PREFIX: A Theoretically Improved Solution for Spatial-Temporal Kernel Density Visualization (Technical Report)

Tsz Nam Chan
Hong Kong Baptist University
edisonchan@comp.hkbu.edu.hk

Pak Lon Ip
University of Macau
paklonip@um.edu.mo

Leong Hou U
University of Macau
ryanlhu@um.edu.mo

Jianliang Xu
Hong Kong Baptist University
xujl@comp.hkbu.edu.hk

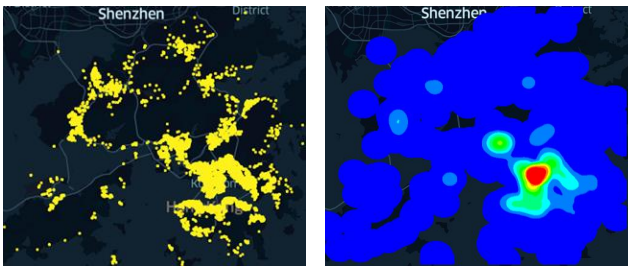
ABSTRACT

Spatial-temporal kernel density visualization (STKDV) has been extensively used for many geospatial analysis tasks, including traffic accident hotspot detection, crime hotspot detection, and disease outbreak detection. However, STKDV is a computationally expensive operation, which is not scalable to large-scale datasets, high resolution sizes, and a large number of timestamps. Although a recent approach, called sliding-window-based solution (SWS), can successfully reduce the time complexity for computing STKDV, it is (i) unable to reduce the time complexity for supporting STKDV-based exploratory analysis, (ii) not yet theoretically efficient, and (iii) incapable of providing optimization techniques for bandwidth tuning. To tackle these drawbacks, we propose a prefix-set-based solution (PREFIX), which consists of three methods, namely $\text{PREFIX}_{\text{single}}$ (for handling (i)), $\text{PREFIX}_{\text{multiple}}$ (for handling (ii)), and $\text{PREFIX}_{\text{tuning}}$ (for handling (iii)). Our theoretical and experimental studies show that PREFIX achieves superior results compared with the state-of-the-art solution (SWS). In particular, PREFIX achieves at least 115x to 1,906x speedup and is the first solution that can efficiently generate multiple high-resolution STKDVs for the large-scale New York taxi dataset with 13.6 million data points.

1 INTRODUCTION

Heatmap (or hotspot map) is an important tool for domain experts to visualize and analyze geospatial data. Among most of the heatmap tools, kernel density visualization (KDV) [15, 17] is the most classic one, which has been widely used by many communities. For example, transportation experts [13, 18, 29, 32, 57] and criminologists [26, 28, 30, 38, 62, 67] adopt KDV to discover the traffic accident and crime hotspots, respectively, in different geographical regions. Epidemiologists [19, 27, 61] adopt KDV to perform the disease outbreak analysis. Figure 1 shows an example to use KDV for discovering the disease outbreak in Hong Kong. Note that there is one COVID-19 hotspot from 1st February 2020 to 1st February 2022. Due to the popularity of KDV, many commonly used scientific software packages, e.g., Scikit-learn [39] and Scipy [54], geographical software packages, e.g., QGIS [7] and ArcGIS [1], and visualization software packages, e.g., Deck.gl [2] and Seaborn [10], can also support this tool.

However, since KDV ignores the occurrence time of each location data point (or event) for generating the hotspot map, it cannot show how the hotspots change with respect to different timestamps. As pointed out by domain experts [19, 28, 32], many geographical



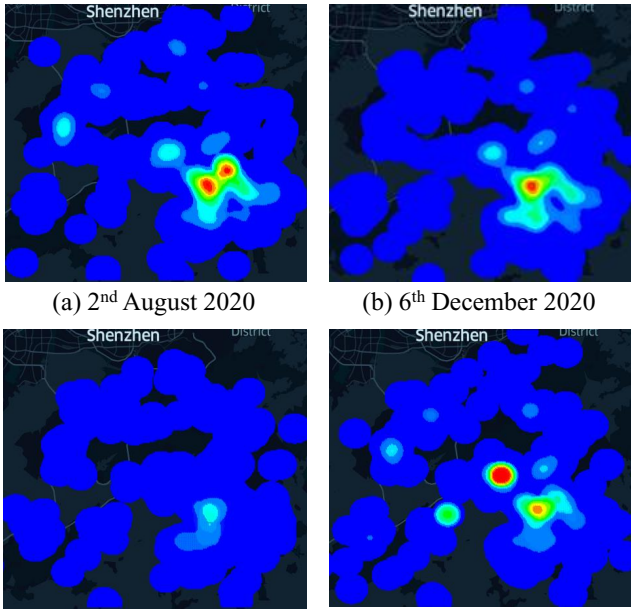
(a) Hong Kong COVID-19 cases (b) Hotspot map

Figure 1: A hotspot map (based on KDV) for the Hong Kong COVID-19 cases (from 1st February 2020 to 1st February 2022), where the region with red color is the hotspot.

events (e.g., traffic accident, crime, and disease outbreak) are time-dependent. Using COVID-19 cases in Hong Kong as an example (cf. Figure 1a), we can see from Figure 2 that the hotspot distributions significantly change over time. Therefore, simply using KDV (cf. Figure 1b) may not reveal the real situation of COVID-19 outbreak. To tackle this issue, geographical researchers [13, 19, 27–30, 32] recently adopt a more advanced and accurate visualization technique, called spatial-temporal kernel density visualization (STKDV), which can generate time-dependent hotspot maps (cf. Figure 2).

Nevertheless, generating STKDV is very time-consuming. Given a resolution size $X \times Y$, T timestamps, and a set of location data points with size n, a naïve implementation of STKDV takes $O(XYTn)$ time, which is not scalable to high resolution sizes, large-scale datasets, and a large number of timestamps. Worse still, domain experts [19, 22, 46, 61] need to generate multiple STKDVs by tuning the bandwidth parameters of spatial and temporal kernels (which will be discussed in Section 2) in order to obtain the time-dependent hotspot maps with the best visual quality, which further exacerbates the inefficiency issue of STKDV. In fact, many domain experts [19, 27, 46] have complained about the inefficiency issue of using STKDV.

To respond to this issue, Chan et al. [16] propose the first solution, called sliding-window-based solution (SWS), that can successfully reduce the worst-case time complexity for generating STKDV from $O(XYTn)$ to $O(XY(T+n))$. Despite this, the time complexity still depends on XYn, which is not yet efficient enough to support high-resolution STKDV for a large-scale dataset. For example, the SWS method cannot generate STKDV with 480×320 resolution for datasets with more than 500,000 data points within one hour (cf.



(a) 2nd August 2020 (b) 6th December 2020

(c) 28th February 2021 (d) 28th January 2022

Figure 2: Using STKDV to show the hotspot maps of Hong Kong COVID-19 cases in different timestamps. The COVID-19 outbreak is more serious on 2nd August 2020, 6th December 2020, and 28th January 2022.

Figure 12 of [16]). In addition, this method requires all timestamps of STKDV to be known in advance. However, domain experts can perform exploratory analysis [33–35], where the timestamps are provided on the fly. Furthermore, Chan et al. [16] do not provide efficient algorithms to generate multiple STKDVs with respect to different bandwidth parameters.

In order to further improve the efficiency of STKDV-based data analytics, we ask the following two research questions in this paper.

(1) *Can we further reduce the time complexity for generating STKDV, without increasing the space complexity, under the following two settings?*
   (a) Every timestamp is given on the fly (i.e., $T = 1$).
   (b) The T timestamps are known in advance.
(2) *Can we reduce the time complexity for generating multiple STKDVs with different spatial and temporal bandwidths, without increasing the space complexity?*

In this paper, we answer the above questions by proposing a prefix-set-based solution (PREFIX), which consists of three methods. To tackle the first question, we develop $\text{PREFIX}_{\text{single}}$ and $\text{PREFIX}_{\text{multiple}}$, which significantly reduce the time complexity for generating STKDV with an on-the-fly timestamp (i.e., $T = 1$) and T known timestamps, respectively. To tackle the second question, we develop $\text{PREFIX}_{\text{tuning}}$, which further reduces the time complexity for generating STKDVs with multiple spatial and temporal bandwidths. As a remark, all our methods retain the same space complexity for solving these problems. Table 1 summarizes the theoretical results of our solution, PREFIX, and the state-of-the-art solution, SWS [16], where $X \times Y$, T, n, M, and N denote the resolution size, the number of timestamps, the dataset size, the number of spatial bandwidths, and the number of temporal bandwidths, respectively.

Table 1: Theoretical results for generating STKDVs.

Problem	Method	Time complexity	Space complexity
STKDV with an on-the-fly timestamp	SWS [16]	$O(XYn)$	$O(XY + n)$
	$\text{PREFIX}_{\text{single}}$ (Section 3.2)	$O(Y(X + n))$ (Theorem 1)	$O(XY + n)$ (Theorem 4)
STKDV with T known timestamps	SWS [16]	$O(XY(T + n))$	$O(XYT + n)$
	$\text{PREFIX}_{\text{multiple}}$ (Section 3.3)	$O(XYT + Yn)$ (Theorem 2)	$O(XYT + n)$ (Theorem 5)
Bandwidth tuning	SWS [16]	$O(MNXY(T + n))$	$O(MNXYT + n)$
	$\text{PREFIX}_{\text{tuning}}$ (Section 3.4)	$O(M(XYTN + Yn))$ (Theorem 3)	$O(MNXYT + n)$ (Theorem 6)

Our experiments on five large-scale location datasets (with at most 5 million data points) show that PREFIX can achieve 115x to 1,906x speedup compared with the state-of-the-art solution, SWS, for all problem settings. In addition, we further conduct a case study to generate high-resolution ($1{,}280 \times 960$) STKDVs for the large-scale New York taxi dataset [5] (with 13.6 million data points), which, to the best of our knowledge, is the first work that can achieve this scalability.

The rest of the paper is organized as follows. We first formally define the problems and overview the state-of-the-art solution, SWS, in Section 2. Then, we illustrate our solution, PREFIX, in Section 3. Next, we discuss the experimental results in Section 4. After that, we review the related work in Section 5. Lastly, we conclude our paper in Section 6. The appendix can be found in Section 7.

2 PRELIMINARIES

In this section, we first formally define two problems, namely STKDV (with two settings) and bandwidth tuning, in Section 2.1. Then, we overview the state-of-the-art method, namely sliding-window-based solution (SWS) [16], in Section 2.2.

2.1 Problem Definitions

To generate STKDV, i.e., the time-dependent hotspot maps (cf. Figure 2), with the resolution size $X \times Y$ for the set of location data points (cf. Figure 1a), domain experts [19, 28, 32] need to determine the color of each pixel for each timestamp based on the spatial-temporal kernel density function, which is formally stated in Problem 1.

Problem 1. *(STKDV) Given a set of spatial-temporal data points $\widehat{P} = \{(\mathbf{p}_1, t_{\mathbf{p}_1}), (\mathbf{p}_2, t_{\mathbf{p}_2}), ..., (\mathbf{p}_n, t_{\mathbf{p}_n})\}$ with size n, a resolution size $X \times Y$, T timestamps $t_1, t_2, ..., t_T$, the bandwidth of the spatial kernel b_σ, and the bandwidth of the temporal kernel b_τ, we need to compute the spatial-temporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ for each pixel $\mathbf{q}$ and timestamp t_i (cf. Equation 1).*

$$\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i) = \sum_{(\mathbf{p}, t_{\mathbf{p}}) \in \widehat{P}} w \cdot K^{(b_\sigma)}_{space}(\mathbf{q}, \mathbf{p}) \cdot K^{(b_\tau)}_{time}(t_i, t_{\mathbf{p}}) \quad (1)$$

where w, $K^{(b_\sigma)}_{space}(\mathbf{q}, \mathbf{p})$, and $K^{(b_\tau)}_{time}(t_i, t_{\mathbf{p}})$ denote the normalization constant, spatial kernel function, and temporal kernel function, respectively. Some of the representative kernel functions are listed in Table 2. In this problem, we focus on two settings: (1) generate STKDV with each on-the-fly timestamp (i.e., $T = 1$) and (2) generate STKDV with T timestamps that are known in advance.

Table 2: Some representative spatial and temporal kernel functions, where $dist$, b_σ, and b_τ denote the Euclidean distance, the bandwidth of the spatial kernel, and the bandwidth of the temporal kernel, respectively.

Kernel	$K_{\text{space}}^{(b_\sigma)}(\mathbf{q}, \mathbf{p})$	$K_{\text{time}}^{(b_\tau)}(t_i, t_\mathbf{p})$	References
Uniform	$\begin{cases} \frac{1}{b_\sigma} & \text{if } dist(\mathbf{q}, \mathbf{p}) \le b_\sigma \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} \frac{1}{b_\tau} & \text{if } dist(t_i, t_\mathbf{p}) \le b_\tau \\ 0 & \text{otherwise} \end{cases}$	[58]
Epanechnikov	$\begin{cases} 1 - \frac{1}{b_\sigma^2} dist(\mathbf{q}, \mathbf{p})^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \le b_\sigma \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 - \frac{1}{b_\tau^2} dist(t_i, t_\mathbf{p})^2 & \text{if } dist(t_i, t_\mathbf{p}) \le b_\tau \\ 0 & \text{otherwise} \end{cases}$	[19, 29, 56, 61]
Quartic	$\begin{cases} \left(1 - \frac{1}{b_\sigma^2} dist(\mathbf{q}, \mathbf{p})^2\right)^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \le b_\sigma \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} \left(1 - \frac{1}{b_\tau^2} dist(t_i, t_\mathbf{p})^2\right)^2 & \text{if } dist(t_i, t_\mathbf{p}) \le b_\tau \\ 0 & \text{otherwise} \end{cases}$	[14, 30]

Based on the definition of Problem 1, we further define the bandwidth-tuning problem for STKDV (cf. Problem 2).

Problem 2. *(Bandwidth tuning) Given M bandwidths of the spatial kernel, i.e., $b_{\sigma_1}, b_{\sigma_2}, ..., b_{\sigma_M}$, and N bandwidths of the temporal kernel, i.e., $b_{\tau_1}, b_{\tau_2}, ..., b_{\tau_N}$, we need to compute STKDV (cf. Problem 1) for each pair of spatial and temporal bandwidths, i.e., $(b_{\sigma_u}, b_{\tau_v})$, where $1 \le u \le M$ and $1 \le v \le N$.*

As a remark, since the Epanechnikov kernel is more popular compared with other kernel functions, we adopt this function in both the spatial kernel $K_{\text{space}}^{(b_\sigma)}(\mathbf{q}, \mathbf{p})$ and the temporal kernel $K_{\text{time}}^{(b_\tau)}(t_i, t_\mathbf{p})$ in this paper. Nevertheless, all our methods can be extended to other kernel functions in Table 2.

2.2 Overview of SWS: the State-of-the-art Method

Recently, Chan et al. [16] propose the sliding-window-based solution (SWS), which can reduce the time complexity for computing STKDV with T timestamps. Here, we first discuss two core ideas of SWS, which are (1) establishment of the sliding window in the time-axis and (2) fast sliding-window-based incremental computation. Then, we summarize the weakness of this method.

Establishment of the sliding window in the time-axis: Consider the temporal kernel $K_{\text{time}}^{(b_\tau)}(t_i, t_\mathbf{p})$ in Table 2. Note that only those data points $(\mathbf{p}, t_\mathbf{p})$ with $dist(t_i, t_\mathbf{p}) \le b_\tau$ can contribute to the kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ (cf. Equation 1). Therefore, they define the sliding window $W(t_i)$ that only includes the data points with $dist(t_i, t_\mathbf{p}) \le b_\tau$ (cf. Equation 2), as shown in Figure 3.

$$W(t_i) = \{(\mathbf{p}, t_\mathbf{p}) \in \widehat{P} : dist(t_i, t_\mathbf{p}) \le b_\tau\} \tag{2}$$

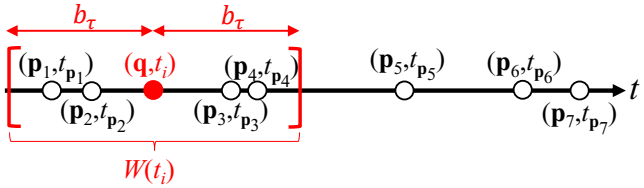


Figure 3: The red sliding window $W(t_i)$ for the pixel $\mathbf{q}$ with the timestamp t_i. $(\mathbf{p}_1, t_{\mathbf{p}_1})$, $(\mathbf{p}_2, t_{\mathbf{p}_2})$, $(\mathbf{p}_3, t_{\mathbf{p}_3})$, and $(\mathbf{p}_4, t_{\mathbf{p}_4})$ are the only data points that can contribute to the kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$.

With the definition of $W(t_i)$, they can represent the spatial-temporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ (cf. Equation 1 with Epanechnikov kernel) as follows.

$$\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i) = \sum_{(\mathbf{p}, t_\mathbf{p}) \in W(t_i)} w \cdot K_{\text{space}}^{(b_\sigma)}(\mathbf{q}, \mathbf{p}) \cdot \left(1 - \frac{1}{b_\tau^2} dist(t_i, t_\mathbf{p})^2\right)$$

Based on the simple mathematical derivations, they have:

$$\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i) = w\left(1 - \frac{t_i^2}{b_\tau^2}\right) \cdot S_{W(t_i)}^{(0)}(\mathbf{q}) + \frac{2wt_i}{b_\tau^2} \cdot S_{W(t_i)}^{(1)}(\mathbf{q}) - \frac{w}{b_\tau^2} \cdot S_{W(t_i)}^{(2)}(\mathbf{q}) \tag{3}$$

where:

$$S_{W(t_i)}^{(u)}(\mathbf{q}) = \sum_{(\mathbf{p}, t_\mathbf{p}) \in W(t_i)} t_\mathbf{p}^u \cdot K_{\text{space}}^{(b_\sigma)}(\mathbf{q}, \mathbf{p}) \tag{4}$$

Therefore, once they can efficiently maintain the statistical terms $S_{W(t_i)}^{(u)}(\mathbf{q})$, where $0 \le u \le 2$, in the sliding window, they can compute the density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ in $O(1)$ time using Equation 3.

Fast sliding-window-based incremental computation: In Figure 4, we can see that the window $W(t_i)$ for the timestamp t_i can be efficiently updated to the consecutive window $W(t_{i+1})$ for the timestamp t_{i+1}, by deleting the yellow points, called deleted set $D(W(t_i), W(t_{i+1}))$, and inserting the green point, called inserted set $I(W(t_i), W(t_{i+1}))$. As such, they have:

$$W(t_{i+1}) = W(t_i) \cup I(W(t_i), W(t_{i+1})) \backslash D(W(t_i), W(t_{i+1})) \tag{5}$$

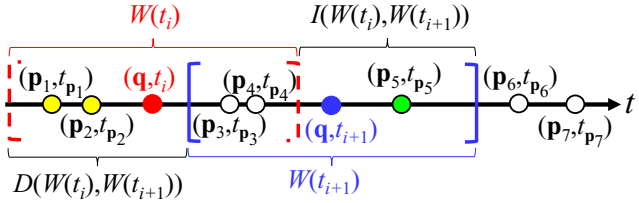


Figure 4: After they change the timestamp from t_i to t_{i+1}, they can update from the red dashed sliding window $W(t_i)$ to the blue sliding window $W(t_{i+1})$, by deleting the yellow points $(\mathbf{p}_1, t_{\mathbf{p}_1})$ and $(\mathbf{p}_2, t_{\mathbf{p}_2})$ and inserting the green point $(\mathbf{p}_5, t_{\mathbf{p}_5})$.

Based on Equation 5, they conclude that updating the statistical terms from $W(t_i)$ to $W(t_{i+1})$ (cf. Equation 6) and computing $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_{i+1})$ (cf. Equation 3) take $O(|I(W(t_i), W(t_{i+1}))| + |D(W(t_i), W(t_{i+1}))|)$ time.

$$S_{W(t_{i+1})}^{(u)}(\mathbf{q}) = S_{W(t_i)}^{(u)}(\mathbf{q}) - \sum_{(\mathbf{p}, t_\mathbf{p}) \in D(W(t_i), W(t_{i+1}))} t_\mathbf{p}^u \cdot K_{\text{space}}^{(b_\sigma)}(\mathbf{q}, \mathbf{p}) + \sum_{(\mathbf{p}, t_\mathbf{p}) \in I(W(t_i), W(t_{i+1}))} t_\mathbf{p}^u \cdot K_{\text{space}}^{(b_\sigma)}(\mathbf{q}, \mathbf{p}) \tag{6}$$

Thus, they first compute the density value of the pixel $\mathbf{q}$ for the first timestamp t_1 along with the statistical terms $S^{(u)}_{W(t_1)}(\mathbf{q})$ ($u = 0, 1, 2$) of the first window $W(t_1)$. Then, they incrementally update to the next window/timestamp (cf. Figure 5) and compute its statistical terms and density value. As such, the time complexity of this approach is:

$$O\Big(|W(t_1)| + \sum_{i=1}^{T-1} |I(W(t_i), W(t_{i+1}))| + \sum_{i=1}^{T-1} |D(W(t_i), W(t_{i+1}))| + T\Big) \tag{7}$$

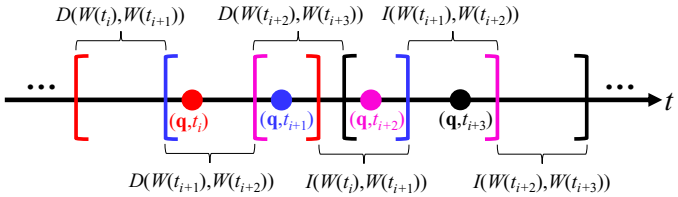


Figure 5: Multiple sliding windows for different timestamps.

In Figure 5, note that the deleted (or inserted) sets of different timestamps do not share the same data points. Based on this reason, they have:

$$\sum_{i=1}^{T-1} |I(W(t_i), W(t_{i+1}))| \leq n \text{ and } \sum_{i=1}^{T-1} |D(W(t_i), W(t_{i+1}))| \leq n \tag{8}$$

Furthermore, $|W(t_1)| \rightarrow n$ in the worst case. As such, the time complexity for computing the density values of a pixel $\mathbf{q}$ with T timestamps $t_1, t_2, ..., t_T$ (cf. Equation 7) is $O(T + n)$. Since there are $X \times Y$ pixels, the time complexity for computing STKDV (cf. Problem 1) is $O(XY(T + n))$. With M spatial bandwidths and N temporal bandwidths, this method also takes $O(MNXY(T + n))$ time to support bandwidth tuning for STKDV (cf. Problem 2).

Weakness of SWS. Although the SWS method can successfully reduce the worst-case time complexity for computing STKDV, the time complexity still depends on XYn. Therefore, this method cannot be scalable to the large-resolution size $X \times Y$ and the large number of data points n. Furthermore, the SWS method adopts the sharing approach for computing the statistical terms between two consecutive sliding windows (which corresponds to two consecutive timestamps) in order to efficiently evaluate the kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ with T timestamps (cf. Figure 5). Therefore, suppose that domain experts provide each timestamp on-the-fly (i.e., $T = 1$) for conducting the exploratory analysis, this approach cannot be utilized to improve the efficiency for computing STKDV. In addition, the SWS method does not offer the optimization techniques for handling multiple bandwidths.

3 OUR SOLUTION: PREFIX

To overcome the weakness of the state-of-the-art solution, we propose the new solution, called PREFIX, in this section. First, we summarize the core ideas of PREFIX in Section 3.1. Based on these ideas, we then discuss the methods, $\text{PREFIX}_{\text{single}}$, $\text{PREFIX}_{\text{multiple}}$, and $\text{PREFIX}_{\text{tuning}}$, in Section 3.2, Section 3.3, and Section 3.4, respectively, which further reduce the time complexity for generating STKDV (cf. Problem 1 with two settings) and supporting bandwidth tuning (cf. Problem 2). Lastly, we discuss the space complexity of our PREFIX solution in Section 3.5.

3.1 Core ideas of PREFIX

Here, we first extend the concept of sliding window $W(t_i)$ (cf. Equation 2) to the general one $W_{\mathcal{I}}$ (cf. Equation 9), where $\mathcal{I}$ denotes any time interval in the time-axis.

$$W_{\mathcal{I}} = \{(\mathbf{p}, t_{\mathbf{p}}) \in \widehat{P} : t_{\mathbf{p}} \in \mathcal{I}\} \tag{9}$$

Then, we further define the statistical terms for the window $W_{\mathcal{I}}$ in Equation 10 ($u = 0, 1, 2$ for the Epanechnikov function as the temporal kernel).

$$S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q}) = \sum_{(\mathbf{p}, t_{\mathbf{p}}) \in W_{\mathcal{I}}} t^u_{\mathbf{p}} \cdot K^{(b_\sigma)}_{\text{space}}(\mathbf{q}, \mathbf{p}) \tag{10}$$

Based on these concepts, we illustrate three core ideas of our solution, PREFIX, namely (1) an $O(Y(X + |W_{\mathcal{I}}|))$-algorithm for computing $S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q})$ with all pixels $\mathbf{q}$, (2) prefix set, and (3) computation of the statistical terms for multiple prefix sets.

Core idea 1 (An $O(Y(X + |W_{\mathcal{I}}|))$-algorithm for computing $S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q})$ with all pixels $\mathbf{q}$): In Figure 6, we can see that all data points in $W_{\mathcal{I}}$ (cf. Equation 9) do not depend on spatial positions of the pixels $\mathbf{q}$. Therefore, $W_{\mathcal{I}}$ can be shared to all pixels $\mathbf{q}$.

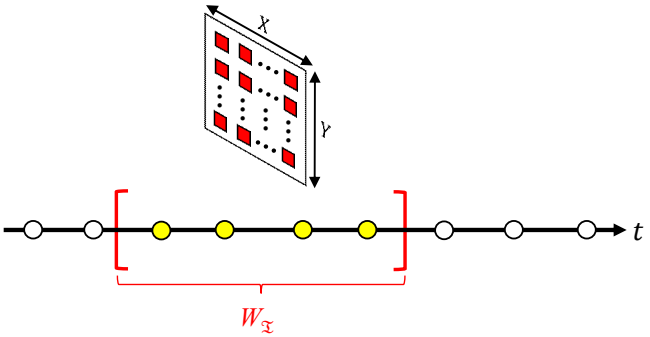


Figure 6: All data points in the sliding window $W_{\mathcal{I}}$ (yellow circles) can be used to compute the statistical terms $S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q})$ for all pixels $\mathbf{q}$ (red rectangles).

Based on the above observation, we aim to efficiently evaluate the statistical terms $S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q})$ for all pixels $\mathbf{q}$. Recently, Chan et al. [17] propose the bucket-based algorithm, $\text{SLAM}_{\text{BUCKET}}$, for solving the kernel density visualization (KDV) problem, which is closely related to our STKDV problem. Inspired by this work, we extend the $\text{SLAM}_{\text{BUCKET}}$ algorithm for computing $S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q})$ with all pixels $\mathbf{q}$. Lemma 1 shows that this approach only takes $O(Y(X + |W_{\mathcal{I}}|))$ time, using the spatial kernel functions in Table 2. Due to space limitations, we only provide the proof sketch for this lemma. The formal proof can be found in Section 7.

LEMMA 1. *Given a sliding window $W_{\mathcal{I}}$ (cf. Equation 9) with any time interval $\mathcal{I}$ in the time-axis and a resolution size $X \times Y$, the statistical terms $S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q})$ (cf. Equation 10) for all pixels $\mathbf{q}$ can be computed in $O(Y(X + |W_{\mathcal{I}}|))$ time, using the spatial kernel functions in Table 2.*

Proof. (Sketch) Chan et al. [17] show that the bucket-based algorithm can use $O(Y(X + |P|))$ time (cf. Theorem 2 in [17]) to generate a single KDV for the location dataset P with the resolution size $X \times Y$, where the color of each pixel $\mathbf{q}$ is based on the kernel density function $\mathcal{F}_P(\mathbf{q})$ (cf. Equation 11).

$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p} \in P} w \cdot K^{(b_\sigma)}_{\text{space}}(\mathbf{q}, \mathbf{p}) \tag{11}$$

Since the statistical terms $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ (cf. Equation 10) are close to $\mathcal{F}_P(\mathbf{q})$ (by replacing the positive constant w with $t^u_\mathbf{p}$), we can simply modify the bucket-based algorithm to compute $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ in $O(Y(X + |W_\mathcal{I}|))$ time. □

Core idea 2 (Prefix set): We first define the prefix set of the timestamp t, which includes the data points $(\mathbf{p}, t_\mathbf{p})$ with $t_\mathbf{p} \le t$ in the dataset $\widehat{P}$, where:

$$\widehat{P}(t) = \{(\mathbf{p}, t_\mathbf{p}) \in \widehat{P} : t_\mathbf{p} \le t\} \tag{12}$$

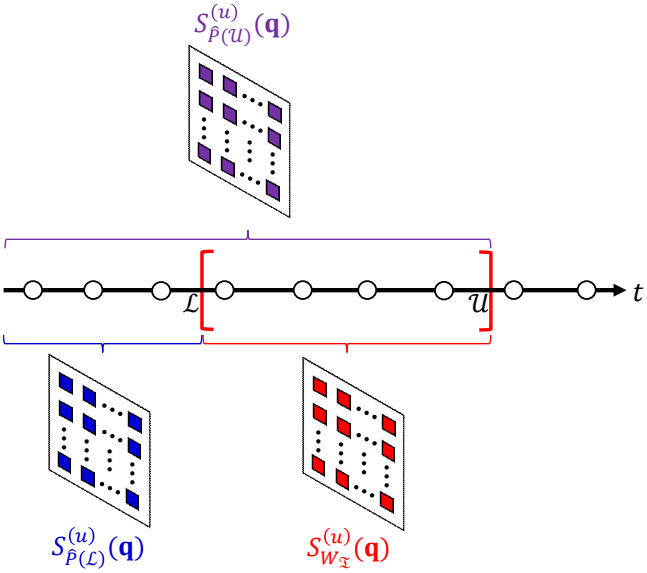


Figure 7: The statistical terms $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ for the window $W_\mathcal{I}$ (i.e., all the red pixels) with the interval $\mathcal{I} = [\mathcal{L}, \mathcal{U}]$ can be computed by subtracting $S^{(u)}_{\widehat{P}(\mathcal{U})}(\mathbf{q})$ (i.e., all the purple pixels) by $S^{(u)}_{\widehat{P}(\mathcal{L})}(\mathbf{q})$ (all the blue pixels).

With this concept, Figure 7 shows that we can then obtain the statistical terms $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ for the window $W_\mathcal{I}$ using Equation 13 if the statistical terms for the prefix sets $\widehat{P}(\mathcal{L})$ and $\widehat{P}(\mathcal{U})$, i.e., $S^{(u)}_{\widehat{P}(\mathcal{L})}(\mathbf{q})$ and $S^{(u)}_{\widehat{P}(\mathcal{U})}(\mathbf{q})$, respectively, are available.

$$S^{(u)}_{W_\mathcal{I}}(\mathbf{q}) = S^{(u)}_{\widehat{P}(\mathcal{U})}(\mathbf{q}) - S^{(u)}_{\widehat{P}(\mathcal{L})}(\mathbf{q}) \tag{13}$$

Since there are $X \times Y$ pixels, we can conclude that the statistical terms $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ for the window $W_\mathcal{I}$ can be computed in $O(XY)$ time once we have $S^{(u)}_{\widehat{P}(\mathcal{L})}(\mathbf{q})$ and $S^{(u)}_{\widehat{P}(\mathcal{U})}(\mathbf{q})$ (cf. Lemma 2).

Lemma 2. *Given the interval $\mathcal{I} = [\mathcal{L}, \mathcal{U}]$ and the statistical terms $S^{(u)}_{\widehat{P}(\mathcal{L})}(\mathbf{q})$ and $S^{(u)}_{\widehat{P}(\mathcal{U})}(\mathbf{q})$, we can compute the statistical terms $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ for all pixels $\mathbf{q}$ with $O(XY)$ time.*

Core idea 3 (Computation of the statistical terms for multiple prefix sets): Suppose that we need to compute the statistical terms for the prefix sets $\widehat{P}(t^{(1)})$, $\widehat{P}(t^{(2)})$,..., $\widehat{P}(t^{(K)})$, which correspond to the arbitrary timestamps $t^{(1)}$, $t^{(2)}$,..., $t^{(K)}$ (cf. Figure 8).

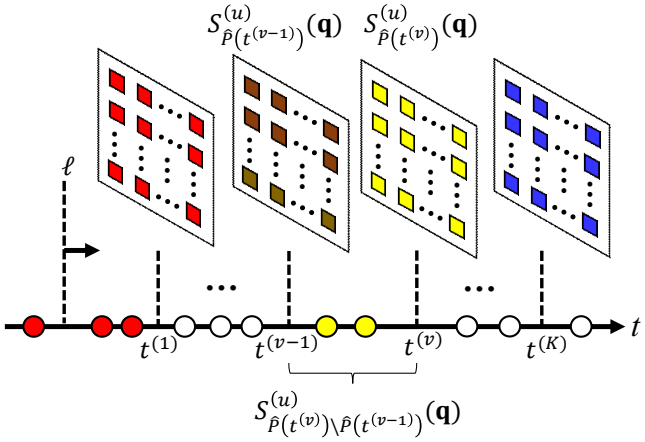


Figure 8: Using the sweep line ℓ to compute the statistical terms $S^{(u)}_{\widehat{P}(t^{(v)})}(\mathbf{q})$ for each prefix set $\widehat{P}(t^{(v)})$, where $1 \le v \le K$.

In Figure 8, note that we can use the sweep line ℓ to compute the statistical terms for each prefix set. Here, we use an inductive approach to show how we can achieve this goal.

Consider the timestamp $t^{(1)}$. This sweep line ℓ needs to scan and store all the red data points $(\mathbf{p}, t_\mathbf{p})$ with $t_\mathbf{p} \le t^{(1)}$. Once this line ℓ intersects with $t^{(1)}$, we can compute the statistical terms $S^{(u)}_{\widehat{P}(t^{(1)})}(\mathbf{q})$ (i.e., the red pixels) with $O(Y(X + |\widehat{P}(t^{(1)})|))$ time (cf. Lemma 1) and remove all these red points from the sweep line ℓ (with $O(|\widehat{P}(t^{(1)})|)$ time).

Consider the timestamp $t^{(v)}$, where $v \ge 2$. Suppose that the statistical terms $S^{(u)}_{\widehat{P}(t^{(v-1)})}(\mathbf{q})$ for all brown pixels are available, the sweep line ℓ needs to scan and store all the yellow points $(\mathbf{p}, t_\mathbf{p})$ with $t^{(v-1)} < t_\mathbf{p} \le t^{(v)}$. Once this sweep line ℓ intersects with $t^{(v)}$, we can compute $S^{(u)}_{\widehat{P}(t^{(v)})\setminus\widehat{P}(t^{(v-1)})}(\mathbf{q})$ for all pixels $\mathbf{q}$ with $O(Y(X + |\widehat{P}(t^{(v)})\setminus\widehat{P}(t^{(v-1)})|))$ time (cf. Lemma 1). Then, we can compute the statistical terms $S^{(u)}_{\widehat{P}(t^{(v)})}(\mathbf{q})$ for all yellow pixels using $O(XY)$ time, based on the following equation.

$$S^{(u)}_{\widehat{P}(t^{(v)})}(\mathbf{q}) = S^{(u)}_{\widehat{P}(t^{(v-1)})}(\mathbf{q}) + S^{(u)}_{\widehat{P}(t^{(v)})\setminus\widehat{P}(t^{(v-1)})}(\mathbf{q}) \tag{14}$$

Lastly, we can remove all the yellow points from the sweep line ℓ (with $O(|\widehat{P}(t^{(v)})\setminus\widehat{P}(t^{(v-1)})|)$ time).

Algorithm 1 shows the detailed pseudocode to compute the statistical terms for multiple prefix sets. Based on the above discussion, the time complexity of this algorithm is:

$$O\Big(Y(X + |\widehat{P}(t^{(1)})|) + \sum_{v=2}^{K} Y(X + |\widehat{P}(t^{(v)})\setminus\widehat{P}(t^{(v-1)})|)\Big)$$

Algorithm 1 Algorithm for Computing Statistical terms of K Prefix Sets (PS_{STAT})

procedure $\text{PS}_{\text{STAT}}(\widehat{P} = \{(\mathbf{p}_1, t_{\mathbf{p}_1}), (\mathbf{p}_2, t_{\mathbf{p}_2}), ..., (\mathbf{p}_n, t_{\mathbf{p}_n})\}$, timestamps $t^{(1)}, t^{(2)}, ..., t^{(K)}$)
  Initialize the sweep line ℓ, where
    $\ell.t \leftarrow -\infty$ and $\ell.\text{set} \leftarrow \phi$
  while $\ell.t < t^{(K)}$ **do**
    ℓ sweeps the next element e
    if e is $(\mathbf{p}_i, t_{\mathbf{p}_i})$ (where $1 \le i \le n$) **then**
      $\ell.t \leftarrow t_{\mathbf{p}_i}$
      $\ell.\text{set} \leftarrow \ell.\text{set} \cup \{(\mathbf{p}_i, t_{\mathbf{p}_i})\}$
    if e is $t^{(v)}$ (where $1 \le v \le K$) **then**
      $\ell.t \leftarrow t^{(v)}$
      //Consider all pixels $\mathbf{q}$
      if $v = 1$ **then**
        $S^{(u)}_{\widehat{P}(t^{(v)})}(\mathbf{q}) \leftarrow S^{(u)}_{\ell.\text{set}}(\mathbf{q})$ ▷ Lemma 1
      else
        $S^{(u)}_{\widehat{P}(t^{(v)})}(\mathbf{q}) \leftarrow S^{(u)}_{\widehat{P}(t^{(v-1)})}(\mathbf{q}) + S^{(u)}_{\ell.\text{set}}(\mathbf{q})$ ▷ Eq. 14
      $\ell.\text{set} \leftarrow \phi$
    Return $S^{(u)}_{\widehat{P}(t^{(v)})}(\mathbf{q})$ (where $1 \le v \le K$)

Since the sets $\widehat{P}(t^{(v)}) \backslash \widehat{P}(t^{(v-1)})$ with different v do not intersect with each other (cf. Figure 8), we have $\sum_{v=2}^{K} |\widehat{P}(t^{(v)}) \backslash \widehat{P}(t^{(v-1)})| \le n$. In addition, $|\widehat{P}(t^{(1)})| \to n$ in the worst case. As such, we can conclude that finding the statistical terms for the prefix sets with respect to K arbitrary timestamps (based on Algorithm 1) is $O(XYK + Yn)$ time (cf. Lemma 3).

Lemma 3. *Given a set of spatial-temporal data points $\widehat{P}$ with size n and K arbitrary timestamps, $t^{(1)}, t^{(2)}, ..., t^{(K)}$, Algorithm 1 takes $O(XYK + Yn)$ time to find the statistical terms for the prefix sets with respect to these K timestamps.*

3.2 PREFIX for Generating STKDV with a Single Timestamp ($\text{PREFIX}_{\text{single}}$)

Recall from the core idea 1 (cf. Section 3.1) that the statistical terms $S^{(u)}_{W_{\mathcal{I}}}(\mathbf{q})$ for all pixels $\mathbf{q}$ can be computed in $O(Y(X + |W_{\mathcal{I}}|))$ time (cf. Lemma 1) no matter which interval $\mathcal{I}$ we choose. Therefore, once we set $\mathcal{I} = [t_i - b_\tau, t_i + b_\tau]$, i.e., $W_{\mathcal{I}} = W(t_i)$ (cf. Equation 2 and Figure 3), we conclude that $S^{(u)}_{W(t_i)}(\mathbf{q})$ for all pixels $\mathbf{q}$ can be computed in $O(Y(X + n))$ time (as $|W(t_i)| \to n$ in the worst case). With these statistical terms $S^{(u)}_{W(t_i)}(\mathbf{q})$, we can then compute the spatiotemporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ for all pixels $\mathbf{q}$ with a single timestamp t_i in $O(XY)$ time (by adopting simple matrix arithmetic operations in Equation 3). Theorem 1 summarizes the theoretical result of this method (named as $\text{PREFIX}_{\text{single}}$).

Theorem 1. *PREFIX_{single} can generate STKDV with a single timestamp (i.e., solving Problem 1 with $T = 1$) using $O(Y(X + n))$ time.*

Compared with the state-of-the-art SWS method (cf. Section 2.2), which takes $O(XYn)$ time, our $\text{PREFIX}_{\text{single}}$ method can further reduce the time complexity to $O(Y(X + n))$ time for generating STKDV with a single timestamp.

3.3 PREFIX for Generating STKDV with Multiple Timestamps ($\text{PREFIX}_{\text{multiple}}$)

In this section, we proceed to investigate how the core ideas of PREFIX can be adopted to generate STKDV with T timestamps. With the core idea 3 (cf. Section 3.1), we can first adopt the sweep line ℓ to maintain all statistical terms $S^{(u)}_{\widehat{P}(t_i - b_\tau)}(\mathbf{q})$ and $S^{(u)}_{\widehat{P}(t_i + b_\tau)}(\mathbf{q})$ for the endpoints of the window $W(t_i)$ with every timestamp t_i (cf. the pixel-planes on the lower time-axis in Figure 9). Since there are T windows and each window contains two endpoints, we need to maintain $O(T)$ statistical terms for the prefix sets, which takes $O(XYT + Yn)$ time (by setting $K = O(T)$ in Lemma 3). Based on the core idea 2 in Section 3.1, we can then obtain the statistical terms $S^{(u)}_{W(t_i)}(\mathbf{q})$ for each window $W(t_i)$ (cf. the pixel-planes on the upper time-axis in Figure 9). Since there are T timestamps (i.e., T windows) and computing $S^{(u)}_{W(t_i)}(\mathbf{q})$ for each $W(t_i)$ takes $O(XY)$ time (cf. Lemma 2), the time complexity of finding $S^{(u)}_{W(t_i)}(\mathbf{q})$ for all timestamps is $O(XYT)$. By adopting Equation 3, we can compute the spatiotemporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ for all pixels $\mathbf{q}$ with T timestamps in $O(XYT)$ time. Based on the above discussion, we can conclude that the time complexity of this method, named as $\text{PREFIX}_{\text{multiple}}$, for generating STKDV with T timestamps is $O(XYT + Yn)$ (cf. Theorem 2).

Theorem 2. *$\text{PREFIX}_{multiple}$ can generate STKDV (cf. Problem 1) with T timestamps in $O(XYT + Yn)$ time.*

Recall that the state-of-the-art method, SWS, takes $O(XY(T + n))$ time for generating STKDV (cf. Section 2.2). $\text{PREFIX}_{\text{multiple}}$ further improves the theoretical result to $O(XYT + Yn)$ time.

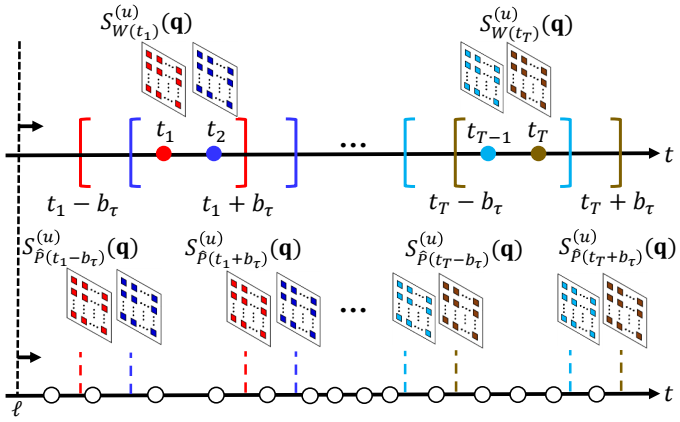


Figure 9: Maintain the statistical terms of the prefix sets, i.e., $S^{(u)}_{\widehat{P}(t_i - b\tau)}(\mathbf{q})$ and $S^{(u)}_{\widehat{P}(t_i + b\tau)}(\mathbf{q})$, for two endpoints of the sliding window $W(t_i)$ with each timestamp t_i (e.g., $S^{(u)}_{\widehat{P}(t_1 - b\tau)}(\mathbf{q})$, $S^{(u)}_{\widehat{P}(t_1 + b\tau)}(\mathbf{q})$, $S^{(u)}_{\widehat{P}(t_T - b\tau)}(\mathbf{q})$, and $S^{(u)}_{\widehat{P}(t_T + b\tau)}(\mathbf{q})$) in order to evaluate the statistical terms $S^{(u)}_{W(t_i)}(\mathbf{q})$ (e.g., $S^{(u)}_{W(t_1)}(\mathbf{q})$ and $S^{(u)}_{W(t_T)}(\mathbf{q})$) for all pixels $\mathbf{q}$.

3.4 PREFIX for Bandwidth Tuning (PREFIX$_{\text{tuning}}$)

Here, we discuss how to solve the bandwidth tuning problem (cf. Problem 2) using the core ideas in Section 3.1. We fix a spatial bandwidth b_σ and generate STKDVs with N temporal bandwidths, i.e., computing $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ for each timestamp t_i with N sliding windows (temporal bandwidths) for all pixels $\mathbf{q}$ (cf. Figure 10).

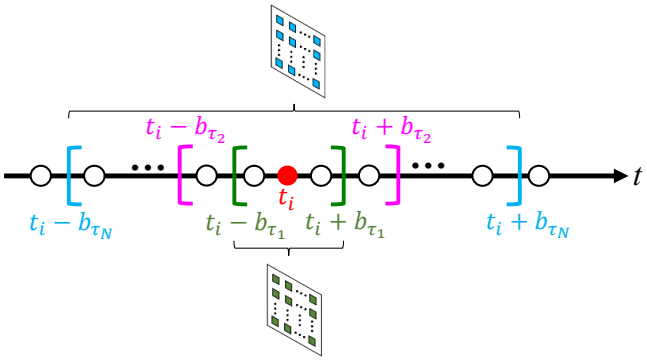


Figure 10: Compute the density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ for all pixels $\mathbf{q}$ with N temporal bandwidths, $b_{\tau_1}, b_{\tau_2}, ..., b_{\tau_N}$. Green and blue pixels denote the density values for the temporal bandwidths b_{τ_1} and b_{τ_N}, respectively.

Since each STKDV contains T timestamps, there are $2TN$ endpoints in total (cf. Figure 11). Based on the core idea 3, finding the statistical terms of all prefix sets takes $O(XYTN + Yn)$ time (by setting $K = 2TN$ in Lemma 3). With these prefix sets, we can adopt the core idea 2 to obtain the statistical terms for each sliding window with $O(XY)$ time (cf. Lemma 2) and compute the corresponding density values $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ (cf. Equation 3) for all pixels $\mathbf{q}$ (with $O(XY)$ time). Since there are TN sliding windows in Figure 11, computing all density values takes $O(XYTN)$ time. Therefore, given a fixed spatial bandwidth, we can conclude that the time complexity for computing STKDVs with N temporal bandwidths is $O(XYTN + Yn)$ time (cf. Lemma 4).

Lemma 4. *Given a fixed spatial bandwidth, i.e., b_σ, and N temporal bandwidths, i.e., $b_{\tau_1}, b_{\tau_2}, ..., b_{\tau_N}$, we can generate STKDVs with $O(XYTN + Yn)$ time for all pairs of spatial and temporal bandwidths.*

Recall that there are M spatial bandwidths. Our method, named as PREFIX$_{\text{tuning}}$, can support the bandwidth tuning problem with $O(M(XYTN + Yn))$ time (cf. Theorem 3).

Theorem 3. *PREFIX$_{tuning}$ can solve the bandwidth tuning problem (i.e., Problem 2) with $O(M(XYTN + Yn))$ time.*

As a remark, the state-of-the-art SWS method takes $O(MNXY(T + n))$ time for solving the bandwidth tuning problem (cf. Section 2.2). Therefore, our PREFIX$_{\text{tuning}}$ method (with $O(M(XYTN + Yn))$ time) significantly reduces the time complexity for solving this problem.

3.5 Space Complexity of PREFIX

In this section, we investigate the space complexity of all our methods, PREFIX$_{\text{single}}$, PREFIX$_{\text{multiple}}$, and PREFIX$_{\text{tuning}}$.

PREFIX$_{single}$: Since PREFIX$_{\text{single}}$ needs to generate STKDV with the resolution size $X \times Y$ for a single timestamp t_i and access the data points in the sliding window $W(t_i)$, this approach needs to take at least $O(XY + n)$ space in the worst case (with $|W(t_i)| \to n$). Since PREFIX$_{\text{single}}$ is based on the simple modification of the bucket-based algorithm (cf. the core idea 1 in Section 3.1), SLAM$_{\text{BUCKET}}$, in [17], this approach does not affect the overall space complexity (cf. Theorem 4 in [17]). As such, PREFIX$_{\text{single}}$ retains in $O(XY + n)$ space (cf. Theorem 4).

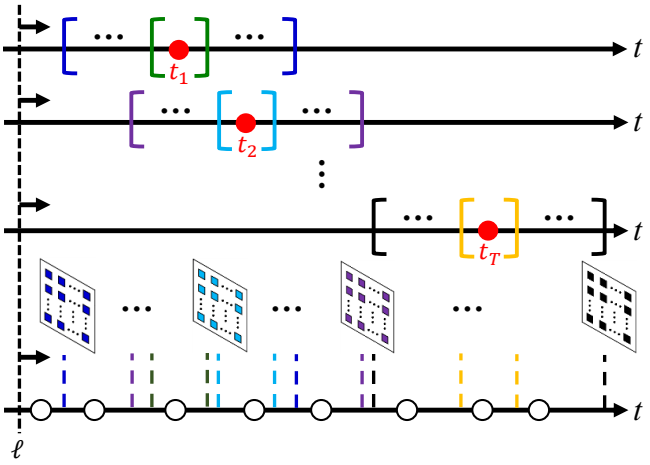


Figure 11: Maintain the statistical terms of prefix sets for all timestamps of the endpoints (of the sliding windows).

Theorem 4. *PREFIX$_{single}$ takes $O(XY + n)$ space to generate STKDV with a single timestamp (i.e., solving Problem 1 with $T = 1$).*

PREFIX$_{multiple}$: To generate STKDV with the resolution size $X \times Y$ and T timestamps for the dataset $\widehat{P}$ with size n, PREFIX$_{\text{multiple}}$ needs to take at least $O(XYT + n)$ space for outputting the visualizations and scanning the dataset. In addition, this method only needs to maintain at most $O(T)$ statistical terms for prefix sets (i.e., pixel-planes in the lower time-axis of Figure 9) and $O(T)$ statistical terms for the windows (i.e., pixel-planes in the upper time-axis of Figure 9), which consume $O(XYT)$ additional space. Therefore, the space complexity of PREFIX$_{\text{multiple}}$ remains in $O(XYT+n)$ (cf. Theorem 5).

Theorem 5. *PREFIX$_{multiple}$ takes $O(XYT + n)$ space to generate STKDV (cf. Problem 1) with T timestamps.*

PREFIX$_{tuning}$: Since we need to generate MN STKDVs (with M spatial bandwidths and N temporal bandwidths) for the dataset $\widehat{P}$ with size n, we need to output the visualizations (with $O(MNXYT)$ space) and scan the dataset (with $O(n)$ space). Therefore, PREFIX$_{\text{tuning}}$ takes at least $O(MNXYT + n)$ space. In addition, Figure 11 shows that PREFIX$_{\text{tuning}}$ needs to maintain the statistical terms of prefix sets for all endpoints ($2TN$ in total), which take $O(NXYT)$ additional space, given a fixed spatial bandwidth. Since the additional space can be cleared before we process the next spatial bandwidth, the space complexity of PREFIX$_{\text{tuning}}$ remains in $O(MNXYT + n)$ (cf. Theorem 6).

Theorem 6. *PREFIX$_{tuning}$ takes $O(MNXYT + n)$ space to solve the bandwidth tuning problem (i.e., Problem 2).*

Based on the above discussion, both PREFIX$_{\text{single}}$, PREFIX$_{\text{multiple}}$, and PREFIX$_{\text{tuning}}$ achieve the same space complexity compared

with the state-of-the-art SWS method (cf. Table 1) under different problem settings.

4 EXPERIMENTAL EVALUATION

In this section, we first introduce the experimental settings in Section 4.1. Then, we investigate the efficiency of all methods for computing STKDV with on-the-fly timestamps and known timestamps in Section 4.2 and Section 4.3, respectively. Next, we further conduct the efficiency experiments for all methods to solve the bandwidth tuning problem in Section 4.4. After that, we discuss the space consumption of all methods for generating STKDV in Section 4.5. Lastly, we conduct the case study in Section 4.6 to analyze spatial-temporal hotspots in the New York taxi dataset (i.e., New $\text{York}_{\text{taxi}}$).

4.1 Experimental Settings

We adopt six large-scale location datasets for testing, which are listed in Table 3. These datasets can be classified into five categories, namely COVID-19 cases, crime events, traffic accidents, 311 calls, and pickup locations. By default, we set the resolution size to be $1{,}280 \times 960$ and the number of timestamps to be 32. Furthermore, we use the Scott's rule [16] to choose the default spatial bandwidth $b_s^{(D)}$ and the default temporal bandwidth $b_t^{(D)}$ for each dataset.

Table 3: Datasets.

Dataset	n	Category	Ref.
Ontario	560,856	COVID-19 cases	[6]
Los Angeles	1,255,668	Crime events	[3]
New York	1,674,261	Traffic accidents	[4]
London	2,075,950	Traffic accidents	[8]
San Francisco	5,003,812	311 calls	[9]
$\text{New York}_{\text{taxi}}$	13,596,055	Pickup locations	[5]

In our experiments, we first compare our methods, $\text{PREFIX}_{\text{single}}$, $\text{PREFIX}_{\text{multiple}}$, and $\text{PREFIX}_{\text{tuning}}$, with the state-of-the-art method, SWS [16], under different problem settings (cf. Table 1), using the first five datasets of Table 3. Then, we use the largest dataset, $\text{New York}_{\text{taxi}}$, to conduct the case study. We implemented all these methods with C++ and conducted experiments on an Intel i7 3.19GHz PC with 32GB memory. In this paper, we use the response time (sec) and memory space (MB) to measure the time and space efficiency, respectively, of each method and only report the results that are within 86,400 sec (i.e., one day). Due to space limitations, we only show the experiment results for using the Epanechnikov function (cf. Table 2), which is the most famous one, as the spatial and temporal kernels.

4.2 STKDV with On-the-fly Timestamps

Although $\text{PREFIX}_{\text{single}}$ can theoretically achieve the lower time complexity for generating STKDV with on-the-fly timestamps (cf. Table 1), we still do not know the practical improvement of $\text{PREFIX}_{\text{single}}$ compared with the state-of-the-art SWS method. In this section, we conduct the following experiments to test the efficiency of all methods under this problem setting.

Varying the resolution size: In this experiment, we randomly generate $T = 32$ timestamps and choose four resolution sizes, which are 320×240, 640×480, $1{,}280 \times 960$, and $2{,}560 \times 1{,}920$, for testing the efficiency of all methods. Since our method, $\text{PREFIX}_{\text{single}}$, has the lower time complexity (with $O(Y(X+n))$ time for each on-the-fly timestamp) compared with the SWS method (with $O(XYn)$ time for each on-the-fly timestamp), $\text{PREFIX}_{\text{single}}$ can achieve at least 115x speedup (cf. Figure 12). In addition, the time gap between these two methods increases for each dataset once we increase the resolution size.

Varying the dataset size: We proceed to investigate how the dataset size affects the response time of all methods. To conduct this experiment, we first randomly sample each location dataset with four ratios, which are 25%, 50%, 75%, and 100% (the original one), and then test the response time of all methods for these sampled datasets. In Figure 13, we can see that $\text{PREFIX}_{\text{single}}$ can provide 346x to 935x speedup compared with the SWS method. The main reason is that $\text{PREFIX}_{\text{single}}$ has the lower time complexity for computing STKDV with on-the-fly timestamps.

Varying the number of timestamps: We further investigate how the number of (on-the-fly) timestamps T can affect the response time of all methods. To conduct this experiment, we first randomly generate $T = 2, 4, 8, 16, 32$ timestamps. Then, for each T, we can measure the response time of all methods using those T timestamps. Due to the lower time complexity of $\text{PREFIX}_{\text{single}}$, this method can consistently achieve better efficiency (with 420x to 986x speedup) compared with the SWS method (cf. Figure 14), no matter which T we adopt.

4.3 STKDV with Known Timestamps

We further investigate the time efficiency of our $\text{PREFIX}_{\text{multiple}}$ method and the state-of-the-art SWS method for computing STKDV with known timestamps. In this section, we discuss the results for the following experiments.

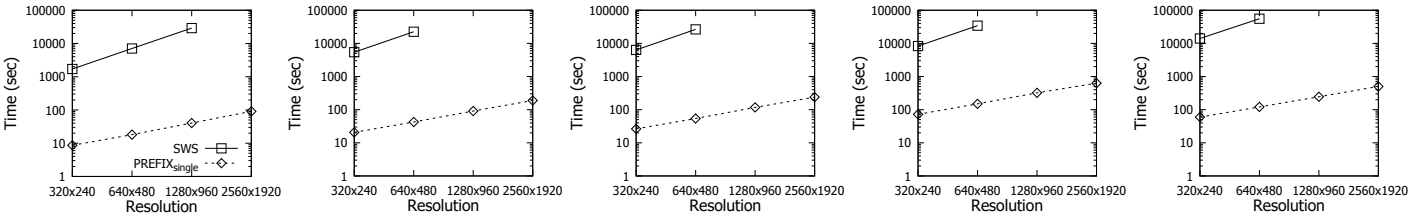
Varying the resolution size: In this experiment, we fix the number of timestamps to be 32 and vary the resolution size from 320×240 to $2{,}560 \times 1{,}920$ for testing the response time of $\text{PREFIX}_{\text{multiple}}$ and SWS. Since $\text{PREFIX}_{\text{multiple}}$ can reduce the time complexity for computing STKDV (with known timestamps), our method can achieve 217x to 1,524x speedup compared with SWS (cf. Figure 15).

Varying the number of timestamps: Here, we further examine how the number of timestamps T can affect the response time of $\text{PREFIX}_{\text{multiple}}$ and SWS. Since the time complexity of $\text{PREFIX}_{\text{multiple}}$ and SWS is $O(XYT + Yn)$ and $O(XY(T+n))$, respectively, and the number of data points n is much larger compared with the number of timestamps T, both the $\text{PREFIX}_{\text{multiple}}$ and SWS methods are insensitive to this parameter T (cf. Figure 16). With the lower time complexity of $\text{PREFIX}_{\text{multiple}}$, this method can achieve 737x to 1,906x speedup.

4.4 Bandwidth Tuning

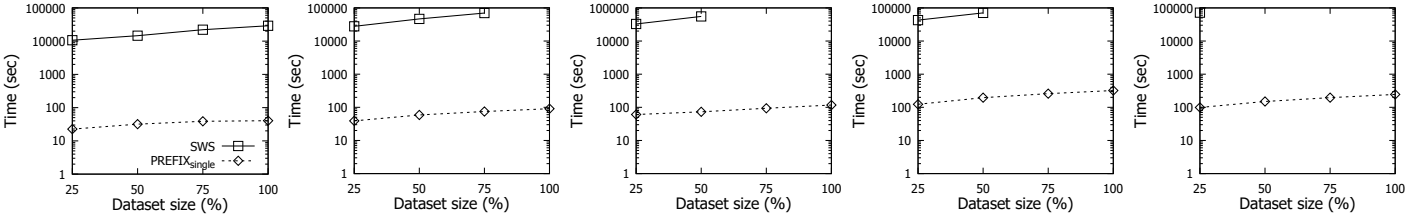
Recall that domain experts need to tune the spatial and temporal bandwidths in order to obtain the best visual quality of STKDV. In this section, we compare the efficiency of our $\text{PREFIX}_{\text{tuning}}$ method with the state-of-the-art SWS method under this problem setting.

Varying the number of spatial bandwidths: Here, we investigate how the number of spatial bandwidths M can affect the efficiency of the $\text{PREFIX}_{\text{tuning}}$ and SWS methods. To conduct this



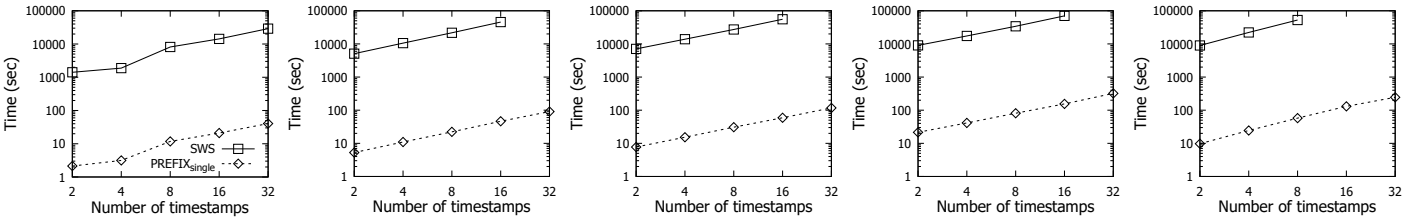
(a) Ontario (b) Los Angeles (c) New York (d) London (e) San Francisco

Figure 12: Response time (in total) for computing STKDV with $T = 32$ on-the-fly timestamps, varying the resolution size.



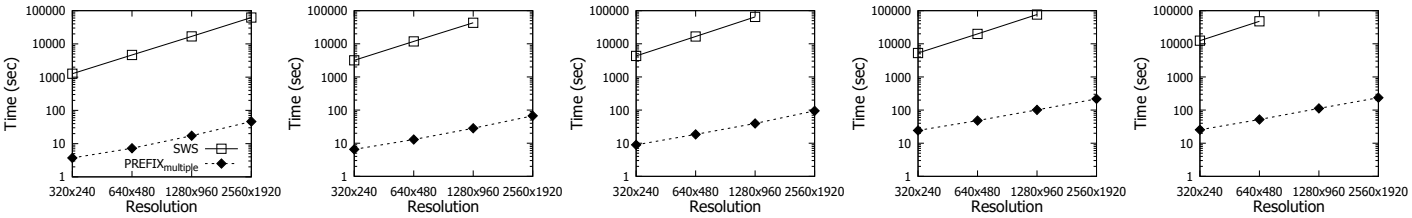
(a) Ontario (b) Los Angeles (c) New York (d) London (e) San Francisco

Figure 13: Response time (in total) for computing STKDV with the default resolution size 1,280 × 960 and $T = 32$ on-the-fly timestamps, varying the dataset size.



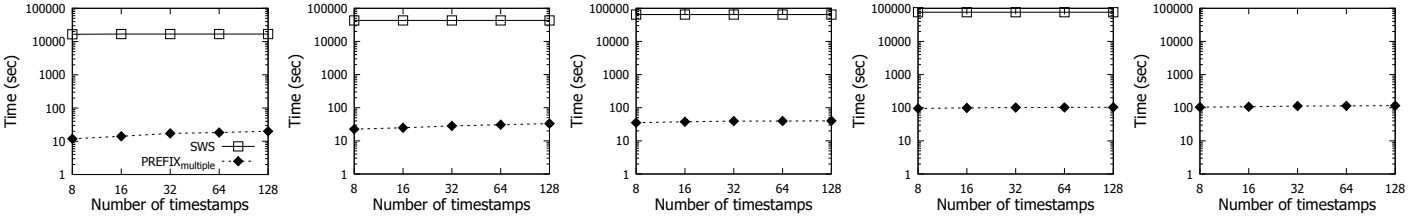
(a) Ontario (b) Los Angeles (c) New York (d) London (e) San Francisco

Figure 14: Response time (in total) for computing STKDV with the default resolution size 1,280 × 960, varying the number of (on-the-fly) timestamps T.



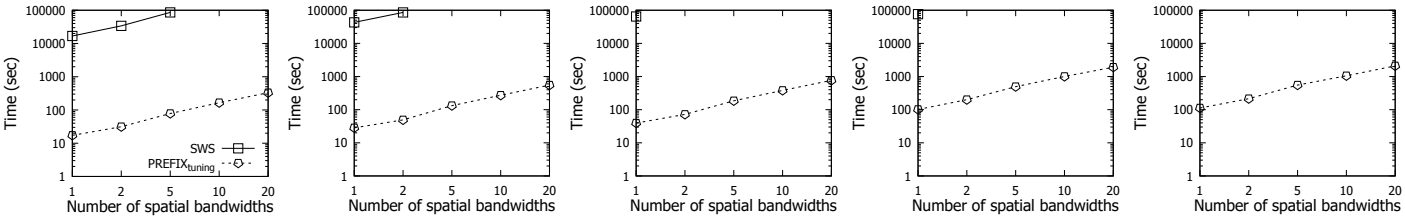
(a) Ontario (b) Los Angeles (c) New New York (d) London (e) San Francisco

Figure 15: Response time for computing STKDV with $T = 32$ known timestamps, varying the resolution size.



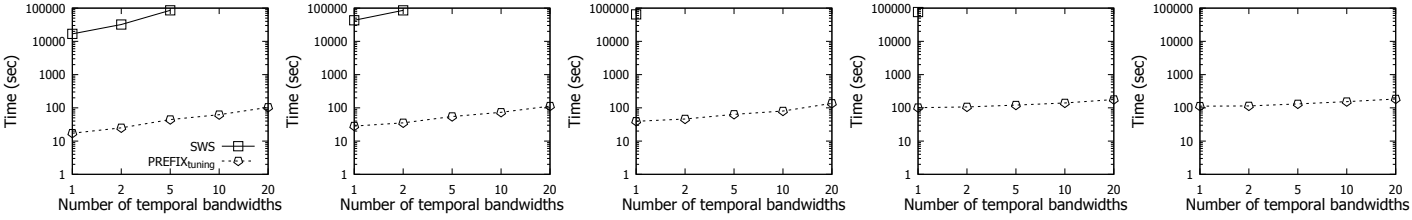
(a) Ontario (b) Los Angeles (c) New York (d) London (e) San Francisco

Figure 16: Response time for computing STKDV with the default resolution size 1,280 × 960, varying the number of (known) timestamps T.



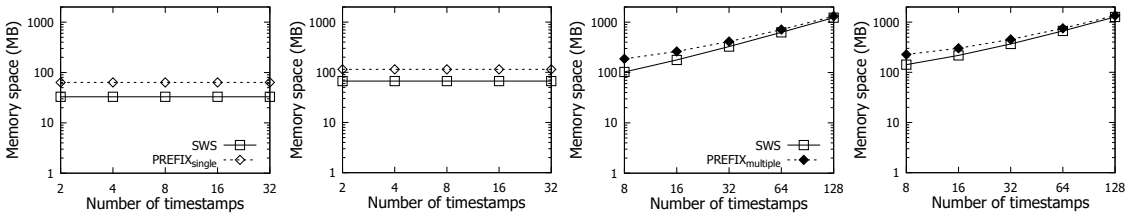
(a) Ontario (b) Los Angeles (c) New York (d) London (e) San Francisco

Figure 17: Response time for computing STKDVs with the default resolution size 1,280 × 960 and the default number of timestamps $T = 32$, varying the number of spatial bandwidths M.



(a) Ontario (b) Los Angeles (c) New York (d) London (e) San Francisco

Figure 18: Response time for computing STKDVs with the default resolution size 1,280 × 960 and the default number of timestamps $T = 32$, varying the number of temporal bandwidths N.



(a) Ontario (on-the-fly timestamps) (b) Los Angeles (on-the-fly timestamps) (c) Ontario (known timestamps) (d) Los Angeles (known timestamps)

Figure 19: Memory space consumption for generating STKDV for the Ontario (a and c) and Los Angeles (b and d) datasets, varying the number of on-the-fly timestamps (a and b) and the number of known timestamps (c and d).

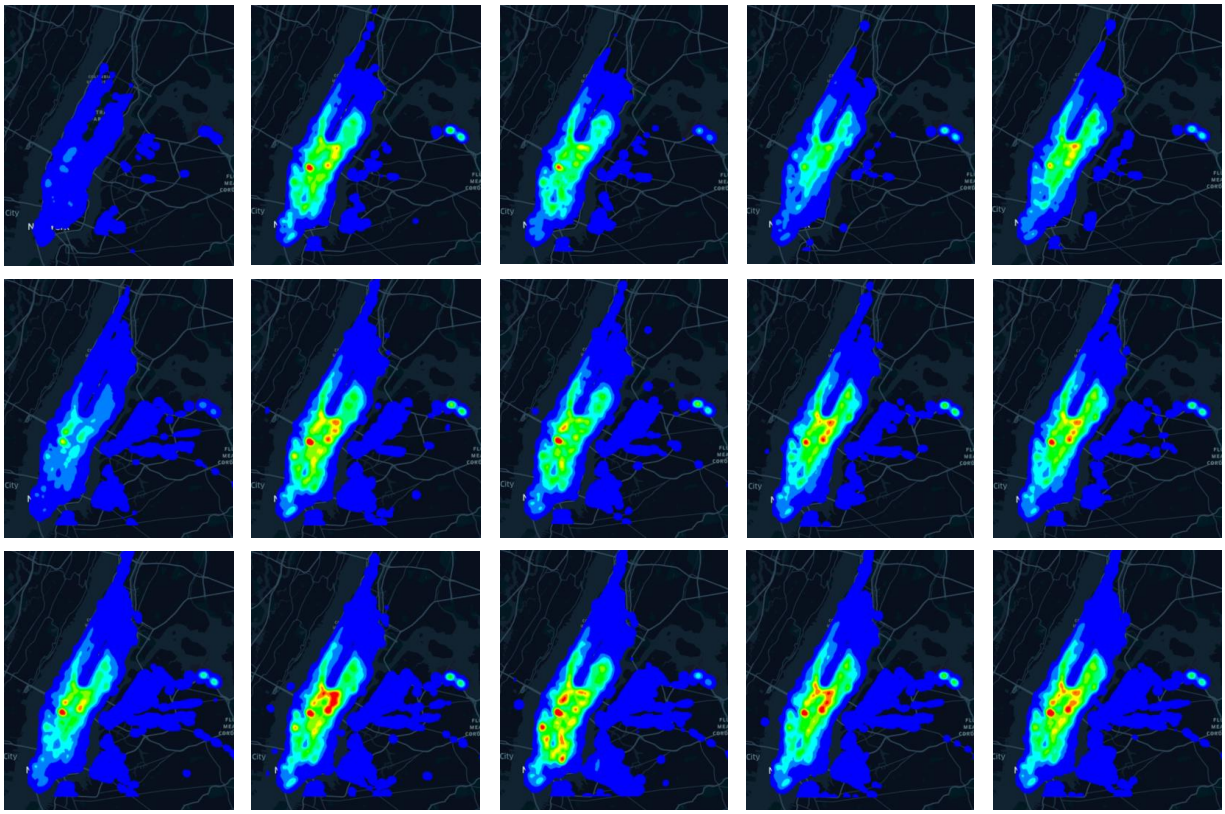
experiment, we first adopt the default resolution size 1,280 × 960, adopt the default number of timestamps $T = 32$, and fix the number of temporal bandwidths N to be 1 (use the default temporal bandwidth $b_t^{(D)}$). Then, we equally divide the range $[0, 2b_s^{(D)}]$ and choose M spatial bandwidths, i.e., $\frac{2b_s^{(D)}}{M}$, $\frac{4b_s^{(D)}}{M}$,..., $2b_s^{(D)}$, where we vary the number M from 1 to 20 for testing. Since the time complexity of our $\text{PREFIX}_{\text{tuning}}$ method and the SWS method (cf. Table 1) is linearly proportional to the number of spatial bandwidths M, the response time increases linearly for these two methods (cf. Figure 17). With the lower time complexity of $\text{PREFIX}_{\text{tuning}}$, this method achieves nearly three-order-of-magnitude speedup compared with the state-of-the-art SWS method.

Varying the number of temporal bandwidths: In this experiment, we further test how the number of temporal bandwidths N can affect the efficiency of the $\text{PREFIX}_{\text{tuning}}$ and SWS methods. We follow the same setting as the previous experiment, except that we fix the number of spatial bandwidths M to be 1 (use the default spatial bandwidth $b_s^{(D)}$) and vary the number N from 1 to 20 for testing, where the N bandwidths are $\frac{2b_t^{(D)}}{N}$, $\frac{4b_t^{(D)}}{N}$,..., $2b_t^{(D)}$. In Figure 18, we can see that $\text{PREFIX}_{\text{tuning}}$ can achieve at least 754x speedup compared with the SWS method. Furthermore, since the $\text{PREFIX}_{\text{tuning}}$ method takes $O(M(XYTN + Yn))$ time and n is normally much larger compared with other parameters, this method is not sensitive to the number of temporal bandwidths N.

4.5 Space Consumption for Computing STKDV

In this section, we investigate the memory space consumption of all methods for generating STKDV with (i) on-the-fly timestamps and (ii) known timestamps. Here, we use the Ontario and Los Angeles datasets for testing. Due to space limitations, we omit the experiment for testing the space consumption of solving the bandwidth tuning problem.

Varying the number of on-the-fly timestamps: In the first experiment, we choose the number of on-the-fly timestamps T to be 2, 4, 8, 16, and 32 and then measure the memory space consumption of the SWS and $\text{PREFIX}_{\text{single}}$ methods for each T. In Figure 19a and



(a) 7/1/2014 (Tue) (b) 17/1/2014 (Fri) (c) 19/1/2014 (Sun) (d) 23/1/2014 (Thu) (e) 24/1/2014 (Fri)

Figure 20: Generate STKDVs with respect to three temporal bandwidths, i.e., 0.1 days (1st row), 0.3 days (2nd row), and 0.5 days (3rd row), where we fix the spatial bandwidth to be 350 meters, choose the resolution size to be 1,280 × 960, and show five timestamps.

Figure 19b, since both SWS and $\text{PREFIX}_{\text{single}}$ methods process each timestamp independently, all these methods are not sensitive to the number T. Due to the same space complexity of these two methods (cf. Table 1), $\text{PREFIX}_{\text{single}}$ only incurs slight space overhead compared with the SWS method.

Varying the number of known timestamps: In the second experiment, we vary the number of known timestamps T from 8 to 128 and test the memory space consumption of all methods, i.e., SWS and $\text{PREFIX}_{\text{multiple}}$. In Figure 19c and Figure 19d, we can see that the space consumption of these two methods is linearly proportional to the number of timestamps T. Since the space complexity of SWS and $\text{PREFIX}_{\text{multiple}}$ is the same (cf. Table 1), the space consumption of $\text{PREFIX}_{\text{multiple}}$ is close to SWS.

4.6 Case Study: STKDV-based Hotspot Analysis in the New $\text{York}_{\text{taxi}}$ Dataset

Since PREFIX is more scalable to large-scale datasets, we adopt this approach to analyze the traffic hotspots of the largest New $\text{York}_{\text{taxi}}$ dataset (cf. Table 3), which, to the best of our knowledge, cannot be feasibly supported by existing solutions. To conduct this case study, we choose the resolution size to be 1,280 × 960, set the number of timestamps T to be 32, and specify multiple spatial bandwidths, including 300, 350, 400, 450, and 500 meters, and multiple temporal bandwidths, including 0.1, 0.2, 0.3, 0.4, and 0.5 days, for tuning the best visual quality of STKDV (cf. Problem 2). *Under this setting, generating these 25 STKDVs only takes 1,383 seconds (~23 minutes), while the state-of-the-art SWS method takes more than 86,400 seconds (i.e., one day) to generate even a single STKDV.*

Due to space limitations, Figure 20 only shows the STKDV results for three temporal bandwidths, i.e., 0.1 days, 0.3 days, and 0.5 days, with the fixed spatial bandwidth, i.e., 350 meters, and five timestamps. Note that the traffic hotspots are mainly in the Manhattan region. Furthermore, suppose that we choose 0.1 days as the temporal bandwidth, we cannot detect any hotspots (cf. the first row of Figure 20). Once we increase the temporal bandwidth (e.g., 0.3 days and 0.5 days), we can discover the meaningful hotspots in this region. In addition, we can notice that different temporal bandwidths can be suitable in different timestamps. For example, using 0.5 days as the temporal bandwidth on 7/1/2014, 19/1/2014, and 24/1/2014 can discover more hidden patterns (red regions). while it can provide the oversmoothing hotspot maps (a large red region) on 17/1/2014 and 23/1/2014. As such, our fast solution enables domain

experts to test more spatial and temporal bandwidths so that they can obtain the best visualization results.

5 RELATED WORK

Kernel density visualization (KDV) [15, 17] is a de facto tool for analyzing hotspots in different geographical regions, which has been extensively used in many communities, including transportation science [47, 52, 57], criminology [26, 43], and disease outbreak analysis [20, 37]. However, KDV does not consider the occurrence time of each event for generating the hotspot map, which can provide inaccurate density estimation [19, 28, 32]. To tackle this issue, many researchers [14, 19, 28, 29, 32, 38, 58, 61, 66] adopt the more advanced visualization tool, called spatial-temporal kernel density visualization (STKDV) to analyze their location data. Although this tool can provide more accurate visualization results, STKDV is computationally expensive, which is not scalable to high resolution sizes, large-scale datasets, and a large number of timestamps. In this section, we review different types of methods for improving the efficiency of STKDV and also review some research studies that are closely related to this work.

Range-query-based methods: Recall from Table 2 that only those data points $(\mathbf{p}, t_\mathbf{p}) \in \widehat{P}$ with $dist(\mathbf{q}, \mathbf{p}) \leq b_\sigma$ and $dist(t_i, t_\mathbf{p}) \leq b_\tau$ can contribute to the spatial-temporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ (cf. Equation 1). Therefore, one possible approach is to first find the range query set $R(\mathbf{q}, t_i) = \{(\mathbf{p}, t_\mathbf{p}) \in \widehat{P} : dist(\mathbf{q}, \mathbf{p}) \leq b_\sigma \text{ and } dist(t_i, t_\mathbf{p}) \leq b_\tau\}$ for each pixel-timestamp pair $(\mathbf{q}, t_i)$ and then evaluate $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ based on this range query set $R(\mathbf{q}, t_i)$. Since many index structures [45, 59], e.g., kd-tree [12] and ball-tree [36], have been developed to improve the efficiency for solving range queries, this approach can also improve the efficiency for computing STKDV. However, range-query-based methods cannot theoretically reduce the time complexity for generating STKDV. In addition, Chan et al. [16] verify that this approach is consistently inferior compared with their SWS method.

Sliding-window-based methods: In both database and data mining communities, sliding-window-based methods have been extensively used to improve the efficiency for many query processing tasks, including aggregate queries (e.g., COUNT, MIN, MAX,...) [23, 31, 48–50, 53, 60], frequency queries [11, 24], top-k queries [55, 68], and skyline queries [51]. Since none of these studies focus on the spatial-temporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ (cf. Equation 1), they cannot be adopted to improve the efficiency for generating STKDV. Recently, Chan et al. [16] successfully develop the first sliding-window-based solution (SWS) to reduce the time complexity for generating STKDV. However, this method achieves inferior theoretical and practical performance compared with PREFIX (cf. Table 1).

Parallel methods: Recently, many research studies [16, 27, 46] adopt the parallel methods to improve the efficiency for generating STKDV. In particular, Chan et al. [16] incorporate this approach into the state-of-the-art method, SWS, which can significantly improve the practical efficiency. Nevertheless, the theoretical improvement of the parallel methods is at most $\mathcal{T}$ times using $\mathcal{T}$ threads (e.g., at most 16 times speedup with 16 threads). Since PREFIX achieves at least two-order-of-magnitude speedup compared with the SWS method, the parallel version of the SWS method is still inferior compared with PREFIX under the single computer setting (with limited number of threads). Due to space limitations, we will investigate how to parallelize our solution, PREFIX, in the future work.

KDV methods: In the literature, many researchers [15, 17, 21, 25, 40–42, 63–65] have proposed efficient algorithms for generating KDV, which is closely related to this work. In these research studies, Gray et al. [25] and Gan et al. [21] incorporate the lower and upper bound functions into an index structure to improve the efficiency for computing the spatial kernel density function $\mathcal{F}_P(\mathbf{q})$ for each pixel $\mathbf{q}$. Zheng et al. [63–65] and Phillips et al. [40–42] propose to first obtain the sampled dataset S from the original location dataset, and then evaluate the approximate spatial kernel density function $\mathcal{F}_S(\mathbf{q})$. Recently, Chan et al. [15, 17] also develop complexity-optimized algorithms to generate KDV. Although the above research studies can theoretically and practically improve the efficiency of generating KDV, all these methods are tailor-made for efficiently evaluating the spatial kernel density function $\mathcal{F}_P(\mathbf{q})$. Hence, none of these research studies can be directly adopted to support STKDV (with the more complex spatial-temporal kernel density function $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ (cf. Equation 1)).

6 CONCLUSION

In this paper, we study spatial-temporal kernel density visualization (STKDV), which has been extensively used in many applications, including traffic accident hotspot detection, crime hotspot detection, and disease outbreak detection. However, STKDV is very time-consuming, which cannot be scalable to large-scale datasets, high resolution sizes, and a large number of timestamps. Although Chan et al. [16] propose the first work, called SWS, that can successfully reduce the time complexity for generating STKDV, this approach (i) is unable to reduce the time complexity for supporting exploratory analysis, (ii) still takes $O(XY(T+n))$ time, and (iii) does not provide any optimization techniques for bandwidth tuning. To overcome these issues, we propose the prefix-set-based solution, namely PREFIX, which consists of three methods, $\text{PREFIX}_{\text{single}}$ for handling (i), $\text{PREFIX}_{\text{multiple}}$ for handling (ii), and $\text{PREFIX}_{\text{tuning}}$ for handling (iii). Our theoretical analysis (cf. Table 1) shows that all these methods can significantly reduce the time complexity, without increasing the space complexity, under different problem settings. Furthermore, our experiment results show that PREFIX (1) achieves 115x to 1906x speedup compared with the state-of-the-art SWS method for all problem settings and (2) is the first solution that can be scalable to generate high-resolution STKDVs (1280×960) for the $\text{New York}_{\text{taxi}}$ dataset (with 13.6 million data points).

In the future, we plan to extend this solution to support other types of kernel functions, including Gaussian kernel and exponential kernel, and other geospatial analysis tasks, e.g., spatial-temporal network kernel density visualization (STNKDV) [44]. Furthermore, we will investigate how to parallelize our solution.

7 APPENDIX (PROOF OF LEMMA 1)

In this proof, we focus on showing that the statistical terms $S^{(u)}_{W_I}(\mathbf{q})$ can be computed in $O(X + |W_I|)$ time for a row of pixels, i.e., $\mathbf{q}_1$, $\mathbf{q}_2$,..., $\mathbf{q}_X$ (cf. the yellow region in Figure 21), where:

$$\mathbf{q}_1.y = \mathbf{q}_2.y = \ldots = \mathbf{q}_X.y = k \tag{15}$$

Based on this statement, we can conclude that the time complexity for finding $S^{(u)}_{W_I}(\mathbf{q})$ for all (i.e., Y rows of) pixels is $O(Y(X + |W_I|))$ time. Therefore, we can compute $\mathcal{F}_{\widehat{P}}(\mathbf{q}, t_i)$ for all pixels $\mathbf{q}$ in $O(XY)$ time using Equation 3 once the statistical terms $S^{(u)}_{W_I}(\mathbf{q})$ are available. Hence, we can show that the overall time complexity is $O(Y(X + |W_I|))$ time.

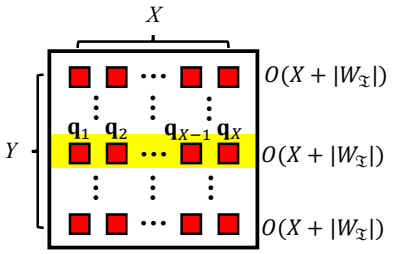


Figure 21: Computing the statistical terms $S^{(u)}_{W_I}(\mathbf{q})$ for each row of pixels takes $O(X + |W_I|)$ time, given the sliding window W_I.

Here, we use three core ideas to show the above statement, which are: (1) decomposition of $S^{(u)}_{W_I}(\mathbf{q})$, (2) lower and upper bound functions, and (3) bucket-based algorithm.

Core idea 1 (Decomposition of $S^{(u)}_{W_I}(\mathbf{q})$): Consider the spatial kernels in Table 2. Note that only those data points $(\mathbf{p}, t_\mathbf{p})$ with $dist(\mathbf{q}, \mathbf{p}) \le b_\sigma$ can contribute to the statistical terms $S^{(u)}_{W_I}(\mathbf{q})$ (cf. Equation 10). Therefore, we can first obtain the range query set $R(\mathbf{q})$ in the sliding window W_I (cf. Equation 9), where:

$$R(\mathbf{q}) = \{(\mathbf{p}, t_\mathbf{p}) \in W_I : dist(\mathbf{q}, \mathbf{p}) \le b_\sigma\} \tag{16}$$

By adopting the Epanechnikov kernel in $K^{(b_\sigma)}_{\text{space}}(\mathbf{q}, \mathbf{p})$ (cf. Table 2) as an example, we can decompose $S^{(u)}_{W_I}(\mathbf{q})$ (cf. Equation 10) into the following expression.

$$\begin{aligned} S^{(u)}_{W_I}(\mathbf{q}) &= \sum_{(\mathbf{p},t_\mathbf{p})\in R(\mathbf{q})} t^u_\mathbf{p} \cdot \left(1 - \frac{1}{b^2_\sigma} dist(\mathbf{q}, \mathbf{p})^2\right) \\ &= \sum_{(\mathbf{p},t_\mathbf{p})\in R(\mathbf{q})} t^u_\mathbf{p} \cdot \left(1 - \frac{1}{b^2_\sigma}(||\mathbf{q}||_2^2 - 2\mathbf{q}^T\mathbf{p} + ||\mathbf{p}||_2^2)\right) \\ &= \left(1 - \frac{1}{b^2_\sigma}||\mathbf{q}||_2^2\right) \cdot C_{R(\mathbf{q})} + \frac{2}{b^2_\sigma}\mathbf{q}^T\mathbf{v}_{R(\mathbf{q})} - \frac{1}{b^2_\sigma} H_{R(\mathbf{q})} \end{aligned}$$

where $C_{R(\mathbf{q})} = \sum_{(\mathbf{p},t_\mathbf{p})\in R(\mathbf{q})} t^u_\mathbf{p}$, $\mathbf{v}_{R(\mathbf{q})} = \sum_{(\mathbf{p},t_\mathbf{p})\in R(\mathbf{q})} t^u_\mathbf{p}\mathbf{p}$, and $H_{R(\mathbf{q})} = \sum_{(\mathbf{p},t_\mathbf{p})\in R(\mathbf{q})} t^u_\mathbf{p}||\mathbf{p}||_2^2$ are the aggregate terms of $R(\mathbf{q})$. Therefore, suppose that we can efficiently maintain these three aggregate terms for a row of pixels $\mathbf{q}$, we can efficiently evaluate $S^{(u)}_{W_I}(\mathbf{q})$ for all these pixels $\mathbf{q}$.

Core idea 2 (Lower and upper bound functions): In Figure 22, note that we can find the envelope $E(k)$ in W_I for a row of pixels $\mathbf{q}_1$, $\mathbf{q}_2$,..., and $\mathbf{q}_X$, where:

$$E(k) = \{(\mathbf{p}, t_\mathbf{p}) \in W_I : |k - \mathbf{p}.y| \le b_\sigma\} \tag{17}$$

Based on Equation 17, we can assign proper[1] lower and upper bound functions for each data point $(\mathbf{p}, t_\mathbf{p})$ (cf. white points in Figure 22) in $E(k)$ with at most $O(|W_I|)$ time, where:

$$LB_k(\mathbf{p}) = \mathbf{p}.x - \sqrt{b^2_\sigma - (k - \mathbf{p}.y)^2} \tag{18}$$

$$UB_k(\mathbf{p}) = \mathbf{p}.x + \sqrt{b^2_\sigma - (k - \mathbf{p}.y)^2} \tag{19}$$

Therefore, once we have $LB_k(\mathbf{p}) \le \mathbf{q}.x \le UB_k(\mathbf{p})$, we can conclude that those data points $\mathbf{p}$ are inside the range query set $R(\mathbf{q})$ (cf. Figure 23), i.e., $dist(\mathbf{q}, \mathbf{p}) \le b_\sigma$.

[1] $|k - \mathbf{p}.y| \le b_\sigma$ implies that Equation 18 and Equation 19 have real values.

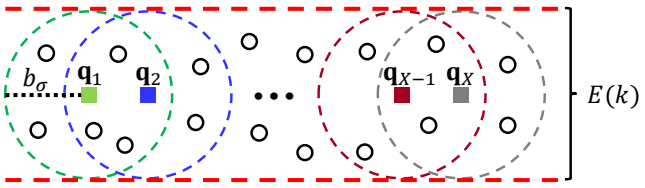


Figure 22: An envelope $E(k)$ for a row of pixels $\mathbf{q}_1$, $\mathbf{q}_2$,..., and $\mathbf{q}_X$, where $\mathbf{q}_1.y = \mathbf{q}_2.y = ... = \mathbf{q}_X.y = k$.

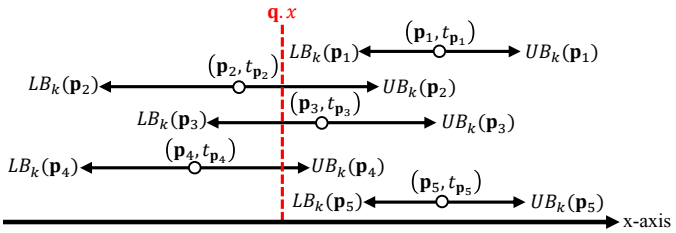


Figure 23: The data points $(\mathbf{p}_2, t_{\mathbf{p}_2})$, $(\mathbf{p}_3, t_{\mathbf{p}_3})$, and $(\mathbf{p}_4, t_{\mathbf{p}_4})$ are in the range query set $R(\mathbf{q})$.

Core idea 3 (Bucket-based algorithm): To evaluate the statistical terms for a row of X pixels (cf. Figure 21), the bucket-based algorithm follows these two steps.

Step 1: We establish the lower and upper bound buckets (cf. Equation 20 and Equation 21, respectively) for the left-open intervals $(\mathbf{q}_0.x, \mathbf{q}_1.x]$, $(\mathbf{q}_1.x, \mathbf{q}_2.x]$,..., $(\mathbf{q}_X.x, \mathbf{q}_{X+1}.x]$, where we let $\mathbf{q}_0.x = -\infty$ and $\mathbf{q}_{X+1}.x = \infty$ be the dummy variables.

$$B_L(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x) = \{(\mathbf{p}, t_\mathbf{p}) \in E(k) : \mathbf{q}_{v-1}.x < LB_k(\mathbf{p}) \le \mathbf{q}_v.x\} \tag{20}$$

$$B_U(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x) = \{(\mathbf{p}, t_\mathbf{p}) \in E(k) : \mathbf{q}_{v-1}.x < UB_k(\mathbf{p}) \le \mathbf{q}_v.x\} \tag{21}$$

Using Figure 24 as an example, since the lower (upper) bound value of the orange (yellow) data point is inside the interval $(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x]$, the lower (upper) bound bucket $B_L(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x)$ ($B_U(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x)$) covers the orange (yellow) data point.

Since we only need to scan all the intervals and assign them into the corresponding buckets, the time complexity of the step 1 is $O(X + |W_I|)$.

Step 2: With these lower and upper bound buckets ($1 \le v \le X$), we can iteratively access $\mathbf{q}_v$ and maintain the accumulative lower and upper bound buckets (cf. Figure 24), i.e., $A_L(\mathbf{q}_v.x) = \{(\mathbf{p}, t_\mathbf{p}) \in E(k) : LB_k(\mathbf{p}) \le \mathbf{q}_v.x\}$ and $A_U(\mathbf{q}_v.x) = \{(\mathbf{p}, t_\mathbf{p}) \in E(k) : UB_k(\mathbf{p}) \le \mathbf{q}_v.x\}$, respectively, using the following equations.

$$A_L(\mathbf{q}_v.x) = \begin{cases} A_L(\mathbf{q}_{v-1}.x) \cup B_L(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x) & \text{if } v \ge 2 \\ B_L(\mathbf{q}_0.x, \mathbf{q}_1.x) & \text{if } v = 1 \end{cases} \tag{22}$$

$$A_U(\mathbf{q}_v.x) = \begin{cases} A_U(\mathbf{q}_{v-1}.x) \cup B_U(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x) & \text{if } v \ge 2 \\ B_U(\mathbf{q}_0.x, \mathbf{q}_1.x) & \text{if } v = 1 \end{cases} \tag{23}$$

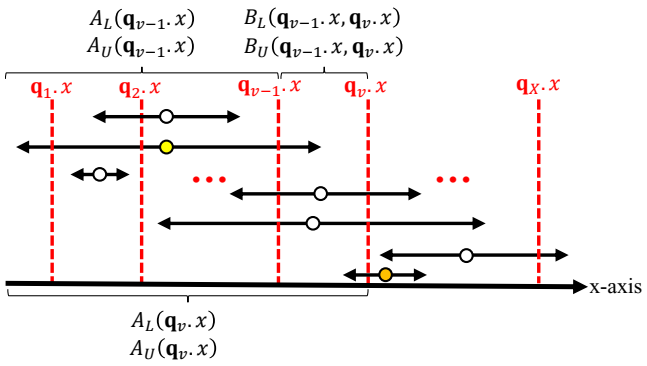


Figure 24: Maintain the lower and upper bound buckets, $B_L(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x)$ and $B_U(\mathbf{q}_{v-1}.x, \mathbf{q}_v.x)$, respectively, and the accumulative lower and upper bound buckets, i.e., $A_L(\mathbf{q}_v.x)$ and $A_U(\mathbf{q}_v.x)$, respectively, where $1 \le v \le X$, for processing a row of pixels.

Like the core idea 1, we also maintain the aggregate terms $C_{A_L(\mathbf{q}_v.x)}$, $C_{A_U(\mathbf{q}_v.x)}$, $\mathbf{v}_{A_L(\mathbf{q}_v.x)}$, $\mathbf{v}_{A_U(\mathbf{q}_v.x)}$, $H_{A_L(\mathbf{q}_v.x)}$, and $H_{A_U(\mathbf{q}_v.x)}$ for these accumulative buckets (e.g., $C_{A_L(\mathbf{q}_v.x)} = \sum_{(\mathbf{p},t_\mathbf{p}) \in A_L(\mathbf{q}_v.x)} t_\mathbf{p}^u$).

Based on the core idea 2, we note that $LB_k(\mathbf{p}) \le \mathbf{q}_v.x$, i.e., $(\mathbf{p}, t_\mathbf{p}) \in A_L(\mathbf{q}_v.x)$, and $\mathbf{q}_v.x > UB_k(\mathbf{p})$, i.e., $(\mathbf{p}, t_\mathbf{p}) \notin A_U(\mathbf{q}_v.x)$ imply $(\mathbf{p}, t_\mathbf{p}) \in R(\mathbf{q}_v)$. Therefore, we have:

$$R(\mathbf{q}_v) = A_L(\mathbf{q}_v.x) \backslash A_U(\mathbf{q}_v.x) \tag{24}$$

Furthermore, if $UB_k(\mathbf{p}) \le \mathbf{q}_v.x$, we also have $LB_k(\mathbf{p}) \le \mathbf{q}_v.x$. Hence, we have:

$$A_U(\mathbf{q}_v.x) \subseteq A_L(\mathbf{q}_v.x) \tag{25}$$

With Equation 24, Equation 25, and the core idea 1, we can use $O(1)$ time to compute $C_{R(\mathbf{q}_v)}$, $\mathbf{v}_{R(\mathbf{q}_v)}$, $H_{R(\mathbf{q}_v)}$ (cf. Equations 26 to 28), and the statistical terms $S_{W_I}^{(u)}(\mathbf{q}_v)$ once the aggregate terms of $A_L(\mathbf{q}_v.x)$ and $A_U(\mathbf{q}_v.x)$ are available.

$$C_{R(\mathbf{q}_v)} = C_{A_L(\mathbf{q}_v.x)} - C_{A_U(\mathbf{q}_v.x)} \tag{26}$$

$$\mathbf{v}_{R(\mathbf{q}_v)} = \mathbf{v}_{A_L(\mathbf{q}_v.x)} - \mathbf{v}_{A_U(\mathbf{q}_v.x)} \tag{27}$$

$$H_{R(\mathbf{q}_v)} = H_{A_L(\mathbf{q}_v.x)} - H_{A_U(\mathbf{q}_v.x)} \tag{28}$$

Recall that we only need to scan each lower (upper) bound bucket once throughout X iterations (i.e., X pixels) in order to maintain the accumulative buckets (cf. Equation 22 and Equation 23) and their aggregate terms. In addition, evaluating $C_{R(\mathbf{q}_v)}$, $\mathbf{v}_{R(\mathbf{q}_v)}$, $H_{R(\mathbf{q}_v)}$ (cf. Equations 26 to 28), and the statistical terms $S_{W_I}^{(u)}(\mathbf{q}_v)$ for all X pixels is $O(X)$ time. The time complexity of the step 2 remains in $O(X + |W_I|)$. Hence, we have proved this lemma.

REFERENCES

[1] [n. d.]. ArcGIS. http://pro.arcgis.com/en/pro-app/tool-reference/spatial-analyst/how-kernel-density-works.htm.

[2] [n. d.]. Deck.gl. https://deck.gl/docs/api-reference/aggregation-layers/heatmap-layer.

[3] [n. d.]. Los Angeles Open Data (last accessed: 8th June 2022). https://data.lacity.org/A-Safe-City/Crime-Data-from-2010-to-2019/63jg-8b9z.

[4] [n. d.]. NYC Open Data (last accessed: 8th June 2022). https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95.

[5] [n. d.]. NYC Yellow Taxi Trip Data. https://data.cityofnewyork.us/Transportation/2014-Yellow-Taxi-Trip-Data/gkne-dk5s.

[6] [n. d.]. Ontario Open Data (last accessed: 8th June 2022). https://data.ontario.ca/dataset/confirmed-positive-cases-of-covid-19-in-ontario.

[7] [n. d.]. QGIS. https://docs.qgis.org/2.18/en/docs/user_manual/plugins/plugins_heatmap.html.

[8] [n. d.]. Road Safety Data (last accessed: 8th June 2022). https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data.

[9] [n. d.]. San Francisco Open Data (last accessed: 8th June 2022). https://data.sfgov.org/City-Infrastructure/311-Cases/vw6y-z8j6.

[10] [n. d.]. Seaborn. https://seaborn.pydata.org/generated/seaborn.kdeplot.html.

[11] Ran Ben-Basat, Roy Friedman, and Rana Shahout. 2018. Stream Frequency Over Interval Queries. *Proc. VLDB Endow.* 12, 4 (2018), 433–445. https://doi.org/10.14778/3297753.3297762

[12] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.

[13] Michal Bíl, Richard Andrášik, and Jiří Sedoník. 2019. A detailed spatiotemporal analysis of traffic crash hotspots. *Applied Geography* 107 (2019), 82–90. https://doi.org/10.1016/j.apgeog.2019.04.008

[14] Chris Brunsdon, Jonathan Corcoran, and Gary Higgs. 2007. Visualising space and time in crime patterns: A comparison of methods. *Comput. Environ. Urban Syst.* 31, 1 (2007), 52–75. https://doi.org/10.1016/j.compenvurbsys.2005.07.009

[15] Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. SAFE: A Share-and-Aggregate Bandwidth Exploration Framework for Kernel Density Visualization. *Proc. VLDB Endow.* 15, 3 (2022), 513–526.

[16] Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. SWS: A Complexity-Optimized Solution for Spatial-Temporal Kernel Density Visualization. *Proc. VLDB Endow.* 15, 4 (2022), 814–827.

[17] Tsz Nam Chan, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. SLAM: Efficient Sweep Line Algorithms for Kernel Density Visualization. In *SIGMOD*. ACM, 2120–2134. https://doi.org/10.1145/3514221.3517823

[18] Michael Crimmins, Seri Park, Virginia Smith, and Peleg Kremer. 2021. A spatial assessment of high-resolution drainage characteristics and roadway safety during wet conditions. *Applied Geography* 133 (2021), 102477. https://doi.org/10.1016/j.apgeog.2021.102477

[19] Eric Delmelle, Coline Dony, Irene Casas, Meijuan Jia, and Wenwu Tang. 2014. Visualizing the impact of space-time uncertainties on dengue fever patterns. *International Journal of Geographical Information Science* 28, 5 (2014), 1107–1127. https://doi.org/10.1080/13658816.2013.871285

[20] Richard Elson, Tilman M. Davies, Iain R. Lake, Roberto Vivancos, Paula B. Blomquist, Andre Charlett, and Gavin Dabrera. 2021. The spatio-temporal distribution of COVID-19 infection in England between January and June 2020. *Epidemiology and Infection* 149 (2021), e73. https://doi.org/10.1017/S0950268821000534

[21] Edward Gan and Peter Bailis. 2017. Scalable Kernel Density Classification via Threshold-Based Pruning. In *ACM SIGMOD*. 945–959.

[22] Song Gao. 2015. Spatio-Temporal Analytics for Exploring Human Mobility Patterns and Urban Dynamics in the Mobile Age. *Spatial Cognition & Computation* 15, 2 (2015), 86–114. https://doi.org/10.1080/13875868.2014.984300

[23] Thanaa M. Ghanem, Moustafa A. Hammad, Mohamed F. Mokbel, Walid G. Aref, and Ahmed K. Elmagarmid. 2007. Incremental Evaluation of Sliding-Window Queries over Data Streams. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 57–72. https://doi.org/10.1109/TKDE.2007.250585

[24] Xiangyang Gou, Long He, Yinda Zhang, Ke Wang, Xilai Liu, Tong Yang, Yi Wang, and Bin Cui. 2020. Sliding Sketches: A Framework using Time Zones for Data Stream Processing in Sliding Windows. In *SIGKDD*. ACM, 1015–1025. https://doi.org/10.1145/3394486.3403144

[25] Alexander G. Gray and Andrew W. Moore. 2003. Nonparametric Density Estimation: Toward Computational Tractability. In *SDM*. 203–211.

[26] Timothy Hart and Paul Zandbergen. 2014. Kernel density estimation and hotspot mapping: examining the influence of interpolation method, grid cell size, and bandwidth on crime forecasting. *Policing: An International Journal of Police Strategies and Management* 37 (2014), 305–323. https://doi.org/10.3390/s80603601

[27] Alexander Hohl, Eric Delmelle, Wenwu Tang, and Irene Casas. 2016. Accelerating the discovery of space-time patterns of infectious diseases using parallel computing. *Spatial and Spatio-temporal Epidemiology* 19 (2016), 10 – 20. https://doi.org/10.1016/j.sste.2016.05.002

[28] Yujie Hu, Fahui Wang, Cecile Guin, and Haojie Zhu. 2018. A spatio-temporal kernel density estimation framework for predictive crime hotspot mapping and evaluation. *Applied Geography* 99 (2018), 89 – 97. https://doi.org/10.1016/j.apgeog.2018.08.001

[29] Youngok Kang, Nahye Cho, and Serin Son. 2018. Spatiotemporal characteristics of elderly population's traffic accidents in Seoul using space-time cube and space-time kernel density estimation. *PLOS ONE* 13, 5 (05 2018), 1–17. https://doi.org/10.1371/journal.pone.0196845

[30] Jay Lee, Junfang Gong, and Shengwen Li. 2017. Exploring spatiotemporal clusters based on extended kernel estimation methods. *Int. J. Geogr. Inf. Sci.* 31, 6 (2017), 1154–1177. https://doi.org/10.1080/13658816.2017.1287371

[31] Jin Li, David Maier, Kristin Tufte, Vassilis Papadimos, and Peter A. Tucker. 2005. Semantics and Evaluation Techniques for Window Aggregates in Data Streams. In *SIGMOD*. 311–322. https://doi.org/10.1145/1066157.1066193

[32] Yunxuan Li, Mohamed Abdel-Aty, Jinghui Yuan, Zeyang Cheng, and Jian Lu. 2020. Analyzing traffic violation behavior at urban intersections: A spatio-temporal kernel density estimation approach using automated enforcement system data. *Accident Analysis & Prevention* 141 (2020), 105509. https://doi.org/10.1016/j.aap.2020.105509

[33] Ross Maciejewski, Ryan Hafen, Stephen Rudolph, Stephen G. Larew, Michael A. Mitchell, William S. Cleveland, and David S. Ebert. 2011. Forecasting Hotspots - A Predictive Analytics Approach. *IEEE Transactions on Visualization and Computer Graphics* 17, 4 (2011), 440–453. https://doi.org/10.1109/TVCG.2010.82

[34] Ross Maciejewski, Stephen Rudolph, Ryan Hafen, Ahmad M. Abusalah, Mohamed Yakout, Mourad Ouzzani, William S. Cleveland, Shaun J. Grannis, and David S. Ebert. 2010. A Visual Analytics Approach to Understanding Spatiotemporal Hotspots. *IEEE Transactions on Visualization and Computer Graphics* 16, 2 (2010), 205–220. https://doi.org/10.1109/TVCG.2009.100

[35] Ross Maciejewski, Stephen Rudolph, Ryan Hafen, Ahmad M. Abusalah, Mohamed Yakout, Mourad Ouzzani, William S. Cleveland, Shaun J. Grannis, Michael Wade, and David S. Ebert. 2008. Understanding syndromic hotspots - a visual analytics approach. In *VAST*. 35–42. https://doi.org/10.1109/VAST.2008.4677354

[36] Andrew W. Moore. 2000. The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data. In *UAI*. 397–405.

[37] Norihiko Muroga, Yoko Hayama, Takehisa Yamamoto, Akihiro Kurogi, Tomoyuki Tsuda, and Toshiyuki Tsutsui. 2011. The 2010 Foot-and-Mouth Disease Epidemic in Japan. *The Journal of veterinary medical science / the Japanese Society of Veterinary Science* 74 (11 2011), 399–404. https://doi.org/10.1292/jvms.11-0271

[38] Tomoki Nakaya and Keiji Yano. 2010. Visualising Crime Clusters in a Space-time Cube: An Exploratory Data-analysis Approach Using Space-time Kernel Density Estimation and Scan Statistics. *Transactions in GIS* 14, 3 (2010), 223–239. https://doi.org/10.1111/j.1467-9671.2010.01194.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9671.2010.01194.x

[39] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[40] Jeff M. Phillips. 2013. ϵ-Samples for Kernels. In *SODA*. 1622–1632. https://doi.org/10.1137/1.9781611973105.116

[41] Jeff M. Phillips and Wai Ming Tai. 2018. Improved Coresets for Kernel Density Estimates. In *SODA*. 2718–2727. https://doi.org/10.1137/1.9781611975031.173

[42] Jeff M. Phillips and Wai Ming Tai. 2018. Near-Optimal Coresets of Kernel Density Estimates. In *SOCG*. 66:1–66:13. https://doi.org/10.4230/LIPIcs.SoCG.2018.66

[43] Alina Ristea, Mohammad Al Boni, Bernd Resch, Matthew S. Gerber, and Michael Leitner. 2020. Spatial crime distribution and prediction for sporting events using social media. *Int. J. Geogr. Inf. Sci.* 34, 9 (2020), 1708–1739. https://doi.org/10.1080/13658816.2020.1719495

[44] Benjamin Romano and Zhe Jiang. 2017. Visualizing Traffic Accident Hotspots Based on Spatial-Temporal Network Kernel Density Estimation. In *SIGSPATIAL*. ACM, 98:1–98:4. https://doi.org/10.1145/3139958.3139981

[45] H. Samet. 2006. *Foundations of Multidimensional and Metric Data Structures*.

[46] Erik Saule, Dinesh Panchananam, Alexander Hohl, Wenwu Tang, and Eric Delmelle. 2017. Parallel Space-Time Kernel Density Estimation. In *ICPP*. 483–492. https://doi.org/10.1109/ICPP.2017.57

[47] Òscar Saladié, Edgar Bustamante, and Aaron Gutiérrez. 2020. COVID-19 lockdown and reduction of traffic accidents in Tarragona province, Spain. *Transportation Research Interdisciplinary Perspectives* 8 (2020), 100218. https://doi.org/10.1016/j.trip.2020.100218

[48] Kanat Tangwongsan, Martin Hirzel, and Scott Schneider. 2019. Optimal and General Out-of-Order Sliding-Window Aggregation. *PVLDB* 12, 10 (2019), 1167–1180. https://doi.org/10.14778/3339490.3339499

[49] Kanat Tangwongsan, Martin Hirzel, and Scott Schneider. 2021. In-order sliding-window aggregation in worst-case constant time. *VLDB J.* 30, 6 (2021), 933–957. https://doi.org/10.1007/s00778-021-00668-3

[50] Kanat Tangwongsan, Martin Hirzel, Scott Schneider, and Kun-Lung Wu. 2015. General Incremental Sliding-Window Aggregation. *PVLDB* 8, 7 (2015), 702–713. https://doi.org/10.14778/2752939.2752940

[51] Yufei Tao and Dimitris Papadias. 2006. Maintaining Sliding Window Skylines on Data Streams. *IEEE Trans. Knowl. Data Eng.* 18, 2 (2006), 377–391. https://doi.org/10.1109/TKDE.2006.48

[52] Lalita Thakali, Tae J. Kwon, and Liping Fu. 2015. Identification of crash hotspots using kernel density estimation and kriging methods: a comparison. *Journal of Modern Transportation* 23, 2 (01 Jun 2015), 93–106. https://doi.org/10.1007/s40534-015-0068-0

[53] Álvaro Villalba, Josep Lluis Berral, and David Carrera. 2019. Constant-Time Sliding Window Framework with Reduced Memory Footprint and Efficient Bulk Evictions. *IEEE Trans. Parallel Distributed Syst.* 30, 3 (2019), 486–500. https://doi.org/10.1109/TPDS.2018.2868960

[54] Pauli Virtanen, Ralf Gommers, and Travis E. et al. Oliphant. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[55] Xiang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin, and Zengfeng Huang. 2016. SKYPE: Top-k Spatial-keyword Publish/Subscribe Over Sliding Window. *PVLDB* 9, 7 (2016), 588–599. https://doi.org/10.14778/2904483.2904490

[56] Qiujun Wei, Jiangfeng She, Shuhua Zhang, and Jinsong Ma. 2018. Using Individual GPS Trajectories to Explore Foodscape Exposure: A Case Study in Beijing Metropolitan Area. *International journal of environmental research and public health* 15 (02 2018). https://doi.org/10.3390/ijerph15030405

[57] Kun Xie, Kaan Ozbay, Abdullah Kurkcu, and Hong Yang. 2017. Analysis of Traffic Crashes Involving Pedestrians Using Big Data: Investigation of Contributing Factors and Identification of Hotspots. *Risk Analysis* 37, 8 (2017), 1459–1476. https://EconPapers.repec.org/RePEc:wly:riskan:v:37:y:2017:i:8:p:1459-1476

[58] Li Xu, Mei-Po Kwan, Sara McLafferty, and Shaowen Wang. 2017. Predicting demand for 311 non-emergency municipal services: An adaptive space-time kernel approach. *Applied Geography* 89 (2017), 133–141. https://doi.org/10.1016/j.apgeog.2017.10.012

[59] P. Zezula, G. Amato, V. Dohnal, and M. Batko. 2006. *Similarity Search: The Metric Space Approach*. Springer US. https://books.google.com.hk/books?id=KTkWXsiPXR4C

[60] Chao Zhang, Reza Akbarinia, and Farouk Toumani. 2021. Efficient Incremental Computation of Aggregations over Sliding Windows. In *SIGKDD*. ACM, 2136–2144. https://doi.org/10.1145/3447548.3467360

[61] Zhijie Zhang, Dongmei Chen, Wenbao Liu, Jeffrey Racine, Seng-Huat Ong, Yue Chen, Genming Zhao, and Qingwu Jiang. 2011. Nonparametric Evaluation of Dynamic Disease Risk: A Spatio-Temporal Kernel Approach. *PloS one* 6 (03 2011), e17381. https://doi.org/10.1371/journal.pone.0017381

[62] Xiangyu Zhao and Jiliang Tang. 2018. Crime in Urban Areas: A Data Mining Perspective. *SIGKDD Explorations* 20, 1 (2018), 1–12. https://doi.org/10.1145/3229329.3229331

[63] Yan Zheng, Jeffrey Jestes, Jeff M. Phillips, and Feifei Li. 2013. Quality and efficiency for kernel density estimates in large data. In *SIGMOD*. 433–444.

[64] Yan Zheng, Yi Ou, Alexander Lex, and Jeff M. Phillips. 2021. Visualization of Big Spatial Data Using Coresets for Kernel Density Estimates. *IEEE Trans. Big Data* 7, 3 (2021), 524–534. https://doi.org/10.1109/TBDATA.2019.2913655

[65] Yan Zheng and Jeff M. Phillips. 2015. L∞ Error and Bandwidth Selection for Kernel Density Estimates of Large Data. In *SIGKDD*. 1533–1542. https://doi.org/10.1145/2783258.2783357

[66] Zhengyi Zhou and David S. Matteson. 2015. Predicting Ambulance Demand: a Spatio-Temporal Kernel Approach. In *SIGKDD*. 2297–2303. https://doi.org/10.1145/2783258.2788570

[67] Haojie Zhu and Fahui Wang. 2021. An agent-based model for simulating urban crime with improved daily routines. *Computers, Environment and Urban Systems* 89 (2021), 101680. https://doi.org/10.1016/j.compenvurbsys.2021.101680

[68] Rui Zhu, Bin Wang, Xiaochun Yang, Baihua Zheng, and Guoren Wang. 2017. SAP: Improving Continuous Top-K Queries Over Streaming Data. *IEEE Trans. Knowl. Data Eng.* 29, 6 (2017), 1310–1328. https://doi.org/10.1109/TKDE.2017.2662236