AGILE: GIScience Series, 3, 41, 2022. https://doi.org/10.5194/agile-giss-3-41-2022
Proceedings of the 25th AGILE Conference on Geographic Information Science, 2022.
Editors: E. Parseliunas, A. Mansourian, P. Partsinevelos, and J. Suziedelyte-Visockiene.
This contribution underwent peer review based on a full paper submission.

Applied open-source Discrete Global Grid Systems

Alexander Kmoch[1], Oleksandr Matsibora[1,2], Ivan Vasilyev[1], and Evelyn Uuemaa[1]

[1]Department of Geography, Institute of Ecology and Earth Sciences, University of Tartu, Vanemuise 46, Tartu, 51003, Estonia
[2]Institute of Geography, National Academy of Sciences of Ukraine, Kyiv, Ukraine

Correspondence: Alexander Kmoch (alexander.kmoch@ut.ee)

Abstract. Discrete Global Grid Systems (DGGS) are spatial reference systems that use a hierarchical tessellation of cells to partition and address the globe and provide alternative spatial data format and indexing methods as compared to traditional vector and raster spatial data. In order to effectively use DGGS, functional software needs to be available and data needs to be indexed into a DGGS. We compare the software APIs of the 5 main open-source DGGS implementations - Uber H3, Google S2, rHEALPix by Landcare Research New Zealand, RiskAware OpenEAGGR, and DGGRID by Southern Oregon University - and present exemplary workflows for converting spatial and vector and raster datasets into DGGS-indexed format. We summarize, that Uber H3 and Google S2 provide more mature software library functionalities and DGGRID provides excellent functionality to construct grids with desired geometric properties and to load point data but does not provide functions for traversal and navigation within a grid after its construction.

Keywords. Coordinate Reference Systems; Spatial Indexing; software API; DGGS

1 Introduction

Two main approaches have arisen for the creation of the Digital Earth: Discrete Global Grid Systems (DGGS), and data cubes (Purss et al., 2019; Alderson et al., 2020). In a DGGS, the surface of the Earth is tessellated into a set of highly regular cells. These cells can then be addressed by using an indexing mechanism that is used to assign and retrieve data. Data cubes are n-dimensional arrays that are used to store query-ready spatial-temporal data ordered according to various attribute/coordinate axes, which can be spatial or non-spatial in nature (Alderson et al., 2020). DGGS theory has been discussed for decades in the scientific community, but the interest among wider public has gained momentum in recent years including emergence of practical implementations, as demonstrated by the recent OGC abstract specification (Open Geospatial Consortium, 2017). Consequently, DGGS are increasingly considered as the globally applicable reference system and spatial data model for congruent hierarchical data cubes and spatial data in general (Goodchild, 2018; Purss et al., 2019). The discretization of the sphere (or ellipsoid) is derived from the faces of a platonic base solid (Fig.1), the most commonly implemented base solids are the cube and the icosahedron.

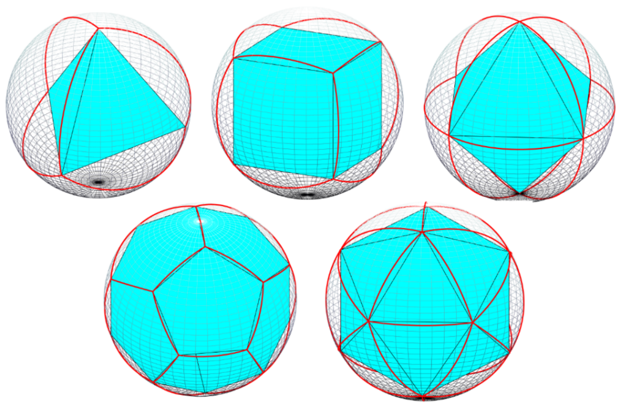
Figure 1. Different platonic base solids and their spherical representations (left to right, top to bottom): tetrahedron, cube, octahedron, dodecahedron, icosahedron (Source: Lei et al., 2020)

Li and Stefanakis (2020) have summarized advantages of DGGS against traditional GIS. However, cell divisions of DGGS implementations have different geometric properties due to the choice of the base solid, tessellation scheme, and geographic referencing or projection method for the cells. A DGGS software needs to implement a set of core steps, 1) select base solid as a partition surface, 2) choose an orientation of the base solid relative to the Earth, 3) define the hierarchical spatial partitioning method (tessellation) on the faces of the base solid, 4) project the planar partition face to the corresponding spherical or ellipsoidal surface, and 5) provide a method of indexing and addressing grid cells. The majority of tessellation types used in the literature and in available software are based on a) the cube as the base solid with rectangular congruent refinements, b) quaternary triangular meshes, c) or hexagonal systems. The aperture of a tessellation system describes the relationship of a parent cell to the children cells of the next higher resolution. For example, a square (or any quadrilateral) or a triangle can be subdivided in 4 equal children, thus an aperture of 4. For hexagons, there are several apertures known, 3,4 and 7. Only in aperture 7 hexagonal systems we can speak of an almost congruent parent-child relationship. Fig.2 shows widely used tessellations.

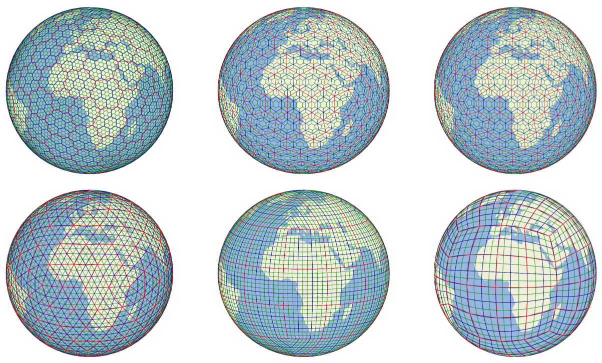
Figure 2. Most widely implemented tessellation schemes (left to right, top to bottom): aperture 7 hexagons (7H), 3H, 4H, aperture 4 triangles (4T), aperture 9 mixed (rHEALPix), aperture 4 squares. The last two instances have a cube as a base solid, the former ones use an icosahedron.

Classic GIS services such as gazetteers are rediscovered for geo-referencing geoscience articles (Adams, 2017). DGGS-based GIS analyses are already being conducted across a variety of domains, including crime analysis (Jendryke and McClure, 2019), wildfire modelling (Robertson et al., 2020), characterisation of coastal environments (Bousquin, 2021), risk analysis for marine traffic (Rawson et al., 2021), or flood mapping (Chaudhuri et al., 2021). Other Various large-scale GIS challenges are being reevaluated on DGGS, including watershed delineation, land use / land cover change statistics, and general earth system modelling (Liao et al., 2020). Such studies rely on available open-source DGGS implementations and authors state unanimously that DGGS are favourable geospatial frameworks for data integration and large scale environmental modelling and analysis.

In this short paper we want to provide an overview of available free and open-source DGGS software, their functionalities and how to use them in a conceptual workflows of indexing spatial data. We conclude the article with a discussion on current perceived shortcomings and a call to action to improve the tooling and interoperability around open-source DGGS software.

2 Available free and open-source DGGS software

2.1 Data and Software Availability

Although predominantly developed in the C/C++ programming languages, all free and open-source DGGS software have some support for the Python programming language. This facilitates a comparative analysis of their functionality.

- Uber H3 (Uber Technologies Inc, 2018): base C-libary, with many implementations, including a Python API, also installable from on PYPI and CONDA https://h3geo.org/;
- rHEALPix (Gibb et al., 2016): a pure Python library depending on Numpy and Scipy, available on PYPI https://github.com/manaakiwhenua/rhealpixdggs-py;
- DGGRID (Sahr, 2019): a C++ based command-line tool, need to be self-compiled, there are wrappers for DGGRID https://www.discreteglobalgrids.org/software/;
- Google S2 (Veach et al., 2017): a C++ library, needs to be compiled with SWIG to make use of Python bindings https://s2geometry.io/;
- OpenEaggr DGGS (Bush, 2017): an often compared C++ DGGS, with Python bindings, however, public development has not continued though https://github.com/riskaware-ltd/open-eaggr/.

We developed a Jupyter/Binderhub notebook-based repository for exploratory work with open-source Discrete Global Grid System implementations https://github.com/allixender/dggs_t1 (Kmoch, 2021).

In order to include DGGRID into Python compatible workflows we developed a Python wrapper tool named *dggrid4py*, also available on PYPI (Kmoch, 2020). This still requires DGGRID to be compiled and installed, but then integrates with the geospatial Python libraries ecosystem including Shapely, Fiona and Geopandas.

2.2 Functionality and API comparison

2.2.1 rHEALPix

The rearranged Hierarchical Equal Area isoLatitude Pixelization (rHEALPix) DGGS is based on the NASA HEALPix system (Gorski et al., 1998; Gibb et al., 2016). In contrast to other DGGS it is based on a ellipsoid and not a sphere. rHEALPix uses a cube as the base solid and a symmetric hierarchical partitioning of the cube faces with 4 different shape variations depending on the latitude - a quad cell (quadrilateral), a dart cell (triangular), a skew-quad cell (quadrilateral), and a cap cell (circular). All 4 shapes are congruent and form an aperture 9 refinement. The rHEALPix projection method is non-conformal, but achieves that the area of each cell is equal throughout each resolution. For indexing, Z (Morton) space-filling curves are used for every face of the cube. Table 1 lists central rHEALPix implementation details, which include conversion to and from geographic coordinates and grid traversing. The ellipsoid as well as rotation of the base solid can be freely defined (Bowater and Stefanakis, 2018).

Table 1. rHEALPix API highlights

Description	API details
DGGS object	rddgs = rhealpixdggs.dggs.WGS84_003
cell from suid	rdggs.cell(suid)
cell from point	rdggs.cell_from_point()
get outline	cell.boundary(n=2, plane=False)
	cell.vertices(plane=False)
get centroid	cell.centroid()
region filler	cells = rdggs.cells_from_region(res, se, nw)
	(returns a nested list indicating topology)
coordinates	cell.centroid(plane=False)
	rdggs.cell_from_point(coords, resolution)
child access	cell.subcells()
neighborhood	cell.neighbor, cell.neighbors

2.2.2 OpenEAGGR

RiskAware developed and published the Open Equal Area Global Grid (OpenEAGGR) library in 2017 (Bush, 2017). The OpenEAGGR library implements an fixed orientation icosahedron as the base solid and uses the ISEA projection. It provides two different tesselations: 1) a fully congruent and functional hierarchical triangle tesselation (ISEA4T), and 2) an non-congruent aperture 3 hexagonal DGGS (ISEA3H), which only supports offset indexing, but not hierarchical or unique identifier-based indexing. The API provides functions to convert to and from geographic coordinates, grid traversing and methods for spatial analysis such as intersections, proximity (Table 2). OpenEAGGR uses an accuracy concept, which does not seem to allow direct access to a hierarchical resolution level configuration.

Table 2. OpenEAGGR API highlights

Description	API details
DGGS object	dggs = Eaggr(Model.ISEA4T)
cell	dggs.convert_point_to_dggs_cell(lat, lon, acc)
cell to point	convert_dggs_cell_to_point()
outline	convert_dggs_cell_outline_to_shape_string()
parent cell(s)	get_dggs_cell_parents()
child cells	get_dggs_cell_children()
sibling cells	get_cell_siblings()
BBOX cell	get_bounding_dggs_cell()

2.2.3 H3

The Hexagonal Hierarchical Geospatial Indexing System (H3) was developed at Uber as a grid system for optimization of ride pricing and dispatch, for visualizing and exploring spatial data (Uber Technologies Inc, 2018). H3 is internally used in different business processes to aggregate and analyze geographic information to set dynamic prices and make other decisions on a city-wide level. The H3 DGGS is based on an aperture 7 hexagonal tessellation of the icosahedron. The orientation of the base icosahedron is fixed with all vertices located in the ocean to minimize distance distortions in cities. Two indexing method provides flexible and computationally efficient methods for neighboring and hierarchical traversing. Table 3 only shows a small excerpt of the user-fiendly H3 functionalities.

Table 3. H3 API highlights

Description	API details
DGGS object	fixed, not necessary
cell	geoToH3(lat, lng, res)
cell to point	h3ToGeo(h3Index)
outline	h3ToGeoBoundary(h3Index)
parent cell(s)	h3ToParent(h3Index, res)
child cells	h3ToChildren(h3Index, res)
sibling cells	kRing(h3Index, ringSize)
	hexRing(h3Index, ringSize)
	hexRing(h3Index, ringSize)
region filler	polyfill(coordinates, res)

2.2.4 Google S2

S2 is an advanced library for computational geometry and spatial indexing on the sphere. S2 uses the cube as the base polyhedron, with a fixed orientation. Two faces are located with their centroids on the poles, one face having its centroid on the intersection of the prime meridian and the equator, and the other faces evenly surrounding the equator. The chosen partition shape for the cube is the square with a fully congruent aperture 4 refinement. For indexing, Hilbert space-filling curve is used to recursively index the cells for all resolutions on the 6 faces of the cube. S2 provides methods for the conversion from geographic coordinates to cells and back, but its strength lies in computational geometry (point indexing, great circle distances, areas and angles). Table 4 demonstrates that the S2 API logic varies from the above mentioned DGGS in that aspect.

2.2.5 DGGRID

DGGRID is an open-source command-line application written in C++(Sahr, 2019) to construct icosahedral DGGSs and allows to define set of design parameters such as tessellation shape (triangles, diamonds, hexagons), orientation, and projection method (ISEA or Fuller-Gray), a

Table 4. S2 API highlights

Description	API details
DGGS object	fixed, not necessary
LatLon obj	S2LatLng.FromDegrees(lat, lon)
cell	cell_id = S2CellId(LatLon)
	cell = S2Cell(S2CellId.FromToken(cell_id)
cell to point	.ToPoint()
outline	cell.GetS2LatLngVertex(pos 0 - 4)
	(or) polygon = S2Polygon(cell)
parent cell	cell.parent(res)
child cells	cell.child(0-3)
region filler	rect = S2LatLngRect(ll, ur)
	S2RegionCoverer().GetCovering(rect)
neighborhood	latLngToNeighborKeys()
S2 point index	S2 points (not cells yet)
	can be in-memory indexed

resolution and area of interest (Sahr, 2011). DGGRID only operates with the icosahedron as the base polyhedron and its core functionality revolves around binning point data into DGGS cells at a chosen resolution. DGGRID allows to export generated DGGS to standard GIS formats. To programmatically utilize the potential of the tool several software packages provide convenient wrappers, the most prominent being dggridR for the R Statistical language (Barnes, 2018). Several Python packages are available as well (Hojati, 2019; Correndo, 2019; Kmoch, 2020). The rather high-level functionalities around the DGGRID tool can be summarized as follows:

- region filler, i.e. constructing are grid with given parameters, incl. tessellation shape, resolution, rotation for either the whole globe or a bounding box;
- retrieve either full geometries, centroids or cell identifiers for the parameterized grid;
- reversely, generate full cells from either cell ids or points for a parameterized grid definition;
- sample/aggregate point data into dggs cells, i.e. binning

3 Conceptual workflows to convert spatial data into DGGS

The most important functionality for any DGGS implementation is to index and convert data into a DGGS (ingestion). We describe considerations for ingesting generic raster data, and point and polygon vector data. Once indexed, data values can be hierarchically aggregated towards lower resolution parent cells by the cell indexes using preferential statistical aggregation method (mean or median in case of numerical or mode in case of categorical values).

3.1 Raster

For ingestion of raster data several strategies can be used. The most straightforward approach is to use coordinates of raster pixels centroids to get corresponding DGGS cells indexes on desired resolution. This method allows to avoid expensive vector geometry generation for point query for zonal statistic. The downside of this method is the necessity to assure that sampling DGGS resolution will match resolution of the ingested raster. A coarser DGGS resolution will lead to data loss due to less frequent sampling points (cells centroids) distribution than raster pixels. In case of significantly finer resolution, there will be an unnecessary overhead during sampling process and loss in performance of the algorithm. To provide decision support, a simple pseudo algorithm is suggested. First, the input raster pixel size is retrieved from the metadata. Then for a range of DGGS resolutions average cell edge size is calculated and compared with raster pixel size. If a cell area is less than a given desired pixel size then this DGGS resolution can be chosen for the data ingestion. If this ratio is about 1/2 or 1/3, all input raster pixels will be sampled by DGGS cell centroids and no data loss occurs. The procedure needs to be conducted band by band in case the raster file contains multiple bands.

Another option is to use cell polygons with a raster zonal statistic algorithms. This method will be more computationally expensive as it is necessary first to construct full cell geometries for the region extent of the raster. An advantage of this approach is the possibility to use any lower desired resolution of DGGS for ingestion with any desired statistical aggregation method of raster data into DGGS cell.

3.2 Points

The ingestion of vector points is the simplest scenario as all DGGS implementations have methods for conversion of geographic coordinates to cells indexes. The procedure is the same as raster pixels centroids ingestion scenario described above. However, vector data can have many attributes with different data types.

Independently of their quantity, attributes types and other characteristics, the input source will be converted into array of DGGS indexes and stored as table rows. The value of the optimal resolution is more difficult. Potentially, nearest neighbour or Voronoi polygon generation could provide hints to smallest distances that could be considered unique cells. However, depending on the data occurrences at very close or identical location might be acceptable. Value of optimal resolution can also be calculated by processing of all objects in a loop, where features are converted to DGGS indexes and duplicates are counted. With each new loop the level of resolution will increase by one unit until there are no duplicates or the resolution value is at a defined maximum. Defining field types for the ingested DGGS data table is based on the recognition of attributes types of the input features.

3.3 Polygons

Polygonal vector data ingestion is more complicated due to necessity in careful selection of DGSS resolution level to avoid data loss due to undersampling. A basic strategy is to get DGGS cell centroids on a manually defined fine enough resolution and use spatial intersections to assign desired polygon attributes to cells. To automatically select an appropriate resolution the following pseudo algorithm might be used: First, calculate an area of the smallest polygon in the input vector dataset. Then, calculate an average cell area of a range of DGGS resolutions. If the cell ratio of polygon area/cell area is smaller than the desired value, then the corresponding resolution is appropriate for ingestion. Depending on the aperture of the DGGS type, each next higher resolution already decreases the DGGS cell size by a factor of 4, 7 or even 9, which will quickly reduce the area discrepancy of the filling cells versus the original size of the converted polygon. In most cases we will have to process not only a single polygon but often it is required to deal with multipolygons, i.e. collection of polygons. The conversion of single polygons should then be done in a loop of conversion of each subpolygon inside parent polygons.

4 Discussion and Conclusions

The H3 system is very popular and has been widely adopted, including support for H3 cell indices in various sofwtare such as Deck.gl and Kepler.gl and the FME software (Safe Software, 2022). DGGS software started out as point location indexing and aggregation systems, in particular DGGRID, H3 and S2. However, for aggregating and area-true statistical analysis of environmental variables they are less suitable than true "raster-type" alternatives. rHEALPix on the other side, and ISEA-based DGGS (Snyder equal area projection in DGGRID and OpenEAGGR), are ideal candidates for gridded representation of continuous phenomena (like in GeoTIFF or NetCDF formats). For example, H3 is an aperture 7 hexagonal grid that use the Gnomonic projection which is not equal area. Google S2 builds on a cube-sphered projection which only guarantees area variations to stay within a factor of 1.5 (Veach et al., 2017). Thus, H3 and S2 should not be considered in environmental modelling or equal-area DGGS-based data cubes, because they do not have equal area cells across the globe. For example, when intending to aggregate large scale landuse, forest cover, soils and climatologies and calculate area-based statistics, rHEALPix and ISEA-based grids should be used.

DGGSs like H3, S2 OpenEAGGR have useful and mostly straight-forward indexing systems that provide practical neighborhood functions, and hierarchical cell access. This is a particular DGGS semantic that the same cell id is always at the same location. Passing that cell ID to another program or platform enables to exactly identify location and resolution under the assumption the DGGS type is known.

There are a few core opportunities to suggest:

- improving access and installation for these tools, S2 and DGGRID are for example still hard to compile and install on windows systems;
- this would support seamless uptake with other open-source communities around Osgeo and Pangeo;
- there is currently no equal-area congruent hierarchical hexagonal DGGS in a library module comparable to the functionalities of H3, S2 or rHEALPix;
- developing a portable C-compatible base library for rHEALPix, so the DGGS can be used across languages, including the web (JavaScript)

We believe that it will be very typical to use these sorts of grids in 10 years. The challenge is adapting our analysis software so that the user experience is as seamless as it is with traditional raster, i.e. rectangularly gridded data.

Author contributions. AK, OM and EU designed the study, AK, OM and IV ran the experiments and created the code. AK and OM drafted the manuscript and created the figures, and all authors contributed to the paper writing.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. This research has been supported by grant number MOBERC34 of the Estonian Research Council (ETAG) and the Estonian national scholarship programme for international researchers and academic staff of the Archimedes foundation.

References

Adams, B.: Wāhi, a discrete global grid gazetteer built using linked open data, International Journal of Digital Earth, 10, 490–503, https://doi.org/10.1080/17538947.2016.1229819, 2017.

Alderson, T., Purss, M., Du, X., Mahdavi-Amiri, A., and Samavati, F.: Digital Earth Platforms, in: Manual of Digital Earth, edited by Guo, H., Goodchild, M. F., and Annoni, A., pp. 25–55, Springer, Singapore, 1 edn., https://doi.org/10.1007/978-981-32-9915-3, 2020.

Barnes, R.: dggridR: Discrete Global Grids for R v2.0.3, [software], https://doi.org/10.5281/zenodo.1322866, last accessed Feb 18, 2022, 2018.

Bousquin, J.: Discrete Global Grid Systems as scalable geospatial frameworks for characterizing coastal environments, Environmental Modelling & Software, 146, 105–210, https://doi.org/10.1016/j.envsoft.2021.105210, 2021.

Bowater, D. and Stefanakis, E.: The rHEALPix discrete global grid system: Considerations for Canada, Geomatica, https://doi.org/10.1139/geomat-2018-0008, 2018.

Bush, I.: OpenEAGGR - Open Equal Area Global GRid, [software], https://github.com/riskawaretd/openaggr, last accessed Feb 18, 2022, 2017.

Chaudhuri, C., Gray, A., and Robertson, C.: InundatEd-v1.0: a height above nearest drainage (HAND)-based flood risk modeling system using a discrete global grid system, Geoscientific Model Development, 14, 3295–3315, https://doi.org/10.5194/gmd-14-3295-2021, 2021.

Correndo, G.: dggridpy: a Python binding to the DGGRID library, [software], https://github.com/correndo/dggridpy, last accessed Feb 18, 2022, 2019.

Gibb, R., Raichev, A., and Speth, M.: The rHEALPix DGGS preprint, https://raichev.net/files/rhealpix_dggs_preprint.pdf, last accessed Jan 08, 2022, 2016.

Goodchild, M. F.: Reimagining the history of GIS, Annals of GIS, 24, 1–8, https://doi.org/10.1080/19475683.2018.1424737, 2018.

Gorski, K. M., Hivon, E., and Wandelt, B. D.: Analysis Issues for Large CMB Data Sets, arXiv, eprint: astro-ph/9812350, 1998.

Hojati, M.: pydggrid: Python wrapper for DGGRID v0.0.1, [software], https://doi.org/10.5281/zenodo.3228406, last accessed Feb 18, 2022, 2019.

Jendryke, M. and McClure, S. C.: Mapping crime – Hate crimes and hate groups in the USA: A spatial analysis with gridded data, Applied Geography, 111, 102 072, https://doi.org/10.1016/j.apgeog.2019.102072, 2019.

Kmoch, A.: dggrid4py: A set of python modules for creating and manipulating Discrete Global Grids with DGGRID. v0.2.3, [software], https://doi.org/10.5281/zenodo.5873330, last accessed Feb 18, 2022, 2020.

Kmoch, A.: Working with Discrete Global Grid Systems - AGILE 2021 workshop materials, [software], https://doi.org/10.5281/zenodo.5873218, last accessed April 18, 2022, 2021.

Lei, K., Qi, D., and Tian, X.: A New Coordinate System for Constructing Spherical Grid Systems, Applied Sciences, 10, 655, https://doi.org/10.3390/app10020655, 2020.

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

Liao, C., Tesfa, T., Duan, Z., and Leung, R.: Watershed delineation on a hexagonal mesh grid, Environmental Modelling & Software, 128, https://doi.org/10.1016/j.envsoft.2020.104702, 2020.

Open Geospatial Consortium: Topic 21: Discrete Global Grid Systems Abstract Specification, https://portal.opengeospatial.org/files/15-104r5.pdf, last accessed Feb 18, 2022, 2017.

Purss, M. B. J., Peterson, P. R., Strobl, P., Dow, C., Sabeur, Z. A., Gibb, R. G., and Ben, J.: Datacubes: A Discrete Global Grid Systems Perspective, Cartographica, 54, 63–71, https://doi.org/10.3138/cart.54.1.2018-0017, 2019.

Rawson, A., Sabeur, Z. A., and Brito, M.: Intelligent geospatial maritime risk analytics using the Discrete Global Grid System, Big Earth Data, pp. 1–29, https://doi.org/10.1080/20964471.2021.1965370, 2021.

Robertson, C., Chaudhuri, C., Hojati, M., and Roberts, S. A.: 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, 2020.

Safe Software: Integrate H3 Using FME, [website], https://www.safe.com/integrate/h3/, last accessed April 18, 2022, 2022.

Sahr, K.: Hexagonal discrete global grid system for geospatial computing, Archives of Photogrammetry, Cartography and Remote Sensing, 22, 363–376, 2011.

Sahr, K.: DGGRID version 7.0, [software], https://www.discreteglobalgrids.org/software/, last accessed Feb 18, 2022, 2019.

Uber Technologies Inc: H3: Hexagonal hierarchical geospatial indexing system, [software], https://h3geo.org/, last accessed Feb 18, 2022, 2018.

Veach, E., Rosenstock, J., Engle, E., Snedegar, R., Basch, J., and Manshreck, T.: S2 Geometry Library: Computational geometry and spatial indexing on the sphere, [software], https://s2geometry.io, last accessed Feb 18, 2022, 2017.