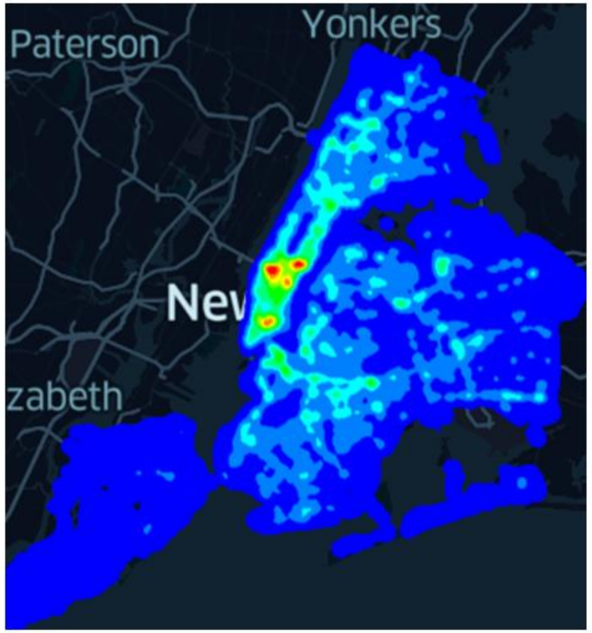
Complexity-Optimized Algorithms for Large-scale Kernel Density Visualization

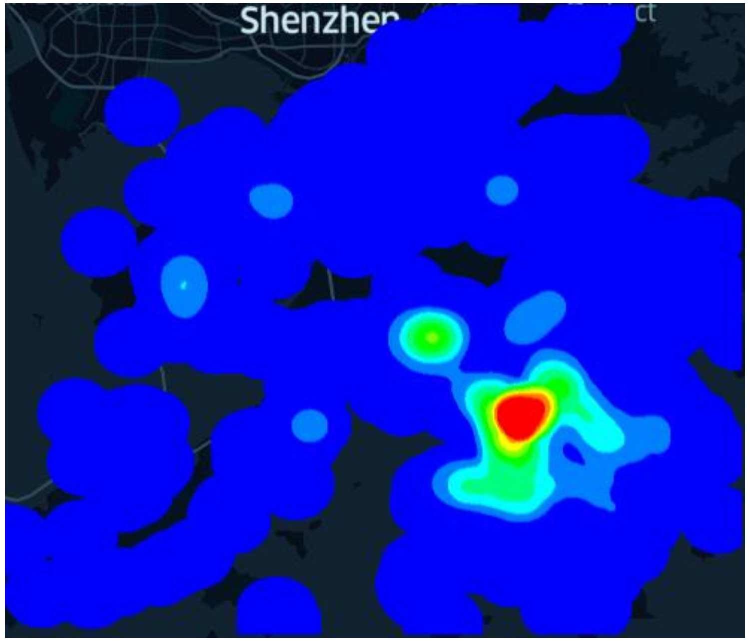
Introduction

Why Data Visualization?

- Find hidden patterns (e.g., hotspots) from a dataset.
- Provide an intuitive way to understand a dataset (by visualizing it).



A heatmap (or hotspot map) for the New York traffic accident dataset

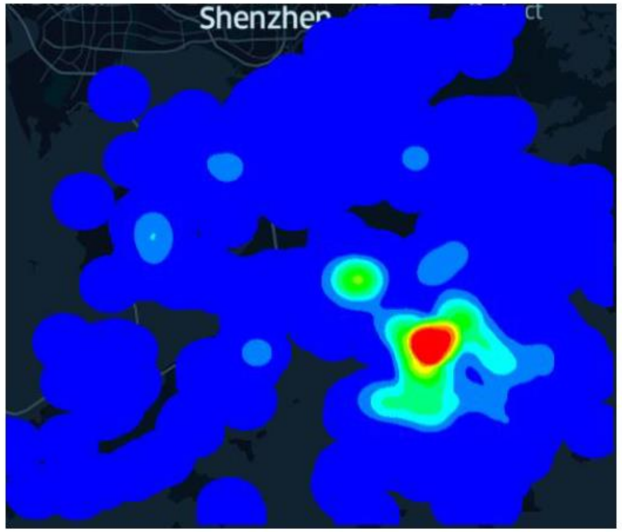


Hong Kong COVID-19 hotspot map

Why Kernel Density Visualization (KDV)?

- KDV can provide better visualization quality compared with many traditional software tools.

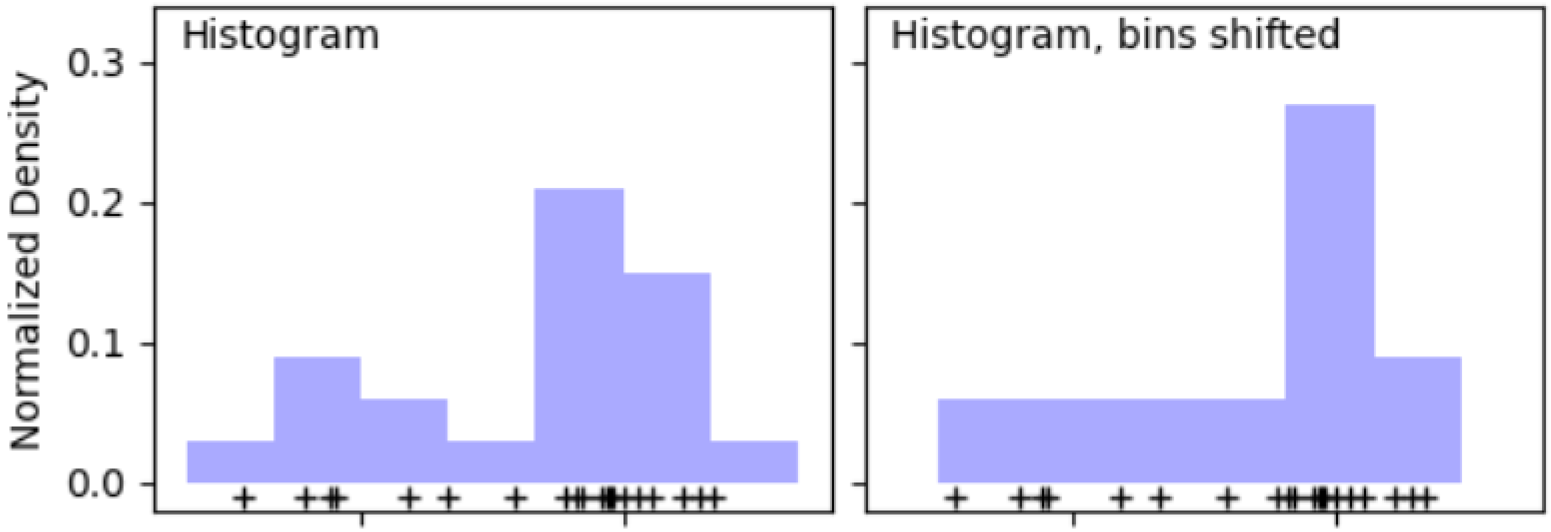
Scatter plot of
Hong Kong COVID-19 cases



Hotspot map of Hong Kong
COVID-19 cases (based on KDV)

Why Kernel Density Visualization (KDV)?

- KDV can provide better visualization quality compared with many traditional software tools.



A major problem with histograms, however, is that the choice of binning can have a disproportionate effect on the resulting visualization. Consider the upper-right panel of the above figure. It shows a histogram over the same data, with the bins shifted right. The results of the two visualizations look entirely different, and might lead to different interpretations of the data.

Why Kernel Density Visualization (KDV)?

- A de facto method
- Supported by many software packages

2.8.2. Kernel Density Estimation

Kernel density estimation in scikit-learn is implemented in the `KernelDensity` estimator, which uses the Ball Tree or KD Tree for efficient queries (see Nearest Neighbors for a discussion of these). Though the above example uses a 1D data set for simplicity, kernel density estimation can be performed in any number of dimensions, though in practice the curse of dimensionality causes its performance to degrade in high dimensions.

In the following figure, 100 points are drawn from a bimodal distribution, and the kernel density estimates are shown for three choices of kernels:

Scikit-learn

How Kernel Density works

ArcGIS Pro 3.0 | Other versions | Help archive

Available with Spatial Analyst license.

The Kernel Density tool calculates the density of features in a neighborhood around those features. It can be calculated for both point and line features.

ArcGIS

scipy.stats.gaussian_kde

class `scipy.stats.gaussian_kde(dataset, bw_method=None, weights=None)` [source]

Representation of a kernel-density estimate using Gaussian kernels.

Kernel density estimation is a way to estimate the probability density function (PDF) of a random variable in a non-parametric way. `gaussian_kde` works for both uni-variate and multi-variate data. It includes automatic bandwidth determination. The estimation works best for a unimodal distribution; bimodal or multi-modal distributions tend to be oversmoothed.

Parameters: **dataset** : *array_like*

Datapoints to estimate from. In case of univariate data this is a 1-D array, otherwise a 2-D array with shape (# of dims, # of data).

bw_method : *str, scalar or callable, optional*

The method used to calculate the estimator bandwidth. This can be 'scott', 'silverman', a scalar constant or a callable. If a scalar, this will be used directly as *kde.factor*. If a callable, it should take a `gaussian_kde` instance as only parameter and return a scalar. If None (default), 'scott' is used. See Notes for more details.

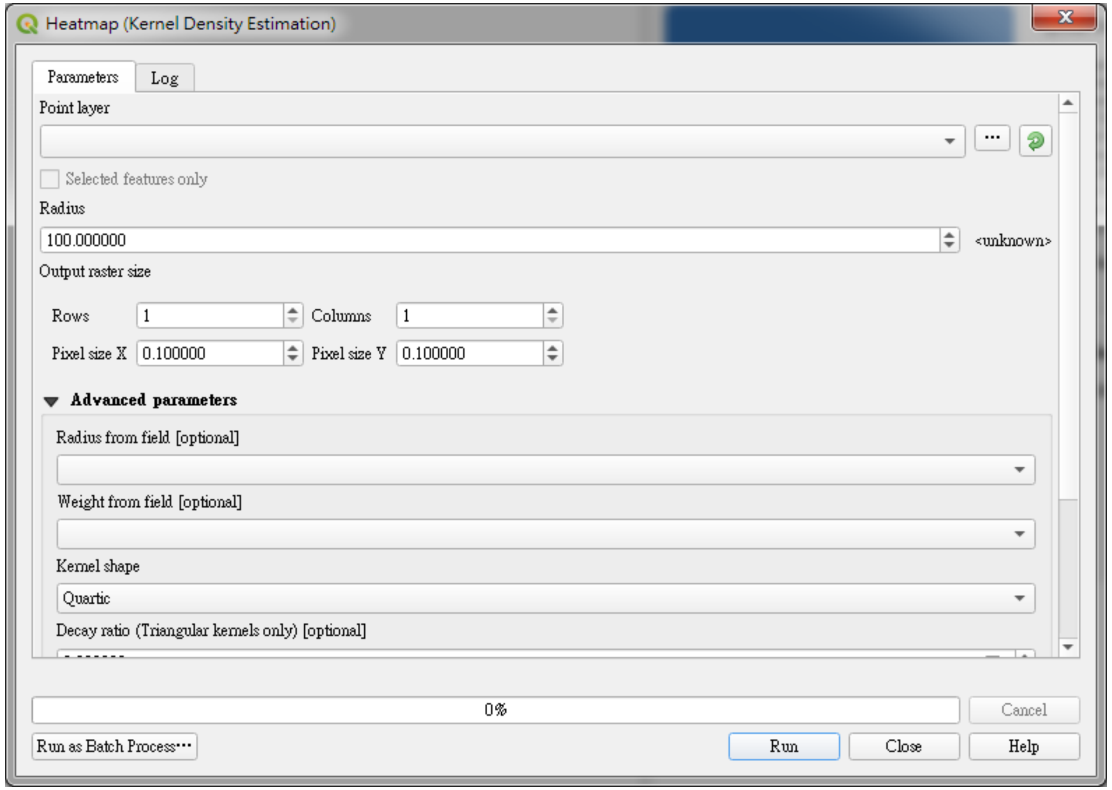
weights : *array_like, optional*

weights of datapoints. This must be the same shape as dataset. If None (default), the samples are assumed to be equally weighted

Scipy

Why Kernel Density Visualization (KDV)?

- A de facto method
- Supported by many software packages



QGIS

HeatmapLayer

`HeatmapLayer` can be used to visualize spatial distribution of data. It internally implements Gaussian Kernel Density Estimation to render heatmaps. Note that this layer does not support all platforms; see "limitations" section below.

Deck.gl

seaborn.kdeplot

```
seaborn.kdeplot(data=None, *, x=None, y=None, hue=None, weights=None, palette=None, hue_order=None, hue_norm=None, color=None, fill=None, multiple='layer', common_norm=True, common_grid=False, cumulative=False, bw_method='scott', bw_adjust=1, warn_singular=True, log_scale=None, levels=10, thresh=0.05, gridsize=200, cut=3, clip=None, legend=True, cbar=False, cbar_ax=None, cbar_kws=None, ax=None, **kwargs)
```

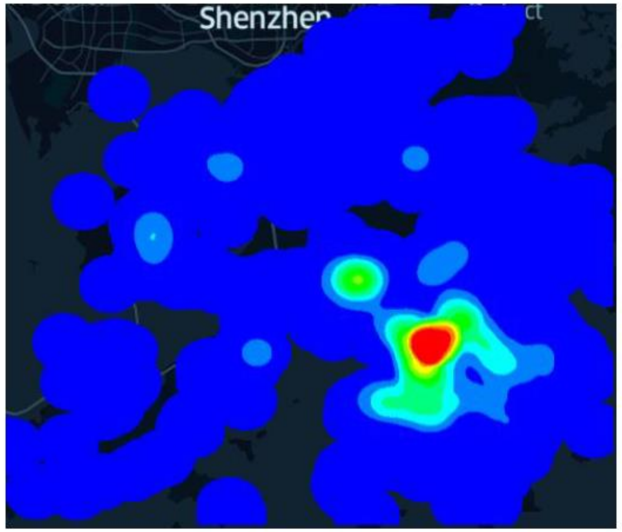
Plot univariate or bivariate distributions using kernel density estimation.

A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

Seaborn

Kernel Density Visualization (KDV)

What is KDV?



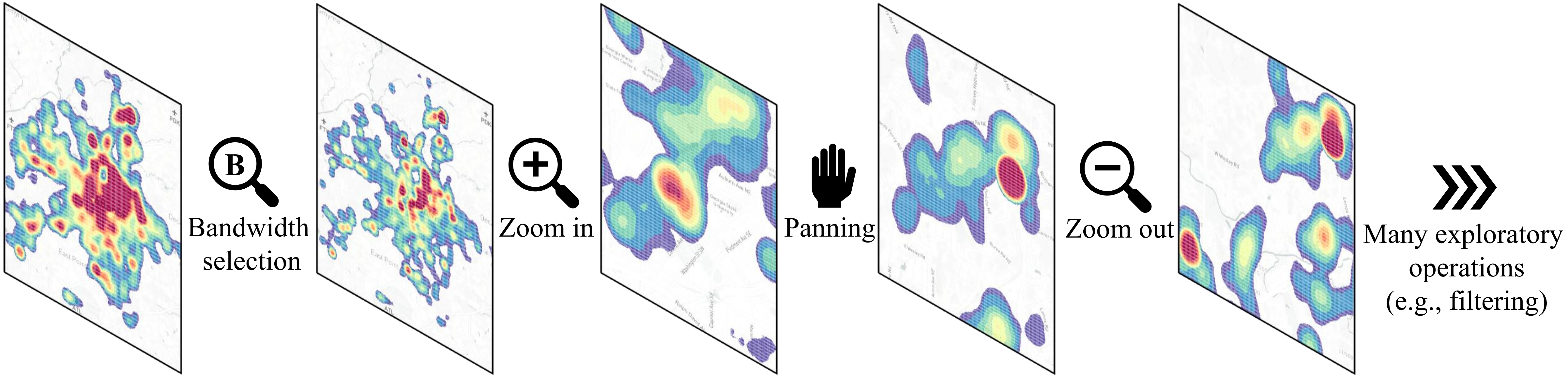
- Each **p** (yellow dot) represents the location of a COVID-19 case.
- Predict the risk of a given location **q** by computing the ***kernel density function*** $\mathcal{F}_P(\mathbf{q})$.

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

KDV is Slow!

- Time complexity: $O(XYn)$
  - Resolution size: $X \times Y$
  - Number of data points in P: n

- Domain experts need to generate multiple KDVs.

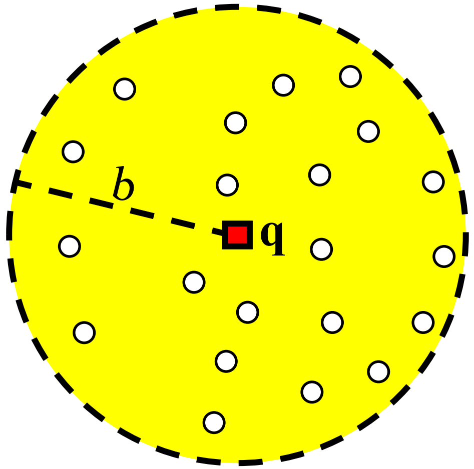


KDV is Slow!

- Complaints:
  - Gan et al. [1] "...the total runtime cost of density estimation is quadratic in dataset size..."
  - Gramacki et al. [2] "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."

[1] Edward Gan and Peter Bailis. Scalable Kernel Density Classification via Threshold-Based Pruning. In SIGMOD 2017. 945–959.
[2] A. Gramacki. Nonparametric Kernel Density Estimation and Its Computational Aspects. Springer International Publishing 2017.

Range-Query-based Solution



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

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

- Simple ☺
- Many tree structures are available to improve the practical efficiency ☺
- Cannot reduce the worst-case time complexity ($b \rightarrow \infty$) ☹

Our Solution: SLAM

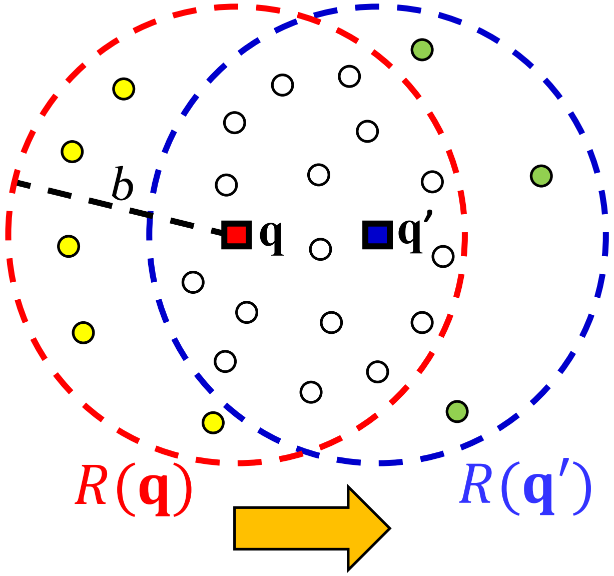
Method	Time complexity	Space complexity
RQS (cf. Section 2.2)	$O(XYn)$	$O(XY + n)$
$\text{SLAM}_{\text{SORT}}$ (cf. Section 3.4)	$O(Y(X + n \log n))$ (cf. Theorem 1)	$O(XY + n)$ (cf. Theorem 4)
$\text{SLAM}_{\text{BUCKET}}$ (cf. Section 3.5)	$O(Y(X + n))$ (cf. Theorem 2)	
$\text{SLAM}^{(\text{RAO})}_{\text{SORT}}$ (cf. Sections 3.4 and 3.6)	$O(\min(X, Y) \times (\max(X, Y) + n \log n))$ (cf. Theorem 3)	
$\text{SLAM}^{(\text{RAO})}_{\text{BUCKET}}$ (cf. Sections 3.5 and 3.6)	$O(\min(X, Y) \times (\max(X, Y) + n))$ (cf. Theorem 3)	

- The first work for generating a single KDV that can:
  - Reduce the worst-case time complexity ☺
  - Retain the same space complexity ☺

- Achieve one to two-order-of-magnitude speedup for supporting KDV, using large-scale datasets ☺

Core Ideas of SLAM

- Core idea 1: two consecutive pixels can share many data points (white circles) in the range set.



Can we share computations between two consecutive pixels?

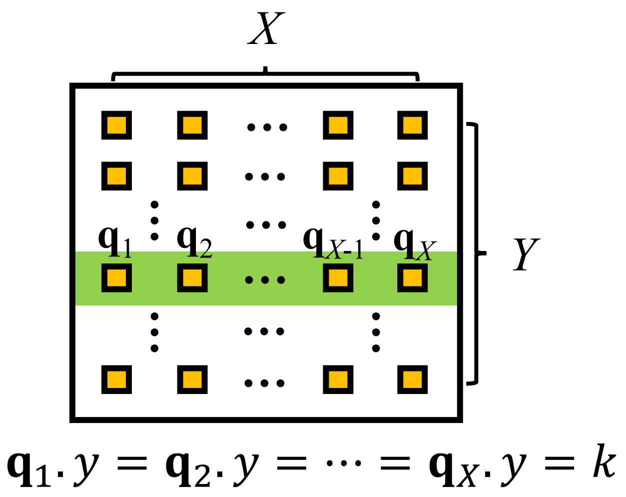
- Core idea 2: $\mathcal{F}_P(\mathbf{q})$ can be decomposed into this expression.

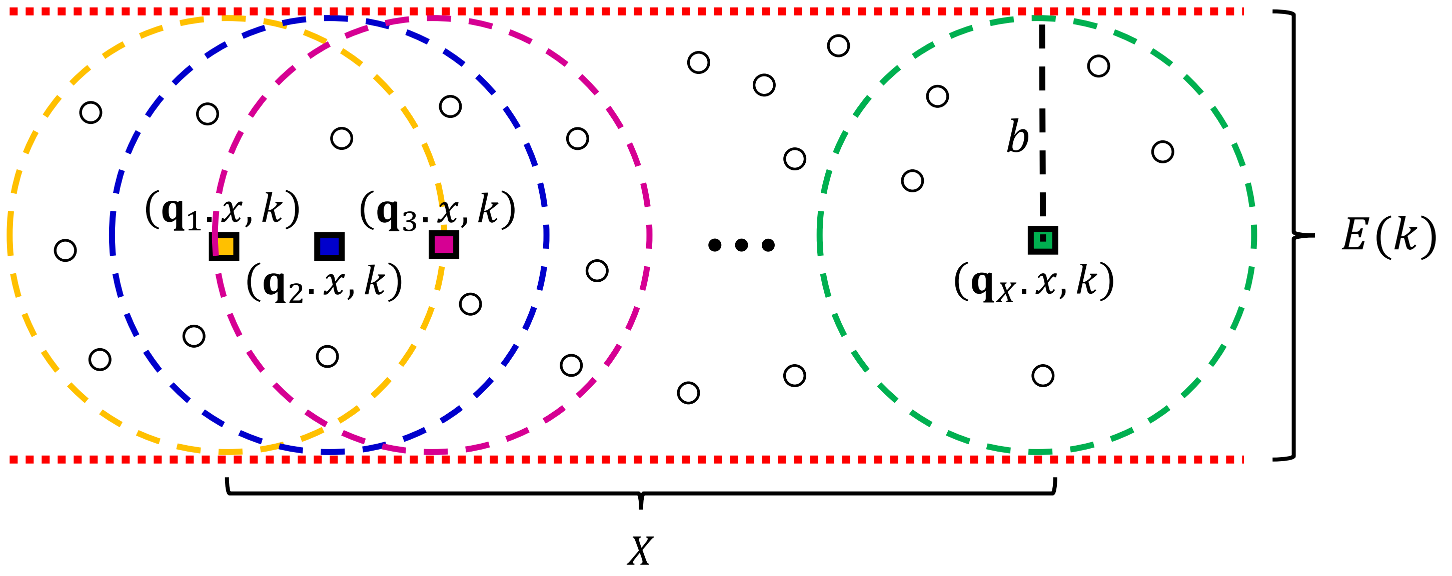
$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p}\in R(\mathbf{q})} w \cdot \left(1 - \frac{1}{b^2} dist(\mathbf{q},\mathbf{p})^2\right)$$

$$= w|R(\mathbf{q})| - \frac{w}{b^2}\left(|R(\mathbf{q})| \times \|\mathbf{q}\|_2^2 - 2\mathbf{q}^T \underbrace{\mathbf{A}_{R_\mathbf{q}}}_{\sum_{\mathbf{p}\in R(\mathbf{q})} \mathbf{p}} + \underbrace{S_{R_\mathbf{q}}}_{\sum_{\mathbf{p}\in R(\mathbf{q})} \|\mathbf{p}\|_2^2}\right)$$

How to efficiently maintain $|R(\mathbf{q})|$, $\mathbf{A}_{R_\mathbf{q}}$, and $S_{R_\mathbf{q}}$?

Envelope for A Row of Pixels

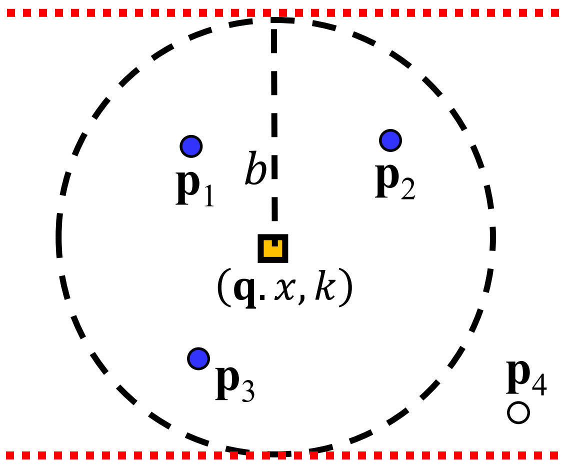




Use $O(n)$ time to find the envelope $E(k)$.

$$E(k) = \{\mathbf{p} \in P : |k - \mathbf{p}.y| \leq b\}$$

Lower and Upper Bound Functions



- Consider the blue data points **p** that are within the range b of the pixel **q**.

$$
\begin{aligned}
dist(\mathbf{q}, \mathbf{p}) &\leq b \\
(\mathbf{q}.x - \mathbf{p}.x)^2 &\leq b^2 - (k - \mathbf{p}.y)^2
\end{aligned}
$$

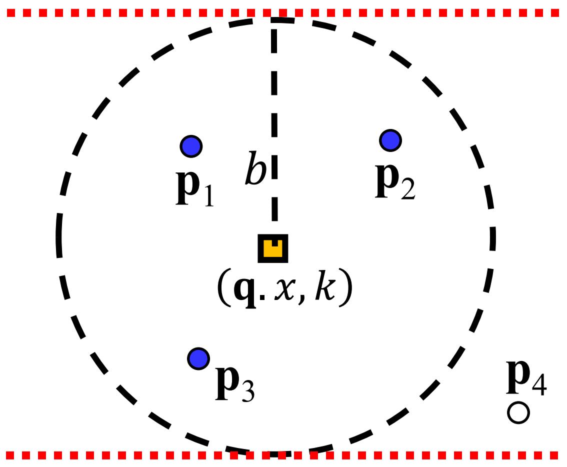
$$\mathbf{p}.x - \sqrt{b^2 - (k - \mathbf{p}.y)^2} \leq \mathbf{q}.x \leq \mathbf{p}.x + \sqrt{b^2 - (k - \mathbf{p}.y)^2}$$

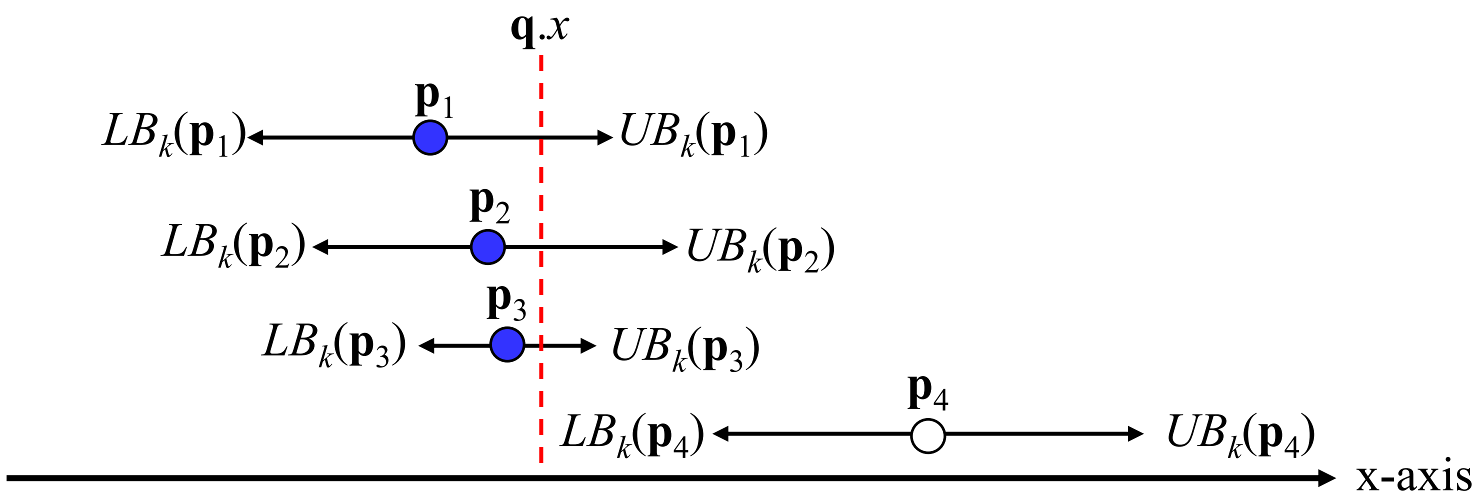
- We can let:

$$
\begin{aligned}
LB_k(\mathbf{p}) &= \mathbf{p}.x - \sqrt{b^2 - (k - \mathbf{p}.y)^2} \\
UB_k(\mathbf{p}) &= \mathbf{p}.x + \sqrt{b^2 - (k - \mathbf{p}.y)^2}
\end{aligned}
$$

- $O(n)$ time to find the bound functions for all data points in $E(k)$.

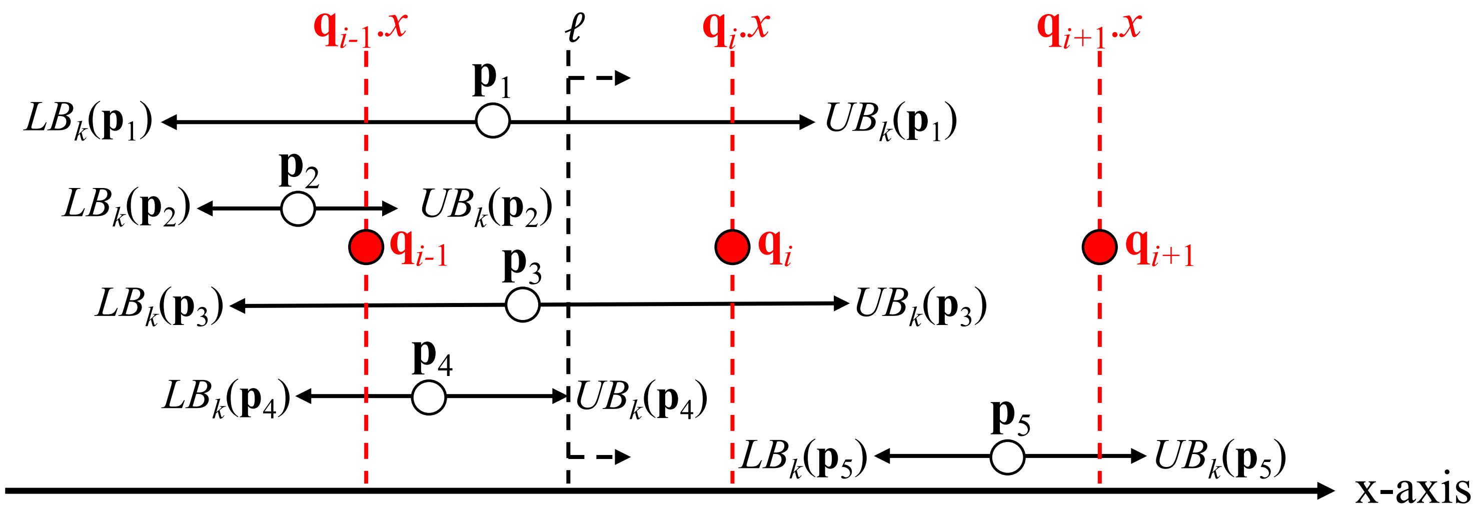
Range Search Problem = Interval Stabbing Problem





LEMMA 2. *Given the lower and upper bound values, i.e., $LB_k(\mathbf{p})$ and $UB_k(\mathbf{p})$, respectively, for each data point $\mathbf{p}$ in the envelope point set $E(k)$, this data point $\mathbf{p}$ is in the range query solution set $R(\mathbf{q})$ if $\mathbf{q}.x$ is within the bound interval $[LB_k(\mathbf{p}), UB_k(\mathbf{p})]$.*

Sorting-based Sweep Line Algorithm ($\text{SLAM}_{\text{SORT}}$)



- Sort the x-coordinates of the end points of all intervals with $O(n \log n)$ time.
- Use the sweep line ℓ to maintain $|R(\mathbf{q})|$, $\mathbf{A}_{R_{\mathbf{q}}}$, and $S_{R_{\mathbf{q}}}$ and calculate $\mathcal{F}_P(\mathbf{q})$ for each pixel $\mathbf{q}$ with $O(X + n)$ time. (Refer to the paper for more details)
- Overall time complexity for processing a row of pixels: $O(X + n \log n)$.
- $\text{SLAM}_{\text{SORT}}$ takes $O(Y(X + n \log n))$ time ☺

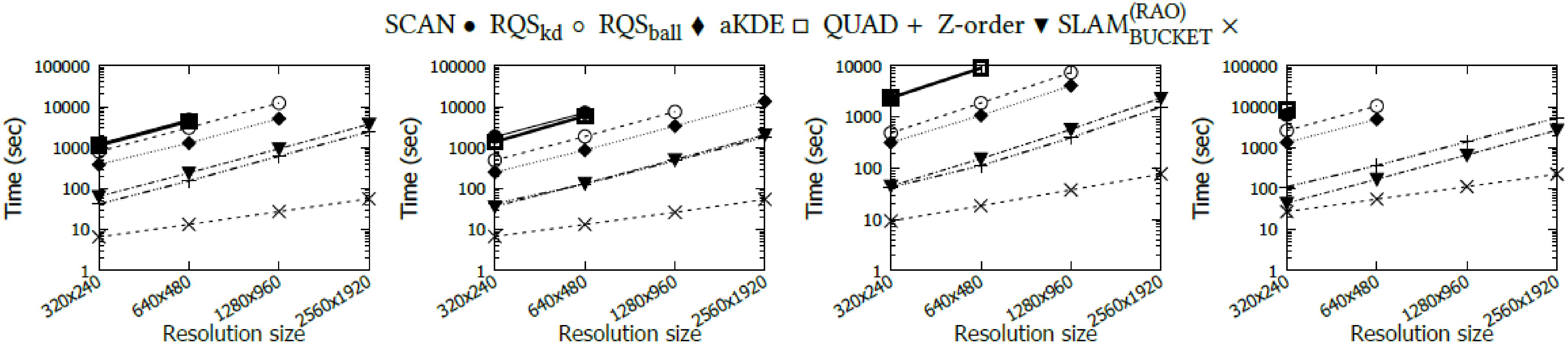
Bucket-based Sweep Line Algorithm ($\mathrm{SLAM}_{\mathrm{BUCKET}}$)

- Can remove the sorting step (How? Refer to the paper [a] for more details).
- $\mathrm{SLAM}_{\mathrm{BUCKET}}$ takes $O(Y(X+n))$ time ☺

[a] Tsz Nam Chan, Leong Hou U, Byron Choi, Jianliang Xu: "SLAM: Efficient Sweep Line Algorithms for Kernel Density Visualization" SIGMOD 2022, pages 2120-2134.

Our Experiment

Dataset name	Dataset size n	Category	Bandwidth b (meters)
Seattle [5]	862873	Crime events	671.39
Los Angeles [2]	1255668	Crime events	1588.47
New York [3]	1499928	Traffic accidents	1062.53
San Francisco [4]	4333098	311 calls	279.27

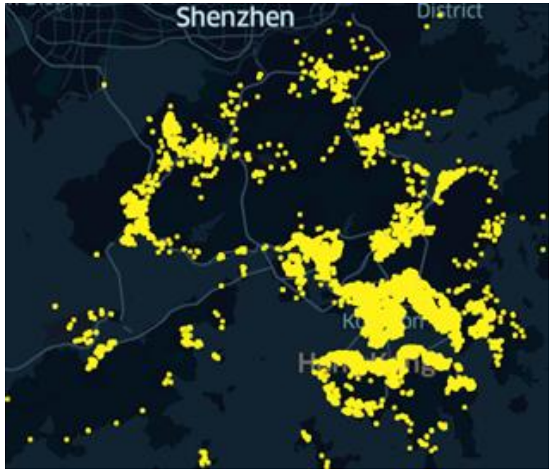


(a) Seattle (b) Los Angeles (c) New York (d) San Francisco

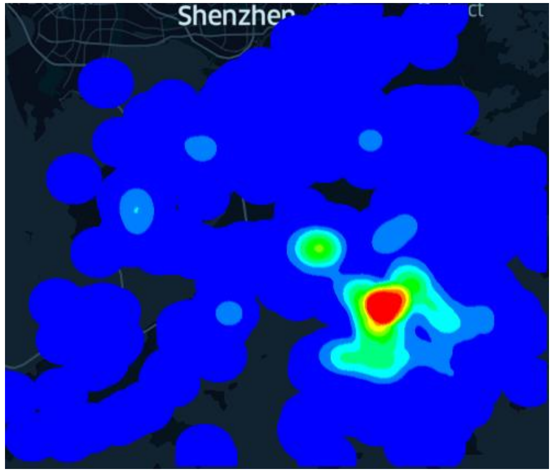
Spatiotemporal Kernel Density Visualization (STKDV)

Weakness of KDV

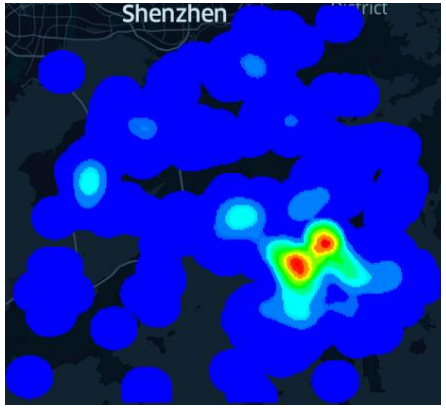
- Does not consider the occurrence time of each geographical event ☹



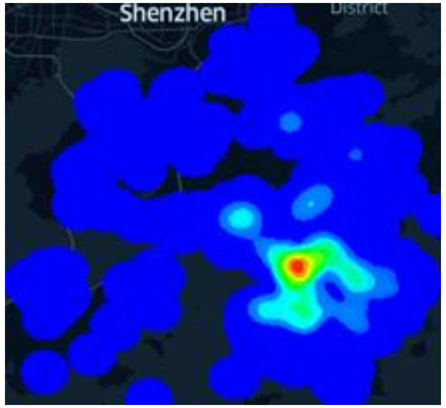
Hong Kong COVID-19 cases



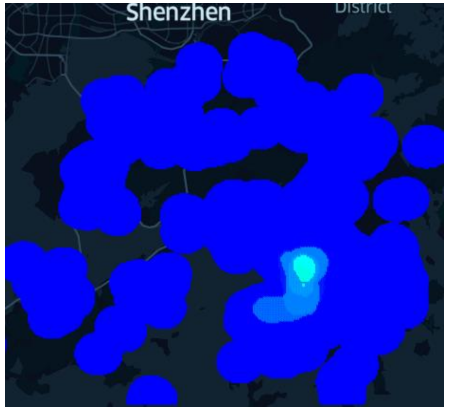
Hotspot map (based on KDV)



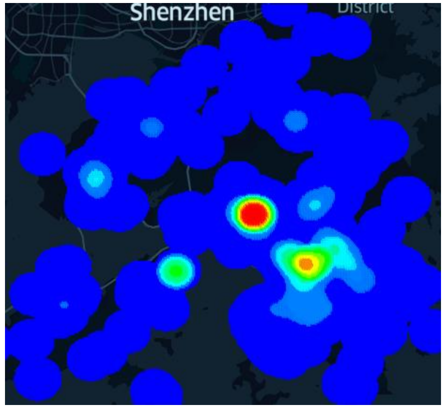
2nd August 2020



6th December 2020

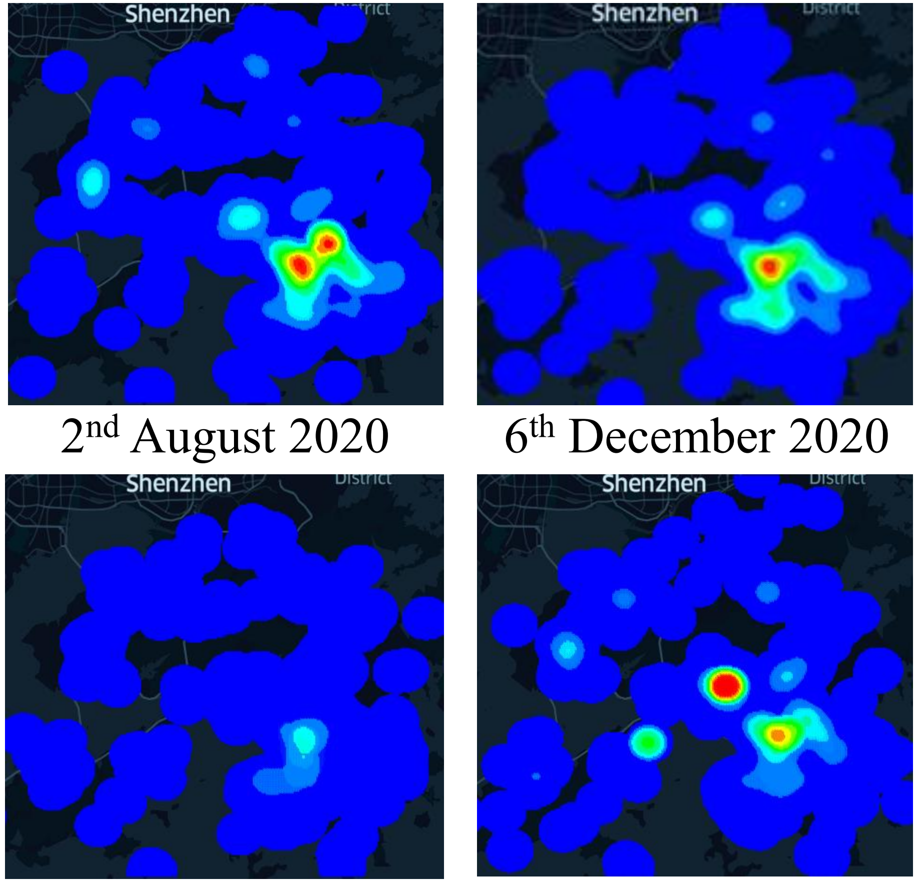


28th February 2021



28th January 2022

Spatial-Temporal Kernel Density Visualization (STKDV)



2nd August 2020 | 6th December 2020 | 28th February 2021 | 28th 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 $\mathcal{F}_{\hat{P}}(\mathbf{q}, t_\mathbf{q})$.

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

Spatial-Temporal Kernel Density Visualization (STKDV)

- Some representative spatial and temporal kernel functions that are used in the spatial-temporal kernel density function $\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}})$.

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

Kernel	$K_{\text{space}}(\mathbf{q}, \mathbf{p})$	$K_{\text{time}}(t_{\mathbf{q}}, t_{\mathbf{p}})$
Triangular	$\begin{cases} 1 - \gamma_s dist(\mathbf{q}, \mathbf{p}) & \text{if } dist(\mathbf{q}, \mathbf{p}) \leq \frac{1}{\gamma_s} \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 - \gamma_t dist(t_{\mathbf{q}}, t_{\mathbf{p}}) & \text{if } dist(t_{\mathbf{q}}, t_{\mathbf{p}}) \leq \frac{1}{\gamma_t} \\ 0 & \text{otherwise} \end{cases}$
Epanechnikov	$\begin{cases} 1 - \gamma_s^2 dist(\mathbf{q}, \mathbf{p})^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \leq \frac{1}{\gamma_s} \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 - \gamma_t^2 dist(t_{\mathbf{q}}, t_{\mathbf{p}})^2 & \text{if } dist(t_{\mathbf{q}}, t_{\mathbf{p}}) \leq \frac{1}{\gamma_t} \\ 0 & \text{otherwise} \end{cases}$
Quartic	$\begin{cases} (1 - \gamma_s^2 dist(\mathbf{q}, \mathbf{p})^2)^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \leq \frac{1}{\gamma_s} \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} (1 - \gamma_t^2 dist(t_{\mathbf{q}}, t_{\mathbf{p}})^2)^2 & \text{if } dist(t_{\mathbf{q}}, t_{\mathbf{p}}) \leq \frac{1}{\gamma_t} \\ 0 & \text{otherwise} \end{cases}$

STKDV is Slow!

- The time complexity of a naïve solution for generating STKDV is $O(XYTn)$ ☹
- Example:
  - The resolution size ($X \times Y$): 128×128
  - The number of timestamps (T): 128
  - The total number of data points (n): 1,000,000
  - The total cost is: **2.09 trillion operations** ☹

Many Complaints from Domain Experts

- Delmelle et al. [1] “Expanding the KDE algorithm to integrate the temporal dimension is **computationally demanding**…”

- Hohl et al. [2] “The temporal extension of the KDE is known as the space-time kernel density estimation (STKDE) and essentially maps a volume of disease intensity along the space-time domain (Nakaya and Yano 2010). However, the above methods are **computationally intensive**…”

[1] Eric Delmelle, Coline Dony, Irene Casas, Meijuan Jia, and Wenwu Tang. 2014. Visualizing the impact of space-time uncertainties on dengue fever patterns. International Journal of Geographical Information Science 28, 5 (2014), 1107–1127.
[2] Alexander Hohl, Eric Delmelle, Wenwu Tang, and Irene Casas. 2016. Accelerating the discovery of space-time patterns of infectious diseases using parallel computing. Spatial and Spatio-temporal Epidemiology 19 (2016), 10 – 20.

Range-Query-based Solution (RQS)

- Recall that:

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

where (with the Epanechnikov kernel):

$$K_{\text{space}}(\mathbf{q}, \mathbf{p}) = \begin{cases} 1 - \gamma_s^2 dist(\mathbf{q}, \mathbf{p})^2 & \text{if } dist(\mathbf{q}, \mathbf{p}) \leq \frac{1}{\gamma_s} \\ 0 & \text{otherwise} \end{cases}$$

$$K_{\text{time}}(t_{\mathbf{q}}, t_{\mathbf{p}}) = \begin{cases} 1 - \gamma_t^2 dist(t_{\mathbf{q}}, t_{\mathbf{p}})^2 & \text{if } dist(t_{\mathbf{q}}, t_{\mathbf{p}}) \leq \frac{1}{\gamma_t} \\ 0 & \text{otherwise} \end{cases}$$

- Only those data points $(\mathbf{p}, t_{\mathbf{p}})$ with $dist(\mathbf{q}, \mathbf{p}) \leq \frac{1}{\gamma_s}$ and $dist(t_{\mathbf{q}}, t_{\mathbf{p}}) \leq \frac{1}{\gamma_t}$ can contribute to $\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}})$.

Range-Query-based Solution (RQS)

- Step 1: Find the range-query set $R_{\mathbf{q}}$ for the pixel $\mathbf{q}$ with the timestamp $t_{\mathbf{q}}$.

$$R_{\mathbf{q}} = \left\{ (\mathbf{p}, t_{\mathbf{p}}) \in \hat{P} \;\middle|\; dist(\mathbf{q}, \mathbf{p}) \leq \frac{1}{\gamma_s} \text{ and } dist(t_{\mathbf{q}}, t_{\mathbf{p}}) \leq \frac{1}{\gamma_t} \right\}$$

- Step 2: Compute $\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}})$ based on $R_{\mathbf{q}}$.

$$\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}}) = \sum_{(\mathbf{p}, t_{\mathbf{p}}) \in R_{\mathbf{q}}} w \cdot (1 - \gamma_s^2 dist(\mathbf{q}, \mathbf{p})^2) \cdot (1 - \gamma_t^2 dist(t_{\mathbf{q}}, t_{\mathbf{p}})^2)$$

- Many index structures can be adopted to improve the practical efficiency for generating STKDV ☺
  - kd-tree
  - ball-tree
- Cannot reduce the time complexity for generating STKDV (remains in $O(XYTn)$ time) ☹

Our Solution: SWS

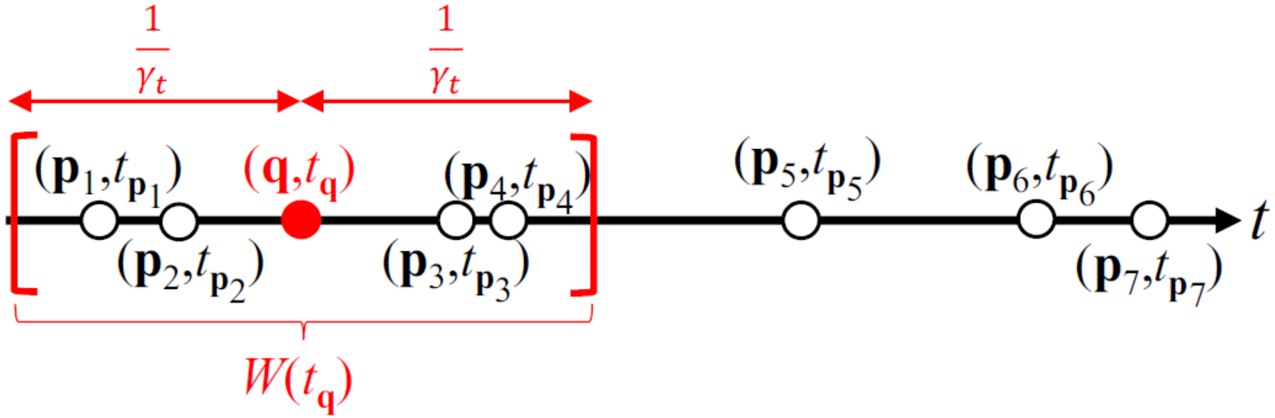
- Theory: Reduce the time complexity for generating exact STKDV, without increasing the space complexity ☺

Method	Time complexity	Space complexity
SCAN	$O(XYTn)$	$O(XYT+n)$
RQS_{kd}		
RQS_{ball}		
SWS	$O(XY(T+n))$	

- Practice: Achieve 1.71x to 24x speedup compared with the state-of-the-art method (RQS) ☺

Core Idea 1 of SWS: Sliding Window

- Establish the sliding window in the temporal dimension.



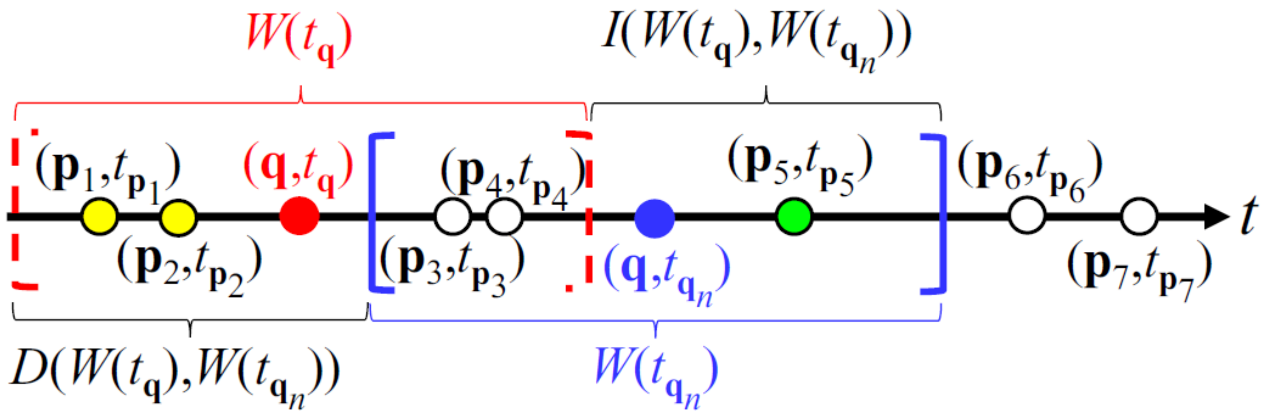
- Only $(\mathbf{p}_1, t_{\mathbf{p}_1})$, $(\mathbf{p}_2, t_{\mathbf{p}_2})$, $(\mathbf{p}_3, t_{\mathbf{p}_3})$, and $(\mathbf{p}_4, t_{\mathbf{p}_4})$ can contribute to $\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}})$.

$$\mathcal{F}_{\hat{P}}(\mathbf{q}, t_{\mathbf{q}}) = \sum_{(\mathbf{p}, t_{\mathbf{p}}) \in W(t_{\mathbf{q}})} w \cdot K_{\text{space}}(\mathbf{q}, \mathbf{p}) \cdot \left(1 - \gamma_t^2 dist(t_{\mathbf{q}}, t_{\mathbf{p}})^2\right)$$

$$= w\left(1 - \gamma_t^2 t_{\mathbf{q}}^2\right) \cdot S_{W(t_{\mathbf{q}})}^{(0)}(\mathbf{q}) + 2w\gamma_t^2 t_{\mathbf{q}} \cdot S_{W(t_{\mathbf{q}})}^{(1)}(\mathbf{q}) - w\gamma_t^2 \cdot S_{W(t_{\mathbf{q}})}^{(2)}(\mathbf{q})$$

$$S_{W(t_{\mathbf{q}})}^{(i)}(\mathbf{q}) = \sum_{(\mathbf{p}, t_{\mathbf{p}}) \in W(t_{\mathbf{q}})} t_{\mathbf{p}}^i \cdot K_{\text{space}}(\mathbf{q}, \mathbf{p})$$

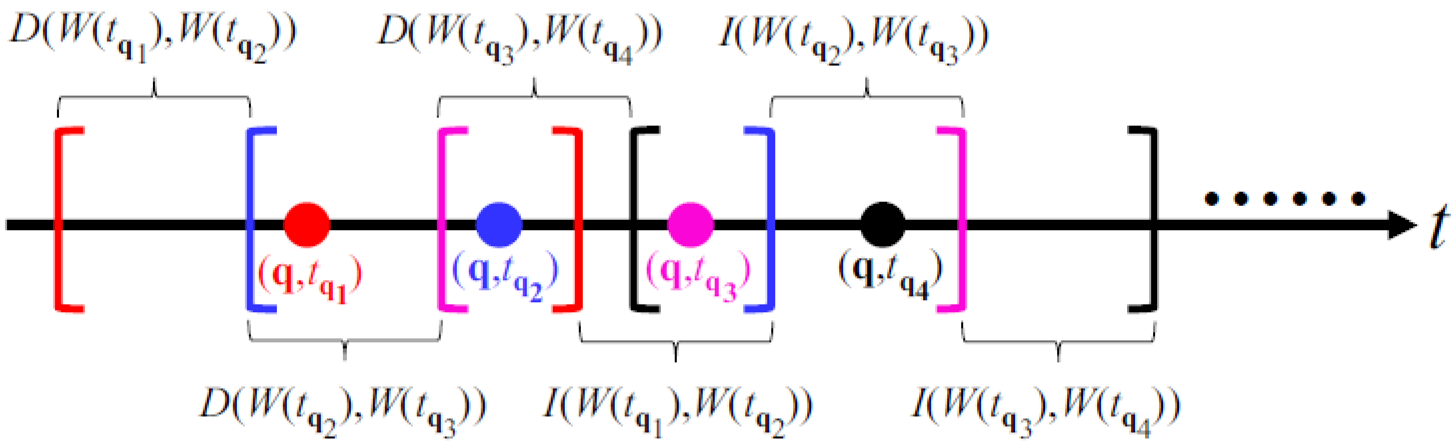
Core Idea 2 of SWS: Incremental Computation



$$S^{(i)}_{W(t_{\mathbf{q}_n})}(\mathbf{q}) = S^{(i)}_{W(t_{\mathbf{q}})}(\mathbf{q}) - \sum_{(\mathbf{p},t_{\mathbf{p}})\in D\left(W(t_{\mathbf{q}}),W(t_{\mathbf{q}_n})\right)} t_{\mathbf{p}}^{i}\cdot K_{\text{space}}(\mathbf{q},\mathbf{p}) + \sum_{(\mathbf{p},t_{\mathbf{p}})\in I\left(W(t_{\mathbf{q}}),W(t_{\mathbf{q}_n})\right)} t_{\mathbf{p}}^{i}\cdot K_{\text{space}}(\mathbf{q},\mathbf{p})$$

The time complexity is $O\left(\left|I\left(W(t_{\mathbf{q}}),W(t_{\mathbf{q}_n})\right)\right| + \left|D\left(W(t_{\mathbf{q}}),W(t_{\mathbf{q}_n})\right)\right|\right)$

Core Idea 2 of SWS: Incremental Computation



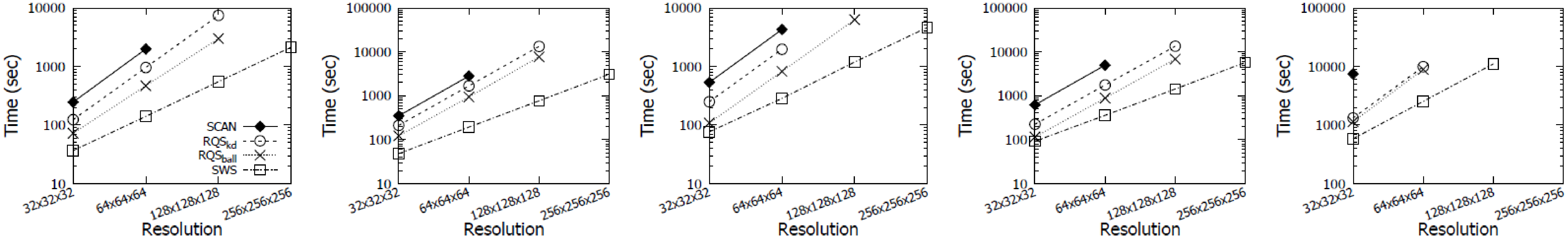
The time complexity is $O\left(\left|W_{t_{\mathbf{q}_1}}\right| + \sum_{i=1}^{T-1}\left|I\left(W(t_{\mathbf{q}_i}), W(t_{\mathbf{q}_{i+1}})\right)\right| + \sum_{i=1}^{T-1}\left|D\left(W(t_{\mathbf{q}_i}), W(t_{\mathbf{q}_{i+1}})\right)\right| + T\right)$

$= O(T+n)$

There are $X \times Y$ pixels $\Rightarrow$ Generating STKDV is $O(XY(T+n))$ time ☺

Our Experiment

Dataset	n	Category
Ontario	560,856	COVID-19
Seattle	839,504	Crime
Los Angeles	1,255,668	Crime
New York	1,499,928	Traffic accident
New York$_{taxi}$	13,596,055	Pickup location

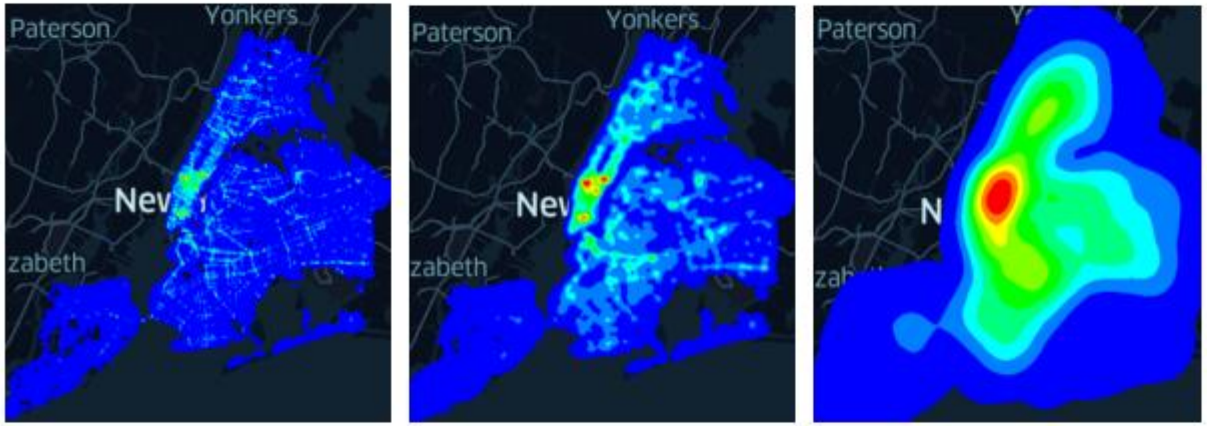


(a) Ontario (b) Seattle (c) Los Angeles (d) New York (e) New York$_{taxi}$

LIBKDV: A Versatile Kernel Density Visualization Library for Heatmap Analytics

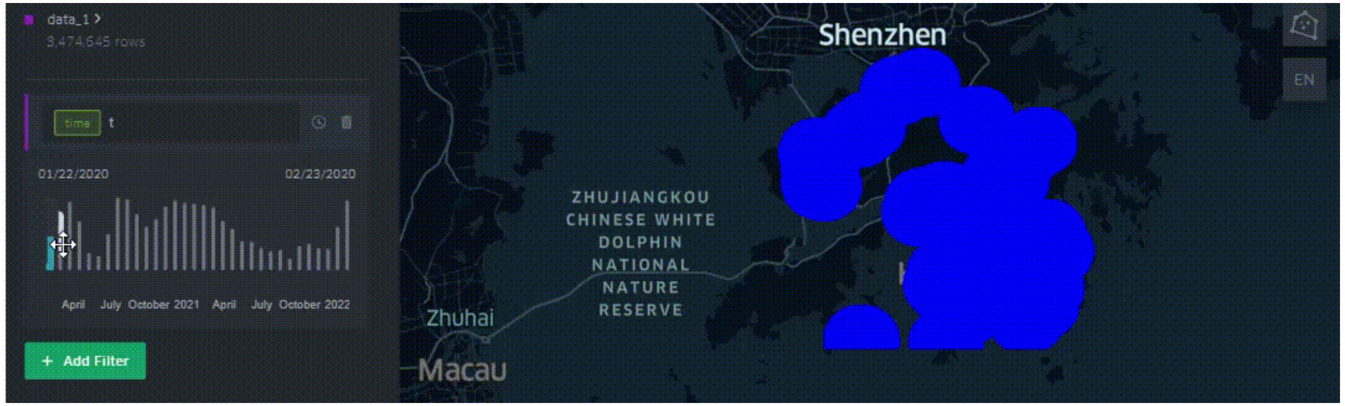
What is LIBKDV?

- A python library for supporting KDV and STKDV.
  - Adopt our solution, SLAM, for computing KDV
  - Adopt our solution, SWS, for computing STKDV
- Webpage: https://github.com/libkdv/libkdv
- Functionalities:



(a) Small b (b) Moderate b (c) Large b

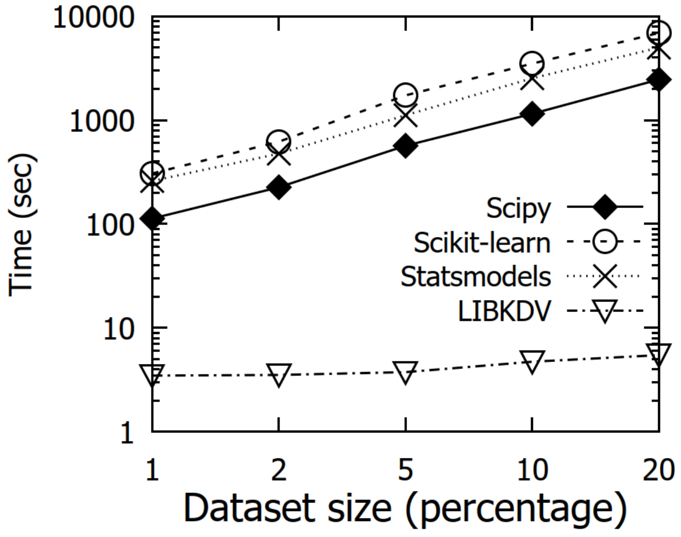
Generate multiple KDVs (based on LIBKDV) with different bandwidths b for the New York traffic accident dataset.



Generate STKDV (based on LIBKDV) with different bandwidths b for the Hong Kong COVID-19 cases

LIBKDV is All You Need!

- Fast ☺



- Easy to use ☺

```
NewYork = pd.read_csv('./Datasets/New_York.csv')
traffic_kdv = kdv(NewYork,KDV_type="KDV",bandwidth_s=1000)
traffic_kdv.compute()
```

KDV

```
libkdv_obj = kdv(dataset, KDV_type,
                 GPS=true,
                 bandwidth_s=1000, row_pixels=800, col_pixels=640,
                 bandwidth_t=6, t_pixels=32,
                 num_threads=8)
libkdv_obj.compute()
```

STKDV

Use Cases: Hong Kong and Macau COVID-19 Hotspot Maps

Hong Kong and Macau COVID-19 Hotspot Maps

- Websites:
  - Hong Kong version (https://covid19.comp.hkbu.edu.hk/)
  - Macau version (http://degroup.cis.um.edu.mo/covid-19/)

- Powered by LIBKDV (https://github.com/libkdv/libkdv)

- Can achieve real-time performance (< 0.5 sec) for computing KDV ☺

- Can achieve nearly real-time performance for computing STKDV ☺

Publicity of Hong Kong COVID-19 Hotspot Map

浸大推新冠確診個案分布圖　實時掌握各地區風險水平

新聞觀看次數：4.5k

11月14日(一) 13:43　推介 9　分享　Tweet　分享

浸大推出「香港新冠病毒熱點分析圖」，可呈現確診個案的地理位置分布。

新冠肺炎疫情仍未平息，為助公眾了解不同地區的感染風險，香港浸會大學領導的研究團隊推出「香港新冠病毒熱點分析圖」，以直觀、實時和動態的方式，呈現新冠病毒個案的地理位置分布。此線上地圖有助及時和準確地掌握新冠感染個案位置分布資訊，並根據新冠個案

Oriental Daily Hong Kong (in Chinese)

浸大推確診分析圖 實時掌握各區風險

【大公報訊】香港浸會大學領導的研究團隊推出「香港新冠病毒熱點分析圖」，以直觀、實時和動態的方式，呈現新冠病毒個案分布。該地圖採用由團隊開發的時空大數據分析演算法，其運算時間，較現有最先進的方法快100倍。研究成果已發表於今年舉行的兩個大數據管理領域最頂級國際會議「國際數據管理會議」及「國際超大型數據庫會議」。

運算時間較現時快100倍

「香港新冠病毒熱點分析圖」由浸大計算機科學系系主任徐建良教授領導的團隊開發，目的是在線上地圖顯示出全港新冠病毒感染個案的數據。團隊的其他浸大學者包括計算機科學系副系主任蔡冠球教授及研究助理教授陳梓楠博士。此地圖亦由澳門大學及香港大學共同開發。

該分析圖以政府發布的香港互動地圖儀表板作為實時數據，直觀、實時和動態化地呈現新型冠狀病毒個案的地理位置分布。然而，現時應用於時空數據分析的「核密度可視化」計算工具，未能支援「香港新冠病毒熱點分析圖」運行所需，故浸大領導的團隊共同開發出一套新的演算方法。新算法配合漸進式可視化框架，產生持續的局部成像，以減少「核密度可視化」的運算時間。團隊運用大規模數據集進行實驗，結果顯示新算法的運算時間，較現有最先進的方法快100倍。而解像度亦提高至1376×960像素（高清解像度），並能以少於0.5秒的計算時間處理100萬個數據點。

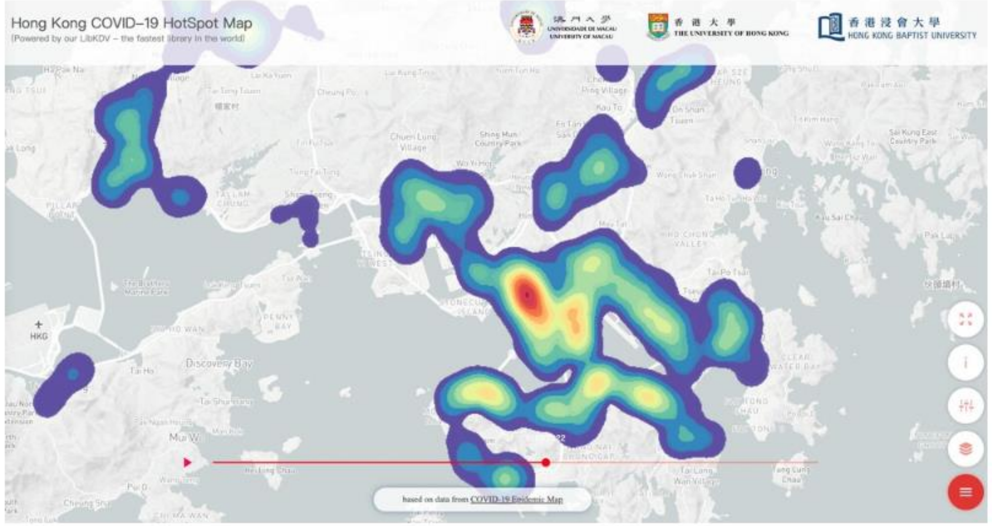
▲浸大領導的研究團隊推出「香港新冠病毒熱點分析圖」，可實時以動態方式呈現新冠病毒個案分布。

徐教授表示，新開發的演算法可以支援更多以「核密度可視化」為基礎的時空大數據分析工作。例如，交通熱點偵測，景區人流控制，樓價可視化分析，以及實時氣象資源管理等。

Ta Kung Pao News (in Chinese)

HKBU-led research team launches Hong Kong COVID-19 hotspot map

Local | 14 Nov 2022 7:16 pm



A research team led by Hong Kong Baptist University has launched the Hong Kong COVID-19 Hotspot Map, which allows the visualisation of the real-time and dynamic geographic distribution of Covid cases in the city.

The standard

Publicity of Macau COVID-19 Hotspot Map

Macau TDM (Video news in Cantonese)

UM COVID-19 research team proposes key points for combating epidemic

NEWS PROVIDED BY
Macao Government News
July 15, 2022, 21:48 GMT

SHARE THIS ARTICLE

MACAU, July 15 - Omicron BA.5 is raging around the world and Macao has also been deeply affected by it. To closely monitor the impact of the virus on the community, the COVID-19 research team of the University of Macau (UM) continues to analyse the latest development trend of the epidemic in the city and underscores some key areas of concerns.

The research team points out that since the discovery of five positive cases in Macao on 18 June, the number of new single-day positive cases had been increasing, with the highest number of new single-day positive cases recorded on 5 July at 146. Since the implementation of 'relatively static' management measures on 9 July, the number of new daily positive cases has been gradually decreasing, with 29 new positive cases on 13 July. Between 18 June and 13 July, 1,644 positive cases have been recorded in the current outbreak, of which over 60 per cent were classified as non-symptomatic. Based on a cautious reading of the development trend of the epidemic in Macao, there are three points worth noting:

1. Strictly adhere to the principles of epidemic prevention to avoid a large increase of cases: Although BA.5 is more transmissible than other subvariants, there has not been an exponential increase in the number of positive cases in Macao as there has been in other affected areas. This reflects the key role that Macao's long-standing anti-epidemic principle of early detection, early

Contact

email us here

More From This Source

Employment survey for August-October 2022

MICE statistics for the 3rd quarter of 2022

[Infographic] Protective measures as part of routine anti-epidemic steps -- Advice for participants in festive, ...

View All Stories From This Source

Newswires

Future Work

Future Work for Research

1. Can we further develop the optimal solution for KDV?
   - Current lower bound time complexity: $\Omega(XY + n)$.
   - State-of-the-art upper bound time complexity: $O(Y(X + n))$/ $O(X(Y + n))$.

2. Can we further develop the optimal solution for STKDV?
   - Current lower bound time complexity: $\Omega(XYT + n)$.
   - State-of-the-art upper bound time complexity: $O(XY(T + n))$.

3. Can we extend our solutions to other kernel functions (e.g., Gaussian kernel and exponential kernel)?

Future Work for Research

4. Can we extend our solution to support other types of spatial visualization tasks?
   - Kriging
   - Inverse distance weighting (IDW)

5. Can we extend our solution to support other spatial analysis tasks?
   - K-function
   - DBSCAN clustering

Future Work for Software Development

1. Can we further extend our python library LIBKDV to support more visualization tools and data analysis operations (e.g., Kriging, IDW, and K-function)?

2. Can we further integrate our library in (1) into the commonly used software packages?
    - Develop plugins for QGIS and ArcGIS
    - Integrate our methods into Scikit-learn and Scipy.

3. Can we further develop an R package for supporting those operations in (1)?

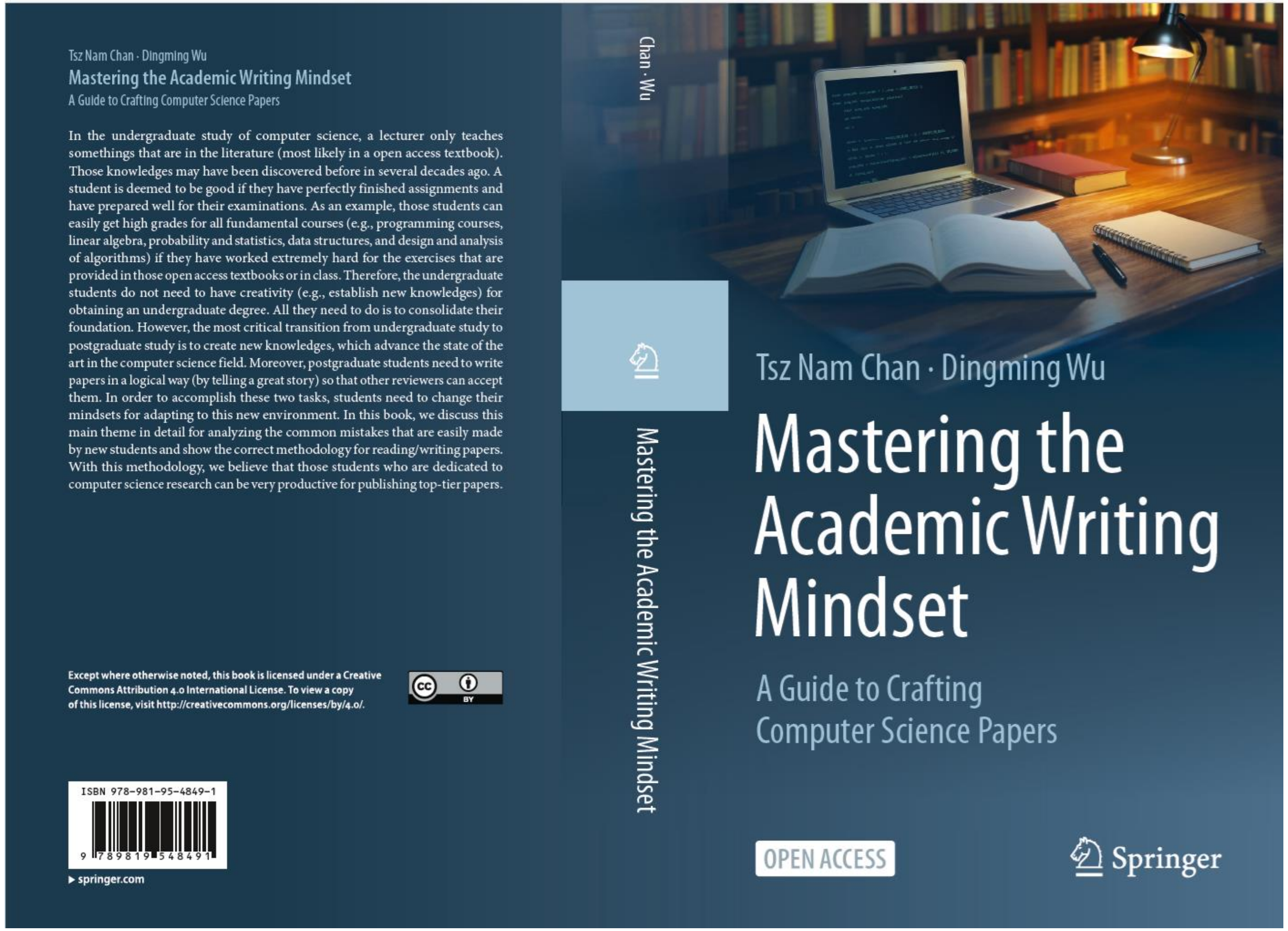
Future Work for Software Development

4. Can we further extend our web-based system (Hong Kong COVID-19 hotspot map) to support more visualization tools and data analysis operations?

5. (**Long-term goal**) Develop a software package (like ArcGIS and Scikit-learn) that includes our complexity-reduced algorithms for different operations.

My New Book

- Related to academic writing mindsets for writing computer science papers.
- Useful for your students who are struggling with writing research papers.
- Available on 31st March 2026.
- My signature is free. ☺



Questions?