LION: Fast and High-Resolution Network Kernel Density Visualization

Tsz Nam Chan
Shenzhen University
edisonchan@szu.edu.cn

Rui Zang
Hong Kong Baptist University
19251017@life.hkbu.edu.hk

Bojian Zhu
Hong Kong Baptist University
csbjzhu@comp.hkbu.edu.hk

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

Dingming Wu
Shenzhen University
dingming@szu.edu.cn

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

ABSTRACT

Network Kernel Density Visualization (NKDV) has often been used in a wide range of applications, e.g., criminology, transportation science, and urban planning. However, NKDV is computationally expensive, which cannot be scalable to large-scale datasets and high resolution sizes. Although a recent work, called aggregate distance augmentation (ADA), has been developed for improving the efficiency to generate NKDV, this method is still slow and does not take the resolution size into account for optimizing the efficiency. In this paper, we develop a new solution, called LION, which can reduce the worst-case time complexity for generating high-resolution NKDV, without increasing the space complexity. Experiment results on four large-scale location datasets verify that LION can achieve 2.86x to 35.36x speedup compared with the state-of-the-art ADA method.

PVLDB Reference Format:
Tsz Nam Chan, Rui Zang, Bojian Zhu, Leong Hou U, Dingming Wu, and Jianliang Xu. LION: Fast and High-Resolution Network Kernel Density Visualization. PVLDB, 17(6): 1255 - 1268, 2024.
doi:10.14778/3648160.3648168

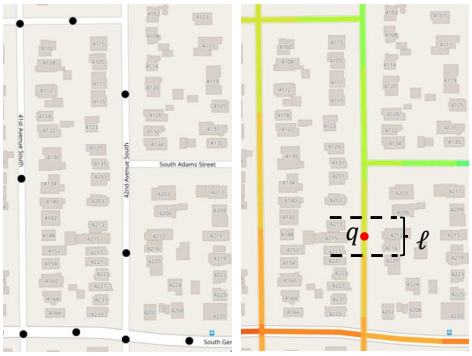
PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/edisonchan2013928/LION.

1 INTRODUCTION

Network Kernel Density Visualization (NKDV) [23] has found extensive use in various fields, including criminology, transportation science, and urban planning. Transportation scientists [12, 15, 40, 43, 47, 75, 76] use NKDV to analyze traffic/traffic accident hotspots in different cities, while criminologists [13, 44, 64, 74] apply it to detect crime hotspots in different geographical regions. Urban planners [34, 43, 53, 65, 68, 71, 73, 78] use NKDV to analyze different types of social phenomena (e.g., human mobility [34, 68, 71]) in order to inform policy-making. Given the broad applicability of NKDV, several software suites, including spNetwork [8, 39] (an R package) and SANET [6] (a plugin for QGIS [60] and ArcGIS [1]), have been developed to support this tool.

Proceedings of the VLDB Endowment, Vol. 17, No. 6 ISSN 2150-8097.
doi:10.14778/3648160.3648168

Dingming Wu is the corresponding author of this paper.



(a) Location dataset (b) NKDV

Figure 1: Illustration of NKDV for a location dataset (black points), where the red color and the green color of a lixel denote the hotspot region and the coldspot region, respectively.

Figure 1 shows an example to illustrate how domain experts generate NKDV for a location dataset P in a road network G. Note that they first divide each road into a set of lixels q (i.e., small road segments with size ℓ).[1] Then, they color each lixel q based on the network kernel density function (cf. Equation 1).

$$\mathcal{F}_P(q) = \sum_{p_i \in P} w \cdot K_G(q, p_i) \quad (1)$$

where w and $K_G(q, p_i)$ denote the normalization constant and the kernel function, respectively. A variety of kernel functions can be used in Equation 1, which is shown in Table 1.

Table 1: Some representative kernel functions, where $d_G(q, p_i)$ and b are the shortest path distance and the bandwidth parameter, respectively.

Kernel	$K_G(q, p_i)$	Ref.
Triangular	$\begin{cases} 1 - \frac{1}{b} d_G(q, p_i) & \text{if } d_G(q, p_i) \le b \\ 0 & \text{otherwise} \end{cases}$	[17, 47]
Epanechnikov	$\begin{cases} 1 - \frac{1}{b^2} d_G(q, p_i)^2 & \text{if } d_G(q, p_i) \le b \\ 0 & \text{otherwise} \end{cases}$	[17, 79]
Quartic	$\begin{cases} (1 - \frac{1}{b^2} d_G(q, p_i)^2)^2 & \text{if } d_G(q, p_i) \le b \\ 0 & \text{otherwise} \end{cases}$	[47, 76]

[1] The size of each lixel is generally fixed, except for those lixels that are the closest to the nodes. For simplicity, we assume that all lixels have the same size ℓ in this paper. However, all methods can be easily extended to support this exceptional case.

Table 2: State-of-the-art methods for generating NKDV, where $|V|$, $|E|$, n, L, T_{SP}, and S_{SP} denote the number of nodes, the number of edges, the dataset size, the number of lixels, the time complexity of the shortest path algorithm, and the space complexity of the shortest path algorithm, respectively.

Method	Time complexity	Space complexity	Ref.
ADA	$O\left(\lvert E\rvert T_{\text{SP}} + L\lvert E\rvert \log\left(\frac{n}{\lvert E\rvert}\right)\right)$ (cf. Theorem 1)	$O(\lvert V\rvert + \lvert E\rvert + n + L + S_{\text{SP}})$ (cf. Theorem 1)	Section 2.2 [23]
LION	$O(\lvert E\rvert T_{\text{SP}} + n\lvert E\rvert + \lvert E\rvert^2 + L)$ (cf. Theorem 2)	$O(\lvert V\rvert + \lvert E\rvert + n + L + S_{\text{SP}})$ (cf. Theorem 3)	Section 3

Despite being a popular spatial analytic tool, generating a single NKDV is computationally expensive, making it difficult to scale to large resolution sizes (i.e., large number of lixels) and large-scale datasets. As an example, consider the Detroit 911-call location dataset [3] (with 1.931 million location data points) and a fixed lixel size $\ell = 5$m. A naïve algorithm takes more than one day for generating a single NKDV in the Detroit road network. Therefore, many domain experts [52, 61, 78] have reported inefficiency issues when using the NKDV tool.

To overcome this issue, Chan et al. [23] propose the aggregate distance augmentation (ADA) algorithm, which theoretically reduces the worst-case time complexity for generating NKDV. Despite this, ADA only aims to improve the efficiency for supporting large-scale datasets, which omits the optimization opportunity for another parameter: the number of lixels in a road network (a.k.a. resolution). However, in practice, the number of lixels in a road network can be larger than the number of data points. Consider the London traffic accident dataset [5] and the London road network. The total number of lixels (with $\ell = 5$m as the lixel size) in this road network is 2.95 million, while the total number of data points is only 0.838 million. Furthermore, domain experts [16, 47, 74] need to perform exploratory analysis for analyzing hotspots with different attribute types (e.g., only analyze the crime location data with the type "possession of weapon" in [74]), which can further increase the difference between the number of lixels and the number of data points.

Therefore, we pose a question. *Can we develop a theoretically improved solution for generating high-resolution NKDV (with a large number of lixels), without increasing the space complexity?* To answer this question, we develop a new solution, called LION, which further reduces the worst-case time complexity for generating NKDV (cf. Table 2) compared with the state-of-the-art ADA method given that the number of lixels L is larger than the dataset size n ($L > n$). In addition, LION also retains the same space complexity for generating NKDV (cf. Table 2). Experiment results on four location datasets demonstrate that LION can achieve 2.86x to 35.36x speedup compared with the state-of-the-art ADA method.

The rest of the paper is structured as follows. We first formally state the NKDV problem and discuss the state-of-the-art ADA method in Section 2. Then, we illustrate our efficient solution, LION, in Section 3. Next, we show our experiment results in Section 4. After that, we discuss the related work in Section 5. Lastly, we conclude this paper in Section 6.

2 PRELIMINARIES

In this section, we first formally define the NKDV problem in Section 2.1. Then, we discuss the state-of-the-art solution [23], namely aggregate distance augmentation (ADA), in Section 2.2.

2.1 Problem Statement

In order to generate NKDV (cf. Figure 1), we need to color each lixel q based on the network kernel density function (cf. Equation 1), which is formally defined in Problem 1.

Problem 1. *(NKDV [23]) Given a road network $G = (V, E)$, L lixels, and a location dataset P with size n, where each data point in P is on one and only one edge, we need to compute the network kernel density function $\mathcal{F}_P(q)$ (with Epanechnikov kernel) for each lixel q in E, where*

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

Due to the popularity of Epanechnikov kernel (cf. Table 1), we adopt this kernel by default. However, all the methods can also be extended to support other kernel functions in Table 1.

2.2 Aggregate Distance Augmentation (ADA)

Recently, Chan et al. [23] propose the state-of-the-art solution, called aggregate distance augmentation (ADA), which successfully reduces the worst-case time complexity for generating NKDV. In this section, we have an overview of this method.

Here, we first define the point set $P(e)$ of each edge e in a road network G (cf. Definition 1).

Definition 1. *Given an edge e in a road network $G = (V, E)$, $P(e)$ is the set of data points in this edge e.*

Based on Definition 1, the network kernel density function $\mathcal{F}_P(q)$ (cf. Equation 2) can be decomposed into the following expression.

$$\mathcal{F}_P(q) = \sum_{e \in E} \mathcal{F}_{P(e)}(q) \tag{3}$$

where $\mathcal{F}_{P(e)}(q)$ denotes the edge-e network kernel density function (which replaces P by $P(e)$ in Equation 2).

$$\mathcal{F}_{P(e)}(q) = \sum_{p_i \in P(e)} w \cdot \begin{cases} 1 - \frac{1}{b^2} d_G(q, p_i)^2 & \text{if } d_G(q, p_i) \le b \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Therefore, the core idea of [23] is to efficiently evaluate $\mathcal{F}_{P(e)}(q)$ in order to reduce the time complexity for computing $\mathcal{F}_P(q)$ (generating NKDV). Figure 2 shows how the ADA method achieves this goal. Note that this method augments the aggregate distance values $a^{(deg)}_{P(u,p)}$ (cf. Equation 5) and $a^{(deg)}_{P(v,p)}$ (cf. Equation 6) for all data points p in each edge $e = (u, v)$ in advance, where deg denotes the degree value (e.g., $deg = 0, 1, 2$ in the Epanechnikov kernel [23]).

$$a^{(deg)}_{P(u,p)} = \sum_{p_i \in P(u,p)} d_G(u, p_i)^{deg} \tag{5}$$

$$a^{(deg)}_{P(v,p)} = \sum_{p_i \in P(v,p)} d_G(v, p_i)^{deg} \tag{6}$$

where $P(u,p)$ and $P(v,p)$ denote the sets of data points from node u to p and from node v to p, respectively.

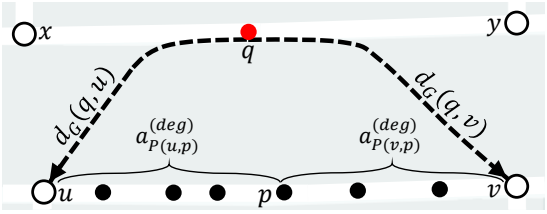


Figure 2: Illustration of the ADA method.

Based on the aggregate distance values $a^{(deg)}_{P(u,p)}$ and $a^{(deg)}_{P(v,p)}$, Chan et al. [23] show that they can compute $\mathcal{F}_{P(e)}(q)$ in $O(\log|P(e)|)$ time (instead of $O(|P(e)|)$ time) once the shortest path distances $d_G(q,u)$ and $d_G(q,v)$ (cf. the black dashed arrows in Figure 2) are available. Consider the case $d_G(q,u) \le b$ and $d_G(q,v) > b$ as an example. Note that the ADA method adopts the binary search approach (with $O(\log|P(e)|)$ time) to find p^* such that (1) $d_G(u,p^*)$ is the largest and (2) $d_G(u,p^*) \le b - d_G(q,u)$. With this p^*, they have:

$$\mathcal{F}_{P(e)}(q) = \sum_{p_i \in P(u,p^*)} w \cdot \left(1 - \frac{1}{b^2} d_G(q,p_i)^2\right)$$

which can be computed in $O(1)$ time using the following expression.

$$\mathcal{F}_{P(e)}(q) = w\left(1 - \frac{d_G(q,u)^2}{b^2}\right) a^{(0)}_{P(u,p^*)} - \frac{2w \cdot d_G(q,u)}{b^2} a^{(1)}_{P(u,p^*)} - \frac{w}{b^2} a^{(2)}_{P(u,p^*)}$$

As a remark, the ADA method can be further extended to support three other cases, namely (i) $d_G(q,u) \le b$ and $d_G(q,v) \le b$, (ii) $d_G(q,u) > b$ and $d_G(q,v) \le b$, and (iii) $d_G(q,u) > b$ and $d_G(q,v) > b$, for computing $\mathcal{F}_{P(e)}(q)$ in $O(\log|P(e)|)$ time based on the binary search approach and the aggregate terms (cf. Equation 5 and Equation 6).

Due to the lower time complexity for computing $\mathcal{F}_{P(e)}(q)$, Chan et al. [23] further show that the time complexity for generating NKDV is $O\big(|E|T_{SP} + L|E|\log\big(\frac{n}{|E|}\big)\big)$. In addition, since this method needs to access the road network, all data points, and all lixels, and adopt the shortest path algorithm, Chan et al. [23] also show that the space complexity of the ADA method is $O(|V|+|E|+n+L+S_{SP})$. Theorem 1 summarizes the theoretical results of this method.

Theorem 1. *[23] Given a road network $G=(V,E)$, L lixels, and a location dataset P with size n, the ADA method takes $O\big(|E|T_{SP} + L|E|\log\big(\frac{n}{|E|}\big)\big)$ time and $O(|V|+|E|+n+L+S_{SP})$ space to generate NKDV.*

Although the ADA method lowers the worst-case time complexity for generating NKDV, this time complexity still depends on the term $L|E|$, which can still be slow if the number of lixels L is large (i.e., generate a high-resolution NKDV).

3 OUR SOLUTION: LION

To further reduce the time complexity for generating high-resolution NKDV (with $L > n$), we propose the new solution, called LION. In this section, we first discuss two core ideas of LION, which are (1) the new expression of $\mathcal{F}_P(q)$ (cf. Section 3.1) and (2) influence regions of a data point (cf. Section 3.2). Based on these two core ideas, we then illustrate LION and analyze its time complexity in Section 3.3. Lastly, we discuss the space complexity of LION in Section 3.4.

3.1 New Expression of $\mathcal{F}_P(q)$

Consider any lixel q in the edge $\widehat{e} = (x,y)$. We can decompose the location dataset P into two parts, which are the data points in the edge $\widehat{e}$, namely $P(\widehat{e})$, and outside the edge $\widehat{e}$, namely $\widetilde{P} = P \backslash P(\widehat{e})$. Therefore, $\mathcal{F}_P(q)$ can be represented by the following expression.

$$\mathcal{F}_P(q) = \mathcal{F}_{P(\widehat{e})}(q) + \mathcal{F}_{\widetilde{P}}(q) \tag{7}$$

Since the first component $\mathcal{F}_{P(\widehat{e})}(q)$ in Equation 7 only depends on those data points in the same edge $\widehat{e}$ as the lixel q, generating NKDV based on $\mathcal{F}_{P(\widehat{e})}(q)$ for all lixels q is equivalent to solving the one-dimensional kernel density visualization problem (the special case of [25]) in each edge $\widehat{e}$ (cf. Figure 3).

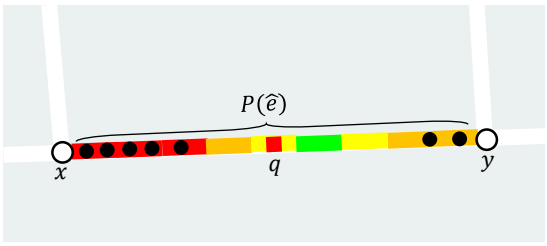


Figure 3: Generating NKDV using $\mathcal{F}_{P(\widehat{e})}(q)$ (all lixels q lie in the same edge $\widehat{e} = (x,y)$.) is equivalent to solving the one-dimensional kernel density visualization problem.

In Lemma 1, we show that the time complexity for using the first component $\mathcal{F}_{P(\widehat{e})}(q)$ of the network kernel density function to generate NKDV is $O(n+L)$.

Lemma 1. *Given a road network $G=(V,E)$, generating NKDV based on the network kernel density function $\mathcal{F}_{P(\widehat{e})}(q)$, where each lixel q lies in the same edge $\widehat{e}$, takes $O(n+L)$ time.*

Proof. In previous work, Chan et al. [25] have shown that generating kernel density visualization with the resolution size $X \times Y$ for a two-dimensional location dataset $L = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$ based on the kernel density function $D_L(\mathbf{q})$ (cf. Equation 8) takes $O(Y(X+m))$ time.

$$D_L(\mathbf{q}) = \sum_{\mathbf{p}_i \in L} w \cdot \begin{cases} 1 - \frac{1}{b^2} dist(\mathbf{q},\mathbf{p}_i)^2 & \text{if } dist(\mathbf{q},\mathbf{p}_i) \le b \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where w and $dist(\mathbf{q},\mathbf{p}_i)$ are the normalization constant and the Euclidean distance, respectively.

Note that generating one-dimensional kernel density visualization for the edge $\widehat{e} = (x,y)$ (cf. Figure 3) is the special case of the above problem based on the following settings.

- Set the x-coordinate and y-coordinate of $\mathbf{p}_i$ to be $d_G(x,p_i)$ in our problem and 0, respectively.
- Set X to be $|L(\widehat{e})|$ ($|L(\widehat{e})|$ denotes the number of lixels in $\widehat{e}$.).
- Set Y to be 1.
- Set m to be $|P(\widehat{e})|$.

As such, computing network kernel density function values for all lixels q in $\widehat{e}$ is $O(|L(\widehat{e})|+|P(\widehat{e})|)$ time. Since there are $|E|$ edges in the road network, we conclude that the time complexity for generating

NKDV in all edges $\widehat{e}$ based on the network kernel density function $\mathcal{F}_{P(\widehat{e})}(q)$ is

$$O\Big(\sum_{\widehat{e}\in E}(|L(\widehat{e})|+|P(\widehat{e})|)\Big)=O(n+L)$$

□

Consider the second component in Equation 7, which depends on those data points that are outside the edge $\widehat{e}=(x,y)$. In Figure 4, note that each lixel q in $\widehat{e}$ can have two possible routes $q\rightarrow x\rightarrow p$ (blue dashed line) and $q\rightarrow y\rightarrow p$ (pink dashed line) to reach the data point p. Moreover, only those data points p with $d_G(q,p)\le b$ can contribute to the second component of network kernel density function $\mathcal{F}_{\widetilde{P}}(q)$ (cf. Equation 7). Based on these two concepts, we consider two sets of data points $R_x(q)$ (cf. Equation 9) and $R_y(q)$ (cf. Equation 10), which cover those data points such that the shortest path from q to each of these data points p only passes through the node x (i.e., the blue dashed line in Figure 4) and the node y (i.e., the pink dashed line in Figure 4), respectively, and its shortest path distance must be within the range b (i.e., $d_G(q,p)\le b$).

$$R_x(q)=\{p\in\widetilde{P}: d_G(q,x)+d_G(x,p)\le \min(d_G(q,y)+d_G(y,p),b)\} \quad (9)$$

$$R_y(q)=\{p\in\widetilde{P}: d_G(q,y)+d_G(y,p)\le \min(d_G(q,x)+d_G(x,p),b)\} \quad (10)$$

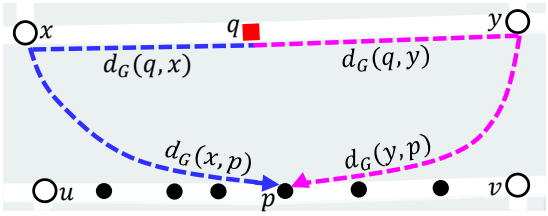


Figure 4: There are two possible routes (i.e., blue dashed line and pink dashed line) from the lixel q in the edge $\widehat{e}=(x,y)$ to the data point p in the edge $e=(u,v)$.

With these two sets of data points, the second component of network kernel density function $\mathcal{F}_{\widetilde{P}}(q)$ can be expressed as follows.

$$\begin{aligned}\mathcal{F}_{\widetilde{P}}(q) &= \sum_{p\in R_x(q)} w\cdot\Big(1-\frac{1}{b^2}(d_G(q,x)+d_G(x,p))^2\Big)\\ &+\sum_{p\in R_y(q)} w\cdot\Big(1-\frac{1}{b^2}(d_G(q,y)+d_G(y,p))^2\Big)\\ &= w\Big(1-\frac{1}{b^2}d_G(q,x)^2\Big)\alpha^{(0)}_{R_x(q)}-\frac{2w}{b^2}d_G(q,x)\alpha^{(1)}_{R_x(q)}-\frac{w}{b^2}\alpha^{(2)}_{R_x(q)}\\ &+ w\Big(1-\frac{1}{b^2}d_G(q,y)^2\Big)\alpha^{(0)}_{R_y(q)}-\frac{2w}{b^2}d_G(q,y)\alpha^{(1)}_{R_y(q)}-\frac{w}{b^2}\alpha^{(2)}_{R_y(q)}\end{aligned}$$

where $\alpha^{(deg)}_{R_x(q)}$ and $\alpha^{(deg)}_{R_y(q)}$ $(deg=0,1,2)$ are the aggregate terms for the lixel q.

$$\alpha^{(deg)}_{R_x(q)}=\sum_{p\in R_x(q)} d_G(x,p)^{deg} \quad\text{and}\quad \alpha^{(deg)}_{R_y(q)}=\sum_{p\in R_y(q)} d_G(y,p)^{deg} \quad (11)$$

Therefore, once we can efficiently maintain the aggregate terms, we can also efficiently evaluate $\mathcal{F}_{\widetilde{P}}(q)$. In this paper, we focus on generating NKDV using $\mathcal{F}_{\widetilde{P}}(q)$, which is the more challenging case.

3.2 Influence Regions of a Data Point

Observe from Figure 4 that we can compute the shortest path distances $d_G(x,p)$ (cf. Equation 12) and $d_G(y,p)$ (cf. Equation 13) in $O(1)$ time if we have obtained the shortest path distances (e.g., $d_G(x,u)$, $d_G(x,v)$, $d_G(y,u)$, and $d_G(y,v)$) from the node x and the node y to other nodes (e.g., node u and node v) in G.

$$d_G(x,p) = \min\begin{cases}d_G(x,u)+d_G(u,p)\\ d_G(x,v)+d_G(v,p)\end{cases} \quad (12)$$

$$d_G(y,p) = \min\begin{cases}d_G(y,u)+d_G(u,p)\\ d_G(y,v)+d_G(v,p)\end{cases} \quad (13)$$

With these shortest path distances $d_G(x,p)$ and $d_G(y,p)$, we can use $O(1)$ time to obtain the influence regions of this data point p from the node x and the node y in the edge $\widehat{e}=(x,y)$ (cf. Definition 2).

Definition 2. *Given a road network $G=(V,E)$, two edges $\widehat{e}=(x,y)$ and $e=(u,v)$, and a data point p in the edge e, the influence region of the data point p from the node x (node y) denotes those lixels q where each of the corresponding sets $R_x(q)$ ($R_y(q)$) covers the data point p, i.e., $p\in R_x(q)$ ($p\in R_y(q)$).*

Figure 5 shows the influence regions of the data point p from the node x (i.e., the blue lixels) and the node y (i.e., the pink lixels).

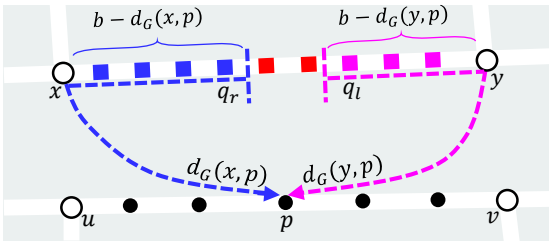


Figure 5: The blue and pink lixels in $\widehat{e}=(x,y)$ represent the influence regions of the data point p from the node x and the node y, respectively.

Based on the above concept, there are four possible cases for the influence regions of the data point p from the node x and the node y.

Case 1 ($d_G(x,p)>b$ and $d_G(y,p)>b$): Since we have $d_G(q,p)>\min(d_G(x,p),d_G(y,p))>b$ (cf. Figure 4) in this case, none of the lixels q in the edge $\widehat{e}=(x,y)$ can be influenced by the data point p.

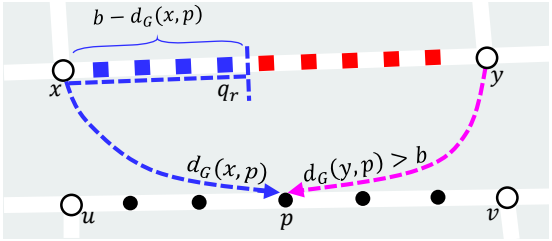


Figure 6: The blue lixels in $\widehat{e}=(x,y)$ represent the influence region of the data point p from the node x and there is no influence region from the node y.

Case 2 ($d_G(x,p)\le b$ and $d_G(y,p)>b$): In this case, only those lixels q in $\widehat{e}=(x,y)$ where the shortest path from each q to p passes

through the node x (cf. the blue dashed curve in Figure 6) can be possibly influenced by the data point p. Since each lixel has the regular size ℓ (cf. Figure 1b), we can use $O(1)$ time to identify the rightmost lixel q_r (i.e., $d_G(q_r, x)$ is the largest that is smaller than or equal to $b - d_G(x, p)$), and thus the influence region from x (i.e., the blue lixels in the edge $\widehat{e} = (x, y)$ in Figure 6).

Case 3 ($d_G(x, p) > b$ and $d_G(y, p) \leq b$): This case is similar to Case 2. Observe from Figure 7 that we can use $O(1)$ time to find the leftmost lixel q_l (i.e., $d_G(q_l, y)$ is the largest that is smaller than or equal to $b - d_G(y, p)$) and its corresponding influence region from the node y (i.e., the pink lixels in the edge $\widehat{e} = (x, y)$).

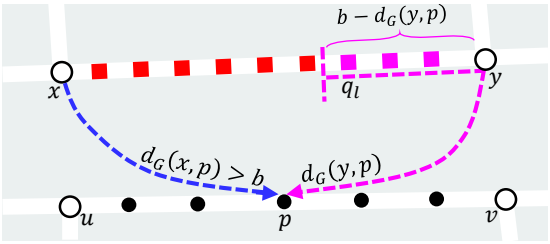


Figure 7: The pink lixels in $\widehat{e} = (x, y)$ represent the influence region of the data point p from the node y and there is no influence region from the node x.

Case 4 ($d_G(x, p) \leq b$ and $d_G(y, p) \leq b$): In this case, the data point p can influence the lixels in the edge $\widehat{e} = (x, y)$ from both the node x and the node y, which can be further classified into the following two cases (i.e., Case 4a and Case 4b).

In Case 4a (i.e., $2b - d_G(x, p) - d_G(y, p) < d_G(x, y)$), observe from Figure 5 that both the blue dashed line and pink dashed line in the edge $\widehat{e} = (x, y)$ do not intersect with each other. Like Case 2 and Case 3, we can use $O(1)$ time to find q_r and q_l, respectively, and identify the corresponding influence regions (i.e., the blue and pink lixels in Figure 5).

In Case 4b (i.e., $2b - d_G(x, p) - d_G(y, p) \geq d_G(x, y)$), note that the blue dashed line and the pink dashed line in the edge $\widehat{e} = (x, y)$ intersect with each other (cf. Figure 8). To correctly identify the influence regions from the node x and the node y (cf. Definition 2), we need to find the rightmost lixel q_r (i.e., with the largest distance $d_G(q_r, x)$) from the node x so that

$$\begin{aligned} d_G(q_r, x) + d_G(x, p) &\leq d_G(q_r, y) + d_G(y, p) \\ d_G(q_r, x) + d_G(x, p) &\leq (d_G(x, y) - d_G(q_r, x)) + d_G(y, p) \\ d_G(q_r, x) &\leq \frac{d_G(x, y) - d_G(x, p) + d_G(y, p)}{2} \end{aligned}$$

which can be computed in $O(1)$ time. With this q_r, we can obtain the right-neighbor lixel q_l (cf. Figure 8). As such, we can further obtain the two influence regions from the node x and the node y of the data point p in $O(1)$ time.

Based on these four cases, we have the following two observations, which are:

- If the lixel q is close to the node x compared with the lixel q_ρ, this lixel q is easily influenced by any data point p (in other edges) from the node x (i.e., $R_x(q_\rho) \subseteq R_x(q)$).
- If the lixel q is close to the node y compared with the lixel q_λ, this lixel q is easily influenced by any data point p (in other edges) from the node y (i.e., $R_y(q_\lambda) \subseteq R_y(q)$).

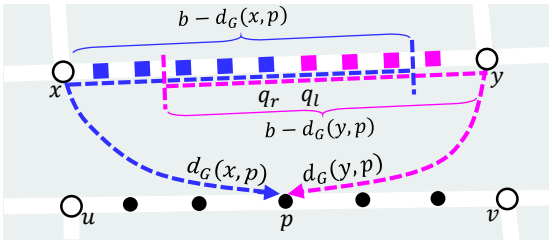


Figure 8: The blue lixels and the pink lixels represent the influence regions of the data point p from the node x and the node y, respectively.

In Lemma 2, we show that the above observations are true.

Lemma 2. *Consider the edge $\widehat{e} = (x, y)$ in a road network $G = (V, E)$ and three lixels q, q_λ, and q_ρ in the edge $\widehat{e}$. If $d_G(q_\lambda, x) \leq d_G(q, x) \leq d_G(q_\rho, x)$, we have*

$$R_x(q_\rho) \subseteq R_x(q) \tag{14}$$

$$R_y(q_\lambda) \subseteq R_y(q) \tag{15}$$

Proof. In this proof, we focus on showing $R_x(q_\rho) \subseteq R_x(q)$. Similar ideas can be adopted for proving $R_y(q_\lambda) \subseteq R_y(q)$.

Suppose that $p \in R_x(q_\rho)$, we have (cf. Equation 9)

$$d_G(q_\rho, x) + d_G(x, p) \leq \min(d_G(q_\rho, y) + d_G(y, p), b)$$

which implies:

$$d_G(q_\rho, x) + d_G(x, p) \leq d_G(q_\rho, y) + d_G(y, p) \tag{16}$$

and

$$d_G(q_\rho, x) + d_G(x, p) \leq b \tag{17}$$

Consider the left part of Inequality 16. We have

$$\begin{aligned} & d_G(q_\rho, x) + d_G(x, p) \\ \geq\ & d_G(q, x) + d_G(x, p) \text{ (since } d_G(q, x) \leq d_G(q_\rho, x)) \end{aligned} \tag{18}$$

Consider the right part of Inequality 16. We have

$$\begin{aligned} & d_G(q_\rho, y) + d_G(y, p) \\ =\ & (d_G(x, y) - d_G(q_\rho, x)) + d_G(y, p) \\ \leq\ & (d_G(x, y) - d_G(q, x)) + d_G(y, p) \text{ (since } d_G(q, x) \leq d_G(q_\rho, x)) \\ =\ & d_G(q, y) + d_G(y, p) \end{aligned} \tag{19}$$

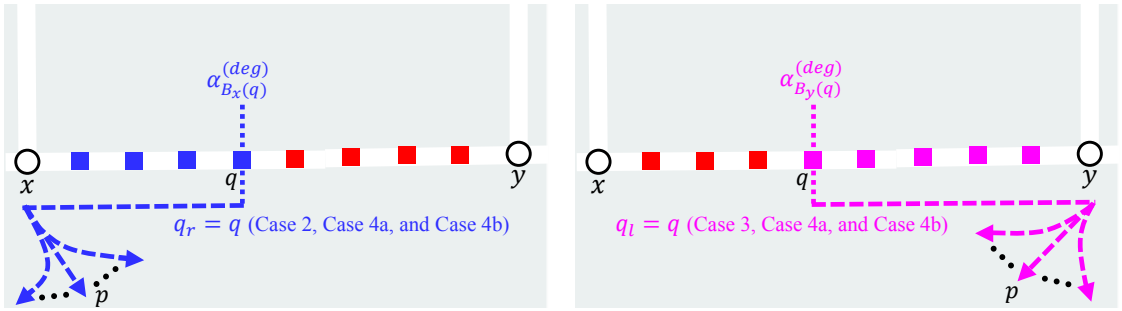
Based on Inequalities 18, 19, and 16, we have $d_G(q, x) + d_G(x, p) \leq d_G(q, y) + d_G(y, p)$. In addition, we also have $d_G(q, x) + d_G(x, p) \leq b$ (based on Inequalities 18 and 17). Therefore, $d_G(q, x) + d_G(x, p) \leq \min(d_G(q, y) + d_G(y, p), b)$, which indicates $p \in R_x(q) \Longrightarrow R_x(q_\rho) \subseteq R_x(q)$. □

3.3 LION: An Augmentation and Aggregation Approach

To boost the efficiency for generating NKDV using $\mathcal{F}_{\widetilde{P}}(q)$, we propose the solution, called LION, which consists of two phases, namely (1) LIxel augmentatiON and (2) LIxel aggregatiON.

Lixel augmentation. In the first phase, we aim to augment the aggregate terms, $\alpha^{(deg)}_{B_x(q)}$ and $\alpha^{(deg)}_{B_y(q)}$ ($deg = 0, 1, 2$), for each lixel q (cf. Equation 20 and Figure 9).

$$\alpha^{(deg)}_{B_x(q)} = \sum_{p \in B_x(q)} d_G(x, p)^{deg} \text{ and } \alpha^{(deg)}_{B_y(q)} = \sum_{p \in B_y(q)} d_G(y, p)^{deg} \tag{20}$$



(a) Influence region from the node x (b) Influence region from the node y

Figure 9: Augment the aggregate terms, $\alpha^{(deg)}_{B_x(q)}$ ($\alpha^{(deg)}_{B_y(q)}$), where $deg = 0, 1, 2$, for the rightmost (leftmost) lixel q.

where $B_x(q)$ and $B_y(q)$ store all data points p which their influence regions cover q as the rightmost lixel (with the largest $d_G(q, x)$) and the leftmost lixel (with the largest $d_G(q, y)$), respectively.

Algorithm 1 Lixel Augmentation

1: **procedure** AUGMENTATION($G = (V, E)$, P, b)
2: **for** each edge $\widehat{e} = (x, y) \in E$ **do**
3: **for** each lixel q in the edge $\widehat{e}$ **do** ▷ Initialization
4: $\alpha^{(deg)}_{B_x(q)} \leftarrow 0$ ($deg = 0, 1, 2$)
5: $\alpha^{(deg)}_{B_y(q)} \leftarrow 0$ ($deg = 0, 1, 2$)
6: Find $SPD(x)$ and $SPD(y)$, where (for each $h \in V$)

$$SPD(x).h = \begin{cases} d_G(x,h) & \text{if } d_G(x,h) \le b \\ \infty & \text{otherwise} \end{cases} \quad (21)$$

7: **for** each edge $e = (u, v) \in E \backslash \widehat{e}$ **do**
8: **for** each $p \in P(e)$ **do**
9: Compute $d_G(x, p)$ ▷ Equation 12, $O(1)$ time
10: Compute $d_G(y, p)$ ▷ Equation 13, $O(1)$ time
11: //Core idea 2 (cf. Section 3.2)
12: **if** Case 2 is true **then**
13: Obtain q_r
14: Update $\alpha^{(deg)}_{B_x(q_r)}$ ($deg = 0, 1, 2$)
15: **if** Case 3 is true **then**
16: Obtain q_l
17: Update $\alpha^{(deg)}_{B_y(q_l)}$ ($deg = 0, 1, 2$)
18: **if** Case 4 is true **then**
19: Obtain q_l and q_r
20: Update $\alpha^{(deg)}_{B_x(q_r)}$ ($deg = 0, 1, 2$)
21: Update $\alpha^{(deg)}_{B_y(q_l)}$ ($deg = 0, 1, 2$)

Algorithm 1 shows how we obtain all these aggregate terms (cf. Equation 20 and Figure 9) in each lixel q. Consider an edge $\widehat{e} = (x, y)$ in this algorithm (line 2). We first initialize these aggregate terms for all lixels q to be 0 (lines 3 to 5) and then compute the single-source shortest path distances from the node x and the node y to other nodes in V (cf. Equation 21 in line 6), which take $O(|L(e)| + T_{SP})$ time, where $|L(e)|$ and T_{SP} denote the number of lixels in the edge e and the time complexity of the shortest path algorithm, respectively. Based on these shortest path distances from the node x and the node y and different cases of the core idea 2 (cf. Section 3.2), we can scan each data point p in other edges $e = (u, v)$ and use $O(1)$ time to obtain the rightmost lixel q_r and the leftmost lixel q_l (cf. Figure 9). With these q_r and q_l, we can update the aggregate terms by the following approach:

$$\alpha^{(deg)}_{B_x(q_r)} \leftarrow \alpha^{(deg)}_{B_x(q_r)} + d_G(x,p)^{deg}$$
$$\alpha^{(deg)}_{B_y(q_l)} \leftarrow \alpha^{(deg)}_{B_y(q_l)} + d_G(y,p)^{deg}$$

As such, the time complexity of line 7 to line 21 is $O(n + |E|)$. In Lemma 3, we show that the time complexity of Algorithm 1 is $O(|E|T_{SP} + n|E| + |E|^2 + L)$.

LEMMA 3. *The time complexity of Algorithm 1 is* $O(|E|T_{SP} + n|E| + |E|^2 + L)$.

PROOF. Since line 3 to line 6 take $O(|L(e)| + T_{SP})$ and line 7 to line 21 take $O(n + |E|)$, the time complexity of one iteration in line 2 is $O(|L(e)| + T_{SP} + n + |E|)$. As such, the time complexity of Algorithm 1 is

$$\sum_{e \in E} O(|L(e)| + T_{SP} + n + |E|) = O(|E|T_{SP} + n|E| + |E|^2 + L)$$

□

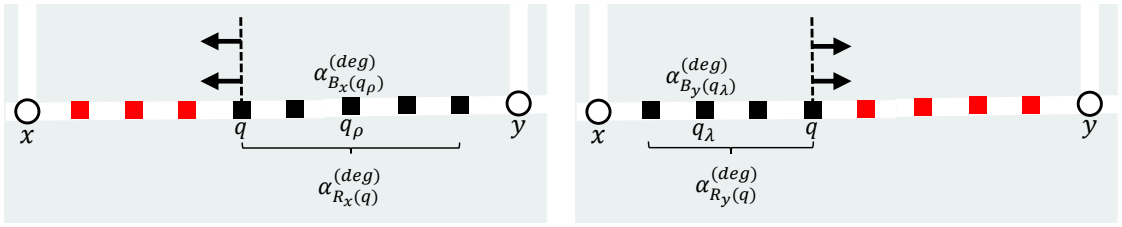
Lixel aggregation. In the second phase, we aim to efficiently compute $\alpha^{(deg)}_{R_x(q)}$ and $\alpha^{(deg)}_{R_y(q)}$ (cf. Equation 11) based on the aggregate terms in the first phase (cf. Figure 9) so that we can efficiently evaluate $\mathcal{F}_P(q)$ for all lixels q. Here, we claim in Lemma 4 that only $B_x(q_\rho)$ with $d_G(q, x) \le d_G(q_\rho, x)$ and only $B_y(q_\lambda)$ with $d_G(q_\lambda, x) \le d_G(q, x)$ can contribute to $R_x(q)$ and $R_y(q)$, respectively (cf. the black lixels in Figure 10).

LEMMA 4. *Given an edge $\widehat{e} = (x, y)$ and the lixel q in the edge $\widehat{e}$, we have*

$$\alpha^{(deg)}_{R_x(q)} = \sum_{q_\rho : d_G(q_\rho, x) \ge d_G(q,x)} \alpha^{(deg)}_{B_x(q_\rho)} \quad (22)$$

$$\alpha^{(deg)}_{R_y(q)} = \sum_{q_\lambda : d_G(q_\lambda, x) \le d_G(q,x)} \alpha^{(deg)}_{B_y(q_\lambda)} \quad (23)$$

PROOF. In this proof, we focus on showing Equation 22. By adopting the same concept, we can also prove Equation 23.



(a) Compute $\alpha^{(deg)}_{R_x(q)}$

(b) Compute $\alpha^{(deg)}_{R_y(q)}$

Figure 10: Compute the aggregate terms, $\alpha^{(deg)}_{R_x(q)}$ and $\alpha^{(deg)}_{R_y(q)}$ (based on $\alpha^{(deg)}_{B_x(q_\rho)}$ ($d_G(q,x) \le d_G(q_\rho,x)$) and $\alpha^{(deg)}_{B_y(q_\lambda)}$ ($d_G(q_\lambda,x) \le d_G(q,x)$), respectively), in order to evaluate $\mathcal{F}_{\widehat{P}}(q)$, by scanning all lixels q two times in each edge $\widehat{e} = (x,y)$.

Consider any lixels q_λ, q, and q_ρ, where $d_G(q_\lambda,x) \le d_G(q,x) \le d_G(q_\rho,x)$. Note that $B_x(q_\lambda)$ denotes the set of data points where the rightmost lixel of the influence region of each data point p from the node x is q_λ. As such, the influence region cannot cover the lixel q. Hence, we have

$$\bigcup_{q_\lambda : d_G(q_\lambda,x) \le d_G(q,x)} B_x(q_\lambda) \not\subseteq R_x(q) \tag{24}$$

Since $B_x(q_\rho) \subseteq R_x(q_\rho) \subseteq R_x(q)$ (based on Lemma 2) and Equation 24 holds, we can further conclude that

$$\bigcup_{q_\rho : d_G(q_\rho,x) \ge d_G(q,x)} B_x(q_\rho) = R_x(q) \tag{25}$$

Based on Equation 25, we prove that Equation 22 is correct. □

As such, we only need to scan all lixels q in each edge $\widehat{e} = (x,y)$ two times (from y to x in Figure 10a and from x to y in Figure 10b) to incrementally compute the aggregate terms $\alpha^{(deg)}_{R_x(q)}$ (cf. Equation 22) and $\alpha^{(deg)}_{R_y(q)}$ (cf. Equation 23), which takes $O(|L(e)|)$ time. Based on the core idea 1 in Section 3.1, we can also compute $\mathcal{F}_{\widehat{P}}(q)$ for all lixels q in the edge $\widehat{e} = (x,y)$ in $O(|L(e)|)$ time. Algorithm 2 shows how to compute $\mathcal{F}_{\widehat{P}}(q)$ for all lixels in a road network $G = (V,E)$.

Algorithm 2 Lixel Aggregation

```
1:  procedure AGGREGATION(G = (V, E))
2:      //deg = 0, 1, 2 for the Epanechnikov kernel.
3:      for each edge ê = (x, y) ∈ E do
4:          α^(deg)_left ← 0 and α^(deg)_right ← 0
5:          for each lixel q from node y to node x do
6:              α^(deg)_left ← α^(deg)_left + α^(deg)_{B_x(q)}
7:              α^(deg)_{R_x(q)} ← α^(deg)_left
8:          for each lixel q from node x to node y do
9:              α^(deg)_right ← α^(deg)_right + α^(deg)_{B_y(q)}
10:             α^(deg)_{R_y(q)} ← α^(deg)_right
11:         for each lixel q in ê do
12:             Compute F_P̂(q)                          ▷ Section 3.1
```

In Lemma 5, we show that the time complexity of Algorithm 2 is $O(L)$.

Lemma 5. *The time complexity of Algorithm 2 is $O(L)$.*

Proof. Note that the internal loops (i.e., line 5, line 8, and line 11) of Algorithm 2 only take $O(|L(e)|)$ time for computing the aggregate terms and evaluating $\mathcal{F}_{\widehat{P}}(q)$. Therefore, the time complexity of this algorithm (cf. line 3) is $\sum_{e \in E} O(|L(e)|) = O(L)$. □

LION. Since Algorithm 1 and Algorithm 2 take $O(|E|T_{\text{SP}} + n|E| + |E|^2 + L)$ (cf. Lemma 3) and $O(L)$ time (cf. Lemma 5), respectively, we conclude that the time complexity of LION is $O(|E|T_{\text{SP}} + n|E| + |E|^2 + L)$ (cf. Theorem 2). We omit the proof of this theorem due to its simplicity.

Theorem 2. *The time complexity of LION is $O(|E|T_{SP} + n|E| + |E|^2 + L)$.*

Compared with the state-of-the-art solution, ADA, which takes $O(|E|T_{\text{SP}} + L|E| \log(\frac{n}{|E|}))$ time, our solution, LION, theoretically lowers the worst-case time complexity if the number of lixels L is larger than the number of location data points n (i.e., $L > n$).

3.4 Space Complexity of LION

In this section, we further investigate the space complexity of LION. Since every algorithm needs to adopt the shortest path algorithm and access the road network $G = (V,E)$, all lixels in G, and the location dataset P, LION takes at least $O(|V| + |E| + n + L + S_{\text{SP}})$ space (where S_{SP} denotes the space complexity of the shortest path algorithm). In addition, LION also needs to augment the aggregate terms $\alpha^{(deg)}_{B_x(q)}$ and $\alpha^{(deg)}_{B_y(q)}$ (cf. Equation 20) and evaluate $\alpha^{(deg)}_{R_x(q)}$ and $\alpha^{(deg)}_{R_y(q)}$ (cf. Equation 22 and Equation 23, respectively) for all lixels q, which takes $O(L)$ additional space. As such, we conclude that LION takes $O(|V| + |E| + n + L + S_{\text{SP}})$ space for generating NKDV (cf. Theorem 3).

Theorem 3. *The space complexity of LION is $O(|V| + |E| + n + L + S_{SP})$.*

4 EXPERIMENTAL EVALUATION

In this section, we first discuss the experiment settings in Section 4.1. Then, we compare the time and space efficiency of different methods in Section 4.2. Next, we conduct the efficiency experiments for other kernels, including triangular kernel and quartic kernel (cf. Table 1), in Section 4.3. Lastly, we conduct two case studies, which are (1) using NKDV to analyze hotspots with different attribute values in two location datasets, namely London traffic accident dataset

and Detroit 911-call dataset, and (2) investigating the visualization quality of NKDVs in the Chicago traffic accident dataset with respect to different lixel sizes, in Section 4.4. Due to space limitations, some additional experiments can be found in the technical report of this paper (cf. Appendix in [31]).

4.1 Experiment Settings

We adopt four large-scale location datasets with three categories, namely crime events, traffic accidents, and 911 calls, for conducting our experiments (cf. Table 3). For each location dataset, we first extract the corresponding road network and then map each location data point into it using the famous OSMNx python library [14].

Table 3: Datasets.

Dataset	$\|V\|$	$\|E\|$	n	Category	Ref.
Gainesville	5,352	7,522	193,795	Crime events	[4]
Seattle	12,030	20,369	241,599	Traffic accidents	[7]
Chicago	40,428	69,219	719,372	Traffic accidents	[2]
Detroit	57,029	92,646	1,931,000	911 calls	[3]

In our experiments, we compare our method, LION, with different NKDV methods, which are summarized in Table 4. Range-query-based solution (RQS) [54, 75] computes the network kernel density function $\mathcal{F}_P(q)$ for each lixel q based on the range query set which covers all data points that are within the bandwidth b from q. Shortest path sharing solution (SPS) [61] improves the efficiency of generating NKDV by sharing the shortest path distances. Aggregate distance augmentation (ADA) [23] is the state-of-the-art method for generating NKDV. We follow [23] and choose the default bandwidth parameter b (cf. Table 1) to be 1000m. In addition, since domain experts [40, 75, 76] normally adopt 10m as the lixel size ℓ (cf. Figure 1), we follow the same setting in this paper. As a remark, the numbers of lixels with the default lixel size $\ell = 10$m for all datasets, Gainesville, Seattle, Chicago, and Detroit, are 208,548, 398,134, 1,184,187, and 2,563,436, respectively, which are larger than the corresponding numbers of data points n (cf. Table 3).

Table 4: Comparisons of different methods for generating NKDV.

Method	RQS	SPS	ADA	LION
Ref.	[54, 75]	[61]	[23] (cf. Section 2.2)	Section 3

We adopted the C++ implementation of existing methods[2] in [23] (i.e., RQS, SPS, and ADA in Table 4), implemented our method using C++[3], and conducted experiments on an Intel i7 3.19GHz PC with 32GB memory. In this paper, we report the response time (sec) and the memory space (MB)[4] for testing the time efficiency and space efficiency of each method, respectively, and omit the results of response time that are more than 14,400 sec (i.e., 4 hours).

4.2 Efficiency of Generating NKDV

In this section, we compare both the time efficiency and space efficiency of our method, LION, with different methods in Table 4 by conducting the following experiments.

[2]The implementation of existing methods can be found in the Github link https://github.com/edisonchan2013928/Network-Kernel-Density-Visualization-NKDV-Code.
[3]The implementation of our method, LION, can be found in the Github link https://github.com/edisonchan2013928/LION.
[4]We adopt the standard C++ function, getrusage(), for measuring the space consumption of each method.

Response time of all methods with different lixel sizes. In this experiment, we vary the lixel size (ℓ in Figure 1) from 20m to 1m and test the response time of different methods. In Figure 11, once we reduce the lixel size ℓ (i.e., increase the number of lixels in the road networks), each method needs to compute more density values, which results in higher response time. With the lower time complexity for generating NKDV, both ADA and LION achieve better efficiency performance compared with the RQS and SPS methods. Since LION is more scalable to the number of lixels L (i.e., small lixel size ℓ), this method can achieve 2.86x to 34.55x speedup compared with the state-of-the-art ADA method.

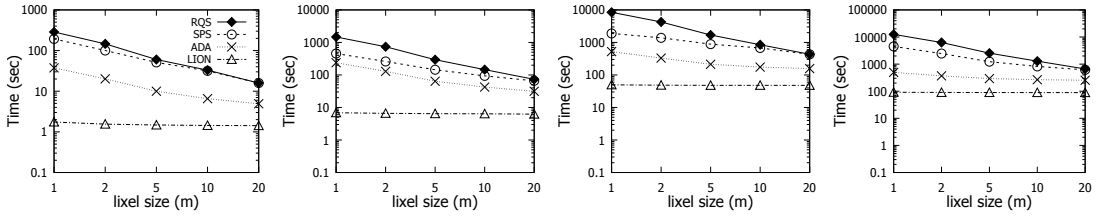
Response time of all methods with different dataset sizes. To conduct this experiment, we first randomly sample each dataset with different percentages, 25%, 50%, 75%, and 100% (original one), and then measure the response time of all methods in these reduced datasets. Figure 12 shows the results of all methods. Since ADA and LION have the lower time complexity for generating NKDV, these two methods have the smaller response time compared with the RQS and SPS methods. In practice, our method, LION, further achieves 3x to 6.73x speedup compared with the existing ADA method.

Response time of all methods with different bandwidth parameters b. Here, we further investigate how the bandwidth parameter b affects the efficiency of all methods. In Figure 13, observe that all methods take larger response time for generating NKDV once we increase the bandwidth parameter b. The main reason is that each method needs to process more data points with the larger bandwidth b. Note that the best method, LION, can further achieve up to 13.03x speedup compared with the state-of-the-art ADA method.

Space consumption of all methods with different dataset sizes. We proceed to investigate how the dataset size can affect the memory space consumption of each method. To conduct this experiment, we first randomly sample each dataset with four ratios, i.e., 25%, 50%, 75%, and 100% (original one), and then measure the memory space consumption of all methods for each reduced dataset. Figure 14 shows the results of all methods. Since both ADA and LION need to augment additional information (e.g., the aggregate terms in Equation 20) in a road network, these two methods need to consume larger memory space compared with the RQS and SPS methods. Recall that ADA and LION augment $2 \times deg$ aggregate terms for each data point (cf. Equation 5 and Equation 6) and each lixel (cf. Equation 20), respectively, in a road network and the number of lixels is larger than the number of data points (i.e., $L > n$). Therefore, once we vary the dataset size for each dataset, ADA consumes smaller memory space (with smaller number of aggregate terms in total) compared with LION in practice. Despite this, LION does not incur high space overhead for generating NKDV (i.e., retains in MB level) since this method has the same space complexity (cf. Theorem 3) compared with other methods (cf. Table 2).

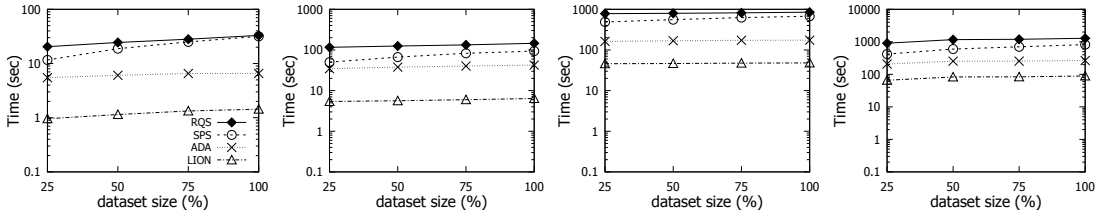
4.3 Other Kernels

In this section, we proceed to investigate the efficiency performance of all methods for generating NKDV using other kernel functions in Table 1, including triangular kernel and quartic kernel. Due to space limitations, we only choose the Seattle and Chicago datasets for testing.



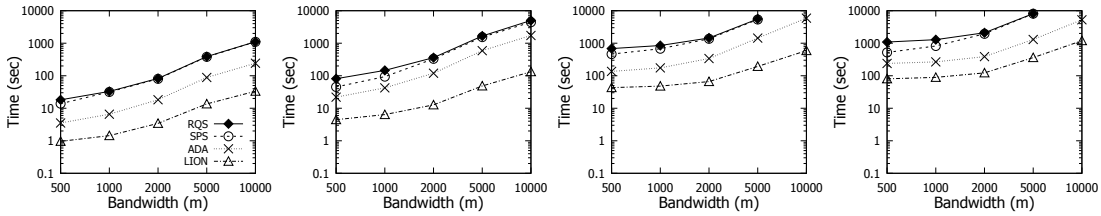
(a) Gainesville (b) Seattle (c) Chicago (d) Detroit

Figure 11: Response time for generating NKDV, varying the lixel size.



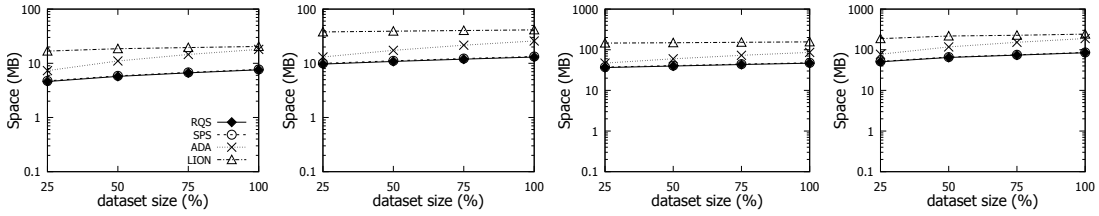
(a) Gainesville (b) Seattle (c) Chicago (d) Detroit

Figure 12: Response time for generating NKDV, varying the dataset size.



(a) Gainesville (b) Seattle (c) Chicago (d) Detroit

Figure 13: Response time for generating NKDV, varying the bandwidth parameter b.



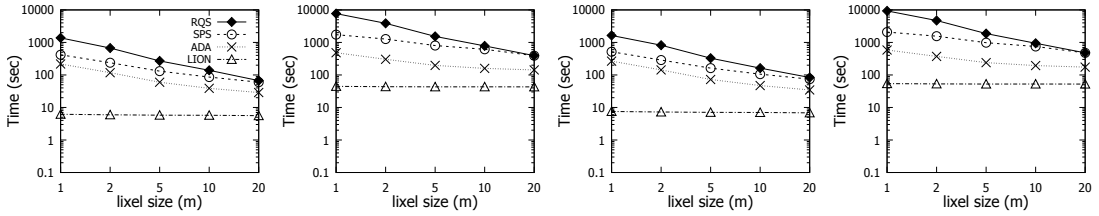
(a) Gainesville (b) Seattle (c) Chicago (d) Detroit

Figure 14: Memory space consumption (MB) for generating NKDV, varying the dataset size.

Response time of all methods with different lixel sizes. In this experiment, we first choose five lixel sizes, which are 1m, 2m, 5m, 10m, and 20m, and test the response time of each method with these lixel sizes using the default bandwidth value $b = 1000$m. In Figure 15, note that our method, LION, achieves 3.32x to 35.36x speedup compared with the state-of-the-art ADA method. Like the previous experiment in Section 4.2, all methods have the similar trends of response time (e.g., Figure 15a and Figure 15c) compared with the Epanechnikov kernel function (e.g., Figure 11b).

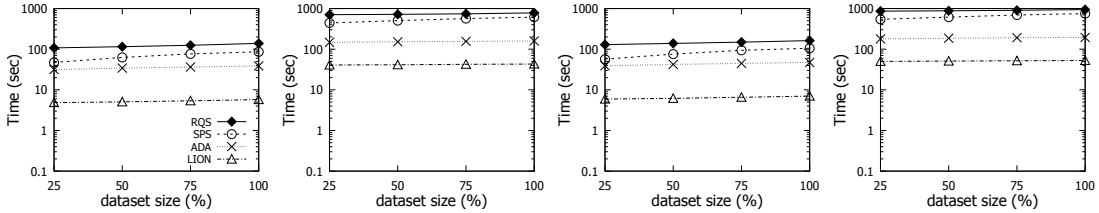
Response time of all methods with different dataset sizes. We proceed to examine how the dataset size affects the response time of each method. Here, we follow the same settings in Section 4.2 to measure the response time of each method for these reduced datasets with different sampling percentages, which are 25%, 50%, 75%, and 100% (original one), using the same bandwidth value $b = 1000$m. In Figure 16, observe that our method, LION, can achieve 3.61x to 6.83x speedup compared with the existing method, ADA, regardless of which kernel function we choose.

Space consumption of all methods with different dataset sizes. Here, we further investigate the memory space consumption of all methods for triangular kernel and quartic kernel with respect to different dataset sizes (i.e., sampling percentages). Figure 17 shows the results of all methods. Like the previous experiments in Figure 14, both ADA and LION need to consume higher memory space compared with RQS and SPS no matter which kernel functions we adopt. Despite this, LION does not significantly incur huge space overhead due to the fact that this method does not increase the worst-case space complexity for generating NKDV.



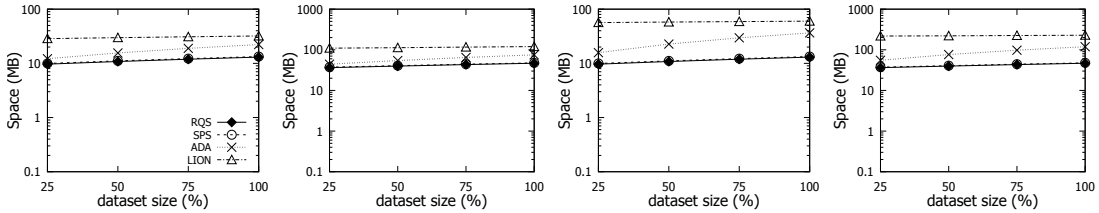
(a) Seattle (Triangular) (b) Chicago (Triangular) (c) Seattle (Quartic) (d) Chicago (Quartic)

Figure 15: Response time for generating NKDV in two datasets, which are Seattle ((a) and (c)) and Chicago ((b) and (d)), using the triangular kernel ((a) and (b)) and the quartic kernel ((c) and (d)), with different lixel sizes.



(a) Seattle (Triangular) (b) Chicago (Triangular) (c) Seattle (Quartic) (d) Chicago (Quartic)

Figure 16: Response time for generating NKDV in two datasets, which are Seattle ((a) and (c)) and Chicago ((b) and (d)), using the triangular kernel ((a) and (b)) and the quartic kernel ((c) and (d)), with different dataset sizes.



(a) Seattle (Triangular) (b) Chicago (Triangular) (c) Seattle (Quartic) (d) Chicago (Quartic)

Figure 17: Memory space consumption (MB) for generating NKDV in two datasets, which are Seattle ((a) and (c)) and Chicago ((b) and (d)), using the triangular kernel ((a) and (b)) and the quartic kernel ((c) and (d)), with different dataset sizes.

4.4 Case Studies

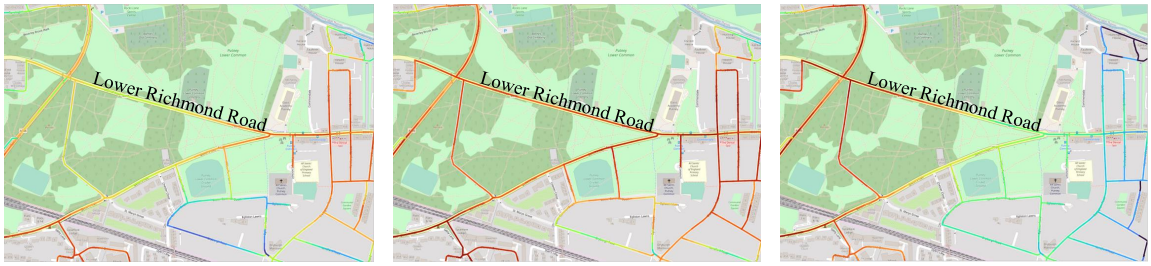
In this section, we further conduct two case studies, which are (1) NKDV-based exploratory analysis and (2) quality analysis of NKDV with different lixel sizes.

NKDV-based exploratory analysis. In the first case study, we aim to analyze hotspots with different attribute values with respect to two location datasets. As a remark, we set the bandwidth parameter b and the lixel size to be 1000m and 10m (i.e., the default values), respectively.

In the London traffic accident dataset [5], we first select the attribute "number of vehicles", which indicates the number of vehicles that are involved in each traffic accident event, and then choose three values for this attribute, which are 1, 2, and 3. For each attribute value (e.g., number of vehicles = 2), we generate NKDV for those traffic accident events that have the same attribute value. Figure 18 shows the visualization results with respect to different numbers of vehicles. Once we choose "one vehicle" and "three vehicles", the hotspots are mainly in the right part and the left part, respectively, of the Lower Richmond Road. In addition, the whole "Lower Richmond Road" is the hotspot region if we choose "two vehicles" for analysis.

In the Detroit 911-call dataset (cf. Table 3), we first select the attribute "priority", which indicates the priority for each 911 call, and then choose three values for this attribute, which are 1 (highest priority), 2 (middle priority), and 3 (smallest priority). We further generate NKDV for those data points with respect to each attribute value. Observe from Figure 19 that the roads that are near the Perrien Park have high density of 911 calls with the middle priority and the smallest priority (i.e., the hotspot regions), while they have low density of 911 calls with the highest priority (i.e., the coldspot regions). This phenomenon may indicate that these roads do not have many serious crime events.

These results show that different attribute values can significantly affect the interpretations for the geographic properties (e.g., is it a dangerous region or a safe region?) of a given location. Therefore, domain experts need to adopt the filtering operation to focus on some parts (e.g., with different attribute values) of a location dataset. In order to obtain better efficiency performance for generating multiple NKDVs with the filtering operation, we need to adopt our method, LION, which can further achieve 4.03x-4.74x speedup compared with the state-of-the-art ADA method (cf. Figure 20).

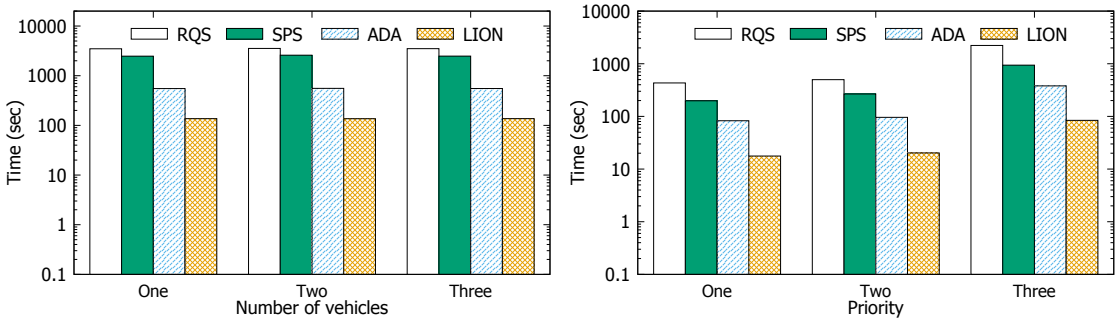


(a) One vehicle (b) Two vehicles (c) Three vehicles

Figure 18: Generate NKDVs for the London traffic accident dataset, where we zoom in to the Putney Lower Common region, by selecting different values of the attribute "number of vehicles".

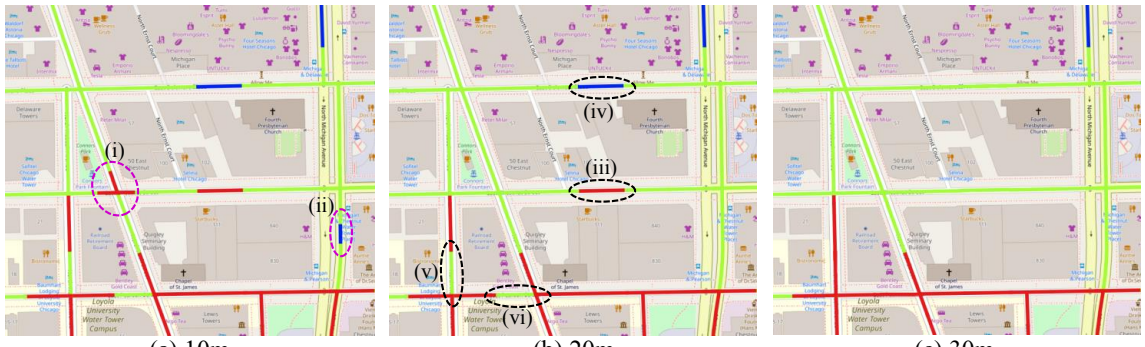
(a) 911 calls with the highest priority (b) 911 calls with the middle priority (c) 911 calls with the smallest priority

Figure 19: Generate NKDVs for the Detroit 911-call dataset, where we zoom in to the Perrien Park region, by selecting different values of the attribute "priority".



(a) London traffic accident dataset (b) Detroit 911-call dataset

Figure 20: Response time for generating NKDVs with different attribute values in London traffic accident dataset and Detroit 911-call dataset.



(a) 10m (b) 20m (c) 30m

Figure 21: Generate NKDVs for the Chicago traffic accident dataset with respect to three lixel sizes, namely 10m, 20m, and 30m, where we zoom in to the Old Chicago Water Tower District. The differences between (a) and (b) and the differences between (b) and (c) are highlighted with the pink dashed circles in (a) and the black dashed circles in (b), respectively.

Quality analysis of NKDVs with different lixel sizes. In the second case study, we aim to analyze the visualization quality of NKDVs with respect to different lixel sizes. Here, we adopt the Chicago traffic accident dataset (cf. Table 3) and the default bandwidth parameter $b = 1000$m for testing. Note that NKDV can be more sensitive to density variation, which can capture more detailed changes of a visualization, when we choose a small lixel size (i.e., a large number of lixels). Comparing Figure 21b with Figure 21c, NKDV with 20m as the lixel size can discover one additional hotspot in (iii), uncover one additional coldspot in (iv), and identify (v) and (vi) as non-hotspot regions. Once we further reduce the lixel size to be 10m (cf. Figure 21a), we can discover one additional hotspot in (i) and one additional coldspot in (ii).

5 RELATED WORK

Network Kernel Density Visualization (NKDV) has been widely used in many application domains, including crime hotspot detection [44, 64], traffic/traffic accident hotspot detection [15, 40], and urban planning [34, 78]. However, generating NKDV is time-consuming, which cannot be scalable to large-scale datasets (i.e., large n) and high resolution sizes (i.e., large L). In this section, we review three camps of research studies, namely (1) efficient algorithms for kernel density visualization and its variants, (2) efficient shortest path algorithms, and (3) efficient algorithms for spatial queries in a road network, which are closely related to this work.

Efficient algorithms for kernel density visualization and its variants. Recently, many research studies have been proposed to improve the efficiency for generating kernel density visualization (KDV) and its variants [18–25, 57–59, 81, 82]. Although all these research studies can successfully reduce the worst-case time complexity for solving these problems, most of them [18–22, 24, 25, 29, 57–59, 81, 82] focus on the Euclidean space, which cannot be extended for supporting our NKDV problem. Among most of these research studies, Chan et al. [23] propose the state-of-the-art solution, called aggregate distance augmentation (ADA), which can reduce the worst-case time complexity for generating NKDV. Despite this, this work does not consider the optimization opportunity of the lixel size. Therefore, unlike our LION method, the ADA method cannot be scalable to large L. However, domain experts may adopt some filtering operations for using NKDV to perform hotspot analysis (e.g., generating NKDV for those 911 calls with the highest priority in Figure 19a). Therefore, it is likely to have the case $L > n$ in practice, which is more suitable for using our LION method. As a remark, this is the first work that considers the new concept, called influence region of a data point (cf. Section 3.2), and analyzes its property (cf. Lemma 2), in order to develop the new LION method, which is technically different from the ADA method (based on augmenting the aggregate distance values for each data point in Figure 2).

Efficient shortest path algorithms. To generate NKDV, all existing methods (cf. Table 4) and our LION method also need to compute the shortest path distances from each node to all other nodes (e.g., line 6 in Algorithm 1 of our LION method). Although many types of shortest path algorithms, including shortest path caching [46, 72], hub labeling [11, 42, 48, 62], and hierarchical indexing [38, 84], have been proposed in the literature, most of them (e.g., shortest path caching algorithms and hub labeling algorithms) only boost the efficiency of solving single-source-single-destination queries, which cannot be used for improving the efficiency of NKDV. Furthermore, note that the main bottleneck of generating NKDV is to process the lixels and data points, using efficient shortest path algorithms cannot significantly boost the efficiency for generating NKDV. As a remark, since we can easily replace the shortest path algorithm for every NKDV method, developing efficient shortest path algorithms is orthogonal to this work.

Efficient algorithms for spatial queries in a road network. In spatial database communities, many researchers have proposed efficient algorithms for various queries in a road network, e.g., kNN queries [10, 41, 45, 55, 66, 69, 83], range queries [32, 49, 56, 70], skyline queries [35, 36, 51, 67], and keyword queries [9, 33, 37, 50, 63, 77, 80]. However, since all these research studies do not focus on the complex network kernel density function $\mathcal{F}_P(q)$ (cf. Equation 1), most of them cannot be directly extended to generate NKDV, not to mention reducing the worst-case time complexity of this operation.

6 CONCLUSION

In this paper, we study network kernel density visualization (NKDV), which has been extensively used in various geospatial analysis fields. However, NKDV is computationally expensive, which is time-consuming (or even infeasible) to support large-scale location datasets and high resolution sizes. Although a recent work [23] has proposed the new algorithm, called aggregate distance augmentation (ADA), to improve the efficiency for generating NKDV, this method is still slow and does not consider the optimization opportunity for another parameter, i.e., the number of lixels. However, in practice, it is possible for the number of lixels L to be larger than the number of data points n in a location dataset (e.g., domain experts can perform filtering operations for conducting exploratory analysis.). To tackle this issue, we develop the new algorithm, called LION, which can further reduce the worst-case time complexity for generating NKDV compared with the state-of-the-art ADA method if $L > n$. Our experiment results and case studies also verify that LION can achieve 2.86x to 35.36x speedup compared with existing methods and show the importance of conducting exploratory analysis, respectively.

In the future, we will develop the python library (like [30]), the QGIS plugin, and the ArcGIS plugin based on this method. Furthermore, we will investigate how to extend LION for supporting other types of interpolation tasks in a road network, e.g., network inverse distance weighting [54] and network kriging [54]. Moreover, we will also develop efficient algorithms for handling other GIS tasks [26, 27] (e.g., network K-function [28]).

ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of China under grants 62202401 and 62372308, the Natural Science Foundation of Guangdong Province of China under grant 2023A1515011619, the Science and Technology Development Fund Macau SAR (0052/2023/RIA1, 0031/2022/A, 001/2024/SKL), the Research Grant of University of Macau (MYRG2022-00252-FST), Wuyi University Hong Kong and Macau joint Research Fund (2021WGALH14), and the Hong Kong RGC Project GRF 12202221.

REFERENCES

[1] 2024. ArcGIS. https://www.arcgis.com/index.html.

[2] 2024. Chicago Data Portal. https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if.

[3] 2024. City of Detroit Open Data Portal. https://data.detroitmi.gov/datasets/detroitmi::911-calls-for-service/about.

[4] 2024. Gainesville's Open Data Portal. https://data.cityofgainesville.org/Public-Safety/Crime-Responses/gvua-xt9q/data.

[5] 2024. Road Safety Data. https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data.

[6] 2024. SANET. http://sanet.csis.u-tokyo.ac.jp/.

[7] 2024. Seattle Open Data. https://data.seattle.gov/Transportation/SDOT-GIS-Datasets/jyjy-n3ap.

[8] 2024. spNetwork (nkde: Network Kernel density estimate). https://www.rdocumentation.org/packages/spNetwork/versions/0.4.3.2/topics/nkde.

[9] Tenindra Abeywickrama, Muhammad Aamir Cheema, and Arijit Khan. 2020. K-SPIN: Efficiently Processing Spatial Keyword Queries on Road Networks. *IEEE Trans. Knowl. Data Eng.* 32, 5 (2020), 983–997. https://doi.org/10.1109/TKDE.2019.2894140

[10] Tenindra Abeywickrama, Muhammad Aamir Cheema, and David Taniar. 2016. k-Nearest Neighbors on Road Networks: A Journey in Experimentation and In-Memory Implementation. *Proc. VLDB Endow.* 9, 6 (2016), 492–503. https://doi.org/10.14778/2904121.2904125

[11] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. 2013. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *SIGMOD*. 349–360. https://doi.org/10.1145/2463676.2465315

[12] Amira K. Al-Aamri, Graeme Hornby, Li-Chun Zhang, Abdullah A. Al-Maniri, and Sabu S. Padmadas. 2021. Mapping road traffic crash hotspots using GIS-based methods: A case study of Muscat Governorate in the Sultanate of Oman. *Spatial Statistics* 42 (2021), 100458. https://doi.org/10.1016/j.spasta.2020.100458 Towards Spatial Data Science.

[13] Adrian Baddeley, Gopalan Nair, Suman Rakshit, Greg McSwiggan, and Tilman M. Davies. 2021. Analysing point patterns on networks — A review. *Spatial Statistics* 42 (2021), 100435.

[14] Geoff Boeing. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65 (2017), 126 – 139.

[15] Darren Boss, Trisalyn Nelson, and Meghan Winters. 2018. Monitoring city wide patterns of cycling safety. *Accident Analysis & Prevention* 111 (2018), 101–108. https://doi.org/10.1016/j.aap.2017.11.008

[16] Álvaro Briz-Redón, Francisco Martínez-Ruiz, and Francisco Montes. 2019. Spatial analysis of traffic accidents near and between road intersections in a directed linear network. *Accident Analysis & Prevention* 132 (2019), 105252.

[17] 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

[18] 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

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

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

[21] 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.

[22] Tsz Nam Chan, Pak Lon Ip, Kaiyan Zhao, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. LIBKDV: A Versatile Kernel Density Visualization Library for Geospatial Analytics. *Proc. VLDB Endow.* 15, 12 (2022), 3606–3609. https://www.vldb.org/pvldb/vol15/p3606-chan.pdf

[23] 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

[24] Tsz Nam Chan, Leong Hou U, Reynold Cheng, Man Lung Yiu, and Shivansh Mittal. 2022. Efficient Algorithms for Kernel Aggregation Queries. *IEEE Trans. Knowl. Data Eng.* 34, 6 (2022), 2726–2739. https://doi.org/10.1109/TKDE.2020.3018376

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

[26] Tsz Nam Chan, Leong Hou U, Byron Choi, Jianliang Xu, and Reynold Cheng. 2023. Kernel Density Visualization for Big Geospatial Data: Algorithms and Applications. In *MDM*. IEEE, 231–234. https://doi.org/10.1109/MDM58254.2023.00046

[27] Tsz Nam Chan, Leong Hou U, Byron Choi, Jianliang Xu, and Reynold Cheng. 2023. Large-scale Geospatial Analytics: Problems, Challenges, and Opportunities. In *SIGMOD Companion*. ACM, 21–29. https://doi.org/10.1145/3555041.3589401

[28] Tsz Nam Chan, Leong Hou U, Yun Peng, Byron Choi, and Jianliang Xu. 2022. Fast Network K-function-based Spatial Analysis. *Proc. VLDB Endow.* 15, 11 (2022), 2853–2866. https://www.vldb.org/pvldb/vol15/p2853-chan.pdf

[29] 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

[30] Tsz Nam Chan, Rui Zang, Pak Lon Ip, Leong Hou U, and Jianliang Xu. 2023. PyNKDV: An Efficient Network Kernel Density Visualization Library for Geospatial Analytic Systems. In *SIGMOD Companion*. ACM, 99–102. https://doi.org/10.1145/3555041.3589711

[31] Tsz Nam Chan, Rui Zang, Bojian Zhu, Leong Hou U, Dingming Wu, and Jianliang Xu. 2024. LION: Fast and High-Resolution Network Kernel Density Visualization (Technical Report). https://github.com/edisonchan2013928/LION/blob/main/LION_TR.pdf.

[32] Muhammad Aamir Cheema, Ljiljana Brankovic, Xuemin Lin, Wenjie Zhang, and Wei Wang. 2011. Continuous Monitoring of Distance-Based Range Queries. *IEEE Trans. Knowl. Data Eng.* 23, 8 (2011), 1182–1199. https://doi.org/10.1109/TKDE.2010.246

[33] Zhida Chen, Lisi Chen, Gao Cong, and Christian S. Jensen. 2021. Location- and keyword-based querying of geo-textual data: a survey. *VLDB J.* 30, 4 (2021), 603–640. https://doi.org/10.1007/s00778-021-00661-w

[34] Javier Delso, Belén Martín, and Emilio Ortega. 2018. 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* 67 (2018), 61–72. https://doi.org/10.1016/j.jtrangeo.2018.02.001

[35] Ke Deng, Xiaofang Zhou, and Heng Tao Shen. 2007. Multi-source Skyline Query Processing in Road Networks. In *ICDE*. IEEE, 796–805. https://doi.org/10.1109/ICDE.2007.367925

[36] Xiaoyi Fu, Xiaoye Miao, Jianliang Xu, and Yunjun Gao. 2017. Continuous range-based skyline queries in road networks. *World Wide Web* 20, 6 (2017), 1443–1467. https://doi.org/10.1007/s11280-017-0444-2

[37] Yunjun Gao, Jingwen Zhao, Baihua Zheng, and Gang Chen. 2016. Efficient Collective Spatial Keyword Query Processing on Road Networks. *IEEE Trans. Intell. Transp. Syst.* 17, 2 (2016), 469–480. https://doi.org/10.1109/TITS.2015.2477837

[38] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *WEA*. 319–333. https://doi.org/10.1007/978-3-540-68552-4_24

[39] Jeremy Gelb. 2021. spNetwork: A Package for Network Kernel Density Estimation. *R Journal* 13, 2 (2021).

[40] Homayoun Harirforoush and Lynda Bellalite. 2019. A new integrated GIS-based analysis to detect hotspots: A case study of the city of Sherbrooke. *Accident Analysis & Prevention* 130 (2019), 62 – 74. https://doi.org/10.1016/j.aap.2016.08.015 Road Safety Data Considerations.

[41] Haibo Hu, Dik Lun Lee, and Victor C. S. Lee. 2006. Distance Indexing on Road Networks. In *VLDB*. 894–905. http://dl.acm.org/citation.cfm?id=1164204

[42] Ruoming Jin, Ning Ruan, Yang Xiang, and Victor E. Lee. 2012. A highway-centric labeling approach for answering distance queries on large sparse graphs. In *SIGMOD*. 445–456. https://doi.org/10.1145/2213836.2213887

[43] Jalal Khalil, Da Yan, Lyuheng Yuan, Mostafa Jafarzadehfadaki, Saugat Adhikari, Virginia P. Sisiopiku, and Zhe Jiang. 2022. Realistic urban traffic simulation with ride-hailing services: a revisit to network kernel density estimation (systems paper). In *SIGSPATIAL*. ACM, 29:1–29:10. https://doi.org/10.1145/3557915.3560963

[44] Pei-Fen Kuo and Dominique Lord. 2021. A visual approach for defining the spatial relationships among crashes, crimes, and alcohol retailers: Applying the color mixing theorem to define the colocation pattern of multiple variables. *Accident Analysis & Prevention* 154 (2021), 106062.

[45] Jiajia Li, Cancan Ni, Dan He, Lei Li, Xiufeng Xia, and Xiaofang Zhou. 2023. Efficient kNN query for moving objects on time-dependent road networks. *VLDB J.* 32, 3 (2023), 575–594. https://doi.org/10.1007/s00778-022-00758-w

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

[47] Qingquan Li, Tong Zhang, Handong Wang, and Zhe Zeng. 2011. Dynamic accessibility mapping using floating car data: a network-constrained density estimation approach. *Journal of Transport Geography* 19, 3 (2011), 379 – 393. https://doi.org/10.1016/j.jtrangeo.2010.07.003 Special Issue : Geographic Information Systems for Transportation.

[48] Ye Li, Leong Hou U, Man Lung Yiu, and Ngai Meng Kou. 2017. An Experimental Study on Hub Labeling based Shortest Path Algorithms. *Proc. VLDB Endow.* 11, 4 (2017), 445–457. https://doi.org/10.1145/3186728.3164141

[49] Zijian Li, Lei Chen, and Yue Wang. 2019. G*-Tree: An Efficient Spatial Index on Road Networks. In *ICDE*. IEEE, 268–279. https://doi.org/10.1109/ICDE.2019.00032

[50] Siqiang Luo, Yifeng Luo, Shuigeng Zhou, Gao Cong, and Jihong Guan. 2014. Distributed Spatial Keyword Querying on Road Networks. In *EDBT*. OpenProceedings.org, 235–246. https://doi.org/10.5441/002/edbt.2014.23

[51] Xiaoye Miao, Yunjun Gao, Su Guo, and Gang Chen. 2018. On Efficiently Answering Why-Not Range-Based Skyline Queries in Road Networks. *IEEE Trans. Knowl. Data Eng.* 30, 9 (2018), 1697–1711. https://doi.org/10.1109/TKDE.2018.2803821

[52] M. Mehdi Moradi, Francisco J. Rodríguez-Cortés, and Jorge Mateu. 2018. On Kernel-Based Intensity Estimation of Spatial Point Patterns on Linear Networks. *Journal of Computational and Graphical Statistics* 27, 2 (2018), 302–311. https://doi.org/10.1080/10618600.2017.1360782

[53] Jianhua Ni, Tianlu Qian, Changbai Xi, Yikang Rui, and Jiechen Wang. 2016. Spatial distribution characteristics of healthcare facilities in Nanjing: Network point pattern analysis and correlation analysis. *International journal of environmental research and public health* 13, 8 (2016), 833.

[54] A. Okabe and K. Sugihara. 2012. *Spatial Analysis Along Networks: Statistical and Computational Methods*. Wiley. https://books.google.com.hk/books?id=48GRqj51_W8C

[55] Dian Ouyang, Dong Wen, Lu Qin, Lijun Chang, Ying Zhang, and Xuemin Lin. 2020. Progressive Top-K Nearest Neighbors Search in Large Road Networks. In *SIGMOD*. ACM, 1781–1795. https://doi.org/10.1145/3318464.3389746

[56] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. 2003. Query Processing in Spatial Network Databases. In *VLDB*. Morgan Kaufmann, 802–813. https://doi.org/10.1016/B978-012722442-8/50076-8

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

[58] 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

[59] 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

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

[61] Suman Rakshit, Adrian Baddeley, and Gopalan Nair. 2019. Efficient Code for Second Order Analysis of Events on a Linear Network. *Journal of Statistical Software, Articles* 90, 1 (2019), 1–37. https://doi.org/10.18637/jss.v090.i01

[62] Michael N. Rice and Vassilis J. Tsotras. 2010. Graph Indexing of Road Networks for Shortest Path Queries with Label Restrictions. *Proc. VLDB Endow.* 4, 2 (2010), 69–80. https://doi.org/10.14778/1921071.1921074

[63] João B. Rocha-Junior, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg. 2011. Efficient execution plans for distributed skyline query processing. In *EDBT*. ACM, 271–282. https://doi.org/10.1145/1951365.1951399

[64] Gabriel Rosser, Toby O. Davies, Kate. Bowers, Shane D. Johnson, and T. Cheng. 2017. Predictive Crime Mapping: Arbitrary Grids or Street Networks? *Journal of Quantitative Criminology* 33 (2017), 569 – 594.

[65] Yikang Rui, Zaigui Yang, Tianlu Qian, Shoaib Khalid, Nan Xia, and Jiechen Wang. 2016. Network-constrained and category-based point pattern analysis for Suguo retail stores in Nanjing, China. *International Journal of Geographical Information Science* 30, 2 (2016), 186–199.

[66] Hanan Samet, Jagan Sankaranarayanan, and Houman Alborzi. 2008. Scalable network distance browsing in spatial databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, Jason Tsong-Li Wang (Ed.). ACM, 43–54. https://doi.org/10.1145/1376616.1376623

[67] Mehdi Sharifzadeh, Cyrus Shahabi, and Leyla Kazemi. 2009. Processing spatial skyline queries in both vector spaces and spatial network databases. *ACM Trans. Database Syst.* 34, 3 (2009), 14:1–14:45. https://doi.org/10.1145/1567274.1567276

[68] Boxi Shen, Xiang Xu, Jun Li, Antonio Plaza, and Qunying Huang. 2020. Unfolding Spatial-Temporal Patterns of Taxi Trip based on an Improved Network Kernel Density Estimation. *ISPRS International Journal of Geo-Information* 9, 11 (2020), 683.

[69] Bilong Shen, Ying Zhao, Guoliang Li, Weimin Zheng, Yue Qin, Bo Yuan, and Yongming Rao. 2017. V-Tree: Efficient kNN Search on Moving Objects with Road-Network Constraints. In *ICDE*. 609–620. https://doi.org/10.1109/ICDE.2017.115

[70] Weiwei Sun, Chunan Chen, Baihua Zheng, Chong Chen, and Peng Liu. 2015. An Air Index for Spatial Query Processing in Road Networks. *IEEE Trans. Knowl. Data Eng.* 27, 2 (2015), 382–395. https://doi.org/10.1109/TKDE.2014.2330836

[71] Luliang Tang, Zihan Kan, Xia Zhang, Fei Sun, Xue Yang, and Qingquan Li. 2016. A network Kernel Density Estimation for linear features in space-time analysis of big trace data. *Int. J. Geogr. Inf. Sci.* 30, 9 (2016), 1717–1737. https://doi.org/10.1080/13658816.2015.1119279

[72] Jeppe Rishede Thomsen, Man Lung Yiu, and Christian S. Jensen. 2012. Effective caching of shortest paths for location-based services. In *SIGMOD*. 313–324. https://doi.org/10.1145/2213836.2213872

[73] Teng Wang, Yandong Wang, Xiaoming Zhao, and Xiaokang Fu. 2018. 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* 71 (2018), 88–97. https://doi.org/10.1016/j.compenvurbsys.2018.04.005

[74] Yuying Wu and Yijing Li. 2022. "Hot street" of crime detection in London borough and lockdown impacts. *Geo-spatial Information Science* (2022), 1–17.

[75] Zhixiao Xie and Jun Yan. 2008. Kernel Density Estimation of traffic accidents in a network space. *Computers, Environment and Urban Systems* 32, 5 (2008), 396 – 406. https://doi.org/10.1016/j.compenvurbsys.2008.05.001

[76] Zhixiao Xie and Jun Yan. 2013. Detecting traffic accident clusters with network kernel density estimation and local spatial statistics: an integrated approach. *Journal of Transport Geography* 31 (2013), 64 – 71. https://doi.org/10.1016/j.jtrangeo.2013.05.009

[77] Hongfei Xu, Yu Gu, Yu Sun, Jianzhong Qi, Ge Yu, and Rui Zhang. 2020. Efficient processing of moving collective spatial keyword queries. *VLDB J.* 29, 4 (2020), 841–865. https://doi.org/10.1007/s00778-019-00583-8

[78] Wenhao Yu, Tinghua Ai, and Shiwei Shao. 2015. The analysis and delimitation of Central Business District using network kernel density estimation. *Journal of Transport Geography* 45 (2015), 32–47. https://doi.org/10.1016/j.jtrangeo.2015.04.008

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

[80] Jingwen Zhao, Yunjun Gao, Gang Chen, and Rui Chen. 2018. Why-Not Questions on Top-k Geo-Social Keyword Queries in Road Networks. In *ICDE*. IEEE, 965–976. https://doi.org/10.1109/ICDE.2018.00091

[81] 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.

[82] 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

[83] Ruicheng Zhong, Guoliang Li, Kian-Lee Tan, Lizhu Zhou, and Zhiguo Gong. 2015. G-Tree: An Efficient and Scalable Index for Spatial Search on Road Networks. *IEEE Trans. Knowl. Data Eng.* 27, 8 (2015), 2175–2189. https://doi.org/10.1109/TKDE.2015.2399306

[84] Andy Diwen Zhu, Hui Ma, Xiaokui Xiao, Siqiang Luo, Youze Tang, and Shuigeng Zhou. 2013. Shortest path and distance queries on road networks: towards bridging theory and practice. In *SIGMOD*. 857–868. https://doi.org/10.1145/2463676.2465277