International Journal of Digital Earth

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tjde20

Utilizing serverless framework for dynamic visualization and operations in geospatial applications

Mingke Li, Charles Tousignant, Chiranjib Chaudhuri & Achraf Chabbouh

To cite this article: Mingke Li, Charles Tousignant, Chiranjib Chaudhuri & Achraf Chabbouh (2024) Utilizing serverless framework for dynamic visualization and operations in geospatial applications, International Journal of Digital Earth, 17:1, 2392835, DOI: 10.1080/17538947.2024.2392835

To link to this article: https://doi.org/10.1080/17538947.2024.2392835

Published online: 09 Sep 2024.

INTERNATIONAL JOURNAL OF DIGITAL EARTH
2024, VOL. 17, NO. 1, 2392835
https://doi.org/10.1080/17538947.2024.2392835

Utilizing serverless framework for dynamic visualization and operations in geospatial applications

Mingke Li, Charles Tousignant, Chiranjib Chaudhuri and Achraf Chabbouh

Research and Development Group, Geosapiens Inc., Quebec City, Canada

ABSTRACT

While substantial efforts have been invested in the development of Discrete Global Grid Systems (DGGS) spatial operations and their potential applications in the geospatial domain, it has become evident that there is a demand for an efficient and scalable system to handle the visualization of large-scale DGGS data. This study demonstrated the potential of DGGS in conjunction with the serverless framework for dynamic visualization at various resolutions, which is based on data storage and effective querying using PostgreSQL integrated into Amazon Aurora Serverless. The use of Amazon Web Services (AWS) Lambda for on-the-fly generation of hexagon geometries significantly reduced the storage requirements and improved the speed of the visualization process. In addition, we implemented on-the-fly spatial operations including point binning, thresholding, aggregation, and neighborhood operations in the DGGS, highlighting the capabilities of DGGS in vector and raster processing. The proposed system has shown promising results in terms of efficiency, scalability, and adaptability, making it a viable solution for large-scale geospatial data processing and visualization. Case studies using flood risk data and terrain data further illustrate the system's practical applicability in on-the-fly spatial operations and rapid visualization.

ARTICLE HISTORY
Received 1 March 2024
Accepted 9 August 2024

KEYWORDS
Discrete global grid systems; serverless; PostgreSQL; geo-visualization; on-the-fly operations;

1. Introduction

Geospatial visualization is the process of representing geographical data in a visual format, often through interactive graphics, to gain insights into spatial relationships, patterns, and trends. It typically serves as the final stage in the data analysis pipeline, acting as the interface through which the insights and patterns extracted from raw data are conveyed to end-users or decision-makers. In the real world, it finds application in diverse fields such as urban planning, environmental monitoring, and disaster management, assisting resource allocation, problem-solving, and decision-making through spatial context. However, on-the-fly geospatial visualization faces challenges in handling vast and dynamic datasets in real-time, ensuring responsive user experiences, and maintaining data accuracy.

Recent studies have demonstrated the importance of Discrete Global Grid Systems (DGGS) in revolutionizing the processing, analysis, and visualization of geospatial data. As a standard spatial reference system adopted by the Open Geospatial Consortium (OGC), DGGS hierarchically partitions the Earth's surface into almost uniform cells, offering a structured geospatial data

CONTACT Mingke Li erin.li@geosapiens.ca 20 Boulevard Charest O suite 102, Québec, QCG1K 1X2, Canada

framework, and facilitating scalable data organization (OGC 2019). Bousquin (2021) emphasized the advantages offered by DGGS implementations such as flexibility, scalability, and effective data aggregation and visualization, which underscored the application of DGGS in characterizing environments. Rawson, Sabeur, and Brito (2021) highlighted how DGGS went beyond grids to produce accurate and illuminating risk maps, particularly in global maritime risk assessment. A framework for a DGGS geoprocessing language was introduced to facilitate building a Digital Earth platform combining various geospatial data sources (Peterson and Shatz 2019). Chaudhuri, Gray, and Robertson (2021) introduced a novel flood inundation modeling framework using a big-data DGGS and web-GIS platform to improve the accessibility and computational efficiency of flood risk models. Bondaruk, Roberts, and Robertson (2020) thoroughly assessed the state of DGGS and emphasized the urgent need for fresh methods of data integration, analysis, and visualization due to the increasing availability of geospatial data sources. These contributions highlighted the capacity of DGGS to provide scalability and adaptability in addressing real-world geospatial challenges. A comprehensive review of spatial operations and applications within the DGGS context has been conducted in Section 2.

Previous research has also demonstrated an interest in the development of visualization systems based on DGGS. For example, a simulation – visualization system was developed based on DGGS, which integrated statistical conditional simulation, array set addressing, and a visualization platform to achieve efficient storage, computation, and near real-time visualization of spatial data (Stough et al. 2020). Particularly, fine-resolution data is generated using a spatial statistical conditional simulation methodology that constrains the simulation output to replicate features of a physical model based on scientific knowledge about the structure of the true physical process (Stough et al. 2020). The study by Djavaherpour, Mahdavi-Amiri, and Samavati (2017) also proposed a strategy that combines fabrication with DGGS, which offered a way to create models representing Earth for visualizing datasets. Raposo, Robinson, and Brown (2019) implemented a visualization system specifically based on the Quaternary Triangular Mesh (QTM; Dutton 1999), to demonstrate the modifiable areal unit problem (MAUP). The system allows users to bin global data points with counts or measures of any theme at multiple levels. Users can interactively select the binning level and translate the tessellation for visualization on a virtual globe. This demonstrates the scaling and zoning aspects of the MAUP with dynamically drawn choropleths and various quantile classifications (Raposo, Robinson, and Brown 2019).

Nevertheless, there remains a demand for a scalable visualization system to effectively render DGGS data, particularly when confronted with the challenge of handling immense cell geometries associated with large-scale datasets. This paper aims to address this need by integrating DGGS with the serverless framework for dynamic visualization. We implemented on-the-fly spatial operations, such as point binning, thresholding, down-sampling, and neighborhood operations within the context of DGGS, and visualized the analysis results utilizing a serverless framework. To illustrate the practicality of our approach, we have conducted two case studies within the domain of flood risk management and terrain data management.

The OGC Specification on DGGS does not define a single configuration but rather outlines a set of criteria that a DGGS should fulfill. In this paper, we specifically adopted the Icosahedral Snyder Equal Area aperture 3 Hexagon (ISEA3H) DGGS for our system. The advantages of the ISEA3H tessellation have been previously highlighted in comparison to other DGGS configurations (Sahr, White, and Kimerling 2003; White et al. 1998). First, as a polyhedral map projection, the Icosahedral Snyder Equal Area projection minimizes area distortion (White et al. 1998). Second, hexagonal grids are characterized by the greatest angular resolution, uniform adjacency, and accurate approximation of Cartesian distance (Golay 1969; Conway and Sloane 1998; Luczak and Rosenfeld 1976; Sahr 2011). Sampling over hexagonal grids offers noticeable advantages because of their uniform adjacency compared to other regular grids with rectangular or triangular geometries. Lastly, the hexagonal tessellation with aperture 3 benefits smoother transitions between DGGS resolution levels than apertures of four or seven (Mahdavi-Amiri, Alderson, and Samavati 2015). The

centroid-aligned, aperture 3 hexagonal tessellation ensures that a parent cell's six vertices are centroids of its child cells (Sahr, White, and Kimerling 2003), leading to the important monotonic convergence characteristic where the representing centroid is infinitely closer to the point to be modeled at finer resolutions. The DGGRID library was compiled into our system architecture because it has demonstrated promising functionality in constructing grids with desired geometric properties (Kmoch et al. 2022; Sahr 2022). The remainder of this paper is arranged as follows. Section 2 conducts a comprehensive literature review on DGGS operations and applications. Section 3 provides an in-depth description of our system architecture. Section 4 describes the quantization processes including both the feature and raster dataset. Section 5 demonstrates the spatial operations that have been implemented within our system. In Section 6, we present two case studies within the domain of flooding risk management and terrain data management. Section 7 discusses the research and outlines potential directions for future work.

2. Literature review on DGGS operations and applications

2.1. Fundamental operations in DGGS

The fundamental operations mandated by the OGC specification include quantization, spatial relation, and interoperability (OGC 2019), forming the solid foundation for various analytical processes, visualization techniques, and practical applications of DGGS. Quantization operations serve as an essential process aimed at converting diverse geospatial data into a uniform format accessible within the context of DGGS. A general understanding of quantization includes dataset crawling, conversion into an internal format, integration, aggregation, and quality control, among which the key step is the conversion between cell addresses and geographic coordinates or among cell addresses. Based on this, previous studies adopted various approaches to quantizing values of data sources on the DGGS cells depending on the data model and nature of the source data, where the cell centroid is commonly used to represent a cell's value. For the raster model, Robertson et al. (2020) extracted the underlying raster value for each DGGS cell centroid where the mean value and nearest pixel value were used for continuous data and categorical data, respectively. Bilinear interpolation was also applied when quantizing raster data, such as integrating multi-source terrain data into a standard DGGS framework and resampling GF-1 satellite imageries on hexagonal discrete grids (Li, McGrath, and Stefanakis 2021; Ma et al. 2021). A precise approach for hexagonal pixel modeling was developed to convert hexagonal DGGS cells into quadrilateral pixels (Liang et al. 2024). These pixels are then stored as regular rectangles with a fixed pattern in any standard format, facilitating the compatibility (Ding et al. 2024). For the vector model, Li and Stefanakis (2020) explored different types of geo-feature modeling in discrete global grids, where geographic coordinates of spatial points were directly converted to the corresponding DGGS cell index at a certain resolution, line features were transformed to a sequence of DGGS cells where they intersect, and polygons were converted to a group of cells whose values were determined by the intersected polygon with dominant area proportions. Robertson et al. (2020) further stored the order of cells when modeling line features and identified interior and boundary cells when modeling polygons.

Spatial relations in the DGGS context involve cell-level and object-level relation queries. Cell-level spatial relation includes sibling navigation and parent–child navigation, which have found support in a few open-source libraries like DGGRID and H3 (Sahr 2022; Uber 2019). The efficiency of spatial relation queries is greatly influenced by the underlying cell indexing mechanism. For instance, hierarchical indexing systems prove advantageous for parent–child relation queries, whereas coordinate-based indexing systems facilitate sibling relation queries (Mahdavi-Amiri, Alderson, and Samavati 2015). As for object-level relations, query engines tend to be more complex. In the context of traditional Geographic Information Systems (GIS), topology is commonly represented by the Dimensionally Extended 9-intersection Model (DE-9IM; Egenhofer and Herring

1990). It is conceivable that this model can be extended to support topology modeling within the context of DGGS, though further deliberation is required (Hojati et al. 2022).

Interoperability serves as a pivotal requirement for a mature DGGS acting as a Spatial Data Infrastructure (SDI), meaning the system's capability to communicate with end-users, alternative DGGS, or other SDIs through standardized APIs and data formats. Such capability is essential for enabling seamless data exchange and collaboration. While advancements have been made in enhancing interoperability (Mahdavi-Amiri, Harrison, and Samavati 2016), this aspect remains subject to ongoing research and development.

2.2. State-of-the-art DGGS libraries

Various open-source DGGS libraries have been developed, including OpenEAGGR, DGGRID, HEALPix, and rHEALPix (Sahr 2022; Gibb 2016; Gorski et al. 2005; OpenEAGGR 2019). HEALPix, developed by NASA's Jet Propulsion Laboratory (JPL), was originally designed to map the sky above the Earth's surface and store background cosmic microwave energy (JPL 2019). Gibb's rHEALPix DGGS was derived from HEALPix by reorganizing the underlying map projection into the rHEALPix projection and constructing a DGGS for ellipsoids of revolution based on the projection (Gibb 2016). The rHEALPix is particularly useful for latitudinal data analysis due to its iso-latitude property (Bowater and Stefanakis 2019). DGGRID and OpenEAGGR are libraries that offer end-users the flexibility to select from various DGGS configurations and support cell navigations, such as hierarchical search and neighborhood search (OpenEAGGR 2019; Sahr 2022). Additionally, Uber H3 and Google S2 are well-established open-source packages with DGGS capabilities. H3 utilizes an icosahedral hexagonal DGGS with a refinement ratio of seven and includes various spatial operators (Uber 2019). On the other hand, Google S2 provides operations for computational geometry and spatial indexing on the sphere, featuring a fully congruent aperture 4 refinement with square cell shapes (Veach et al. 2017). However, it's worth noting that H3 and S2 exhibit significant variation in the cell area, making them unsuitable for applications requiring equal area cells (Kmoch et al. 2022). Equi7 Grid is another attempt to create a global grid, dividing the globe into seven zones that align with the continents and defining customized equidistant azimuthal projections for each zone (Bauer-Marschallinger, Sabel, and Wagner 2014). Nonetheless, there are overlaps between various Cartesian spaces at the boundaries between the zones (Bauer-Marschallinger, Sabel, and Wagner 2014).

2.3. Common spatial analysis implemented in DGGS

Prior research has investigated the adaptation and migration of spatial analysis algorithms originally designed for traditional GIS to the context of Discrete Global Grid Systems (DGGS). Notably, researchers have implemented vector-based operations, such as union, intersection, clipping, and buffering, in an in-database manner, leveraging set operations as a foundation (Robertson et al. 2020). Investigations have been conducted into image algebra, including local, focal, and zonal operations (Robertson et al. 2020; Li, McGrath, and Stefanakis 2022a), and topographical and hydrological analysis algorithms were developed accordingly in the hexagonal DGGS, such as terrain slope, flow routing, and watershed delineation (Li, McGrath, and Stefanakis 2022a; 2022b). In addition, cosmological data statistical algorithms were developed in the HEALPix framework, along with the basic geometric operations including distance, area, and angle computation (Fryer, Li, and Olenko 2020). In certain circumstances involving discrete grids, distance can also be interpreted as the number of rings. For example, an inverse distance weighting interpolation technique was utilized to estimate missing temperature data, where the weight was assigned based on the number of cells away from the target location (Bousquin 2021).

The hierarchical nature of DGGS makes multi-resolution spatial aggregation a distinctive operation. This process involves aggregating data across various levels within the DGGS, allowing

for efficient representation and manipulation of geospatial data. In the field of topography, the QTM, among the earliest modern DGGS designs, was specifically introduced for the assembly and management of global terrain data across various scales (Dutton 1984; 1999). Additionally, the Ellipsoidal Cube Map and Crusta were developed to enhance the rendering of global-scale terrain data based on quadtrees (Bernardin et al. 2010; Lambers and Kolb 2012). Aggregation methods based on mean, maximum, and minimum terrain values were also introduced for specific application scenarios (Li, McGrath, and Stefanakis 2021). A down-sampling kernel was defined to facilitate the visualization of internal data representations with efficient storage and computation (Stough et al. 2020). Furthermore, DGGS has been employed as bins with spherical cells to grid and aggregate point data, thereby facilitating spatial statistics without the spatial bias arising from varying cell sizes (Konig et al. 2019; Somveille, Manica, and Rodrigues 2018).

2.4. Real-world applications driven by DGGS

In recent years, DGGS has seen growing utilization across diverse applications aimed at addressing real-world challenges, broadly categorized based on their usage in three main areas. First, DGGS has been employed as bins with spherical cells to grid and aggregate geospatial data, thereby facilitating spatial statistics without the spatial bias arising from varying cell sizes. For instance, Konig et al. (2019) integrated plant diversity data in a DGGS framework, demonstrating the significance of data resolution. Similarly, Somveille, Manica, and Rodrigues (2018) aggregated the global distribution of land bird species on a DGGS framework and explored migratory differences among terrestrial bird species. Spatial gaps in taxonomic knowledge and expertise were explored by mapping taxonomic effort and amphibian diversity on the DGGS, which aimed to examine their relationship with economy and biodiversity (Rodrigues et al. 2010). Qiu et al. (2022) proposed continuous digital expression of large-scale population spatial disaggregation data and improved spatial statistical analysis using DGGS. In addition, a software application based on the QTM was developed to bin global data points geodetically, enabling aggregation with counts or measures of any thematic content across multiple levels (Raposo, Robinson, and Brown 2019). Users can interactively choose the level at which the data are binned by the QTM and explore the scaling and zoning characteristics of the MAUP through the dynamically generated choropleths presented on the surface of the virtual globe.

Second, DGGS serves as a data fabric for heterogeneous data integration and multi-scale data queries due to its cells' uniform properties, often followed by data mining analysis. Rawson, Sabeur, and Brito (2021) integrated multiple maritime datasets, predicting ship grounding occurrences through a random forest algorithm within a DGGS. Li, McGrath, and Stefanakis (2022c) explored multi-scale flood mapping under climate change scenarios using four machine learning algorithms, organizing heterogeneous predictor datasets in DGGS. A combination of DGGS and Artificial Neural Network was employed to predict the spatial distribution of hate crimes in the USA, incorporating numerous social and economic parameters for enhanced accuracy (Jendryke and McClure 2019; 2021). Additionally, Eco-ISEA3H established a machine learning-ready spatial database, quantizing over 3000 variables from 17 sources in DGGS at multiple granularities for ecometric and species distribution modeling (Mechenich and Zliobaite 2023).

Third, because of the discrete nature of DGGS cells, it is possible to realize bottom-up predictive models such as cellular automata and agent-based modeling in DGGS. For example, Hojati and Robertson (2020) coupled cellular automata with DGGS in a distributed database system and carried out a case study to model the spread of wildfire events. Kiester and Sahr (2008) conducted a hierarchical, multi-resolution cellular automaton using a topology-independent discrete simulation library based on DGGS, where the behavior of a central cell was influenced by its neighboring cells, parent cells, and child cells. The study highlighted the effectiveness of cell-based simulation techniques with DGGS on a global scale (Kiester and Sahr 2008). While less

explored, DGGS-driven agent-based modeling holds promise with well-established neighboring navigation rules for grid cells.

3. Description of the architecture

3.1. Backend architecture based on the IDEAS

Our backend data architecture is based on the Integrated Discrete Environmental Analysis System (IDEAS; Robertson et al. 2020). The primary aim of the IDEAS is to establish an operational Geographic Information System (GIS) based on DGGS. This system is specifically tailored for large-scale environmental modeling and analysis. Robertson et al. (2020) demonstrated the feasibility of such a system by implementing it within a relational database environment, integrating commonly used data analytics tools. The IDEAS data model is based on a hybrid relational/key-value database approach, consisting of three core sub-models: the spatial data model, the temporal data model, and the attribute data model, implemented by five main tables, including base, attribute, temporal lookup, geometry, and metadata tables (Figure 1).

The spatial data model in the IDEAS is based on the ISEA3H DGGS. This model defines the base regular polyhedron, cell shapes, subdivision method, and the method for projecting cells onto the Earth's surface. The chosen aperture 3 hexagon grid allows for efficient global-scale processing and accurate representation of the Earth's surface. Here the base table serves as the main table for all cell objects and their associated values, while the geometry table complements it by providing hexagonal geometry information. The temporal data model, implemented by the temporal lookup table,

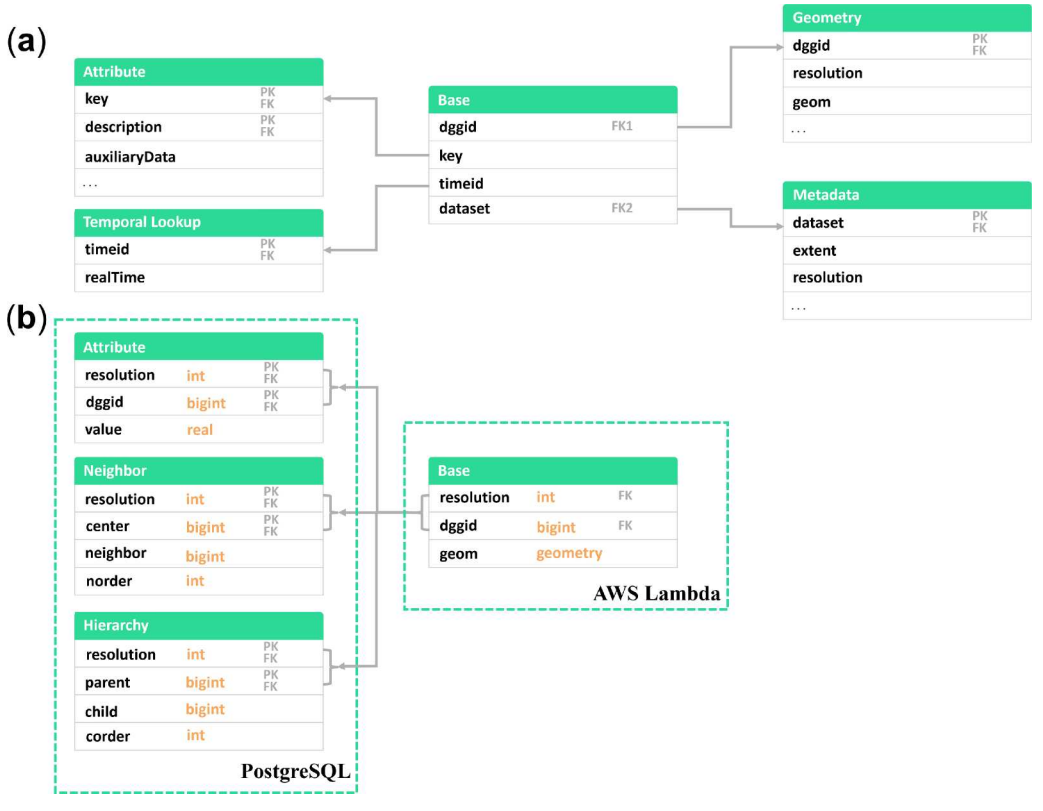

Figure 1. The database structure of (a) the Integrated Discrete Environmental Analysis System (IDEAS), and (b) a modified version of our system.

enables the representation of time durations, accommodating various levels of nesting. The attribute data model stores cell-object attributes as key/value pairs, including thematic, topological, and measurement data, where the attribute table and metadata table provide semantic interpretations of the attribute keys and incorporate dataset-level information.

Different from the IDEAS implementation, our system adopts an on-the-fly approach for generating the base table and DGGS cell geometries through a serverless setup, based on the user's view extent, rather than storing them in the database (Figure 2). Furthermore, within the backend database, we have pre-calculated and stored the neighbor table, which records neighborhood relationships, and the hierarchy table, which records parent–child relationships (Figure 1). The resolution, in conjunction with the cell address, functions as the primary key or foreign key within the system to facilitate the joining of various tables, including the base table, attribute table, neighbor table, or hierarchy table (Figure 1). The rest of Section 3 elaborates on the detailed description of the specific components comprising our system architecture.

3.2. Storage and parallel querying with PostgreSQL integrated on Amazon Aurora Serverless

PostgreSQL is a robust and open-source relational database management system (RDBMS) known for its advanced features, extensibility, and adherence to SQL standards (PostgreSQL 2023). It has gained significant popularity for its capabilities in handling complex data management tasks, supporting both structured and unstructured data, and providing extensibility through custom functions and procedural languages. PostgreSQL has been used as a versatile and scalable database

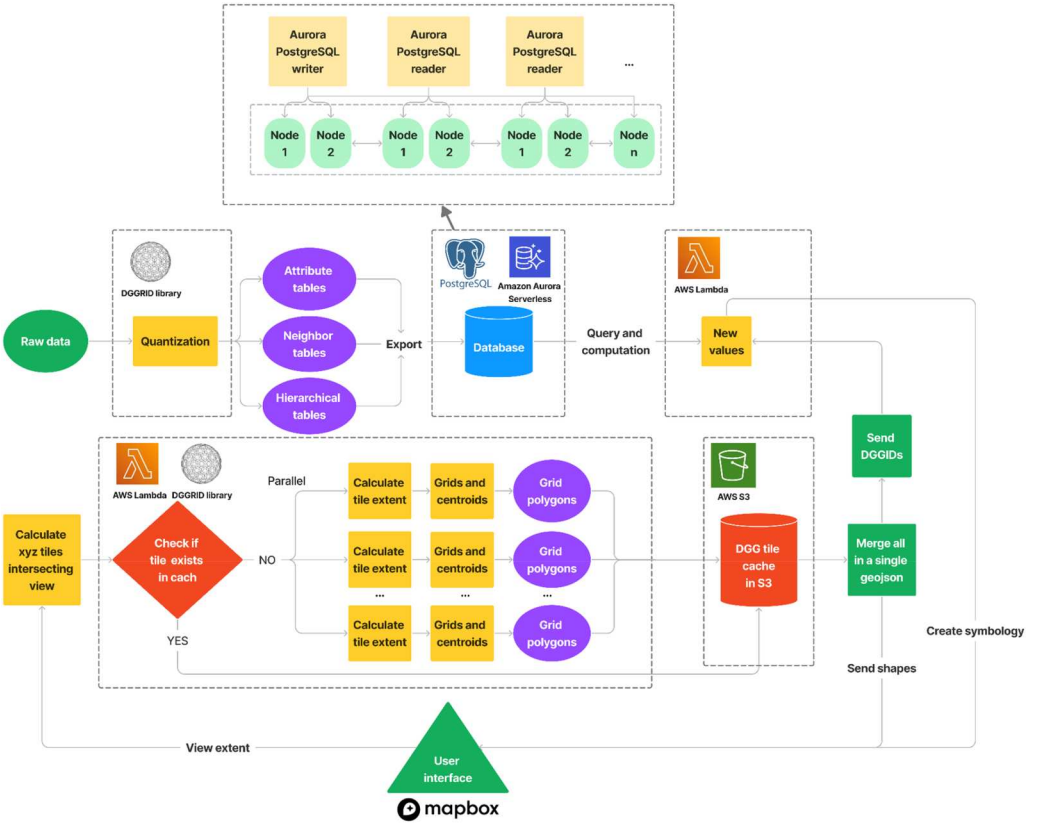

Figure 2. The overall architecture of our system implementation.

solution for various applications from small projects to large enterprise systems. Amazon Aurora stands out as a cloud-native RDBMS that fully embraces the power of PostgreSQL compatibility, and Amazon Aurora Serverless is an on-demand, autoscaling configuration for Amazon Aurora (Amazon 2023).

Our implementation is based on the integration of PostgreSQL and Amazon Aurora Serverless, which combines the strength of PostgreSQL with the scalability and resilience of the cloud. Specifically, attribute tables, neighbor tables, and hierarchical tables across various resolutions were stored in the PostgreSQL integrated into Amazon Aurora Serverless (Figure 2). Within the neighbor table, the fields 'center' and 'neighbor' correspond to the DGGS cell indices of the center cell and its neighboring cells, respectively. The field 'norder' has integers denoting the order of the surrounding cells, ranging from 1 to 6. In the hierarchy table, the 'parent' and 'child' fields represent DGGS cell indices for the parent cell and its child cells, respectively. The 'corder' field also has integers indicating the order of the child cells, starting from 0 to 6, with the center-aligned child cell assigned an order of 0. Both the neighbor table and hierarchy table are generated at various DGGS resolutions using the DGGRID library (Sahr 2022). The process of creating the attribute table, which involves converting raw data into DGGS cell values, is explained in Section 4. We pre-calculated the hierarchical tables, neighborhood tables, and attribute tables as static relations stored in the PostgreSQL integrated on Amazon Aurora Serverless in this stage of development. They are relatively small in size compared to the DGGS geometries and are frequently utilized in diverse spatial operations. With them pre-calculated, the 'spatial operations' are essentially a set of join-operations of relations based on the DGGS cell indices, rather than relying on any topological or geometric computations within the database. Storing them statically ensures stable and efficient performance. The query performance is enhanced by the parallel query capabilities enabled by PostgreSQL-compatible Aurora Serverless (Figure 2), which facilitates operations such as table joining, grouping, and generating descriptive statistics. These operations were essential for processes such as point binning, up-sampling, and down-sampling.

3.3. Parallel processing on serverless AWS Lambda

Our system leverages the serverless framework, which is compatible with various cloud platforms such as Google Cloud, Microsoft Azure, and Amazon Web Services (AWS) Lambda. In our implementation, we specifically employed AWS Lambda. AWS Lambda is a serverless computing paradigm, offering a robust approach to parallelization and enabling efficient task processing across concurrent executions (Baldini et al. 2017). The AWS Lambda functions operate in an event-driven manner, automatically scaling in response to triggers like Application Programming Interface (API) requests or data updates (Klems 2018). As events occur, AWS Lambda dynamically provisions containers to handle parallel requests, ensuring swift execution. Concurrency in AWS Lambda allows multiple instances of a function to operate concurrently, dependent on incoming events and configured concurrency settings. AWS Lambda also provides fine-grained control over concurrency, enabling users to set the number of simultaneous executions based on specific workload requirements (Klems 2018). This automatic scaling capability ensures adequate resources for concurrent execution, minimizing costs and optimizing resource usage (Baldini et al. 2017). AWS Lambda's serverless architecture, characterized by event-driven scalability and fine-tuned parallelization, stands as a powerful choice for cost-effective and efficient parallelized serverless computing (Shafiei, Khonsari, and Mousavi 2022).

In our implementation, AWS Lambda was used in two stages of our workflow (Figure 2). First, AWS Lambda operates as a singular instance responsible for forwarding queries to the Aurora PostgreSQL database. The architecture initially facilitated querying in parallel on a per-tile basis. However, we had to consolidate the requests and submit them as a single query because the pricing model of the database was designed as a charge per query, despite the potential increase in latency. Second, AWS Lambda was employed to dynamically

Table 1. Summary of the data sizes for ISEA3H DGGS geometries all over the globe at various resolutions if stored in the PostgreSQL database.

Resolution	Cell count	Data size
0	12	2.8 KB
1	32	7.5 KB
2	92	21.6 KB
3	272	64.0 KB
4	812	191.1 KB
5	2432	572.2 KB
6	7292	1.4 MB
7	21872	4.3 MB
8	65612	12.8 MB
9	196832	38.5 MB
10	590492	115.4 MB
11	1771472	346.2 MB
12	5314412	1.0 GB
13	15943232	3.0 GB
14	47829692	9.1 GB
15	143489072	27.4 GB
16	430467212	82.2 GB
17	1285401632	245.3 GB

generate hexagon geometries for tiles intersecting the current user view. These geometries were generated on-demand in the geoJSON format and cached in AWS S3. When caching hexagon geometries, the cost-effectiveness of AWS S3 makes it well-suited for caching frequently accessed, static geometries, resulting in faster retrieval times. In other words, geometries were swiftly created by AWS Lambda in parallel instead of being stored in the static PostgreSQL database. To achieve this, we compiled the DGGRID library to run on AWS Lambda. To ensure portability and seamless integration with the AWS Lambda environment, we encapsulated the execution of the DGGRID command within a Docker container hosted on AWS Lambda. This approach eliminates the necessity to manage external dependencies or concerns with environmental variations. The use of a serverless setup conserves significant storage space within the database and enhances the speed and efficiency of the visualization process. Table 1 provides a summary of the data sizes for ISEA3H DGGS geometries all over the globe at resolutions ranging from 0 to 17, showcasing the storage requirements that would be incurred if these geometries were to be stored in the PostgreSQL database. For ISEA3H DGGS where the refinement ratio is three, the size of geometry data approximately triples at the next finer resolution level (Table 1).

While the Hypertext Transfer Protocol (HTTP) requests serve as a prominent example of event-driven triggers for AWS Lambda functions in our study, it is important to note that AWS Lambda can effectively handle various other event sources, including AWS S3 events, DynamoDB feeds, Amazon Simple Notification Service (SNS) notifications, and more. Here we emphasize the use case of HTTP requests because it closely aligns with the specific requirements of our system architecture and the intended functionality of the application. Although the managed PostgreSQL Aurora instance effectively remains active, it operates according to the principles of Amazon Aurora Serverless, where scaling is likewise dictated by demand. This means the database scales up or down as necessary based on incoming requests, following a pay-as-you-go approach. The underlying idea is that charges should only apply to resources that have been used. The Amazon Aurora Serverless offers an auto-pause feature, allowing databases to be paused when not in use. This feature helps save costs by eliminating charges for unused capacity during idle periods. Compared to traditional provisioned PostgreSQL instances, where resources remain static regardless of usage and are allocated based on peak demand, Amazon Aurora Serverless has significant cost optimization benefits by adjusting resource consumption according to the workload.

3.4. Dynamic vector tile-based visualization for DGGS

Our system supports dynamic visualization based on the Google Maps tiling scheme, as shown in Figure 2. The Google map tiling scheme relies on the Web Mercator projection, where the world is divided into square tiles, each represented by a unique x, y, and zoom level. This tiling scheme allows for efficient rendering and display of map data at various zoom levels. Initially, the system captures the view extent via the user interface, which is implemented using Mapbox (Mapbox 2023). Based on the view status of the user, the relevant pre-defined vector tiles are determined. If hexagonal DGGS cells contained within these vector tiles have not been previously stored in AWS S3, they are generated in parallel for each tile using the DGGRID library compiled within AWS Lambda, with the output in geoJSON format. In other words, each tile within the current view serves as a unit for parallel processing, ensuring that at each tiling scheme level, with a corresponding fixed DGGS level, the number of DGGS cells within each tile remains constant and quantifiable. This uniform distribution guarantees an even workload for each thread in parallel processing. These newly generated geometries are combined with any cached geometries in AWS S3 if available, resulting in a single file that stores both geometry information and DGGS cell indices. The cell indices are used to extract values calculated by in-database operations in AWS Lambda, where tables are joined based on the common cell indices (e.g. DGGIDs) and extracted values are used for symbology. Finally, cell geometries are joined with the cell values for user-end visualization.

4. Data preparation

4.1. Quantization of feature dataset

In the current developmental stage of our system, our focus is primarily on modeling spatial points, or centroids of spatial polygons. We perform a direct conversion of their geographic coordinates into the corresponding ISEA3H DGGS cell indices, at a predetermined resolution, using the DGGRID library (Sahr 2022). To facilitate point binning analysis, a coarser DGGS resolution relative to the point density is chosen, which allows the grouping of multiple points together for subsequent summarization and analysis. The specific DGGS resolution used to aggregate point data depends on various factors such as thematic attributes, point density, and the purpose of analysis and visualization. This aligns with the concept of the modifiable areal unit problem in geography, where the outcomes of data aggregation are influenced by the mapmaker's decision regarding the choice of modifiable areal unit for their analysis. Different criteria for determining the aggregation level can lead to different outcomes in analysis and visualization.

4.2. Quantization of raster dataset

For the raster dataset, we employed the largest-share principle in our quantization algorithm for continuous data such as elevation and temperature. As shown in Figure 3, hexagonal cells receive a combination of intersected pixel values with corresponding weights. The weights are determined based on the area of intersection, where pixels with larger intersected areas gain higher weights in influencing the cell's final value. For example, in Figure 3, the quantized value of the colored hexagonal cell is calculated as:

$$v_{hex} = \sum w_i n_i \tag{1}$$

$$w_i = \frac{S_i}{S_{hex}} \tag{2}$$

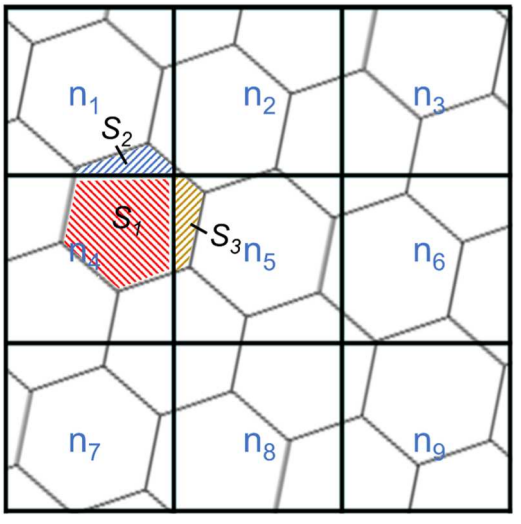

Figure 3. Quantization of raster with continuous data on the ISEA3H DGGS.

where n_i is the original pixel value, w_i is the corresponding weight, S_i is the intersected area of the certain pixel, and S_{hex} is the hexagonal cell area. This is equivalent to the bilinear interpolation when the hexagonal cell's centroid aligns with the pixel vertex, and the cell is uniformly divided by four pixels. The DGGRID library was used when generating hexagon geometries in the raster quantization process (Sahr 2022).

5. On-the-fly spatial operations

5.1. Point binning and spatial statistics

Point binning involves aggregating point feature data using the ISEA3H DGGS cells as spatial unit bins, applying a specified value aggregation rule. Values within each hexagonal cell are calculated from the attribute values of original point features falling within that cell, utilizing statistics such as mean, max, min, deviation, range, count, density, or more sophisticated calculations (Figure 4). The calculation of the binned value can be defined as:

$$v_{bin} = F(v_1, v_1, \ldots, v_n) \tag{3}$$

where v_1 to v_n are the original point values within the target bin, n is the number of points, and F defines the function to be applied to the original point values, such as mean, max, min, count, etc.

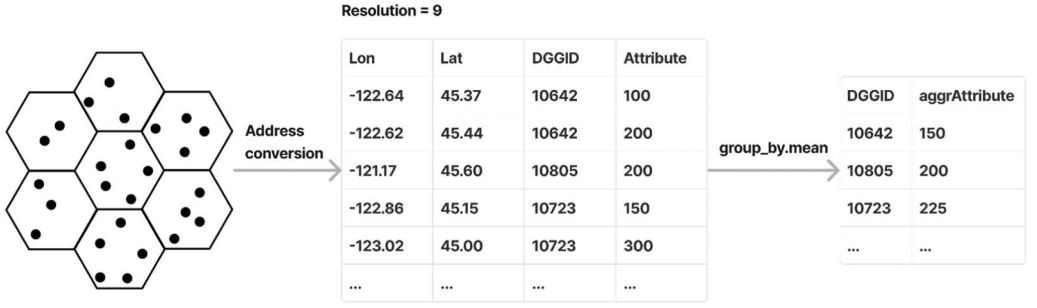

Figure 4. Illustration of point binning through averaging values on the ISEA3H DGGS at resolution 9.

Thresholding is an example of a post-point binning operation, enabling the filtration of specific hexagonal cells that meet particular criteria. As illustrated in Figure 4, to implement the process, the geographic coordinates of point features were initially converted to DGGS cell addresses at a certain DGGS resolution. This resolution is pre-determined based on both the spatial scale of the visualization scene and the original density of the point data, which can vary across different usage scenarios. Subsequently, the DGGS cell values were computed by grouping the data based on their cell addresses following a certain rule at the determined DGGS level. This approach replaces the point-in-polygon spatial query with an efficient group-by operation in the database, improving the processing speed and overall performance of the process.

5.2. Neighborhood operations on quantized datasets

Neighborhood operations aim to calculate a new value for a target cell by taking into account the values of its six directly connected neighbors. While the specific equation may vary depending on the use case, it can generally be conceptualized as a convolution kernel applied to the center cell in conjunction with its six neighbors. The overarching calculation of the center cell value can be defined as:

$$v_c = \sum_{i=1}^{7} w_i v_i \tag{4}$$

where v_i is the value of the original cell, w_i is the corresponding weight of the cell, and i ranges from 1 to 7 representing the center cell and its six neighbors. Figure 5 illustrates the application of Sobel kernels on hexagonal grids along three axes, employed for edge detection on a surface in three directions. The numbers displayed on the cells indicate the weight assigned to each specific cell (He et al. 2008). Vleugels and Palmblad (2020) showcased the application of the Gaussian smoothing filter on hexagonal grids in Cartesian coordinates. Figure 6 depicts the adjacent hexagonal cells in Cartesian coordinates in units of the hexagon side length. The weight of each hexagonal cell is computed based on the Gaussian function:

$$w_{(x,y)} = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{5}$$

where x and y are Cartesian coordinates of hexagonal cells in units of hexagon side length, and σ is the standard deviation of the Gaussian distribution. Neighborhood operations heavily depend on the neighbor table stored within the PostgreSQL database. This table, established through queries, determines the neighboring cells for each cell at a specific resolution. Subsequently, these cells are joined with the attribute table, facilitating the following calculations.

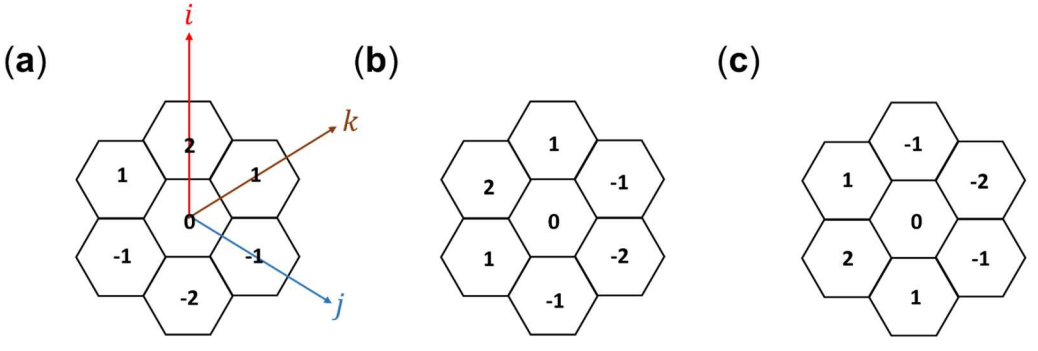

Figure 5. Sobel kernel on hexagonal grids along (a) *i* axis, (b) *j* axis, and (c) *k* axis.

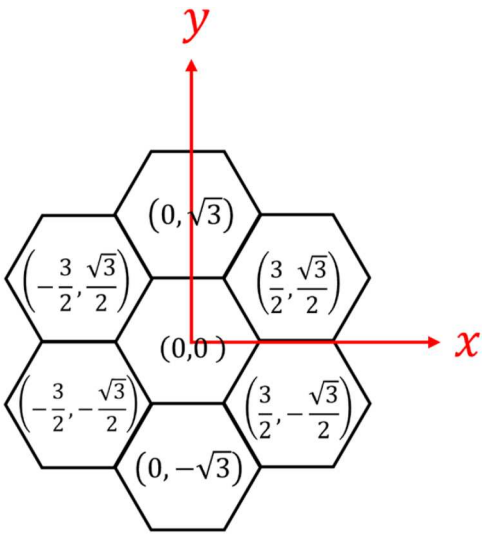

Figure 6. Hexagonal cells in Cartesian coordinates in units of hexagon side length.

5.3. Hierarchical operations on quantized datasets

In this study, we explored the aggregation operation within the context of DGGS across resolution levels, assuming that data has already been quantized in DGGS at a specific level. Aggregation refers to the combination of data at coarser resolutions. The ISEA3H schema guarantees that a parent cell has one central child cell and six non-central child cells at the next finer resolution. In this schema, the central child cell's centroid aligns with the parent cell's centroid, and the centroids of the non-central child cells align with the parent cell's vertices (Figure 7). Aggregation operations extensively rely on the hierarchical table stored within the PostgreSQL database. This table is instrumental in identifying the parent cells or child cells associated with a given cell through queries, and these cells are joined with the attribute table, allowing for subsequent calculations to be conducted. When aggregating continuous data, the parent cell combines values from its seven child cells, with the central child cell having three times the weight of the non-central child cells (Figure 7). The value of the parent cell is computed using the formula:

$$v_p = w_{cen} v_0 + w_{non-cen} \sum_{i=1}^{6} v_i \tag{6}$$

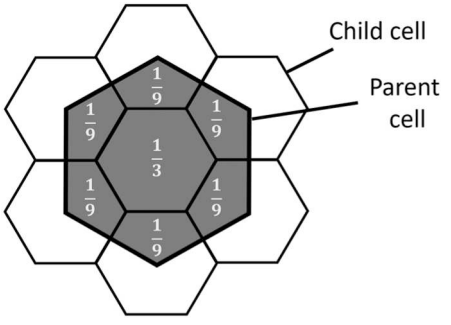

Figure 7. Illustration of down-sampling kernel of continuous data.

where $w_{cen} = \frac{1}{3}$ represents the weight of the central child cell, $w_{non-cen} = \frac{1}{9}$ represents the weight of the non-central child cells, v_0 is the value of the central child cell, and v_1 to v_6 are the values of the non-central child cells.

6. Case study

6.1. Visualization of aggregated building-loss data and associated thresholding

Building risk visualization is a crucial aspect of urban planning and disaster management, which involves assessing the vulnerability of buildings to various hazards such as earthquakes, floods, and fires, and representing such information clearly and intuitively. In this case study, we employed our system to visualize building loss, flood depth, and associated thresholding resulting from flood hazards around the southern Quebec area in Canada in an on-the-fly manner. The original building loss data consisted of polygons and was prepared by Geosapiens Inc.

In our context of flood hazard and loss visualization, buildings are assigned DGGS cell indices (i.e. DGGIDs) based on the geographical locations of their centroids. The associated building-specific hazard risk levels are pre-calculated and stored in the PostgreSQL database. It included the water depth in meters of 12 return period scenarios (return period of 2, 5, 10, 20, 25, 50, 100, 150, 200, 500, 1000, and 1500 years), average annual loss, and geometry information of each building. To dynamically visualize building risk on a web map, we used point binning and spatial statistics, as previously described in Section 5, to aggregate and summarize the flood depth and mean annual loss values of buildings within each DGGS cell at a specific level. In our case, we defined the DGGS resolution $r = t + 5$, where t represents the vector tile level in the current user's visualization scene. It should be noted that, in this case study, the chosen DGGS resolution was for illustration purposes and different criteria can lead to distinct outcomes in analysis and visualization. Our primary focus was to demonstrate our system's capability of point binning and dynamic visualizations, instead of exploring the MAUP in our system.

As explained in Section 3.4, vector tiles are dynamically generated via the serverless AWS Lambda function on-demand, based on the current user view area, incorporating the fundamental DGGS geometry required for visualization. During user interactions with the map interface, a backend query is dispatched. This query, which encapsulates the user's defined view area and zoom level, prompts the serverless AWS Lambda function to craft vector tiles for visible DGGS cells within the defined extent. Concurrently, the AWS Lambda function dispatches a query to the PostgreSQL database utilizing the DGGIDs of these visible cells. The database subsequently responds with the flood depth and building loss data corresponding to the requested cells. The AWS Lambda function then undertakes the task of incorporating the DGGS geometry from the freshly generated vector tiles with the flood depth and building loss information acquired from the PostgreSQL database. This integrated dataset is then transmitted to users, facilitating the dynamic visualization of flood depth and building loss. Our

Table 2. Summary of ISEA3H DGGS cell information and hexagon geometry generation timing for each vector tile across various levels.

Vector tile level	DGGS level	DGGS cell area (km^2)	Average number of DGGS cells	Average time of intersecting bounding box (ms)	Average time of creating hexagons (ms)	Total time of hexagon geometry generation (ms)
7	12	95.98	474	115	1623	1738
8	13	31.99	368	119	1234	1353
9	14	10.66	278	116	932	1048
10	15	3.55	211	119	735	854
11	16	1.18	159	117	613	730
12	17	0.39	120	120	512	632
13	18	0.13	89	118	435	553
14	19	0.04	68	117	370	487

system also facilitated the thresholding operations on the flood depth data. It involved querying buildings within each DGGS cell based on common DGGIDs, followed by an additional filter applied to the query results based on the values of flood depth. The remaining buildings after the filter were then passed to the visualization pipeline. Table 2 provides a summary of ISEA3H DGGS cell generation information across various vector tile levels. It includes the vector tile level, associated ISEA3H DGGS level, DGGS cell area, the average number of DGGS cells per tile, the average time to retrieve the intersected bounding box, the average time to create hexagon geometries, and the total time for hexagon geometry generation. Specifically, retrieving the intersected bounding box has a consistent response time regardless of resolution (Table 2). Requests for hexagon geometry creation are sent in parallel based on the number of vector tiles in the current user view. The average time for these parallel requests increases at finer levels due to the higher number of DGGS cells per tile (Table 2). It should be noted that the reported geometry generation time refers to the initial generation. Once generated and cached, retrieval from the S3 cache takes about 200 ms.

Figure 8 shows the application interface for building risk visualization. Figure 8(a) illustrates the mean average annual loss for a return period of 50 years within the current user view, with the

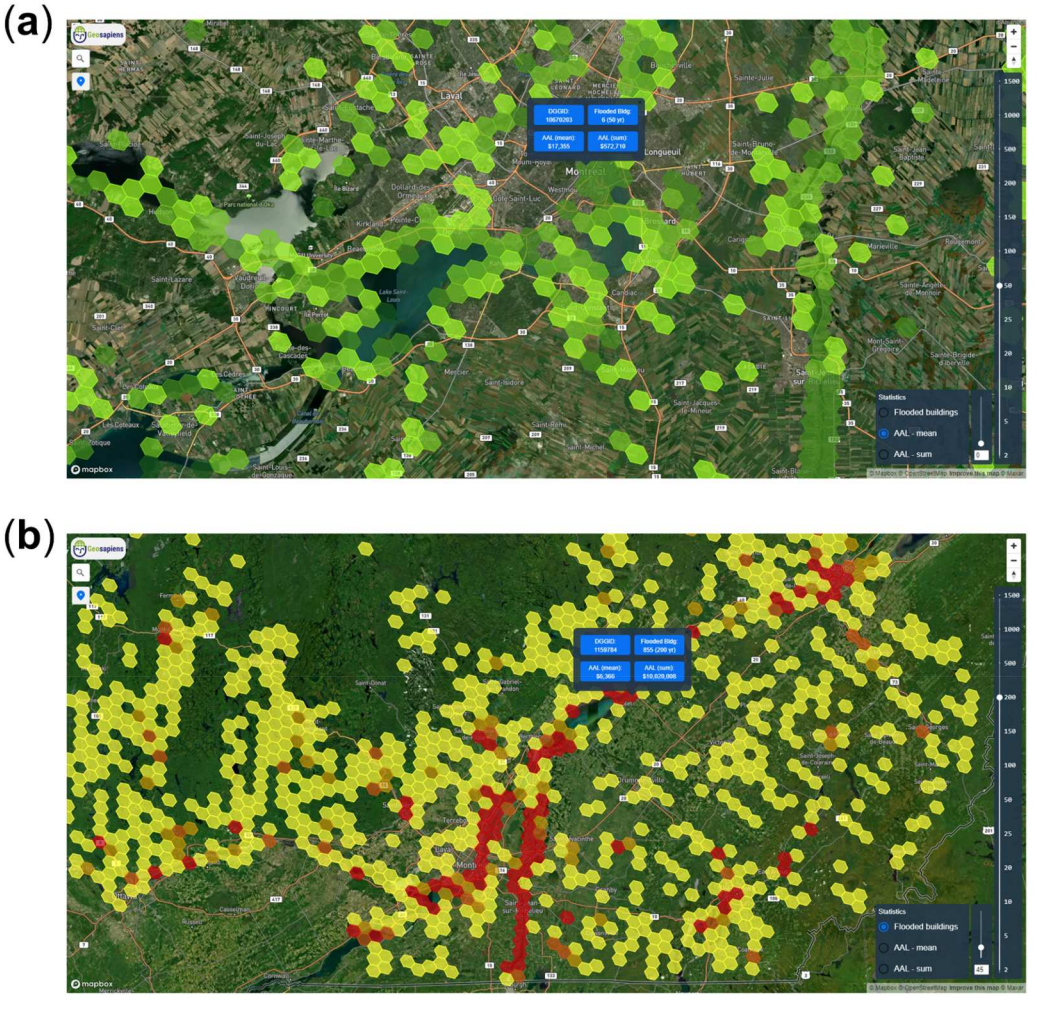

Figure 8. The application interface of building risk visualization: a. mean average annual loss for a return period of 50 years at ISEA3H DGGS level 15, and b. total count of buildings with flood depths greater than 45 cm for a return period of 200 years at ISEA3H DGGS level 13, within the current user view.

vector tile level of 10 and the DGGS level of 15. Figure 8(b) illustrates the total count of buildings with flood depths greater than 45 cm for a return period of 200 years within the current user view, where the vector tile is at level 8 and the DGGS is at level 13. Users can also query the number of flooded buildings, mean average annual loss, and total average annual loss for each hexagon bin via the interactive pop-ups (Figure 8).

6.2. Visualization of the MERIT DEM dataset at multiple resolutions

The Multi-Error-Removed Improved-Terrain (MERIT) Digital Elevation Model (DEM) was created by eliminating various error components, including absolute bias, stripe noise, speckle noise, and tree height bias, from previously available spaceborne DEMs, such as SRTM3 v2.1 and AW3D-30 m v1 (Yamazaki et al. 2017). It provides terrain elevations at a 3-arcsecond resolution, approximately 90 m at the equator, and covers land areas between 90° N and 60° S (Yamazaki et al. 2017). In this case study, we visualize the MERIT DEM, along with the results of down-sampling and neighborhood operations, within the context of a DGGS vector tile framework.

The original MERIT DEM raster data were quantized on the ISEA3H DGGS at level 12 following the largest-share principle detailed in Section 4.2. This data was organized within a structured attribute table with columns for DGGS cell indices (i.e. DGGIDs), resolution, and the corresponding DEM values. We first implemented the dynamic down-sampling operations using the MERIT DEM and realized the on-the-fly visualization of the outcomes. In this process, seven child cells were first determined by querying the hierarchical table, associated values were collected by joining the attribute table, and finally, the DEM value of each cell at the next coarser resolution was computed following Equation 6, where the central child cells triple the weight of the non-central child cells. After the computation of aggregated values for each DGGS cell, these values are subsequently integrated into the visualization process, as previously detailed. Figure 9(a) presents the down-sampled DEM at level 6 based on quantized data at DGGS level 12.

Furthermore, we integrated on-the-fly Sobel and Gaussian filters into our system to process MERIT DEM data and facilitate dynamic visualization. The procedure included identifying the neighboring cells of each central cell by querying the neighbor table at a certain resolution level, joining the attribute table based on the DGGIDs, applying the designated filter kernel, and integrating it into the visualization pipeline. The Sobel and Gaussian filter kernels applied in this case study were derived from the previous research, as explained in Section 5.2. We applied the Sobel filters along three hexagonal axes and the Gaussian filter with $\sigma = 1$ on the quantized MERIT DEM. The Gaussian kernel has been normalized so that the sum of the kernel weight equals 1. Figure 9(b–d) visualize the Sobel filtering results along the i, j, and k axis, respectively, on DGGS resolution 6. Figure 9(e) shows the Gaussian filtering results with $\sigma = 1$ on DGGS resolution 6.

7. Discussion and future directions

This study demonstrated the potential of DGGS in conjunction with serverless approaches for dynamic vector tile-based visualization. DGGS provided a uniform and scalable framework for geospatial data, enabling efficient data processing and visualization. The integration of serverless AWS Lambda further enhanced the system's scalability and adaptability, allowing for efficient parallel processing and on-the-fly operations. This can also be easily applied on other platforms such as Google Cloud and Microsoft Azure. We adopted PostgreSQL for data storage and querying, which has proven effective, particularly in handling complex data management tasks and supporting both structured and unstructured data. Integrating PostgreSQL with Amazon Aurora Serverless made our system suitable for large-scale applications. The dynamic vector tile-based visualization approach proposed in this study has shown promising results in terms of efficiency and adaptability.

Specifically, the use of AWS Lambda for on-the-fly generation of hexagon geometries significantly reduced the storage requirements. In this study, we intended to minimize storage by

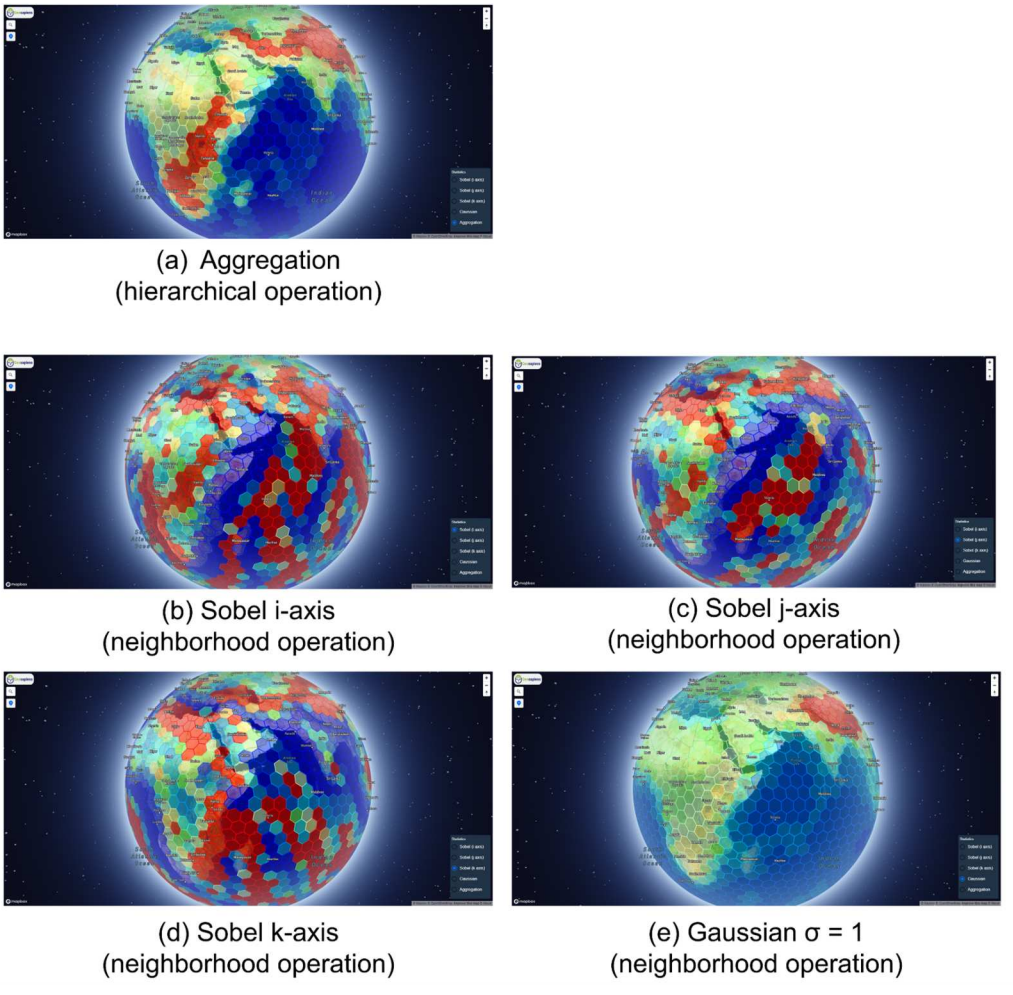

Figure 9. Visualizations for a. aggregation, b. Sobel filter along i axis, c. Sobel filter along j axis, d. Sobel filter along k axis, and e. Gaussian filter applied to the MERIT DEM on ISEA3H DGGS resolution 6.

adopting on-the-fly generation of geometries, even though this adds extra computational workload when processing queries. While pre-storing offers faster retrieval, it consumes significantly more space and exhibits limited scalability with sizable datasets. Furthermore, since the geometries are exclusively utilized for visualization rather than any DGGS operations, it is unnecessary to store an extensive volume of geometries permanently in the database. In addition to the storage efficiency, our system also ensures the query processing speed, overall system scalability, and simultaneously responsive visualization for all users. Initially, query processing times may be prolonged as information is computed at runtime through on-the-fly generation. Nevertheless, the scalability of AWS Lambda and the performance capabilities of Amazon Aurora Serverless can facilitate faster query processing, which can further benefit from cached results within AWS S3, reducing the need for repetitive computations. Moreover, the system can dynamically adjust its scale in response to varying workload demands, using combinations of services such as AWS Lambda, Amazon Aurora Serverless, and AWS S3. Lastly, our system can handle multiple users' view extents simultaneously and ensure accurate and responsive visualization for all users by tiling, multi-threading, and caching. By partitioning the data space into tiles, every user only asks for the tiles necessary for their view, thus minimizing data transfer and processing per user. AWS Lambda was used to enable

multi-threaded generation on the server side, and different users can have DGGS geometries generated in parallel, which speeds up overall response time. In addition, frequently accessed geometries are also cached in the AWS S3 on the server side, which prevents redundant generation when users request similar views.

Although scalable geospatial visualization was a central focus of our research, we also managed to implement quantization, neighborhood operations, and hierarchical operations, aligning with the fundamental operations emphasized by OGC (OGC 2019) and enabling dynamic visualization of the operation outcomes. Our use of PostgreSQL for storing static tables and the on-the-fly generation of geoJSON by the serverless AWS Lambda function reflects the compatibility of mainstream RDBMS and geometry formats in contemporary SDI, suggesting the potential interoperability of our system. The case studies presented in this paper demonstrated the practical applicability of the proposed system. The visualization of the building-loss data set showcased the system's capability to handle complex geospatial data and provide intuitive and informative visualizations. The visualization of the MERIT DEM dataset and associated aggregation and neighborhood operations further demonstrated the system's efficiency in supporting on-the-fly operations and adaptability in handling large-scale geospatial data.

While this study has shown promising results, several areas are worth further investigation. One potential area of research is the extension of the system to support other types of geospatial data, such as 3D data and time-series data (Robertson et al. 2020). This requires the development of new algorithms and techniques for data processing and visualization. Another potential area of research is the optimization of the system's performance. While the use of serverless AWS Lambda has significantly improved the system's scalability, there may be further opportunities for optimization, particularly in terms of data processing and querying. In our implementation, we pre-calculated and stored the attribute table containing quantized data values, the neighborhood table containing sibling navigation information, and the hierarchy table storing parent–child navigation information within PostgreSQL. The capacity of AWS Lambda is dynamically adjusted based on the workload, and AWS Lambda allows configuring the amount of memory allocated to each function, which also affects the CPU allocation and performance. Thus, these calculations can potentially be performed on the fly based on the current user view, as long as the computation remains within the serverless AWS Lambda's capacity. Furthermore, the incorporation of machine learning and artificial intelligence techniques can enhance the system's capabilities in data analysis and predictive modeling (Li, McGrath, and Stefanakis 2022c; Hojati and Robertson 2020). This involves the development of new models for geospatial data analysis as well as the incorporation of existing machine learning and artificial intelligence tools.

Acknowledgements

We acknowledge our colleagues at Geosapiens Inc. for their valuable feedback on our research. Special thanks to Jasmin Gill-Fortin for the assistance in preparing the analysis datasets, and to Haydn Laurence for his help with linguistic edits.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research was funded by the Geosapiens Inc.

Data availability statement

The building-loss data used in the first case study is not available due to commercial restrictions. The MERIT DEM data that support the findings of the second case study are openly available at https://hydro.iis.u-tokyo.ac.jp/~yamadai/MERIT_DEM.

References

Amazon. 2023. "Amazon Aurora: Unparalleled High Performance and Availability at Global Scale with full MySQL and PostgreSQL Compatibility." Accessed 1 September 2023. https://aws.amazon.com/rds/aurora/.

Baldini, I., P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, et al. 2017. "Serverless Computing: Current Trends and Open Problems." In *Research Advances in Cloud Computing*, edited by S. Chaudhary, G. Somani, and R. Buyya, 1–20. Singapore: Springer.

Bauer-Marschallinger, Bernhard, Daniel Sabel, and Wolfgang Wagner. 2014. "Optimisation of Global Grids for High-resolution Remote Sensing Data." *Computers & Geosciences* 72:84–93. https://doi.org/10.1016/j.cageo.2014.07.005.

Bernardin, Tony, Eric Cowgill, Oliver Kreylos, Christopher Bowles, Peter Gold, Bernd Hamann, and Louise H. Kellogg. 2010. "Crusta: A New Virtual Globe for Real-time Visualization of Sub-meter Digital Topography at Planetary Scales." *Computers & Geosciences* 37 (1): 75–85. https://doi.org/10.1016/j.cageo.2010.02.006.

Bondaruk, Ben, Steven A. Roberts, and Colin Robertson. 2020. "Assessing the State of the art in Discrete Global Grid Systems: OGC Criteria and Present Functionality." *Geomatica* 74 (1): 9–30. https://doi.org/10.1139/geomat-2019-0015.

Bousquin, Justin. 2021. "Discrete Global Grid Systems as Scalable Geospatial Frameworks for Characterizing Coastal Environments." *Environmental Modelling & Software* 146:105210. https://doi.org/10.1016/j.envsoft.2021.105210.

Bowater, David, and Emmanuel Stefanakis. 2019. "An Open-source Web Service for Creating Quadrilateral Grids Based on the rHEALPix Discrete Global Grid System." *International Journal of Digital Earth* 13 (9): 1055–1071. https://doi.org/10.1080/17538947.2019.1645893.

Chaudhuri, Chiranjib, Annie Gray, and Colin Robertson. 2021. "InundatEd-v1.0: A Height Above Nearest Drainage (HAND)-based Flood Risk Modeling System Using a Discrete Global Grid System." *Geoscientific Model Development* 14 (6): 3295–3315. https://doi.org/10.5194/gmd-14-3295-2021.

Conway, J. H., and N. J. A. Sloane. 1998. *Sphere Packings, Lattices and Groups*. New York, USA: Springer Science & Business Media.

Ding, Junjie, Xinhai Huang, Jin Ben, Jianbin Zhou, Qishuang Liang, Yihang Chen, and Jinchi Dai. 2024. "Encoding and Operation Scheme for the Rhombic Triacontahedron Aperture 4 Hexagonal Discrete Global Grid System." *International Journal of Digital Earth* 17 (1): 2316112. https://doi.org/10.1080/17538947.2024.2316112.

Djavaherpour, H., A. Mahdavi-Amiri, and F. F. Samavati. 2017. "Physical Visualization of Geospatial Datasets." *IEEE Computer Graphics and Applications* 37 (3): 61–69. https://doi.org/10.1109/MCG.2017.38.

Dutton, Geoffrey. 1984. "Part 4: Mathematical, Algorithmic and Data Structure Issues: Geodesic Modelling of Planetary Relief." *Cartographica: The International Journal for Geographic Information and Geovisualization* 21 (2-3): 188–207. https://doi.org/10.3138/R613-191U-7255-082N.

Dutton, Geoffrey. 1999. "Scale, Sinuosity, and Point Selection in Digital Line Generalization." *Cartography and Geographic Information Science* 26 (1): 33–54. https://doi.org/10.1559/152304099782424929.

Egenhofer, M. J., and J. R. Herring. 1990. *Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases*. Orono, ME: Department of Surveying Engineering, University of Maine.

Fryer, Daniel, Ming Li, and Andriy Olenko. 2020. "rcosmo: R Package for Analysis of Spherical, HEALPix and Cosmological Data." *The R Journal* 12 (1): 206–225. https://doi.org/10.32614/RJ-2020-012.

Gibb, Robert. 2016. "The rHEALPix Discrete Global Grid System." In *IOP Conference Series: Earth and Environmental Science, 9th Symposium of the International Society for Digital Earth (ISDE)*, Vol. 34, 012012. Halifax, Canada: IOP Publishing Ltd.

Golay, Marcel J.E. 1969. "Hexagonal Parallel Pattern Transformations." *IEEE Transactions on Computers* C-18 (8): 733–740. https://doi.org/10.1109/T-C.1969.222756.

Gorski, K. M., E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. 2005. "HEALPix — A Framework for High Resolution Discretization, and Fast Analysis of Data Distributed on the Sphere." *The Astrophysical Journal* 622 (2): 759–779. https://doi.org/10.1086/427976.

He, X., Q. Wu, W. Jia, and T. Hintz. 2008. "Edge Detection on Hexagonal Structure." *Journal of Algorithms & Computational Technology* 2 (1): 61–78. https://doi.org/10.1260/174830108784300321.

Hojati, Majid, and Colin Robertson. 2020. "Integrating Cellular Automata and Discrete Global Grid Systems: A Case Study into Wildfire Modelling." *Proceedings of the 23rd AGILE Conference on Geographic Information Science* 1 (6): 1–23. https://doi.org/10.5194/agile-giss-1-6-2020.

Hojati, Majid, Colin Robertson, Steven Roberts, and Chiranji Chaudhuri. 2022. "GIScience Research Challenges for Realizing Discrete Global Grid Systems as a Digital Earth." *Big Earth Data* 6 (3): 358–379. https://doi.org/10.1080/20964471.2021.2012912.

Jendryke, Michael, and Stephen C. McClure. 2019. "Mapping Crime – Hate Crimes and Hate Groups in the USA: A Spatial Analysis with Gridded Data." *Applied Geography* 111:102072. https://doi.org/10.1016/j.apgeog.2019.102072.

Jendryke, Michael, and Stephen C. McClure. 2021. "Spatial Prediction of Sparse Events Using a Discrete Global Grid System; A Case Study of Hate Crimes in the USA." *International Journal of Digital Earth* 14 (6): 789–805. https://doi.org/10.1080/17538947.2021.1886356.

JPL. 2019. "HEALPix: Data Analysis, Simulations and Visualization on the Sphere." Accessed 20 November 2019. https://healpix.sourceforge.io/index.php.

Kiester, A. Ross, and Kevin Sahr. 2008. "Planar and Spherical Hierarchical, Multi-resolution Cellular Automata." *Computers, Environment and Urban Systems* 32 (3): 204–213. https://doi.org/10.1016/j.compenvurbsys.2008.03.001.

Klems, M. 2018. *AWS Lambda Quick Start Guide: Learn How to Build and Deploy Serverless Applications on AWS*. Birmingham, UK: Packt Publishing Ltd.

Kmoch, Alexander, Ivan Vasilyev, Holger Virro, and Evelyn Uuemaa. 2022. "Area and Shape Distortions in Open-source Discrete Global Grid Systems." *Big Earth Data* 6 (3): 256–275. https://doi.org/10.1080/20964471.2022.2094926.

Konig, C., P. Weigelt, J. Schrader, A. Taylor, J. Kattge, and H. Kreft. 2019. "Biodiversity Data Integration-the Significance of Data Resolution and Domain." *PLoS Biology* 17 (3): e3000183. https://doi.org/10.1371/journal.pbio.3000183.

Lambers, M., and A. Kolb. 2012. "Ellipsoidal Cube Maps for Accurate Rendering of Planetary-scale Terrain Data." *Proceedings of the 20th Pacific Conference on Computer Graphics and Applications* PCG (2012): 5–10. https://doi.org/10.2312/PE/PG/PG2012short/005-010.

Li, Mingke, Heather McGrath, and Emmanuel Stefanakis. 2021. "Integration of Heterogeneous Terrain Data into Discrete Global Grid Systems." *Cartography and Geographic Information Science* 48 (6): 546–564. https://doi.org/10.1080/15230406.2021.1966648.

Li, Mingke, H. McGrath, and E. Stefanakis. 2022a. "Multi-resolution Topographic Analysis in Hexagonal Discrete Global Grid Systems." *International Journal of Applied Earth Observation and Geoinformation* 113:102985. https://doi.org/10.1016/j.jag.2022.102985.

Li, Mingke, Heather McGrath, and Emmanuel Stefanakis. 2022b. "Geovisualization of Hydrological Flow in Hexagonal Grid Systems." *Geographies* 2 (2): 227–244. https://doi.org/10.3390/geographies2020016.

Li, Mingke, Heather McGrath, and Emmanuel Stefanakis. 2022c. "Multi-scale Flood Mapping under Climate Change Scenarios in Hexagonal Discrete Global Grids." *ISPRS International Journal of Geo-Information* 11 (12): 627. https://doi.org/10.3390/ijgi11120627.

Li, Mingke, and E. Stefanakis. 2020. "Geospatial Operations of Discrete Global Grid Systems—a Comparison with Traditional GIS." *Journal of Geovisualization and Spatial Analysis* 4 (2): 26. https://doi.org/10.1007/s41651-020-00066-3.

Liang, Qishuang, Jianbin Zhou, Jin Ben, Yihang Chen, Xinhai Huang, Junjie Ding, and Jinchi Dai. 2024. "Precise Hexagonal Pixel Modeling and an Easy-sharing Storage Scheme for Remote Sensing Images Based on Discrete Global Grid System." *International Journal of Digital Earth* 17 (1): 2328824. https://doi.org/10.1080/17538947.2024.2328824.

Luczak, Ed, and Azriel Rosenfeld. 1976. "Distance on a Hexagonal Grid." *IEEE Transactions on Computers* C-25 (5): 532–533. https://doi.org/10.1109/TC.1976.1674642.

Ma, Yue, Guoqing Li, Xiaochuang Yao, Qianqian Cao, Long Zhao, Shuang Wang, and Lianchong Zhang. 2021. "A Precision Evaluation Index System for Remote Sensing Data Sampling Based on Hexagonal Discrete Grids." *ISPRS International Journal of Geo-Information* 10 (3): 194. https://doi.org/10.3390/ijgi10030194.

Mahdavi-Amiri, Ali, Troy Alderson, and Faramarz Samavati. 2015. "A Survey of Digital Earth." *Computers and Graphics* 53:95–117. https://doi.org/10.1016/j.cag.2015.08.005.

Mahdavi-Amiri, Ali, Erika Harrison, and Faramarz Samavati. 2016. "Hierarchical Grid Conversion." *Computer-Aided Design* 79:12–26. https://doi.org/10.1016/j.cad.2016.04.005.

Mapbox. 2023. "Mapbox: Maps and Location for Developers." Accessed 1 September 2023. https://www.mapbox.com/.

Mechenich, M. F., and I. Zliobaite. 2023. "Eco-ISEA3H, a Machine Learning Ready Spatial Database for Ecometric and Species Distribution Modeling." *Scientific Data* 10:77. https://doi.org/10.1038/s41597-023-01966-x.

OGC. 2019. "Topic 21: Discrete Global Grid System Abstract Specification." Accessed 15 November 2019. http://www.opengis.net/doc/AS/dggs/1.0.

OpenEAGGR. 2019. "Open Equal Area Global GRid." Riskaware Ltd., Accessed 26 November 2019. https://github.com/riskaware-ltd/open-eaggr.

Peterson, Perry, and Idan Shatz. 2019. "An Application Development Framework for Open Geospatial Consortium Discrete Global Grid System Standard." In *IGARSS 2019 IEEE International Geoscience and Remote Sensing Symposium 2019*, 5668–5670. https://doi.org/10.1109/IGARSS.2019.8898394.

PostgreSQL. 2023. "PostgreSQL: The World's Most Advanced Open Source Relational Database." Accessed 30 April 2023. https://www.postgresql.org/.

Qiu, Yue, Xuesheng Zhao, Deqin Fan, Songnian Li, and Yijing Zhao. 2022. "Disaggregating Population Data for Assessing Progress of SDGs: Methods and Applications." *International Journal of Digital Earth* 15 (1): 2–29. https://doi.org/10.1080/17538947.2021.2013553.

Raposo, Paulo, Anthony Robinson, and Randall Brown. 2019. "A Virtual Globe Using a Discrete Global Grid System to Illustrate the Modifiable Areal Unit Problem." *Cartographica: The International Journal for Geographic Information and Geovisualization* 54 (1): 51–62. https://doi.org/10.3138/cart.54.1.2018-0015.

Rawson, Andrew, Zoheir Sabeur, and Mario Brito. 2021. "Intelligent Geospatial Maritime Risk Analytics Using the Discrete Global Grid System." *Big Earth Data* 6 (3): 294–322. https://doi.org/10.1080/20964471.2021.1965370.

Robertson, Colin, Chiranjib Chaudhuri, Majid Hojati, and Steven A. Roberts. 2020. "An Integrated Environmental Analytics System (IDEAS) Based on a DGGS." *ISPRS Journal of Photogrammetry and Remote Sensing* 162:214–228. https://doi.org/10.1016/j.isprsjprs.2020.02.009.

Rodrigues, Ana S. L., Claudia L. Gray, Ben J. Crowter, Robert M. Ewers, Simon N. Stuart, Tony Whitten, and Andrea Manica. 2010. "A Global Assessment of Amphibian Taxonomic Effort and Expertise." *BioScience* 60 (10): 798–806. https://doi.org/10.1525/bio.2010.60.10.6.

Sahr, Kevin. 2011. "Hexagonal Discrete Global Grid System for Geospatial Computing." *Archives of Photogrammetry, Cartography and Remote Sensing* 22:363–376.

Sahr, K. 2022. "DGGRID Version 7.5." Accessed 12 March 2022. https://github.com/sahrk/DGGRID.

Sahr, Kevin, Denis White, and A. Jon Kimerling. 2003. "Geodesic Discrete Global Grid Systems." *Cartography and Geographic Information Science* 30 (2): 121–134. https://doi.org/10.1559/152304003100011090.

Shafiei, H., A. Khonsari, and P. Mousavi. 2022. "Serverless Computing: A Survey of Opportunities, Challenges, and Applications." *ACM Computing Surveys* 54 (11s): 1–32. https://doi.org/10.1145/3510611.

Somveille, Marius, Andrea Manica, and Ana S. L. Rodrigues. 2018. "Where the Wild Birds Go: Explaining the Differences in Migratory Destinations across Terrestrial Bird Species." *Ecography* 42 (2): 225–236. https://doi.org/10.1111/ecog.03531.

Stough, T., N. Cressie, E. L. Kang, A. M. Michalak, and K. Sahr. 2020. "Spatial Analysis and Visualization of Global Data on Multi-resolution Hexagonal Grids." *Japanese Journal of Statistics and Data Science* 3 (1): 107–128. https://doi.org/10.1007/s42081-020-00077-w.

Uber. 2019. "H3: A Hexagonal Hierarchical Geospatial Indexing System." Accessed 25 November 2019. https://github.com/uber/h3.

Veach, E., J. Rosenstock, E. Engle, R. Snedegar, J. Basch, and T. Manshreck. 2017. "S2 Geometry Library: Computational Geometry and Spatial Indexing on the Sphere." Accessed 5 June 2017. https://s2geometry.io.

Vleugels, R., and M. Palmblad. 2020. "Non-uniform Gaussian blur of hexagonal bins in cartesian coordinates." *arXiv preprint* arXiv:2005.09941. doi:10.48550/arXiv.2005.09941.

White, Denis, A. Jon Kimerling, Kevin Sahr, and Lian Song. 1998. "Comparing Area and Shape Distortion on Polyhedral-based Recursive Partitions of the Sphere." *International Journal of Geographical Information Science* 12 (8): 805–827. https://doi.org/10.1080/136588198241518.

Yamazaki, Dai, Daiki Ikeshima, Ryunosuke Tawatari, Tomohiro Yamaguchi, Fiachra O'Loughlin, Jeffery C. Neal, Christopher C. Sampson, Shinjiro Kanae, and Paul D. Bates. 2017. "A High-accuracy Map of Global Terrain Elevations." *Geophysical Research Letters* 44 (11): 5844–5853. https://doi.org/10.1002/2017GL072874.