual fidelity loss function. Next, we devise an *efficient approximate algorithm* named VFGS, which provides theoretical visual fidelity guarantee for the sampled results.Quadtree Part? Last, we *explicitly tackle the visual clutter problem* by taking data distribution and human perception characteristics into consideration in an advance algorithm named CheetahTraj.

We illustrates the merits of our proposals in Figure 1. Figure 1(D) and (E) are the results of our CheetahTraj algorithm on the Porto dataset with sampling rate 0.1% and 1%, respectively. Compared with their uniform random sampling (i.e., RAND) counterparts Figure 1(B) and (C), they are obviously more similar to the full dataset visualization in Figure 1(A). The advantage of our proposals over RAND is also consistent across different zoom levels, e.g., Figure 1(E) vs. Figure 1(C) at level 11, and Figure 1(I) vs. Figure 1(H) at level 15. Figure 1(F) is produced by our CheetahTraj algorithm using the same parameters as Figure 1(E) but the trajectories are colored according to their algorithm-generated representativeness (warmer color means more representative). Observe that the visual clutter problem in Figures 1(A) and (E) are significantly alleviated in Figure 1(F) with color encoding. This advantage is even more prominent when comparing Figure (I) with Figure (G) and (H).

To sum up, our contributions in this paper include:

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

applications.

II. Related work

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

A. Trajectory Visualization Methods

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

B. Interactive Visualization for Large Datasets

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

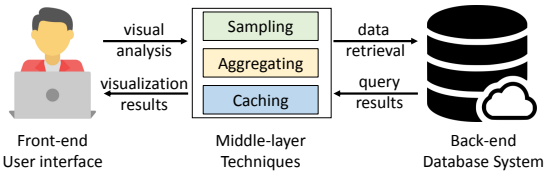


Fig. 2. System architecture for interactive visualization.

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

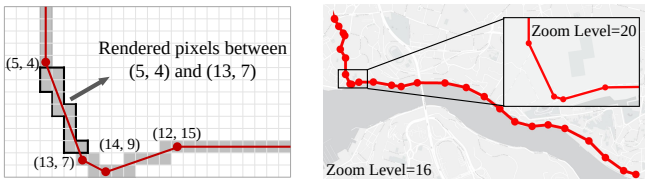
Sampling-based techniques: Sampling techniques are widely used in both visualization and database communities [12]–[16], [40]. The work most relevant to ours is [15], which is designed for scatter plots (a form of point-based visualization). It solves the problem of point overdrawing and preserves the spatial point distribution of the original dataset at the same time. The techniques in [15] cannot be adapted to our trajectory visualization problem as trajectory is more complex than scatter points (e.g., varying lengths, contiguous GPS points).

Caching-based and other techniques: Chan et al. present ATLAS [34], which utilizes caching for efficient data communication between server and client. In addition, ATLAS also exploits a powerful multi-core server to accelerate visual analysis task processing from the middle-layer to the back-end. Piringer et al. [41] propose a multi-threading architecture for interactive visual exploration, which utilizes multi-core devices and avoids the pitfalls of multi-threading to provide quick visual feedback. Our techniques in this work are orthogonal to researches in this category.

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

III. Problem Formulation

In this part, we first formally define the *fidelity-optimal sampling problem* in Section III-A and then show that it is



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

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

NP-hard to solve the problem exactly in Section III-B.

A. Problem Definition

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

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

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

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

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

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

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

One subtlety is that trajectory visualization needs to work at different *zoom levels* upon user request. For example, Google map [42] provides a zoom levels from 0 to 20, with level 0 providing the largest visualization range (i.e., the whole world) but the lowest resolution, and level 20 providing the smallest visualization range (e.g., individual building, if available) but the highest resolution. We also provided an illustration of zoom level in Figure 3(B). Ideally, we want a sample to be *zoom-level-independent*, providing a consistent fidelity guarantee at different zoom levels. This turns out to be straightforward as trajectory visualization merges several pixels in a high-level result (by pixel-wise OR) to obtain a pixel in a lower-level visualization result. The following theorem shows that it suffices to satisfy the fidelity guarantee at the highest zoom level.

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

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

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

□

We are aware that lower zoom levels need more coarse-grained visualization and thus the sampling rate at the highest level may be larger than necessary. We left more sophisticated designs, such as re-sampling a sample set or generating multiple sample sets for future work, and consider only sampling for the highest zoom level. As our fidelity loss function considers the entire visible region (i.e., it is a *global fidelity measure*), the visual difference between our sample and the ground-truth visualization may be large for some specific regions, especially when the marked points are sparse in these regions. *Local visual fidelity* can be important for some tasks(e.g., outlier discovery [43], [44]) and we leave it for future work. We want to note that a fidelity-guaranteed sample $\mathcal{R}$ is also *query independent* from the definition of the fidelity loss function, which means $\mathcal{R}$ only needs to be constructed once to answer all queries.

B. Hardness Analysis

We use $t_i \in \mathcal{T}$ to denote a trajectory in the dataset. According to the working mechanism of line-based trajectory

visualization, t_i corresponds to a set of marked pixels on the canvas in the ground-truth visualization and we also use t_i to denote this set of pixels. Thus, we have $V(\mathcal{T}) = \cup_{t_i \in \mathcal{T}} t_i$ and $V(\mathcal{R}) = \cup_{t_i \in \mathcal{R}} t_i$. We can transform Problem 1 as follows

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

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

IV. Our Solution: VFGS

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

A. Visual Fidelity-Guaranteed Sampling

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

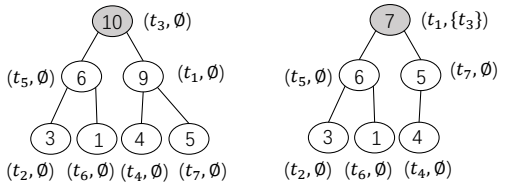
Algorithm 1 VFGS$(\mathcal{T}, k = \alpha|\mathcal{T}|)$

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

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

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

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



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

Fig. 4. Lazy computing manner.

Lemma 1 (Submodularity)**.** *Define the contribution of a trajectory t to the result set $\mathcal{R}$ as $\Delta(\mathcal{R}, t) = |V(\mathcal{R} \cup t)| - |V(\mathcal{R})|$. Given a trajectory t and two result sets $\mathcal{R}, \mathcal{R}^{'}$, if $\mathcal{R} \subset \mathcal{R}^{'}$, then $\Delta(\mathcal{R}, t) \geq \Delta(\mathcal{R}^{'}, t)$.*

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

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

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

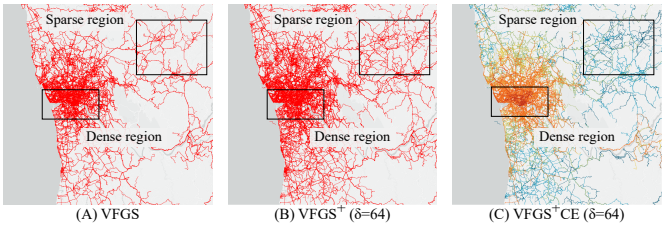
Proof. In each iteration, Algorithm 1 computes the uncovered pixels of each trajectory in dataset $\mathcal{T}$ with $O(m)$ cost. As the dataset $\mathcal{T}$ has $O(|\mathcal{T}|)$ trajectories and Algorithm 1 runs for $k = \alpha|\mathcal{T}|$ iterations, the total cost is $O(k \cdot m \cdot |\mathcal{T}|)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

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

B. Heap-based Lazy Computation

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

To implement this idea, we maintain a max-heap for the number of uncovered pixels of each trajectory and update the



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

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

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

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

V. Advanced Approach: CheetahTraj

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

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

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

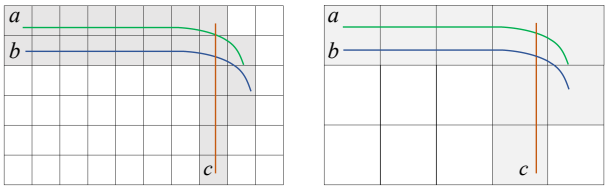
Based on the two observations above, we can improve VFGS by delivering richer information in the sparse regions and reducing visual clutter in the dense regions. CheetahTraj in Algorithm 2 achieves both objectives using a perception tolerance parameter δ, which models the perception capability of humans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

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

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

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

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



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

Fig. 6. CheetahTraj at different zoom levels.

details in the sparse regions compared with the VFGS result in Figure 5(A), and we can identify the main roads in the dense region with very warm colors.

VI. Experimental Evaluation

We evaluate our techniques on two real-world trajectory datasets: Porto, Shenzhen and Chengdu. Porto [45] contains a total of 2.39 million taxi trajectories and 75.67 million of GPS points, and the longest trajectory has 3,490 GPS points. Shenzhen [11] consists of 3.07 million taxi trajectories with 53.53 million GPS points, and the longest trajectory has 2,268 GPS points. Chengdu [46] consists of 0.28 million taxi trajectories with 6.69 million GPS points, and the longest trajectory has 4025 GPS points. All experiments are conducted on a machine with an Intel i7-8700 CPU, 24 GB memory and an NVIDIA GeForce GTX1080 GPU with 8 GB on-chip memory, running on Windows 10. All methods are implemented in Java 1.8, and the Processing 3 library [47] is used for rendering. All datasets and source codes required to reproduce our results are available at [48].

This section is organized as follows. In Section VI-A, we present several case studies of the visualization results on the Porto datasets to demonstrate the merits of our methods. In Section VI-B, we conduct a comprehensive user study to test the effectiveness of our visualization results on practical tasks including *region center identification*, *reachable route inspection* and *traffic flow comparison*. In Section VI-C, we quantitatively evaluate the visual quality and efficiency of our methods on all these three datasets.

A. Case studies

1) Effectiveness of overview visualization: We conduct case study in Porto, shown as the visualization results in Figure 7, to demonstrate the effectiveness our proposals from the following aspects.

Consistently good visual fidelity at overview: At zoom level 11, Figure 7(A) is the visualization result of the full Porto dataset. With a sampling rate $\alpha = 1\%$, Figures 7(B) and (C) are the visualizations produced by uniform random sampling (RAND), baseline [17], and our visual quality-guaranteed sampling method (CheetahTraj, Algorithm 2), respectively. Comparing with Figure 7(B) and (C), it is obvious that Figure 7(E) is more similar to Figure 7(A). In particular, Figure 7(E) not only preserves the overall visual structure of the entire region but also keeps the details of cities that are far from the center (marked by the dashed cycles in the figure). However, the details of these cities are lost in Figure 7(C) as RAND mostly preserve trajectories in the dense region. On the other sides, the distance based sampling keeps the trajectories in the sparse regions but cannot guarantee visual quality.

Consistently good visual fidelity under different sampling rates: Figures 7(D) and (E) are the visualizations produced by CheetahTraj with a sampling rate of 0.1% and 1%. We can make two observations: (i) the larger the sampling rate, the better the visual fidelity, i.e., Figures 7(C) and (E) are more similar to Figure 7(A) compared with Figures 7(B) and (D); (ii) the visualization of CheetahTraj with a sampling rate of 0.1% (i.e., Figure 7(D)) looks even more appealing than the visualization of RAND with a sampling rate of 1% (i.e., Figure 7(C)) as Figure 1(D) better captures the overall visual structure of Figure 7(A).

Color encoding effectively mitigates visual clutter: At a zoom level of 11 and with a sampling rate of 1%, Figures 7(E) and (F) are the visualizations produced our CheetahTraj and CheetahTrajCE (i.e., CheetahTraj with color encoding), respectively. Visual clutter is severe for the full dataset (i.e., Figure 7(A)) and Figures 7(E) since almost all pixels in the center regions are colored, and it is difficult to identify patterns in the these regions. The visualization of CheetahTrajCE in Figure 7(F) migrate this problem by encoding the trajectories with color, and thus it is easy to identify some prominent trajectories and busy routes.

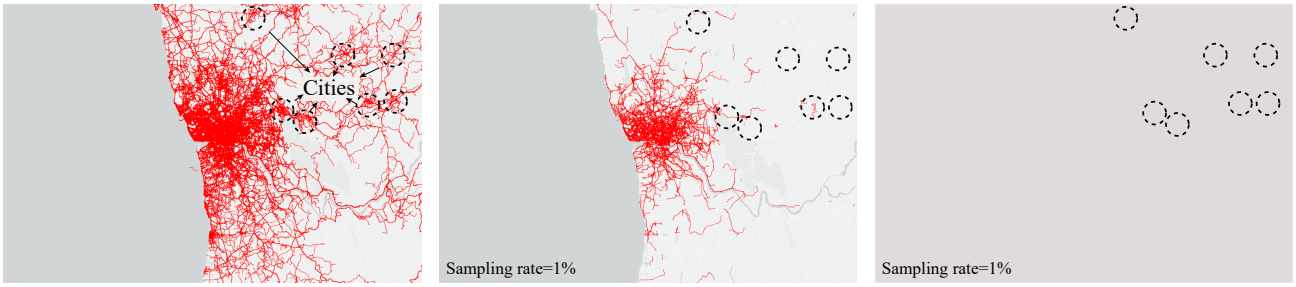
2) Case Studies on the effectiveness of detail view: We next present the effectiveness of our proposals with detail views by investigating two regions of interest in Porto, the dense region as shown in Fig. 8(8A).

Dense region: R1 is the center of Porto, which has the highest concentration of the trajectories and causes serious visual clutter, as visualized in Fig. 8(B4). For example, the circular structures of the main route(shown as Fig. **??**(B5$**$)) is unclear. local + fix alleviates the visual clutter by preserve the 1% trajectories of the total regions but the clutter is still serious. Furthermore, CheetahTraj performs better than local + fix by preserving less trajectories and reduce the visual clutter.

Sparse region: R2 indicates the city of Casino Espinho at zoom level $**$, which contains less trajectories than the center of Porto as the visualization result of full dataset shown in Fig. 8(B1). Given fix sampling rate $\alpha = 1\%$, Fig. 8(B2) indicates the visualization of local + fix. This visualization result misses a lot if detail information in this region, because the fix sampling rate preserves too few trajectories(*** trajectories) in the spares region. While CheetahTraj in Fig. 8(B3) performs much better than local + fix as the sampling rate is automatically adjusted by the visual quality. In this visualization, the trajectory sketch of Casino Espinho is almost the same as it in Fig. 8(B1), the visualized result of full dataset.

B. User Study for Practical Tasks

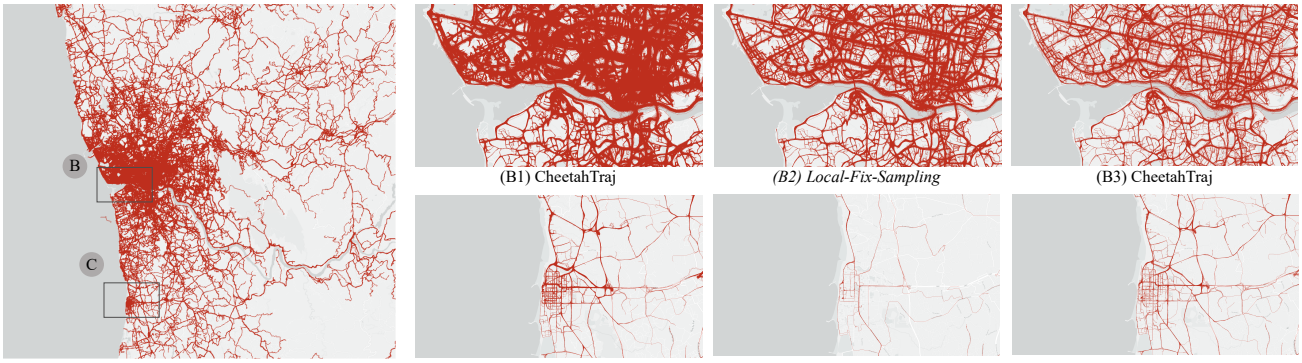
In this part, we evaluate the effectiveness of our proposals by recruiting 186 participants to conduct practical spatial tasks. These spatial tasks (illustrated in Figure 9) are designed by our



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

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

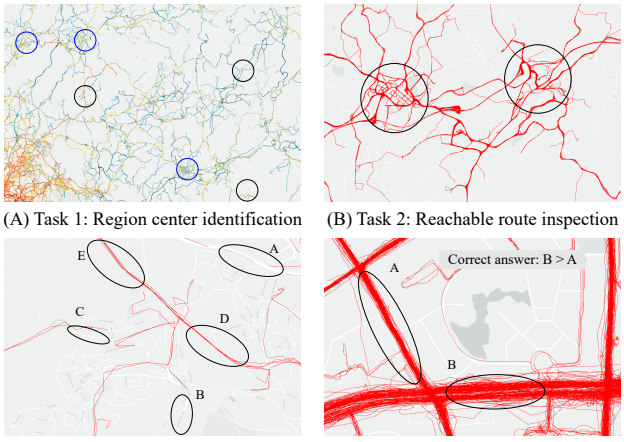
Fig. 7. Effectiveness studies of CheetahTraj at overview visualization in Porto.



(A) FULL(Overview) (B1) CheetahTraj *(B2) Local-Fix-Sampling* (B3) CheetahTraj

(C1) FULL *(C2) Local-Fix-Sampling* (C3) CheetahTraj

Fig. 8. Effectiveness studies of CheetahTraj at detail visualization.



(A) Task 1: Region center identification (B) Task 2: Reachable route inspection

(C) Task 3: Traffic flow comparison (D) Task 3 tutorial

Fig. 9. The three tasks in the user study.

industry partner, Tencent Map [49], according to their user study. The results show that our algorithms provide informative visualizations, which lead to good user performance in these tasks.

Settings: We recruited 186 participants with 24 females, 162 males, aged 18-29 with a mean of 21.16 for the user study[1]. The user study system is a web-based platform, in which all visualizations are displayed with a resolution of 450*300. We used the two taxi trajectory datasets, i.e., Porto and Shenzhen, to generate the visualizations.

We test five visualization methods: (i) exact visualization using the full dataset, denoted as FULL; (ii) uniform random sampling, denoted as RAND; (iii) our VFGS algorithm; (iv) our CheetahTraj algorithm; (v) CheetahTraj with color encoding, denoted as CheetahTrajCE. The sampling rate is $\alpha = 0.5\%$ for RAND and our algorithms, and $\delta = 64$ for CheetahTraj and CheetahTrajCE. The tasks are described as follows:

(T1): region center identification. The downtown areas or

[1]We are aware of the gender and age bias in the participants as we mainly recruited students from a university due to some practical constraints. However, we think the test results are still informative as we observed that the performance of a participant in the tasks is not significantly affected by these factors.

commercial regions of a city are hubs for human activities and crucial for traffic management. We also observed that the taxi trajectories are denser in these regions than the other regions for both datasets. T1 contains 30 visualizations of six selected regions, which include city or commercial centers from the Porto dataset. For each visualization, we manually label three locations as region centers, and randomly generate three locations far away from the correct centers as fake centers. Each participant is asked to check six randomly selected visualizations and choose three region centers from each visualization, as illustrated in Figure 9(A).

(T2): reachable route inspection. Visualized trajectories should show reachable routes that connect different regions and allow users to identify them. In this task, a participant is presented with a visualization containing two circular areas (as shown in Figure 9(B)) and asked to draw the representative reachable routes between the two areas. We assume that a route is more representative if there are a larger number of trajectories containing it. For each visualization, we manually generate at least 5 reachable routes and a user answer is considered correct if it contains 3 of these routes. T2 includes 35 visualizations of seven different regions, and for each visualization, two cities/commercial districts are marked by circles. Each participant is asked to check five randomly selected visualizations.

(T3): traffic flow comparison. A road with a large traffic flow (i.e., having many trajectories passing it) should have dense and broad trajectory brunch in the visualization. For the CheetahTraj algorithm with color encoding, such large track flow can also be identified by the concentration of trajectories with darker colors. In this task, we ask the participants to choose the road with larger traffic flow from two roads according to the visualization results, as shown in Figure 9(C). They can also choose "I am not sure" if they could not decide the answer. T3 includes the visualizations of five randomly selected regions, 25 visualization results, and 60 comparison road pairs. We count the exact number of passing trajectories for each road to deduce the ground-truth answer.

User study procedure: When the participants enter the user study system, they are first given a brief introduction about the motivation of the study and the tasks. For each tasks, we include a tutorial (with the correct result) to help the participants to get familiar with the interface and tasks. For example, Figure 9(D) shows a tutorial for T3, where road B has larger traffic flow than road A and the correct answer is displayed on upper right corner. For each participant and task, we randomly choose visualizations and question instances from our task base. The participants are interviewed to collect feedbacks after finishing the test and their answers are saved for result analysis. We are mainly interested in how different visualization methods affect the accuracy in processing the three tasks. The readers can refer to our supplementary video for more details about our user study tasks and procedures.

Result analysis: Figure 10 reports the average accuracy of the five visualization methods in the three tasks.

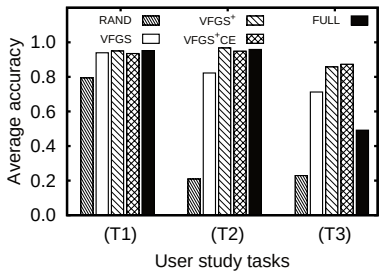


Fig. 10. Average accuracy of the three tasks. X-axis shows different tasks while Y-axis indicates the accuracy.

For (T1) region center identification, the accuracies of our proposals are very similar to that of visualizing the full dataset (i.e., FULL). In contrast, the accuracy of RAND is significantly lower than FULL. These results suggest that our proposals successfully preserve the centers of human activities even with a low sampling rate of 0.5%. CheetahTrajCE performs slightly worse than CheetahTraj and FULL, and some participants reported that the colors of the trajectories distract them in the post-interview.

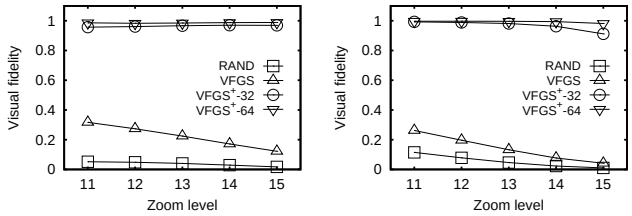
For (T2) reachable route inspection, RAND has a very low accuracy compared with the other 4 visualization methods. This is because RAND lost many fine-grained details of the trajectories due to uniform sampling, especially in the sparse regions. However, these details can be crucial for determining the existences of a route. Moreover, our advanced methods CheetahTraj and CheetahTrajCE provide noticeably better performance than VFGS. This is because CheetahTraj and CheetahTrajCE take data distribution and human perception intro consideration, and sample more trajectories in the sparse regions. As a result, more details are preserved.

(T3) traffic flow comparison is more difficult than T1 and T2, and thus the accuracy drops for all visualization methods. Similar to the case of T1 and T2, RAND has the worst performance among all methods. Remarkably, the accuracies of our proposals (i.e., VFGS, CheetahTraj and CheetahTrajCE) are even higher than FULL. In the post-interview, the participants reported that FULL has severe visual clutter problem, which makes it difficult to compare the traffic flows on two roads. Therefore, the higher accuracies of our proposals indicate that sampling effectively mitigates visual clutter. In addition, the accuracies of CheetahTraj and CheetahTrajCE are higher than that of VFGS, suggesting that our advanced method and color encoding are effective in alleviating visual clutter.

To sum up, the results in this part shows that our proposals (i.e., (VFGS, CheetahTraj, and CheetahTrajCE) are effective in practical spatial tasks. All our methods consistently outperforms RAND by a large margin across different tasks, the advanced methods (i.e., CheetahTraj, and CheetahTrajCE) often achieve comparable or even better performance than FULL.

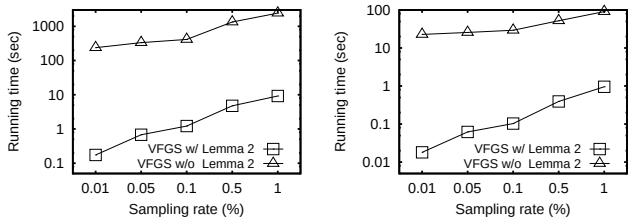
C. Quantitative Evaluation

In this part, we quantitatively evaluate our proposals on the Porto and Shenzhen trajectory datasets from two aspects: (i)



(A) Porto (B) Shenzhen

Fig. 11. Visual fidelity vs. zoom levels.



(A) Porto (B) Shenzhen

Fig. 12. Running time of VFGS w/ and w/o Lemma 1.

visual fidelity at different zoom levels and (ii) running time under different sampling rates.

Visual fidelity evaluation: We report the *visual fidelity* of different visualization methods in Figure 11. Visual fidelity is defined as the $1 - loss$, in which $loss$ is the fidelity loss function in Equation (1). The sampling rate is $\alpha = 0.5\%$ and the visualization using the full dataset is used as the ground-truth. The results show that RAND always has the lowest visual fidelity. VFGS has better visual fidelity than RAND but is significantly outperformed by CheetahTraj. With $\delta = 32$ and $\delta = 64$, the minimum visual fidelity values of CheetahTraj are 0.95 and 0.91 for Porto and Shenzhen, respectively. Moreover, the fidelity of all methods increases when the zoom level drops, which in line with Theorem 1.

Running time evaluation: We first conduct an experiment to evaluate the rendering cost by datasize. We vary the number of trajectories from 1000 to 1 million, which are randomly selected from Porto dataset. The experimental results are summarized in Table I. We observe that the rendering cost is linear with the input data trajectories. Then, we report the running time of our VFGS algorithm in Figure 12 by varying the sampling rate from 0.01% to 1%. The results show that VFGS is quite slow without the submodularity of contribution value, which agrees with Lemma 1 in Section IV-B. The optimized VFGS (e.g., VFGS with Lemma 1) outperforms VFGS by one to three orders of magnitudes on both datasets. The result show that running time of our VFGS algorithm is below 1 second in most cases. We have shown that VFGS provides good visualization performance with a low sampling rate (e.g., 0.1% or 1%) in Section 6.1 and 6.2, and Table I suggests that the rendering latency scales almost linearly with dataset cardinality. By significantly reducing the dataset cardinality with sampling, VFGS can effectively reduces the rendering latency to make interactive visualization possible without sacrificing visual fidelity. For example, rendering the full Porto dataset takes about 34 seconds, with a sampling rate of 1%, VFGS can bring down the rendering latency to less than 1 second.

TABLE I
VISUALIZATION RENDERING COST

No. of trajectories	No. of GPS points	Rendering time (s)
1,000	32,648	0.016
10,000	331,583	0.143
100,000	3,262,278	1.416
1,000,000	32,660,845	13.950

VII. CONCLUSIONS AND FUTURE DIRECTIONS

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

[13] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang, "Sample + seek: Approximating aggregates with distribution precision guarantee," in *SIGMOD*. ACM, 2016, pp. 679–694.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[40] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K.-L. Ma, "Visual abstraction and exploration of multi-class scatterplots," *TVCG*, vol. 20, no. 12, pp. 1683–1692, 2014.

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

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

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

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

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

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

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

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

[49] "Tencent map service," https://map.qq.com/, 2020.