Efficient Approximate Analytics of Georeferenced Big Data

Subjects: Computer Science, Interdisciplinary Applications | Environmental Sciences | Geography, Physical
Contributor: Isam Mashhour Al Jawarneh, Luca Foschini, Paolo Bellavista

The unprecedented availability of sensor networks and GPS-enabled devices has caused the accumulation of voluminous georeferenced data streams. These data streams offer an opportunity to derive valuable insights and facilitate decision making for urban planning.

Keywords: approximate query processing ; Spark Streaming ; stratified sampling ; spatial sampling ; Douglas–Peucker

1. Introduction

The unprecedented overdependence on the ubiquitous Internet of Things (IoT) in all aspects of our lives has caused the accumulation of massive amounts of multidimensional and heterogeneous data that are mostly georeferenced [1][2][3][4]. To extract insightful knowledge from such an abundance of data, the fast-track adoption of a breed of big data processing frameworks such as Apache Spark has occurred.

In particular, with most data from IoT arriving in streams, unprecedented efforts in the relevant literature have focused on designing and supporting the promotion of a constellation of tools known as data stream processing systems (SPSs) [5]. Those systems typically search for a good trade-off between two indispensable QoS requirements specified in service-level agreements (SLAs): low latency and high accuracy. As those requirements are normally contradicting, where improving one of them leads naturally to the deterioration of the other, a constellation of different SPSs, characterized by different trade-offs and based on approximate computation, has recently emerged [3][4][6][7]. Approximate computing is driven by the fact that users are typically satisfied with approximate results and willing to give up tiny error-bounded losses in accuracy in exchange for a lower latency [4][6]. There are numerous methods applied in approximate computing. However, they are mostly based on either selecting subsets of the arriving data streams or discarding (also known as shedding) extra data loads that may otherwise introduce additional latencies. Sampling is, by and large, one of the most widely adopted methods for such purposes [3].

Embracing randomness has been the defining factor for most sampling methods that have been designed in the relevant literature so far. This, in part, is attributed to the fact that application scenarios were increasingly known to generate data that exhibited normal distributions [3]. However, things have changed dramatically with the emergence of smart environments (e.g., smart cities) and big data scenarios. Currently, those scenarios are generating big data characterized by high skewness and thus are non-uniformly distributed. In short, this means that the dependence on random-based sampling methods is not going to perform well when it comes to approximating geo-statistics (e.g., the average speed of a vehicle moving in a specific trajectory in a city).

Online sampling that is aware of the geographical characteristics of big data has been demonstrated to be computationally expensive and could cause SPSs to halt at high spikes in data arrival rates [6]. This problem is exacerbated by the fact that well-performing spatial sampling algorithms have to be based on advanced data structures as opposed to simple random sampling. In fact, they are mostly based on stratification, which divides the data into groups corresponding to the real geometrical areas where they have been collected in order to be able to draw roughly equal amounts of data objects from each area independently.

2. Spatial Approximate Query Processing

Unprecedently, massive amounts of big, geotagged data streams are hitting database management systems (DBMSs) continuously, with fluctuations in such a way that in peak hours, data size typically overwhelms those systems and far exceeds their computational capacity [3][6][7]. Seeking to derive insights from such data [6], DBMSs need to manage and store such data efficiently. In such cases where size is overwhelming, systems resort to either elasticity or adaptivity [8]. In

the former, systems overprovision or deprovision resources depending on the fluctuation in data arrival rates, whereas in the latter, systems resort to applying approximate query processing (AQP). The disadvantage of elasticity is that it keeps the online DBMS busy with configurations while those resources can be better allocated to data stream processing instead.

The typical form of adaptivity is represented by AQP. This entails reducing the size of the arriving data in such a way that enables the system to operate gracefully in response to the fluctuations in data arrival rates. The most adopted method for AQP in the literature is sampling, which works by selecting a representative subset of the arrived data in a way that guarantees bounded error numbers in exchange for significant gains in processing times and reduced latencies. AQP is highly sought after in the state-of-the-art simply because the tiny loss in accuracy in exchange for a significant gain in speedup is desirable and acceptable by decision makers in smart city scenarios [4].

Authors in [7] proposed an AQP system for online data stream processing. Their system is composed of three main components: data learning, dynamic sampling strategy, and error control. Their sampling strategy is based on stratification and binary-trees division. They split arriving data streams into tree-based nodes (nodes are analogous to partitions in data distributed parallel processing terms). The other component is a loop-feedback-based error controller to compute the sample sizes adaptively for subsequent processing cycles. The shortcoming of this framework includes the fact the stratification methodology is expensive as it is tree-based. Also, the system is unaware of spatial characteristics and is not reliable for spatial online data processing.

In the same vein, SnappyData [9] has been designed to operate over Apache Spark [10] and incorporates AQP for data stream processing and analytics. It features data structure modules that balance latency and accuracy QoS SLA targets. However, it is not optimized for spatial AQP and data analytics. It does not host a spatial-aware sampler.

Authors in [11] have designed Taster, which encapsulates a few components, including a collector that employs synopses reduction techniques as front stages to select a subset of data. In addition, it hosts a tuner and cost planner, which is responsible for producing on-the-fly approximate running plans. Plans are then run against a synopsis (e.g., samples or sketches) that either persists in a data warehouse or is poured from an online data stream. Plans should be able to meet the accuracy/latency targets from an SLA. Those plans are passed to a tuner, which is then responsible for selecting the plan that optimizes the performance. The system then utilizes the selected plan to capture the synopsis (persisted in disk or from online data streams). The system then computes approximate results from those synopses to answer a query posed by the user. This system, however, is not designed with spatial data processing in mind; it also regularly stores synopsis in disks, which adds extra overhead.

Stock versions of those AQP systems do not include over-the-counter support for geospatial data processing, which leaves handling those logistics to the presentation layer, thus overwhelming the shoulders of front-end developers and distracting their focus orientation away from the main task of developing dependable spatial data stream analytics. This is so because geospatial data are parametrized and represented in the form of pairs of coordinates (typically longitudes and latitudes). Having said that, spatial data lose their characteristics while moving and floating around in a network, and bringing them back to their multidimensional shape is a computationally expensive task that normally involves costly spatial join processing [12][13]. In conclusion, for a data stream processing system to be able to work with big geospatial data at scale with QoS guarantees, it must feature by-design products that incorporate support for geospatial data processing and be aware of other spatial characteristics intrinsically. It also should support interactive and proactive mechanisms to respond to sudden spikes in data arrival rates in such a manner that guarantees, to a good extent, the stability of the system. It should be well-noted, however, that online spatial sampling is challenging basically because of the multidimensionality of data. Stock versions of state-of-the-art data stream processing engines do not currently offer spatial awareness and AQP with QoS guarantees together [3][6][13][14].

Within the same consortium, ApproxHadoop [15] features a two-level sampling mechanism atop Apache Hadoop, which drops tuples to reduce data size in order to reduce I/O overhead. Nevertheless, it relies on an offline profiling method for reconfiguring the latency/accuracy estimates and, therefore, is hardly applied for data streaming workloads.

Other big data systems employ sampling for spatial data partitioning in the Cloud. For example, Simba [16] and SpatialHadoop [17]. However, they do not work for dynamic spatial data stream processing scenarios, where data arrival rates fluctuate periodically.

Also, EXPLORA [18] has been designed to support AQP and geo-visualization of spatiotemporal data at scale with QoS guarantees. It hosts a sampling method that selects online representative samples of arriving data, which aims to speed

up processing while keeping accuracy in check. However, it does not host a latency-aware controller that pulses the fluctuation in data arrival rates and reacts proactively.

Also, spatial-aware approximate query processing (SAQP) is employed for integrating calibrated granular heterogeneous spatially tagged pollution and mobility data for joint analytics at scale with QoS guarantees, as in [19]. Authors have designed EMDI (Environmental Mobility Data Integrator) for integrating mobility data with environmental data, e.g., pollution, climatological, and meteorological data, for complex unified analytics with QoS guarantees. It features a sampler that samples data from both worlds for unified AQP analytics.

However, those systems do not take advantage of the line simplification algorithms. As stipulated in Tobler's first law of geography, objects that are closer in real geometries are more related. Taking advantage of this fact has been widely proven in the literature [3][4][6][20][21]. Having said so, most analytics do not consider spatial objects floating on the outskirts of cities, and most analytics are focused inside cities and livable areas. Furthermore, line simplification algorithms are indispensable for efficient, timely analytics of big georeferenced data with QoS guarantees, such as reducing latency while keeping accuracy in check.

3. Line Generalization Algorithms

Vector line generalization algorithms can be divided into several categories, as discussed in [22]. Those are independent point algorithms (such as the nth point routine and the routine of random selection of points), local processing routines (such as the routine of distance between points and the perpendicular distance routine), unconstrained extended local processing (such as the Reumann–Witkam routine [23] and the sleeve-fitting polyline simplification algorithm [24]), constrained extended local processing (such as the Opheim simplification algorithm and Lang simplification algorithm [25]), and global routines (such as Douglas–Peucker algorithm [26] and an algorithm by Visvalingam and Whyatt [27]). For example, the independent point algorithms work by grouping points on a line into independent sets of consecutive points and then retaining random points. Local processing routines depend on the idea of eliminating points within a radial or perpendicular distance that is less than a user-supplied threshold (tolerance bandwidth) while retaining points within a distance greater than that of the tolerance threshold. From the family of unconstrained extended local processing, the Reumann–Witkam routine [23] works by first passing a strip (rectangular) that shifts stepwise through all segments of an original line, where the start and end points within the strip are retained while in-between points are eliminated. The sleeve-fitting polyline simplification algorithm works in a way similar to that of the Reumann–Witkam routine, where the strip is known as a sleeve that is passed through the original segments of the original line. The Opheim simplification algorithm from the category of constrained extended local processing algorithms works in a way that is similar to that of unconstrained extended local processing routines, in addition to minimum and maximum additional constraints applied within a search region (similar to strips and sleeves), where points within the minimum tolerance or within the search region bounded by the maximum tolerance are removed, while the other points of the original line are retained. Lang simplification algorithm works in a similar way as compared to the Opheim simplification algorithm utilizing search regions, in addition to a distance of intermediate points to a segment connecting start and end points in the region, which should not exceed a user-supplied tolerance for in-between points to be retained.

Global routines, e.g., DP, take the entire line into consideration while processing, contrary to what appears in the other four categories, which consider segments of the entire line stepwise from the start point to the end point. Visvalingam–Whyatt from this category works on the basis of eliminating the original point with the minimum effective area (triangular area formed by any point and its direct in-between neighboring points). The DP algorithm performs favorably for tiny weeding (minor simplification), while the Visvalingam–Whyatt algorithm wins for eliminating entire shapes (e.g., caricatural generalization) [22].

Extensive experiments conducted by [22] show that the Douglas–Peucker algorithm produces the most accurate generalization in terms of measures such as mean distance and mean dissimilarity (which means it has the best performance in the language of shape distortion).

4. Applications of Line Simplification in Approximate Geospatial Analysis

Line generalization has been widely adopted in several aspects related to complex geospatial analysis, including trajectory compression. For example, a recent work by [28] proposed to parallelize several trajectory compression algorithms atop Apache Spark, including an algorithm that is based on Douglas–Peucker. So, it has been applied to trajectories formed from GPS points so that trajectories are lines formed by connecting points, and then parts of those

lines are cut, and their corresponding points are discarded accordingly. The process does not involve reducing a polygon; thus, no spatial join is involved.

In the same vein, ref. [29] have designed a novel geospatial-aware big data analytics framework that features trajectory compression using DP algorithm line simplification as an integral part of its operation. The same applies to a work by [30], where authors proposed an Enhanced Douglas–Peucker (EDP) algorithm which applies a group of enhanced spatial–temporal constraints (ESTC) while simplifying trajectory data, aiming basically to preserve few spatial–temporal features while applying the line simplification for compressing trajectories.

Within the same consortium, ref. [31] designed a novel method to unleash hotspot routes in mobility data by employing massive taxi trajectory data. To reduce the burden on storage and processing resources, they employed the DP algorithm as a quick-and-approximate filter in the front stage of their system to reduce the number of trajectory points accepted for clustering downstream in their system.

Also, variations of the plain application of the DP algorithm are available. For example, ref. [32] proposed a velocity-preserving trajectory DP-based simplification algorithm, which divides the source trajectory into sub-trajectories based on their average velocity such that the velocity in every sub-trajectory differs largely from the others. This way, each sub-trajectory is assigned a different threshold based on those figures; thereafter, each sub-trajectory is simplified using the plain DP algorithm independently, then all simplified sub-trajectories will be merged into a single simplified trajectory.

In addition, ref. [33] designed an adaptive DP-based algorithm with automatic thresholding for AIS-based vessel trajectory compression. Also, another DP-based algorithm has been designed by [34] for compressing automatic identification system (AIS) ocean massive datasets. Also, in oceanography, refs. [35][36][37] designed a DP-based algorithm for the compression of AIS ocean ship data.

Another line of applications of line simplification algorithms for scale-aware geospatial data analytics includes big georeferenced data visualization. In this direction, ref. [38] applied a DP-based algorithm for reducing the georeferenced data to be visualized at scale with QoS guarantees. Authors in [39] propose a novel algorithm that they term locally adaptive line simplification (LOCALIS) for GPU-based geographic polyline data visualization at scale with quality guarantees. Similar work appears in [40], where a method for Level-of-Details (LoD) visualization for polygonal data was designed, which incorporates a polygon simplification method that is based on the DP algorithm. Ref. [41] applies the DP algorithm for generating trajectory data for thematic heatmaps at the city scale for tourist activity analysis. Similarly, authors in [42] have designed an approach for the efficient generation of heatmaps using methods based on the DP algorithm. Also, ref. [43] designed RectMap by combining DP-based simplification with a gridding algorithm to generate simplified versions of plain reference maps.

The picture that emerges from the recent state-of-the-art is the following: systems are mostly run on beefed-up servers and not deployed in parallel. Also, they are not applied with stratified-like spatial sampling, and they are mostly applied for simplifying polylines, not polygons. Simplifying polygons has a great impact on the processing and qualified analytics of big geospatial data streams for several reasons, including the fact that georeferenced sensor data are parametrized and typically need to be joined with polygons for insightful analytics, which adds extra I/O and computational overheads. Also, sending those polygons to the Edge node in Edge–Cloud deployments that are designed for spatial data analytics means adding extra layers of network overhead that slow down the performance overall. Also, federated learning is a line of machine learning (ML) that parallelizes ML algorithms so that they run on Edge devices near the data, so being able to join the spatial data quickly on those devices with polygons data can provide significant speedups.

References

1. Jiang, D. The construction of smart city information system based on the Internet of Things and cloud computing. Comput. Commun. 2020, 150, 158–166.
2. Chen, G.; Zou, W.; Jing, W.; Wei, W.; Scherer, R. Improving the Efficiency of the EMS-Based Smart City: A Novel Distributed Framework for Spatial Data. IEEE Trans. Ind. Inform. 2022, 19, 594–604.
3. Al Jawarneh, I.M.; Bellavista, P.; Corradi, A.; Foschini, L.; Montanari, R. Spatially Representative Online Big Data Sampling for Smart Cities. In Proceedings of the 2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Pisa, Italy, 14–16 September 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.

4. Al Jawarneh, I.M.; Bellavista, P.; Corradi, A.; Foschini, L.; Montanari, R. QoS-Aware Approximate Query Processing for Smart Cities Spatial Data Streams. Sensors 2021, 21, 4160.
5. Armbrust, M.; Das, T.; Torres, J.; Yavuz, B.; Zhu, S.; Xin, R.; Ghodsi, A.; Stoica, I.; Zaharia, M. Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark. In Proceedings of the 2018 International Conference on Management of Data, Houston, TX, USA, 10–15 June 2018.
6. Al Jawarneh, I.M.; Bellavista, P.; Foschini, L.; Montanari, R. Spatial-Aware Approximate Big Data Stream Processing. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
7. Wei, X.; Liu, Y.; Wang, X.; Gao, S.; Chen, L. Online adaptive approximate stream processing with customized error control. IEEE Access 2019, 7, 25123–25137.
8. Herbst, N.R.; Kounev, S.; Reussner, R. Elasticity in cloud computing: What it is, and what it is not. In Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), San Jose, CA, USA, 26–28 June 2013; pp. 23–27.
9. Ramnarayan, J.; Mozafari, B.; Wale, S.; Menon, S.; Kumar, N.; Bhanawat, H.; Chakraborty, S.; Mahajan, Y.; Mishra, R.; Bachhav, K. Snappydata: A hybrid transactional analytical store built on spark. In Proceedings of the 2016 International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016; pp. 2153–2156.
10. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache spark: A unified engine for big data processing. Commun. ACM 2016, 59, 56–65.
11. Olma, M.; Papapetrou, O.; Appuswamy, R.; Ailamaki, A. Taster: Self-tuning, elastic and online approximate query processing. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 482–493.
12. Al Jawarneh, I.M.; Bellavista, P.; Casimiro, F.; Corradi, A.; Foschini, L. Cost-effective strategies for provisioning NoSQL storage services in support for industry 4.0. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 01227–01232.
13. Al Jawarneh, I.M.; Bellavista, P.; Corradi, A.; Foschini, L.; Montanari, R. Efficient QoS-Aware Spatial Join Processing for Scalable NoSQL Storage Frameworks. IEEE Trans. Netw. Serv. Manag. 2020, 18, 2437–2449.
14. Al Jawarneh, I.M.; Bellavista, P.; Corradi, A.; Foschini, L.; Montanari, R. Big Spatial Data Management for the Internet of Things: A Survey. J. Netw. Syst. Manag. 2020, 28, 990–1035.
15. Goiri, I.; Bianchini, R.; Nagarakatte, S.; Nguyen, T.D. Approxhadoop: Bringing approximations to mapreduce frameworks. In Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, Istanbul, Turkey, 14–18 March 2015; pp. 383–397.
16. Xie, D.; Li, F.; Yao, B.; Li, G.; Zhou, L.; Guo, M. Simba: Efficient in-memory spatial analytics. In Proceedings of the 2016 International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016; pp. 1071–1085.
17. Eldawy, A.; Mokbel, M.F. Spatialhadoop: A mapreduce framework for spatial data. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1352–1363.
18. Ordonez-Ante, L.; Van Seghbroeck, G.; Wauters, T.; Volckaert, B.; De Turck, F. EXPLORA: Interactive Querying of Multidimensional Data in the Context of Smart Cities. Sensors 2020, 20, 2737.
19. Al Jawarneh, I.M.; Foschini, L.; Bellavista, P. Efficient Integration of Heterogeneous Mobility-Pollution Big Data for Joint Analytics at Scale with QoS Guarantees. Future Internet 2023, 15, 263.
20. Al Jawarneh, I.M.; Bellavista, P.; Corradi, A.; Foschini, L.; Montanari, R. Efficient Geospatial Analytics on Time Series Big Data. In Proceedings of the ICC 2022-IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 3002–3008.
21. Al Jawarneh, I.M.; Bellavista, P.; Corradi, A.; Foschini, L.; Montanari, R. Efficiently Integrating Mobility and Environment Data for Climate Change Analytics. In Proceedings of the 2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Porto, Portugal, 25–27 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5.
22. Shi, W.; Cheung, C. Performance evaluation of line simplification algorithms for vector generalization. Cartogr. J. 2006, 43, 27–44.
23. Reumann, K.; Witkam, A. Optimizing Curve Segmentation in Computer Graphics; International Computing Symposium: Amsterdam, The Netherlands, 1974.
24. Zhao, Z.; Saalfeld, A. Linear-time sleeve-fitting polyline simplification algorithms. Proc. AutoCarto 1997, 13, 214–223.

25. Lang, T. Rules for the robot draughtsmen. Geogr. Mag. 1969, 42, 50–51.
26. Douglas, D.H.; Peucker, T.K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartogr. Int. J. Geogr. Inf. Geovis. 1973, 10, 112–122.
27. Visvalingam, M.; Whyatt, J.D. Line generalization by repeated elimination of points. In Landmarks in Mapping; Routledge: Oxfordshire, UK, 2017; pp. 144–155.
28. Xiong, W.; Wang, X.; Li, H. Efficient Large-Scale GPS Trajectory Compression on Spark: A Pipeline-Based Approach. Electronics 2023, 12, 3569.
29. Gao, S.; Li, M.; Rao, J.; Mai, G.; Prestby, T.; Marks, J.; Hu, Y. Automatic urban road network extraction from massive GPS trajectories of taxis. In Handbook of Big Geospatial Data; Springer: Berlin/Heidelberg, Germany, 2021; pp. 261–283.
30. Qian, H.; Lu, Y. Simplifying GPS Trajectory Data with Enhanced Spatial-Temporal Constraints. ISPRS Int. J. Geo-Inf. 2017, 6, 329.
31. Zheng, L.; Feng, Q.; Liu, W.; Zhao, X. Discovering trip hot routes using large scale taxi trajectory data. In Advanced Data Mining and Applications, Proceedings of the 12th International Conference, ADMA 2016, Gold Coast, QLD, Australia, 12–15 December 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 534–546.
32. Lin, C.-Y.; Hung, C.-C.; Lei, P.-R. A velocity-preserving trajectory simplification approach. In Proceedings of the 2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI), Hsinchu, Taiwan, 25–27 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 58–65.
33. Liu, J.; Li, H.; Yang, Z.; Wu, K.; Liu, Y.; Liu, R.W. Adaptive douglas-peucker algorithm with automatic thresholding for AIS-based vessel trajectory compression. IEEE Access 2019, 7, 150677–150692.
34. Zhou, Z.; Zhang, Y.; Yuan, X.; Wang, H. Compressing AIS Trajectory Data Based on the Multi-Objective Peak Douglas–Peucker Algorithm. IEEE Access 2023, 11, 6802–6821.
35. Tang, C.; Wang, H.; Zhao, J.; Tang, Y.; Yan, H.; Xiao, Y. A method for compressing AIS trajectory data based on the adaptive-threshold Douglas-Peucker algorithm. Ocean Eng. 2021, 232, 109041.
36. Lee, W.; Cho, S.-W. AIS Trajectories Simplification Algorithm Considering Topographic Information. Sensors 2022, 22, 7036.
37. Zhao, L.; Shi, G. A method for simplifying ship trajectory based on improved Douglas–Peucker algorithm. Ocean Eng. 2018, 166, 37–46.
38. Ma, S.; Zhang, S. Map vector tile construction for arable land spatial connectivity analysis based on the Hadoop cloud platform. Front. Earth Sci. 2023, 11, 1234732.
39. Amiraghdam, A.; Diehl, A.; Pajarola, R. LOCALIS: Locally-adaptive Line Simplification for GPU-based Geographic Vector Data Visualization. In Computer Graphics Forum; Wiley Online Library: Hoboken, NJ, USA, 2020; Volume 39, pp. 443–453.
40. Wu, M.; Chen, T.; Zhang, K.; Jing, Z.; Han, Y.; Chen, M.; Wang, H.; Lv, G. An Efficient Visualization Method for Polygonal Data with Dynamic Simplification. ISPRS Int. J. Geo-Inf. 2018, 7, 138.
41. Sasaki, I.; Arikawa, M.; Lu, M.; Sato, R. Thematic Geo-Density Heatmapping for Walking Tourism Analytics using Semi-Ready GPS Trajectories. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 4944–4951.
42. Sasaki, I.; Arikawa, M.; Lu, M.; Sato, R. Mobile Collaborative Heatmapping to Infer Self-Guided Walking Tourists' Preferences for Geomedia. ISPRS Int. J. Geo-Inf. 2023, 12, 283.
43. Sun, G.; Zhai, S.; Li, S.; Liang, R. RectMap: A Boundary-Reserved Map Deformation Approach for Visualizing Geographical Map. Chin. J. Electron. 2018, 27, 927–933.