A Client-based Visual Analytics Framework for Large Spatiotemporal Data under Architectural Constraints

Guizhen Wang, Abish Malik, Chittayong Surakitbanharn, José Florencio de Queiroz Neto, Shehzad Afzal, Siqiao Chen, David Wiszowaty and David S. Ebert, *Fellow, IEEE*

Abstract—A primary aim of visual analytics is to provide end-users interactive and scalable environments to facilitate their decision making tasks. Researchers have often utilized several server-client solutions to support interactive data exploration (e.g., building the data cube, parallelizing data processing). However, these solutions can suffer from scalability issues especially in the absence of adequate computation functionality provided by servers. Organizational policies can also prohibit the transfer of data to external data servers because of security or budgetary concerns; thereby, severely limiting the capability of the visual analytic systems. Therefore, in this paper, we propose an interactive client-based visual analytics framework for large-scale spatiotemporal data. The proposed framework follows a sampling based incremental visual analysis approach to sustain the real-time responsiveness, meanwhile, with affordable computation resources in a client machine. General sampling methods [34] preprocess the entire dataset to build data indexing, which can bring the client unaffordable computation overhead. Instead, our framework proposes a novel data management model, using the spatiotemporal clustering pattern to predictively organize and sample data based on historical data acquisition activities. We demonstrate the capabilities and usefulness of our framework by applying it on crime data and twitter data. We also conduct several experimental evaluations to determine the efficacy of our framework.

Index Terms—Large spatiotemporal data, data management, incremental visualization

1 Introduction

In the big data era, interactive visual analytic environments often require advanced computing platforms or advanced client-server architectures with sufficient computing abilities to enable interactive analysis. These solutions typically offload the expensive computational tasks to a high-performance server or a distributed computing platform (e.g., Hadoop), while leaving the client-side application to mainly focus on user interactions and visual representations.

Unfortunately, in many businesses and local governmental organizations, policies or budgetary concerns may prevent deployment of such solutions, or the transfer of data to external servers [21, 31], thereby, only allowing the server to provide data, and requiring the client to take care of all the computational workload. However, for data data, typical client machines probably lack adequate computation resources to process the entire data, let alone providing real-time responsiveness to end users.

In order to address this architectural constraint, we propose an incremental visual analytics framework that enables interactive analysis of large spatiotemporal data under these client-server constraints: (1) a fixed server that only serves as a data provider (e.g., a relational database), and (2) a local client-side system (e.g., desktop, web-based) subject to limited computational and memory resources. In our framework, we utilize a sophisticated spatiotemporal data structure suitable for the client to progressively query meaningful samples from the database on the server. The approximate visualizations are created in the client based on these samples, which are continuously updated since the client keeps fetching and applying new samples until reaching 100% accuracy or when canceled by the end user. The entire workflow is well coordinated and provides real-time visual representations to the users. Furthermore, it can be completed by a typical client machine on account of the significantly reduced computation latency and resource consumption to process a few data samples. Existing spatiotemporal data sampling methods (e.g., STORM [34] and TrajStore [10]) cannot always be completed by a client machine, for the lengthy computation time and large volume of computer resource consumption to index all the data. Instead, our novel method is to propose a predictive data management strategy to index based on the spatiotemporal patterns discovered from previous data acquisition, within the client computation affordability, and support data sampling progressively from the server.

Specifically, our work contributes the following:

- An incremental visual analytics environment for large spatiotemporal datasets that supports interactive exploration for environments where the visual analytic software runs on an average client machine with a server constrained to only provide data query.
- A predictive data management model that supports incremental sampling from indeterminate data distribution, effectively guaranteeing each incremental cycle not only obtains representative samples but also adapts to the computation capability of the client machine.
- A user-driven data acquisition scheme that prioritizes data fetching based on the spatiotemporal range of users' interest and minimize data acquisition between the server and client through reusing client memory cache and disk cache.

2 Related Work

Researchers have proposed various techniques to achieve interactive exploration of big data. Computational latencies can be remarkably reduced through parallelizing the data processing workflow, e.g., MapReduce [11] and imMens [23]. Data aggregation can save query time through indexing the entire data based on the specification of analytical tasks, e.g, techniques including *imMens* [23], Nanocubes [22], and Hashedcubes [3] can speed up the data retrieval performance through aggregation by spatial, temporal or categorical attributes. Data prefetching techniques [4, 7] can effectively shorten latencies as well through predicting user behaviors and fetching data in advance.

To reduce the demand on computational resources, techniques have been explored to minimize resource consumption. For example, out-of-core precomputation that divides and processes data one block at a time, can address memory limitations and advance scientific visualization ([6, 8, 9, 16, 33]). A combination of in-memory indexing and disk-resident techniques have also been proposed to mitigate memory shortage. For

• *Guizhen Wang, Abish Malik, Chittayong Surakitbanharn, Shehzad Afzal, Siqiao Chen, David Wiszowaty and David S. Ebert are from Purdue University. E-mail: {wang1908|amalik|csurakit|safzal|chen1722|dwiszowa|ebertd} @purdue.edu.*
José Florencio de Queiroz Neto is from Federal University of Ceara. Email: florencio@lia.ufc.br

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

example, *EdiFlow* [5] is an interactive workflow for visual analysis with DBMS as the temporary storage of computation results.

Different from the aforementioned techniques to process all the data and generate accurate results, sampling-based approximate data query techniques (e.g., Sample+Seek [12] and STORM [34]) can use a small ratio of samples to generate approximate results. Many research efforts have been made to help end users utilize approximate results to make decisions [14, 15, 19, 28, 29]. To reduce the high computational workload to sample spatiotemporal data, our work aims in the architectural constraint environment having a client with limited computation capabilities and a server reluctant to be customized, and proposes a framework to allow a typical client machine responsible for incremental spatiotemporal visual analytics.

3 Framework Overview

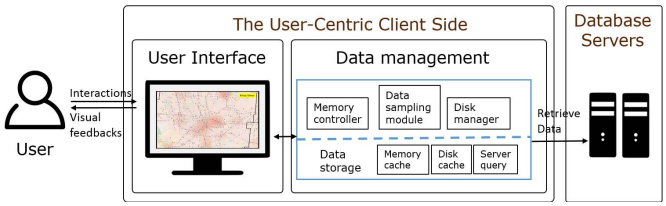

Fig. 1: Framework components and workflow between the client side and the remote server.

Our interactive client-based visual analytics framework can be decomposed into three main components, the client-side user interface, remote data servers, and the client-side data management model (introduced in Section 4), seen in Figure 1. In what follows, we explain the key technical ideas of these three components.

The **user interface** is a multiview visual interface that supports user interactions and provides incremental visual feedback to users. After users issue a new query, the user interface will send the user-specific spatiotemporal range to the data management model, and in each subsequent cycle, receive data samples from the data management model, filter data based on filtering predicates, update the visualization, and show the data loading percentage, a value calculated by the data management model to denote the ratio of nodes that have been sampled, to tell users how reliable the approximate results can be. Users can either wait for more accurate results, suspend incremental updates, or stop the incremental updates when the approximate data visualization is accurate enough for their tasks. As an example, Figure 2 illustrates how a heatmap progressively updates.

Data servers can be any platform that hosts the entire dataset and allows clients to fetch data. Despite the architectural constraints to prevent deploying customized functions, databases on the server side can host the entire dataset and provide the function to retrieve data based on the spatiotemporal data. For example, popular spatiotemporal indexing techniques (e.g., R-Tree [18] and data cube techniques [17, 30]), widely provided in commercial database systems, can efficiently reduce the data query time. Therefore, in our framework, we simply assume that the data server query time is proportional to the number of data records requested by the client, as is the data transfer time from the server to the client. As long as the client retrieves a smaller number of points from the server, the latency between sending the request and receiving the data can be reduced to an appropriate value.

The **data management model** is associated with a single client application, consisiting of three components: the data sampling module, the memory controller, and the disk manager. The **data sampling module**, introduced in Section 4, incrementally fetches data from the server in accordance with the sample size specified by an upper bound for interactive performance. The **memory controller** monitors, predicts, and swaps in-memory data to the disk when necessary. When the application launches, memory will be consumed for initialization, including views in the user interface, modules in the data management model and so on. After initialization, more client memory is gradually consumed

Fig. 2: Incremental visual updates of the heatmap using the kernel density estimation technique [27] to show the distribution of tweets in Chicago (introduced in Section 5) at different stages of the data loading process, indicated by the number below the image.

for loading new data, but the memory cost of the user interface will remain almost constant despite incrementally accumulating data samples for these visualizations. For example, the spatial heatmap in the system initialization stage requires memory allocation for a grid to store the spatial histogram. In the incremental data update, the user interface only updates bin values in the grid, without allocating new memory. To prevent the out-of-memory issue, the controller tracks the memory utilization of each incremental update cycle and estimates the memory usage in next cycle, since the data sampling module can predict the number of points in a node that is going to be fetched from the server. That information can be applied to coarsely calculate the memory cost of the next cycle. If the memory is predicted to be insufficient for the next cycle, one in-memory data node that has not been recently accessed is chosen based on the LRU rule [32] to be swapped to the disk cache. The **disk manager** organizes the data in the disk, stores data swapped out from the memory, and schedules disk data read/write operations.

To minimize data acquisition from the server side, our framework supports user-driven data acquisition: only loading data of user interest into the client side, since the data management model only samples nodes whose spatiotemporal ranges overlap the user query range, and avoiding data that are not requested by user interactions. The other acquisition minimization method is to reuse the data that have been obtained from the server side to the client side, including memory cache and disk cache.

4 Incremental Spatiotemporal Data Sampling under Architectural Constraints

This section describes our client-based spatiotemporal data sampling model design for constrained environments. In our framework, the model is responsible for maintaining a spatiotemporal index of the data in a predictive way, and conducting incremental data sampling to ensure visually approximate results.

4.1 Client-side Data Organization

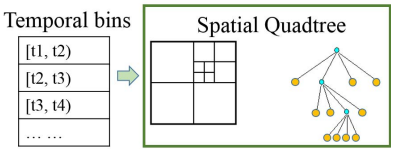

Fig. 3: Illustration of client-side data organization using the two-level indexing.

The proposed data management model organizes data with a two-level organization shown in Figure 3. The first level is the temporal indexing, dividing the entire temporal range into equal bins (e.g., one month). The second level uses spatial indexing to organize data within the same temporal bin into a quadtree [24, 25] based on their spatial distribution. Every node in the quad tree covers a rectangular spatial range, with only leaf nodes having data, and an upper bound of the number of points that one leaf node could contain at most - the framework takes care of this limit, equally dividing a node into four children

when it is going to have more points than the upper bound. Thus, the spatial space decomposition in the quadtree will follow the spatial data density in its related temporal range: denser areas will have more nodes and vice versa.

4.2 Spatiotemporal Data Sampling

Our sampling method incrementally samples nodes, in accordance with the data distribution where denser areas can have a greater likelihood of nodes being sampled. However, in the constrained environment, it is infeasible to preprocess the entire data collection to determine the data density distribution and build the index. Therefore, we incrementally sample data from the server and refine the index based on data densities determined from the historical sampling. We have developed a predictive way to estimate data densities, refine the node organization and further sample data in accordance with its actual statistical distribution.

The client-side data organization initially assumes that the data is uniformly distributed in the spatiotemporal space, since the client side does not have any information about the data distribution. Hence, the client runs a server query to retrieve the total count of data and the entire spatiotemporal range in the dataset. This information is used to divide the time range equally into temporal bins, and to build, for each temporal bin, a spatial quadtree.The quadtree is progressively built until the leaf has fewer points than the upper-bound framework parameter.

After receiving a data request with specified spatial and temporal ranges, the data management model incrementally samples the data using the following steps:

1. Select all leaf nodes that spatiotemporally overlap with the specified ranges, and form a node set S
2. Randomly choose a node in the selected node set S
3. If the density of the node is unknown, estimate the data density of the node, with possible adjusts in the quadtree in case of upper-bound violation or merging the node and its three siblings into a parent node in case of sparseness.
4. Fetch the data of the sampled node N from the server
5. Update the corresponding quadtree leaf with the exact number of points
6. Send the data for visualization, and remove N from S
7. Repeat steps 2-6 until S is empty or canceled by end users

The data density estimation procedure in Step 3 is necessary because, sampling a node with unknown data densities actually has an excessively large number of points, the cycle can take longer to fetch data from the server and process the data on the client side, making the client application unresponsive. Predicting the number of points in a node $N_{predict}$ involves three steps. The first step is to collect nodes in other temporal bins that have acquired data from the server and have the same spatial range as $N_{predict}$. Then, the number of points in $N_{predict}$ is calculated using Equation 1. If predicted to have a larger point number than the upper-bound, the node can be split into four children. If predicted to be smaller than the upper bound, the father node will be estimated to see whether the four children nodes can be merged as long as the predicted data size is no more than the upper bound. The last step is to get the final node for sampling. If the original node is split, a child is randomly selected; if the node is merged, its father is selected; otherwise, the node itself is finally selected.

$$p_{predict} = \frac{\sum_{i=1}^{n} p_i}{|n|}, \tag{1}$$

where n is the total number of nodes that have the same spatial range with $N_{predict}$ and know their data densities.

p_i is the number of points in the i-th node.

5 Evaluation and Results

In this section, we conducted experiments to verify the effectiveness of our proposed framework. The client is a 32-bit desktop application implemented on the .NET framework, with one thread for the user interface (Figure 4) and the other for the data management model. The client connects a MySQL database hosted by a data server in the same network domain and utilizes SQLite [2] to store data in the disk. We set the upper bound of a node to be 4096, a multiple of a Windows disk block size, 4KB. The data in the server side is grouped into multiple tables with each table for one temporal bin. The test datasets included two years of traffic incident reports for the state of Ohio, and six months of Twitter data for the city of Chicago (Table 1). The temporal range is evenly divided into multiple bins: one month for the osp datasets, and one week for the tweet dataset. The map view was set to overview the entire Ohio geospatial range for osp, and the entire Chicago geospatial range for tweets. For the osp data, the experiment loaded all the data. For the tweets, a specific temporal range from April 3 to May 9 in 2013, nearly 1.7 million, was selected for experiment. The application was launched with a clean disk cache, and the process was repeated five times for each dataset. All experiments were conducted on a client machine with an Intel(R) Xeon(R) E5-2630 CPU with twelve cores at 2.60GHz, 32GB main memory, and a 256GB solid state drive.

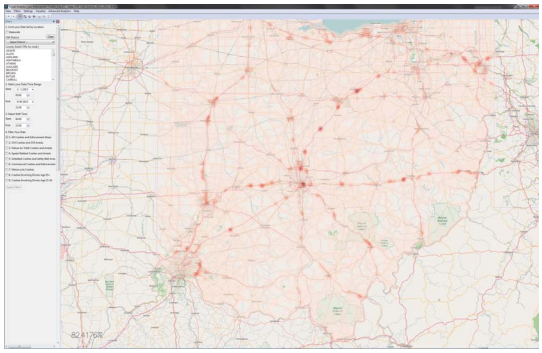

Fig. 4: The user interface of an implemented incremental spatiotemporal data visual analytics prototype. The left is the data filter panel, and the right is a map view showing the spatial heatmap rendered with OpenGL [1] and the bottom left of the map view displays the data loading percentage.

Data	Description	Size
osp	Ohio crime data from January 1, 2012 to December 31, 2013	3.2 million
tweet	Chicago Twitter data from April 1, 2013 to September 30, 2013	9.7 million

Table 1: Evaluated traffic incident reports and Twitter datasets.

Table 2 measured the latency in each cycle, including the node sampling time, the server side query time, and data transfer time, which averaged less than 500ms for all datasets. The low resulting transfer time support the interactive analysis of end users.

Data	Average time per cycle(ms)	Total cycles
osp	456.8754	1915.8
tweets	407.3	4681

Table 2: Measurements of incremental visual updates in our prototype for one incremental visual update, averaged in five trials.

Figure 5(a-c) measures the number of points fetched from the server at each incremental cycle. Figure 5(a) shows the histogram of the number of points fetched from the server averaged in the five trials. The

distribution of the osp data has a higher density around [1000, 3000], and the tweet data is more concentrated with a higher density around [0, 2000]. Figure 5(b) shows the number of points fetched per cycle in one trial of the osp dataset. We can see that the value in the earlier 8% cycles is sometimes significantly larger than the upper bound, 4096, with the peak at 16,962, and in the subsequent cycles, going down to around or below the upper bound. Figure 5(c) shows the number of points fetched per cycle in one experiment of the tweet dataset. In this case, the total points fetched per cycle is, most of the time, around or below the upper bound, with only four examples in the first 50% cycles being significantly larger than the upper bound.

Figure 5(d) uses Root-Mean-Square-Error (RMSE) [20] to measure the accuracy between the approximate spatial data distribution per incremental cycle and the final exact data distribution. A Kernel Density Estimation (KDE) method [13] measured the spatial distribution, with the spatial resolution of the 2D histogram to be 256 by 256 pixels. To reduce the measure inaccuracy from the geospatial sparsity, we only compare KDE values in denser areas defined as spatial bins with a KDE value no less than 0.05 on a normalized scale of 0 to 1.We find that the RMSE value starts around 0.15 in the first cycle, gradually decreases with increased data sampling, and is reduced to 0.1 when 10% of the dataset are sampled with the attribute-based sampling procedure.

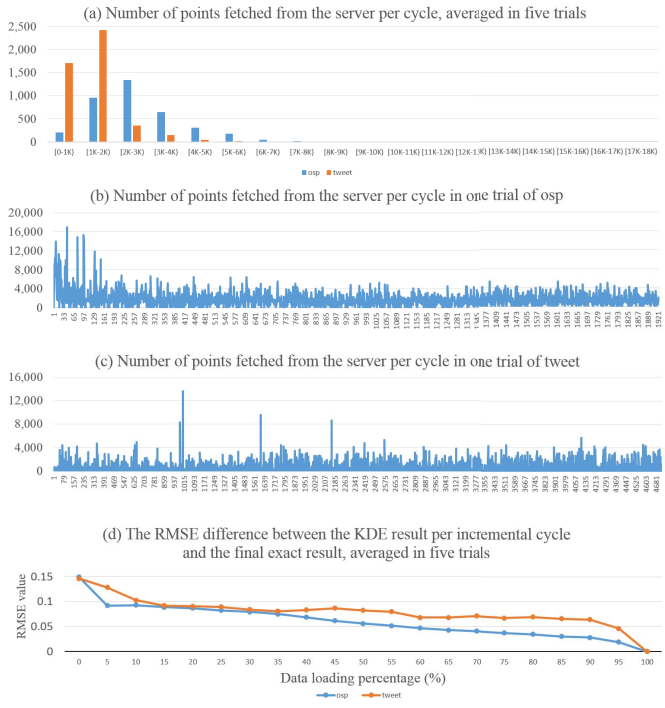

Fig. 5: Statistics of the experiment testing our proposed framework in five trials. In (b-c), the x-axis indicates the cycle index.

6 Discussion

Our work is motivated by the challenge one law enforcement agency met to use our visual analytics system VALET [26] to interactively analyze a large scale data. Subject to security policies, their servers cannot be allowed to use, except database functions. Therefore, we propose the incremental spatiotemporal visual analytics system introduced in this paper to enable them to interactively explore data. Our domain users expressed the concern that the solution cannot consume a significant portion of the computer resources, since they need the computer to process their other work as well, and thereby, we developed a client-based application, and allow users to specify the amount of computer memory dedicated to the system. Experiments in Section 5 have demonstrated the capability of our framework to assist end users interactively explore data within the server limit.

The server query performance is vital to assure each cycle can be completed in real time. In our experiment, the entire dataset in the server side database was split into multiple tables, making one table for each temporal bin. We tested that, in the same condition, with the entire data being hosted within a single table, each server query increased by about 400ms in the osp data.

There are two parameters in our proposed incremental spatiotemporal data sampling method, the upper bound to specify the maximum number of points a leaf node is allowed to have and the length of a temporal bin. The upper bound of a node can impact the interactivity of the framework. If it is set to a value inappropriately large, a cycle can take an excessive time to retrieve and compute data. Therefore, the value of the upper bound is decided with the concern of the system responsiveness. Concerning the length of a temporal bin, the shorter the length of the temporal bin, the more cycles the approximate visualization takes. In our experiment, the total number of cycles for loading the 1.7 million tweet data is twice of the 3.2 million osp crime incidents. This difference happens even though the osp data is almost twice as large as the size of the tweet data. The reason is that the temporal bin is a week for the tweet and a month for the osp.

Results in Section 5 reveals that the spatial data distribution can have an impact on the number of points fetched per cycle. The rightmost picture in Figure 2 shows the exact spatial distribution of the tweet data, and the heatmap in Figure 4 overviews the spatial distribution of the osp data. We can see that the tweet data is more spatially concentrated than the osp data, having a relatively high density only in the downtown region. Therefore, in the data organization initialization stage, with the uniform assumption, the majority of tweet nodes have a smaller number of points than the upper bound, and osp has relatively fewer nodes below the upper bound. That can explain the fact that in Figure 5(c), in the first 10% cycles, the number of points fetched in the tweet is much smaller than that of osp in Figure 5(b), since it is a high chance to select nodes in the sparser area. However, for the osp, nodes in denser area has a relatively greater chance to pick, and that is why, in its first 10% cycles, the number of points per cycle is sometimes larger than the upper bound. Along the cycles proceed, the client can refine the data indexing structure through data that have been fetched in previous cycles. That can explain the number of points fetched per cycle in the tweet is quite similar to the osp in the last 60% cycles. Overall, the diverse spatial distribution can significantly affect cycles in the first percentages, and gradually decease the impact in later cycles.

There are some improvement points for our framework in the near future. First, the data density prediction method in Section 4.2 can be enhanced by using more sophisticated spatiotemporal patterns, such as spatial correlations of nodes. Our current strategy merely pertain to the pattern that in the same spatial region, the number of points in one temporal bin is quite possibly close to the count in another bin. Second, the data size demonstrated in the experiment is around several million data, which cannot fully show the capability of our framework. In the future, we will test our framework on a larger data scale, in line with the data scalability showing in existing works (e.g., STORM [34]). The last one is to conduct a user study with domain experts in law enforcement agencies. Since many agencies enforce strict policies to manage and access crime records with designated devices equipped with limited hardware configurations, the domain users would be the target group to acknowledge the benefits of using our proposed framework and provide valuable suggestions to improve the usability.

7 Conclusion

In this paper, we presented a visual analysis client-based framework that enables the interactive exploration of large-scale spatiotemporal data in constrained computer infrastructure settings. Our framework incorporates an incremental data analysis workflow that provides users approximate visual representation in real time and be within the limited computation capability of a client machine. Experiments have validated that our framework can successfully conduct interactive spatiotemporal data exploration in a typical client machine. Our future work will primarily focus on domain user feedback in law enforcement agencies.

REFERENCES

[1] OpenGL. https://www.opengl.org. Accessed: 2017-03-09.

[2] SQLite. https://www.sqlite.org. Accessed: 2017-03-09.

[3] Hashedcubes: Simple, low memory, real-time visual exploration of big data. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):671–680, Jan 2017. doi: 10.1109/TVCG.2016.2598624

[4] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pp. 1363–1375. ACM, New York, NY, USA, 2016. doi: 10.1145/2882903.2882919

[5] V. Benzaken, J.-D. Fekete, P.-L. Hémery, W. Khemiri, and I. Manolescu. Ediflow: Data-intensive interactive workflows for visual analytics. *IEEE International Conference on Data Engineering*, pp. 780–791, April 2011. doi: 10.1109/ICDE.2011.5767914

[6] R. Bruckschen, F. Kuester, B. Hamann, and K. I. Joy. Real-time out-of-core visualization of particle traces. In *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pp. 45–50, Oct 2001. doi: 10.1109/PVGS.2001.964403

[7] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. pp. 59–66, Oct 2008. doi: 10.1109/VAST.2008.4677357

[8] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Adaptive tetrapuzzles: Efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. *ACM Transactions on Graphics*, 23(3):796–803, aug 2004. doi: 10.1145/1015706.1015802

[9] M. Cox and D. Ellsworth. Application-controlled demand paging for out-of-core visualization. In *Proceedings of the 8th Conference on Visualization*, VIS '97, pp. 235–ff. IEEE Computer Society Press, Los Alamitos, CA, USA, 1997.

[10] P. Cudre-Mauroux, E. Wu, and S. Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pp. 109–120, March 2010.

[11] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pp. 10–10. USENIX Association, Berkeley, CA, USA, 2004.

[12] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample + seek: Approximating aggregates with distribution precision guarantee. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pp. 679–694. ACM, New York, NY, USA, 2016. doi: 10.1145/2882903.2915249

[13] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969. doi: 10.1137/1114019

[14] D. Fisher. Big data exploration requires collaboration between visualization and data infrastructures. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA '16, pp. 16:1–16:5. ACM, 2016. doi: 10.1145/2939502.2939518

[15] D. Fisher, I. Popov, S. M. Drucker, and mc schraefel. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1673–1682. ACM, May 2012.

[16] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pp. 247–254. ACM, New York, NY, USA, 1993. doi: 10.1145/166117.166149

[17] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53, Jan. 1997. doi: 10.1023/A:1009726021843

[18] A. Guttman. R-trees: A dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57, June 1984. doi: 10.1145/971697.602266

[19] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. *SIGMOD Rec.*, 26(2):171–182, June 1997. doi: 10.1145/253262.253291

[20] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pp. 679–688, 2006.

[21] A. J. Lattanze. *Architecting Software Intensive Systems: A Practitioners Guide*. Auerbach Publications, Boston, MA, USA, 1st ed., 2008.

[22] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, Dec 2013. doi: 10.1109/TVCG.2013.179

[23] Z. Liu, B. Jiang, and J. Heer. *imMens*: Real-time visual querying of big data. *Computer Graphics Forum (Proceedings of the 15th Eurographics Conference on Visualization 2013)*, 32(3):421–430, 2013. doi: 10.1111/cgf.12129

[24] A. Magdy, A. M. Aly, M. F. Mokbel, S. Elnikety, Y. He, and S. Nath. Mars: Real-time spatio-temporal queries on microblogs. *IEEE 30th International Conference on Data Engineering (ICDE 2014)*, pp. 1238–1241, March 2014. doi: 10.1109/ICDE.2014.6816750

[25] A. Magdy, M. F. Mokbel, S. Elnikety, S. Nath, and Y. He. Mercury: A memory-constrained spatio-temporal real-time search on microblogs. *IEEE 30th International Conference on Data Engineering (ICDE 2014)*, pp. 172–183, March 2014. doi: 10.1109/ICDE.2014.6816649

[26] A. Malik, R. Maciejewski, T. F. Collins, and D. S. Ebert. Visual analytics law enforcement toolkit. In *2010 IEEE International Conference on Technologies for Homeland Security (HST)*, pp. 222–228, Nov 2010. doi: 10.1109/THS.2010.5655057

[27] A. Malik, R. Maciejewski, N. Elmqvist, Y. Jang, D. S. Ebert, and W. Huang. A correlative analysis process in a visual analytics environment. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 33–42, Oct 2012. doi: 10.1109/VAST.2012.6400491

[28] D. Moritz and D. Fisher. What users don't expect about exploratory data analysis on approximate query processing systems. In *Proceedings of the 2Nd Workshop on Human-In-the-Loop Data Analytics*, HILDA'17, pp. 9:1–9:4. ACM, New York, NY, USA, 2017. doi: 10.1145/3077257.3077258

[29] D. Moritz, D. Fisher, B. Ding, and C. Wang. Trust, but verify: Optimistic visualizations of approximate queries for exploring big data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 2904–2915. ACM, New York, NY, USA, 2017. doi: 10.1145/3025453.3025456

[30] D. Papadias, Y. Tao, P. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In *Proceedings 18th International Conference on Data Engineering*, pp. 166–175, 2002. doi: 10.1109/ICDE.2002.994706

[31] M. Peterson and U. S. B. of Justice Assistance. *Intelligence-led Policing: The New Intelligence Architecture*. Bureau of Justice Assistance, 2005.

[32] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. Wiley Publishing, 8th ed., 2008.

[33] C. Silva, Y. jen Chiang, W. Corrła, J. El-sana, and P. Lindstrom. Out-of-core algorithms for scientific visualization and computer graphics. In *Course Notes for IEEE Visualization 2002*, 2002.

[34] L. Wang, R. Christensen, F. Li, and K. Yi. Spatial online sampling and aggregation. 9(3):84–95, nov 2015. doi: 10.14778/2850583.2850584