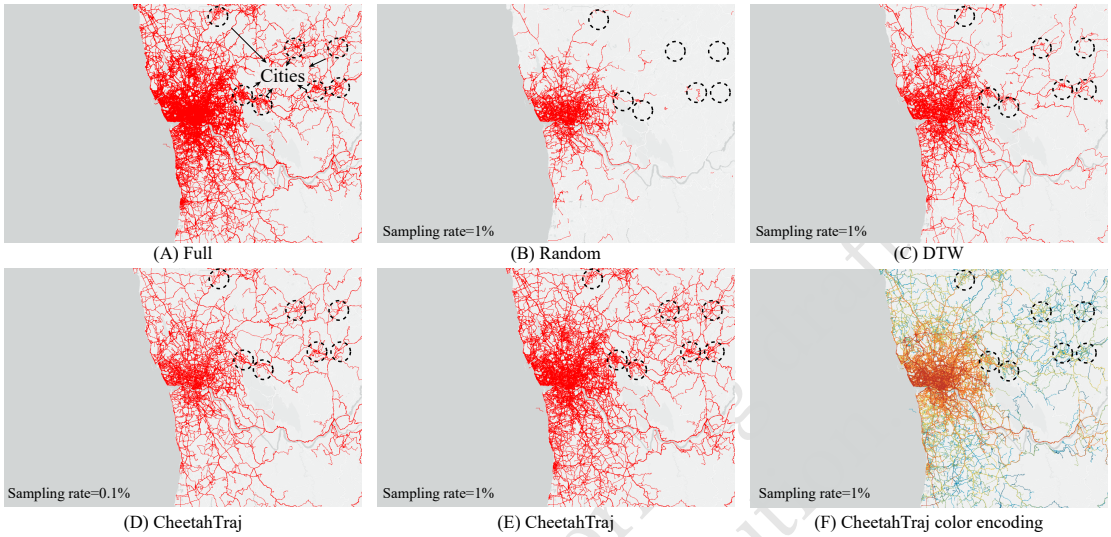
# CheetahTraj: **Quality and Efficiency in Large-scale Trajectory Data Visual Exploration**




(A) Full (B) Random (C) DTW

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

**Figure 1: Effectiveness of** CheetahTraj **for overview visualization on the Porto dataset.**


## ABSTRACT

Visualizing large-scale trajectory data is a core subroutine for many applications, e.g., traffic management, urban planning, and route recommendation. However, naively visualizing all trajectories for a target region could result in long delay due to large data volume. Ad-hoc sampling reduces visualization time but may harm visual quality, i.e., generating visualizations that look substantially different from the exact one. In this paper, we propose the CheetahTraj framework to provide high quality trajectory visualization with low latency. To this end, we first define a natural pixel-based *visual quality function* to measure the similarity between two visualizations and formulate a quality optimal trajectory sampling problem. Then, we design the VQGS and VQGS$^+$ algorithms to solve the trajectory sampling problem, which not only provide guaranteed visual quality but also reduce visual clutter. To generate quality guaranteed trajectory samples with high efficiency, we develop a quad-tree-based index (InvQuad) that allows to use trajectory samples computed offline. Extensive experiments (i.e., case-, user-, and quantitative- studies) are conducted on 3 real-world trajectory datasets and the results show that CheetahTraj consistently provide higher visualization quality and better efficiency than the baselines.




*Woodstock '18, June 03–05, 2018, Woodstock, NY*



ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456



## CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

## KEYWORDS

Trajectory visualization; Interactive data exploration; Sampling

**ACM Reference Format:**

. 2018. CheetahTraj: Quality and Efficiency in Large-scale Trajectory Data Visual Exploration. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456


## 1 INTRODUCTION

«««< HEAD

The ubiquity of location acquisition devices leads to an explosive growth 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 [34, 38, 46] and location-based services [27, 45]. 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 [8, 9, 16]. To support interactive exploration on trajectory data, it is crucial to conduct line-based trajectory visualization for an arbitrary region with high quality and low latency.

======= The ubiquity of location acquisition devices leads to an explosive growth 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 [34, 38, 42, 46] and location-based services [27, 45]. Among various visualization

**Table 1: Visualization time on the Porto dataset (in seconds)**

| No. of trajectories | No. of GPS points | Mapping time | Rendering time | Total time |
|---|---|---|---|---|
| 1,000 | 31,300 | 0.027 | 0.003 | 0.03 |
| 10,000 | 31,6531 | 0.169 | 0.005 | 0.174 |
| 100,000 | 316,7120 | 1.701 | 0.057 | 1.758 |
| 1,000,000 | 31,646,379 | 15.562 | 0.592 | 16.154 |

methods, line-based trajectory visualization, which connects the locations of a moving object by polylines, is widely adopted for spatial-temporal data analytics [8, 9, 16]. To support interactive exploration on trajectory data, it is crucial to conduct line-based trajectory visualization for an arbitrary region with high quality and low latency. »»»> 717904052d402b9c6da6451dd6510c80d2a1b21e

**Challenge for visualizing full datasets:** To visualize the trajectories in a target region $Q$, a natural solution is to find the trajectories in it and visualize all these trajectories (Full for short). However, Full may suffer from a long visualization time as the trajectory datasets can be extremely large. For example, Shenzhen has 24,237 taxis which collectively generate more than 7.72 million GPS locations each day [4]. Our profiling results in Table 1[1] also show that visualizing the 1,000,000 taxi trajectories needs 16.154 seconds which is far more than 2 seconds delay suggested for interactive exploration [33]. Another problem of applying Full on large-scale datasets is *visual clutter* [26], where there are too many points in the visualization such that it is difficult for users to gain insights. One such example can be found in Figure 1(A), for which it is difficult to recognize the road networks in the dense center region.

**Ad-hoc sampling has poor visual quality:** Sampling is widely used to accelerate large-scale data analysis in both information retrieval and data visualization [18, 24, 29, 31]. One such example is ScalaR [10], which samples records *uniformly at random* (denoted as Random) when the query results are too large. However, Random has poor visual quality as its visualization can be significantly different from the ground-truth in the sparse areas as shown in Figure 1(B). Another natural idea is to sample trajectories with good diversity and we develop such a baseline using the famous Dynamic Time Warping (DTW) distance between trajectories [11]. As shown in Figure 1(C), DTW provides better visualization than Random but there are still obvious differences between DTW and the ground-truth in Figure 1(A). Without explicit visual quality guarantee, sampling trajectories in ad-hoc ways may produce visualizations with poor quality and mislead visual exploration.

**The** CheetahTraj **framework:** We explore novel algorithm and efficient index jointly in the CheetahTraj framework to provide visualizations with high quality and low latency. To conduct quality guaranteed sampling, we first propose a novel pixel-based *visual quality function* to measure how similar an approximate visualization is to the ground-truth. We also show that it is NP-hard to select an optimal set of trajectories that maximize the visual quality function. Next, we devise a *visual quality guaranteed sampling algorithm* named VQGS, which provides theoretical visual quality guarantee for the sampled trajectories. Then, we *tackle the visual clutter problem* by taking data distribution and human perception into consideration in an advance algorithm named VQGS$^+$. To avoid running the somehow complex VQGS$^+$ algorithm on-line for interactive visual exploration, we design an InvQuad-tree index based on quad-tree, which allows to use the sampling results computed in an offline index building phase.

We conduct extensive case study, user study and quantitative performance evaluation to validate the visualization quality and efficiency of the CheetahTraj framework. The case study shows that CheetahTraj consistently provides high quality visualizations for both large target regions and small target regions. The user study with 35 participants confirms that CheetahTraj effectively reduces visual clutter and produces visualizations that are plausible to human inspectors. The quantitative performance evaluation shows that CheetahTraj provides good visual quality by sampling only a small number of trajectories. In addition, CheetahTraj produces high quality visualizations for arbitrary target regions in less than 1 second for all 3 experiment datasets and the visualization delay is below 0.1 second in most cases.

We illustrate the merits of our CheetahTraj framework in Figure 1. Figure 1(D) and (E) are the visualizations produced by CheetahTraj on the Porto dataset with sampling rate 0.1% and 1%, respectively. Compared with uniform random sampling (i.e., Random) and diversity based sampling (i.e., DTW) in Figure 1(B) and (C), Figure 1(D) and (E) are obviously more similar to the full dataset visualization in Figure 1(A). Figure 1(F) is produced by CheetahTraj using the same parameters as Figure 1(E) but the trajectories are colored according to their algorithm-generated representativeness (warmer color means more representative). Compared with Figure 1(A), the main routes in the dense region can be identified much more easily, which shows that CheetahTraj effectively reduces visual clutter. Last but not least, it takes CheetahTraj only 0.116 seconds and 0.339 seconds to generate Figure 1(D) and (E), respectively, while the full visualization in Figure 1(A) takes 16.154 seconds.

## 2 BACKGROUND AND RELATED WORK

In this part, we discuss related works on *trajectory visualization methods* and *interactive data visualization for large datasets*.

**Trajectory Visualization Methods:** A trajectory is a sequence of spatial locations (e.g., GPS positioning results) and trajectories are the most common representations of object movements. Existing trajectory visualization methods can be classified into three categories according to the form of visualization [16], i.e., *point-based*, *region-based*, and *line-based*. We give a brief introduction to these methods and refer the interested readers to [16] for more details.

Point-based visualization plots the locations in the trajectories independently and captures the overall spatial distribution of the moving objects. Many density-based methods [12, 13, 28, 40], 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 [21, 36]. In this work, we focus on line-based visualization, which uses polylines to connect the locations in each trajectory (see an example in Figure 2) and preserves the continuous movement information of objects [22, 23, 43]. However, line-based visualization is known to suffer from severe visual clutter, especially when the dataset is large. Several techniques have

[1]Mapping is the time to map the GPS points to the screen points, and rendering is the time to render the graphics. The details can be found in Section 6.

been proposed to alleviate visual clutter, such as clustering-based techniques [36] and advanced interaction techniques [19].

**Interactive Visualization for Large Datasets:** Visualization generation has a long latency for large datasets due to heavy data processing/graphic rendering, which harms the responsiveness of interactive visualization. Therefore, both the information retrieval and visualization communities began to advance techniques to reduce visualization latency for large datasets. We briefly elaborate these techniques as follows.

*Aggregation-based techniques* divide the entire area into basic units and visualize the aggregated information of the trajectories for each unit [21, 36, 39]. For more details on aggregation-based techniques, we refer the reader to [6, 7]. Our problem and solutions are different from these works as we focus on visualizing the raw trajectories, instead of aggregated statistics. *Sampling-based techniques* [10, 15, 18, 24, 29, 31, 32, 41] try to reduce the dataset to a subset with some special characteristics: such as blue noise property [32], multi-class property [15] or maximize some user-defined quality [41]. The work most relevant to ours is [29], which is designed for scatter plots (a form of point-based visualization). It reduces the number of points in a plot while preserving the spatial distribution of the points in the original dataset. The techniques in [29] cannot be applied to our trajectory visualization problem as trajectory is more complex than individual scatter points (e.g., the order of GPS points is essential and the trajectories could have a large variance in length). Some works simplify a trajectory by sampling important points to reduce data size [35, 44] or alleviate visual clutter [11, 37]. These works are orthogonal to ours as *we are sampling complete trajectories instead of points in a trajectory.* Some *execution optimizations* have also been proposed to reduce visualization latency. Chan et al. propose ATLAS [14], which utilizes caching for efficient data communication between server and client. ATLAS also exploits a powerful multi-core server to accelerate visual analysis tasks. Piringer et al. [30] propose an architecture for interactive visual exploration, which utilizes multi-core devices and avoids the common pitfalls of multi-threading to provide quick visual feedback. Our work is complementary to these execution optimizations as we mainly focus on the algorithm perspective.

**Novelties of our work.** To the best of our knowledge, we are the first to formulate the quality optimal trajectory sampling problem to accelerate visualization on large-scale datasets. We devise effective algorithms for this problem, which not only provide visual quality guarantee but also reduce the well-know visual clutter in trajectory visualization. Based on these algorithm, we design the CheetahTraj framework with a tailored InvQuad-tree index to produce high quality visualization for arbitrary target region with low latency.

## 3 PROBLEM FORMULATION

In this section, we first formally define the *quality optimal sampling problem* (QOSP) for large-scale trajectory data visualization, and then show that it is NP-hard to solve the problem exactly.

### 3.1 Problem Definition

We motivate our definition of the *visualization quality function* by introducing how line-based trajectory visualization works. As elaborated earlier, a trajectory contains a sequence of 2-dimensional

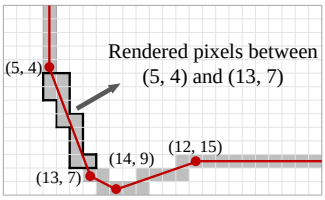



**Figure 2: Illustration of line-based trajectory visualization.**

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 2(A), the result of line-based trajectory visualization can be regarded as a 2-dimensional array of boolean variables with 1 indicting that a pixel has been marked. Thus, we can treat a visualization result as a set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{n}$ that contains all marked pixels. This observation leads to the following definition of visualization quality function

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

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

Problem 1 (Quality Optimal Sampling Problem, QOSP). *Let the entire trajectory dataset be $\mathcal{T}$ and a sample set that contains some trajectories from $\mathcal{T}$ be $\mathcal{R}$. Using $\mathtt{V}(\mathcal{U})$ to denote the visualization result set derived from a trajectory set $\mathcal{U}$, with a sampling rate $\alpha$, the quality optimal sampling problem finds a set $\mathcal{R}$ that satisfies*

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

Note that we are sampling *complete trajectories* instead of *individual locations* in QOSP such that the lines and orientations in the trajectories are persevered. Given a visualization quality threshold $\tau$, the QOSP problem can be transformed to find the sampled trajectory set $\mathcal{R}$ with the smallest $\alpha$ that meets the quality requirement, i.e, $\mathtt{Q}(\mathtt{V}(\mathcal{T}), \mathtt{V}(\mathcal{R})) \geq \tau$.

### 3.2 Hardness Analysis

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

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

The transformations use the fact that $\mathbb{V}(\mathcal{R})\subseteq\mathbb{V}(\mathcal{T})$ as $\mathcal{R}\subseteq\mathcal{T}$, and the ground-truth marked point set $\mathbb{V}(\mathcal{T})$ has constant cardinality. The last line shows that QOSP is equivalent to the famous set cover maximization problem[2]. Specifically, given an integer $k$, and a collection of sets $\mathcal{T}=\{t_1,t_2,\cdots,t_n\}$, set cover maximization finds a subset $\mathcal{R}\subset\mathcal{T}$ such that $|\mathcal{R}|=k$ and the number of covered elements $|\cup_{t_i\in\mathcal{R}} t_i|$ is maximized. The set cover maximization problem is well-known to be NP-hard [17]. For sampling-based methods, the visualization quality is determined by the sample set $\mathcal{R}$, and thus we use $\mathbb{Q}(\mathcal{R})$ to denote $\mathbb{Q}(\mathbb{V}(\mathcal{T}),\mathbb{V}(\mathcal{R}))$ for conciseness.

## 4 SOLUTIONS FOR QOSP

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

### 4.1 Visual Quality Guaranteed Sampling VQGS

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

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

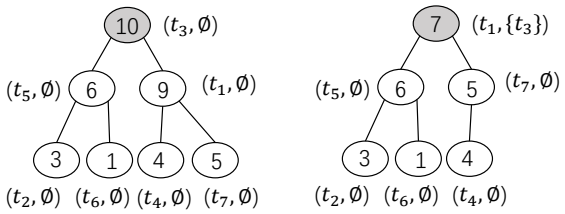
Our visual quality guaranteed sampling method (VQGS) is presented in Algorithm 1, which takes the trajectory dataset $\mathcal{T}$ and a sampling rate $\alpha$ as input (i.e., $k=\lceil\alpha|\mathcal{T}|\rceil$). VQGS employs a greedy paradigm and finds the trajectory tmp in $\mathcal{T}$ that maximizes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

Theorem 1. *Given a sample rate $\alpha$, and let the optimal solution to QOSP defined in Equation (2) be $\mathcal{R}^\star$ and the solution provided by Algorithm 1 be $\mathcal{R}$, we have $\mathbb{Q}(\mathcal{R})\geq 0.632*\mathbb{Q}(\mathcal{R}^\star)$.*

Theorem 1 follows directly from the submodularity of the visualization quality function $\mathbb{Q}(\mathcal{R})$, and it is well known that greedy solution provides a 0.632 approximation of the optimal solution for a submodular function [20]. As $\mathbb{Q}(\mathcal{R})$ is a linear scaling of $|\mathbb{V}(\mathcal{R})|$ as shown in Equation (3), we prove $|\mathbb{V}(\mathcal{R})|$ is submodular as follows.

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

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




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

**Figure 3: Heap-based lazy computation.**

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

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

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

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

Proof. In each iteration, Algorithm 1 computes the trajectory with the largest number of uncovered pixels in dataset $\mathcal{T}$. It take $O(m)$ cost to compute the number of uncovered pixels for each trajectory in $\mathcal{T}$. Algorithm 1 runs for $k=\lceil\alpha|\mathcal{T}|\rceil$ iterations. Hence, the total cost is $O(k\cdot m\cdot|\mathcal{T}|)=O(\alpha\cdot m\cdot|\mathcal{T}|^2)$. □

The high complexity of Algorithm 1 hurts its scalability for large-scale trajectory datasets. For example, the Porto dataset contains 2.39 millions taxi trajectories, Algorithm 1 takes 413.6 seconds to obtain the result set $\mathcal{R}$ with sampling rate 0.1%.

**Heap-based lazy Computation:** Algorithm 1 essentially adds the trajectory that maximizes $\Delta(\mathcal{R},t)=|\mathbb{V}(\mathcal{R}\cup t)|-|\mathbb{V}(\mathcal{R})|$ to $\mathcal{R}$ in each iteration. Lemma 1 shows that the contribution of a trajectory (i.e., $\Delta(\mathcal{R},t)$) cannot increase when Algorithm 1 runs for more iterations because $\Delta(\mathcal{R}',t)\leq\Delta(\mathcal{R},t)$ for $\mathcal{R}\subset\mathcal{R}'$. For example, the contribution of $t_1$ is 9 at the first iteration (i.e., $\mathcal{R}=\emptyset$), see Figure 3(a). Its contribution turns 7 when $\mathcal{R}=\{t_3\}$ at the second iteration, as shown in Figure 3(b). Based on this property, we can use $\Delta(\mathcal{R},t)$ calculated in the previous iterations to prune it from contribution computation. Specifically, if we have $\Delta(\mathcal{R},t)\leq\Delta(\mathcal{R}',t')$, in which $\mathcal{R}$ and $\mathcal{R}'$ are a previous and the current sample set, respectively, we know that $t$ can not be added to the sample set in the current iteration as $\Delta(\mathcal{R}',t)\leq\Delta(\mathcal{R}',t')$ and $t'$ is a better choice. As shown in Figure 3(b), even if we do not know the exact value of $\Delta(\mathcal{R}'=\{t_3\},t_7)$, we can conclude that $t_7$ will not be added to the sample set in the second iteration as $\Delta(\mathcal{R}=\emptyset,t_7)=5<\Delta(\mathcal{R}'=\{t_3\},t_1)=7$.

To implement this idea, we maintain a max-heap for the number of uncovered pixels in each trajectory and update the contribution of a trajectory only when necessary, i.e., computing in a lazy manner. Consider a tiny example with 7 trajectories, i.e., $t_1$ to $t_7$. Figure 3(a) shows the initial max-heap and the contributions of

trajectories $t_1$ to $t_7$ w.r.t. result set $\mathcal{R} = \emptyset$. At the first iteration, the root node of the max-heap, $t_3$ in Figure 3(A), is selected. At the second iteration, the number of uncovered pixels of the new root node $t_1$ is updated to 7 w.r.t. result set $\mathcal{R} = \{t_3\}$ (the gray node in Figure 3(B)). Then $t_1$ is selected at the second iteration without computing the contributions of other trajectories w.r.t. $\mathcal{R} = \{t_3\}$. This is because the contributions of these trajectories are less than 7 when $\mathcal{R} = \emptyset$, according to Lemma 1, their contributions must be smaller than 7 when $\mathcal{R} = \{t_3\}$. The efficiency of Algorithm 1 improves significantly with heap-based lazy computation. Recall that Algorithm 1 takes 413.6 seconds with $\alpha = 0.1\%$ on Porto while our performance-optimized VQGS needs only 1.2 seconds.

## 4.2 Advanced Approach VQGS$^+$

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

**Trajectory data distribution:** Considering the Porto trajectory dataset, Figure 4(A) is the visualization result of VQGS with sampling rate 0.5%. It is obvious that the trajectories follow a non-uniform distribution, and there are some dense regions and sparse regions as illustrated by the two rectangles in Figure 4(A). There are many points in the dense region, which creates visual clutter and makes it difficult to identify the main roads.

**Human perception capability:** Comparing Figures 4(A) and (B), it is easier to tell their differences in the sparse regions than in the dense regions. This is because human perception has limited capability, and hence two visualizations look indistinguishable if both of them contain a large number of points in the same area. The two dense regions look similar although Figure 4(B) contain fewer points in this region than Figure 4(A). However, for the sparse region, VQGS loses some trajectories and it is easy to tell the differences between Figures 4(A) and 4(B).

Based on the two observations above, we can improve VQGS by delivering richer information in the sparse regions and reducing visual clutter in the dense regions. VQGS$^+$ in Algorithm 2 achieves both objectives using a perception tolerance parameter $\delta$, which models the perception capability of human. Specifically, if pixel $(x, y)$ in the canvas is marked by the result set $\mathcal{R}$, the pixels around $(x, y)$, i.e., from $(x-\delta, y-\delta)$ to $(x+\delta, y+\delta)$, do not need to be marked as they are close to $(x, y)$ and human perception cannot tell nearby pixels apart. We modify VQGS in Algorithm 1 to incorporate the perception tolerance parameter $\delta$ in Algorithm 2. VQGS$^+$ measures the contribution of each trajectory $t_i$ w.r.t. the augmented visualized point set $\mathsf{V}(\mathcal{R})^+$ in Line 4, where $\mathsf{V}(\mathcal{R})^+$ includes both pixels on the selected trajectories and their tolerance pixels (in Line 6). We also use the heap-based lazy computation to speedup VQGS$^+$.

VQGS$^+$ in Algorithm 2 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). Compared with VQGS, VQGS$^+$ is more likely to sample trajectories in the sparse regions as their pixels are less likely to be covered by other trajectories as shown in Figure 4. Moreover, reducing the number of trajectories sampled from the dense regions helps to reduce visual clutter.

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

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 $\leftarrow \arg\max_{t_i \in \mathcal{T}} |t_i \cup \mathsf{V}(\mathcal{R})^+|$
5: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{\text{tmp}\}$
6: $\quad \mathsf{V}(\mathcal{R})^+ \leftarrow \mathsf{V}(\mathcal{R})^+ \cup \text{augment}(\text{tmp}, \delta)$
7: **for** each $t$ in $\mathcal{T}$ **do** ▷ Representative encoding
8: $\quad tr \leftarrow \arg\min_{t_i \in \mathcal{R}} |t - \text{augment}(t_i, \delta)|$
9: $\quad tr.\text{cnt}++$
10: Return $\mathcal{R}$

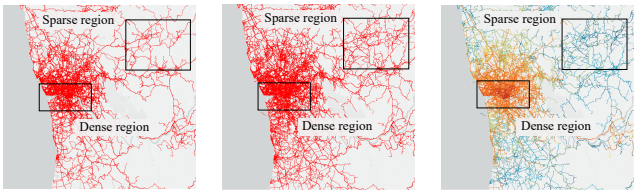
One subtlety is that different $\delta$ needs to used for different *zoom levels* (or regions with different sizes). For example, Google map [2] provides zoom levels from 0 to 20, with level 0 providing the largest visualization range (i.e., the whole world) but the lowest resolution, and level 20 providing the smallest visualization range (e.g., individual building) but the highest resolution. We provided an illustration of zoom level in Figure 6 and users may select different zoom levels for visualization according their needs. Note that we define $\delta$ on the highest zoom level (i.e., using the raw distance of the locations) to account for different resolutions. If the zoom level is small (i.e., the visualization region is large), we can apply a large $\delta$ because locations with a large raw distance look close to each other in the visualization and we can afford to lose more details. If the zoom level is large (i.e., the visualization region is small), we need to use a small $\delta$ as users typically want to investigate some fine-grained details in this case and using a large $\delta$ will lose these details.

**Color encoding scheme:** 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 $t_i$ as the size of its *reverse nearest neighbor set*, which contains the trajectories in $\mathcal{T}$ that has $t_i$ as its nearest neighbor in $\mathcal{R}$. The distance between trajectory $t$ and $t_i$ is defined as the number of pixels in $t$ that can not be covered by the augmented pixels of $t_i$. We compute the representativeness of each trajectory in $\mathcal{R}$ in Lines 7-9 in Algorithm 2. Figure 4(C) shows the visualization result by encoding trajectories with larger representativeness with warmer colors, for which the main roads in the dense region is clearer than Figure 4(B) with very warm colors.

# 5 THE CheetahTraj FRAMEWORK

Recall that our goal is to provide high quality trajectory visualization for any user selected query region with low latency. In this section, we first introduce the motivation behind the CheetahTraj framework, then present its two key procedures: *index building*, and *query processing*.

**Motivation of CheetahTraj:** Given a user selected region query $Q$, a naive visualization procedure with our sampling algorithms works as follows: it first retrieves all trajectories (or trajectory segments) that are in this region (a.k.a, WayPoint query [25]), then it invokes VQGS$^+$ (or VQGS) to obtain a set $\mathcal{R}$ of sample trajectories, and finally the trajectories in $\mathcal{R}$ are rendered to the canvas (e.g.,




(A) VQGS (B) VQGS$^+$ (C) VQGS$^+$CE

**Figure 4: QOSP solution VQGS$^+$ on Porto ($\alpha = 0.5\%, \delta = 64$)**

.

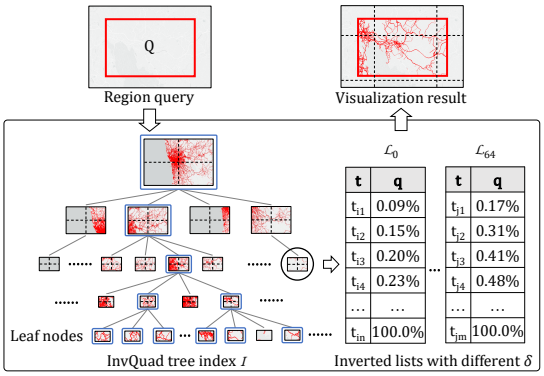



**Figure 6: The CheetahTraj framework.**

displaying device). VQGS$^+$ has short *visualization time* as it effectively reduces the number of processed locations by sampling. However, VQGS$^+$ has a long *sampling time* (e.g., several seconds to tens of seconds) even with our performance optimization techniques. Hence, the naive procedure can not achieve low latency for large-scale trajectory visualization. To tackle this problem, we propose the CheetahTraj framework as illustrated in Figure 6. CheetahTraj consists of three modules: (i) index building, (ii) query processing, and (iii) result visualization, we elaborate them as follows.

## 5.1 Index Building

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

As shown in the example InvQuad-tree index $\mathcal{I}$ at the bottom of Figure 6, we exploit a quad-tree to recursively partition the entire area (spanned by the trajectory dataset) into smaller areas and manage each area with a tree node. For each tree node, we run VQGS$^+$ using the trajectories (or trajectory segments) in its associated area as input to compute the *visualization quality inverted lists* for this area. $\mathcal{L}_0$ and $\mathcal{L}_{64}$ in Figure 6 are two example visualization quality inverted lists, in which the subscripts are the values of $\delta$ for this list. Specifically, we compute several inverted lists with different $\delta$ values[3] to support the efficient quality guaranteed result visualization at various zoom levels. For each inverted list, (i) VQGS$^+$ terminates until the quality of the sample set is 100%, i.e., the visualization result of the sample set is the same as the full dataset; (ii) the trajectory selected at each iteration of VQGS$^+$ is stored in the inverted list with its *cumulative quality* in ascending order. Take inverted list $\mathcal{L}_0$ in Figure 6 for example, $t_{i4}$ is the trajectory selected at the 4th iteration, $t_{i4}$'s cumulative quality is 0.23%, which means that the quality achieved by $\{t_{i1}, t_{i2}, t_{i3}, t_{i4}\}$ as a whole is 0.23%. With the quality inverted list, searching a quality guaranteed sample set for a query region can be conducted efficiently via binary search.

[3]We set $\delta$ as 0 (i.e., VQGS), 4, 8, 16, 32, 64. We need quality inverted lists with different $\delta$ for one area as the area may be covered by query regions of different sizes, and we use lists with larger $\delta$ for larger query region as discussed in Section 4.2.

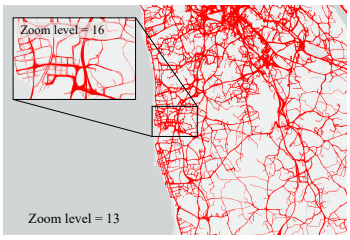



**Figure 5: An illustration of zoom levels.**

## 5.2 Query Processing

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

**Algorithm 3** Query($Q$, $\tau$, InvQuad node $\mathcal{N}$, result $\mathcal{R}$)

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

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

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

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

**Table 2: Statistics of the datasets used in the experiments**

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

In more general cases, we also select some areas that only intersect with the query region $Q$ and the sample set $\mathcal{R}$ may not satisfy $\frac{m_k}{n_k} \geq \tau$ for these areas. This does not significantly affect visualization quality for two reasons: (i) these areas are the leaf nodes of the InvQuad-tree index and thus reside on the border of the query region. When exploring the map, human tends to move the region of interest to the screen center, where is more "close" to eyes [1]. (ii) the areas of the border regions are small w.r.t. the query region if the InvQuad-tree has a sufficient height (i.e., the leaf nodes have a small area).

# 6 EXPERIMENTAL EVALUATION

We first present a case study of the visualizations provided by CheetahTraj in Section 6.1 to demonstrate its good visualization quality. In Section 6.2, we conduct a comprehensive user study to compare the visualization quality of different methods. In Section 6.3, we quantitatively compare the visual quality and efficiency of CheetahTraj with the baselines.

**Experiment Settings:** We use 3 real-world trajectory datasets, i.e., Porto, Shenzhen and Chengdu, and their statistics are summarized in Table 2. The experiments are conducted on a machine with an Intel i7-8700 CPU, 24 GB memory and an NVIDIA GeForce GTX1080 GPU with 8 GB on-chip memory, running on Windows 10. All methods are implemented using Java 1.8. UnfoldingMap 0.9.92 [5] is used to provide interactive map and GPS mapping, and the Processing 3 library [3] is used for rendering. All timing results are measured in single-thread mode.

**Baselines:** We compare CheetahTraj with three baselines, i.e., Full, Random and DTW. Full visualizes all trajectories in the user selected region while Random samples trajectories in the user selected region uniformly at random. DTW is based on the DTW distance between trajectories [11] and designed by us to select trajectories with good diversity. Specifically, DTW samples the trajectory that maximizes the aggregate DTW distance to all remaining trajectories in each step. It takes DTW several days to run on the experiment datasets because it needs to compute expensive DTW distance (quadratic complexity w.r.t. trajectory length) between all trajectory pairs. For fair comparison, we ensure that Random and DTW use the same number of trajectories as CheetahTraj.

## 6.1 Case Study

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

*6.1.1 Overview visualization.* We illustrate the visualization results of different methods for the entire Porto dataset in Figure 1.

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

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

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

*6.1.2 Detail visualization.* We analyze the visualizations produced by different methods for small areas with details by investigating two regions of interest in the Porto dataset in Figure 7.

**Reduce visual clutter and preserve micro structures for dense region:** At zoom level 15, region B in Figure 7(A) is the center of Porto and has the highest concentration of trajectories. Therefore, Full suffers from severe visual clutter and it is difficult to identify the road networks in Figure 7(B1). Random and DTW in Figure 7(B2) and (B3) reduce the visual clutter to some extent by sampling some trajectories. CheetahTraj in Figure 7(B4) is more successful in reducing the visual clutter of Full and allows to identify a much larger number of routes. In addition, CheetahTraj preserves more micro structures of the trajectories than Random and DTW, e.g., the circular route in the dashed circular region.

**Preserve overall layout for sparse region:** At zoom level 14, region C in Figure 7(A) contains the city of Casino Espinho and has fewer trajectories than the dense region in the center. In this case, the sampling methods need to keep the overall layout of the trajectories to provide good visualization quality. Compared with Full in Figure 7(C1), Random and DTW in Figure 7(C2) and (C3) fail to meet this requirement as they do not show any trajectory for areas far from the city, e.g., in the dashed circle. This makes their entire visualization layout very different from Full. In contrast, CheetahTraj in Figure 7(C4) preserves the trajectories in areas far from the city and has a overall layout similar to Full.

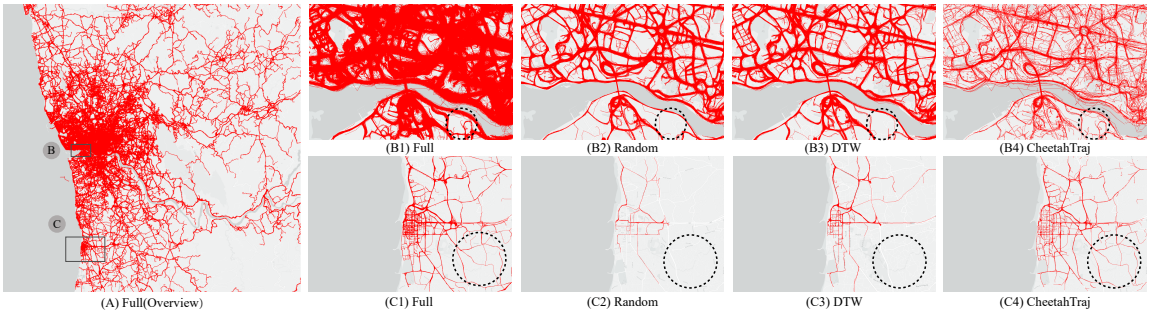
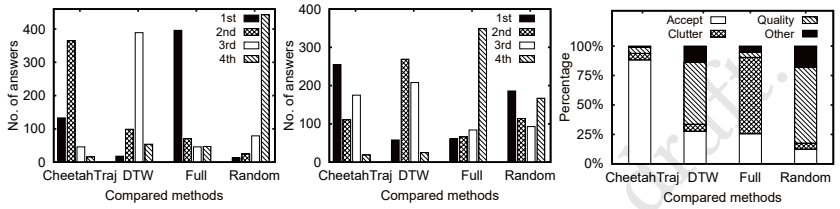



**Figure 7: Case study of the visualization quality of CheetahTraj for two detail regions.**




(A) T1: visual quality study (B) T2: visual clutter study (C) T3: acceptance study

**Figure 8: User study results of different visualization methods.**

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

## 6.2 User Study

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

**Settings:** We recruited 35 participants with 10 females, 25 males, aged 19 to 31 with a mean of 24.78 for the user study. The user study is conducted on the Porto and Shenzhen datasets, and four visualization methods are investigated, i.e., Full, Random, DTW and CheetahTraj. We define a *comparable group* as the set of visualizations generated by different methods with the same center point and zoom scale. We manually generated 66 comparable groups and 264 visualizations results (66 groups × 4 visualizations). The user study system is a web-based platform, in which all visualizations are displayed with a resolution of 450*300. We are interested in the visual quality and visual clutter of the visualizations, and thus designed three tasks for a comparable group:

- Task 1 (T1): rank the visualizations in a group from the highest visual quality to the lowest visual quality.
- Task 2 (T2): rank the visualizations in a group from the least visual clutter to the most severe visual clutter.
- Task 3 (T3): select the visualizations considered acceptable (multiple choices allowed) and choose the reason (including "severe visual clutter", "poor visual quality" and "others") for the visualizations considered unacceptable.

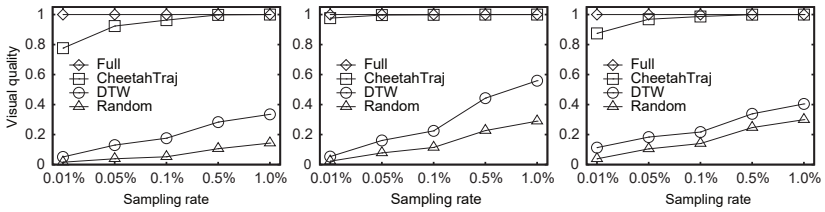
**User study procedure:** When the participants enter the user study system, they are given a tutorial on how to conduct the tasks to get familiar with the interface and tasks. For each participant, we randomly select 16 comparable groups. For each comparable group, the 4 visualizations (*without specifying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

**Result analysis:** Figure 8(A) reports the visual quality ranking of the 4 methods in T1. The results show that Full ranks 1st in most cases while CheetahTraj usually ranks 1st or 2nd. In contrast, DTW and Random rank 3rd and 4th at most times. This suggests that the visualizations generated by CheetahTraj are more appealing to the participants than DTW and Random. Figure 8(B) reports the anti-visual clutter ranking of the 4 methods in T2. The results show that visual clutter is most severe for Full, ranking 4th in most cases. While CheetahTraj is the most successful in reducing visual clutter, ranking 1st in 255 out of the 560 cases and ranking 4th for only 19 cases. We report the acceptance frequency of the 4 methods in T3 using bar chart in Figure 8(C). Each column corresponds to a method, and from bottom to top, the lengths of the bars indicate the percentage of participants choosing "acceptable", "not acceptable due to visual clutter", "not acceptable due to poor visual quality" and "not acceptable for other reasons". The results show that CheetahTraj is regarded acceptable for about 88.2% of the cases, and the other methods have much lower acceptance rate than CheetahTraj.

## 6.3 Quantitative Evaluation

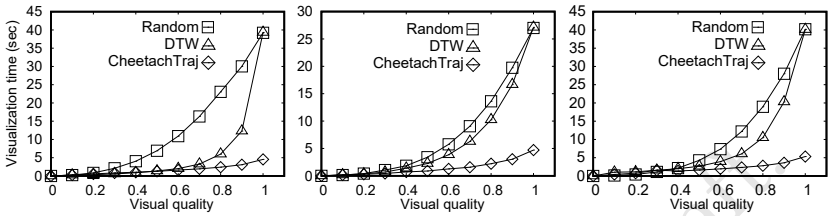
In this part, we quantitatively evaluate the visual quality and efficiency of CheetahTraj on the three real-world trajectory datasets.

**Visual quality:** Figure 9 reports the visualization quality (defined in Equation (1)) of the trajectory sampling methods under different sampling rates. We consider the entire region in each dataset for this experiment. The results show that CheetahTraj achieves significantly higher quality than Random and DTW under the same sampling rate. This is because our sampling algorithms VQGS and VQGS$^+$ are designed with explicit considerations for visual quality. The quality of CheetahTraj approaches 1 when the sampling rate is still less than 1% for all 3 datasets. DTW has a higher quality than Random because it considers the diversity of trajectories.



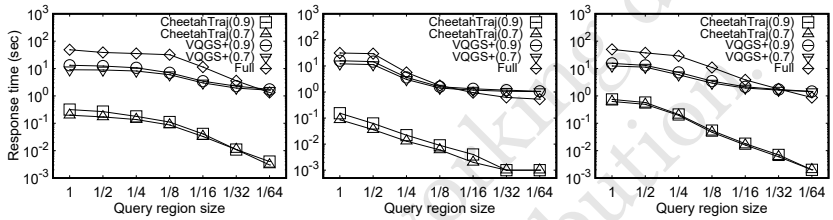

(A) Porto (B) Shenzhen (C) Chengdu

**Figure 9: Effect of sampling rate on visual quality.**



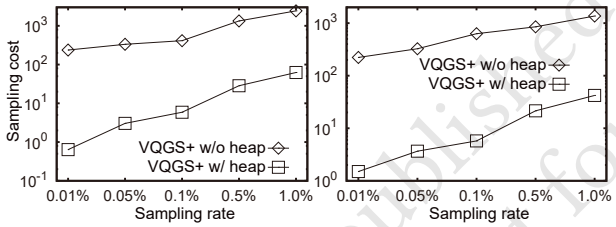

(A) Porto (B) Shenzhen (C) Chengdu

**Figure 10: Effect of visual quality on visualization time.**




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

**Figure 11: Effect of region size on end-to-end response time.**




(A) Porto (B) Shenzhen

**Figure 12: Running time of VQGS$^+$ w/o optimization.**

In Figure 10, we report the visualization cost (i.e., the wall clock time to generate visualization result using the sampled trajectories) for the methods under different quality requirements. We still consider the entire region in this experiment and the results show that CheetahTraj has significantly shorter visualization time than Random and DTW under the same quality requirement. This is because CheetahTraj is designed to sample trajectories that optimize visual quality and thus requires a smaller number of trajectories to achieve the same quality compared with Random and DTW.

**Efficiency of** CheetahTraj**:** We evaluate the *response time* of our CheetahTraj framework under different quality guarantees and region sizes in Figure 11. The response time of CheetahTraj is the end-to-end time for generating visualization for a selected region, which includes querying the CheetahTraj index and computing the visualization. We also include the response time of VQGS$^+$ (with $\delta = 8$), which uses online sampling instead of querying the index in CheetahTraj framework. We constrain the regions to be rectangles with a constant height/width ratio and measure the size of a region by dividing its height over the height of the entire region. For each region size, we report the average response time of three typical regions, i.e., a dense region, a sparse region and a medium region. The results show that CheetahTraj achieves a short response time (less than 1 second in all cases and 0.2 second for most cases) for all region sizes and quality guarantees. VQGS$^+$ is 1 to 2 orders of magnitude slower than CheetahTraj and takes at most 14.802 seconds in all cases. These results show that VQGS$^+$ cannot support interactive visual exploration and the InvQuad-tree index in CheetahTraj is effective in improving efficiency. The response time decreases rapidly when the region size shrinks as there are fewer trajectories in a smaller region. However, the response time required to achieve a high quality (e.g., 0.9) is not significantly longer than a low quality (e.g., 0.7) as quality improves quickly with the number of sampled trajectories as a shown in Figure 9.

**Optimizations and costs:** In Figure 12, we report the running time of VQGS$^+$ with and without the heap-based lazy computation. The results show that the heap-based optimization reduces the running time of VQGS$^+$ by around 2 orders of magnitude. We also report the building time and memory cost of the InvQuad-tree index in Table 3. For all three datasets, it takes less than 10 minutes to build the InvQuad index with a height of 13. The memory cost of the InvQuad index in the last column is also comparable with the size of the raw data shown in the first column.

## 7 CONCLUSIONS

This paper presents the CheetahTraj framework, which achieves high visual quality and low visualization latency for large-scale

**Table 3: The cost of InvQuad-tree index**

| Dataset (size) | Height | Building time | Memory size |
|---|---|---|---|
| Porto (1.44G) | 13 | 526.390s | 3.65GB |
| Shenzhen (1.02G) | 13 | 435.291s | 3.12GB |
| Chengdu (1.49G) | 13 | 454.151s | 3.71GB |

trajectory datasets. CheetahTraj provides guaranteed visual quality in trajectory sampling by formulating a quality optimal sampling problem and developing effective solutions including VQGS and VQGS. Low visualization latency is achieved with the InvQuad-tree index, which allows to use the sampling results computed offline. Experiment results show that CheetahTraj consistently provides high quality visualization in different cases and its visualization time is orders of magnitude shorter than full visualization.

## REFERENCES


[1] 2020. Fitts' Law: Tracking users' clicks. https://www.interaction-design.org/literature/article/fitts-law-tracking-users-clicks.
[2] 2020. Google map. https://www.google.com/maps/preview.
[3] 2020. The open-source graphical library. https://processing.org.
[4] 2020. Shenzhen dataset. http://jtys.sz.gov.cn/.
[5] 2020. Unfolding maps. http://unfoldingmaps.org/.
[6] Natalia Adrienko and Gennady Adrienko. 2010. Spatial generalization and aggregation of massive movement data. *TVCG* 17, 2 (2010), 205–219.
[7] Gennady Andrienko and Natalia Andrienko. 2008. Spatio-temporal aggregation for visual analysis of movements. In *IEEE symposium on visual analytics science and technology*. 51–58.
[8] Gennady L. Andrienko, Natalia V. Andrienko, Wei Chen, Ross Maciejewski, and Ye Zhao. 2017. Visual Analytics of Mobility and Transportation: State of the Art and Further Research Directions. *TITS* 18, 8 (2017), 2232–2249.
[9] Gennady L. Andrienko, Natalia V. Andrienko, Steven Mark Drucker, Jean-Daniel Fekete, Danyel Fisher, Stratos Idreos, Tim Kraska, Guoliang Li, Kwan-Liu Ma, Jock D. Mackinlay, Antti Oulasvirta, Tobias Schreck, Heidrun Schumann, Michael Stonebraker, David Auber, Nikos Bikakis, Panos K. Chrysanthis, George Papastefanatos, and Mohamed A. Sharaf. 2020. Big Data Visualization and Analytics: Future Research Challenges and Emerging Applications. In *EDBT/ICDT joint conference (CEUR Workshop Proceedings, Vol. 2578)*. CEUR-WS.org.
[10] Leilani Battle, Michael Stonebraker, and Remco Chang. 2013. Dynamic reduction of query result sets for interactive visualizaton. In *IEEE BigData*. 1–8.
[11] OM Borcan. 2012. *Improving visualization of trajectories by dataset reduction and line simplification*. Master's thesis.
[12] Giuseppe Borruso. 2008. Network density estimation: a GIS approach for analysing point patterns in a network space. *Transactions in GIS* 12, 3 (2008), 377–402.
[13] Junghoon Chae, Dennis Thom, Yun Jang, SungYe Kim, Thomas Ertl, and David S Ebert. 2014. Public behavior response analysis in disaster events utilizing visual analytics of microblog data. *Computers & Graphics* 38 (2014), 51–60.
[14] Sye-Min Chan, Ling Xiao, John Gerth, and Pat Hanrahan. 2008. Maintaining interactivity while exploring massive time series. In *IEEE Symposium on Visual Analytics Science and Technology*. 59–66.
[15] Haidong Chen, Wei Chen, Honghui Mei, Zhiqi Liu, Kun Zhou, Weifeng Chen, Wentao Gu, and Kwan-Liu Ma. 2014. Visual abstraction and exploration of multi-class scatterplots. *TVCG* 20, 12 (2014), 1683–1692.
[16] Wei Chen, Fangzhou Guo, and Fei-Yue Wang. 2015. A survey of traffic data visualization. *TITS* 16, 6 (2015), 2970–2984.
[17] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2009. *Introduction to algorithms*. MIT press.
[18] Bolin Ding, Silu Huang, Surajit Chaudhuri, Kaushik Chakrabarti, and Chi Wang. 2016. Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee. In *SIGMOD*. 679–694.
[19] Nivan Ferreira, Jorge Poco, Huy T Vo, Juliana Freire, and Cláudio T Silva. 2013. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *TVCG* 19, 12 (2013), 2149–2158.
[20] Satoru Fujishige. 2005. *Submodular functions and optimization*.
[21] Diansheng Guo. 2009. Flow mapping and multivariate visualization of large spatial interaction data. *TVCG* 15, 6 (2009), 1041–1048.
[22] Hanqi Guo, Zuchao Wang, Bowen Yu, Huijing Zhao, and Xiaoru Yuan. 2011. Tripvista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection. In *IEEE Pacific Visualization Symposium*. 163–170.
[23] Christophe Hurter, Benjamin Tissoires, and Stéphane Conversy. 2009. Fromdady: Spreading aircraft trajectories across views to support iterative queries. *TVCG* 15, 6 (2009), 1017–1024.
[24] Albert Kim, Eric Blais, Aditya G. Parameswaran, Piotr Indyk, Samuel Madden, and Ronitt Rubinfeld. 2015. Rapid Sampling for Visualizations with Ordering Guarantees. *PVLDB* 8, 5 (2015), 521–532.
[25] Robert Krüger, Dennis Thom, Michael Wörner, Harald Bosch, and Thomas Ertl. 2013. TrajectoryLenses–a set-based filtering and exploration technique for long-term trajectory data. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 451–460.
[26] Bum Chul Kwon, Janu Verma, Peter J Haas, and Cagatay Demiralp. 2017. Sampling for scalable visual analytics. *IEEE Computer Graphics and Applications* 37, 1 (2017), 100–108.
[27] Dongyu Liu, Di Weng, Yuhong Li, Jie Bao, Yu Zheng, Huamin Qu, and Yingcai Wu. 2016. Smartadp: Visual analytics of large-scale taxi trajectories for selecting billboard locations. *TVCG* 23, 1 (2016), 1–10.
[28] Siyuan Liu, Jiansu Pu, Qiong Luo, Huamin Qu, Lionel M Ni, and Ramayya Krishnan. 2013. VAIT: A visual analytics system for metropolitan transportation. *TITS* 14, 4 (2013), 1586–1596.
[29] Yongjoo Park, Michael Cafarella, and Barzan Mozafari. 2016. Visualization-aware sampling for very large databases. In *ICDE*. 755–766.
[30] Harald Piringer, Christian Tominski, Philipp Muigg, and Wolfgang Berger. 2009. A multi-threading architecture to support interactive visual exploration. *TVCG* 15, 6 (2009), 1113–1120.
[31] Xuedi Qin, Yuyu Luo, Nan Tang, and Guoliang Li. 2020. Making data visualization more efficient and effective: A survey. *The VLDB Journal* 29, 1 (2020), 93–117.
[32] Tobias Rapp, Christoph Peters, and Carsten Dachsbacher. 2019. Void-and-Cluster Sampling of Large Scattered Data and Trajectories. *TVCG* 26, 1 (2019), 780–789.
[33] Ben Shneiderman. 1984. Response time and display rate in human performance with computers. *Comput. Surveys* 16, 3 (1984), 265–285.
[34] Bo Tang, Man Lung Yiu, Kyriakos Mouratidis, and Kai Wang. 2017. Efficient motif discovery in spatial trajectories using discrete fréchet distance. In *EDBT*.
[35] Marc van Kreveld, Maarten Löffler, and Lionov Wiratma. 2018. On Optimal Polyline Simplification using the Hausdorff and Fréchet Distance. *arXiv e-prints*, Article arXiv:1803.03550 (March 2018), arXiv:1803.03550 pages. arXiv:1803.03550 [cs.CG]
[36] Tatiana Von Landesberger, Felix Brodkorb, Philipp Roskosch, Natalia Andrienko, Gennady Andrienko, and Andreas Kerren. 2015. Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *TVCG* 22, 1 (2015), 11–20.
[37] K. Vrotsou, H. Janetzko, C. Navarra, G. Fuchs, D. Spretke, F. Mansmann, N. Andrienko, and G. Andrienko. 2015. SimpliFly: A Methodology for Simplification and Thematic Enhancement of Trajectories. *TVCG* 21, 1 (2015), 107–121.
[38] Zuchao Wang, Tangzhi Ye, Min Lu, Xiaoru Yuan, Huamin Qu, Jacky Yuan, and Qianliang Wu. 2014. Visual exploration of sparse traffic trajectory data. *TVCG* 20, 12 (2014), 1813–1822.
[39] Jo Wood, Jason Dykes, and Aidan Slingsby. 2010. Visualisation of origins, destinations and flows with OD maps. *The Cartographic Journal* 47, 2 (2010), 117–129.
[40] Xiping Yang, Zhiyuan Zhao, and Shiwei Lu. 2016. Exploring spatial-temporal patterns of urban human mobility hotspots. *Sustainability* 8, 7 (2016), 674.
[41] Jia Yu and Mohamed Sarwat. 2020. Turbocharging Geospatial Visualization Dashboards via a Materialized Sampling Cube Approach. In *ICDE*. 1165–1176.
[42] Wei Zeng, Chi-Wing Fu, Stefan Müller Arisona, Alexander Erath, and Huamin Qu. 2014. Visualizing mobility of public transportation system. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 1833–1842.
[43] Wei Zeng, Qiaomu Shen, Yuzhe Jiang, and Alexandru Telea. 2019. Route-Aware Edge Bundling for Visualizing Origin-Destination Trails in Urban Traffic. In *Computer Graphics Forum*, Vol. 38. 581–593.
[44] Dongxiang Zhang, Mengting Ding, Dingyu Yang, Yi Liu, Ju Fan, and Heng Tao Shen. 2018. Trajectory simplification: an experimental study and quality analysis. *VLDB* 11, 9 (2018), 934–946.
[45] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative location and activity recommendations with GPS history data. In *WWW*. 1029–1038.
[46] Yu Zheng and Xing Xie. 2011. Learning travel recommendations from user-generated GPS traces. *TIST* 2, 1 (2011), 1–29.