Reference: S2_MPC_DGGS_Tasks_1_2_Report
Issue: 06
Date: 2024-06-21

S2 MPC

Sentinel-2 ARD handling with DGGS

Ref. S2_MPC_DGGS_Tasks_1_2_Report

Authors Table

	Name	Company	Responsibility	Date	Signature
Written by	F. Benatia, G. Salgues, N. Vila	CS	Remote sensing engineer	2024-06-21	
Approved by	L. Pessiot	CS	Project Manager	2024-06-21	 Signature numérique de Laëtitia Pessiot Date : 2024.06.21 15:23:34 +02'00'

Change Log

Issue	Date	Reason for change	Pages(s)/Section(s)
01	2022-11-03	Creation	
02	2023-01-12	JRC RIDs + additional remarks	
03	2023-07-10	Addition of Task 2 outcomes	Section 3.1.1: H3 retrieving performance Section 3.2.1: Complement on rHEALPix Section 4.2: Example for cloud masks
04	2023-12-18	Addition of tables and precisions about Task 2 outcomes	Sections 1 and 5 added
05	2024-03-15	Addition of new outcomes and dispatching of section 5 content within the document	Section 5 removed (content merged with the rest of the document)
06	2024-06-21	Updates following ESA comments	Section 2.3: update of the "ARD Product Resolution vs DGGS resolution" bullet Section 2.7.2.4: addition of more details regarding the comparison between COG vs Parquet (H3) storage Section 2.8: delete of the conclusion regarding the S2 suitability assessment evaluation

Table of contents

1. Introduction

1.1 Context of the study

Currently, remote sensing data is increasingly freely and openly available from different satellite platforms. However, the variety of images in terms of sensor types, spatial, spectral, and temporal resolutions, image georeferencing, file formats, etc. generate complexity that constrains and slows down the exploitation of satellite imagery. This heterogeneity requires advanced skills and tools to transform Earth Observation (EO) imagery into useful information, making it almost inaccessible for non-expert users.

From another point of view, dealing with geospatial data at a global scale – either trying to make sense of events or modelling a physical phenomenon – requires subdividing the globe into regions (mainly for computing reasons: CPU, parallelization, …). This usually requires defining a way to cut the space in "homogeneous" regions, adopting a local coordinate reference system (to eliminate many of the inconsistencies and distortions inherently present with traditional "flat Earth" map projections), having to reproject constantly to deal with a new region of space, or to use a gridding system like geo-hashing, where the distortion at the poles makes it difficult to make correct analysis at global scales.

Addressing these issues requires innovative approaches to organize, manage, and analyze remote-sensing imagery. Discrete Global Grid Systems (DGGS for short), currently under study by the OGC DGGS SWG, allow to subdivide the Earth's sphere into (almost) equal size regions that can be used to index geospatial data into homogeneous cells.

DGGS can act as a unified framework for EO data integration, multisource data fusion and cloud computing on a global scale. Therefore, DGGS can play a key role in the development of new products within the European EO downstream sector, which must incorporate interoperability best practices, automatization, systemization, visualization, and on-the-fly processing through integrated web-services.

1.2 Objective of the study

The aim of this study is to assess the use of a DGGS for Sentinel-2 ARD manipulation and analysis. A first task, which is the subject of this document, is to select a suitable DGGS framework for Sentinel-2 data in order to perform more testing in a dedicated proof of concept (Task 02).

The selection criteria are based on the core characteristics of any DGG system (Sahr et al., 2003). Considering the Sentinel-2 data specifications these criteria can be filtered to keep the most relevant for S2 ARD regarding constraints like storage size, data retrieval efficiency, data restoration accuracy and bands resolutions.

To support this analysis multiple type of DGG systems need to be generated and compared. This was achieved by testing existing libraries and by implementing a python dggs toolbox based on existing reference tools. The toolbox is described in Appendix A.

1.3 Definition of DGGS

As introduced in the previous chapter, a Discrete Global Grid System (DGGS) is a spatial reference system that uses a hierarchy of global tessellations to partition the surface of the Earth into grid cells.

DGGS starts from a base polyhedron inscribed on the Earth's sphere to obtain regular subdivisions and further subdivide those into more fine-grained regions. Individual observations can be assigned to cell corresponding to both the position and size of the phenomenon being observed – meaning that the resolution and precision of the data capture is inherently part of the stored data, and not something that needs to be explained in metadata – and potentially overlooked.

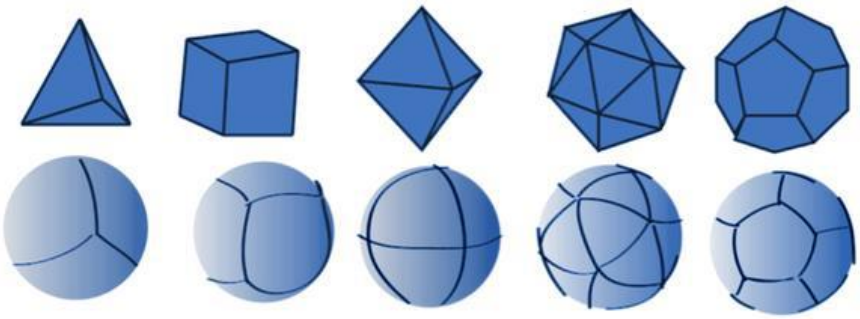


Figure 1: Regular polyhedral (top) and their corresponding initial tessellation (bottom) source: (Purss & al., 2017)

A DGGS – to be compliant with the OGC DGGS definition[1] - must satisfy 18 requirements amongst them:

- ✓ Earth is partitioned into cells which should have geometries of equal area (for a given resolution and within the specified level of precision),
- ✓ Partitioning has no arbitrary limits (poles, dateline, …),
- ✓ Each cell has a unique identifier which characterizes its resolution and geographical extend,
- ✓ Cells do not overlap at a given resolution,
- ✓ Cells have well defined parent/child/neighbor relationships usable in multi-resolution queries.

The use of a discrete set of cells rather than coordinate pairs has several advantages in space-time algebraic calculations. The first is that it provides a homogeneous environment for integrating, aggregating, and visualizing both vector/point cloud geometries and raster-based geospatial data sources as depicted in the following figure.

[1] https://docs.opengeospatial.org/as/15-104r5/15-104r5.html

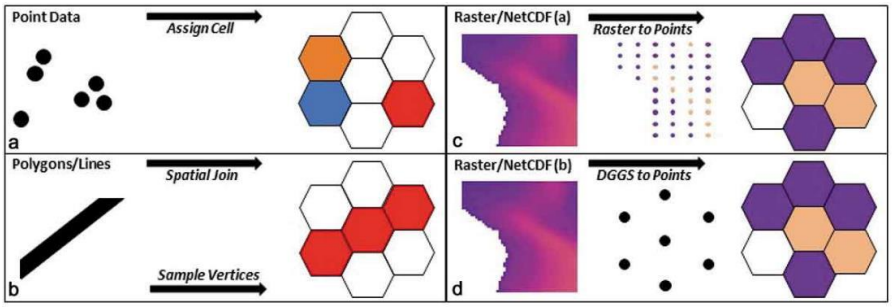


Figure 2: Data integration via DGGS source: (Rawson et al., 2022)

Secondly, the immutability of the value of a cell at a spatio-temporal address (constituted by cell ID and time pair) and the partitioning into cells of equal area facilitates algebraic ensemble calculations and statistics on some or all the aggregated measurements in a DGGS-based platform. It allows calculations on large sets of cells to be trivially distributed among an arbitrary number of processors. And once a calculation has been performed, the result itself can be indexed in the system and subsequent queries requiring that calculation become a lookup operation, eliminating duplication of work throughout the system. Moreover, the global scope of a DGGS immediately means that algorithms defined for one region of the DGGS reference frame become deployable anywhere within the DGGS.

All these advantages make DGGS a particularly well-suited data organization system to meet the needs of data cubes (Purss and al, 2019), and to provide ready-to-use (already aligned) data (ARD).

2. S2 DGGS selection criteria

2.1 The characteristics of a DGGS

The main components of any DGGS are the following:

- ✓ **Base polyhedron:** the platonic solid or regular polyhedron used as a base for the DGGS. The most used solids are icosahedron (20 faces polyhedron) and hexahedron (cube). The icosahedron has the smallest face size and, therefore any DGGSs defined on it tend to display relatively small distortions. The icosahedron is thus the most common choice for a base platonic solid.(Sahr et al., 2003).
  - Uber H3 and ISEA DGGS are based on an icosahedron.
  - Google S2 and rHEALPix are based on a hexahedron.
- ✓ **The orientation**: a fixed positioning of the base polyhedron vertices on the surface of the earth. Fuller's Dymaxion orientation is the only known icosahedron placement with no icosahedron vertices falling on land. This projection places all 12 vertices of the icosahedron in the ocean (Sahr et al., 2003)

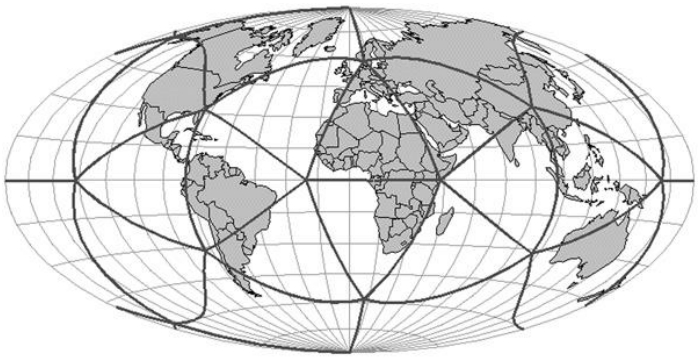


Figure 3: Spherical icosahedron (in Mollweide projection) oriented using Fuller's Dymaxion orientation. Note that all vertices fall in the ocean.(Sahr et al., 2003)

- ✓ **The projection**:
  - For equal area: Icosahedral Snyder Equal Area (ISEA): "Snyder's Icosahedral Equal Area map projections on polyhedral globes for the dodecahedron and truncated icosahedron offer relatively low scale and angular distortion […] The interruptions remain a disadvantage, as with any low-error projection system applied to the entire globe" (Snyder, 1992)
  - For shape preservation: Gnomonic (H3) with more distortions away from the center of the face
  - A compromise between equal areas and shape preservation is Fuller-gray grid (https://observablehq.com/@fil/gray-fuller-grid)
- ✓ **Face of the polyhedron** that can be squares (S2), triangles (ISEA4T), diamonds (ISEA4D) or hexagons (H3, ISEA7H).
- ✓ **The aperture** and **resolution levels**:
  - Aperture: parent cell subdivision into N child (ex: Uber H3 is Aperture 7)
  - Resolution levels: Consecutive subdivisions (by the aperture value) of the base level (resolution 0) give the resolution levels of a DGGS.

Figure 4: Uber H3 aperture 7

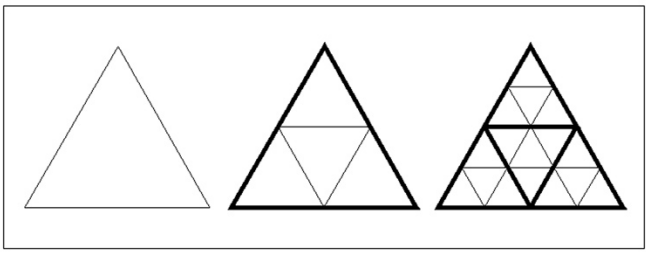


Figure 5: Aperture 4 on a triangle source: (Sahr et al., 2003)

- ✓ **Indexing strategy**:

  DGGS can have different ways to calculate cells ID according to different strategies. Here we are referring to the way each DGGS calculates children IDs based on parents' cells IDs:

  - Hierarchical indexing (rHEALPix, H3, ISEA)
  - Axes-based indexing
  - Space-filling curvers (Google S2)
  - GeoHash

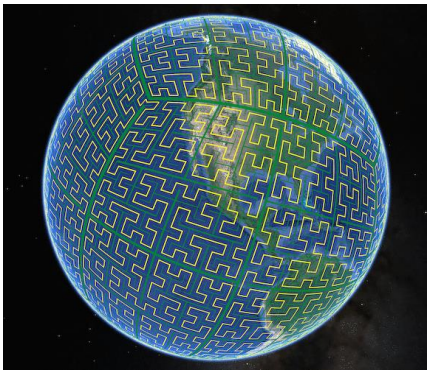
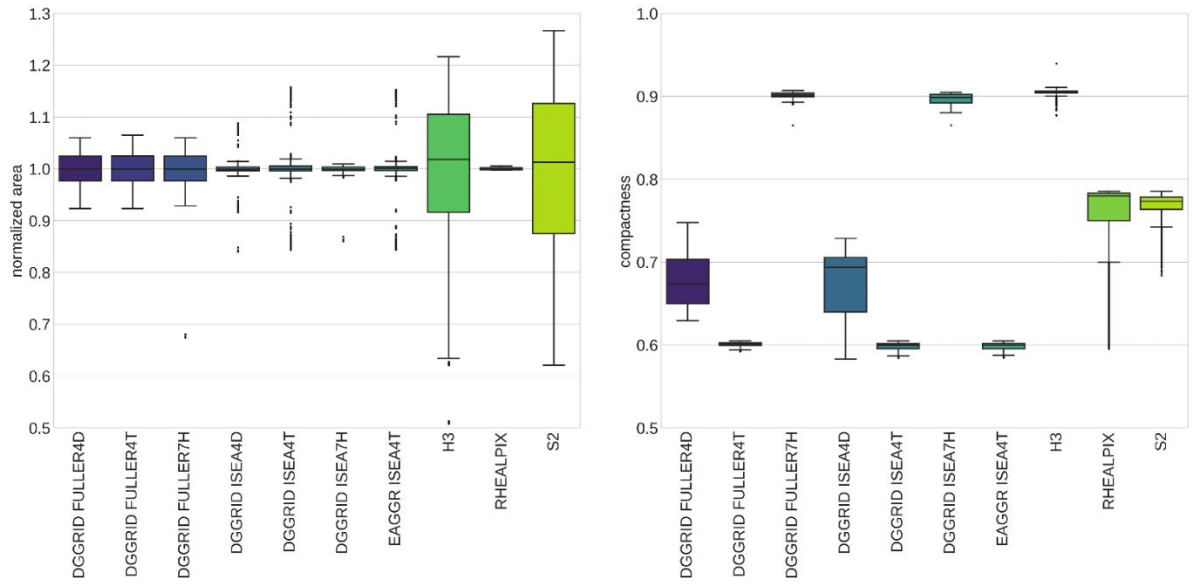


Figure 6: S2 Space-Filling curve indexing

2.2 Area and shape distortion



(a) Area distortions over normalized areas of tested DGGSs.

(b) Compactness distribution across tested DGGS.

Figure 7: Area distortions and compactness distribution across multiple DGGS (Kmoch et al., 2022)

The cell shape and projection are two key aspects of the selection of a DGGS. These aspects have a direct impact on the usability of a DGGS for certain applications. (Kmoch et al., 2022) compared popular DGGS implementations (Figure 7) and we can see that a DGGS with a significant area distortion cannot be recommended for large/global scale (area based) statistical comparisons. This is the case of H3 that relies on a gnomonic projection. A better choice in this case would be an equal area DGGS like ISEA7H.

ISEA based DGGS with hexagonal cells can combine the benefits of an equal area projection and hexagonal shape preservation. Although area and shape preservation are important aspect for many Earth Observation applications, many users prefer to use H3 grid for the convenience and the large software support (Kmoch et al., 2022).

2.3 DGGS selection for ARD Product

A perfect DGGS does not exist (and will probably never exist) so it is all a question of tradeoff. Here are the points to be discussed for the DGGs selection:

- ✓ **Shape of cells**: the closer it is to a circle, the less radiometry will be impacted.
- ✓ **Aperture** choice: the aperture can have a significant impact on the storage size to match the native resolution. The higher the aperture and the less choice we will have for intermediate resolutions.
- ✓ **Parent-Children relationship**: most DGGS will make one cell to have a single parent, but this is not a golden rule. With ISEA7H for instance a cell can have 2 parents which makes DGGS data even more complicated to handle.

- ✓ **ARD Product Resolution vs DGGS resolution**: Whatever the DGGS the resolution levels are fixed by definition and cannot be adjusted to a particular product. The imagery products have to fit the pre-defined grids' cells. One DGGS resolution level could fit one product resolution (for example S2) but could not fit another product resolution (for example L8/9)

  As DGGS and target products' resolutions are not aligned, a compromise must be found to pick the best DGGS and associated resolution levels (the closest to the native product resolutions). For H3, the most suitable resolutions for S2 ARD are 11, 12, 13 (respectively to address 60 m, 20 m, 10 m per pixel resolutions).

  We selected the resolution levels 11, 12 and 13 by comparing the cells area with the native product pixel area so that we don't lose information. Ex: level 11 cell area < 60 m product pixel area (3600 m²) < level 12 cell area.

  In the table below, the DGGS cells resolutions, which better suit the Sentinel-2 imagery resolutions (60 m, 20 m and 10 m), are highlighted in green. To do so, the resolution level must have cells with area smaller than one pixel area.

  - 60 m resolution pixel = 3600 m² = 0,0036 km²
  - 20 m resolution pixel = 400 m² = 0,0004 km²
  - 10 m resolution pixel = 100 m² = 0,0001 km²

  We notice that for H3 the best resolutions are levels 11, 12 and 13, like for rHEALPix (aperture 9).

  ISEA4T has interesting resolution for 20 m and 10 m (levels 18 and 19) but the best resolution level for 60 m resolution (level 17) implies an important oversampling (cell size is 1484 m² for 3600 m² pixels)

  rHEALPix aperture 4 has the same issue as ISEA4T, but with levels 18, 19 and 20.

  The following table shows the different DGGS resolutions and corresponding cells areas. This table tries to summarize which DGGS resolution level is suitable for a Sentinel-2 imagery resolution (highlighted in green).

  Cells area MUST be smaller than the native pixel area:

60m resolution pixel =	0,0036 km²
20m resolution pixel =	0,0004 km²
10m resolution pixel =	0,0001 km²

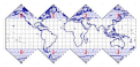
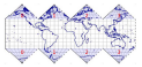
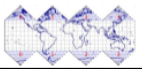
  Two consequences: if DGGS cell area is smaller than the native pixel area, there is an oversampling. In case the cell area is bigger, then we have a down sampling and an information loss.

Table 1: DGGS/S2 ARD Product resolutions alignment







	ISEA3H	ISEA4D	ISEA4T	ISEA4H	H3	H3	rHEALPix	rHEALPix	rHEALPix
	Aperture 3 Hexagon	Aperture 4 Diamond DGG	Aperture 4 Triangle DGG	Aperture 4 Hexagon DGG	Aperture 7 Hexagon DGG	Aperture 7 Hexagon DGG	Aperture 4 DGG	Aperture 9 DGG	Aperture 14 DGG
Resolution	Hex Area2 (km²)*	Cell Area (km²)*	Cell Area (km²)*	Hex Area2 (km²)*	Hex Area2 (km²)	Pentagon Area2 (km²)	Cell Area (km²)**	Cell Area (km²)**	Cell Area (km²)**
0	N/A	51006587,635000	25503293,817000	N/A	4357449,416078	2562182,162955	100263556,426512	100263556,426512	100263556,426512
1	17002195,878000	12751646,909000	6375823,454000	12751646,909000	609788,441794	328434,586246	25065889,156694	11140395,158501	6266472,289173
2	5667398,626000	3187911,727000	1593955,864000	3187911,727000	86801,780399	44930,898498	6266472,289173	1237821,684278	391654,514944
3	1889132,875000	796977,932000	398488,966000	796977,932000	12393,434655	6315,472268	1566618,059777	137535,745170	24478,407966
4	629710,958000	199244,483000	99622,241000	199244,483000	1770,347654	896,582383	391654,514944	15281,748639	1529,900302
5	209903,653000	49811,121000	24905,560000	49811,121000	252,903858	127,785583	97913,631865	1697,972346	95,618867
6	69967,884000	12452,780000	6226,390000	12452,780000	36,129062	18,238750	24478,407966	188,663686	5,976167
7	23322,628000	3113,195000	1556,598000	3113,195000	5,161293	2,604669	6119,601209	20,962662	0,373517
8	7774,209000	778,299000	389,149000	778,299000	0,737328	0,372048	1529,900302	2,329195	0,023345
9	2591,403000	194,575000	97,287000	194,575000	0,105333	0,053147	382,475076	0,258796	0,001459
10	863,801000	48,644000	24,322000	48,644000	0,015048	0,007592	95,618867	0,028754	0,000091
11	287,934000	12,161000	6,080000	12,161000	0,002150	0,001085	23,904668	0,003195	0,000006
12	95,978000	3,040000	1,520000	3,040000	0,000307	0,000155	5,976167	0,000355	0,000000
13	31,993000	0,760000	0,380000	0,760000	0,000044	0,000022	1,494042	0,000039	0,000000
14	10,664000	0,190000	0,095000	0,190000	0,000006	0,000003	0,373517	0,000004	0,000000
15	3,555000	*0,047500*	*0,023750*	*0,047500*	0,000001	0,000000	0,093379	0,000000	0,000000
16	1,185000	*0,011875*	*0,005938*	*0,011875*			*0,023345*		
17	0,395000	*0,002969*	*0,001484*	*0,002969*			*0,005836*		
18	0,132000	*0,000742*	*0,000371*	*0,000742*			*0,001459*		
19	*0,044000*	*0,000186*	*0,000093*	*0,000186*			*0,000365*		
20	*0,014667*	*0,000046*	*0,000023*	*0,000046*			*0,000091*		
21	*0,004889*	*0,000012*	*0,000006*	*0,000012*			*0,000023*		
22	*0,001630*						*0,000006*		
23	*0,000543*								
24	*0,000181*								
25	*0,000060*								
26	*0,000020*								

60m resolution pixel =	0,0036 km²
20m resolution pixel =	0,0004 km²
10m resolution pixel =	0,0001 km²

* Source: https://webpages.sou.edu/~sahrk/dgg/isea.old/tables/tables.html

** Source: https://doi.org/10.1080/20964471.2021.2017539

DGGS software support: DGGS software landscape is poor, and a lot of libraries and tools are not updated regularly. Some projects seem abandoned. Development of new software and libraries will be necessary to address the needs for all the steps conversion, ingestion, storage, visualization, and processing/analysis.

In the table below, we tried to summarize three key aspects of the major software implementations for each DGGS grid:

- ✓ The language support (**yes** if supported, **no** if it's not supported and no wrapper is currently available): This depends mostly on the language used for the library implementation. Please note that it is technically possible to create new wrappers for C++ implementations in other languages.
- ✓ The storage support: this column highlights different options to store DGGS data, either in database or files. Here, we focused on solution that are provided an API (functions) to handle DGGS data directly from the database.
- ✓ Please note Parquet format is a specific column-oriented cloud-native file format, which can be queried remotely with SQL language (with DuckDB 2for instance)
- ✓ The visualization engines which support DGGS grids. **TODO** means an implementation/integration is possible without having to directly modify the library.

Table 2: Current DGGS software and libraries functionalities support

		Language support (native language library or wrapper support)					Compatibility with raster GIS library	Storage applicable to DGGS (Databases & File format implementing a DGGS API for data and grid handling)				Visualization capabilities (support of DGGS by existing toolkits, independently from storage)			
DGGS	Libraries	Python	Java	C/C++	R	Javascript	GDAL Proj	PostGreSQL PostGIS	ElasticSearch	ClickHouse	Apache Parquet	OpenLayers	MapLibreGL MapboxGL	KeplerGL	DeckGL
S2	**s2geometry** (22/11/2023, Go around 40% complete, Java, Kotlin (Complete except binary serialization)	Yes (native)	Yes (native)	Yes (native)	Yes (native)	Yes (native)		Yes (wrapper)		Yes (native)					
H3	**H3** (multiple bindings)	Yes (native)	Yes (native)	Yes (native)	Yes (native)	Yes (native)		Yes (native)	Yes (native)	Yes (native)	TODO	TODO	TODO	Yes (native)	Yes (native)
rHEALPix	**rhealpixdggs-py** 0.5.3 (Python, 15/12/2020) **GeoTools** (Java org/geotools/dggs/rhealpix)	Yes (native)	Yes (wrapper)	Yes (native)	Yes (native)		Yes (native)				TODO				
ISEA3H	**OpenEAGGR** (C/C++, 14/08/2017), **EAGGRPython** (Python wrapper, 14/08/2017)	Yes (wrapper)	Yes (wrapper)	Yes (native)	Yes (native)			Yes (native)	Yes (native)		TODO				
ISEA4H	**EAGGRJava** (Java wrapper, 14/08/2017) **EAGGRPython** (Python wrapper: ISEA4T, ISEA3H,	Yes (wrapper)	Yes (wrapper)	Yes (native)	Yes (native)			Yes (native)	Yes (native)						
ISEA7H	14/08/2017) **EAGGRElasticsearch** (14/08/2017),	Yes (wrapper)	Yes (wrapper)	Yes (native)	Yes (native)			Yes (native)	Yes (native)						
ISEA43H	**EAGGRPostgres** (14/08/2017),	Yes (wrapper)	Yes (wrapper)	Yes (native)	Yes (native)			Yes (native)	Yes (native)						
ISEA4T	**DGGRID** 7.8, 8.0b (C/C++, 24/10/2023), **dggrid4py** (Python wrapper, DGGRID 7.5, 01/02/2023)	Yes (wrapper)	Yes (wrapper)	Yes (native)	Yes (native)			Yes (native)	Yes (native)		TODO				
ISEA4D	**pydggrid** 0.0.13 (DGGRID 6.4, 22/05/2020) **dggridR** (DGGRID 6.2, 18/10/2023)	Yes (wrapper)	Yes (wrapper)	Yes (native)	Yes (native)			Yes (native)	Yes (native)						

Regarding the current DGGS characteristics and constraints, it might be also interesting to consider setting up a "**customized DGGS**" for **S2 ARD Products**. While DGGS have fixed characteristics, it is possible to adjust some DGGS parameters: for example with DGGRID, prime meridian can be moved, as well as orientation can be changed.

Tha is for instance what Australia did with AusPIX while focusing on local data usage (AusPIX is not suitable for other countries like European ones). Canada did the same so that rHEALPix usage only concerns a single cell geometry type: squares (as reminder, rHEALPIX implies use of triangles, diamonds and squares).

In general, modifying a global grid is often for local applications, but this point was relevant to be mentioned for potential further analysis.

[2] https://duckdb.org/

2.4 Available open source DGGS tools and libraries

There are relatively few libraries, and these libraries do not have a large user community. The developments around these libraries are often not very active, and many of them have not benefited from any development activity during the last 12 months. This is a cutting-edge area that needs to be addressed.

These libraries are often built around a single reference implementation (rhealpixdggs-py for rHEALPix DGGS, H3 for Uber's H3). Only grids built on ISEA (e.g. ISEA3H) have several implementations (DGGRID and OpenEAGRR).

Uber H3 provides implementations for multiple popular programming languages (Python, C, Java and Javascript).

Apart from rhealpixdggs-py (and the Geoserver demonstrator presented at the OGC testbed 16), none of the currently analyzed libraries has the stated objective of ensuring compatibility with OGC recommendations (OGC Topic 21 v2.0 / ISO 19170-1:20203). This does not mean that they are not compatible.

This first analysis also highlighted that these tools were mainly designed to allow quick statistical analysis on data, and by aggregating vector data. Only two examples dealing with raster data were identified (rHEALPIx, and TB16)

Only Uber H3 stands out, and has a fairly large community, thanks to the support of H3 in ClickHouse OLAP Database, PostgreSQL, MapBox [3], CartoDB [4], Kepler.gl[5], Deck.gl which are among the leaders in the manipulation, analysis and display of spatial data – but mainly vector data.

2.4.1 DGGRID

DGGRID is a very complete tool for creating and manipulating ISEA based Discrete Global Grids. This tool was primarily written in C++ by Kevin Sahr with 2 main contributors Richard Barnes and Casey Ghilardi.

Supported grids:

- ✓ SUPERFUND,
- ✓ PLANETRISK,
- ✓ ISEA3H, ISEA4H, ISEA7H, ISEA43H, ISEA4T, ISEA4D,
- ✓ FULLER3H, FULLER4H, FULLER7H, FULLER43H, FULLER4T, FULLER4D

Main features:

- ✓ Construct a variety of icosahedral DGGSs with the choice of tessellation shape, orientation, projection method and resolution.
- ✓ Presets of DGGSs: ISEA4T, ISEA4D, ISEA3H, ISEA7H, PlanetRisk, …
- ✓ Address conversions
- ✓ Binning point values

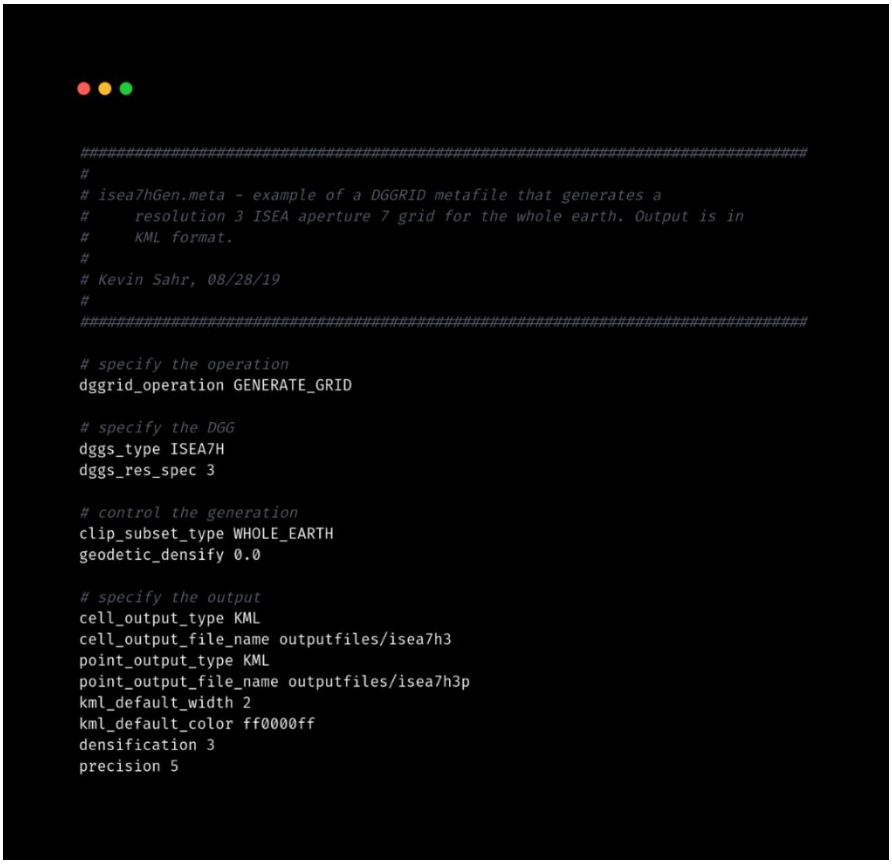
3 https://observablehq.com/collection/@nrabinowitz/h3-tutorial

4 https://carto.com/blog/spatial-functions-bigquery-uber/

5 https://docs.kepler.gl/docs/user-guides/c-types-of-layers/j-h3

The user interacts with the tool using a Command Line Interface (CLI). The configuration relies on a meta file specifying the parameters of the DGGS to create. The installation of the tool is well documented with all the instructions needed to install it on Linux, the installation process could be made easier using a docker image (especially for windows users).

Multiple bindings/wrappers for DGGRID exist: pydggrid, dggrid4py for python and Rdggrid for R.



```
##############################################################################
#
# isea7hGen.meta - example of a DGGRID metafile that generates a
#     resolution 3 ISEA aperture 7 grid for the whole earth. Output is in
#     KML format.
#
# Kevin Sahr, 08/28/19
#
##############################################################################

# specify the operation
dggrid_operation GENERATE_GRID

# specify the DGG
dggs_type ISEA7H
dggs_res_spec 3

# control the generation
clip_subset_type WHOLE_EARTH
geodetic_densify 0.0

# specify the output
cell_output_type KML
cell_output_file_name outputfiles/isea7h3
point_output_type KML
point_output_file_name outputfiles/isea7h3p
kml_default_width 2
kml_default_color ff0000ff
densification 3
precision 5
```

Figure 8: DGGRID metafile used to generate ISEA7H grid for the whole earth at resolution 3

2.4.2 H3 tools

The Uber H3 DGGS can be created and managed using a variety of tools and extensions. The main implementation (in C) is provided by Uber[6] with bindings to multiple languages (Python, Javascript, …)[7].

On top of the bindings, multiple Databases, and data services support H3 like ClickHouse, AWS Athena (via the aws-athena-udfs-h3 connector)[8] or PostgreSQL (via h3-pg)[9].

The full list of bindings and extensions can be found here:
https://h3geo.org/docs/community/bindings/

[6] https://h3geo.org/

[7] https://h3geo.org/docs/community/bindings/

[8] https://github.com/daniel-cortez-stevenson/aws-athena-udfs-h3

[9] https://github.com/zachasme/h3-pg

For this study the library H3-Pandas was used to easily generate H3 grids from the extent of a raster file.

2.4.3 RHEALPixdggs-py

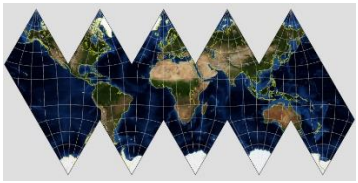
rHEALPixdggs-py implements code to both define an rHEALPix DGGS reference system and to perform topological queries on its identifiers. The roadmap for this tool is to be fully compliant with OGC Topic 21 v2.0 / ISO 19170-1:2020.

2.5 DGGS custom tooling to be developed

For DGGS testing in phase 2, we had to setup tools to project input data products (S2) onto the DGGS grid, and to store the result in a Database. A toolbox was developed to gather multiple DGGS libraries (H3 and rHEALPix), as well as Jupyter notebooks.

The mains concerns were the data handling once in database and the file export of the data once projected onto the DGGS, so:

- ✓ For H3, we focused on indexing properties thanks to the H3 SQL library.
- ✓ For rHEALPIx, we focused on testing DGGS storing as raster, thanks to good support in Proj4.
- ✓ Some ISEA DGGSs are in theory also supported by Proj4, but no test was successful due to libraries limitation. The idea here was to transform data to ISEA, like we did for rHEALPix. The fact that ISEA is partially implemented in latest Proj [10] versions was promising but conversion of raster data failed, causing segmentation faults in GDAL libraries. According to GDAL developers, contacted on Github, GDAL cannot reproject rasters unless the PROJ library supports both Forward AND Inverse on any projection. ISEA has no PROJ inverse so is not capable of raster output.



A specific effort will be necessary to industrialize a set of consistent tools and libraries to convert to and from DGGS data. Data conversion, i.e. discretizing and loading data into a DGGS-based data structure, is clearly a bottleneck.

2.6 Sentinel-2 data storage key points

An analysis of the DGGS characteristics in the context of S2 data storage immediately raises the question of the optimal resolution for referencing S2 pixels. All DGGS have different cells size for a given level function of apertures and shape. In addition, S2 MSI covers 13 spectral bands (443–2190 nm), with a spatial resolution of 10 m (four visible and near-infrared bands), 20 m (six red edge and shortwave infrared bands) and 60 m (three atmospheric correction bands). Considering H3 DGGS, at level 12, a cell has an area of 307 m², at level 13 an area of 44 m² and requires a proper resampling of S2

[10] https://proj.org/en/9.3/operations/projections/isea.html

data. Which cells size to choose for each S2 band? The wrapping of the data in such a grid involves a tradeoff between storage size and information loss. So, the question is also "What precision is needed for my application?".

As an example, during the OGC Testbed 16 – designed to test DGGS and the APIs to be implemented to exploit them – each band of Sentinel 2 data were resampled at level 11 on H3 and rHEALPix cells, which have a cell area respectively of 2150 m² and 2712 m² - far away from the original pixel area from a visible band. This produces a "Pixelized" effect when visualizing the data as shown in the following figure... but on the other hand, global statistics on NDVI could be calculated in seconds.

As we see, the answer is not trivial and will have to be studied according to the expectations of the system. It might be tempting to take the highest possible resolution, in order not to lose spatial resolution, but this would lead to a storage problem.

Indeed, at level 13, it is almost 12,000 billion [11] cells that a "whole world" system must handle. At the highest resolution level, it would require storing 597 trillion records. Few storage systems can manage such a large quantity of references, which must then be multiplied by the number of occurrences of the same cell over time (almost 50 by year for S2 data).

In contrast to the usual raster data, the storage of DGGS grid-indexed data is often addressed as vector storage and stored in a database structure. For local use, e.g. on zone of interest, GeoPackage [12] databases or HDF5 file formats (KEA [13] or NetCDF [14]) may be sufficient to store table containing cell id and measurements at several resolutions; but on a global scale, to manage such a large number of records, other solutions must be found. The use of OLAP databases such as ClickHouse may be a good solution, but only a deep analysis will tell us if this type of database can index full resolution raster data and if the resulting disk usage is acceptable.

The OGC's "DGGS and DGGS API Engineering Report" mentions that the storage in the OLAP database is "a couple of times larger than an equivalent compressed and non-lossy raster storage covering the same area, at the same resolution". Doubling the space needed for data storage may be an important parameter given the difficulty DIAS already have in keeping Sentinel 2 data online. Some ClickHouse optimization parameters could help in managing large database (partitioned and distributed on multiple computation nodes [15]).

The storage solution to consider is perhaps a hybrid one. Like the Sentinel-2 tiling based on MGRS, one could imagine a raster tiling based on a DGGS grid. The tiles thus formed would be stored according to the COG (Cloud Optimized Geotiff) format and projected according to a projection local to the cell in order to have the least possible distortion (ISEA and RHEALPix DGGS allow locally to represent the data on a regular square grid). This solution may seem interesting, but it is probably not compatible with the OGC recommendations, nor with the few tools currently available on the market.

Indeed, the DGGS specifications [16] requires the DGGS to have a global datum:

"The structure for the DGGS geometry is provided by a strictly controlled process of recursive tessellation of the parent geometry that creates the DGGS RS's units of geometry. The region occupied by each unit of geometry is called a zone. Each zone is

[11] 11,626,681,248,842 to be exact.

[12] https://www.geopackage.org/

[13] http://www.kealib.org/

[14] https://www.unidata.ucar.edu/software/netcdf/

[15] https://clickhouse.com/docs/en/sql-reference/statements/create/table/#codecs

[16] https://docs.ogc.org/as/20-040r3/20-040r3.html

given a unique name, called a zonal identifier. Each zonal identifier is associated with a representative spatio-temporal position in a base CRS (coordinate reference system) defined by a datum for the DGGS's global world.

Best practice is for a zonal identifier to be an encoding of both its position and its topology."

2.7 DGGS Technical challenges for ARD products

Independently from the strict DGGS selection, some exciting challenges need to be addressed to validate its usability and adoption for ARD products in a multi-missions/multi-sensors context.

2.7.1 Imagery processing new paradigm

Image processing is the biggest challenge by far, beyond technological aspects. Switching from traditional gridded imagery files such as COG to DGGS will have significant impacts regarding current geospatial ecosystem and habits.

Imagery processing is based on arrays and gridded data. DGGS processing requires to change and to work on data without arrays, with unstructured data (non-gridded data).

This change of paradigm will have impacts:

- ✓ On technical aspects: storage, software, processing libraries, etc.
- ✓ On user-side aspects: algorithms development, global (worldwide) approach.

That said some current works are promising. For example, UXarray [17] is a component of the Project Raijin [18] 8 efforts, whose goals are to produce and support Python tools for the analysis and visualization of unstructured grids, primarily those arising from global weather and climate models. DGGS could benefit from such developments.

2.7.2 DGGS ARD Products storage challenge

Whatever the selected DGGS, the technical key points are the same and so are the implementation solutions. Indeed, with DGGS we store only information (band values) attached to DGGS cells identifiers. To summarize, DGGS data should be stored using a column-based storage format, with each row representing a single DGGS cell, to ensure optimal performance.

Regarding the storage, whether it is on database or files, the dataset structure will then look like a super "CSV file", just as simple as that. The question is how to scale up and store all the records for S2 (and other missions) data.

The technical solution for a PoC will have to be scalable and easy to maintain. As a reminder, a single 10 m resolution image represents 10 980 x 10 980 = 120 560 400 pixels (= sparse DGGS records).

[17] https://github.com/UXARRAY/uxarray

[18] https://raijin.ucar.edu

2.7.2.1 Overlapping issue of Sentinel-2 MGRS grid

Before discussing potential solutions, we want to emphasize a major issue with the current system used for Sentinel-2: the MGRS grid and the resulting storage inefficiency caused by overlapping tiles. During our attempts to estimate the cost of a switch to a DGGS for ARD storage, and our experiments over Northern Europe, we found out that the loss of storage is significant. Here, we will explain why this issue is important and why it must be considered when estimating DGGS-based storage costs, as DGGS does not suffer from this problem.

Indeed, "UTM-MGRS's grid-inherent overlaps inflate global land remote sensing datasets by 33%. E.g. yearly Sentinel-2 Level-1C/-2A products volume is increased from 3 to 4 petabytes", according to a study [19] on this side effect.

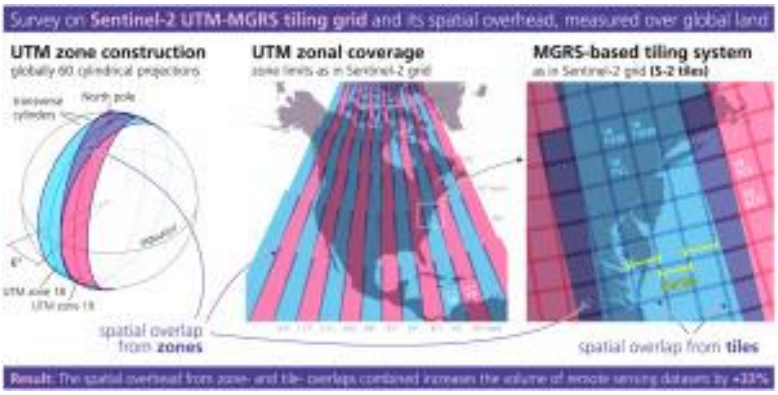


Figure 9: Sentinel-2 UTM tiling grid and its spatial overhead

The S2 MGRS tiling grid has 2 major implications:

- ✓ UTM zones are overlapping themselves when we get closer to the North or South.
- ✓ MGRS tiles within a single UTM zone are also overlapping each other on their edges.

Between two adjacent tiles, we observe pixel redundancy along the edges. At the intersection of four tiles, pixels are duplicated four times, as shown hereafter.

[19] https://doi.org/10.1016/j.isprsjprs.2023.07.015

Figure 10: Sentinel-2 UTM tiling grid tiles overlap

The advantage is that DGGS will completely eliminate duplication caused by overlapping MGRS/Sentinel-2 grid tiles, thereby avoiding the 33% storage loss.

2.7.2.2 Database storage

Storage of DGGS data can be a real challenge. Except the rHEALPix DGGS which allows to store imagery data in a traditional way (raster files such as GeoTiff/COG), other DGGS' data need to be processed as atomic values, attached to cell IDs. This representation of the pixels and bands data is column oriented.

```
"zoneId","resolution","date","B01","B02","B03","B04","B05","B06","B07","B08","B09","B10","B11","B12","B8A","NDVI","NDBI","NDWI"
"86184c00fffffff",6,"2023-05-13 12:03:56",7034,6676,6197,5935,6053,5763,5691,0,0,0,4954,4627,5542,-1,1,1
"86184c01fffffff",6,"2023-05-13 12:03:56",8340,7920,7366,7053,7178,6855,6784,0,0,0,5982,5378,6624,-1,1,1
"86184c047ffffff",6,"2023-05-13 12:03:56",2341,2290,2170,2045,2075,2005,1991,0,0,0,2037,2110,1959,-1,1,1
"86184c04fffffff",6,"2023-05-13 12:03:56",2194,2114,1996,1889,1915,1846,1833,0,0,0,1969,2004,1814,-1,1,1
"86184c057ffffff",6,"2023-05-13 12:03:56",6108,5831,5427,5194,5297,5037,4973,0,0,0,4340,4084,4846,-1,1,1
"86184c05fffffff",6,"2023-05-13 12:03:56",3125,3003,2810,2668,2713,2593,2561,0,0,0,2575,2699,2512,-1,1,1
"86184c067ffffff",6,"2023-05-13 12:03:56",3257,3154,2970,2811,2868,2733,2696,0,0,0,2736,2908,2646,-1,1,1
"86184c06fffffff",6,"2023-05-13 12:03:56",1917,1876,1784,1675,1695,1652,1648,0,0,0,1728,1733,1630,-1,1,1
"86184c077ffffff",6,"2023-05-13 12:03:56",4319,4090,3815,3629,3712,3518,3463,0,0,0,3367,3500,3391,-1,1,1
"86184c0afffffff",6,"2023-05-13 12:03:56",4418,4072,3739,3538,3592,3412,3384,0,0,0,3422,3330,3327,-1,1,1
"86184c0c7ffffff",6,"2023-05-13 12:03:56",5061,4790,4425,4188,4261,4045,3986,0,0,0,3877,3911,3892,-1,1,1
"86184c0cfffffff",6,"2023-05-13 12:03:56",4698,4445,4100,3893,3977,3770,3704,0,0,0,3685,3901,3614,-1,1,1
"86184c0d7ffffff",6,"2023-05-13 12:03:56",4023,3928,3608,3397,3455,3266,3215,0,0,0,3202,3311,3132,-1,1,1
"86184c0dfffffff",6,"2023-05-13 12:03:56",5638,5428,4988,4716,4828,4570,4485,0,0,0,4319,4400,4364,-1,1,1
"86184c0e7ffffff",6,"2023-05-13 12:03:56",6865,6517,6047,5770,5866,5593,5527,0,0,0,4932,4605,5385,-1,1,1
"86184c0efffffff",6,"2023-05-13 12:03:56",4749,4503,4172,3977,4045,3845,3796,0,0,0,3645,3683,3708,-1,1,1
"86184c0f7ffffff",6,"2023-05-13 12:03:56",6366,6064,5613,5347,5440,5177,5124,0,0,0,4605,4242,5000,-1,1,1
"86184c14fffffff",6,"2023-05-13 12:03:56",2164,2014,1904,1794,1817,1762,1756,0,0,0,1837,1906,1737,-1,1,1
"86184c207ffffff",6,"2023-05-13 12:03:56",1516,1484,1410,1319,1324,1298,1304,0,0,0,1308,1297,1294,-1,1,1
"86184c20fffffff",6,"2023-05-13 12:03:56",1380,1355,1285,1216,1210,1168,1171,0,0,0,1191,1188,1170,-1,1,1
"86184c217ffffff",6,"2023-05-13 12:03:56",1801,1744,1628,1503,1506,1460,1455,0,0,0,1479,1489,1432,-1,1,1
"86184c21fffffff",6,"2023-05-13 12:03:56",1389,1355,1278,1171,1165,1143,1148,0,0,0,1121,1111,1132,-1,1,1
"86184c227ffffff",6,"2023-05-13 12:03:56",2625,2558,2397,2265,2307,2209,2190,0,0,0,2329,2468,2172,-1,1,1
"86184c22fffffff",6,"2023-05-13 12:03:56",1423,1399,1328,1241,1240,1214,1220,0,0,0,1209,1203,1215,-1,1,1
"86184c237ffffff",6,"2023-05-13 12:03:56",2984,2886,2698,2553,2602,2482,2447,0,0,0,2561,2732,2404,-1,1,1
"86184c247ffffff",6,"2023-05-13 12:03:56",1641,1745,1820,1624,1711,1859,1938,0,0,0,1795,1651,1970,-1,1,1
"86184c24fffffff",6,"2023-05-13 12:03:56",1644,1771,1904,1629,1681,1758,1820,0,0,0,1750,1643,1837,-1,1,1
"86184c257ffffff",6,"2023-05-13 12:03:56",1420,1418,1350,1230,1232,1216,1225,0,0,0,1200,1192,1220,-1,1,1
"86184c25fffffff",6,"2023-05-13 12:03:56",1474,1495,1431,1263,1261,1240,1250,0,0,0,1223,1215,1242,-1,1,1
"86184c267ffffff",6,"2023-05-13 12:03:56",1467,1567,1734,1628,1901,2551,2780,0,0,0,2292,1874,2913,-1,1,1
"86184c26fffffff",6,"2023-05-13 12:03:56",1541,1641,1757,1540,1590,1651,1709,0,0,0,1602,1509,1717,-1,1,1
```

Figure 11: Sentinel2 data converted to H3 DGGS

OLAP database

ClickHouse

During the phase 2, we made some tests with Clickhouse OLAP database. This solution was challenged to store S2 ARD Products with H3 DGGS. The preliminary results were very promising.

Advantages:

- ✓ Efficient storage performances (compression ratio around 3:1)

- ✓ Very high performances: processing of millions of records in milliseconds
- ✓ Direct Integration of H3 API in ClickHouse (SQL functions)

Points to be evaluated if this storage is selected for the phase 3 demonstrator:

- ✓ The **scalability** of the database through clustering.
- ✓ The use of a **distributed storage** filesystem for the database, to scale up.

Traditional databases

This option seems irrelevant according to our tests because of scalability and maintenance issues. PostGreSQL worked well for our limited use case and use on a few products.

Postgres-XL [20] would have been an interesting option to test massive DGGS storage. This massively scalable version of PostGreSQL has been used for European Space Agency "Gaia Variability Studies", but the project seems stopped by now.

2.7.2.3 File-based storage

Apache Parquet format

Apache Parquet was in the late part of the study considered as an alternative solution for storage. To date this format still have to be evaluated, to check its viability in real-use case. We focused on simple Parquet files and not GeoParquet [21], because storing cell geometry is too expensive in terms of storage. Indeed, geometry can be calculated from the cell identifier.

Advantages:

- ✓ Granularity could be similar to current Sentinel missions' storage (files structure),
- ✓ Scalability through S3 object storage (no server or database required except maybe for catalog),
- ✓ Cloud-native format (like COG) and direct query is possible (DuckDB),
- ✓ Efficient file compression.

Points to be evaluated if this storage is selected for the Task 3 PoC:

- ✓ Performances? (for direct access, visualization, processing)
- ✓ Storage evaluation impact
- ✓ Tools and libraries:
  - imports/exports? (to and from Geotiff/COG)
  - Data visualization toolkit?
  - Compatibility with existing software stacks? Most of imagery processing relies on arrays (e.g. XArray)

[20] https://www.postgres-xl.org/

[21] https://geoparquet.org/

2.7.2.4 A quick storage comparison

Some additional tests have been made, to compare alternative options to store the ARD data. The following graph shows the disk storage necessary for 7 Sentinel-2 Level-2A tiles (all bands in 60 m) covering Belgium.

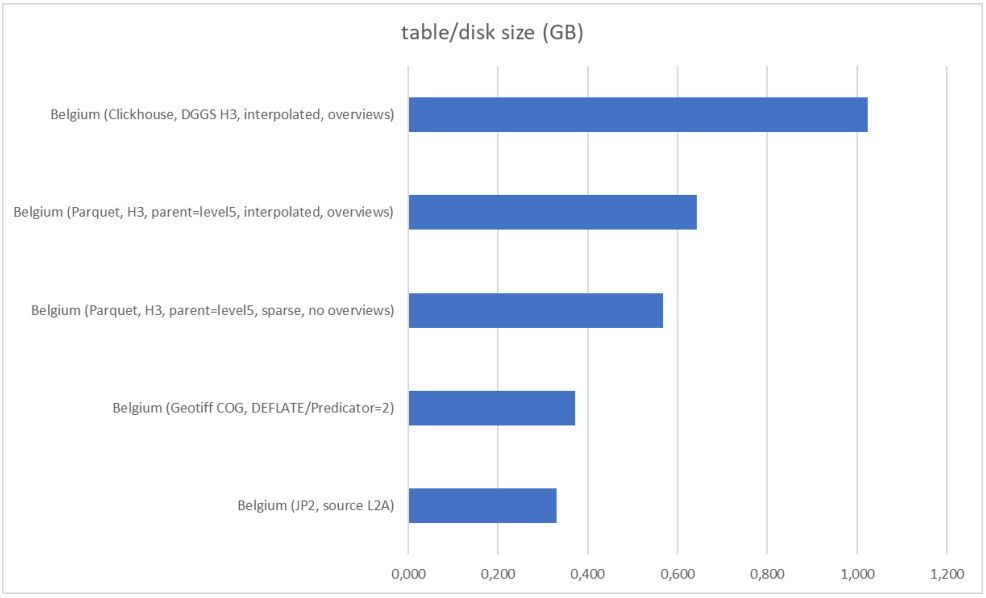


Figure 12: Storage comparison for S2 ARD data

This storage evaluation is not perfect though, because the S2 tiles have been converted AS-IS and overlapping between S2 tiles have not been considered. This has an impact on the number of cells either in ClickHouse database or in Parquet files.

To illustrate this issue, here are below the "level 5" H3 cells covering Belgium at the same date, generated from 7 S2 tiles. We can see that the overlapping might have a significant impact on the estimated storage.

In phase 3, a filtering will be necessary to remove overlapped data. As seen before, S2 tiling system (MGRS) generates around 33% of duplicated information and we obtain the same overhead if we do not filter duplicated information (see below).

Figure 13: Duplicated information in DGGS dataset caused by current S2 tiling system

To provide more reliable insights, the following table and graph independently display the figures for the seven Sentinel-2 tiles covering Belgium, as shown above. Parquet files are only containing sparse non-interpolated data (=pixels) at level 11, H3 DGGS.

The idea here is to compare the same number of pixels from raster imagery, converted into sparse DGGS cells (1 input pixel = 1 output DGGS cell) Indeed, conversion of raster data pixels to a higher resolution DGGS may create "holes" in the DGGS grid. The most convenient solution today is to interpolate DGG cells, to fill the gaps.

As of today, we lack software to interpolate DGGS cells data on the fly (on server side or client side), the following table gives an overview of what could be the impact on storage, if we decide later to store only sparse DGGS data into Parquet files. When we have the possibility of handling "sparse" cells, this solution would be cost effective, and scalability would be assured by the object storage itself.

Table 3: COG vs Parquet (H3) storage requirements and overhead

S2 dataset	COG file size in MB (overviews)	Parquet file size in MB (sparse, no overviews)	Overhead
T31UDS_20230604T165401_60m_UInt16	44	66	50,00%
T31UER_20230604T165401_60m_UInt16	56	78	39,29%
T31UES_20230604T165401_60m_UInt16	56	79	41,07%
T31UFR_20230601T215503_60m_UInt16	54	80	48,15%
T31UFS_20230601T215503_60m_UInt16	56	80	42,86%
T31UGR_20230601T215503_60m_UInt16	54	81	50,00%
T31UGS_20230601T215503_60m_UInt16	55	82	49,09%

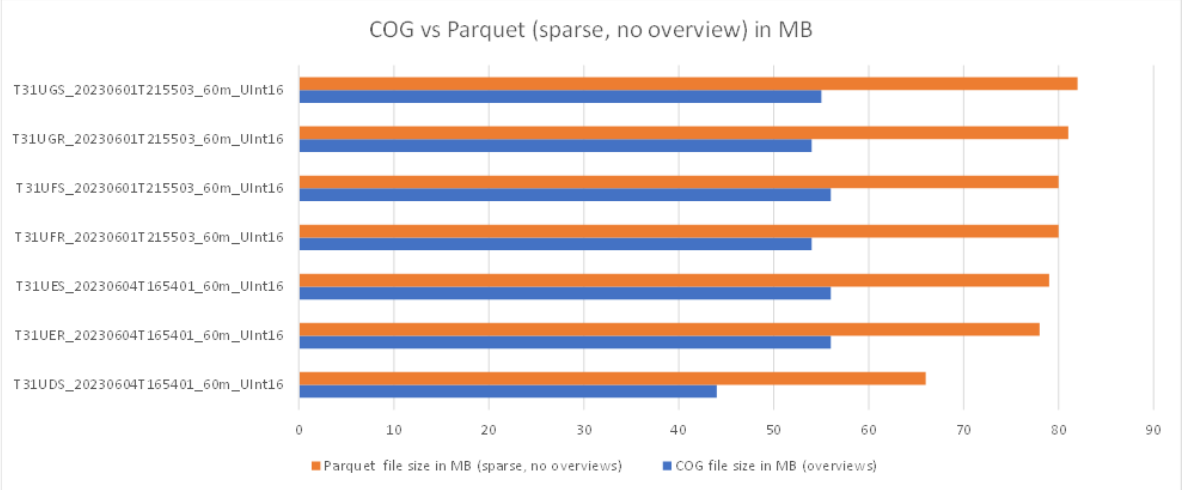
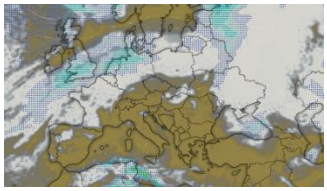


Figure 14: COG vs Parquet (H3) storage requirements and overhead comparison

We notice that the size increase for raw pixel data in Parquet format is between +39% and +50% compared to COG files, when the files are taken independently, to avoid COG overlapping issues. This figure should be considered in light of the fact that in a Discrete Global Grid, no pixel data is duplicated.

These Parquet files are storing ARD data in "sparse" form: data are not interpolated if there are "holes" (no data cells) while we align 60 m resolution pixels to the discrete global grid (at level 11 H3 grid).

Adding overviews would have implied to add +15% overhead using H3 DGGS, which is not that much. Indeed, this can be explained because of the Aperture (7) of H3 and the rapid decrease in each level cells number: 1 parent for 7 children. In traditional raster world with aperture 4, overviews represent up to +50% of the COG file.



The capability of Discrete Global Grids to address "sparse" data can be challenging for users and data scientists but it could also help to reduce the storage footprint in an Earth Observation context, by ignoring areas covered by clouds. Indeed, it is widely admitted that clouds can represent up to half of the pixels data acquired by satellites.

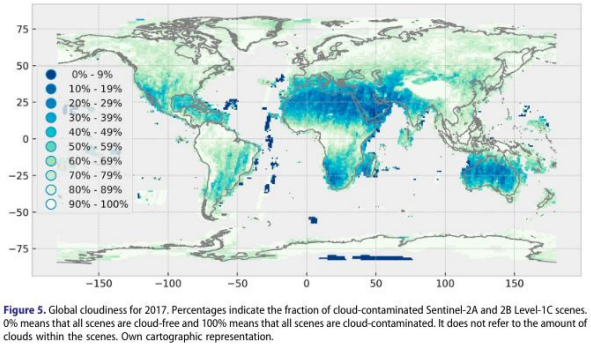


Figure 5. Global cloudiness for 2017. Percentages indicate the fraction of cloud-contaminated Sentinel-2A and 2B Level-1C scenes. 0% means that all scenes are cloud-free and 100% means that all scenes are cloud-contaminated. It does not refer to the amount of clouds within the scenes. Own cartographic representation.

Figure 15: Average cloud coverage in Sentinel-2 image (in 2017)

2.8 Conclusion on the DGGS selection criteria

The following table summarises the main features of the candidate DGGS for EO data. Overall ranking values are expressed in four classes: Poor, Limited, Good, Excellent.

Table 4: Summary of selection criteria by DGGS

	H3	rHEALPix	ISEA7H	ISEA4T	ISEA4D
Cell	Hexagon	Multiple	Hexagon	Triangle	Diamond
Aperture	7	9	7	4	4
Projection	Gnomonic	Custom (EA)	ISEA	ISEA	ISEA

	H3	rHEALPix	ISEA7H	ISEA4T	ISEA4D
Shape/Area preservation (Kmoch et al., 2022)	Very good shape preservation High area distortion	Good shape preservation by shape group Very low area distortion	Very good shape preservation Very low area distortion	Low shape preservation Very low area distortion	Low shape preservation Very low area distortion
Indexing strategy	Dual indexing: Axis and hierarchical	Z space filling curves	Sequential id	Sequential id	Sequential id
Children extend fully contained in Parent	No	Yes	No	Yes	Yes
Orientation	Dymaxion	custom	customizable	customizable	customizable
Software support	Excellent (Multi-language support, DBs support, Cloud extensions, …)	Limited (implemented in the Proj.4 Cartographic Library)	Good (DGGRID + bindings)	Good (DGGRID + bindings)	Good (DGGRID + bindings)

ISEA7H and rHEALPix are great DGGS candidates for handling Sentinel-2 ARD data. ISEA7H brings the benefits of an equal area projection, high compactness, and great neighbors' relation. Where rHEALPix is more suitable for raster processing especially in the mid latitudes (square faces) and provide some unique features like the distribution of cell nuclei along iso-latitude rings that can facilitate latitudinal data analysis at multiple resolutions (Bowater and Stefanakis, 2019).

However, if we consider the maturity of the software tools, H3 is one of the most used DGGS and by far the most supported. A good support and user experience are key to the adoption of DGGS. This makes H3 the easiest go-to grid for many studies and applications demonstration, despite the showed limitations.

3. DGGS review for Sentinel-2 data

3.1 Uber H3

H3 is first an indexing system developed by Uber for its purpose. Uber created this indexing system to not only help with its location and positioning of rides but also for handling a dynamic market, with the H3 system handling dynamic pricing and other marketplace dynamic data that deal with supply and demand. It, therefore, also integrates a variety of spatial data together.

Unlike ISEA, H3 uses gnomonic projection which sacrifices accuracy for higher computing speed.

3.1.1 Sentinel-2 on H3

Sentinel-2 bands	Central wavelength (µm)	Resolution (m)
Band 1 – Coastal aerosol	0.443	60
Band 2 – Blue	0.490	10
Band 3 – Green	0.560	10
Band 4 – Red	0.665	10
Band 5 – Vegetation red edge	0.705	20
Band 6 – Vegetation red edge	0.740	20
Band 7 – Vegetation red edge	0.783	20
Band 8 – NIR	0.842	10
Band 8A – Vegetation red edge	0.865	20
Band 9 – Water vapour	0.945	60
Band 10 – SWIR – Cirrius	1.375	60
Band 11 – SWIR	1.610	20
Band 12 – SWIR	2.190	20

Res	Average Hexagon Area (m²)	Pentagon Area* (m²)
0	4,357,449,416,078.392	2,562,182,162,955.496
1	609,788,441,794.134	328,434,586,246.469
2	86,801,780,398.997	44,930,898,497.879
3	12,393,434,655.088	6,315,472,267.516
4	1,770,347,654.491	896,582,383.141
5	252,903,858.182	127,785,583.023
6	36,129,062.164	18,238,749.548
7	5,161,293.360	2,604,669.397
8	737,327.598	372,048.038
9	105,332.513	53,147.195
10	15,047.502	7,592.318
11	2,149.643	1,084.609
12	307.092	154.944
13	43.870	22.135
14	6.267	3.162
15	0.895	0.452

Figure 16: Sentinel-2 band resolutions and H3 area per resolution.

To store Sentinel-2 bands we could use resolution 13 for uber H3 that guarantees to keep most of the information from 10m Sentinel-2 bands. But using resolution 12 for 20m bands and resolution 8 for 60m bands could help reduce the storage/compute needed.

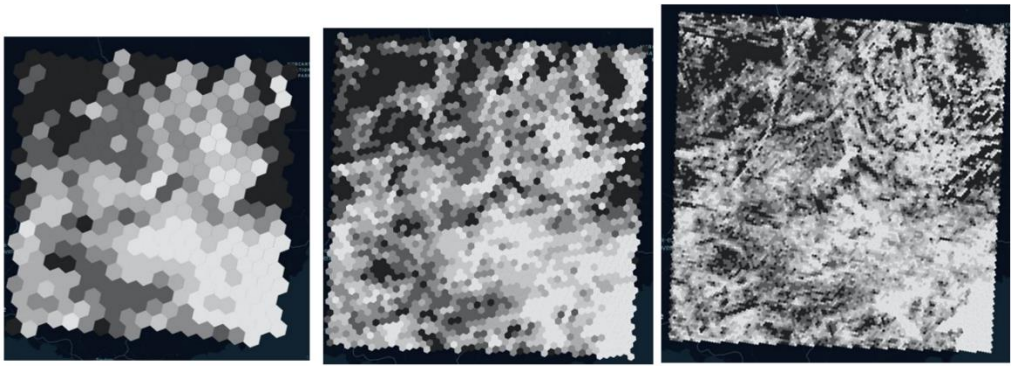


Figure 17: Sentinel-2 Band 2 on H3 at resolutions 6, 7 and 8

Also, to assess the size of a Sentinel-2 10m band at different H3 resolution, the H3 data were inserted in a PostgreSQL/PostGIS database using a simple model using only the H3 index of the cell, the band value, with and without the cell geometry. The different tables were generated for resolution from 10 to 12. The table below provides an overview of the table statistics:

	Nb rows	Table size	Index size	Insertion time
H3_10_nogeo	843,330	35.62 MB	18.09 MB	~30 sec
H3_11_nogeo	5,893,785	248.9 MB	126.27 MB	~2 min
H3_12_nogeo	41,231,339	1.7 GB	883.21 MB	~10 min
H3_10_geo	843,330	164.72 MB	18.09 MB	~30 sec

The table storing the resolution 10 with the cell geometry is about five times heavier than the one without the geometry. The H3 cell geometry can obviously be reconstructed from its index and was added to the table only for size comparison.

Knowing the native S2 band stored as raster was only ~223MB, the storage in a database of large number S2 product, with an H3 resolution close to the native resolution is not sustainable. The H3 resolution 13, which is the most suitable for the 10m band resolution, could not be handle properly.

Also, it is important to mention that to obtain the band values stored for each resolution, an average of the value of each pixel contained in the cell was applied. If this is sufficient when the H3 resolution is lower than the original pixel resolution, for higher H3 resolutions, a proper resampling at cell centre coordinates is essential to preserve the accuracy of the data.

Finally, we also assessed the performance for retrieving data from the database. We compared the time between querying spatial data entries using standard PostGIS polygon intersect method (on table with geographic information) and the H3 approach based on parent H3 index (on table with only the H3 index information. The first query was executed in about 1.3 sec while the second one only took 0.5 sec, thus representing more than a factor 2 gain.

Indeed, H3 proposes functions to get the cells overlapping a given geometry (like polygonToCell, polygonToCellAddresses). Resulting cells ids/addresses can then be used for lists comparisons, which are indeed faster than geographic intersections.

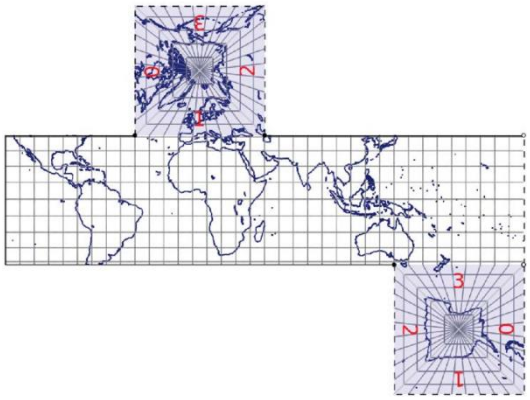
3.2 rHEALPix

The rHEALPix DGGS (acronym for Hierarchical Equal Area iso-Latitude Pixelization) is a pixelization produced by a subdivision of a spherical surface in which each “pixel” covers the same surface area as every other “pixel”. Internally, it can be thought of as a mapping of an ellipsoid onto a cube, followed by a symmetric hierarchical partitioning of the polyhedral faces.

This DGGS can be customized using diverse options, like the capability to shift the prime meridian. The default RHEALPix based DGGS is called AusPix and is specially configured to provide easy to use (only square) system in Australia. Shifting the prime meridian to 50◦ east longitude moves all cube's corners and edges into water, Arctic polar triangle boundary cuts to less densely populated regions, and rearranged polar triangles to optimize Europe and New Zealand views. Such a rotation was used in the OGC testbed-16, and the resulting reference system called TB16Pix.

The rHEALPix mathematics are implemented in the PROJ.4 cartographic library, allowing the use of standard raster manipulation tools and provides a fundamental building block for satisfying the proposed OGC DGGS Standard's requirement.

3.2.1 Sentinel-2 on rHEALPix



Resolution	Cell area (m^2)	Nominal cell edge	Number of cells
0	85.11E+12	9,225 km	6
1	9.456E+12	3,075 km	54
2	1.051E+12	1,025 km	486
..			
11	2,712	54 m	188.3E+09
12	301	18 m	1,694.6E+09
13	33.5	6 m	15,251E+09
14	3.72	2 m	137.26E+12

Figure 18: rHEALPix cells per resolution

In order to store Sentinel-2 data we could use rHEALPix resolutions 13, 12 and 11 that guarantee to keep all information from respectively Sentinel-2 10 m, 20 m and 60 m bands. But it is important to notice that under the mid Europe latitudes, the rHEALPix grid can lead to, at least, double the number of cells along the longitude axe.

rHEALPix projection is natively supported by PROJ4 library, thus allowing to continue working with geotiffs.

The conversion can be performed using a simple command line:

gdalwarp -r bilinear -t_srs '+proj=rHEALPix +ellps=WGS84' B02.tif B02_rHEALPix.tif

Therefore, when using gdal warp to convert a S2 10 m bands, originally composed of 10980*10980 pixels, it results into a 13839*33571 grid at resolution 13. If we compare the GeoTIFFs files, the original band (compressed) is 226MB and the rHEALPix one (also compressed) is 547MB.

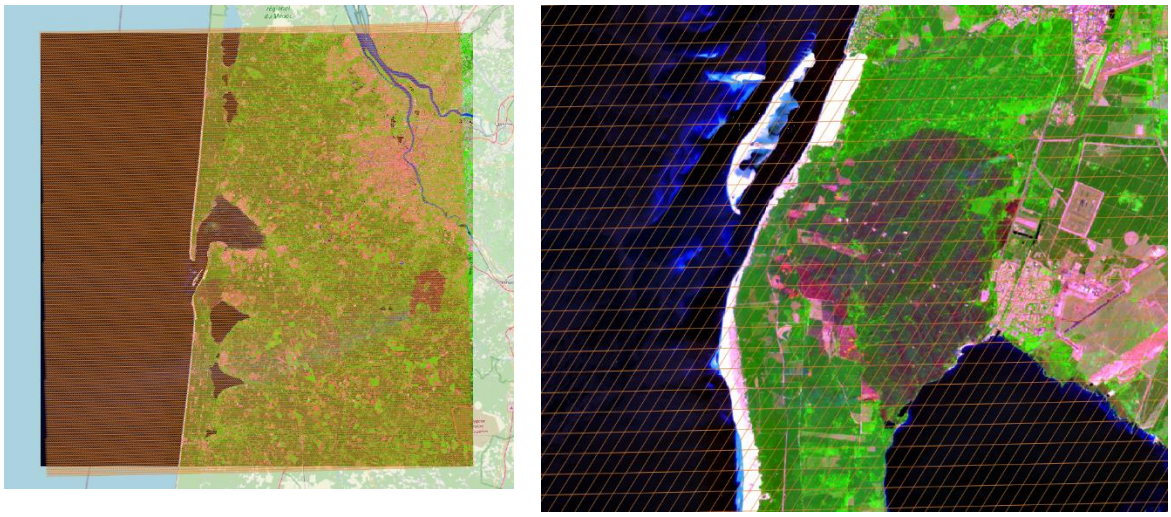


Figure 19: Sentinel-2 on rHEALPix resolution 9

If we consider storing a rHEALPix grid into a database, the size is proportional to the number of cells in the grid and can be compared to the H3 grid size. However, because rHEALPix requires more cells to preserve the S2 information than H3, the database will require more storage space.

The resulting COG rHEALPix raster remains larger than the original raster, because at the data location and to maintain the information it requires more pixels than the native square pixels. Under our latitudes it almost doubles the size of the raster after application of the COG formatting.

B02.tif	225,6 MB
B02_rhealpix.tif	929,3 MB
B02_rhealpix_cog.tif	547,1 MB

Figure 20: Sentinel-2 on **rHEALPix**, almost double the rows resolution

3.3 ISEA-based grids

These grid systems are constructed upon the Snyder's polyhedral globe, which is a truncated icosahedron with 32 same area faces (20 hexagons and 12 pentagons centered on the 12 vertices of the icosahedron). It uses the Snyder equal-area projection, which is based on a modified Lambert azimuthal equal-area projection adapted to this polyhedral globe.

On the application to the truncated icosahedron, the angular deformation does not exceed 3.75°, and the scale variation is less than 3.3 percent. These advantages are at the expense of increased interruptions at the polygon edges when the polyhedral globe is unfolded.

Different grids are based on this projection:

- ✓ **ISEA4T** extends the globe and projection to use a hierarchical equilateral triangle grid with aperture 4.
- ✓ ISEA3H uses a hexagonal grid of aperture 3 with offset coordinate cell indexing system.
- ✓ Other combinations of aperture and cell topology are also available: ISEA4H, ISEA7H, ISEA43H, ISEA4D.

Cell networks are available at increasingly finer resolutions to allow locations to be represented by cells up to the order of one square centimeter in area. For non-exact approximations (to equal area) it can be replaced by Gnomonic projection, as in H3 Uber.

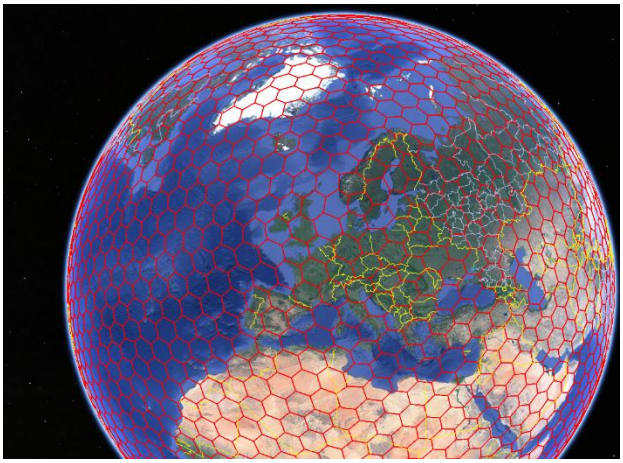


Figure 21: ISEA7H3 (Hexagon based aperture 7 resolution 3) generated using DGGRID

4. DGGS use cases

A Discrete Global Grid System could enable an optimized approach for storing and querying Sentinel-2-like ARD products and any other missions' products.

DGGS could help:

- ✓ to define a unique standard for ARD products storage (DGGS-based datacube)
- ✓ to propose a homogeneous way to access data all over the world, in a multi-mission / multi-resolution / multi-temporal context.
- ✓ to enable ARD product indexation. Indexation within product based on DGGS cells.

4.1 Data integration

One of the key application for DGGS is enabling multi-source geospatial analysis. Using a DGGS like H3 we combine data from multiple sensors using the same grid. Using the same python code, we can populate the H3 grid with either Sentinel-2 data or Copernicus DEM for example (Figure 22). Nevertheless, the resolution of the different data products must be aligned with the DGGS grid resolutions. One couple DGGS/resolution could fit one raster product but not the other: indeed, the chosen DGGS level must have at least equal or higher resolution than the product resolution:

- ✓ If the DGGS level resolution is lower than the product, then a data loss will occur (down sampling).
- ✓ If the DGGS level resolution is way higher, then an oversampling will occur, which can lead to a storage increase (depending on the DGGS aperture) if the data is interpolated to fill the gaps in the DGGS grid.

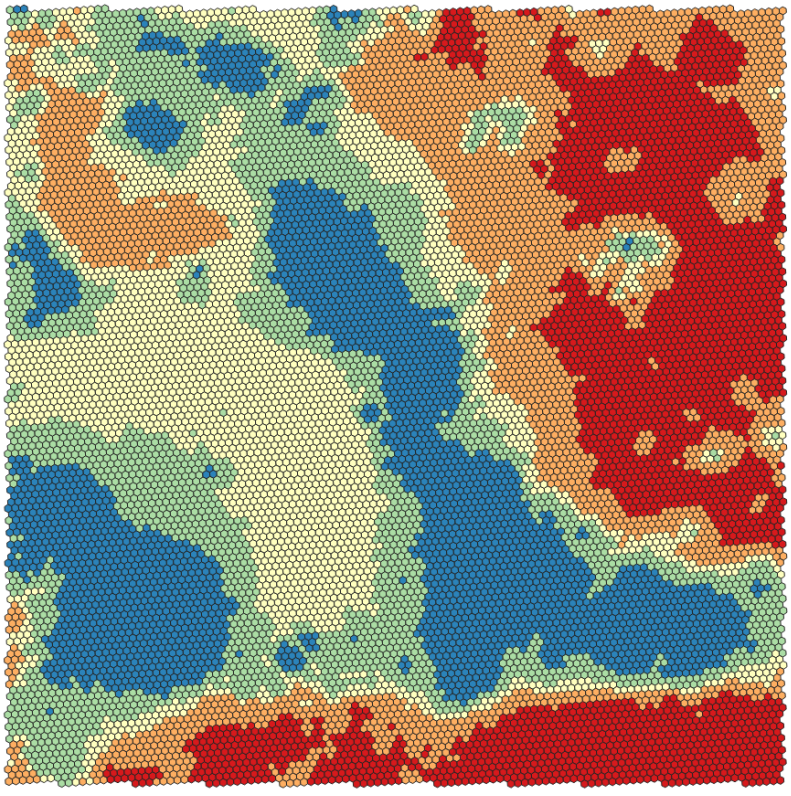


Figure 22: Copernicus DEM classes represented using H3 grid

It is also possible to add Sentinel-3 data onto the H3 grid, and an H3 resolution of 7 or 8 is sufficient to have a conservative approach. If we consider the footprint of an S2 tile, the S3 data (SL_2_LST layer for example) on the H3 grid represent less than 1MB.

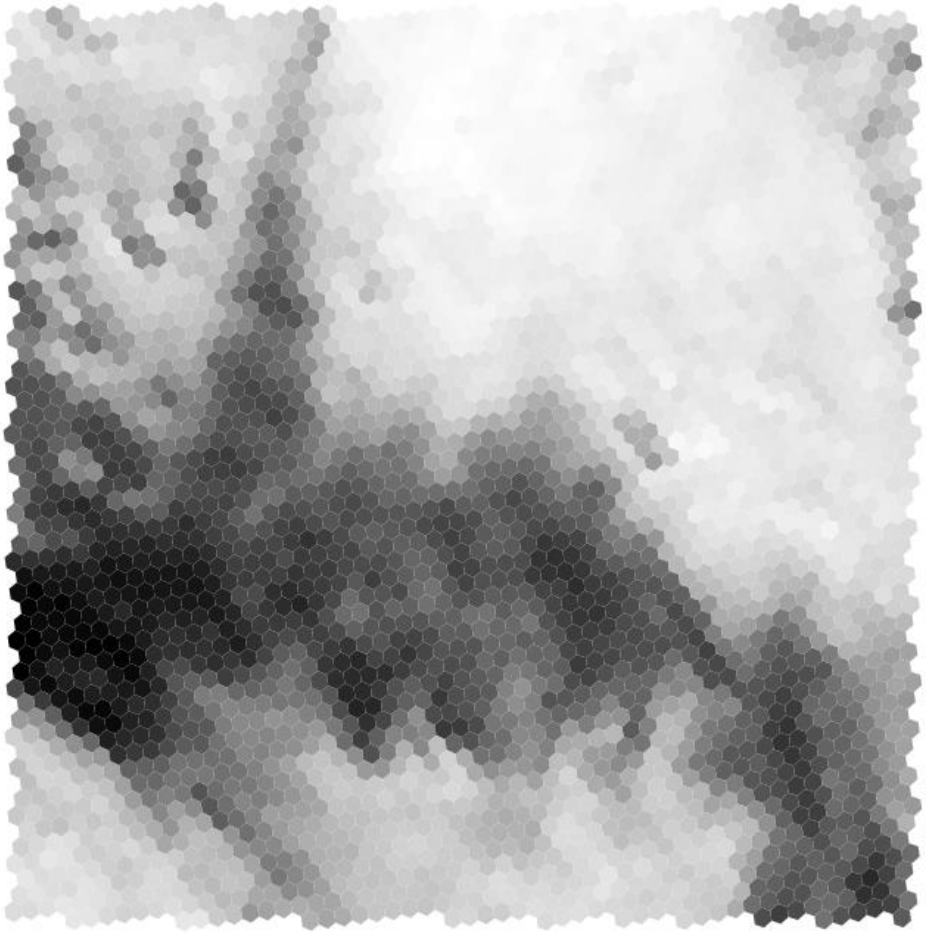


Figure 23: S3 data (SL_2_LST) represented on H3 grid (res 7)

4.2 ARD products indexing

DGGS provides a great way to index imagery product or any geospatial data. During this study we found that using H3 for fast Sentinel-2 products indexing can be an efficient way of querying satellite imagery based on cell level cloud cover. This will result in a faster more precise Sentinel-2 products search. What has been experimented with H3 would be also applicable for other DGGS but for indexing, the fact that H3 neighbors' cells centers are equidistant is an advantage.

Indexing consists in covering with a binary mask DGGS cells, which resolution level has to be selected according to the target application. No need to reach the native resolution to index cloud masks.

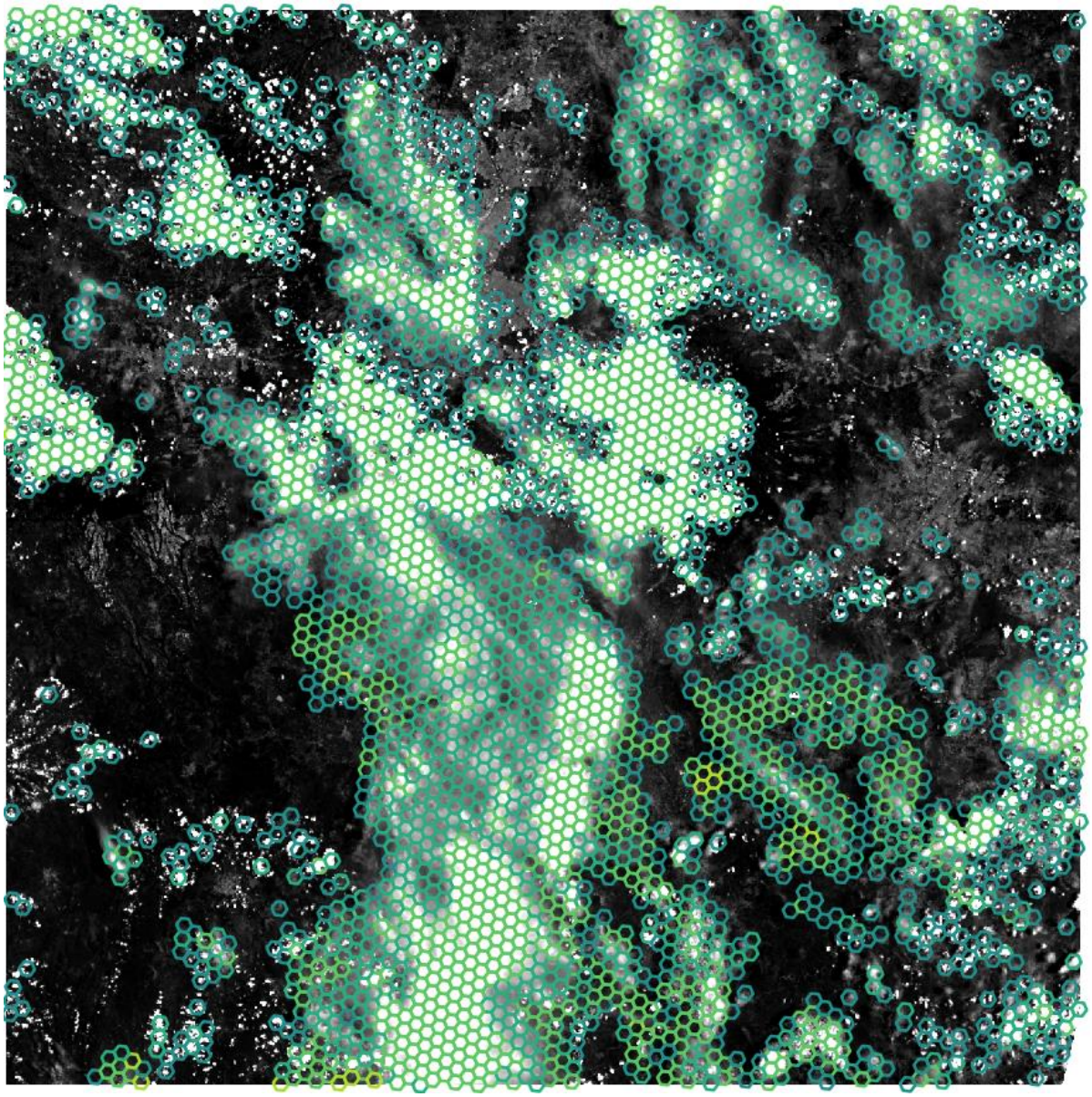


Figure 24: Sentinel-2 cloud filtering using the SCL layer (on H3)

It is possible to use H3 properties to simplify the geometries and gather binary information at a lower resolution.

Applied to the cloud mask, it then becomes possible to precisely filter clear sky data at a specific location.

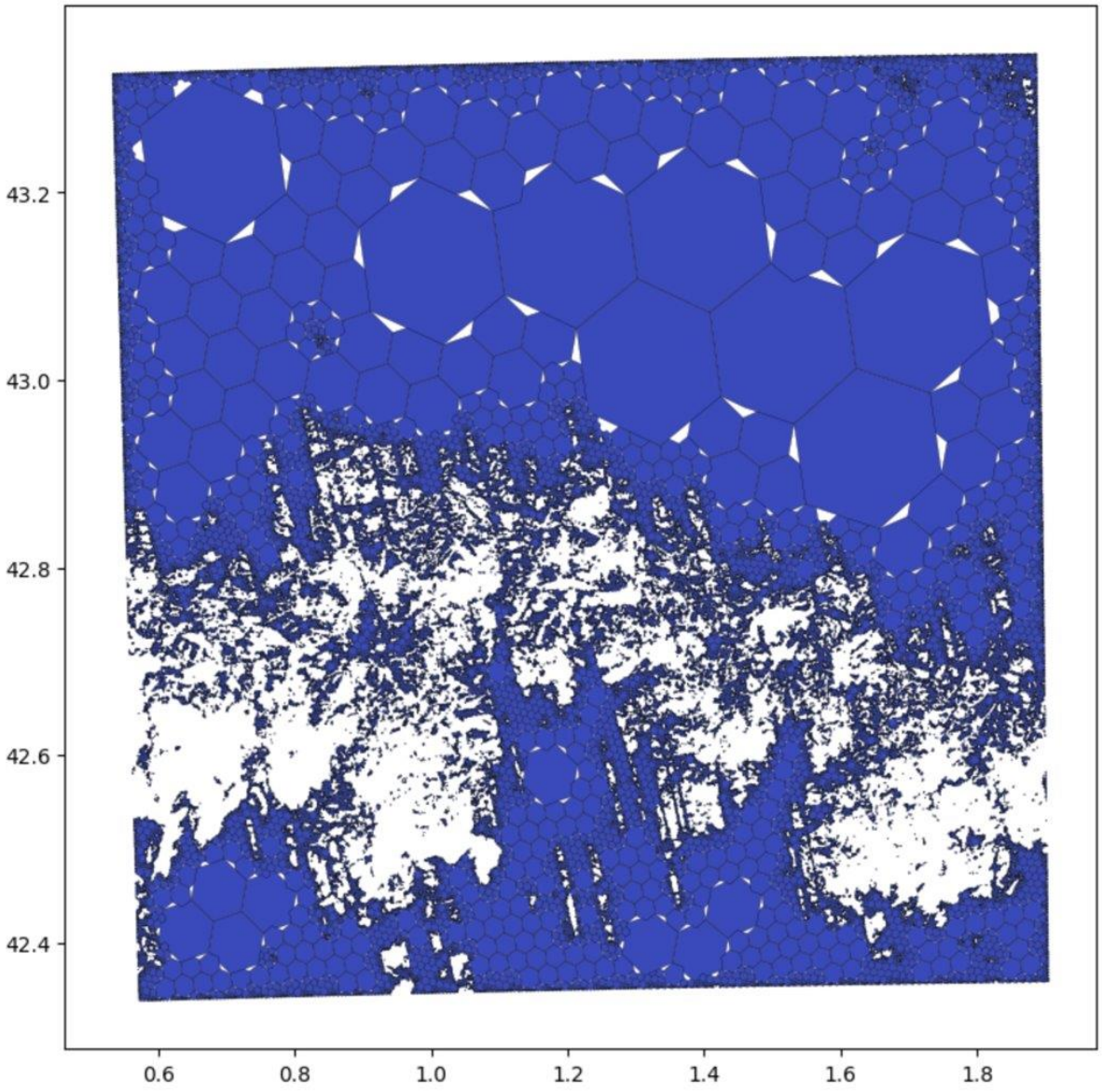


Figure 25: Compacted cloud mask (on H3)

Other masks could also be included in the data model: snow mask, quality mask, etc.

5. Lessons learned after Tasks 1 and 2

This study focused on Sentinel-2 products, but its conclusions are applicable to other EO products, such as Sentinel-1, Sentinel-3, and Landsat 8. Further theoretical studies could help refine the DGGS choice or adjust its properties to focus on specific usage needs.

As mentioned earlier, a perfect DGGS does not exist. The choice of a specific DGGS to handle different missions and sensor characteristics is always a matter of compromise. The choice of a DGGS depends on various properties that are strongly interrelated. Some parameters can be adjusted to re-center the grid on a specific location (ex: rHEALPix centered over Canada) but this usually breaks the universal nature of Discrete Global Grid systems. Also, there are no infinite combinations of parameters like geometry, aperture, resolution, etc. and most possible way to create DGG systems have already been explored:

- ✓ **Geometry:** cell geometry, which is important regarding the DGGS compactness, dictates the possible resolution levels. More compact geometry causes bigger resolution gaps. We remind here that products' native resolution must adapt to the closest DGGS resolution, not the other way around.
- ✓ **Equal Area cells:** for a data cube implementation for managing environmental data and spatial statistics, having equal area cells is required.
- ✓ **Compactness**: hexagonal DGGSs have undeniable advantages, including clear neighborhood relationships and improved visual data representation.

That said the selected DGGS have all some caveats:

- ✓ rHEALPix has 3 different types of cells: squares, diamonds, and triangles and not very convenient at northern and southern latitudes. Nevertheless, it allows to keep on working with traditional imagery formats and tools.
- ✓ H3 is interesting for indexation and statistics, provides very low deformation but cells are not equal area and parent cell does not perfectly overlap children cells.
- ✓ ISEA4T using triangles cells has interesting properties but is the less suitable for representing optical effects. Triangle cell shape also implies a complex neighborhood.
- ✓ ISEA7H, seems a good candidate regarding shape and area preservation but has an arguable parent/children link between resolution and lacks software support.

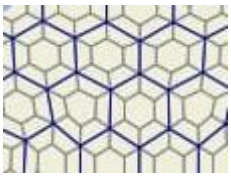


Figure 26: Parent-Children relationships in ISEA7H

During the study we have tested several DGGS grids and identified the pros and cons of the major implementations, as well as some of the available tools and libraries.

This study has highlighted two different issues. The first one is obviously the selection of the best DGGS for S2 ARD products, and the second is how to implement such DGGS.

From our experiments, we conclude that these two problematics can be treated independently. Development could start with a widely supported DGG System and later move to another implementation.

Independently from the DGGS choice, here are the main upcoming technical challenges to tackle, to pretend replacing the completely the native product format:

- ✓ **Storage** optimization: challenges regarding scalability, storage resources utilization, and cost-effectiveness. This point is critical for high resolution levels, which require good compression.
- ✓ **Import/export capabilities and conversions** (time consuming processing, etc.) Dissemination of DGGS projected data requires an export storage format, but only few tools are currently available.
- ✓ **Compatibility** with existing software stacks and **processing libraries**
  - Processing requires arrays. Earth observation processing libraries usually work on two-dimensional arrays (square cells), whereas DGGS returns lists of indexes, for which neighbouring work is not trivial (call an API to find neighbours) and "pixels" are not square.
- ✓ **Integration** in existing processing chains
- ✓ **DGGS Indexing** capabilities are interesting and very efficient, it could be a distinct proposition of application by itself.
  - Only light information would be stored, such as **simplified quality masks or low-resolution overviews** (either the bands data, or computed indexes such as NDVI...)
  - The link with the full resolution raster would be maintained (i.e. the database can be used as **product catalogue**).

6. Final considerations

The first task of this study allowed us to compare multiple DGGS for Sentinel-2 ARD handling. One of the main outcomes of this part of the study was that the user experience is very important. This was confirmed by our research and feedback from DGGS users like the ESA SMOS Cesbio team, because of the lack of ergonomic tools to handle SMOS data.

DGGS is a great integration framework with promising functionalities. Nonetheless, unlocking these features and driving adoption will need an investment in new and existing tools focused on the user experience.

Considering the criteria mentioned in this report, ISAE7H, ISEA4T and rHEALPix are interesting candidates for ARD data as the three implementations guarantee equal-area cells (Table 4). Using an equal area projection for most applications is a very important aspect in conserving accuracy in statistical analysis over a large area of interest. We hope that this MVP will generate more interest in DGGS and help develop tooling for ISEA based grids.

In the context of using DGGS in a production context, developing software ecosystem around a specific DGGS (ISEA4T? ISEA7H?), is required to migrate ARD products: to reduce data size, to preserve the information at high resolution and to focus on file exchange format definition.

Nonetheless to date, Uber H3 is by far the most convenient grid because of the great community and software support. Thus, adopting H3 for the rest of the study (Tasks 3 and 4) will allow the preparation of a dedicated PoC for Sentinel-2 and other missions/sensors like S3, Landsat 8, or Sen2Like products (L2H).

It is important to stress that the selection of H3 for next phases of this project is motivated by the fact that the envelope of such study cannot consider poor software supported DGGS. H3 is anyhow deemed able to fully demonstrate the capabilities and benefits of a DGGS applied to Earth Observation data.

As a DGGS selection and technical implementation are independent, the first could be replaced later by another DGGS, to take advantage of new libraries and developments.

7. References

Bowater, D., Stefanakis, E., 2019. On the isolatitude property of the rHEALPix Discrete Global Grid System. Big Earth Data 3, 362–377. https://doi.org/10.1080/20964471.2019.1658494

Kmoch, A., Vasilyev, I., Virro, H., Uuemaa, E., 2022. Area and shape distortions in open-source discrete global grid systems. Big Earth Data 1–20. https://doi.org/10.1080/20964471.2022.2094926

Purss, M., Gibb, R., Samavati, F., Peterson, P., Rogers, J., Ben, J., Dow, C., 2017. Topic 21: Discrete Global Grid Systems Abstract Specification. Open Geospatial Consortium: Wayland, MA, USA.

Rawson, A., Sabeur, Z., Brito, M., 2022. Intelligent geospatial maritime risk analytics using the Discrete Global Grid System. Big Earth Data 6, 294–322. https://doi.org/10.1080/20964471.2021.1965370

Sahr, K., White, D., Kimerling, A.J., 2003. Geodesic Discrete Global Grid Systems. Cartography and Geographic Information Science 30, 121–134. https://doi.org/10.1559/152304003100011090

Snyder, J.P., 1992. An Equal-Area Map Projection For Polyhedral Globes. Cartographica 29, 10–21. https://doi.org/10.3138/27H7-8K88-4882-1752

Purss, M. B. J., Peterson, P. R., Strobl, P., Dow, C., Sabeur, Z. A., Gibb, R. G., & Ben, J. (2019, March). Datacubes: A Discrete Global Grid Systems perspective. Cartographica, 54(1), 63–71. https://doi.org/10.3138/cart.54.1.2018-0017

8. Appendix A

To support the previous analysis, multiple type of DGG systems have been generated and compared. This was achieved by implementing a python DGGS toolbox based on existing reference tools.

The toolbox can be found on GitHub here (access must be granted by CS on demand): https://github.com/CS-SI/dggs_tbx