SAFE: A Share-and-Aggregate Bandwidth Exploration Framework for Kernel Density Visualization

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

Pak Lon Ip
University of Macau
SKL of Internet of Things for Smart City
paklonip@um.edu.mo

Leong Hou U
University of Macau
SKL of Internet of Things for Smart City
ryanlhu@um.edu.mo

Byron Choi
Hong Kong Baptist University
bchoi@comp.hkbu.edu.hk

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

ABSTRACT

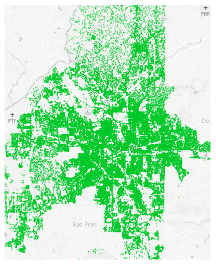
Kernel density visualization (KDV) has been the de facto method in many spatial analysis tasks, including ecological modeling, crime hotspot detection, traffic accident hotspot detection, and disease outbreak detection. In these tasks, domain experts usually generate multiple KDVs with different bandwidth values. However, generating a single KDV, let alone multiple KDVs, is time-consuming. In this paper, we develop a share-and-aggregate framework, namely SAFE, to reduce the time complexity of generating multiple KDVs given a set of bandwidth values. On the other hand, domain experts can specify bandwidth values on the fly. To tackle this issue, we further extend SAFE and develop the exact method SAFE$_{\text{all}}$ and the 2-approximation method SAFE$_{\text{exp}}$ which reduce the time complexity under this setting. Experimental results on four large-scale datasets (up to 4.33M data points) show that these three methods achieve at least one-order-of-magnitude speedup for generating multiple KDVs in most of the cases without degrading the visualization quality.

PVLDB Reference Format:
Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Byron Choi, and Jianliang Xu. SAFE: A Share-and-Aggregate Bandwidth Exploration Framework for Kernel Density Visualization. PVLDB, 15(3): 513-526, 2022.
doi:10.14778/3494124.3494135

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/edisonchan2013928/SAFE.

1 INTRODUCTION

Kernel-density-estimation-based visualization, a.k.a. kernel density visualization (KDV), [12, 24, 54] is a commonly used visualization tool for exploring and discovering patterns from a dataset that has been extensively used in a wide range of applications, including ecological modeling [8, 36, 37, 57, 63, 66], crime hotspot detection [11, 28, 40, 51], traffic accident hotspot detection [60, 65, 68], and disease outbreak detection [4, 14, 33, 42]. Due to its wide applicability, this operation has been supported in many scientific and geographical software packages, including Scikit-learn [43], QGIS [48], ArcGIS [1], and KDV-Explorer [14]. Figure 1b shows an example usage of KDV for visualizing the crime hotspots in Atlanta (this crime location dataset (cf. Figure 1a) can be found from [2]), using KDV-Explorer [14].


(a) Crime location data

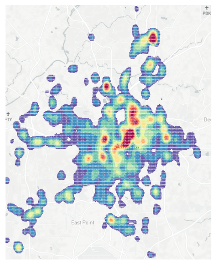

(b) Crime hotspot map

Figure 1: Using KDV to generate the crime hotspot map for Atlanta, where the red color denotes the high density (crime rate) of that region.

To generate such a hotspot map for a given spatial dataset P, we need to determine the color of each pixel $\mathbf{q}$, based on the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$ (cf. Equation 1), where w and $K_b(\mathbf{q},\mathbf{p})$ denote the normalization constant and the kernel function with bandwidth value b, respectively. The representative kernel functions are summarized in Table 1.

$$\mathcal{F}_P^{(b)}(\mathbf{q}) = \sum_{\mathbf{p}\in P} w \cdot K_b(\mathbf{q},\mathbf{p}) \tag{1}$$

Table 1: Commonly-used kernel functions.

Kernel	$K_b(\mathbf{q},\mathbf{p})$	Used in
Triangular	$\begin{cases} 1-\frac{1}{b}dist(\mathbf{q},\mathbf{p}) & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise}\end{cases}$	[8, 28]
Epanechnikov	$\begin{cases} 1-\frac{1}{b^2}dist(\mathbf{q},\mathbf{p})^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise}\end{cases}$	[10, 57]
Quartic	$\begin{cases} (1-\frac{1}{b^2}dist(\mathbf{q},\mathbf{p})^2)^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise}\end{cases}$	[65, 68]

Even though KDV has been extensively used for generating hotspot maps (cf. Figure 1), choosing the suitable bandwidth b

Proceedings of the VLDB Endowment, Vol. 15, No. 3 ISSN 2150-8097.
doi:10.14778/3494124.3494135

Table 2: Theoretical results of different methods.

Method	Time complexity	Space complexity	Quality	Bandwidth properties
Baseline (cf. Section 2.2)	$O(LXYn)$	$O(XYL+n)$	Exact	Known in advance On-the-fly
SAFE (cf. Section 3.2)	$O(XY(n\log L+L))$ (cf. Theorem 1)	$O(XYL+n)$ (cf. Theorem 3)	Exact	Known in advance
SAFE$_{\text{all}}$ (cf. Section 4.1)	$O(XY(n+L)\log n)$ (cf. Theorem 4)	$O(XY(n+L))$ (cf. Theorem 5)	Exact	On-the-fly
SAFE$_{\text{exp}}$ (cf. Section 4.2)	$O(XY(n\log n+L\log(\log n)))$ (cf. Theorem 8)	$O(XY(\log n+L)+n)$ (cf. Theorem 7)	2-approximation (cf. Theorem 6)	On-the-fly

for hotspot visualization is a challenging issue. In Figure 2a, we can observe that we cannot identify any hotspot region once we utilize a small b (undersmoothing). On the other hand, the hotspot region tends to be very large (cf. Figure 2c) if we choose a large b (oversmoothing).

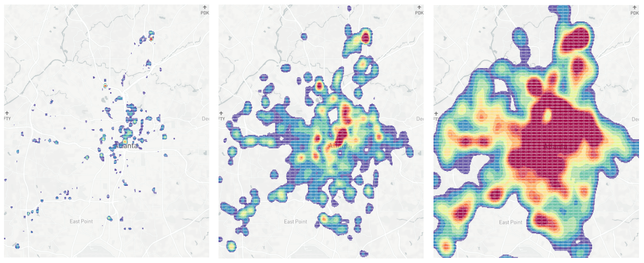

(a) Small (b) Moderate (c) Large

Figure 2: The crime hotspot maps for Atlanta, varying the bandwidth b of the Epanechnikov kernel function.

To obtain a high-quality hotspot map (e.g., Figure 2b), domain experts [8, 11, 60, 68] iteratively generate multiple hotspot maps by tuning different bandwidth values b and then select the best one. The main reason for using this approach is that choosing the suitable bandwidth b is subjective for hotspot detection in practice, which has been pointed out in many research studies.

- *"The choice of bandwidth is quite subjective..."* [60]
- *"An iterative procedure was followed to test the performance of various bandwidths,..."* [68]
- *"There is flexibility when setting parameters such as the grid cell size and bandwidth (search radius); however, despite many useful recommendations (see Ratcliffe and McCullagh, 1999; Chainey and Ratcliffe, 2005; Eck et al., 2005), there is no universal doctrine on how to set these and in what circumstances."* [11]

On the other hand, one of the most important tasks of hotspot analysis [28, 68] is to understand the risk of different regions by varying the bandwidth values. To provide better exploration experiences, a better approach is for domain experts to thoroughly view the changes in the hotspots versus the bandwidth values in different regions. Therefore, we need to generate multiple KDVs with many bandwidth values to support this task.

Although many efficient algorithms [12, 17, 72, 74] have been developed to improve the efficiency of generating a single KDV, these methods are not scalable to million-scale datasets [12], let alone to generate multiple KDVs. Since the above tasks need to compute KDVs for many bandwidth values, it is inefficient to utilize the existing methods to support them.

Motivated by this, this paper investigates the following research question: *Can we efficiently generate multiple KDVs by exploring the bandwidth values b in the commonly-used kernel functions (cf. Table 1)?* To provide an affirmative answer to this question, we develop a Share-and-Aggregate FramEwork, called SAFE, which significantly reduces the time complexity of generating multiple KDVs, without increasing the space complexity, if the bandwidth values are known in advance. However, users (e.g., geoscientists, criminologists) may not know all of the bandwidth values for testing in advance and they can fine-tune the bandwidth values or perform the online exploratory analysis. For this case, we propose a method, called SAFE$_{\text{all}}$, which reduces the response time for generating multiple KDVs with on-the-fly bandwidth values. Due to the high space complexity of using SAFE$_{\text{all}}$, we further develop a 2-approximation method, called SAFE$_{\text{exp}}$, which, compared with SAFE$_{\text{all}}$, reduces the space complexity and the time complexity. Table 2 summarizes the theoretical results of different methods. Here, we denote the resolution size, number of data points, and number of bandwidth values to be $X \times Y$, n, and L, respectively. Our experiments on four large-scale datasets show that all of our methods achieve at least one-order-of-magnitude speedup for generating multiple KDVs without degrading the visualization quality. As a remark, we also integrate our SAFE method in a system prototype to support bandwidth exploration (cf. Section 5.4), which has not been efficiently supported by other software packages.

The rest of the paper is organized as follows. First, we formally define the problem and introduce the background in Section 2. Then, in Section 3, we discuss our solution, SAFE, for efficiently generating multiple KDVs where the bandwidth values are known in advance. In Section 4, we extend SAFE and develop two solutions, SAFE$_{\text{all}}$ and SAFE$_{\text{exp}}$, to efficiently support generating multiple KDVs with on-the-fly bandwidth values. We discuss the experimental results in Section 5. Then, we review the existing work in Section 6. Lastly, we conclude our paper and discuss future work in Section 7.

2 PRELIMINARIES

In this section, we first formally define the problem in Section 2.1. Then, we present the baseline method, called range-query-based solution, in Section 2.2.

2.1 Problem Definition

Recall from Section 1, the users generate multiple KDVs (cf. Figure 2), by tuning the bandwidth b of the kernel function (cf. Table 1). In the following, we define the concept of KDV in Definition 1. Here, we choose the Epanechnikov kernel as the default kernel function for discussion. As a remark, our method also supports other kernel functions, which will be discussed in Section 3.

Definition 1. *(KDV) Given a plane with $X \times Y$ pixels, a point set P with n data points, a bandwidth value b and the Epanechnikov kernel (cf. Table 1), we color each pixel $\mathbf{q}$, using the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$, where:*

$$\mathcal{F}_P^{(b)}(\mathbf{q}) = \sum_{\mathbf{p} \in P} w \cdot \begin{cases} 1 - \frac{1}{b^2} dist(\mathbf{q},\mathbf{p})^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Based on the above definition, we can formulate the problem, termed as $\text{KDV}_{\text{explore}}$, in Problem 1.

Problem 1. *($\text{KDV}_{explore}$) Given a plane with $X \times Y$ pixels, a point set P with n data points and L bandwidth values $b_1, b_2, ..., b_L$, where: $b_1 < b_2 < ... < b_L$, we generate KDV (cf. Definition 1) for each bandwidth value.*

In practice, the users (e.g., geoscientists, criminologists, etc.) can either know the set of bandwidth values for testing in advance or select some bandwidth values on the fly. We will discuss both scenarios in this paper.

2.2 Range-Query-based Solution (RQS)

In this section, we proceed to discuss the baseline method, called range-query-based solution (RQS). Observe from Table 1, the data point $\mathbf{p}$ can contribute to the kernel function $K_b(\mathbf{q},\mathbf{p})$, once the distance value $dist(\mathbf{q},\mathbf{p})$ is within the bandwidth value b. Therefore, computing $\mathcal{F}_P^{(b)}(\mathbf{q})$ (cf. Equation 2) can be regarded as the following two-step method:

(1) Find the range query solution set $R_{\mathbf{q}}^{(b)}$, where:

$$R_{\mathbf{q}}^{(b)} = \{\mathbf{p} \in P | dist(\mathbf{q},\mathbf{p}) \le b\} \tag{3}$$

(2) Compute $\mathcal{F}_P^{(b)}(\mathbf{q})$ (cf. Equation 2), based on $R_{\mathbf{q}}^{(b)}$, where:

$$\mathcal{F}_P^{(b)}(\mathbf{q}) = \sum_{\mathbf{p} \in R_{\mathbf{q}}^{(b)}} w \cdot \left(1 - \frac{1}{b^2} dist(\mathbf{q},\mathbf{p})^2\right) \tag{4}$$

Even though RQS can improve the practical efficiency for generating a single KDV, the worst case time complexity for computing KDV remains in $O(XYn)$ time. The main reason is that $|R_{\mathbf{q}}^{(b)}| \to n$, once we use a large bandwidth value, e.g., $b \to \infty$. As such, the worst case time complexity for solving $\text{KDV}_{\text{explore}}$ (i.e., Problem 1) is $O(LXYn)$.

3 OUR SOLUTION

In this section, we first describe the core ideas of our method in Section 3.1. Then, we propose the Share-and-Aggregate FramEwork (SAFE) to solve $\text{KDV}_{\text{explore}}$ (cf. Problem 1), which reduces the time complexity from $O(LXYn)$ (RQS) to $O(XY(n \log L + L))$ for all kernel functions (cf. Table 1), without increasing the space complexity, in Section 3.2.

3.1 Core Ideas

One major drawback for using RQS (cf. Section 2.2) is that this method suffers from repetitive computation. Given the bandwidth values $b_1, b_2, ..., b_L$, RQS obtains $R_{\mathbf{q}}^{(b_1)}, R_{\mathbf{q}}^{(b_2)}, ..., R_{\mathbf{q}}^{(b_L)}$ and then evaluates $\mathcal{F}_P^{(b_1)}(\mathbf{q}), \mathcal{F}_P^{(b_2)}(\mathbf{q}), ..., \mathcal{F}_P^{(b_L)}(\mathbf{q})$, respectively. However, since $b_1 \le b_2 \le ... \le b_L$, those range query solution sets exhibit the following property:

$$R_{\mathbf{q}}^{(b_1)} \subseteq R_{\mathbf{q}}^{(b_2)} \subseteq ... \subseteq R_{\mathbf{q}}^{(b_L)} \tag{5}$$

Based on this property, suppose that we have already obtained the range query solution set $R_{\mathbf{q}}^{(b_L)}$ with the largest bandwidth b_L, this set $R_{\mathbf{q}}^{(b_L)}$ contains the sufficient and necessary information to compute all the kernel density function values $\mathcal{F}_P^{(b_1)}(\mathbf{q}), \mathcal{F}_P^{(b_2)}(\mathbf{q}), ..., \mathcal{F}_P^{(b_L)}(\mathbf{q})$ for the pixel $\mathbf{q}$. Therefore, we ask a question: *After we have computed $R_{\mathbf{q}}^{(b_L)}$, can we share the computations to all kernel density functions, $\mathcal{F}_P^{(b_1)}(\mathbf{q}), \mathcal{F}_P^{(b_2)}(\mathbf{q}), ..., \mathcal{F}_P^{(b_L)}(\mathbf{q})$, in order to efficiently solve $\text{KDV}_{explore}$?*

On the other hand, the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$ (cf. Equation 4) can be expanded in the following expression.

$$\begin{aligned} \mathcal{F}_P^{(b)}(\mathbf{q}) &= \sum_{\mathbf{p} \in R_{\mathbf{q}}^{(b)}} w \cdot \left(1 - \frac{1}{b^2} dist(\mathbf{q},\mathbf{p})^2\right) \\ &= \sum_{\mathbf{p} \in R_{\mathbf{q}}^{(b)}} w - \frac{w}{b^2} \sum_{\mathbf{p} \in R_{\mathbf{q}}^{(b)}} dist(\mathbf{q},\mathbf{p})^2 \\ &= w|R_{\mathbf{q}}^{(b)}| - \frac{w}{b^2} S_{R_{\mathbf{q}}^{(b)}} \end{aligned} \tag{6}$$

where $S_{R_{\mathbf{q}}^{(b)}} = \sum_{\mathbf{p} \in R_{\mathbf{q}}^{(b)}} dist(\mathbf{q},\mathbf{p})^2$. Here, we term $|R_{\mathbf{q}}^{(b)}|$ and $S_{R_{\mathbf{q}}^{(b)}}$ as the *cardinality* and *aggregate distance*, respectively, of the set $R_{\mathbf{q}}^{(b)}$.

Observe from the above expression, suppose that we can efficiently compute these aggregate values $|R_{\mathbf{q}}^{(b)}|$ and $S_{R_{\mathbf{q}}^{(b)}}$ for different bandwidth values b, we can efficiently compute the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$. Therefore, we ask another question: *Can we efficiently maintain these aggregate values for different bandwidth values $b_1, b_2, ..., b_L$ in order to efficiently solve $\text{KDV}_{explore}$?*

3.2 SAFE: Share-and-Aggregate Framework

Based on the two core ideas in Section 3.1, we propose the Share-and-Aggregate FramEwork (SAFE), which follows these two steps, (1) share and (2) aggregate.

Share: Observe from Figure 3, once we have obtained the range query solution set $R_{\mathbf{q}}^{(b_L)}$, we can share each data point $\mathbf{p}$ in $R_{\mathbf{q}}^{(b_L)}$ into the corresponding bandwidth gap $(b_{i-1}, b_i]$[1]. Here, we denote this set as $G_{\mathbf{q}}^{(b_{i-1},b_i)}$ (cf. Equation 7) and the dummy bandwidth b_0 to be $-\infty$.

$$G_{\mathbf{q}}^{(b_{i-1},b_i)} = \{\mathbf{p} \in P | b_{i-1} < dist(\mathbf{q},\mathbf{p}) \le b_i\} \tag{7}$$

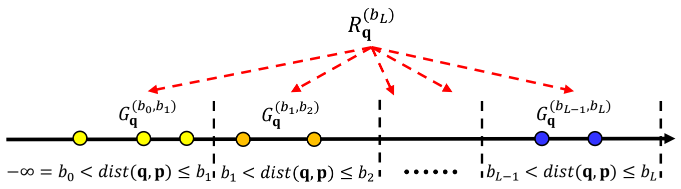

$R_{\mathbf{q}}^{(b_L)}$
$G_{\mathbf{q}}^{(b_0,b_1)}$
$G_{\mathbf{q}}^{(b_1,b_2)}$
$G_{\mathbf{q}}^{(b_{L-1},b_L)}$
$-\infty = b_0 < dist(\mathbf{q},\mathbf{p}) \le b_1$
$b_1 < dist(\mathbf{q},\mathbf{p}) \le b_2$
$\cdots\cdots$
$b_{L-1} < dist(\mathbf{q},\mathbf{p}) \le b_L$

Figure 3: Share each data point p in the range query solution set $R_{\mathbf{q}}^{(b_L)}$ into the corresponding bandwidth gap $(b_{i-1}, b_i]$, where yellow, orange, and blue circles denote the data points that are within the bandwidth gaps $(b_0, b_1]$, $(b_1, b_2]$, and $(b_{L-1}, b_L]$, respectively.

[1]Following the mathematical convention, we use] and (to denote that this bandwidth gap contains b_i and does not contain b_{i-1}, respectively.

Algorithm 1 Share Operation

```
1: procedure SHARE($R_{\mathbf{q}}^{(b_L)}$, $b_1, b_2, ..., b_L$)
2:     $b_0 \leftarrow -\infty$                                   ▷ Dummy bandwidth
3:     $G_{\mathbf{q}}^{(b_{i-1},b_i)} \leftarrow \phi$, where $i = 1, ..., L$          ▷ Initialization
4:     for each point $\mathbf{p} \in R_{\mathbf{q}}^{(b_L)}$ do
5:         $d \leftarrow dist(\mathbf{q}, \mathbf{p})$
6:         $i \leftarrow$ SEARCH($d$), where $b_{i-1} < d \le b_i$ ▷ Binary search
7:         $G_{\mathbf{q}}^{(b_{i-1},b_i)} \leftarrow G_{\mathbf{q}}^{(b_{i-1},b_i)} \cup \mathbf{p}$
8:     Return $G_{\mathbf{q}}^{(b_{i-1},b_i)}$, where $i = 1, ..., L$
```

Algorithm 1 shows the pseudocode of this share operation. We claim in Lemma 1 that this algorithm only incurs $O(n \log L + L)$ time.

LEMMA 1. *The worst case time complexity of Algorithm 1 is* $O(n \log L + L)$.

PROOF. Recall that we have $b_1 < b_2 < ... < b_L$ (cf. Problem 1), we can adopt the binary search method to find the correct bandwidth gap $(b_{i-1}, b_i]$ (line 6) for each $\mathbf{p}$ in $R_{\mathbf{q}}^{(b_L)}$, which incurs $O(\log L)$ time. Therefore, the loop (lines 4-7) takes $O(|R_{\mathbf{q}}^{(b_L)}| \log L)$ time to evaluate. Besides this loop, the computational time for lines 2, 3 and 8 is $O(L)$. As such, Algorithm 1 consumes $O(|R_{\mathbf{q}}^{(b_L)}| \log L + L)$ time. Theoretically, if $b_L \to \infty$, the size $|R_{\mathbf{q}}^{(b_L)}| \to n$. Therefore, we can conclude that the worst case time complexity of Algorithm 1 is $O(n \log L + L)$. □

Aggregate: Figure 4 summarizes the aggregate operation. Here, we denote the aggregate distance $S_{\mathcal{I}}$ for a set $\mathcal{I}$ of data points as:

$$S_{\mathcal{I}} = \sum_{\mathbf{p} \in \mathcal{I}} dist(\mathbf{q}, \mathbf{p})^2 \tag{8}$$

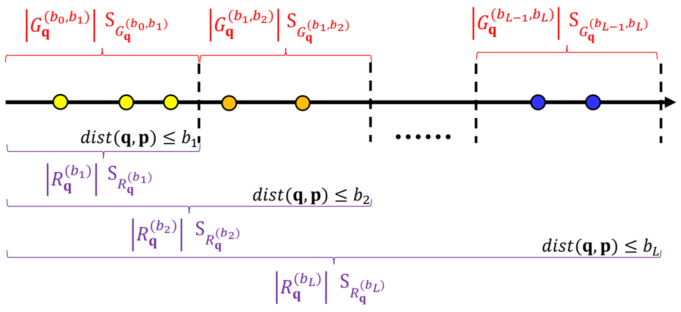

$|G_{\mathbf{q}}^{(b_0,b_1)}|$ $S_{G_{\mathbf{q}}^{(b_0,b_1)}}$
$|G_{\mathbf{q}}^{(b_1,b_2)}|$ $S_{G_{\mathbf{q}}^{(b_1,b_2)}}$
$|G_{\mathbf{q}}^{(b_{L-1},b_L)}|$ $S_{G_{\mathbf{q}}^{(b_{L-1},b_L)}}$
......
$dist(\mathbf{q},\mathbf{p}) \le b_1$
$|R_{\mathbf{q}}^{(b_1)}|$ $S_{R_{\mathbf{q}}^{(b_1)}}$
$dist(\mathbf{q},\mathbf{p}) \le b_2$
$|R_{\mathbf{q}}^{(b_2)}|$ $S_{R_{\mathbf{q}}^{(b_2)}}$
$dist(\mathbf{q},\mathbf{p}) \le b_L$
$|R_{\mathbf{q}}^{(b_L)}|$ $S_{R_{\mathbf{q}}^{(b_L)}}$

Figure 4: (1) Compute the aggregate values (in red color) for each bandwidth gap $(b_{i-1}, b_i]$ and (2) compute the aggregate values (in purple color) with $dist(\mathbf{q}, \mathbf{p}) \le b_i$.

Once we have obtained the set $G_{\mathbf{q}}^{(b_{i-1},b_i)}$ of the data points for each bandwidth gap $(b_{i-1}, b_i]$ (cf. Figure 3), we compute the cardinality of the point set $|G_{\mathbf{q}}^{(b_{i-1},b_i)}|$ and the aggregate distance $S_{G_{\mathbf{q}}^{(b_{i-1},b_i)}}$ (in red color) in this range.

Based on these red aggregate values, we can further compute the purple aggregate values (cf. Figure 4), including cardinality (cf. Equation 9), and aggregate distance (cf. Equation 10), for the range query solution set $R_{\mathbf{q}}^{(b_i)}$ (cf. Equation 3) with each bandwidth value b_i.

$$|R_{\mathbf{q}}^{(b_i)}| = \begin{cases} |R_{\mathbf{q}}^{(b_{i-1})}| + |G_{\mathbf{q}}^{(b_{i-1},b_i)}| & \text{if } i > 1 \\ |G_{\mathbf{q}}^{(b_0,b_1)}| & \text{if } i = 1 \end{cases} \tag{9}$$

$$S_{R_{\mathbf{q}}^{(b_i)}} = \begin{cases} S_{R_{\mathbf{q}}^{(b_{i-1})}} + S_{G_{\mathbf{q}}^{(b_{i-1},b_i)}} & \text{if } i > 1 \\ S_{G_{\mathbf{q}}^{(b_0,b_1)}} & \text{if } i = 1 \end{cases} \tag{10}$$

Recall from Equation 6, once we have these purple aggregate values $|R_{\mathbf{q}}^{(b_i)}|$ and $S_{R_{\mathbf{q}}^{(b_i)}}$, we can compute the kernel density function $\mathcal{F}_P^{(b_i)}(\mathbf{q})$ for the bandwidth b_i.

Algorithm 2 Aggregate Operation

```
1: procedure AGGREGATE($G_{\mathbf{q}}^{(b_0,b_1)}$,$G_{\mathbf{q}}^{(b_1,b_2)}$,...,$G_{\mathbf{q}}^{(b_{L-1},b_L)}$)
2:     for $i \leftarrow 1$ to $L$ do
3:         //Red aggregate values (cf. Figure 4)
4:         Compute $|G_{\mathbf{q}}^{(b_{i-1},b_i)}|$
5:         Compute $S_{G_{\mathbf{q}}^{(b_{i-1},b_i)}}$                    ▷ Equation 8
6:         //Purple aggregate values (cf. Figure 4)
7:         Compute $|R_{\mathbf{q}}^{(b_i)}|$                          ▷ Equation 9
8:         Compute $S_{R_{\mathbf{q}}^{(b_i)}}$                           ▷ Equation 10
9:         //Compute the kernel density value with bandwidth $b_i$
10:        Compute $\mathcal{F}_P^{(b_i)}(\mathbf{q})$                      ▷ Equation 6
11:    Return $\mathcal{F}_P^{(b_i)}(\mathbf{q})$, where $1 \le i \le L$
```

Algorithm 2 describes the aggregate operation in detail. Here, we claim that this algorithm can compute $\mathcal{F}_P^{(b)}(\mathbf{q})$, for all bandwidth values $b = b_1, b_2, ..., b_L$ in $O(n + L)$ time (cf. Lemma 2).

LEMMA 2. *The worst case time complexity of Algorithm 2 is* $O(n + L)$.

PROOF. Since we need to scan the set $G_{\mathbf{q}}^{(b_{i-1},b_i)}$ of data points in each bandwidth gap $(b_{i-1}, b_i]$ in order to compute the cardinality $|G_{\mathbf{q}}^{(b_{i-1},b_i)}|$ and the aggregate distance $S_{G_{\mathbf{q}}^{(b_{i-1},b_i)}}$, lines 4 and 5 in Algorithm 2 incur at most $O(L + |R_{\mathbf{q}}^{(b_L)}|)$ time in L iterations. Based on Equations 9 and 10, we can obtain the purple aggregate values $|R_{\mathbf{q}}^{(b_i)}|$ and $S_{R_{\mathbf{q}}^{(b_i)}}$ in $O(1)$ time, since $|R_{\mathbf{q}}^{(b_{i-1})}|$ and $S_{R_{\mathbf{q}}^{(b_{i-1})}}$ are available in the previous iteration and $|G_{\mathbf{q}}^{(b_{i-1},b_i)}|$ and $S_{G_{\mathbf{q}}^{(b_{i-1},b_i)}}$ are available in lines 4 and 5. Furthermore, once the purple aggregate values are available, computing $\mathcal{F}_P^{(b_i)}(\mathbf{q})$ (line 10) is in $O(1)$ time (cf. Equation 6). Therefore, lines 7, 8 and 10 in Algorithm 2 take $O(L)$ time for all iterations. In the worst case, Algorithm 2 takes $O(n + L)$ time to evaluate. □

SAFE for solving KDV$_{explore}$: To generate KDVs for all bandwidth values $b_1, b_2, ..., b_L$, each pixel $\mathbf{q}$ in the plane needs to (1) find the range query solution set $R_{\mathbf{q}}^{(b_L)}$ for the largest bandwidth value b_L, (2) share the data points in this $R_{\mathbf{q}}^{(b_L)}$ into different bandwidth gaps (cf. Algorithm 1), i.e., $G_{\mathbf{q}}^{(b_0,b_1)}$,..., $G_{\mathbf{q}}^{(b_{L-1},b_L)}$, and (3) compute the kernel density values for all bandwidth values $b_1, b_2,..., b_L$ based on the aggregate values (cf. Algorithm 2). Algorithm 3 shows the pseudocode for SAFE.

Algorithm 3 Share-and-Aggregate Framework (SAFE)

```
1:  procedure SAFE(Point set P, X, Y, $b_1,..., b_L$)
2:      Create $L$ planes $\mathcal{P}_1,..., \mathcal{P}_L$ with $X \times Y$ pixels
3:      for $x \leftarrow 1$ to $X$ do
4:          for $y \leftarrow 1$ to $Y$ do
5:              $\mathbf{q} \leftarrow$ COORD($x, y$)                    ▷ Obtain the coordinates
6:              $R_{\mathbf{q}}^{(b_L)} \leftarrow$ RANGE($\mathbf{q}, P, b_L$)          ▷ Range query
7:              Obtain $\{G_{\mathbf{q}}^{(b_0,b_1)}, ..., G_{\mathbf{q}}^{(b_{L-1},b_L)}\}$      ▷ Algorithm 1
8:              Obtain $\{\mathcal{F}_P^{(b_1)}(\mathbf{q}), ..., \mathcal{F}_P^{(b_L)}(\mathbf{q})\}$     ▷ Algorithm 2
9:              for $\ell \leftarrow 1$ to $L$ do
10:                 $\mathcal{P}_\ell(x, y) \leftarrow \mathcal{F}_P^{(b_\ell)}(\mathbf{q})$             ▷ Line 8
11:     Return $\mathcal{P}_\ell$, where $1 \le \ell \le L$
```

Based on the cost for performing range search ($O(n)$ time), the share operation (cf. Lemma 1) and the aggregate operation (cf. Lemma 2), we can conclude that each pixel $\mathbf{q}$ only takes $O(n \log L + L)$ time to compute the kernel density values for all bandwidth values $b_1, b_2,..., b_L$. Therefore, the worst case time complexity for using SAFE to solve $\text{KDV}_{\text{explore}}$ (cf. Problem 1) is $O(XY(n \log L + L))$ (cf. Theorem 1).

Theorem 1. *The worst case time complexity for using SAFE (cf. Algorithm 3) to solve $KDV_{explore}$ is $O(XY(n \log L + L))$.*

As such, we can conclude that our method SAFE can reduce the worst case time complexity for solving $\text{KDV}_{\text{explore}}$ (cf. Problem 1) compared with the method RQS (cf. Section 2.2), which takes $O(LXYn)$ time.

Further improvement: The main bottleneck of SAFE is to evaluate the share operation using Algorithm 1 (i.e., line 7 in Algorithm 3), which takes $O(n \log L + L)$ time for each pixel $\mathbf{q}$. We need this additional $\log L$ factor, since we need to utilize the binary search method (line 6 in Algorithm 1) to share each point into the correct bandwidth gap (cf. Figure 3). However, the users (e.g., geoscientists) can also visualize multiple KDVs, by uniformly increasing the bandwidth values (e.g., $b_1 = 100$, $b_2 = 200$,..., $b_{100} = 10000$). In this case, since all bandwidth gaps have the same distance (e.g., 100), we can mathematically derive the bandwidth gap i for each point $\mathbf{p}$ from $R_{\mathbf{q}}^{(b_L)}$ (cf. Figure 3) without using the binary search method, where:

$$i = \left\lceil \frac{dist(\mathbf{q}, \mathbf{p})}{b_2 - b_1} \right\rceil$$

By adopting this approach, we can further remove this $\log L$ factor for using SAFE (cf. Theorem 2).

Theorem 2. *Suppose that we have L bandwidth values $b_1, b_2, ..., b_L$ and these bandwidth values have the same bandwidth gap, i.e., $b_2 - b_1 = b_3 - b_2 = ... = b_L - b_{L-1}$, the time complexity for using SAFE to solve $KDV_{explore}$ is $O(XY(n + L))$.*

Space complexity of SAFE: We proceed to discuss the space complexity for using SAFE to solve $\text{KDV}_{\text{explore}}$. Observe from Algorithm 3, since we need to return multiple KDVs for all bandwidth values $b_1, b_2,..., b_L$, it takes $O(XYL)$ space for storing the L planes $\mathcal{P}_1,...,\mathcal{P}_L$. Moreover, we need to use $O(n)$ space for storing the point set P. Since both the range query operation[2] (line 6), the share operation (line 7) and the aggregate operation (line 8) only consume at most $O(n)$ space throughout the loops (lines 3 to 4), we can conclude that SAFE only requires $O(XYL + n)$ space (cf. Theorem 3), which is the same as the baseline method (cf. Section 2.2).

Theorem 3. *The space complexity for using SAFE (cf. Algorithm 3) to solve $KDV_{explore}$ is $O(XYL + n)$.*

SAFE for other kernel functions: In previous discussion, we illustrate SAFE with Epanechnikov kernel function. Here, we extend SAFE for supporting other commonly-used kernel functions (cf. Table 1). Recall that one of the core ideas of our method (cf. Section 3.1) is to efficiently maintain the aggregate values for evaluating $\mathcal{F}_P^{(b)}(\mathbf{q})$ (e.g., the cardinality $|R_{\mathbf{q}}^{(b)}|$ and the aggregate distance $S_{R_{\mathbf{q}}^{(b)}}$ in Epanechnikov kernel). Like Equation 6, we can also decompose $\mathcal{F}_P^{(b)}(\mathbf{q})$ into different aggregate values for other kernel functions (cf. Table 1).

Triangular kernel: We can decompose $\mathcal{F}_P^{(b)}(\mathbf{q})$ in the following expression.

$$\mathcal{F}_P^{(b)}(\mathbf{q}) = w|R_{\mathbf{q}}^{(b)}| - \frac{w}{b} D_{R_{\mathbf{q}}^{(b)}}$$

where:

$$D_{R_{\mathbf{q}}^{(b)}} = \sum_{\mathbf{p} \in R_{\mathbf{q}}^{(b)}} dist(\mathbf{q}, \mathbf{p})$$

Quartic kernel: We can decompose $\mathcal{F}_P^{(b)}(\mathbf{q})$ in the following expression.

$$\mathcal{F}_P^{(b)}(\mathbf{q}) = w|R_{\mathbf{q}}^{(b)}| - \frac{2w}{b^2} S_{R_{\mathbf{q}}^{(b)}} + \frac{w}{b^4} Q_{R_{\mathbf{q}}^{(b)}}$$

where:

$$Q_{R_{\mathbf{q}}^{(b)}} = \sum_{\mathbf{p} \in R_{\mathbf{q}}^{(b)}} dist(\mathbf{q}, \mathbf{p})^4$$

Therefore, we can extend SAFE for supporting these kernel functions with the same time (cf. Theorem 1 and Theorem 2) and space complexity (cf. Theorem 3), based on the above aggregate values, which are summarized in Table 3.

Table 3: Aggregate values for all kernel functions.

Kernel	Aggregate values
Triangular	$\lvert R_{\mathbf{q}}^{(b)}\rvert$, $D_{R_{\mathbf{q}}^{(b)}}$
Epanechnikov	$\lvert R_{\mathbf{q}}^{(b)}\rvert$, $S_{R_{\mathbf{q}}^{(b)}}$
Quartic	$\lvert R_{\mathbf{q}}^{(b)}\rvert$, $S_{R_{\mathbf{q}}^{(b)}}$, $Q_{R_{\mathbf{q}}^{(b)}}$

4 SAFE FOR ON-THE-FLY BANDWIDTH VALUES

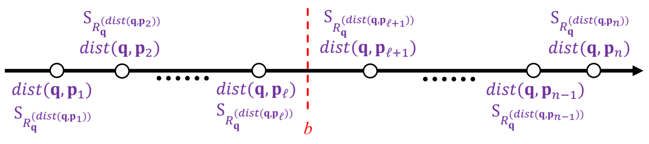
In Section 3, we illustrate how to use SAFE to efficiently generate multiple KDVs with L bandwidth values, which are known in advance. However, the users may not know the suitable set of bandwidth values and they can explore many bandwidth values on the fly. In this section, we propose two methods, namely SAFE_{all} (cf. Section 4.1) and SAFE_{exp} (cf. Section 4.2), which can solve $\text{KDV}_{\text{explore}}$

[2]Some range query solutions can take more than $O(n)$ space. Here, we refer to the commonly used index structures, e.g., kd-tree [9] and ball-tree [41, 43], which only take $O(n)$ space, for solving the range query.

(cf. Problem 1) with lower time complexity compared with the baseline method RQS, without knowing L bandwidth values in advance. We also discuss the practical implementation of these methods (cf. Section 4.3).

4.1 First Solution: SAFE$_{\text{all}}$

Unlike the share operation in Figure 3, we do not know those L bandwidth values in advance under this setting. Therefore, our core idea (cf. Figure 5) is to store the sorted distance values and the aggregate values, i.e., $dist(\mathbf{q}, \mathbf{p}_i)$ and $S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_i))}}$, respectively, for all data points $\mathbf{p}_i$, in each pixel $\mathbf{q}$. To simplify the notations, we let $dist(\mathbf{q}, \mathbf{p}_1) \leq dist(\mathbf{q}, \mathbf{p}_2) \leq ... \leq dist(\mathbf{q}, \mathbf{p}_n)$. Theoretically, different pixels $\mathbf{q}$ can induce different orders of distance values.



$S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_2))}}$
$dist(\mathbf{q},\mathbf{p}_2)$
$S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_{\ell+1}))}}$
$dist(\mathbf{q},\mathbf{p}_{\ell+1})$
$S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_n))}}$
$dist(\mathbf{q},\mathbf{p}_n)$
$dist(\mathbf{q},\mathbf{p}_1)$
$S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_1))}}$
$dist(\mathbf{q},\mathbf{p}_\ell)$
$S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_\ell))}}$
b
$dist(\mathbf{q},\mathbf{p}_{n-1})$
$S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_{n-1}))}}$

Figure 5: The core idea of our method SAFE$_{\text{all}}$.

Based on these aggregate values, we can compute $\mathcal{F}_P^{(b)}(\mathbf{q})$ with Epanechnikov kernel, for any bandwidth b (red dashed line in Figure 5), using Equation 11, which is the analogy of Equation 6.

$$\mathcal{F}_P^{(b)}(\mathbf{q}) = w \times \ell - \frac{w}{b^2} S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_\ell))}} \tag{11}$$

where $\mathbf{p}_\ell$ is the data point in P such that $dist(\mathbf{q}, \mathbf{p}_\ell)$ is the largest value that is smaller than b (cf. Figure 5).

Therefore, SAFE$_{\text{all}}$ follows these two steps, (1) store the distance values (with increasing order) and aggregate values for each pixel $\mathbf{q}$ (cf. Figure 5), (2) generate KDV for each bandwidth b on the fly.

In the first step, we can compute all distance values for each pixel $\mathbf{q}$, sort these distance values, and then compute the aggregate values, based on Equation 12, which is the analogy of Equation 10.

$$S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_i))}} = \begin{cases} S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_{i-1}))}} + dist(\mathbf{q}, \mathbf{p}_i)^2 & \text{if } i > 1 \\ dist(\mathbf{q}, \mathbf{p}_1)^2 & \text{if } i = 1 \end{cases} \tag{12}$$

In the second step, we need to first find the largest distance value $dist(\mathbf{q}, \mathbf{p}_\ell)$, which is just smaller than the bandwidth b (cf. Figure 5) and its corresponding aggregate value $S_{R_{\mathbf{q}}^{(dist(\mathbf{q},\mathbf{p}_\ell))}}$. Then, we use Equation 11 to obtain the kernel density value $\mathcal{F}_P^{(b)}(\mathbf{q})$, for each pixel $\mathbf{q}$.

Since there are $X \times Y$ pixels and the bottleneck in the first step is to perform the sorting operation, which takes $O(n \log n)$ time, the time complexity of the first step is $O(XYn \log n)$. In the second step, the most time-consuming operation is to find $\mathbf{p}_\ell$, which takes $O(\log n)$ time, using the binary search method. As such, the second step consumes $O(XY \log n)$ time for each bandwidth b. Therefore, we claim that SAFE$_{\text{all}}$ takes $O(XY(n + L) \log n)$ time for generating multiple KDVs with L bandwidth values (cf. Theorem 4).

Theorem 4. *The time complexity of SAFE$_{all}$ is $O(XY(n+L) \log n)$ for solving KDV$_{explore}$ without knowing those L bandwidth values in advance.*

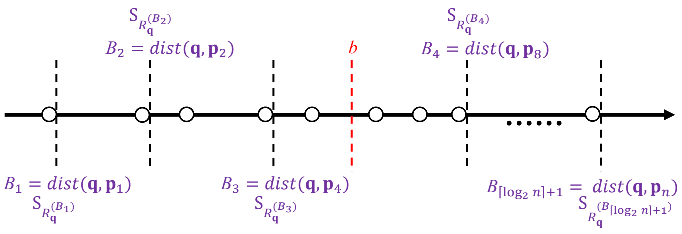
Compared with SAFE (cf. Section 3.2), this method SAFE$_{\text{all}}$ is inferior in terms of time complexity. However, SAFE$_{\text{all}}$ can outperform the RQS method when $L > \log n$. We argue that this is practical when the users need to continuously tune the bandwidth b to visualize the changes of KDV.

However, since each pixel needs to store n distance and aggregate values and we need to generate multiple KDVs with L bandwidth values, SAFE$_{\text{all}}$ consumes $O(XY(n+L))$ space (cf. Theorem 5), which is space-consuming in practice.

Theorem 5. *The space complexity of SAFE$_{all}$ is $O(XY(n + L))$.*

4.2 Second Solution: SAFE$_{\text{exp}}$

Observe that SAFE$_{\text{all}}$ takes $O(XY(n + L))$ space (cf. Theorem 5). Therefore, we ask a question: *Can we avoid storing n distance and aggregate values (cf. Figure 5) for each pixel* $\mathbf{q}$*, which can still provide the accurate answer for solving KDV$_{explore}$ (cf. Problem 1) without knowing all bandwidth values?* Here, we provide an affirmative answer to this question.



$S_{R_{\mathbf{q}}^{(B_2)}}$
$B_2 = dist(\mathbf{q},\mathbf{p}_2)$
b
$S_{R_{\mathbf{q}}^{(B_4)}}$
$B_4 = dist(\mathbf{q},\mathbf{p}_8)$
$B_1 = dist(\mathbf{q},\mathbf{p}_1)$
$S_{R_{\mathbf{q}}^{(B_1)}}$
$B_3 = dist(\mathbf{q},\mathbf{p}_4)$
$S_{R_{\mathbf{q}}^{(B_3)}}$
$B_{\lceil \log_2 n \rceil + 1} = dist(\mathbf{q},\mathbf{p}_n)$
$S_{R_{\mathbf{q}}^{(B_{\lceil \log_2 n \rceil + 1})}}$

Figure 6: The core idea of SAFE$_{\text{exp}}$.

Figure 6 summarizes the core idea of our method SAFE$_{\text{exp}}$. SAFE$_{\text{exp}}$ chooses $\lceil \log_2 n \rceil + 1$ bandwidth values, i.e., B_1, B_2,..., $B_{\lceil \log_2 n \rceil + 1}$, where:

$$B_i = dist(\mathbf{q}, \mathbf{p}_{\min(2^{i-1}, n)}) \tag{13}$$

These $\lceil \log_2 n \rceil + 1$ bandwidth values cover the exponential numbers, i.e., $1, 2, 4, 8, ..., n$, of data points in P, i.e.,

$$|R_{\mathbf{q}}^{(B_i)}| = \min(2^{i-1}, n) \tag{14}$$

Like Equation 10, we can also obtain the aggregate values $S_{R_{\mathbf{q}}^{(B_i)}}$ in all these bandwidth values B_i, using the concept of bandwidth gap (cf. Equation 7), where $B_0 = -\infty$ is the dummy bandwidth.

$$S_{R_{\mathbf{q}}^{(B_i)}} = \begin{cases} S_{R_{\mathbf{q}}^{(B_{i-1})}} + S_{G_{\mathbf{q}}^{(B_{i-1}, B_i)}} & \text{if } i > 1 \\ S_{G_{\mathbf{q}}^{(B_0, B_1)}} & \text{if } i = 1 \end{cases} \tag{15}$$

After we have computed the purple aggregate values (cf. Figure 6) for these $\lceil \log_2 n \rceil + 1$ bandwidth values, a simple idea for approximating $\mathcal{F}_P^{(b)}(\mathbf{q})$ with the on-the-fly bandwidth value b (e.g., red dashed line in Figure 6) is to return either $\mathcal{F}_P^{(B_\ell)}(\mathbf{q})$ or $\mathcal{F}_P^{(B_{\ell+1})}(\mathbf{q})$ (based on Equation 6) as an answer, where $B_\ell < b < B_{\ell+1}$. However, b can be anywhere in between B_ℓ and $B_{\ell+1}$ and $B_\ell << B_{\ell+1}$ theoretically (e.g., $B_\ell \rightarrow 0$ and $B_{\ell+1} \rightarrow \infty$). Therefore, both $\mathcal{F}_P^{(B_\ell)}(\mathbf{q})$ and $\mathcal{F}_P^{(B_{\ell+1})}(\mathbf{q})$ cannot provide the approximation guarantee for computing $\mathcal{F}_P^{(b)}(\mathbf{q})$.

Here, we describe how to obtain the approximate result of $\mathcal{F}_P^{(b)}(\mathbf{q})$ with an approximation ratio of 2. We first define the v-truncated kernel density function $\tau_P^{(b,v)}(\mathbf{q})$ (with bandwidth b) in Equation 16. Observe that $\tau_P^{(b,v)}(\mathbf{q})$ only calculates the distance values which are smaller than the value v (e.g., we only consider the five white dots before v for calculating $\tau_P^{(b,v)}(\mathbf{q})$ in Figure 7).

$$\tau_P^{(b,v)}(\mathbf{q}) = \sum_{\mathbf{p}\in R_{\mathbf{q}}^{(v)}} w \cdot \begin{cases} 1-\frac{1}{b^2}dist(\mathbf{q},\mathbf{p})^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

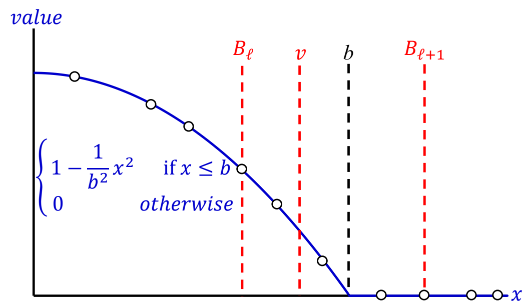

value
B_ℓ
v
b
$B_{\ell+1}$
$\begin{cases} 1-\frac{1}{b^2}x^2 & \text{if } x \le b \\ 0 & otherwise \end{cases}$
x

Figure 7: Illustration of v-truncated kernel density function, each white dot denotes the distance value $x = dist(\mathbf{q},\mathbf{p})$ for each p in P. The red dashed lines ($v, B_\ell, B_{\ell+1}$) are the possible truncated lines.

Observe from Figure 7, we claim that $\tau_P^{(b,v)}(\mathbf{q})$ exhibits the following two properties in Lemma 3.

Lemma 3. *The v-truncated kernel density function $\tau_P^{(b,v)}(\mathbf{q})$ exhibits these two properties:*

(1) $\tau_P^{(b,b)}(\mathbf{q}) = \mathcal{F}_P^{(b)}(\mathbf{q})$.

(2) $\tau_P^{(b,v)}(\mathbf{q})$ is monotonic increasing with respect to v.

Proof. Regarding the first property, we can simply substitute v by b in Equation 16. Then, based on Equation 4, we can prove the first property.

Regarding the second property, if $v_1 < v_2$, we have:

$$\tau_P^{(b,v_2)} = \sum_{\mathbf{p}\in R_{\mathbf{q}}^{(v_2)}} w \cdot \begin{cases} 1-\frac{1}{b^2}dist(\mathbf{q},\mathbf{p})^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases}$$
$$= \tau_P^{(b,v_1)} + \sum_{\mathbf{p}\in G_{\mathbf{q}}^{(v_1,v_2)}} w \cdot \begin{cases} 1-\frac{1}{b^2}dist(\mathbf{q},\mathbf{p})^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases}$$

Since the second term is non-negative, we conclude $\tau_P^{(b,v_1)} \le \tau_P^{(b,v_2)}$. Hence, we prove the second property. □

By adopting these two properties, we have the following inequality:

$$\tau_P^{(b,B_\ell)}(\mathbf{q}) \le \tau_P^{(b,b)}(\mathbf{q}) = \mathcal{F}_P^{(b)}(\mathbf{q}) \le \tau_P^{(b,B_{\ell+1})}(\mathbf{q}) \tag{17}$$

where we let b be in between B_ℓ and $B_{\ell+1}$ (e.g., b is in between B_3 and B_4 in Figure 6).

Here, we claim that $\tau_P^{(b,B_{\ell+1})}(\mathbf{q}) \le 2\tau_P^{(b,B_\ell)}(\mathbf{q})$ in Lemma 4.

Lemma 4. $\tau_P^{(b,B_{\ell+1})}(\mathbf{q}) \le 2\tau_P^{(b,B_\ell)}(\mathbf{q})$.

Proof. We first prove these two properties:

(1) Let $\mathbf{p}_r$ and $\mathbf{p}_g$ be two data points in $R_{\mathbf{q}}^{(B_\ell)}$ and $G_{\mathbf{q}}^{(B_\ell,B_{\ell+1})}$, respectively, we have $dist(\mathbf{q},\mathbf{p}_r) \le dist(\mathbf{q},\mathbf{p}_g)$.

(2) $|G_{\mathbf{q}}^{(B_\ell,B_{\ell+1})}| \le |R_{\mathbf{q}}^{(B_\ell)}|$.

We can easily prove the first property by comparing these two sets $R_{\mathbf{q}}^{(B_\ell)}$ (cf. Equation 3) and $G_{\mathbf{q}}^{(B_\ell,B_{\ell+1})}$ (cf. Equation 7).

Regarding the second property, since each bandwidth value covers exponential number, i.e., 1, 2, 4, 8, ..., n, of data points, we have:

$$\begin{aligned} |R_{\mathbf{q}}^{(B_{\ell+1})}| &\le 2|R_{\mathbf{q}}^{(B_\ell)}| \\ |R_{\mathbf{q}}^{(B_\ell)}| + |G_{\mathbf{q}}^{(B_\ell,B_{\ell+1})}| &\le 2|R_{\mathbf{q}}^{(B_\ell)}| \\ |G_{\mathbf{q}}^{(B_\ell,B_{\ell+1})}| &\le |R_{\mathbf{q}}^{(B_\ell)}| \end{aligned}$$

Based on these two properties, once we replace each $\mathbf{p}_g$ in $G_{\mathbf{q}}^{(B_\ell,B_{\ell+1})}$ by each $\mathbf{p}_r$ in $R_{\mathbf{q}}^{(B_\ell)}$, we have:

$$\begin{aligned} &\tau_P^{(b,B_{\ell+1})}(\mathbf{q}) \\ =\ & \tau_P^{(b,B_\ell)}(\mathbf{q}) + \sum_{\mathbf{p}_g\in G_{\mathbf{q}}^{(B_\ell,B_{\ell+1})}} w \cdot \begin{cases} 1-\frac{1}{b^2}dist(\mathbf{q},\mathbf{p}_g)^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases} \\ \le\ & \tau_P^{(b,B_\ell)}(\mathbf{q}) + \sum_{\mathbf{p}_r\in R_{\mathbf{q}}^{(B_\ell)}} w \cdot \begin{cases} 1-\frac{1}{b^2}dist(\mathbf{q},\mathbf{p}_r)^2 & \text{if } dist(\mathbf{q},\mathbf{p}) \le b \\ 0 & \text{otherwise} \end{cases} \\ =\ & 2\tau_P^{(b,B_\ell)}(\mathbf{q}) \end{aligned}$$

□

Based on the inequality (cf. Equation 17) and Lemma 4, once we use $\tau_P^{(b,B_\ell)}(\mathbf{q})$ as the approximate result of $\mathcal{F}_P^{(b)}(\mathbf{q})$, we can achieve the approximation ratio 2.

Theorem 6. *If $B_\ell \le b \le B_{\ell+1}$, $\tau_P^{(b,B_\ell)}(\mathbf{q})$ is the approximate value of $\mathcal{F}_P^{(b)}(\mathbf{q})$ with approximation ratio 2.*

Our method SAFE$_{\text{exp}}$ follows the same two-step method as SAFE$_{\text{all}}$ (cf. Section 4.1), except that we only store those $\lceil \log_2 n \rceil + 1$ distance and aggregate values (cf. Figure 6) in step 1 and compute $\tau_P^{(b,B_\ell)}(\mathbf{q})$, instead of $\mathcal{F}_P^{(b)}(\mathbf{q})$, in step 2. Therefore, we can conclude that the space complexity of SAFE$_{\text{exp}}$ is $O(XY(\log n + L) + n)$ (cf. Theorem 7).

Theorem 7. *The space complexity of SAFE$_{exp}$ is $O(XY(\log n + L) + n)$.*

Regarding the time complexity of SAFE$_{\text{exp}}$, step 1 still needs $O(XYn\log n)$ time for scanning and sorting all data points for each pixel. In step 2, the bottlenecks are to first adopt the binary search for finding B_ℓ, which takes $O(\log(\log n))$ time and then evaluate $\tau_P^{(b,B_\ell)}(\mathbf{q})$. Like Equation 6, we can compute $\tau_P^{(b,B_\ell)}(\mathbf{q})$ (with $b > B_\ell$) in $O(1)$ time, using those distance and aggregate values, where:

$$\tau_P^{(b,B_\ell)}(\mathbf{q}) = w|R_{\mathbf{q}}^{(B_\ell)}| - \frac{w}{b^2}S_{R_{\mathbf{q}}^{(B_\ell)}}$$

Therefore, we conclude that the time complexity for solving KDV$_{\text{explore}}$ is $O(XY(n\log n + L\log(\log n)))$.

Theorem 8. *The time complexity of SAFE$_{exp}$ is $O(XY(n\log n + L\log(\log n)))$ for solving KDV$_{explore}$ without knowing L bandwidth values in advance.*

4.3 Practical Implementation of SAFE$_{\text{all}}$ and SAFE$_{\text{exp}}$

In real applications, users (e.g., transportation experts [65, 68], criminologists [40], etc.) do not use the kernel function $K_b(\mathbf{q},\mathbf{p})$ with an extremely large bandwidth value b. Imagine that we have a crime event $\mathbf{p}$ in New York, we do not expect that this event can influence a location $\mathbf{q}$ in San Francisco. Therefore, the users can know the largest bandwidth value b_L for testing in advance. After they have specified the largest bandwidth value b_L, both SAFE$_{\text{all}}$ and SAFE$_{\text{exp}}$

only need to scan those data points $\mathbf{p}$ with distance values $dist(\mathbf{q}, \mathbf{p})$ smaller than b_L, for each pixel $\mathbf{q}$, which can further improve the practical efficiency and space consumption for these two methods.

5 EXPERIMENTAL EVALUATION

In this section, we first discuss the experimental settings in Section 5.1. Then, we show the experimental results for solving $\text{KDV}_{\text{explore}}$ (cf. Problem 1) with known bandwidth values in Section 5.2. After that, we further provide the experimental results for solving $\text{KDV}_{\text{explore}}$ with on-the-fly bandwidth values in Section 5.3. Lastly, we show the system prototype for supporting $\text{KDV}_{\text{explore}}$, using our solution SAFE, in Section 5.4.

5.1 Experimental Settings

We use four large-scale datasets (up to 4.33 million data points), which correspond to four different categories, for conducting the experiments. All these datasets are the open data from the local governments in different cities. Table 4 gives the details for each dataset. To conduct the experiments, we follow [12, 23] and utilize Scott's rule [54] to generate the bandwidth value b_S and randomly sample L bandwidth values from the range $[0.5b_S, 2b_S]$, based on the uniform distribution. By default, we choose $L = 20$ and the resolution size $X \times Y$ as 640×480. In addition, as discussed in Section 4.3, we also assume that the users can know the largest bandwidth value b_L in advance.

Table 4: Datasets.

Dataset	n	Category	Ref.
Chicago	237255	Environmental inspections	[3]
Seattle	839504	Crime events	[7]
New York	1499928	Traffic accidents	[5]
San Francisco	4333098	311 cases	[6]

In our experiments, we compare different state-of-the-art methods for solving $\text{KDV}_{\text{explore}}$, which are summarized in Table 5. SCAN is the sequential scan method for computing $\mathcal{F}_P^{(b)}(\mathbf{q})$ for each pixel $\mathbf{q}$. Both RQS_{kd} and RQS_{ball} are the range-query-based solutions (cf. Section 2.2), which adopt the kd-tree and ball-tree index structures, respectively. Z-order [72–74] is the data sampling method, which provides the probabilistic error guarantee for computing $\mathcal{F}_P^{(b)}(\mathbf{q})$. Both aKDE [25] and QUAD [12] approximate the kernel functions (cf. Table 1) with the simple curves, e.g., constant value and quadratic function, respectively, and combine with the indexing framework, to improve the efficiency for evaluating $\mathcal{F}_P^{(b)}(\mathbf{q})$. In contrast to these research studies, our methods SAFE and SAFE_{all} are the exact solutions, which can reduce the time complexity for solving $\text{KDV}_{\text{explore}}$ with known and on-the-fly bandwidth values, respectively. To further reduce the space consumption of SAFE_{all}, our SAFE_{exp} can provide the deterministic approximation ratio 2 for solving $\text{KDV}_{\text{explore}}$ with on-the-fly bandwidth values.

We implemented all 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 of all methods, respectively. Here, we only report the results with the response time smaller than 14400 sec (i.e., 4 hours) and the memory space smaller than 24GB. Due to space limitations, we do not include the experiment for testing the efficiency of all methods with the sequence of uniformly increasing bandwidth values.

5.2 Experiments for $\text{KDV}_{\text{explore}}$ with Known Bandwidth Values

Even though our method SAFE can reduce the time complexity for solving $\text{KDV}_{\text{explore}}$, we do not know the practical improvement for using this method compared with the existing solutions. In this section, we conduct the following experiments, where the bandwidth values are known in advance.

Response time of all methods with the default setting of parameters: We adopt the default parameters, i.e., $L = 20$ and $X \times Y = 640 \times 480$, and report the response time of all methods for solving $\text{KDV}_{\text{explore}}$ (cf. Figure 8). Since our method SAFE reduces the time complexity for solving $\text{KDV}_{\text{explore}}$ from $O(LXYn)$ to $O(XY(n \log L + L))$ (cf. Theorem 1), our method achieves at least 3.96x to 7.55x speedup compared with different state-of-the-art methods for generating KDVs with $L = 20$ bandwidth values.

Response time of all methods with different resolution sizes: We proceed to investigate how the resolution size $X \times Y$ affects the response time for generating multiple KDVs with $L = 20$. Here, we choose four resolution sizes, which are 160×120, 320×240, 640×480, and 1280×960. In Figure 9, once we increase the resolution size by 4 times (e.g., from 160×120 to 320×240), the response time of all methods also increases roughly by 4 times. Observe that no matter which resolution size we choose, our method SAFE consistently achieves better efficiency compared with other methods.

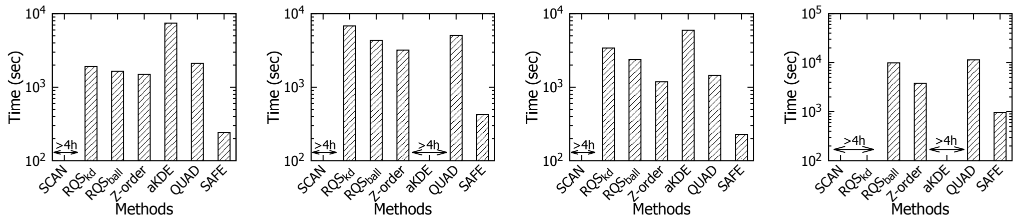
Response time of all methods with different dataset sizes: In this experiment, we test how the dataset sizes affect the response time of all methods. For each dataset, we obtain four sampled subsets, which contain 25%, 50%, 75% and 100% (original one) data points of the original dataset. In Figure 10, we observe that our method SAFE significantly outperforms the state-of-the-art methods by 3.96x to 12.3x, using different dataset sizes.

Response time of all methods with different numbers L of bandwidth values: We discuss how the number L of bandwidth values affects the response time of all methods. Observe from Figure 11, our method SAFE is not sensitive to the number L of bandwidth values. Therefore, the larger the number L, the larger the time gap between SAFE and other methods. The main reason is that the worst case time complexity of our method SAFE is $O(XY(n \log L + L))$ (cf. Theorem 1), which is much smaller than the time complexity of other methods $O(LXYn)$, especially for the large L.

Response time of all methods for other kernel functions: We proceed to investigate the response time of all methods, when we utilize other kernel functions (cf. Table 1), including triangular and quartic kernels. Here, we choose two largest datasets, i.e., New York and San Francisco, and adopt the default parameters, i.e., $L = 20$ and $X\times Y = 640\times480$, for testing. We report the response time of all methods in Figure 12. Since both existing methods and our method SAFE can be easily extended to support other kernel functions without incurring significant time overhead, the response time of all methods is similar to Figures 8c and 8d. As such, like the results in Epanechnikov kernel, our method SAFE can outperform the existing methods by a visible margin for both triangular and quartic kernels.

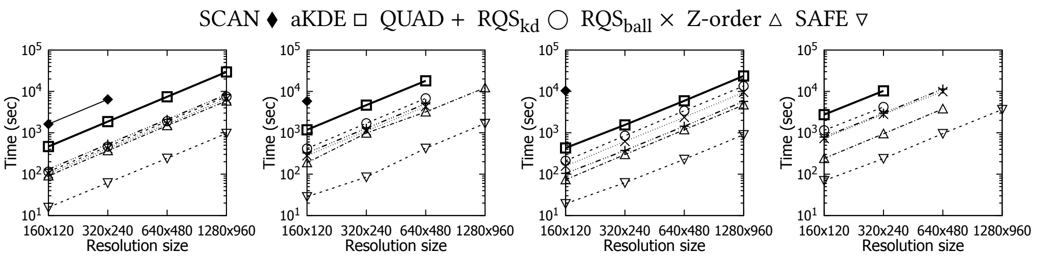
Table 5: Methods for solving $KDV_{explore}$.

Method	SCAN	RQS_{kd}	RQS_{ball}	Z-order	aKDE	QUAD	SAFE	$SAFE_{all}$	$SAFE_{exp}$
Ref.	[54]	Section 2.2 with [9]	Section 2.2 with [41]	[72]	[25]	[12]	Section 3.2	Section 4.1	Section 4.2



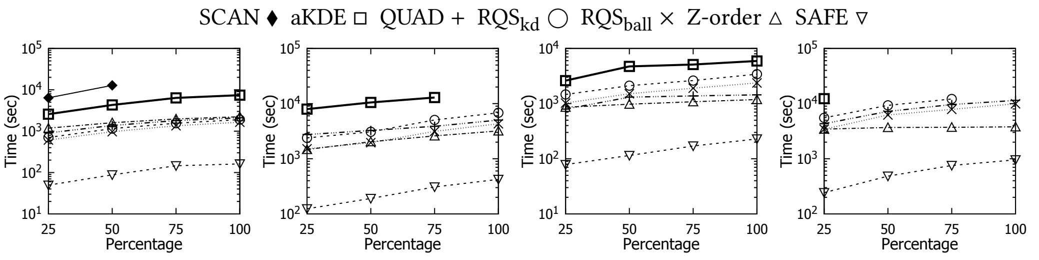
(a) Chicago (b) Seattle (c) New York (d) San Francisco

Figure 8: Response time of all methods, using the default parameters (L = 20 and $X \times Y$ = 640 × 480).



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

Figure 9: Response time of all methods with L = 20, varying the resolution sizes (from 160 × 120 to 1280 × 960).



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

Figure 10: Response time of all methods with L = 20 and $X \times Y$ = 640 × 480, varying the dataset sizes (sampling different percentages of the original datasets).

5.3 Experiments for $KDV_{explore}$ with On-The-Fly Bandwidth Values

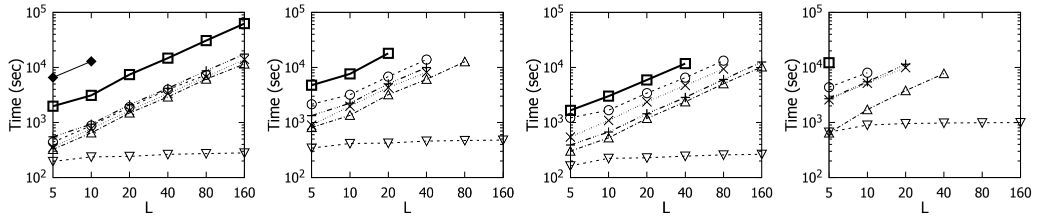
We further verify the efficiency of solving $KDV_{explore}$ with on-the-fly bandwidth values. Recall that $SAFE_{all}$ and $SAFE_{exp}$ need to consume more memory space (cf. Theorem 5 and Theorem 7, respectively) and $SAFE_{exp}$ provides the approximation result for solving $KDV_{explore}$ (cf. Theorem 6). Therefore, except for only performing experimental evaluation for the running time, we also test the memory space conumption and the visualization quality of all methods in this section. Here, we conduct the following experiments.

Response time of all methods with different numbers L of bandwidth values: We examine how the parameter L affects the response time of all methods (cf. Figure 13). Since the methods SCAN, aKDE, QUAD, RQS_{kd}, RQS_{ball}, Z-order can be seamlessly adopted for solving $KDV_{explore}$ with on-the-fly bandwidth values, the response time of these methods is the same as the one in Figure 11. Observe that our best method $SAFE_{exp}$ outperforms the state-of-the-art methods by a visible margin, due to the small time complexity of these methods. Here, we omit the results of $SAFE_{all}$ in both Seattle and San Francisco datasets, since this method consumes more than 24GB memory space.

Memory space consumption of all methods with different resolution sizes: We test how the resolution size affects the memory space consumption of all methods. In Figure 14, we observe that $SAFE_{all}$ is more space-consuming compared with other methods. The main reason is that $SAFE_{all}$ needs to store n distance and aggregate values for each pixel in the worst case, which takes $O(XYn)$ memory space (cf. Theorem 5). Since $SAFE_{exp}$ only takes $O(\log n)$ additional space for each pixel, $SAFE_{exp}$ can significantly reduce the space consumption compared with $SAFE_{all}$. Here, we further observe that $SAFE_{exp}$ achieves competitive memory consumption compared with other methods, regardless of the resolution size.

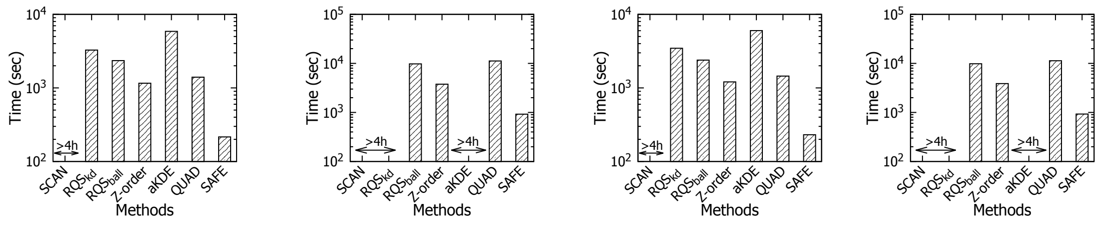
Memory space consumption of all methods with different dataset sizes: We further investigate how different dataset sizes

SCAN ♦ aKDE □ QUAD + RQS_{kd} ○ RQS_{ball} × Z-order △ SAFE ▽



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

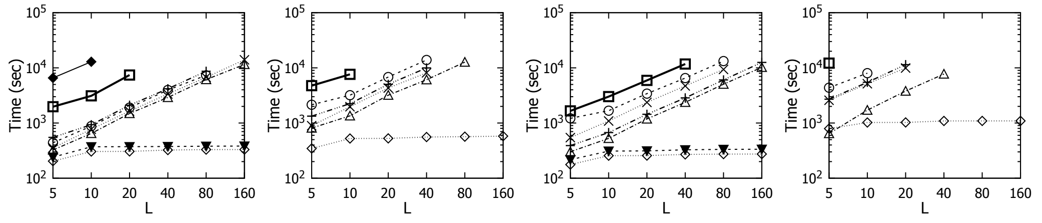
Figure 11: Response time of all methods with $X \times Y = 640 \times 480$, varying the number L of bandwidth values.



(a) New York (Triangular) (b) San Francisco (Triangular) (c) New York (Quartic) (d) San Francisco (Quartic)

Figure 12: Response time of all methods for other kernel functions, including triangular (a and b) and quartic kernels (c and d), in the New York (a and c) and San Francisco (b and d) datasets, using the default parameters ($L = 20$ and $X \times Y = 640 \times 480$).

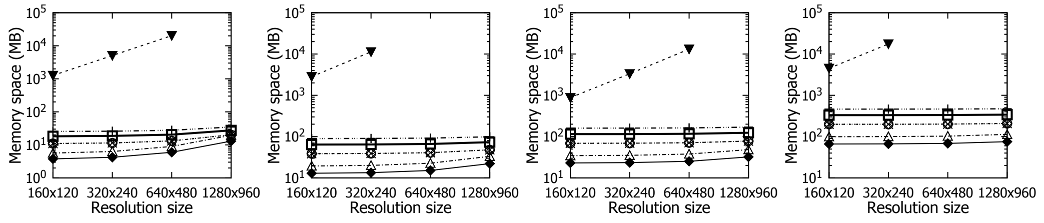
SCAN ♦ aKDE □ QUAD + RQS_{kd} ○ RQS_{ball} × Z-order △ $SAFE_{all}$ ▼ $SAFE_{exp}$ ◇



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

Figure 13: Response time of all methods with $X \times Y = 640 \times 480$, varying the number L of on-the-fly bandwidth values.

SCAN ♦ aKDE □ QUAD + RQS_{kd} ○ RQS_{ball} × Z-order △ $SAFE_{all}$ ▼ $SAFE_{exp}$ ◇



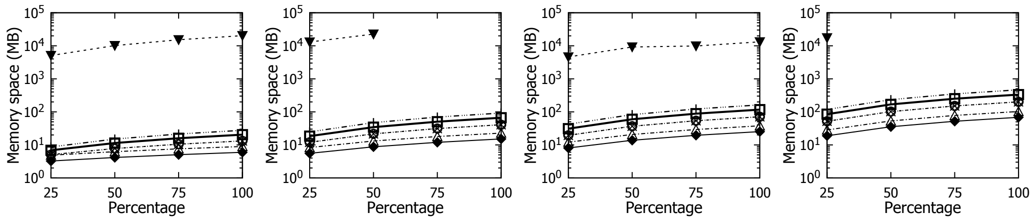
(a) Chicago (b) Seattle (c) New York (d) San Francisco

Figure 14: Memory space (MB) consumption of all methods with $L = 20$, varying the resolution sizes.

affect the memory space consumption of all methods (cf. Figure 15). Observe that the larger the dataset size, the larger the memory space consumption of all methods. Compared with other methods, our method $SAFE_{exp}$ achieves competitive memory space consumption, no matter which dataset size we adopt.

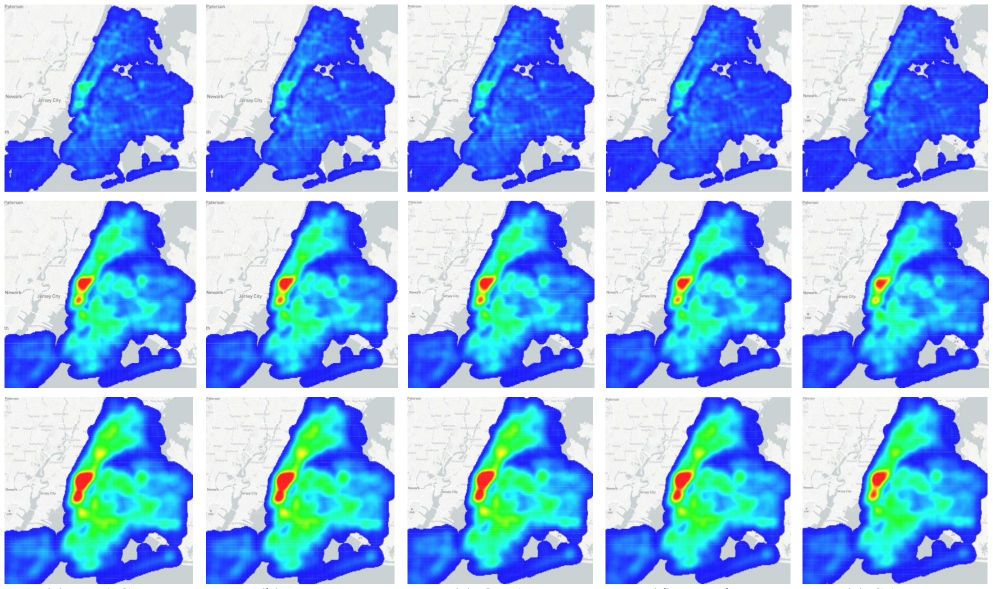
Visualization quality of all approximation methods: We proceed to compare the visualization quality of different approximation methods, including aKDE, QUAD, Z-order, and $SAFE_{exp}$, with the exact solution (EXACT). In this experiment, we adopt the default setting of parameters (i.e., $X \times Y = 640 \times 480$ and $L = 20$) and select three KDVs, which correspond to the 25^{th}, 50^{th} and 75^{th} smallest bandwidth values, for each method. Figure 16 reports the visual results of all methods. Since all approximation methods have the approximation guarantees (e.g., Theorem 6 in $SAFE_{exp}$), they yield similar visualization results compared with exact methods (e.g., SCAN), no matter which bandwidth value we choose.

SCAN ♦ aKDE □ QUAD + RQS_{kd} ○ RQS_{ball} × Z-order △ SAFE_{all} ▼ SAFE_{exp} ◇



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

Figure 15: Memory space consumption of all methods with $L = 20$ and $X \times Y = 640 \times 480$, varying the dataset sizes (sampling different percentages of the original datasets).



(a) EXACT (b) aKDE (c) QUAD (d) Z-order (e) SAFE_{exp}

Figure 16: Visual quality of the exact and approximation methods with the 25th (upper figures), 50th (middle figures) and 75th (lower figures) smallest bandwidth values.

5.4 System Prototype with SAFE for $\text{KDV}_{\text{explore}}$

In practice, domain experts need to (1) select the reasonable hotspot map and (2) understand how the hotspots change with different bandwidth values (e.g., they need to determine the risk of different regions). Therefore, we develop a system prototype[3] for supporting these two tasks. Figure 17 summarizes the general idea of this system prototype. First, domain experts need to specify L bandwidth values (e.g., $L = 80$ in our demonstration (cf. footnote 3)). Then, our solution, SAFE, generates KDVs for these bandwidth values. Lastly, domain experts can continuously slide the bandwidth bar to visualize the changes in the hotspot map. In addition, this system prototype also offers the plot of density value versus bandwidth to further help domain experts to determine the risk in different regions. Since our solution, SAFE, is much faster than the state-of-the-art solutions, domain experts can generate multiple KDVs with more bandwidth values.

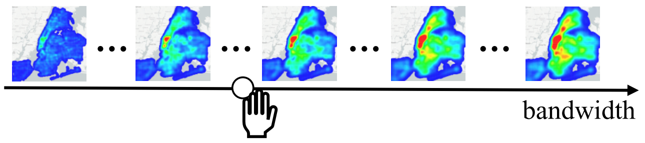


Figure 17: A system prototype to display multiple KDVs in the New York traffic accident dataset with different bandwidth values.

6 RELATED WORK

Kernel density visualization (KDV) has been the de facto method to identify the hotspots in different applications, including crime hotspot detection [11, 28, 40, 51] and traffic accident hotspot detection [60, 65, 68]. In these applications, domain experts need to generate multiple hotspot maps, by varying different bandwidth values [8, 11, 60, 68], which we term this problem as $\text{KDV}_{\text{explore}}$ (cf. Problem 1). However, generating multiple KDVs with different bandwidth values can be very time-consuming, which takes $O(LXYn)$ time in the worst case with L bandwidth values. In this section, we review five camps of research studies, which are mostly related to this work.

Efficient methods for computing a single KDV: Since KDV is a computationally expensive operation, many algorithms have been developed to boost the efficiency for computing KDV. Raykar et

[3]The demonstration video can be found at https://github.com/edisonchan2013928/SAFE/blob/main/supplementary_files/SAFE_bandwidth_exploration.mp4.

al. [50] and Yang et al. [67] utilize the fast Gauss transform to approximate the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$ (cf. Equation 1), which can improve the efficiency for generating KDV. However, this approach cannot provide the approximation guarantee between the returned result and the density value $\mathcal{F}_P^{(b)}(\mathbf{q})$. Chan et al. [12, 14, 16, 17], Gan et al. [23], and Gray et al. [25] develop the bound functions to approximate $\mathcal{F}_P^{(b)}(\mathbf{q})$. By combining the indexing structures (e.g., ball-tree [41] and kd-tree [9]) with these bound functions, they can boost the efficiency for generating KDV. However, these methods cannot reduce the worst case time complexity for generating KDV. Zheng et al. [72, 74] and Phillips et al. [44–46] develop the data sampling method to wisely sample the original dataset. Then, they apply the exact KDV (with slight modification of the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$) for this reduced dataset. Even though this approach can improve the efficiency for generating a single KDV, they still need to evaluate the exact KDV for the reduced dataset, which can still be time-consuming for generating multiple KDVs. As a remark, since these data sampling methods are orthogonal to this work, we can combine our methods and these methods to further improve the efficiency for solving KDV$_{\text{explore}}$.

Range-query-based methods: Recall from Section 2.2 that we can evaluate the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$ (cf. Equation 4) for each pixel $\mathbf{q}$ based on the range-query-solution set $R_{\mathbf{q}}^{(b)}$ (cf. Equation 3). In the literature, many efficient index structures [20, 52, 70] have been developed for supporting range queries. Among most of these index structures, both kd-tree [9] and ball-tree [41] are the most popular ones (supported by Scikit-learn [43]), which can efficiently solve the range queries in low-dimensional datasets. Unlike our method SAFE, even though these index structures can improve the efficiency for generating a single KDV, this approach cannot reduce the worst case time complexity (i.e., $O(LXYn)$ time) for solving KDV$_{\text{explore}}$ (cf. Section 2.2).

Bandwidth selection methods: Bandwidth selection is a long-studied problem in the statistics [24, 30, 32, 54, 56, 58, 59, 61] and data mining [47, 49, 74] communities. In these four decades, many algorithms have been proposed to choose the best bandwidth value for the kernel density estimation model. However, choosing the best bandwidth value is subjective [11, 60, 68], which depends on different applications and datasets. As such, there is no one-size-fit-all bandwidth selection method for finding the best bandwidth value. In addition, due to a huge amount of research work in this topic, it is hard for domain experts to directly select the correct bandwidth selection method without tuning the bandwidth values. For example: Yu et al. [68] adopt the rule-of-thumb approach [54] to select the bandwidth value of the kernel function and then multiply this value by different constants for generating multiple hotspot maps. Compared with the above bandwidth selection methods, our method SAFE offers the users to efficiently tune the bandwidth values for obtaining multiple KDVs.

Multiple-query optimization (or batch query processing): The general idea of our method SAFE is to first compute KDV with the largest bandwidth value b_L and share its information to multiple KDVs with smaller bandwidth values, i.e., b_1, b_2,..., and b_{L-1}, which can be regarded as a kind of multiple-query optimization (MQO). Even though MQO has been extensively studied in the database community for improving the efficiency in different types of queries, e.g., relational database queries [31, 55, 62], spatial queries [34, 64, 71], and graph queries [18, 19], none of these studies focuses on the kernel density function $\mathcal{F}_P^{(b)}(\mathbf{q})$. As such, all these techniques cannot be adopted to solve the KDV$_{\text{explore}}$ problem.

Spatial visualization tools: Recently, many spatial visualization tools [14, 21, 22, 26, 27, 35, 38, 39, 69] have been proposed in both database and visualization communities. Among these spatial visualization tools, KDV-Explorer [14], HadoopViz [22], and GeoSparkViz [69] can support the generation of multiple KDVs. However, KDV-Explorer [14] (based on QUAD [12]) is not scalable to large resolution sizes (e.g., 640×480). In addition, even though HadoopViz [22] and GeoSparkViz [69] use a computer cluster to efficiently support generating KDV with large-scale datasets, these methods consume many computational resources, which may not be suitable for domain experts who normally adopt the QGIS [48] and ArcGIS [1] software packages and do not necessarily have many computational resources. As a remark, our methods can also be fully parallelized to further boost the efficiency of generating multiple KDVs. The details can be found in this supplementary document https://github.com/edisonchan2013928/SAFE/blob/main/supplementary_files/SAFE_parallel.pdf.

7 CONCLUSION

KDV is an important operation, which has been extensively used in many spatial analysis tasks. In these tasks, domain experts typically generate multiple KDVs by tuning different bandwidth values. However, generating multiple KDVs is time-consuming, which is not scalable to a large number L of bandwidth values and million-scale datasets. Therefore, we propose a share-and-aggregate framework, called SAFE, which reduces the time complexity for generating multiple KDVs with L bandwidth values from $O(LXYn)$ to $O(XY(n \log L + L))$ (cf. Theorem 1), given that those L bandwidth values are known in advance. To tackle on-the-fly bandwidth values, we extend SAFE and develop the exact method SAFE$_{\text{all}}$ and 2-approximation method SAFE$_{\text{exp}}$, which also reduce the time complexity (cf. Theorem 4 and Theorem 8) and retain the visualization quality (cf. Theorem 6). Our experimental results on four large-scale datasets show that all methods SAFE, SAFE$_{\text{all}}$, and SAFE$_{\text{exp}}$ provide at least one-order-of-magnitude speedup over different state-of-the-art methods in most of the cases. In addition, our approximation method SAFE$_{\text{exp}}$ retains the good visualization quality in practice.

In the future, we plan to extend our methods to support other types of KDV, e.g., NKDV [15] and STKDV [13]. In addition, we will study other types of statistical and machine learning models, e.g., kernel smoothing [29] and kernel PCA [53].

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Plan of China (No.2019YFB2102100), the Science and Technology Development Fund Macau (SKL-IOTSC-2021-2023, 0015/2019/AKP), University of Macau (MYRG2019-00119-FST), IRCMs/19-20/H01, RIF R2002-20F, Research Grants Council of Hong Kong (projects 12200021, 12201520, C6030-18GF), and Guangdong Basic and Applied Basic Research Foundation (Project No. 2019B1515130001).

REFERENCES

[1] [n. d.]. ArcGIS. http://pro.arcgis.com/en/pro-app/tool-reference/spatial-analyst/how-kernel-density-works.htm (last accessed: 1st May 2021).

[2] [n. d.]. Atlanta Police Department Open Data. http://opendata.atlantapd.org/ (last accessed: 1st May 2021).

[3] [n. d.]. Chicago Data Portal. https://data.cityofchicago.org/Environment-Sustainable-Development/CDPH-Environmental-Inspections/i9rk-duva (last accessed: 1st May 2021).

[4] [n. d.]. IKCEST: Disaster Risk Reduction. http://drr.ikcest.org/knowledge_service/ncp.html (last accessed: 1st May 2021).

[5] [n. d.]. NYC Open Data. https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95 (last accessed: 1st May 2021).

[6] [n. d.]. San Francisco Open Data. https://data.sfgov.org/City-Infrastructure/311-Cases/vw6y-z8j6 (last accessed: 1st May 2021).

[7] [n. d.]. Seattle Open Data. https://data.seattle.gov/Public-Safety/SPD-Crime-Data-2008-Present/tazs-3rd5 (last accessed: 1st May 2021).

[8] Giuseppe Amatulli, Fernando Peréz-Cabello, and Juan de la Riva. 2007. Mapping lightning/human-caused wildfires occurrence under ignition point location uncertainty. *Ecological Modelling* 200, 3 (2007), 321 – 333. https://doi.org/10.1016/j.ecolmodel.2006.08.001

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

[10] Michal Bíl, Richard Andrášik, and Zbyněk Janoška. 2013. Identification of hazardous road locations of traffic accidents by means of kernel density estimation and cluster significance evaluation. *Accident Analysis & Prevention* 55 (2013), 265–273. https://doi.org/10.1016/j.aap.2013.03.003

[11] Spencer Chainey, Lisa Tompson, and Sebastian Uhlig. 2008. The Utility of Hotspot Mapping for Predicting Spatial Patterns of Crime. *Security Journal* 21, 1 (01 Feb 2008), 4–28. https://doi.org/10.1057/palgrave.sj.8350066

[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. SWS: A Complexity-Optimized Solution for Spatial-Temporal Kernel Density Visualization. *Proc. VLDB Endow. (To appear)* (2022).

[14] Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Weng Hou Tong, Shivansh Mittal, Ye Li, and Reynold Cheng. 2021. KDV-Explorer: A Near Real-Time Kernel Density Visualization System for Spatial Analysis. *Proc. VLDB Endow.* 14, 12 (2021), 2655–2658.

[15] Tsz Nam Chan, Zhe Li, Leong Hou U, Jianliang Xu, and Reynold Cheng. 2021. Fast Augmentation Algorithms for Network Kernel Density Visualization. *Proc. VLDB Endow.* 14, 9 (2021), 1503–1516. http://www.vldb.org/pvldb/vol14/p1503-chan.pdf

[16] Tsz Nam Chan, Leong Hou U, Reynold Cheng, Man Lung Yiu, and Shivansh Mittal. 2020. Efficient Algorithms for Kernel Aggregation Queries. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

[17] 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

[18] Lu Chen, Chengfei Liu, Xiaochun Yang, Bin Wang, Jianxin Li, and Rui Zhou. 2016. Efficient Batch Processing for Multiple Keyword Queries on Graph Data. In *CIKM*. ACM, 1261–1270. https://doi.org/10.1145/2983323.2983806

[19] James Cheng, Yiping Ke, Ada Wai-Chee Fu, and Jeffrey Xu Yu. 2011. Fast graph query processing with a low-cost index. *VLDB J.* 20, 4 (2011), 521–539. https://doi.org/10.1007/s00778-010-0212-8

[20] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. 2008. *Computational geometry: algorithms and applications, 3rd Edition.* Springer. https://www.worldcat.org/oclc/227584184

[21] Liming Dong, Qiushi Bai, Taewoo Kim, Taiji Chen, Weidong Liu, and Chen Li. 2020. Marviq: Quality-Aware Geospatial Visualization of Range-Selection Queries Using Materialization. In *SIGMOD*. ACM, 67–82. https://doi.org/10.1145/3318464.3389730

[22] Ahmed Eldawy, Mohamed F. Mokbel, and Christopher Jonathan. 2016. HadoopViz: A MapReduce framework for extensible visualization of big spatial data. In *ICDE*. IEEE Computer Society, 601–612. https://doi.org/10.1109/ICDE.2016.7498274

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

[24] A. Gramacki. 2017. *Nonparametric Kernel Density Estimation and Its Computational Aspects.* Springer International Publishing. https://books.google.com.hk/books?id=PCpEDwAAQBAJ

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

[26] Tao Guo, Kaiyu Feng, Gao Cong, and Zhifeng Bao. 2018. Efficient Selection of Geospatial Data on Maps for Interactive and Visualized Exploration. In *SIGMOD*. 567–582. https://doi.org/10.1145/3183713.3183738

[27] Tao Guo, Mingzhao Li, Peishan Li, Zhifeng Bao, and Gao Cong. 2018. POIsam: a System for Efficient Selection of Large-scale Geospatial Data on Maps. In *SIGMOD*. 1677–1680. https://doi.org/10.1145/3183713.3193549

[28] 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

[29] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition.* Springer. https://doi.org/10.1007/978-0-387-84858-7

[30] M. C. Jones, J. S. Marron, and S. J. Sheather. 1996. A Brief Survey of Bandwidth Selection for Density Estimation. *J. Amer. Statist. Assoc.* 91, 433 (1996), 401–407. https://doi.org/10.1080/01621459.1996.10476701

[31] Tarun Kathuria and S. Sudarshan. 2017. Efficient and Provable Multi-Query Optimization. In *PODS*. ACM, 53–67. https://doi.org/10.1145/3034786.3034792

[32] K. B. Kulasekera and W. J. Padgett. 2006. Bayes bandwidth selection in kernel density estimation with censored data. *Journal of Nonparametric Statistics* 18, 2 (2006), 129–143. https://doi.org/10.1080/10485250600556744

[33] P.C. Lai, Chit-Ming Wong, Anthony Hedley, S.V. Lo, P.Y. Leung, J Kong, and G.M. Leung. 2004. Understanding the Spatial Clustering of Severe Acute Respiratory Syndrome (SARS) in Hong Kong. *Environmental health perspectives* 112 (12 2004), 1550–6. https://doi.org/10.1289/ehp.7117

[34] Lei Li, Mengxuan Zhang, Wen Hua, and Xiaofang Zhou. 2020. Fast Query Decomposition for Batch Shortest Path Processing in Road Networks. In *ICDE*. IEEE, 1189–1200. https://doi.org/10.1109/ICDE48307.2020.00107

[35] Mingzhao Li, Zhifeng Bao, Farhana Murtaza Choudhury, and Timos Sellis. 2018. Supporting Large-scale Geographical Visualization in a Multi-granularity Way. In *WSDM*. 767–770. https://doi.org/10.1145/3159652.3160587

[36] Yu-Pin Lin, Hone-Jay Chu, Chen-Fa Wu, Tsun-Kuo Chang, and Chiu-Yang Chen. 2011. Hotspot Analysis of Spatial Environmental Pollutants Using Kernel Density Estimation and Geostatistical Techniques. *International Journal of Environmental Research and Public Health* 8, 1 (2011), 75–88. https://doi.org/10.3390/ijerph8010075

[37] Yajie Ma, Mark Richards, Moustafa Ghanem, Yike Guo, and John Hassard. 2008. Air Pollution Monitoring and Mining Based on Sensor Grid in London. *Sensors* 8, 6 (2008), 3601–3623. https://doi.org/10.3390/s80603601

[38] A. Mayorga and M. Gleicher. 2013. Splatterplots: Overcoming Overdraw in Scatter Plots. *IEEE Transactions on Visualization and Computer Graphics* 19, 9 (Sept 2013), 1526–1538. https://doi.org/10.1109/TVCG.2013.65

[39] Luana Micallef, Gregorio Palmas, Antti Oulasvirta, and Tino Weinkauf. 2017. Towards Perceptual Optimization of the Visual Design of Scatterplots. *IEEE Trans. Vis. Comput. Graph.* 23, 6 (2017), 1588–1599. https://doi.org/10.1109/TVCG.2017.2674978

[40] George Mohler. 2014. Marked point process hotspot maps for homicide and gun crime prediction in Chicago. *International Journal of Forecasting* 30, 3 (2014), 491 – 497. https://doi.org/10.1016/j.ijforecast.2014.01.004

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

[42] 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

[43] 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.

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

[45] 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

[46] 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

[47] Abdulhakim Ali Qahtan, Suojin Wang, and Xiangliang Zhang. 2017. KDE-Track: An Efficient Dynamic Density Estimator for Data Streams. *IEEE Trans. Knowl. Data Eng.* 29, 3 (2017), 642–655. https://doi.org/10.1109/TKDE.2016.2626441

[48] QGIS Development Team. 2009. *QGIS Geographic Information System.* Open Source Geospatial Foundation. http://qgis.osgeo.org

[49] Vikas C. Raykar and Ramani Duraiswami. 2006. Fast optimal bandwidth selection for kernel density estimation. In *SDM*. SIAM, 524–528.

[50] Vikas C. Raykar, Ramani Duraiswami, and Linda H. Zhao. 2010. Fast Computation of Kernel Estimators. *Journal of Computational and Graphical Statistics* 19, 1 (2010), 205–220. https://doi.org/10.1198/jcgs.2010.09046

[51] 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

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

[53] Bernhard Schölkopf and Alexander Johannes Smola. 2002. *Learning with Kernels: support vector machines, regularization, optimization, and beyond.* MIT Press.

http://www.worldcat.org/oclc/48970254

[54] D.W. Scott. 1992. *Multivariate Density Estimation: Theory, Practice, and Visualization.* Wiley. https://books.google.com.hk/books?id=7crCUS_F2ocC

[55] Timos K. Sellis. 1988. Multiple-Query Optimization. *ACM Trans. Database Syst.* 13, 1 (1988), 23–52. https://doi.org/10.1145/42201.42203

[56] S. J. Sheather and M. C. Jones. 1991. A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation. *Journal of the Royal Statistical Society: Series B (Methodological)* 53, 3 (1991), 683–690. https://doi.org/10.1111/j.2517-6161.1991.tb01857.x arXiv:https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1991.tb01857.x

[57] Xun Shi, Meifang Li, Olivia Hunter, Bart Guetti, Angeline Andrew, Elijah Stommel, Walter Bradley, and Margaret Karagas. 2019. Estimation of environmental exposure: interpolation, kernel density estimation or snapshotting. *Annals of GIS* 25, 1 (2019), 1–8. https://doi.org/10.1080/19475683.2018.1555188 PMID: 30687456.

[58] B.W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis.* Taylor & Francis. https://books.google.com.hk/books?id=e-xsrjsL7WkC

[59] J.S. Simonoff. 2012. *Smoothing Methods in Statistics.* Springer New York. https://books.google.com.hk/books?id=dgHaBwAAQBAJ

[60] 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

[61] M.P. Wand and M.C. Jones. 1994. *Kernel Smoothing.* Taylor & Francis. https://books.google.com.hk/books?id=GTOOi5yE008C

[62] Song Wang, Elke A. Rundensteiner, Samrat Ganguly, and Sudeept Bhatnagar. 2006. State-Slice: New Paradigm of Multi-query Optimization of Window-based Stream Queries. In *VLDB*. ACM, 619–630. http://dl.acm.org/citation.cfm?id=1164181

[63] Zuyuan Wang, Christian Ginzler, and Lars T. Waser. 2020. Assessing structural changes at the forest edge using kernel density estimation. *Forest Ecology and Management* 456 (2020), 117639. https://doi.org/10.1016/j.foreco.2019.117639

[64] Dingming Wu, Man Lung Yiu, Gao Cong, and Christian S. Jensen. 2012. Joint Top-K Spatial Keyword Query Processing. *IEEE Trans. Knowl. Data Eng.* 24, 10 (2012), 1889–1903. https://doi.org/10.1109/TKDE.2011.172

[65] 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

[66] Liting Xu, Shuhe Zhao, Sophia Shuang Chen, Cheng Yu, and Buyun Lei. 2020. Analysis of arable land distribution around human settlements in the riparian area of Lake Tanganyika in Africa. *Applied Geography* 125 (2020), 102344. https://doi.org/10.1016/j.apgeog.2020.102344

[67] Changjiang Yang, Ramani Duraiswami, and Larry S. Davis. 2004. Efficient Kernel Machines Using the Improved Fast Gauss Transform. In *NIPS*. 1561–1568. http://papers.nips.cc/paper/2550-efficient-kernel-machines-using-the-improved-fast-gauss-transform

[68] Hao Yu, Pan Liu, Jun Chen, and Hao Wang. 2014. Comparative analysis of the spatial analysis methods for hotspot identification. *Accident Analysis and Prevention* 66 (2014), 80 – 88. https://doi.org/10.1016/j.aap.2014.01.017

[69] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. 2018. GeoSparkViz: a scalable geospatial data visualization framework in the apache spark ecosystem. In *SSDBM*. ACM, 15:1–15:12. https://doi.org/10.1145/3221269.3223040

[70] 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

[71] M. Zhang, L. Li, W. Hua, and X. Zhou. 2020. Stream Processing of Shortest Path Query in Dynamic Road Networks. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1. https://doi.org/10.1109/TKDE.2020.3010005

[72] 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.

[73] Y. Zheng, Y. Ou, A. Lex, and J. M. Phillips. 2021. Visualization of Big Spatial Data Using Coresets for Kernel Density Estimates. *IEEE Transactions on Big Data* 7, 03 (2021), 524–534. https://doi.org/10.1109/TBDATA.2019.2913655

[74] 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