Visual Fidelity-Guaranteed Sampling for Large-scale Trajectory Data Visualization

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 fidelity, i.e., generating visualizations that look different from the exact one. In this work, we propose sampling techniques that provide *theoretically guaranteed visual fidelity* for large-scale trajectory data visualization. We first define a natural pixel-based *fidelity 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 named VFGS with a suite of optimization techniques, which runs efficiently but still provides guaranteed fidelity loss. By taking data distribution and human perception characteristics into consideration, we further improve VFGS to an advanced algorithm named VFGS$^+$ 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 [**?**], [**?**], [**?**] and location-based services [**?**], [**?**]. 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 [**?**], [**?**], [**?**]. However, large-scale line-based trajectory visualization suffers from two problems– (i) *scalability* and (ii) *visual clutter*, which we elaborate as follows.

Scalability: Trajectory datasets can be extremely large and thus take a long time to render for visualization. For example, Shenzhen has 24,237 taxis which collectively generate more than 82.8 million GPS locations each day [**?**]. Our benchmark on the Porto taxi trajectory dataset shows that it takes 13.95 seconds to render 1 million real-world trajectories using an NVIDIA GeForce GTX 1080 GPU with 8GB video memory. The long delay caused by large dataset cardinality makes interactive visual exploration difficult. Thus, we want visualization to scale to very large datasets while maintaining a short delay.

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

Sampling techniques are widely used for large-scale data analysis in both database and visualization communities [**?**], [**?**], [**?**], [**?**]. By sampling a subset of records from the raw large-scale dataset, it helps to reduce the rendering latency on graphics devices for visualization. One such example is ScalaR [**?**], 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. However, RAND does not work well for large-scale trajectory data visualization as it cannot provide fidelity 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).

In this work, we set out to design efficient sampling algorithms that provides visual fidelity guarantee for line-based large-scale trajectory visualization. This goal leads to three 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?* (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. Last, we *explicitly tackle the visual clutter problem* by taking data distribution and human perception

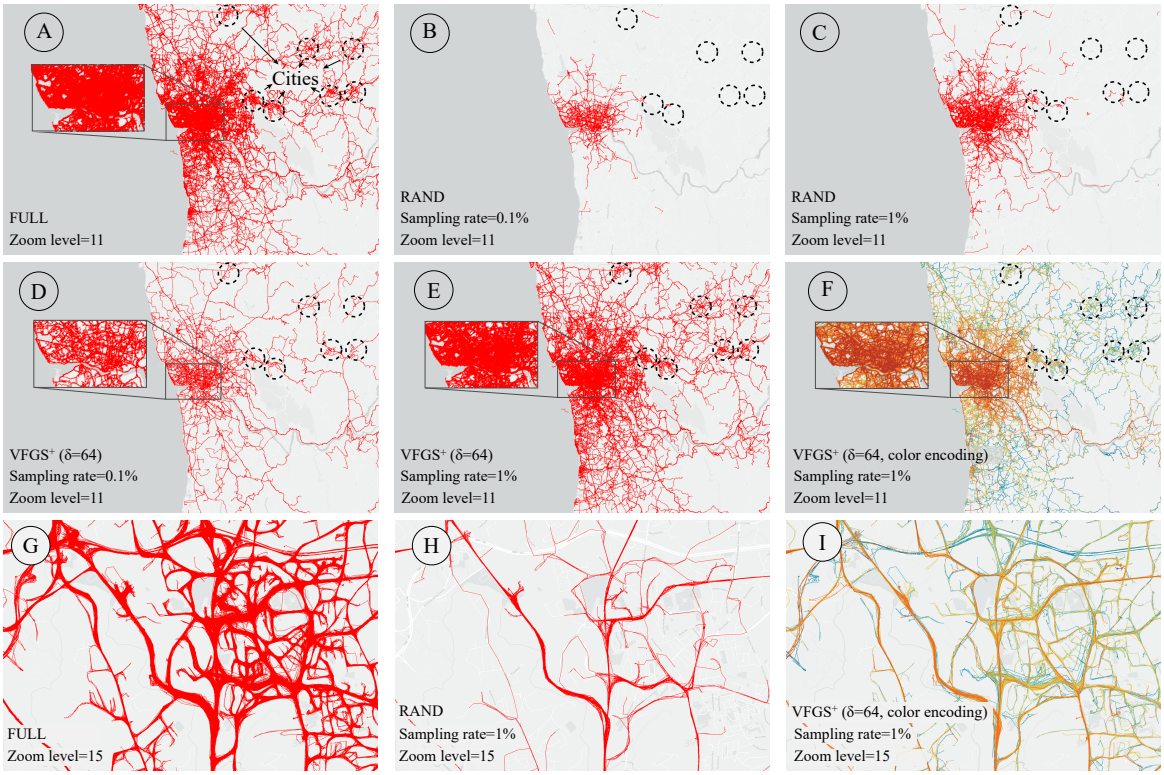


Fig. 1. A comparison of visualization results. (I) A is the visualization of the full Porto taxi trajectory dataset at zoom level 11. At the same zoom level, B and C are produced by uniform random sampling, while D, E, F are produced by our VFGS$^+$ algorithm. (II) G is the visualization of the full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

characteristics into consideration in an advance algorithm named VFGS$^+$.

We illustrates the merits of our proposals in Figure 1. Figure 1(D) and (E) are the results of our VFGS$^+$ 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 VFGS$^+$ 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 VFGS$^+$ 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 [**?**], 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 [**?**] for more detailed discussions. Point-based visualization plots the locations in each trajectory independently and captures the overall spatial distribution of

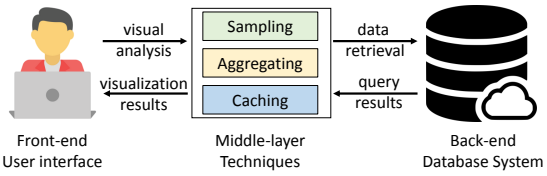
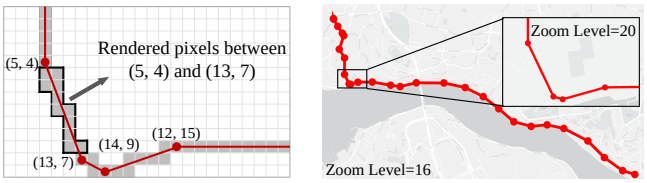


Fig. 2. System architecture for interactive visualization.



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

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

the moving objects. Many density-based methods, e.g., kernel density estimation, are applied in point-based visualization [?], [?], [?], [?], [?] to preserve the spatial distribution. Region-based visualization slices the entire region into sub-regions and visualizes the aggregated information in each sub-region [?], [?], [?]. 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 poly-lines 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 [?], [?], 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 [?], [?], [?] and advanced interaction techniques [?], [?].

B. Interactive Visualization for Large Datasets

Figure 2 illustrates the general architecture of interactive visualization systems, e.g., Spotfire [?], Tableau [?], ATLAS [?], and Viate [?]. 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 [?] 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 [?] 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.

Aggregating-based techniques: These works [?], [?], [?] 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 [?], [?]. 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 [?], [?], [?], [?], [?], [?]. The work most relevant to ours is [?], 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 [?] 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 [?], 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. [?] 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 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 [**?**] 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 $l \leq l'$ [1], with larger l indicating higher resolution.

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 [**?**], [**?**]) 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

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

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 section, we focus on addressing 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 several optimizations to improve the efficiency of VFGS in Section IV-B.

A. Visual Fidelity-Guaranteed Sampling

Due to the hardness of Problem 1, the straight-forward solution is uniform random sampling RAND. This solution randomly selects k trajectories from the dataset $\mathcal{T}$ and stores them in the result set $\mathcal{R}$. The selected trajectories in $\mathcal{R}$

[1] We omitted the proofs of all theorems and lemmas due to space limit, and refer the interested readers to our technical report [**?**].

are rendered as the visualization result. Obviously, uniform random sampling RAND does not provide any guarantee on the visual fidelity of the sampled set.

We next present our visual fidelity-guaranteed sampling method VFGS in Algorithm 1, which employs a greedy paradigm. In particular, it finds the trajectory tmp in $\mathcal{T}$ that maximizes $|\mathcal{R} \cup tmp|$ 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: $\quad tmp \leftarrow argmax_{t_i \in \mathcal{T}} |\mathcal{R} \cup t_i|$
4: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{tmp\}$
5: 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 $f(\mathcal{R}) \geq 0.632 * f(\mathcal{R}^\star)$.

B. Optimization Techniques

With the above theoretical analysis, Algorithm 1 offers a visual fidelity-guaranteed sampling algorithm for the large-scale trajectory data visualization problem. However, as the time complexity analyzed in Lemma 1, it is not scalable to large-scale trajectory datasets (e.g., millions of trajectories).

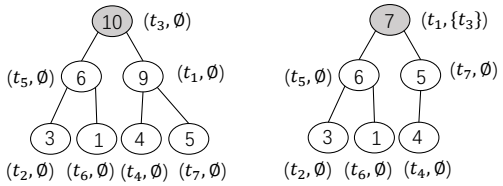
Lemma 1 (Time Complexity): Given trajectory dataset $\mathcal{T}$ and an integer $k = \alpha|\mathcal{T}|$, the time complexity of Algorithm 1 is $O(\alpha \cdot m \cdot |\mathcal{T}|^2)$, where m is the maximum length of all trajectories in dataset $\mathcal{T}$.

For example, the Porto trajectory dataset has 2.39 millions of taxi trajectories. The maximum length of the trajectories in it has 3,490 GPS points. It takes 413.6 seconds to return a subset $\mathcal{R}$ with sampling rate 0.1%. Obviously, it is impractical for interactive trajectory explorations.

Due to the inefficient of our visual fidelity-guaranteed sampling approach in Algorithm 1, we then devise performance optimization techniques to accelerate its running time. The core idea is utilizing the submodularity of the covered pixels of result set $\mathcal{R}$, as shown in Lemma 2.

Lemma 2 (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 $\Delta(\mathcal{R}, t) \geq \Delta(\mathcal{R}^{'}, t)$.

The submodularity in Lemma 2 reduces many unnecessary trajectory contribution value computations. We maintain a max-heap for the number of uncovered pixels of each trajectory and employ a lazy computing manner. That is, the contribution of a given trajectory is computed when necessary. Figure 4(a) shows a tiny max-heap example about the numbers of uncovered pixels of each trajectory 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 gray node at Figure 4(B)). Then t_1 is selected at the second iteration without computing the number of uncovered pixels in other trajectories, i.e., all white nodes at Figure 4(B). The reason is that the contribution of the trajectories in all white nodes is less than 7 according to the submodularity in Lemma 2.



(A) 1st iteration (B) 2nd iteration
Fig. 4. An illustration of lazy computing.

The performance of Algorithm 1 is improved significantly because its contribution values are computed only when necessary. To exemplify, Algorithm 1 costs 413.6 seconds to return the results with sampling rate 0.1% on the Porto taxi trajectory dataset. However, it only needs 1.2 seconds in our performance-optimized VFGS.

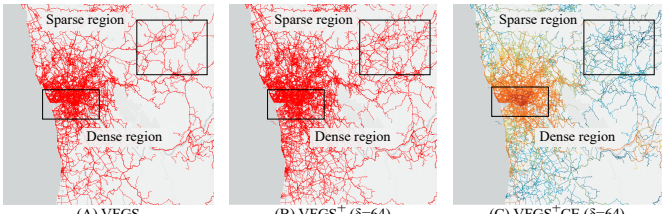
V. Advanced Approach: VFGS$^+$

Until now, VFGS in Algorithm 1 offers a visual fidelity-guaranteed sampling approach for the large-scale trajectory visualization problem (Problem 1), which returns the fidelity-guaranteed result efficiently via the optimization techniques in Section IV-B. 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 the advanced approach VFGS$^+$ to alleviate it by considering (i) trajectory data distribution, and (ii) human perception capability. We elaborate (i) and (ii) by the examples in Figure 5 shortly.

Trajectory data distribution: Considering the Porto trajectory dataset, Figure 5(A) is the visualization result of VFGS with sampling ratio 0.5%. Obviously, the real-world trajectory dataset is non-uniform distributed. For example, the trajectories in dense region are much more than those in the sparse region, as illustrated by the rectangles in Figure 5(A).

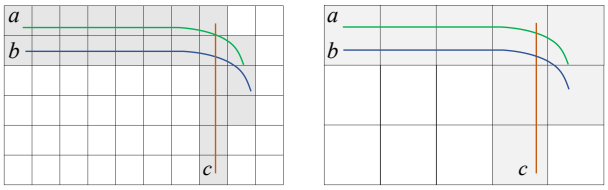
Human perception capability: Intuitively, humans can more easily distinguish the difference between sparse regions rather than dense regions in Figures 5(A) and (B). The core reason is the limited perception capability of human beings. In particular, the visual difference of human beings diminishes when the visualized trajectories are large enough with a given zoom level. For example, the difference between two dense regions in Figures 5(A) and (B) almost cannot be recognized by human beings.

Taking the above two observations into consideration, we can further improve the returning result of visual fidelity-guaranteed sampling approach VFGS by delivering rich information at sparse regions and reducing visual clutter in dense



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

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



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

Fig. 6. VFGS$^+$ at different zoom levels.

regions. In this section, we devise the advanced approach VFGS$^+$ (see Algorithm 2) to achieve the above two objectives. In specific, we introduce perception tolerance parameter δ in VFGS$^+$, which models the perception capability of humans at the highest level of details. In other words, suppose the 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 $(x + \delta, y + \delta)$, are not necessary to cover because they are in the perception tolerance of human beings.

Fortunately, we can slightly revise VFGS in Algorithm 1 to incorporate the perception tolerance parameter δ in advance approach VFGS$^+$, as shown in Algorithm 2. It 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 VFGS$^+(\mathcal{T}, k = \alpha|\mathcal{T}|, \delta)$

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

Interestingly, the visual clutter large trajectory visualization problem can be further reduced by encoding representative trajectories in $\mathcal{R}$ (the returning result of VFGS$^+$) with colors. In particular, VFGS$^+$ selects the trajectory with the largest uncovered pixels by taking the perception tolerance capability of humans into account at each iteration, instead of only choosing the trajectory with the largest uncovered pixels in VFGS (Algorithm 1). During its selection process, some trajectories will not be included into the result set $\mathcal{R}$ even they have more uncovered pixels w.r.t. $\mathcal{R}$. The reason is their uncovered pixels are too close to the pixels in the selected trajectories (i.e., within the tolerance area of selected pixels). With Figure 6(A) as an example, suppose $\delta = 1$ and trajectory a was selected at the first iteration, the selected trajectory in the second iteration is c instead of b because almost all pixels in b is in the tolerance area of a's.

Inherently, the VFGS$^+$ trajectory selection process embeds the representativeness of each trajectory in the result set $\mathcal{R}$. We define the representativeness of a trajectory as the number of influenced trajectories in the dataset $\mathcal{T}$. We compute the representativeness of each trajectory in $\mathcal{R}$ from Line 7 to Line 9 in Algorithm 2, then visualize them by encoding with different colors. Figure 5(C) shows the visualized result of VFGS$^+$ by encoding the trajectory representativeness with colors. Obviously, the trajectories in dense region are darker than those in sparse region, indicating there are more trajectories in dense region.

Notably, VFGS$^+$ provides excellent visual fidelity over VFGS at arbitrary zooming resolutions naturally. The key technique to achieve that is it considers the zooming resolutions inherently when introducing the perception tolerance δ. For example, the zoom level in Figure 6(A) is higher than that in Figure 6(B). As our above elaboration, VFGS$^+$ selects trajectory a and c at Figure 6(A). When it zoomed out, as shown in Figure 6(b), it still captures the main sketch of the underlying dataset (as gray cells shown). We will elaborate it by the experimental study in Section VI.

VI. Experimental Evaluation

We evaluate our techniques on two real-world trajectory datasets: Porto and Shenzhen. Porto [**?**] 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 [**?**] consists of 3.07 million taxi trajectories with 53.53 million GPS points, and the longest trajectory has 2,268 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 [**?**] is used for rendering. All datasets and source codes required to reproduce our results are available at [**?**].

This section is organized as follows. In Section VI-A, we present a case study of the visualization results on the Porto dataset 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 fidelity loss and efficiency of our methods.

A. Case Study of Visualization Results

We use the visualization results in Figure 1 to demonstrate the effectiveness our proposals from the following three as-

pects.

Consistently good visual fidelity at different zoom-levels: At zoom level 11, Figure 1(A) is the visualization result of the full Porto dataset. With a sampling rate $\alpha = 1\%$, Figures 1(C) and (E) are the visualizations produced by uniform random sampling (RAND) and our advanced visual fidelity-guaranteed sampling method (VFGS$^+$, Algorithm 2), respectively. Comparing with Figure 1(C), it is obvious that Figure 1(E) is more similar to Figure 1(A). In particular, Figure 1(E) not only preserves the overall visual structure of the entire region but also keeps the details of cities that are far from the center (marked by the dashed cycles in the figure). However, the details of these cities are lost in Figure 1(C) as RAND mostly samples trajectories in the dense region.

Figures 1(H) and (I) are the visualizations generated by RAND and VFGS$^+$ (with color encoding), respectively, at zoom level 15 with a sampling rate $\alpha = 1\%$. Compared with the visualization using the full dataset at this level (i.e., Figure 1(G)), Figure 1(H) only shows a few trajectories and most of the information in the raw data is lost. In contrast, Figure 1(I) captures the overall structure of Figure 1(G), and the details are even clearer than Figure 1(G) thanks to color encoding.

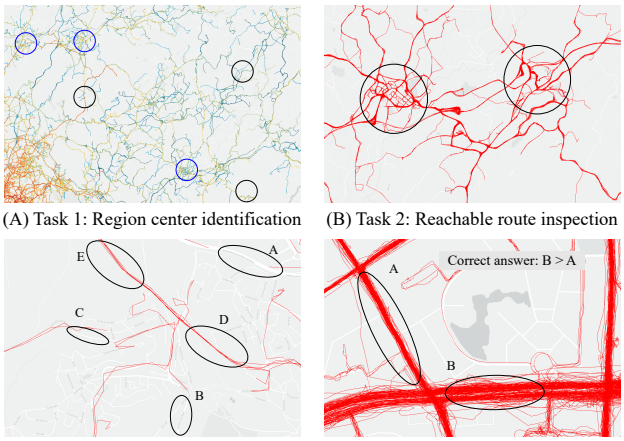
Consistently good visual fidelity under different sampling rates: Figures 1(B) and (C) are the visualizations produced by RAND with a sampling rate of 0.1% and 1%, respectively, while Figures 1(D) and (E) are the visualizations generated by our VFGS$^+$ algorithm at the same sampling rates. We can make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

Color encoding effectively mitigates visual clutter: At a zoom level of 11 and with a sampling rate of 1%, Figures 1(E) and (F) are the visualizations produced our VFGS$^+$ and VFGS$^+$CE (i.e., VFGS$^+$ with color encoding), respectively. Visual clutter is severe for the full dataset (i.e., Figure 1(A)) and Figures 1(E), and it is difficult to identify a specific trajectory in the dense region. The visualization of VFGS$^+$CE in Figure 1(F) reduces the visual clutter by encoding the trajectories with color, and thus it is easy to identify some prominent trajectories. The comparison between Figure 1(G) (full dataset) and Figure 1(I) (VFGS$^+$CE) at a sampling rate of 0.1% also validates the effectiveness of color encoding.

Due to the page limit, we refer the readers to our technical report [**?**] for more comprehensive visualization results on both datasets.

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.



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

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

Fig. 7. The three tasks in the user study.

These spatial tasks (illustrated in Figure 7) are designed by our industry partner, Tencent Map [**?**], 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[2]. 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 VFGS$^+$ algorithm; (v) VFGS$^+$ with color encoding, denoted as VFGS$^+$CE. The sampling rate is $\alpha = 0.5\%$ for RAND and our algorithms, and $\delta = 64$ for VFGS$^+$ and VFGS$^+$CE. The tasks are described as follows:

(T1): region center identification. The downtown areas or 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 7(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 7(B)) and asked to draw the representative

[2]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.

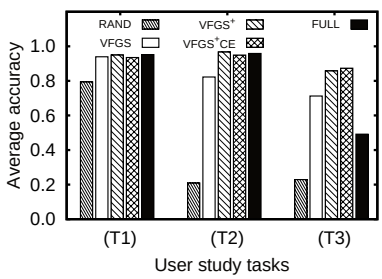


Fig. 8. Average accuracy of the three tasks. X-axis shows different tasks while Y-axis indicates the accuracy.

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 VFGS$^+$ 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 7(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 7(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 8 reports the average accuracy of the five visualization methods in the three tasks.

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%. VFGS$^+$CE performs slightly worse than VFGS$^+$ and FULL, and some participants reported that the colors of the trajectories distract them in the post-interview.

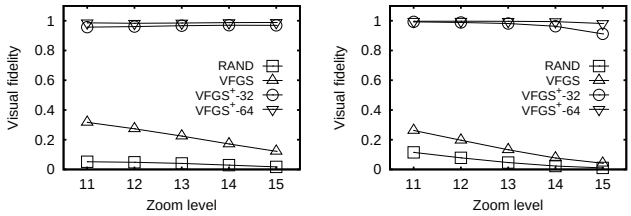


Fig. 9. Visual fidelity vs. zoom levels.

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 VFGS$^+$ and VFGS$^+$CE provide noticeably better performance than VFGS. This is because VFGS$^+$ and VFGS$^+$CE take data distribution and human perception intro consideration, and sample more trajectories in the sparse regions. As a result, more details are preserved.

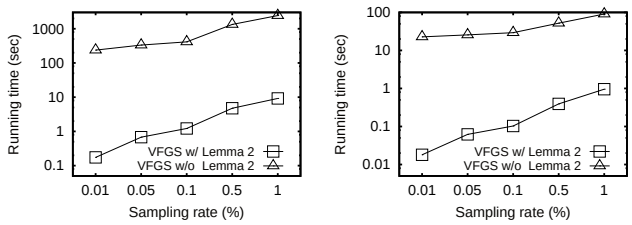
(T3) traffic flow comparison is more difficult than T1 and T2, and thus the accuracy drops for all visualization methods. Similar to the case of T1 and T2, RAND has the worst performance among all methods. Remarkably, the accuracies of our proposals (i.e., VFGS, VFGS$^+$ and VFGS$^+$CE) are even higher 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 VFGS$^+$ and VFGS$^+$CE 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, VFGS$^+$, and VFGS$^+$CE) are effective in practical spatial tasks. All our methods consistently outperforms RAND by a large margin across different tasks, the advanced methods (i.e., VFGS$^+$, and VFGS$^+$CE) often achieve comparable or even better performance than FULL.

C. Quantitative Evaluation

In this part, we quantitatively evaluates our proposals on the Porto and Shenzhen trajectory datasets from two aspects: (i) 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 9. 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



(A) Porto (B) Shenzhen

Fig. 10. Running time of VFGS w/ and w/o Lemma 2.

is significantly outperformed by VFGS$^+$. With $\delta = 32$ and $\delta = 64$, the minimum visual fidelity values of VFGS$^+$ 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 report the running time of our VFGS algorithm in Figure 10 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 2 in Section IV-B. The optimized VFGS (e.g., VFGS with Lemma 2) 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 our technical report 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.

VII. Conclusions and Future Directions

This paper presents a novel sampling technique, VFGS$^+$, 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.