A Fast and Accurate Block Compression Solution for Spatiotemporal Kernel Density Visualization

Yue Zhong
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
2310273008@email.szu.edu.cn

Tsz Nam Chan*
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
edisonchan@szu.edu.cn

Leong Hou U
Department of Computer and Information Science, University of Macau
Macao, China
ryanlhu@um.edu.mo

Dingming Wu
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
dingming@szu.edu.cn

Wei Tu
Ruisheng Wang
School of Architecture and Urban Planning, Shenzhen University
Shenzhen, China
{tuwei,ruiswang}@szu.edu.cn

Joshua Zhexue Huang
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
zx.huang@szu.edu.cn

Abstract

Spatiotemporal Kernel Density Visualization (STKDV) has been widely used across various domains in geospatial analysis, e.g., urban planning, traffic/traffic accident hotspot analysis, crime hotspot analysis, and disease spread modeling. However, STKDV is a computationally expensive tool, which has been complained by many domain experts. Although many recent solutions, including the sliding-window-based solution (SWS) and the prefix-matrix-based solution (PREFIX), have been proposed for improving the efficiency of generating an exact STKDV, these solutions still cannot be scalable to handle large-scale location datasets. To tackle this efficiency issue, we propose the pioneering block compression solution, called COMP, which can compress (or represent) a location dataset by a small amount of blocks. By combining COMP with the existing exact solutions, i.e., SWS and PREFIX, we show that COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can generate approximate STKDV with an ϵ-absolute error guarantee based on properly tuning the block size. Experimental results on four large-scale location datasets (up to 6.782 million data points) also verify that COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can achieve speedups of 4.1x to 677.16x and 1.45x to 143.52x compared with SWS and PREFIX, respectively, without degrading the visualization results. The code of this paper can be found in https://github.com/YovelaZ/COMP.

CCS Concepts

• **Information systems** → **Geographic information systems**.

Keywords

STKDV; block compression; approximation; efficiency

ACM Reference Format:
Yue Zhong, Tsz Nam Chan, Leong Hou U, Dingming Wu, Wei Tu, Ruisheng Wang, and Joshua Zhexue Huang. 2025. A Fast and Accurate Block Compression Solution for Spatiotemporal Kernel Density Visualization. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3736821

1 Introduction

Heatmap [42] is a powerful visualization/data analysis tool that has been extensively used for understanding the spatial distribution of data points. Among most of the heatmap tools, Kernel Density Visualization (KDV) [12, 15, 18, 23, 48] is commonly used across various domains in geospatial analysis, including urban planning [32, 46], traffic/traffic accident hotspot analysis [11, 45], crime hotspot analysis [27, 39], and disease spread modeling [22, 29]. Figure 1 shows an example for using KDV to visualize hotspots using the San Francisco 311-call dataset [7]. Observe that the red region and the blue region represent the 311-call hotspot and coldspot, respectively, in San Francisco. Due to the importance of KDV, many contemporary software packages have been developed for supporting this tool, including Scikit-learn [35], ArcGIS [1], QGIS [6], and Seaborn [8].

(a) San Francisco 311 calls (b) Heatmap

Figure 1: Generating a heatmap (based on KDV) for the San Francisco 311-call dataset.

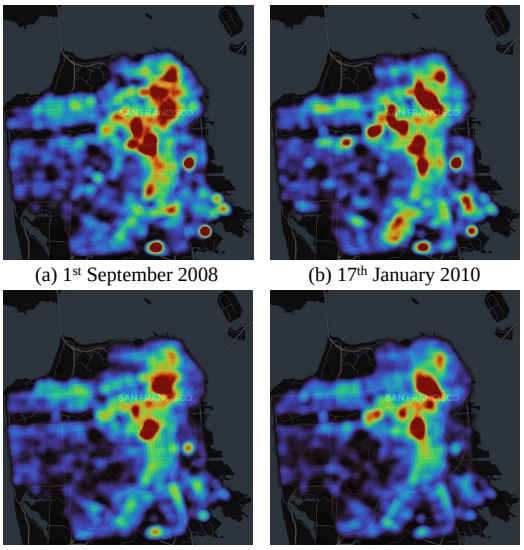
However, a severe limitation of using KDV is that this tool solely takes the location of each data point into account for generating

*Corresponding author.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3736821

a heatmap, which disregards its occurrence time, leading to inaccurate data analysis (or inaccurate interpretation of visualization). Observe from Figure 2 that hotspots can dramatically change with respect to different timestamps. For example, there are many 311-call hotspot regions on 17th January 2010, while there are relatively a few 311-call hotspot regions on 7th October 2011. As such, simply using KDV (see Figure 1b) cannot provide correct hotspot interpretation during a period of time.



(a) 1st September 2008 (b) 17th January 2010

(c) 7th October 2011 (d) 26th June 2013

Figure 2: Generating time-dependent heatmaps (based on STKDV) for the San Francisco 311-call dataset.

To address this limitation, domain experts [27, 30] propose an advanced tool, called Spatiotemporal Kernel Density Visualization (STKDV), which generates time-dependent heatmaps (see Figure 2) by considering both the spatial and temporal coordinates of each location data point (e.g., yellow points in Figure 1a). Although this tool has been well recognized (and extensively used) by domain experts (e.g., [22, 27, 29, 30]), this tool suffers from high response time. Consider a $X \times Y$-resolution plane, T timestamps (e.g., $T = 4$ in Figure 2), and n data points. Generating STKDV takes $O(XYTn)$ time. Using the Montgomery dataset [9] (with 1.83 million data points) as an example, generating a 1280×960-resolution STKDV with 32 timestamps takes 71.96 trillion operations, which cannot be scalable to high-resolution sizes and large datasets.

Due to the efficiency issues of STKDV, various efficient solutions have been recently proposed to reduce the time complexity for using this tool, including the sliding-window-based solution (SWS) [14] and the prefix-matrix-based solution (PREFIX) [16]. Among these two solutions, PREFIX is the state-of-the-art one, which significantly reduces the time complexity of generating STKDV to $O(XYT + Yn)$ without theoretically degrading the visualization quality. Despite this, the time complexity of PREFIX depends on the term Yn, which can still be large (especially for million-scale datasets) and is still dominated by n. Furthermore, many data points in a location dataset are spatiotemporally close to each other (see Figure 3), which can possibly be represented by a compact dataset (i.e., with a small dataset size) without significantly losing information (i.e., degrading the visualization quality of STKDV). Therefore, a key research question arises: *can we develop an effective data compression method for a location dataset so that generating STKDV (with an existing solution) in a returned compact dataset retains the similar visualization quality?*

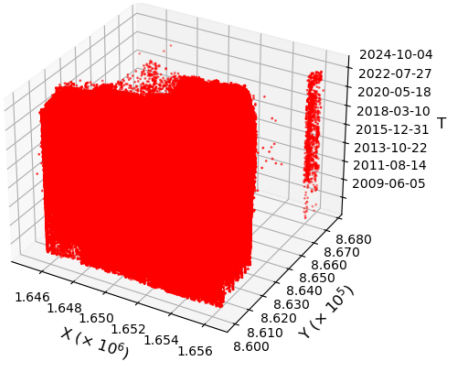


Figure 3: The distribution of San Francisco 311-call data points.

To provide a definite answer to this question, we propose a new block compression solution, called COMP, which, to the best of our knowledge, is the first solution that can successfully compress (or represent) a location dataset by a small amount of blocks. Once we combine COMP with the existing exact solutions, namely SWS and PREFIX, we show that COMP$_{\text{SWS}}$ and COMP$_{\text{PREFIX}}$ can achieve a non-trivial ϵ-absolute error guarantee for generating STKDV by rigorously tuning the block size based on the ϵ parameter. Our experimental results with four large-scale location datasets (with up to 6.782 million data points) verify that COMP$_{\text{SWS}}$ and COMP$_{\text{PREFIX}}$ can achieve speedups of 4.1x to 677.16x and 1.45x to 143.52x compared with the existing SWS and PREFIX solutions, respectively, without degrading the visualization results.

The rest of the paper is structured as follows. We first formally define the STKDV problem and outline the existing solutions (i.e., SWS and PREFIX) in Section 2. Then, we discuss our block compression solution (i.e., COMP) in Section 3. Next, we provide the experimental evaluation in Section 4. After that, we review the related studies in Section 5. Lastly, we conclude this paper in Section 6. Appendix can be found in Section 7.

2 Preliminaries

In this section, we first formally define the STKDV problem in Section 2.1. Then, we discuss two existing solutions in Section 2.2.

2.1 Problem Statement

In order to generate STKDV (see Figure 2) with a location dataset (see Figure 1a), we need to color each pixel-timestamp pair $(\mathbf{q}, t_i)$ based on the spatiotemporal kernel density function $\mathcal{F}_P(\mathbf{q}, t_i)$, which is formally stated in Definition 1.

Definition 1. *Given a resolution size $X \times Y$, T timestamps (i.e., t_1, t_2,..., t_T), and a location dataset $P = \{(\mathbf{p}_1, t_{\mathbf{p}_1}), (\mathbf{p}_2, t_{\mathbf{p}_2}), ..., (\mathbf{p}_n, t_{\mathbf{p}_n})\}$ with size n, we need to compute the spatiotemporal kernel density function $\mathcal{F}_P(\mathbf{q}, t_i)$ for each pixel-timestamp pair $(\mathbf{q}, t_i)$.*

$$\mathcal{F}_P(\mathbf{q}, t_i) = \frac{1}{n} \sum_{(\mathbf{p}, t_{\mathbf{p}}) \in P} K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}}) \quad (1)$$

where $K_{space}(\mathbf{q}, \mathbf{p})$ and $K_{time}(t_i, t_{\mathbf{p}})$ are the spatial and temporal kernel functions, which are summarized in Table 1.

Table 1: Some widely used spatial kernel functions ($K_{\text{space}}(\mathbf{q},\mathbf{p})$) and temporal kernel functions ($K_{\text{time}}(t_i,t_{\mathbf{p}})$), where b_σ and b_τ denote the spatial bandwidth and the temporal bandwidth, respectively.

Kernel	$K_{\text{space}}(\mathbf{q},\mathbf{p})$	$K_{\text{time}}(t_i,t_{\mathbf{p}})$	References
Triangular	$\begin{cases}1-\frac{1}{b_\sigma}\|\|\mathbf{q}-\mathbf{p}\|\|_2 & \text{if } \|\|\mathbf{q}-\mathbf{p}\|\|_2 \le b_\sigma\\ 0 & \text{otherwise}\end{cases}$	$\begin{cases}1-\frac{1}{b_\tau}\|t_i-t_{\mathbf{p}}\| & \text{if } \|t_i-t_{\mathbf{p}}\| \le b_\tau\\ 0 & \text{otherwise}\end{cases}$	[14, 33]
Epanechnikov	$\begin{cases}\frac{3}{4}\cdot\left(1-\frac{1}{b_\sigma^2}\|\|\mathbf{q}-\mathbf{p}\|\|_2^2\right) & \text{if } \|\|\mathbf{q}-\mathbf{p}\|\|_2 \le b_\sigma\\ 0 & \text{otherwise}\end{cases}$	$\begin{cases}\frac{3}{4}\cdot\left(1-\frac{1}{b_\tau^2}(t_i-t_{\mathbf{p}})^2\right) & \text{if } \|t_i-t_{\mathbf{p}}\| \le b_\tau\\ 0 & \text{otherwise}\end{cases}$	[22, 27]
Quartic	$\begin{cases}\frac{15}{16}\cdot\left(1-\frac{1}{b_\sigma^2}\|\|\mathbf{q}-\mathbf{p}\|\|_2^2\right)^2 & \text{if } \|\|\mathbf{q}-\mathbf{p}\|\|_2 \le b_\sigma\\ 0 & \text{otherwise}\end{cases}$	$\begin{cases}\frac{15}{16}\cdot\left(1-\frac{1}{b_\tau^2}(t_i-t_{\mathbf{p}})^2\right)^2 & \text{if } \|t_i-t_{\mathbf{p}}\| \le b_\tau\\ 0 & \text{otherwise}\end{cases}$	[16, 30]

2.2 Existing Solutions

Here, we discuss the core ideas of two existing solutions, which are (1) sliding-window-based solution (SWS) [14] and (2) prefix-matrix-based solution (PREFIX) [16], that can improve the efficiency for generating STKDV.

Sliding-window-based solution (SWS). Observe from Table 1 that only those data points $(\mathbf{p}_j, t_{\mathbf{p}_j})$ with $|t_i - t_{\mathbf{p}_j}| \le b_\tau$ can contribute to the spatiotemporal kernel density function (see Equation 1). Therefore, Chan et al. [14] propose to first maintain a sliding window $W(t_i)$ (i.e., the orange one in Figure 4) in the time-axis. With this sliding window, they can represent $\mathcal{F}_P(\mathbf{q},t_i)$ based on the statistical terms $S^{(u)}_{W(t_i)}(\mathbf{q})$ (where u is the positive integer that depends on the temporal kernel function). Using the Epanechnikov function as an example of the temporal kernel (see Table 1), $\mathcal{F}_P(\mathbf{q},t_i)$ can be decomposed as follows.

$$\mathcal{F}_P(\mathbf{q},t_i) = \frac{3}{4n}\left(\left(1-\frac{t_i^2}{b_\tau^2}\right)S^{(0)}_{W(t_i)}(\mathbf{q}) + \frac{2t_i}{b_\tau^2}S^{(1)}_{W(t_i)}(\mathbf{q}) - \frac{1}{b_\tau^2}S^{(2)}_{W(t_i)}(\mathbf{q})\right) \tag{2}$$

where

$$S^{(u)}_{W(t_i)}(\mathbf{q}) = \sum_{(\mathbf{p},t_{\mathbf{p}})\in W(t_i)} t_{\mathbf{p}}^u \cdot K_{\text{space}}(\mathbf{q},\mathbf{p}) \tag{3}$$

Then, they propose to avoid the redundant computation (i.e., avoid scanning the green circles in $W(t_i)\cap W(t_{i+1})$ in Figure 4) between two sliding windows by utilizing the following property.

$$S^{(u)}_{W(t_{i+1})}(\mathbf{q}) = S^{(u)}_{W(t_i)}(\mathbf{q}) + S^{(u)}_{I(W(t_i),W(t_{i+1}))}(\mathbf{q}) - S^{(u)}_{D(W(t_i),W(t_{i+1}))}(\mathbf{q}) \tag{4}$$

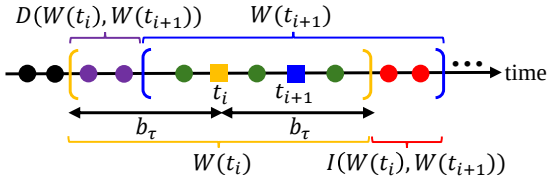


Figure 4: Illustration of SWS, where those circles denote the data points $(\mathbf{p}_j, t_{\mathbf{p}_j})$ in a location dataset and the squares represent the pixel-timestamp pairs, i.e., $(\mathbf{q},t_i)$ and $(\mathbf{q},t_{i+1})$.

With this approach, they further show that evaluating the density values with T timestamps for a pixel $\mathbf{q}$ is $O(T+n)$, which indicates that generating STKDV (with $X\times Y$ pixels) is $O(XY(T+n))$ time.

Prefix-matrix-based solution (PREFIX). Instead of considering each pixel independently in SWS, Chan et al. [16] aim to first maintain the prefix-matrix with respect to all $X\times Y$ pixels for each end point in the time axis (see Figure 5). Here, we let t_e be one of the end points. We have $P(t_e) = \{(\mathbf{p},t_{\mathbf{p}}) \in P : t_{\mathbf{p}} \le t_e\}$. As an example, $P(t_i+b_\tau)$ denotes all black data points that are just before t_i+b_τ in Figure 5. Once they have all prefix-matrices, they then evaluate each $S^{(u)}_{W(t_i)}(\mathbf{q})$ based on the following equation and compute $\mathcal{F}_P(\mathbf{q},t_i)$ (see Equation 2) for all pixels $\mathbf{q}$.

$$S^{(u)}_{W(t_i)}(\mathbf{q}) = S^{(u)}_{P(t_i+b_\tau)}(\mathbf{q}) - S^{(u)}_{P(t_i-b_\tau)}(\mathbf{q}) \tag{5}$$

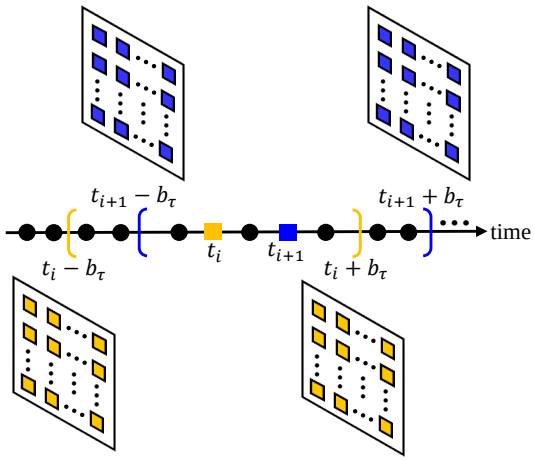


Figure 5: Illustration of PREFIX, where the orange planes (the blue planes) denote the prefix matrices, $S^{(u)}_{P(t_i-b_\tau)}(\mathbf{q})$ and $S^{(u)}_{P(t_i+b_\tau)}(\mathbf{q})$ ($S^{(u)}_{P(t_{i+1}-b_\tau)}(\mathbf{q})$ and $S^{(u)}_{P(t_{i+1}+b_\tau)}(\mathbf{q})$), for the end points, t_i-b_τ and t_i+b_τ ($t_{i+1}-b_\tau$ and $t_{i+1}+b_\tau$).

By adopting this approach, they show that the time complexity of generating STKDV with T timestamps can be further reduced to $O(XYT+Yn)$.

3 Our Solution

In this section, we first discuss the core idea of block compression in Section 3.1. Then, we illustrate how to tune the proper block size in Section 3.2. Lastly, we provide the block compression algorithm in Section 3.3.

3.1 Core Idea of Block Compression

Observe from Figure 3 that many red data points are spatiotemporally close to each other. Therefore, we ask a question. *Can we compress this dataset so that generating the STKDV for this compact dataset can provide similar visualization quality compared with the STKDV for the original dataset?* To provide an affirmative answer to this question, we propose to adopt a set of blocks $\mathcal{S}$ to cover a dataset (see Figure 6). Hence, we can represent $\mathcal{F}_P(\mathbf{q},t_i)$ (see

Equation 1) based on $\mathcal{S}$.

$$\mathcal{F}_P(\mathbf{q}, t_i) = \frac{1}{n} \sum_{\mathcal{B} \in \mathcal{S}} \sum_{(\mathbf{p}, t_\mathbf{p}) \in \mathcal{B}} K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_\mathbf{p}) \quad (6)$$

where $\mathcal{B}$ denotes each block that covers its data points. As an example, the block $\mathcal{B}$ in Figure 6 covers four data points (the black spheres), i.e., $\mathbb{C}(\mathcal{B}) = 4$.

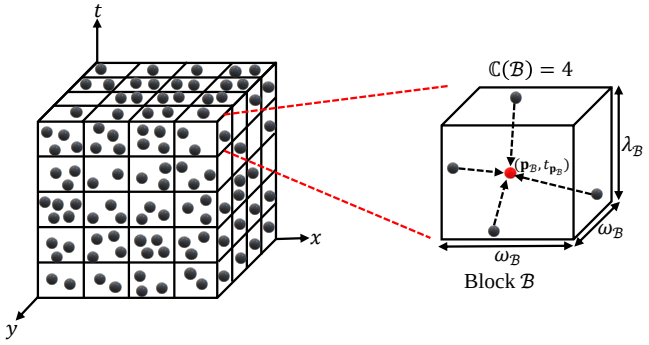


Figure 6: Illustration of the core idea of block compression, where each black sphere denotes the location data point $(\mathbf{p}, t_\mathbf{p})$, the red sphere denotes the center point $(\mathbf{p}_\mathcal{B}, t_{\mathbf{p}_\mathcal{B}})$ of the block $\mathcal{B}$ (with size $\omega_\mathcal{B} \times \omega_\mathcal{B} \times \lambda_\mathcal{B}$), and $\mathcal{B}$ covers four location data points (i.e., $\mathbb{C}(\mathcal{B}) = 4$).

Observe that we aim to use the center $(\mathbf{p}_\mathcal{B}, t_{\mathbf{p}_\mathcal{B}})$ (i.e., the red sphere) to represent all data points (i.e., the black spheres) in each block $\mathcal{B}$. Based on this concept, we have the approximate spatiotemporal kernel density function $A_\mathcal{S}(\mathbf{q}, t_i)$ based on $\mathcal{S}$.

$$A_\mathcal{S}(\mathbf{q}, t_i) = \frac{1}{n} \sum_{\mathcal{B} \in \mathcal{S}} \mathbb{C}(\mathcal{B}) \cdot K_{\text{space}}(\mathbf{q}, \mathbf{p}_\mathcal{B}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_\mathcal{B}}) \quad (7)$$

In order to ensure that this block compression approach does not degrade the visualization quality, the approximate value $A_\mathcal{S}(\mathbf{q}, t_i)$ should only deviate from the exact value $\mathcal{F}_P(\mathbf{q}, t_i)$ by a small absolute error ϵ (see Definition 2).

Definition 2. *Given a location dataset P and an absolute error ϵ, we need to construct a set of blocks $\mathcal{S}$ in order to generate an approximate STKDV based on $A_\mathcal{S}(\mathbf{q}, t_i)$, where*

$$|A_\mathcal{S}(\mathbf{q}, t_i) - \mathcal{F}_P(\mathbf{q}, t_i)| \leq \epsilon \quad (8)$$

3.2 How to Tune the Proper Block Size?

One major challenge for using the block compression approach is that it is non-trivial to tune the proper block size $\omega_\mathcal{B} \times \omega_\mathcal{B} \times \lambda_\mathcal{B}$ (see Figure 6). Suppose that the block size is very large. Each block $\mathcal{B}$ can cover more data points (with large $\mathbb{C}(\mathcal{B})$), which can achieve the fast evaluation of $A_\mathcal{S}(\mathbf{q}, t_i)$ (due to a smaller number of blocks in $\mathcal{S}$). Despite this, it can cause an inaccurate value of $A_\mathcal{S}(\mathbf{q}, t_i)$, which violates the error guarantee (see Definition 2). In contrast, a small block size can easily achieve the error guarantee, while it provides a large number of blocks in $\mathcal{S}$ (resulting in the slow evaluation of $A_\mathcal{S}(\mathbf{q}, t_i)$). Therefore, we ask a question. *Can we find a set of blocks $\mathcal{S}$ so that (1) the error guarantee (Definition 2) is fulfilled and (2) the block size $\omega_\mathcal{B} \times \omega_\mathcal{B} \times \lambda_\mathcal{B}$ is the largest?*

To answer the above question, we analyze Equation 8. Consider each data point $(\mathbf{p}, t_\mathbf{p})$ in any block $\mathcal{B}$ (with the center $(\mathbf{p}_\mathcal{B}, t_{\mathbf{p}_\mathcal{B}})$). Once $|K_{\text{space}}(\mathbf{q}, \mathbf{p}_\mathcal{B}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_\mathcal{B}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_\mathbf{p})| \leq \epsilon$ for any pixel-timestamp pair $(\mathbf{q}, t_i)$, we can conclude that Equation 8 holds (see Lemma 1). The proof of this lemma can be found in Section 7.1 of Appendix.

Lemma 1. *Given any block $\mathcal{B}$ (with its center $(\mathbf{p}_\mathcal{B}, t_{\mathbf{p}_\mathcal{B}})$) and any data point $(\mathbf{p}, t_\mathbf{p})$ in $\mathcal{B}$, the error guarantee (i.e., Equation 8) holds if the following property holds for any pixel-timestamp pair $(\mathbf{q}, t_i)$.*

$$|K_{\text{space}}(\mathbf{q}, \mathbf{p}_\mathcal{B}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_\mathcal{B}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_\mathbf{p})| \leq \epsilon \quad (9)$$

With Lemma 1, instead of considering all data points in P and all blocks in $\mathcal{S}$ (see Equation 8), we can simplify this constraint by only considering those data points $(\mathbf{p}, t_\mathbf{p})$ in a single block $\mathcal{B}$ (see Equation 9).

In order to fulfill the constraint of Equation 9, we first consider the property of spatial and temporal kernel functions, i.e., $K_{\text{space}}(\mathbf{q}, \mathbf{p})$ and $K_{\text{time}}(t_i, t_\mathbf{p})$, respectively. Here, after we let $x = ||\mathbf{q} - \mathbf{p}||_2$ (or $x = |t_i - t_\mathbf{p}|$), we can have another function $\mathcal{K}(x)$ so that $\mathcal{K}(x) = K_{\text{space}}(\mathbf{q}, \mathbf{p})$ (or $\mathcal{K}(x) = K_{\text{time}}(t_i, t_\mathbf{p})$). Using the Epanechnikov kernel as an example (see Table 1), we have (by setting $b = b_\sigma$ and $b = b_\tau$ for spatial kernel and temporal kernel, respectively)

$$\mathcal{K}(x) = \frac{3}{4} \cdot \begin{cases} 1 - \frac{x^2}{b^2} & \text{if } x \leq b \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

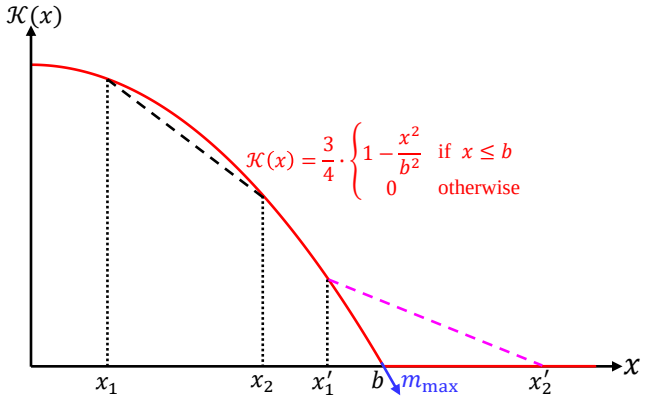


Figure 7: Illustration of $\mathcal{K}(x)$ using the Epanechnikov kernel, where $m_{\max}$ denotes the maximum absolute value of the slope of $\mathcal{K}(x)$ for all x.

Figure 7 depicts the $\mathcal{K}(x)$ function, which corresponds to the Epanechnikov kernel, with respect to x. Observe that the absolute value of the slope between any two points, e.g., the black dashed line that connects $(x_1, \mathcal{K}(x_1))$ and $(x_2, \mathcal{K}(x_2))$ and the pink dashed line that connects $(x_1', \mathcal{K}(x_1'))$ and $(x_2', \mathcal{K}(x_2'))$, must be smaller than $m_{\max}$ (where $m_{\max}$ denotes the maximum absolute value of the slopes of $\mathcal{K}(x)$ for all x). Hence, we have the following Lemma 2. The proof of this lemma can be found in Section 7.2 of Appendix.

Lemma 2. *Given any two positive values, x_1 and x_2, in the x-axis and $\mathcal{K}(x)$ that corresponds to any kernel function in Table 1, we have*

$$\left| \frac{\mathcal{K}(x_1) - \mathcal{K}(x_2)}{x_1 - x_2} \right| \leq m_{\max} \quad (11)$$

where $m_{\max} = \max_x \left(\left| \frac{d\mathcal{K}(x)}{dx} \right| \right)$.

Table 2 summarizes the $m_{\max}^{(\sigma)}$ and $m_{\max}^{(\tau)}$ values with respect to the spatial and temporal kernel functions, respectively, in Table 1. We discuss how to derive them in Section 7.3 of Appendix.

Table 2: Summarization of different $m_{\max}^{(\sigma)}$ (for spatial kernels) and $m_{\max}^{(\tau)}$ (for temporal kernels) values.

Kernel	$m_{\max}^{(\sigma)}$	$m_{\max}^{(\tau)}$
Triangular	$\frac{1}{b_\sigma}$	$\frac{1}{b_\tau}$
Epanechnikov	$\frac{3}{2b_\sigma}$	$\frac{3}{2b_\tau}$
Quartic	$\frac{5\sqrt{3}}{6b_\sigma}$	$\frac{5\sqrt{3}}{6b_\tau}$

Based on Lemma 2, we further show that the deviation between two spatial (or temporal) kernel values, i.e., $|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p})|$ (or $|K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{time}}(t_i, t_{\mathbf{p}})|$), is bounded by the parameter of the block size $\omega_{\mathcal{B}}$ (or $\lambda_{\mathcal{B}}$) and $m_{\max}^{(\sigma)}$ (or $m_{\max}^{(\tau)}$), which is stated in Lemma 3. The proof of this lemma can be found in Section 7.4 of Appendix.

Lemma 3. *Given any pixel-timestamp pair $(\mathbf{q}, t_i)$, the block $\mathcal{B}$ with the center $(\mathbf{p}_{\mathcal{B}}, t_{\mathbf{p}_{\mathcal{B}}})$ and the size $\omega_{\mathcal{B}} \times \omega_{\mathcal{B}} \times \lambda_{\mathcal{B}}$, and any data point $(\mathbf{p}, t_{\mathbf{p}})$ in $\mathcal{B}$, the deviations between two spatial kernel values and two temporal kernel values have the following bounds, i.e., Equation 12 and Equation 13, respectively.*

$$|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p})| \le \frac{\sqrt{2} \cdot \omega_{\mathcal{B}} \cdot m_{\max}^{(\sigma)}}{2} \tag{12}$$

$$|K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{time}}(t_i, t_{\mathbf{p}})| \le \frac{\lambda_{\mathcal{B}} \cdot m_{\max}^{(\tau)}}{2} \tag{13}$$

After we have obtained these bounds in Lemma 3, we can conclude in Theorem 1 that the error guarantee (see Definition 2) can be fulfilled (by illustrating how Equation 9 in Lemma 1 is satisfied) if we set the proper block size $\omega_{\mathcal{B}} \times \omega_{\mathcal{B}} \times \lambda_{\mathcal{B}}$ in each kernel function.

Theorem 1. *Given a location dataset P and an absolute error ϵ, we can achieve the absolute error guarantee ϵ in Definition 2 if $\omega_{\mathcal{B}}$ and $\lambda_{\mathcal{B}}$ of each block $\mathcal{B}$ in $\mathcal{S}$ have the following settings.*

(1) $\omega_{\mathcal{B}} = \frac{\sqrt{2} \cdot \epsilon \cdot b_\sigma}{2}$ *and* $\lambda_{\mathcal{B}} = \epsilon \cdot b_\tau$ *using the triangular spatial and temporal kernels.*
(2) $\omega_{\mathcal{B}} = \frac{4\sqrt{2} \cdot \epsilon \cdot b_\sigma}{9}$ *and* $\lambda_{\mathcal{B}} = \frac{8 \cdot \epsilon \cdot b_\tau}{9}$ *using the Epanechnikov spatial and temporal kernels.*
(3) $\omega_{\mathcal{B}} = \frac{16\sqrt{6} \cdot \epsilon \cdot b_\sigma}{75}$ *and* $\lambda_{\mathcal{B}} = \frac{32\sqrt{3} \cdot \epsilon \cdot b_\tau}{75}$ *using the quartic spatial and temporal kernels.*

Proof. In this proof, we focus on (2) (i.e., the Epanechnikov spatial and temporal kernels). Based on the similar concept, we can easily extend this proof for other kernels.

By considering Equation 12 in Lemma 3 and the temporal kernel $K_{\text{time}}(t_i, t_{\mathbf{p}}) > 0$, we have

$$\begin{aligned} &|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})| \\ \le\ & \frac{\sqrt{2} \cdot \omega_{\mathcal{B}} \cdot m_{\max}^{(\sigma)}}{2} K_{\text{time}}(t_i, t_{\mathbf{p}}) \end{aligned} \tag{14}$$

By considering Equation 13 in Lemma 3 and the spatial kernel $K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) > 0$, we also have

$$\begin{aligned} &|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})| \\ \le\ & \frac{\lambda_{\mathcal{B}} \cdot m_{\max}^{(\tau)}}{2} K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \end{aligned} \tag{15}$$

Therefore, we consider the term $|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})|$ in Lemma 1. Based on Equation 14 and Equation 15, we can obtain the following bound.

$$\begin{aligned} &|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})| \\ =\ & |K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}}) \\ +\ & K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})| \\ \le\ & |K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})| \\ +\ & |K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})| \\ \le\ & \frac{\lambda_{\mathcal{B}} \cdot m_{\max}^{(\tau)}}{2} K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) + \frac{\sqrt{2} \cdot \omega_{\mathcal{B}} \cdot m_{\max}^{(\sigma)}}{2} K_{\text{time}}(t_i, t_{\mathbf{p}}) \end{aligned}$$

Since we consider the Epanechnikov spatial and temporal kernels, we have $K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \le \frac{3}{4}$, $K_{\text{time}}(t_i, t_{\mathbf{p}}) \le \frac{3}{4}$, $m_{\max}^{(\sigma)} = \frac{3}{2b_\sigma}$, and $m_{\max}^{(\tau)} = \frac{3}{2b_\tau}$ (see Table 1 and Table 2). Therefore, we have

$$\begin{aligned} &|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})| \\ \le\ & \frac{9\lambda_{\mathcal{B}}}{16b_\tau} + \frac{9\sqrt{2}\omega_{\mathcal{B}}}{16b_\sigma} \end{aligned}$$

Hence, by setting $\frac{9\lambda_{\mathcal{B}}}{16b_\tau} = \frac{\epsilon}{2}$ and $\frac{9\sqrt{2}\omega_{\mathcal{B}}}{16b_\sigma} = \frac{\epsilon}{2}$, i.e., $\lambda_{\mathcal{B}} = \frac{8 \cdot \epsilon \cdot b_\tau}{9}$ and $\omega_{\mathcal{B}} = \frac{4\sqrt{2} \cdot \epsilon \cdot b_\sigma}{9}$, respectively, Equation 9 is fulfilled, which indicates that the error guarantee (see Equation 8) holds (based on Lemma 1). □

3.3 Block Compression Algorithm (COMP)

Once we have determined the block size $\omega_{\mathcal{B}} \times \omega_{\mathcal{B}} \times \lambda_{\mathcal{B}}$, we can establish the set of blocks $\mathcal{S}$ in order to compute $A_{\mathcal{S}}(\mathbf{q}, t_i)$ (see Equation 7).

Algorithm 1 Block Compression Algorithm for Generating Approximate STKDV

```
1:  procedure COMP(P = {(p_1, t_p1), (p_2, t_p2), ..., (p_n, t_pn)}, spatial
    bandwidth b_σ, temporal bandwidth b_τ, error parameter ε)
2:      Compute ω_B and λ_B                                  ▷ Theorem 1
3:      S ← φ
4:      for each (p, t_p) ∈ P do
5:          Identify the correct block B for (p, t_p)
6:          if B ∈ S then
7:              C(B) ← C(B) + 1
8:          else
9:              C(B) ← 1
10:             S ← S ∪ B
11:     Generate STKDV with A_S(q, t_i) (based on S)
```

Algorithm 1 shows the pseudocode of this method, namely COMP. We first find $\omega_{\mathcal{B}}$ and $\lambda_{\mathcal{B}}$ (see line 2) in order to obtain the block size. Then, we assign each data point into the correct block $\mathcal{B}$ for establishing $\mathcal{S}$ (see line 4 to line 10). Note that many blocks $\mathcal{B}$ can have no data point (i.e., $\mathbb{C}(\mathcal{B}) = 0$), which cannot contribute to $A_{\mathcal{S}}(\mathbf{q}, t_i)$. Using Figure 3 as an example, observe that some regions do not contain any data point. Therefore, instead of maintaining all blocks in $\mathcal{S}$ (which can consume huge space and response time), we only keep those blocks $\mathcal{B}$ with $\mathbb{C}(\mathcal{B}) > 0$ in $\mathcal{S}$ (see line 10). After we have obtained $\mathcal{S}$, we can generate approximate STKDV with $A_{\mathcal{S}}(\mathbf{q}, t_i)$ by adopting the existing solutions, including

SWS and PREFIX (see Section 2.2), since $A_{\mathcal{S}}(\mathbf{q}, t_i)$ (see Equation 7) is similar to the original spatiotemporal kernel density function $\mathcal{F}_P(\mathbf{q}, t_i)$ (see Equation 1). Theorem 2 shows the time complexity of COMP.

Theorem 2. *The time complexity of Algorithm 1 is* $O(\mathbb{T}_{\text{ALG}(|\mathcal{S}|)} + n \log |\mathcal{S}|)$, *where* $\mathbb{T}_{\text{ALG}(|\mathcal{S}|)}$ *represents the time complexity of any exact algorithm ALG to compute STKDV with the size* $|\mathcal{S}|$.

Proof. Since we do not maintain all blocks in the set $\mathcal{S}$, we need to use the vector structure (instead of the cube structure in Figure 6), in which each entry stores the position index of the block $\mathcal{B}$ and the block center $(\mathbf{p}_{\mathcal{B}}, t_{\mathbf{p}_{\mathcal{B}}})$, to maintain $\mathcal{S}$. Therefore, for each data point $(\mathbf{p}, t_{\mathbf{p}})$, we first use $O(1)$ time to identify the correct (position index of) block $\mathcal{B}$ (in line 5), due to the same size $\omega_{\mathcal{B}} \times \omega_{\mathcal{B}} \times \lambda_{\mathcal{B}}$ of each block. Then, we need to adopt the binary search method to check whether $\mathcal{B}$ is in $\mathcal{S}$ (line 6 and line 8), which takes $O(\log |\mathcal{S}|)$ time in the worst case. If $\mathcal{B}$ is in $\mathcal{S}$, we need to increase its count value $\mathbb{C}(\mathcal{B})$ (with $O(1)$ time in line 7). Otherwise, we need to set the count value $\mathbb{C}(\mathcal{B})$ of this new block $\mathcal{B}$ to be 1 (with $O(1)$ time in line 9) and insert this block (based on the binary search method) into $\mathcal{S}$ (with $O(\log |\mathcal{S}|)$ time in line 10). Hence, the time complexity of the loop (from line 4 to line 10) is $O(n \log |\mathcal{S}|)$. With this set $\mathcal{S}$, we further adopt any exact algorithm ALG, which takes $O(\mathbb{T}_{\text{ALG}(|\mathcal{S}|)})$ time, for computing STKDV (line 11). Hence, the time complexity of this algorithm is $O(\mathbb{T}_{\text{ALG}(|\mathcal{S}|)} + n \log |\mathcal{S}|)$. □

As a remark, the smaller the size of $\mathcal{S}$ (with $|\mathcal{S}| << n$), the faster the performance compared with existing methods (with $O(\mathbb{T}_{\text{ALG}(n)})$ time).

4 Experimental Evaluation

In this section, we first discuss the experimental setup in Section 4.1. Then, we verify the compression effectiveness of COMP in Section 4.2. Lastly, we test the efficiency performance and accuracy performance of all methods in Section 4.3 and Section 4.4, respectively. Additional experiments can be found in Section 7.5 of Appendix.

4.1 Experimental Setup

We adopt four large-scale location datasets for testing, which are summarized in Table 3. All these datasets are open to public from different governments. We combine our block compression algorithm (a.k.a. COMP) with the state-of-the-art STKDV methods, SWS [14] and PREFIX [16], which are denoted as COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$, respectively, and measure their efficiency performance and accuracy performance against these two existing methods.

Table 3: Datasets.

Name	n	Category	Ref.
Montgomery	1,830,268	Traffic violations	[9]
New York	1,867,735	Traffic accidents	[5]
Chicago	5,286,309	Taxi pick-up locations	[2]
San Francisco	6,782,726	311 calls	[7]

To conduct our experiments, we follow the settings of [14, 16], by choosing the default spatial resolution and the default number of timestamps to be 1280×960 and 32, respectively, and adopting the Scott's rule [14, 42] to set the default spatial bandwidth b_σ and temporal bandwidth b_τ. Furthermore, we also set the default absolute error ϵ to be 0.05 for COMP. All the methods are implemented in C++ and experiments are conducted on an Intel i7 2.4GHz PC with 16GB memory. We omit the experiment results for those methods that take more than 259,200 sec (i.e., three days). As a remark, we only test all methods using the Epanechnikov spatial and temporal kernels in this section. All those experiments that are related to other kernel functions, including triangular kernel and quartic kernel, can be found in Section 7.5 of Appendix.

4.2 Compression Effectiveness of COMP

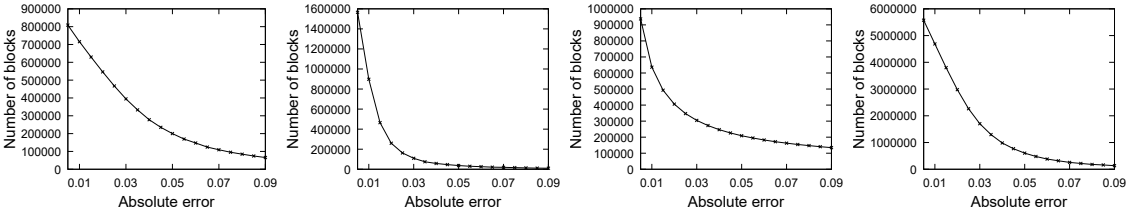
In this experiment, we investigate the compression effectiveness (i.e., the number of blocks in $\mathcal{S}$) of COMP by using different absolute error ϵ (ranging from 0.005 to 0.09). Figure 8 shows the results in different location datasets. Observe that the smaller the absolute error ϵ, the smaller the block size $\omega_{\mathcal{B}} \times \omega_{\mathcal{B}} \times \lambda_{\mathcal{B}}$ (see Theorem 1), resulting in the larger number of blocks $|\mathcal{S}|$. Note that COMP can already achieve a small $|\mathcal{S}|$ although we adopt a small ϵ. For example, $|\mathcal{S}| = 36{,}970$ for the New York traffic accident dataset with $\epsilon = 0.05$, which is only 1.979% compared with the original dataset size (with $n = 1{,}867{,}735$). With the high compression effectiveness, we anticipate that the combined versions of COMP, i.e., COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$, can achieve better efficiency performance compared with the corresponding methods, i.e., SWS and PREFIX, respectively.

4.3 Efficiency Performance of All Methods

In this section, we examine the efficiency performance of all methods by conducting the following three experiments. Due to space limitations, some experiments, e.g., varying the number of timestamps and varying the temporal bandwidth, can be found in Section 7.5 of Appendix.

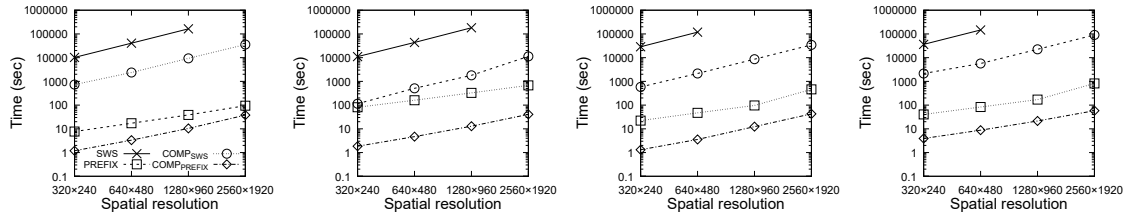
Varying the spatial resolution. We test how the spatial resolution $X \times Y$ affects the response time of each method. To conduct this experiment, we choose four resolution sizes, which are 320×240, 640×480, 1280×960, and 2560×1920, for generating STKDVs with respect to all methods. Since the number of blocks $|\mathcal{S}|$ is much smaller than the number of data points n (see Figure 8), COMP_{ALG} has small time complexity compared with the corresponding existing method no matter which ALG (either SWS or PREFIX) we choose (as $\mathbb{T}_{\text{ALG}}(|\mathcal{S}|) << \mathbb{T}_{\text{ALG}}(n)$ in Theorem 2). As such, observe from Figure 9 that COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ achieve speedups of 13.94x to 100.98x and speedups of 2.49x to 44.01x compared with SWS and PREFIX, respectively.

Varying the spatial bandwidth. We examine how the spatial bandwidth b_σ affects the response time of each method. To conduct this experiment, we first specify four spatial bandwidth values by multiplying the default spatial bandwidth with four ratios, which are 1 (the default one), 2, 4, and 8. Then, we measure the response time of each method with respect to these spatial bandwidth values for every dataset. Figure 10 shows the results of all methods. Due to the smaller number of block size $|\mathcal{S}|$ of COMP (see Figure 8) compared with n, COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can achieve speedups of 17.52x to 677.16x and 3.7x to 79.14x compared with SWS and PREFIX, respectively. In addition, we also note that the larger the spatial bandwidth b_σ, the smaller the response time of COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$. The main reason is that $\omega_{\mathcal{B}}$ is linearly proportional to b_σ (see Theorem 1), which indicates that the block size $\omega_{\mathcal{B}} \times \omega_{\mathcal{B}} \times \lambda_{\mathcal{B}}$ is larger (i.e., the number of blocks $|\mathcal{S}|$ is smaller) if



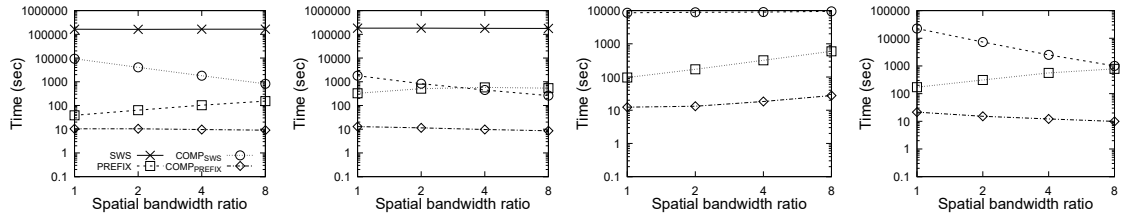
(a) Montgomery (b) New York (c) Chicago (d) San Francisco

Figure 8: The number of blocks in $\mathcal{S}$, varying the absolute error ϵ.



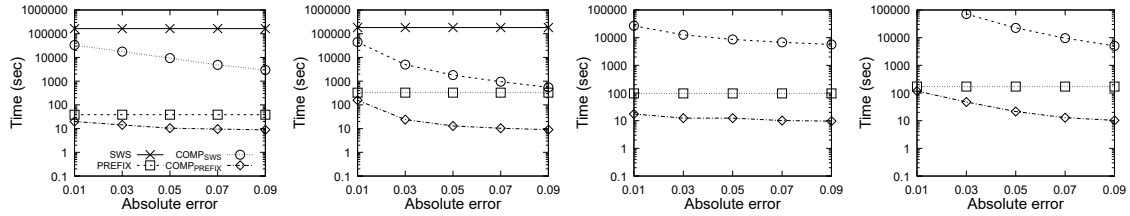
(a) Montgomery (b) New York (c) Chicago (d) San Francisco

Figure 9: Response time for computing STKDV, varying the spatial resolution.



(a) Montgomery (b) New York (c) Chicago (d) San Francisco

Figure 10: Response time for computing STKDV, varying the spatial bandwidth.



(a) Montgomery (b) New York (c) Chicago (d) San Francisco

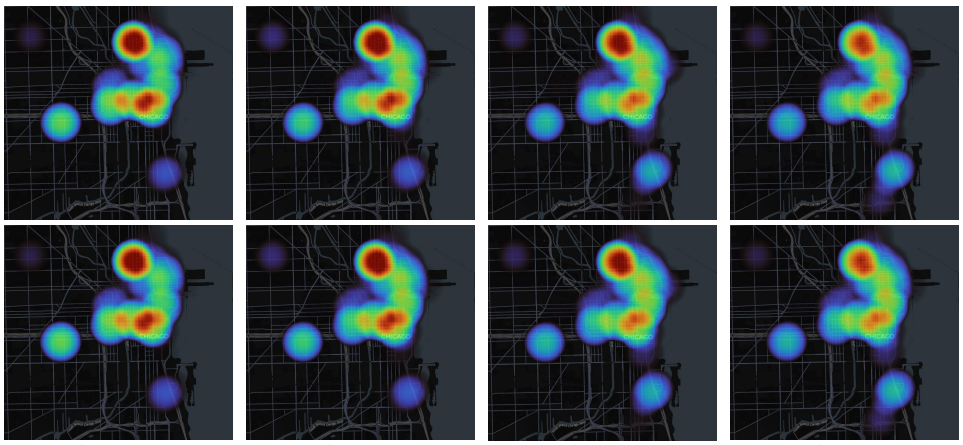
Figure 11: Response time for computing STKDV, varying the absolute error.

we adopt a larger spatial bandwidth value b_σ. With a small $|\mathcal{S}|$, the response time is also smaller (see Theorem 2).

Varying the absolute error. We further test how the absolute error ϵ affects the response time of each method by considering five absolute error values, which are 0.01, 0.03, 0.05, 0.07, and 0.09. Figure 11 shows the results of all methods. Since all exact methods, including SWS and PREFIX, do not depend on the absolute error ϵ, the response time of these two methods does not change across this parameter. Once we reduce ϵ, we note that the response time of COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ increases. The main reason is that both $\omega_{\mathcal{B}}$ and $\lambda_{\mathcal{B}}$ decrease (according to Theorem 1), which results in a smaller block size, i.e., a larger number of blocks $|\mathcal{S}|$ for COMP. Despite this, both COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can still achieve speedups of 4.1x to 332.48x and 1.45x to 36.32x compared with SWS and PREFIX, respectively, no matter which ϵ we choose.

4.4 Accuracy Performance of All Methods

We further investigate the accuracy performance of all exact (i.e., SWS and PREFIX) and approximate (i.e., COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$) methods for generating STKDVs with the default settings. Due to space limitations, we only test the subjective accuracy using the Chicago taxi pick-up location dataset in this section. Some additional experiments about the objective accuracy for all datasets can be found in Section 7.5 of Appendix. Figure 12 shows the STKDV results with four timestamps using the exact (the upper ones) and approximate (the lower ones) methods. Observe that there is nearly no difference between the exact STKDV and the approximate STKDV. The main reason is that COMP can provide the approximation guarantee between the approximate density value $A_{\mathcal{S}}(\mathbf{q}, t_i)$ and the exact density value $\mathcal{F}_P(\mathbf{q}, t_i)$ for each pixel-timestamp pair $(\mathbf{q}, t_i)$ (see Definition 2). As a remark, both exact and approximate STKDV methods can clearly demonstrate how the



(a) 10th January 2024 (b) 5th April 2024 (c) 9th July 2024 (d) 22nd October 2024

Figure 12: Accuracy comparison between the exact method (upper figures) and the approximate method (lower figures) for generating STKDVs with four timestamps in the Chicago taxi pick-up location dataset.

hotspot changes with respect to different timestamps. For example, the taxi pick-up location hotspot in the upper part of Chicago is reduced (with light red color) on 22nd October 2024 compared with the one (with dark red color) on 10th January 2024.

5 Related Work

We review two types of research studies, which are closely related to our work, in this section.

5.1 Efficient Algorithms for Kernel Density Visualization

In recent decades, many research studies have been proposed for solving the kernel density visualization (KDV) problem, which aims to generate a visualization based on computing the following kernel density function (by ignoring all timestamps and the temporal kernel function in Equation 1) for each pixel $\mathbf{q}$.

$$\mathcal{F}_P(\mathbf{q}) = \frac{1}{n} \sum_{\mathbf{p} \in P} K_{\text{space}}(\mathbf{q}, \mathbf{p}) \tag{16}$$

Here, we summarize the representative approaches, including (1) data indexing, (2) data sampling, and (3) computational sharing.

Data indexing. Observe from Table 1 that only those data points $\mathbf{p}$ with $||\mathbf{q} - \mathbf{p}||_2 \leq b_\sigma$ can be possibly used for computing $\mathcal{F}_P(\mathbf{q})$. As such, Gray et al. [24] and Gan et al. [23] adopt the kd-tree [10] and ball-tree [34] to improve the efficiency for evaluating $\mathcal{F}_P(\mathbf{q})$. Note that other types of indexing structures [40] can also be adopted for improving the efficiency of solving this problem. Recently, Chan et al. [12, 17, 19] first develop some simple bound functions to approximate the more complex kernel density function $\mathcal{F}_P(\mathbf{q})$ and incorporate these bound functions into the indexing structures (e.g., kd-tree and ball-tree) in order to boost the efficiency of generating approximate KDV. Although these indexing techniques can improve the efficiency of generating KDV and can be easily extended for solving the STKDV problem, Chan et al. [14, 16] verify that they can provide inferior efficiency performance compared with their state-of-the-art solutions (i.e., SWS [14] and PREFIX [16]), let alone to our advanced block compression solution.

Data sampling. In both data mining, database, and theoretical computer science communities, many data sampling methods [20, 21, 28, 31, 36–38, 43, 47, 48] have been proposed to boost the efficiency of generating KDV. However, instead of providing the deterministic error guarantees (e.g., within a predefined absolute error with the probability 1), all these methods can only provide the probabilistic error guarantees (e.g., within a predefined absolute error with the probability 0.8). Worse still, some of these methods either suffer from high time-complexity for constructing the data samples (or coresets) [36, 38] or cannot support those commonly used kernel functions in Table 1 [31, 43]. In addition, since all these methods do not focus on the more complex spatiotemporal kernel density function $\mathcal{F}_P(\mathbf{q}, t_i)$ (see Equation 1), none of these methods, to the best of our knowledge, can be extended for generating STKDV with non-trivial accuracy guarantees.

Computational sharing. Recently, Chan et al. [13, 18] propose to share computations between consecutive pixels [18] and multiple spatial bandwidth parameters b_σ [13], which can successfully reduce the time complexity for generating KDV. Despite this, these research studies only focus on $\mathcal{F}_P(\mathbf{q})$, which ignores the more complicated spatiotemporal kernel density function $\mathcal{F}_P(\mathbf{q}, t_i)$ (see Equation 1). Therefore, all these research studies cannot be directly extended to solve our STKDV problem.

5.2 Efficient Algorithms for Spatiotemporal Kernel Density Visualization

There are also two representative approaches for solving the more complicated spatiotemporal kernel density visualization (STKDV) problem, namely (1) complexity-reduced methods and (2) parallel/distributed methods.

Complexity-reduced methods. Recently, Chan et al. [14, 16] develop two pioneering methods, namely the sliding-window-based solution (SWS) [14] and the prefix-matrix-based solution (PREFIX) [16], which can successfully reduce the time complexity of generating exact STKDV to $O(XY(T + n))$ and $O(XYT + Yn)$, respectively. The technical details have been briefly discussed in Section 2.2. Although the best method, PREFIX, can significantly improve the efficiency of STKDV, the term Yn can still be large, which cannot be scalable to a large n. By combining COMP with PREFIX, this integrated solution can achieve the best efficiency without degrading the visualization quality.

Parallel/Distributed methods. Some research studies [22, 25, 26, 41] adopt the parallel/distributed methods to improve the efficiency of generating STKDV. However, all of them only aim to boost the efficiency of the naïve solution (with $O(XYTn)$ time), which cannot be scalable to large-scale datasets. Recently, Chan et al. [14, 16] also parallelize their PREFIX and SWS, which can boost the efficiency of their solutions. Despite this, domain experts mainly adopt contemporary software packages (e.g., ArcGIS, QGIS, and Scikit-learn) for analyzing their datasets, who may not have many computational resources to perform parallel/distributed computing. Therefore, our work does not consider this setting. In addition, COMP only reduces the size of each dataset, which can combine with any backbone algorithm (e.g., SWS and PREFIX). As such, these parallel/distributed methods can also improve the efficiency of our solution. Besides, incorporating the parallel/distributed methods into our solution is orthogonal to this work.

6 Conclusion

We study the spatiotemporal kernel density visualization problem, which is regarded to be the inefficient visualization/data analysis tool in many domains of geospatial analysis. Although many recent algorithms, including SWS [14] and PREFIX [16], have been proposed to tackle the efficiency issues for generating STKDV, these algorithms cannot be scalable to handle large location datasets (with large n). In this paper, we observe that many data points are spatiotemporally close to each other. Based on this observation, we propose the pioneering block compression solution (COMP) for this problem, which can significantly compress (or represent) each location dataset by a small amount of blocks, without theoretically incurring large error for generating approximate STKDV. By combining COMP with the existing methods, our experimental results verify that COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can achieve speedups of 4.1x to 677.16x and 1.45x to 143.52x compared with SWS and PREFIX, respectively, without degrading the visualization results.

In the future, we plan to investigate how to extend this block compression solution to other geospatial analysis tasks, including K-function [4] and inverse distance weighted interpolation [3]. Furthermore, we will develop a python library, an R package, and a QGIS/ArcGIS plugin based on COMP to support efficient and accurate STKDV.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China under Grants 23IAA00610, 62202401, 62372308, 42311530335, 42471496, the Science and Technology Development Fund Macau SAR (0003/2023/RIC, 0052/2023/RIA1, 0031/2022/A, 001/2024/SKL for SKL-IOTSC), GuangDong Basic and Applied Basic Research Foundation 2023A1515011619, the Innovation Team of the Department of Education of Guangdong Province (2024KCXTD013), the Key Technological Innovation Program of Ningbo City under Grant No. 2024Z297, and the Key Basic Research Foundation of Shenzhen under Grant No. JCYJ20220818100205012.

References

[1] [n. d.]. ArcGIS. http://pro.arcgis.com/en/pro-app/tool-reference/spatial-analyst/how-kernel-density-works.htm.
[2] [n. d.]. Chicago Open Data. https://catalog.data.gov/dataset/taxi-trips-2024.
[3] [n. d.]. How IDW works. https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/multi-distance-spatial-cluster-analysis.htm.
[4] [n. d.]. Multi-Distance Spatial Cluster Analysis (Ripley's K Function) (Spatial Statistics). https://pro.arcgis.com/en/pro-app/latest/tool-reference/3d-analyst/how-idw-works.htm.
[5] [n. d.]. NYC Open Data. https://catalog.data.gov/dataset/motor-vehicle-collisions-crashes.
[6] [n. d.]. QGIS. https://docs.qgis.org/2.18/en/docs/user_manual/plugins/plugins_heatmap.html.
[7] [n. d.]. San Francisco Open Data. https://data.sfgov.org/City-Infrastructure/311-Cases/vw6y-z8j6.
[8] [n. d.]. Seaborn. https://seaborn.pydata.org/generated/seaborn.kdeplot.html.
[9] [n. d.]. Traffic Violations in the Montgomery County of Maryland. https://catalog.data.gov/dataset/traffic-violations.
[10] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.
[11] 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
[12] Tsz Nam Chan, Reynold Cheng, and Man Lung Yiu. 2020. QUAD: Quadratic-Bound-based Kernel Density Visualization. In *SIGMOD*. 35–50. https://doi.org/10.1145/3318464.3380561
[13] 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.
[14] 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.
[15] Tsz Nam Chan, Pak Lon Ip, Kaiyan Zhao, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. LIBKDV: A Versatile Kernel Density Visualization Library for Geospatial Analytics. *Proc. VLDB Endow.* 15, 12 (2022), 3606–3609. https://www.vldb.org/pvldb/vol15/p3606-chan.pdf.
[16] Tsz Nam Chan, Pak Lon Ip, Bojian Zhu, Leong Hou U, Dingming Wu, Jianliang Xu, and Christian S. Jensen. 2025. Large-scale Spatiotemporal Kernel Density Visualization. In *ICDE*. 99–113. https://doi.org/10.1109/ICDE65448.2025.00015
[17] Tsz Nam Chan, Leong Hou U, Reynold Cheng, Man Lung Yiu, and Shivansh Mittal. 2022. Efficient Algorithms for Kernel Aggregation Queries. *IEEE Trans. Knowl. Data Eng.* 34, 6 (2022), 2726–2739. https://doi.org/10.1109/TKDE.2020.3018376
[18] 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
[19] Tsz Nam Chan, Man Lung Yiu, and Leong Hou U. 2019. KARL: Fast Kernel Aggregation Queries. In *ICDE*. 542–553. https://doi.org/10.1109/ICDE.2019.00055
[20] Moses Charikar, Michael Kapralov, and Erik Waingarten. 2024. A Quasi-Monte Carlo Data Structure for Smooth Kernel Evaluations. In *SODA*. SIAM, 5118–5144. https://doi.org/10.1137/1.9781611977912.184
[21] Moses Charikar and Paris Siminelakis. 2017. Hashing-Based-Estimators for Kernel Density in High Dimensions. In *FOCS*. 1032–1043. https://doi.org/10.1109/FOCS.2017.99
[22] 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
[23] Edward Gan and Peter Bailis. 2017. Scalable Kernel Density Classification via Threshold-Based Pruning. In *ACM SIGMOD*. 945–959.
[24] Alexander G. Gray and Andrew W. Moore. 2003. Nonparametric Density Estimation: Toward Computational Tractability. In *SDM*. 203–211.
[25] 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
[26] Alexander Hohl, Erik Saule, Eric Delmelle, and Wenwu Tang. 2020. *Spatiotemporal Domain Decomposition for High Performance Computing: A Flexible Splits Heuristic to Minimize Redundancy.* Springer International Publishing, Cham, 27–50. https://doi.org/10.1007/978-3-030-47998-5_3
[27] 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
[28] Sarang C. Joshi, Raj Varma Kommaraju, Jeff M. Phillips, and Suresh Venkatasubramanian. 2011. Comparing distributions and shapes using the kernel distance. In *SOCG*. 47–56. https://doi.org/10.1145/1998196.1998204
[29] Zihan Kan, Mei-Po Kwan, Jianwei Huang, Jiannan Cai, and Dong Liu. 2024. A spatial network-based assessment of individual exposure to COVID-19. *Annals of the American Association of Geographers* 114, 8 (2024), 1693–1703.
[30] Jay Lee. 2023. *Spatiotemporal Analytics.* CRC Press.
[31] Runze Lei, Pinghui Wang, Rundong Li, Peng Jia, Junzhou Zhao, Xiaohong Guan, and Chao Deng. 2021. Fast Rotation Kernel Density Estimation over Data Streams. In *SIGKDD*. 892–902. https://doi.org/10.1145/3447548.3467356

[32] Zhaoyang Liu, Yanyan Shen, and Yanmin Zhu. 2018. Where Will Dockless Shared Bikes be Stacked?: - Parking Hotspots Detection in a New City. In *SIGKDD*. 566–575. https://doi.org/10.1145/3219819.3219920
[33] Jonas Lukasczyk, Ross Maciejewski, Christoph Garth, and Hans Hagen. 2015. Understanding hotspots: a topological visual analytics approach. In *SIGSPATIAL*. 36:1–36:10. https://doi.org/10.1145/2820783.2820817
[34] Andrew W. Moore. 2000. The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data. In *UAI*. 397–405.
[35] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, and et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[36] Jeff M. Phillips. 2013. ϵ-Samples for Kernels. In *SODA*. 1622–1632. https://doi.org/10.1137/1.9781611973105.116
[37] 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
[38] 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
[39] Amanda Gadelha Ferreira Rosa, Caroline Maria de Miranda Mota, and Ciro José Jardim de Figueiredo. 2023. A spatial multi-criteria decision analysis framework to reveal vulnerabilities of areas to incidences of street robberies. *Applied Geography* 151 (2023), 102840. https://doi.org/10.1016/j.apgeog.2022.102840
[40] Hanan Samet. 2006. *Foundations of Multidimensional and Metric Data Structures.*
[41] 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
[42] David W Scott. 2015. *Multivariate density estimation: theory, practice, and visualization.* John Wiley & Sons.
[43] Wai Ming Tai. 2022. Optimal Coreset for Gaussian Kernel Density Estimation. In *SoCG (LIPIcs, Vol. 224)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 63:1–63:15. https://doi.org/10.4230/LIPICS.SOCG.2022.63
[44] Brian S Thomson, Judith B Bruckner, and Andrew M Bruckner. 2008. *Elementary real analysis*. Vol. 1. ClassicalRealAnalysis.com.
[45] Yuchen Yan, Wei Quan, and Hua Wang. 2024. A data-driven adaptive geospatial hotspot detection approach in smart cities. *Transactions in GIS* 28, 2 (2024), 303–325. https://doi.org/10.1111/tgis.13137
[46] Wenting Zhang, Haochun Guan, Shan Li, Bo Huang, Wuyang Hong, and Wenping Liu. 2024. The impact of street-scale built environments on urban park visitations: A case study in Wuhan. *Applied Geography* 171 (2024), 103374. https://doi.org/10.1016/j.apgeog.2024.103374
[47] 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.
[48] 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

7 Appendix

7.1 Proof of Lemma 1

Proof. Consider the expression $|A_{\mathcal{S}}(\mathbf{q}, t_i) - \mathscr{F}_P(\mathbf{q}, t_i)|$. We have

$$\begin{aligned}
&|A_{\mathcal{S}}(\mathbf{q}, t_i) - \mathscr{F}_P(\mathbf{q}, t_i)| \\
=\ & \frac{1}{n}\Big| \sum_{\mathcal{B}\in\mathcal{S}} \mathbb{C}(\mathcal{B}) \cdot K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) \\
-\ & \sum_{\mathcal{B}\in\mathcal{S}} \sum_{(\mathbf{p},t_{\mathbf{p}})\in\mathcal{B}} K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_i, t_{\mathbf{p}})\Big| \\
=\ & \frac{1}{n}\Big| \sum_{\mathcal{B}\in\mathcal{S}} \sum_{(\mathbf{p},t_{\mathbf{p}})\in\mathcal{B}} (K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}})K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p})K_{\text{time}}(t_i, t_{\mathbf{p}}))\Big| \\
\le\ & \frac{1}{n} \sum_{\mathcal{B}\in\mathcal{S}} \sum_{(\mathbf{p},t_{\mathbf{p}})\in\mathcal{B}} \Big|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}})K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p})K_{\text{time}}(t_i, t_{\mathbf{p}})\Big|
\end{aligned}$$

Therefore, if $|K_{\text{space}}(\mathbf{q}, \mathbf{p}_{\mathcal{B}})K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{space}}(\mathbf{q}, \mathbf{p})K_{\text{time}}(t_i, t_{\mathbf{p}})| \le \epsilon$, we can conclude that

$$|A_{\mathcal{S}}(\mathbf{q}, t_i) - \mathscr{F}_P(\mathbf{q}, t_i)| \le \frac{1}{n} \sum_{\mathcal{B}\in\mathcal{S}} \sum_{(\mathbf{p},t_{\mathbf{p}})\in\mathcal{B}} \epsilon = \epsilon$$

□

7.2 Proof of Lemma 2

Proof. In order to prove this lemma, we consider three cases, which are (1) $x_1 < x_2 \le b$, (2) $x_1 \le b < x_2$, and (3) $b < x_1 < x_2$.

Case 1 **($x_1 < x_2 \le b$):** In this case, both $(x_1, \mathcal{K}(x_1))$ and $(x_2, \mathcal{K}(x_2))$ are on the polynomial (and differentiable) curve (e.g., $\frac{3}{4}(1 - \frac{x^2}{b^2})$ in Figure 7). Therefore, based on the mean value theorem [44], there exist a value ζ, where $x_1 \le \zeta \le x_2$, so that

$$\frac{\mathcal{K}(x_1) - \mathcal{K}(x_2)}{x_1 - x_2} = \frac{d\mathcal{K}(x)}{dx}\Big|_{x=\zeta}$$

Hence, we have

$$\left|\frac{\mathcal{K}(x_1) - \mathcal{K}(x_2)}{x_1 - x_2}\right| = \left|\frac{d\mathcal{K}(x)}{dx}\Big|_{x=\zeta}\right| \le m_{\max}$$

Case 2 **($x_1 \le b < x_2$):** In this case, we have $\mathcal{K}(x_2) = 0$ and $|x_1 - b| \le |x_1 - x_2|$. Therefore, we can conclude that

$$\left|\frac{\mathcal{K}(x_1) - \mathcal{K}(x_2)}{x_1 - x_2}\right| \le \left|\frac{\mathcal{K}(x_1)}{x_1 - b}\right|$$

Moreover, since the data points $(x_1, \mathcal{K}(x_1))$ and $(b, 0)$ are on the polynomial (and differentiable) curve (see Figure 7 as an example), we adopt the mean value theorem [44] to show that we have (where $x_1 \le \zeta \le b$)

$$\frac{\mathcal{K}(x_1)}{x_1 - b} = \frac{d\mathcal{K}(x)}{dx}\Big|_{x=\zeta}$$

Based on the above two equations, we conclude that

$$\left|\frac{\mathcal{K}(x_1) - \mathcal{K}(x_2)}{x_1 - x_2}\right| \le \left|\frac{d\mathcal{K}(x)}{dx}\Big|_{x=\zeta}\right| \le m_{\max}$$

Case 3 **($b < x_1 < x_2$):** Note that $\mathcal{K}(x_1) = \mathcal{K}(x_2) = 0$ in this case. Therefore, we have

$$\left|\frac{\mathcal{K}(x_1) - \mathcal{K}(x_2)}{x_1 - x_2}\right| = 0 \le m_{\max}$$

□

7.3 Derivation of $m_{\max}^{(\sigma)}$ and $m_{\max}^{(\tau)}$ for Different Kernel Functions

Here, we only focus on the derivation of $m_{\max}^{(\sigma)}$. We can simply replace b_σ by b_τ to obtain $m_{\max}^{(\tau)}$.

Consider $\mathcal{K}(x)$ for the triangular spatial kernel (see Table 1). We have

$$\mathcal{K}(x) = \begin{cases} 1 - \frac{x}{b_\sigma} & \text{if } x \le b_\sigma \\ 0 & \text{otherwise} \end{cases} \implies \frac{d\mathcal{K}(x)}{dx} = \begin{cases} -\frac{1}{b_\sigma} & \text{if } x \le b_\sigma \\ 0 & \text{otherwise} \end{cases}$$

Hence, we can conclude that $m_{\max}^{(\sigma)} = \frac{1}{b_\sigma}$.

Consider $\mathcal{K}(x)$ for the Epanechnikov spatial kernel (see Table 1). We have

$$\mathcal{K}(x) = \frac{3}{4} \cdot \begin{cases} 1 - \frac{x^2}{b_\sigma^2} & \text{if } x \le b_\sigma \\ 0 & \text{otherwise} \end{cases} \implies \frac{d\mathcal{K}(x)}{dx} = \begin{cases} -\frac{3x}{2b_\sigma^2} & \text{if } x \le b_\sigma \\ 0 & \text{otherwise} \end{cases}$$

Hence, we can conclude that $m_{\max}^{(\sigma)} = \frac{3}{2b_\sigma}$.

Consider $\mathcal{K}(x)$ for the quartic spatial kernel (see Table 1). We have

$$\begin{aligned}
\mathcal{K}(x) &= \frac{15}{16} \cdot \begin{cases} \left(1 - \frac{x^2}{b_\sigma^2}\right)^2 & \text{if } x \le b_\sigma \\ 0 & \text{otherwise} \end{cases} \\
\implies \frac{d\mathcal{K}(x)}{dx} &= \begin{cases} -\frac{15}{4}\left(\frac{x}{b_\sigma^2} - \frac{x^3}{b_\sigma^4}\right) & \text{if } x \le b_\sigma \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

Hence, we can conclude that $m_{\max}^{(\sigma)} = \frac{5\sqrt{3}}{6b_\sigma}$.

7.4 Proof of Lemma 3

Proof. To prove the bound in Equation 12, we let $x_1 = ||\mathbf{q}-\mathbf{p}_{\mathcal{B}}||_2$, $x_2 = ||\mathbf{q}-\mathbf{p}||_2$, and $m_{\max} = m_{\max}^{(\sigma)}$. Then, based on Lemma 2, we have

$$\begin{aligned}|K_{\text{space}}(\mathbf{q},\mathbf{p}_{\mathcal{B}}) - K_{\text{space}}(\mathbf{q},\mathbf{p})| &\leq m_{\max}^{(\sigma)}|||\mathbf{q}-\mathbf{p}_{\mathcal{B}}||_2 - ||\mathbf{q}-\mathbf{p}||_2| \\ &\leq m_{\max}^{(\sigma)}||\mathbf{p}-\mathbf{p}_{\mathcal{B}}||_2\end{aligned}$$

The last inequality is based on the triangle inequality. Observe from Figure 6 that each black sphere $(\mathbf{p}, t_{\mathbf{p}})$ must be within the block $\mathcal{B}$. Therefore, we also have $||\mathbf{p}-\mathbf{p}_{\mathcal{B}}||_2 \leq \sqrt{\left(\frac{\omega_{\mathcal{B}}}{2}\right)^2 + \left(\frac{\omega_{\mathcal{B}}}{2}\right)^2} = \frac{\sqrt{2}\cdot\omega_{\mathcal{B}}}{2}$. Based on these two inequalities, we conclude that

$$|K_{\text{space}}(\mathbf{q},\mathbf{p}_{\mathcal{B}}) - K_{\text{space}}(\mathbf{q},\mathbf{p})| \leq \frac{\sqrt{2}\cdot\omega_{\mathcal{B}}\cdot m_{\max}^{(\sigma)}}{2}$$

To prove the bound in Equation 13, we let $x_1 = |t_i - t_{\mathbf{p}_{\mathcal{B}}}|$, $x_2 = |t_i - t_{\mathbf{p}}|$, and $m_{\max} = m_{\max}^{(\tau)}$. Based on Lemma 2 and the triangle inequality, we have

$$\begin{aligned}|K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{time}}(t_i, t_{\mathbf{p}})| &\leq m_{\max}^{(\tau)}||t_i - t_{\mathbf{p}_{\mathcal{B}}}| - |t_i - t_{\mathbf{p}}|| \\ &\leq m_{\max}^{(\tau)}|t_{\mathbf{p}} - t_{\mathbf{p}_{\mathcal{B}}}|\end{aligned}$$

In Figure 6, observe that every black sphere $(\mathbf{p}, t_{\mathbf{p}})$ must be within the temporal distance $\frac{\lambda_{\mathcal{B}}}{2}$ from the red sphere $(\mathbf{p}_{\mathcal{B}}, t_{\mathbf{p}_{\mathcal{B}}})$, i.e., $|t_{\mathbf{p}} - t_{\mathbf{p}_{\mathcal{B}}}| \leq \frac{\lambda_{\mathcal{B}}}{2}$. Therefore, we conclude that

$$|K_{\text{time}}(t_i, t_{\mathbf{p}_{\mathcal{B}}}) - K_{\text{time}}(t_i, t_{\mathbf{p}})| \leq \frac{\lambda_{\mathcal{B}}\cdot m_{\max}^{(\tau)}}{2}$$

□

7.5 Additional Experiments

In this section, we provide those experiments that have not been presented in the main content, including (1) additional efficiency experiments of all methods, (2) other kernels, and (3) objective accuracy of all methods.

7.5.1 Additional Efficiency Experiments of All Methods. We further conduct these two additional experiments for testing the efficiency of all methods.

Varying the number of timestamps. We investigate how the number of timestamps T affects the response time of each method by choosing the number of timestamps from 8 to 64. Figure 13 shows the results of all methods. With the smaller number of blocks $|\mathcal{S}|$ of COMP (see Figure 8), both COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can achieve speedups of up to 105.14x and 45.97x compared with the corresponding methods, SWS and PREFIX, respectively, no matter which T we adopt.

Varying the temporal bandwidth. We proceed to investigate how the temporal bandwidth b_τ affects the response time of each method. Note that this experiment is similar to the one for varying the spatial bandwidth. Instead, we multiply the default temporal bandwidth with four ratios, i.e., 1 (the default one), 2, 4, and 8, in order to obtain the four temporal bandwidths for testing. Observe from Figure 14 that COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can normally achieve speedups of 17.53x to 603.78x and 3.15x to 138.06x compared with SWS and PREFIX, respectively. In addition, the larger the temporal bandwidth b_τ, the larger the $\lambda_{\mathcal{B}}$ value (see Theorem 1), which indicates that COMP produces larger blocks (i.e., produces the smaller number of blocks $|\mathcal{S}|$). Based on Theorem 2, COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can achieve the smaller response time if we adopt a larger b_τ (see Figure 14).

7.5.2 Other Kernels. Here, we conduct the following experiments to test the efficiency of all methods using the triangular kernel and the quartic kernel. As a remark, we only consider SWS and COMP_{SWS} for testing the triangular kernel since the PREFIX method cannot support this kernel function.

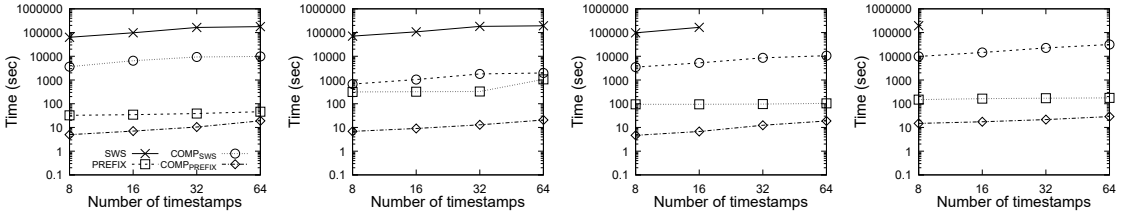
Varying the spatial bandwidth. In this experiment, we follow the same settings as in Section 4.3 for testing the efficiency of all methods with respect to different spatial bandwidths. Figure 15 shows the results of all methods. Like the results in Figure 10a and Figure 10b, the response time of COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ (only for the quartic kernel) is reduced when we adopt the large spatial bandwidth value b_σ. The main reason is that the number of blocks $|\mathcal{S}|$ of COMP is small (i.e., $\omega_{\mathcal{B}}$ becomes large) if we have a large b_σ (see Theorem 1), which indicates that COMP_{ALG} can provide fast performance for any exact algorithm ALG (see Theorem 2) no matter which kernel function we choose. Note that COMP can significantly improve the performance for generating STKDV. Regarding the triangular kernel, COMP_{SWS} can achieve speedups of 20.03x to 528.41x compared with SWS. Regarding the quartic kernel, COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ can achieve speedups of 10x to 271.53x and 3.22x to 54.87x compared with SWS and PREFIX, respectively.

Varying the temporal bandwidth. To conduct this experiment, we also follow the same settings as in Section 7.5.1 for testing the efficiency of all methods with respect to different temporal bandwidths. Figure 16 shows the results of all methods. Observe that COMP_{SWS} achieves speedups of 20.03x to 237.25x compared with SWS in the triangular kernel, while COMP_{SWS} and $\text{COMP}_{\text{PREFIX}}$ achieve speedups of 10x to 356.98x and 3.22x to 143.52x compared with SWS and PREFIX, respectively. The main reason is that COMP can achieve the small number of blocks $|\mathcal{S}|$ compared with the dataset size n, which results in the small time complexity (see Theorem 2).

7.5.3 Objective Accuracy of All Methods. Although we have performed an experiment to test the subjective accuracy of all methods in Section 4.4, we still do not know the objective accuracy of these methods. To conduct this experiment, we utilize the following measure, which represents the maximum density deviation between an exact STKDV and an approximate STKDV, to obtain the practical maximum error (*PME*) of the approximate method.

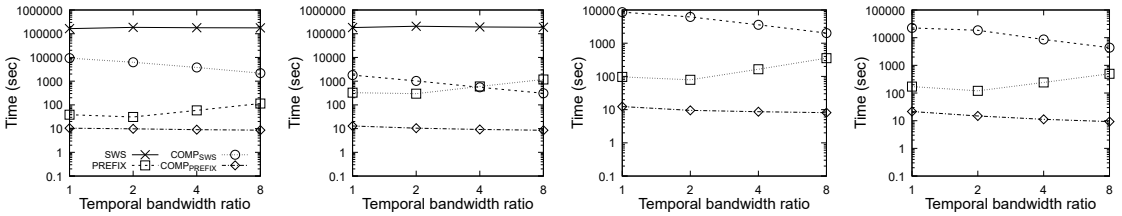
$$PME = \max_{\forall(\mathbf{q},t_i)} |A_{\mathcal{S}}(\mathbf{q},t_i) - \mathcal{F}_P(\mathbf{q},t_i)| \tag{17}$$

Figure 17 shows the results of the approximate method in all datasets. Observe that the *PME* values are normally in the range of $[10^{-5}, 10^{-3}]$, which can be much smaller compared with the absolute error ϵ. Therefore, this further explains why the approximate method does not degrade the STKDV result compared with the one that is generated by the exact method in the Chicago taxi pick-up location dataset (see Figure 12).



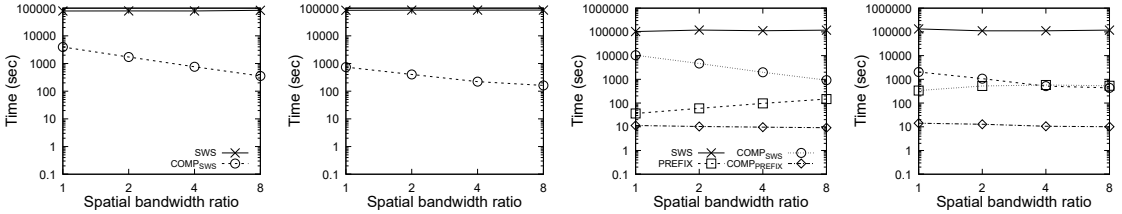
(a) Montgomery (b) New York (c) Chicago (d) San Francisco

Figure 13: Response time for computing STKDV, varying the number of timestamps.



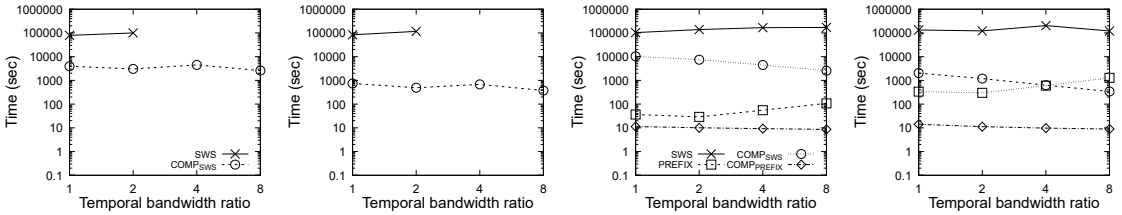
(a) Montgomery (b) New York (c) Chicago (d) San Francisco

Figure 14: Response time for computing STKDV, varying the temporal bandwidth.



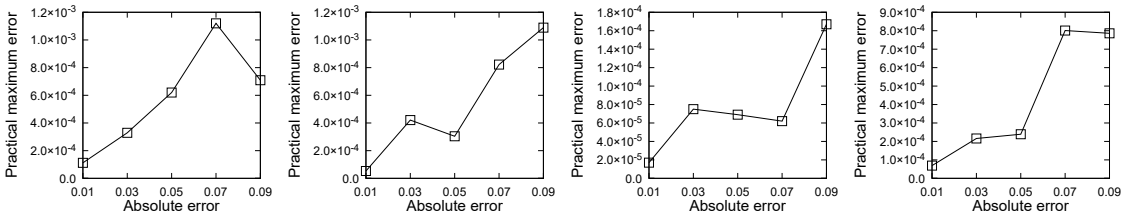
(a) Montgomery (Triangular kernel) (b) New York (Triangular kernel) (c) Montgomery (Quartic kernel) (d) New York (Quartic kernel)

Figure 15: Response time for computing STKDV in the Montgomery (a and c) and New York (b and d) datasets with the triangular kernel (a and b) and the quartic kernel (c and d), varying the spatial bandwidth.



(a) Montgomery (Triangular kernel) (b) New York (Triangular kernel) (c) Montgomery (Quartic kernel) (d) New York (Quartic kernel)

Figure 16: Response time for computing STKDV in the Montgomery (a and c) and New York (b and d) datasets with the triangular kernel (a and b) and the quartic kernel (c and d), varying the temporal bandwidth.



(a) Montgomery (b) New York (c) Chicago (d) San Francisco

Figure 17: Accuracy (practical maximum error) of the approximate method for computing STKDV, varying the absolute error.