Spatial Knowledge and Information Canada, 2019, 7(6), 1

Discrete Global Grid Systems: Operational Capability of the Current State of the Art

BEN BONDARUK
Department of Geography & Environmental Management
University of Waterloo,
Department of Geography & Environmental Studies
Wilfrid Laurier University
vbondaruk@uwaterloo.ca

STEVEN A. ROBERTS
Department of Geography & Environmental Studies
Wilfrid Laurier University
sroberts@wlu.ca

COLIN ROBERTSON
Department of Geography & Environmental Studies
Wilfrid Laurier University
crobertson@wlu.ca

ABSTRACT

The paper compares two current implementations of Discrete Global Grid Systems as potential new data models for spatial data representation, integration, and analysis. It outlines suitability of such structures for spatial data modelling and GIS applications, as well as documents the core criteria necessary for their successful implementation. An experimental analysis is performed in order to determine the current state of their development and their practical applicability for data integration, analysis and visualization. The work concludes with a reflection on the current implementations compared to the industry standards and some future projections of geospatial analysis within Discrete Global Grid Systems framework.

1. Introduction

Spatial data handling and integration has become one of the prevailing needs in geospatial analysis and computation. In the modern digital context spatial data are usually collected and stored as either raster or vector representations, along with attribute data for these spatial features (Mahdavi-Amiri, Alderson & Samavati, 2016). These representations have evolved to serve a number of specialist communities and workflows in GIS analysis (e.g., satellite-based remote sensing); however with continuing increases in spatial data heterogeneity and volume, the necessity for efficient data integration has become essential. As a result, new methods for integrating, transmitting and representing spatial data are required. Discrete Global Grid Systems (DGGS) have been proposed as a new model for spatial data representation, integration and analysis suited to the current data-rich environment (Li, 2013; Mahdavi-Amiri et el., 2016).

DGGS are hierarchical tessellations of regular shaped polygons (Figure 1.0) initially designed as a global reference system for mapping and navigational purposes (Purss, Gibb, Samavati, Peterson & Ben, 2016).

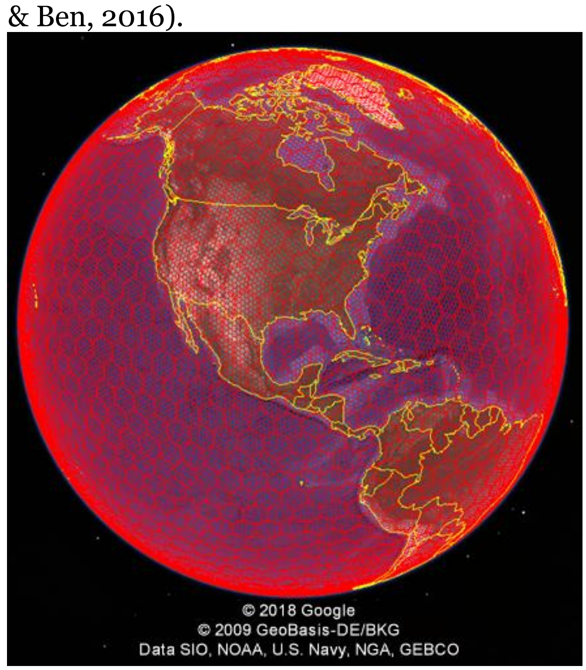


Figure 1.0. Example of a hierarchical structure of the Earth surface using hexagon shapes for the cell refinement generated for the purposes of this study.

Over time, due to its discrete construction DGGS have also started to be used as a data structure for consistent storage, reference and analysis of spatial data and its attribute information. DGGS suggest a different approach for geospatial data handling that allows interoperability of resources and elimination of inaccurate and complex data synthesis operations (Purss et el., 2016).

In order for a grid network to qualify as DGGS it must consist of core elements documented and summarized in the Open Geospatial Consortium (OGC) standard data protocol (Open Geospatial Consortium, 2017). The work of Goodchild and Kimerling (2002), on defining DGGS and some of their core requirements, have contributed greatly to the overall advancement of this new data standard; yet some of the earlier research on DGGS had already begun in the 1980s (Dutton, 1984). Their initial ideas and thoughts served as the basis for a fully functional and well-designed DGGS data standards set by OGC. OGC requirements were put in place in order to guarantee the explicit resolution, area preservation and positional uniqueness at each level of hierarchy which account for scale differences and spatial distortion globally.

In addition, the unique topological properties of regular shape tessellation might also naturally suggest looking for new forms of spatial analysis. The referencing and indexing mechanisms, on the other hand, provide reliable methods to access, store and retrieve data. As a result, various algorithms might integrate the existing indexing system for data assimilation via the refinement methods using the above properties (Peterson, 2017; Purss el al., 2016).

The aim of this paper is an in-depth analysis of two DGGS implementation: the H3 (Uber, 2015) and OpenEAGGR (Riskaware, 2017) open source software libraries. In addition, their operational and practical applications are also reviewed.

2. Methods

The following section outlines the methodology for comparing core data model requirements of the selected software libraries to the released OGC standards. It is very likely that certain requirements might not be met by either of the software, suggesting further advancing of DGGS packages. Both H3 and OpenEAGGR are open source software available via the GitHub source code repository (Uber, 2015; Riskaware, 2017).

For the testing purposes both libraries were built directly from their source in their natural development environments. This step is necessary in order to gain access to the full list of available functionality, which might not be available through other language bindings. In particular, the H3 library is built from its source using CMake packaging software, Visual Studio development environment with integrated C++ compiler; whereas OpenEAGGR is built using the MinGW compiler and Eclipse development environment. In addition, JavaScript binding for H3 and Python binding for OpenEAGGR libraries were also used in order to evaluate their flexibility and ease of use. Although not a requirement the libraries were also evaluated based on their availability for programing language bindings for the user’s convenience.

In this study the main focus is put on hexagon shape structures due to their availability on both platforms, as well as their advantages in sampling, circularity, packing and uniform connectivity properties over the other regular shapes (e.g., triangles, squares) (Li, 2013; Peterson, 2017). The particular interest of this study is to evaluate the operational capability of both libraries. In other words, the aim is to test practical applications of available DGGS software for their basic functionality, such as importing of spatial data, querying spatial analysis algorithms as well as exporting and visualizing the end results.

3. Results

A summary of H3 and OpenEAGGR implementations compared to OGC standards is outlined in Table 1.0. The existing DGGS software provide a comprehensive approach for geocoding, indexing, addressing and processing geospatial data into discrete forms. Due to their hierarchical structures, positional uniqueness and discrete representation of spatial resolution, DGGS are gaining popularity and acceptance in the modern geospatial analysis and data integration.

3.1 Technical specifications

A detailed technical analysis show basic functionality of DGGS for modeling Earth's surface via hierarchical networks of equal area cells. Both libraries support hierarchical tessellation of regular polygons at increasingly fine resolutions up to a m^2 and cm^2 in areal size for H3 and OpenEAGGR respectively (Uber, 2015; Riskaware, 2017). Each cell has a unique index and is accessible throughout the hierarchies. The given software also provides functionality to convert from latitude-longitude coordinates to DGGS indexes and vice versa referencing all cell centroids.

Since addressing and referencing are two major properties of DGGS (criteria 11-12) (Table 1.0) it is important to mention that H3 and OpenEAGGR use hierarchy-based and offset coordinate addressing structure for hexagonal cell systems respectively. Hierarchical addressing is based on the lower resolution grid to find next consecutive cell's location of a higher resolution, whereas offset addressing uses fixed axis orientation and offset distances from their origin to determine location of a cell (Bush, 2017). Both implementations use an icosahedron as a base polyhedron for creating planar faces approximating a sphere. The grid partitioning method of H3 uses hexagon aperture 7, whereas OpenEAGGR incorporates both hexagon aperture 3 and triangle aperture 4 hierarchical models (Uber, 2015; Riskaware, 2017). Aperture is a method known to partition a DGGS cell using additional partial self-similar shapes (e.g., hexagons) in order to preserve equal area property across multiple resolutions (Figure 2.0).

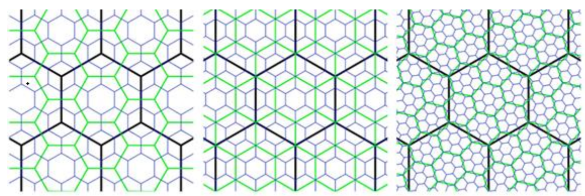


Figure 2.0. Hierarchical partition of space using hexagon apertures 3 (left), 4 (middle) and 7 (right) (Sahr, 2013).

The results of this subsection indicate that both implementations fail to meet the complete list of required criteria outlined by OGC and therefore cannot be classified as fully functional DGGS (Table 1.0).

Table 1.0: The following table outlines a core set of criteria to be met by software and classified as DGGS. The table also summarizes technical specifications of H3 and OpenEAGGR libraries and compares them to the OGC standards.

Criteria	*OGC Requirement*	*H3*	*OpenEAGGR*	*Notes*
1	Core Data Model	(✓×) Partial fulfillment	(✓×) Partial fulfillment	This requirement includes definition of conceptual data model of DGGS including reference frame (criteria 2-13) and functional algorithms (criteria 14-18) elements, which are partially fulfilled by each library.
2	Area	(✓) Fulfilled	(✓) Fulfilled	Guarantees the coverages of the entire globe. Each library fulfills the requirement for covering the entire surface of the earth.
3	Overlap	(✓) Fulfilled	(×) Not fulfilled	Ensures positional uniqueness without overlapping cells. Theoretically, this requirement is met by both libraries; however practical application of OpenEAGGR fails to meet the requirement (see Figure 3.0).
4	Tessellation sequence	(✓) Fulfilled	(✓) Fulfilled	Forms a sequence of hierarchical tessellations at multiple spatial resolutions. Both libraries are capable of generating hierarchical grids at various resolutions.
5	Area preservation	(✓) Fulfilled	(×) Not fulfilled	A total surface area must be preserved throughout hierarchical tessellations. The following requirement is not met by OpenEAGGR due to the overlapping cells in criterion 3 and perhaps inconsistent geometry of the offset coordinate system.

Table 1.0: Continued.

Criteria	*OGC Requirement*	*H3*	*OpenEAGGR*	*Notes*
6	Shape	(✓) Fulfilled	(✓) Fulfilled	DGGS cells must be formed of simple regular polygons. Both libraries have met the requirement with H3 using mostly hexagons and OpenEAGGR mostly hexagons and triangles (see criterion 8).
7	Equal area precision	(×) Not fulfilled	(✓×) Partial fulfillment	Any DGGS implementation will have equal area uncertainties of cells caused by the factors such as converging calculation, the rate of convergence or the precision of real numbers (e.g., π) used to calculate DGGS cell geometry. H3 seems to omit such technical details for the computational precision of equal area cells, whereas OpenEAGGR summarizes some technical benchmarks in its prototype evaluation framework (Bush, 2017).
8	Equal area	(✓) Fulfilled	(✓) Fulfilled	For each successive resolution equal area cells must be defined within the specified level of precision. Both libraries are constructed on the icosahedron with H3 using equal area hexagons and OpenEAGGR – hexagons and triangles. The only exception is that both hexagon grid libraries contain 12 pentagon cells centered at each icosahedron vertices and resolution. Pentagon cells are necessary in order to tile the sphere completely.
9	Initial tessellation	(✓) Fulfilled	(✓) Fulfilled	The initial partition of a sphere must be specified as a base unit polyhedron. Both libraries meet the requirement and use an icosahedron as a base.
10	Refinement (aperture)	(✓) Fulfilled	(✓) Fulfilled	Cell refinement methods and maximum number of refinements must be specified for each DGGS. H3 uses hexagonal aperture 7 grid partitioning method, whereas OpenEAGGR triangular aperture 4 and hexagonal aperture 3 cell partitioning.
11	Addressing	(✓) Fulfilled	(✓) Fulfilled	A spatial referencing method for an assignment of a unique identifier (index) must be specified. H3 implements hierarchy-based indexing method, whereas OpenEAGGR uses hierarchical indexing for triangular and offset coordinate indexing for hexagonal cell systems.
12	Spatial reference	(✓) Fulfilled	(✓) Fulfilled	A unique identifier must be assigned to each DGGS cell. Both libraries meet this requirement by assigning unique index to each DGGS cell using both hierarchical-based and offset coordinate indexing methods.
13	Cell centroid	(✓) Fulfilled	(✓) Fulfilled	The location of each DGGS cell must be referenced by the location of their centroids. Both libraries meet this property. It was tested by converting random latitude-longitude coordinates to a DGGS cell and vice versa. The new output coordinates were assigned to the cell centroids.
14	Quantization	(×) Not fulfilled	(×) Not fulfilled	Quantization methods for assigning and retrieval of data to individual DGGS cells must be documented; however such functionalities are not supported at this stage of the development.
15	Cell navigation	(✓) Fulfilled	(✓×) Partial fulfillment	Methods for hierarchical and neighbourhood navigation must be provided. The H3 library is equipped with functions for navigating between different resolutions and neighbouring cells. The OpenEAGGR library, however, does not support the neighbourhood query, but only the navigation queries through hierarchy.
16	Spatial analysis	(×) Not fulfilled	(✓) Fulfilled	Methods for performing simple spatial analysis operations on the grids must be provided. At this stage of the development only OpenEAGGR library is equipped with spatial analysis functions, such as equals, contains, intersects, etc. for two DGGS shape objects.
17	Query	(×) Not fulfilled	(✓×) Partial fulfillment	Methods for receiving, interpreting and processing data queries by DGGS algorithms must be provided. The OpenEAGGR library supports integration with third party software, however those extensions are challenging to use due to the outdated technical support for the newer software releases.
18	Broadcast	(×) Not fulfilled	(✓×) Partial fulfillment	Methods for integration, processing and transmitting data to external applications or web-based clients must be provided. The OpenEAGGR library also provides theoretical broadcasting functionality to external applications, however due to the limited technical support this property was not deployed in this study.

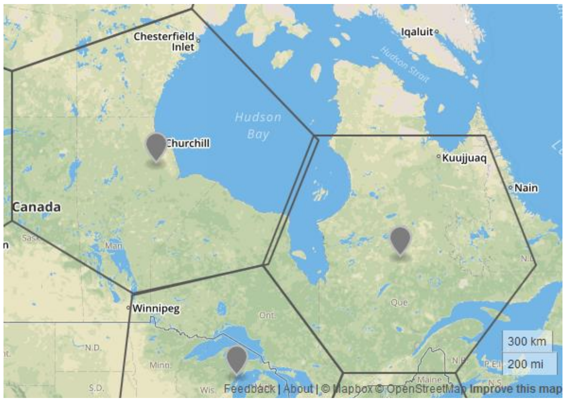


Figure 3.0. Example of the failed requirements for positional uniqueness and area preservation due to the overlapping cells generated via OpenEAGGR software library.

3.2 Operational proficiency

Both H3 and OpenEAGGR libraries deliver a reasonable amount of functionality (Table 2.0) in order to meet basic DGGS requirements for operational capability of conversion, query and search across DGGS hierarchy.

Table 2.0: Outlines the list of available language bindings, software extensions and API functions for H3 and OpenEAGGR libraries.

Evaluation	*H3*	*OpenEAGGR*
Language bindings	Erlang Go Java JavaScript OCaml PHP Python R	C C++ Java Python
Software extensions		PostgreSQL/PostGIS Elasticsearch
Basic API functions	geoToH3 h3ToGeo h3ToGeoBoundary h3GetResolution h3GetBaseCell stringToH3 h3ToString h3IsValid h3IsPentagon kRing kRingDistances h3ToParent h3ToChildren compact uncompact polyfill hexAreaKm2 hexAreaM2 numHexagons	convertPointToDggsCell convertShapesToDggsShapes convertShapeStringToDggsSh apes convertDggsCellToPoint convertDggsCellsToPoints convertDggsCellsToShapeStri ng getCellParents getCellChildren getCellSiblings getBoundingCell createKmlFile convertDggsCellOutlineToSha peString compareShapes

However, their applications are limited to their development environments and must be executed via direct function calls from within. In other words, the libraries do not support a user friendly interface.

In addition, their practical applications for importing user data, performing spatial analysis as well as exporting and visualizing the end result still requires further development. The integration with other third party software should be more effortless, with up-to-date technical support.

On the bright side, both libraries provide seamless functionality of coordinate conversion to a DGGS cell indexes for individual point locations at different resolutions, which is also illustrated in practice (Figures 4.0, 5.0).

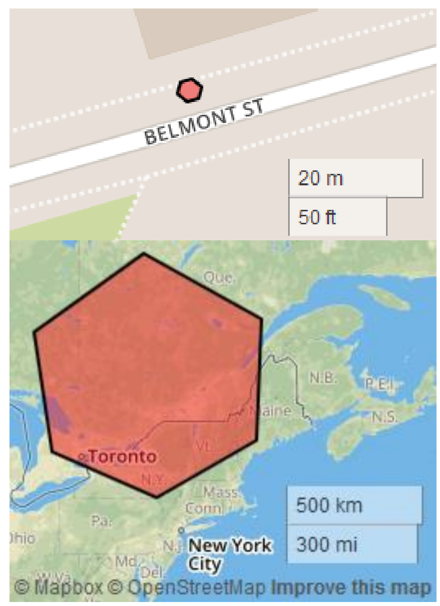


Figure 4.0. Conversion of Toronto's latitude longitude coordinates into DGGS cells via H3 library. The images are at DGGS resolution 1 (top) and 14 (bottom), which are equivalent to the 6.3 m^2 and 607,221 km^2 average surface area.

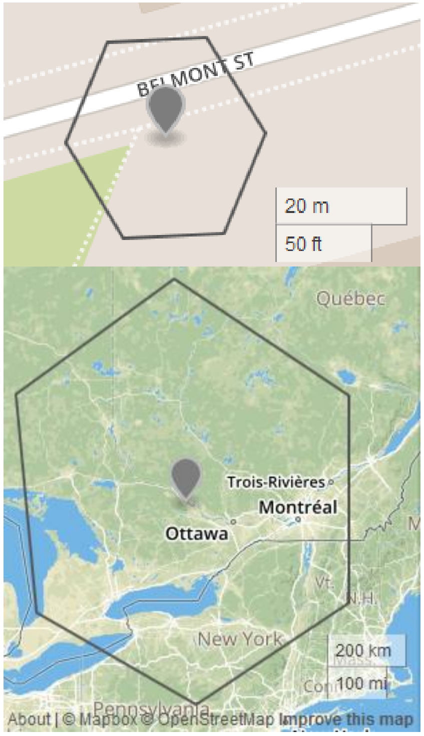


Figure 5.0. Conversion of Toronto's latitude longitude coordinates into DGGS cells via OpenEAGGR library. The resolutions accuracy were approximated to 1000 m^2 (top) and 1,000,000,000 km^2 (bottom) of the surface area.

Cell navigation functionality is also well implemented by the H3 library, which allows performing basic distance and search queries in order to identify and generate neighboring hexagons within a desired distance from a cell of interest (Figure 6.0). The OpenEAGGR library, however, lacks such functionality and only supports basic parent-child cell relationship (Figure 7.0). Furthermore, the developers indicate that parent-child queries perform significantly worse for hexagonal grids due to the implemented offset coordinate indexing system (Bush, 2017). With offset coordinates the parent-child identification is based purely on the grid geometry, which might lead to the sources of error for positional uniqueness and area preservation. As a result, it is suggested to use caution when integrating offset coordinate indexing system for DGGS implementation.

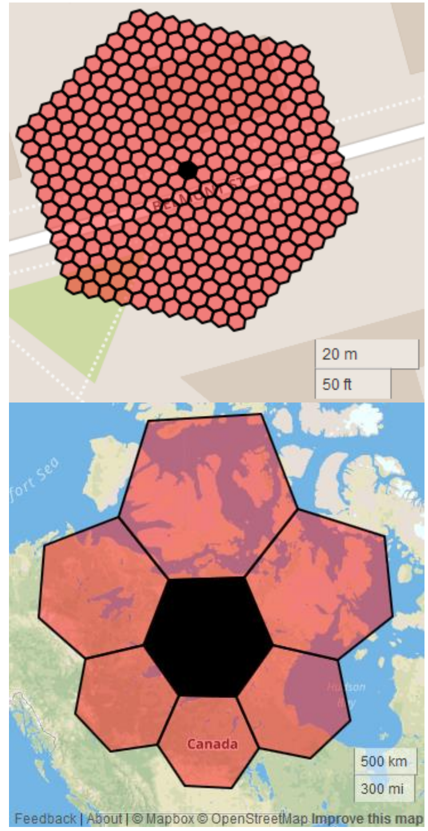


Figure 6.0. Performing a search query of neighboring cells via H3 kRing function with ring distance of 10 (top) and 1 (bottom) from a cell of interest (black).

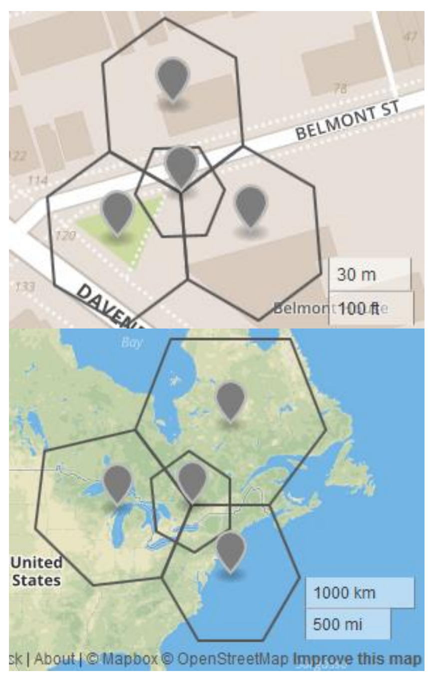


Figure 7.0. Performing a parent search query via OpenEAGGR getCellParents function for high (top) and low (bottom) resolution cells.

The H3 library also supports more advanced functionalities, such as filling polygon areas with hexagons as well as compressing them into more efficient representation (Figure 8.0).

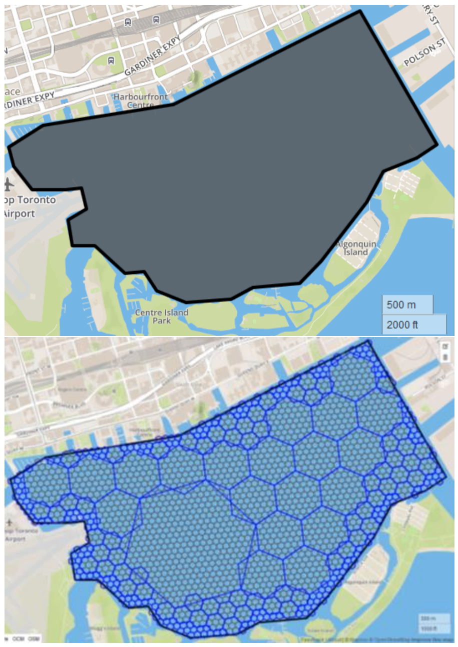
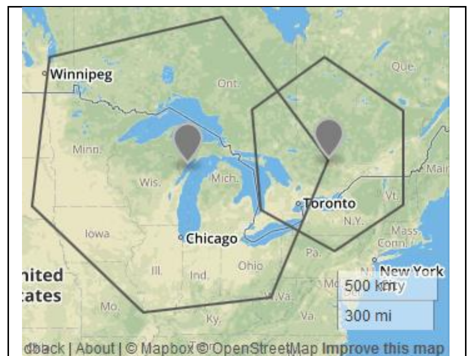


Figure 8.0. The above figure demonstrates H3 functionality for tessellating area of interest (top) with hexagons (bottom, grey), as well as the ability to compact them into a more concise shape (bottom, blue)

Although they are useful, these APIs might not be classified as spatial analysis functions for performing operations and determining relationship between DGGS cells. The OpenEAGGR, on the other hand, does support spatial analysis APIs for shape comparison of DGGS cells, linestrings and polygons with variety of available operations (Figure 9.0).

Visualization is not inherently available in the tested libraries. In other words, in order to visualize the output results the object or shape must be exported into one of the available file formats, such as GeoJSON or KML via OpenEAGGR APIs; however, this functionality might not be applicable to all DGGS shapes. If export is not possible the spatial data objects will remain stored in memory and could be accessed via direct memory calls as a workaround.

In comparison, the H3 library does not support built-in functionality for exporting shape geometries directly. As a result, a script for converting such data objects into GeoJSON file format was implemented separately in order to visualize the output via third party applications such as Google Earth or geojson.io.



EQUALS	False
CONTAINS	False
WITHIN	False
TOUCHES	False
DISJOINT	False
INTERSECTS	True
COVERS	False
COVERED_BY	False
CROSSES	False
OVERLAPS	True

Figure 9.0. The above figure demonstrates output of spatial analysis queries performed on two DGGS cell shapes via Open EAGGR library.

4. Conclusion

Traditional spatial analysis includes multiple techniques to study geographic phenomena by interacting with existing data related to a specific geographic location. Such data are then used to extract meaningful information with the help of computer processing and applications. Several problems might occur during such a chain of events, but integration of multiple data sources at once is an important aspect of the problem. DGGS set a benchmark for a more scalable and comprehensive data handling that can be distributed across different platforms and accessed via the web, and therefore have been investigated in detail in this paper.

Both H3 and OpenEAGGR software deliver basic functionality of DGGS, however cannot be classified as such due to the unfulfilled OGC requirements (Table 1.0). It was found that H3 library is missing some of the key functionality for assigning and retrieval of spatial data, data quantization as well as basic spatial analysis, query and broadcasting functionalities. The OpenEAGGR library is more successful with spatial analysis, data query and broadcasting implementations, yet still lacks support for data quantization and some essential properties of positional uniqueness and area preservation.

Both implementations provide great variety of language bindings available for integration with third party applications (Table 2.0), however not all of them are at the same level of development. In terms of the current progress, it also seems that the H3 project undergoes more rapid development compared to the OpenEAGGR and has greater functional availability. New H3 features and corrections are being implemented regularly, and connections with other third party software continuously explored for data query and broadcasting. This includes Uber's own operational needs for dynamic optimization of ride prices, as well as spatial decision making on a city level (Uber, 2018). As of now, however, the current open source implementations are not at the point where they can be used at the larger scales with convenience and minimal technical experience.

Therefore, all these components should receive additional attention in order to make them more practical and possible for average users to tailor for their specific needs. Once accomplished, however, it is very likely that DGGS will set new standards for geospatial analysis and open up new research prospects.

Acknowledgements

This work was funded by the Global Water Futures Programme through the Global Water Citizenship Project.

References

Bush, I. (2017). OpenEAGGR literature review & prototype evaluation. [online] Retrieved August 2018, from https://github.com/riskaware-ltd/open-eaggr/blob/master/Documents/Literature%20Review%20%26%20Prototype%20Evaluation.pdf

Dutton, G. (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. doi:10.3138/r613-191u-7255-082n

Goodchild, M. F, & Kimerling, A. (2002). *Discrete Global Grids: A Web Book*. Retrieved August 2018, from https://escholarship.org/uc/item/9492q6sm

Li, X. (2013). Storage and addressing scheme for practical hexagonal image processing. *Journal of Electronic Imaging, 22*(1), 010502. doi:10.1117/1.jei.22.1.010502

Mahdavi-Amiri, A., Alderson, T., & Samavati, F. (2016). Data Management Possibilities for Aperture 3 Hexagonal Discrete Global Grid Systems. Retrieved August 2018, from http://dx.doi.org/10.5072/PRISM/30988

Open Geospatial Consortium. (2017). Discrete Global Grid Systems Abstract Specification. [online] Retrieved August 2018, from http://docs.opengeospatial.org/as/15-104r5/15-104r5.html.

Peterson, P. (2017). Discrete Global Grid Systems. *International Encyclopedia Of Geography: People, The Earth, Environment And Technology*, 1-10. doi: 10.1002/9781118786352.wbieg1050

Purss, M., Gibb, R., Samavati, F., Peterson, P., & Ben, J. (2016). The OGC® Discrete Global Grid System core standard: A framework for rapid geospatial integration. *2016 IEEE International Geoscience And Remote Sensing Symposium (IGARSS)*. doi: 10.1109/igarss.2016.7729935

Riskaware Ltd. (2017). OpenEAGGR (Open Equal Area Global GRid). Retrieved August 2018, from https://github.com/riskaware-ltd/open-eaggr/

Sahr, K. (2013). On the Optimal Representation of Vector Location using Fixed-Width Multi-Precision Quantizers. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-4/W2, 1-8. doi:10.5194/isprsarchives-xl-4-w2-1-2013

Uber Technologies Inc. (2015). H3: A hexagonal hierarchical geospatial indexing system. Retrieved August 2018, from https://uber.github.io/h3/#/

Uber Technologies Inc. (2018). H3: Uber's hexagonal hierarchical spatial index. Retrieved November 2018, from https://eng.uber.com/h3/