PyNKDV: An Efficient Network Kernel Density Visualization Library for Geospatial Analytic Systems*

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

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

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

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

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

ABSTRACT

Network kernel density visualization (NKDV) is an important tool for many application domains, including criminology and transportation science. However, all existing software tools, e.g., SANET (a plug-in for QGIS and ArcGIS) and spNetwork (an R package), adopt the naïve implementation of NKDV, which does not scale to large-scale location datasets and high-resolution sizes. To overcome this issue, we develop the first python library, called PyNKDV, which adopts our complexity-reduced solution and its parallel implementation to significantly improve the efficiency for generating NKDV. Moreover, PyNKDV is also user-friendly (with four lines of python code) and can support commonly used geospatial analytic systems (e.g., QGIS and ArcGIS). In this demonstration, we will use three large-scale location datasets (up to 7.71 million data points), provide different python scripts (in the Jupyter Notebook), and install existing software tools (i.e., SANET and spNetwork) for participants to (1) explore different functionalities of our PyNKDV library and (2) compare its practical efficiency with existing software tools.

CCS CONCEPTS

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

KEYWORDS

NKDV, geospatial analytic systems, python library

ACM Reference Format:

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 *Companion of the 2023 International Conference on Management of Data (SIGMOD-Companion '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3555041.3589711

*This work was supported by the NSFC grant 62202401, the Science and Technology Development Fund Macau SAR 0015/2019/AKP, 0031/2022/A, SKL-IOTSC-2021-2023, the Research Grant of University of Macau MYRG2022-00252-FST, Wuyi University Hong Kong and Macau joint Research Fund 2021WGALH14, HKRGC C2004-21GF, and GDNSF 2019B1515130001. Pak Lon Ip and Leong Hou U are also affiliated with the State Key Laboratory of Internet of Things for Smart City, UM, and with the Centre for Data Science, UM.

SIGMOD-Companion '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 978-1-4503-9507-6/23/06…$15.00
https://doi.org/10.1145/3555041.3589711

1 INTRODUCTION

Network kernel density visualization (NKDV) [14] has been extensively used in many application domains. Some representative examples include criminology [18, 21] and transportation science [16, 17]. Figure 1 shows an example for generating NKDV using the 311-call location dataset in San Francisco [5]. Note that those red road segments (e.g., in the Lower Nob Hill district) indicate that they have higher density values for the 311 calls. In contrast, other road segments with green color (e.g., in the Western Addition district) have lower density values.

(a) 311-call location data points (b) NKDV

Figure 1: Generate NKDV (in (b)) for the 311-call location data points (i.e., the black dots in (a)) around the Lower Nob Hill and Western Addition districts in San Francisco, where the road segments with the red color and green color (in (b)) are the hotspot and coldspot regions, respectively.

Due to the wide applicability of NKDV, some recent geospatial software packages, including spNetwork [7] (an R package) and SANET [6] (a plugin for QGIS [4] and ArcGIS [1]), have been developed for supporting this tool. However, all these packages are based on the naïve implementation of NKDV, which suffers from high time complexity. With the rapid growth of geospatial data, many large-scale location datasets can be collected and analyzed nowadays. Therefore, existing packages are inefficient (or even infeasible) to generate NKDV for such large-scale datasets. Using the Chicago crime dataset [2] (with 7.69 million data points) as an example, both SANET and spNetwork take more than four hours for generating a single NKDV. Worse still, domain experts [17, 21] need to perform exploratory analysis, who may need to generate multiple NKDVs for a single dataset. This further amplifies the inefficiency issue for using off-the-shelf software packages to support this task.

To tackle this issue, we propose the first python library, called PyNKDV, for efficiently generating NKDV in different geospatial

analytic systems, including QGIS and ArcGIS. There are two main features for this library that have not been considered by existing software packages. First, we adopt our state-of-the-art algorithm, namely aggregate distance augmentation (ADA) [14], which reduces the time complexity for computing NKDV. Second, we parallelize the ADA method in order to further boost the efficiency of generating NKDV. Table 1 compares different software tools. As a remark, there are also some fast software tools, e.g., LIBKDV [13] (based on [11, 15]) and KDV-Explorer [12] (based on [10]), for generating kernel density visualization (KDV), which is the variant of this problem. However, KDV does not consider the road network, which can provide inaccurate visualization results [14, 22].

Table 1: Comparisons of different software tools for generating NKDV.

Software tool	Time-complexity reduction	Support parallelization	Used in
SANET [6]	No	No	QGIS, ArcGIS
spNetwork [7]	No	No	R
PyNKDV (ours)	Yes	Yes	Python, QGIS, ArcGIS

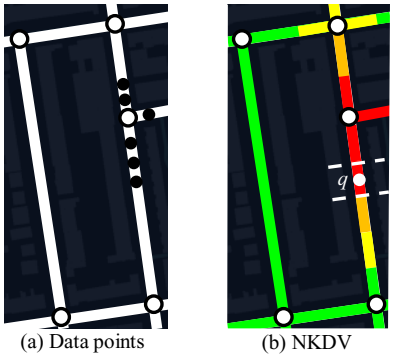
In this demonstration paper, we first overview the technical details of PyNKDV in Section 2. Then, we discuss how to use PyNKDV in Section 3. Lastly, we provide the demonstration plan for PyNKDV in Section 4.

2 TECHNICAL OVERVIEW OF PyNKDV

In this section, we have a technical overview of our library, PyNKDV. First, we formally define the NKDV problem in Section 2.1. Then, we illustrate our complexity-reduced method, aggregate distance augmentation (ADA) [14], in Section 2.2. Lastly, we discuss how to parallelize our ADA method to further improve its practical efficiency in Section 2.3.

2.1 Network Kernel Density Visualization (NKDV)

To generate NKDV for a location dataset (cf. the black data points in Figure 2a), we need to first divide each road into different lixels q (i.e., road segments). Then, we color each lixel q (cf. Figure 2b) based on the network kernel density function value (cf. Definition 1).



(a) Data points (b) NKDV

Figure 2: Illustration of NKDV, where we use different colors to denote the density values. For example, the red color and green color denote the high-density (i.e., hotspot) region and low-density (i.e., coldspot) region, respectively.

Definition 1. *[14] Given a road network $G = (V, E)$, a location dataset $P = \{p_1, p_2, ..., p_n\}$ in G, we need to compute the network kernel density function value $\mathcal{F}_P(q)$ (cf. Equation 1) for each lixel q.*

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

where w and $K_G(q, p)$ are the normalization constant and the kernel function between the lixel q and the data point p. Some representative kernel functions are summarized in Table 2.

Table 2: Some representative kernel functions, where $dist_G(q, p)$ and b denote the shortest path distance and the bandwidth parameter, respectively.

Kernel	$K_G(q, p)$	Used in
Triangular	$\begin{cases} 1 - \frac{1}{b} dist_G(q,p) & \text{if } dist_G(q,p) \le b \\ 0 & \text{otherwise} \end{cases}$	[8, 19]
Epanechnikov	$\begin{cases} 1 - \frac{1}{b^2} dist_G(q,p)^2 & \text{if } dist_G(q,p) \le b \\ 0 & \text{otherwise} \end{cases}$	[8, 23]
Quartic	$\begin{cases} (1 - \frac{1}{b^2} dist_G(q,p)^2)^2 & \text{if } dist_G(q,p) \le b \\ 0 & \text{otherwise} \end{cases}$	[19, 22]

As a remark, we adopt the Epanechnikov kernel for discussion in this paper due to space limitations. However, our method can be extended to other kernel functions (cf. Table 2).

2.2 Aggregate Distance Augmentation (ADA)

We consider $P(e)$ to be the set of data points in an edge e. Since every data point is on one and only one edge, we can decompose the network kernel density function $\mathcal{F}_P(\mathbf{q})$ (cf. Equation 1 with the Epanechnikov kernel in Table 2) into the following expression.

$$\mathcal{F}_P(q) = \sum_{e \in E} f_e(q) \tag{2}$$

where

$$f_e(q) = \sum_{p \in P(e)} w \cdot \begin{cases} 1 - \frac{1}{b^2} dist_G(q,p)^2 & \text{if } dist_G(q,p) \le b \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Hence, if we can reduce the time complexity for computing $f_e(q)$, we can also reduce the time complexity for computing $\mathcal{F}_P(q)$ (i.e., generating NKDV). To achieve this goal, we augment the aggregate distances (cf. Equation 4 and Equation 5) for all data points p in each edge $e = (u, v)$ in advance (cf. Figure 3).

$$a^{(deg)}_{P(u,p)} = \sum_{p_i \in P(u,p)} dist_G(u, p_i)^{deg} \tag{4}$$

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

where $P(u, p)$ and $P(v, p)$ are two sets of data points from node u and node v, respectively, to the data point p and deg denotes the degree value (e.g., $deg = 0$, $deg = 1$, and $deg = 2$ are used in the Epanechnikov kernel).

Therefore, if we have obtained the shortest path distances from the lixel q to the node u and node v (cf. two orange dashed lines in Figure 3), i.e., $dist_G(q, u)$ and $dist_G(q, v)$, respectively, we can use the binary search algorithm to compute $f_e(q)$ in $O(\log |P(e)|)$ time (instead of $O(|P(e)|)$ time), based on the aggregate distances (cf. Equation 4 and Equation 5).

As an example, we consider the case $dist_G(q, u) \le b$ and $dist_G(q, v) > b$ in Figure 3. Note that only those data points with

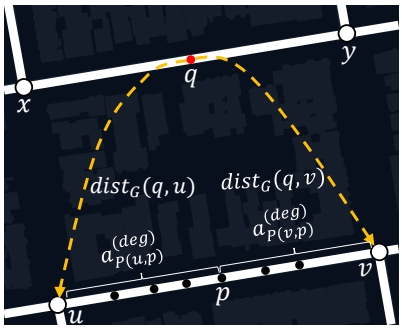


Figure 3: Core idea of ADA.

$dist_G(u,p) \leq b - dist_G(q,u)$ can contribute to $f_e(q)$ (cf. Equation 3). Therefore, we can use the binary search algorithm to first obtain the data point p^* in $e = (u,v)$ such that $dist_G(u,p^*)$ attains the maximum value that still fulfills the inequality. Then, we can evaluate $f_e(q)$ in $O(1)$ time using the following expression (based on the simple mathematical derivation).

$$f_e(q) = w\Big(1-\frac{dist_G(q,u)^2}{b^2}\Big)a^{(0)}_{P(u,p^*)} - \frac{2w\cdot dist_G(q,u)}{b^2}a^{(1)}_{P(u,p^*)} - \frac{w}{b^2}a^{(2)}_{P(u,p^*)}$$

By adopting the similar idea, we can also handle three other cases (i.e., (1) $dist_G(q,u) > b$ and $dist_G(q,v) > b$, (2) $dist_G(q,u) > b$ and $dist_G(q,v) \leq b$, and (3) $dist_G(q,u) \leq b$ and $dist_G(q,v) \leq b$) in $O(\log|P(e)|)$ time. The details can be found in [14]. In Theorem 1, we further state that the ADA method only takes $O\big(|E|\big(T_{\text{SP}} + L\log\big(\frac{n}{|E|}\big)\big)\big)$ time, where T_{SP} denotes the time complexity of the shortest path algorithm.

Theorem 1. *[14] Given a road network $G = (V,E)$, a location dataset $P = \{p_1, p_2, ..., p_n\}$ with size n, the ADA method takes $O\big(|E|\big(T_{SP} + L\log\big(\frac{n}{|E|}\big)\big)\big)$ time to generate NKDV (cf. Definition 1).*

Compared with the state-of-the-art method [20], which takes $O(|E|T_{\text{SP}} + nL)$ time, our ADA method achieves the lower worst-case time complexity (as $O\big(|E|L\log\big(\frac{n}{|E|}\big)\big) < O(nL)$).

2.3 Parallelization of ADA

In this section, we extend our previous work [14] to parallelize the ADA method in order to further improve its efficiency for generating NKDV. Consider two arbitrary lixels, e.g., q_1 and q_2, in the edge $\widehat{e} = (x,y)$ in Figure 4. Note that computing $f_e(q_1)$ ($\mathcal{F}_P(q_1)$) and computing $f_e(q_2)$ ($\mathcal{F}_P(q_2)$) (1) do not modify the same computational resources and (2) are not dependent with each other (i.e., can execute them concurrently).

Figure 4: Parallelization of the ADA method.

Based on these two reasons, we adopt the round robin approach to assign CPU threads to handle different lixels (e.g., assign the thread 1, thread 2, and thread 3 to compute the network kernel density function values (cf. Equation 2) for the lixels q_1, q_2, and q_3, respectively, in Figure 4), which can highly parallelize our ADA method.

3 HOW TO USE PyNKDV?

In order to adopt our library, PyNKDV, for generating NKDV in a location dataset, users (or domain experts) only need to write four lines of python code (cf. Figure 5), which correspond to these four steps, (1) map data points into a road network, (2) initialize the parameters for generating NKDV, (3) compute NKDV, and (4) output NKDV into a shapefile for displaying in geospatial analytic systems.

```
road_data = map_road_network(location_data)
model = PyNKDV(road_data, bandwidth=1000,
               lixel_size=5, num_threads=8)
results = model.compute()
output(results, output_file_name)
```

Figure 5: Four lines of python code to invoke PyNKDV for outputting NKDV.

Map data points into a road network: In the first step, users need to provide a csv/json file, which stores the latitude and longitude values for each geographical event (e.g., crime and traffic accident). With this input, the function, map_road_network (cf. the first line in Figure 5), adopts the OSMnx python library [9] to first extract all nodes and edges that are inside the minimum bounding rectangle of these location data points, and then further integrates these data points into the road network.

Initialize the parameters for generating NKDV: In the second step, users need to specify multiple parameters for generating NKDV, namely (1) the bandwidth parameter b (cf. Table 2), which controls the smoothness of the visualization (i.e., the size of a hotspot region) and (2) the lixel size (cf. Figure 2), which determines the resolution of the visualization, and (3) the number of CPU threads for parallelizing our ADA method (cf. Section 2.3).

Compute NKDV: In the third step, our library adopts the complexity-reduced solution, ADA (cf. Section 2.2), and its parallel implementation (cf. Section 2.3) to generate NKDV based on the parameters that are chosen in the second step.

Output NKDV into a shapefile for displaying in geospatial analytic systems: In the fourth step, our library can output the NKDV result (obtained in the third step) into a shapefile, which can be visualized in the commonly used geospatial analytic systems, QGIS [4] and ArcGIS [1].

4 DEMONSTRATION PLAN

We will use three large-scale location datasets, which are New York traffic accident dataset [3], Chicago crime dataset [2], and San Francisco 311-call dataset [5], for demonstrating our PyNKDV in the Jupyter Notebook. Here, we consider four demonstration scenarios.

Generating NKDVs for different location datasets: We prepare the python script in the Jupyter Notebook for using our PyNKDV to generate NKDV in each location dataset. Moreover, we also install two state-of-the-art NKDV software tools (cf. Table 1), i.e., SANET [6] and spNetwork [7], in our computer. In this demonstration scenario, participants can run different parts of the python

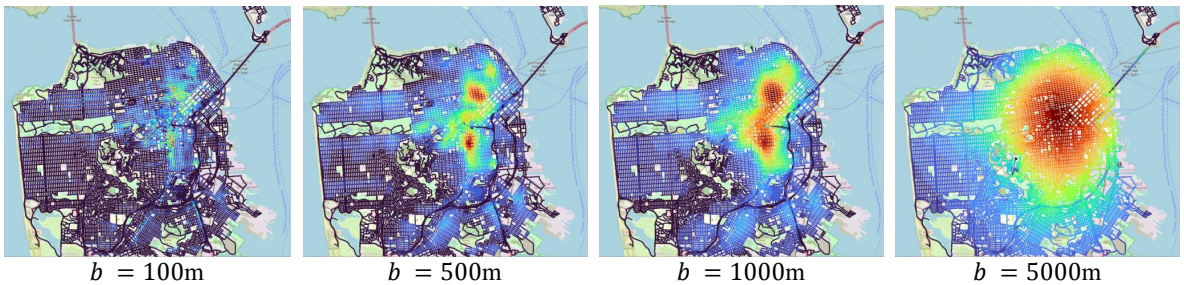


Figure 6: Generate NKDVs for the San Francisco 311-call dataset with four bandwidth parameters, namely 100m, 500m, 1000m, and 5000m, in QGIS.

script in order to visualize the hotspots of different datasets in QGIS/ArcGIS (e.g., NKDV in Figure 1b, which is obtained by our PyNKDV, is displayed in QGIS). Furthermore, they can compare the efficiency of different software tools.

Varying the bandwidth parameter b***:*** Since the bandwidth parameter b can significantly affect the hotspot regions, domain experts [17, 18] need to generate the meaningful NKDV with the proper bandwidth parameter b. To achieve this goal, they first adopt the exploratory (or trial-and-error) approach to generate NKDVs with multiple bandwidth parameters and then choose the most meaningful one. Using Figure 6 as an example, we cannot detect any hotspot if b = 100m and we can find an overly large hotspot region if b = 1000m or b = 5000m. Compared with Figure 6a, Figure 6c, and Figure 6d, using b = 500m as the bandwidth parameter (cf. Figure 6b) can detect more meaningful hotspots.

Varying the lixel size: To generate NKDV with different resolutions, we can tune the lixel size in our PyNKDV library (cf. the second line in Figure 5). If the lixel size is smaller (i.e., more lixels), the NKDV result is more accurate but the response time is higher. In this demonstration scenario, participants can choose multiple lixel sizes in order to understand how this parameter affects the visualization quality and efficiency of NKDV. Using the road r (i.e., the black arrow) in Figure 7 as an example, the smaller lixel size can show more changes in the hotspot map when we zoom in to the street level.

(a) Lixel size = 20m (b) Lixel size = 200m

Figure 7: Generate NKDVs for the San Francisco 311-call dataset with two lixel sizes, which are 20m and 200m.

Varying the number of CPU threads: Recall from Section 2.3 that we have parallelized our ADA method in the PyNKDV library. Therefore, we also provide the python script in the Jupyter Notebook to show the practical performance of this implementation with respect to different numbers of CPU threads. In this demonstration scenario, participants can specify the number of threads for calling our PyNKDV library.

REFERENCES

[1] 2023. ArcGIS. https://www.arcgis.com/index.html.
[2] 2023. Chicago Data Portal. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2.
[3] 2023. NYC Open Data. https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95.
[4] 2023. QGIS. https://www.qgis.org/en/site/.
[5] 2023. San Francisco Open Data. https://data.sfgov.org/City-Infrastructure/311-Cases/vw6y-z8j6.
[6] 2023. SANET: Spatial Analysis along Networks. http://sanet.csis.u-tokyo.ac.jp/.
[7] 2023. spNetwork: Spatial Analysis on Network. https://cran.r-project.org/web/packages/spNetwork/index.html.
[8] 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.
[9] Geoff Boeing. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65 (2017), 126–139.
[10] Tsz Nam Chan, Reynold Cheng, and Man Lung Yiu. 2020. QUAD: Quadratic-Bound-based Kernel Density Visualization. In *SIGMOD*. 35–50.
[11] 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.
[12] 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. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2655–2658.
[13] 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.
[14] 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.
[15] Tsz Nam Chan, Leong Hou U, Byron Choi, and Jianliang Xu. 2022. SLAM: Efficient Sweep Line Algorithms for Kernel Density Visualization. In *SIGMOD*. 2120–2134.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] 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.
[22] 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.
[23] 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.