# Subsampled coordinates in CF-netCDF

CF Community Meeting 2020

L. Gaultier, D. Hassell, S. Herlédan, A. Jelenak,

T. Lavergne, D. Lee, A. Soerensen

# Content

1. The problem: Coordinates use too much data volume!
2. Our proposed solution: Parameterized coordinate interpolation
3. Sources of inspiration
4. Proposed solution in CF
5. Open issues & way forward

# 1. The problem

- Coordinates can comprise a significant amount of data volume in products
- This is particularly true for products that don't have their observations on a regular grid
- This has been a problem in remote sensing for a long time, and can be a problem in other places, e.g. in the world of unstructured grids

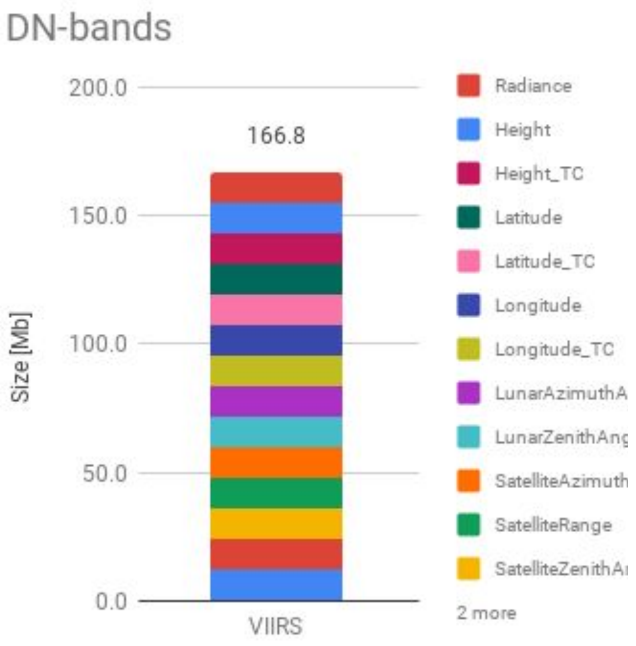
# 1. The problem: An example

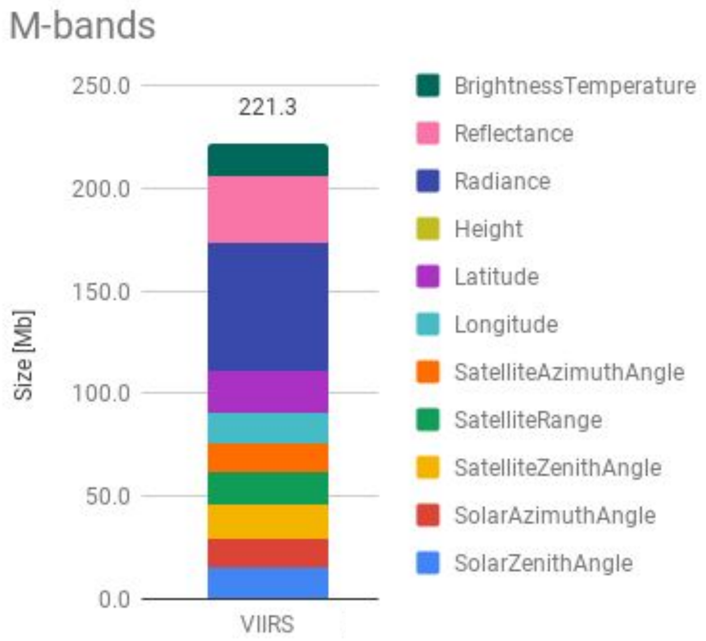
- Instrument characteristics can affect spatial distribution of observations
- A given set of coordinates typically is unique to that observation - they cannot be reused


Source: OceanDataLab

# 1. The problem: An example

- Coordinates have a high information content and expose the limits of off-the-shelf compression
- Example satellite data products (right) demonstrates amount of information needed to interpret L1 product



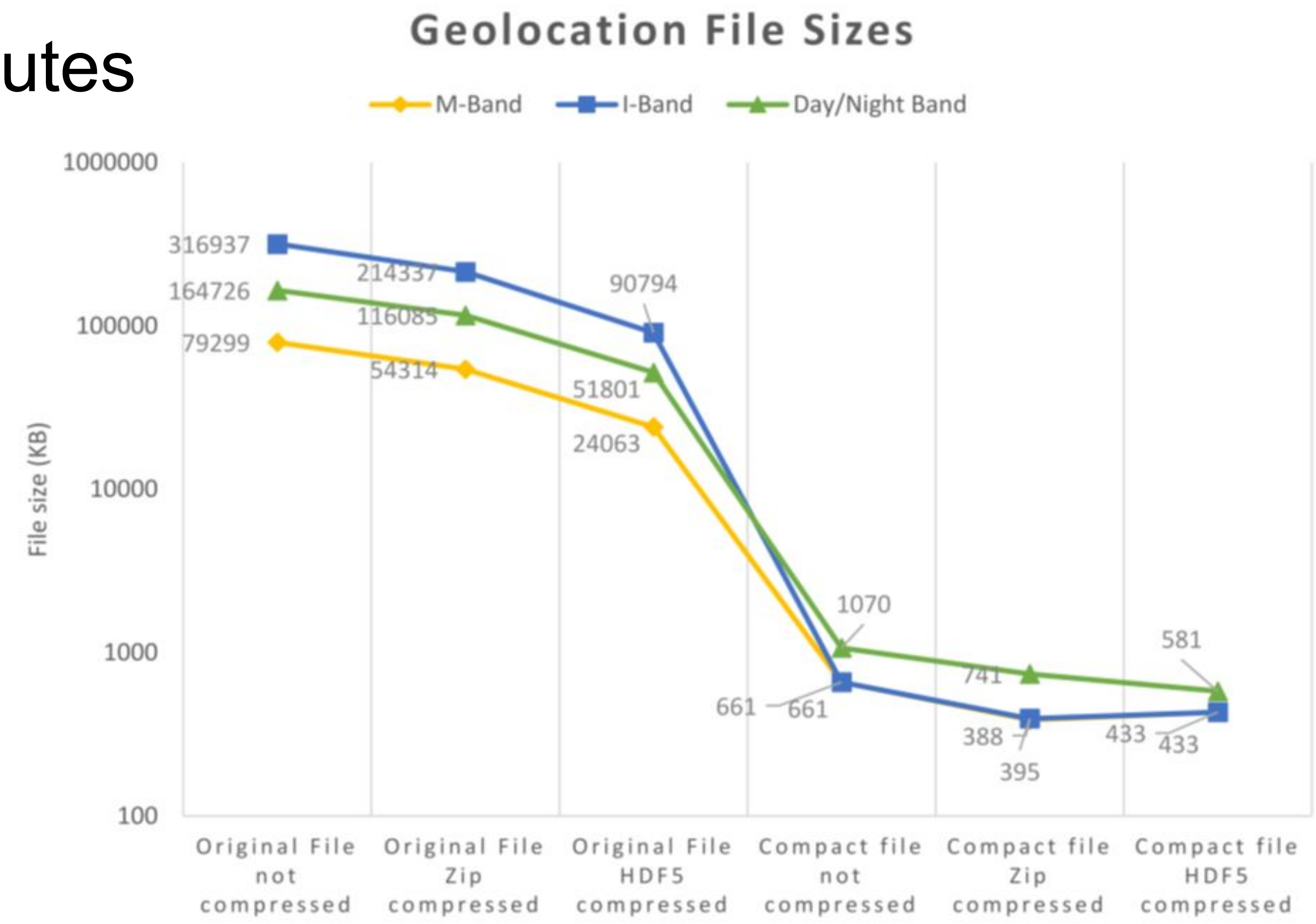



EUMETSAT

# 2. Our solution: Parameterized coordinate interpolation

>0.5GB/3 minutes

= 😵💀




We can beat log scale and outperform off-the-shelf compression! 🤓🥰



EUMETSAT

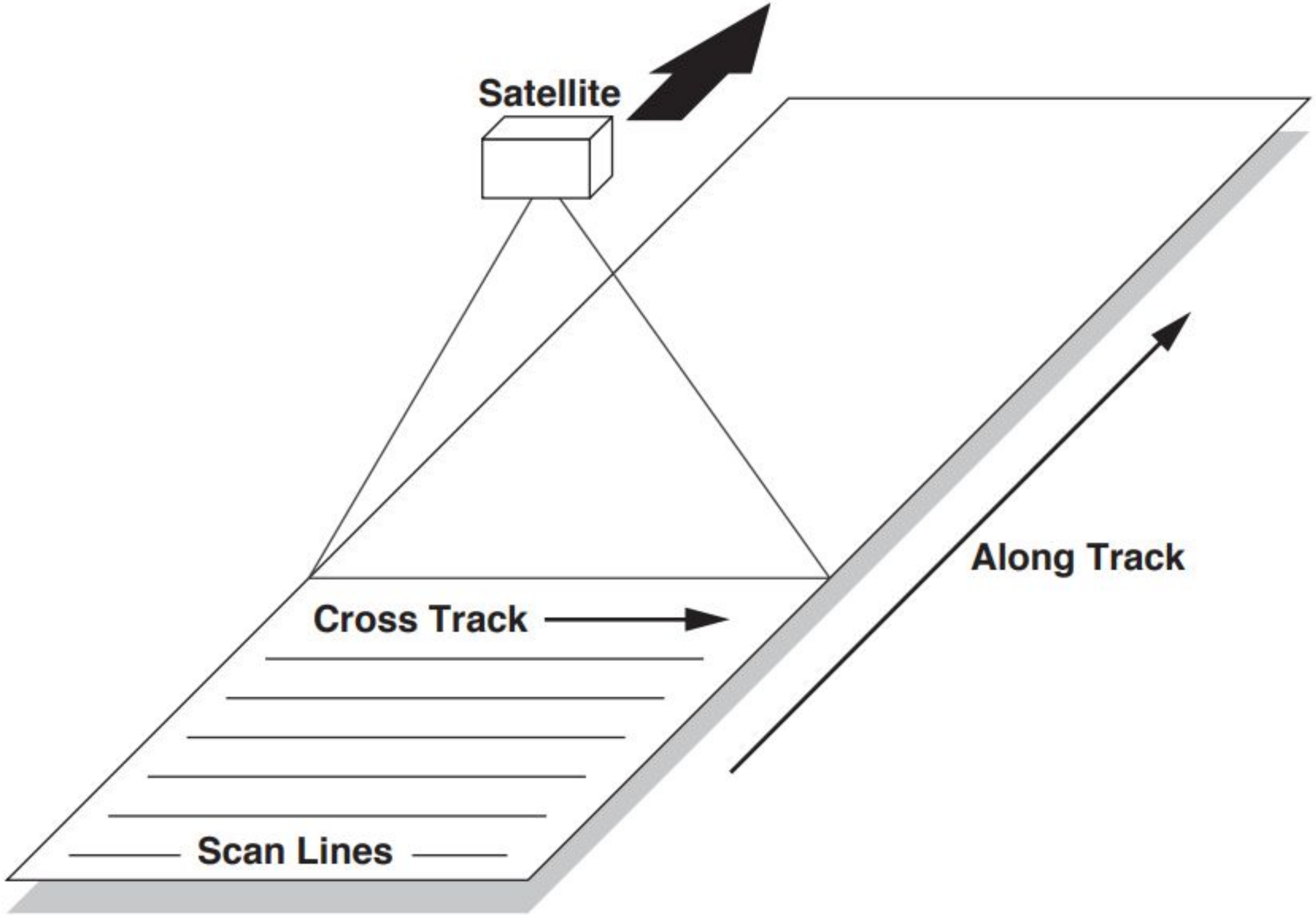
# 2. Our solution: Parameterized coordinate interpolation

- Accommodate:
  - Gridded and ungridded data
  - Regular and irregular data point distributions
  - Need for reusability for multiple variables, potentially on different resolutions
  - Existing practices from "thinning" data to more complex interpolation and extrapolation schemes

# 2. Our solution: Parameterized coordinate interpolation

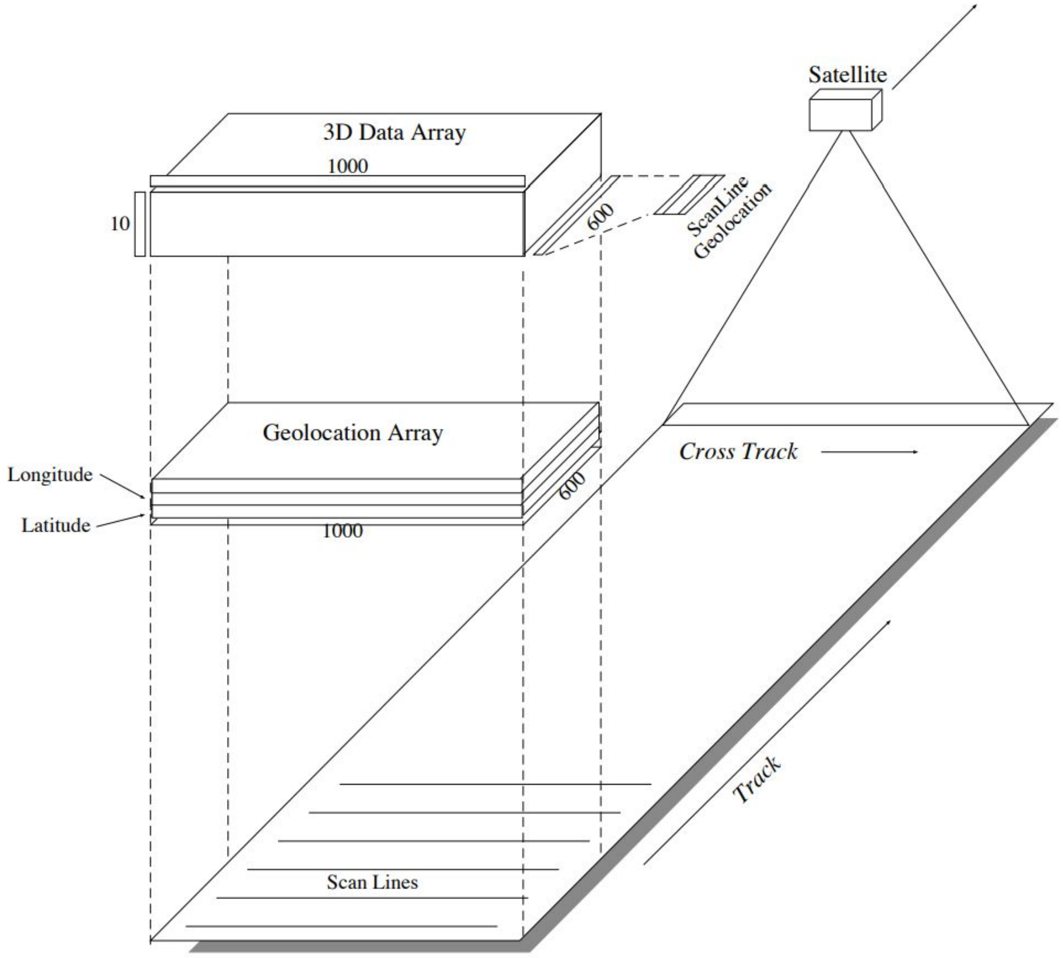
- The approach:
  - Re-use and generalise existing practices
  - "Compact" coordinates by providing a set of coordinates ("tie points") from which the "uncompacted" coordinates can be recovered
  - Stay compatible with CF Data Model, taking inspiration from compression by gathering, grid mappings, reduced horizontal grids, discrete sampling geometries, geometry cell bounds

# 3. Inspiration: Some terms




Source: NASA

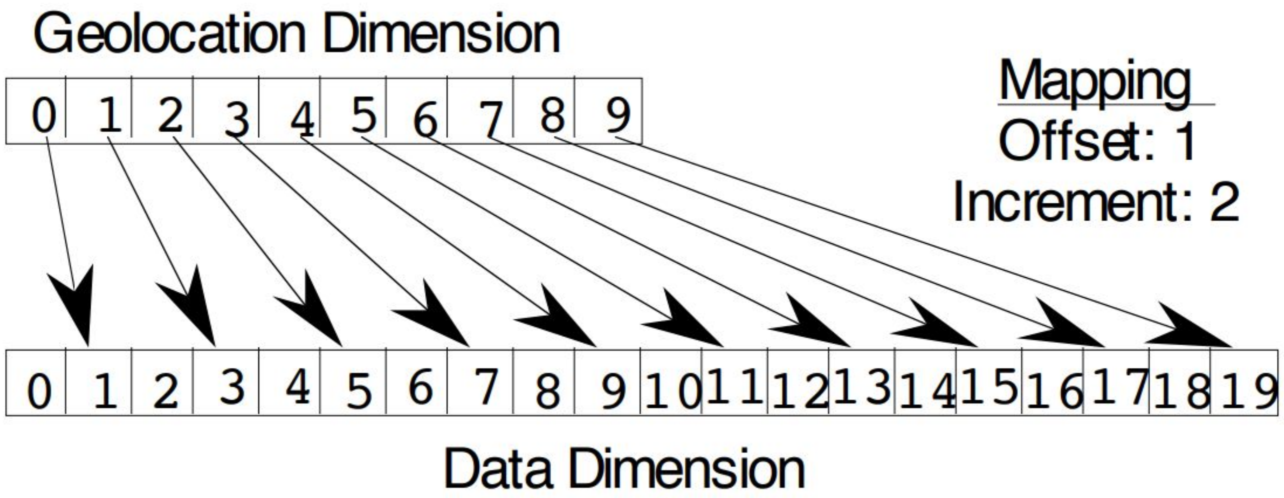
# 3. Inspiration: Some terms




Source: NASA

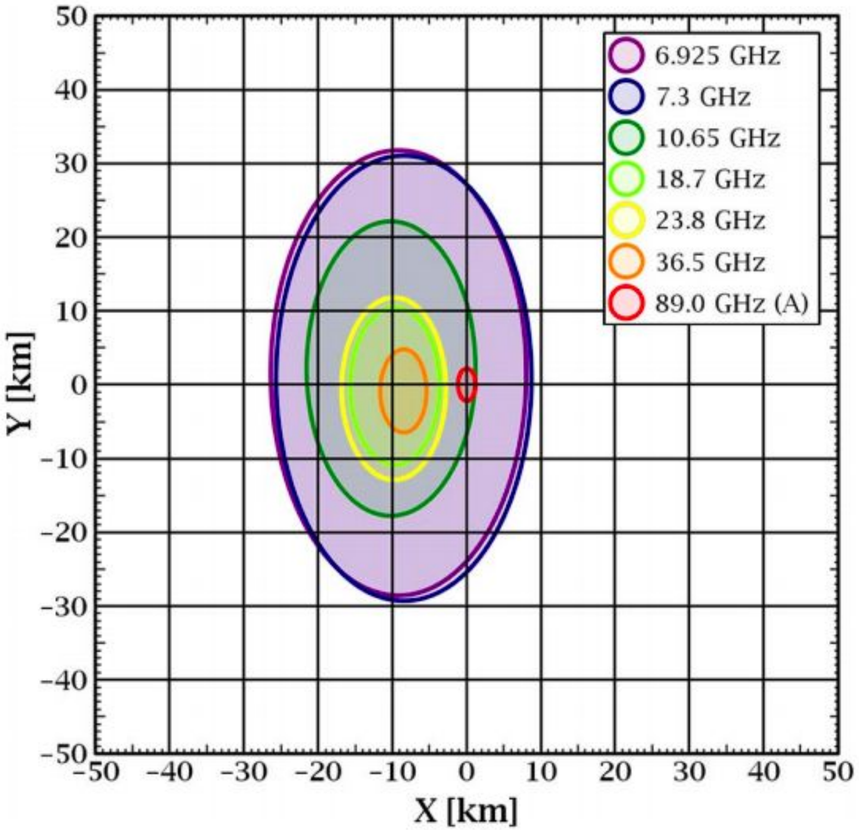
# 3. Inspiration: HDF-EOS

- Provide a "thinned" set of coordinates that can be used to interpolate / extrapolate the remaining data points



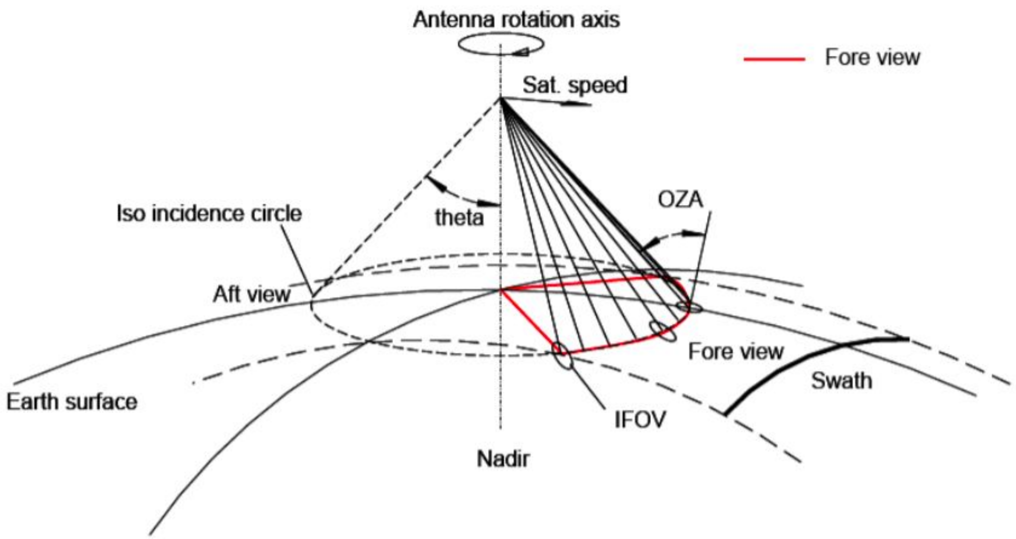

Source: NASA

# 3. Inspiration: Microwave imagers




GCOM-W AMSR2 channel footprints.

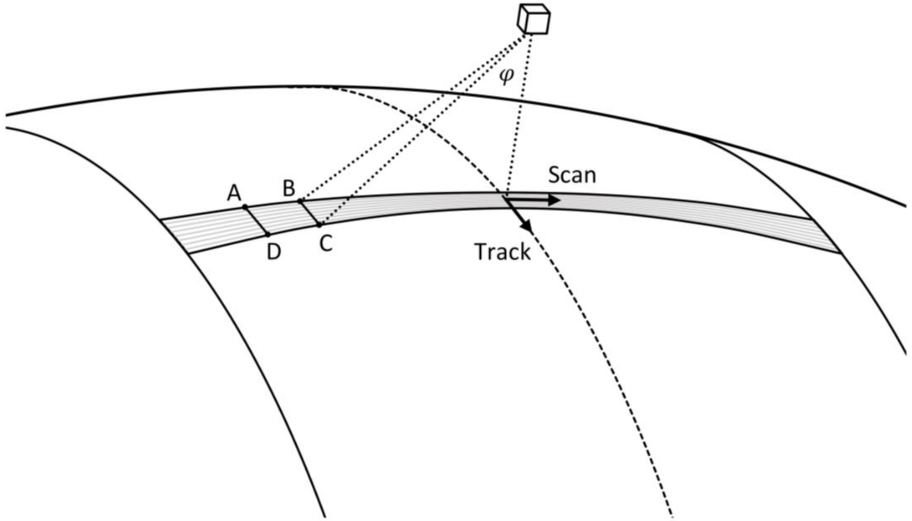
Source: [Maeda et al.]

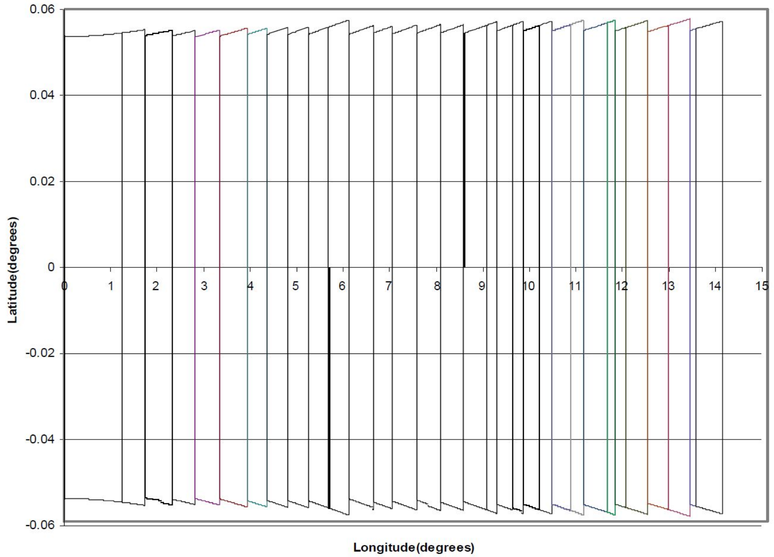



EPS-SG MWI scanning principle.

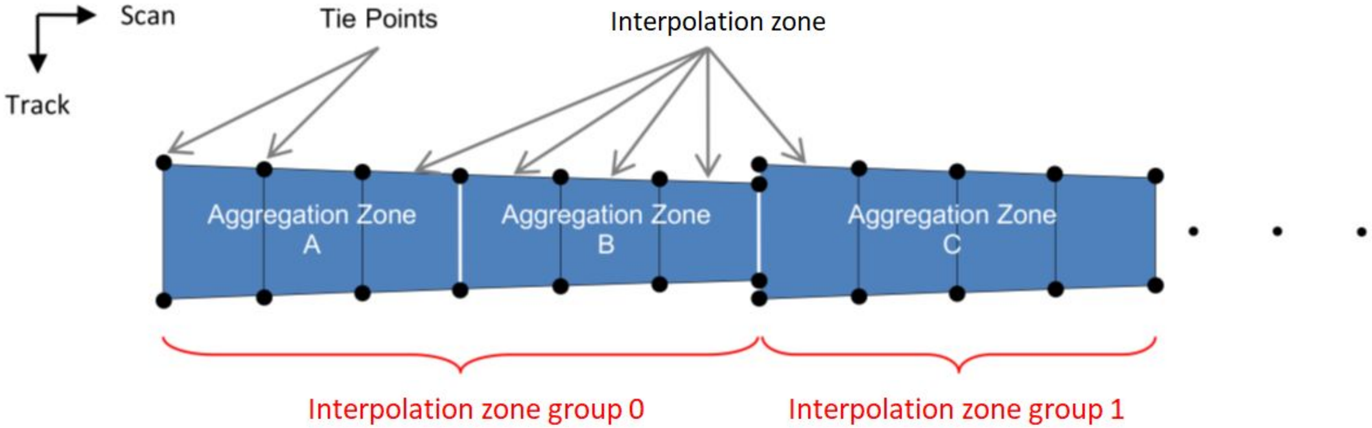
- Provide a set of reference coordinates that can be used to extrapolate the remaining data points based on knowledge of the instrument

# 3. Inspiration: VIIRS



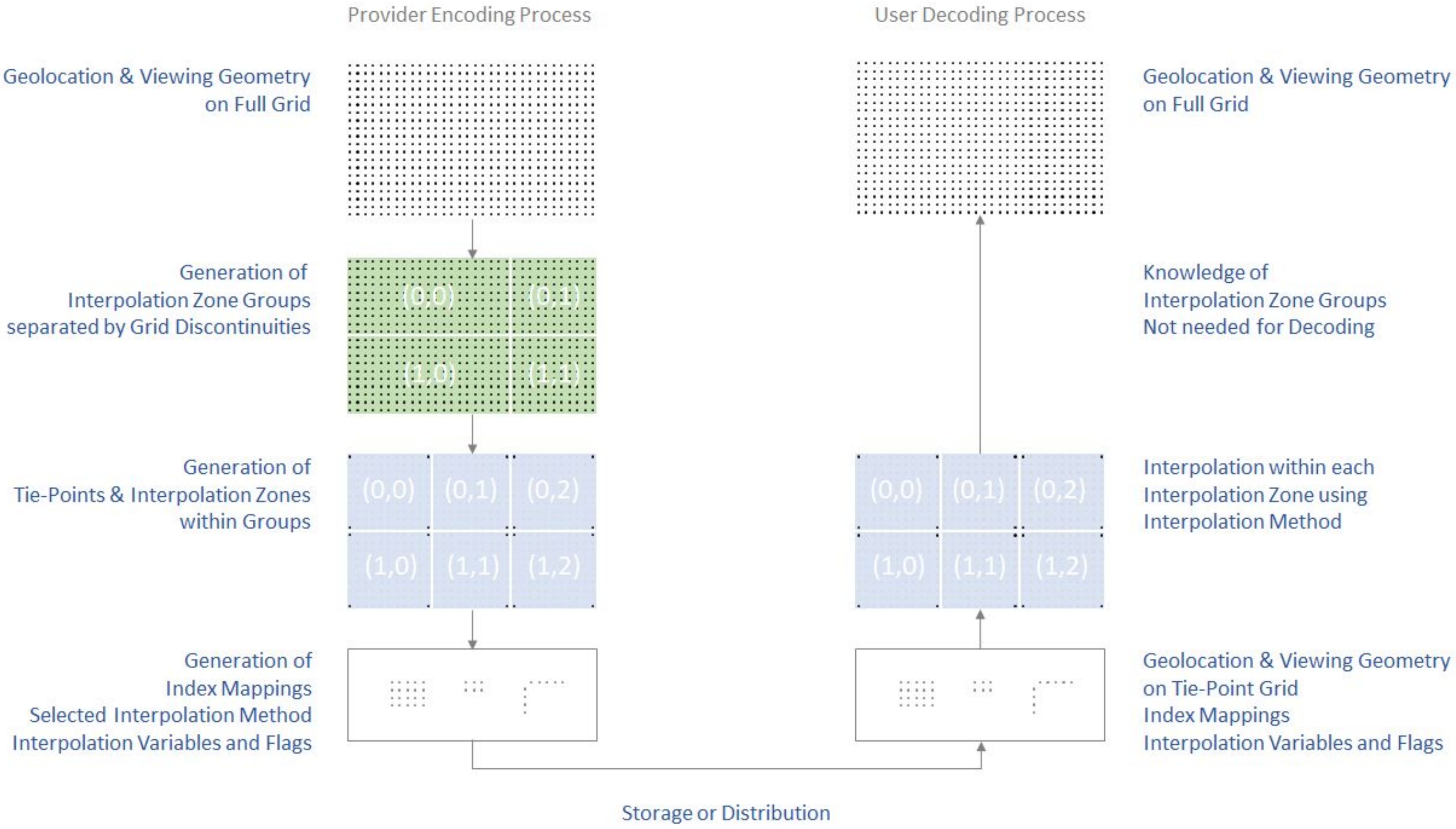

- Bowtie effect is compensated by aggregating views in different "aggregation zones"






EUMETSAT

# 4. Solution: Overview



# 4. Solution: Straightforward example

```
netcdf NDVI_example_compacted {
dimensions:
  // NDVI product grid
  lat = 6 ;
  lon = 5 ;
  // Tie points (tp)
  tp_lat = 2 ;
  tp_lon = 2 ;

variables:
  float ndvi(lat, lon) ;
    ndvi : standard_name = "normalized_difference_vegetation_index" ;
    ndvi : interpolation = "tp_interpolation" ;
    ndvi : interpolation_indices = "lat:lat_indices lon:lon_indices" ;
  int lat_indices(tp_lat) ;
  int lon_indices(tp_lon) ;

  char tp_interpolation ;
    tp_interpolation : interpolation_name = "bi_linear" ;
    tp_interpolation : location_tie_points = "lat lon" ;
  float lat(tp_lat) ;
    lat : standard_name = "latitude" ;
    lat : units = "degrees_north" ;
  float lon(tp_lon) ;
    lon : standard_name = "longitude" ;
    lon : units = "degrees_east" ;

data:
  ndvi =
    0.5119157, 0.04983568, 0.5414233, 0.3076001, 0.8931185,
    0.8581991, 0.7848567, 0.2485297, 0.9762608, 0.4546139,
    0.1063213, 0.8751125, 0.9819403, 0.9346204, 0.2765055,
    0.2011242, 0.7634977, 0.7657007, 0.3465044, 0.9491135,
    0.9431587, 0.04104269, 0.5652257, 0.5340118, 0.8907427,
    0.3514801, 0.1451995, 0.1523716, 0.1563433, 0.7384073 ;
  lat_indices = 0, 5 ;
  lon_indices = 0, 4 ;
  lat = 10, 20 ;
  lon = 10, 30 ;
}
```

Uncompacted grid is 6x5

Payload is on uncompacted grid

Pointer to interpolation description

Interpolation indices declared instead of coordinates

Indices linked to uncompacted dimensions

Interpolation method & tie points defined on compacted grid

Coordinates same as usual, but with fewer data points

1 interpolation zone, 1 interpolation zone group, 4 corners

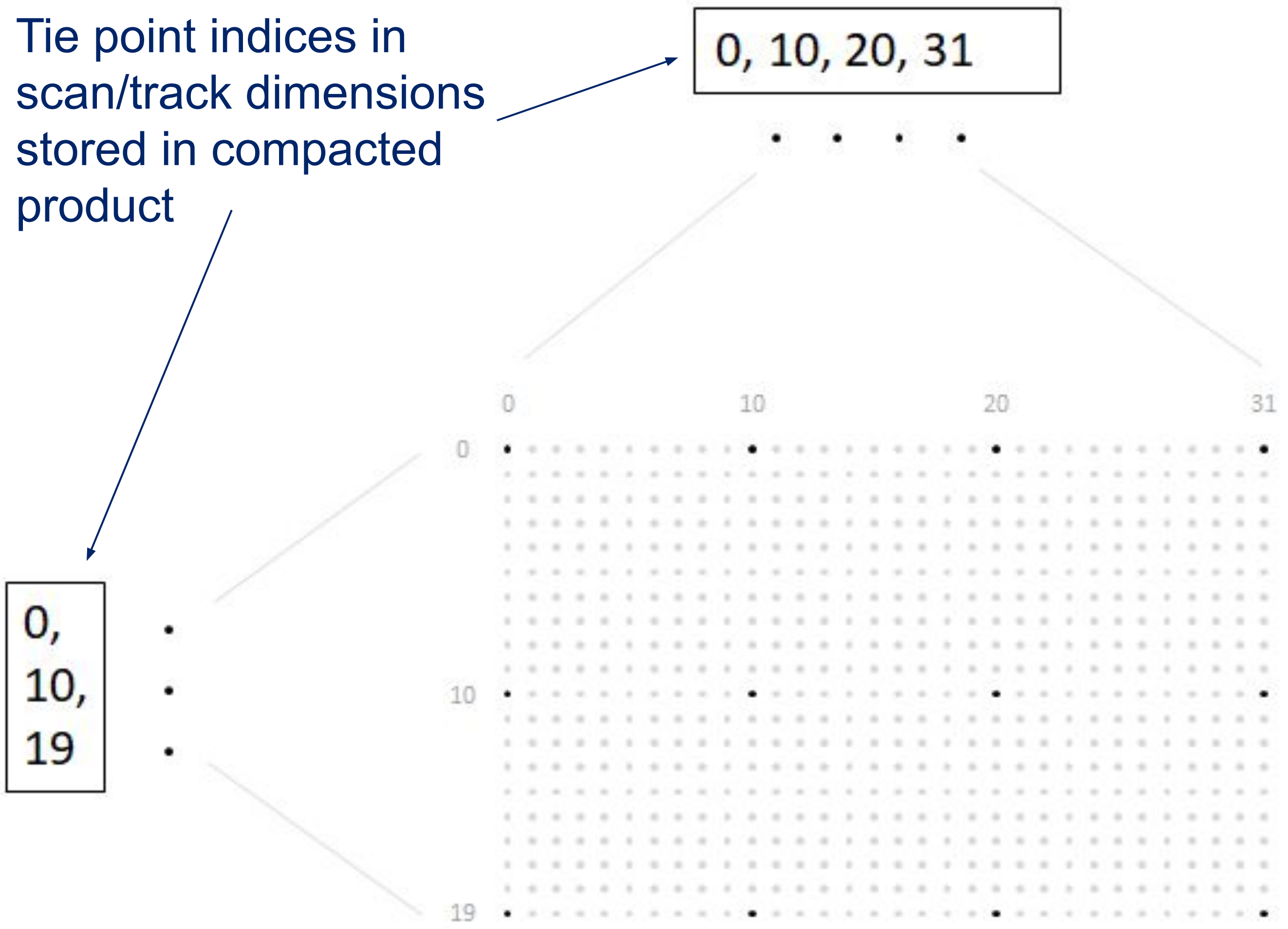
Current version on GitHub



EUMETSAT

# 4. Solution: Straightforward example

- 4 points stored rather than 640
- Tie point centres are collocated with corresponding views in uncompacted product
- Indices map between compacted and uncompacted dimensions






EUMETSAT

# 4. Solution: Example with grid mapping

```
netcdf NDVI_grid_mapping_example_compacted {
dimensions:
  // NDVI product grid
  y = 6 ;
  x = 5 ;
  // Tie points (tp)
  tp_y = 2 ;
  tp_x = 2 ;

variables:
  float ndvi(lat, lon) ;
    ndvi : standard_name = "normalized_difference_vegetation_index" ;
    ndvi : interpolation = "tp_interpolation" ;
    ndvi : interpolation_indices = "y_indices x_indices" ;
    ndvi : grid_mapping = "crs" ;

  int y_indices(tp_y) ;
    y_indices : interpolation_dimension = "y" ;
  int x_indices(tp_x) ;
    x_indices : interpolation_dimension = "x" ;

  char crs ;
    crs : grid_mapping_name = "lambert_conformal_conic";
    crs : standard_parallel = 25.0;
    crs : longitude_of_central_meridian = 265.0;
    crs : latitude_of_projection_origin = 25.0;

  char tp_interpolation ;
    tp_interpolation : interpolation_name = "bi_linear" ;
    tp_interpolation : location_tie_points = "y x" ;
  float y(tp_y) ;
    y : standard_name = "projection_y_coordinate" ;
    y : units = "km" ;
  float x(tp_x) ;
    x : standard_name = "projection_x_coordinate" ;
    x : units = "km" ;
}
```

Reference to grid mapping

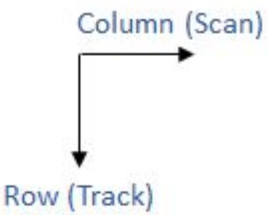
Grid mapping included as usual

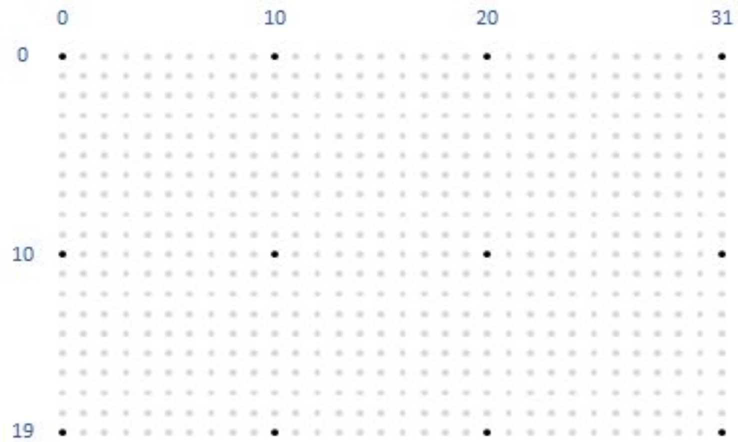
Current version on GitHub



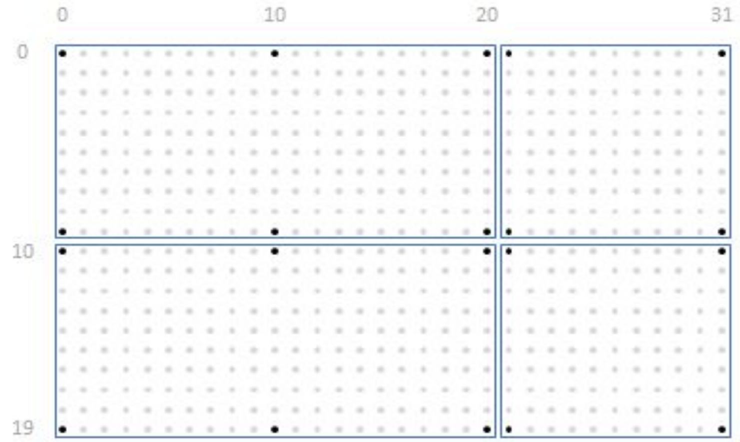
EUMETSAT

# 4. Solution: Generalisations

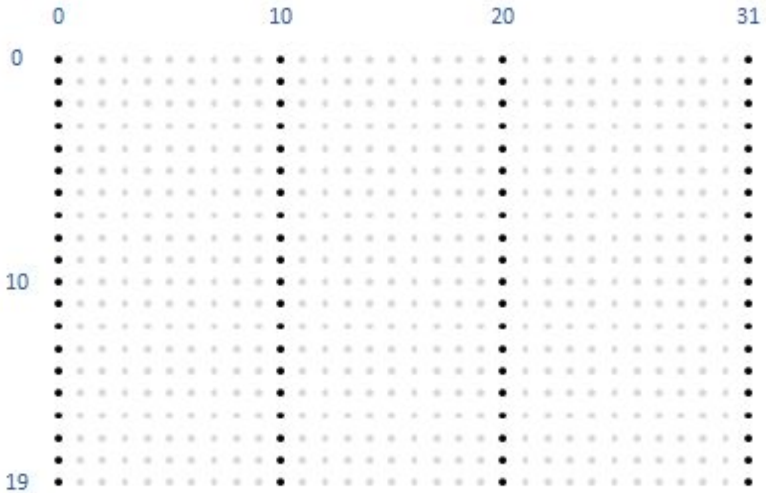


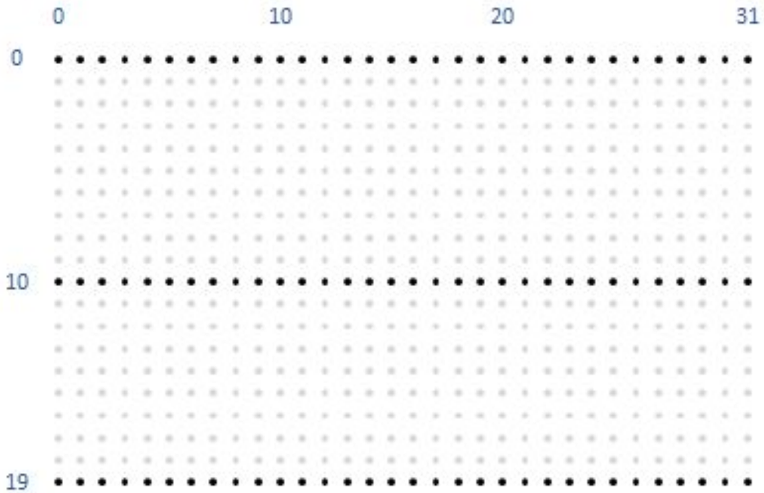

2D interpolation




2D interpolation with interpolation groups



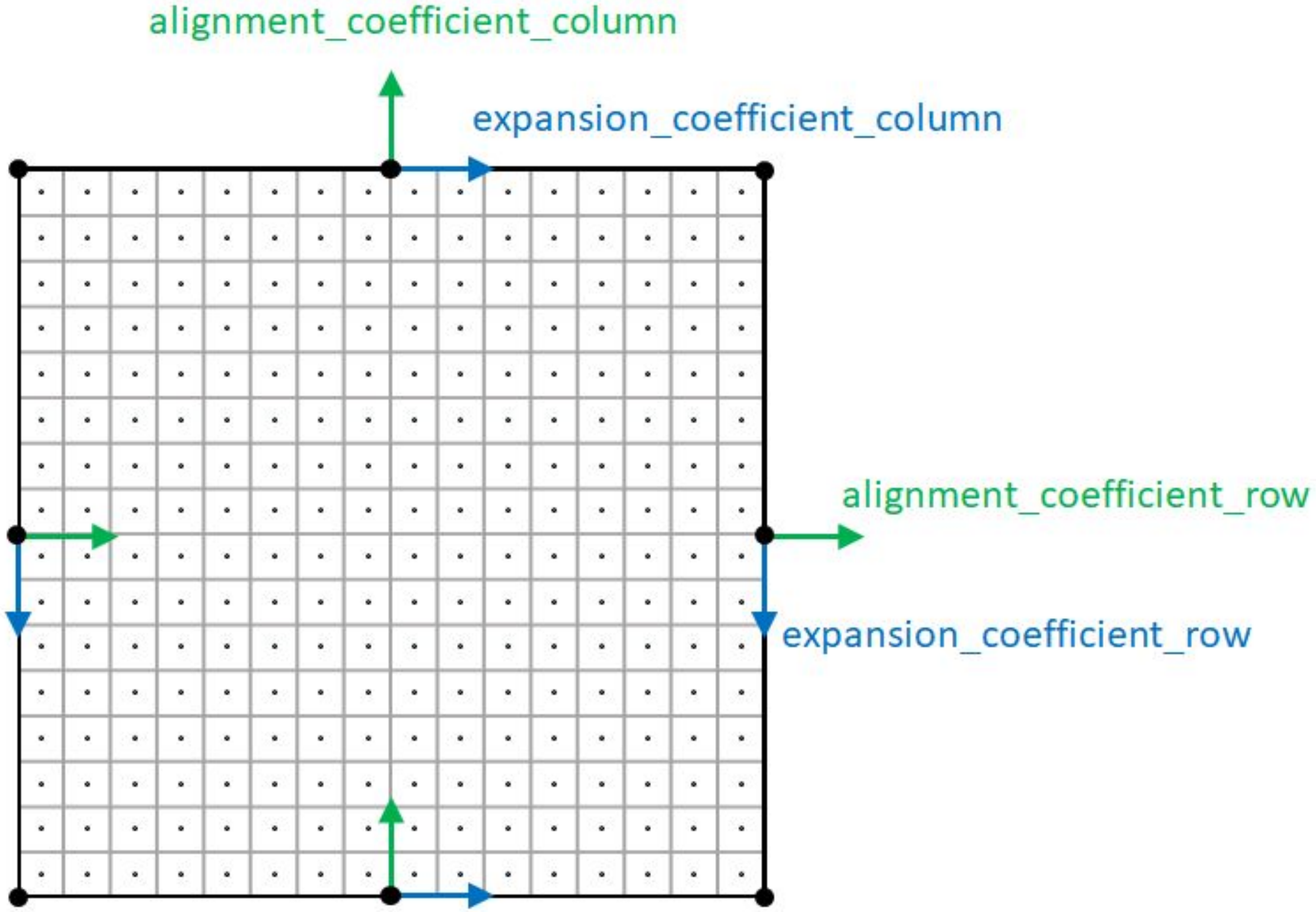

Cross-track interpolation




Along-track interpolation

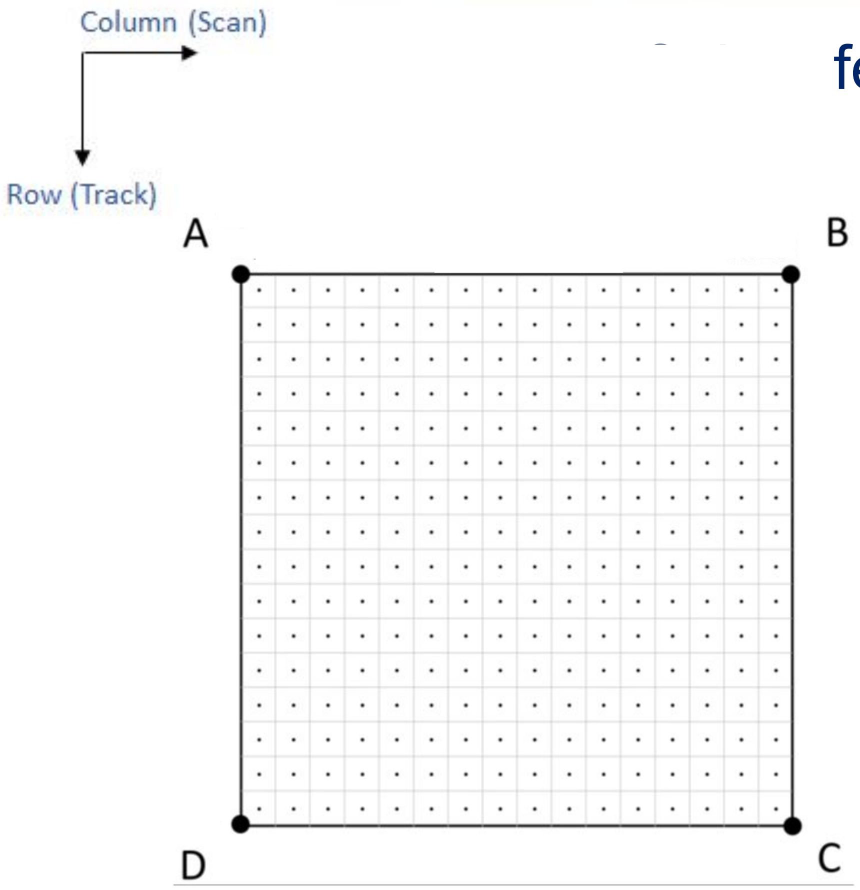
# 4. Solution: Parameterisation

- Sometimes linear interpolation is not precise enough
- This applies especially for remote sensing data, where observations' positions are determined by the intersection of the satellite's view and the point it views
- We can capture this complexity with 4 parameters per interpolation zone group that are applied to each interpolation zone within the group

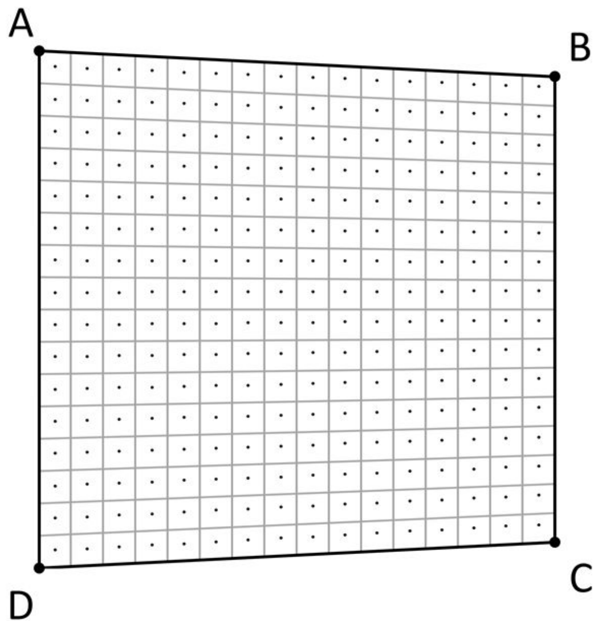


# 4. Solution: Dive into parameters

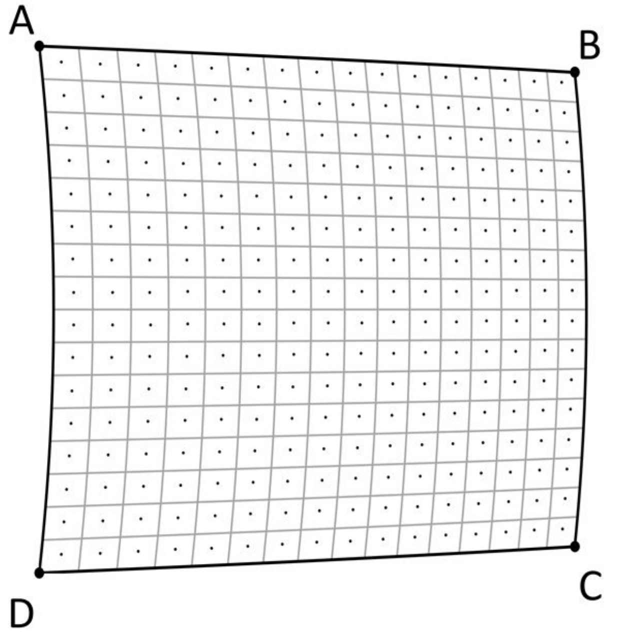
feature added: Offset between tie point and view centres




Default coefficients




Default coefficients
Tie points adapted to data grid




Alignment & expansion
coefficients set

# 4. Solution: Fully parameterized example

```
netcdf VIIRS_M_and_I_Band_example_compacted {
dimensions :
  // VIIRS M-Band (750 m resolution imaging)
  m_track = 768 ;
  m_scan = 3200 ;
  m_channel = 16 ;

  // VIIRS I-Band (375 m resolution imaging)
  i_track = 1536 ;
  i_scan = 6400 ;
  i_channel = 5 ;

  // Tie points and interpolation zones (shared between VIIRS M-Band and I-Band)
  tp_track = 96 ;
  tp_scan = 205 ;
  track_interpolation_zone = 48 ;
  scan_interpolation_zone = 200 ;

  // Time, stored at scan-start and scan-end of each scan
  time_scan = 2;
```

Now interpolation is applied to 2 variables

Multiple interpolation zones provided

Time for individual observations is interpolated too

Current version on GitHub

# 4. Solution: Fully parameterized example

```
variables:

  // VIIRS M-Band
  float m_radiance(m_track, m_scan, m_channel) ;
    m_radiance : interpolation = "tp_interpolation time_interpolation" ;
    m_radiance : interpolation_indices = "m_track: m_track_indices m_scan:m_scan_indices" ;
    m_radiance : interpolation_offsets = "m_track: 0.5 m_scan: 0.5" ;   // unit is cells
  int m_track_indices(tp_track) ;
  int m_scan_indices(tp_scan) ;

  // VIIRS I-Band
  float i_radiance(i_track, i_scan, i_channel) ;
    i_radiance : interpolation = "tp_interpolation time_interpolation" ;
    i_radiance : interpolation_indices = "i_track: i_track_indices i_scan:i_scan_indices" ;
    i_radiance : interpolation_offsets = "i_track: 0.5 i_                unit is cells
  int i_track_indices(tp_track) ;
  int i_scan_indices(tp_scan) ;

  // Reusable interpolation containers for space and time shared by VIIRS M-Band and I-Band
  char tp_interpolation ;
    tp_interpolation : interpolation_name = "bi_quadratic" ;
    tp_interpolation : interpolation_coefficients = "expansion_coefficient_track alignment_coefficient_track expansion_coefficient_scan alignment_coefficient_scan" ;
    tp_interpolation : interpolation_flags = "interpolation_zone_flags" ;
    tp_interpolation : location_tie_points = "lat lon" ;
    tp_interpolation : sensor_direction_tie_points = "sen_azi_ang sen_zen_ang" ;
    tp_interpolation : solar_direction_tie_points = "sol_azi_ang sol_zen_ang" ;
  char time_interpolation ;
    time_interpolation : interpolation_name = "bi_linear" ;
    time_interpolation : time_tie_points = "t" ;
```

Tie point offsets from pixel centres

Interpolation container shared across variables

New interpolation name

Warping coefficients supplied

Flags supplied per interpolation zone

Time interpolation container can be simpler

Current version on GitHub



EUMETSAT

# 4. Solution: Fully parameterized example

```
  // Tie points
  float lat(tp_track, tp_scan) ;
    lat : standard_name = "latitude" ;
    lat : units = "degrees_north" ;
  float lon(tp_track, tp_scan) ;
    lon : standard_name = "longitude" ;
    lon : units = "degrees_east" ;
  float sen_azi_ang(tp_track, tp_scan) ;
    sen_azi_ang : standard_name = "sensor_azimuth_angle" ;
    sen_azi_ang:units = "degrees" ;
  float sen_zen_ang(tp_track, tp_scan) ;
    sen_zen_ang : standard_name = "sensor_zenith_angle" ;
    sen_zen_ang : units = "degrees" ;
  float sol_azi_ang(tp_track, tp_scan) ;
    sol_azi_ang : standard_name = "solar_azimuth_angle" ;
    sol_azi_ang : units = "degrees" ;
  float sol_zen_ang(tp_track, tp_scan) ;
    sol_zen_ang : standard_name = "solar_zenith_angle" ;
    sol_zen_ang : units = "degrees" ;

  // Interpolation coefficients and flags
  short expansion_coefficient_track(track_interpolation_zone, tp_scan) ;
  short alignment_coefficient_track(track_interpolation_zone, tp_scan) ;
  short expansion_coefficient_scan(tp_track, scan_interpolation_zone) ;
  short alignment_coefficient_scan(tp_track, scan_interpolation_zone) ;
  byte interpolation_zone_flags(track_interpolation_zone, scan_interpolation_zone) ;
    interpolation_zone_flags:valid_range = "1b, 7b" ;
    interpolation_zone_flags:flag_masks = "1b, 2b, 4b" ;
    interpolation_zone_flags:flag_meanings = "location_use_cartesian  sensor_direction_use_cartesian  solar_direction_use_cartesian" ;

  // Time
  double t(tp_track, time_scan) ;
    t:long_name = "time" ;
    t:units = "days since 1990-1-1 0:0:0" ;
}
```

Formerly, all of these auxiliary coordinate variables were encoded in full!

Current version on GitHub



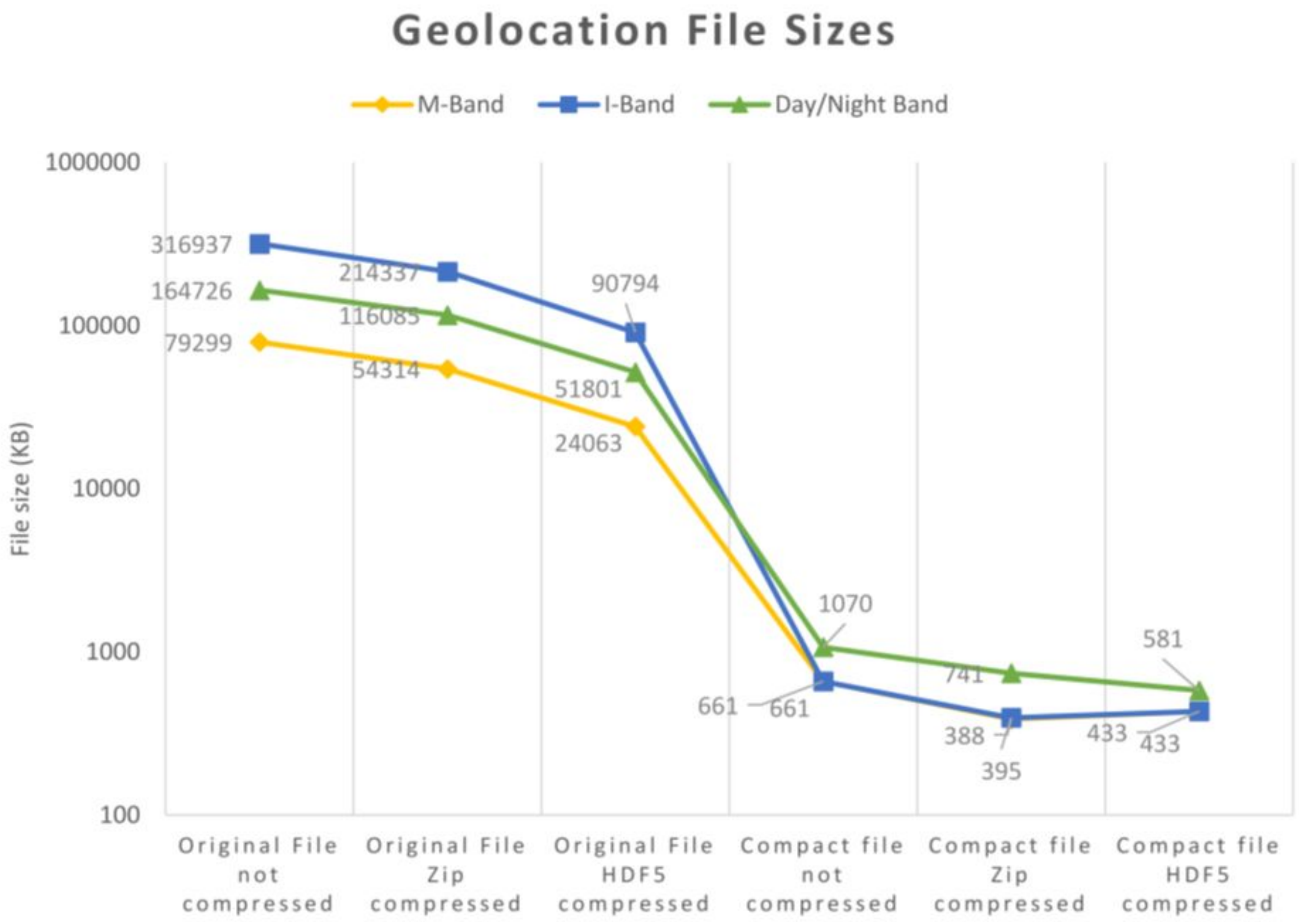
EUMETSAT

# 5. Open issues: Questions

- Can we demonstrate using this approach for extrapolated coordinates such as the microwave imaging examples?
- All relevant use cases covered?
- How can this be combined with terrain correction? (this will probably be the subject of a separate, future proposal)

# 5. Open issues: Way forward

- Detailed explanation probably out of scope of text changes in Conventions, therefore we will need to publish and reference methodology
- Finalisation of naming conventions, CDL structure, etc. needed
- We will produce a full set of sample data
- Possibility: Release of reference software for compacting and uncompacting products




# Interested? [Get involved on GitHub!]



EUMETSAT