Supplementary Document for "Large-scale Spatiotemporal Kernel Density Visualization"

Tsz Nam Chan[1], Pak Lon Ip[2], Bojian Zhu[3], Leong Hou U[2], Dingming Wu[1], Jianliang Xu[3], Christian S. Jensen[4]

[1]Shenzhen University, [2]University of Macau
[3]Hong Kong Baptist University, [4]Aalborg University

In this supplementary document, we further include some content that cannot be presented in the paper [4] due to space limitations.

1 Efficiency Comparisons with RQS Methods

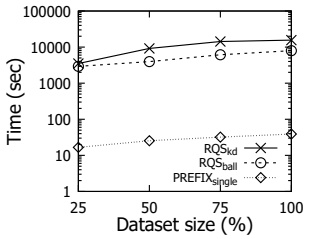
Recall from Section II that the range-query-based methods, including kd-tree [1] and ball-tree [6], can be adopted to improve the efficiency of generating STKDV. Therefore, we compare the time efficiency of our PREFIX solution with these two methods, RQS_{kd} and RQS_{ball}, under two settings (i.e., STKDV with on-the-fly timestamps and STKDV with known timestamps) using two large-scale datasets, namely Los Angeles and New York (see Table III), in this section. As a remark, since the response time of RQS_{kd} and RQS_{ball} are always larger than 86,400 sec for supporting bandwidth tuning, we omit this experiment in the technical report. In the following experiments, we (1) fix the spatial resolution $X \times Y$ and the number of on-the-fly/known timestamps T to be 320×240[1] and 32, respectively, (2) sample each dataset with four sampling ratios, which are 25%, 50%, 75%, and 100% (the original one), and (3) measure the response time of each method.

STKDV with on-the-fly timestamps. Figure SD1a and Figure SD1b show the results of all methods. Since RQS_{kd} and RQS_{ball} cannot reduce the time complexity for generating STKDV with an on-the-fly timestamp (i.e., with $O(XYn)$ time), $\text{PREFIX}_{\text{single}}$ (with $O(Y(X+n))$ time in Theorem 1) can achieve speedups of 155.91x to 674.33x compared with these two methods.

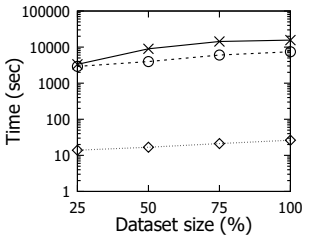
STKDV with known timestamps. Observe from Figure SD1c and Figure SD1d that $\text{PREFIX}_{\text{multiple}}$ achieves speedups of 611.89x to 1223.34x compared with the RQS_{kd} and RQS_{ball} methods. The main reason is that the time complexity of $\text{PREFIX}_{\text{multiple}}$ is $O(XYT + Yn)$ (see Theorem 2), which is must smaller than the one of RQS_{kd} and RQS_{ball} (i.e., $O(XYTn)$).

As a remark, we expect that the time gaps between PREFIX and SWS can be larger once we adopt the larger spatial resolution (e.g., 1280×960, i.e., the default one in [4]).

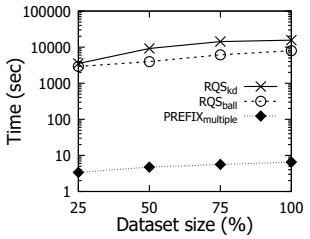
[1]We do not adopt the default spatial resolution 1280×960 since the response time of RQS_{kd} and RQS_{ball} is very large.



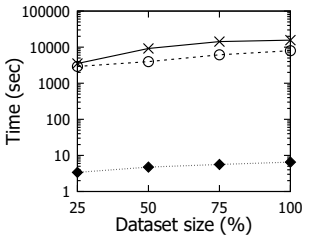
(a) Los Angeles
(STKDV with T on-the-fly timestamps)



(b) New York
(STKDV with T on-the-fly timestamps)



(c) Los Angeles
(STKDV with T known timestamps)



(d) New York
(STKDV with T known timestamps)

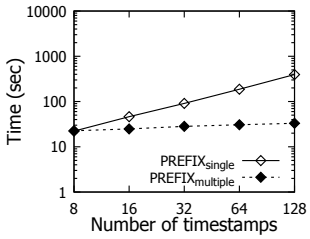
Figure SD1: Response time (in total) for computing a single STKDV with T on-the-fly timestamps (a and b) and T known timestamps (c and d), varying the dataset size.

2 Efficiency Comparisons of Our Methods

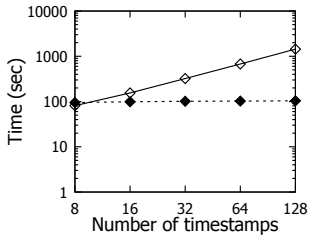
Since PREFIX$_{\text{single}}$ is used to improve the efficiency for generating STKDV with on-the-fly timestamps, this method can also be used to generate STKDV with T known timestamps and support bandwidth tuning. For the similar reason, PREFIX$_{\text{multiple}}$ can also be used to support bandwidth tuning. Therefore, we conduct two experiments for comparing the efficiency of our methods in these two problems. For brevity, we report findings on two datasets only, namely Los Angeles and London.

Varying the number of timestamps: In the first experiment, we aim to test the efficiency of PREFIX$_{\text{single}}$ and PREFIX$_{\text{multiple}}$ for computing STKDV with T known timestamps. To conduct this experiment, we vary the number of timestamps, by following the same settings in Section V-C, and measure the response time of these two methods. Since the time complexity of using PREFIX$_{\text{single}}$ is $O(TY(X+n))$ (based on Theorem 1), the response time of PREFIX$_{\text{single}}$ is linearly proportional to the number of timestamps (see Figures SD2a and b). Note that PREFIX$_{\text{multiple}}$ achieves speedups of up to 13.9x over PREFIX$_{\text{single}}$, which is more scalable to large numbers of known timestamps T.

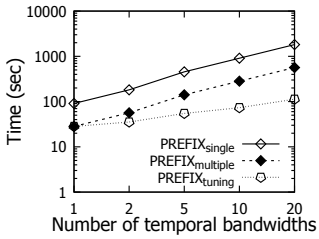
Varying the number of temporal bandwidths: In the second experiment, we follow the same settings in Section V-D and vary the number of temporal bandwidths to test the efficiency of PREFIX$_{\text{single}}$, PREFIX$_{\text{multiple}}$, and PREFIX$_{\text{tuning}}$. Figures SD2c and d show that the response time of PREFIX$_{\text{single}}$ (with $O(MNTY(X+n))$ time, based on Theorem 1) and PREFIX$_{\text{multiple}}$ (with $O(MN(XYT+Yn))$ time, based on Theorem 2) is linearly proportional to the number of temporal bandwidths. Note that PREFIX$_{\text{tuning}}$ is more scalable to large numbers of temporal bandwidths N, achieving speedups of up to 36.22x and 11.4x over PREFIX$_{\text{single}}$ and PREFIX$_{\text{multiple}}$, respectively.



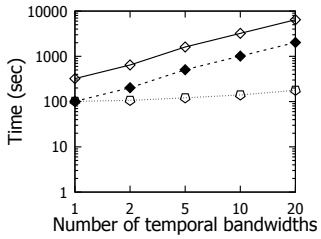
(a) Los Angeles
(STKDV with T known timestamps)



(b) London
(STKDV with T known timestamps)



(c) Los Angeles
(bandwidth tuning)



(d) London
(bandwidth tuning)

Figure SD2: Response time for computing a single STKDV (a and b), varying the T known timestamps, and supporting bandwidth tuning (c and d), varying the number of temporal bandwidths N, following the experimental settings in Section V-C and Section V-D, respectively.

3 Parallelization of PREFIX

Recall from the core idea 1 (see Section V-A) that our PREFIX solution needs to compute the statistical terms, $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ (where $u = 0, 1, 2$), for all pixels $\mathbf{q}$ (see Figure 5 in [4]), which is based on the SLAM$_{\text{BUCKET}}$ method [5]. In this modified method, we can compute $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ for each row of pixels with $O(X + |W_\mathcal{I}|)$ time, which results in the time complexity of $O(Y(X + |W_\mathcal{I}|))$ in Lemma 1.

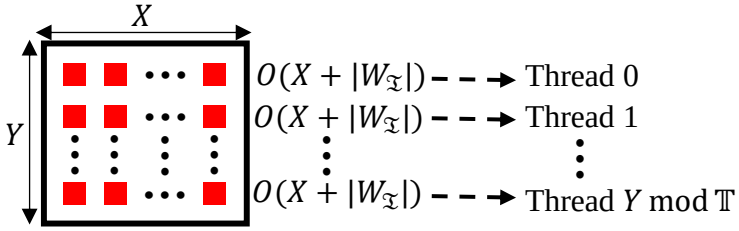
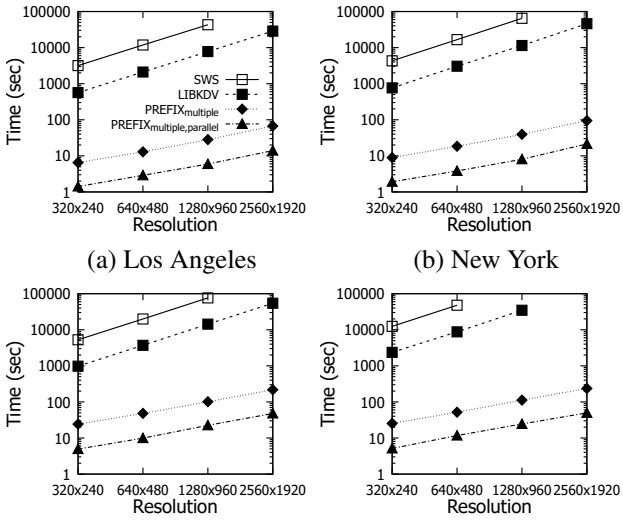


Figure SD3: Using the round robin approach to compute the statistical terms $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ for each row of pixels in a sliding window $W_\mathcal{I}$, where each computer has $\mathbb{T}$ threads.

Observe from Figure SD3 that the computations of $S^{(u)}_{W_\mathcal{I}}(\mathbf{q})$ in different rows are independent to each other. Therefore, a simple approach, which has also been considered by LIBKDV [3] to parallelize the SLAM$_{\text{BUCKET}}$ method, is to assign these computations using the round robin approach. For example, the computations in the first row and the second row of pixels are assigned to thread 0 and thread 1, respectively.

Here, we test the efficiency performance of four methods, which are SWS, LIBKDV, PREFIX$_{\text{multiple}}$, and PREFIX$_{\text{multiple,parallel}}$, for generating STKDV with T

known timestamps. SWS [2] is the state-of-the-art sequential method, while LIBKDV parallelizes SWS for generating STKDV. $\text{PREFIX}_{\text{multiple}}$ and $\text{PREFIX}_{\text{multiple,parallel}}$ are our sequential method and parallel method (based on the above description), respectively, for generating STKDV. To conduct this experiment, we choose four large-scale datasets, which are Los Angeles, New York, London, and San Francisco, for testing. In addition, we adopt the default number of known timestamps to be 32, fix the number of threads to be 8 (which is the same setting as [3]), and vary the spatial resolution from 320×240 to 2560×1920. Figure SD4 shows the results of all methods. Although LIBKDV can highly parallelize the SWS method, our sequential method $\text{PREFIX}_{\text{multiple}}$ is still significantly faster than LIBKDV (with speedups of 40.39x to 486.28x). In addition, $\text{PREFIX}_{\text{multiple,parallel}}$ further achieves speedups of 4.42x to 4.95x compared with $\text{PREFIX}_{\text{multiple}}$.



(a) Los Angeles (b) New York

(c) London (d) San Francisco

Figure SD4: Response time for generating STKDV with known timestamps using the sequential (i.e., SWS and $\text{PREFIX}_{\text{multiple}}$) and parallel (i.e., LIBKDV and $\text{PREFIX}_{\text{multiple,parallel}}$) methods.

4 Space Consumption for Computing STKDV

Here, we investigate the memory space consumption of all methods for computing STKDV with (i) on-the-fly timestamps and (ii) known timestamps, by using the Ontario and Los Angeles datasets. For brevity, we omit the results for solving the bandwidth tuning problem.

Varying the number of on-the-fly timestamps: In the first experiment, we choose the number of on-the-fly timestamps T to be 2, 4, 8, 16, and 32 and then measure the memory space consumption of the SWS and $\text{PREFIX}_{\text{single}}$ methods for each T. In Figures SD5a and b, since both methods process each timestamp independently, these two methods are insensitive to the number T. Due to the same space complexity of these two methods (see Table II), $\text{PREFIX}_{\text{single}}$ only incurs a slight space overhead compared with the SWS method.

Varying the number of known timestamps: In the second experiment, we vary the number of known timestamps T from 8 to 128 and measure the memory space consumption of all methods, i.e., SWS and PREFIX$_{\text{multiple}}$. Figures SD5c and d show that the space consumption of these two methods is linearly proportional to the number of timestamps T. Since the space complexity of SWS and PREFIX$_{\text{multiple}}$ is the same (see Table II in [4]), the space consumption of PREFIX$_{\text{multiple}}$ is close to that of SWS.

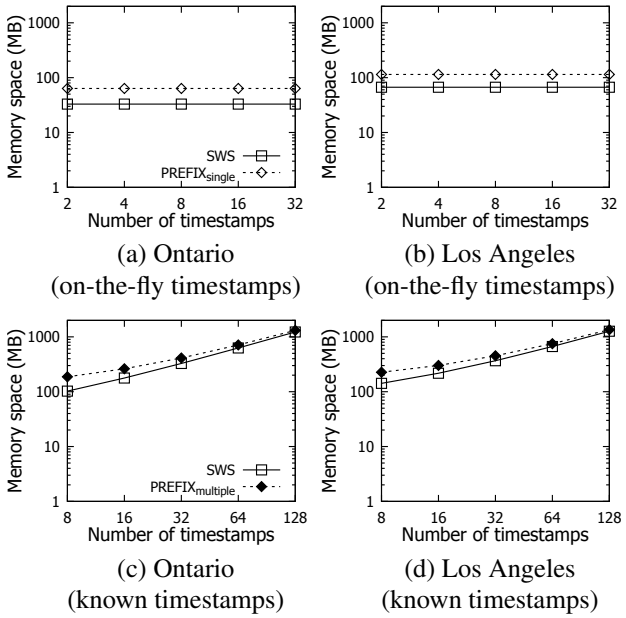


Figure SD5: Memory space consumption for computing STKDV for the Ontario (a and c) and Los Angeles (b and d) datasets, varying the number of on-the-fly timestamps (a and b) and the number of known timestamps (c and d).

References

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[2] T. N. Chan, P. L. Ip, L. H. U, B. Choi, and J. Xu. SWS: A complexity-optimized solution for spatial-temporal kernel density visualization. *Proc. VLDB Endow.*, 15(4):814–827, 2022.

[3] T. N. Chan, P. L. Ip, K. Zhao, L. H. U, B. Choi, and J. Xu. LIBKDV: A versatile kernel density visualization library for geospatial analytics. *Proc. VLDB Endow.*, 15(12):3606–3609, 2022.

[4] T. N. Chan, P. L. Ip, B. Zhu, L. H. U. D. Wu, J. Xu, and C. S. Jensen. Large-scale spatiotemporal kernel density visualization. In *ICDE*, 2025 (In submission).

[5] T. N. Chan, L. H. U, B. Choi, and J. Xu. SLAM: efficient sweep line algorithms for kernel density visualization. In *SIGMOD*, pages 2120–2134. ACM, 2022.

[6] A. W. Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *UAI*, pages 397–405, 2000.