Utilizing a DGGS for point cloud integration

..... *A journey into a world without map projections*

Neeraj Sirdeshmukh

Edward Verbree
Peter Van Oosterom

Stella Psomadaki
Martin Kodde

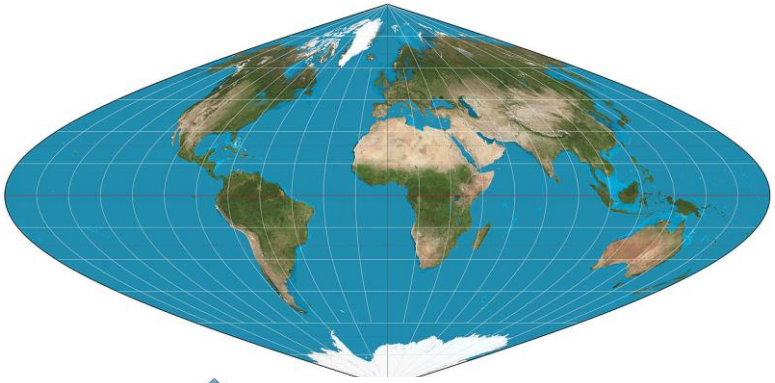
TU Delft

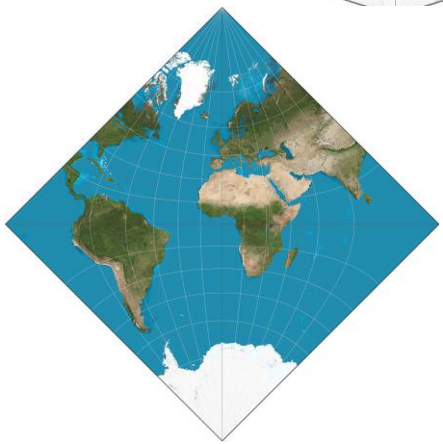
FUGRO

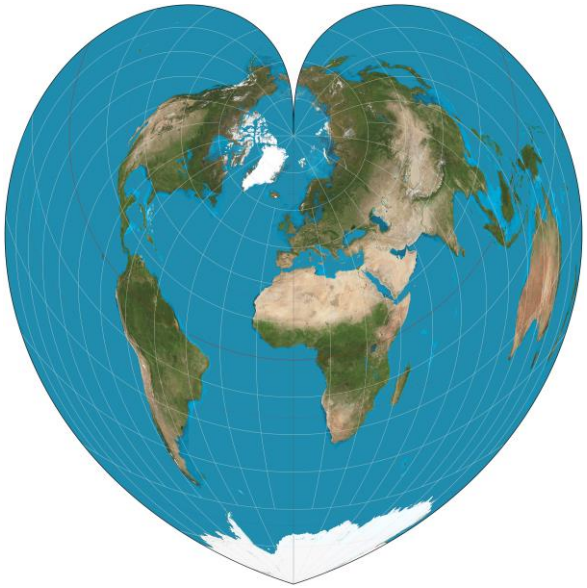
Challenges with map projections

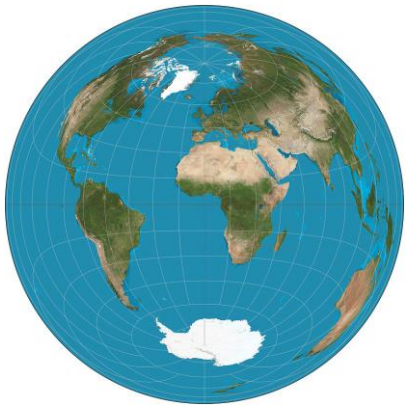
- Cannot preserve all of *angles, area, distance, or direction* at same time
- Spatial error *increases* with repeated projections
- Distortion
  - increases at small scales (whole Earth)
  - increases away from central meridian

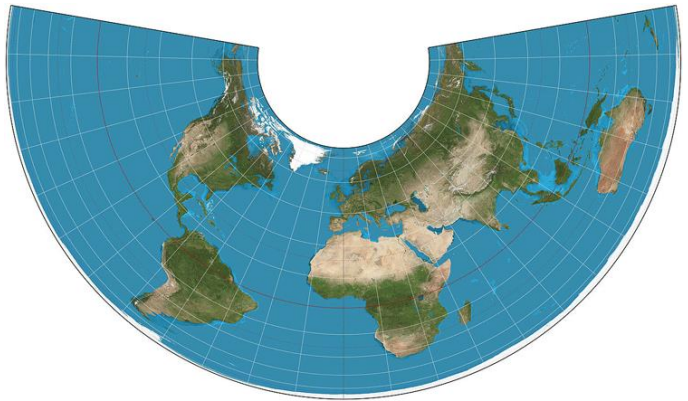
  (*very complex and variable pattern* !)
- ‘tuned’ to certain areas
- In theory, *infinitely* many projections possible !
  - Lack of common standard for geospatial data integration/fusion











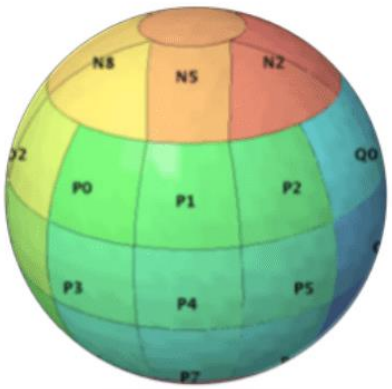
Problem statement

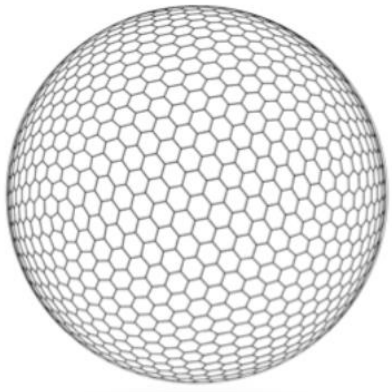
- Need to integrate point clouds originating from different **location**s (countries), each having different coordinate systems, into a **common reference system**
- Need to integrate point clouds with varying initial densities/LOD's to achieve **appealing visualization** of point cloud data when viewed from any **scale**
- " How has it changed ? " – how point clouds of the same area taken at different **times** can be analyzed for changes

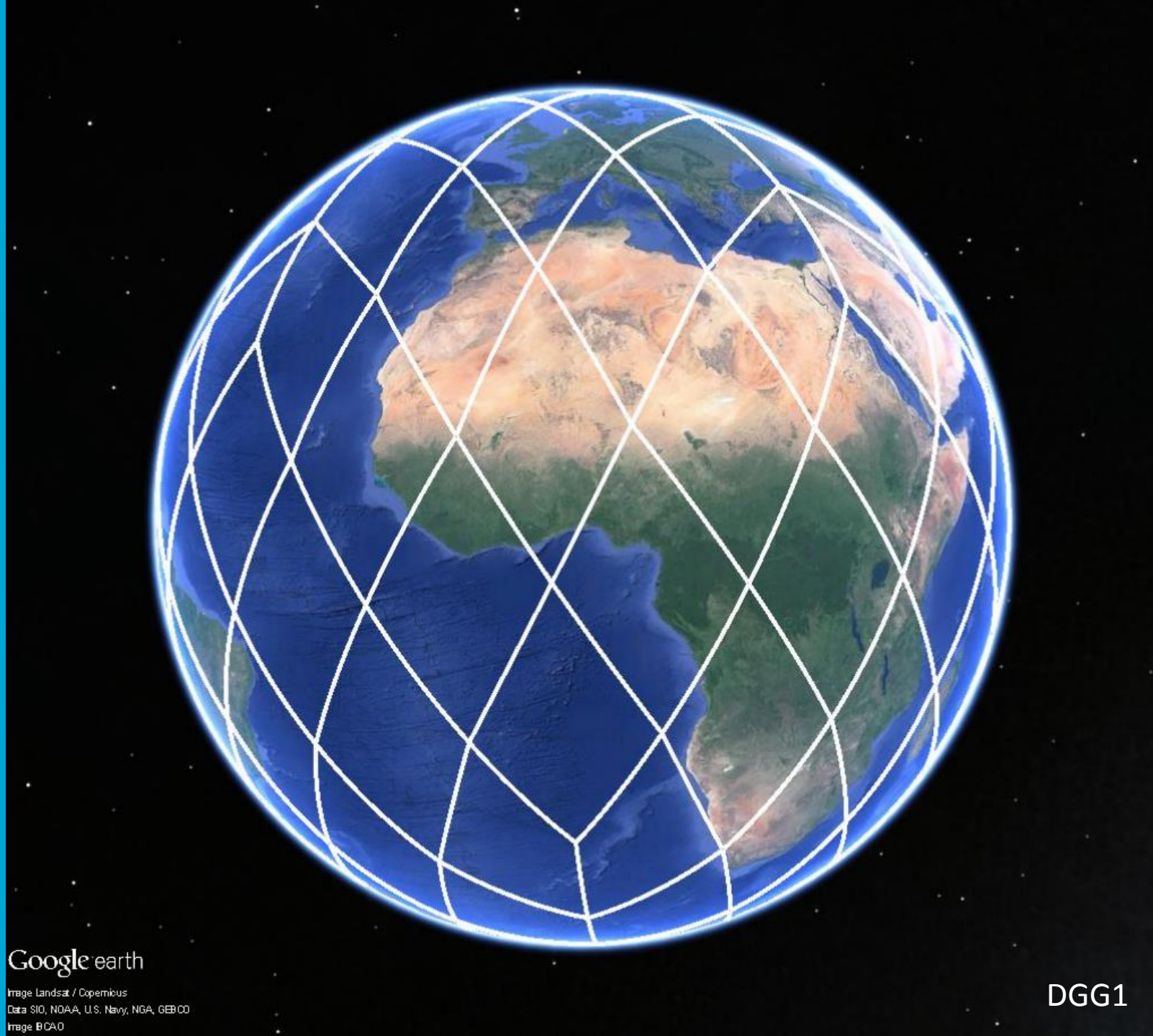
Discrete Global Grid Systems (DGGS)

“a spatial reference system that uses a hierarchical tessellation of equal-area cells to partition and address the globe.”
[OGC]

DGGS = [DGG1 + DGG2 + DGG3 + ……. DGGn]

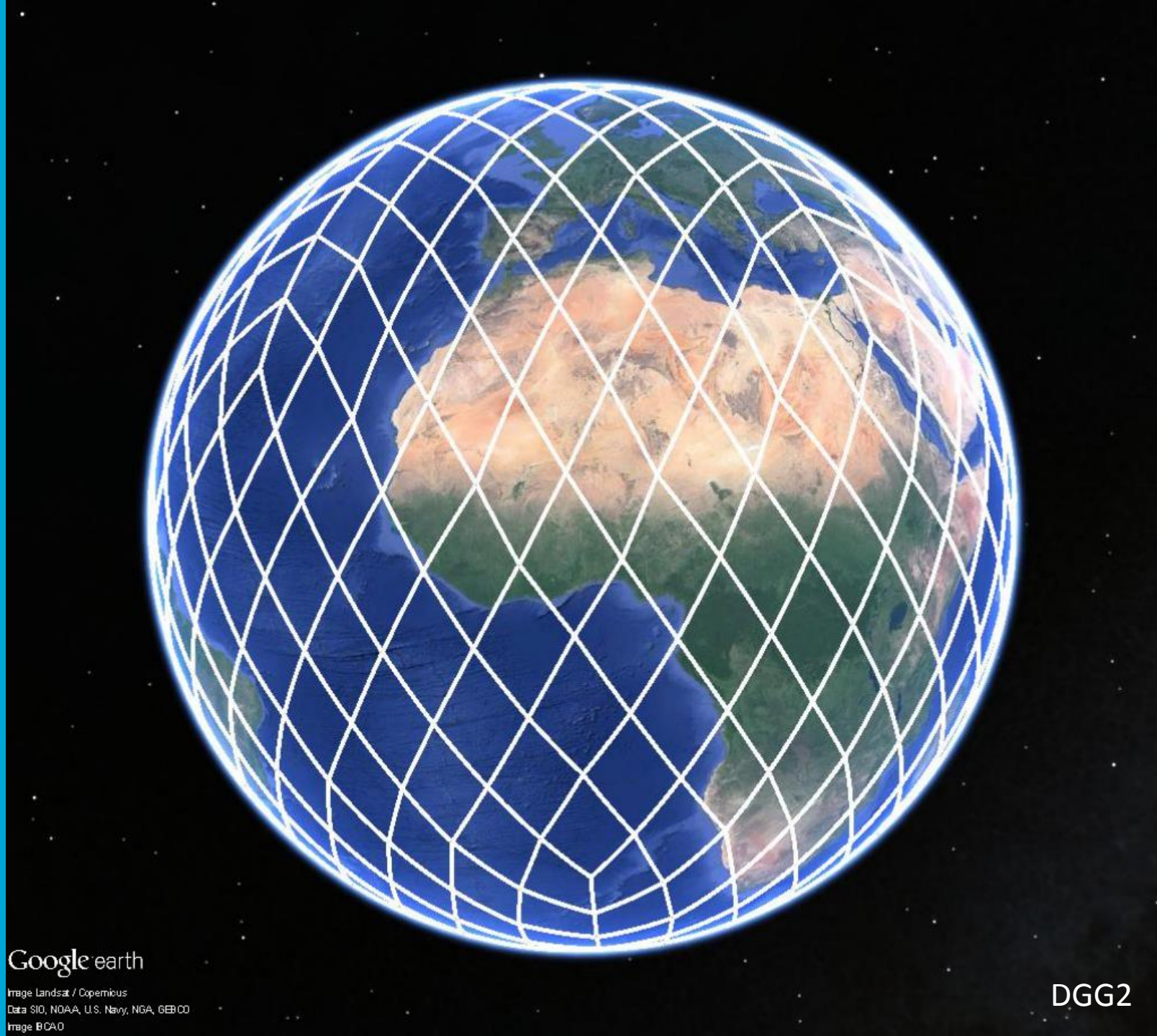






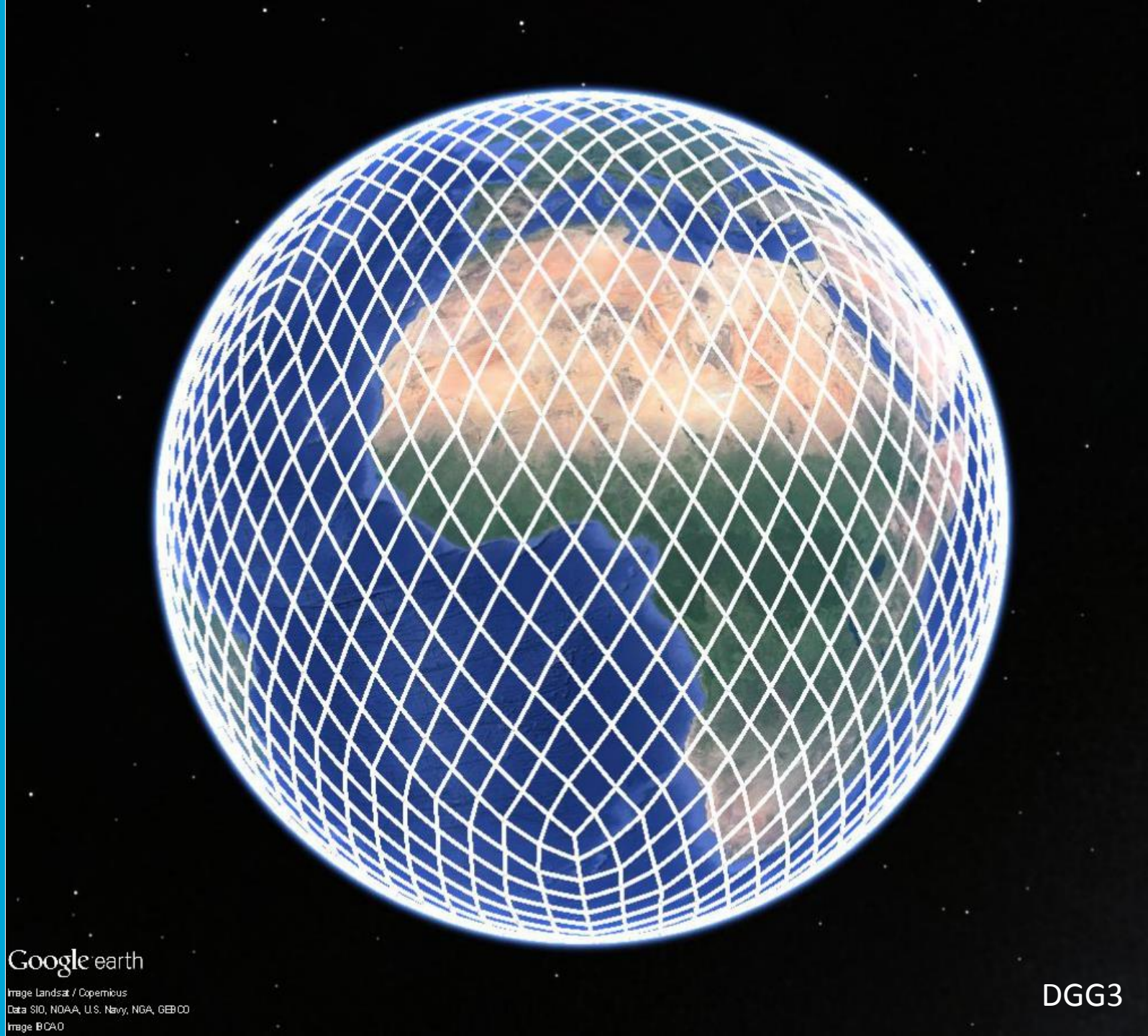
TU Delft

DGG1



TU Delft

DGG2

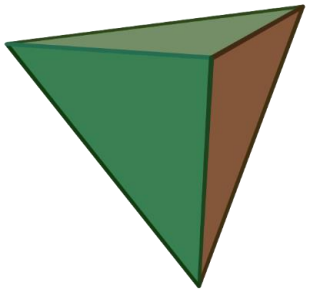


TU Delft

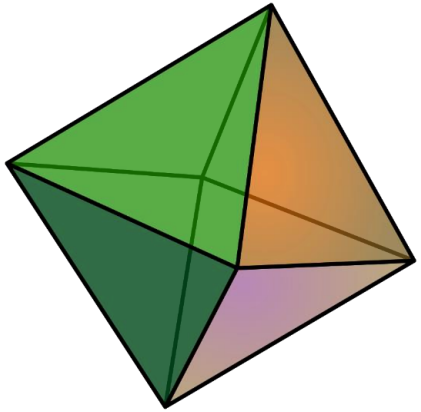
DGG3

Platonic polyhedrons

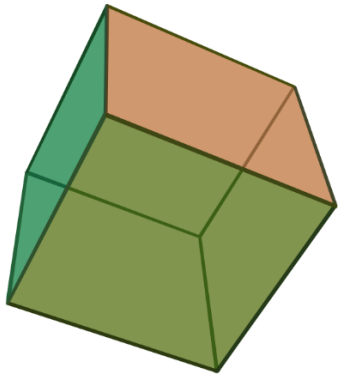
Tetrahedron

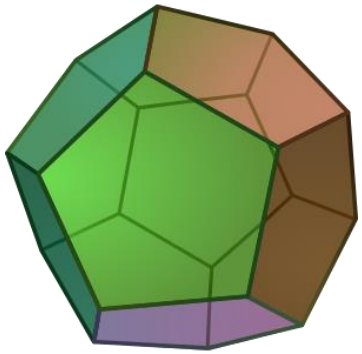


Octahedron

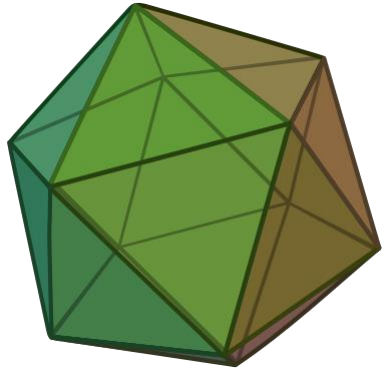


Cube



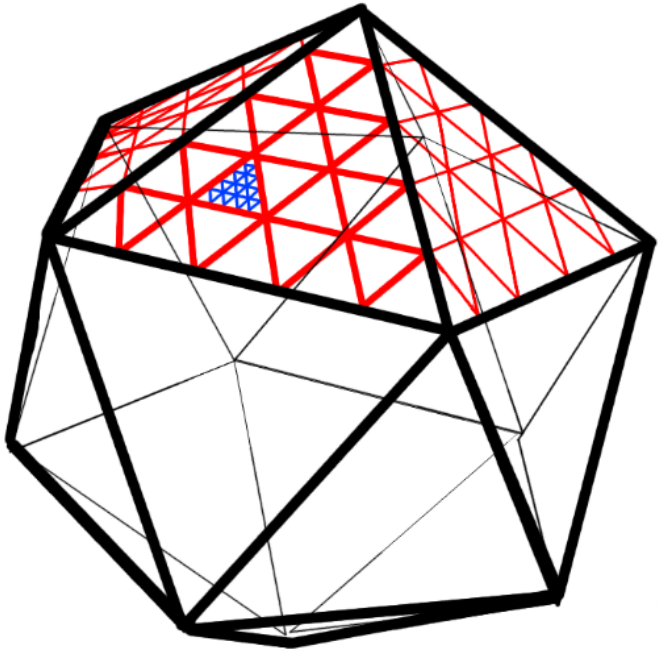


Dodecahedron

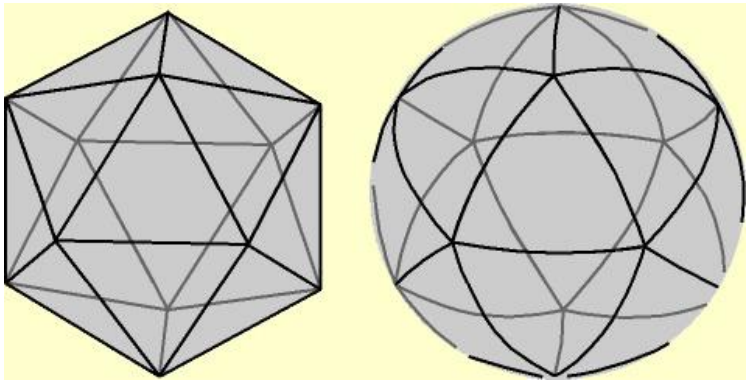


Icosahedron

Representing the Earth as a polyhedron



Subdividing the polyhedron faces – equal-area cells!



Icosahedron

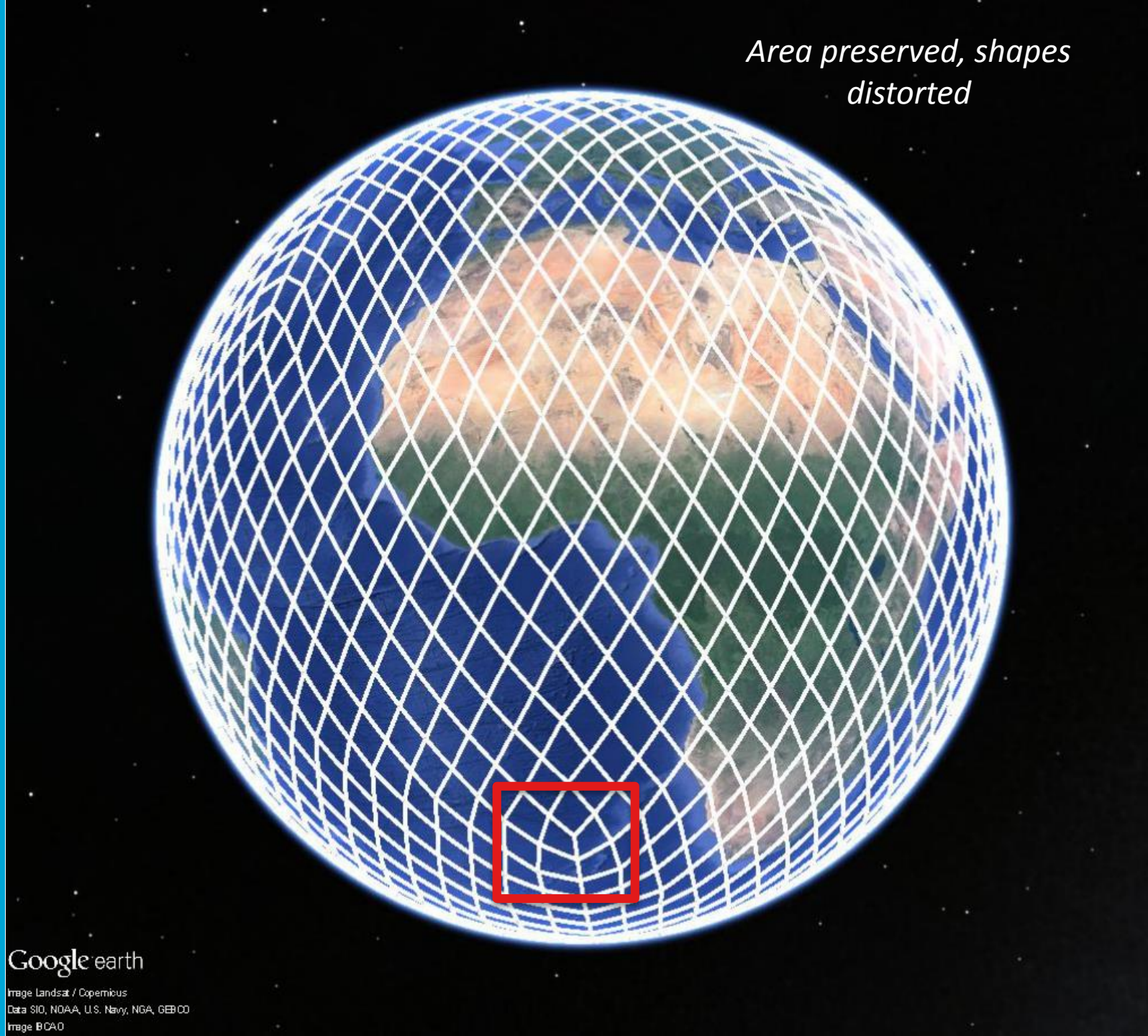
…..projected onto sphere/ellipsoid with equal-area projection === **equal-area cells**!

Discrete Global Grid Systems (DGGS)

DGGS =

[Polyhedron] +
[Polyhedron Orientation] +
Subdivision shape +
Refinement ratio (aperture) +
[Projection]

IMPORTANT: Equal-area cells!


Area preserved, shapes distorted
Google earth
Image Landsat / Copernicus
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image IBCAO

Research question

To what extent can a Discrete Global Grid System (DGGS) be used to handle point clouds with varying locations, times, and densities/levels of detail?

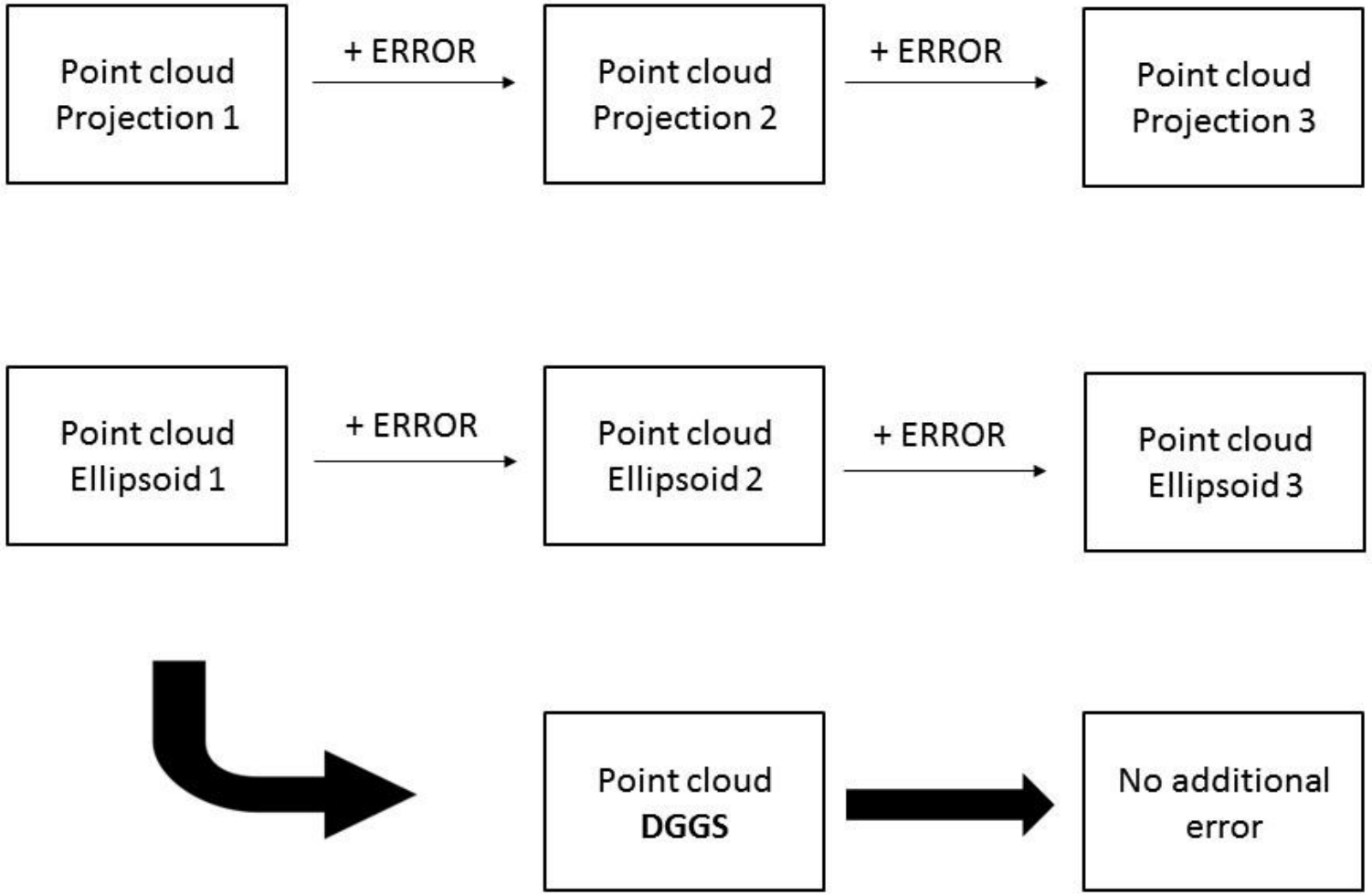
Sub-questions:

- **How can a variable-scale structure be implemented to support smooth-zoom in DGGS?**
- **How can a 4D DGGS be constructed to store the time component associated with point clouds?**
- **What are the advantages and disadvantages of using a DGGS as compared with conventional coordinate reference systems?**

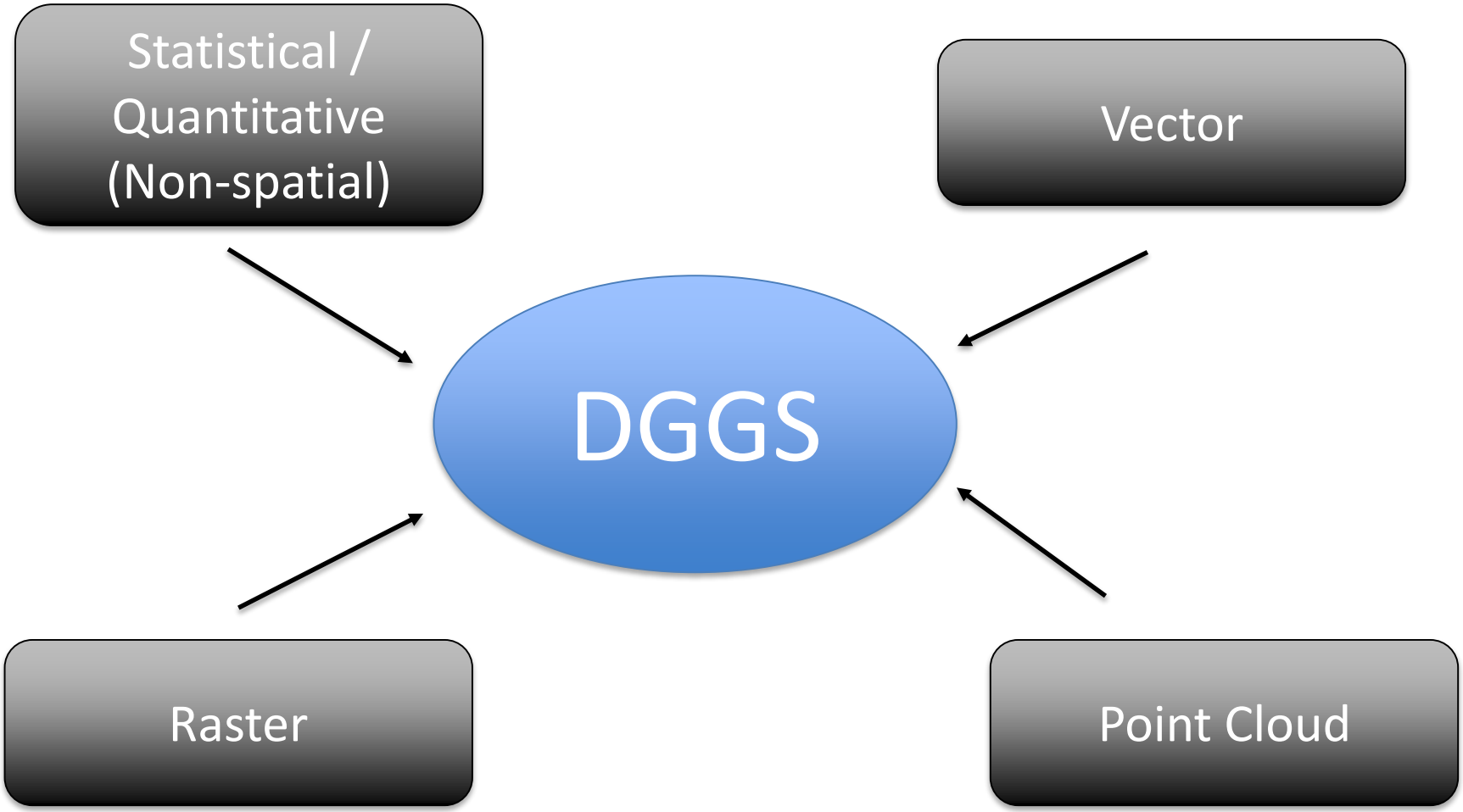
DGGS Vs. Map Projection

Why would one ever use a DGGS?

- **Multi-resolution** instead of single resolution- can analyze at arbitrary levels of detail !
- DGGS requires you to **import** your data into it **only once.**
  - **no** need to **re-project** your data once imported.
- DGGS are optimal for data integration for **vector**, **raster**, **point cloud**, and **non-spatial** datasets !
- From analogue → **digital** spatial analysis.
- DGGS **directly** works with ellipsoid/geoid without map projections!


Point cloud Projection 1
+ ERROR
Point cloud Projection 2
+ ERROR
Point cloud Projection 3
Point cloud Ellipsoid 1
+ ERROR
Point cloud Ellipsoid 2
+ ERROR
Point cloud Ellipsoid 3
Point cloud DGGS
No additional error

DGGS for geospatial data fusion

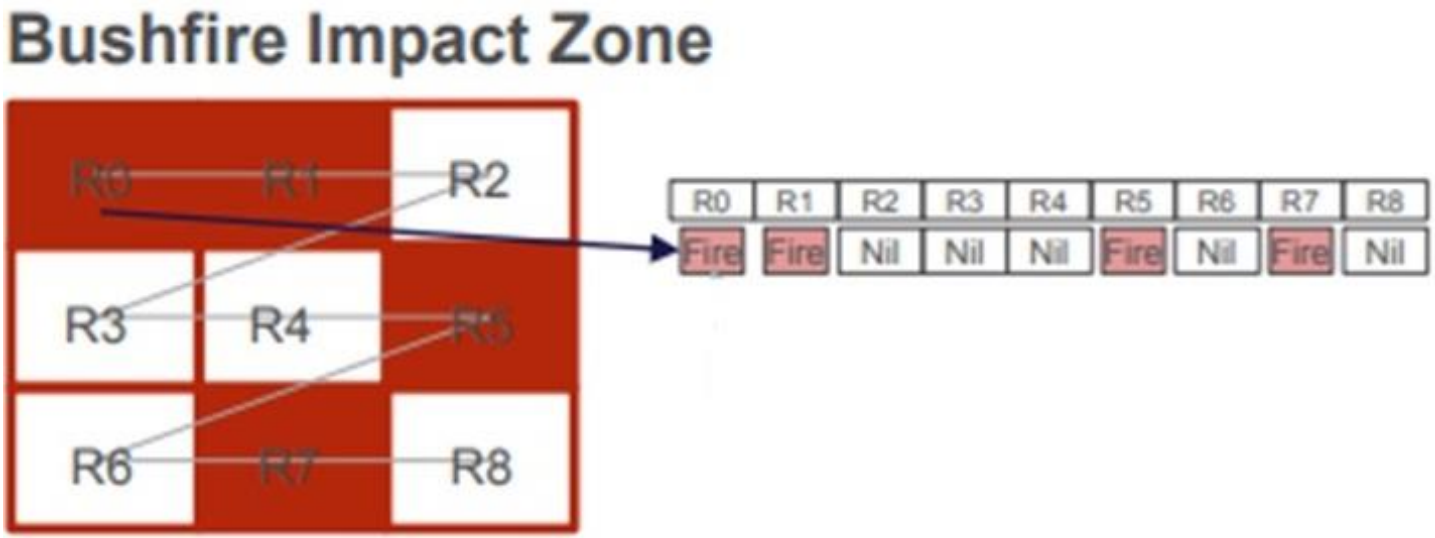


Spatial Analysis in a DGGS

“ Every spatial analysis done inside a DGGS is an array process using set theory. This is scalable. ”

![TU Delft logo]

Array Processing Using Set Theory



Using Statistical Boundary only

Fire Impacting whole area

Using DGGS

Fire Impacting ~44% of area

Array Processing Using Set Theory

Population (per dwelling)

R00	R01	R02	R03	R04	R05	R06	R07	R08
0	0	3	0	0	5	0	9	0

Using Statistical Boundary only

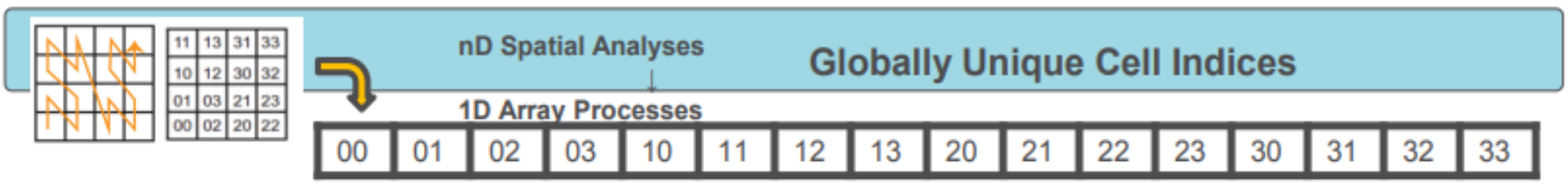
65 People Impacted

Using DGGS

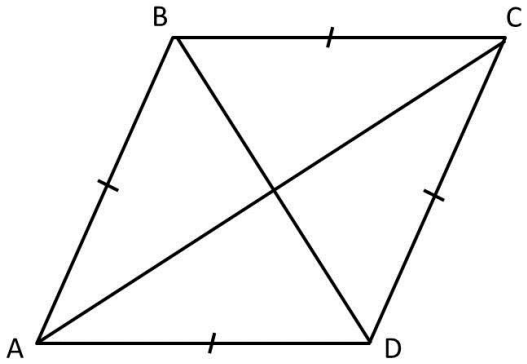
33 People Impacted

Spatial Analysis in a DGGS

Encoding a feature into a DGGS reduces its geography to a series of 1- dimensional addresses along a SFC well-suited for big data geoprocessing.

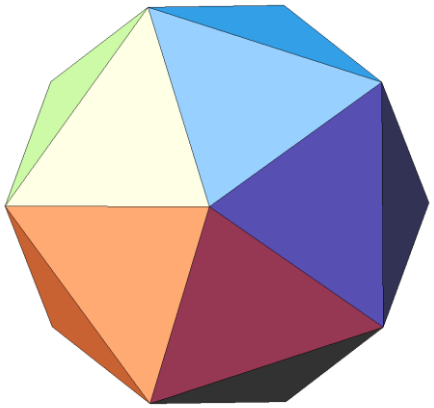


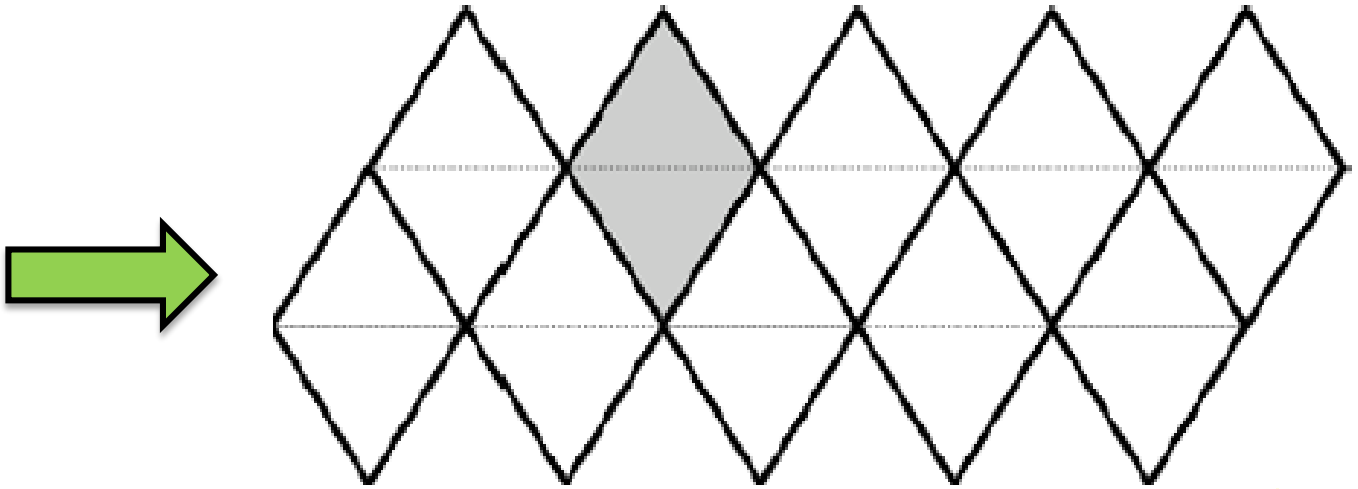
Choice of subdivision shape - rhombus



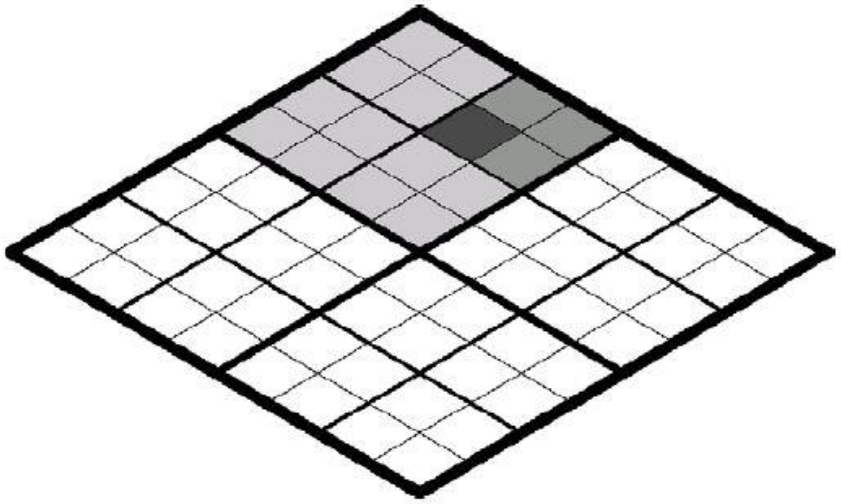
- **Unlike triangles**
  - Exhibits uniform orientation
  - Have radial symmetry
- **Unlike squares**
  - Used on triangle-faced polyhedrons common in DGGS, such as icosahedron… better approximations of Earth!
- **Unlike hexagons**
  - **Congruent**: Each parent rhombus can be exactly split into four smaller child rhombuses
  - Easier to hierarchically index
  - Can benefit from quadrant-recursive algorithms like the **Morton ordering**
  - Have simpler geometry

Flattened Icosahedron



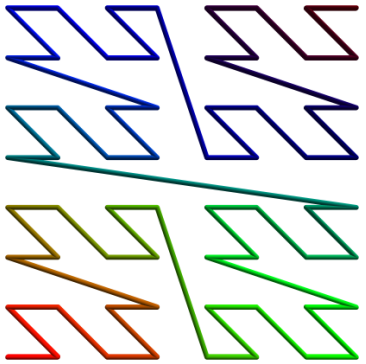


Flattened icosahedron composed of **rhombus** cells



An aperture 4 **rhombus** nested hierarchy

Morton SFC Indexing

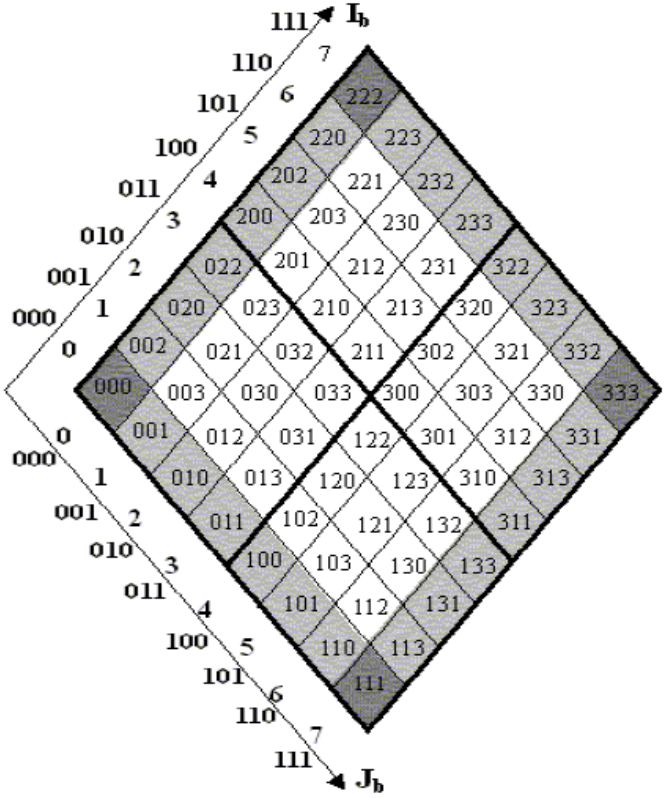


- Morton indexing can be applied on each face of a rhombus DGGS to ensure each cell has a unique ID

And extended into n-D……

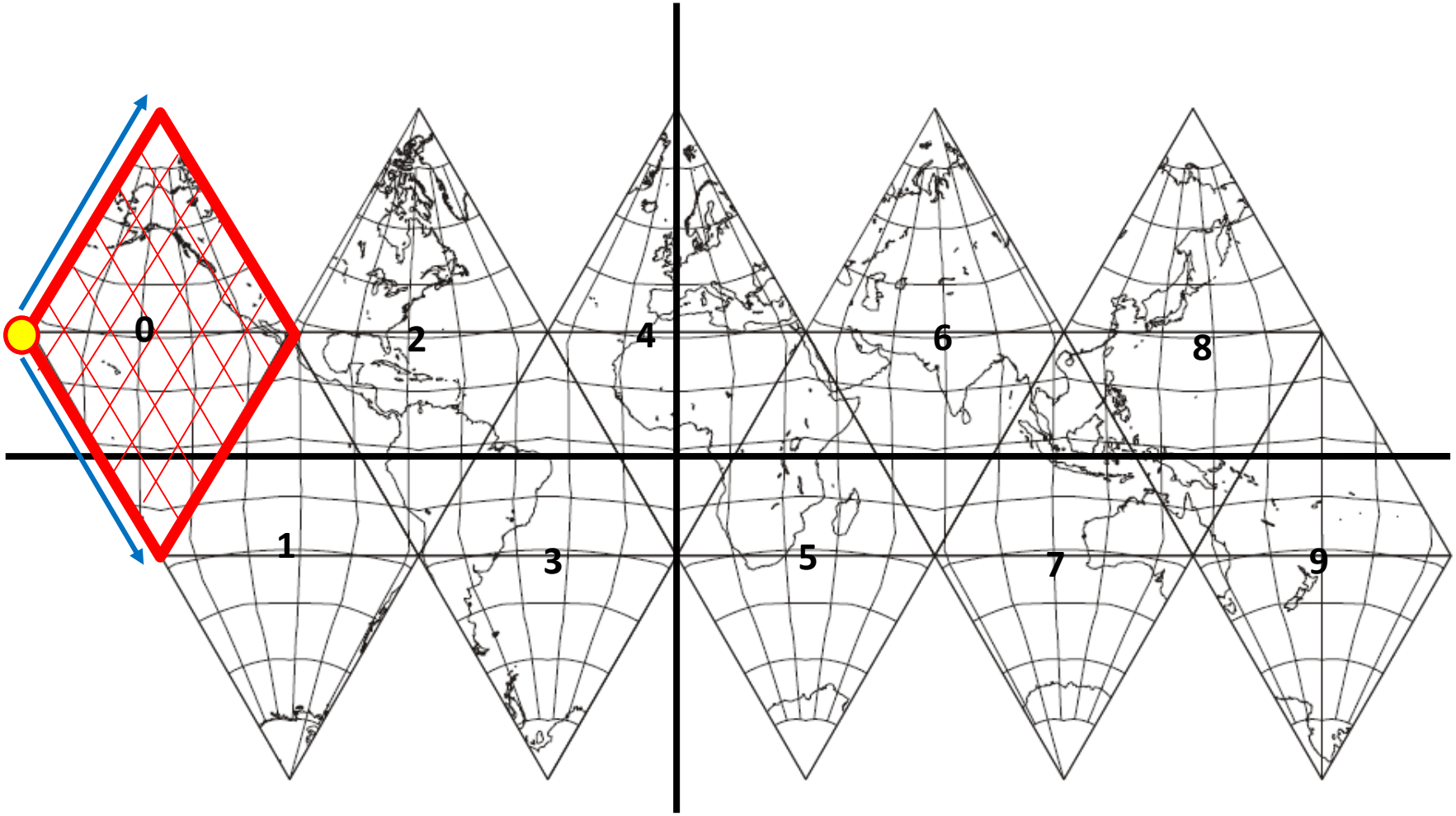
MC = x + 2y + 4z + 8t + …. 2^(n-1) * dimension

Where **x, y, z**, and **t** all have been scaled to the same range of values



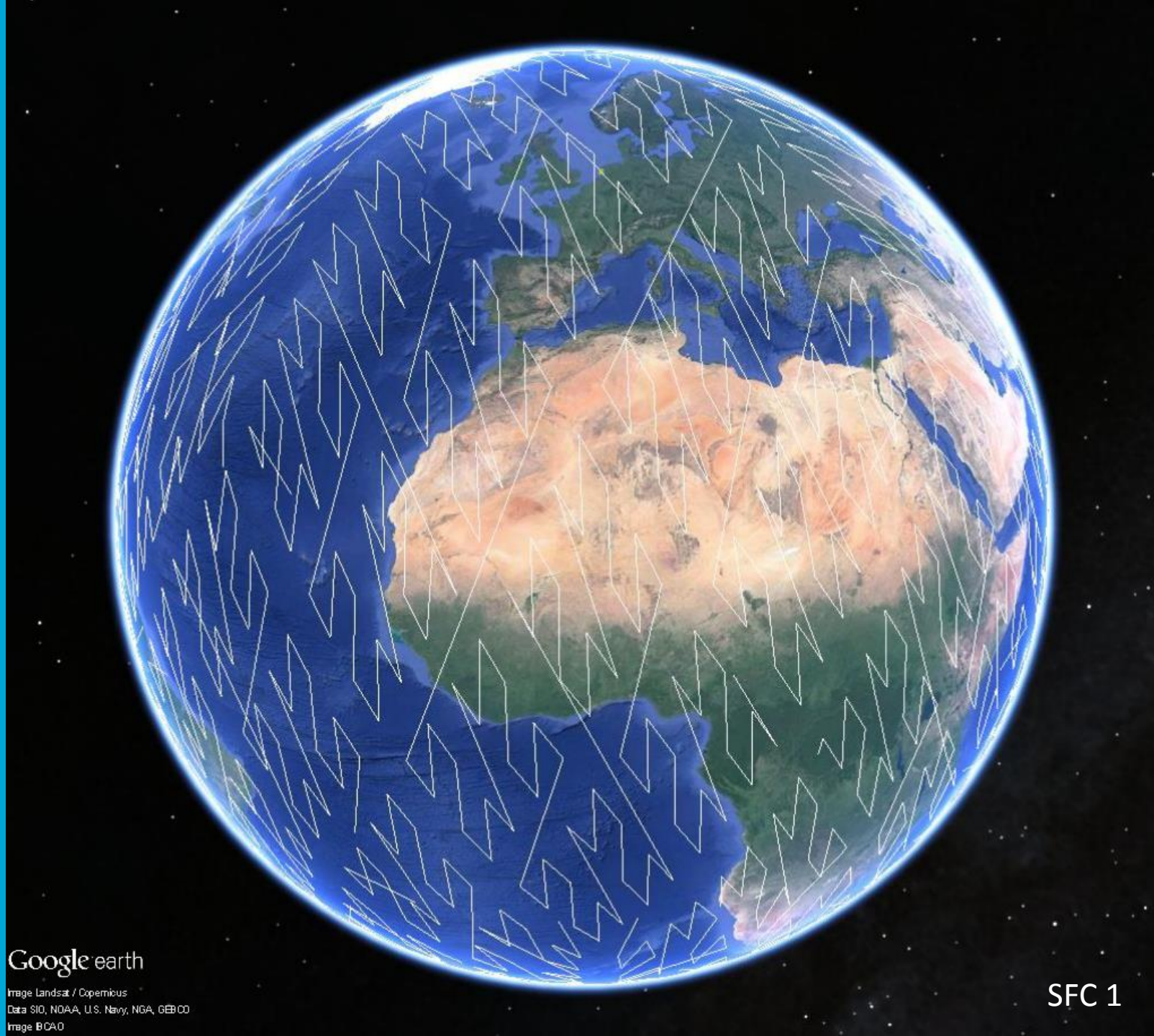
TU Delft

Icosahedral Snyder Equal Area (ISEA) Projection



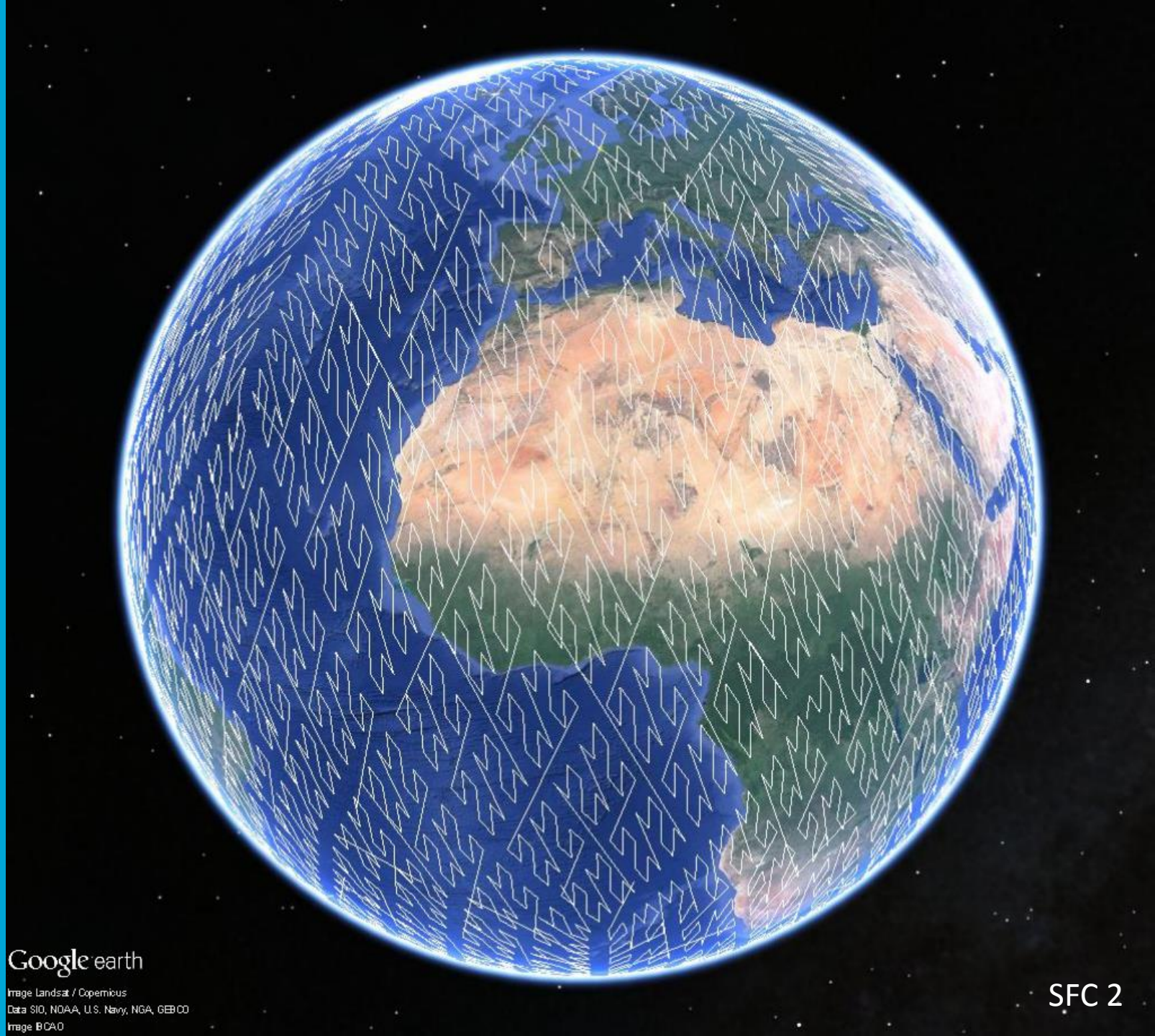
Path addressing

- Path through the tree structure of DGGS
- **Does not** store redundant information
- **Simplifies** hierarchical algorithms and performance of spatial queries
- Rhombus DGGS lends itself **exceptionally well** to a path-based address
- Can get rather long, especially for high resolutions
- Coarser resolution cells can be **used as a filter** for spatial queries at higher resolutions
  - EXAMPLE
    - **4657600661367006630661054334633470**



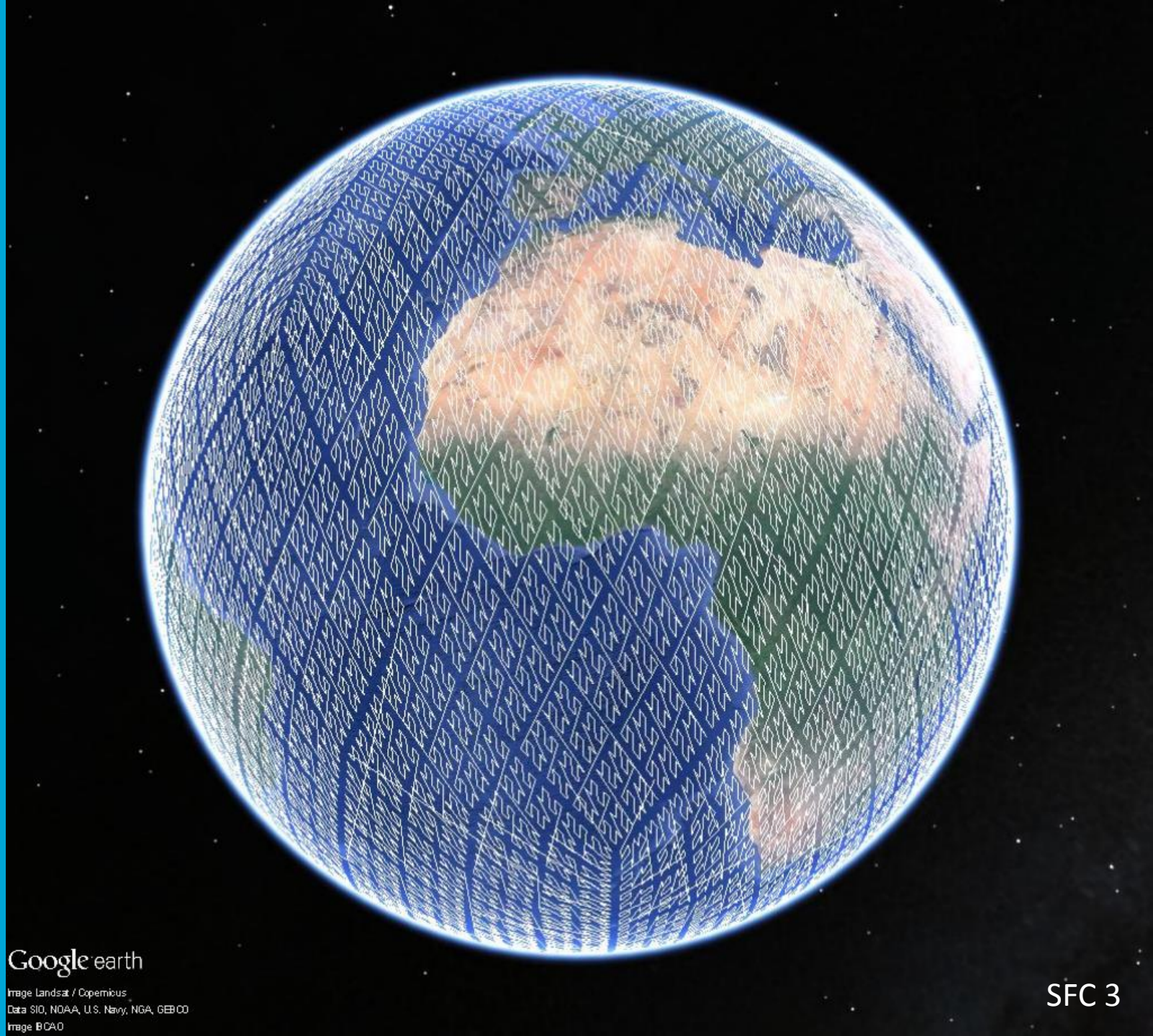
TU Delft

SFC 1



TU Delft

SFC 2


Google earth
Image Landsat / Copernicus
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image IBCAO

TU Delft

Morton Code properties

***e.g.* 465760066136700 6**

→ *Resolution:* ***15***

→ *Rhombus face*: **4**

→ *Discrete precision:* **47,503 sq. meters**.

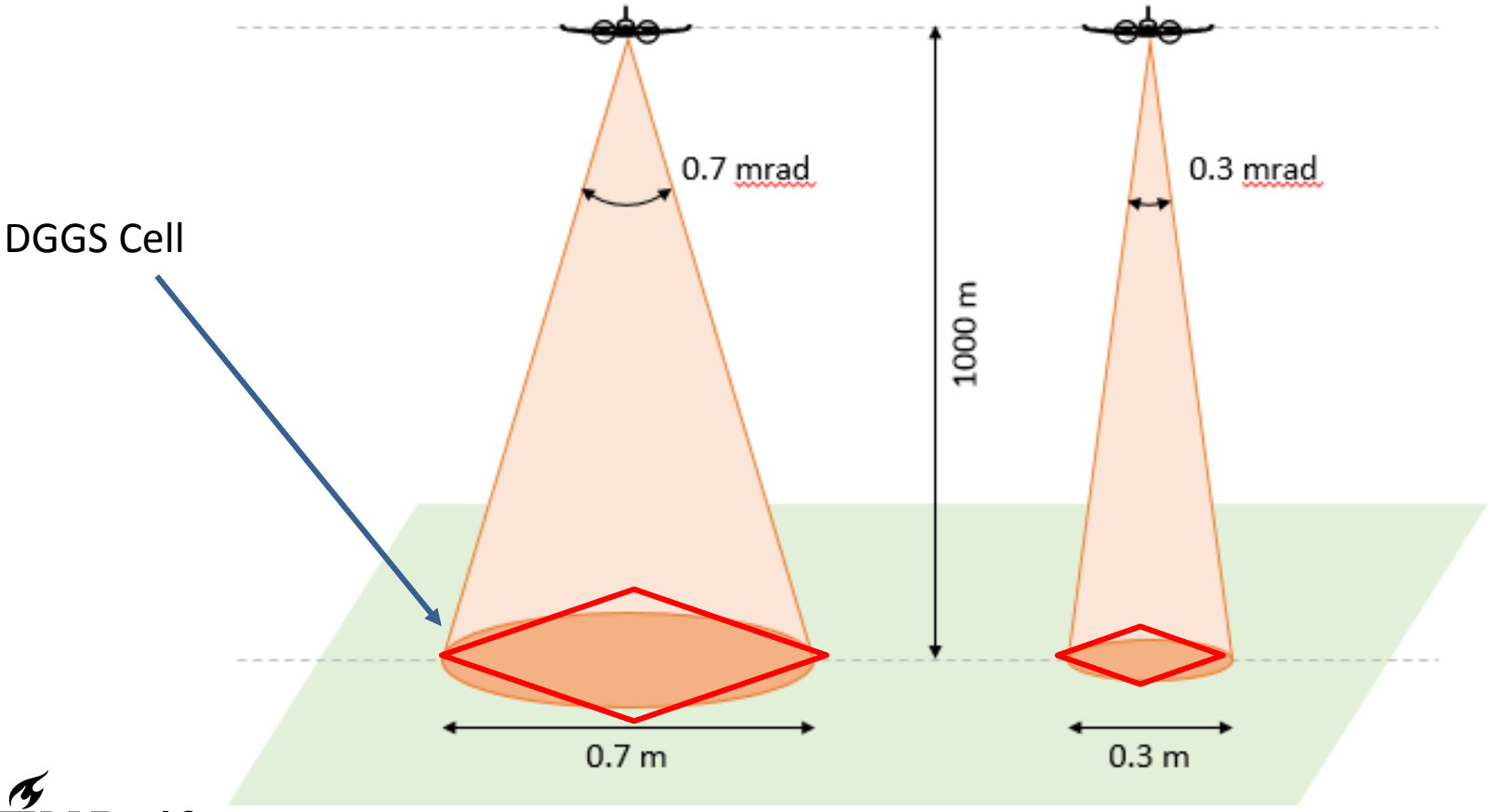
→ *Number of cells:* ***10,737,418,240***

→ *Intercell spacing:* ***215 meters***

- First digit → rhombus face.
- Length of code indicates discrete point precision / DGGS resolution.
  - Number of cells/ intercell spacing.
- Incorporate *n* dimensions.
- Full-resolution code transforms nD cells into (0-D) points.
- **Power of DGGS** comes from the ability to “move” observations up and down DGGS hierarchy.
  - *Truncation/ concatenation operations on Morton codes!*

Smooth Zoom in DGGS

"**Importance**" value assigned to points based upon their precision → relationship with LIDAR beam footprint



Computing point precisions

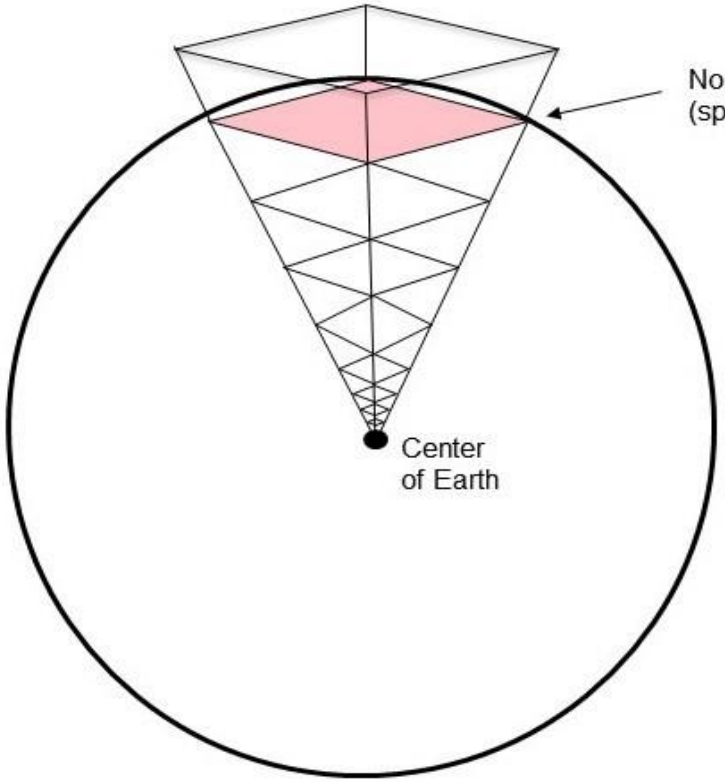
- Time component matched

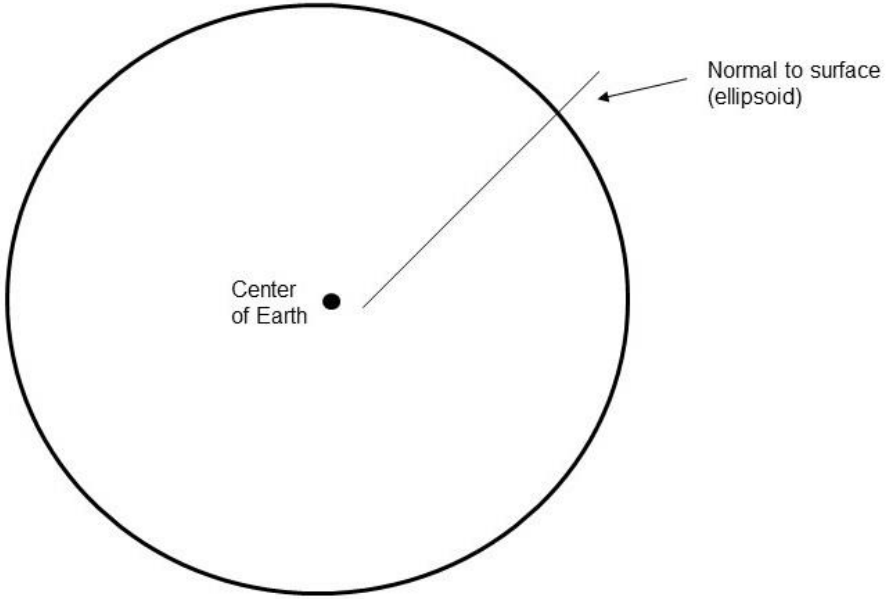
Riegl VQ-250 scanner

Distance	Beam Footprint
Exit aperture	7 mm
50 m	18 mm
100 m	36 mm

3-D DGGS

- 3D – the 3D location in space in an earth centered system of the point
  - Including what is above and below the surface!



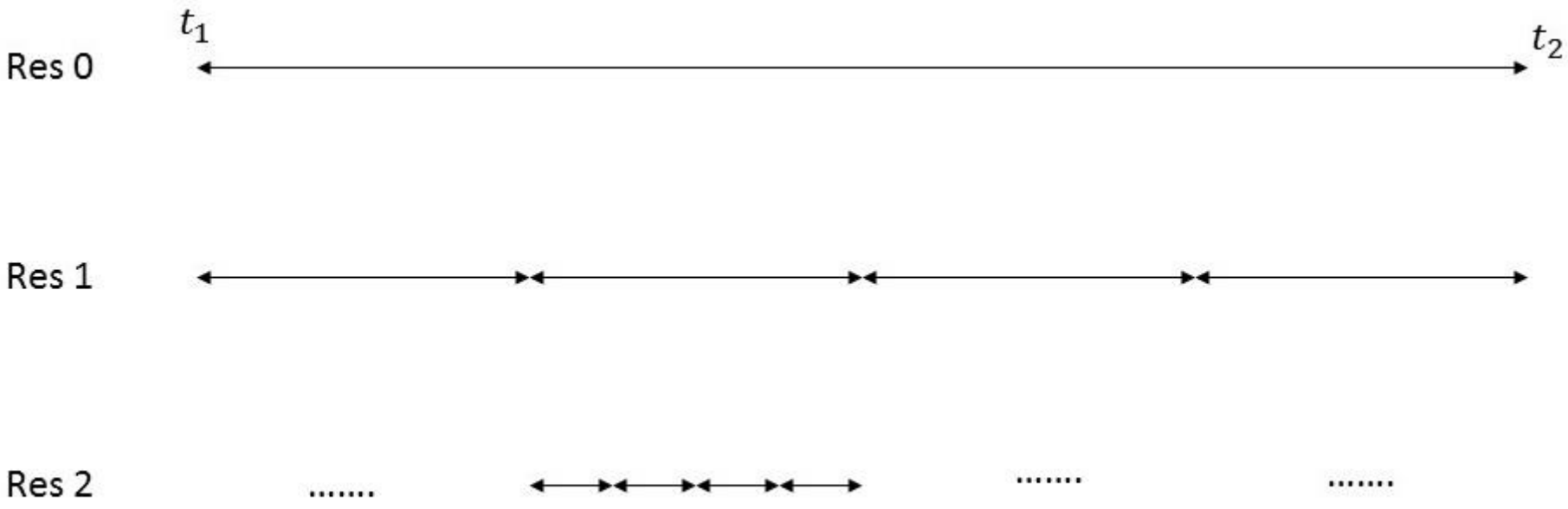


3-D DGGS

- Extended on either base polyhedron or ellipsoid
- Volume increases outwards from center
- Normal only for sphere
- Z-range application dependent
- Z-range extends below surface
- Separate hypercube on every face of polyhedron
- *If extended on ellipsoid:* Works for *any* shape and *any* polyhedron

4-D DGGS

- 4D – inclusion of time (**that has been discretized as a hierarchy of nested temporal ranges**) into the DGGS cell and its ID



3D DGGS Morton Code example

Bouwkunde, TU Delft
Latitude: 52.005891 N
Longitude: 4.370296 E
Altitude: 56 meter above MSL, WGS84

Code	Resolution	Horiz. Precision
465760	5	49,811 sq. kms.
46576006613	10	48.64 sq. kms.
465760061367	12	3.04 sq. kms.
4657600613670066306610	22	2.89 sq. ms.
465760066136700663066105433463	29	1.77 sq. cms.
4657600661367006630661054334633347	32	→ ~ sq. millimeter

4D DGGS Morton Code example

Bouwkunde, TU Delft

Latitude: 52.005891 N

Longitude: 4.370296 E

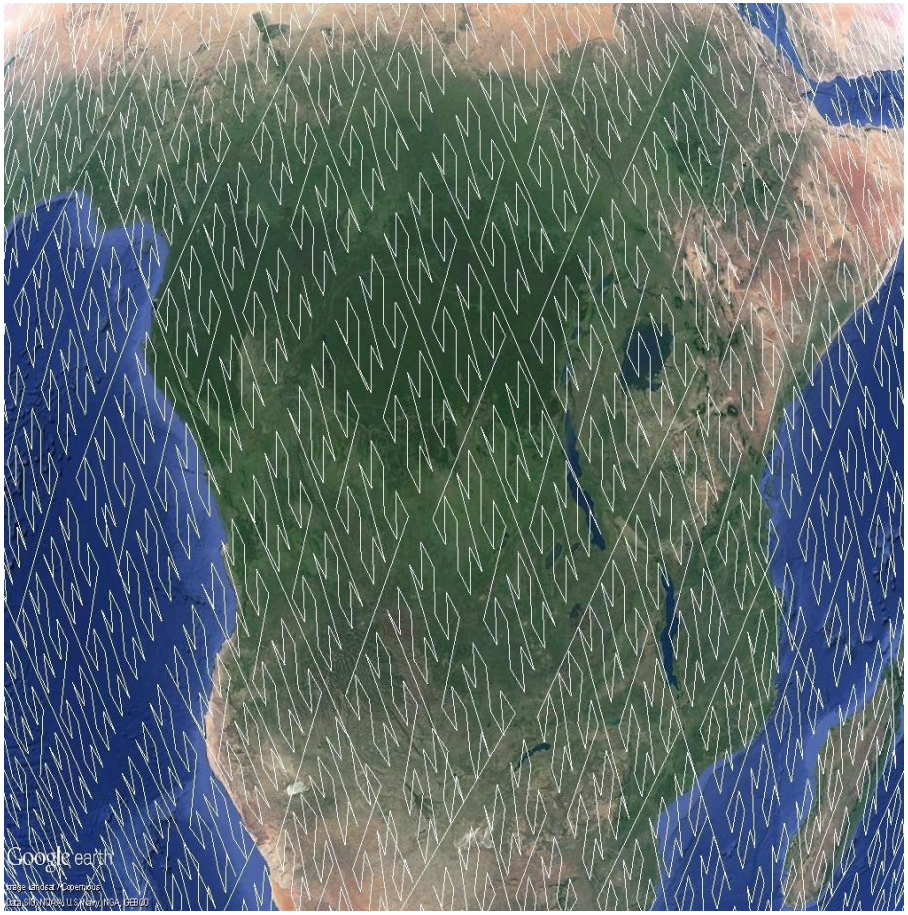
Altitude: 56 meter

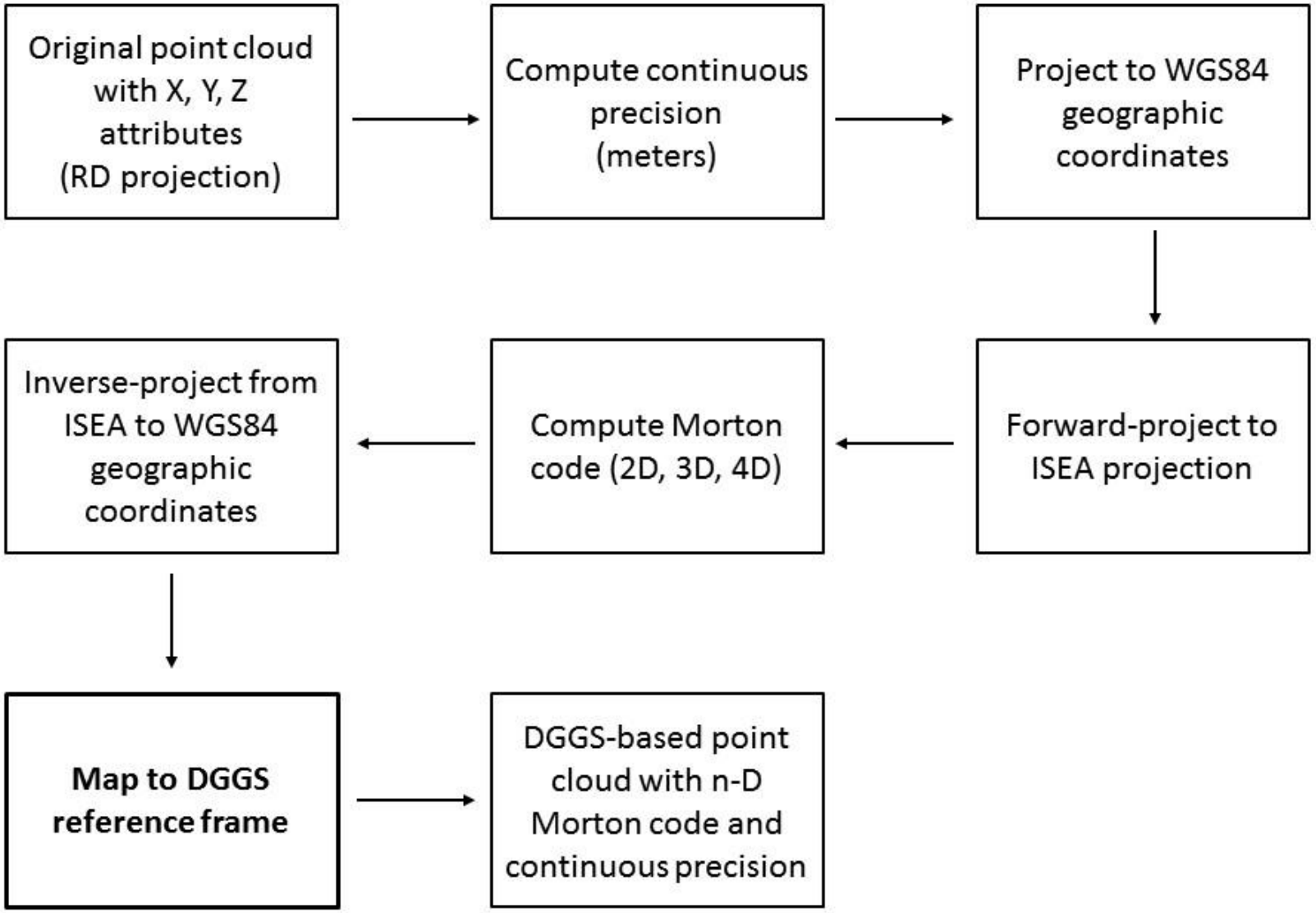
Time: March 1, 2016 13:35:26 pm UTC

Code (hexadecimal)	Resolution
4EDFE0	5
4EDFE00EE93	10
4EDFE00EE9367	12
4EDFE00EE936780E6B86E98	22
4EDFE00EE936780E6B86E9854B3C6334F	32

DGGS Morton SFC Indexing of the Earth

- ✓ Path-based
- ✓ Hierarchical
- ✓ Space-filling
- ✓ Continuous
- ✓ Order-consistent
- ✓ Adjacency-Preserving
- ✓ Precision-encoding
- ✓ N-dimensional




Original point cloud with X, Y, Z attributes (RD projection)
Compute continuous precision (meters)
Project to WGS84 geographic coordinates
Forward-project to ISEA projection
Compute Morton code (2D, 3D, 4D)
Inverse-project from ISEA to WGS84 geographic coordinates
Map to DGGS reference frame
DGGS-based point cloud with n-D Morton code and continuous precision

SFC convergence

Point observation:
Latitude: 36° N
Longitude: 25° E
Elevation (m): 44 m (WGS84)
Time: February 1, 2008 12 AM UTC = 885,859,218 seconds GPS Time

4D Morton code (hexadecimal): ***4F38AAA1D23699081A69D5B67D79A2928B***

Resolution	Latitude	Longitude	Elevation (m)	GPS Time (s)	Equivalent UTC
2	30.2809	18.7822	0.0	599,616,027	Jan 6, 1999 00:00:09 AM
5	35.7917	23.7038	0.0	861,948,031	Apr 30, 2007 06:00:13 AM
10	35.9975	24.9653	39.0625	885,370,531	Jan 26, 2008 08:15:13 AM
15	36.0006	24.9989	43.9453	885,846,301	Jan 31, 2008 20:24:43 PM
25	36.0000	24.9999	43.9999	885,859,203	Jan 31, 2008 11:59:45 PM
32	36.0000	25.0000	44.0000	885,859,218	Feb 1, 2008 00:00:00 AM

~ 8 years

15 sec

TU Delft

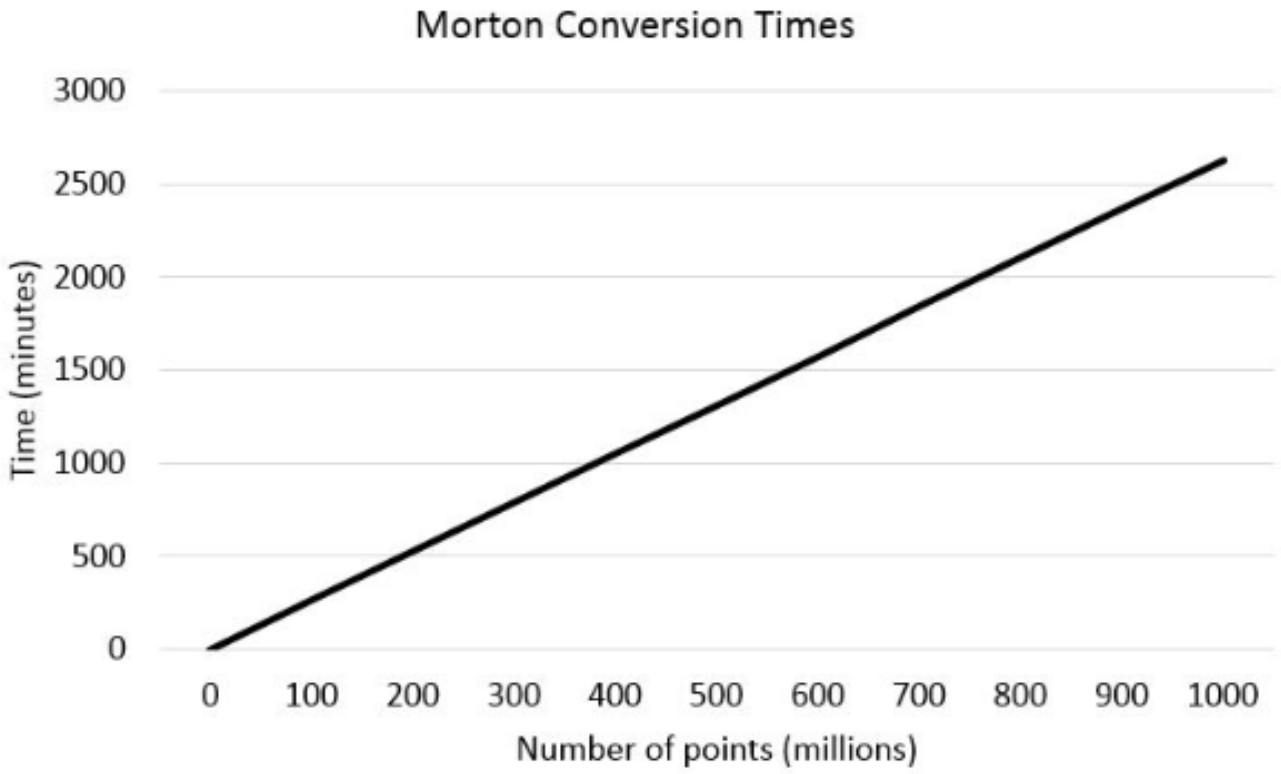
Datasets

- Two scans of the Forepark area in the Hague, Netherlands
- Acquired as a part of the FUGRO DRIVEMAP project using LIDAR

Dataset	File Size	Number Of Points
2010	5.44 GB	162,918,748
2016	6.64 GB	198,365,000

- Datasets stored in two separate flat tables (one point per row)

Morton Conversion Times



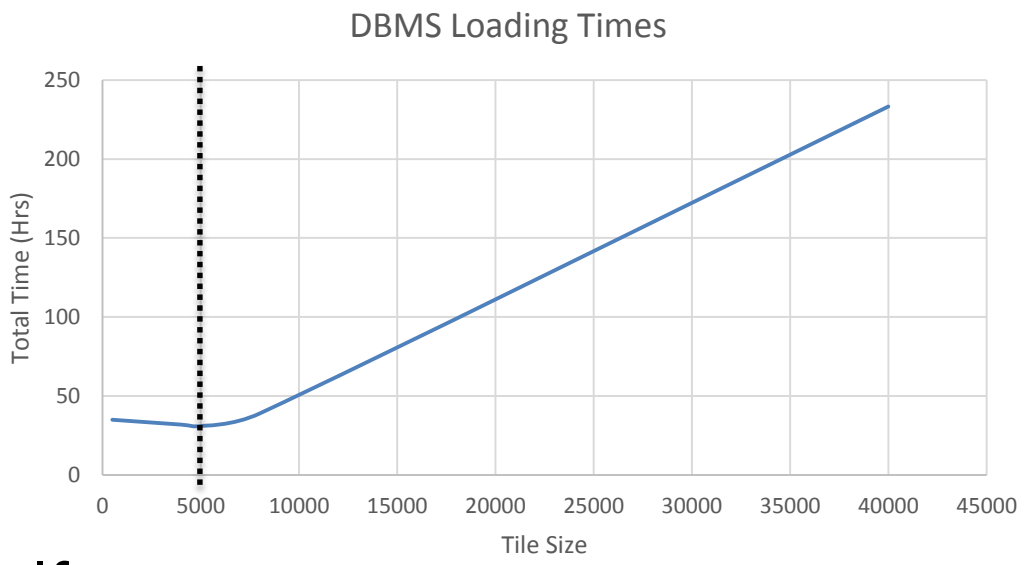
2010 dataset
-162 million points
- Conversion took ~ 420 minutes

2016 dataset
-198 million points
- Conversion took ~ 510 minutes

DBMS Loading Times

Split into tiles of 5,000 points each for optimal loading times

Number Of Points	Number Of Files	File Size (KB)	Est. Total Time (Hrs.)
40,000	4,073	1,329	233.25
8,000	20,365	266	38.91
5,000	32,584	167	30.98
4,000	40730	134	31.85
500	325,838	17	34.93



--- reduced to only **17 minutes** using Psycopg2 COPY_FROM method !

TU Delft

Loading/storage into DBMS

- Psycopg2 COPY_FROM method reduced loading time to only 17 minutes for 162 million points
- Datasets stored in two separate **flat tables** (one point per row)
- Patches not used
- Storage requires up to:
  - 2 bits / digit for 2D Morton code
  - 3 bits / digit for 3D Morton code
  - 4 bits / digit for 4D Morton code

Storage of points

- Morton codes can be stored up to discrete level, or full-resolution (i.e. res 32)
  - Discrete Morton codes make it impossible to uniquely identify the location and time of each and every point within a 2D/3D/4D cell → necessitates storage of additional attributes
  - **Storing full-resolution code lowers storage significantly → get distinct points**
- Attributes stored: 3D Morton code, 4D Morton code, discrete precision, continuous precision
- Morton codes stored as **TEXT**

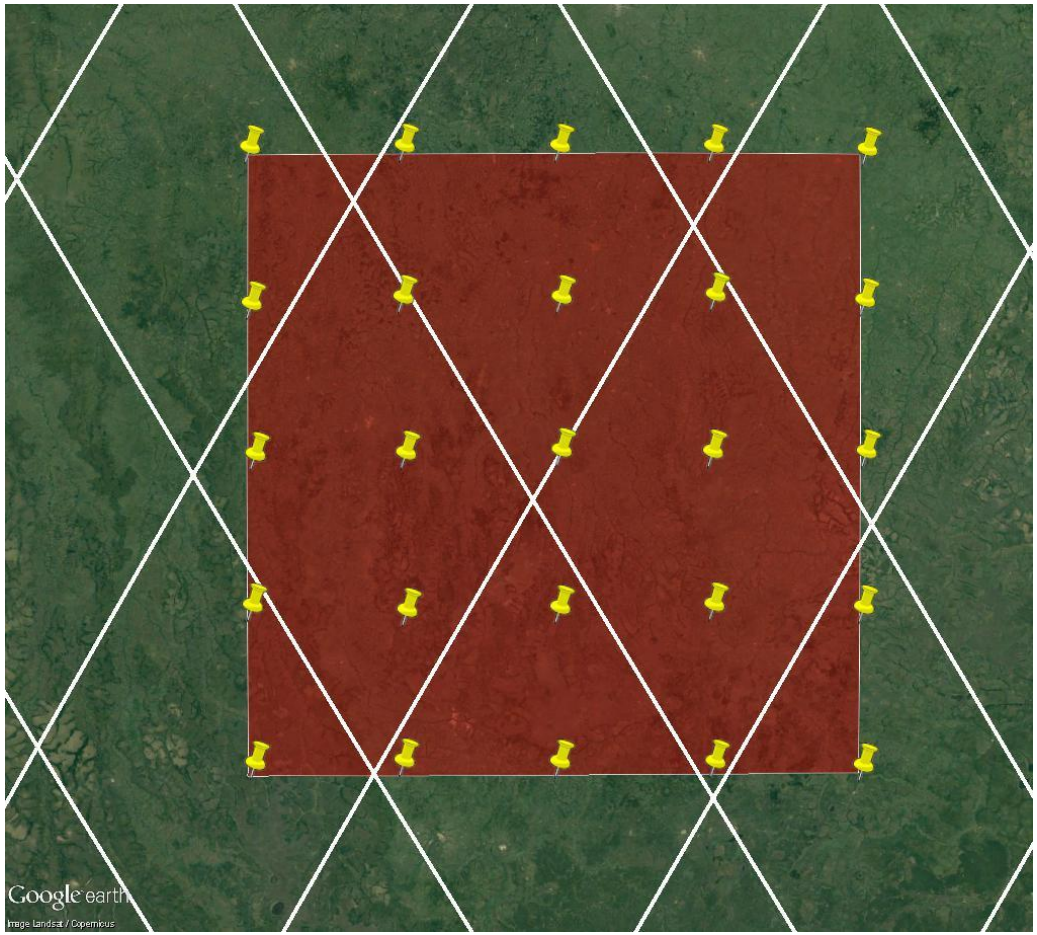
Storage of points

- Power of DGGS comes from ability to **interpolate** or **decimate** your observations to move to a higher or lower level in the hierarchy
- Full-resolution codes stored (until res 32), but indexed and clustered on only a substring of the code (until res 17)..... i.e. Trimmed to lower precision
- Retrieved from DBMS **using only the substring**

e.g. 4657600661367006306610543346334 70

Query procedure

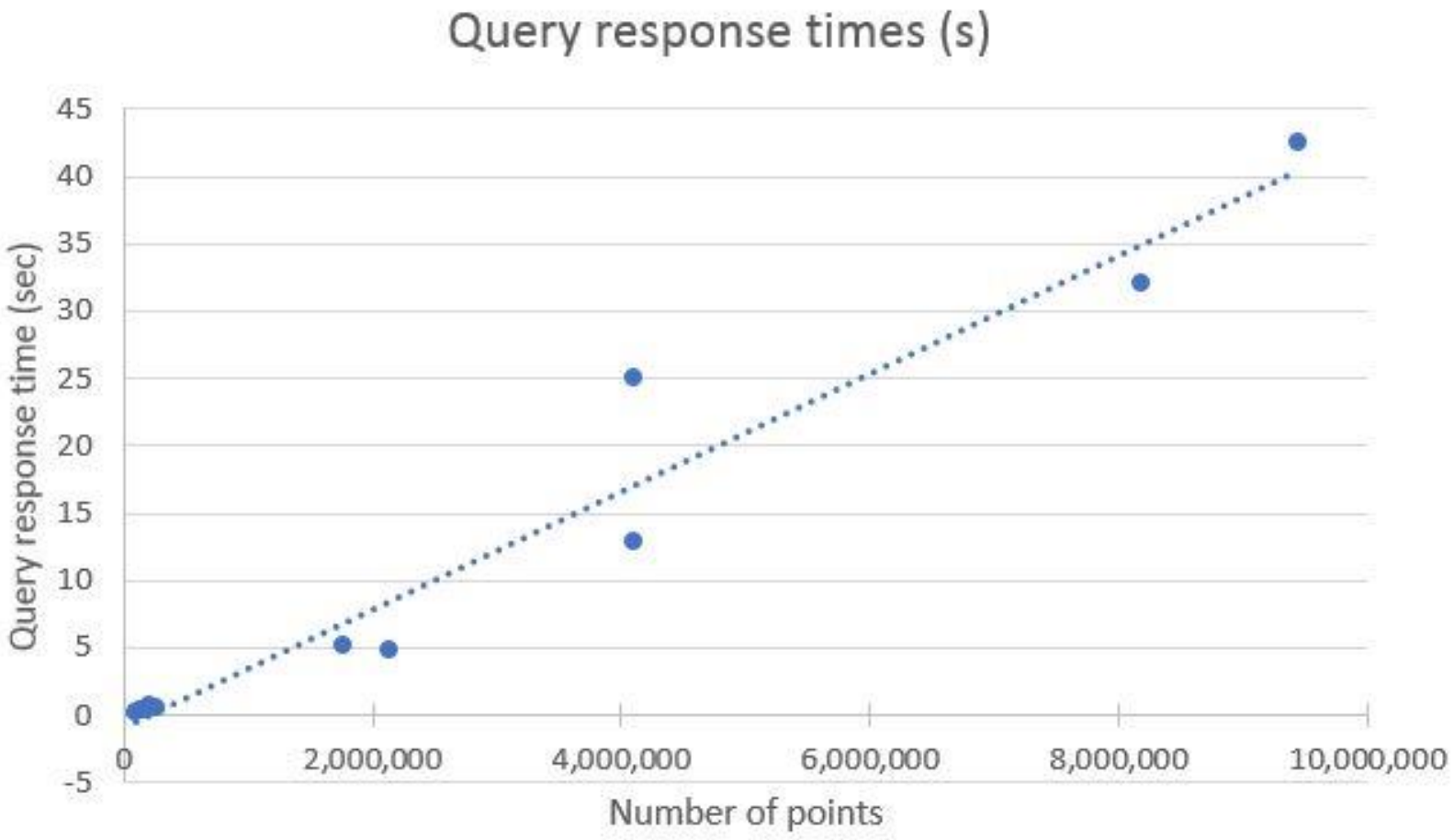
- Geometry construction, comparison, and querying requires large overhead
- DGGS structure **does not** require storage of cell geometries
  - Points have same SFC Morton codes as cell (in nD)
- Query window can be a user-defined simple polygon or series of cells acquired from user clicks
- Point spacing < DGGS intercell spacing



Index on full key vs. part

- Index on full key:
  - Extremely large (lot of disk space)
  - Query procedure cannot be utilized; spacing between points will have to be made very, very small
  - Index creation will take longer
  - Query execution will take longer
- Index on part (e.g. res 17):
  - Allows to exploit *cell* nature of DGGS
  - Allows utilization of cell navigation operations
  - Can perform more meaningful spatial analysis

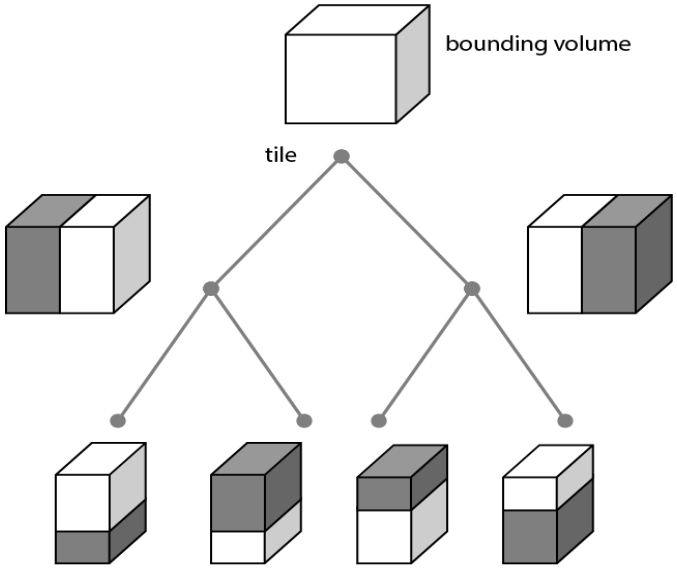
Query execution times

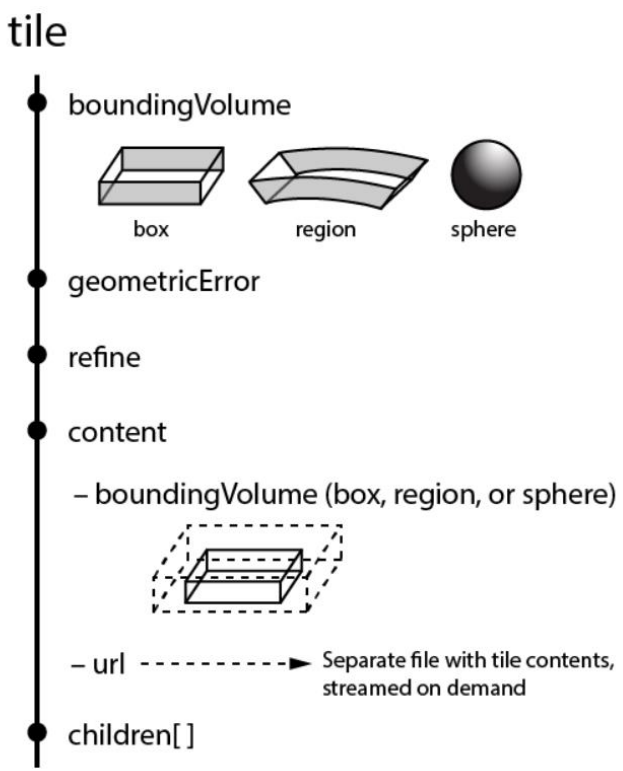


Web visualization

- Decode Morton codes
- Sort by importance
- Group into blocks--- number of blocks arbitrary, ***the more the better for smooth zoom***
- Put each block in a 3D Tile
- Visualize in Cesium JS using Node JS/Express JS

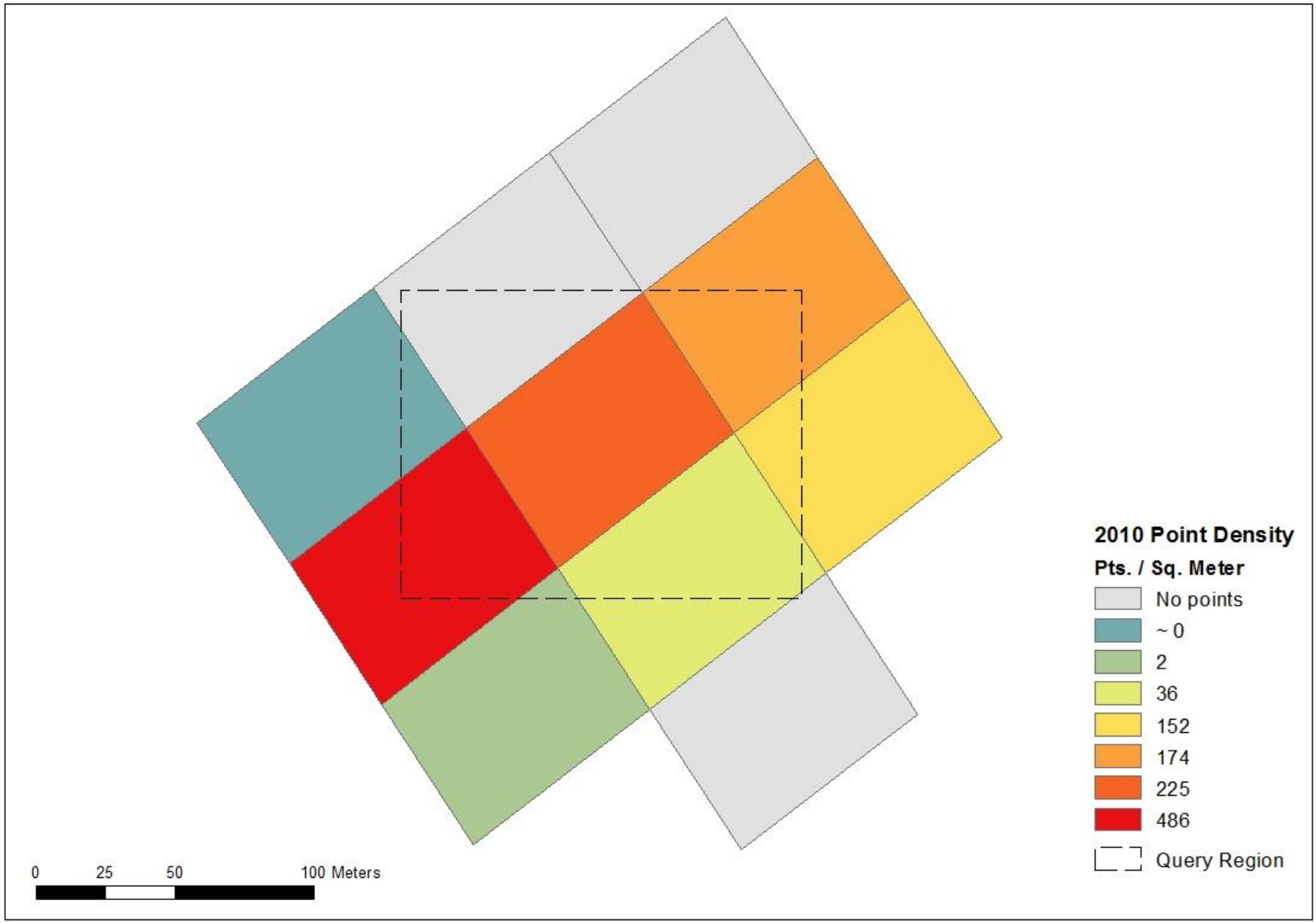
- **Geometric Error** property and **Additive Refinement**

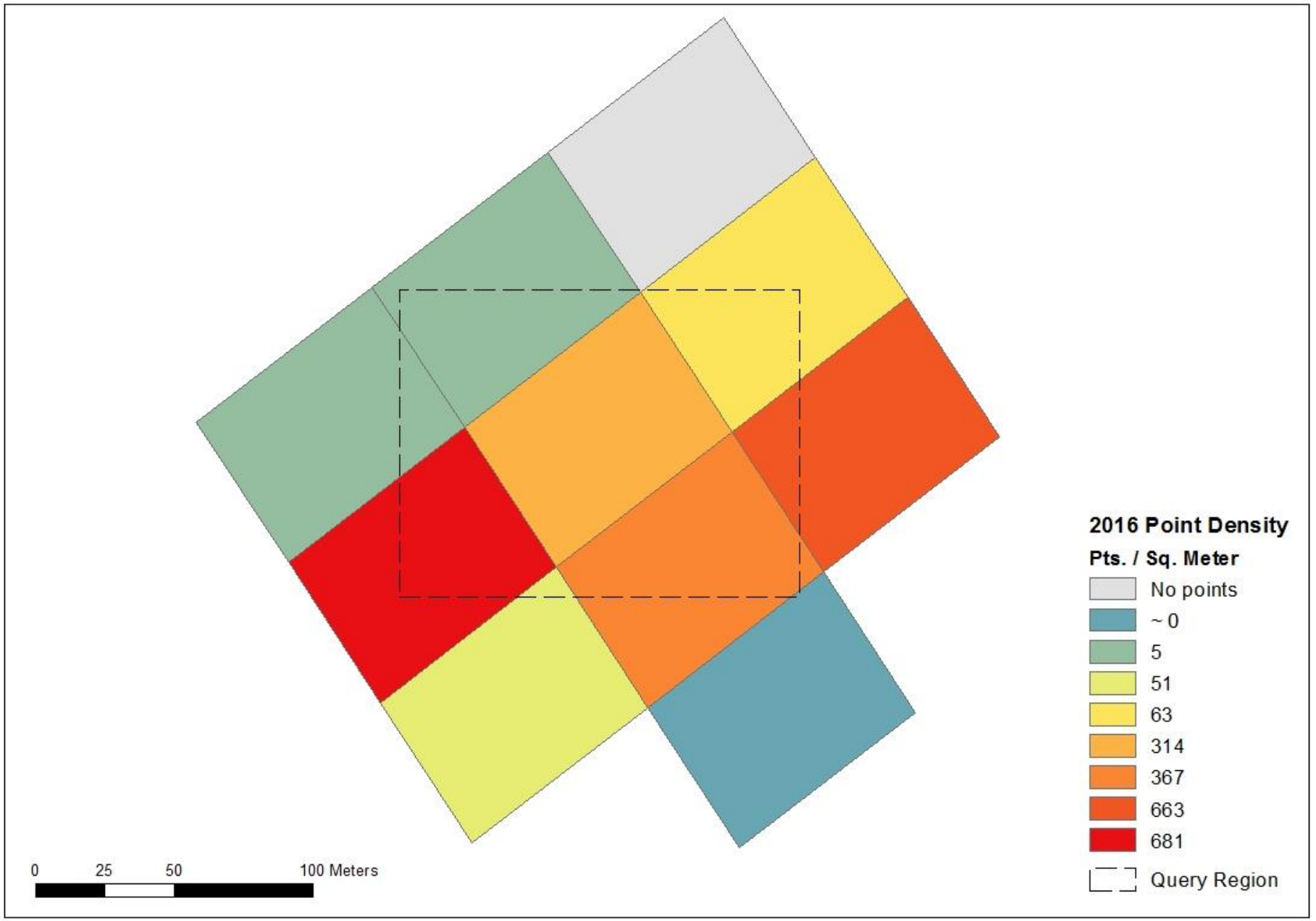


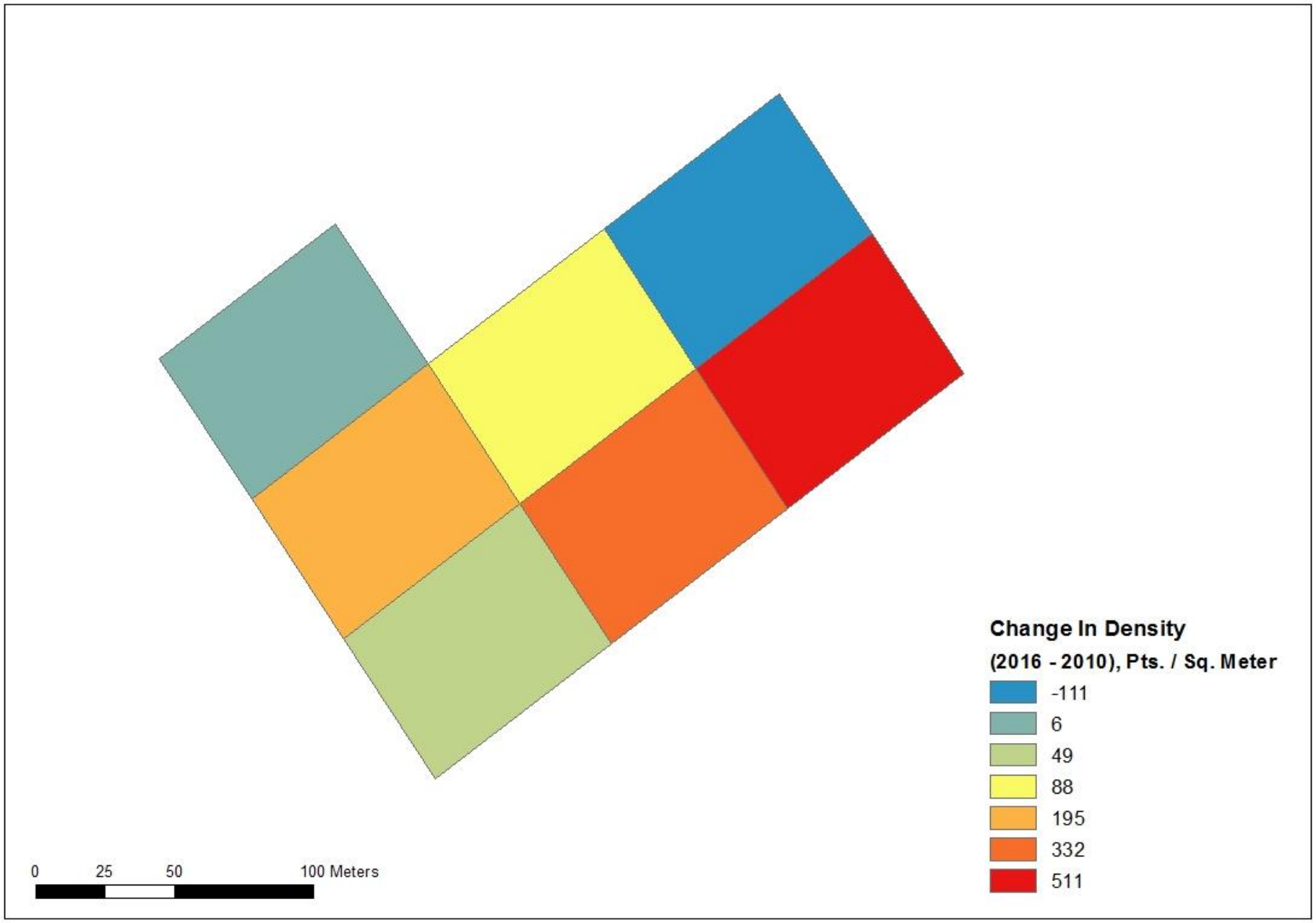


DGGS for change analysis

- DGGS is an **ECEF** system
  - Cells of DGGS “fixed” onto Earth once orientation defined
  - Coordinates of cells do **not** change due to Earth rotation and tectonic movement
  - **Ideal** for change identification and analysis because differences in the exact same area/volume on Earth are compared over time
  - Scalable operations


2010 Point Density
Pts. / Sq. Meter
No points
~ 0
2
36
152
174
225
486
Query Region
0
25
50
100 Meters

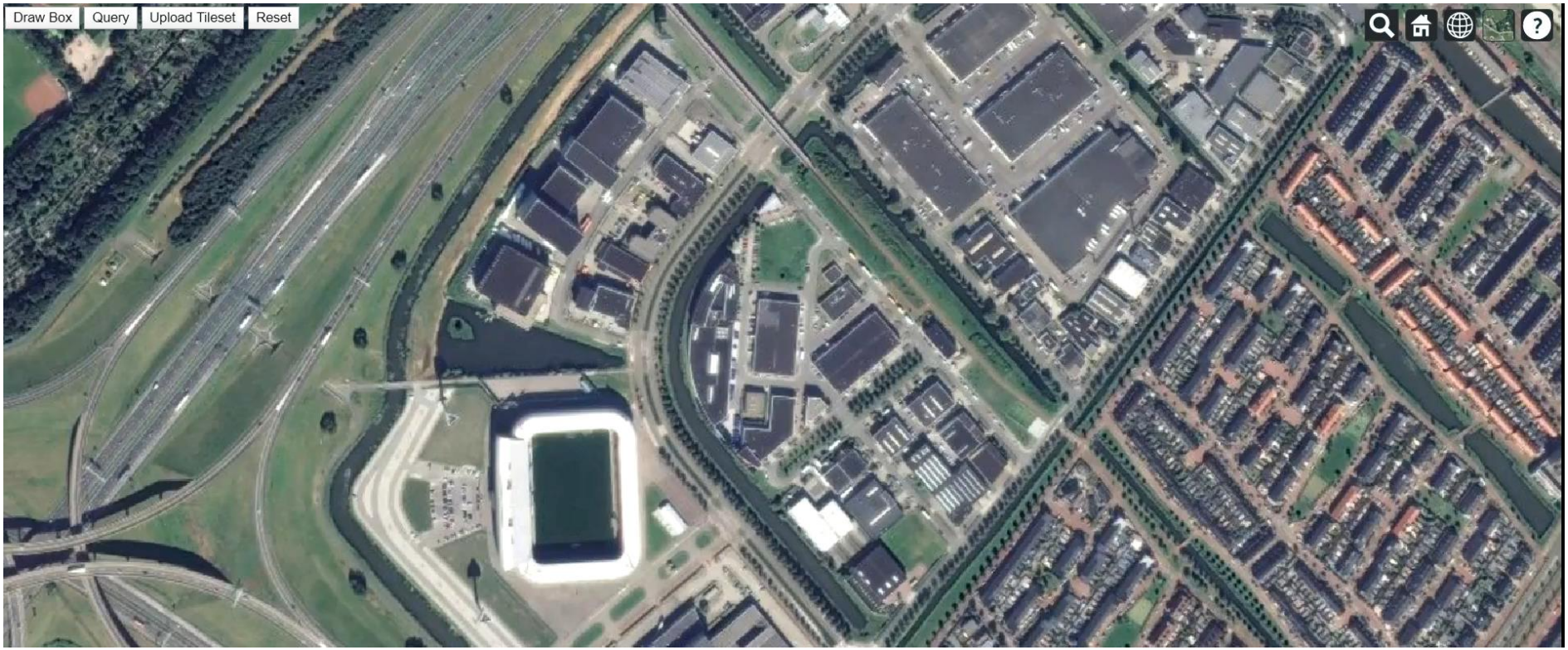

2016 Point Density
Pts. / Sq. Meter
No points
~ 0
5
51
63
314
367
663
681
Query Region
0
25
50
100 Meters


Change In Density
(2016 - 2010), Pts. / Sq. Meter
-111
6
49
88
195
332
511
0
25
50
100 Meters

Conclusions

- DGGS is an *ideal* reference frame for:
  - Indexing, clustering, & querying (**nD**) global point clouds
  - Integrating point clouds from heterogeneous CRS's
  - Conducting temporal analysis
  - Analyzing observations at any arbitrary discrete LOD
  - Limiting accumulated error in geospatial data
  - Geo-statistical analysis
  - Smooth point cloud visualization
  - Scalable geospatial data analytics → array process

LIVE DEMO


Draw Box
Query
Upload Tileset
Reset