KDV-Explorer: A Near Real-Time Kernel Density Visualization System for Spatial Analysis*

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

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

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

Weng Hou Tong
University of Macau
mb85407@um.edu.mo

Shivansh Mittal
The University of Hong Kong
shivansh@connect.hku.hk

Ye Li
University of Macau
yb47438@um.edu.mo

Reynold Cheng
The University of Hong Kong
ckcheng@cs.hku.hk

ABSTRACT

Kernel density visualization (KDV) is a commonly used visualization tool for many spatial analysis tasks, including disease outbreak detection, crime hotspot detection, and traffic accident hotspot detection. Although the most popular geographical information systems, e.g., QGIS, and ArcGIS, can also support this operation, these solutions are not scalable to generate a single KDV for datasets with million-scale data points, let alone to support exploratory operations (e.g., zoom in, zoom out, and panning operations) with KDV in near real-time (< 5 sec). In this demonstration, we develop a near real-time visualization system, called KDV-Explorer, that is built on top of our prior study on the efficient kernel density computation. Participants will be invited to conduct some kernel density analysis on three large-scale datasets (up to 1.3 million data points), including the traffic accident dataset, crime dataset and COVID-19 dataset. We will also compare the performance of our solution and the solutions in QGIS and ArcGIS.

PVLDB Reference Format:
Tsz Nam Chan, Pak Lon Ip, Leong Hou U, Weng Hou Tong, Shivansh Mittal, Ye Li, and Reynold Cheng. KDV-Explorer: A Near Real-Time Kernel Density Visualization System for Spatial Analysis. PVLDB, 14(12): 2655-2658, 2021.
doi:10.14778/3476311.3476312

1 INTRODUCTION

Kernel-density-estimation-based visualization, a.k.a. kernel density visualization (KDV) [6], is the de facto method for hotspot detection, which has been used in many application domains. Epidemiologists [10, 12] utilize KDV to detect the disease outbreak. Criminologists [9, 15] utilize KDV to discover the crime hotspots. Transportation experts [17] utilize KDV to find the traffic accident blackspots. Besides the above analytics tasks, IKCEST [3] utilizes the heatmap, based on KDV, to show the distribution of COVID-19 cases in different regions of China to the general public (cf. Figure 1). Here, each color of the pixel represents the density of that particular region, e.g., orange color is used to represent the hotspots (i.e., more confirmed cases) in IKCEST system.

Figure 1: KDV of COVID-19 cases in China (from [3])

Due to its wide applicability, KDV has been integrated into many geographical information systems, including QGIS [14], and ArcGIS [1]. However, KDV is a computationally expensive operation, which is not scalable to large-scale datasets. Using the New York traffic accident dataset [5] (with 1.3 million spatial data points) as an example, computing KDV takes over 0.676 trillion operations on a 512×512 screen, which is infeasible to compute in a short period of time (e.g., seconds) on a moderate machine. Worse still, there are more large-scale spatial point datasets available nowadays, which can further deteriorate the infeasibility issue for computing a single KDV in different tasks. Moreover, the practical users (e.g., transportation experts [17], epidemiologists [10]) do not generate only a single visualization for a dataset. They would perform the exploratory operations (e.g., zoom in, zoom out, and panning) with KDV to explore the hotspots in different regions, including the province level, city level, district level, and street level. For example, transportation experts can zoom in to Manhattan and perform KDV for analyzing the traffic accident hotspots (using the New York traffic accident dataset [5]) in this region. Therefore, an efficient KDV system is important to these users for performing spatial analysis.

Although both QGIS and ArcGIS have adopted some fast algorithms for evaluating KDV, these types of software cannot generate

*This work was supported by the National Key Research and Development Plan of China (No.2019YFB2102100), the Science and Technology Development Fund Macau (SKL-IOTSC-2021-2023, 0015/2019/AKP), University of Macau (MYRG2019-00119-FST), the Research Grants Council of Hong Kong (RGC Projects 17229116 and 17205015), University of Hong Kong (Projects 104005858, 104005994), HKU-TCL Joint Research Center for Artificial Intelligence (Project no. 200009430), and Guangdong-Hong Kong-Macau Joint Laboratory Program 2020 (Project No: 2020B1212030009). Pak Lon Ip, Leong Hou U, Weng Hou Tong and Ye Li are also affiliated with the State Key Laboratory of Internet of Things for Smart City. Reynold Cheng is also affiliated with Guangdong-Hong Kong-Macau Joint Laboratory for Smart Cities.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476312

KDV within a reasonable time for a large-scale dataset and cannot support exploratory operations. For example, QGIS needs to use more than 30 minutes to generate a single KDV for New York traffic accident dataset [5]. Therefore, we ask a question, *can we develop a system that can achieve the near real-time KDV for datasets with million-scale data points?*

In this demonstration, we develop the web-based system, called KDV-Explorer, which incorporates our state-of-the-art solution QUAD [6] for generating KDV on different datasets, including traffic accident dataset [5], crime dataset [2], and COVID-19 dataset [4]. As a remark, this web-based system offers the opportunity for the users to perform hotspot analysis in their personal computers/smartphones, without installing new software. Furthermore, we illustrate how to efficiently support different types of exploratory operations, e.g., zoom in, zoom out, and panning, with KDV, using representative kernel functions.

Related Work. In recent years, many efficient algorithms for supporting KDV with exploratory operations (zoom in, zoom out and panning) have been developed [11, 13, 18] in which all of these studies focus on using the modern hardware, e.g., GPU [11, 13], or distributed computation [18], to boost the efficiency for computing KDV. However, these studies need to consume many computational resources (e.g., 16 computers (in parallel) with one GPU for each computer in [13]) in order to generate KDV for million-scale datasets in real-time (< 0.5 sec) [11, 13, 18]. Therefore, these approaches are not well-suited for domain experts (e.g., transportation experts [17]), who do not necessarily have many computational resources, to analyze big spatial data (e.g., 1.3M traffic accident dataset [5]). Compared with these research studies, our KDV-Explorer is the first system, which supports near real-time KDV (i.e., time efficient) on a moderate machine (using only one core of the process (i.e., resource efficient)) with exploratory operations (zoom in, zoom out and panning). In Table 1, we summarize the overall performance of well-known visualization software for KDV, QGIS and ArcGIS, compared with KDV-Explorer.

Table 1: Comparisons of different visualization software

Software (or system)	Exploratory operations	Time efficiency	Resource efficiency
QGIS [14]	×	×	✓
ArcGIS [1]	×	×	✓
KDV-Explorer (ours)	✓	✓	✓

2 TECHNICAL OVERVIEW OF KDV-EXPLORER

In this section, we describe the details for evaluating KDV. Since evaluating the exact KDV is time-consuming, we evaluate the approximate version of KDV (ϵKDV), which is formulated as follows.

Problem 1. *(ϵKDV [6]) Given the visualized region with resolution $M \times N$, the set P of data points and the relative error ϵ, the color of each pixel $\mathbf{q}$ in this region depends on the approximate value $A_P(\mathbf{q})$ such that:*

$$(1-\epsilon)\mathcal{F}_P(\mathbf{q}) \leq A_P(\mathbf{q}) \leq (1+\epsilon)\mathcal{F}_P(\mathbf{q}) \tag{1}$$

where $\mathcal{F}_P(\mathbf{q})$ is the kernel density estimation value of pixel $\mathbf{q}$, such that:

$$\mathcal{F}_P(\mathbf{q}) = \frac{1}{|P|}\sum_{\mathbf{p_i}\in P} K(\mathbf{q}, \mathbf{p_i}) \tag{2}$$

In practice, we can achieve similar visualization performance once we specify the error parameter ϵ to be 0.05 [6] in Problem 1. Here, we describe two main components of KDV-Explorer, which are (1) our quadratic bound functions, i.e., QUAD (cf. Section 2.1) and (2) indexing framework (cf. Section 2.2). These two components can significantly improve the efficiency for the evaluation of ϵKDV, which can then support exploratory operations with KDV in near real-time for large-scale datasets.

2.1 QUAD

In order to efficiently obtain $A_P(\mathbf{q})$ for each pixel $\mathbf{q}$, we adopt the state-of-the-art lower and upper bound functions [6] for $\mathcal{F}_P(\mathbf{q})$. Here, we use the triangular kernel as an example to illustrate the concept of these lower and upper bound functions, where:

$$K(\mathbf{q}, \mathbf{p_i}) = \max\left(1 - \frac{1}{b}\cdot dist(\mathbf{q}, \mathbf{p_i}), 0\right) \tag{3}$$

Here, b and $dist(\mathbf{q}, \mathbf{p_i})$ denote the bandwidth of the triangular kernel function and Euclidean distance, respectively.

In [6], we illustrate that once we let $x_i = \frac{1}{b}\cdot dist(\mathbf{q}, \mathbf{p_i})$ in the kernel function (cf. Equation 3), we can utilize two quadratic bounds $Q^L(x_i) = a_l {x_i}^2 + c_l$ and $Q^U(x_i) = a_u {x_i}^2 + c_u$ for approximating this kernel function $\max(1 - x_i, 0)$, as shown in Figure 2.

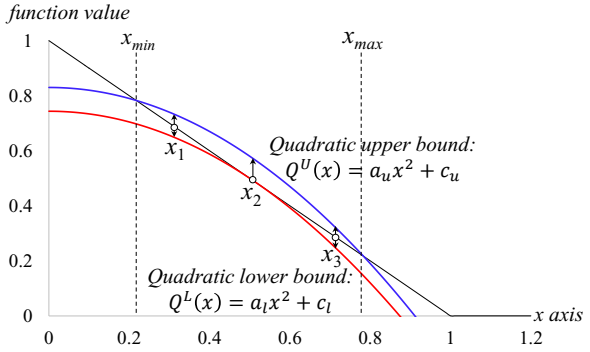


Figure 2: Approximation of triangular kernel function, by the quadratic bound functions ($Q^L(x)$ and $Q^U(x)$)

Then, we can replace each kernel function $K(\mathbf{q}, \mathbf{p_i})$ by its lower and upper bounds, which provide the lower and upper bound functions for $\mathcal{F}_P(\mathbf{q})$, where:

$$LB(\mathbf{q}) = \frac{1}{|P|}\sum_{\mathbf{p_i}\in P}\left(a_l\left(\frac{1}{b}dist(\mathbf{q}, \mathbf{p_i})\right)^2 + c_l\right) \tag{4}$$

$$UB(\mathbf{q}) = \frac{1}{|P|}\sum_{\mathbf{p_i}\in P}\left(a_u\left(\frac{1}{b}dist(\mathbf{q}, \mathbf{p_i})\right)^2 + c_u\right) \tag{5}$$

Since $\sum_{\mathbf{p_i}\in P} dist(\mathbf{q}, \mathbf{p_i})^2$ can be efficiently evaluated in $O(d)$ time [6], both $LB(\mathbf{q})$ and $UB(\mathbf{q})$ can be computed in $O(d)$ time. In [6], we also mention how to obtain the suitable parameters a_l/ c_l and a_u/ c_u in order to make the tightest lower and upper bound values. Once we obtain these two bound functions, we can utilize the following condition (cf. Lemma 1) to check whether it fulfills the theoretical guarantee (cf. Equation 1).

Lemma 1. *[8] If $\frac{UB(\mathbf{q})-LB(\mathbf{q})}{UB(\mathbf{q})+LB(\mathbf{q})} \leq \epsilon$, we can achieve the theoretical guarantee (cf. Equation 1) for Problem 1, where:*

$$A_P(\mathbf{q}) = \frac{2LB(\mathbf{q})UB(\mathbf{q})}{LB(\mathbf{q}) + UB(\mathbf{q})} \tag{6}$$

Table 2 summarizes the kernel functions that our system KDV-Explorer can support. Compared with our preliminary work [6], KDV-Explorer further extends QUAD for handling other commonly used kernel functions, including quartic kernel, which is the default kernel for QGIS/ ArcGIS software, and Epanechnikov kernel.

Table 2: Kernel functions

Kernel	$K(\mathbf{q}, \mathbf{p_i})$
Gaussian	$\exp(-\frac{1}{b^2} dist(\mathbf{q}, \mathbf{p_i})^2)$
Quartic	$\max((1 - \frac{1}{b^2} dist(\mathbf{q}, \mathbf{p_i})^2)^2, 0)$
Epanechnikov	$\max(1 - \frac{1}{b^2} dist(\mathbf{q}, \mathbf{p_i})^2, 0)$
Triangular	$\max(1 - \frac{1}{b} dist(\mathbf{q}, \mathbf{p_i}), 0)$
Cosine	$\begin{cases} \cos(\frac{1}{b} dist(\mathbf{q}, \mathbf{p_i})) & \text{if } dist(\mathbf{q}, \mathbf{p_i}) \leq \frac{\pi}{2} b \\ 0 & \text{otherwise} \end{cases}$
Exponential	$\exp(-\frac{1}{b} dist(\mathbf{q}, \mathbf{p_i}))$

2.2 Indexing framework for Bound Functions

Since both $LB(\mathbf{q})$ and $UB(\mathbf{q})$ (cf. Equations 4 and 5, respectively) are derived based on the full dataset P, these two bound functions may not be tight. Therefore, we can first build the index structure (cf. Figure 3) on the dataset P and then utilize the indexing framework to further tighten the bound functions until they fulfill the condition in Lemma 1. For details, please refer to our preliminary work [6, 7].

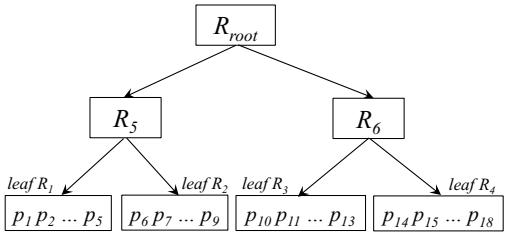


Figure 3: Index structure (e.g., kd-tree) on the dataset P [6]

3 KDV-EXPLORER

In this section, we first illustrate the user interface of our system in Section 3.1. Then, we describe the system architecture in Section 3.2. After that, we discuss the use case of the exploratory operations, including zoom in, zoom out, and panning, for our system in Section 3.3. Lastly, we show the use case of the representative kernel functions (cf. Table 2) in Section 3.4.

3.1 User Interface

In this section, we illustrate the user interface and the important functionalities (i.e., (a) to (e) in Figure 4) of KDV-Explorer. To support KDV by our system, the users need to declare these three inputs, which are (1) kernel type, (2) number of pixels (i.e., $M \times N$) and (3) dataset.

For (1), we support all kernel types in Table 2, which are also supported by existing geographical information systems, e.g., QGIS, and ArcGIS. The users can select the desired kernel type in the list box (a). In addition, the users can also tune the bandwidth of different kernel functions, i.e., the variable b in Equation 3, to control the smoothness of the chosen kernel function in (b). With the larger bandwidth b, the density plot can be smoother.

To specify the number of pixels in KDV-Explorer for (2), we allow the users to specify different levels in the bar (c). Here, the larger

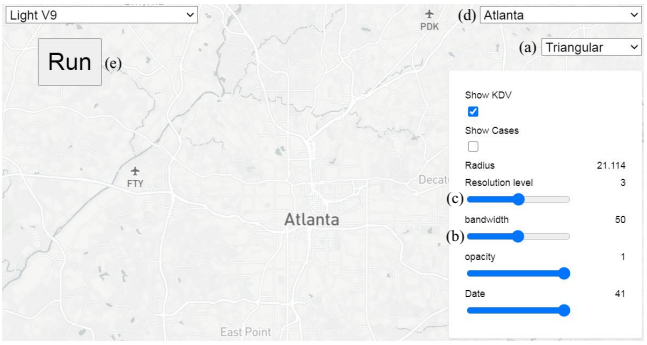


Figure 4: The user interface of KDV-Explorer

the number of levels, the larger the number of pixels. Therefore, the response time is also longer, but the visualization quality is better.

For (3), KDV-Explorer offers three datasets for demonstration, including traffic accident dataset in New York [5], crime dataset in Atlanta [2], and COVID-19 dataset [4]. The users can select the dataset in the list box (d) for visualization.

After the users specify the parameters of (1), (2) and (3), they can click the "Run" button (in (e)), KDV-Explorer adopts the efficient methods of both Sections 2.1 and 2.2 to output the visualization to the users.

3.2 System Architecture

Figure 5 shows the system architecture of KDV-Explorer. Each user can interact with the system using the exploratory operations, e.g., zoom in, zoom out, and panning. In the server side, Apache HTTP Server is used to receive user operations and then asks Express.JS to take the corresponding actions (e.g., zoom in). Once the user requests for the heatmap of a geographical region (on her screen), i.e., click the "Run" button in Figure 4, the KDV computation module (i.e., QUAD) computes the density value for each pixel and then Apache HTTP Server returns the result to the client side. After that, DECK.GL, which is a WebGL-powered framework, can generate the heatmap, based on our result, to the user.

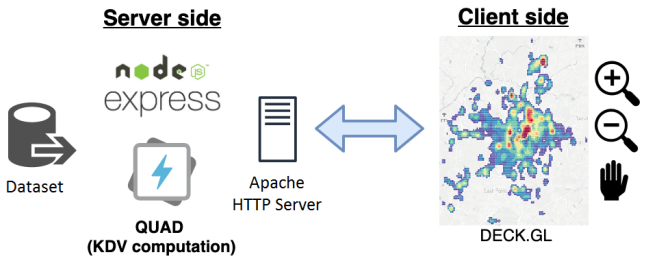
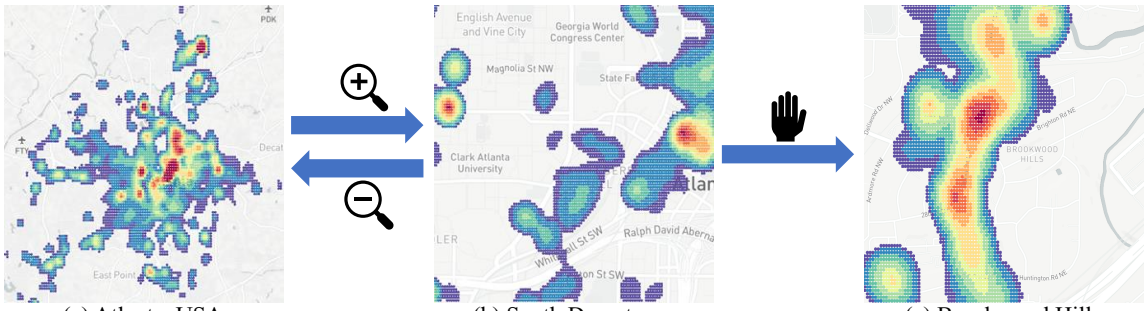


Figure 5: System architecture

3.3 Use Case of Exploratory Operations

Recall from Section 3.2, KDV-Explorer can support three types of exploratory operations, including zoom in, zoom out, and panning. Figure 6 shows the use case of these operations to generate KDV in different geographical regions, using the crime dataset in Atlanta [2].

Suppose that we generate KDV in Atlanta (cf. Figure 6a), we can discover there are multiple crime hotspots (with red color) in this region. Observe that there is a relatively huge hotspot region in South Downtown. However, we cannot distinguish which parts



(a) Atlanta, USA (b) South Downtown (c) Brookwood Hills

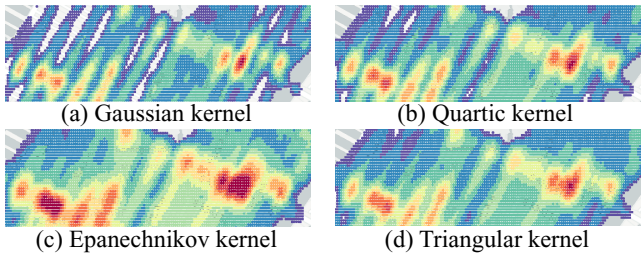
Figure 6: Generating KDV in different geographical regions via exploratory operations, including zoom in, zoom out, and panning, with the crime dataset in Atlanta

of South Downtown contain higher density of crime events. To provide more detailed visualization, we can further zoom in to this region and request for generating KDV again (cf. Figure 6b). Observe that KDV-Explorer can show more detailed visualization in this region (i.e., two crime hotspots in the South Downtown).

Since the users can also be interested in visualizing different regions, KDV-Explorer also provides the panning operation for the users to select the interested region for visualization. As an example, after we pan the visualized region from South Downtown to Brookwood Hills, we can generate another KDV for this region (cf. Figure 6c).

3.4 Use Case of Kernel Functions

In real applications, domain experts (e.g., transportation experts, criminologists) adopt different types of kernel functions, e.g., Gaussian [9], quartic [17], Epanechnikov [16], and triangular [9] kernels, to perform KDV analysis. Therefore, KDV-Explorer extends our preliminary work [6] to support more kernel functions, including Epanechnikov and quartic kernels. Figure 7 shows the use case for generating KDV in Upper Manhattan, using the traffic accident dataset [5] in New York. Observe that KDV with different kernel functions can provide different shapes of hotspot regions.



(a) Gaussian kernel (b) Quartic kernel

(c) Epanechnikov kernel (d) Triangular kernel

Figure 7: Generating KDV in Upper Manhattan with different kernel functions, using the traffic accident dataset in New York

4 DEMONSTRATION PLAN

In this demonstration, we have two objectives:

- Demonstrate that KDV-Explorer can efficiently support online (or web-based) KDV with exploratory operations (including zoom in, zoom out, and panning) and commonly-used kernel functions (cf. Table 2) for multiple users.
- Demonstrate that KDV-Explorer can achieve significant speedup, compared with existing geographical information systems, including QGIS, and ArcGIS.

For the first objective, we use three large-scale datasets (up to 1.3 million data points), including traffic accident [5], crime [2], and COVID-19 [4] datasets, as the case studies. For ease of use, we develop a web-based prototype system to demonstrate our efficient KDV tool. In the demonstration, audiences can simultaneously access to the website (by their computers or smartphones) and issue the KDV queries on different regions of the world.

For the second objective, we will pre-install ArcGIS and QGIS and configure our web-based solution KDV-Explorer in a laptop. Then, the audiences can compare the response time between three solutions.

REFERENCES

[1] ArcGIS. https://www.arcgis.com/.
[2] Atlanta Police Department Open Data. http://opendata.atlantapd.org/.
[3] IKCEST: Disaster Risk Reduction. http://drr.ikcest.org/knowledge_service/ncp.html.
[4] Johns Hopkins University: Coronavirus Resource Center. https://coronavirus.jhu.edu/map.html.
[5] New York State Government Open Data. https://data.ny.gov/browse.
[6] T. N. Chan, R. Cheng, and M. L. Yiu. QUAD: Quadratic-bound-based kernel density visualization. In *SIGMOD*, pages 35–50. ACM, 2020.
[7] T. N. Chan, M. L. Yiu, and L. H. U. KARL: Fast kernel aggregation queries. In *ICDE*, pages 542–553, 2019.
[8] T. N. Chan, M. L. Yiu, and L. H. U. The power of bounds: Answering approximate earth mover's distance with parametric bounds. *IEEE Transactions on Knowledge and Data Engineering*, 33(2):768–781, 2021.
[9] T. Hart and P. Zandbergen. Kernel density estimation and hotspot mapping: examining the influence of interpolation method, grid cell size, and bandwidth on crime forecasting. *Policing: An International Journal of Police Strategies and Management*, 37:305–323, 2014.
[10] P. Lai, C.-M. Wong, A. Hedley, S. Lo, P. Leung, J. Kong, and G. Leung. Understanding the spatial clustering of severe acute respiratory syndrome (sars) in hong kong. *Environmental health perspectives*, 112:1550–6, 12 2004.
[11] O. D. Lampe and H. Hauser. Interactive visualization of streaming data with kernel density estimation. In *PacificVis*, pages 171–178, 2011.
[12] N. Muroga, Y. Hayama, T. Yamamoto, A. Kurogi, T. Tsuda, and T. Tsutsui. The 2010 foot-and-mouth disease epidemic in japan. *The Journal of veterinary medical science / the Japanese Society of Veterinary Science*, 74:399–404, 11 2011.
[13] A. Perrot, R. Bourqui, N. Hanusse, F. Lalanne, and D. Auber. Large interactive visualization of density functions on big data infrastructure. In *LDAV*, pages 99–106, 2015.
[14] QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2009.
[15] A. Ristea, M. A. Boni, B. Resch, M. S. Gerber, and M. Leitner. Spatial crime distribution and prediction for sporting events using social media. *Int. J. Geogr. Inf. Sci.*, 34(9):1708–1739, 2020.
[16] X. Shi, M. Li, O. Hunter, B. Guetti, A. Andrew, E. Stommel, W. Bradley, and M. Karagas. Estimation of environmental exposure: interpolation, kernel density estimation or snapshotting. *Annals of GIS*, 25(1):1–8, 2019. PMID: 30687456.
[17] K. Xie, K. Ozbay, A. Kurkcu, and H. Yang. Analysis of traffic crashes involving pedestrians using big data: Investigation of contributing factors and identification of hotspots. *Risk Analysis*, 37(8):1459–1476, 2017.
[18] Y. Zheng, Y. Ou, A. Lex, and J. M. Phillips. Visualization of big spatial data using coresets for kernel density estimates. In *VDS, to appear*. IEEE, 2017.