# Large-scale Geospatial Analytics: Problems, Challenges, and Opportunities


**(Edison) Tsz Nam Chan**[1] Leong Hou U[2] Byron Choi[1]

Jianliang Xu[1] Reynold Cheng[3]

[1]Hong Kong Baptist University [2]University of Macau [3]The University of Hong Kong

# Tutorial Outline

1. Background of Geospatial Analytics

2. Overview of Different Geospatial Analysis Tools

3. Kernel Density Visualization (KDV)

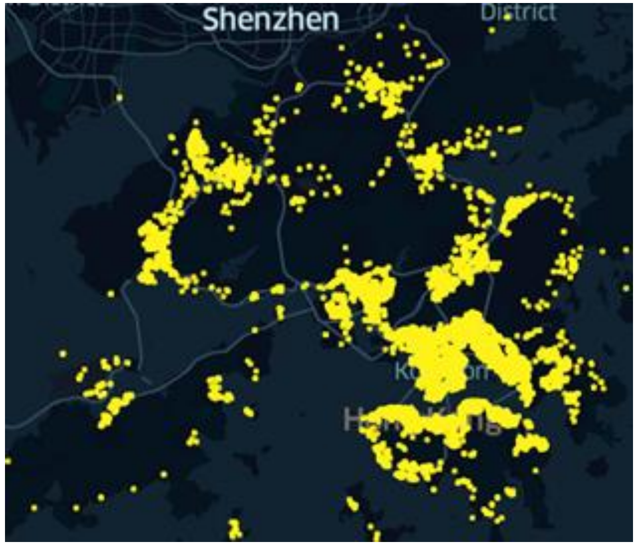
4. K-function

5. Future Opportunities

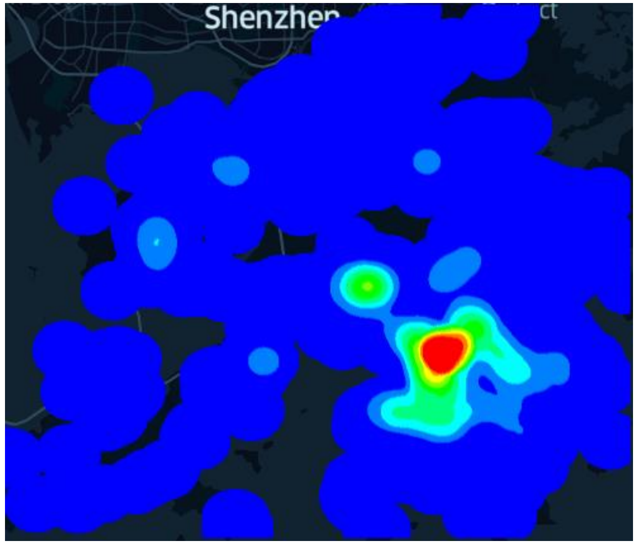
# Background of Geospatial Analytics

# Why Geospatial Analytics?

- Epidemiologists analyze disease outbreak in different regions.



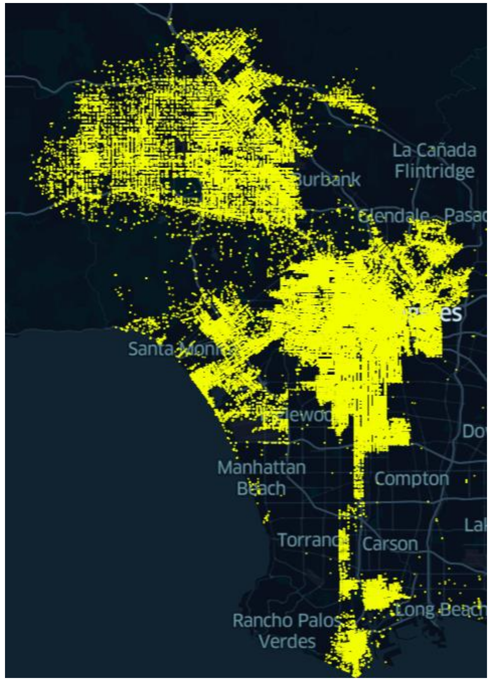

(a) Hong Kong COVID-19 cases



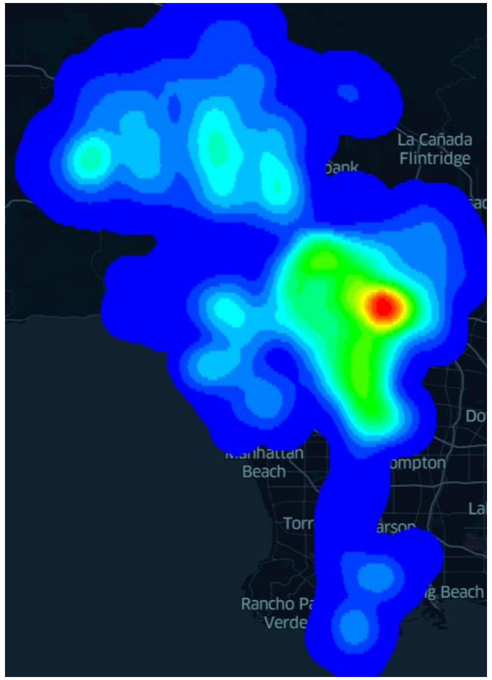

(b) Hotspot map

# Why Geospatial Analytics?

- Criminologists/Transportation experts need to detect the crime/traffic accident hotspots in different regions.



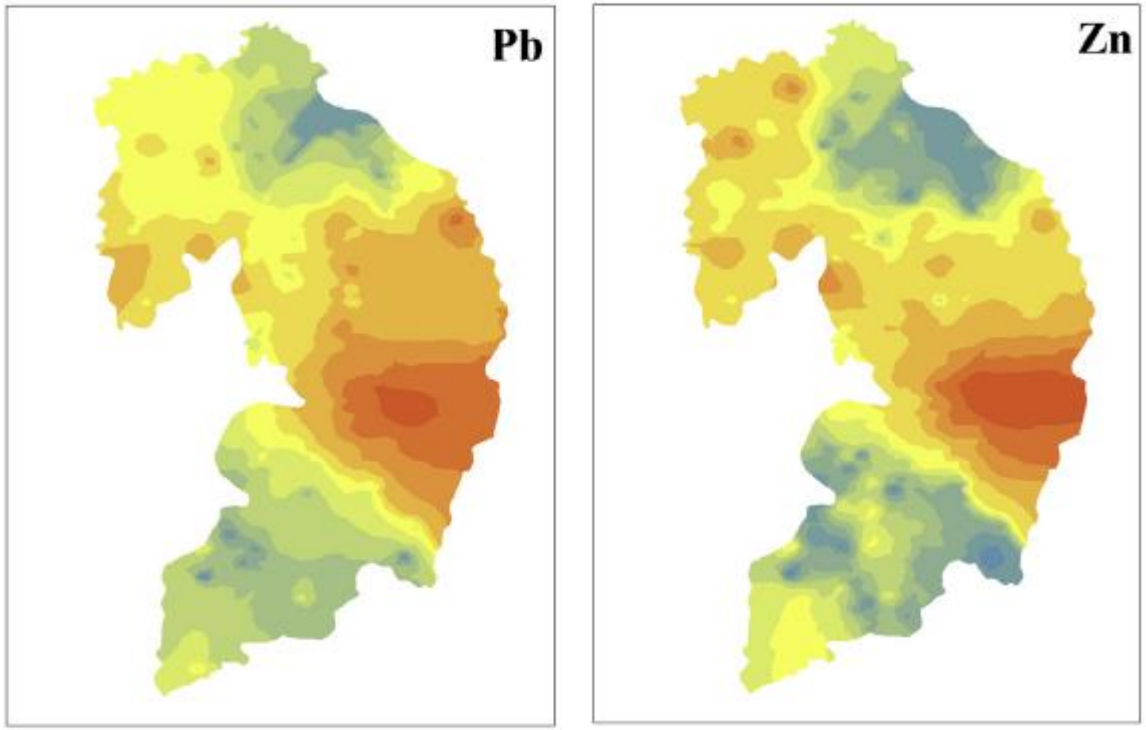

(a) Crime events in Los Angeles




(b) Crime hotspots

# Why Geospatial Analytics?

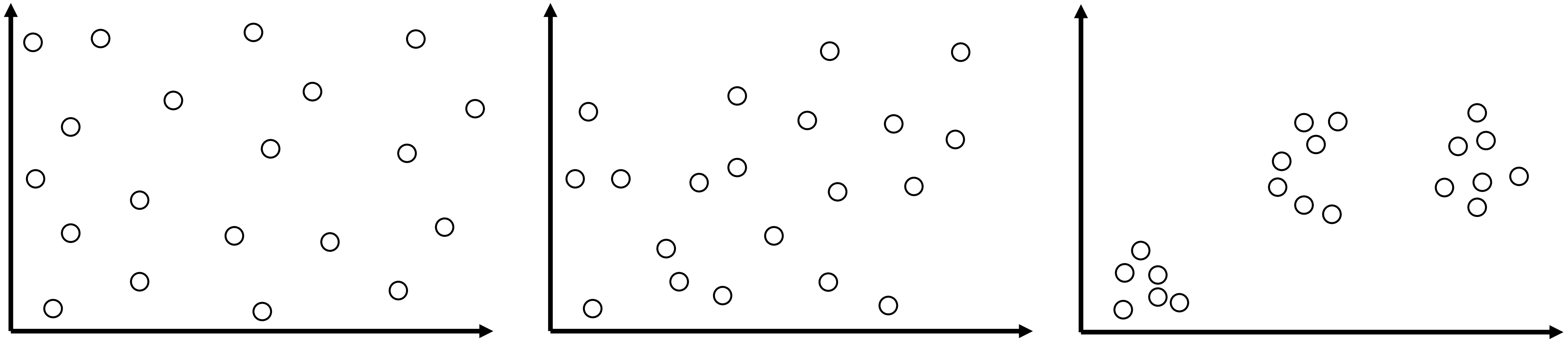
- Ecologists need to analyze the air pollution levels in different geographical regions.




[JEM18] Q. Ding, Y. Wang, D. Zhuang. Comparison of the common spatial interpolation methods used to analyze potentially toxic elements surrounding mining regions. Journal of Environmental Management 2018.

# Why Geospatial Analytics?

- Geographical researchers need to analyze the cluster properties of a location dataset.



(a) Dispersed (b) Random (c) Clustered

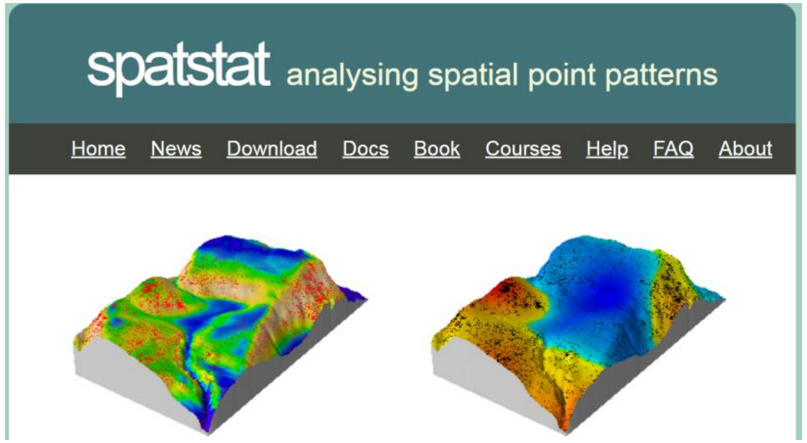
# Representative Tools in Geospatial Analytics

| Application type | Geospatial analytic tool |
|---|---|
| Hotspot detection | Kernel density visualization (KDV) |
| | Inverse distance weighting (IDW) |
| | Kriging |
| Correlation analysis | $K$-function |
| | Moran's I |
| | Getis-Ord General G |

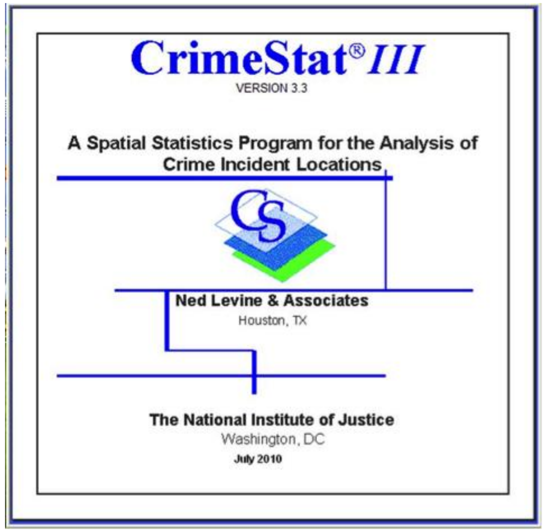
# Software Packages for Supporting Geospatial Analysis Tools

ArcGIS

PySAL




spNetwork



# Geospatial Analysis Tools are Slow!

- At least quadratic time complexity for these tools, where $n$ is the number of data points ☹

- Large-scale location datasets are available ☹
  - San Francisco 311-call location dataset contains more than 8 million data points.
  - New York taxi location dataset contains nearly 14 million data points.

- Lack of efficient algorithms for handling these tools ☹

- Lack of efficient software packages for handling these tools ☹

# Geospatial Analysis Tools are Slow!

- Many complaints from domain experts ☹
  - Gramacki et al. [SIP17] “However, many (or even most) of the practical algorithms and solutions designed in the context of KDE are very time-consuming with quadratic computational complexity being a commonplace.”
  - Zhang et al. [IJGIS16] “Given what we have seen above, conducting this type of analysis using a sequential Ripley’s K function is extremely time-consuming, even to the level which prohibits this comprehensive analysis.”
  - Hohl et al. [SSE16] “The detection of space-time clusters can be computationally demanding, and this issue is exacerbated with spatiotemporal datasets of increasing size, diversity and availability (Grubesic et al., 2014; Robertson et al., 2010).”

[SIP17] A. Gramacki. Nonparametric Kernel Density Estimation and Its Computational Aspects. Springer International Publishing, 2017.
[IJGIS16] G. Zhang, Q. Huang, A. X. Zhu, J. H. Keel. Enabling point pattern analysis on spatial big data using cloud computing: optimizing and accelerating Ripley’s K function. International Journal of Geographical Information Science 2016.
[SSE16] A. Hohl, E. Delmelle, W. Tang, I. Casas. Accelerating the discovery of space-time patterns of infectious diseases using parallel computing. Spatial and Spatio-temporal Epidemiology 2016.

# What Should Database Researchers Do?

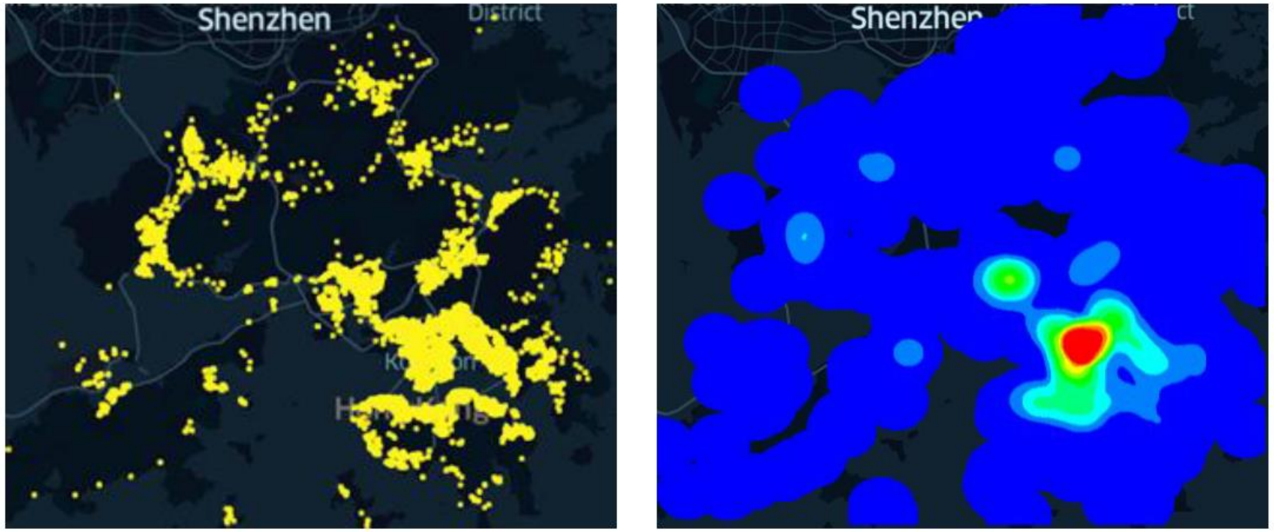
- Regard different tools as the spatial query processing problems.

| Application type | Geospatial analytic tool |
|---|---|
| Hotspot detection | Kernel density visualization (KDV) |
| | Inverse distance weighting (IDW) |
| | Kriging |
| Correlation analysis | *K*-function |
| | Moran's I |
| | Getis-Ord General G |

- Develop efficient algorithms (based on some techniques in database (e.g., indexing)) for these spatiotemporal query processing problems.

# Overview of Different Geospatial Analysis Tools

# Kernel Density Visualization (KDV)



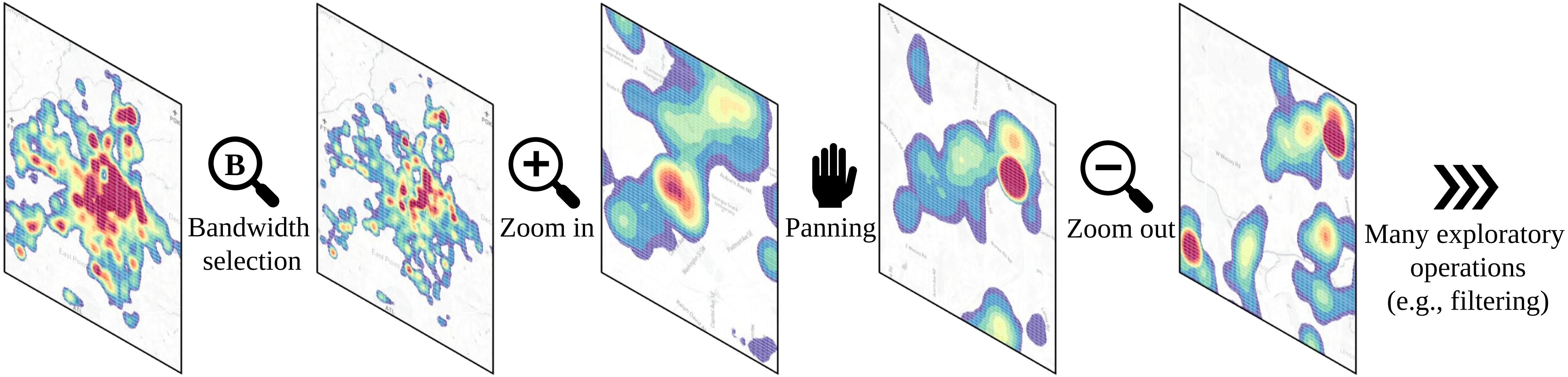

- Each **p** (yellow dot) represents the location of a COVID-19 case.
- Predict the risk of a given location **q** by computing the ***kernel density function*** $\mathcal{F}_P(\mathbf{q})$.

$$\underbrace{\mathcal{F}_{P}}_{\text{dataset}}(\overbrace{\mathbf{q}}^{\text{2D pixel}}) = \sum_{\mathbf{p}\in P} \overbrace{w}^{\text{weighting}} \cdot \begin{cases} 1 - \frac{1}{\underbrace{b^2}_{\text{bandwidth}}} \overbrace{dist(\mathbf{q},\mathbf{p})}^{\text{Euclidean distance}}{}^2 & \text{If } dist(\mathbf{q},\mathbf{p}) \leq b \\ 0 & \text{Otherwise} \end{cases}$$

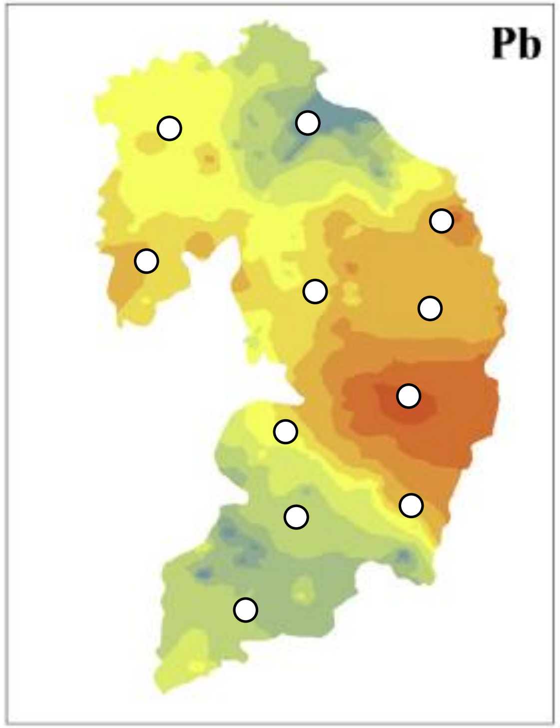
# Challenges of KDV

- The time complexity is $O(XYn)$ ☹
  - $X \times Y$ denotes the number of pixels.
  - $n$ denotes the number of location data points.

- Domain experts need to generate multiple KDVs ☹



# Inverse Distance Weighting (IDW)




- Each white point **p** denotes the location of a sensor, which has the value $v_{\mathbf{p}}$ for measuring the level of Pb.

- Predict the level of Pb for the location **q** based on the inverse distance weighting function $I_P(\mathbf{q})$.

$$I_P(\mathbf{q}) = \begin{cases} \dfrac{\sum_{\mathbf{p}\in P}\left(\dfrac{v_{\mathbf{p}}}{dist(\mathbf{q},\mathbf{p})^{deg}}\right)}{\sum_{\mathbf{p}\in P}\left(\dfrac{1}{dist(\mathbf{q},\mathbf{p})^{deg}}\right)} & \text{If } dist(\mathbf{q},\mathbf{p}) \neq 0 \text{ for all } \mathbf{p} \\ v_{\mathbf{p}} & \text{Otherwise} \end{cases}$$

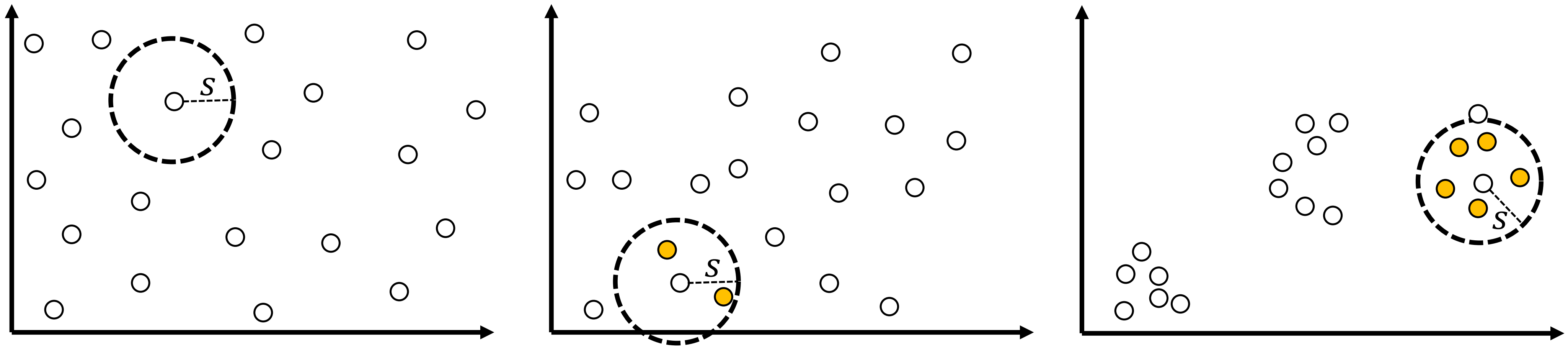
# Challenges of IDW

- The time complexity is $O(XYn)$ ☹
  - $X \times Y$ denotes the number of pixels
  - $n$ denotes the number of sensors.

- Need to achieve real-time performance (< 0.5 sec) ☹
  - Liang et al. [TGIS18] "With very large numbers of concurrent observation streams, novel algorithms are necessary that integrate streams into rasters, or other continuous representations, continuously in real time."

[TGIS18] Q. Liang, S. Nittel, J. C. Whittier, S. Bruin. Real-time inverse distance weighting interpolation for streaming sensor data. Transactions in GIS. 2018.

# K-function




(a) Dispersed (b) Random (c) Clustered

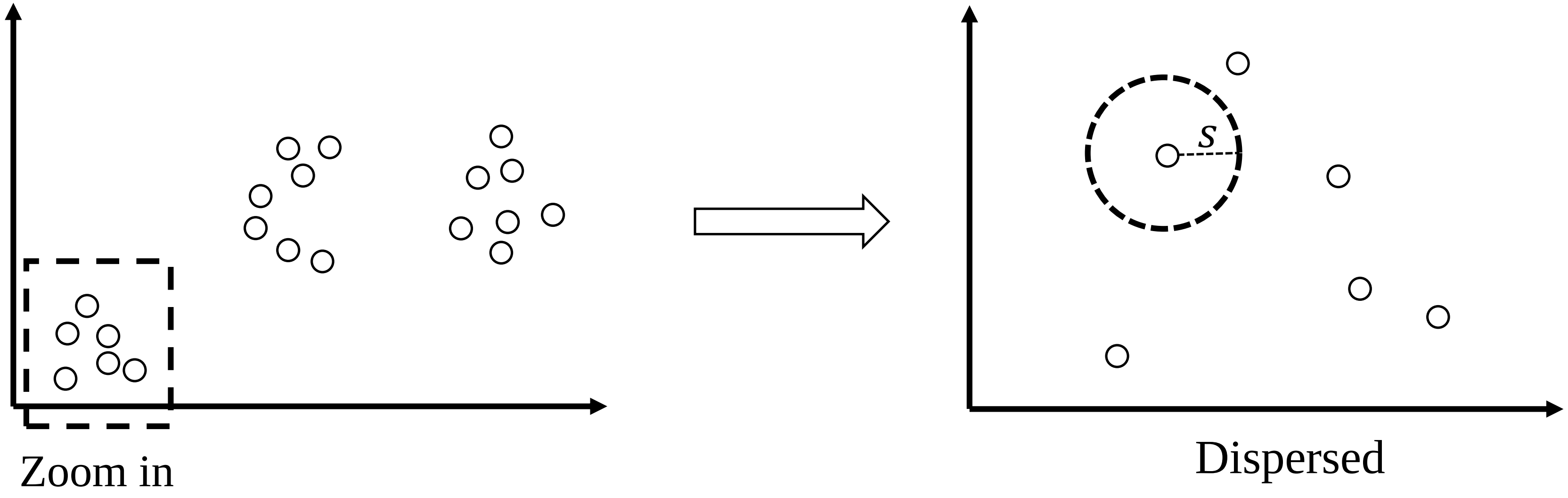
- Each **p** (white dot) represents the location of a geographical event (e.g., COVID-19 case or traffic accident).

- Domain experts need to know the cluster property of each dataset for a given spatial threshold *s* using the K-function.

$$K_P(s) = \sum_{\mathbf{p}_i \in P} \sum_{\substack{\mathbf{p}_j \in P \\ \mathbf{p}_j \neq \mathbf{p}_i}} \mathbb{I}(dist(\mathbf{p}_i, \mathbf{p}_j) \leq s) \quad \text{where } \mathbb{I}(dist(\mathbf{p}_i, \mathbf{p}_j) \leq s) = \begin{cases} 1 & \text{if } dist(\mathbf{p}_i, \mathbf{p}_j) \leq s \\ 0 & \text{otherwise} \end{cases}$$

# K-function

- A location dataset may exhibit different cluster properties under different thresholds.




- Domain experts need to know the cluster properties under different spatial thresholds.

# K-function Plot

- Provide a location dataset $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and $D$ thresholds, which are $s_1, s_2, \ldots, s_D$.

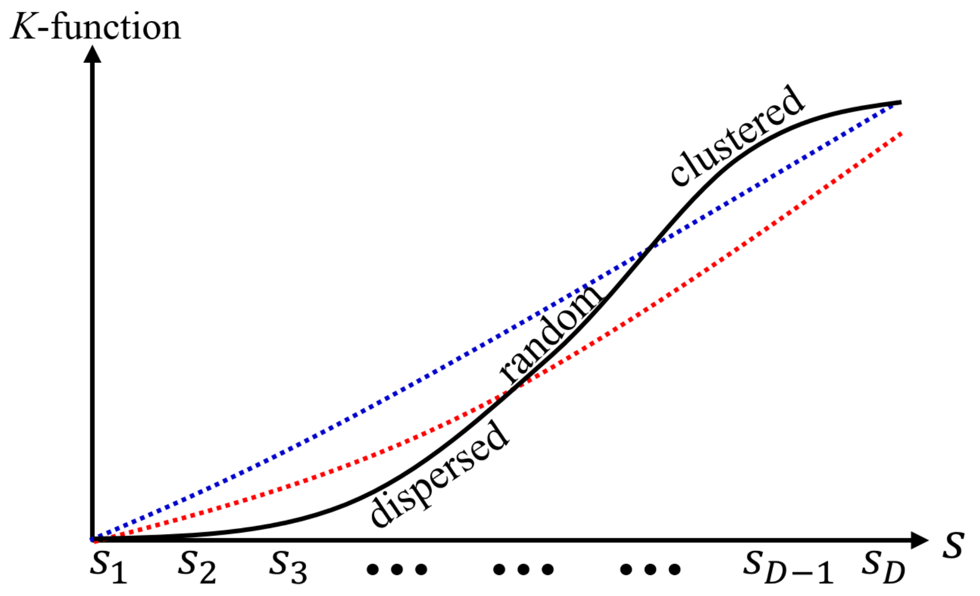
- Randomly generate $L$ datasets, which are $R_1, R_2, \ldots, R_L$.

- For each threshold $s_d$ $(1 \leq d \leq D)$, compute the following three terms.

  (1) $K_P(s_d)$

  (2) $\mathcal{L}(s_d) = \min\left(K_{R_1}(s_d), K_{R_2}(s_d), \ldots, K_{R_L}(s_d)\right)$

  (3) $U(s_d) = \max\left(K_{R_1}(s_d), K_{R_2}(s_d), \ldots, K_{R_L}(s_d)\right)$



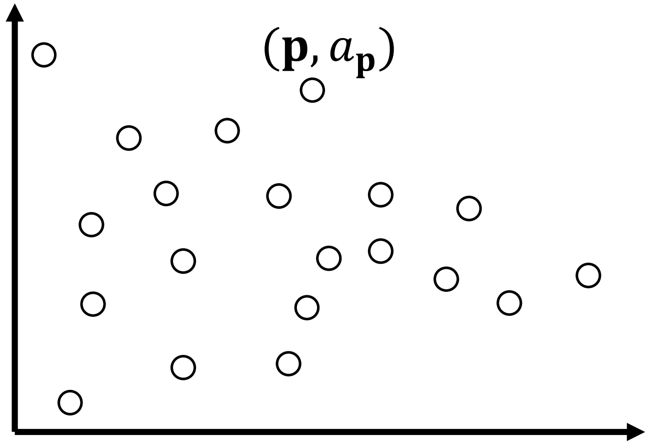
# Challenges of K-function

- The time complexity of K-function is $O(n^2)$ ☹

- Need to compute multiple K-functions in order to generate a K-function plot, which takes $O(LDn^2)$ time ☹

# Moran's I

- Each white data point $(\mathbf{p}, a_{\mathbf{p}})$ is represented by the location $\mathbf{p}$ (e.g., location of a traffic accident) and one attribute $a_{\mathbf{p}}$ (e.g., age and number of injuries).




- Analyze the autocorrelation between the attributes of these data points based on the Moran's I function.

$$M_P = \frac{\sum_{(\mathbf{p}_i, a_{\mathbf{p}_i}) \in P} \sum_{(\mathbf{p}_k, a_{\mathbf{p}_k}) \in P, k \neq i} \frac{(a_{\mathbf{p}_i} - \mu_a)(a_{\mathbf{p}_k} - \mu_a)}{dist(\mathbf{p}_i, \mathbf{p}_k)^{deg}}}{\left(\sum_{(\mathbf{p}_i, a_{\mathbf{p}_i}) \in P} \sum_{(\mathbf{p}_k, a_{\mathbf{p}_k}) \in P, k \neq i} \frac{1}{dist(\mathbf{p}_i, \mathbf{p}_k)^{deg}}\right) \sum_{(\mathbf{p}_i, a_{\mathbf{p}_i}) \in P} (a_{\mathbf{p}_i} - \mu_a)^2}$$

where $\mu_a = \frac{\sum_{(\mathbf{p}_i, a_{\mathbf{p}_i}) \in P} a_{\mathbf{p}_i}}{|P|}$

# Challenges of Moran's I

- The time complexity of Moran's I is $O(n^2)$ ☹
- Cannot be scalable to moderate-scale datasets ☹
  - Amgalan et al. [ICDM20] "Although statistics like Moran's I and Geary's C are widely used to measure spatial autocorrelation, they are slow: all popular methods run in $\Omega(n^2)$ time, rendering them unusable for large data sets, or long time-courses with moderate numbers of points."

[ICDM20] A. Amgalan, L. Mujica-Parodi, S. S. Skiena. Fast Spatial Autocorrelation. ICDM 2020.

# Kernel Density Visualization (KDV)

# State-of-the-art Solutions for Generating KDV

- Function approximation [**TKDE22**, **SIGMOD20**, **ICDE19**, SIGMOD17, SDM03]
- Data sampling [SOCG18, SODA18, SODA13, SIGMOD13]
- Computational sharing [**SIGMOD22**, **VLDB22a**, AISTATS03]

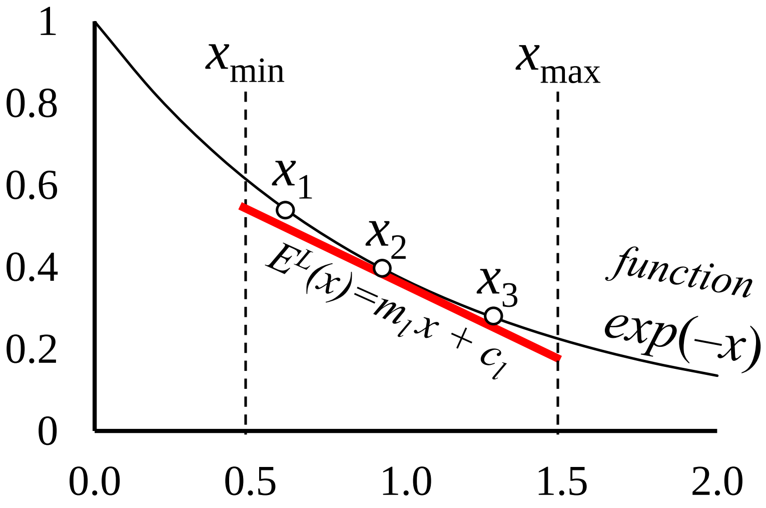
[TKDE22] T. N. Chan, L. H. U, R. Cheng, M. L. Yiu, Shivansh Mittal. Efficient Algorithms for Kernel Aggregation Queries. TKDE 2022.
[SIGMOD22] T. N. Chan, L. H. U, B. Choi, J. Xu. SLAM: Efficient Sweep Line Algorithms for Kernel Density Visualization. SIGMOD 2022.
[VLDB22a] T. N. Chan, P. L. Ip, L. H. U, B. Choi, J. Xu. SAFE: A Share-and-Aggregate Bandwidth Exploration Framework for Kernel Density Visualization. VLDB 2022.
[SIGMOD20] T. N. Chan, R. Cheng, M. L. Yiu. QUAD: Quadratic-Bound-based Kernel Density Visualization. SIGMOD 2020.
[ICDE19] T. N. Chan, M. L. Yiu, L. H. U. KARL: Fast Kernel Aggregation Queries. ICDE 2019.
[SOCG18] J. M. Phillips and W. M. Tai. Near-Optimal Coresets of Kernel Density Estimates. SOCG 2018.
[SODA18] J. M. Phillips and W. M. Tai. Improved Coresets for Kernel Density Estimates. SODA 2018.
[SIGMOD17] E. Gan and P. Bailis. Scalable Kernel Density Classification via Threshold-Based Pruning. SIGMOD 2017.
[SODA13] J. M. Phillips. $\epsilon$-Samples for Kernels. In SODA 2013.
[SIGMOD13] Y. Zheng, J. Jestes, J. M. Phillips, F. Li. Quality and Efficiency for Kernel Density Estimates in Large Data. SIGMOD 2013.
[SDM03] A. G. Gray and A. W. Moore. Nonparametric Density Estimation: Toward Computational Tractability. SDM 2003.
[AISTATS03] A. G. Gray and A. W. Moore. Rapid Evaluation of Multiple Density Models. AISTATS 2003.

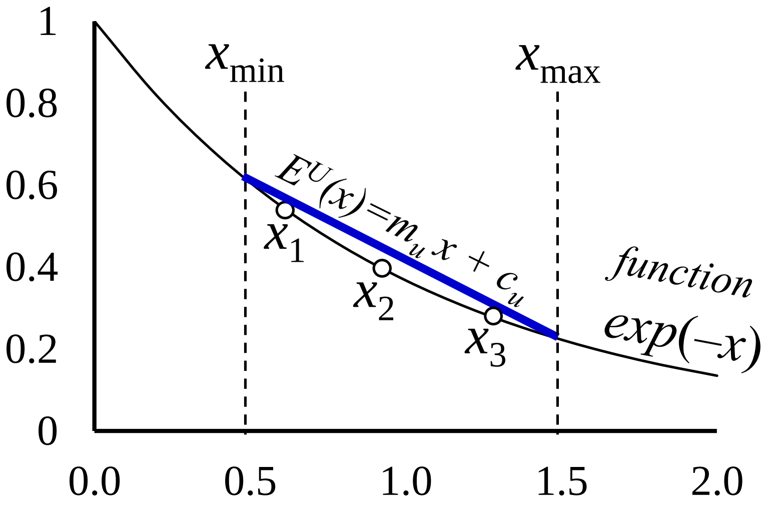
# Function Approximation

- Consider the kernel density function (with the Gaussian kernel).

$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p}\in P} w \cdot \exp\left(\underbrace{-\frac{1}{b^2} dist(\mathbf{q},\mathbf{p})^2}_{x}\right)$$

- Use some simple functions (e.g., linear functions) to approximate the exponential function so that we can obtain the lower and upper bounds of $\mathcal{F}_P(\mathbf{q})$.



# Function Approximation

$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p}\in P} w \cdot \exp\Bigg(\underbrace{-\frac{1}{b^2} dist(\mathbf{q},\mathbf{p})^2}_{x}\Bigg) \qquad O(n) \text{ time}$$

We have $LB_P(\mathbf{q}) \leq \mathcal{F}_P(\mathbf{q}) \leq UB_P(\mathbf{q})$.

Lower bound of $\mathcal{F}_P(\mathbf{q})$:

$$LB_P(\mathbf{q}) = \sum_{\mathbf{p_i}\in P} w\left(m_l \underbrace{\left(\frac{1}{b^2} dist(\mathbf{q},\mathbf{p})^2\right)}_{x} + c_l\right)$$

$$= wm\frac{1}{b^2}\big(\underbrace{|P|\|\mathbf{q}\|^2}_{\mathrm{O}(1)} - \underbrace{2\mathbf{q}\cdot\mathbf{a_P}}_{\mathrm{O}(1)} + b_P\big) + wc|P| \qquad O(1) \text{ time}$$

We can further tighten these bound values using some index structures (e.g., kd-tree) until they fulfill the relative error guarantees.

# Advantages and Disadvantages of Function Approximation

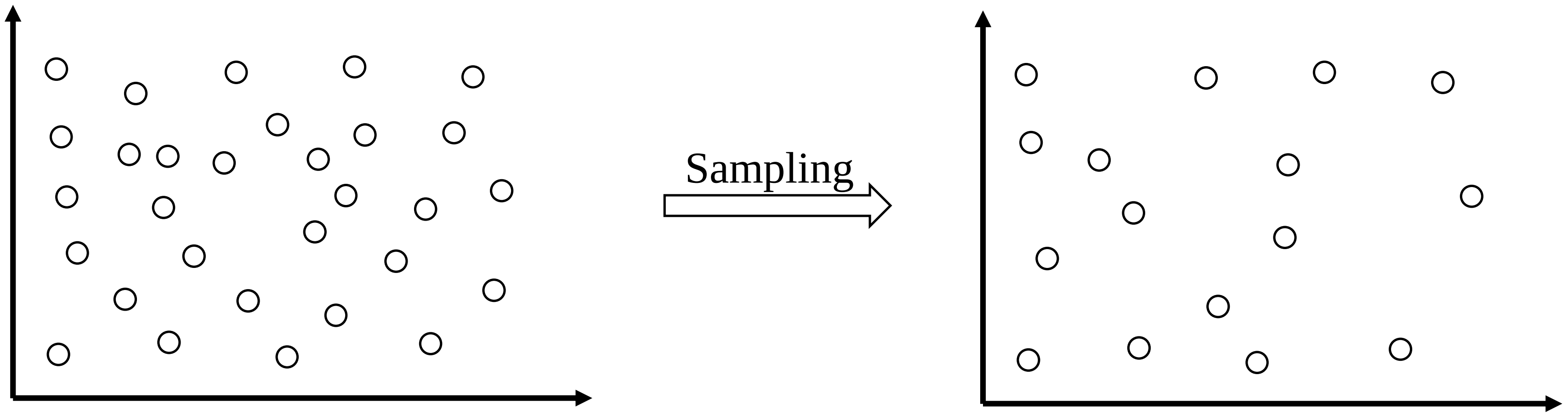
- Advantages ☺
  - Achieve better practical performance.
  - Can handle all kernel functions.
  - Can achieve approximation guarantees for generating KDV.

- Disadvantages ☹
  - Cannot reduce the worst-case time complexity for generating KDV.
  - Cannot achieve exact solution.
  - Can still be slow for generating KDV with some famous kernel functions (Epanechnikov and quartic kernels).

# Data Sampling

- Consider the kernel density function (with the Gaussian kernel).

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




- Compute the modified kernel density function based on the sampled dataset $S$.

$$\mathcal{F}_S^{(M)}(\mathbf{q}) = \sum_{\mathbf{p}_i \in S} w_i \cdot \exp\left(-\frac{1}{b^2} dist(\mathbf{q}, \mathbf{p}_i)^2\right)$$

# Advantages and Disadvantages of Data Sampling

- Advantages ☺
  - Can achieve probabilistic approximation guarantees for generating KDV.
  - Can reduce the worst-case time complexity for generating KDV.
  - Can handle all kernel functions.

- Disadvantages ☹
  - Cannot achieve exact solution.
  - Can still be slow for generating KDV.
  - Can degrade the practical visualization quality.

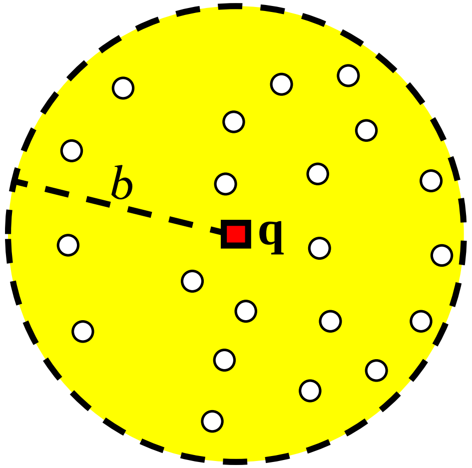
# Computational Sharing

- Consider the kernel density function (with the Epanechnikov kernel).

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

- Only those white data points can contribute to $\mathcal{F}_P(\mathbf{q})$.

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

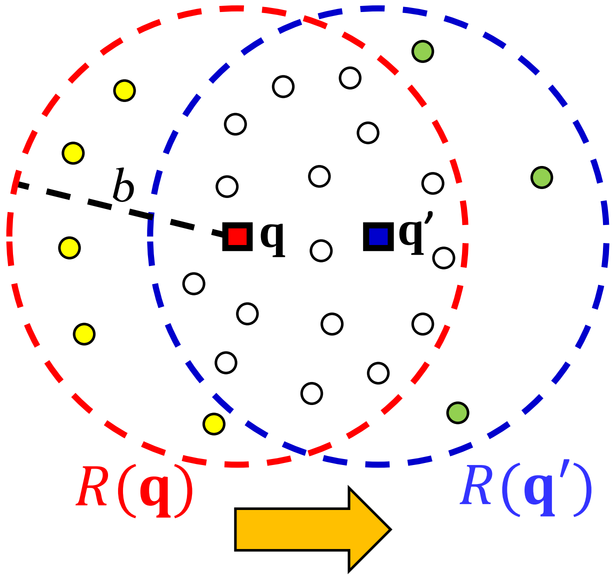



$R(\mathbf{q})$

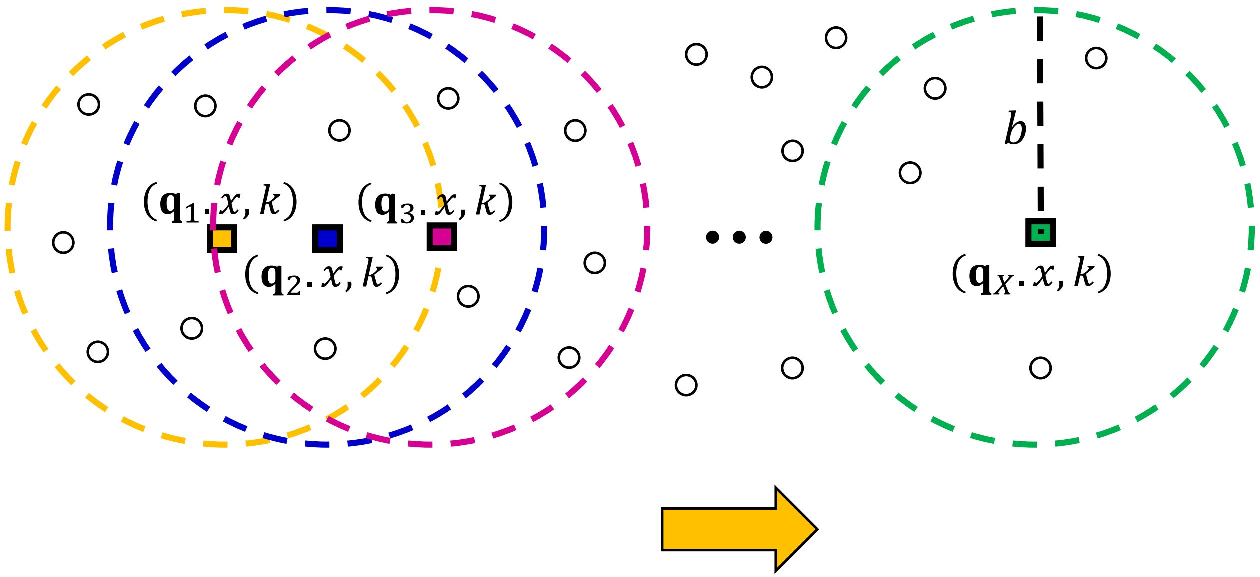
- Efficiently maintaining $R(\mathbf{q})$ for each pixel $\mathbf{q}$ can improve the efficiency.

# Computational Sharing

- Two consecutive pixels can share many data points (white circles) in the range set.




- Consider a row of pixels. If we can efficiently share the computations of $R(\mathbf{q})$ between these pixels **q**, we can improve the efficiency of generating KDV.



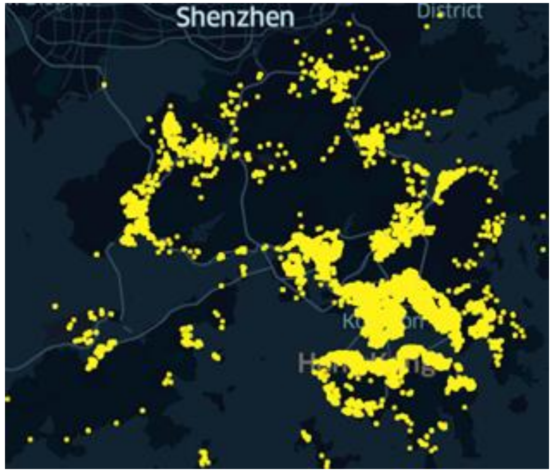
# Advantages and Disadvantages of Computational Sharing

- Advantages ☺
  - Can achieve the exact solution
  - Can reduce the worst-case time complexity
  - Can achieve the best practical efficiency
  - Can combine with data sampling methods

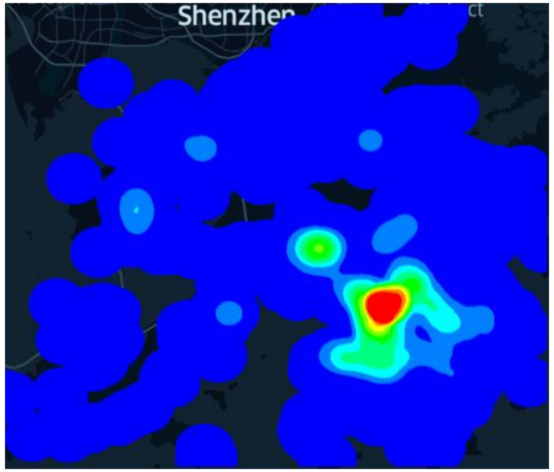
- Disadvantages ☹
  - Cannot support all kernel functions (e.g., cannot support Gaussian kernel).
  - Cannot achieve optimal worst-case time complexity.

# Variant 1: Spatiotemporal Kernel Density Visualization (STKDV)

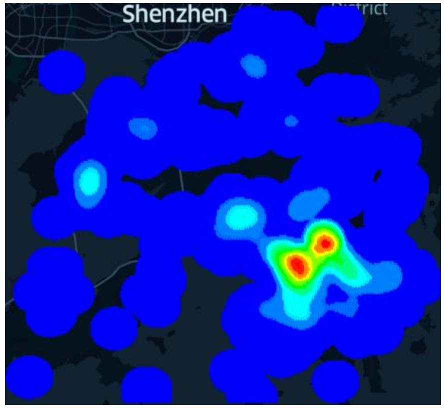
- KDV does not consider the occurrence time of each geographical event, which may provide misleading visualization results.



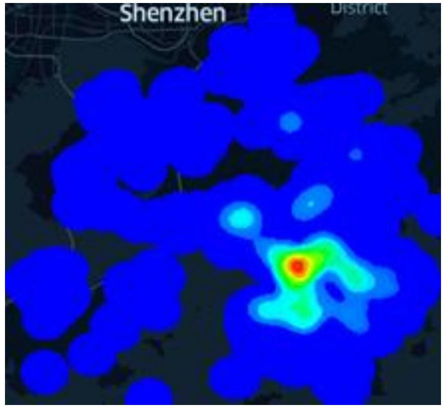

Hong Kong COVID-19 cases



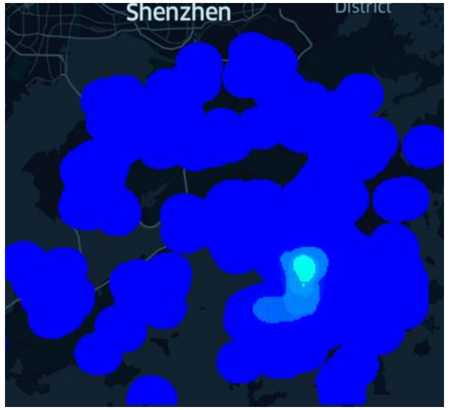
Hotspot map (based on KDV)



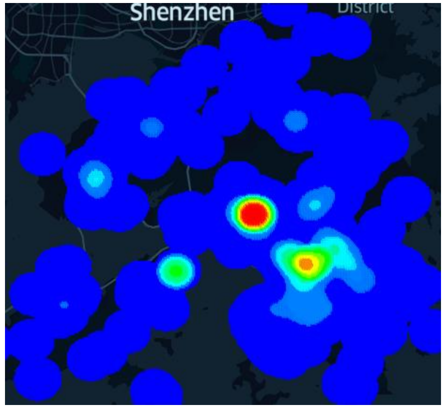

2nd August 2020




6th December 2020

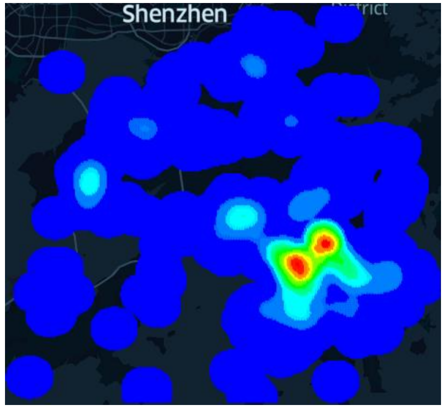



28th February 2021

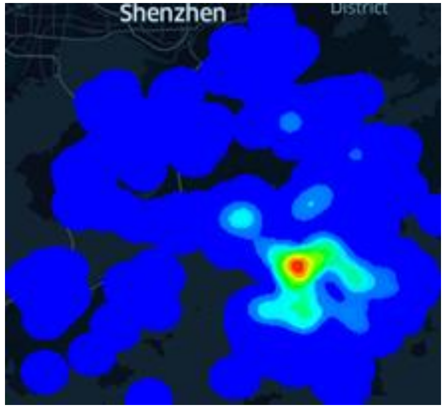



28th January 2022

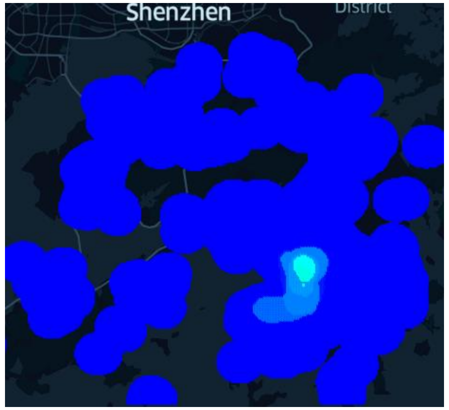
# Variant 1: Spatiotemporal Kernel Density Visualization (STKDV)



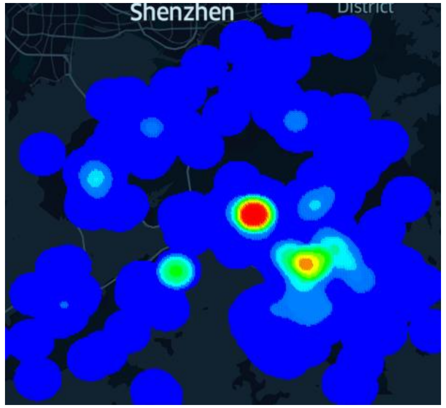

$2^{nd}$ August 2020




$6^{th}$ December 2020




$28^{th}$ February 2021




$28^{th}$ January 2022

- Consider a location dataset $\hat{P} = \{(\mathbf{p}_1, t_{\mathbf{p}_1}), (\mathbf{p}_2, t_{\mathbf{p}_2}), \dots, (\mathbf{p}_n, t_{\mathbf{p}_n})\}$ with size $n$.

- Color each pixel $\mathbf{q}$ with the timestamp $t_{\mathbf{q}}$ based on the spatial-temporal kernel density function $\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}})$.

$$\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}}) = \sum_{(\mathbf{p}, t_{\mathbf{p}}) \in \hat{P}} w \cdot K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot K_{\text{time}}(t_{\mathbf{q}}, t_{\mathbf{p}})$$

# Variant 1: Spatiotemporal Kernel Density Visualization (STKDV)

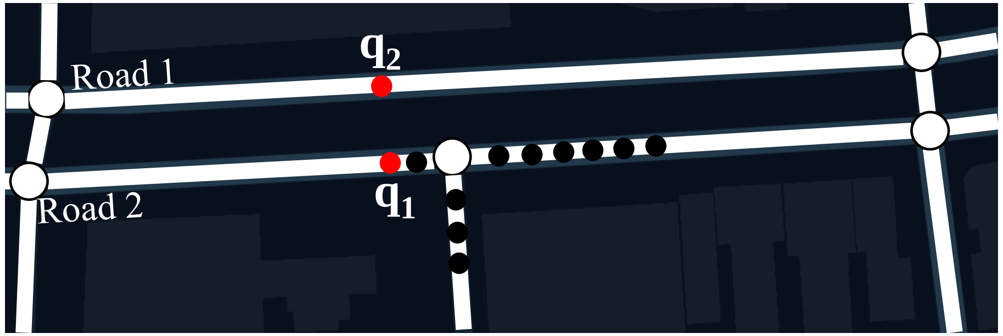
- Time complexity of a naïve solution is $O(XYTn)$ (Very slow!) ☹

- The time complexity of the best solution, called SWS [**VLDB22b**], is $O(XY(T+n))$ ☺

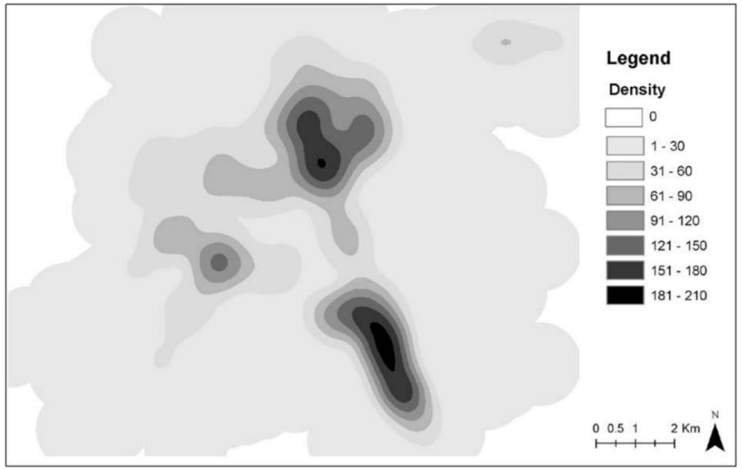
[VLDB22b] T. N. Chan, P. L. Ip, L. H. U, B. Choi, J. Xu. SWS: A Complexity-Optimized Solution for Spatial-Temporal Kernel Density Visualization. VLDB 2022.

# Variant 2: Network Kernel Density Visualization (NKDV)

- KDV ignores the road network
  1. Can overestimate the density value of some regions (e.g., $\mathbf{q}_2$)




  2. Cannot correctly identify which road segments are the hotspot.



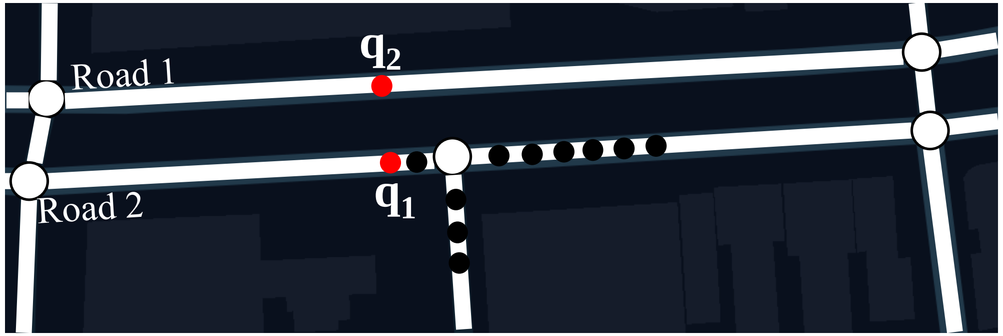

Kernel Density Visualization (KDV)

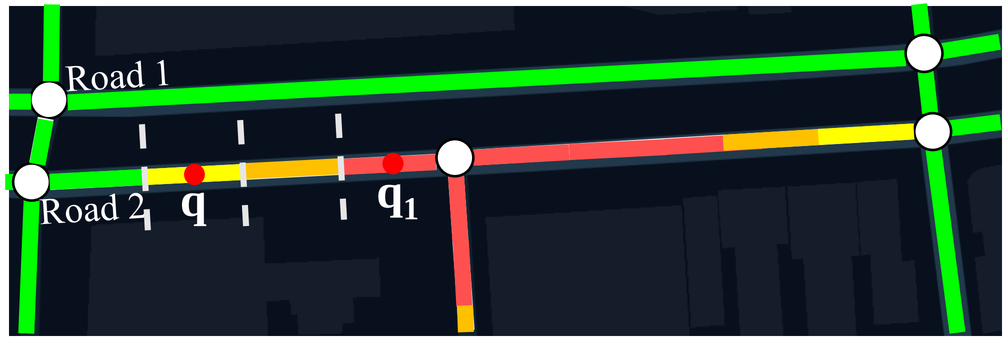

Network Kernel Density Visualization (NKDV)

# Variant 2: Network Kernel Density Visualization (NKDV)

- Divide each road in the road network $G = (V, E)$ into a set of lixels.
- Color each lixel $\mathbf{q}$, based on the network kernel density function.

$$\mathcal{F}_{\underbrace{P}_{\text{dataset}}}(\overbrace{\mathbf{q}}^{\text{lixel}}) = \sum_{\mathbf{p_i}\in P} \overbrace{w}^{\text{weighting}} \cdot \begin{cases} 1 - \frac{1}{\underbrace{b}_{\text{bandwidth}}^2} \overbrace{dist_G(\mathbf{q},\mathbf{p_i})}^{\text{shortest path distance}}{}^2 & \text{if } dist_G(\mathbf{q},\mathbf{p_i}) \le b \\ 0 & \text{otherwise} \end{cases}$$



# Variant 2: Network Kernel Density Visualization (NKDV)

- Time complexity of a naïve solution is $O(L(T_{\mathrm{SP}} + n))$ (Very slow!) ☹
  - $L$ is the number of lixels.
  - $T_{SP}$ is the time complexity of a shortest path algorithm.
  - $n$ is the number of data points.

- Time complexity of the best solution, ADA [**VLDB21a**], is $O\left(|E|\left(T_{\mathrm{SP}} + L\log\left(\frac{n}{|E|}\right)\right)\right)$ time (Why?).

$$O\left(\log\left(\frac{n}{|E|}\right)\right) \quad < \quad O\left(\frac{n}{|E|}\right)$$
$$O\left(|E|L\log\left(\frac{n}{|E|}\right)\right) \quad < \quad O(nL)$$

[VLDB21a] T. N. Chan, Z. Li, L. H. U, J. Xu, R. Cheng. Fast Augmentation Algorithms for Network Kernel Density Visualization. VLDB 2021.

# Software Development of KDV and its Variants

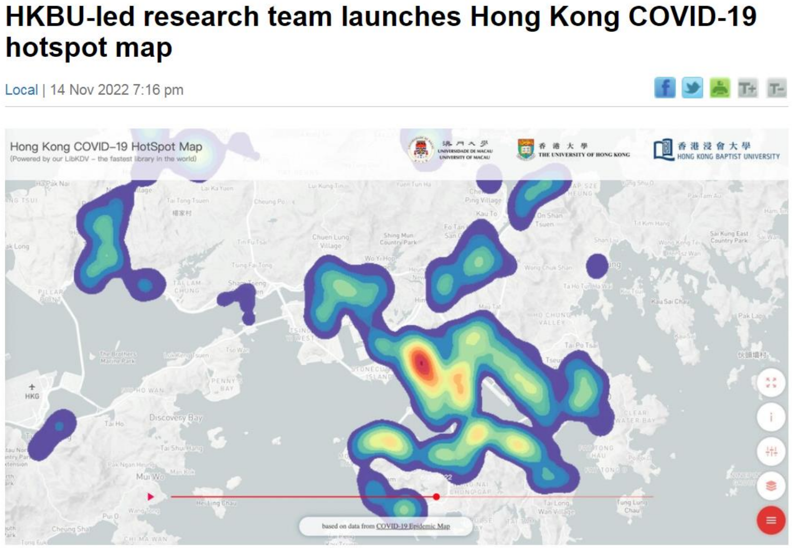
- KDV-Explorer (an online system for KDV) [**VLDB21b**]
- LIBKDV (a python library for KDV and STKDV) [**VLDB22c**]
- PyNKDV (a python library for NKDV) [**SIGMOD23**]

[VLDB21b] T. N. Chan, P. L. Ip, L. H. U, W. H. Tong, S. Mittal, Y. Li, R. Cheng. KDV-Explorer: A Near Real-Time Kernel Density Visualization System for Spatial Analysis. VLDB 2021.
[VLDB22c] T. N. Chan, P. L. Ip, K. Zhao, L. H. U, B. Choi, J. Xu. LIBKDV: A Versatile Kernel Density Visualization Library for Geospatial Analytics. VLDB 2022.
[SIGMOD23] T. N. Chan, R. Zang, P. L. Ip, L. H. U, J. Xu. PyNKDV: An Efficient Network Kernel Density Visualization Library for Geospatial Analytic Systems. SIGMOD 2023.

# Software Development of KDV and its Variants

- [Hong Kong COVID-19 hotspot map]() (based on LIBKDV and KDV-Explorer)

- [Macau COVID-19 hotspot map]() (based on LIBKDV and KDV-Explorer)




# K-function

# State-of-the-art Solutions for Computing K-function

- Range-query-based methods [Springer08, UAI00, ACM75]

- Parallel/distributed and hardware-based methods [IJGIS16, IJGIS15]

[IJGIS16] G. Zhang, Q. Huang, A. X. Zhu, J. H. Keel. 2016. Enabling Point Pattern Analysis on Spatial Big Data using Cloud Computing: Optimizing and Accelerating Ripley's K function. International Journal of Geographical Information Science 2016.

[IJGIS15] W. Tang, W. Feng, M. Jia. Massively Parallel Spatial Point Pattern Analysis: Ripley's K function Accelerated using Graphics Processing Units. International Journal of Geographical Information Science 2015.

[Springer08] M. Berg, O. Cheong, M. J. Kreveld, and M. H. Overmars. Computational Geometry: Algorithms and Applications, 3rd Edition. Springer 2008.

[UAI00] A. W. Moore. The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data. UAI 2000.

[ACM75] J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. Commun. ACM 1975.

# Range-Query-based Methods

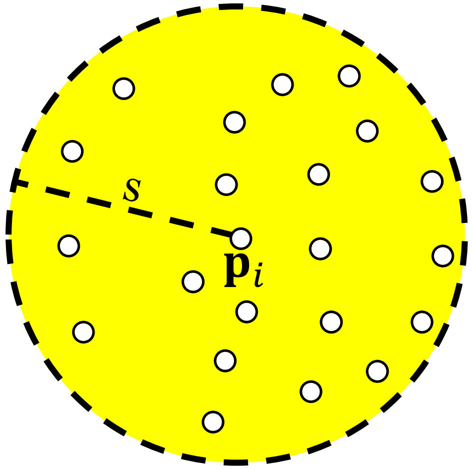
- Consider the K-function.

$$K_P(s) = \sum_{\mathbf{p}_i \in P} \sum_{\substack{\mathbf{p}_j \in P \\ \mathbf{p}_j \neq \mathbf{p}_i}} \mathbb{1}(dist(\mathbf{p}_i, \mathbf{p}_j) \leq s) \quad \text{where } \mathbb{1}(dist(\mathbf{p}_i, \mathbf{p}_j) \leq s) = \begin{cases} 1 & \text{if } dist(\mathbf{p}_i, \mathbf{p}_j) \leq s \\ 0 & \text{otherwise} \end{cases}$$

- Only those white data points that are within the spatial threshold $s$ (i.e., $R(\mathbf{p}_i)$) can contribute to $K_P(s)$.

$$R(\mathbf{p}_i) = \{\mathbf{p}_j \in P: dist(\mathbf{p}_i, \mathbf{p}_j) \leq s, \mathbf{p}_j \neq \mathbf{p}_i\}$$

$$K_P(s) = \sum_{\mathbf{p}_i \in P} |R(\mathbf{p}_i)|$$




$R(\mathbf{p}_i)$

# Range-Query-based Methods

- Many index structures can be adopted for improving the efficiency of finding $R(\mathbf{p}_i)$.
  - kd-tree [ACM75]
  - Ball-tree [UAI00]
  - Range-tree [Springer08]

[Springer08] M. Berg, O. Cheong, M. J. Kreveld, and M. H. Overmars. Computational Geometry: Algorithms and Applications, 3rd Edition. Springer 2008.
[UAI00] A. W. Moore. The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data. UAI 2000.
[ACM75] J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. Commun. ACM 1975.

# Advantages and Disadvantages of Range-Query-based Methods

- Advantages ☺
  - Can practically improve the efficiency for computing K-function.
  - Many index structures are available for improving the efficiency of computing $R(\mathbf{p}_i)$.
  - Can achieve exact solution.

- Disadvantages ☹
  - Cannot reduce the worst-case time complexity for computing K-function (remains in $O(n^2)$ time).
  - Do not investigate the optimization opportunity for computing multiple K-functions (generating K-function plot).

# Parallel/Distributed and Hardware-based Methods

- Aim to assign computations into different computers/GPUs/threads.

- Based on the naïve implementation of K-function.

# Advantages and Disadvantages of Parallel/Distributed and Hardware-based Methods

- Advantages ☺
  - Significantly improve the efficiency of K-function, given many resources.
  - Simple (No new algorithm)
  - Can retain exact results.

- Disadvantages ☹
  - Domain experts may not have enough computational resources (32 CPUs and 96 GPUs are used in [IJGIS15]).
  - Can still not be scalable for large-scale datasets.
  - Cannot reduce the time complexity of this problem.

[IJGIS15] W. Tang, W. Feng, M. Jia. Massively Parallel Spatial Point Pattern Analysis: Ripley's K function Accelerated using Graphics Processing Units. International Journal of Geographical Information Science 2015.

# Variant 1: Spatiotemporal K-function

- Many geographical events (e.g., COVID-19 cases) depend on both space and time.

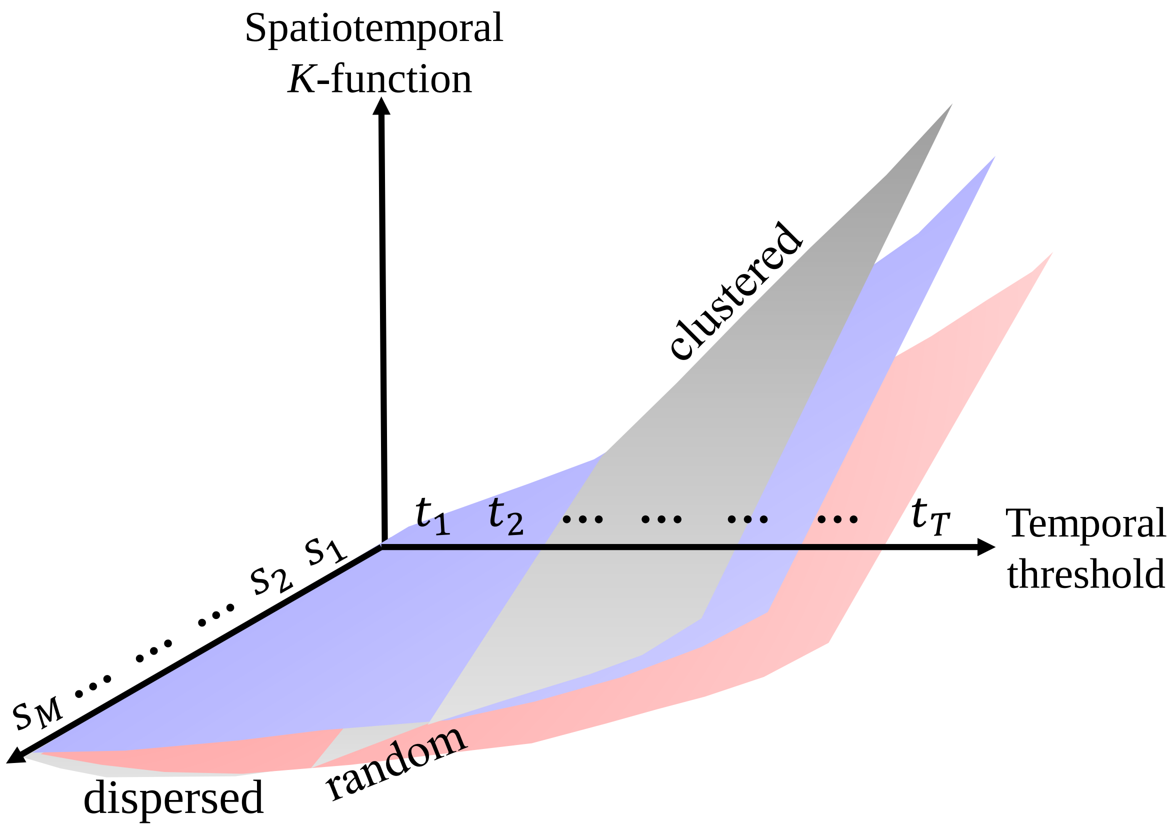
- Domain experts need to understand the spatiotemporal cluster properties of a location dataset.

- Given a location dataset $\hat{P} = \{(\mathbf{p}_1, t_{\mathbf{p}_1}), (\mathbf{p}_2, t_{\mathbf{p}_2}), \dots, (\mathbf{p}_n, t_{\mathbf{p}_n})\}$ with size $n$, the spatial threshold $s$, and the temporal threshold $t$, the spatiotemporal K-function is:

$$K_{\hat{P}}(s,t) = \sum_{(\mathbf{p}_i, t_{\mathbf{p}_i}) \in \hat{P}} \sum_{\substack{(\mathbf{p}_j, t_{\mathbf{p}_j}) \in \hat{P} \\ j \neq i}} \mathbb{I}(dist(\mathbf{p}_i, \mathbf{p}_j) \leq s, dist(t_{\mathbf{p}_i}, t_{\mathbf{p}_j}) \leq t)$$

# Variant 1: Spatiotemporal K-function

- Generate a spatiotemporal K-function plot.



# Variant 1: Spatiotemporal K-function

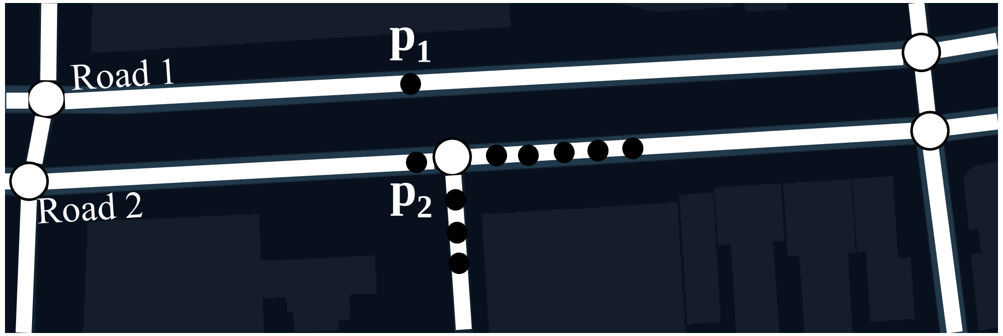
- The naïve solution for computing spatiotemporal K-function is $O(n^2)$ ☹

- The naïve solution for generating spatiotemporal K-function plot is $O(LMTn^2)$ ☹
  - $L$ is the number of random datasets.
  - $M$ is the number of spatial thresholds.
  - $T$ is the number of temporal thresholds.

- There is no complexity-reduced solution for supporting spatiotemporal K-function and generating spatiotemporal K-function plot ☹

# Variant 2: Network K-function

- Many geographical events (e.g., traffic accidents) may be in/along with a road network.




- Two data points, which are close to each other in terms of Euclidean distance, may be far away from each other in a road network.

- Domain experts propose to adopt the network K-function.

$$K_P(s) = \sum_{\mathbf{p}_i \in P} \sum_{\substack{\mathbf{p}_j \in P \\ \mathbf{p}_j \neq \mathbf{p}_i}} \mathbb{I}(dist_G(\mathbf{p}_i, \mathbf{p}_j) \leq s) \text{ where } \mathbb{I}(dist_G(\mathbf{p}_i, \mathbf{p}_j) \leq s) = \begin{cases} 1 & \text{if } dist_G(\mathbf{p}_i, \mathbf{p}_j) \leq s \\ 0 & \text{otherwise} \end{cases}$$

# Variant 2: Network K-function

- The naïve solution for computing network K-function is $O(n(T_{SP} + n))$ ☹

- The naïve solution for computing network K-function plot is $O(LDn(T_{SP} + n))$ ☹

- The best solution for computing network K-function is $O(|E|T_{SP} + n|E| + n \log n)$ [**VLDB22d**] ☺

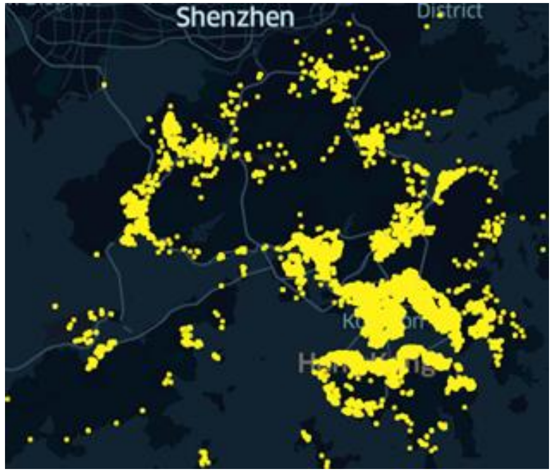
- The best solution for generating network K-function plot is $O(|E|T_{SP} + nLD|E| + Ln \log n)$ [**VLDB22d**] ☺

[VLDB22d] T. N. Chan, L. H. U, Y. Peng, B. Choi, J. Xu. Fast Network K-function-based Spatial Analysis. VLDB 2022.

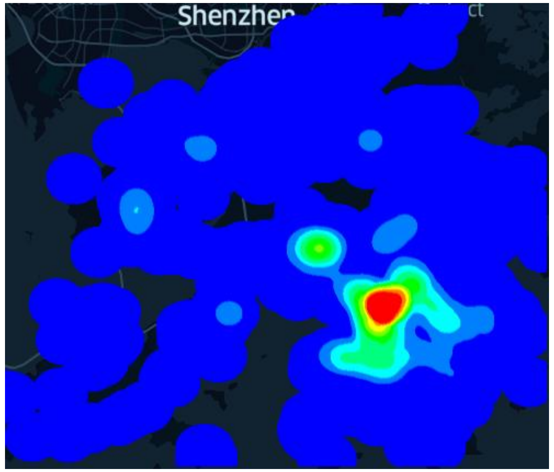
# Future Opportunities

# KDV and its Variants

- The time complexity of the state-of-the-art method for generating KDV is $O(Y(X+n))$.
- The current lower bound time complexity is $O(XY+n)$.
- Can be further achieve the optimal solution for generating KDV?
- This question applies to NKDV and STKDV.



Hong Kong COVID-19 cases


Hotspot map (based on KDV)

# KDV and its Variants

- Complexity-reduced solutions for KDV [**SIGMOD22**], NKDV [**VLDB21a**], and STKDV [**VLDB22b**], can only support polynomial-based kernel functions, which cannot support all kernel functions (e.g., Gaussian kernel).

| Kernel | $\mathcal{K}(\mathbf{q}, \mathbf{p})$ |
|---|---|
| Uniform | $\begin{cases} \frac{1}{b} & \text{if } dist(\mathbf{q}, \mathbf{p}) \leq b \\ 0 & \text{otherwise} \end{cases}$ |
| Epanechnikov | $\begin{cases} 1 - \frac{1}{b^2} dist(\mathbf{q}, \mathbf{p})^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \leq b \\ 0 & \text{otherwise} \end{cases}$ |
| Quartic | $\begin{cases} \left(1 - \frac{1}{b^2} dist(\mathbf{q}, \mathbf{p})^2\right)^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \leq b \\ 0 & \text{otherwise} \end{cases}$ |
| Gaussian | $\exp\left(-\frac{1}{b^2} dist(\mathbf{q}, \mathbf{p})^2\right)$ |

- Can we develop complexity-reduced algorithms for generating KDV with all kernel functions with non-trivial accuracy guarantees?

[SIGMOD22] T. N. Chan, L. H. U, B. Choi, J. Xu. SLAM: Efficient Sweep Line Algorithms for Kernel Density Visualization. SIGMOD 2022.
[VLDB22b] T. N. Chan, P. L. Ip, L. H. U, B. Choi, J. Xu. SWS: A Complexity-Optimized Solution for Spatial-Temporal Kernel Density Visualization. VLDB 2022.
[VLDB21a] T. N. Chan, Z. Li, L. H. U, J. Xu, R. Cheng. Fast Augmentation Algorithms for Network Kernel Density Visualization. VLDB 2021.

# K-function and its Variants

- There is no advanced solution for improving the efficiency of computing K-function and spatiotemporal K-function.
  - Remain in $O(n^2)$ time.
  - Cannot be scalable to support the K-function plot and spatiotemporal K-function plot.

- Can we develop complexity-reduced algorithms for supporting these tools with exact guarantees?

- Can we further develop optimal solutions for all K-function-based tools?

# K-function and its Variants

- No approximation solution has been proposed for these tools.

- Many approximation solutions have been proposed for supporting KDV and its variants.
  - Function approximation
  - Data sampling

- Can we extend these techniques for supporting all K-function-based tools with non-trivial accuracy guarantees?

# Other Geospatial Analysis Tools

- No complexity-reduced solution has been developed for supporting other geospatial analysis tools.

- Many complexity-reduced solutions have been developed for generating KDV and its variants.
  - Computational sharing
  - Data sampling

- Can we extend these solutions for supporting other geospatial analysis tools?

# Other Geospatial Analysis Tools

- No researcher has investigated the lower-bound time complexity of these geospatial analysis tools.
- Without this knowledge, it is hard to develop optimal solutions for supporting these geospatial analysis tools.
- Can we tighten the lower-bound time complexity for different geospatial analysis tools?

# Other Geospatial Analysis Tools

- Many parallel/distributed/hardware-based solutions are based on naïve implementation (e.g., [IJGIS15]).
  - Can consume many computational resources ☹
  - Can still be not scalable to large-scale datasets ☹

- Can we combine parallel/distributed/hardware-based approaches with (new) complexity-reduced solutions?

[IJGIS15] W. Tang, W. Feng, M. Jia. Massively Parallel Spatial Point Pattern Analysis: Ripley's K function Accelerated using Graphics Processing Units. International Journal of Geographical Information Science 2015.

# Software Development

- Existing software packages are based on naïve solutions for supporting geospatial analysis tools.

- Goal: Replace all these naïve solutions with efficient solutions.

- Target users:
  - GIS researchers with some basic programming skills: Can call some python and R libraries (e.g., spatstat, spNetwork, and PySAL) for using geospatial analysis tools.
  - Laymen: Only use some well-known GIS software packages with UI (e.g., QGIS, ArcGIS, QGIS Cloud, and ArcGIS Online).

# Software Development

- Can we develop new python and R libraries, based on new solutions, for supporting all geospatial analysis tools?

- Can we develop new QGIS and ArcGIS plugins, based on new solutions, for supporting all geospatial analysis tools?

- Can we integrate new solutions into web-based (online) GIS systems (e.g., QGIS Cloud and ArcGIS Online)?