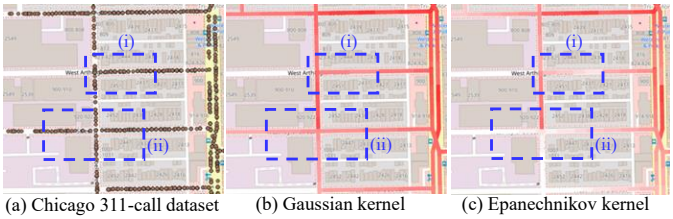
PLAN: Fast and Approximate Gaussian Kernel Density Visualization in Road Networks

Tsz Nam Chan*, Hongwei Ye*, Bojian Zhu†, Leong Hou U‡, Dingming Wu*, Ruisheng Wang§, Joshua Zhexue Huang*
*College of Computer Science and Software Engineering, Shenzhen University
{edisonchan, dingming, zx.huang}@szu.edu.cn, yehongwei2023@email.szu.edu.cn
†Department of Computer Science, Hong Kong Baptist University
csbjzhu@comp.hkbu.edu.hk
‡Department of Computer and Information Science, University of Macau
ryanlhu@um.edu.mo
§School of Architecture and Urban Planning, Shenzhen University
ruiswang@szu.edu.cn

***Abstract*—Network Kernel Density Visualization (NKDV) is a widely used spatial analysis tool in various communities, including transportation science, criminology, and urban planning. However, generating NKDV is very time-consuming, which does not scale to support large-scale datasets. Although many efficient algorithms have been developed for supporting this tool, most of these algorithms cannot be used for handling the most popular Gaussian kernel function. To tackle this issue, we first develop the pioneering Piecewise-Linear Approximate solutioN (PLAN), which can simultaneously (1) reduce the time complexity, (2) retain the similar space complexity, and (3) achieve the accuracy guarantee for generating NKDV with the Gaussian kernel. By providing further optimization for PLAN, our best method, called PLAN+, can achieve the lowest time complexity for supporting this tool. Experiment results on four large-scale datasets verify that both PLAN and PLAN+ achieve 32.47x to 2936.88x speedups, only incur 1.26x to 2.15x space overhead, and provide accurate NKDV results compared with the state-of-the-art solution (SOTA). The implementation of all methods can be found in https://github.com/edisonchan2013928/PLAN.**

I. Introduction

Network Kernel Density Visualization (NKDV) [1]–[5] has been frequently adopted in many communities, e.g., transportation science, criminology, and urban planning. Transportation scientists and criminologists [3], [5]–[17] adopt NKDV to investigate traffic/traffic accident and crime hotspots, respectively. Urban planners [6], [18]–[24] adopt NKDV to either examine the human mobility or analyze distributions of different points of interest (e.g., stores) in order to provide policy for the government. Figure 1b shows the hotspot map (based on NKDV with the Gaussian kernel) in a road network for the Chicago 311-call location dataset [25] (data points in Figure 1a). Observe that the deep red and light red regions denote the hotspots and coldspots, respectively.



(a) Chicago 311-call dataset (b) Gaussian kernel (c) Epanechnikov kernel

Fig. 1: Generating NKDV for the Chicago 311-call location dataset (in a) with the Gaussian kernel (in b) and the Epanechnikov kernel (in c), where we zoom in to the region near the West Arthington Street of Chicago.

In order to generate this type of visualization (like Figure 1b), domain experts first divide each road into a set of lixels[1] and then color them based on the network kernel density function (see Figure 2a), which is denoted as $\mathcal{F}_P(q)$ (see Equation 1).[2]

$$\mathcal{F}_P(q) = \frac{1}{|P|} \sum_{p \in P} \exp\Big(- \frac{1}{b^2} d_G(q,p)^2 \Big) \tag{1}$$

where P, b, and $d_G(q,p)$ are the location dataset, the bandwidth parameter,[3] and the shortest path distance, respectively.

Although many kernel functions (e.g., Epanechnikov kernel and quartic kernel) can be adopted in the network kernel density function, we mainly focus on the Gaussian kernel function, i.e., $\exp\big(- \frac{1}{b^2} d_G(q,p)^2 \big)$ (see Figure 2b), in this paper, which are based on these three main reasons. First,

This work was supported by the Natural Science Foundation of China under grants 62572326, 23IAA00610, and 62372308, Scientific Foundation for Youth Scholars of Shenzhen University, the Science and Technology Development Fund Macau SAR (0003/2023/RIC, 0052/2023/RIA1, 0011/2025/RIC, 001/2024/SKL for SKLIOTSC), Shenzhen-Hong Kong-Macau Science and Technology Program Category C (SGDX20230821095159012), and the Key Technological Innovation Program of Ningbo City under Grant No. 2024Z297. This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau. Dingming Wu is the corresponding author of this paper.

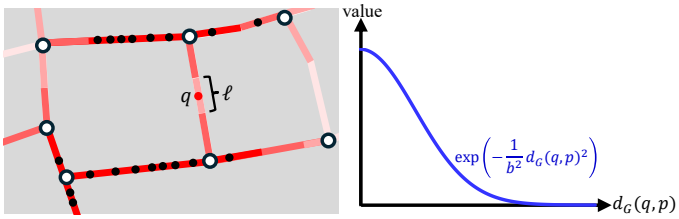
[1]"Lixel" refers to a small line segment on a road network (with the size ℓ in Figure 2a), which is an analogy to "pixel" on a two-dimensional plane.

[2]After the density values $\mathcal{F}_P(q)$ of all lixels q have been computed, we can first divide the range of these density values into several (e.g., 20) equal intervals, where each interval is assigned one color (e.g., we can assign the deep red color for the high density value), and then map each density value to the corresponding color.

[3]This is the positive value that is provided by domain experts to control the shape of the kernel function (see Figure 2b).

TABLE I: Theoretical results for generating NKDV with the Gaussian kernel, where $|V|$, $|E|$, L, $|P|$, T_{SP}, and S_{SP} denote the number of nodes, number of edges, number of lixels, number of data points, the time complexity of the shortest path algorithm, and the space complexity of the shortest path algorithm, respectively. M is a small value that depends on an absolute error ϵ.

Method	Time complexity	Space complexity	Accuracy guarantee	Reference
SOTA	$O(\|E\|T_{\text{SP}} + \|P\|L)$	$O(\|V\| + \|P\| + L + S_{\text{SP}})$	Exact	[26]
PLAN	$O\left(\|E\|T_{\text{SP}} + LM\|E\| \log\left(\frac{\|P\|}{\|E\|}\right)\right)$ (Theorem 1)	$O(\|V\| + \|P\| + L + S_{\text{SP}} + M)$ (Theorem 3)	ϵ-absolute error (Lemma 1)	Section III-A to Section III-C
PLAN+	$O\left(\|E\|T_{\text{SP}} + LM\|E\| \log\left(\frac{\|P\|}{M\|E\|}\right)\right)$ (Theorem 2)			Section III-A to Section III-D



(a) NKDV (b) Gaussian kernel

Fig. 2: Illustration of NKDV for an example dataset (with black data points) using the Gaussian kernel function.

Gaussian kernel is the most popular one, which has been supported by various software packages, including ArcGIS [27], spNetwork [28], Deck.gl [29], and Seaborn [30]. In particular, those visualization software packages (e.g., Deck.gl and Seaborn) set Gaussian kernel to be the default kernel function. Second, the majority of existing studies [5], [6], [8], [9], [12]–[14], [17]–[21], [24], [31], [32] also adopt Gaussian kernel for generating NKDV in the recent decades. Third, Gaussian kernel can provide smooth (and yet more reasonable) visualization. Consider Figure 1 as an example. Observe that NKDV with the Epanechnikov kernel can suffer from abrupt color change (see (i) and (ii) in Figure 1c), which is not reasonable because two positions that are very close to each other in a road network should have similar density values. In contrast, due to the smoothness of the Gaussian kernel, there is no abrupt color change for the corresponding NKDV (see (i) and (ii) in Figure 1b).

However, generating NKDV with the Gaussian kernel function suffers from two main challenges.

Challenge 1 (High time complexity of generating NKDV with the Gaussian kernel): A simple approach is to access all data points in P (see Equation 1) for each lixel q. However, consider the Chicago 311-call dataset [25] (with 10.893 million data points) and the corresponding road network as an example. Generating NKDV with the lixel size $\ell = 10$m (with 0.67 million lixels) takes at least 7.298 trillion operations. Therefore, this approach is not scalable to support large-scale location datasets and a large number of lixels.

Challenge 2 (Hard to extend the state-of-the-art NKDV solutions for supporting the Gaussian kernel): Recently, Chan et al. [1], [2] have proposed efficient algorithms that can successfully reduce the worst-case time complexity for generating NKDV. However, these two research studies mainly focus on polynomial-based kernel functions (including Epanechnikov kernel and quartic kernel), which utilize two main properties of the kernel functions, namely (1) short-tail property and (2) sum-of-distance property, for developing efficient algorithms. Here, we consider the network kernel density function $\mathcal{F}_P^{\mathcal{E}}(q)$ with the Epanechnikov kernel for illustration (see Equation 2).

$$\mathcal{F}_P^{\mathcal{E}}(q) = \frac{1}{|P|} \sum_{p \in P} \begin{cases} 1 - \frac{1}{b^2} d_G(q,p)^2 & \text{if } d_G(q,p) \leq b \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Short-tail property. Observe from Equation 2 that all those data points that are not within the bandwidth b cannot contribute to the network kernel density function $\mathcal{F}_P^{\mathcal{E}}(q)$. Therefore, computing $\mathcal{F}_P^{\mathcal{E}}(q)$ can be transferred to solving the range query problem (with the bandwidth b) for the lixel q, which can avoid scanning all data points for generating NKDV.

Sum-of-distance property. In Equation 2, we note that the network kernel density function $\mathcal{F}_P^{\mathcal{E}}(q)$ can be represented by the following sum-of-distance function (see Equation 3), where S is any subset of the location dataset P.

$$D_S(q) = \sum_{p \in S} d_G(q,p)^2 \quad (3)$$

By investigating $D_S(q)$, ADA [2] and LION [1] augment some terms into data points and lixels, respectively, so that these methods can reduce the worst-case time complexity for generating NKDV.

Unfortunately, the network kernel density function $\mathcal{F}_P(q)$ (see Equation 1) does not exhibit the above two properties. Consider the first property. Note that the exponential function in Equation 1 still has a positive value no matter how small the exponent is (or how large $d_G(q,p)$ is). Consider the second property. Since there is an exponential function in $\mathcal{F}_P(q)$, it is not possible to represent $\mathcal{F}_P(q)$ in terms of $D_S(q)$. Without these two properties, the state-of-the-art methods [1], [2] cannot support NKDV with the Gaussian kernel.

Our contributions: To tackle the above two challenges, we have the following three contributions in this paper.

- We propose the pioneering Piecewise-Linear Approximate solutioN (PLAN), which can re-enable the above two properties. By investigating (i) how the piecewise-linear function can be utilized to accurately approximate the network kernel density function (see Equation 1) and (ii) how the piecewise-linear function can be efficiently handled in each edge, we show that **PLAN can simultaneously (1) reduce the time complexity of generating NKDV, (2) retain the similar space complexity, and (3) provide the approximation guarantee (see Table I).**

- We provide the further optimization for PLAN, namely PLAN+, by (1) observing the connection between evaluating approximate network kernel density function and another computational problem that is to find the rightmost data points in multiple intervals and (2) developing the complexity-reduced algorithm for this computational problem. Note that **PLAN+ can achieve the lowest time complexity for generating NKDV (see Table I).**
- We conduct extensive experiments on four large-scale datasets, which demonstrate that **PLAN and PLAN+ achieve 32.47x to 2936.88x speedups and only incur 1.26x to 2.15x space overhead over the state-of-the-art solution (SOTA) without degrading the (subjective and objective) visualization quality.**

The rest of the paper is organized as follows. We first formally define NKDV with the Gaussian kernel and outline SOTA in Section II. Then, we discuss PLAN and PLAN+ with theoretical analysis in Section III. Next, we provide the experimental evaluation for all methods in Section IV. After that, we review the related work in Section V. Lastly, we conclude this paper in Section VI. Supplementary materials can be found in Section VII.

II. Preliminaries

We first formally define the NKDV problem (using the Gaussian kernel) in Section II-A. Then, we discuss the state-of-the-art solution (SOTA) in Section II-B.

A. Problem Definition

Recall from Figure 2a that we need to color each lixel q on a road network in order to generate NKDV for a location dataset. Here, we formally define NKDV in Problem 1.

Problem 1. *Given a road network $G = (V, E)$ that consists of L lixels and a location dataset P, where each data point lies on one and only one edge, we need to compute $\mathcal{F}_P(q)$ (see Equation 1) for each lixel q.*

B. State-of-the-art Solution (SOTA)

Although many recent advanced algorithms, including ADA [2] and LION [1], cannot be extended for generating NKDV with the Gaussian kernel (as discussed in Section I), the shortest path sharing approach, which is proposed in [26], can still be used for reducing the time complexity of generating exact NKDV, which can act as the SOTA solution.

Instead of calling the shortest path algorithm for each lixel q, SOTA only calls this algorithm in the node s and node t for each edge $e = (s, t)$ and share the shortest path distances (from the node s and node t to other nodes) with all lixels q in this edge e (see Figure 3). Therefore, this approach only needs to call $O(|E|)$ times (instead of $O(L)$ times) shortest path algorithms, which reduces the time complexity for generating NKDV to $O(|E|T_{\text{SP}}+|P|L)$, where T_{SP} denotes the time complexity of the shortest path algorithm (e.g., $T_{\text{SP}} = O(|V|\log|V|+|E|)$ in the Dijkstra's algorithm). Despite this, this solution is still not scalable to large location datasets (i.e., large $|P|$).

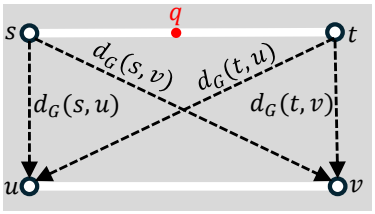


Fig. 3: Illustration of the SOTA solution, where those shortest path distances (black dashed lines) can be shared to all lixels q in the edge $e = (s, t)$.

III. Our Solution: PLAN

To overcome the efficiency issue of SOTA, we propose the Piecewise-Linear Approximate solutioN (PLAN). First, we discuss the core ideas of PLAN in Section III-A. Based on these core ideas, we then discuss the approximation of network kernel density function and the fast computation of approximate NKDV in Section III-B and Section III-C, respectively. After that, we illustrate the improved version of PLAN, called PLAN+, in Section III-D. Lastly, we discuss the space complexity of PLAN and PLAN+ in Section III-E.

A. Core ideas

Recall from the Challenge 2 in Section I that the network kernel density function $\mathcal{F}_P(q)$ (see Equation 1) does not exhibit the short-tail property and the sum-of-distance property, making it difficult to be efficiently computed. To address these two issues, we have the following core ideas.

Core idea 1: Observe from Figure 4b that the exponential function $\exp(-x)$ is very small once the value x is very large. Using $x = 10$ as an example, $\exp(-x) = 4.54 \times 10^{-5}$. Hence, if a lixel q and any data point p (e.g., the orange data points, p_4 and p_5, in Figure 4a) are far away from each other, the value $\exp(-\frac{1}{b^2}d_G(q,p)^2)$ cannot significantly contribute to $\mathcal{F}_P(q)$. Therefore, this core idea is to set a small threshold δ so that all data points p with $\exp(-\frac{1}{b^2}d_G(q,p)^2) \leq \delta$ (see the black dotted line in Figure 4b) are discarded. Note that we have

$$\exp\left(-\frac{1}{b^2}d_G(q,p)^2\right) \leq \delta \implies d_G(q,p) \geq b\sqrt{\ln\left(\frac{1}{\delta}\right)} \quad (4)$$

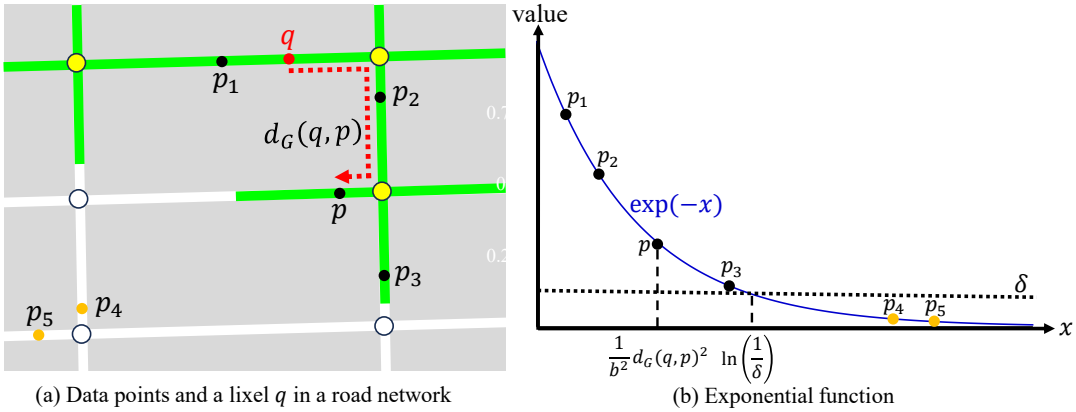
As such, we only need to process those data points p that are within the distance $b\sqrt{\ln(\frac{1}{\delta})}$ from each lixel q (i.e., those data points that lie on the green region in Figure 4a). Hence, using this approach can enable the short-tail property.

Core idea 2: Observe from Figure 5 that it is possible to approximate the more complicated exponential function $\exp(-x)$ (the blue curve) by the simple linear function (i.e., the red line) for any interval $[\ell, u]$, where m and k can be computed based on ℓ and u in $O(1)$ time.

$$m = \frac{\exp(-u)-\exp(-\ell)}{u-\ell} \text{ and } k = \frac{u\exp(-\ell)-\ell\exp(-u)}{u-\ell}$$

With this concept, we consider the set of data points $\mathbb{P} = \{p \in P : \ell \leq \frac{1}{b^2}d_G(q,p)^2 \leq u\}$ (e.g., the black points in Figure 5). The network kernel density function of the lixel q with this set $\mathbb{P}$ can be approximated by the following upper bound value.

$$\mathcal{F}_{\mathbb{P}}(q) \leq \frac{1}{|\mathbb{P}|}\sum_{p\in\mathbb{P}}\left(\frac{m}{b^2}d_G(q,p)^2+k\right) = \frac{m}{|\mathbb{P}|b^2}\sum_{p\in\mathbb{P}}d_G(q,p)^2+k$$



(a) Data points and a lixel q in a road network

(b) Exponential function

Fig. 4: Illustration of Core idea 1.

Hence, by eliminating the exponential function, this upper bound function exhibits the sum-of-distance property (see Equation 3).

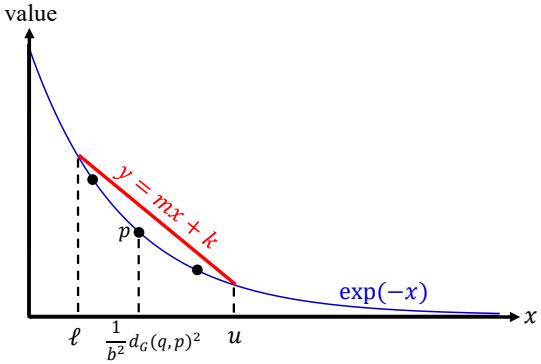


Fig. 5: Illustration of Core idea 2.

Core idea 3: Note that Core ideas 1 and 2 are based on approximating the exponential function (and thus $\mathcal{F}_P(q)$ in Equation 1). In order to retain the similar quality of NKDV, this core idea is to generate NKDV with a non-trivial approximation guarantee (i.e., solving Problem 1 approximately), which is formally stated in Problem 2.

Problem 2. *Given a road network $G = (V, E)$ that consists of L lixels, a location dataset P, where each data point lies on one and only one edge, and an absolute error ϵ, we need to compute the approximate network kernel density function value $A_P(q)$ for each lixel q, where*

$$|A_P(q) - \mathcal{F}_P(q)| \leq \epsilon \tag{5}$$

B. Approximation of Network Kernel Density Function

Based on Core ideas 1 and 2, we propose to adopt the piecewise-linear function $PL(x)$ (with M intervals) to approximate the exponential function $\exp(-x)$ (see Equation 6).

$$PL(x) = \begin{cases} m_1 x + k_1 & \text{if } x \in \mathcal{I}_1 \\ m_2 x + k_2 & \text{if } x \in \mathcal{I}_2 \\ \vdots & \vdots \\ m_{M-1} x + k_{M-1} & \text{if } x \in \mathcal{I}_{M-1} \\ 0 & \text{if } x \in \mathcal{I}_M \end{cases} \tag{6}$$

where $\mathcal{I}_1 = [l_0, l_1)$, $\mathcal{I}_2 = [l_1, l_2)$,..., $\mathcal{I}_{M-1} = [l_{M-2}, l_{M-1})$, and $\mathcal{I}_M = [l_{M-1}, l_M)$. In addition, $l_0 = 0$ and $l_M = \infty$ are dummy variables and $l_{M-1} = \ln(\frac{1}{\delta})$ (i.e., $\exp(-l_{M-1}) = \delta$).

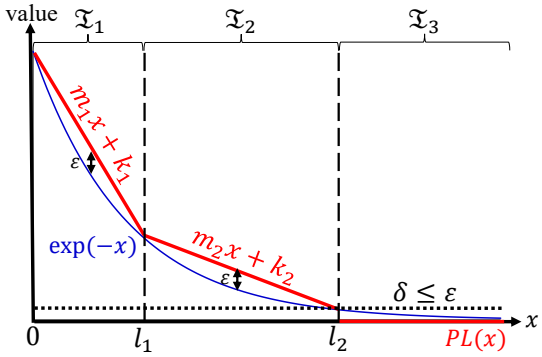


Fig. 6: An example for using the piecewise-linear function to approximate the exponential function.

Figure 6 shows an example of the piecewise-linear function with $M = 3$. Here, we ensure that (1) each line segment (except the last one) is the upper bound of $\exp(-x)$, i.e., $m_i x + k_i \geq \exp(-x)$ for $x \in \mathcal{I}_i$ (where $1 \leq i \leq M-1$) and (2) each line segment must not deviate from $\exp(-x)$ by at most ϵ, i.e., $m_i x + k_i - \exp(-x) \leq \epsilon$ for $x \in \mathcal{I}_i$ (where $1 \leq i \leq M-1$) and $0 \leq \exp(-x) \leq \exp(-l_{M-1}) = \delta \leq \epsilon$ for $x \in \mathcal{I}_M$. With these two conditions, once we replace $\exp(-x)$ by $PL(x)$ in $\mathcal{F}_P(q)$ (see Equation 1), we have the approximate network kernel density function value $A_P(q)$ in Equation 7, which can achieve the ϵ-error guarantee in Problem 2 (see Lemma 1). The proof of this lemma can be found in Section VII-A.

$$A_P(q) = \frac{1}{|P|} \sum_{p \in P} PL\left(\frac{1}{b^2} d_G(q,p)^2\right) \tag{7}$$

Lemma 1. *Given a piecewise-linear function $PL(x)$ (Equation 6) and an absolute error ϵ, if (1) $m_i x + k_i \geq \exp(-x)$ for $x \in \mathcal{I}_i$ (where $1 \leq i \leq M-1$) and (2) $m_i x + k_i - \exp(-x) \leq \epsilon$ for $x \in \mathcal{I}_i$ (where $1 \leq i \leq M-1$) and $0 \leq \exp(-x) \leq \exp(-l_{M-1}) = \delta \leq \epsilon$ for $x \in \mathcal{I}_M$, we have the approximate network kernel density function $A_P(q)$ (Equation 7) so that $|A_P(q) - \mathcal{F}_P(q)| \leq \epsilon$ (Problem 2).*

In order to obtain this $PL(x)$ (given an absolute error ϵ), we aim to reduce the number of intervals M since the large M can

result in slow performance (will be discussed in Section III-C). Therefore, we iteratively find the interval $\mathcal{I}_i = [l_i, l_{i+1})$ (in the i^{th} iteration) with the maximum value of $l_{i+1} - l_i$ so that the deviation is less than ϵ (i.e., $m_i x + k_i - \exp(-x) \leq \epsilon$) until we reach the last interval $\mathcal{I}_M$ with $\delta \leq \epsilon$. Algorithm 1 shows the pseudocode for obtaining these M intervals.

Algorithm 1 Partition

1: **procedure** PARTITION(Error parameter ϵ)
2: Let $S_{\mathcal{I}}$ be the set of intervals, where $S_{\mathcal{I}} \leftarrow \phi$
3: $\ell \leftarrow 0$
4: $\delta \leftarrow \exp(-\ell)$ ▷ Core idea 1
5: **while** $\delta > \epsilon$ **do**
6: Find u in order to maximize $u-\ell$ so that $mx+k \leq \epsilon$ for all $x \in [\ell, u)$, where (based on Core idea 2)

$$m = \frac{\exp(-u) - \exp(-\ell)}{u - \ell}$$

$$k = \frac{u\exp(-\ell) - \ell\exp(-u)}{u - \ell}$$

7: $S_{\mathcal{I}} \leftarrow S_{\mathcal{I}} \cup \{[\ell, u)\}$
8: $\ell \leftarrow u$
9: $\delta \leftarrow \exp(-\ell)$ ▷ Core idea 1
10: $S_{\mathcal{I}} \leftarrow S_{\mathcal{I}} \cup [\ell, \infty)$
11: Return $S_{\mathcal{I}}$

Note that the most challenging part of Algorithm 1 is to find u (i.e., l_{i+1}), given ℓ (i.e., l_i), in line 6. In Lemma 2, we show that finding the best u is equivalent to solving the equation with one variable u, which indicates that we can easily solve it based on some numerical methods [33] (e.g., the bisection method). The proof of this lemma can be found in Section VII-B.

Lemma 2. *Given an absolute error ϵ and ℓ, finding u for maximizing $u-\ell$ so that $\frac{\exp(-u)-\exp(-\ell)}{u-\ell}x + \frac{u\exp(-\ell)-\ell\exp(-u)}{u-\ell} - \exp(-x) \leq \epsilon$ for all $\ell \leq x \leq u$ is equivalent to solving Equation 8 with one variable u.*

$$h(u) = \epsilon \tag{8}$$

where

$$h(u) = \frac{1}{u-\ell}\Big((\exp(-u) - \exp(-\ell))\ln\Big(\frac{u-\ell}{\exp(-\ell) - \exp(-u)}\Big) + (u-1)\exp(-\ell) - (\ell-1)\exp(-u)\Big)$$

Since this partition method does not depend on a location dataset P and a road network G, all intervals and the corresponding parameters (including δ, m_i, and k_i, where $1 \leq i \leq M$, in Equation 6) can be obtained and stored in advance for a given ϵ (especially for those commonly used ϵ in practice). Figure 7 shows how the number of intervals M changes with respect to ϵ. Observe that the smaller the ϵ, the larger the M. Moreover, even though we adopt the small ϵ, M is still not very large (e.g., $M = 8$ for $\epsilon = 0.01$).

C. Fast Computation of Approximate NKDV

Once we have obtained $PL(x)$, our goal is to efficiently compute $A_P(q)$ (see Equation 7). Here, we define $P(e)$ to be the set of data points in an edge e. With this definition, we can represent $A_P(q)$ (see Equation 9) by the (non-normalized) edge-e approximate network kernel density function $\widehat{A}_{P(e)}(q)$ (see Equation 10).

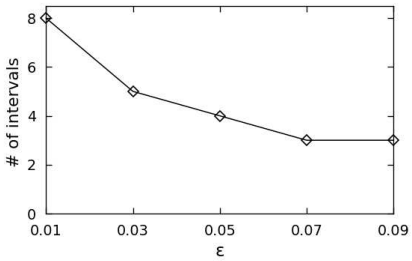


Fig. 7: The number of intervals M of the piecewise-linear function $PL(x)$, varying the absolute error ϵ.

$$A_P(q) = \frac{1}{|P|}\sum_{e \in E} \widehat{A}_{P(e)}(q) \tag{9}$$

where

$$\widehat{A}_{P(e)}(q) = \sum_{p \in P(e)} PL\Big(\frac{1}{b^2} d_G(q,p)^2\Big) \tag{10}$$

Here, we illustrate how to efficiently evaluate $\widehat{A}_{P(e)}(q)$ for each edge $e = (v, w)$ in order to efficiently compute $A_P(q)$. Like [2], we augment the aggregate distance values, $a^{(deg)}_{P(v,p)}$ (see Equation 11) and $a^{(deg)}_{P(w,p)}$ (see Equation 12), for each data point p in advance (see Figure 8), where $deg = 0, 1, 2$.

$$a^{(deg)}_{P(v,p)} = \sum_{p_j \in P(v,p)} d_G(v, p_j)^{deg} \tag{11}$$

$$a^{(deg)}_{P(w,p)} = \sum_{p_j \in P(w,p)} d_G(w, p_j)^{deg} \tag{12}$$

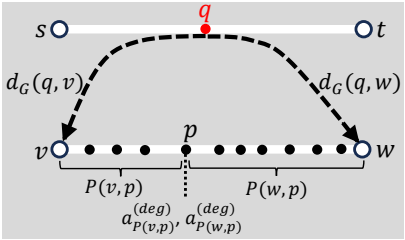


Fig. 8: Augmentation of aggregate distance values, which are $a^{(deg)}_{P(v,p)}$ and $a^{(deg)}_{P(w,p)}$ (where $deg = 0, 1, 2$), for each p.

Recall from Core idea 1 that we only need to process those data points that are within the threshold $b\sqrt{\ln(\frac{1}{\delta})}$. In this section, we focus on the most complicated case with $d_G(q,v) < b\sqrt{\ln(\frac{1}{\delta})}$ and $d_G(q,w) < b\sqrt{\ln(\frac{1}{\delta})}$ (see Figure 8). Note that this case can be further divided into two subcases.

First subcase. Figure 9 illustrates the first subcase (i.e., $d_G(v,w) > 2b\sqrt{\ln(\frac{1}{\delta})} - d_G(q,v) - d_G(q,w)$), which indicates that the search range from node v and the search range from node w do not intersect with each other.

Observe that each data point in $P(e)$ can be in one of the intervals (e.g., the purple data points and pink data points are in $\mathcal{I}_3^{(v)}$ and $\mathcal{I}_3^{(w)}$, respectively). Therefore, we can represent $\widehat{A}_{P(e)}(q)$ (see Equation 10) with respect to $a^{(deg)}_{\mathcal{I}_i^{(v)}}$ and $a^{(deg)}_{\mathcal{I}_i^{(w)}}$

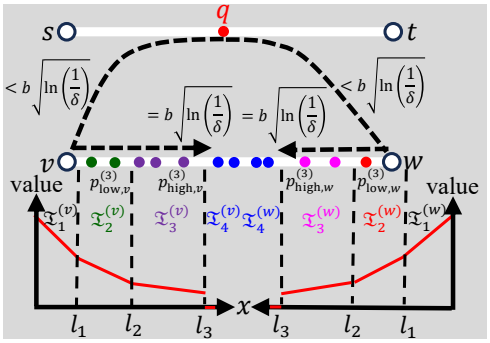


Fig. 9: The search range from node v does not intersect with the search range from node w.

$(deg = 0, 1, 2)$. Note that $a^{(deg)}_{\mathcal{I}^{(v)}_i}$ (or $a^{(deg)}_{\mathcal{I}^{(w)}_i}$) is similar to Equation 11 (or Equation 12), except that we replace $P(v,p)$ (or $P(w,p)$) (see Figure 8) by $\mathcal{I}^{(v)}_i$ (or $\mathcal{I}^{(w)}_i$) (see Figure 9).

$$\begin{aligned}
&\widehat{A}_{P(e)}(q)\\
&= \sum_{i=1}^{M-1}\Bigg(\sum_{\frac{1}{b^2}d_G(q,p)^2\in\mathcal{I}^{(v)}_i}\Big(m_i\Big(\frac{1}{b^2}(d_G(q,v)+d_G(v,p))^2\Big)+k_i\Big)\\
&+\sum_{\frac{1}{b^2}d_G(q,p)^2\in\mathcal{I}^{(w)}_i}\Big(m_i\Big(\frac{1}{b^2}(d_G(q,w)+d_G(w,p))^2\Big)+k_i\Big)\Bigg)\\
&= \sum_{i=1}^{M-1}\Big(\Big(\frac{m_i}{b^2}d_G(q,v)^2+k_i\Big)a^{(0)}_{\mathcal{I}^{(v)}_i}+\frac{2m_i d_G(q,v)}{b^2}a^{(1)}_{\mathcal{I}^{(v)}_i}+\frac{m_i}{b^2}a^{(2)}_{\mathcal{I}^{(v)}_i}\Big)\\
&+\sum_{i=1}^{M-1}\Big(\Big(\frac{m_i}{b^2}d_G(q,w)^2+k_i\Big)a^{(0)}_{\mathcal{I}^{(w)}_i}+\frac{2m_i d_G(q,w)}{b^2}a^{(1)}_{\mathcal{I}^{(w)}_i}+\frac{m_i}{b^2}a^{(2)}_{\mathcal{I}^{(w)}_i}\Big)
\end{aligned}$$

Here, we discuss how to find $a^{(deg)}_{\mathcal{I}^{(v)}_i}$. Note that all data points p that are inside $\mathcal{I}^{(v)}_i$ must fulfill the following inequality.

$$\begin{aligned}
&l_{i-1}\le\frac{1}{b^2}(d_G(q,v)+d_G(v,p))^2<l_i\\
\Longrightarrow\quad &b\sqrt{l_{i-1}}-d_G(q,v)\le d_G(v,p)<b\sqrt{l_i}-d_G(q,v)
\end{aligned}$$

Therefore, we can adopt the binary search method to find $p^{(i)}_{\text{low},v}$ and $p^{(i)}_{\text{high},v}$ so that $d_G(v,p^{(i)}_{\text{low},v})$ and $d_G(v,p^{(i)}_{\text{high},v})$ are just smaller than $b\sqrt{l_{i-1}}-d_G(q,v)$ and $b\sqrt{l_i}-d_G(q,v)$, respectively (e.g., $p^{(3)}_{\text{low},v}$ and $p^{(3)}_{\text{high},v}$ in Figure 9). With $p^{(i)}_{\text{low},v}$ and $p^{(i)}_{\text{high},v}$, we can obtain $a^{(deg)}_{\mathcal{I}^{(v)}_i}$ in $O(1)$ time, where

$$a^{(deg)}_{\mathcal{I}^{(v)}_i}=a^{(deg)}_{P(v,p^{(i)}_{\text{high},v})}-a^{(deg)}_{P(v,p^{(i)}_{\text{low},v})}\tag{13}$$

Based on the same idea, we can also first adopt the binary search method (with $O(\log|P(e)|)$ time) to find $p^{(i)}_{\text{low},w}$ and $p^{(i)}_{\text{high},w}$ and then obtain $a^{(deg)}_{\mathcal{I}^{(w)}_i}$ in $O(1)$ time, where

$$a^{(deg)}_{\mathcal{I}^{(w)}_i}=a^{(deg)}_{P(w,p^{(i)}_{\text{high},w})}-a^{(deg)}_{P(w,p^{(i)}_{\text{low},w})}\tag{14}$$

Since there are at most $2M$ intervals on each edge in the worst case (see Figure 9), the time complexity of this subcase is $O(M\log|P(e)|)$.

Second subcase. Figure 10 illustrates the second subcase (i.e., $d_G(v,w)\le 2b\sqrt{\ln(\frac{1}{\delta})}-d_G(q,v)-d_G(q,w)$), which indicates

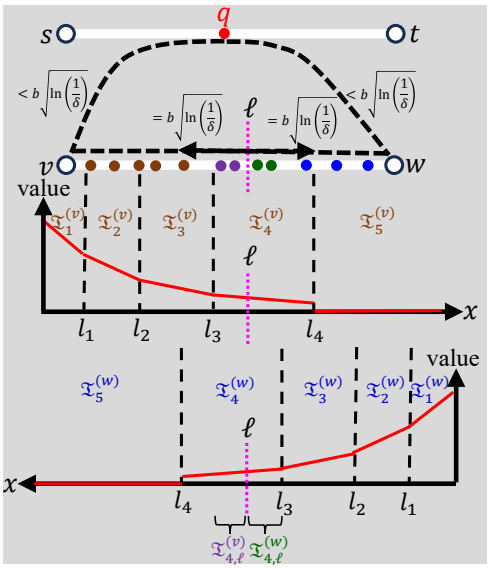


Fig. 10: The search range from node v intersects with the search range from node w.

that the search range from node v and the search range from node w intersect with each other. In order to evaluate $\widehat{A}_{P(e)}(q)$ (see Equation 10), we need to first use the binary search method to find ℓ so that $d_G(v,\ell)=\frac{d_G(v,w)+d_G(q,w)-d_G(q,v)}{2}$ (based on $d_G(q,v)+d_G(v,\ell)=d_G(q,w)+d_G(w,\ell)$), which takes $O(\log|P(e)|)$ time.

Suppose that ℓ is in $\mathcal{I}^{(v)}_\alpha$ and $\mathcal{I}^{(w)}_\alpha$ ($\alpha=4$ in Figure 10)[4]. We can then have $\mathcal{I}^{(v)}_{\alpha,\ell}$ ($\mathcal{I}^{(w)}_{\alpha,\ell}$) that covers those data points in $\mathcal{I}^{(v)}_\alpha$ ($\mathcal{I}^{(w)}_\alpha$) but not in $P(w,\ell)$ ($P(v,\ell)$), i.e., $\mathcal{I}^{(v)}_{\alpha,\ell}=\mathcal{I}^{(v)}_\alpha\backslash P(w,\ell)$ ($\mathcal{I}^{(w)}_{\alpha,\ell}=\mathcal{I}^{(w)}_\alpha\backslash P(v,\ell)$). Using Figure 10 as an example, the purple data points (the green data points) are in $\mathcal{I}^{(v)}_{4,\ell}$ ($\mathcal{I}^{(w)}_{4,\ell}$). By adopting the similar concepts of the first subcase for those left intervals from the node v, i.e., $\mathcal{I}^{(v)}_i$ $(1\le i\le\alpha-1)$ and $\mathcal{I}^{(v)}_{\alpha,\ell}$, and those right intervals from the node w, i.e., $\mathcal{I}^{(w)}_i$ $(1\le i\le\alpha-1)$ and $\mathcal{I}^{(w)}_{\alpha,\ell}$, we can compute $\widehat{A}_{P(e)}(q)$ in $O(M\log|P(e)|)$ time (α can be at most $O(M)$).

Remark. It is possible to have some cases with $d_G(q,v)\ge b\sqrt{\ln(\frac{1}{\delta})}$ or $d_G(q,w)\ge b\sqrt{\ln(\frac{1}{\delta})}$. However, those cases can be regarded as the simplified versions of the first subcase. Observe from Figure 9 that we can discard all those left (or right) intervals from the node v (or the node w) if $d_G(q,v)\ge b\sqrt{\ln(\frac{1}{\delta})}$ (or $d_G(q,w)\ge b\sqrt{\ln(\frac{1}{\delta})}$). Therefore, the time complexity is still at most $O(M\log|P(e)|)$.

Based on the above discussion, we can conclude in Lemma 3 that computing $\widehat{A}_{P(e)}(q)$ takes $O(M\log|P(e)|)$ time, given the shortest path distances $d_G(q,v)$ and $d_G(q,w)$ from q to the edge $e=(v,w)$.

Lemma 3. *Given an edge $e=(v,w)$ and a lixel q in a road network $G=(V,E)$ and the shortest path distances $d_G(q,v)$ and $d_G(q,w)$, the time complexity for computing $\widehat{A}_{P(e)}(q)$ (see Equation 10) is $O(M\log|P(e)|)$.*

[4]Note that ℓ must be in the same intervals in both sides (i.e., $\mathcal{I}^{(v)}_\alpha$ and $\mathcal{I}^{(w)}_\alpha$ with the same α). The main reason is that $d_G(q,\ell)=d_G(q,v)+d_G(v,\ell)=d_G(q,w)+d_G(w,\ell)$.

Algorithm 2 Piecewise-Linear Approximation Solution

procedure PLAN($G = (V, E)$, P, b)
  for each edge $e = (v, w) \in E$ **do**
    for each $p \in P(e)$ **do**
      Augment $a^{(deg)}_{P(v,p)}$ in p ▷ Equation 11
      Augment $a^{(deg)}_{P(w,p)}$ in p ▷ Equation 12
  for each edge $\widehat{e} = (s, t) \in E$ **do**
    Obtain $SPD(s)$ and $SPD(t)$, where (each $v \in V$)

$$SPD(s).v = \begin{cases} d_G(s,v) & \text{if } d_G(s,v) < b\sqrt{\ln\left(\frac{1}{\delta}\right)} \\ \infty & \text{otherwise} \end{cases}$$

    for each lixel q in $\widehat{e}$ **do**
      Set $R(q)$ to be 0
      for each edge $e = (v, w) \in E$ **do**
        Obtain $d_G(q, v)$ and $d_G(q, w)$, where

$$d_G(q,v) = \min \begin{cases} d_G(q,s) + SPD(s).v \\ d_G(q,t) + SPD(t).v \end{cases}$$

        $R(q) \leftarrow R(q) + \widehat{A}_{P(e)}(q)$ ▷ Lemma 3
      $A_P(q) \leftarrow \frac{R(q)}{|P|}$ ▷ Equation 9
  Return $A_P(q)$ for all lixels q

Algorithm 2 shows the pseudocode of our PLAN solution. With the low time complexity for computing $\widehat{A}_{P(e)}(q)$ (see Lemma 3), we conclude in Theorem 1 that the time complexity of PLAN is $O\big(|E|T_{\text{SP}} + LM|E|\log\left(\frac{|P|}{|E|}\right)\big)$. The proof of this theorem can be found in Section VII-C.

Theorem 1. *The time complexity of PLAN (i.e., Algorithm 2) is* $O\big(|E|T_{SP} + LM|E|\log\left(\frac{|P|}{|E|}\right)\big)$.

Compared with SOTA (see Section II-B), which takes $O(|E|T_{\text{SP}} + |P|L)$ time, PLAN is theoretically faster than this solution if $M < \frac{|P|}{|E|\log\left(\frac{|P|}{|E|}\right)}$. This inequality normally holds in practice. Using the San Francisco crime dataset [34], which consists of $|E| = 11,593$ edges and $|P| = 2,985,502$ data points, as an example, we have $\frac{|P|}{|E|\log\left(\frac{|P|}{|E|}\right)} = 32.16$. However, even though we set the absolute error ϵ to be a small value (e.g., 0.01), M is still much smaller compared with $\frac{|P|}{|E|\log\left(\frac{|P|}{|E|}\right)}$ (e.g., $M = 8$ in this case, as shown in Figure 7.).

D. Further Improvement: PLAN+

In this section, we provide an additional optimization approach for computing $\widehat{A}_{P(e)}(q)$ (see Equation 10), which can further reduce the time complexity for generating approximate NKDV compared with PLAN.

Here, we consider another computational problem (see Problem 3), which is closely related to the evaluation of $\widehat{A}_{P(e)}(q)$ (i.e., the first and second subcases in Section III-C).

Problem 3. *Given Γ intervals and m one-dimensional data points that are sorted in an axis, we need to find the rightmost data point for each interval.*

Using Figure 11 as an example, observe that the number of intervals $\Gamma = 6$, the number of data points $m = 18$, and all red data points are the solution to this instance of Problem 3.

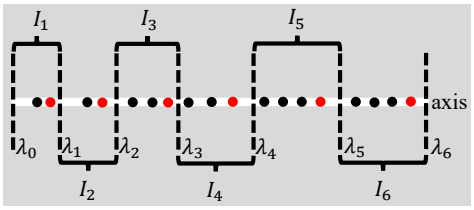


Fig. 11: Finding the rightmost (red) data point in each interval.

In order to solve this problem, a direct approach is to adopt the binary search method Γ times in this axis, which takes $O(\Gamma \log m)$ time. However, observe from Figure 11 that each interval covers a relatively small amount of data points compared with all data points in this axis (e.g., $l_3 = [\lambda_2, \lambda_3)$ only covers three data points.). Therefore, we have this question. *Can we avoid calling the binary search method with all (i.e., m) data points in this axis Γ times?*

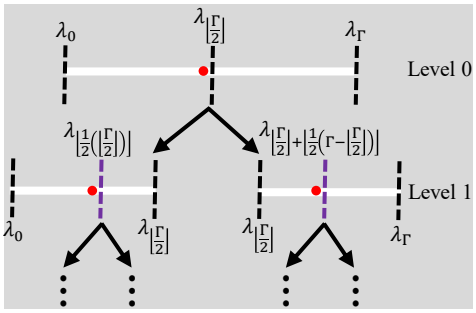


Fig. 12: A divide-and-conquer approach for obtaining the rightmost data point of each interval.

To provide an affirmative answer to this question, we adopt a divide-and-conquer approach for finding the rightmost (red) data points in an axis. Observe from Figure 12 that we can first use the binary search method to find the rightmost (red) data point that is just before the λ value with the middle index, i.e., $\lambda_{\lfloor\frac{\Gamma}{2}\rfloor}$ (the rightmost data point in the interval $I_{\lfloor\frac{\Gamma}{2}\rfloor}$), in the level 0. Then, we iteratively divide this axis into the left part and right part and follow the same step as the level 0 (see the level 1) until each part only covers one interval (e.g., $I_3 = [\lambda_2, \lambda_3)$ in Figure 11). By adopting this approach, we show in Lemma 4 that Problem 3 can be solved in $O\big(\Gamma \log\left(\frac{m}{\Gamma}\right)\big)$ time. The proof of this lemma can be found in Section VII-D.

Lemma 4. *The time complexity of solving Problem 3 is* $O\big(\Gamma \log\left(\frac{m}{\Gamma}\right)\big)$.

Recall from the first and second subcases in Section III-C that the main bottleneck for evaluating $\widehat{A}_{P(e)}(q)$ is to find the rightmost data point (or the leftmost data point) of each interval, e.g., $p^{(3)}_{\text{low},v}$ and $p^{(3)}_{\text{high},v}$ (or $p^{(3)}_{\text{low},w}$ and $p^{(3)}_{\text{high},w}$) in Figure 9, which is the same problem as (or the slightly modified version of) Problem 3. Hence, the worst-case time complexity of computing $\widehat{A}_{P(e)}(q)$ is further reduced to $O\big(M \log\left(\frac{|P(e)|}{M}\right)\big)$ (by setting $\Gamma = M$ and $m = |P(e)|$ in Lemma 4), which is stated in Lemma 5 (the modified version of Lemma 3).

Lemma 5. *Given an edge $e = (v, w)$ and a lixel q in a road network $G = (V, E)$ and the shortest path distances $d_G(q, v)$ and $d_G(q, w)$, the time complexity for computing $\widehat{A}_{P(e)}(q)$ (see Equation 10) is $O\big(M \log\big(\frac{|P(e)|}{M}\big)\big)$.*

With Lemma 5, we state in Theorem 2 that the improved version of PLAN, namely PLAN+, can further lower the worst-case time complexity for generating NKDV, which is $O\big(|E|T_{\text{SP}} + LM|E| \log\big(\frac{|P|}{M|E|}\big)\big)$. The proof of this theorem can be found in Section VII-E.

Theorem 2. *The time complexity of PLAN+ is $O\big(|E|T_{SP} + LM|E| \log\big(\frac{|P|}{M|E|}\big)\big)$.*

E. Space Complexity of PLAN and PLAN+

Note that every NKDV method needs to (1) access all nodes, all edges, all data points, and all lixels in a road network and (2) call the shortest path algorithm. Therefore, the space complexity of every method must be at least $O(|V|+|P|+L+S_{\text{SP}})$,[5] where S_{SP} denotes the space complexity of the shortest path algorithm. Here, we further illustrate the additional space consumption of PLAN and PLAN+.

PLAN. Recall that PLAN needs to (1) store M intervals (see Figure 6) and (2) augment aggregate distance values, $a^{(deg)}_{P(v,p)}$ and $a^{(deg)}_{P(w,p)}$, for each data point p in every edge $e = (v, w)$ (see Figure 8). Therefore, the additional space complexity of PLAN is $O(M + |P|)$.

PLAN+. Since PLAN+ only changes the order for calling the binary search method (based on the divide-and-conquer approach), PLAN+ does not incur additional space overhead compared with PLAN. As such, the additional space complexity of PLAN+ is also $O(M + |P|)$.

Hence, we conclude in Theorem 3 that the space complexity of PLAN and PLAN+ is $O(|V| + |P| + L + S_{\text{SP}} + M)$.

Theorem 3. *The space complexity of PLAN and PLAN+ is $O(|V| + |P| + L + S_{SP} + M)$.*

IV. Experimental Evaluation

We first discuss the experimental settings in Section IV-A. Then, we evaluate the time and space efficiency of all methods in Section IV-B and Section IV-C, respectively. Next, we test the accuracy of all methods in Section IV-D. Lastly, we compare the accuracy and efficiency of generating NKDVs based on Gaussian kernel and Epanechnikov kernel in Section IV-E.

A. Experimental Settings

We utilize four large-scale location datasets for testing. To process each dataset, we adopt the OSMNx library [35] to (1) extract a road network and (2) map all data points into this road network. Table II summarizes the details of all datasets.

TABLE II: Datasets.

Dataset	$\|V\|$	$\|E\|$	$\|P\|$	Category
Minneapolis [36]	7,388	12,913	1,598,877	911 calls
New York [37]	71,019	120,623	1,897,418	Traffic accidents
San Francisco [34]	6,627	11,593	2,985,502	Crime events
Chicago [25]	41,111	70,067	10,893,129	311 calls

[5]We omit $|E|$ since L must be larger than $|E|$.

To conduct our experiments, we test these four methods, which are SOTA, PLAN-, PLAN, and PLAN+. SOTA is the state-of-the-art solution, which is discussed in Section II-B. PLAN- combines the core idea 1 of Sections III-A with SOTA, which performs the range search with the distance $b\sqrt{\ln\big(\frac{1}{\delta}\big)}$ for each lixel q (see the green region in Figure 4a). PLAN and PLAN+ are based on the techniques from Sections III-A to III-C and from Sections III-A to III-D, respectively. We implemented all these methods with C++ and conducted experiments on an Intel i7 2.1 GHz PC with 16 GB memory. In this paper, we measure the response time (sec), memory space consumption (MB), and maximum density deviation (MDD)/average density deviation (ADD) as the time efficiency, space efficiency, and objective accuracy, respectively, of each method, where (based on the notations in Problem 2):

$$MDD = \max_{e \in E}(\max_{q \in e} |A_P(q) - \mathcal{F}_P(q)|) \tag{15}$$

$$ADD = \frac{1}{L} \sum_{e \in E} \sum_{q \in e} |A_P(q) - \mathcal{F}_P(q)| \tag{16}$$

We omit the results which are more than 172,800 sec (i.e., 2 days). By default, we set the lixel size ℓ, the bandwidth parameter b, and the absolute error ϵ to be 10m, 1000m, and 0.05 (i.e., $M = 4$ in Figure 7), respectively.

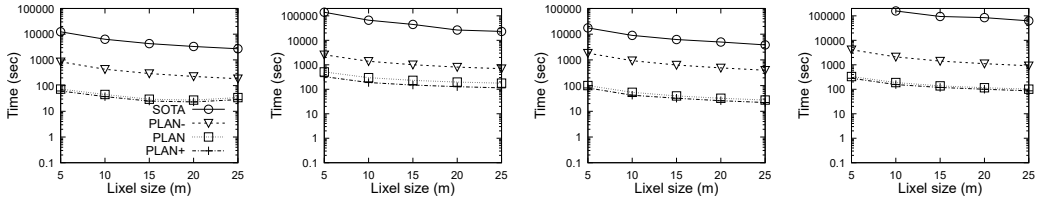
B. Time Efficiency of All Methods

In this section, we report the time efficiency results of all methods by conducting the following experiments.

Varying the lixel size. We investigate how the lixel size affects the response time of each method, by varying it from 5m to 25m. Observe from Figure 13 that the smaller the lixel size (i.e., the larger the number of lixels L), the larger the response time of each method. Since PLAN- avoids scanning all data points on a road network, PLAN- can achieve 9.61x to 75.27x speedups compared with SOTA. With the lower time complexity of PLAN and PLAN+, these two methods can further achieve up to 22.53x speedups compared with PLAN-.

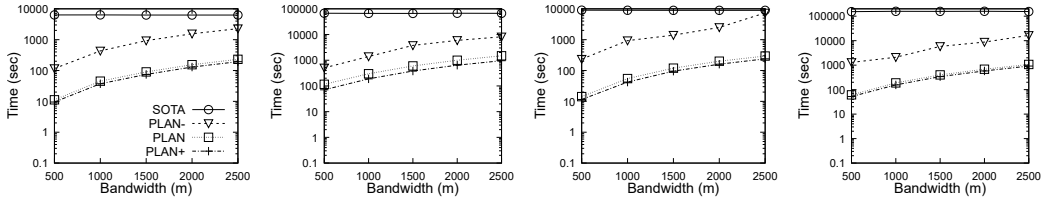
Varying the bandwidth parameter. We proceed to examine how the bandwidth parameter b affects the response time of each method. To conduct this experiment, we vary b from 500m to 2500m. Figure 14 shows the results of all methods. Since SOTA does not exhibit the short-tail property (see Section I), this method needs to (1) scan all data points and (2) access all edges in a road network no matter which bandwidth parameter b we choose. Therefore, the response time of SOTA is not sensitive to b. With the short-tail property of PLAN-, PLAN and PLAN+ (see the core idea 1 in Section III-A), these methods only need to process those data points that are within the range $b\sqrt{\ln(\frac{1}{\delta})}$, which are sensitive with respect to b. As such, once we increase b, the time gaps between PLAN-/PLAN/PLAN+ with SOTA are smaller for all datasets. Based on the lower time complexity of PLAN and PLAN+, these two methods achieve speedups of 32.47x to 2936.88x and 4.36x to 30.83x compared with SOTA and PLAN-, respectively.

Varying the dataset size. We test how the dataset size affects the response time of each method. To conduct this experiment,



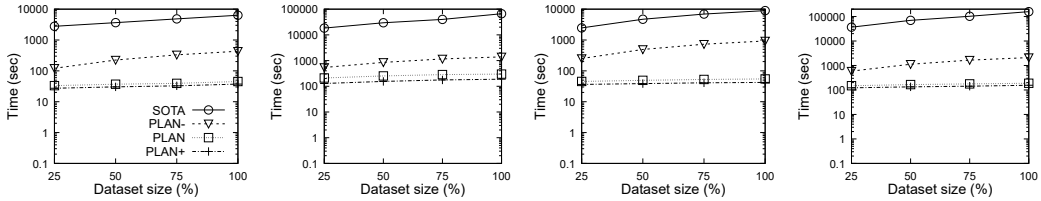
(a) Minneapolis (b) New York (c) San Francisco (d) Chicago

Fig. 13: Response time for generating NKDV, varying the lixel size.



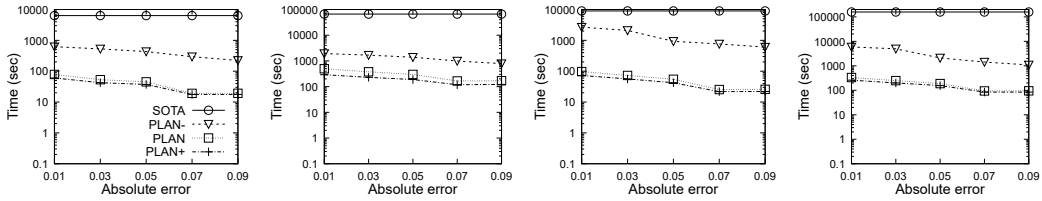
(a) Minneapolis (b) New York (c) San Francisco (d) Chicago

Fig. 14: Response time for generating NKDV, varying the bandwidth parameter.



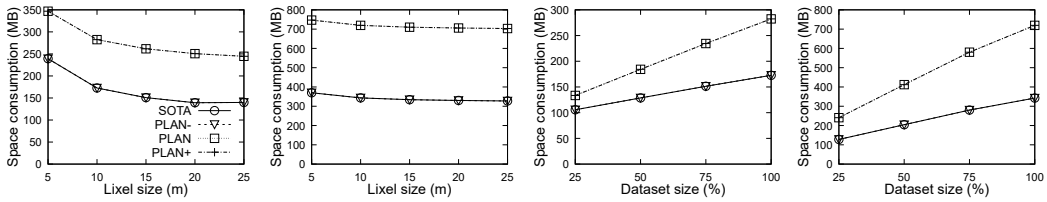
(a) Minneapolis (b) New York (c) San Francisco (d) Chicago

Fig. 15: Response time for generating NKDV, varying the dataset size.



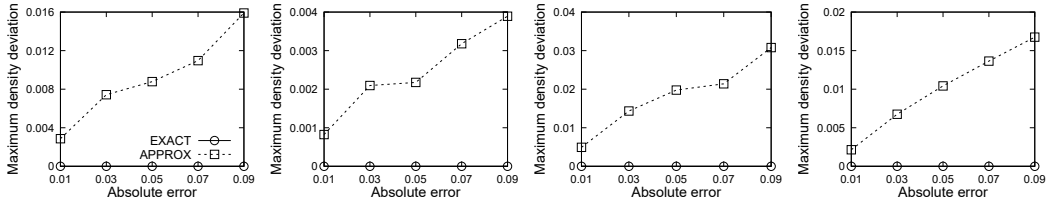
(a) Minneapolis (b) New York (c) San Francisco (d) Chicago

Fig. 16: Response time for generating NKDV, varying the absolute error.



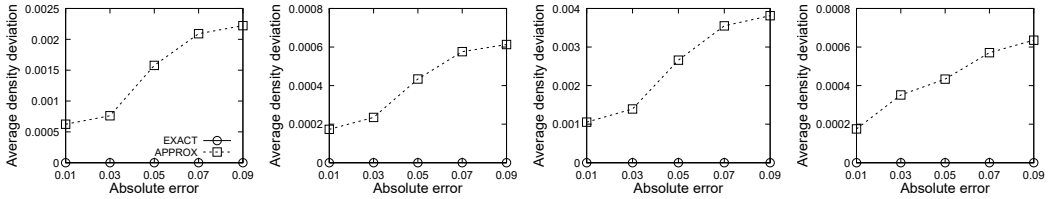
(a) New York (b) Chicago (c) New York (d) Chicago

Fig. 17: Space consumption for generating NKDV in the New York (a and c) and Chicago (b and d) datasets, varying the lixel size (a and b) and the dataset size (c and d).



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

Fig. 18: Objective accuracy (MDD) for generating NKDV, varying the absolute error.



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

Fig. 19: Objective accuracy (ADD) for generating NKDV, varying the absolute error.

we first sample each dataset with different ratios, which are 25%, 50%, 75%, and 100% (i.e., no sampling), and then measure the response time of each method. Observe from Figure 15 that PLAN and PLAN+ are more scalable with respect to the dataset size compared with SOTA and PLAN- since the time complexity of these two methods only depends on the logarithm of $|P|$ (see Table I). As a remark, PLAN and PLAN+ achieve 54.39x to 1015.04x speedups and 2.56x to 22.05x speedups over SOTA and PLAN-, respectively.

Varying the absolute error. We proceed to investigate how the absolute error ϵ affects the response time of each method.[6] Figure 16 shows the results of all methods. Since SOTA is the exact method, the response time does not change across ϵ. Recall from Figure 4 that PLAN-, PLAN, and PLAN+ need to process more data points with a smaller δ, i.e., a smaller ϵ (based on Lemma 1), which results in a longer response time for these methods. Due to the lower time complexity of PLAN and PLAN+, these two methods achieve speedups of 81.01x to 1876.32x and 3.91x to 37.3x compared with SOTA and PLAN-, respectively.

C. Space Efficiency of All Methods

In this section, we investigate the space consumption of each method by conducting the following experiments in the two datasets, which are New York and Chicago.

Varying the lixel size. To conduct this experiment, we first choose the lixel size from 5m to 25m and then measure the space consumption of each method. Figures 17a and b show the results of all methods. Observe that the smaller the lixel size (i.e., the larger number of lixels L), the larger the memory space consumption of each method. Since PLAN and PLAN+ need to augment additional aggregate distance values for each data point of a road network (see Figure 8), the space consumption of these two methods are 1.45x to 2.15x larger than SOTA/PLAN-.

Varying the dataset size. We proceed to test how the dataset size affects the space consumption of each method. By conducting this experiment, we first sample each dataset with four ratios, 25%, 50%, 75%, and 100% (i.e., no sampling), and then measure the space consumption of each method in each reduced dataset. Observe from Figures 17c and d that all methods have higher space consumption if we adopt the larger dataset sizes. PLAN/PLAN+ only incur 1.26x to 2.1x space overhead compared with SOTA/PLAN-.

[6]As a remark, existing research studies [38], [39] in kernel density estimation also set this parameter ϵ to be within the range of $[0.01, 0.1]$. Therefore, we also adopt the similar range $[0.01, 0.09]$ for varying ϵ in this paper.

D. Accuracy of All Methods

Here, we provide the following experiments to test the objective accuracy (i.e., MDD in Equation 15 and ADD in Equation 16) and the subjective accuracy of each method. Note that we use EXACT and APPROX to represent SOTA and PLAN/PLAN+, respectively.

Objective accuracy. In this experiment, we vary the absolute error ϵ from 0.01 to 0.09 and measure the MDD and ADD values for EXACT and APPROX, which are shown in Figure 18 and Figure 19, respectively. Since EXACT provides the same result for each density value (i.e., $A_P(q) = \mathcal{F}_P(q)$), MDD and ADD are always zero for EXACT. Although we choose ϵ from 0.01 to 0.09, MDD and ADD for APPROX are much smaller compared with the corresponding ϵ. For example, MDD and ADD are at most 0.0198 and 0.0027, respectively, in these four datasets if we use $\epsilon = 0.05$.

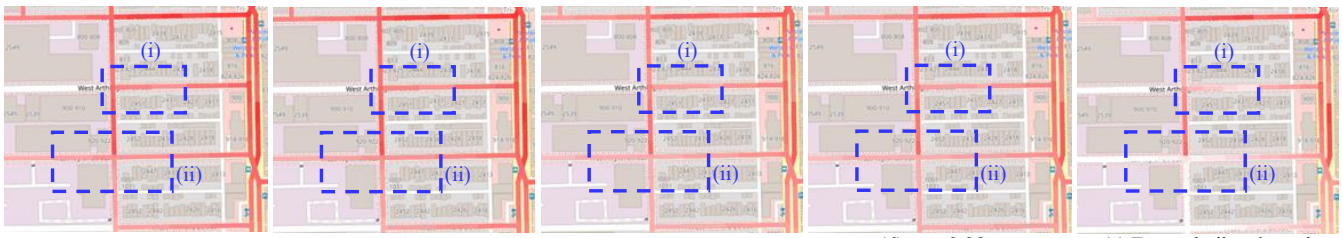
Subjective accuracy. We proceed to investigate the subjective accuracy of different NKDV methods. To conduct this experiment, we generate NKDVs based on EXACT and APPROX (by setting ϵ from 0.01 to 0.09) for the Chicago 311-call dataset. Observe from Figures 20a to d that the larger the absolute error ϵ, the larger the difference between the NKDVs of EXACT and APPROX. As an example, the color in the horizontal road of (i) becomes lighter (i.e., lower density value) when we set a larger ϵ value. In practice, the difference is not very large since the objective accuracy values (in the previous experiment) are very small (see Figures 18 and 19).

E. Comparisons with Epanechnikov Kernel

In this section, we proceed to compare the accuracy and efficiency of generating NKDVs with the Gaussian kernel (based on PLAN/PLAN+) and the Epanechnikov kernel (based on ADA/LION) in the Chicago 311-call location dataset under the default settings (see Section IV-A).

Accuracy. Observe from Figure 20e that NKDV with the Epanechnikov kernel suffers from abrupt (or non-smooth) color changes between two close positions (e.g., (i) or (ii)), which can possibly provide misleading interpretation. As an example, it is hard to conclude whether (ii) is a safe region. In contrast, approximate NKDV with the Gaussian kernel can provide more reasonable (or smooth) visualization results no matter which ϵ we adopt.

Efficiency. Table III summarizes the response time of generating NKDVs with the Gaussian kernel and the Epanechnikov kernel. For each kernel function, we report the smallest response time of the corresponding methods. Since the Gaussian kernel is more complicated than the Epanechnikov kernel, the



(a) EXACT (b) $\epsilon = 0.01$ (c) $\epsilon = 0.05$ (d) $\epsilon = 0.09$ (e) Epanechnikov kernel

Fig. 20: Subjective accuracy for generating NKDV with EXACT (a), APPROX (b to d) (varying the absolute error), and the Epanechnikov kernel (e) in the Chicago 311-call location dataset.

TABLE III: Response time (sec) of generating NKDV with the Gaussian kernel and the Epanechnikov kernel.

Dataset	Epanechnikov kernel	Gaussian kernel
Minneapolis	12.76	37.26
New York	129.03	188.65
San Francisco	17.04	42.22
Chicago	79.68	155.79

response time for supporting NKDV with the Gaussian kernel is normally (1.46x to 2.92x) slower then the one with the Epanechnikov kernel.

V. Related Work

In this section, we review three camps of research studies that are closely related to this work.

Efficient algorithms for kernel density visualization. In the literature, many research studies [38]–[54] have been developed for improving the efficiency of computing kernel density visualization (KDV). However, all these studies mainly focus on generating planar KDV, which does not consider the road network. As such, all these methods cannot be extended to support the network kernel density function (see Equation 1). On the other hand, Chan et al. [49], [50] have adopted the linear and quadratic functions for approximately calculating the kernel density function, which are similar to this work. Since these studies do not consider any properties of road networks, these approaches are unable to reduce the worst-case time complexity (like Table I). Recently, Chan et al. [1], [2], [40], [55] further develop complexity-reduced solutions for generating NKDV with other kernels (e.g., Epanechnikov and quartic kernels). However, these solutions do not fulfill the short-tail property and the sum-of-distance property (as stated in the Challenge 2 of Section I), which cannot support NKDV with the Gaussian kernel. As such, the state-of-the-art solution for supporting NKDV is still based on the shortest path sharing approach [26], which is theoretically and practically inefficient (see Table I and Section IV-B, respectively).

Efficient algorithms for shortest path computation. Recall that computing $\mathcal{F}_P(q)$ (see Equation 1) needs to find the shortest path distances. Therefore, fast shortest path algorithms can improve the efficiency of generating NKDV. In the literature, various types of shortest path algorithms, including heuristic searching [56]–[58], shortest path indexing (e.g., hub labeling structures [59]–[64] and hierarchical structures [59], [62], [65]–[70]), and shortest path caching [71]–[73], can also be used for improving the efficiency of finding the shortest path distances. Nevertheless, most of these research studies, e.g., all heuristic searching and shortest path caching methods and some of the shortest path indexing methods (i.e., hub labeling), can only improve the efficiency of the shortest distance query, i.e., computing the shortest path distance between a single source node and a single destination node in a road network. Therefore, these methods cannot be extended to improve the efficiency of solving this problem since we need to obtain all nodes with their shortest path distance values that are within $b\sqrt{\ln\left(\frac{1}{\delta}\right)}$ (see the core idea 1 in Section III-A). Note that some of the shortest path indexing methods (i.e., the hierarchical indexing approach) may further improve the efficiency of computing shortest path distances. Yet, these methods cannot significantly improve the efficiency of generating NKDV since the main bottleneck lies in processing the data points in a road network (which has been verified by [1], [2]).

Function approximation methods. In the literature, many researchers have adopted the function approximation methods [74]–[87] to approximate the complicated kernel functions in order to improve the efficiency of kernel-based machine learning tasks, which are closely related to Section III-B. However, like [49], [50], these research studies do not consider any properties of road networks, which cannot be simply extended to support our problem (see Problem 2). Furthermore, to the best of our knowledge, this is the first work that utilizes the piecewise-linear function to approximate the Gaussian kernel so that our solution can simultaneously (1) achieve the non-trivial absolute-error guarantee and (2) lower the worst-case time-complexity for generating NKDV.

VI. Conclusion

In this paper, we study network kernel density visualization (NKDV) with the Gaussian kernel. Although many efficient algorithms (e.g., [1], [2]) have been developed to improve the efficiency of NKDV, none of them can be extended to support the commonly used Gaussian kernel. Therefore, NKDV with the Gaussian kernel still suffers from high time complexity, which does not scale to large datasets (e.g., with million data points). To tackle this efficiency issue, we develop the pioneering Piecewise-Linear Approximate solutioN (PLAN), which can simultaneously (1) reduce the time complexity, (2) achieve the similar space complexity, and (3) achieve the absolute error guarantee. By providing further optimization, PLAN+ attains the lowest time complexity. In practice, PLAN and PLAN+ achieve 32.47x to 2936.88x speedups and only incur 1.26x to 2.15x space overhead over the state-of-the-art (SOTA) solution without degrading the visualization quality.

In the future, we will investigate how to extend PLAN to distributed settings for handling extremely large-scale location datasets (e.g., with billion/trillion-scale data points). Moreover, we will examine whether PLAN can support other analysis tasks in road networks [4], e.g., network clustering.

VII. Supplementary Materials

A. Proof of Lemma 1

Proof. Here, we consider $|A_P(q) - \mathcal{F}_P(q)|$. Note that

$$\begin{aligned}
&|A_P(q) - \mathcal{F}_P(q)| \\
&= \Big|\frac{1}{|P|}\sum_{p\in P} PL\Big(\frac{1}{b^2}d_G(q,p)^2\Big) - \frac{1}{|P|}\sum_{p\in P}\exp\Big(-\frac{1}{b^2}d_G(q,p)^2\Big)\Big| \\
&\leq \frac{1}{|P|}\sum_{p\in P}\Big|PL\Big(\frac{1}{b^2}d_G(q,p)^2\Big) - \exp\Big(-\frac{1}{b^2}d_G(q,p)^2\Big)\Big|
\end{aligned}$$

Since $\frac{1}{b^2}d_G(q,p)^2$ must be either in $\mathcal{I}_1$, $\mathcal{I}_2$,..., $\mathcal{I}_{M-1}$, or $\mathcal{I}_M$ for any data point p in P, we can further conclude that

$$\begin{aligned}
&|A_P(q) - \mathcal{F}_P(q)| \\
&\leq \frac{1}{|P|}\Big(\sum_{i=1}^{M-1}\sum_{\frac{1}{b^2}d_G(q,p)^2\in\mathcal{I}_i}\Big|m_i\Big(\frac{1}{b^2}d_G(q,p)^2\Big) + k_i \\
&- \exp\Big(-\frac{1}{b^2}d_G(q,p)^2\Big)\Big| + \sum_{\frac{1}{b^2}d_G(q,p)^2\in\mathcal{I}_M}\Big|\exp\Big(-\frac{1}{b^2}d_G(q,p)^2\Big)\Big|\Big) \\
&\leq \frac{1}{|P|}\sum_{i=1}^{M}\sum_{\frac{1}{b^2}d_G(q,p)^2\in\mathcal{I}_i}\epsilon = \frac{|P|\epsilon}{|P|} = \epsilon
\end{aligned}$$

Note that the last inequality is based on the conditions 1 and 2, i.e.,

$$\begin{cases} 0 \leq m_i x + k_i - \exp(-x) \leq \epsilon & \text{if } x \in \mathcal{I}_i \ (1 \leq i \leq M-1) \\ 0 \leq \exp(-x) \leq \epsilon & \text{if } x \in \mathcal{I}_M \end{cases}$$

which indicates that

$$\begin{cases} |m_i x + k_i - \exp(-x)| \leq \epsilon & \text{if } x \in \mathcal{I}_i \ (1 \leq i \leq M-1) \\ |\exp(-x)| \leq \epsilon & \text{if } x \in \mathcal{I}_M \end{cases}$$

□

B. Proof of Lemma 2

Proof. Here, we consider the gap function $G(x)$ between the linear function and the exponential function $\exp(-x)$ (see Figure 5), where

$$G(x) = \frac{\exp(-u) - \exp(-\ell)}{u-\ell}x + \frac{u\exp(-\ell) - \ell\exp(-u)}{u-\ell} - \exp(-x)$$

By setting the differentiation of this function to be 0, we have

$$\begin{aligned}
&\frac{dG(x)}{dx}\Big|_{x=x^*} = \frac{\exp(-u)-\exp(-\ell)}{u-\ell} + \exp(-x^*) = 0 \\
\implies\ & x^* = \ln\Big(\frac{u-\ell}{\exp(-\ell)-\exp(-u)}\Big)
\end{aligned}$$

Since $\frac{d^2G(x)}{dx^2} = -\exp(-x) \leq 0$ (no matter which x we choose), this x^* attains the local maxima. Furthermore, we also note that (1) $\lim_{x\to-\infty}G(x) = \lim_{x\to\infty}G(x) = -\infty$, (2) there is only one local maxima, and (3) $G(x)$ is continuous with respect to x. As such, we conclude that x^* is also the global optima for $G(x)$.

Hence, by setting the gap $G(x)$ for $x = x^*$ to be the largest value ϵ, we can maximize $u-\ell$, which results in $h(u) = \epsilon$. □

C. Proof of Theorem 1

Proof. In Algorithm 2, we need to augment $a^{(deg)}_{P(v,p)}$ and $a^{(deg)}_{P(w,p)}$ for each data point p in every edge $e = (v,w)$ in advance (from lines 2 to 5). Therefore, the time complexity is $O(|V| + |E| + |P|)$. Moreover, we also need to call the shortest path algorithm in the nodes s and t (line 7) of every edge $\widehat{e} = (s,t)$ (line 6). As such, it takes $O(|E|T_{\text{SP}})$ time in line 7 throughout all iterations in line 6. With these shortest path distances and a lixel q in $\widehat{e}$, we can (1) evaluate $d_G(q,v)$ and $d_G(q,w)$ in $O(1)$ time and (2) compute $\widehat{A}_{P(e)}(q)$ in $O(M\log|P(e)|)$ time (based on Lemma 3) for each edge $e = (v,w)$. Therefore, the time complexity of lines 9 to 13 is

$$O\Big(\sum_{e\in E} M\log|P(e)|\Big) \leq O\Big(M|E|\log\Big(\frac{|P|}{|E|}\Big)\Big)$$

Note that this inequality is based on the proof in Theorem 2 of [2]. Since there are L lixels in total, the time complexity of lines 9 to 13 throughout all iterations in line 6 and line 8 is $O\big(LM|E|\log\big(\frac{|P|}{|E|}\big)\big)$. Hence, the time complexity of Algorithm 2 is $O\big(|E|T_{\text{SP}} + LM|E|\log\big(\frac{|P|}{|E|}\big)\big)$. □

D. Proof of Lemma 4

Proof. Since we need to iteratively divide this axis until each part only covers one interval, the number of levels in Figure 12 is $O(\log\Gamma)$. Here, we let the number of data points of the β^{th} part in the α^{th} level be $\mathcal{P}_{\alpha,\beta}$. Moreover, the number of parts in the α^{th} level is at most 2^α. Therefore, we can conclude that the time complexity of the α^{th} level is $O(\sum_{\beta=1}^{2^\alpha}\log\mathcal{P}_{\alpha,\beta})$, where $\sum_{\beta=1}^{2^\alpha}\mathcal{P}_{\alpha,\beta} = m$. With the AM-GM inequality [88], we have

$$\sum_{\beta=1}^{2^\alpha}\log\mathcal{P}_{\alpha,\beta} = \log\Big(\prod_{\beta=1}^{2^\alpha}\mathcal{P}_{\alpha,\beta}\Big) \leq 2^\alpha\log\Big(\frac{m}{2^\alpha}\Big)$$

Since there are $O(\log\Gamma)$ levels in Figure 12, we conclude that the time complexity for solving Problem 3 is

$$\begin{aligned}
&O\Big(\sum_{\alpha=0}^{\log\Gamma}\sum_{\beta=1}^{2^\alpha}\log\mathcal{P}_{\alpha,\beta}\Big) \leq O\Big(\sum_{\alpha=0}^{\log\Gamma}2^\alpha\log\Big(\frac{m}{2^\alpha}\Big)\Big) \\
&= O\Big(\log m\sum_{\alpha=0}^{\log\Gamma}2^\alpha - \log 2\sum_{\alpha=0}^{\log\Gamma}\alpha\cdot 2^\alpha\Big) = O\Big(\Gamma\log\frac{m}{\Gamma}\Big)
\end{aligned}$$

As a remark, the last equality is derived based on (1) obtaining the value of the arithmetic series in the first term and (2) obtaining the value of the arithmetico-geometric series in the second term of the first equality. More details about the closed forms of these two series can be found in [89]. □

E. Proof of Theorem 2

Proof. Since the main difference between PLAN and PLAN+ is the evaluation of $\widehat{A}_{P(e)}(q)$, the pseudocodes of these two methods are the same (i.e., Algorithm 2). Moreover, this proof is similar to the proof of Theorem 1 except that the time complexity of line 9 to line 13 (based on Lemma 5) is

$$O\Big(\sum_{e\in E}M\log\Big(\frac{|P(e)|}{M}\Big)\Big) \leq O\Big(M|E|\log\Big(\frac{|P|}{M|E|}\Big)\Big)$$

Hence, we conclude that the time complexity of PLAN+ is $O\big(|E|T_{\text{SP}} + LM|E|\log\big(\frac{|P|}{M|E|}\big)\big)$. □

AI-Generated Content Acknowledgement

We do not use any AI tools (e.g., ChatGPT and DeepSeek) to write this article.

References

[1] T. N. Chan, R. Zang, B. Zhu, L. H. U, D. Wu, and J. Xu, "LION: fast and high-resolution network kernel density visualization," *Proc. VLDB Endow.*, vol. 17, no. 6, pp. 1255–1268, 2024. [Online]. Available: https://www.vldb.org/pvldb/vol17/p1255-chan.pdf

[2] T. N. Chan, Z. Li, L. H. U, J. Xu, and R. Cheng, "Fast augmentation algorithms for network kernel density visualization," *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1503–1516, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p1503-chan.pdf

[3] Z. Xie and J. Yan, "Detecting traffic accident clusters with network kernel density estimation and local spatial statistics: an integrated approach," *Journal of Transport Geography*, vol. 31, pp. 64 – 71, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0966692313000860

[4] A. Okabe and K. Sugihara, *Spatial Analysis Along Networks: Statistical and Computational Methods*, ser. Statistics in Practice. Wiley, 2012. [Online]. Available: https://books.google.com.hk/books?id=48GRqj51_W8C

[5] Z. Xie and J. Yan, "Kernel density estimation of traffic accidents in a network space," *Computers, Environment and Urban Systems*, vol. 32, no. 5, pp. 396 – 406, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0198971508000318

[6] I. G. B. Putra, P.-F. Kuo, and D. Lord, "Estimating the effectiveness of marked sidewalks: An application of the spatial causality approach," *Accident Analysis & Prevention*, vol. 206, p. 107699, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457524002446

[7] H. Chen, X. Gao, H. Li, and Z. Yang, "A framework for the optimal deployment of police drones based on street-level crime risk," *Applied Geography*, vol. 162, p. 103178, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0143622823003090

[8] Y. Wu and Y. Li, ""hot street" of crime detection in london borough and lockdown impacts," *Geo-spatial Information Science*, vol. 26, no. 4, pp. 716–732, 2023.

[9] M. Zhu, H. Li, N. Sze, and G. Ren, "Exploring the impacts of street layout on the frequency of pedestrian crashes: A micro-level study," *Journal of Safety Research*, vol. 81, pp. 91–100, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022437522000093

[10] H. Harirforoush and L. Bellalite, "A new integrated gis-based analysis to detect hotspots: A case study of the city of sherbrooke," *Accident Analysis & Prevention*, vol. 130, pp. 62 – 74, 2019, road Safety Data Considerations. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0001457516303013

[11] M. Deng, X. Yang, Y. Shi, J. Gong, Y. Liu, and H. Liu, "A density-based approach for detecting network-constrained clusters in spatial point events," *International Journal of Geographical Information Science*, vol. 33, no. 3, pp. 466–488, 2019. [Online]. Available: https://doi.org/10.1080/13658816.2018.1541177

[12] Álvaro Briz-Redón, F. Martínez-Ruiz, and F. Montes, "Identification of differential risk hotspots for collision and vehicle type in a directed linear network," *Accident Analysis & Prevention*, vol. 132, p. 105278, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457519301095

[13] D. Boss, T. Nelson, and M. Winters, "Monitoring city wide patterns of cycling safety," *Accident Analysis & Prevention*, vol. 111, pp. 101–108, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457517303950

[14] M. M. Moradi, F. J. Rodríguez-Cortés, and J. Mateu, "On kernel-based intensity estimation of spatial point patterns on linear networks," *Journal of Computational and Graphical Statistics*, vol. 27, no. 2, pp. 302–311, 2018.

[15] G. Rosser, T. O. Davies, K. Bowers, S. D. Johnson, and T. Cheng, "Predictive crime mapping: Arbitrary grids or street networks?" *Journal of Quantitative Criminology*, vol. 33, pp. 569 – 594, 2017.

[16] J. Benedek, S. M. Ciobanu, and T. C. Man, "Hotspots and social background of urban traffic crashes: A case study in cluj-napoca (romania)," *Accident Analysis & Prevention*, vol. 87, pp. 117–126, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000145751530141X

[17] Q. Li, T. Zhang, H. Wang, and Z. Zeng, "Dynamic accessibility mapping using floating car data: a network-constrained density estimation approach," *Journal of Transport Geography*, vol. 19, no. 3, pp. 379 – 393, 2011, special Issue : Geographic Information Systems for Transportation. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0966692310001158

[18] J. Khalil, D. Yan, L. Yuan, M. Jafarzadehfadaki, S. Adhikari, J. Han, V. Sisiopiku, and Z. Jiang, "Urban traffic simulation with shared mobility services: An approach using spatiotemporal network kernel density estimation and matsim," *ACM Trans. Spatial Algorithms Syst.*, Nov. 2024, just Accepted. [Online]. Available: https://doi.org/10.1145/3704918

[19] K. Qian and Y. Li, "Safer traffic recovery from the pandemic in london–spatiotemporal data mining of car crashes," *Applied Spatial Analysis and Policy*, vol. 17, no. 1, pp. 87–113, 2024.

[20] J. Khalil, D. Yan, L. Yuan, M. Jafarzadehfadaki, S. Adhikari, V. P. Sisiopiku, and Z. Jiang, "Realistic urban traffic simulation with ride-hailing services: a revisit to network kernel density estimation (systems paper)," in *SIGSPATIAL*. ACM, 2022, pp. 29:1–29:10. [Online]. Available: https://doi.org/10.1145/3557915.3560963

[21] T. Wang, Y. Wang, X. Zhao, and X. Fu, "Spatial distribution pattern of the customer count and satisfaction of commercial facilities based on social network review data in Beijing, China," *Computers, Environment and Urban Systems*, vol. 71, pp. 88–97, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0198971517302843

[22] J. Delso, B. Martín, and E. Ortega, "A new procedure using network analysis and kernel density estimations to evaluate the effect of urban configurations on pedestrian mobility. the case study of vitoria –gasteiz," *Journal of Transport Geography*, vol. 67, pp. 61–72, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0966692317305513

[23] Y. Rui, Z. Yang, T. Qian, S. Khalid, N. Xia, and J. Wang, "Network-constrained and category-based point pattern analysis for suguo retail stores in nanjing, china," *International Journal of Geographical Information Science*, vol. 30, no. 2, pp. 186–199, 2016.

[24] J. Ni, T. Qian, C. Xi, Y. Rui, and J. Wang, "Spatial distribution characteristics of healthcare facilities in nanjing: Network point pattern analysis and correlation analysis," *International journal of environmental research and public health*, vol. 13, no. 8, p. 833, 2016.

[25] "Chicago Data Portal," https://data.cityofchicago.org/Service-Requests/311-Service-Requests/v6vf-nfxy/about_data.

[26] S. Rakshit, A. Baddeley, and G. Nair, "Efficient code for second order analysis of events on a linear network," *Journal of Statistical Software, Articles*, vol. 90, no. 1, pp. 1–37, 2019. [Online]. Available: https://www.jstatsoft.org/v090/i01

[27] "Arcgis," https://www.arcgis.com/index.html, 2025.

[28] "spnetwork (nkde: Network kernel density estimate)," https://www.rdocumentation.org/packages/spNetwork/versions/0.4.3.2/topics/nkde, 2025.

[29] "Deck.gl," https://deck.gl/, 2025.

[30] "seaborn: statistical data visualization," https://seaborn.pydata.org/, 2025.

[31] J. Gelb and P. Apparicio, "Temporal network kernel density estimation," *Geographical Analysis*, vol. 56, no. 1, pp. 62–78, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/gean.12368

[32] B. Shen, X. Xu, J. Li, A. Plaza, and Q. Huang, "Unfolding spatial-temporal patterns of taxi trip based on an improved network kernel density estimation," *ISPRS International Journal of Geo-Information*, vol. 9, no. 11, p. 683, 2020.

[33] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis*. Cengage Learning, 2015.

[34] "DataSF: Police Department Incident Reports," https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-Historical-2003/tmnf-yvry/about_data and https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-2018-to-Present/wg3w-h783/about_data.

[35] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126 – 139, 2017.

[36] "Open data Minneapolis," https://opendata.minneapolismn.gov/search?q=911\%20calls.

[37] "NYC open data," https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95.

[38] Y. Zheng and J. M. Phillips, "L∞ error and bandwidth selection for kernel density estimates of large data," in *SIGKDD*, 2015, pp. 1533–1542. [Online]. Available: http://doi.acm.org/10.1145/2783258.2783357

[39] Y. Zheng, J. Jestes, J. M. Phillips, and F. Li, "Quality and efficiency for kernel density estimates in large data," in *SIGMOD*, 2013, pp. 433–444.

[40] T. N. Chan, B. Zhu, L. H. U, D. Wu, W. Tu, and J. Xu, "A fast, versatile, and user-friendly plugin for kernel density analysis," in *ICDE*, 2026 (To appear).

[41] T. N. Chan, L. H. U, B. Choi, J. Xu, and R. Cheng, "Large-scale geospatial analytics: Problems, challenges, and opportunities," in *SIGMOD Companion*. ACM, 2023, pp. 21–29. [Online]. Available: https://doi.org/10.1145/3555041.3589401

[42] ——, "Kernel density visualization for big geospatial data: Algorithms and applications," in *MDM*. IEEE, 2023, pp. 231–234. [Online]. Available: https://doi.org/10.1109/MDM58254.2023.00046

[43] T. N. Chan, L. H. U, B. Choi, and J. Xu, "SLAM: efficient sweep line algorithms for kernel density visualization," in *SIGMOD*. ACM, 2022, pp. 2120–2134. [Online]. Available: https://doi.org/10.1145/3514221.3517823

[44] T. N. Chan, P. L. Ip, L. H. U, B. Choi, and J. Xu, "SAFE: A share-and-aggregate bandwidth exploration framework for kernel density visualization," *Proc. VLDB Endow.*, vol. 15, no. 3, pp. 513–526, 2022.

[45] T. N. Chan, L. H. U, R. Cheng, M. L. Yiu, and S. Mittal, "Efficient algorithms for kernel aggregation queries," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2726–2739, 2022. [Online]. Available: https://doi.org/10.1109/TKDE.2020.3018376

[46] T. N. Chan, P. L. Ip, K. Zhao, L. H. U, B. Choi, and J. Xu, "LIBKDV: A versatile kernel density visualization library for geospatial analytics," *Proc. VLDB Endow.*, vol. 15, no. 12, pp. 3606–3609, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol15/p3606-chan.pdf

[47] T. N. Chan, P. L. Ip, L. H. U, W. H. Tong, S. Mittal, Y. Li, and R. Cheng, "KDV-Explorer: A near real-time kernel density visualization system for spatial analysis," *Proc. VLDB Endow.*, vol. 14, no. 12, pp. 2655–2658, 2021.

[48] Y. Zheng, Y. Ou, A. Lex, and J. M. Phillips, "Visualization of big spatial data using coresets for kernel density estimates," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 524–534, 2021. [Online]. Available: https://doi.org/10.1109/TBDATA.2019.2913655

[49] T. N. Chan, R. Cheng, and M. L. Yiu, "QUAD: Quadratic-bound-based kernel density visualization," in *SIGMOD*, 2020, pp. 35–50. [Online]. Available: https://doi.org/10.1145/3318464.3380561

[50] T. N. Chan, M. L. Yiu, and L. H. U, "KARL: Fast kernel aggregation queries," in *ICDE*, 2019, pp. 542–553. [Online]. Available: https://doi.org/10.1109/ICDE.2019.00055

[51] J. M. Phillips and W. M. Tai, "Improved coresets for kernel density estimates," in *SODA*, 2018, pp. 2718–2727. [Online]. Available: https://doi.org/10.1137/1.9781611975031.173

[52] ——, "Near-optimal coresets of kernel density estimates," in *SOCG*, 2018, pp. 66:1–66:13. [Online]. Available: https://doi.org/10.4230/LIPIcs.SoCG.2018.66

[53] E. Gan and P. Bailis, "Scalable kernel density classification via threshold-based pruning," in *ACM SIGMOD*, 2017, pp. 945–959.

[54] J. M. Phillips, "ϵ-samples for kernels," in *SODA*, 2013, pp. 1622–1632. [Online]. Available: https://doi.org/10.1137/1.9781611973105.116

[55] T. N. Chan, R. Zang, P. L. Ip, L. H. U, and J. Xu, "PyNKDV: An efficient network kernel density visualization library for geospatial analytic systems," in *SIGMOD Companion*. ACM, 2023, pp. 99–102. [Online]. Available: https://doi.org/10.1145/3555041.3589711

[56] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[57] H. Aljazzar and S. Leue, "K$*$: A heuristic search algorithm for finding the k shortest paths," *Artificial Intelligence*, vol. 175, no. 18, pp. 2129–2154, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370211000865

[58] L. Fu, D. Sun, and L. Rilett, "Heuristic shortest path algorithms for transportation applications: State of the art," *Computers & Operations Research*, vol. 33, no. 11, pp. 3324–3343, 2006, part Special Issue: Operations Research and Data Mining. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030505480500122X

[59] D. Ouyang, D. Wen, L. Qin, L. Chang, X. Lin, and Y. Zhang, "When hierarchy meets 2-hop-labeling: efficient shortest distance and path queries on road networks," *VLDB J.*, vol. 32, no. 6, pp. 1263–1287, 2023. [Online]. Available: https://doi.org/10.1007/s00778-023-00789-x

[60] M. Zhang, L. Li, W. Hua, R. Mao, P. Chao, and X. Zhou, "Dynamic hub labeling for road networks," in *ICDE*. IEEE, 2021, pp. 336–347. [Online]. Available: https://doi.org/10.1109/ICDE51399.2021.00036

[61] W. Li, M. Qiao, L. Qin, Y. Zhang, L. Chang, and X. Lin, "Scaling distance labeling on small-world networks," in *SIGMOD*. ACM, 2019, pp. 1060–1077. [Online]. Available: https://doi.org/10.1145/3299869.3319877

[62] D. Ouyang, L. Qin, L. Chang, X. Lin, Y. Zhang, and Q. Zhu, "When hierarchy meets 2-hop-labeling: Efficient shortest distance queries on road networks," in *SIGMOD*. ACM, 2018, pp. 709–724. [Online]. Available: https://doi.org/10.1145/3183713.3196913

[63] Y. Li, L. H. U, M. L. Yiu, and N. M. Kou, "An experimental study on hub labeling based shortest path algorithms," *Proc. VLDB Endow.*, vol. 11, no. 4, pp. 445–457, 2017. [Online]. Available: http://www.vldb.org/pvldb/vol11/p445-li.pdf

[64] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and distance queries via 2-hop labels," *SIAM J. Comput.*, vol. 32, no. 5, pp. 1338–1355, 2003. [Online]. Available: https://doi.org/10.1137/S0097539702403098

[65] D. Ouyang, L. Yuan, L. Qin, L. Chang, Y. Zhang, and X. Lin, "Efficient shortest path index maintenance on dynamic road networks with theoretical guarantees," *Proc. VLDB Endow.*, vol. 13, no. 5, pp. 602–615, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p602-ouyang.pdf

[66] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou, "Shortest path and distance queries on road networks: towards bridging theory and practice," in *SIGMOD*, 2013, pp. 857–868. [Online]. Available: https://doi.org/10.1145/2463676.2465277

[67] P. Sanders and D. Schultes, "Engineering highway hierarchies," *ACM J. Exp. Algorithmics*, vol. 17, no. 1, 2012. [Online]. Available: https://doi.org/10.1145/2133803.2330080

[68] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner, "Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm," *ACM J. Exp. Algorithmics*, vol. 15, 2010. [Online]. Available: https://doi.org/10.1145/1671970.1671976

[69] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu, and S. Zhou, "Shortest path and distance queries on road networks: An experimental evaluation," *Proc. VLDB Endow.*, vol. 5, no. 5, pp. 406–417, 2012. [Online]. Available: http://vldb.org/pvldb/vol5/p406_lingkunwu_vldb2012.pdf

[70] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *WEA*, 2008, pp. 319–333. [Online]. Available: https://doi.org/10.1007/978-3-540-68552-4_24

[71] L. Yuan, K. Hao, X. Lin, and W. Zhang, "Batch hop-constrained s-t simple path query processing in large graphs," in *ICDE*. IEEE, 2024, pp. 2557–2569. [Online]. Available: https://doi.org/10.1109/ICDE60146.2024.00201

[72] L. Li, M. Zhang, W. Hua, and X. Zhou, "Fast query decomposition for batch shortest path processing in road networks," in *ICDE*, 2020, pp. 1189–1200. [Online]. Available: https://doi.org/10.1109/ICDE48307.2020.00107

[73] J. R. Thomsen, M. L. Yiu, and C. S. Jensen, "Effective caching of shortest paths for location-based services," in *SIGMOD*, 2012, pp. 313–324. [Online]. Available: https://doi.org/10.1145/2213836.2213872

[74] J. Wacker, M. Kanagawa, and M. Filippone, "Improved random features for dot product kernels," *Journal of Machine Learning Research*, vol. 25, no. 235, pp. 1–75, 2024. [Online]. Available: http://jmlr.org/papers/v25/22-0118.html

[75] T. N. Chan, Z. Li, L. H. U, and R. Cheng, "PLAME: piecewise-linear approximate measure for additive kernel SVM," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 9985–9997, 2023. [Online]. Available: https://doi.org/10.1109/TKDE.2023.3253263

[76] A. Tripp, S. Bacallado, S. Singh, and J. M. Hernández-Lobato, "Tanimoto random features for scalable molecular machine learning," in *NeurIPS*, 2023. [Online]. Available: http://papers.nips.cc/paper_files/paper/2023/hash/6a69d44b3386e50c06f7107ef4f29302-Abstract-Conference.html

[77] F. Liu, X. Huang, Y. Chen, and J. A. K. Suykens, "Random features for kernel approximation: A survey on algorithms, theory, and beyond,"

IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 10, pp. 7128–7148, 2022. [Online]. Available: https://doi.org/10.1109/TPAMI.2021.3097011

[78] V. Kurková and D. Coufal, "Translation-invariant kernels for multivariable approximation," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 11, pp. 5072–5081, 2021. [Online]. Available: https://doi.org/10.1109/TNNLS.2020.3026720

[79] T. Erdélyi, C. Musco, and C. Musco, "Fourier sparse leverage scores and approximate kernel learning," in *NeurIPS*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/012d9fe15b2493f21902cd55603382ec-Abstract.html

[80] R. Agrawal, T. Campbell, J. H. Huggins, and T. Broderick, "Data-dependent compression of random features for large-scale kernel approximation," in *AISTATS*, ser. Proceedings of Machine Learning Research, vol. 89. PMLR, 2019, pp. 1822–1831. [Online]. Available: http://proceedings.mlr.press/v89/agrawal19a.html

[81] H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, and A. Zandieh, "Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees," in *ICML*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 253–262. [Online]. Available: https://proceedings.mlr.press/v70/avron17a.html

[82] J. Pennington, F. X. Yu, and S. Kumar, "Spherical random features for polynomial kernels," in *NIPS*, 2015, pp. 1846–1854. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/f7f580e11d00a75814d2ded41fe8e8fe-Abstract.html

[83] J. Wu and H. Yang, "Linear regression-based efficient SVM learning for large-scale classification," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 26, no. 10, pp. 2357–2369, 2015. [Online]. Available: https://doi.org/10.1109/TNNLS.2014.2382123

[84] N. Pham and R. Pagh, "Fast and scalable polynomial kernels via explicit feature maps," in *SIGKDD*. ACM, 2013, pp. 239–247. [Online]. Available: https://doi.org/10.1145/2487575.2487591

[85] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, 2012. [Online]. Available: https://doi.org/10.1109/TPAMI.2011.153

[86] J. Wu, W. Tan, and J. M. Rehg, "Efficient and effective visual codebook generation using additive kernels," *Journal of Machine Learning Research*, vol. 12, pp. 3097–3118, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2078205

[87] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *NIPS*, 2007, pp. 1177–1184. [Online]. Available: https://proceedings.neurips.cc/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html

[88] J. Steele, *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*, ser. MAA problem books series. Cambridge University Press, 2004. [Online]. Available: https://books.google.com.hk/books?id=7GDyRMrlgDsC

[89] K. F. Riley, M. P. Hobson, and S. J. Bence, *Mathematical methods for physics and engineering: a comprehensive guide*. Cambridge university press, 2006.