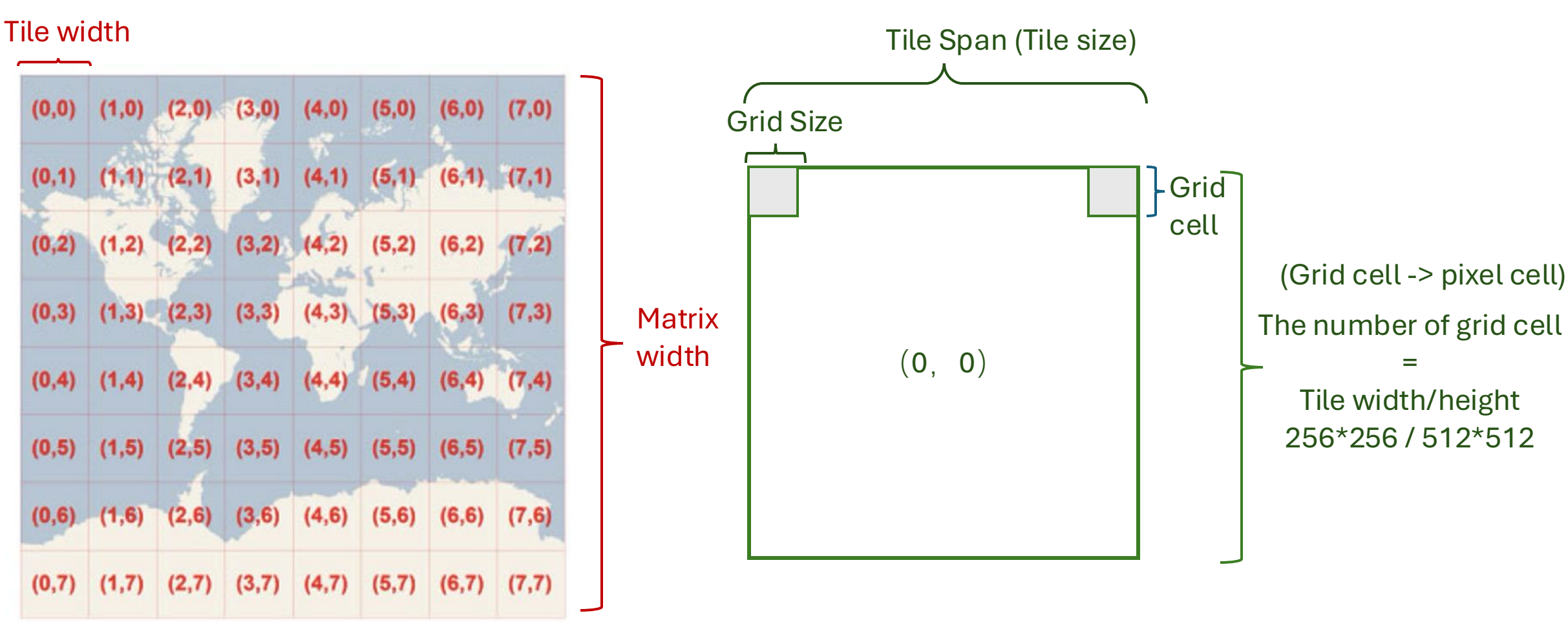
AI-Driven Mapping Analysis

Tiling system

References:

- Microsoft Bing Tiling System
- Open Geospatial Consortium



Given a pair of pixel XY coordinates, you can easily determine the tile XY coordinates of the tile containing that pixel:

```
tileX = floor(pixelX / 256)
```

```
tileY = floor(pixelY / 256)
```

Reference: https://docs.ogc.org/is/17-083r4/17-083r4.html#toc15

Tile Size /Matrix Size Calculation

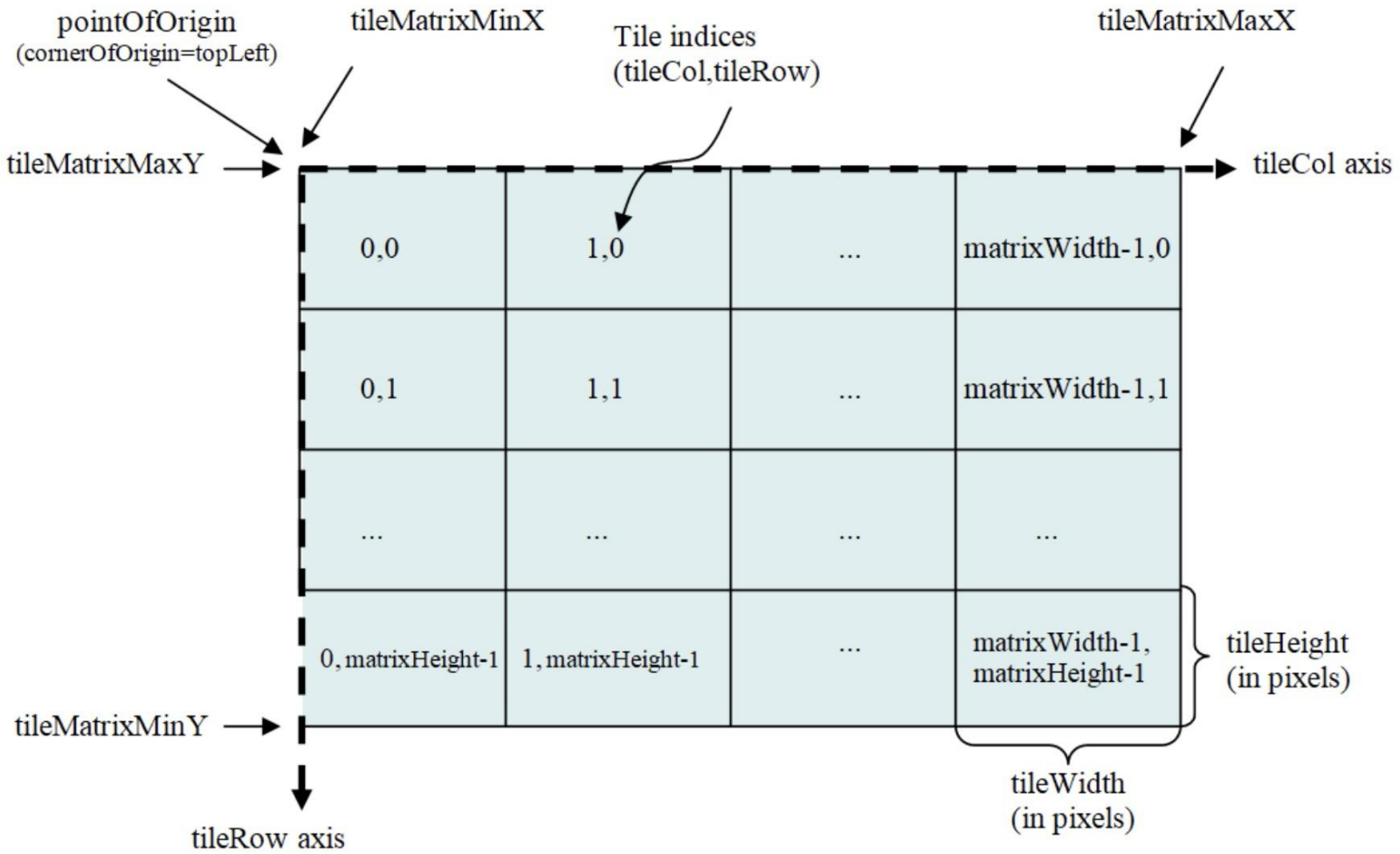
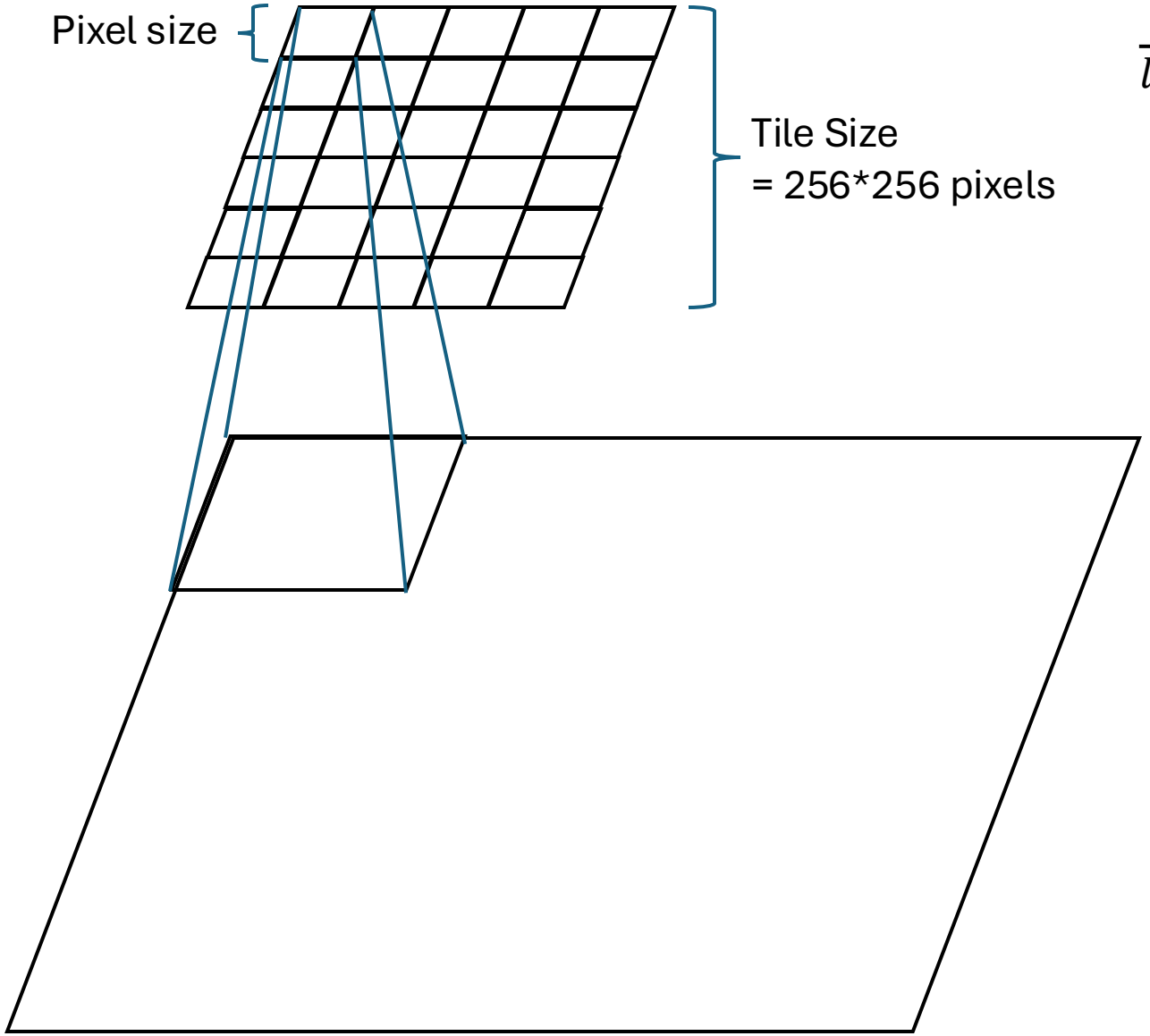


Figure 5 – Tile Space (the corner of origin is topLeft)

Tile Size /Matrix Size Calculation



$$\frac{each\ pixel\ size\ on\ map}{length\ on\ the\ real\ world\ per\ pixel} = \frac{1}{Scale\ Denominator}$$

Tile Size /Matrix Size Calculation

We use the tile in Spherical Mercato projection (EPSG:3857)

For the case of a two-dimensional space, given the top left point of the tile matrix in CRS coordinates (tileMatrixMinX, tileMatrixMaxY), the width and height of the tile matrix in tile units (matrixWidth, matrixHeight), the *rendering cells in a tile* values (tileWidth, tileHeight), the coefficient to convert the coordinate reference system (CRS) units into meters (metersPerUnit), and the scale (1:scaleDenominator), the bottom right corner of the bounding box of a tile matrix (tileMatrixMaxX, tileMatrixMinY) can be calculated as follows:

Rendering pixel: 0.28mm * 0.28mm, tileWidth = 256

$$\text{cellSize} = \text{scaleDenominator} \times 0.2810^{-3} / \text{metersPerUnit}(crs)$$

$$\text{tileSpanX} = \text{tileWidth} \times \text{cellSize}$$

tileSpanX = tileWidth * (scaleDenominator * 0.28 * 10^{-3} /metersPerUnit)
= pixel count * pixel size on map * SD
= 256 * 0.28 * 10^{-3} * SD

$$\text{tileSpanY} = \text{tileHeight} \times \text{cellSize}$$

$$\text{tileMatrixMaxX} = \text{tileMatrixMinX} + \text{tileSpanX} \times \text{matrixWidth}$$

$$\text{tileMatrixMinY} = \text{tileMatrixMaxY} - \text{tileSpanY} \times \text{matrixHeight}$$

NOTE 7 In a CRS with coordinates expressed in meters, metersPerUnit(*crs*) equals 1.

Since the tile is in Spherical Mercato projection (EPSG:3857), metersPerUnit(crs) equals 1.

NOTE 8 In CRS with coordinates expressed in degrees metersPerUnit(*crs*) equals 360/(EquatorialRadius * 2 * PI) (360 degrees are equivalent to the EquatorialPerimeter). E.g for WGS84 metersPerUnit(*crs*) is 111319.4908 meters/degree.

SD = Ground resolution

Ground Resolution and Map Scale

In addition to the projection, the ground resolution or map scale must be specified in order to render a map. At the lowest level of detail (Level 1), the map is 512 x 512 pixels. At each successive level of detail, the map width and height grow by a factor of 2: Level 2 is 1024 x 1024 pixels, Level 3 is 2048 x 2048 pixels, Level 4 is 4096 x 4096 pixels, and so on. In general, the width and height of the map (in pixels) can be calculated as:

$$\text{map width} = \text{map height} = 256 * 2^{\text{level}} \text{ pixels}$$

The **ground resolution** indicates the distance on the ground that's represented by a single pixel in the map. For example, at a ground resolution of 10 meters/pixel, each pixel represents a ground distance of 10 meters. The ground resolution varies depending on the level of detail and the latitude at which it's measured. Using an earth radius of 6378137 meters, the ground resolution (in meters per pixel) can be calculated as:

$$\text{ground resolution} = \cos(\text{latitude} * \text{pi}/180) * \text{earth circumference} / \text{map width}$$

$$= (\cos(\text{latitude} * \text{pi}/180) * 2 * \text{pi} * 6378137 \text{ meters}) / (256 * 2^{\text{level}} \text{ pixels})$$

The **map scale** indicates the ratio between map distance and ground distance, when measured in the same units. For instance, at a map scale of 1 : 100,000, each inch on the map represents a ground distance of 100,000 inches. Like the ground resolution, the map scale varies with the level of detail and the latitude of measurement. It can be calculated from the ground resolution as follows, given the screen resolution in dots per inch, typically 96 dpi:

$$\text{map scale} = 1 : \text{ground resolution} * \text{screen dpi} / 0.0254 \text{ meters/inch}$$

$$= 1 : (\cos(\text{latitude} * \text{pi}/180) * 2 * \text{pi} * 6378137 * \text{screen dpi}) / (256 * 2^{\text{level}} * 0.0254)$$

Tile Matrix Set

6.1.2. Tile Matrix Set

Depending on the range of scales needed to be represented in the screen of a client, a single tile matrix is impractical and might force the software to spend too much time simplifying/generalizing the dataset prior to rendering.

Commonly, several tile matrices are progressively defined covering the expected ranges of scales needed for the application. A *Tile Matrix Set* is a tile scheme composed of a collection of tile matrices, optimized for a particular scale and identified by a tile matrix identifier. Each Tile Matrix Set has an optional approximated bounding box, but each tile matrix has an exact bounding box that is deduced indirectly from other parameters. Tile matrix bounding boxes at each scale will usually vary slightly due to their cell alignment.

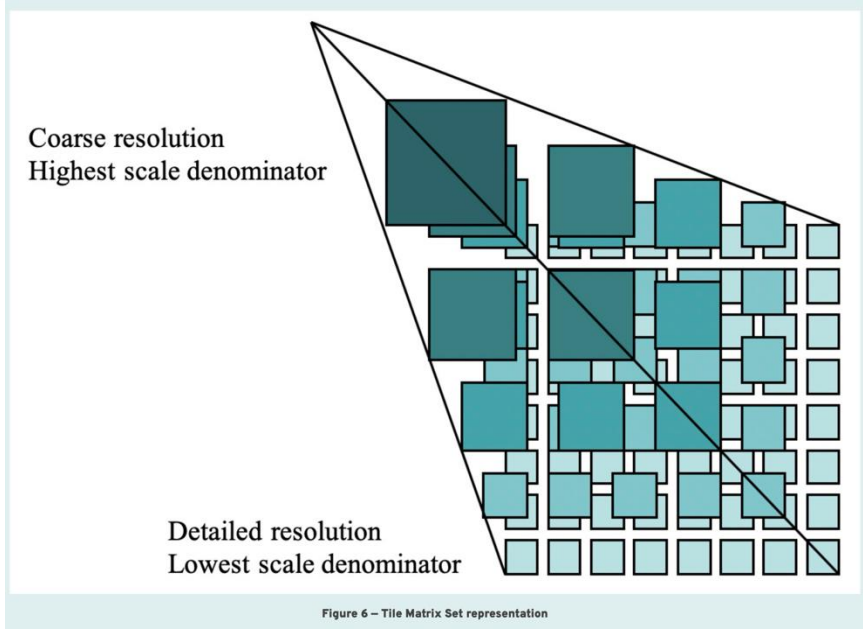
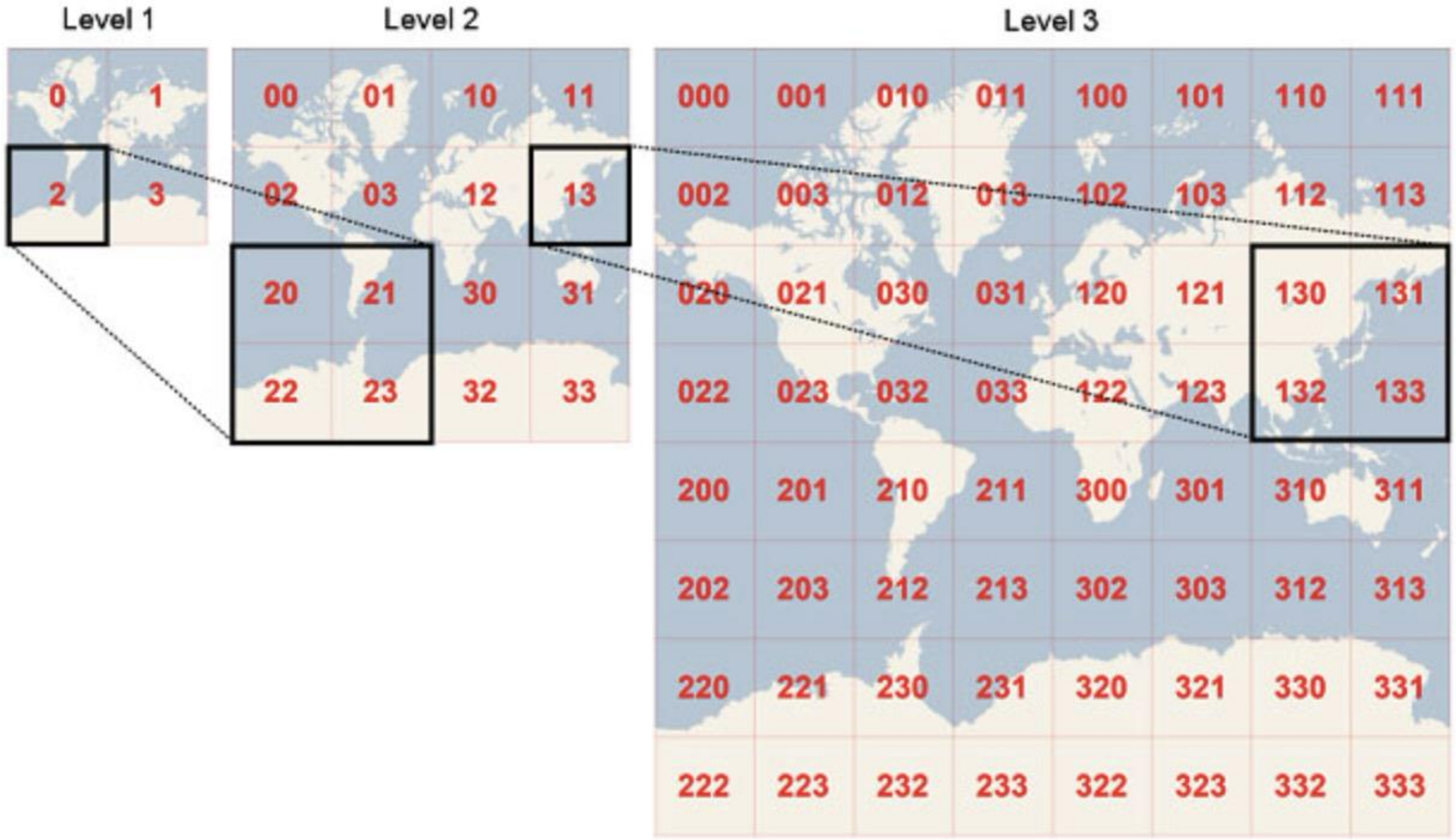


Figure 6 – Tile Matrix Set representation

Tile Matrix Set



Workflow of the AI-Driven analysis

Step 1: Get BBOX of each tile given X, Y, Z (refer to Juliet's code)
https://github.com/PermafrostDiscoveryGateway/viz-info/blob/main/helpful-code/prepocessing/bounding_box_tiles.ipynb

Step 2: Find a library that can generate a heatmap image for polygons (centroid) in this tile.

Step 3: Try to generate tile-based heatmap at different zoom levels
Start from zoom level 15, and go lower (limit: level 3-4 would be good enough), try different parameters

[Step 4: If our focus is Alaska, how to get which tiles are in Alaska]
Given a bounding box of an area of interest, can the library return which tiles are in Alaska?

Process

a) Manually choose a latitude and longitude and give a zoom level -> Tile (X, Y, Z)

b) Get the bounding box of the above tile

c) Projection transform: Bounding box: 4326 -> Ice wedge polygons: 3413

d) Query all the centroids of ice wedge polygons (multipolygons) within the bounding box

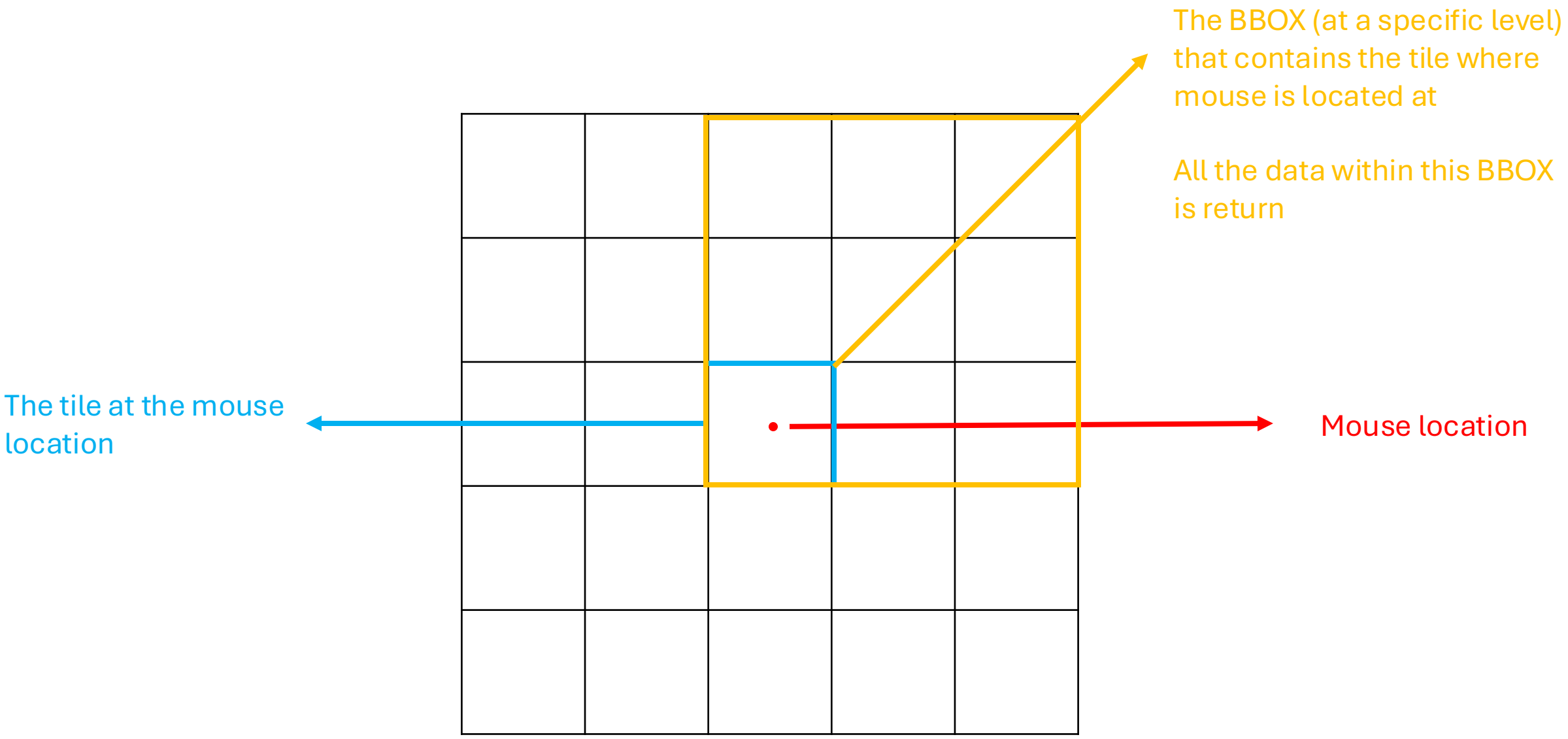
e) Generate a heatmap in **image format** for that tile.

f) Expand this calculation to all the tiles within northern Alaska for example.

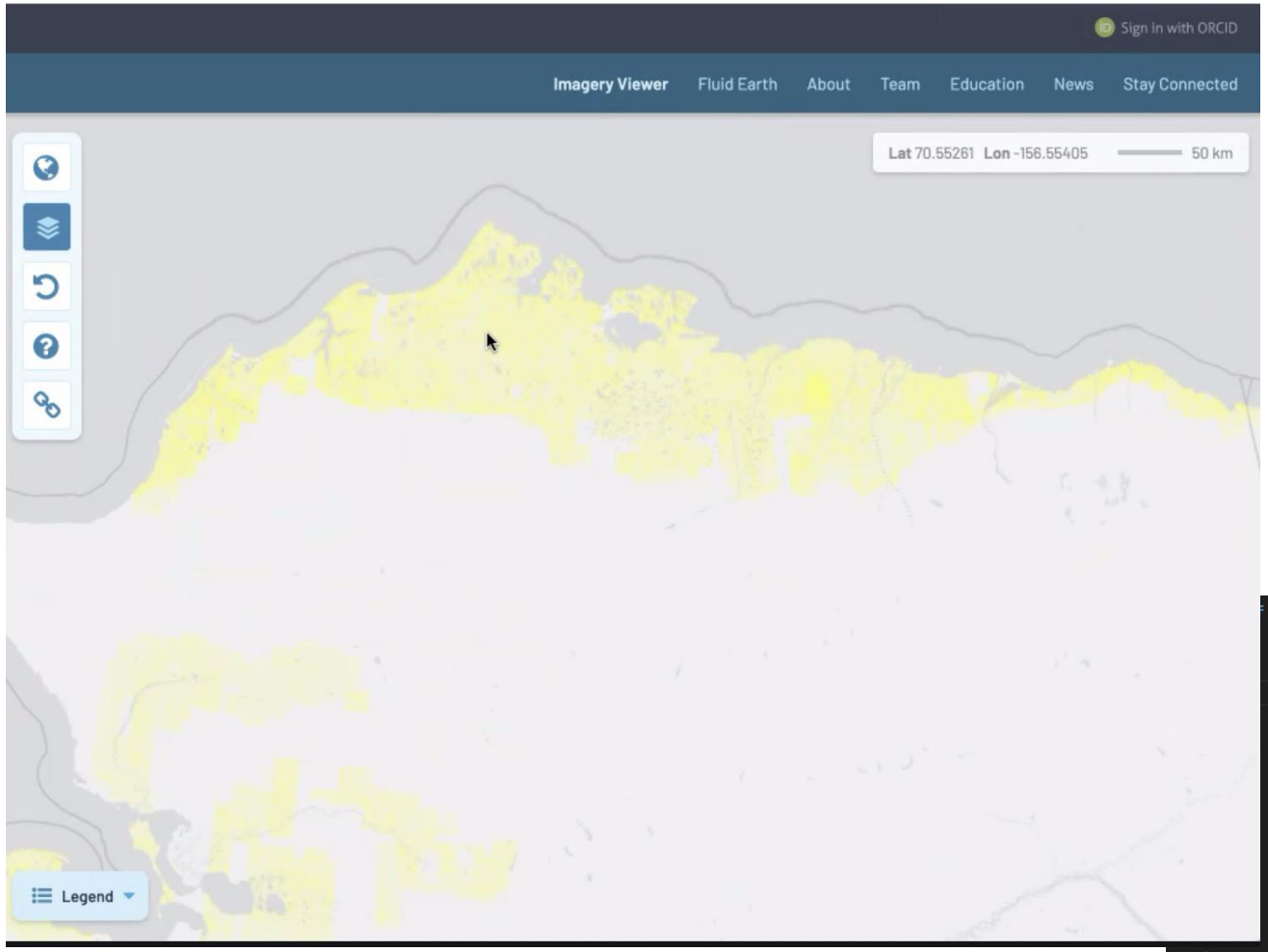
Meeting 8/6/2024

- Get BBOX of each tile given X, Y, Z (refer to Juliet's code) https://github.com/PermafrostDiscoveryGateway/viz-info/blob/main/helpful-code/prepocessing/bounding_box_tiles.ipynb

BBOX & Tiles Matrix



a) Manually choose a latitude and longitude and give a zoom level -> Tile (X, Y, Z)
b) Get the bounding box at this locatin




```
lat_long_to_BBOX(lat, long, zoom_level):
# Get Tile X, Y, Z from lat and long
Tile_XYZ = mercantile.tile(lng = long, lat = lat, zoom = zoom_level)
print("Tile information:" + str(Tile_XYZ))
# Set default TileMatrixSets into Spherical Mercator  (EPGS:3857)
tms = morecantile.tms.get("WebMercatorQuad")
# Given TMS grid (WebMercatorQuad), get the bounding box of a specific XYZ, and return in Geographical C
bbox = tms.bounds(Tile_XYZ)
# print(bbox)
# bbox[0]: xmin, bbox[1]: ymin, bbox[2]: xmax, bbox[3]: ymax
# print("bounding box:" + str(bbox[0],bbox[1],bbox[2],bbox[3]))
print(f"--------------The bounding box of this location is: {bbox[0]}, {bbox[1]}, {bbox[2]}, {bbox[3]}--
longs = (bbox[0], bbox[2])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
/home/xchen/code/alaska_shpMapping2/bin/python3 /home/xchen/code/shpMapping/Heatmap.py
(alaska_shpMapping2) (base) xchen@cici2labasuedu:~$ /home/xchen/code/alaska_shpMapping2/bin/python3 /home/xchen/code/shp
Mapping/Heatmap.py
Connection established successfully!
Tile information:Tile(x=4268, y=14368, z=16)
--------------The bounding box of this location is: -156.55517578125009, 70.55234969643529, -156.5496826171876, 70.55417
853776086----------
```

c) Projection transform: Bounding box: 4326 -> Ice wedge polygons: 3413

```
PROJCS["WGS_84_NSIDC_Sea_Ice_Polar_Stereographic_North",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SP
HEROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]
],PROJECTION["Stereographic_North_Pole"],PARAMETER["standard_parallel_1",70],PARAMETER["central_mer
idian",-45],PARAMETER["false_easting",0],PARAMETER["false_northing",0],UNIT["Meter",1]]
```

```
140     print(f"--------------The bounding box of this location is: {bbox[0]}, {bbox[1]}, {bbox[2]}, {bbox[3]}
141     longs = (bbox[0], bbox[2])
142     lats = (bbox[1], bbox[3])
143     p = Proj("EPSG:3413", preserve_units=False)
144     x,y = p(longs, lats)
145     # print(x, y)
146     xmin = x[0]
147     xmax = x[1]
148     ymin = y[1]
149     ymax = y[0]
150     print(xmin, xmax, ymin, ymax)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
/home/xchen/code/alaska_shpMapping2/bin/python3 /home/xchen/code/shpMapping/Heatmap.py
(alaska_shpMapping2) (base) xchen@cici2labasuedu:~$ /home/xchen/code/alaska_shpMapping2/bin/python3 /home/xchen/code/shpMapping
/Heatmap.py
Connection established successfully!
Tile information:Tile(x=4268, y=14368, z=16)
--------------The bounding box of this location is: -156.55517578125009, 70.55234969643529, -156.5496826171876, 70.554178537760
86----------
-1977647.862519466 -1977533.2904949822 780952.5983109941 781217.0271017228
```

All the tiles within bounding box

X = 4268
Y = 14368
Z = 10

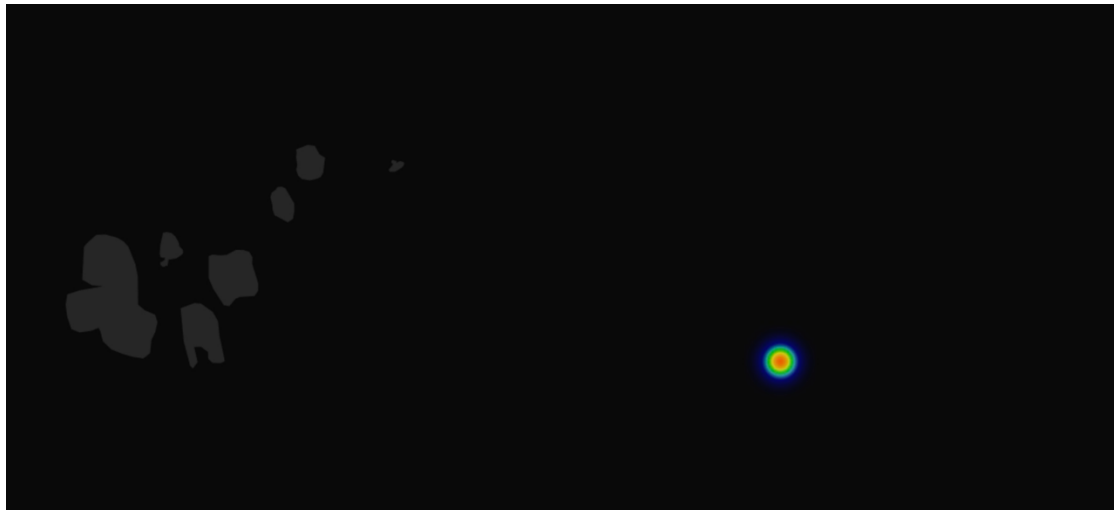
```
023, y=941, z=10), Tile(x=1023, y=942, z=10), Tile(x=1023, y=943, z=10), Tile(x=1023, y=944, z=10), Tile(x=1023, y=945, z=10),
Tile(x=1023, y=946, z=10), Tile(x=1023, y=947, z=10), Tile(x=1023, y=948, z=10), Tile(x=1023, y=949, z=10), Tile(x=1023, y=950,
 z=10), Tile(x=1023, y=951, z=10), Tile(x=1023, y=952, z=10), Tile(x=1023, y=953, z=10), Tile(x=1023, y=954, z=10), Tile(x=1023
, y=955, z=10), Tile(x=1023, y=956, z=10), Tile(x=1023, y=957, z=10), Tile(x=1023, y=958, z=10), Tile(x=1023, y=959, z=10), Til
e(x=1023, y=960, z=10), Tile(x=1023, y=961, z=10), Tile(x=1023, y=962, z=10), Tile(x=1023, y=963, z=10), Tile(x=1023, y=964, z=
10), Tile(x=1023, y=965, z=10), Tile(x=1023, y=966, z=10), Tile(x=1023, y=967, z=10), Tile(x=1023, y=968, z=10), Tile(x=1023, y
=969, z=10), Tile(x=1023, y=970, z=10), Tile(x=1023, y=971, z=10), Tile(x=1023, y=972, z=10), Tile(x=1023, y=973, z=10), Tile(x
=1023, y=974, z=10), Tile(x=1023, y=975, z=10), Tile(x=1023, y=976, z=10), Tile(x=1023, y=977, z=10), Tile(x=1023, y=978, z=10)
, Tile(x=1023, y=979, z=10), Tile(x=1023, y=980, z=10), Tile(x=1023, y=981, z=10), Tile(x=1023, y=982, z=10), Tile(x=1023, y=98
3, z=10), Tile(x=1023, y=984, z=10), Tile(x=1023, y=985, z=10), Tile(x=1023, y=986, z=10), Tile(x=1023, y=987, z=10), Tile(x=10
23, y=988, z=10), Tile(x=1023, y=989, z=10), Tile(x=1023, y=990, z=10), Tile(x=1023, y=991, z=10), Tile(x=1023, y=992, z=10), T
ile(x=1023, y=993, z=10), Tile(x=1023, y=994, z=10), Tile(x=1023, y=995, z=10), Tile(x=1023, y=996, z=10), Tile(x=1023, y=997,
z=10), Tile(x=1023, y=998, z=10), Tile(x=1023, y=999, z=10), Tile(x=1023, y=1000, z=10), Tile(x=1023, y=1001, z=10), Tile(x=102
3, y=1002, z=10), Tile(x=1023, y=1003, z=10), Tile(x=1023, y=1004, z=10), Tile(x=1023, y=1005, z=10), Tile(x=1023, y=1006, z=10
), Tile(x=1023, y=1007, z=10), Tile(x=1023, y=1008, z=10), Tile(x=1023, y=1009, z=10), Tile(x=1023, y=1010, z=10), Tile(x=1023,
 y=1011, z=10), Tile(x=1023, y=1012, z=10), Tile(x=1023, y=1013, z=10), Tile(x=1023, y=1014, z=10), Tile(x=1023, y=1015, z=10),
 Tile(x=1023, y=1016, z=10), Tile(x=1023, y=1017, z=10), Tile(x=1023, y=1018, z=10), Tile(x=1023, y=1019, z=10), Tile(x=1023, y
=1020, z=10), Tile(x=1023, y=1021, z=10), Tile(x=1023, y=1022, z=10), Tile(x=1023, y=1023, z=10)]
------------------The number of tiles within bbox is: 1048576------------------
```

d) Select the centroid of ice wedge polygons (multipolygons) within the bounding box

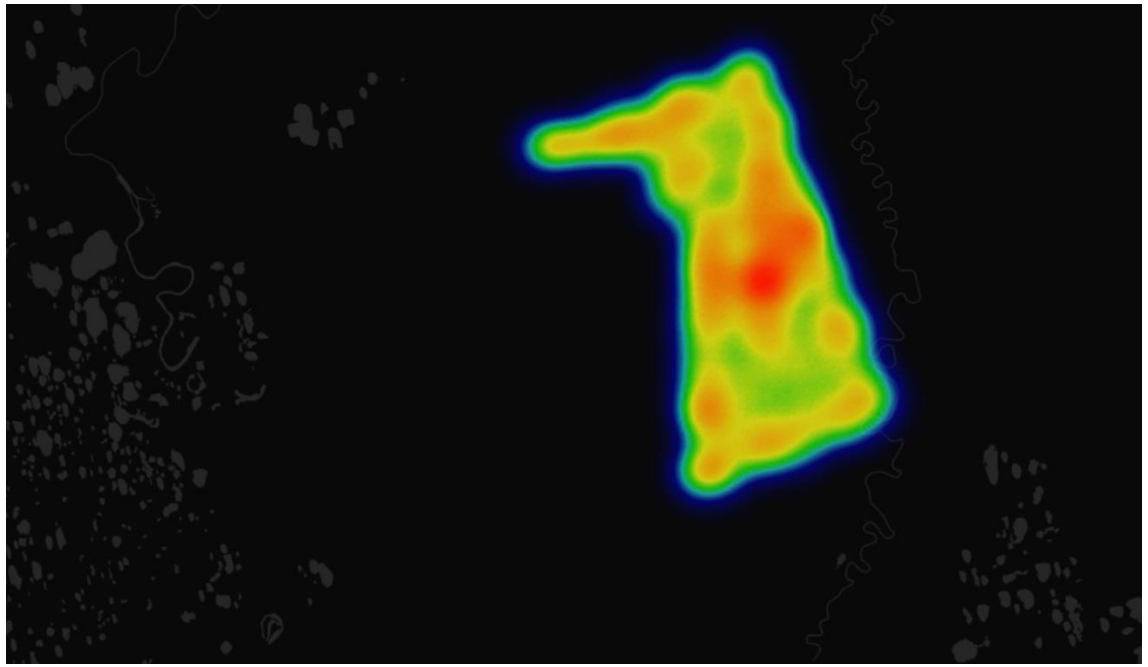
```
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168 WHERE centroidx > -1980206.9719926012 AND centroidx < -1972895.05
17446548 AND centroidy > 774969.5728202369 AND centroidy < 791901.0028554909;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only supports SQL
Alchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not teste
d. Please consider using SQLAlchemy.
  df = pd.read_sql(
         centroidx   centroidy                                              geom
0     -1973943.222  783555.324  MULTIPOLYGON (((-1973946.421 783545.559, -1973...
1     -1975062.973  784025.898  MULTIPOLYGON (((-1975063.134 784016.307, -1975...
2     -1976831.168  783399.403  MULTIPOLYGON (((-1976832.477 783394.823, -1976...
3     -1976823.571  783388.202  MULTIPOLYGON (((-1976820.497 783380.468, -1976...
4     -1975333.295  783341.091  MULTIPOLYGON (((-1975330.332 783333.213, -1975...
...            ...         ...                                               ...
79878 -1972974.852  774985.795  MULTIPOLYGON (((-1972976.069 774980.57, -19729...
79879 -1973367.652  775036.399  MULTIPOLYGON (((-1973368.272 775030.217, -1973...
79880 -1972954.849  775025.382  MULTIPOLYGON (((-1972945.859 775017.057, -1972...
79881 -1972946.820  774991.520  MULTIPOLYGON (((-1972950.547 774985.355, -1972...
79882 -1972935.639  774992.343  MULTIPOLYGON (((-1972934.4 774987.149, -197293...

[79883 rows x 3 columns]
--------------Data Extracted! The runtime of getting data from database is 339.01107835769653-------------
```

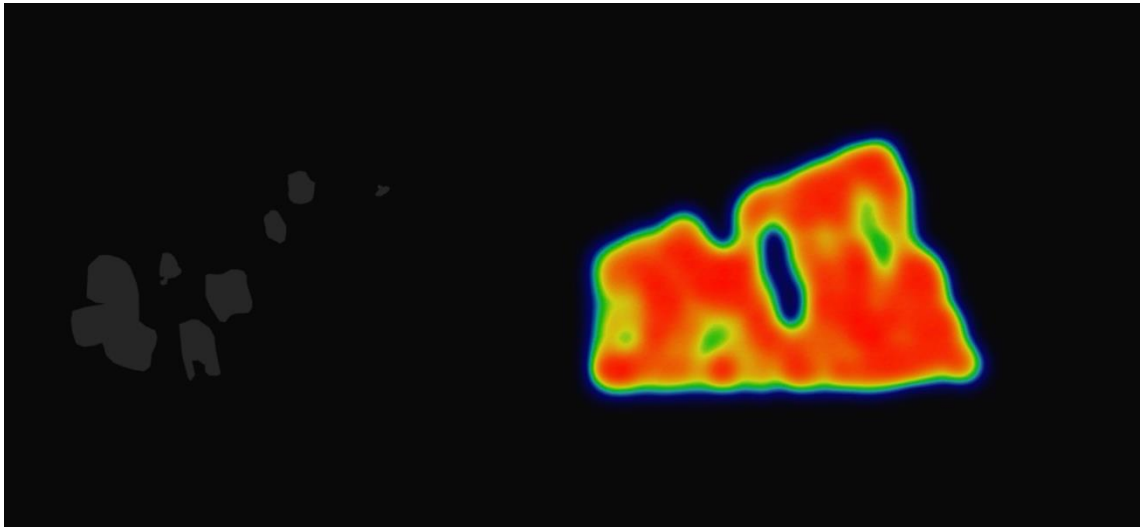
e) Select the centroid of ice wedge polygons (multipolygons) within the bounding box



Zoom level : 16



Zoom level: 8, Tile count: 65,536



Zoom level: 10, Tile count: 1,048,576

x=4268, y=14368, z=16
The bounding box of this location is: -156.55517578125009, 70.55234969643529, -156.5496826171876, 70.55417853776086

x=66, y=224, z=10
The bounding box of this location is: -156.79687500000009, 70.49557354093146, -156.44531250000006, 70.61261423801933

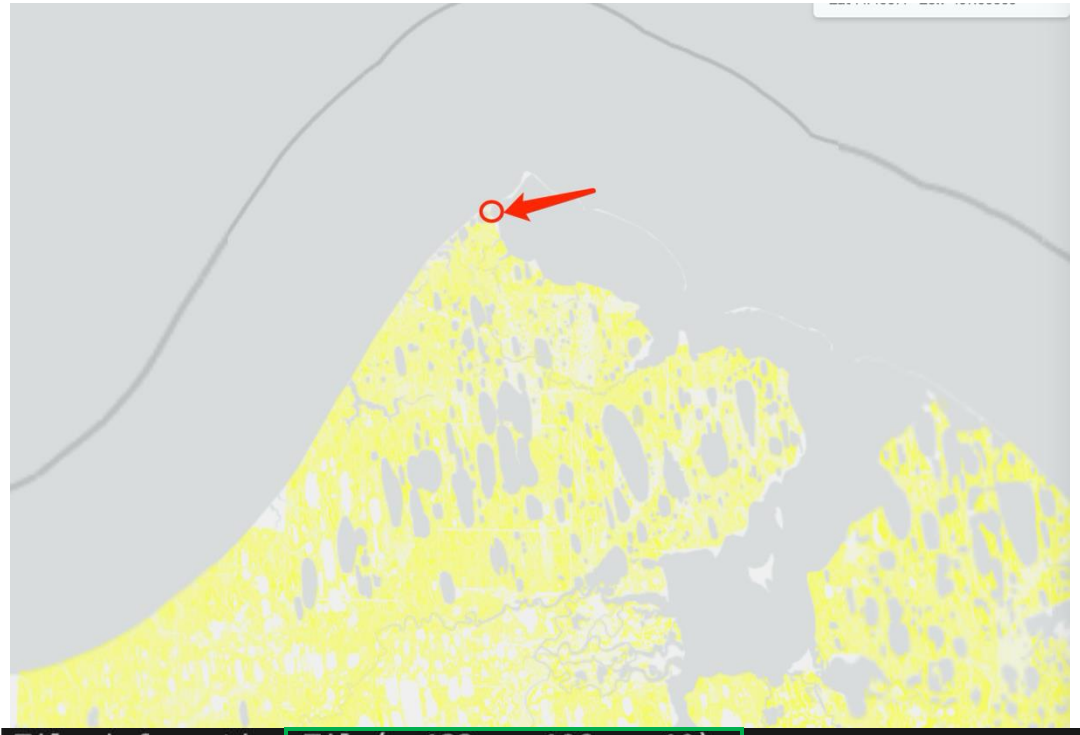
x=16, y=56, z=8
The bounding box of this location is: -157.50000000000003, 70.14036427207168, -156.09375000000003, 70.61261423801925

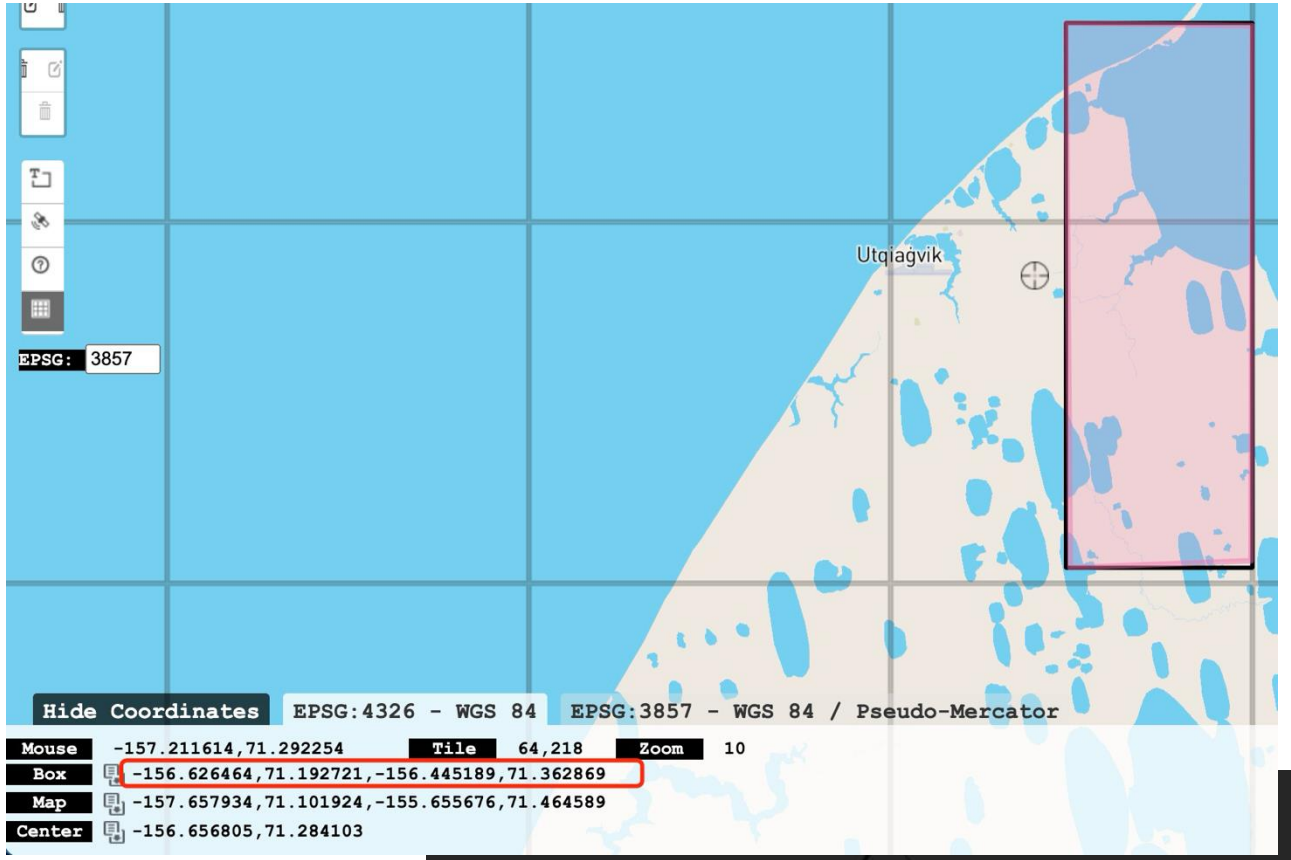
Meeting (8/13/2024)

- Single Tile heatmap (png format / folium)
- Multiple Tile heatmap (png format / folium)

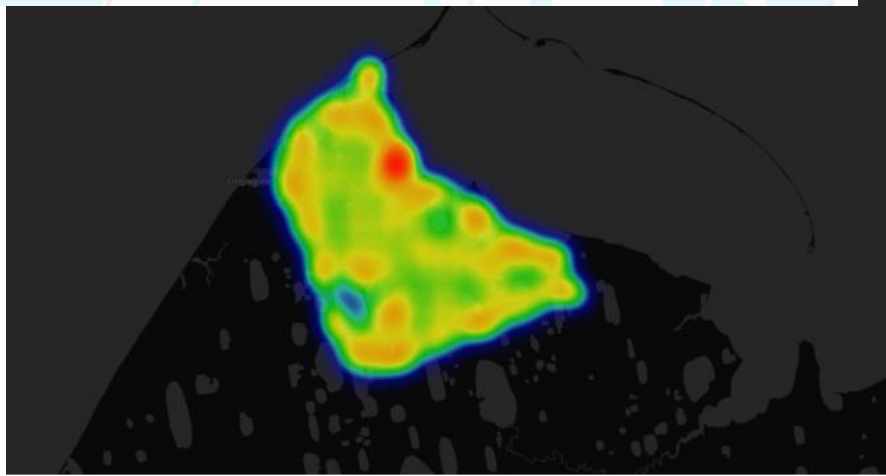
Single Tile Heatmap

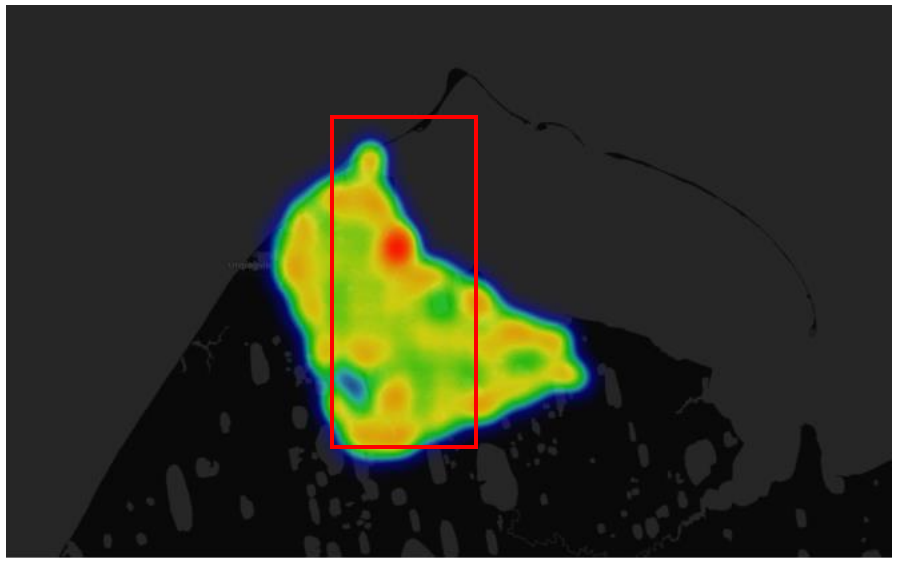
Using morecantile: project to WGS84 (WGS1984Quad)

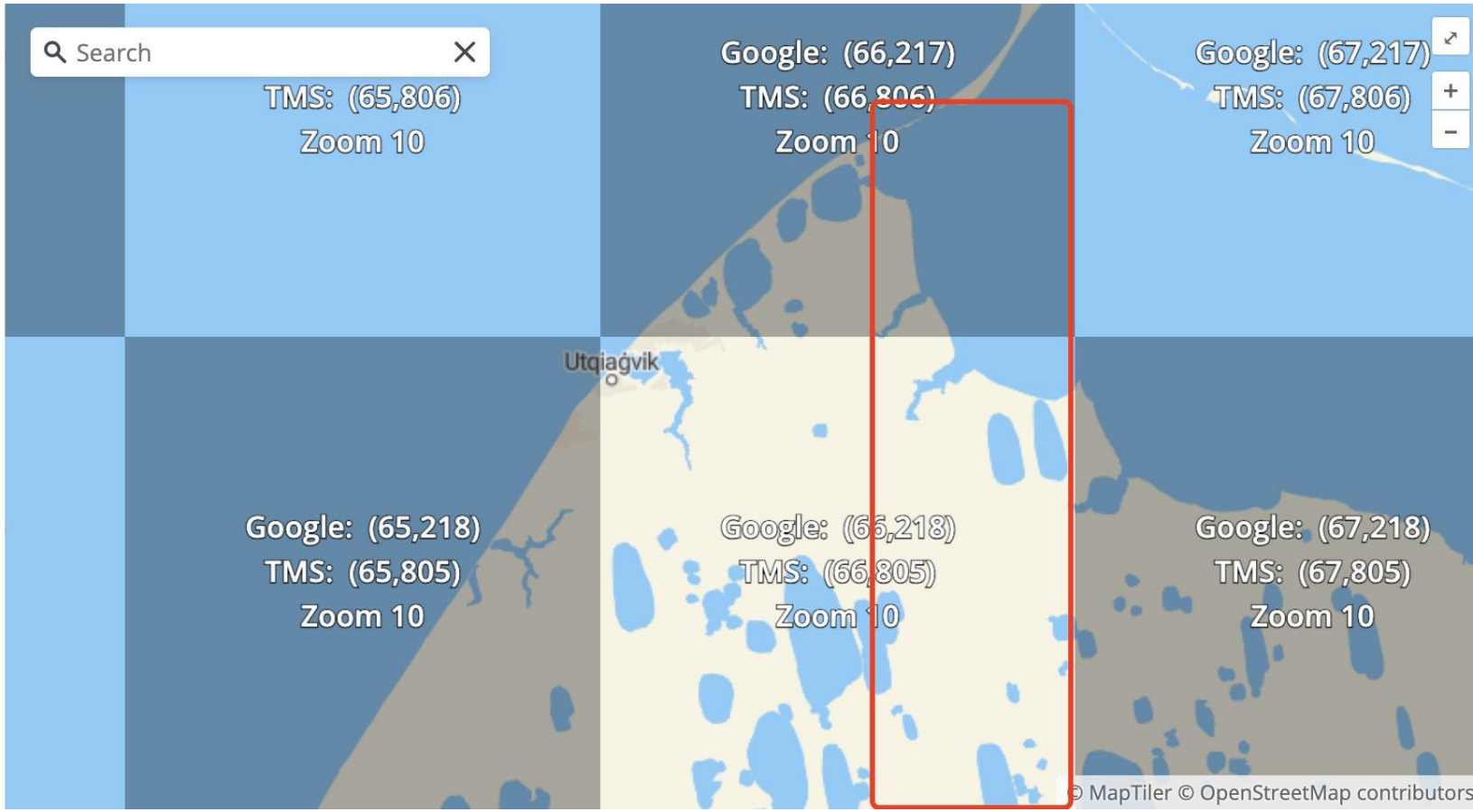




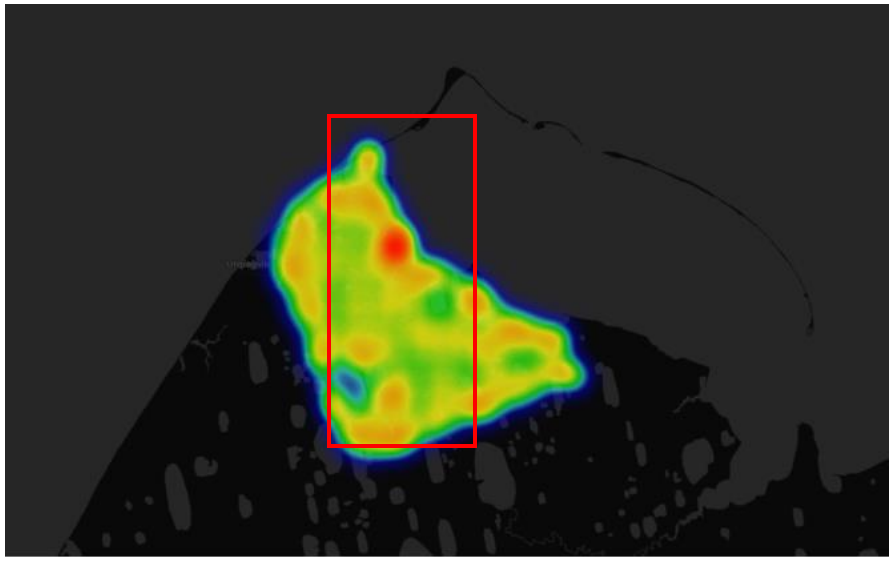

```
Tile information:Tile(x=133, y=106, z=10)
Tile(x=133, y=106, z=10)
--------------The bounding box of this location is: -156.62109375, 71.19140625, -156.4453125,
 71.3671875-----------
-1910636.4785807736 -1894764.7639939869 744279.3464602896 757288.2922898965
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1910636.4
785807736 AND centroidx < -1894764.7639939869 AND centroidy > 744279.3464602896 AND centroidy
 < 757288.2922898965;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: Use
rWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string
URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
```

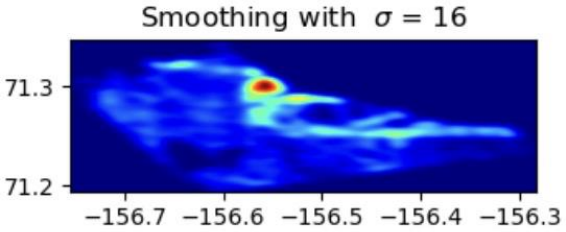


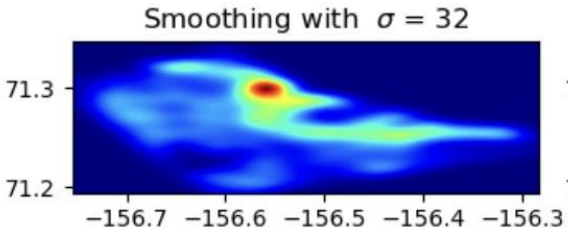


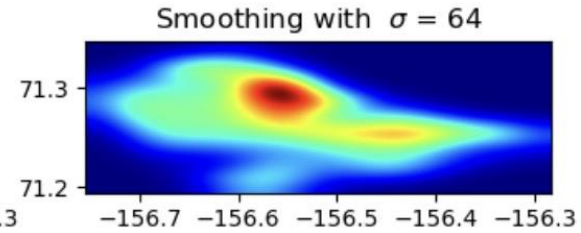


```
Tile information:Tile(x=133, y=106, z=10)
Tile(x=133, y=106, z=10)
--------------The bounding box of this location is: -156.62109375, 71.19140625, -156.4453125,
 71.3671875-----------
-1910636.4785807736 -1894764.7639939869 744279.3464602896 757288.2922898965
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1910636.4
785807736 AND centroidx < -1894764.7639939869 AND centroidy > 744279.3464602896 AND centroidy
 < 757288.2922898965;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: Use
rWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string
URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using
SQLAlchemy.
```







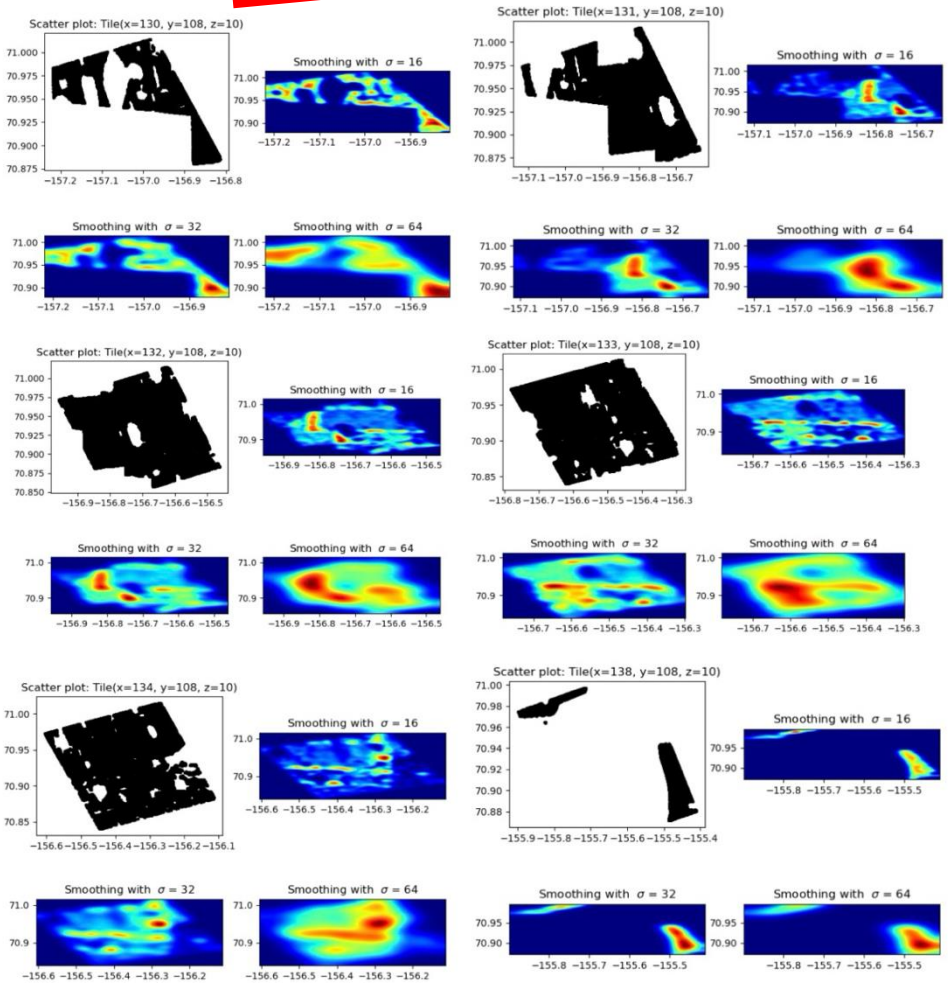


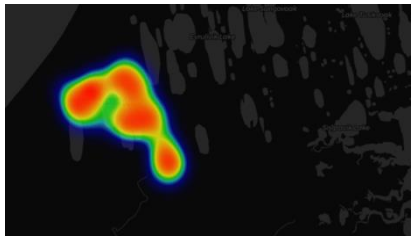

```
Tile information:Tile(x=133, y=106, z=10)
--------------The bounding box of this location is: -156.62109375, 71.19140625, -156.4453125, 71.3671875-----------
-1910636.4785807736 -1894764.7639939869 744279.3464602896 757288.2922898965
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1910636.4785807736 AND centroidx < -1
894764.7639939869 AND centroidy > 744279.3464602896 AND centroidy < 757288.2922898965;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only suppor
ts SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects a
re not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(
---------------Data Extracted! The runtime of getting data from database is 22.26591992378235-------------
Creating heatmap...
-156.75456547765768
-156.28325826694945
71.19428135182447
71.34647038148489
Creating heatmap...
------------------------Heatmap Created!-----------------------
```

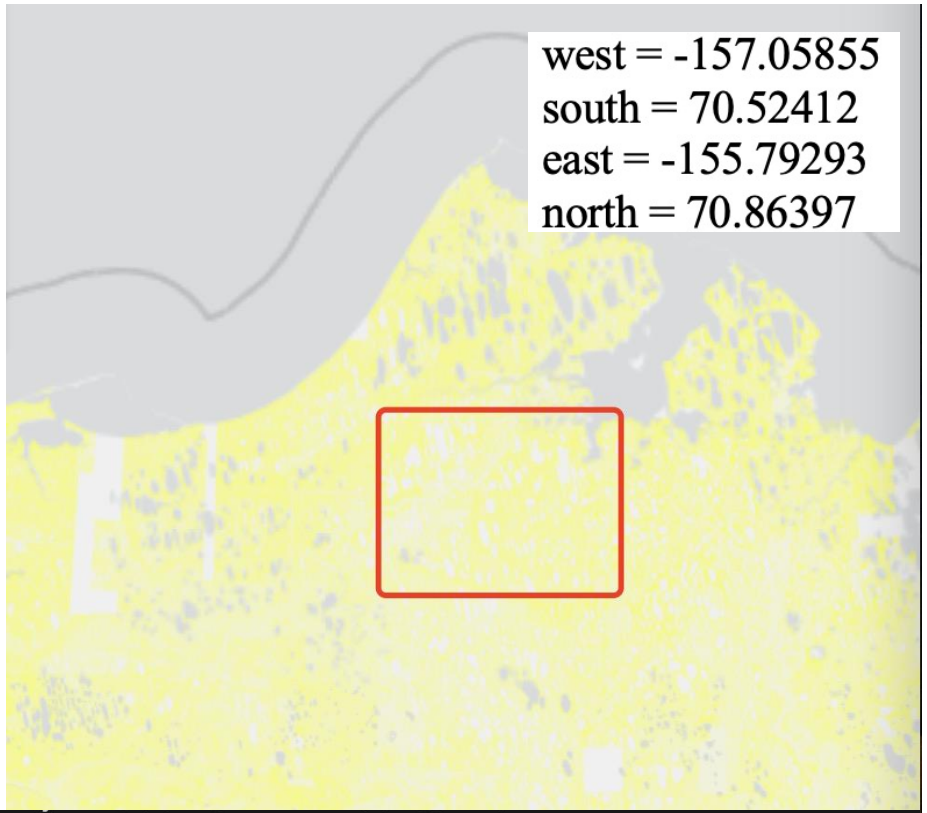
Some of the points are outside of BBOX

Multiple Tile heatmap

Using morecantile: project to WGS84 (WGS1984Quad)



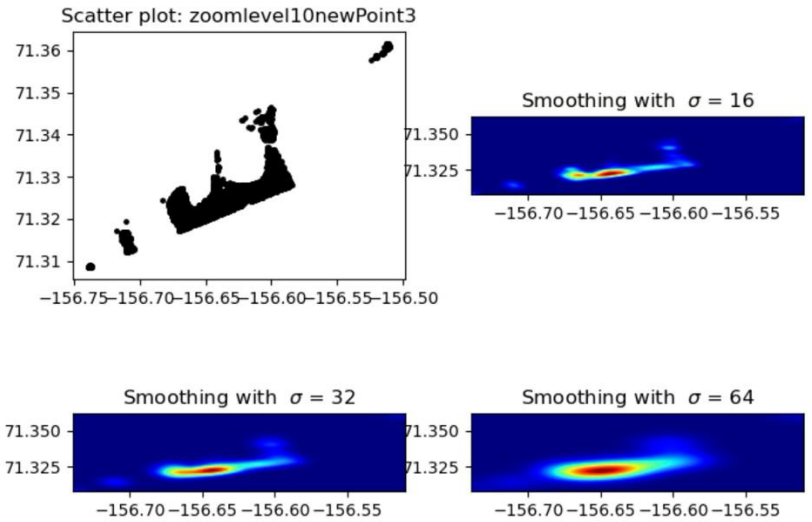


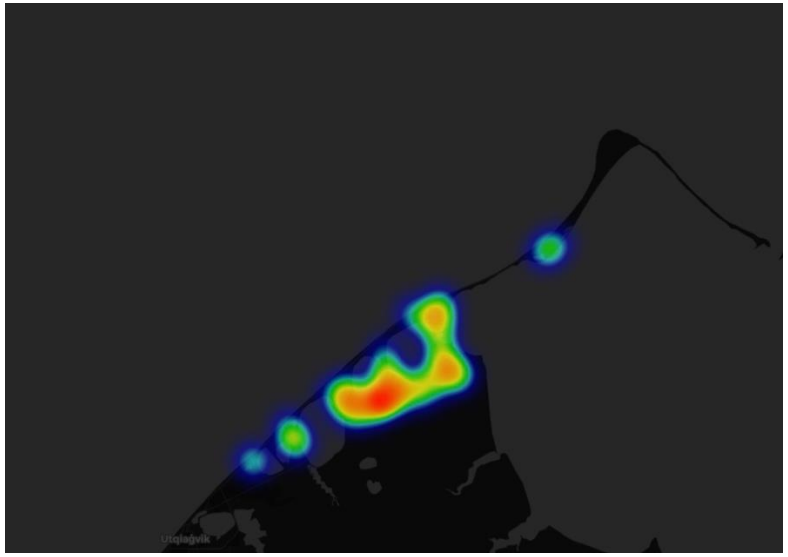


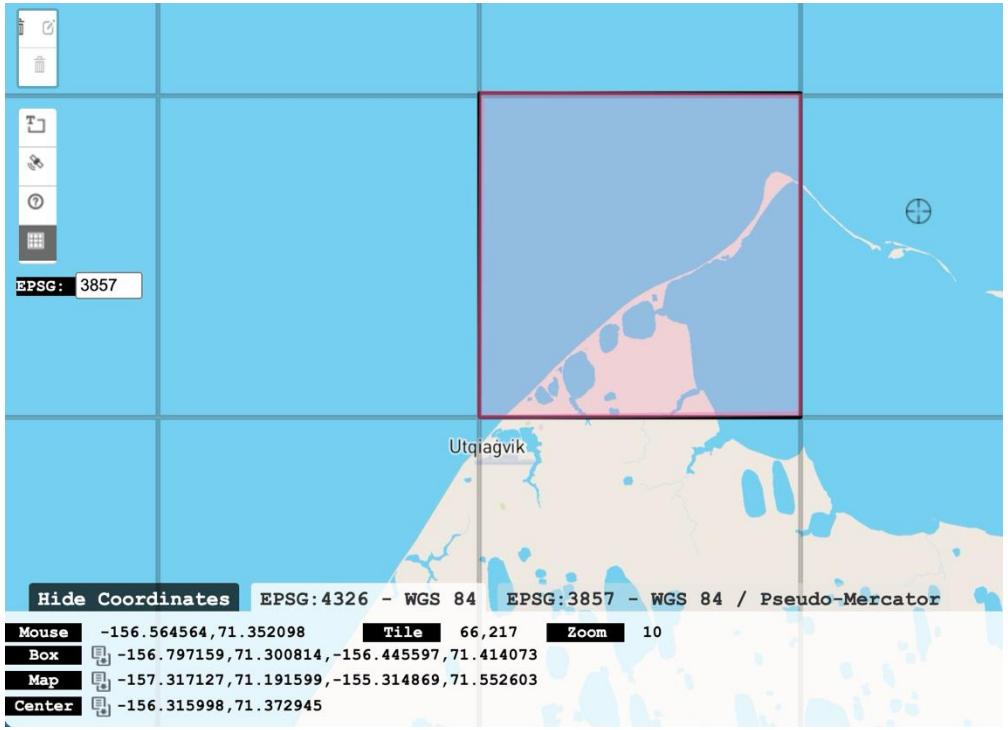

```
------------------The number of tiles within the geographic bbox is: 1187136----------------
_
1
Tile(x=130, y=108, z=10)
--------------The bounding box of this location is: -157.1484375, 70.83984375, -156.97265625,
 71.015625-----------
```

Single Tile Heatmap

Using morecantile: project to EPSG:3857 (WebMercatorQuad)



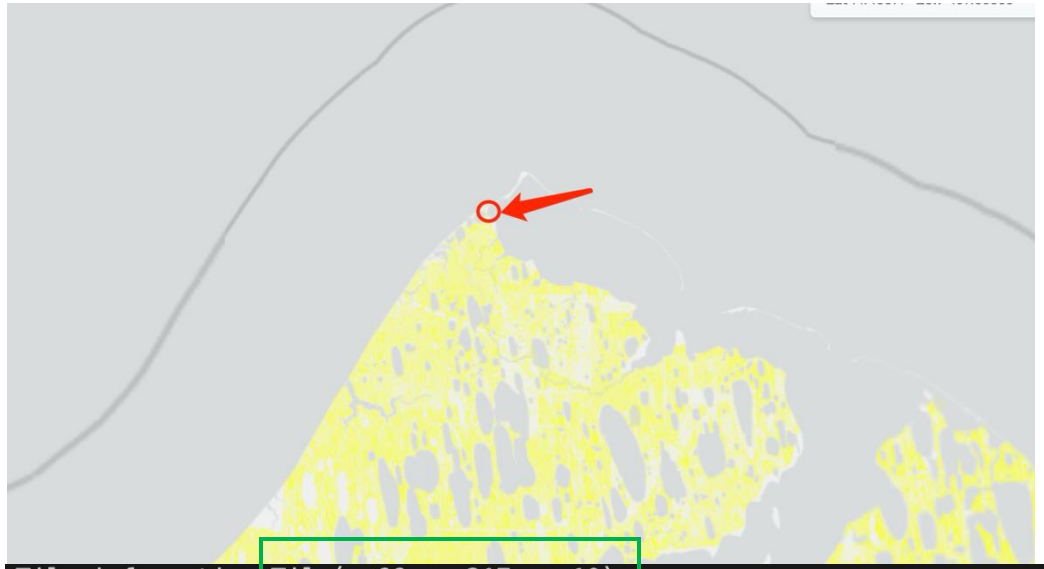


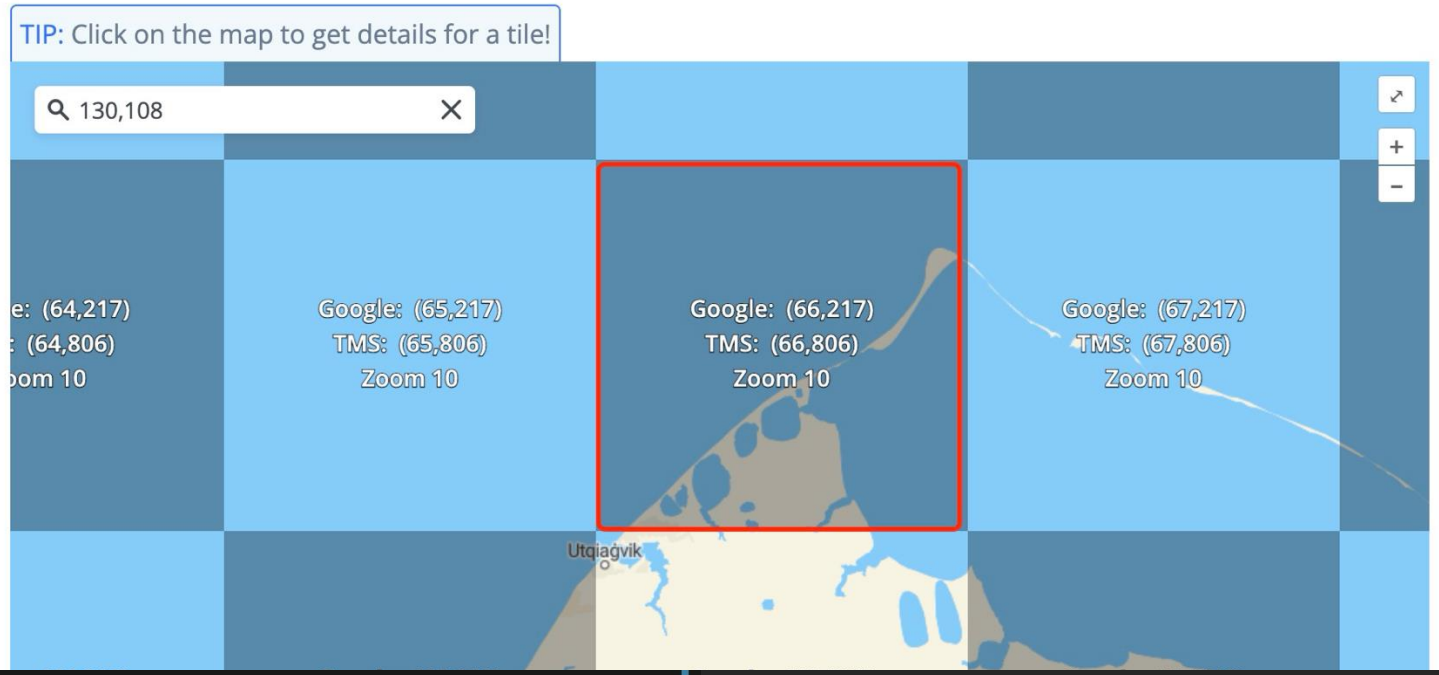



```
Tile information:Tile(x=66, y=217, z=10)
--------------The bounding box of this location is: -156.796875, 71.30079291637452, -156.4453125, 71.41317683396565------
------
0     POINT (-1897013.64 758631.307)
1    POINT (-1890008.237 742410.943)
Name: geometry, dtype: geometry
-1897013.6399017135 -1890008.2366487072 742410.9429881338 758631.3072905991
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1897013.6399017135 AND centroidx < -1
890008.2366487072 AND centroidy > 742410.9429881338 AND centroidy < 758631.3072905991;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only suppor
ts SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects a
re not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(
---------------Data Extracted! The runtime of getting data from database is 10.205280065536499-------------
Creating heatmap...
Creating heatmap...
```

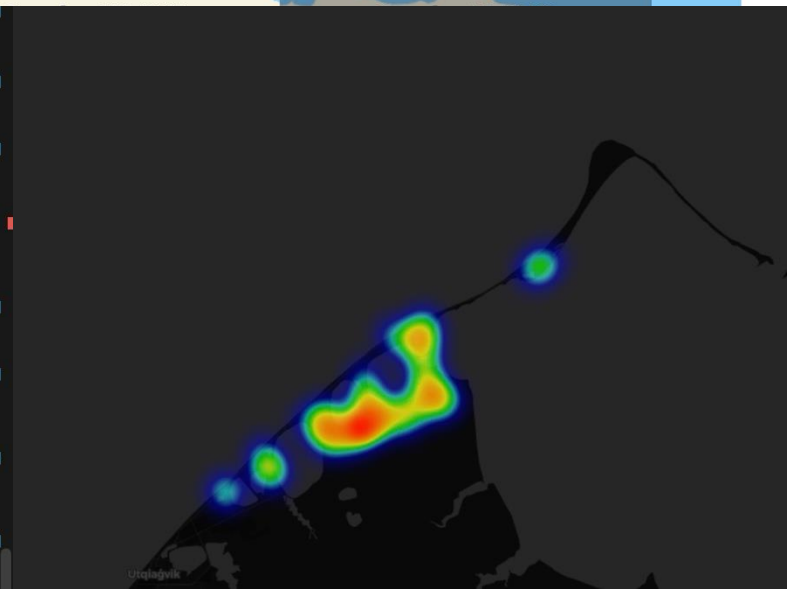
Single Tile Heatmap

Using mercantile

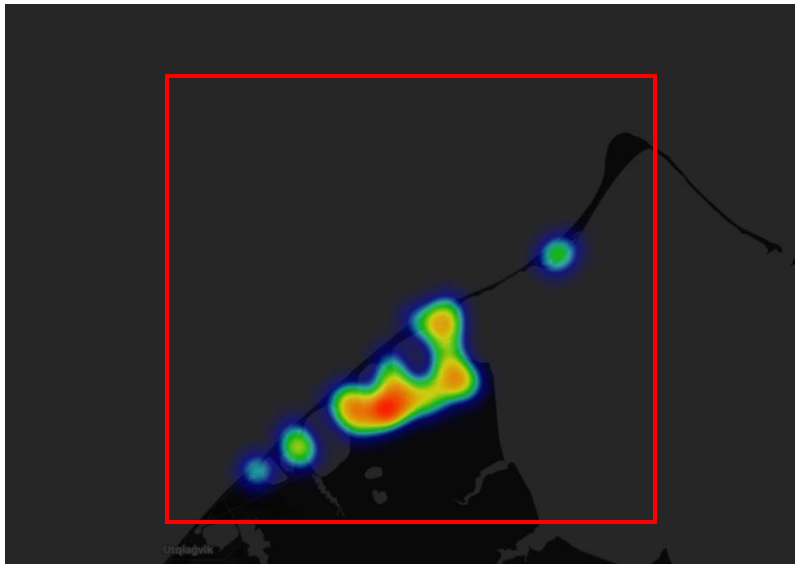


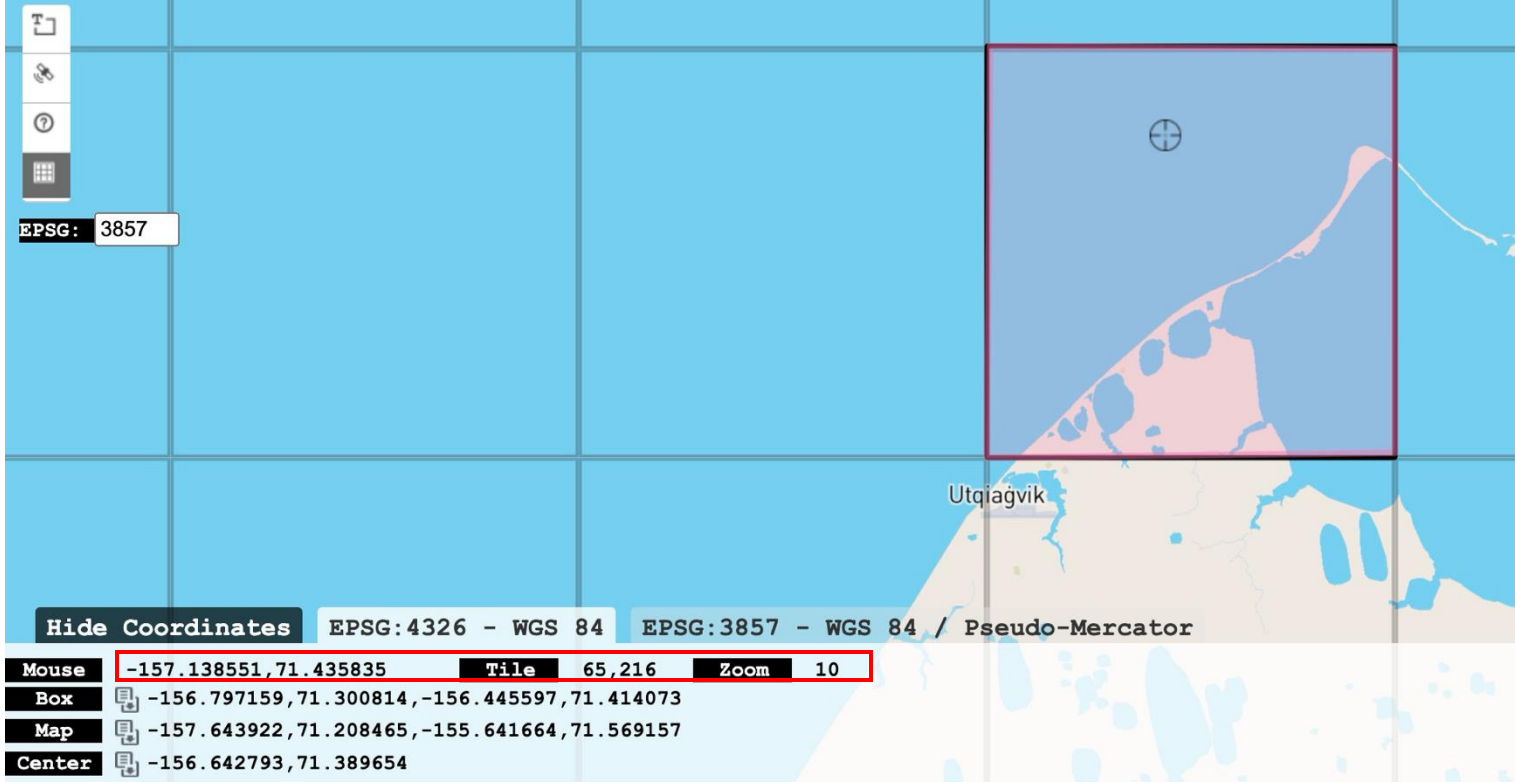


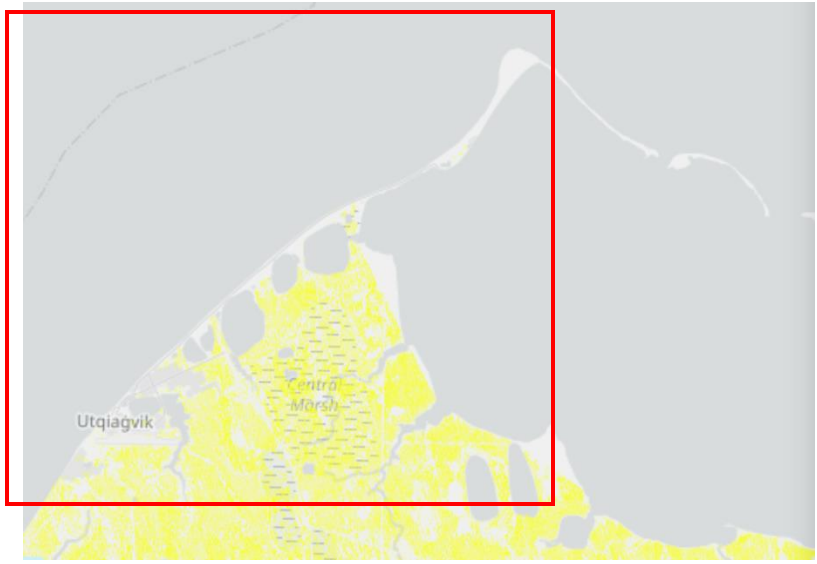

```
Tile information:Tile(x=66, y=217, z=10)
Tile(x=66, y=217, z=10)
--------------The bounding box of this location is: -156.796875, 71.30079291637452, -156.4453125, 71.4131768339656
5----------
-1897013.6399017135 -1890008.2366487072 742410.9429881338 758631.3072905991
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1897013.6399017135 AND centroi
dx < -1890008.2366487072 AND centroidy > 742410.9429881338 AND centroidy < 758631.3072905991;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only
 supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DB
API2 objects are not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(
---------------Data Extracted! The runtime of getting data from database is 10.024113893508911-------------
Creating heatmap...
-156.73926813038608
-156.51000922842084
71.30834542722577
71.36177072594981
Creating heatmap...
```



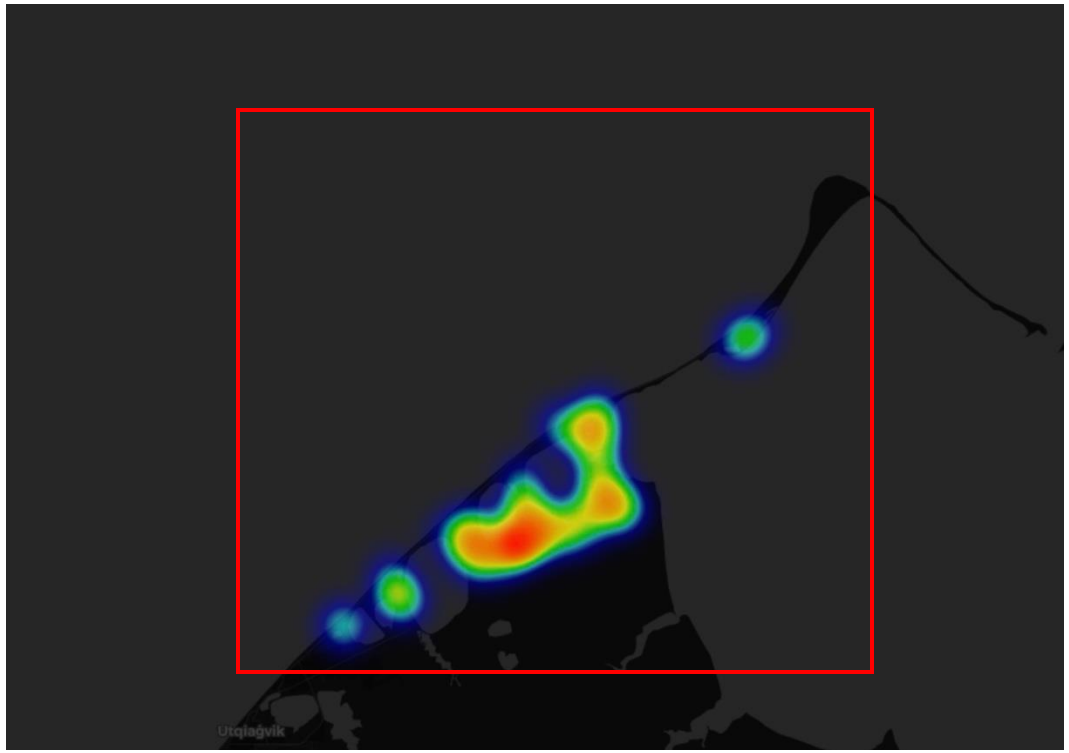
Using mercantile and Proj

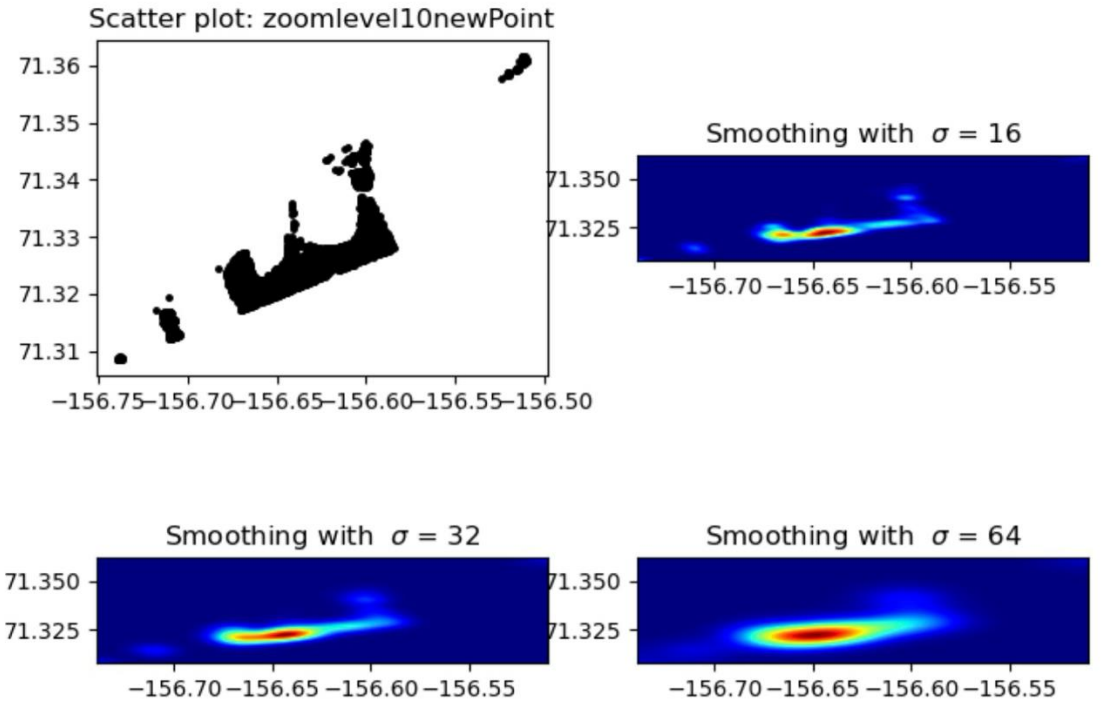







```
Tile information:Tile(x=66, y=217, z=10)
Tile(x=66, y=217, z=10)
---------------The bounding box of this location is: -156.796875, 71.30079291637452, -156.4453125, 71.4131768339656
5-----------
-1897013.6399017135 -1890008.2366487072 742410.9429881338 758631.3072905991
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1897013.6399017135 AND centroi
dx < -1890008.2366487072 AND centroidy > 742410.9429881338 AND centroidy < 758631.3072905991;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only
 supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DB
API2 objects are not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(
---------------Data Extracted! The runtime of getting data from database is 10.024113893508911-------------
Creating heatmap...
-156.73926813038608
-156.51000922842084
71.30834542722577
71.36177072594981
Creating heatmap...
                    Heatmap Created!
```

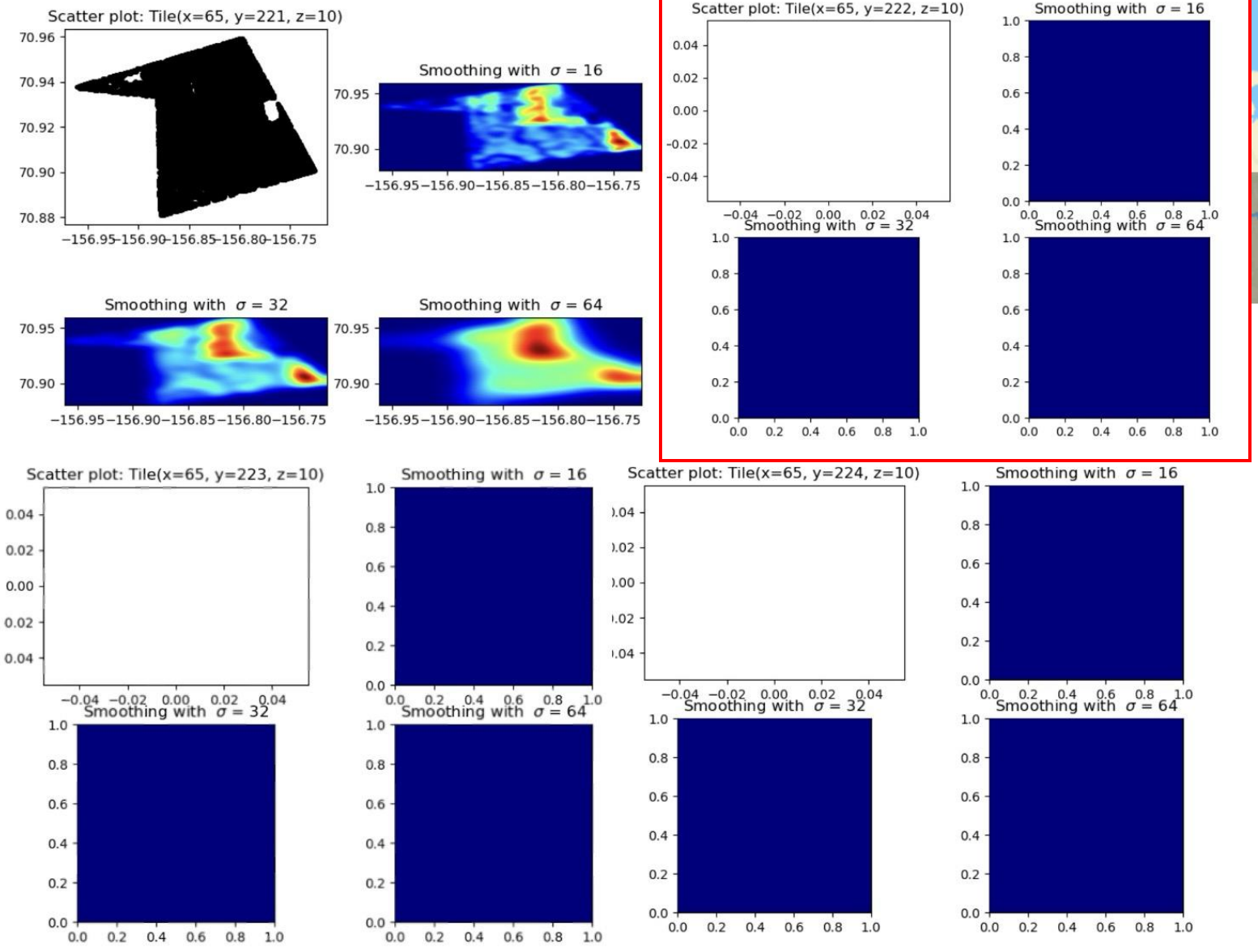


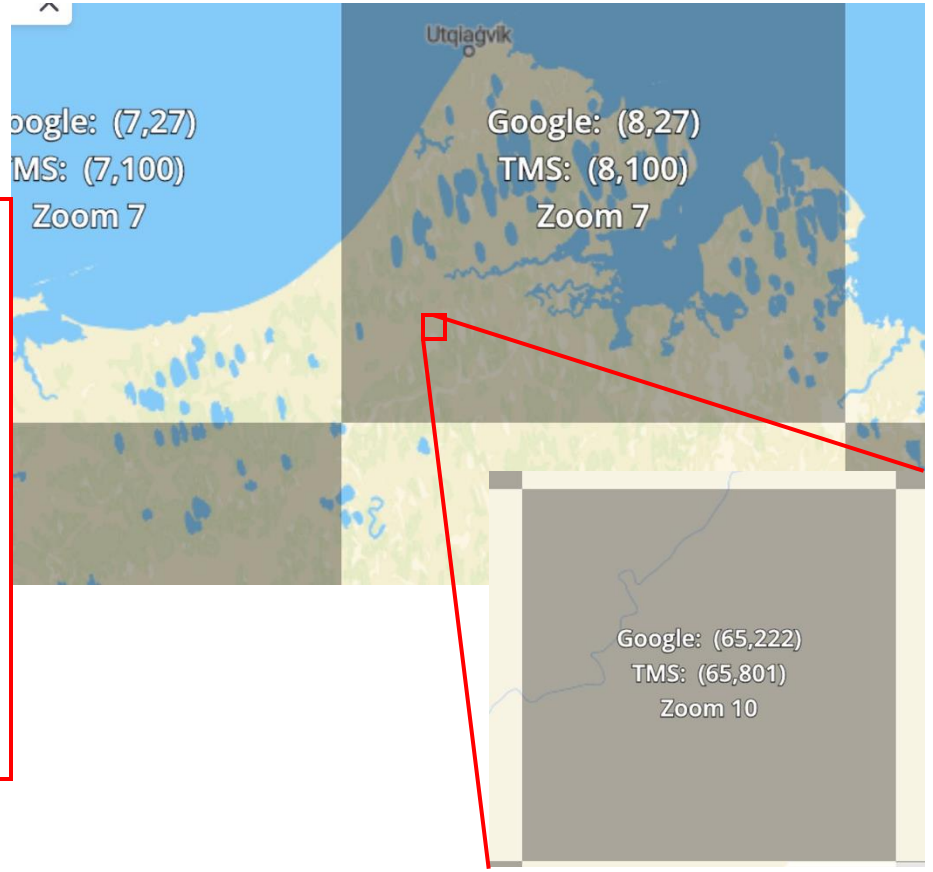


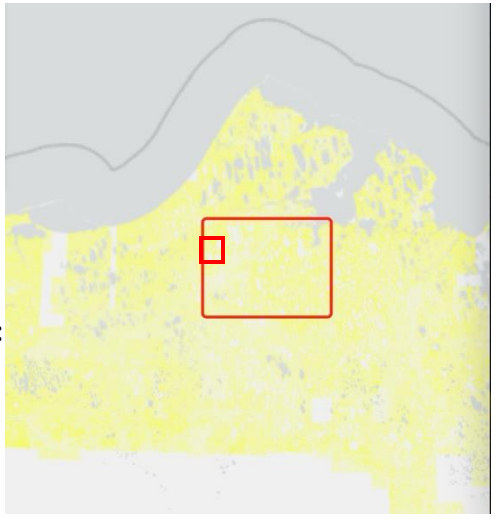

```
Tile information:Tile(x=66, y=217, z=10)
Tile(x=66, y=217, z=10)
--------------The bounding box of this location is: -156.796875, 71.30079291637452, -156.4453125, 71.4131768339656
5----------
-1897013.6399017135 -1890008.2366487072 742410.9429881338 758631.3072905991
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1897013.6399017135 AND centroi
dx < -1890008.2366487072 AND centroidy > 742410.9429881338 AND centroidy < 758631.3072905991;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only
 supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DB
API2 objects are not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(
--------------Data Extracted! The runtime of getting data from database is 10.024113893508911-------------
Creating heatmap...
-156.73926813038608
-156.51000922842084
71.30834542722577
71.36177072594981
Creating heatmap...
                   Heatmap Created!
```

Only parts of data are shown

Multiple Tiles Heatmaps





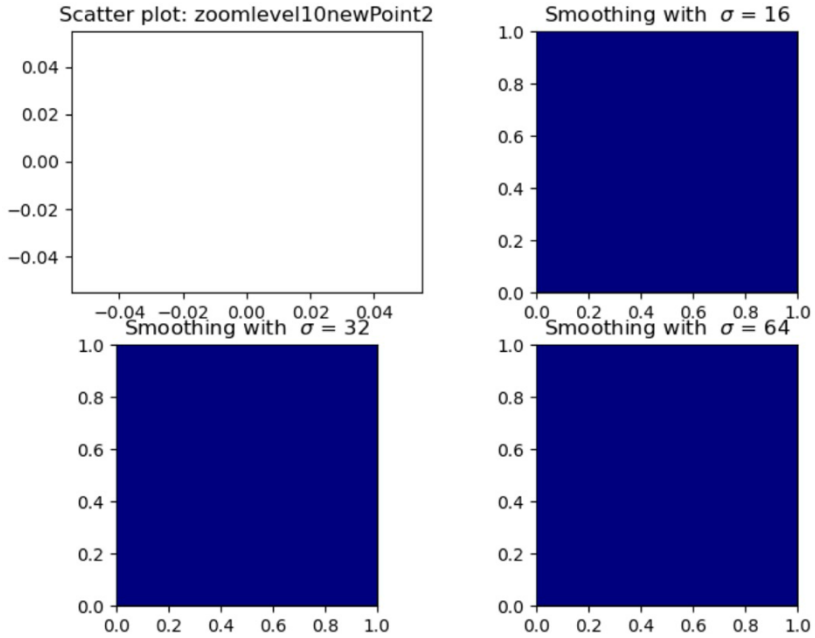


ROI Bounding box:

west = -157.05855
south = 70.52412
east = -155.79293
north = 70.86397

Using mercantile + morecantile

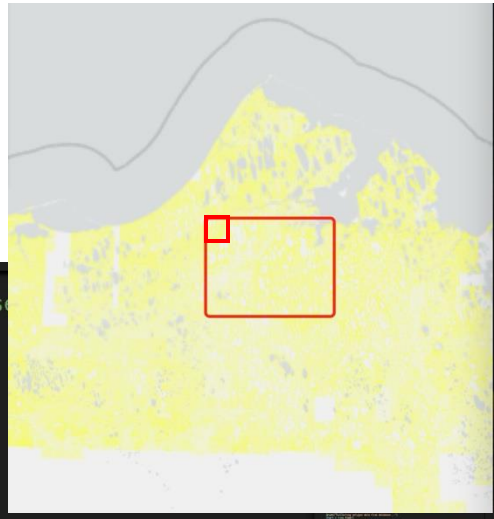
Single Tiles heatmaps




```
File information:Tile(x=66, y=217, z=10)
--------------The bounding box of this location is: -168.3984375, 51.6796875, -168.22265625, 51.85546875----------
-3595650.5545682446 -3585135.531807242 2348072.1963431486 2370754.0183629603
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -3595650.5545682446 AND centroidx < -3
585135.531807242 AND centroidy > 2348072.1963431486 AND centroidy < 2370754.0183629603;
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only suppor
ts SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects a
re not tested. Please consider using SQLAlchemy.
```

WTK: Rectangular -> Polygon -> Rectangular

Projection transformation



Input coordinate system

EPSG:3413 WGS 84 / NSIDC Sea Ice Polar Stereographic North Change

Input coordinates

X: -1897013.64

Y: 742410.943

Show position on a map

Using best available transformer Change

Unit: metre

Transform

Swap ⇄

Output coordinate system

EPSG:4326 WGS 84 Change

Output coordinates

Longitude: -156°22'23.788"

Latitude: 71°21'16.144"

Format: D°M'S"

Show position on a map

Unit: degree (supplier to define represent

Area of use: World: Afghanistan, Albania,

Online converter

Input coordinate system

EPSG:3413 WGS 84 / NSIDC Sea Ice Polar Stereographic North Change

Input coordinates

X: -1890008.237

Y: 758631.307

Show position on a map

Using best available transformer Change

Unit: metre

Transform

Swap ⇄

Output coordinate system

EPSG:4326 WGS 84 Change

Output coordinates

Longitude: -156°52'12.202"

Latitude: 71°21'33.497"

Format: D°M'S"

Show position on a map

Unit: degree (supplier to define represen

Area of use: World: Afghanistan, Albania,

```python
    # When using PROJ. longitude first, latitude second, but it won't work, so I s
    transformer = Transformer.from_crs("EPSG:4326", "EPSG:3413")
    xmin, ymax = transformer.transform(bbox_mer[1], bbox_mer[0])
    xmax, ymin = transformer.transform(bbox_mer[3], bbox_mer[2])
    # source = osr.SpatialReference()
    # source.ImportFromEPSG(4326)
    # target = osr.SpatialReference()
    # target.ImportFromEPSG(3413)
    # transform = osr.CoordinateTransformation(source, target)
    # xmin, ymin, z = transform.TransformPoint(bbox_mer[0], bbox_mer[1])
    # xmax, ymax, z = transform.TransformPoint(bbox_mer[2], bbox_mer[3])
    df = pd.DataFrame(
        {
            "label": ["min", "max"],
            "Latitude": [bbox_mer[1], bbox_mer[3]],
            "Longitude": [bbox_mer[0], bbox_mer[2]],
        })
    gdf = gpd.GeoDataFrame(
        df, geometry=gpd.points_from_xy(df.Longitude, df.Latitude), crs="EPSG:4326"
    )
    gdf['geometry'] = gdf['geometry'].to_crs(epsg=3413)
    print(gdf.geometry)
    print(xmin, xmax, ymin, ymax)
    return xmin, xmax, ymin, ymax

def lat_long_to_BBOX(lat, long, zoom_level):
```

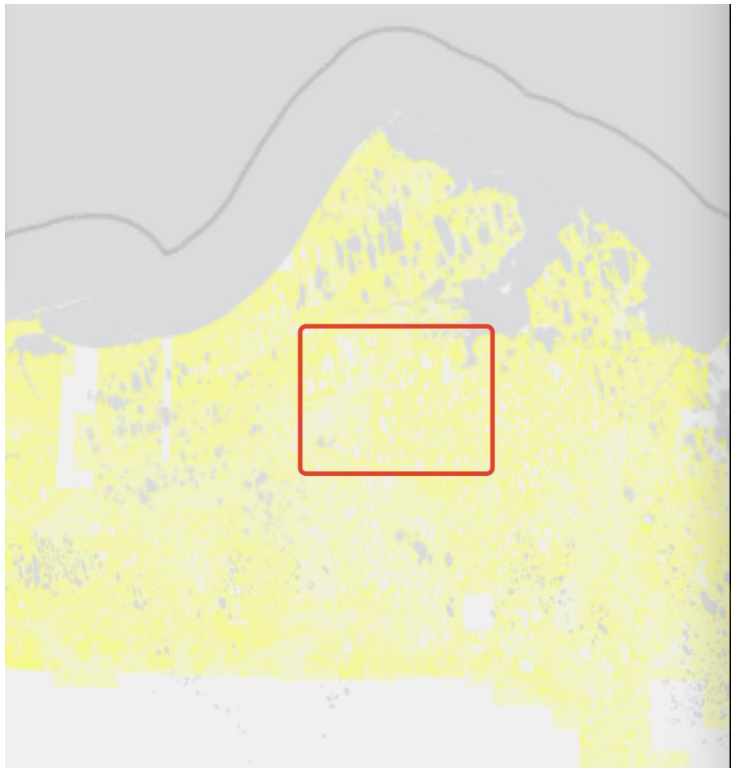
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
Tile information:Tile(x=66, y=217, z=10)
--------------The bounding box of this location is: -156.796875, 71.30079291637452, -156.4453125, 71.41317683396565------
-----
0     POINT (-1897013.64 758631.307)
1    POINT (-1890008.237 742410.943)
Name: geometry, dtype: geometry
-1897013.6399017135 -1890008.2366487072 742410.9429881338 758631.3072905991
SELECT centroidx, centroidy, geom FROM alaska_146_157_167_168_v2 WHERE centroidx > -1897013.6399017135 AND centroidx < -1
890008.2366487072 AND centroidy > 742410.9429881338 AND centroidy < 758631.3072905991;
```

Get all tiles within the ROI

Returns all tiles at this level

```
[Tile(x=65, y=221, z=10), Tile(x=65, y=222, z=10), Tile(x=65, y=223, z=10), Tile(x=65, y=224, z=10), Tile(x=66, y=221, z=
10), Tile(x=66, y=222, z=10), Tile(x=66, y=223, z=10), Tile(x=66, y=224, z=10), Tile(x=67, y=221, z=10), Tile(x=67, y=222
, z=10), Tile(x=67, y=223, z=10), Tile(x=67, y=224, z=10), Tile(x=68, y=221, z=10), Tile(x=68, y=222, z=10), Tile(x=68, y
=223, z=10), Tile(x=68, y=224, z=10)]
------------------The number of tiles within the geographic bbox is: 16------------------
```

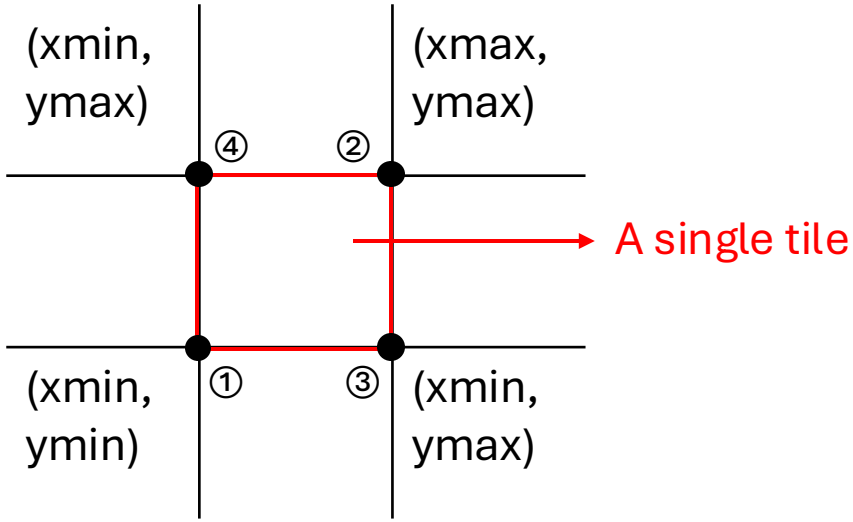


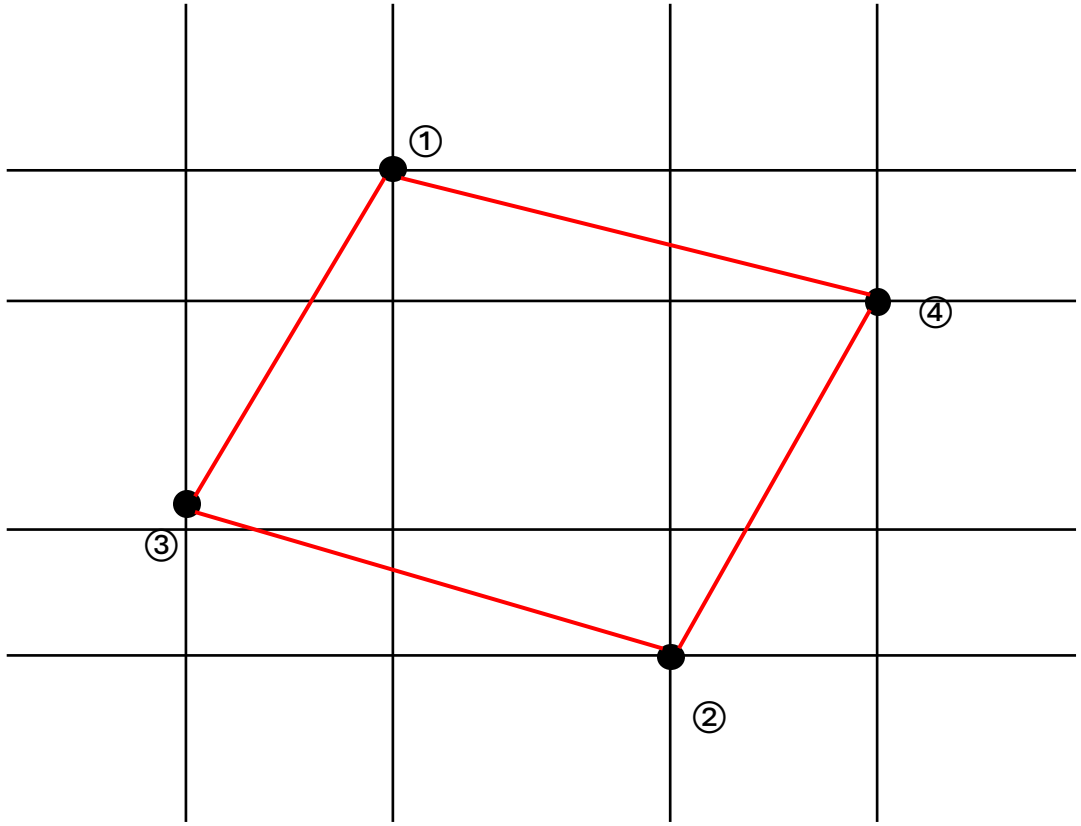
Meeting (8/14/2024)

BBOX -> Polygon shape

WKT: Arc

BBOX transformation



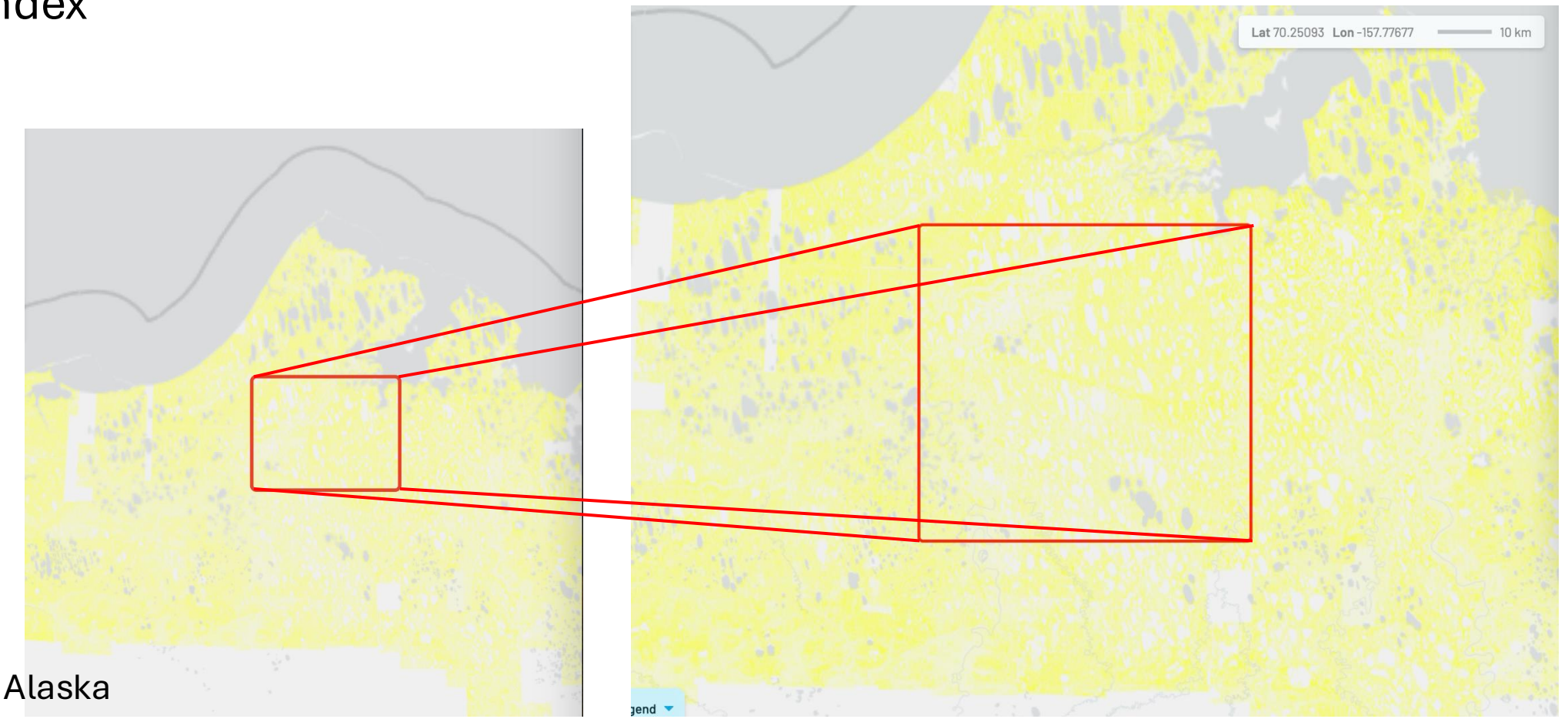


EPSG: 4326

EPSG: 3413

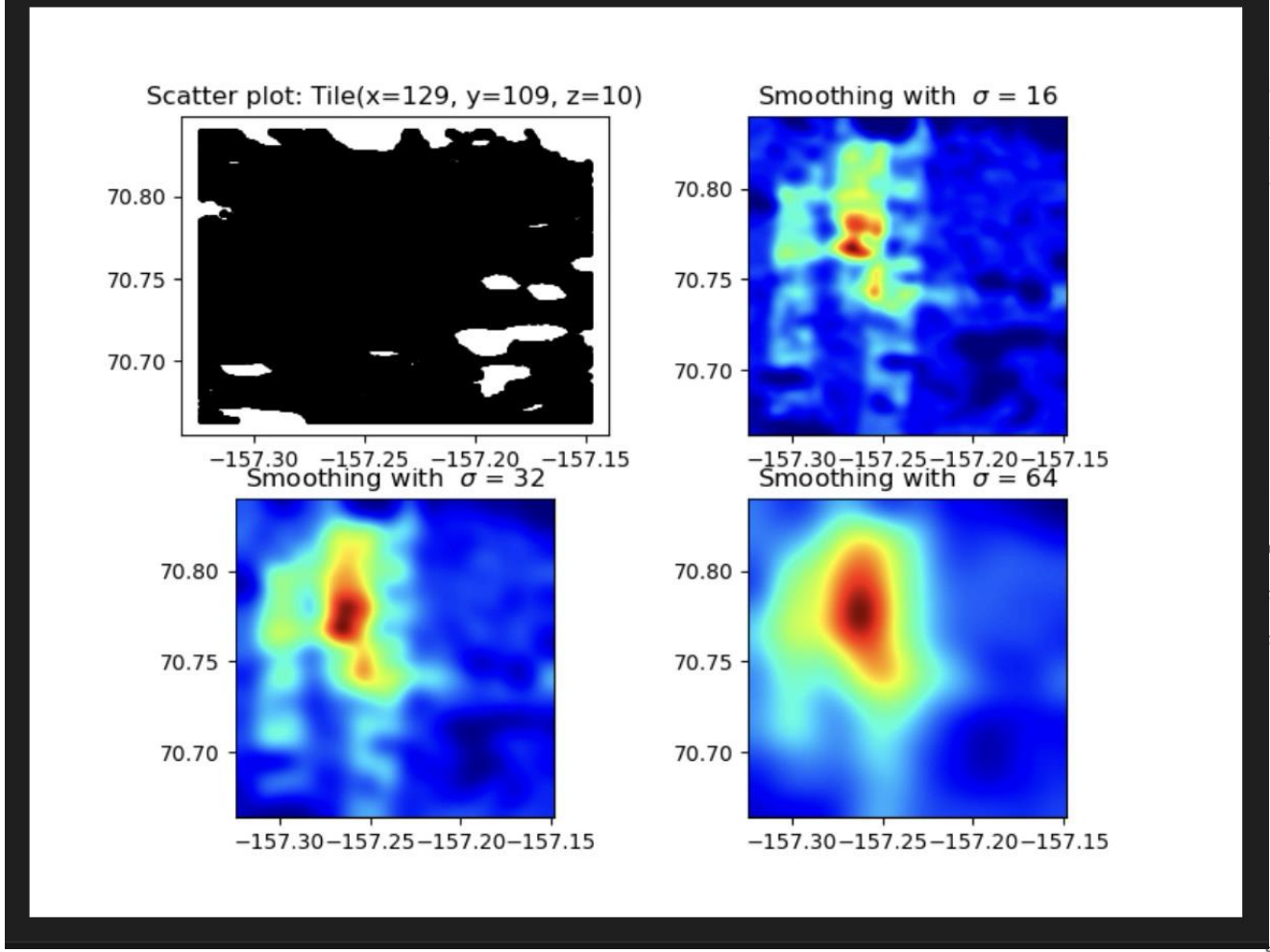
Meeting (8/20/2024)

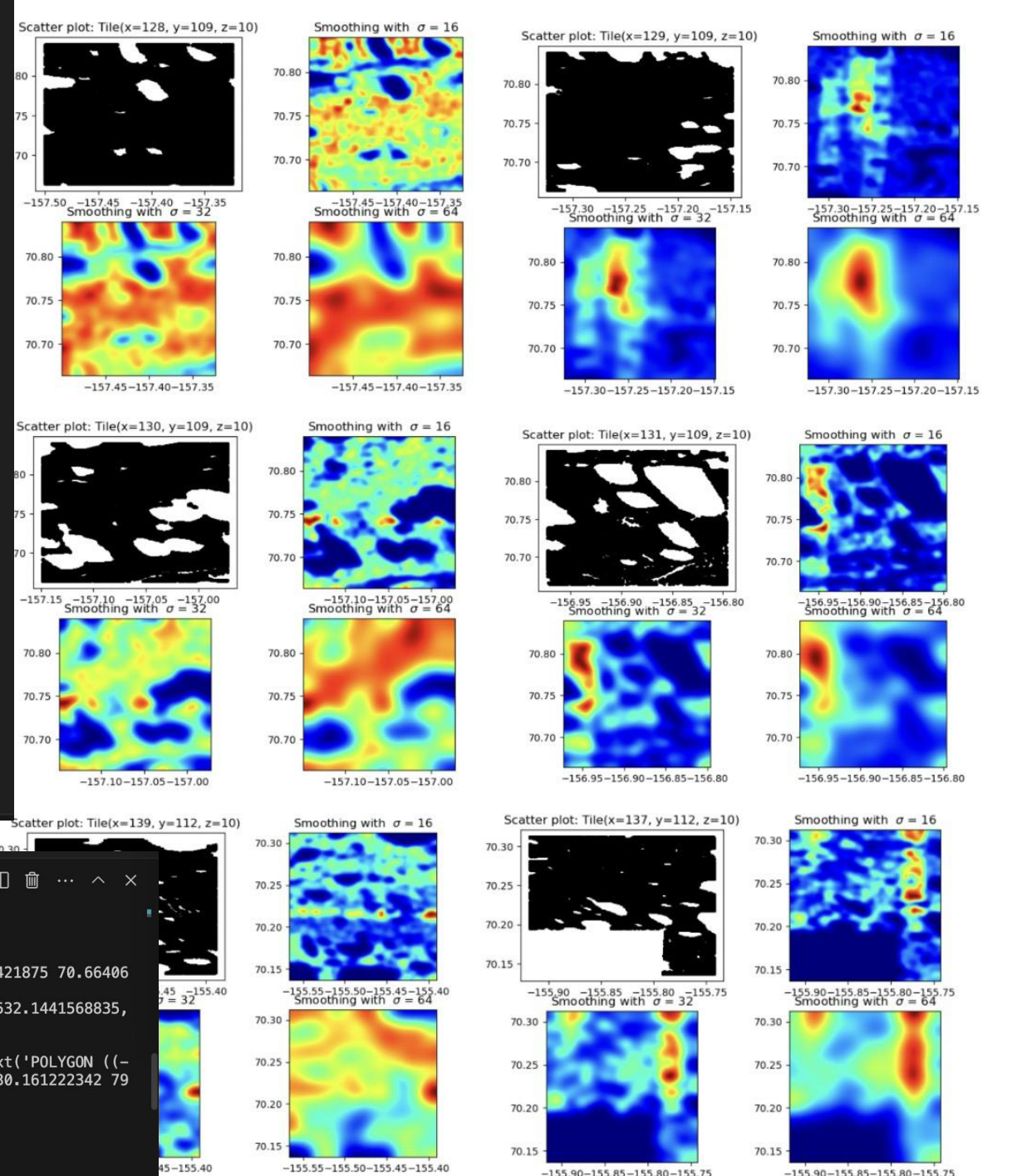
1. Import all of the data into database (polygon-based, EPSG: 3413)
2. Add spatial index



43,000,000

119,124,921 Records in Alaska





PROBLEMS OUTPUT TERMINAL PORTS python - shpMapping

```
Tile(x=129, y=109, z=10)
2
----------The bounding box of this location is: -157.32421875, 70.6640625, -157.1484375, 70.83984375----------
POLYGON ((-157.32421875 70.6640625, -157.32421875 70.83984375, -157.1484375 70.83984375, -157.1484375 70.6640625, -157.32421875 70.6640625))
----------The reprojected bbox is : POLYGON ((-1955475.9024936033 802964.8919575055, -1937374.7882867341 795532.1441568835, -1939806.328853881 789584.6181470009, -1957930.161222342 796961.7975319484, -1955475.9024936033 802964.8919575055))
----------The query is:----------
SELECT centroidx, centroidy, geom FROM alaska_all_3413 polygons WHERE ST_within(ST_centroid(polygons.geom), ST_GeomFromText('POLYGON ((-1955475.9024936033 802964.8919575055, -1937374.7882867341 795532.1441568835, -1939806.328853881 789584.6181470009, -1957930.161222342 796961.7975319484, -1955475.9024936033 802964.8919575055))', 3413))
Extracting polygon data from database...
----------Data Extracted! The runtime of getting data from database is 789.1875863075256----------
Creating heatmap...
Creating heatmap...
----------Heatmap Created!----------
```

Meeting (8/27/2024)

- Spatial index
- Heatmap border effect
- Color schema (heatmap value range)
- Scalibility

	Index	vacuum	Database size	Speed
Polygon	No	No	192GB	~800s
Polygon	Yes	No		~800s
Polygon + Point	No	Yes	376GB	~1200s
Point	No	Yes	376GB	
Point	Yes	Yes	382GB	2~12s

Query | Query History

```
1  SELECT indexname, indexdef FROM pg_indexes WHERE tablename = 'alaska_all_3413' AND schemaname = 'public';
2
```

Data Output | Messages | Notifications

SQL

	indexname name	indexdef text
1	alaska_all_3413_pkey	CREATE UNIQUE INDEX alaska_all_3413_pkey ON public.alaska_all_3413 USING btree (g...
2	alaska_ind_3413	CREATE INDEX alaska_ind_3413 ON public.alaska_all_3413 USING gist (geom)

With spatial index

Tile(x=128, y=109, z=10)
1
----------The bounding box of this location is: -157.5, 70.6640625, -157.32421875, 70.83984375----------
POLYGON ((-157.5 70.6640625, -157.5 70.83984375, -157.32421875 70.83984375, -157.32421875 70.6640625, -157.5 70.6640625))
----------The reprojected bbox is :
POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))
----------The query is:----------
SELECT centroidx, centroidy, geom
FROM alaska_all_3413 polygons
WHERE ST_within(ST_centroid(polygons.geom), ST_GeomFromText('POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))', 3413))
Extracting polygon data from database...
----------Data Extracted! The runtime of getting data from database is 830.5624363422394----------
Creating heatmap...

```
1  SELECT indexname, indexdef FROM pg_indexes WHERE tablename = 'alaska_all_3413' AND schemaname = 'public';
```

Data Output | Messages | Notifications

	indexname name	indexdef text
1	alaska_all_3413_pkey	CREATE UNIQUE INDEX alaska_all_3413_pkey ON public.alaska_all_3413 USING btree (g...

Without spatial index

Tile(x=128, y=109, z=10)
1
----------The bounding box of this location is: -157.5, 70.6640625, -157.32421875, 70.83984375----------
POLYGON ((-157.5 70.6640625, -157.5 70.83984375, -157.32421875 70.83984375, -157.32421875 70.6640625, -157.5 70.6640625))
----------The reprojected bbox is : POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))
----------The query is:----------
SELECT centroidx, centroidy, geom
FROM alaska_all_3413 polygons
WHERE ST_within(ST_centroid(polygons.geom), ST_GeomFromText('POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))', 3413))
Extracting polygon data from database...
----------Data Extracted! The runtime of getting data from database is 797.6694705486298----------
Creating heatmap...

Create a new geom column with centroid points as the geometry

Create the column

```
SELECT AddGeometryColumn('public', 'alaska_all_3413','geom_centroid',3413,'POINT',2) ;
```

Update the values in the column

```
update public.Alaska_all_3413
set geom = ST_SetSRID(
ST_MakePoint(
"Longitude"::double precision,
"Latitude"::double precision
), 4326)
```

Without index

Query | Query History

```
1  SELECT pg_size_pretty( pg_total_relation_size('alaska_all_3413') );
```

Data Output | Messages | Notifications

	pg_size_pretty text
1	192 GB

```
Tile(x=128, y=109, z=10)
1
--------------The bounding box of this location is: -157.5, 70.6640625, -157.32421875, 70.83984375
-----------
POLYGON ((-157.5 70.6640625, -157.5 70.83984375, -157.32421875 70.83984375, -157.32421875 70.66406
25, -157.5 70.6640625))
-----------------------The reprojected bbox is : POLYGON ((-1953003.2380809318 808960.4285716926,
-1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033
802964.8919575055, -1953003.2380809318 808960.4285716926))
------------------The query is:--------------
SELECT centroidx, centroidy, geom FROM alaska_all_3413 polygons WHERE ST_within(ST_centroid(polygo
ns.geom), ST_GeomFromText('POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 80
1472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -19
53003.2380809318 808960.4285716926))', 3413))
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarn
ing: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sql
ite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(
---------------Data Extracted! The runtime of getting data from database is 807.6313257217407------
--------
Creating heatmap...
```

Without Index, but contains polygon and point geometry column

```
1   SELECT pg_size_pretty( pg_total_relation_size('alaska_all_3413') );
```

Data Output | Messages | Notifications

	pg_size_pretty text
1	376 GB

Without Index, delete polygon geometry, but have point geometry column

```
1   SELECT pg_size_pretty( pg_total_relation_size('alaska_all_3413') );
```

Data Output | Messages | Notifications

	pg_size_pretty text
1	376 GB

1

----------The bounding box of this location is: -157.5, 70.6640625, -157.32421875, 70.83984375----------

POLYGON ((-157.5 70.6640625, -157.5 70.83984375, -157.32421875 70.83984375, -157.32421875 70.6640625, -157.5 70.6640625))

----------The reprojected bbox is : POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))

----------The query is:----------

SELECT centroidx, centroidy, geom_centroid FROM alaska_all_3413 polygons WHERE ST_within(polygons.geom_centroid, ST_GeomFromText('POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))', 3413))

Extracting polygon data from database...

/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

df = pd.read_sql(

----------Data Extracted! The runtime of getting data from database is 1351.8806235790253----------

With Index, and point geometry column

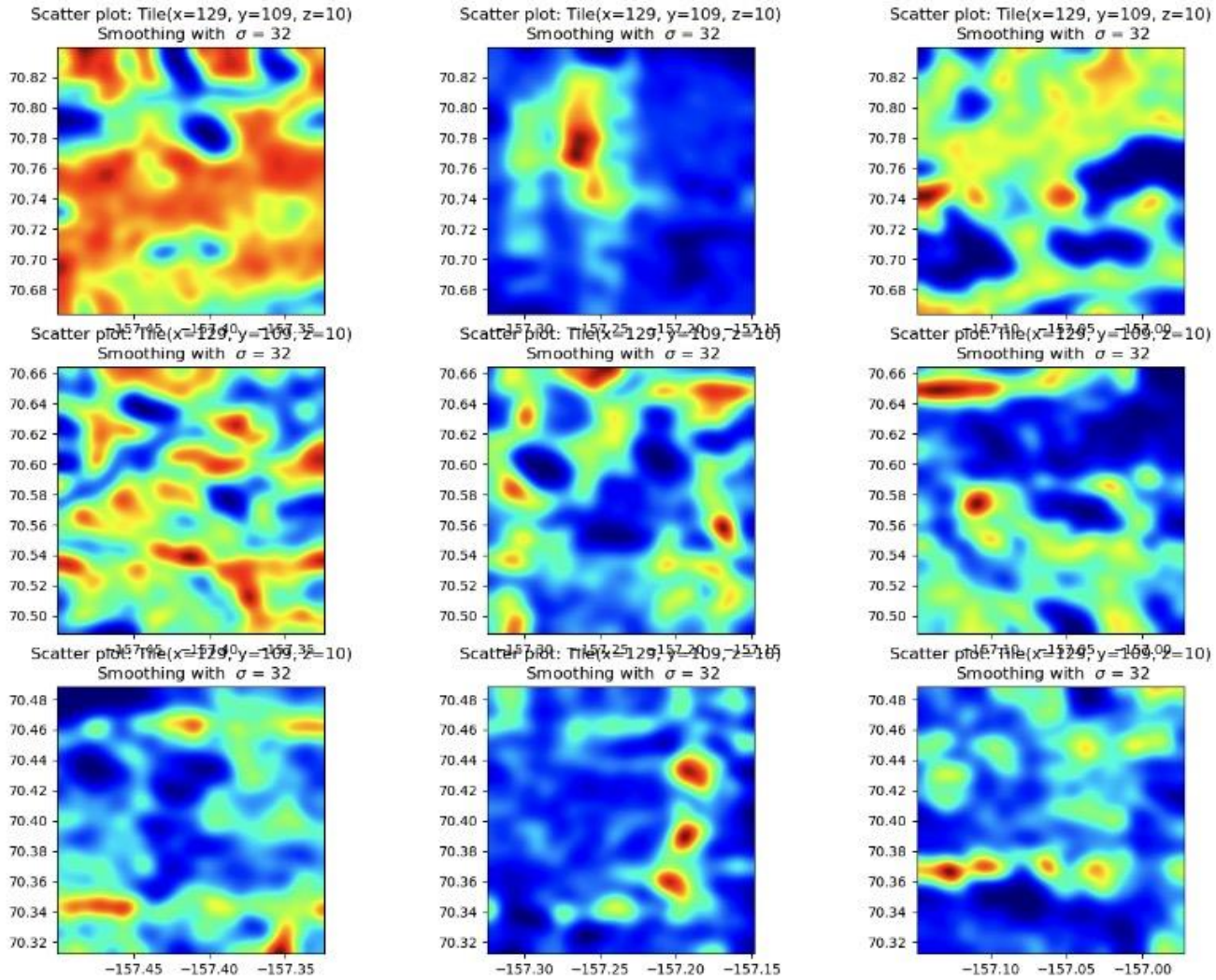
```
1   SELECT pg_size_pretty( pg_total_relation_size('alaska_all_3413') );
```

Data Output | Messages | Notifications

	pg_size_pretty text
1	382 GB

Tile(x=128, y=109, z=10)
1
----------The bounding box of this location is: -157.5, 70.6640625, -157.32421875, 70.83984375----------
POLYGON ((-157.5 70.6640625, -157.5 70.83984375, -157.32421875 70.83984375, -157.32421875 70.6640625, -157.5 70.6640625))
----------The reprojected bbox is : POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))
----------The query is:----------
SELECT centroidx, centroidy, geom_centroid FROM alaska_all_3413 polygons WHERE ST_within(polygons.geom_centroid, ST_GeomFromText('POLYGON ((-1953003.2380809318 808960.4285716926, -1934925.0124102354 801472.1823152486, -1937374.7882867341 795532.1441568835, -1955475.9024936033 802964.8919575055, -1953003.2380809318 808960.4285716926))', 3413))
Extracting polygon data from database...
/home/xchen/code/alaska_shpMapping2/lib/python3.12/site-packages/geopandas/io/sql.py:185: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(
----------Data Extracted! The runtime of getting data from database is 7.471695423126221----------
Creating heatmap...

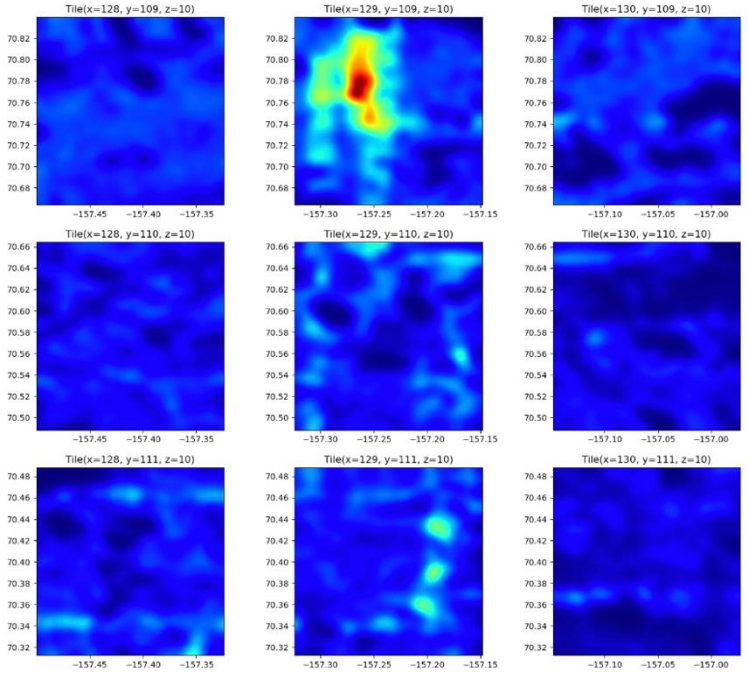
Heatmaps for 3 * 3 Tiles, haven't rescale yet

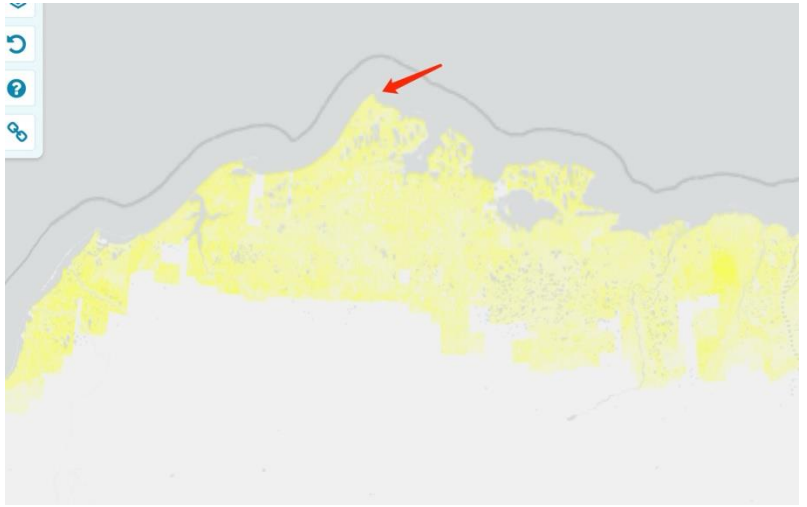


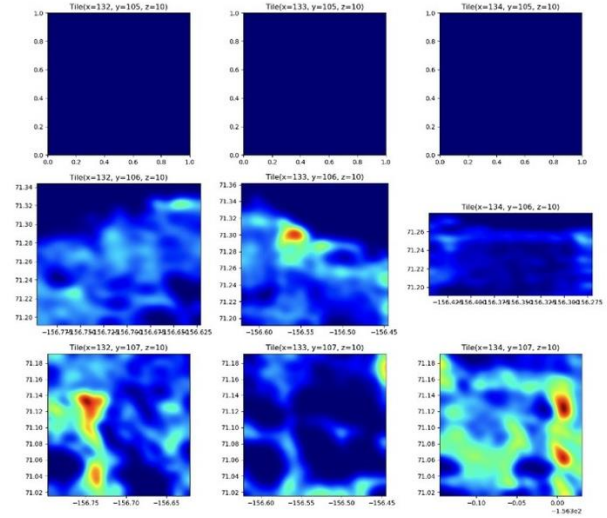
Meeting 09/03/2024 – 09/10/2024

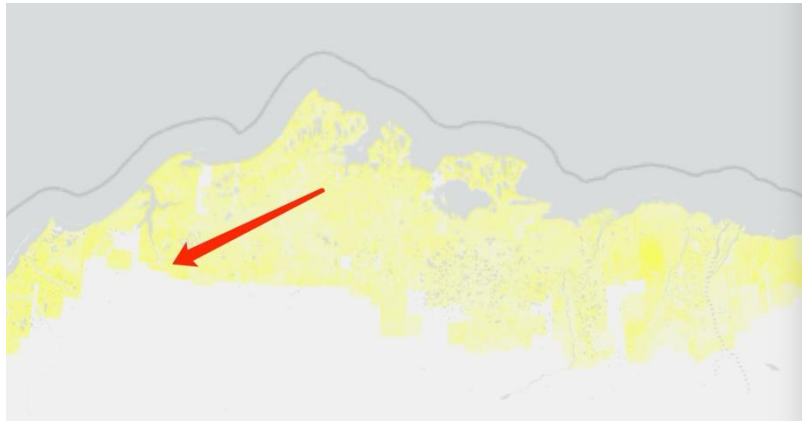
1. Rescale the color schema
2. Extent alignment
3. Border effect
4. Applied for each tile in a larger ROI (Scalability)

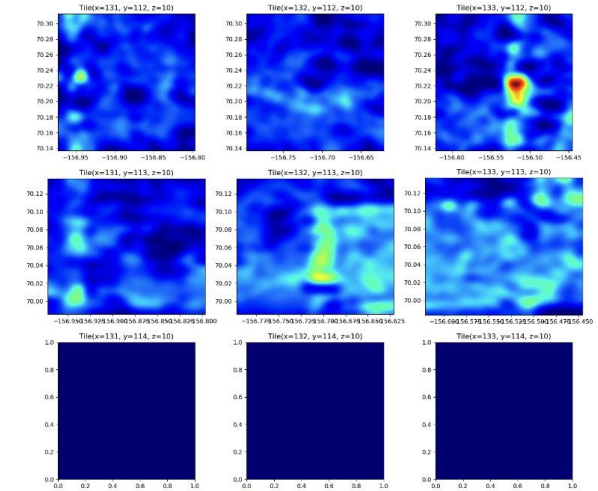
Rescale the range of color schema, based on global max and min





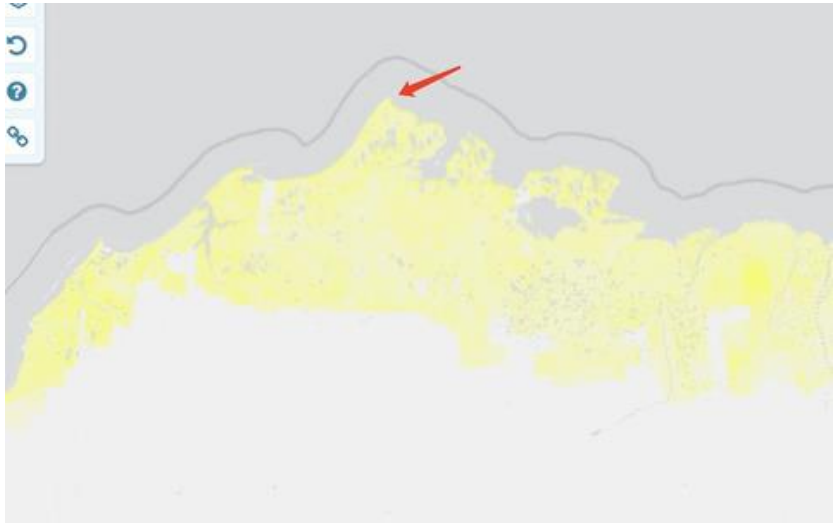


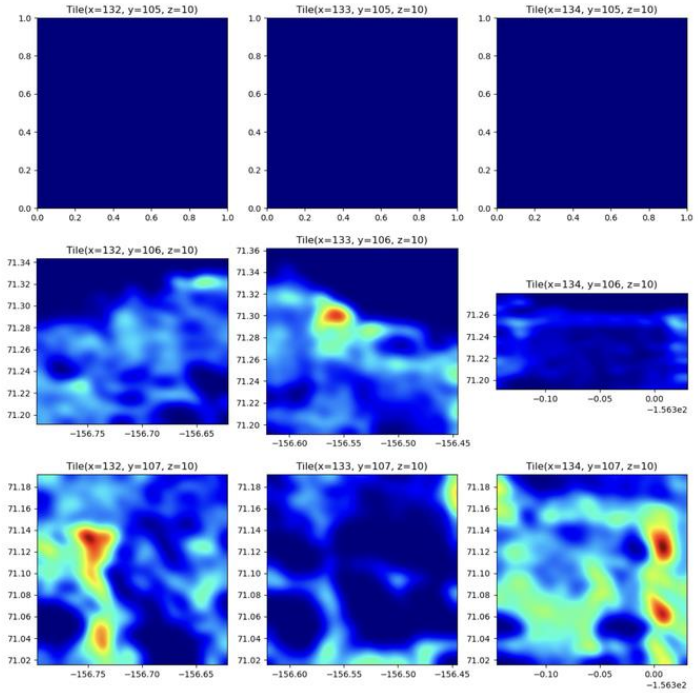


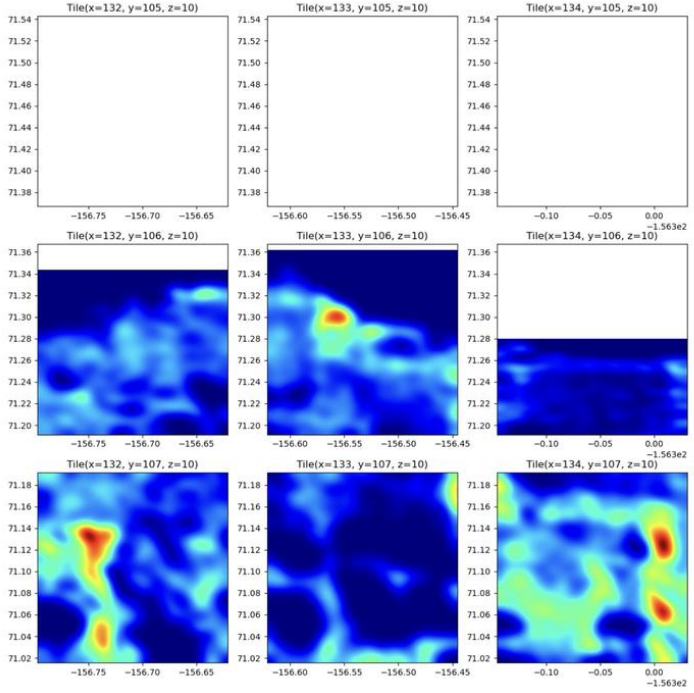


Extent alignment

Before extent alignment

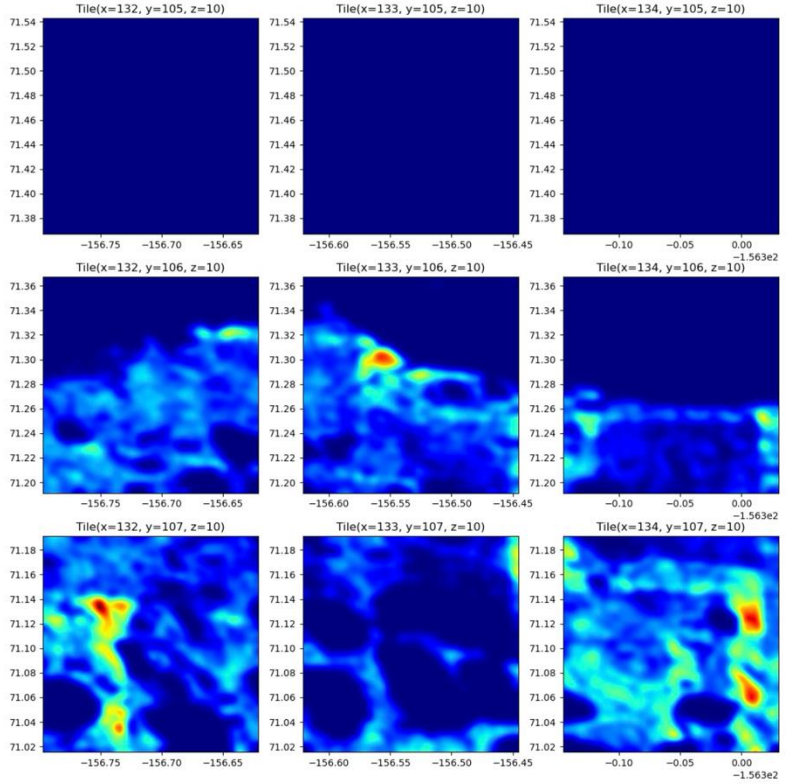


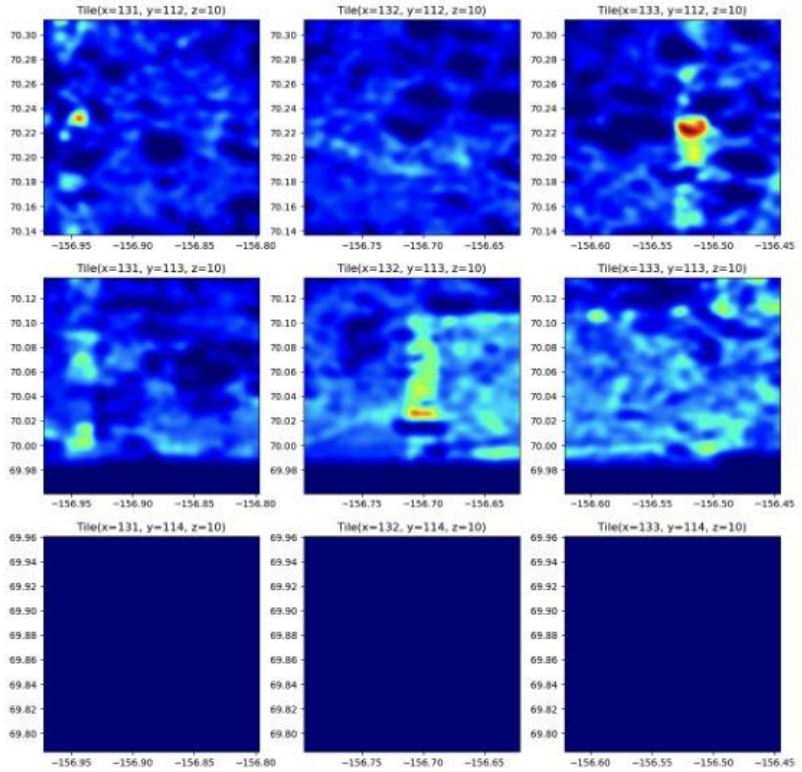


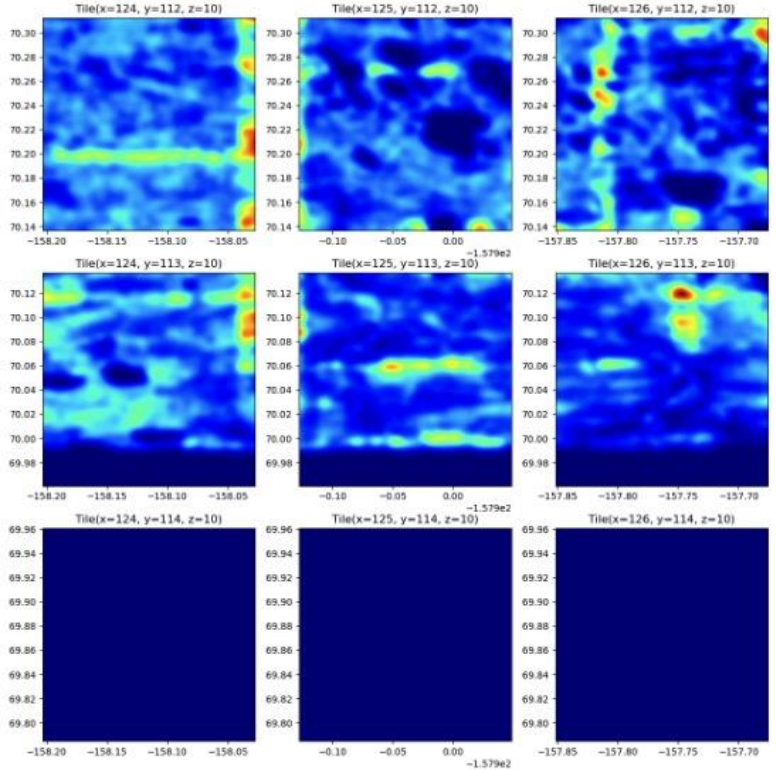


Extent alignment

Align the data extent returned from database with the tile extent where the heatmap is plotted within



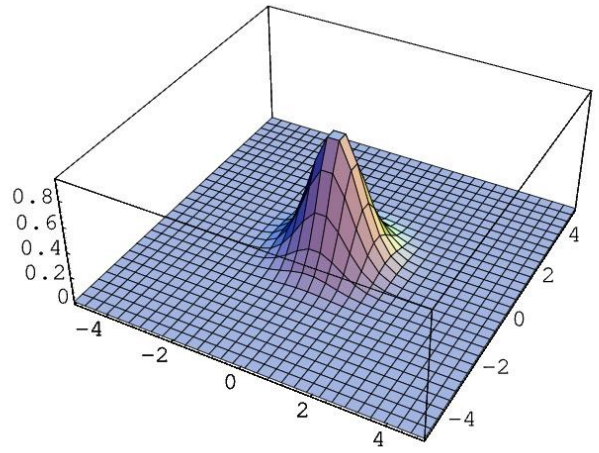




Border effect

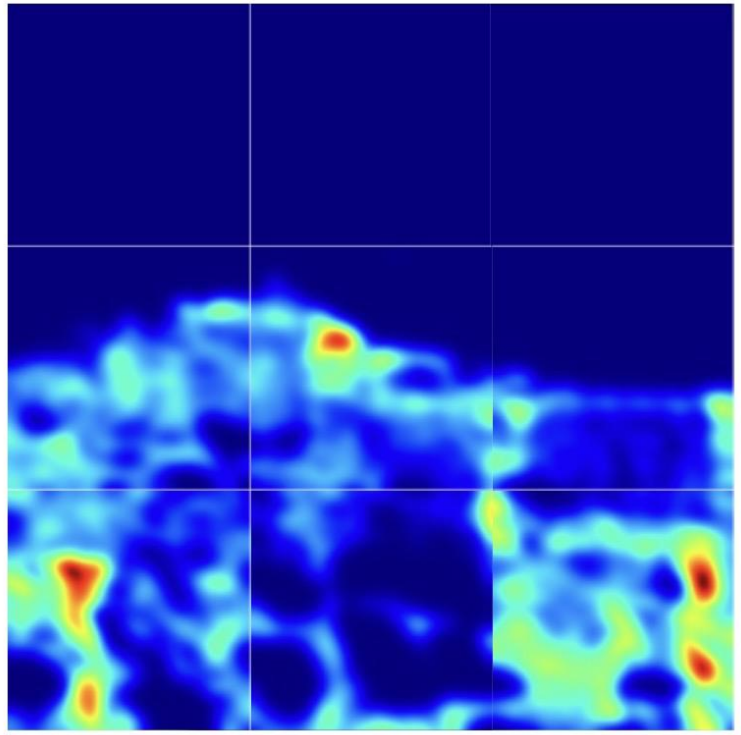
- “**Sigma**” is the standard deviation for Guassian kernal, controlling how much the weights in the Gaussian kernel decrease as you move away from the center.

- “**Radius**” is the radius of the Guassian kernal, controlling the actual size of the kernel, i.e., how far from the center the Gaussian function is evaluated.

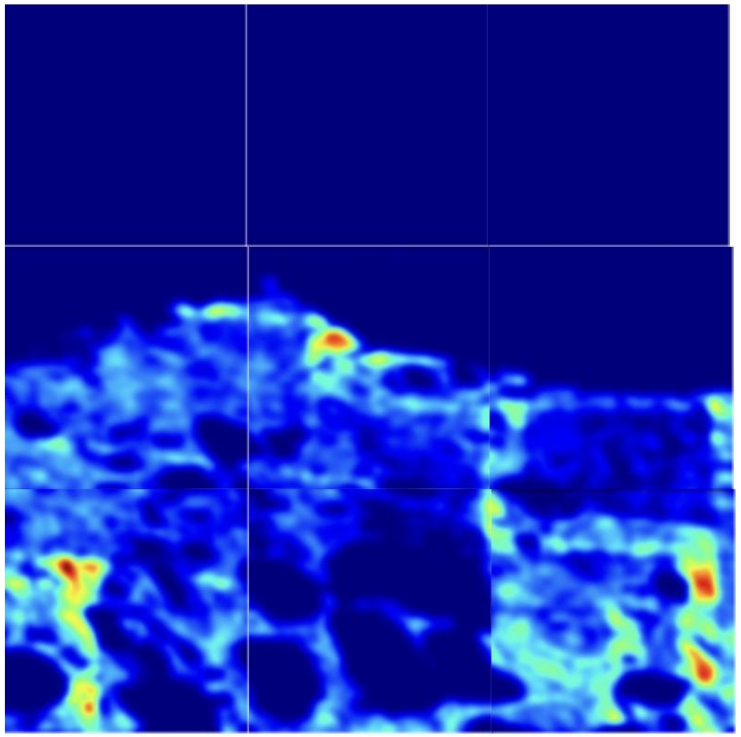


Border effect

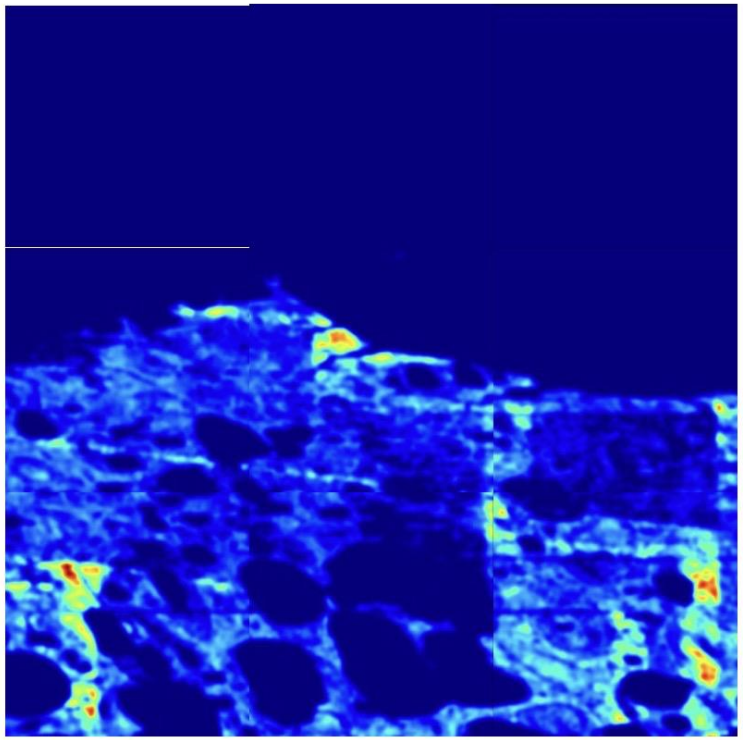
With **σ**, no radius



σ = 64



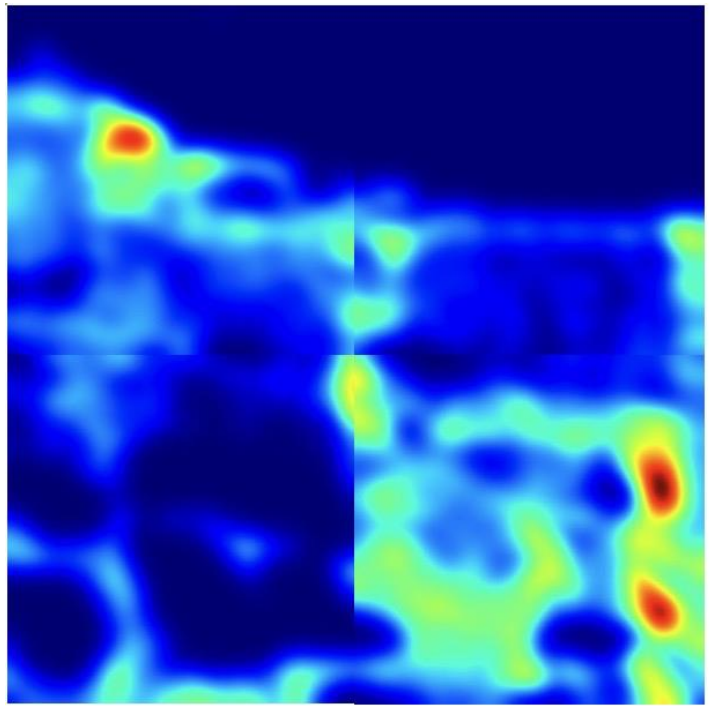
σ = 32



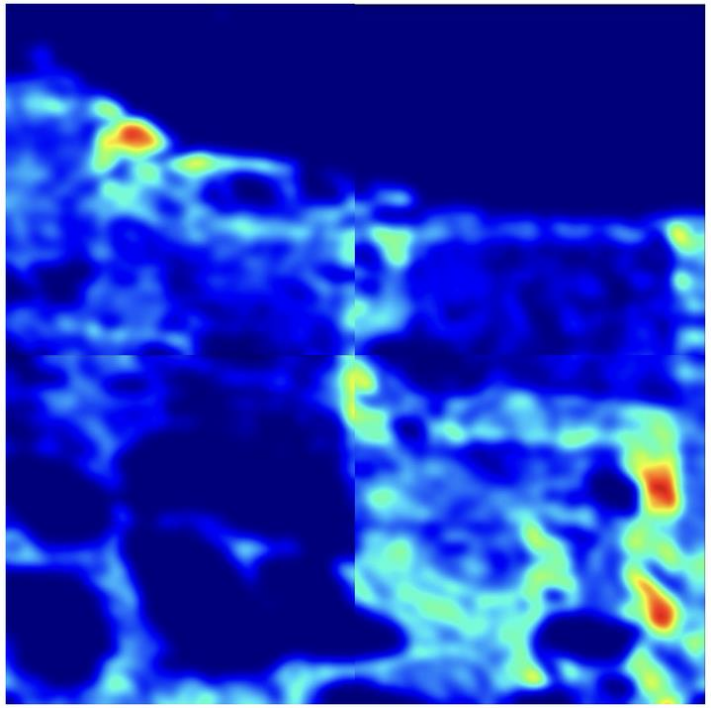
σ = 16

Border effect

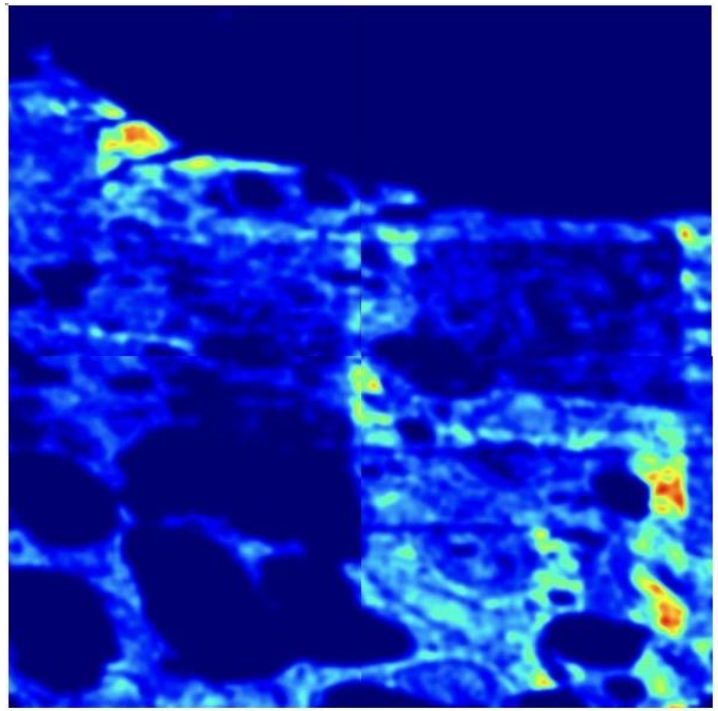
With **σ**, no radius



σ = 64



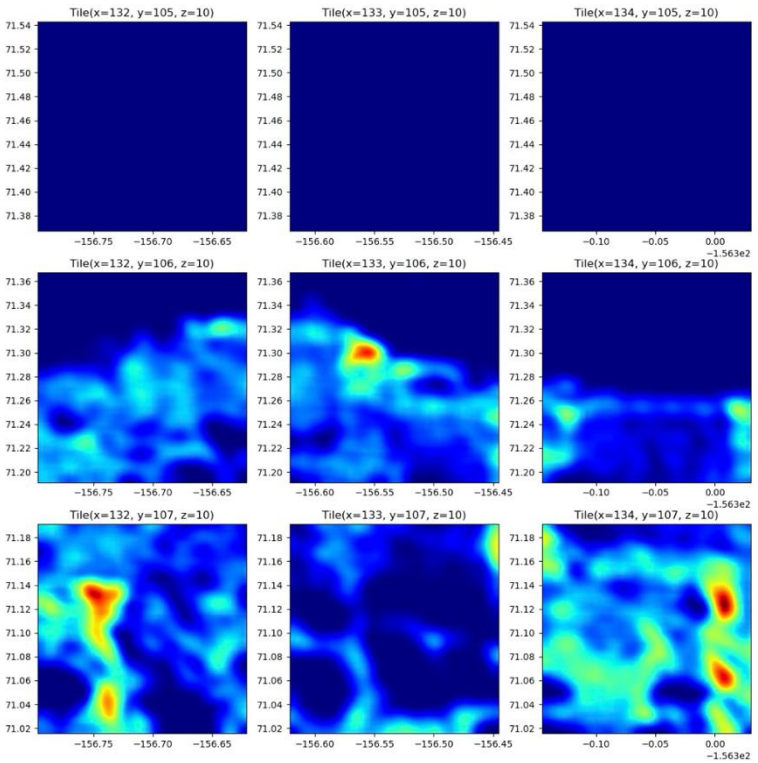
σ = 32



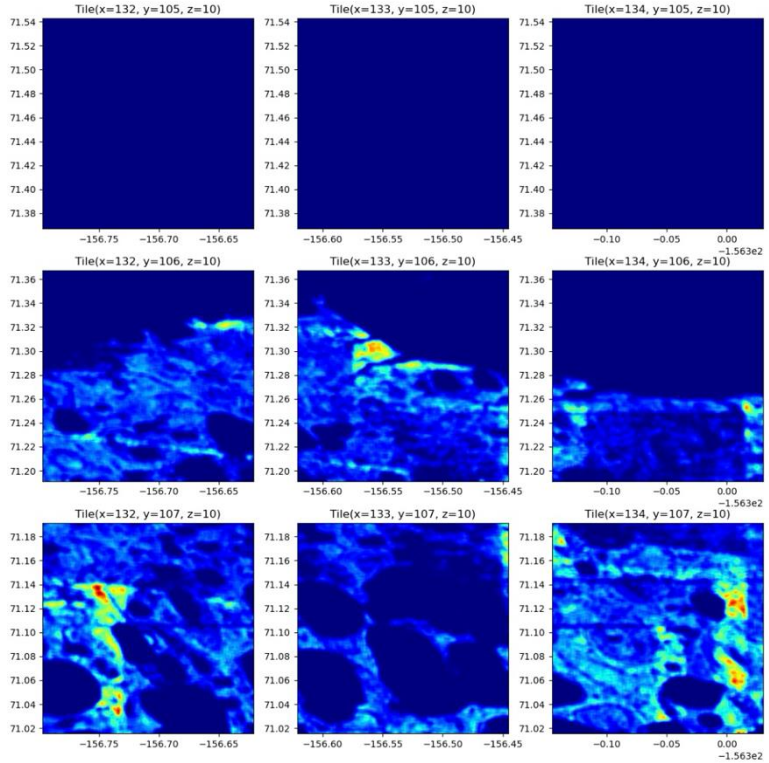
σ = 16

Border effect

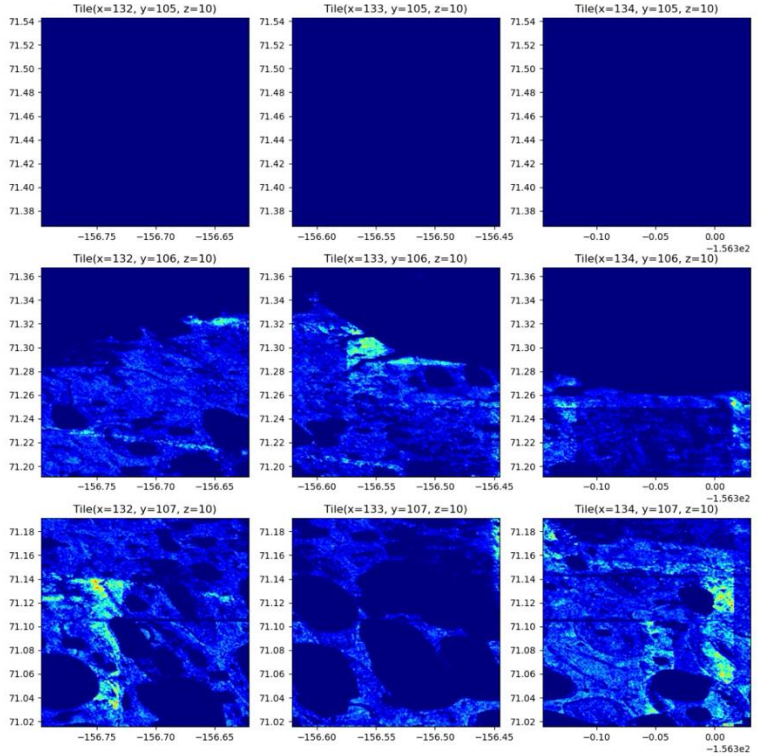
Sigma = 64



Radius = 100



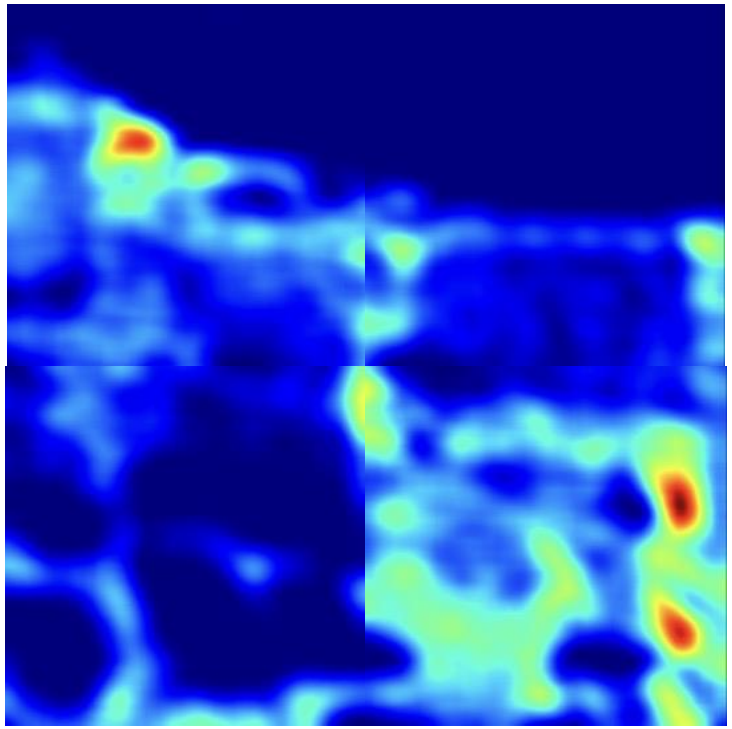
Radius = 20



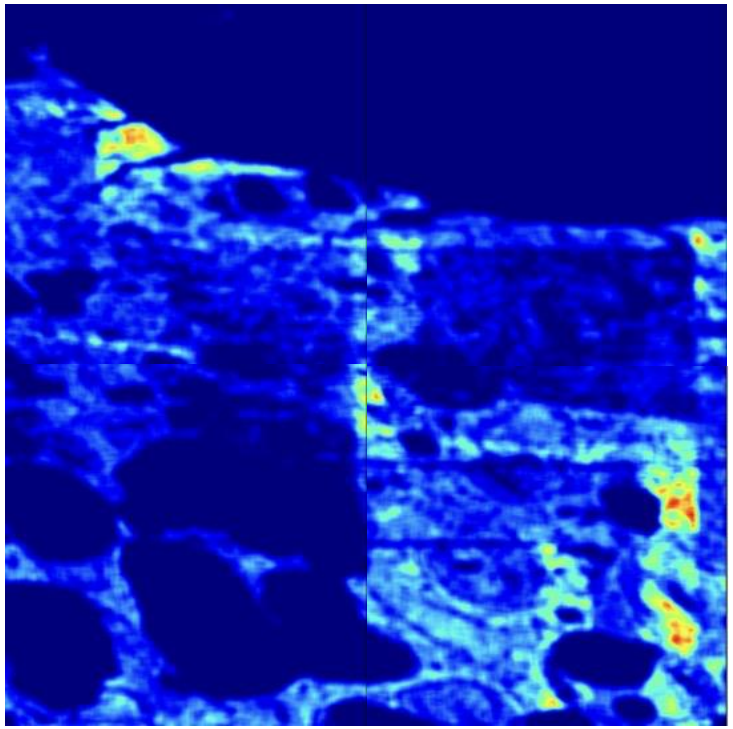
Radius = 5

Border effect

Sigma = 64



Radius = 100

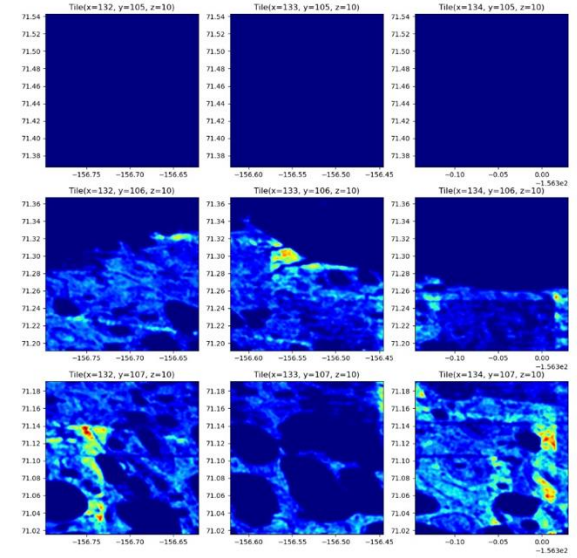


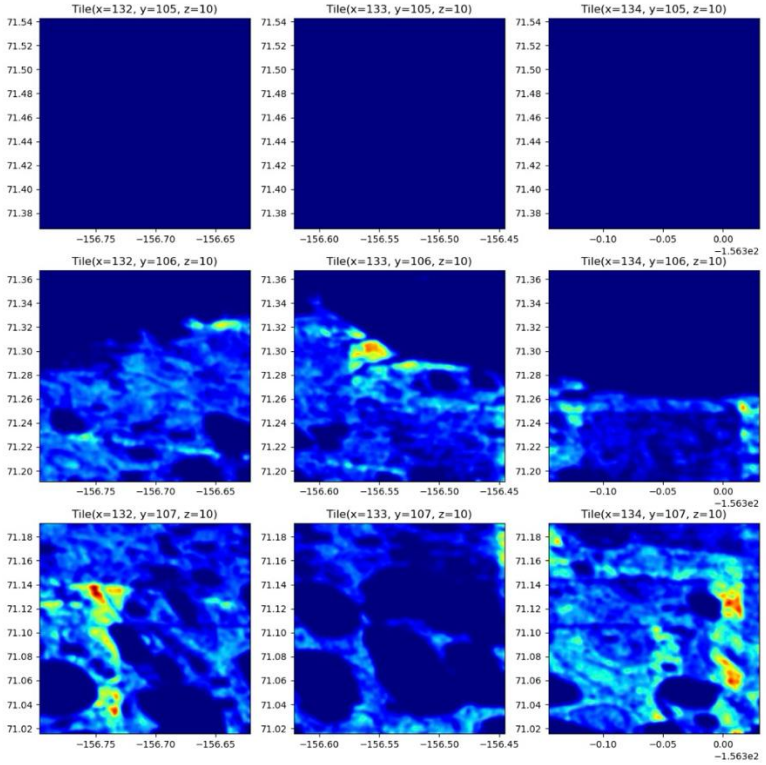
Radius = 20

Border effect

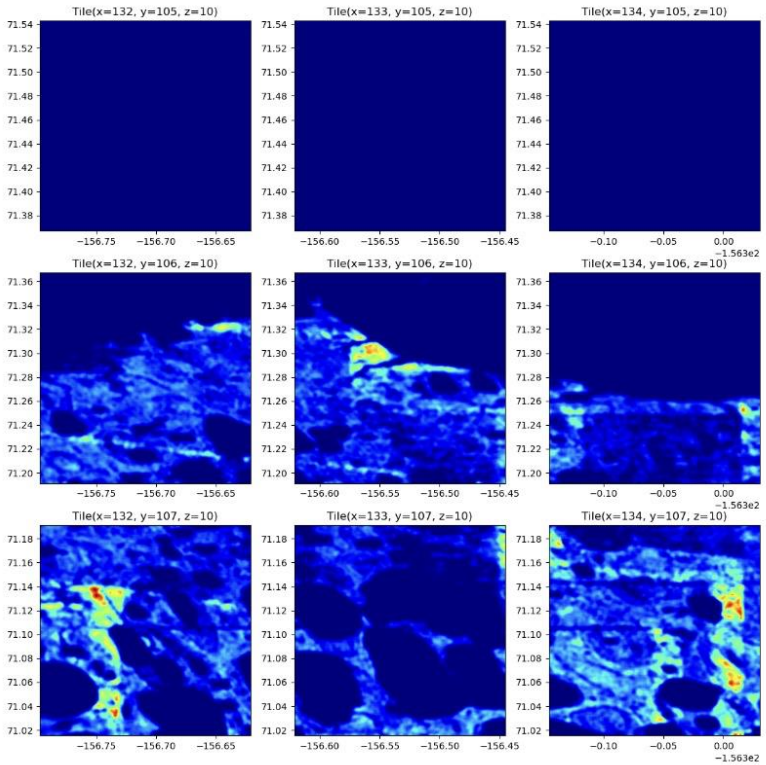
Sigma = 16

Radius = 10

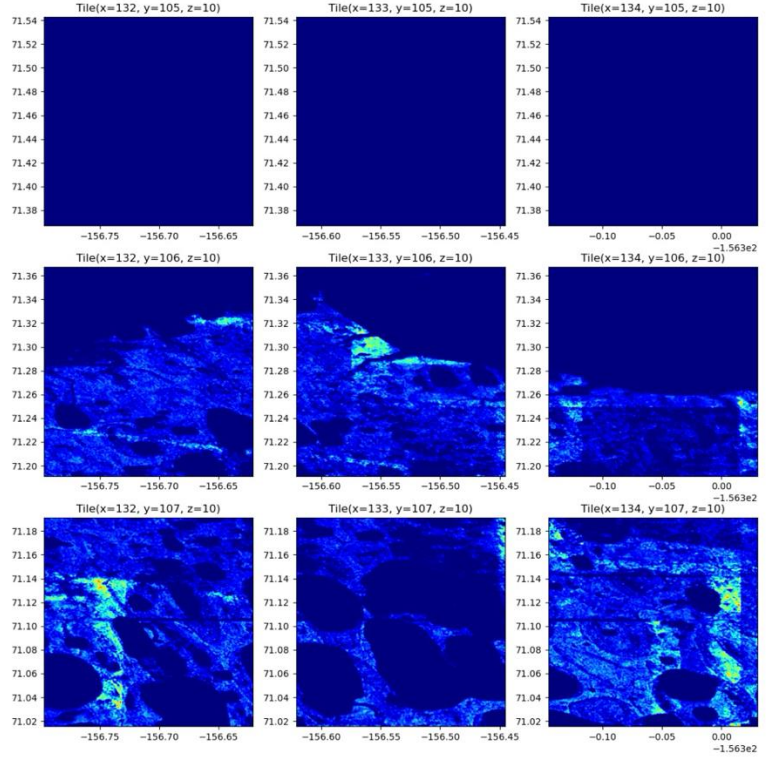




Radius = 100



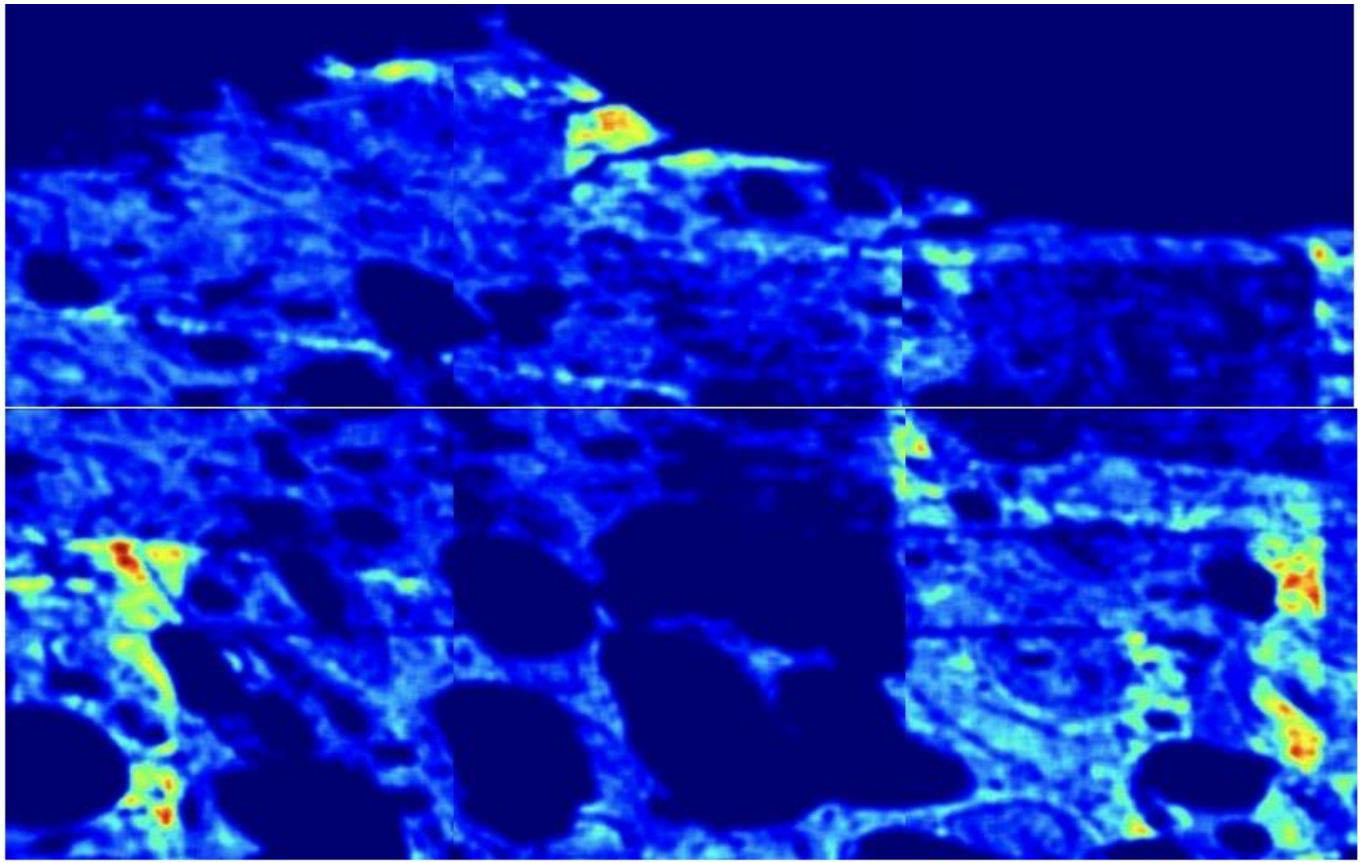
Radius = 20



Radius = 5

Border effect

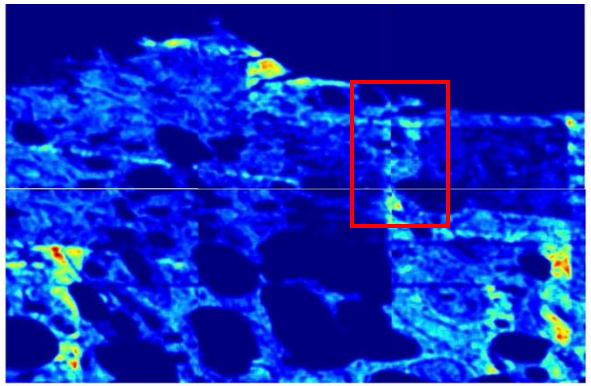
σ = 32, radius = 20

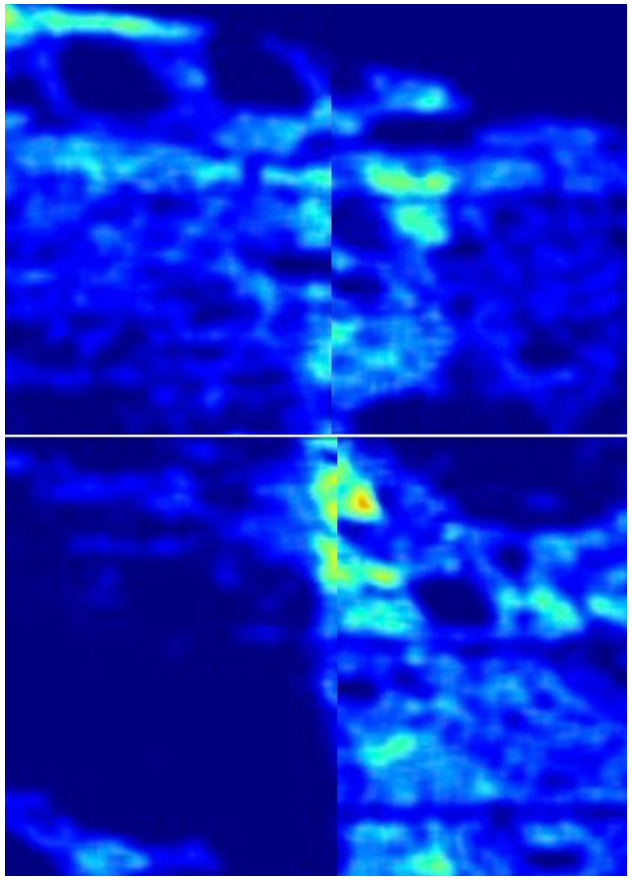


Border effect

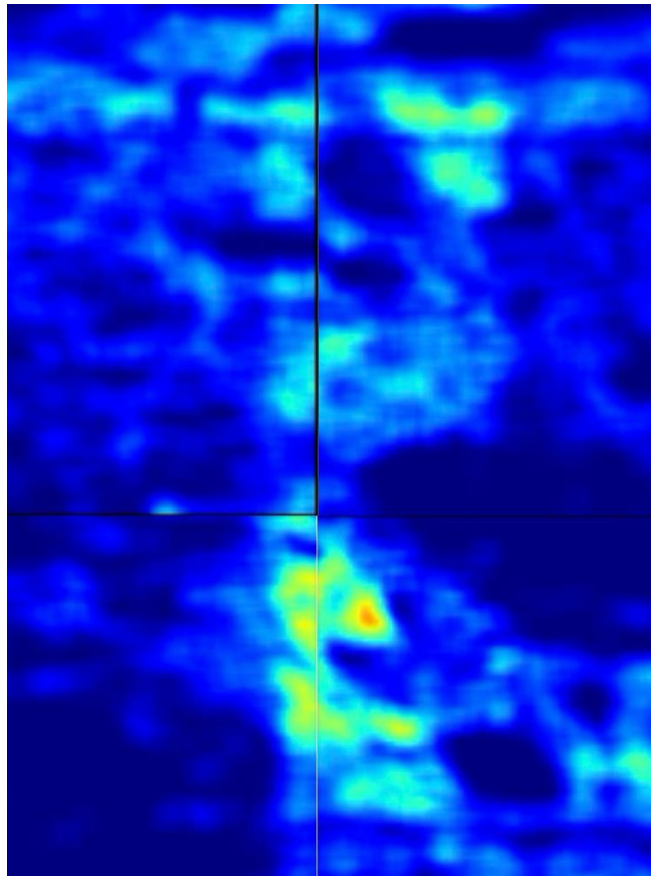
σ = 32

Radius = 20, no buffer

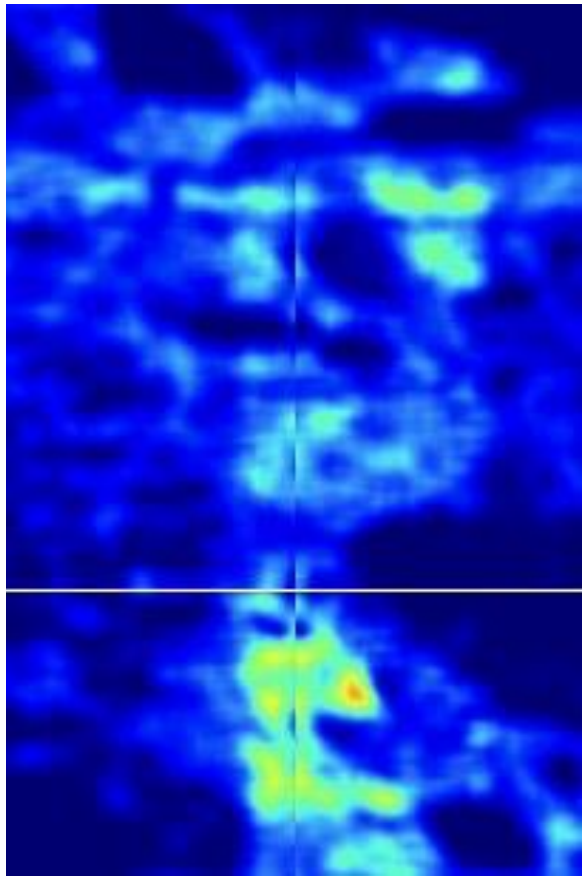




Radius = 20, no buffer



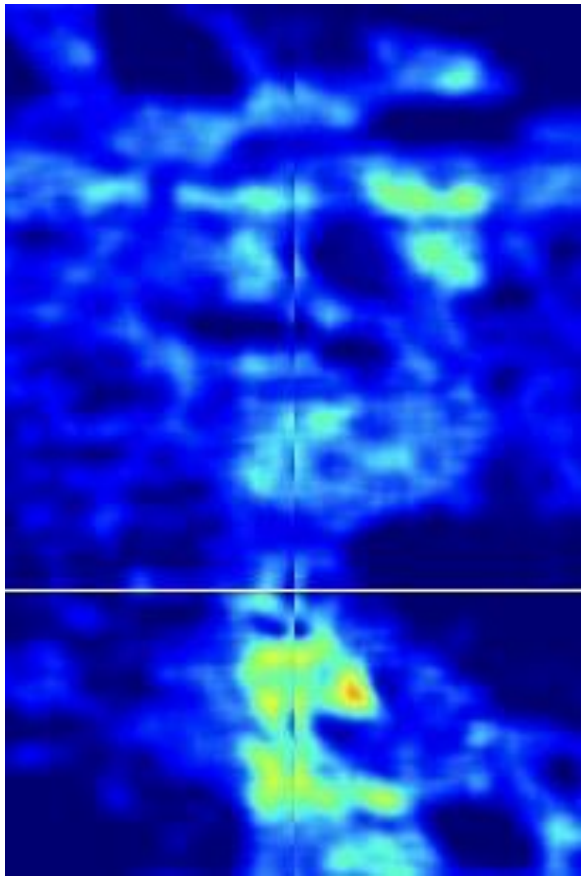
Radius = 20, buffer = 20



Radius = 20, buffer = 100

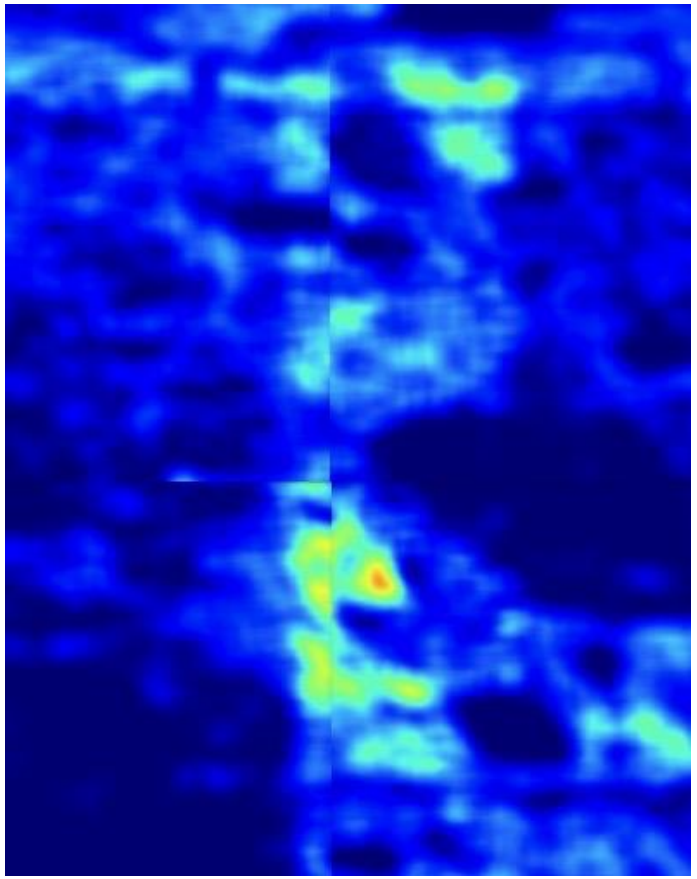
Border effect

σ = 32



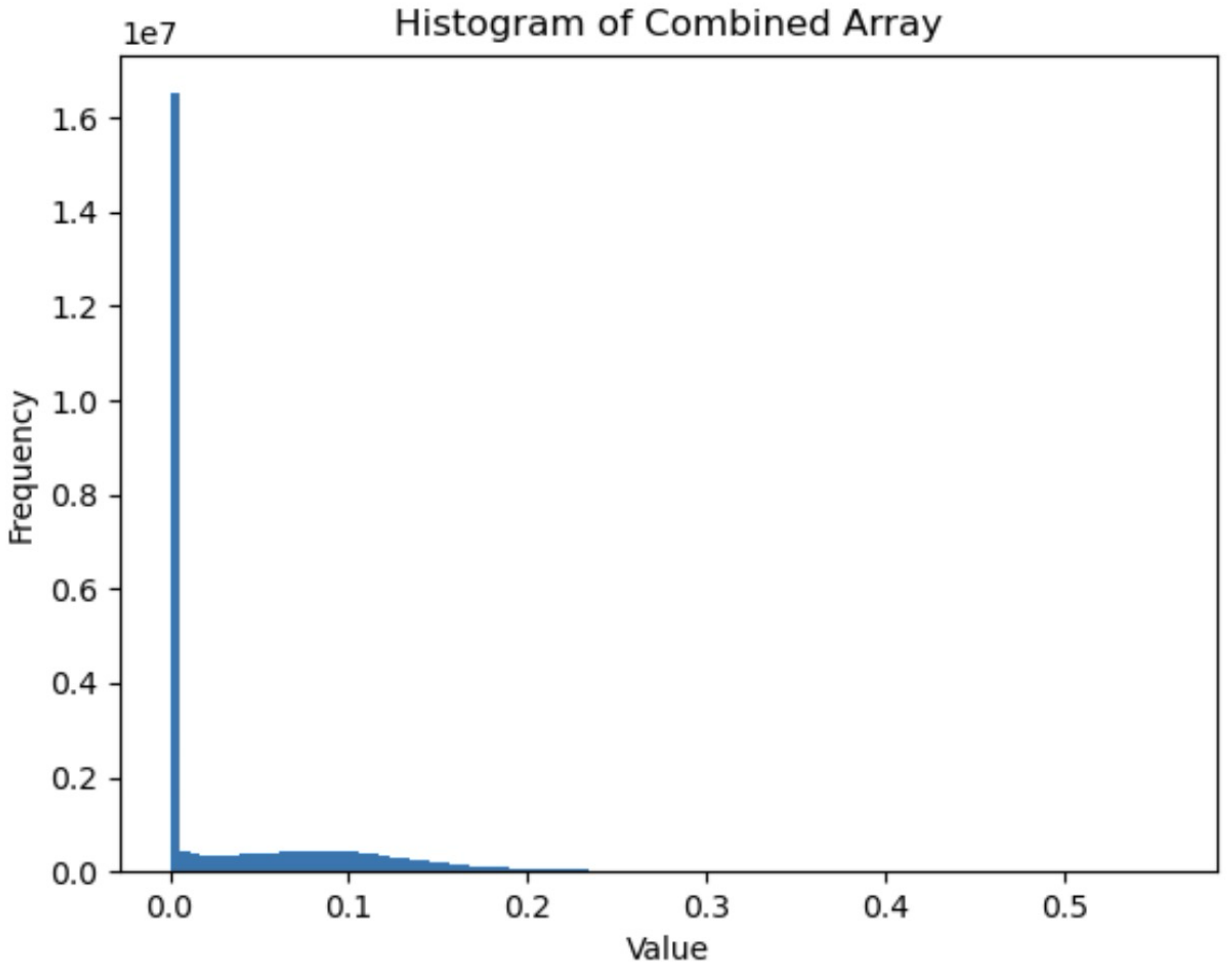
Radius = 20, buffer = 100

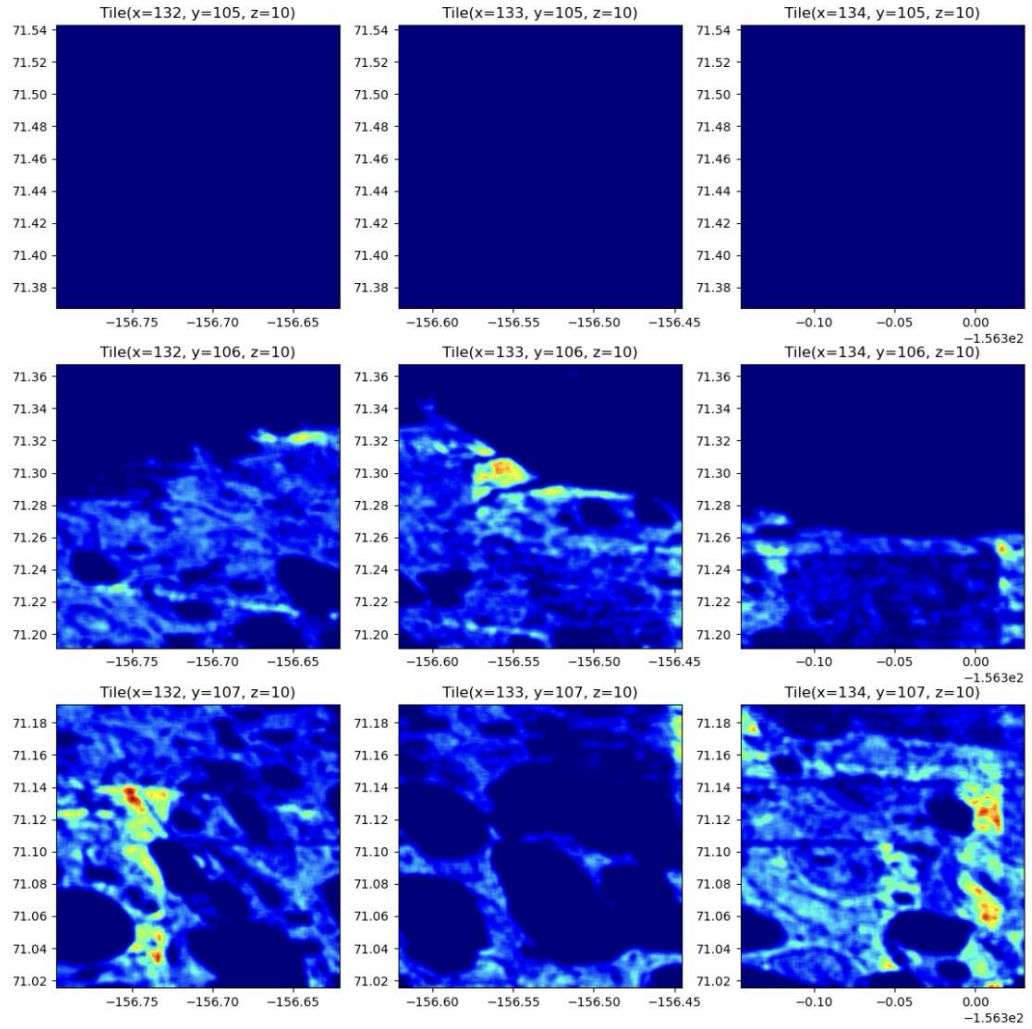
After clip



Radius = 20, buffer = 100

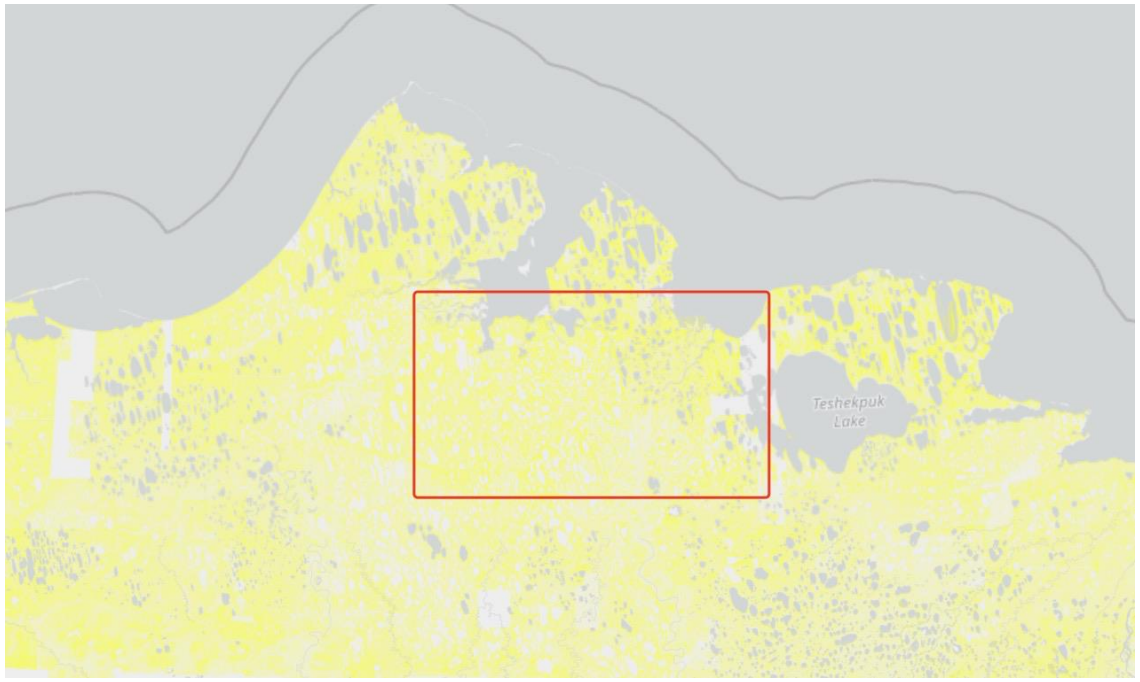
Statistics of the 9 Tiles






```
Overall Min: 0.0
Overall Max: 0.5602904048200853
Overall Mean: 0.038851426040109 17
Overall Standard Deviation: 0.062685601178517 74
```

Switch to another ROI to test



Tile count: 64

Sigma = 32, radius = 20, buffer = 20

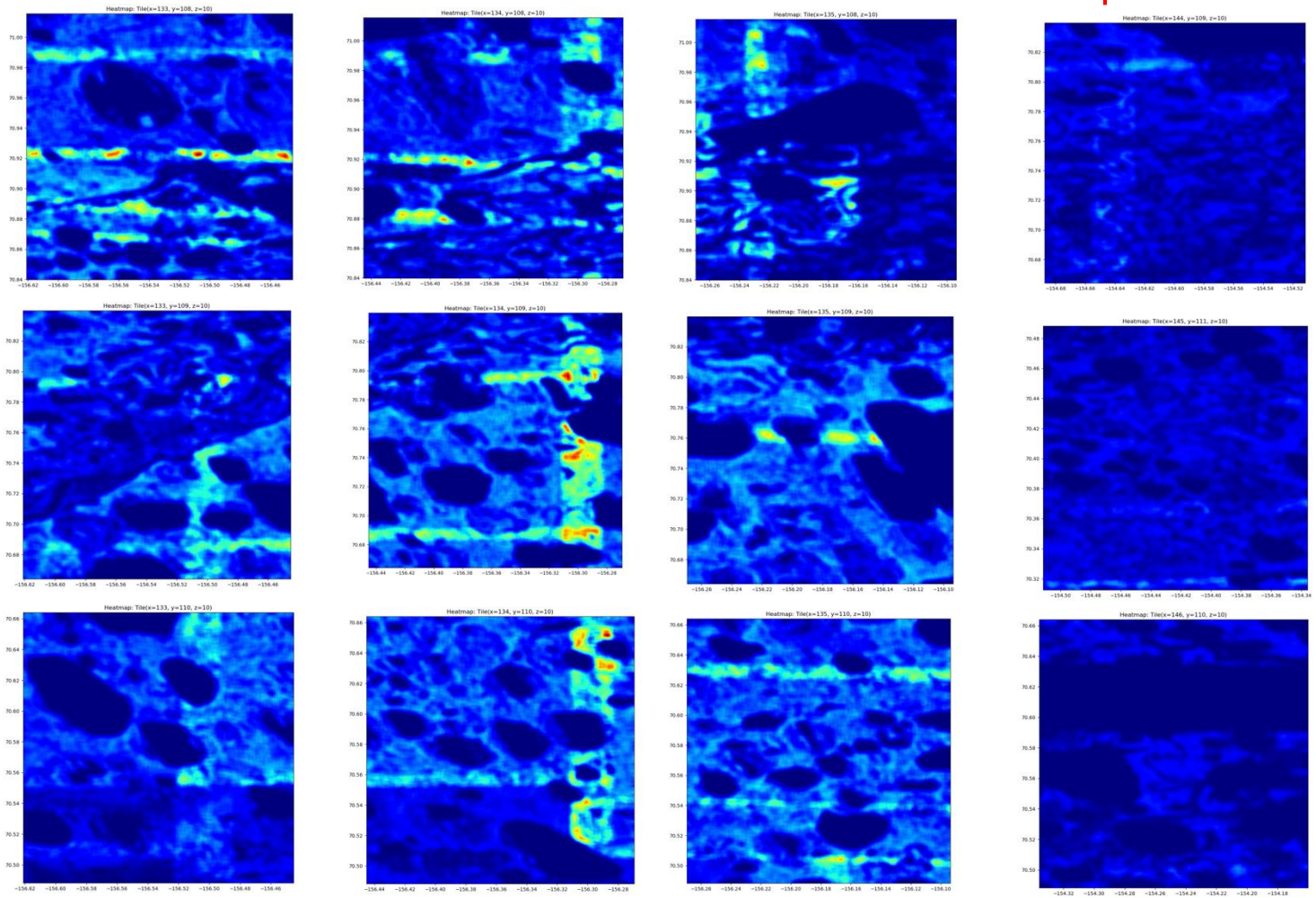
```
[Tile(x=133, y=108, z=10), Tile(x=134, y=108, z=10), Tile(x=135, y=108, z=10), Tile(x=136, y=108, z=10), Tile(x
=137, y=108, z=10), Tile(x=138, y=108, z=10), Tile(x=139, y=108, z=10), Tile(x=140, y=108, z=10), Tile(x=141, y
=108, z=10), Tile(x=142, y=108, z=10), Tile(x=143, y=108, z=10), Tile(x=144, y=108, z=10), Tile(x=145, y=108, z
=10), Tile(x=146, y=108, z=10), Tile(x=147, y=108, z=10), Tile(x=148, y=108, z=10), Tile(x=133, y=109, z=10), T
ile(x=134, y=109, z=10), Tile(x=135, y=109, z=10), Tile(x=136, y=109, z=10), Tile(x=137, y=109, z=10), Tile(x=1
38, y=109, z=10), Tile(x=139, y=109, z=10), Tile(x=140, y=109, z=10), Tile(x=141, y=109, z=10), Tile(x=142, y=1
09, z=10), Tile(x=143, y=109, z=10), Tile(x=144, y=109, z=10), Tile(x=145, y=109, z=10), Tile(x=146, y=109, z=1
0), Tile(x=147, y=109, z=10), Tile(x=148, y=109, z=10), Tile(x=133, y=110, z=10), Tile(x=134, y=110, z=10), Til
e(x=135, y=110, z=10), Tile(x=136, y=110, z=10), Tile(x=137, y=110, z=10), Tile(x=138, y=110, z=10), Tile(x=139
, y=110, z=10), Tile(x=140, y=110, z=10), Tile(x=141, y=110, z=10), Tile(x=142, y=110, z=10), Tile(x=143, y=110
, z=10), Tile(x=144, y=110, z=10), Tile(x=145, y=110, z=10), Tile(x=146, y=110, z=10), Tile(x=147, y=110, z=10)
, Tile(x=148, y=110, z=10), Tile(x=133, y=111, z=10), Tile(x=134, y=111, z=10), Tile(x=135, y=111, z=10), Tile(
x=136, y=111, z=10), Tile(x=137, y=111, z=10), Tile(x=138, y=111, z=10), Tile(x=139, y=111, z=10), Tile(x=140,
y=111, z=10), Tile(x=141, y=111, z=10), Tile(x=142, y=111, z=10), Tile(x=143, y=111, z=10), Tile(x=144, y=111,
z=10), Tile(x=145, y=111, z=10), Tile(x=146, y=111, z=10), Tile(x=147, y=111, z=10), Tile(x=148, y=111, z=10)]
The number of tiles within the geographic bbox is: 64
```

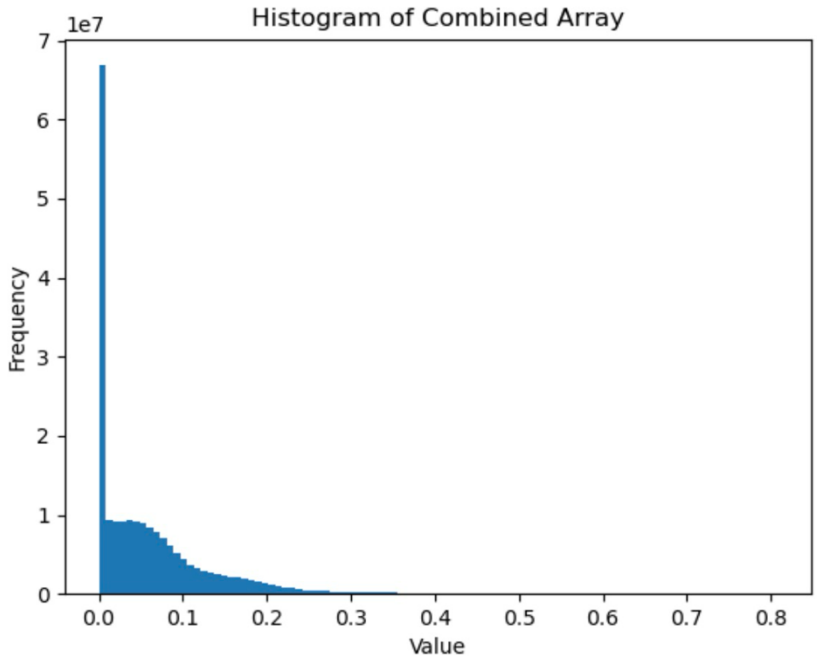
Switch to another ROI to test

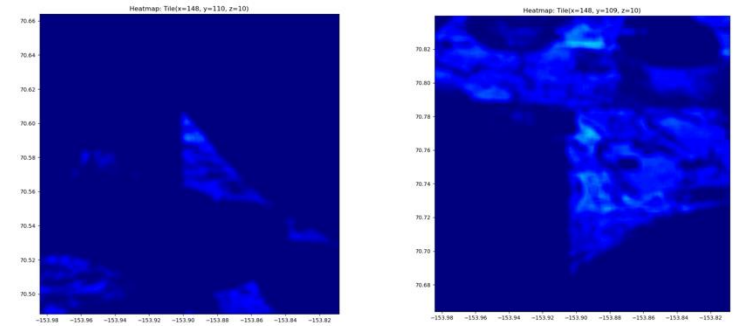
Sample of tiles

Around 37/64 tiles

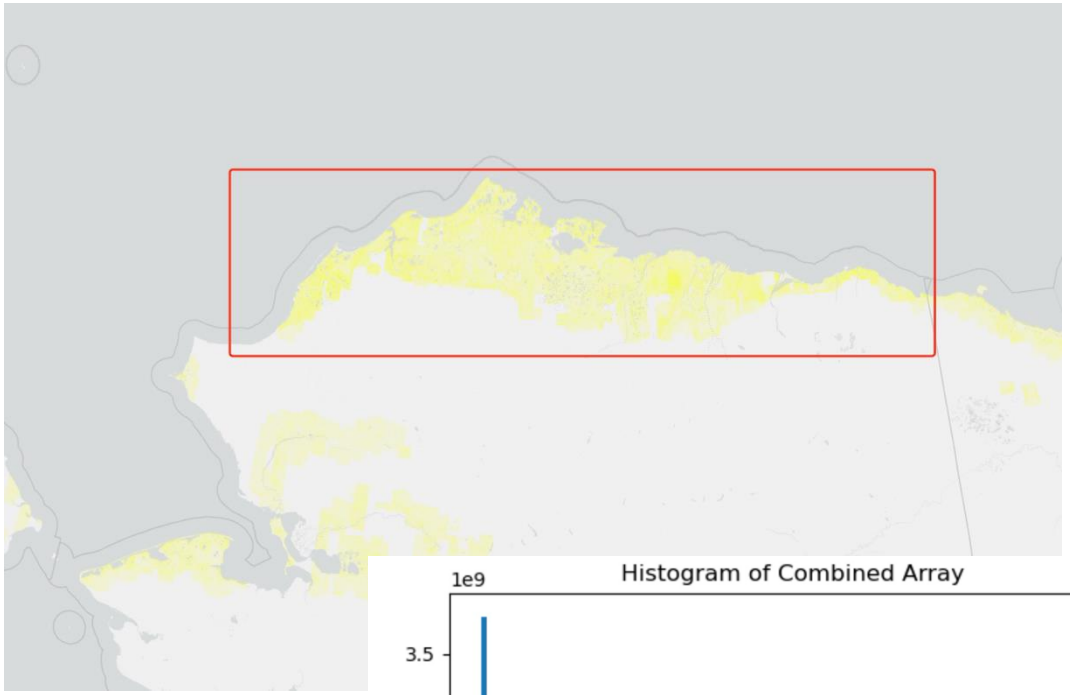
```
Overall Min: 0.0
Overall Max: 0.8075497098561949
Overall Mean: 0.0583910743655836
Overall Standard Deviation: 0.07212593808997543
```

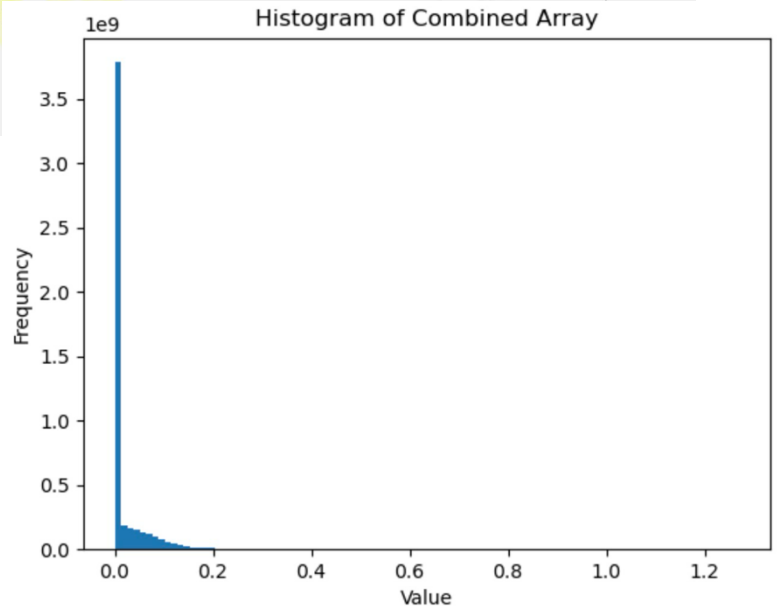


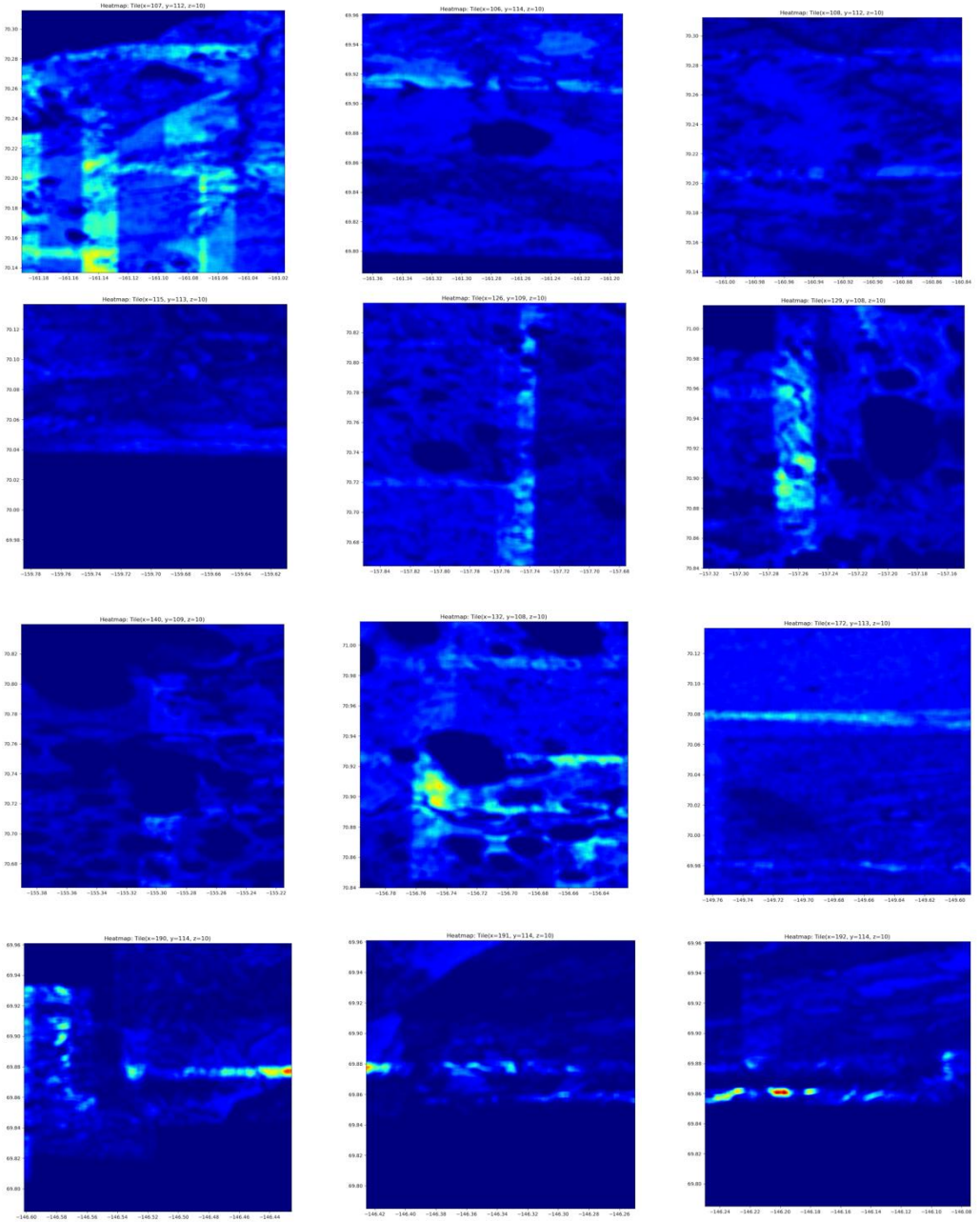




Scalability







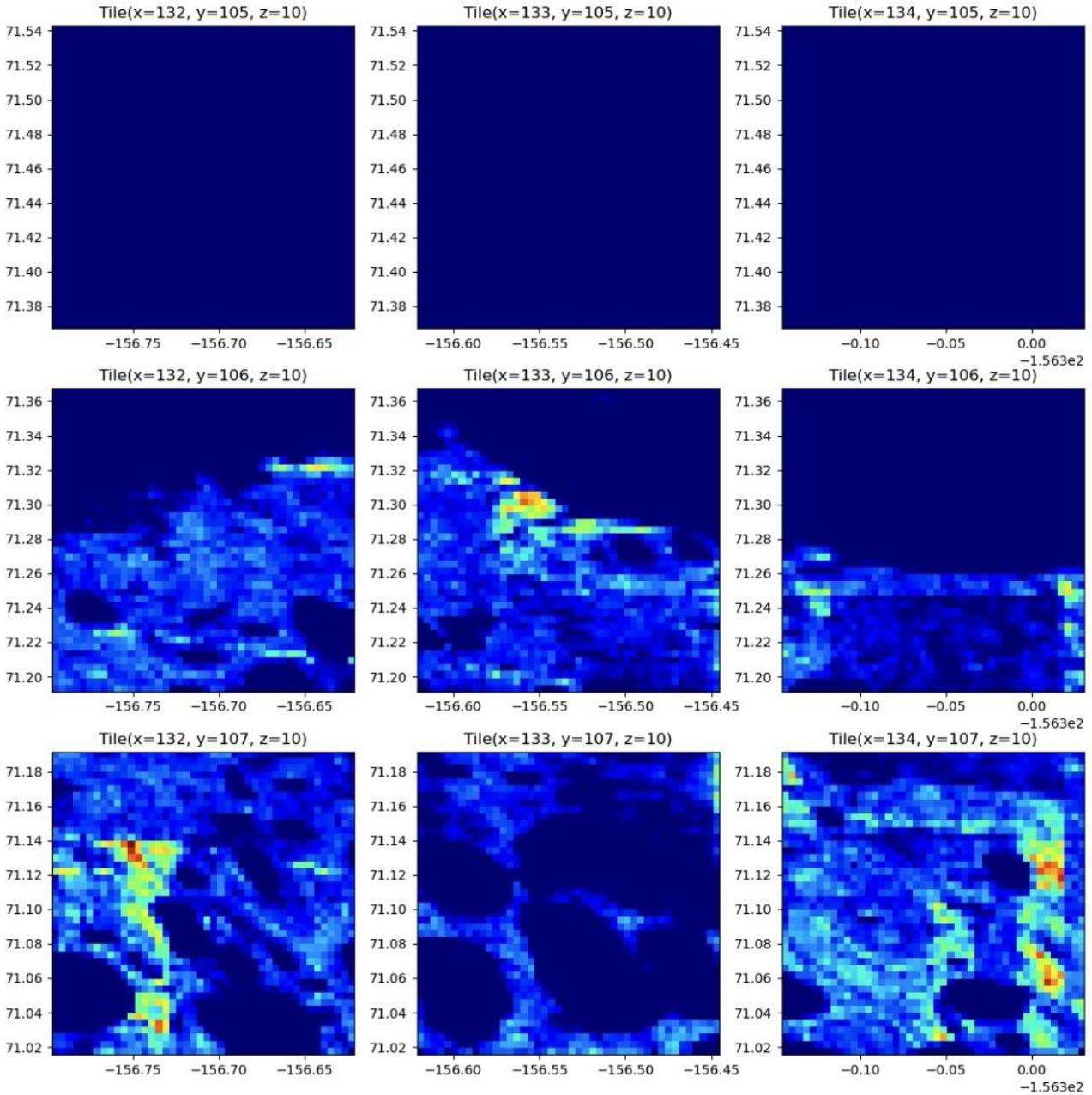
Tile count: 1595

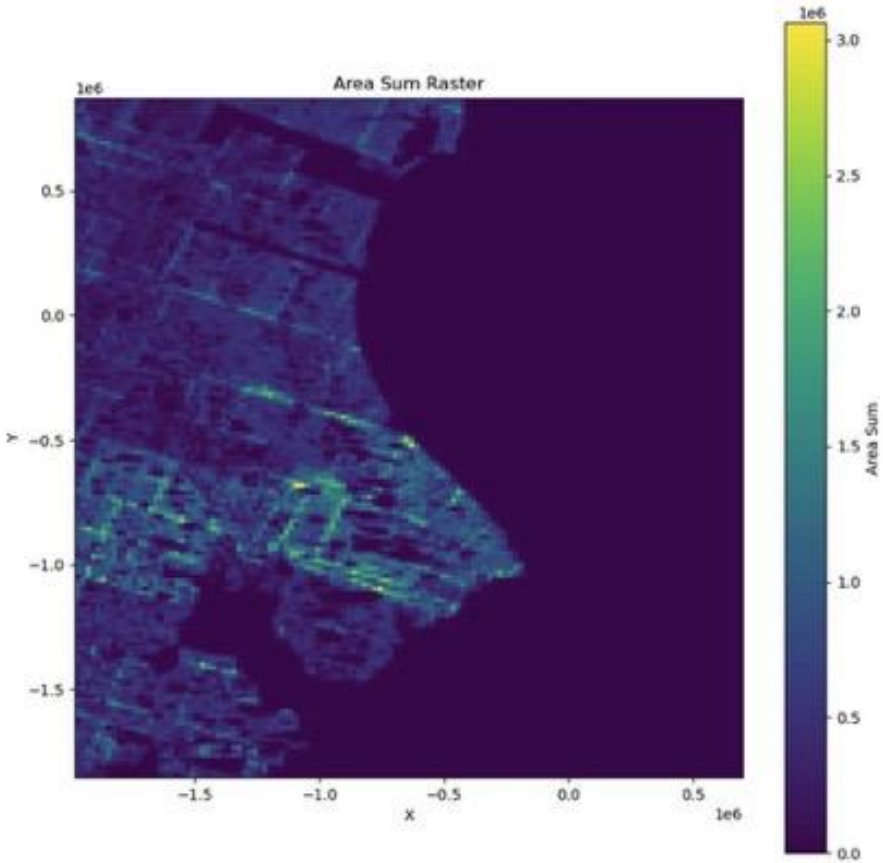
Sigma = 64

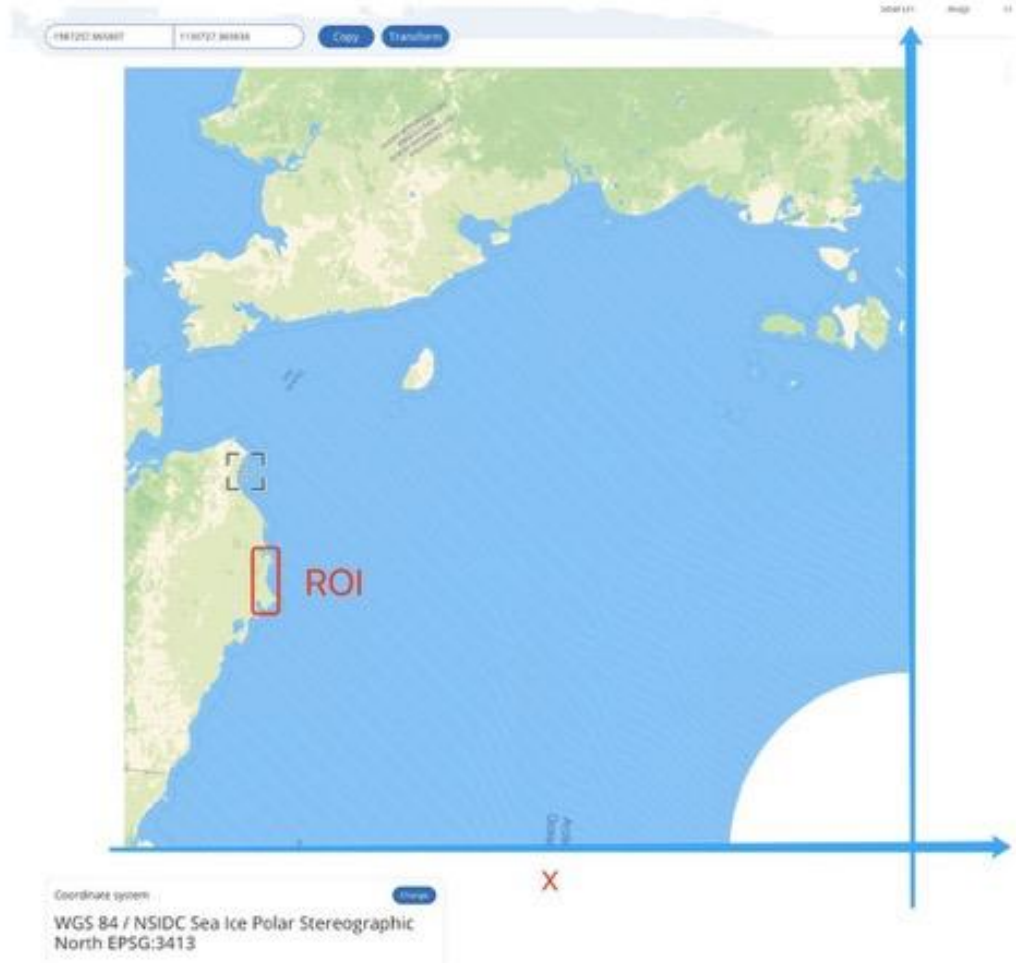
Meeting 9/17/2024

1. Map the count map with 256 * 256 pixels in a tile
2. Map the coverage ratio map
3. Pixel: 1km * 1km

Count map

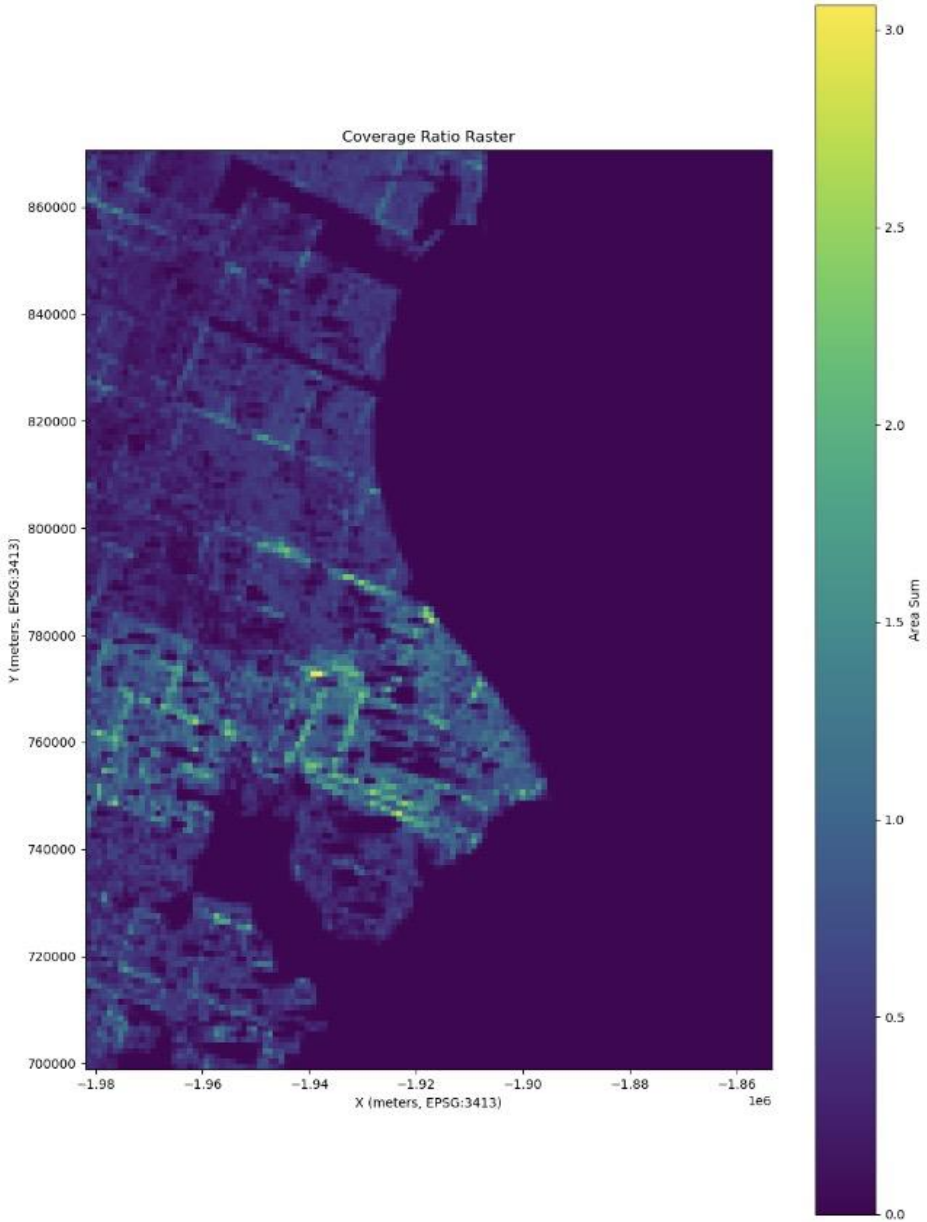

Tile(x=132, y=105, z=10)
Tile(x=133, y=105, z=10)
Tile(x=134, y=105, z=10)
Tile(x=132, y=106, z=10)
Tile(x=133, y=106, z=10)
Tile(x=134, y=106, z=10)
Tile(x=132, y=107, z=10)
Tile(x=133, y=107, z=10)
Tile(x=134, y=107, z=10)
−1.563e2

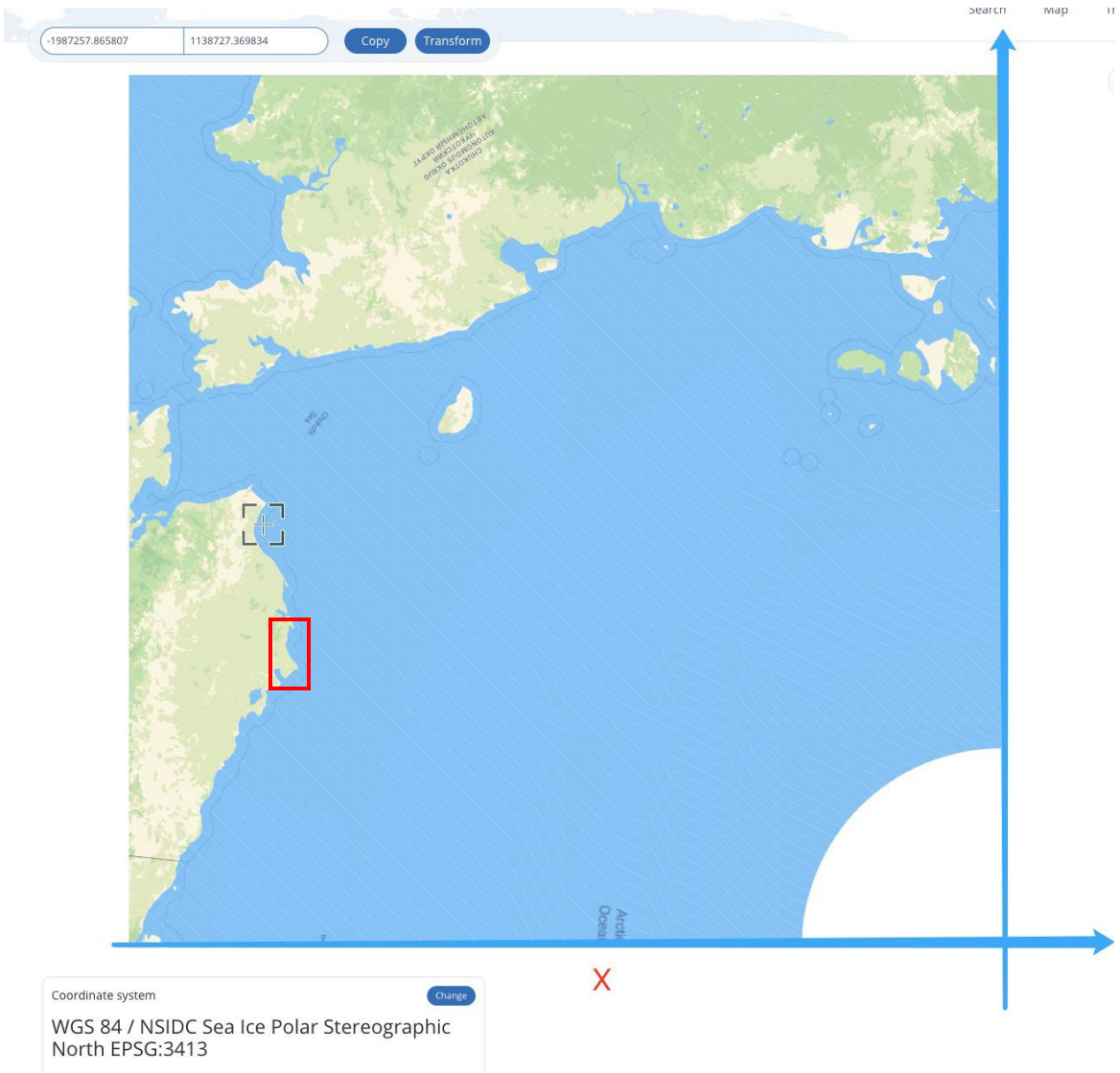

Area Sum Raster
1e6
0.5
0.0
−0.5
−1.0
−1.5
Y
−1.5
−1.0
−0.5
0.0
0.5
1e6
X
1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Area Sum

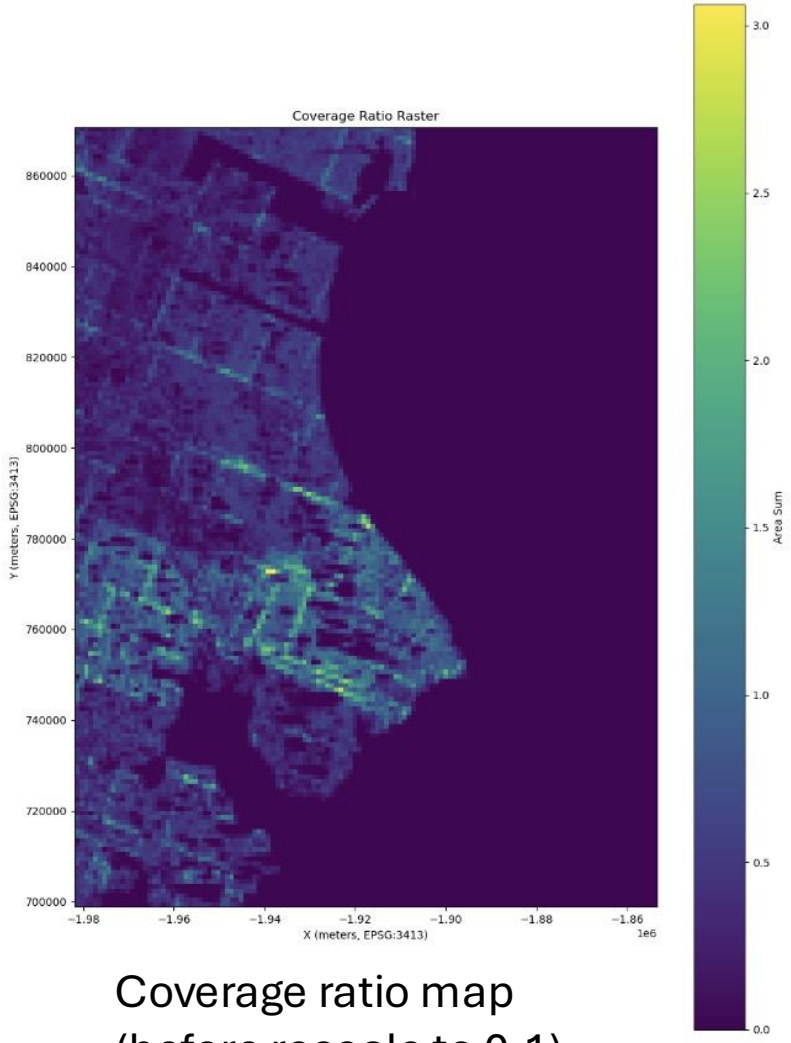

Copy
Transform
ROI
X
Coordinate system
WGS 84 / NSIDC Sea Ice Polar Stereographic North EPSG:3413

Meeting 9/24/2024

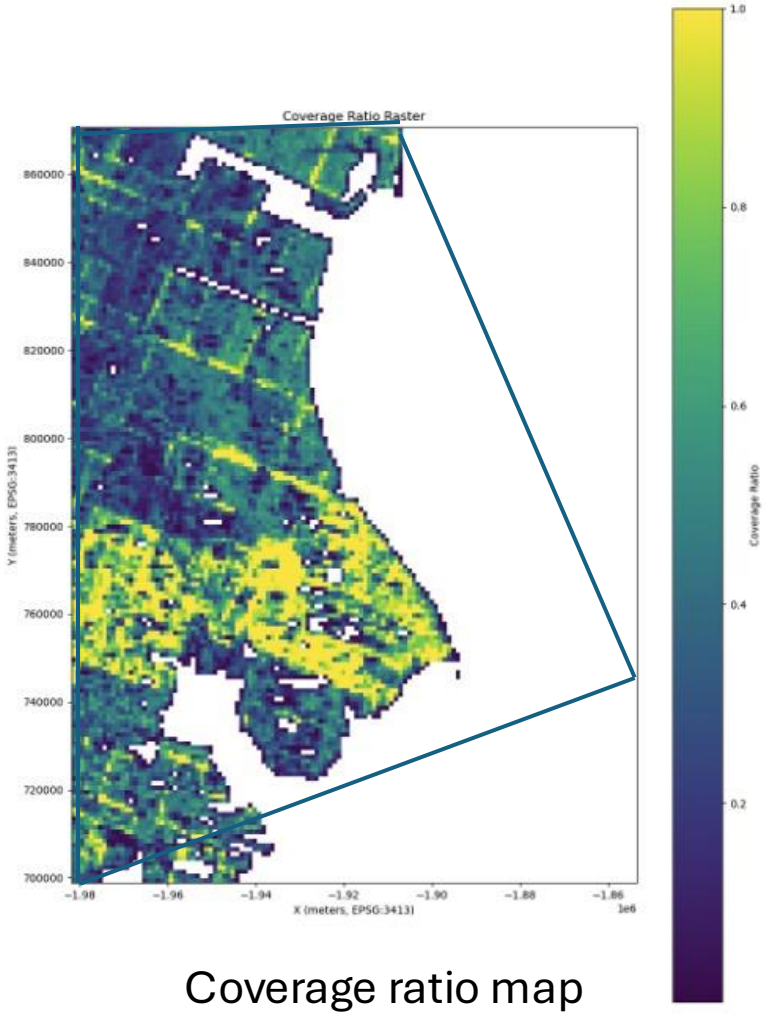
1. Resize the plot
2. Change area sum --> coverage percentage
3. 0 values --> Transparent
4. Check the rotation/flip
5. Check the data (QGIS/2km*2km)
6. Check the data via the statistics histogram
7. Apply to larger scale


Coverage Ratio Raster
Y (meters, EPSG:3413)
860000
840000
820000
800000
780000
760000
740000
720000
700000
−1.98
−1.96
−1.94
−1.92
−1.90
−1.88
−1.86
1e6
X (meters, EPSG:3413)
Area Sum
3.0
2.5
2.0
1.5
1.0
0.5
0.0

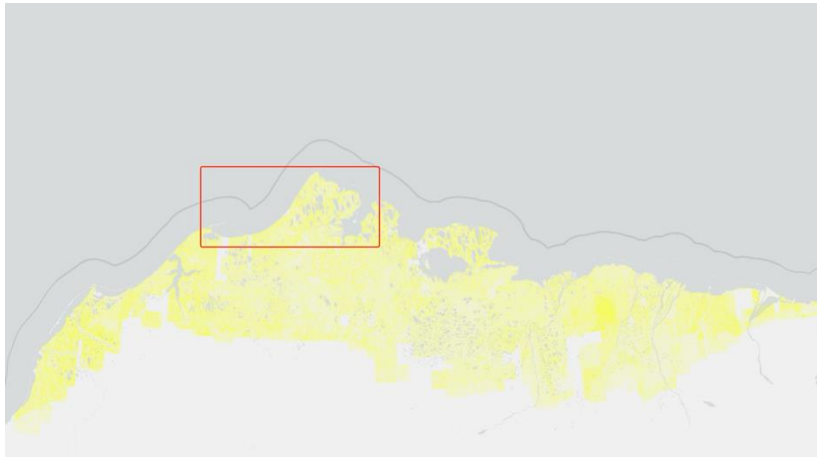

-1987257.865807
1138727.369834
Copy
Transform
X
Coordinate system
Change
WGS 84 / NSIDC Sea Ice Polar Stereographic North EPSG:3413



Coverage ratio map
(before rescale to 0-1)



Coverage ratio map



Polygon:

X: -1981837.4924319359 ~ -1853396.1111618795

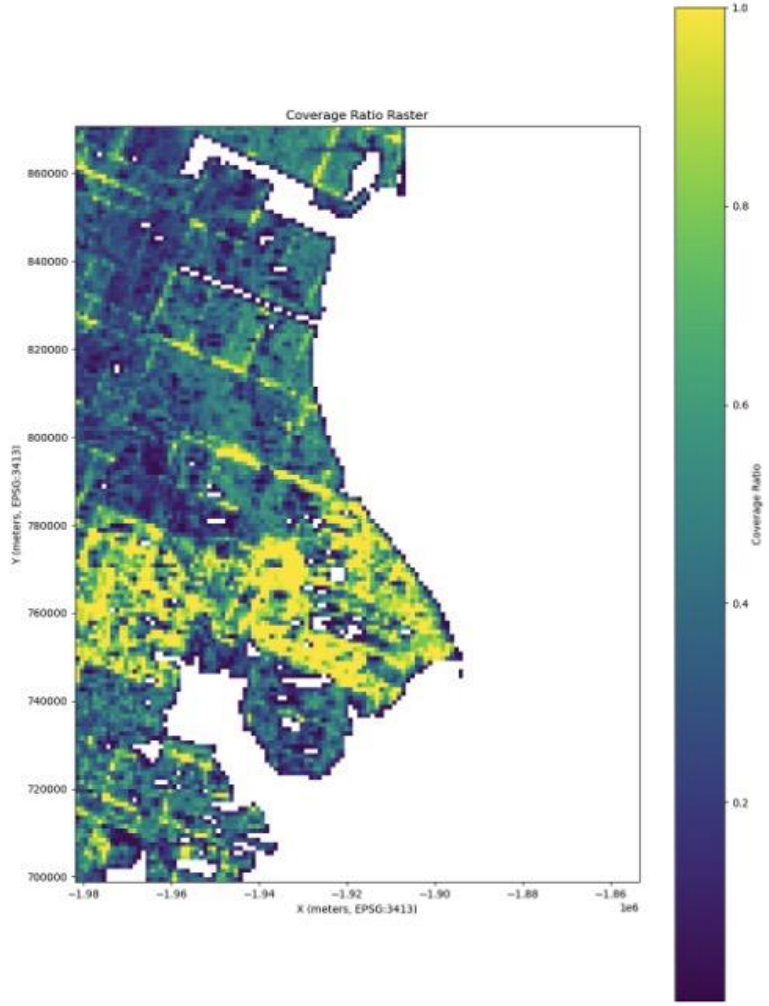
Y: 698801.3846376436 ~ 870714.395168513

Extent:

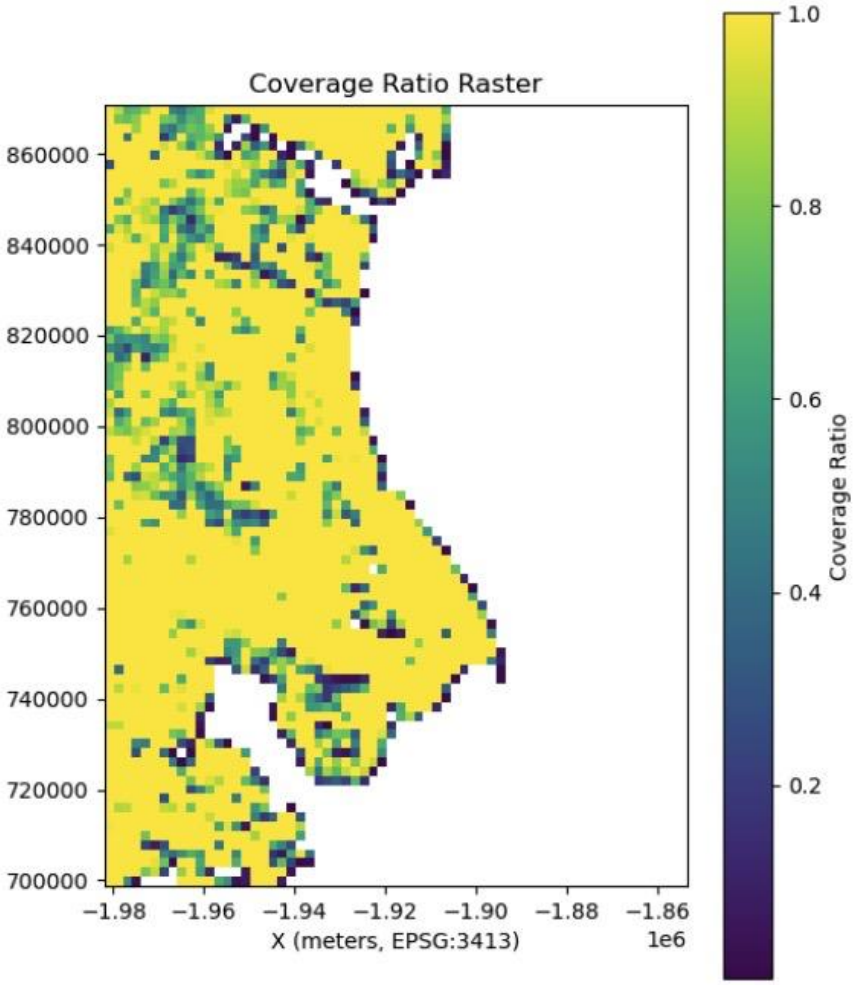
X: 128,441 Y: 171,913

```
POLYGON ((-159.3697 70.69748, -159.3697 71.37637, -155.0869 71.37637, -155.0869 70.69748, -159.3697 70.69748))
The coordinates of reprojected ROI is: POLYGON ((-1922180.5912919727 870714.395168513, -1853396.1111618795 839556.2213294988, -191091
8.2134438816 698801.3846376436, -1981837.4924319359 724735.7705290421, -1922180.5912919727 870714.395168513))
The polygon is: POLYGON ((-1922180.5912919727 870714.395168513, -1853396.1111618795 839556.2213294988, -1910918.2134438816 698801.384
6376436, -1981837.4924319359 724735.7705290421, -1922180.5912919727 870714.395168513)), the bounds of the polygon is: (-1981837.49243
19359, 698801.3846376436, -1853396.1111618795, 870714.395168513)
The polygon bounds of this ROI is: (-1981837.4924319359, 698801.3846376436, -1853396.1111618795, 870714.395168513)
The width size is: 128441.38127005636
The height size is: 171913.01053086936
The height and width of this raster is: (171, 128)
```

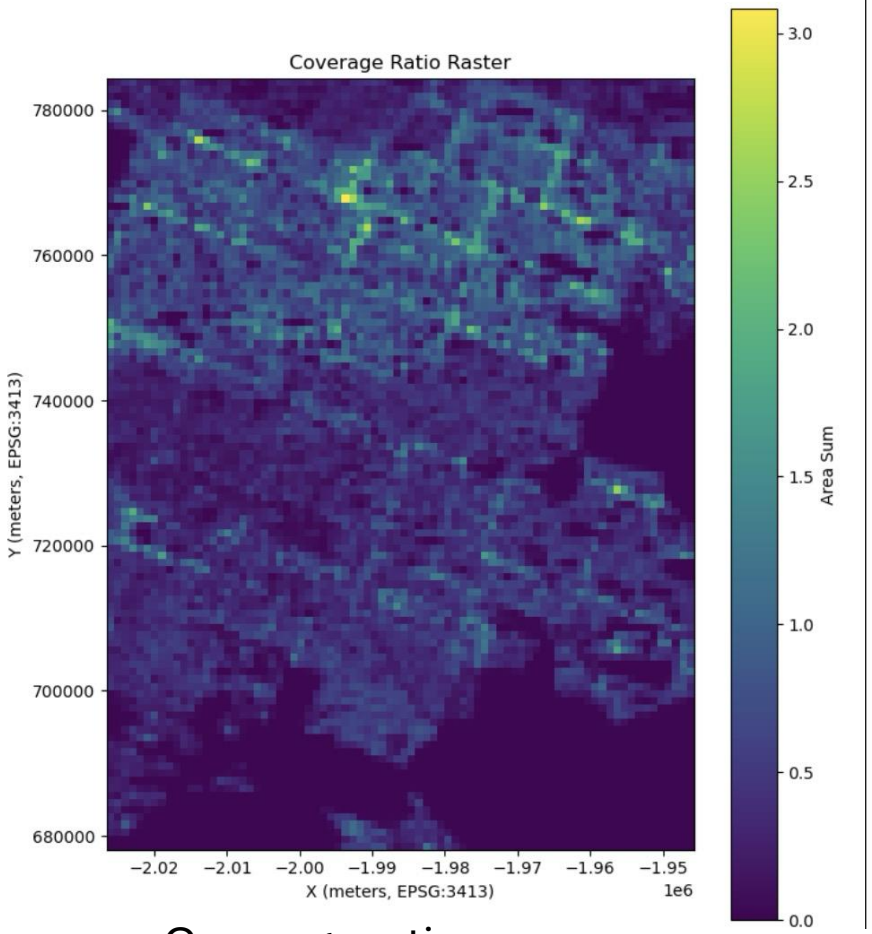
Check the data in 2km resolution



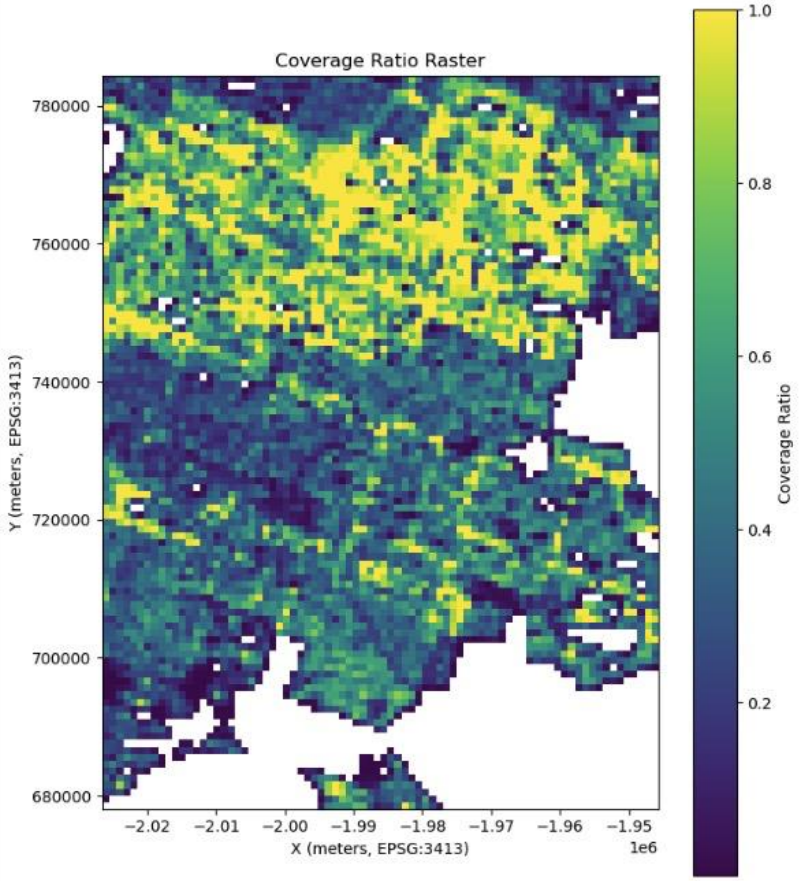
1km resolution



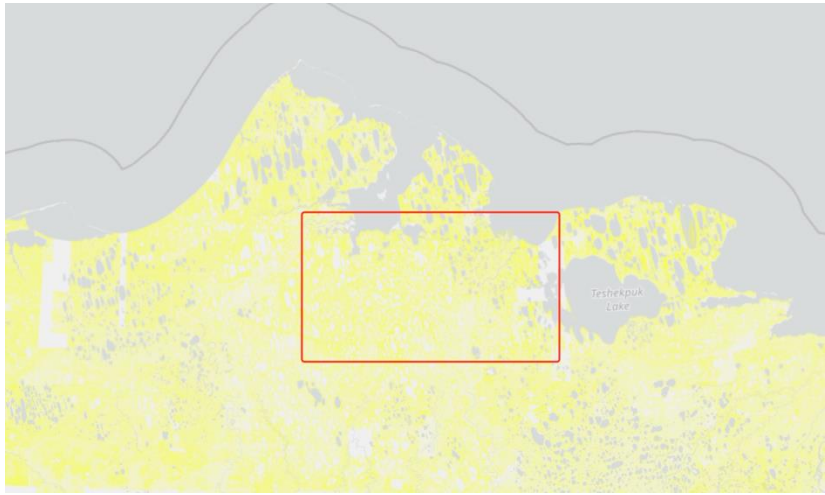
2km resolution



Coverage ratio map
(before rescale to 0-1)



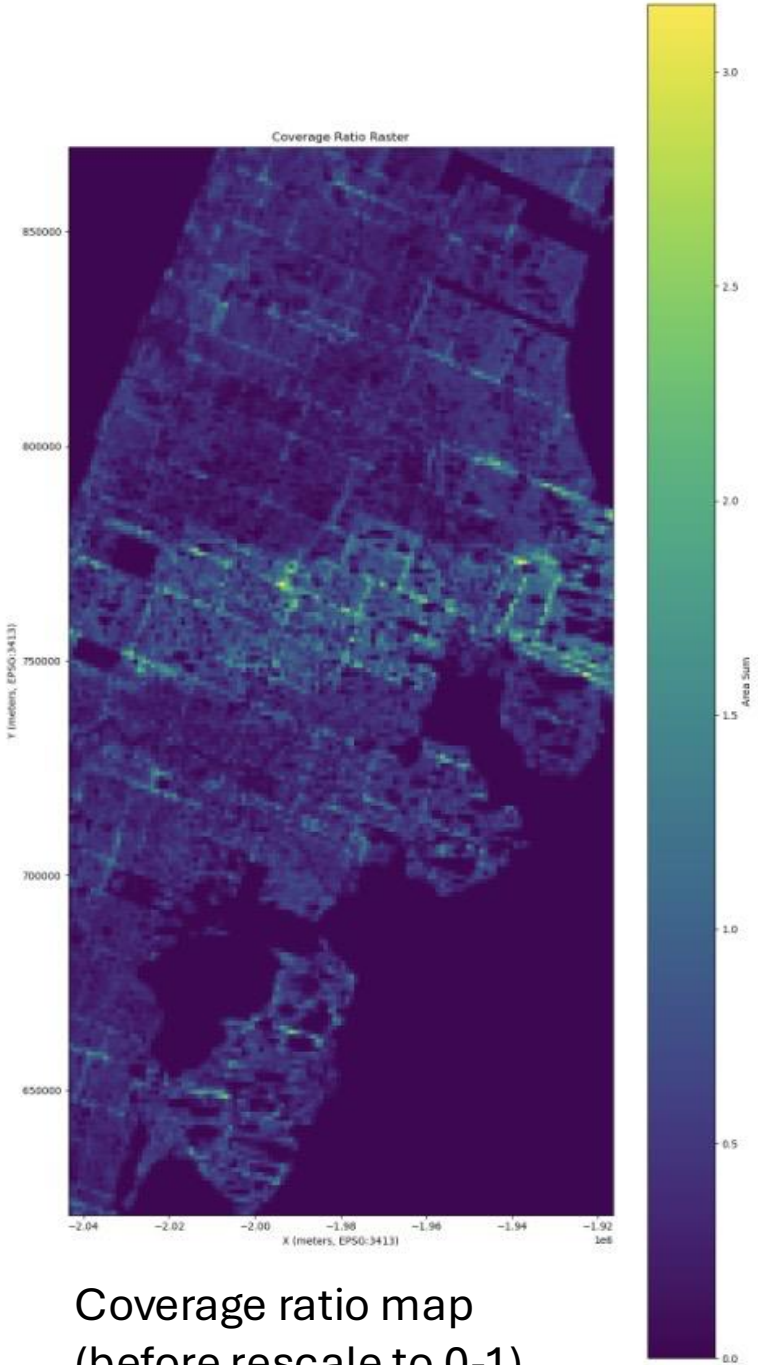
Coverage ratio map



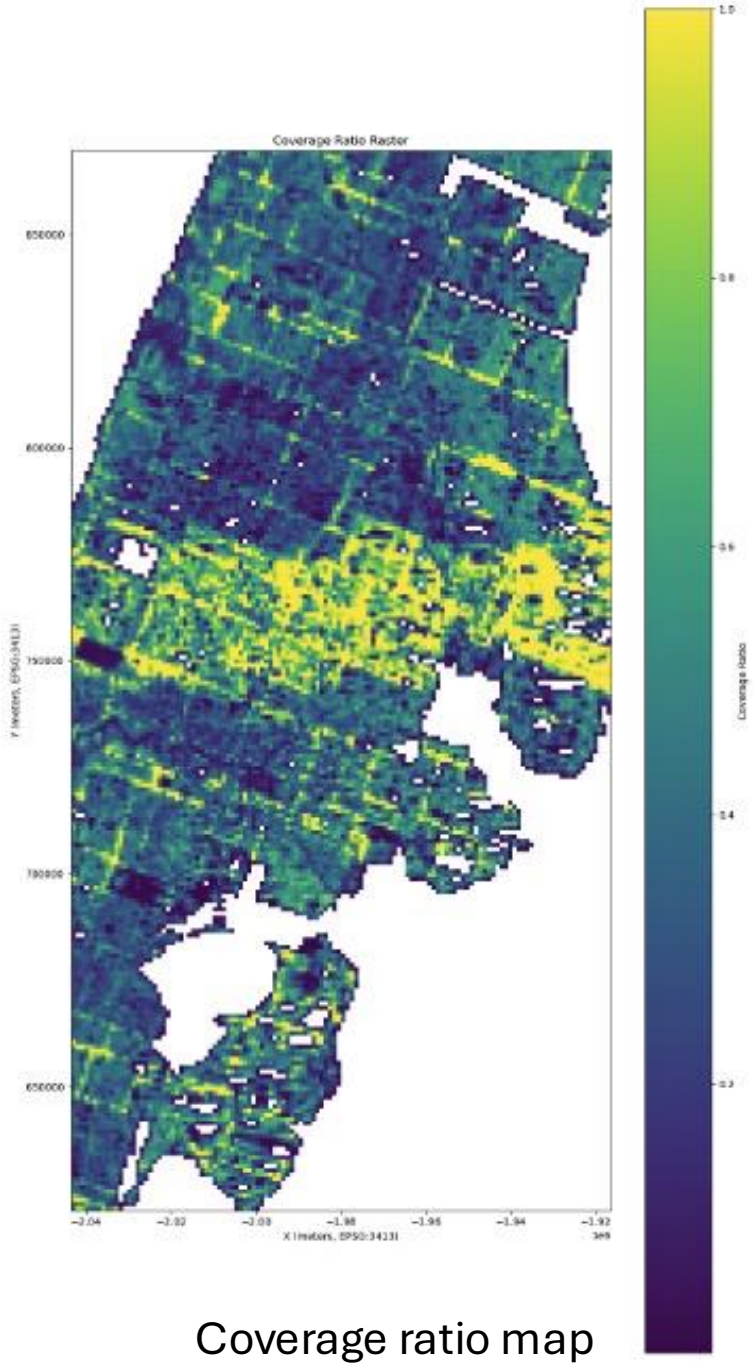
Polygon:
X: -2026533.5803334767 ~ -1945685.3317847361
Y: 678120.4396953756 ~ 784280.1615397114

Extent:
X: 80,848 Y:106,160

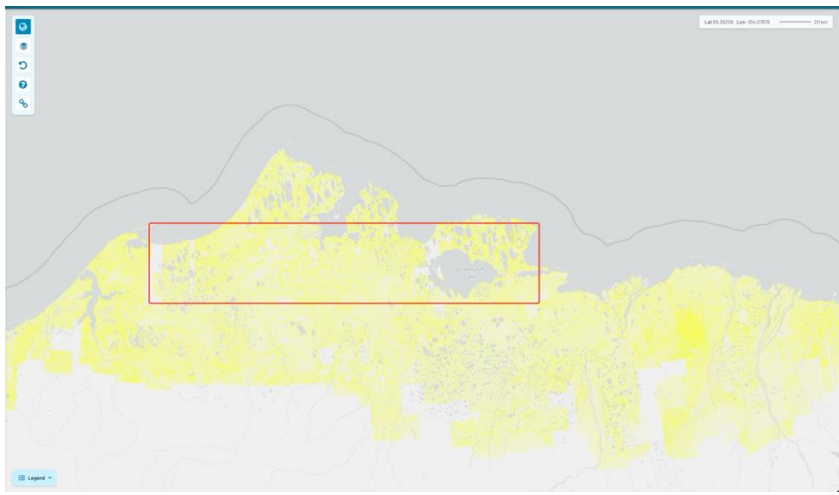
```
(alaska_shpMapping2) (base) xchen@cici2labasuedu:~/code/shpMapping$ python CoveragePercentage.py
Connection established successfully!
POLYGON ((-156.47441 70.40887, -156.47441 70.87147, -153.92505 70.87147, -153.92505 70.40887, -156.47441 70.40887))
The coordinates of reprojected ROI is: POLYGON ((-1993621.9044003987 784280.1615397114, -1945685.3317847361 765422.1710492934, -19778
05.647560812 678120.4396953756, -2026533.5803334767 694827.5449855227, -1993621.9044003987 784280.1615397114))
The polygon is: POLYGON ((-1993621.9044003987 784280.1615397114, -1945685.3317847361 765422.1710492934, -1977805.647560812 678120.439
6953756, -2026533.5803334767 694827.5449855227, -1993621.9044003987 784280.1615397114)), the bounds of the polygon is: (-2026533.5803
334767, 678120.4396953756, -1945685.3317847361, 784280.1615397114)
The polygon bounds of this ROI is: (-2026533.5803334767, 678120.4396953756, -1945685.3317847361, 784280.1615397114)
The width size is: 80848.24854874052
The height size is: 106159.72184433579
The height and width of this raster is: (106, 80)
```

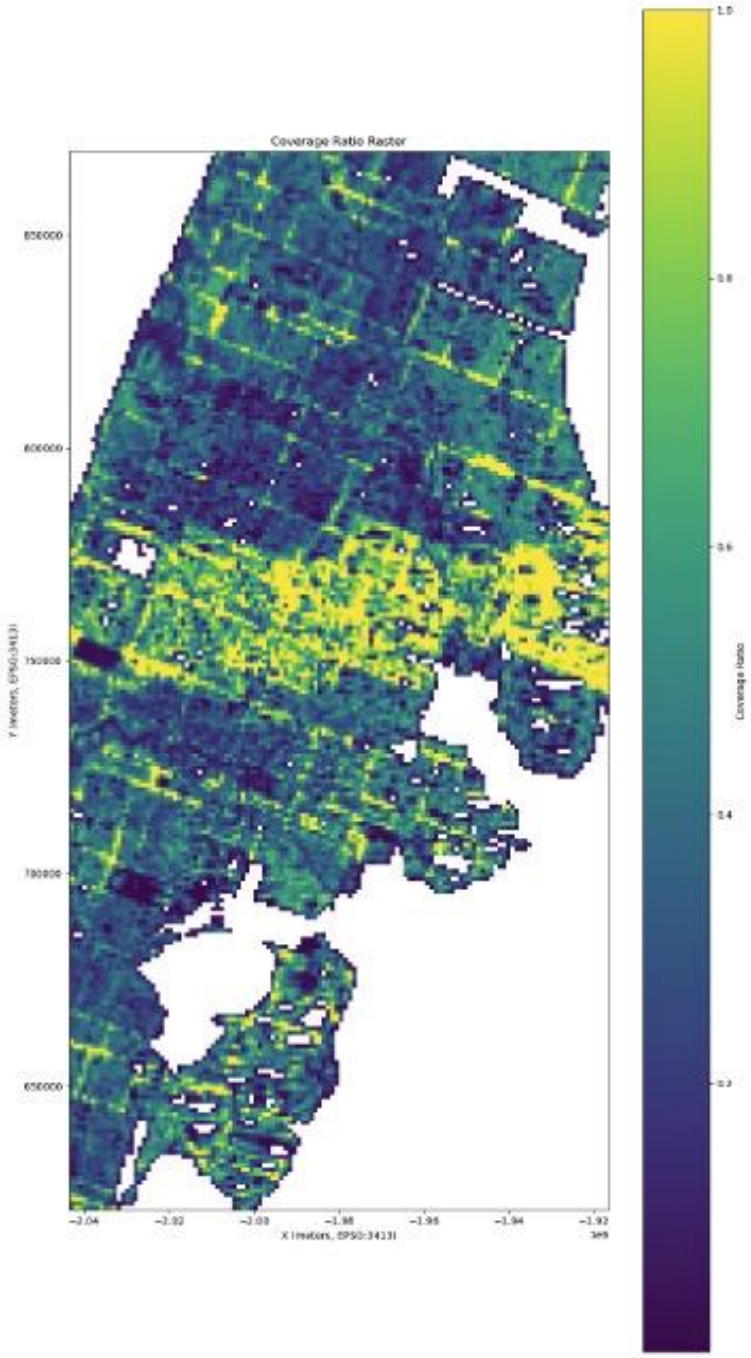


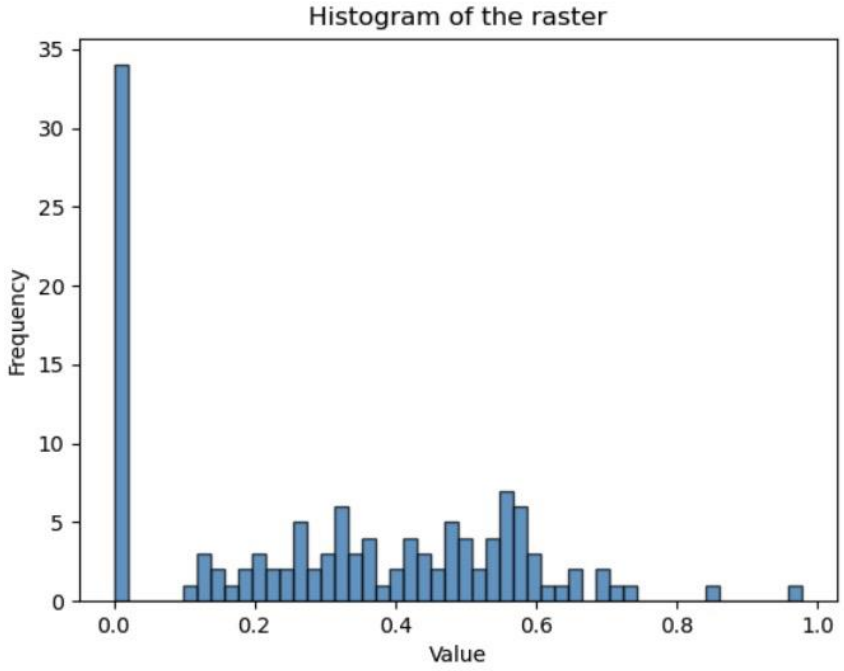
Coverage ratio map
(before rescale to 0-1)

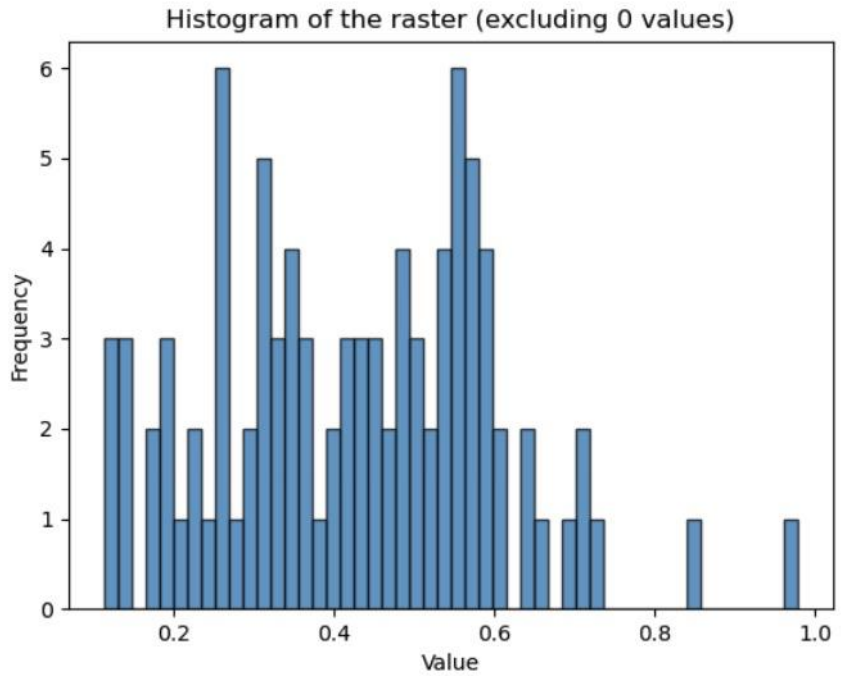


Coverage ratio map

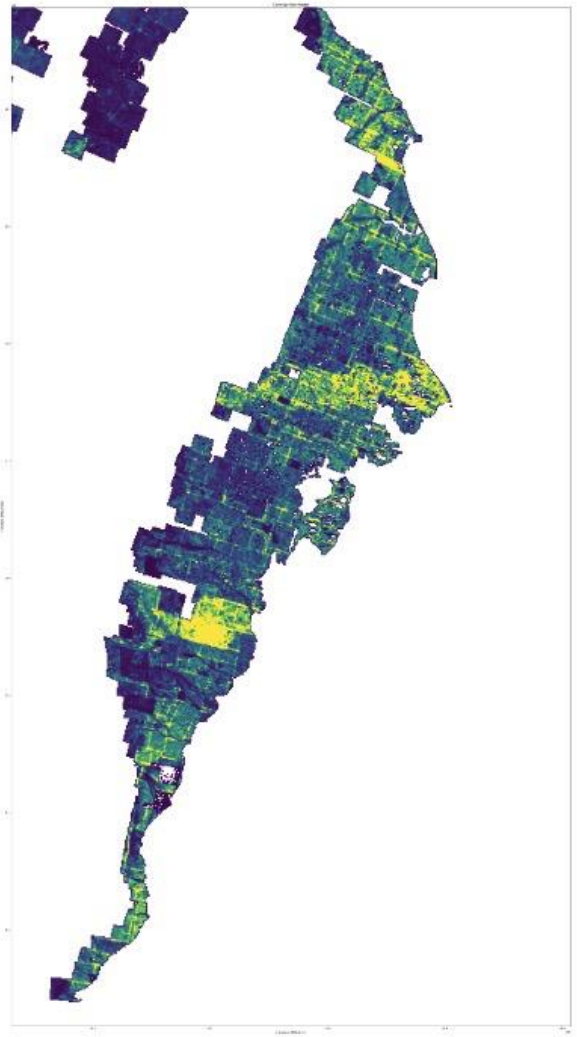


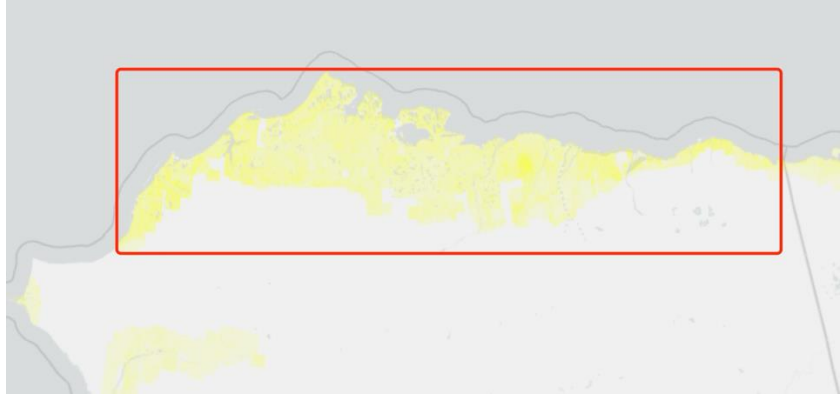


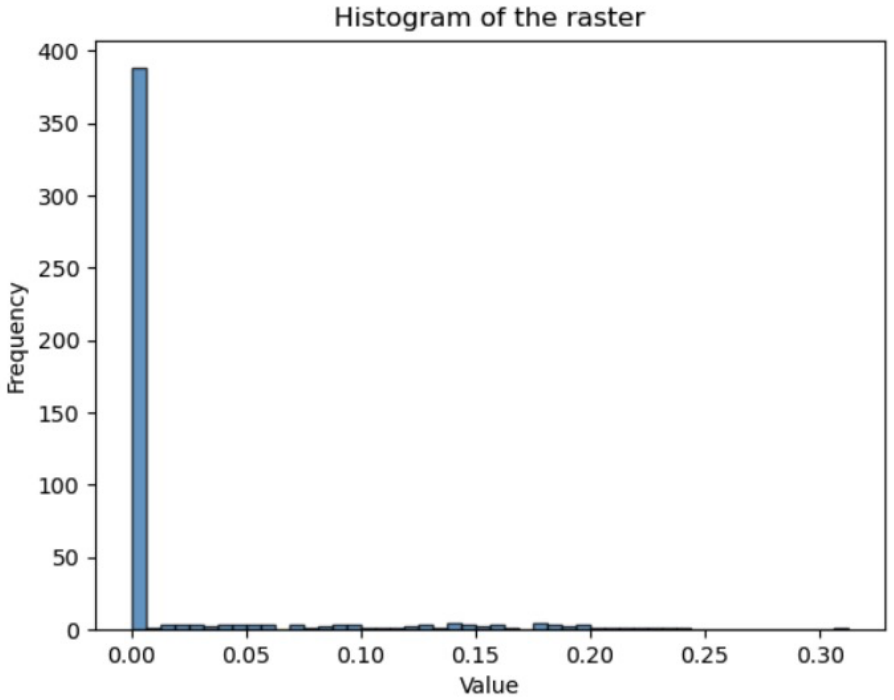


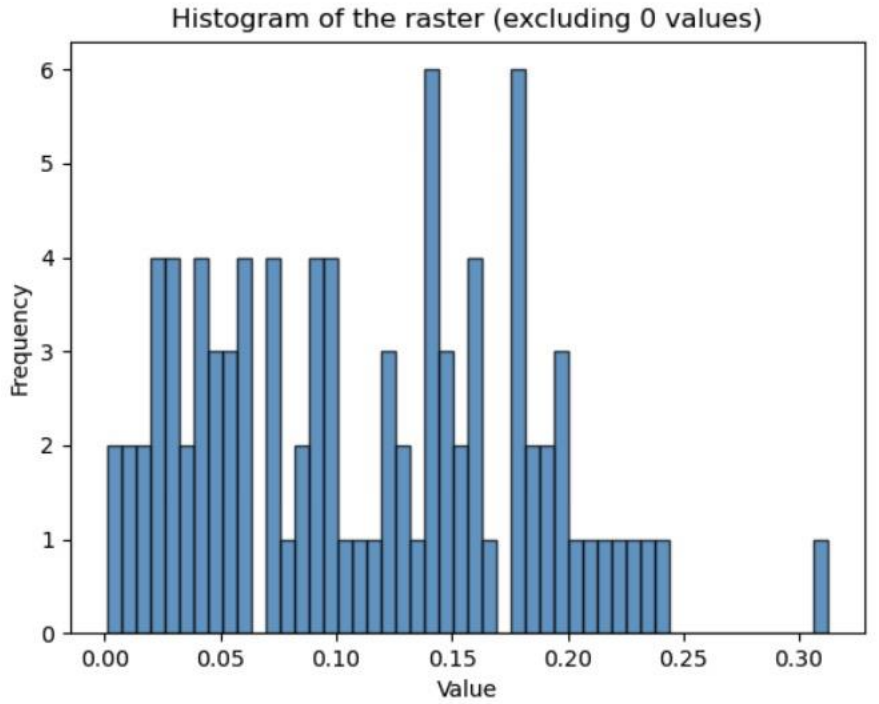



```
Overall Min: 0.0
Overall Max: 0.9791127
Overall Mean: 0.31104845
Overall Standard Deviation: 0.24028301
```










```
Overall Min: 0.0
Overall Max: 0.31268838
Overall Mean: 0.02122589
Overall Standard Deviation: 0.052890483
```

Meeting 10/01/2024

- Check the coverage ratio values
  - Store original area sum at 1*1km resolution → Other resolutions/calculating ratio
    - Try 2*2km → 10*10km
    - Then aggregate and store every raster based on this original area sum
  - Check if the area sum is correct
    - Delete the buffer of the cell
    - Check the bounds of the cell
    - Try on ArcGIS

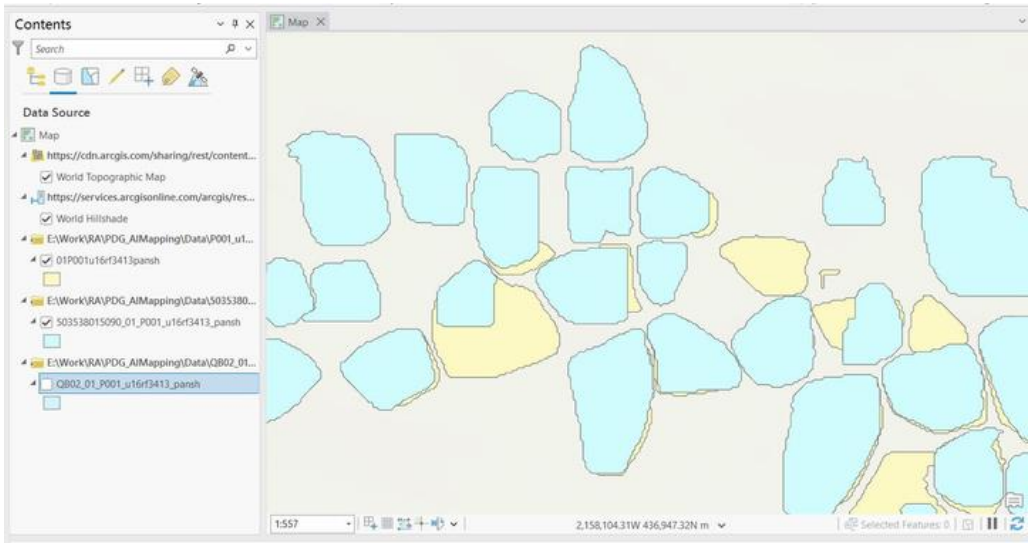
Check the bounds & IWP counts

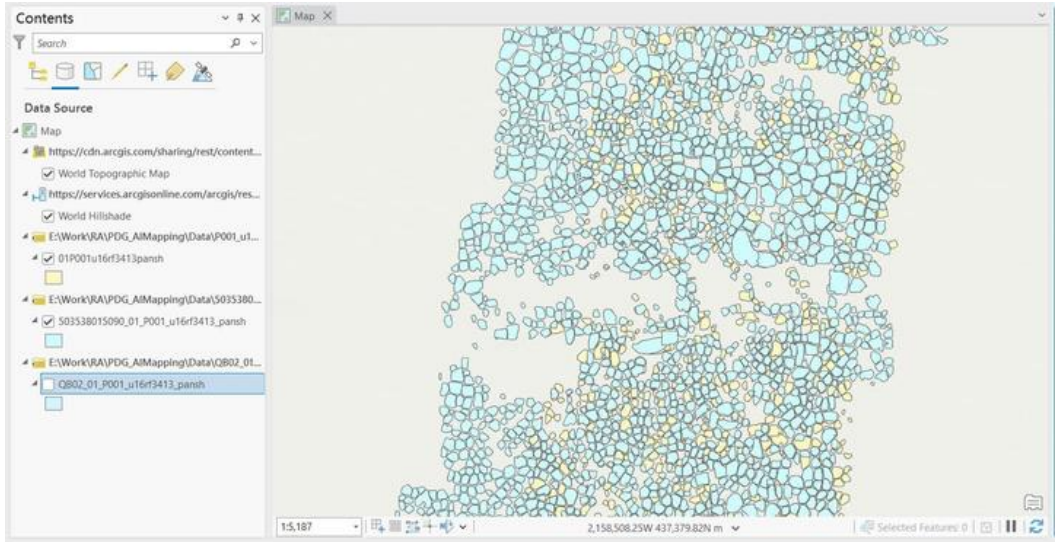
```
Cell (66, 13) has the bounds: (-2013533.5803334767, 718280.1615397114, -2012533.5803334767, 717280.1615397114)
----------------------------------------
Cell (66, 14) has the bounds: (-2012533.5803334767, 718280.1615397114, -2011533.5803334767, 717280.1615397114)
----------------------------------------
Cell (66, 15) has the bounds: (-2011533.5803334767, 718280.1615397114, -2010533.5803334767, 717280.1615397114)
----------------------------------------
Cell (66, 16) has the bounds: (-2010533.5803334767, 718280.1615397114, -2009533.5803334767, 717280.1615397114)
----------------------------------------
Cell (66, 17) has the bounds: (-2009533.5803334767, 718280.1615397114, -2008533.5803334767, 717280.1615397114)
----------------------------------------
Cell (66, 18) has the bounds: (-2008533.5803334767, 718280.1615397114, -2007533.5803334767, 717280.1615397114)
----------------------------------------
Cell (66, 19) has the bounds: (-2007533.5803334767, 718280.1615397114, -2006533.5803334767, 717280.1615397114)
```

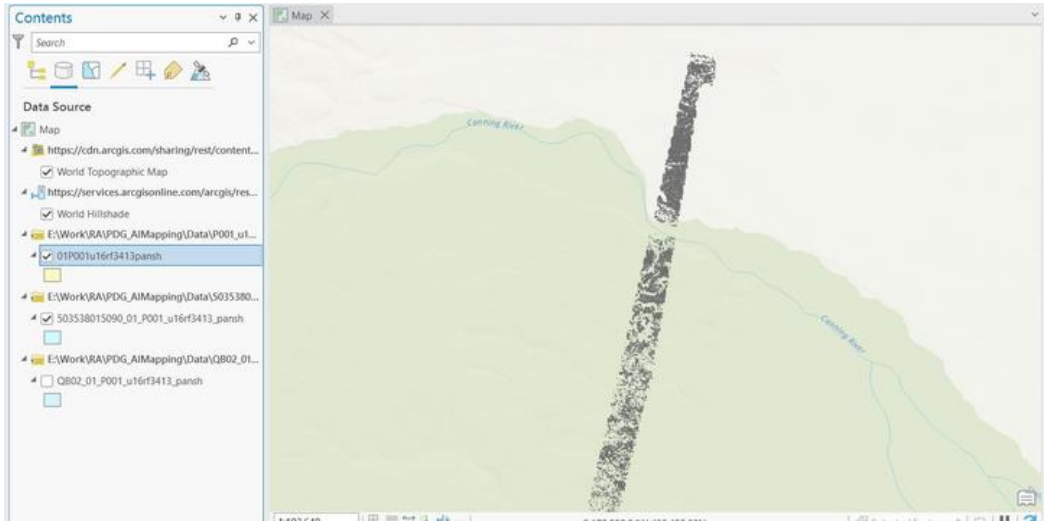
```
----------------------------------------
Cell (0, 89) has the bounds: (-1954436.506794451, 869675.0042933973, -1953436.506794451, 868675.0042933973)
The number of IWP within the bounding box is: 353
----------------------------------------
Cell (0, 90) has the bounds: (-1953436.506794451, 869675.0042933973, -1952436.506794451, 868675.0042933973)
The number of IWP within the bounding box is: 1101
----------------------------------------
Cell (0, 91) has the bounds: (-1952436.506794451, 869675.0042933973, -1951436.506794451, 868675.0042933973)
The number of IWP within the bounding box is: 1255
----------------------------------------
Cell (0, 92) has the bounds: (-1951436.506794451, 869675.0042933973, -1950436.506794451, 868675.0042933973)
The number of IWP within the bounding box is: 1772
----------------------------------------
Cell (0, 93) has the bounds: (-1950436.506794451, 869675.0042933973, -1949436.506794451, 868675.0042933973)
The number of IWP within the bounding box is: 1714
----------------------------------------
Cell (0, 94) has the bounds: (-1949436.506794451, 869675.0042933973, -1948436.506794451, 868675.0042933973)
The number of IWP within the bounding box is: 1357
----------------------------------------
Cell (0, 95) has the bounds: (-1948436.506794451, 869675.0042933973, -1947436.506794451, 868675.0042933973)
The number of IWP within the bounding box is: 1665
----------------------------------------
Cell (0, 96) has the bounds: (-1947436.506794451, 869675.0042933973, -1946436.506794451, 868675.0042933973)
```

Check area sum

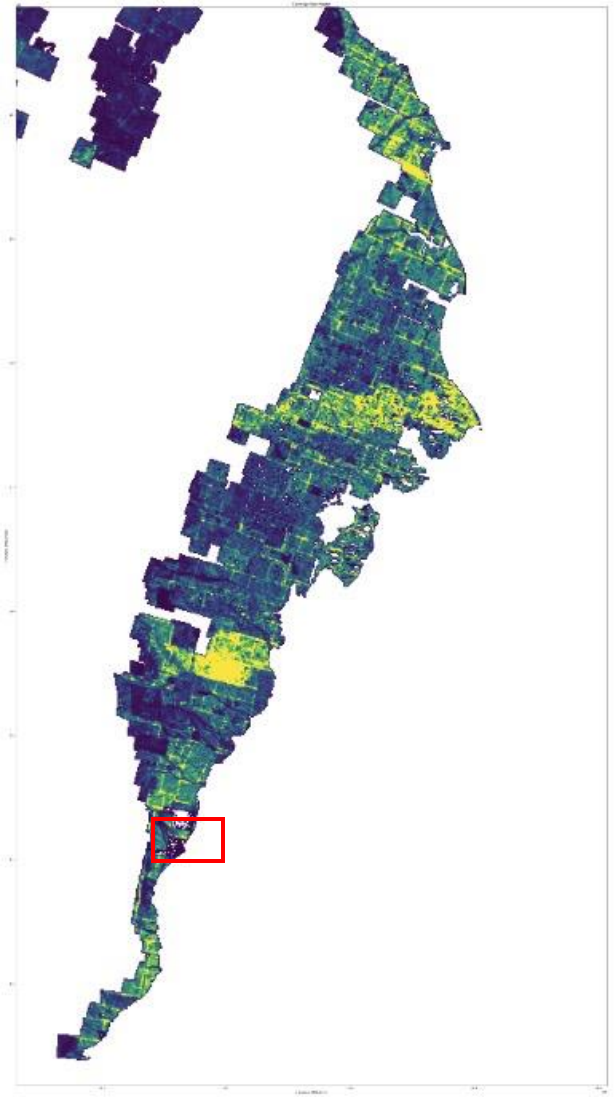
ArcGIS



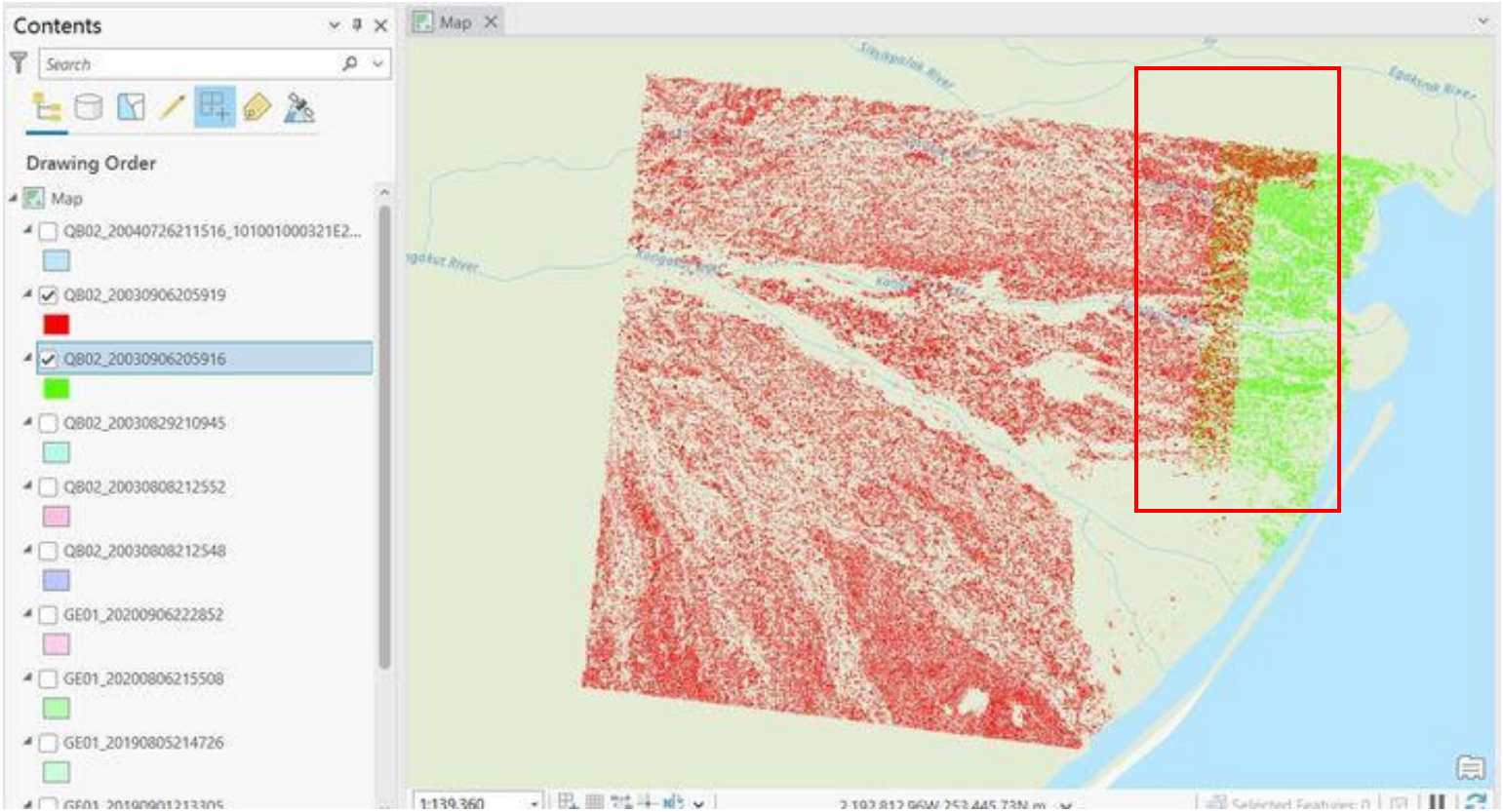


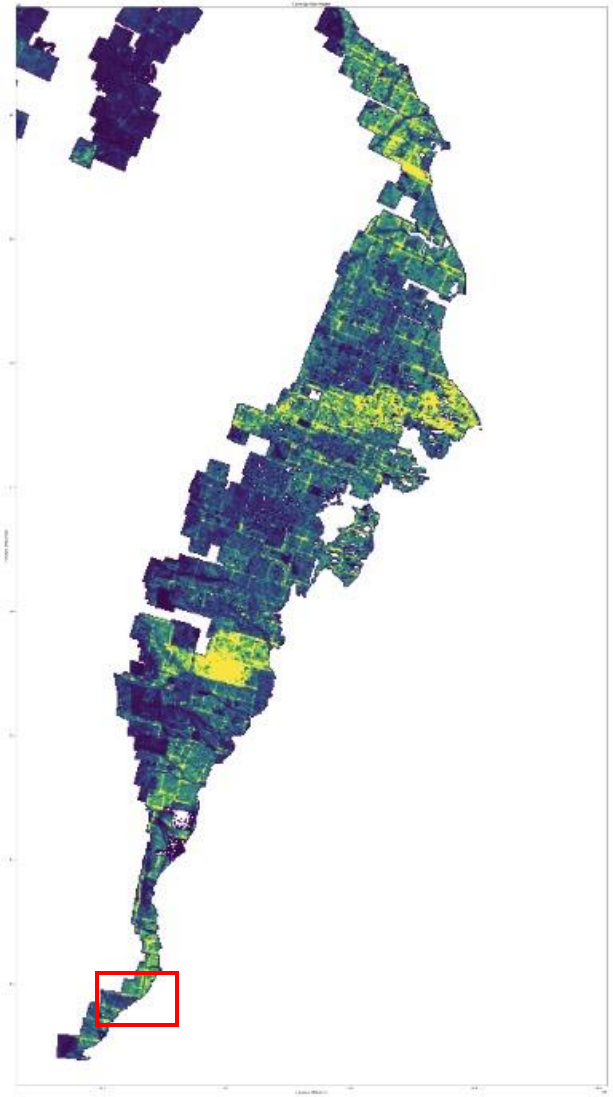


Overlapped area

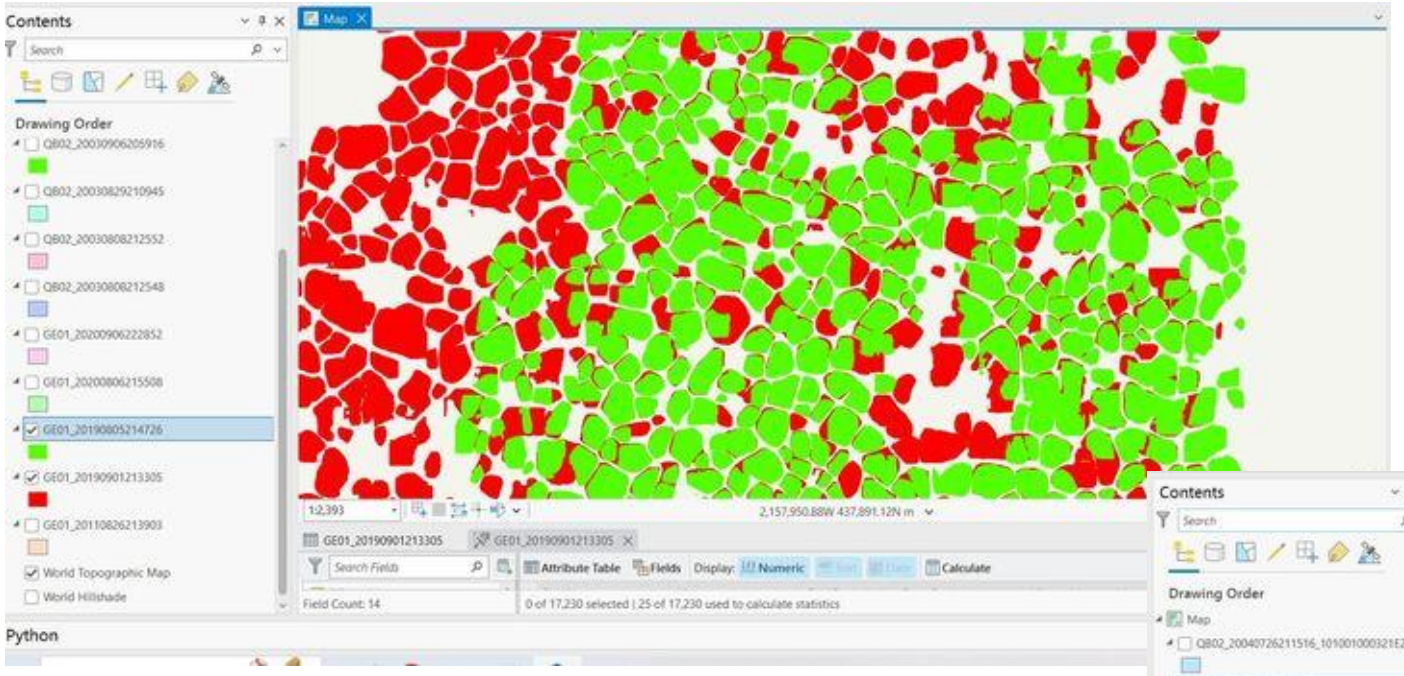


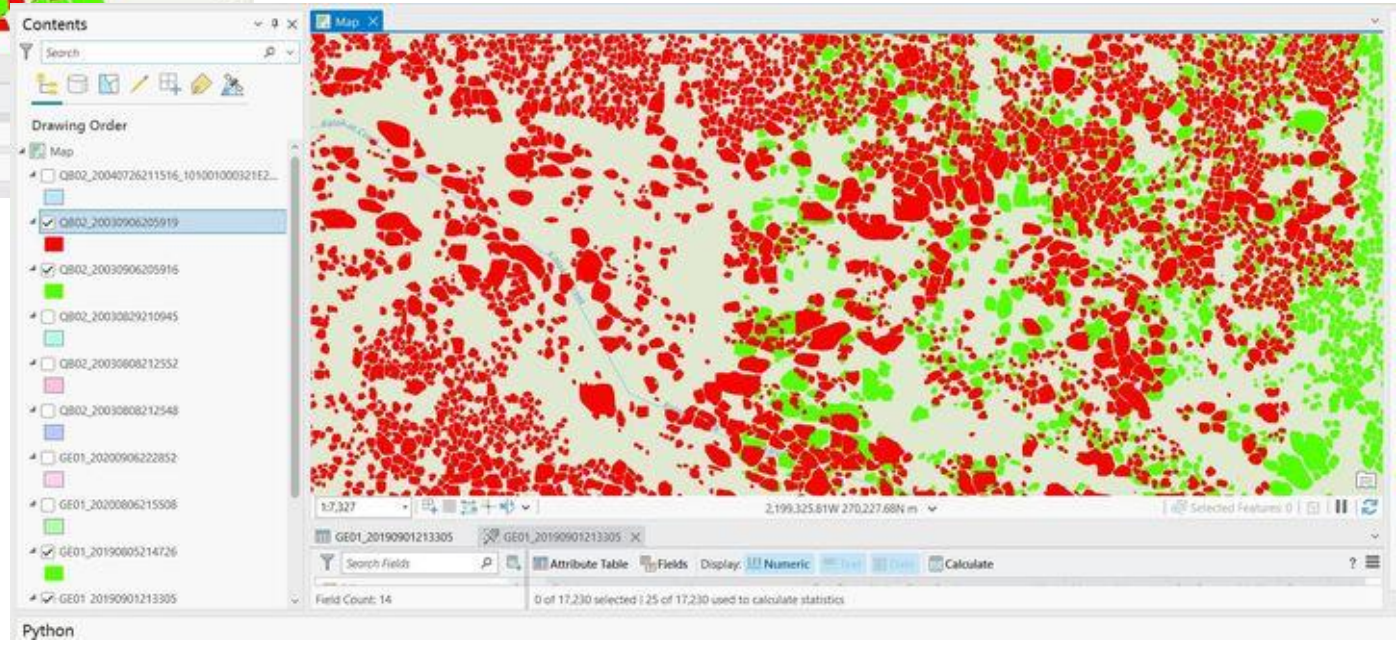
Overlapped area



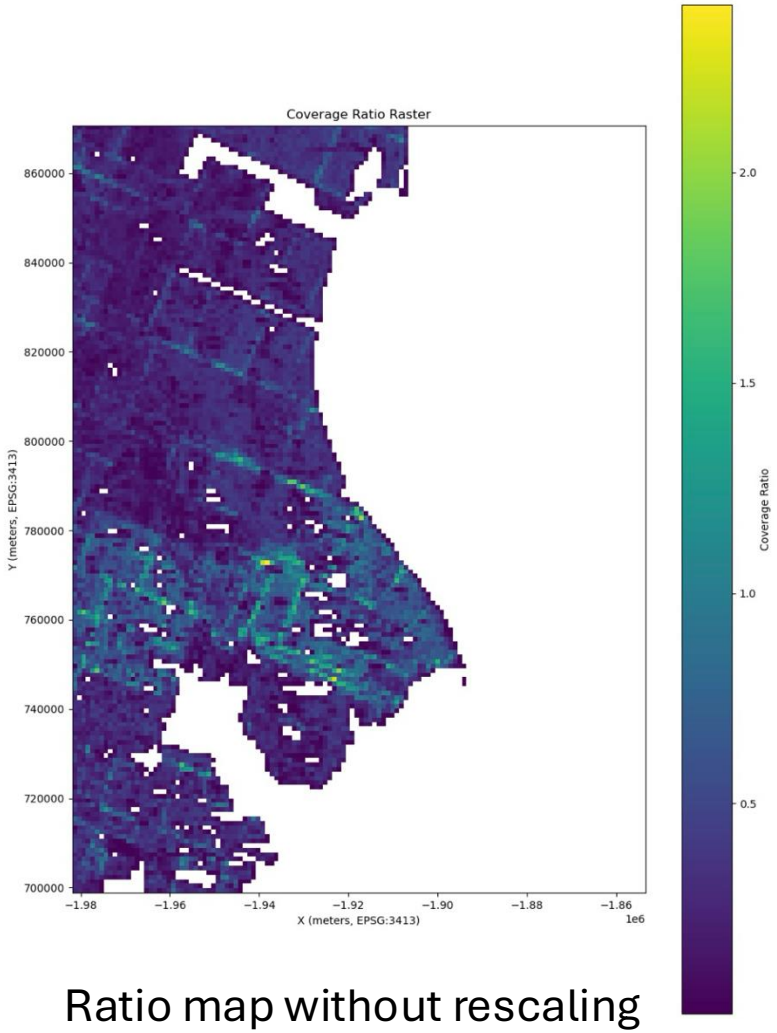


Overlapped area

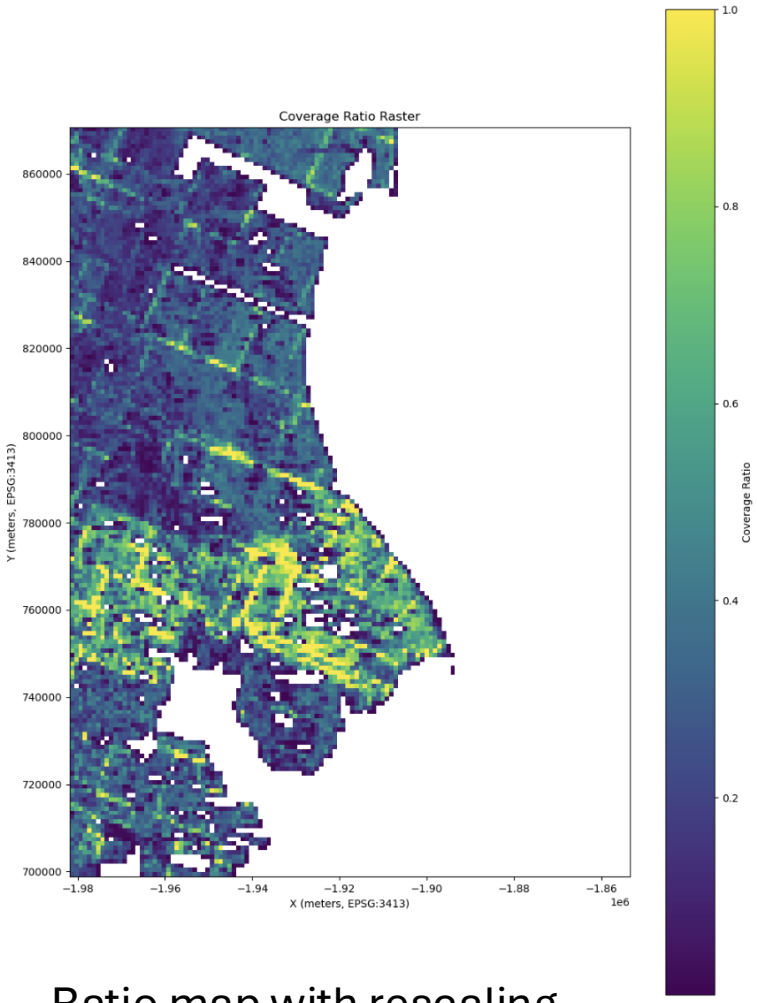




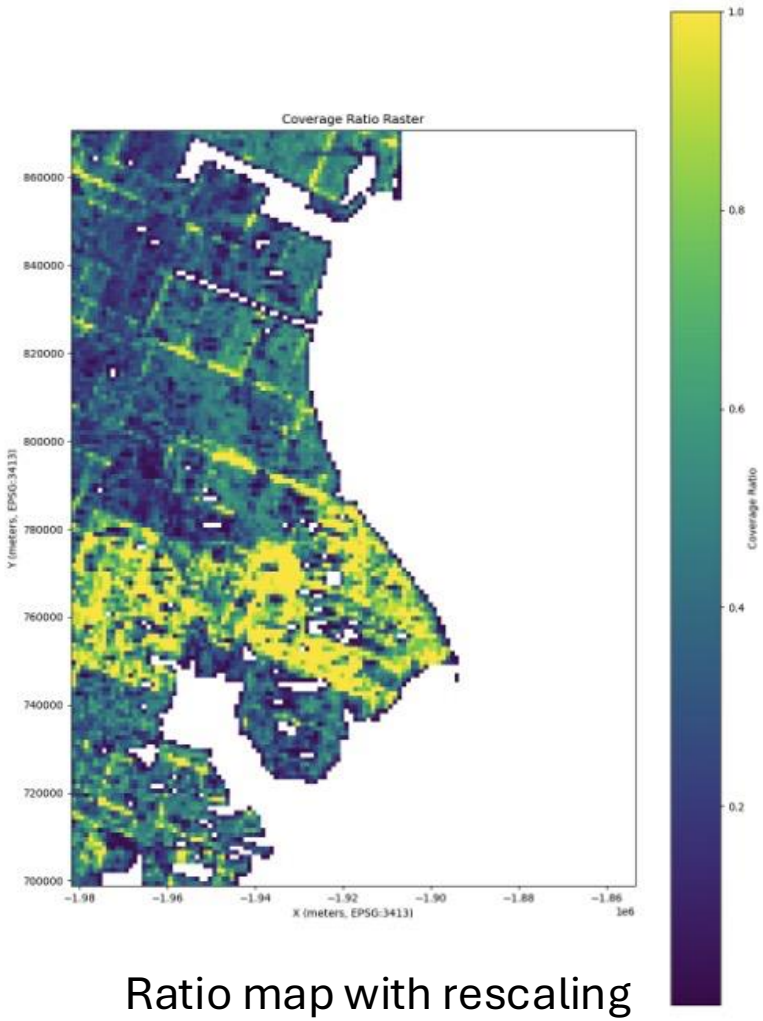
Delete buffer



Ratio map without rescaling
(after deleting buffer)

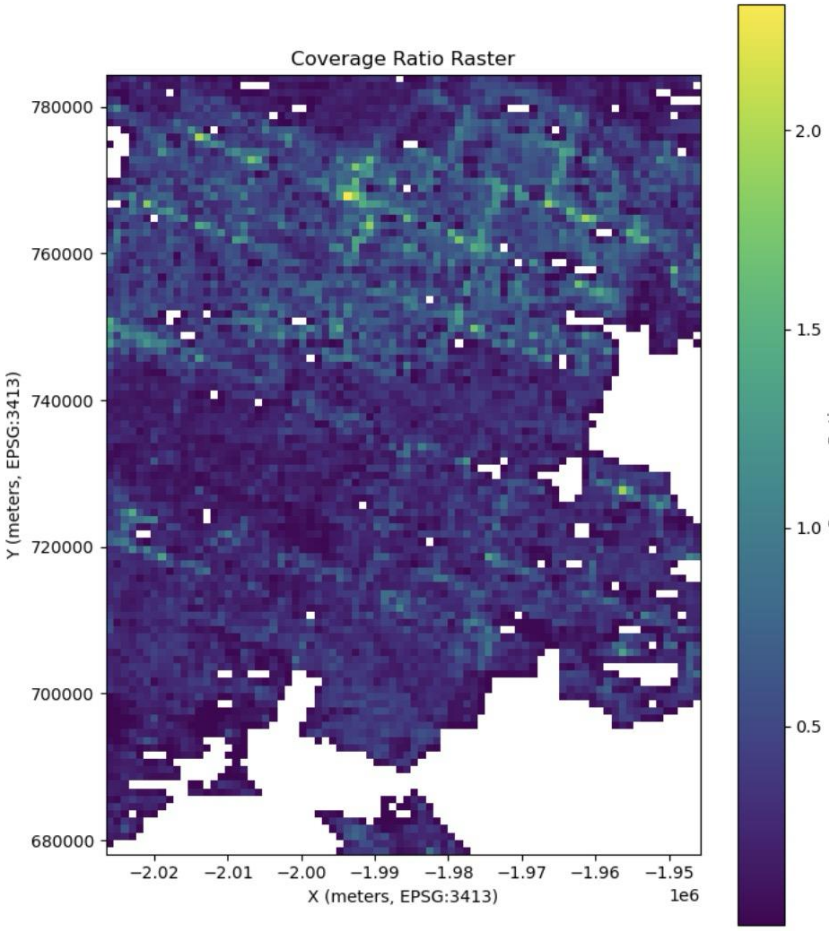


Ratio map with rescaling
(after deleting buffer)

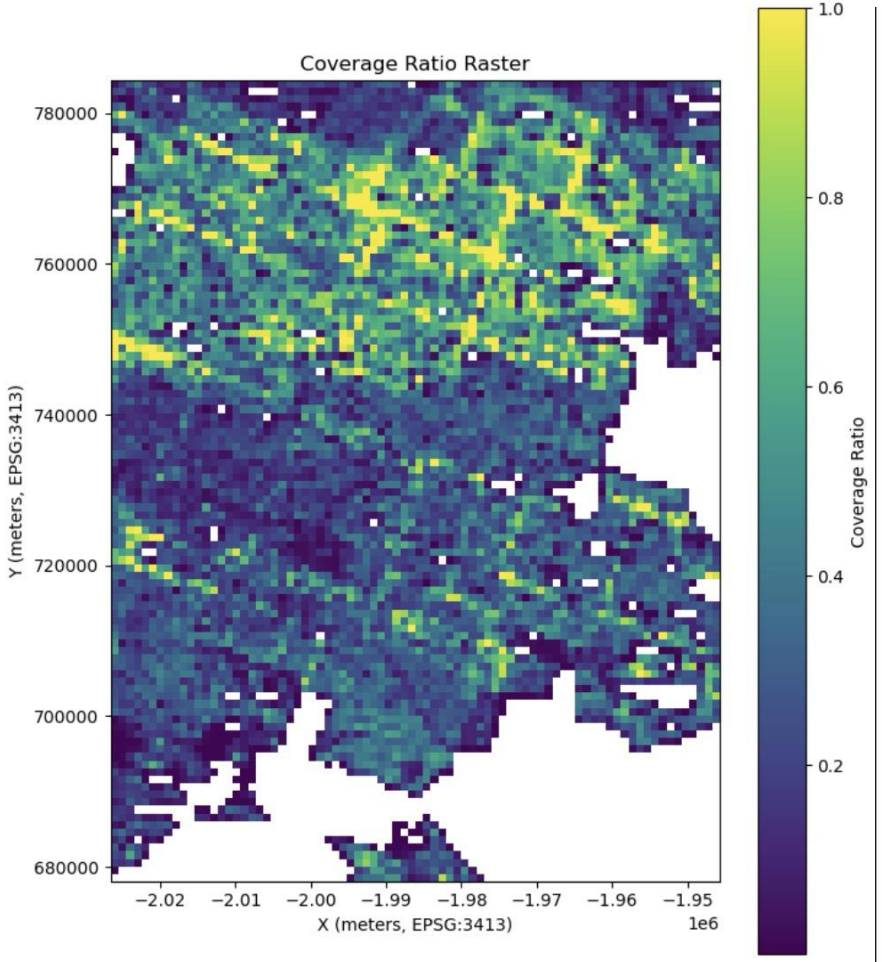


Ratio map with rescaling
(before deleting buffer)

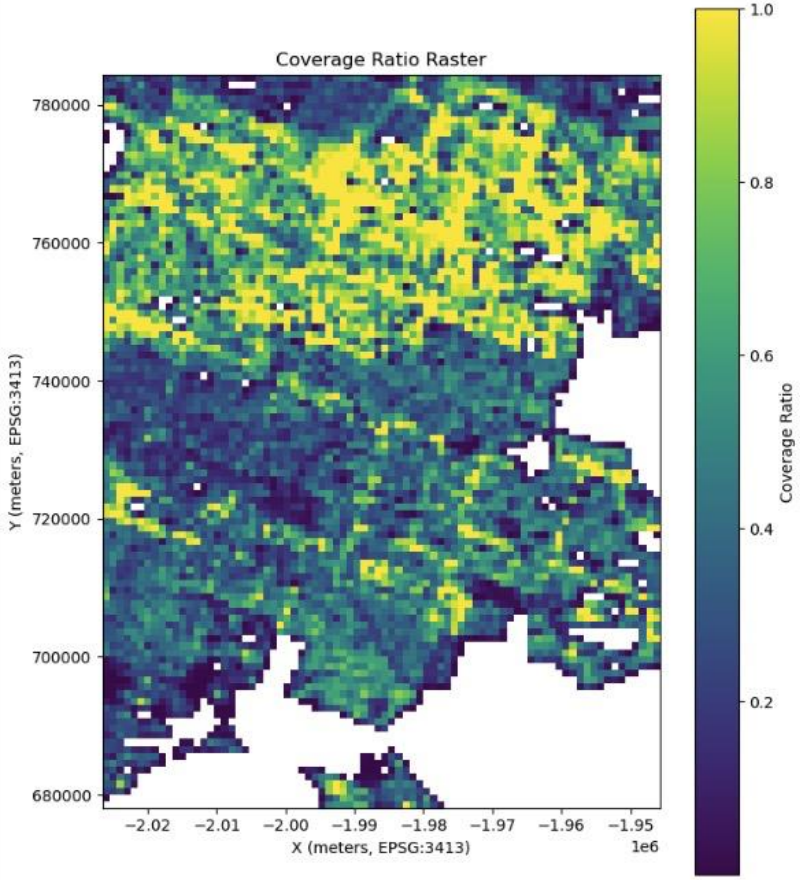
Delete buffer



Ratio map without rescaling
(after deleting buffer)

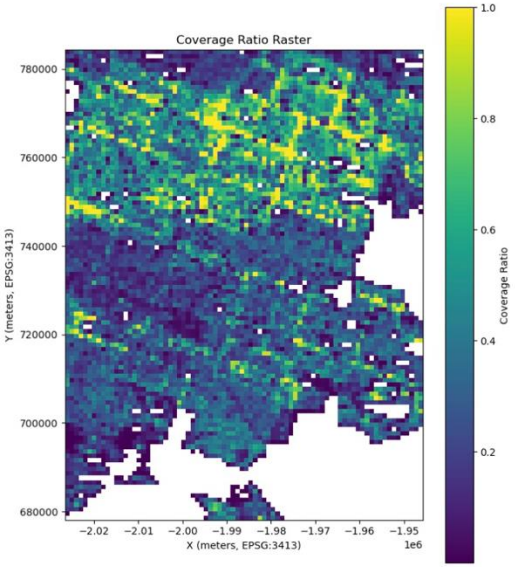


Ratio map with rescaling
(after deleting buffer)

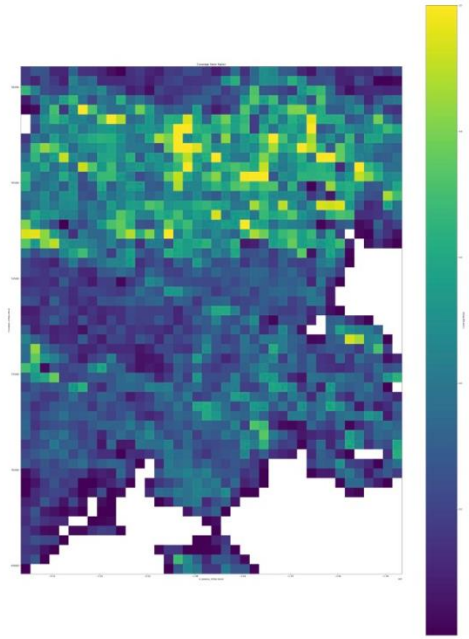


Ratio map with rescaling
(before deleting buffer)

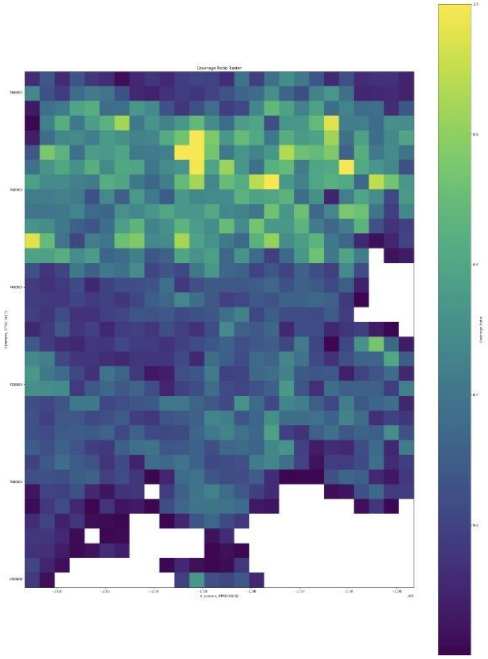
Aggregate from 1*1km



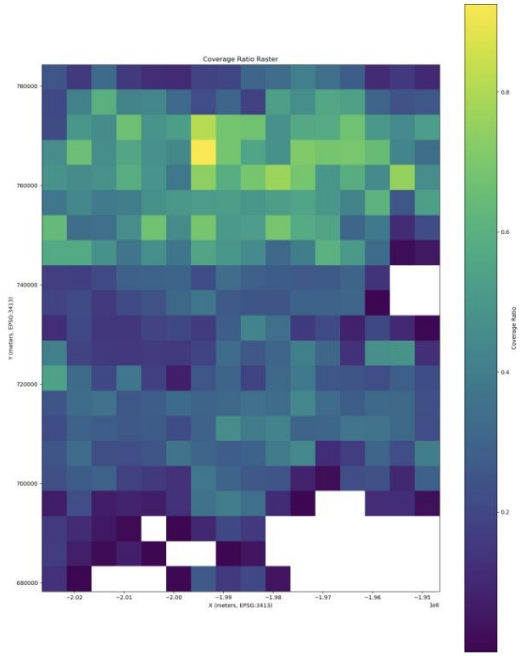
1*1km



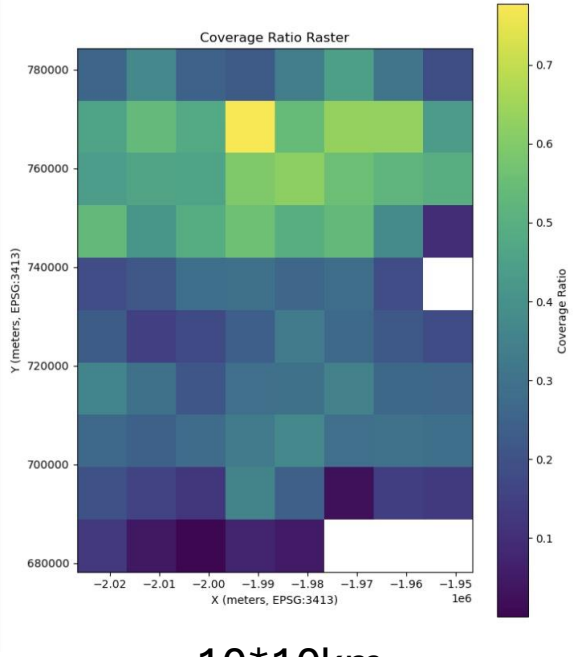
2*2km



3*3km



5*5km



10*10km

Meeting 10/08/2024

Deduplication of footprint

- Using geopackages (tile index)
- Deduplicate in query per pixel
- Dedupilcate from the shapefile

Geopackage downloading

```
(base) xchen@cici2labasuedu:~/data/geopackages/iwp_geopackage_high/WGS1984Quad/15$ ls
100     11599   13200   14786   16434   18029   19727   2408    3994    43410   44996   4658    4950    6536    8375
1000    116     13201   14787   16435   1803    19728   2409    3995    43411   44997   46580   4951    6537    8376
10000   11600   13202   14788   16436   18030   19729   241     3996    43412   44998   46581   4952    6538    8377
10001   11601   13203   14789   16437   18031   19730   2410    3997    43413   44999   46582   4953    6539    8378
10002   11602   13204   1479    16438   18032   19731   2411    3998    43414   45      46583   4954    654     8379
10003   11603   13205   14790   16439   18033   19732   2412    3999    43415   450     46584   4955    6540    8380
10004   11604   13206   14791   1644    18034   19733   2413    40      43416   4500    46585   4956    6541    8381
10005   11605   13207   14792   16440   18035   19734   2414    400     43417   45000   46586   4957    6542    8382
10006   11606   13208   14793   16441   18036   19735   2415    4000    43418   45001   46587   4958    6543    8383
10007   11607   13209   14794   16442   18037   19736   2416    4001    43419   45002   46588   4959    6544    8384
10008   11608   1321    14795   16443   18038   19737   2417    4002    4342    45003   46589   496     6545    8385
10009   11609   13210   14796   16444   18039   19738   2418    4003    43420   45004   4659    4960    6546    8386
1001    11610   13211   14797   16445   1804    19739   2419    4004    43421   45005   46590   4961    6547    8387
10010   11611   13212   14798   16446   18040   19740   242     4005    43422   45006   46591   4962    6548    8388
10011   11612   13213   14799   16447   18041   19741   2420    4006    43423   45007   46592   4963    6549    8389
10012   11613   13214   148     16448   18042   19742   2421    4007    43424   45008   46593   4964    655     8390
10013   11614   13215   1480    16449   18043   19743   2422    4008    43425   45009   46594   4965    6550    8391
10014   11615   13216   14800   1645    18044   19744   2423    4009    43426   4501    46595   4966    6551    8392
10015   11616   13217   14801   16450   18045   19745   2424    401     43427   45010   46596   4967    6552    8393
10016   11617   13218   14802   16451   18046   19746   2425    4010    43428   45011   46597   4968    6553    8394
10017   11618   13219   14803   16452   18047   19747   2426    4011    43429   45012   46598   4969    6554    8395
10018   11619   1322    14804   16453   18048   19748   2427    4012    4343    45013   46599   497     6555    8396
10019   11620   13220   14805   16454   18049   19749   2428    4013    43430   45014   466     4970    6556    8397
1002    11621   13221   14806   16455   1805    19750   2429    4014    43431   45015   4660    4971    6557    8398
10020   11622   13222   14807   16456   18050   19751   243     4015    43432   45016   46600   4972    6558    8399
10021   11623   13223   14808   16457   18051   19752   2430    4016    43433   45017   46601   4973    6559    84
10022   11624   13224   14809   16458   18052   19753   2431    4017    43434   45018   46602   4974    656     8400
10023   11625   13225   1481    16459   18053   19754   2432    4018    43435   45019   46603   4975    6560    8401
10024   11626   13226   14810   1646    18054   19755   2433    4019    43436   4502    46604   4976    6561    8402
10025   11627   13227   14811   16460   18055   19756   2434    402     43437   45020   46605   4977    6562    8403
10026   11628   13228   14812   16461   18056   19757   2435    4020    43438   45021   46606   4978    6563    8404
10027   11629   13229   14813   16462   18057   19758   2436    4021    43439   45022   46607   4979    6564    8405
10028   11630   1323    14814   16463   18058   19759   2437    4022    4344    45023   46608   498     6565    8406
10029   11631   13230   14815   16464   18059   19760   2438    4023    43440   45024   46609   4980    6566    8407
1003    11632   13231   14816   16465   1806    19761   2439    4024    43441   45025   4661    4981    6567    8408
10030   11633   13232   14817   16466   18060   19762   244     4025    43442   45026   46610   4982    6568    8409
```

Directories: >40,000

Z-x-y

After we process the input data, the output tiles are organized by z-y-x indices for the tileset, and we no longer use the original file hierarchy. There are two main reasons for this

You could check which tiles intersect with Alaska, e.g. in python you could use morecantile and and an overlay operation in geopandas.

~ 90 geopackages

```
(alaska_shpMapping2) (base) xchen@cici2labasuedu:~/data/geopackages/iwp_geopackage_high/WGS1984Quad/15/100$ ls
3358.gpkg   3368.gpkg   3378.gpkg   3388.gpkg   3398.gpkg   3408.gpkg   3418.gpkg   3428.gpkg   3452.gpkg
3359.gpkg   3369.gpkg   3379.gpkg   3389.gpkg   3399.gpkg   3409.gpkg   3419.gpkg   3429.gpkg   3455.gpkg
3360.gpkg   3370.gpkg   3380.gpkg   3390.gpkg   3400.gpkg   3410.gpkg   3420.gpkg   3430.gpkg   3465.gpkg
3361.gpkg   3371.gpkg   3381.gpkg   3391.gpkg   3401.gpkg   3411.gpkg   3421.gpkg   3431.gpkg   3470.gpkg
3362.gpkg   3372.gpkg   3382.gpkg   3392.gpkg   3402.gpkg   3412.gpkg   3422.gpkg   3432.gpkg   3471.gpkg
3363.gpkg   3373.gpkg   3383.gpkg   3393.gpkg   3403.gpkg   3413.gpkg   3423.gpkg   3433.gpkg   index.html
3364.gpkg   3374.gpkg   3384.gpkg   3394.gpkg   3404.gpkg   3414.gpkg   3424.gpkg   3434.gpkg   'index.html?C=D;O=A'
3365.gpkg   3375.gpkg   3385.gpkg   3395.gpkg   3405.gpkg   3415.gpkg   3425.gpkg   3435.gpkg   'index.html?C=M;O=A'
3366.gpkg   3376.gpkg   3386.gpkg   3396.gpkg   3406.gpkg   3416.gpkg   3426.gpkg   3436.gpkg   'index.html?C=N;O=D'
3367.gpkg   3377.gpkg   3387.gpkg   3397.gpkg   3407.gpkg   3417.gpkg   3427.gpkg   3440.gpkg   'index.html?C=S;O=A'
```

Alaska tiles

5509 Dell – front desk hard drive

```
Tile(x=33055, y=3378, z=15)
Tile(x=33056, y=3378, z=15)
Tile(x=33057, y=3378, z=15)
Tile(x=33058, y=3378, z=15)
Tile(x=33059, y=3378, z=15)
Tile(x=33060, y=3378, z=15)
Tile(x=33061, y=3378, z=15)
Tile(x=33062, y=3378, z=15)
Tile(x=33063, y=3378, z=15)
Tile(x=33064, y=3378, z=15)
Tile(x=33065, y=3378, z=15)
```

Meeting 10/15/2024

- Geopackage small region test

Docker upgrade

Before

```
root@7bf794b0c6af:/# lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
```

```
root@7bf794b0c6af:/# ogrinfo --version
GDAL 2.4.0, released 2018/12/14
```

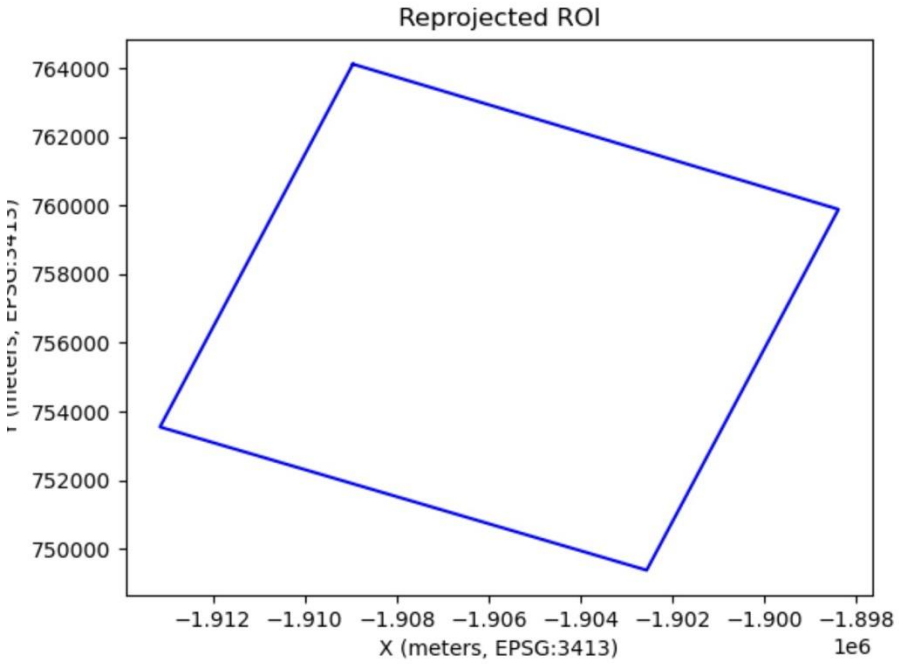
Upgrade

After

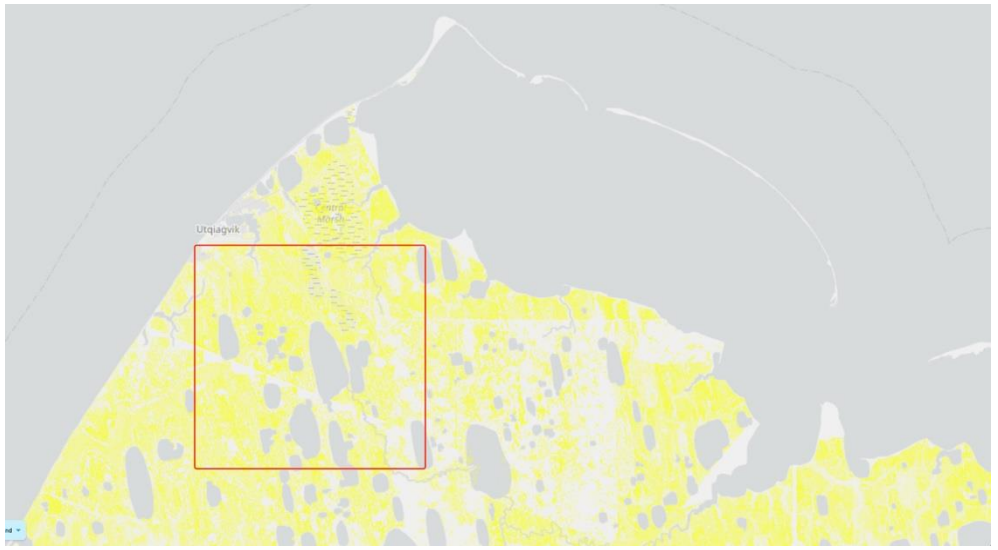
```
root@506f34fcbdca:/# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.4 LTS
Release:        22.04
Codename:       jammy
```

```
root@506f34fcbdca:/# ogr2ogr --version
GDAL 3.6.4, released 2023/04/17
```

Test data region



EPSG: 3413

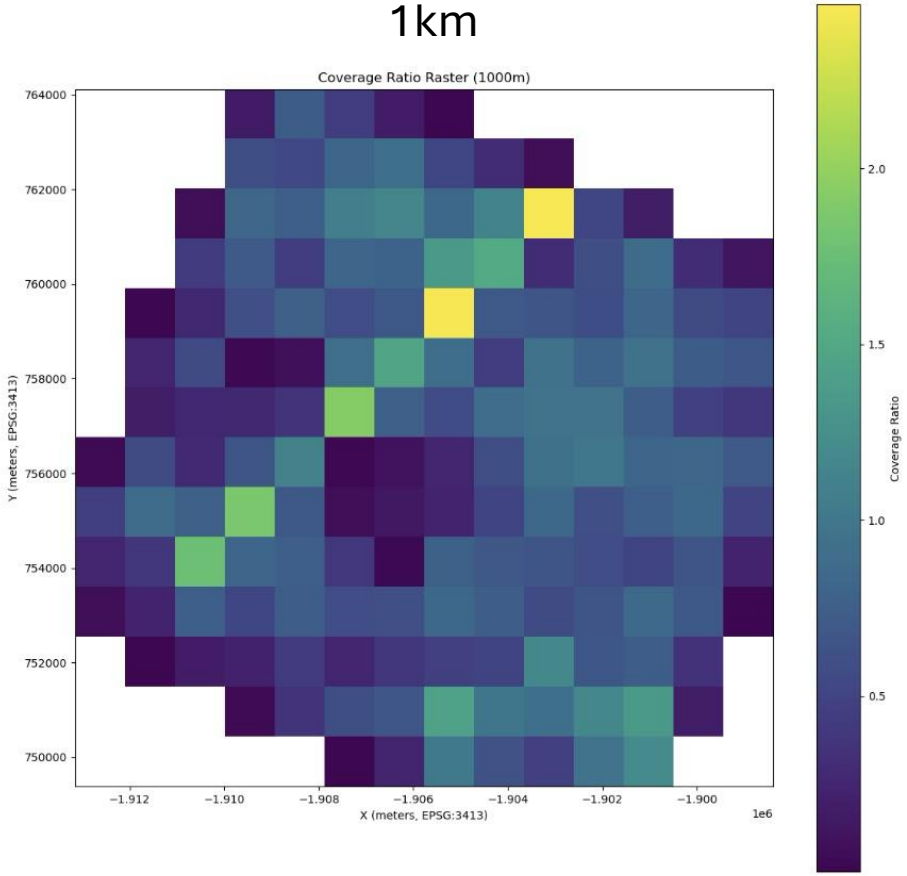


EPSG: 4326

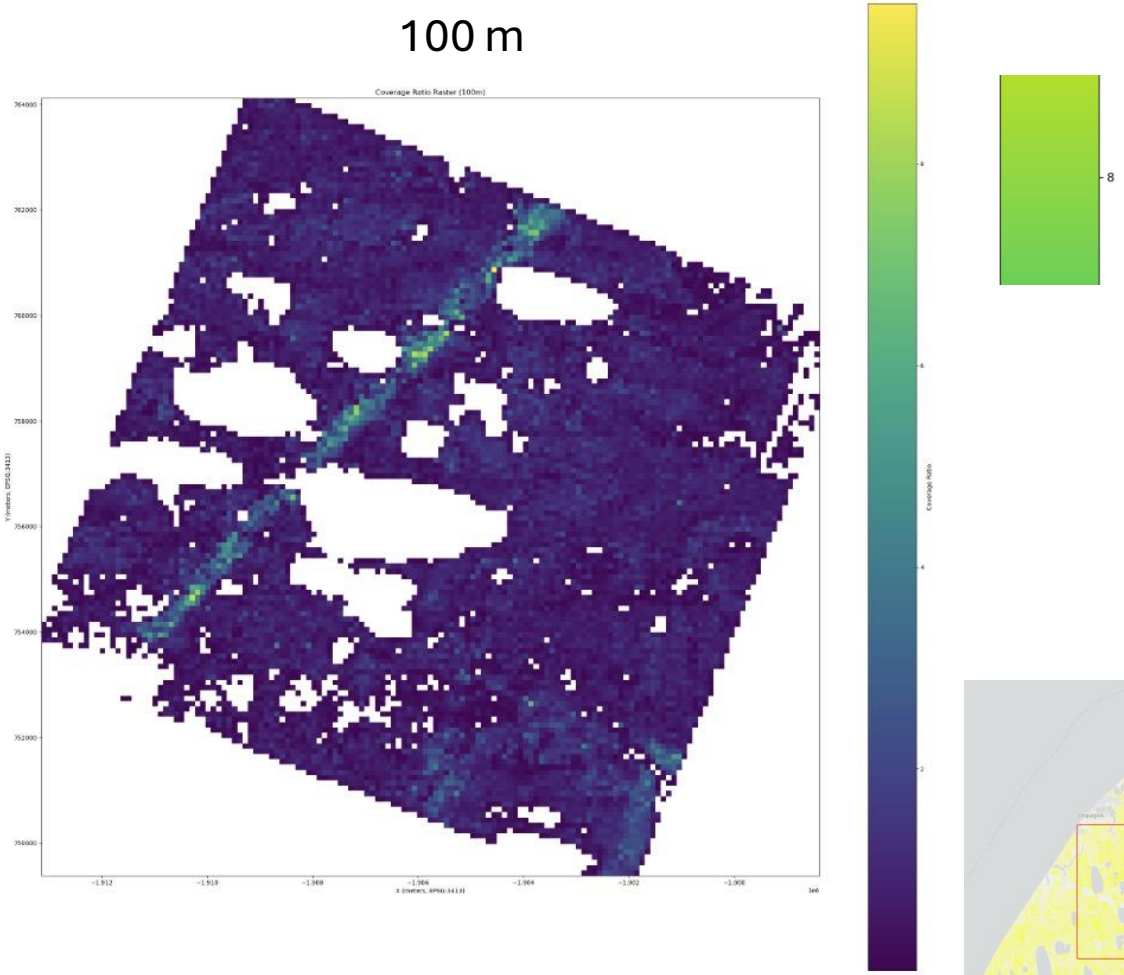
Coverage ratio raster at small scale

Before Deduplicated

1km



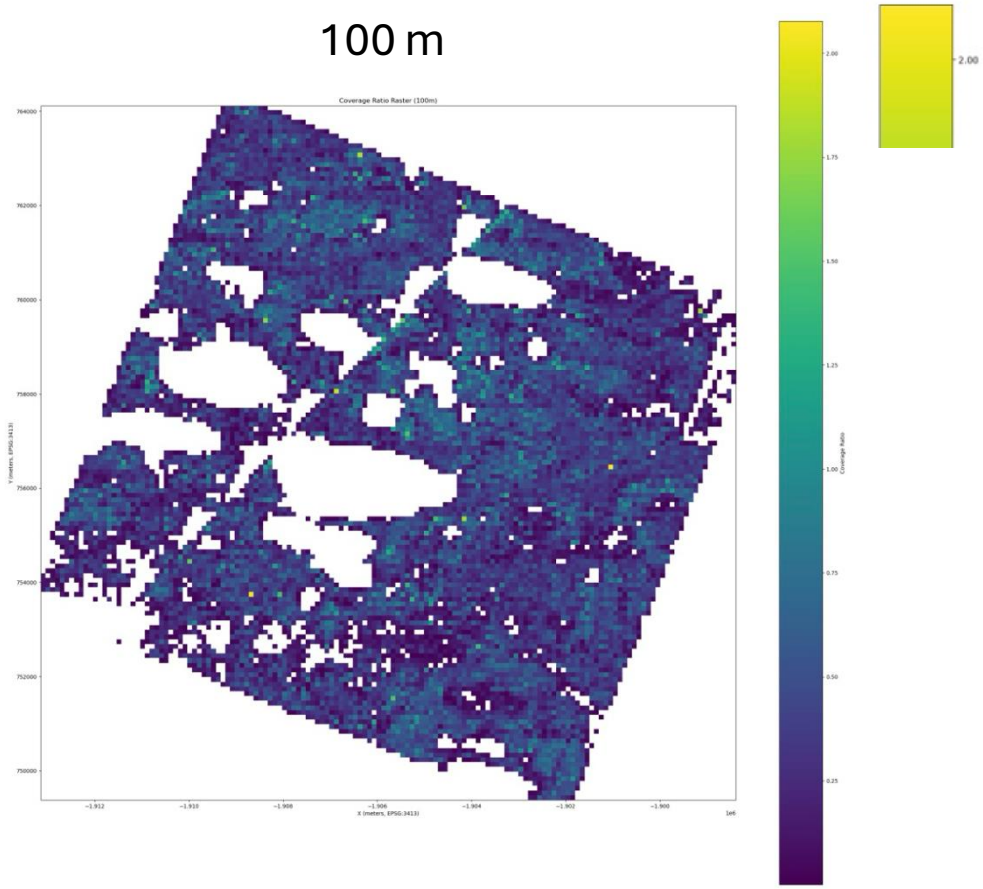
100 m



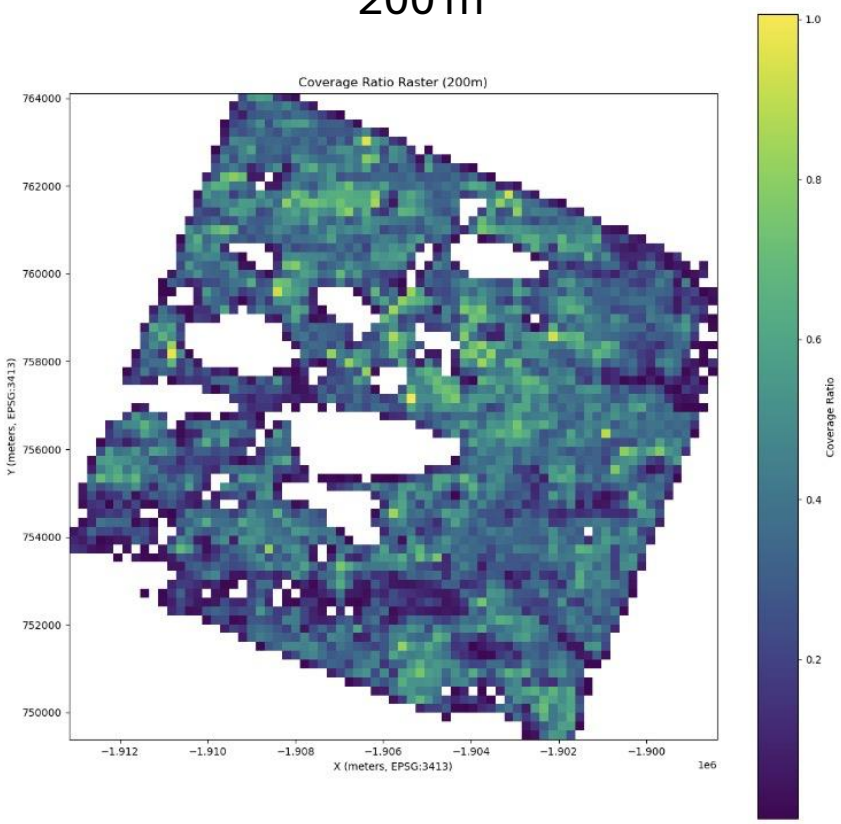
Coverage ratio raster at median scale

After Deduplicated

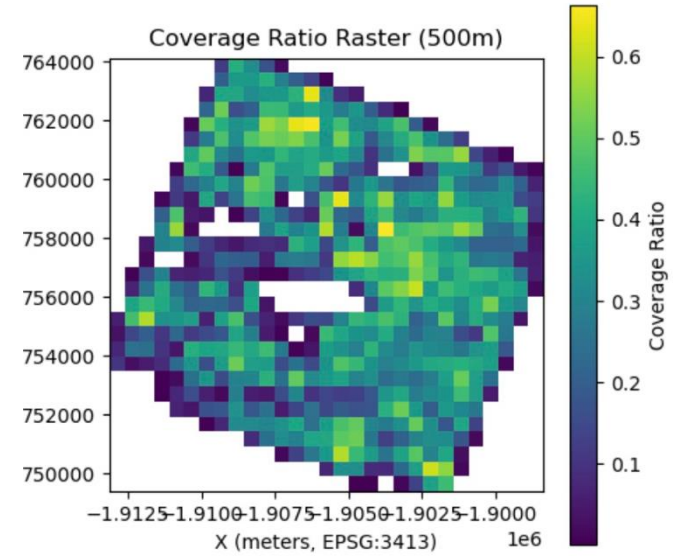
100 m



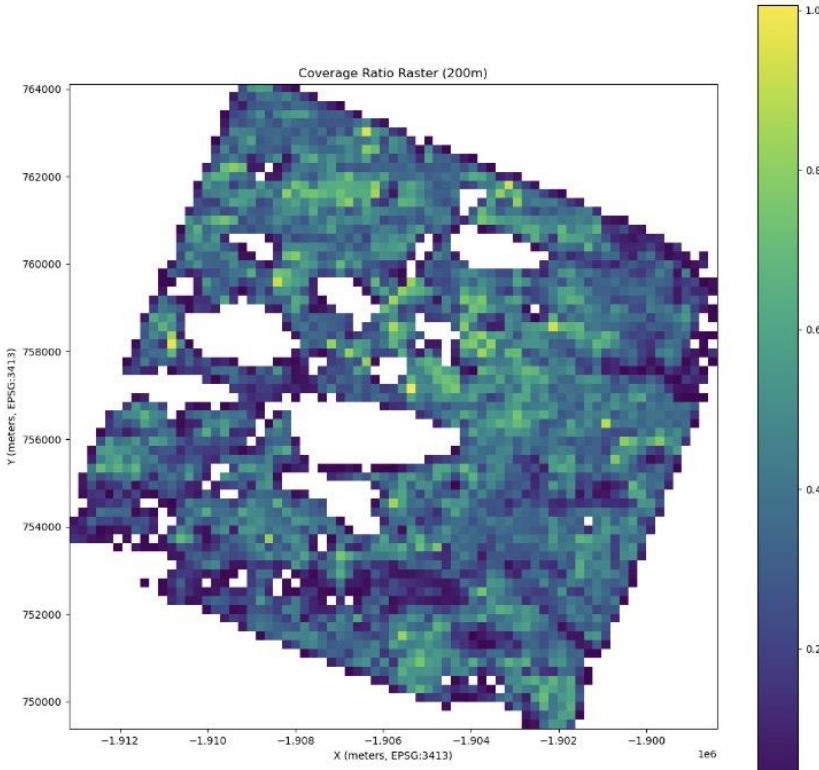
200 m



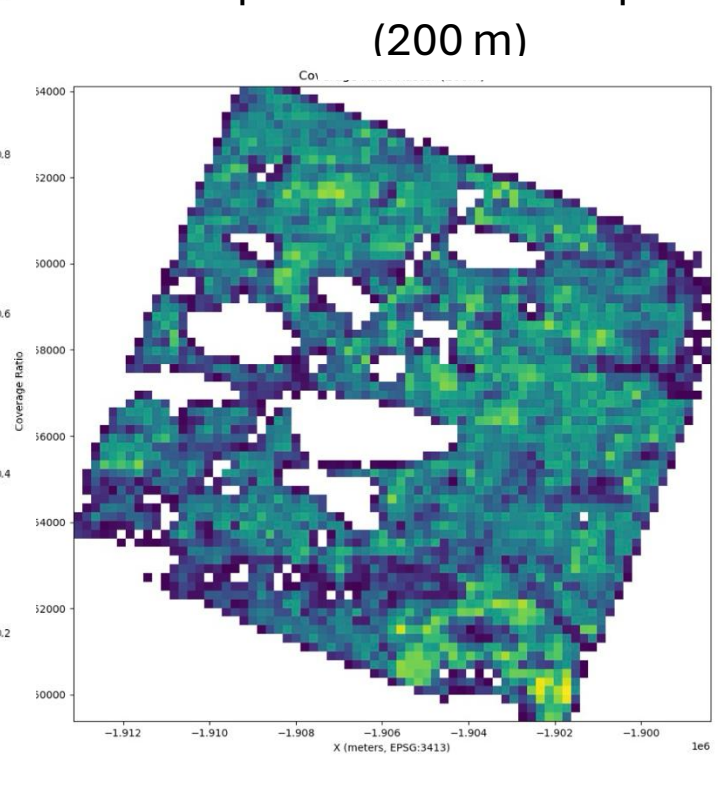
500 m



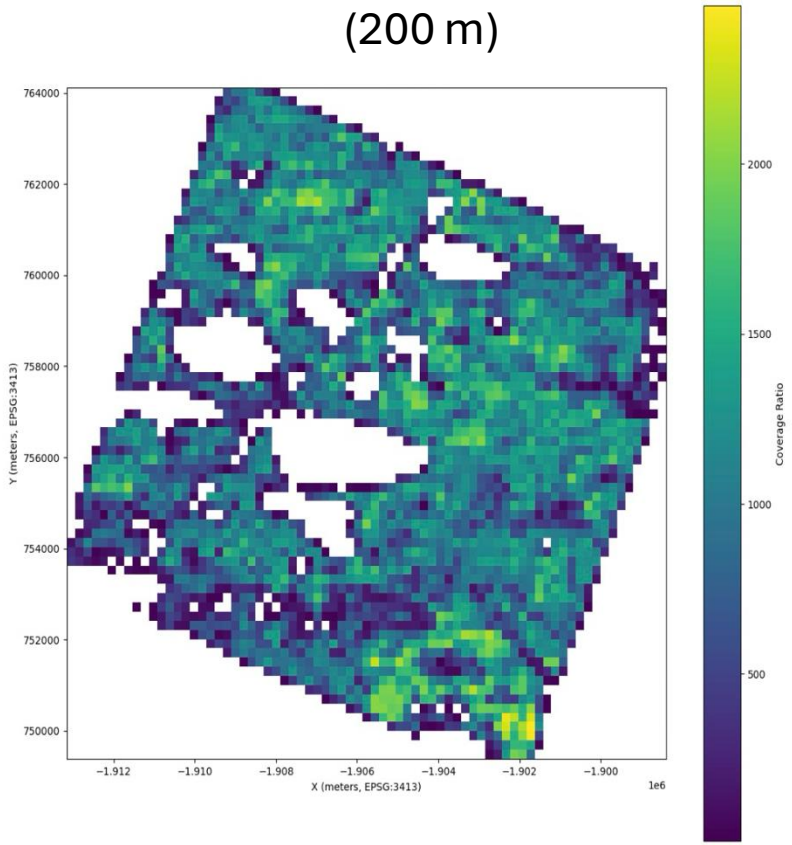
Different maps



perimeter sum map (200 m)



Length sum map (200 m)



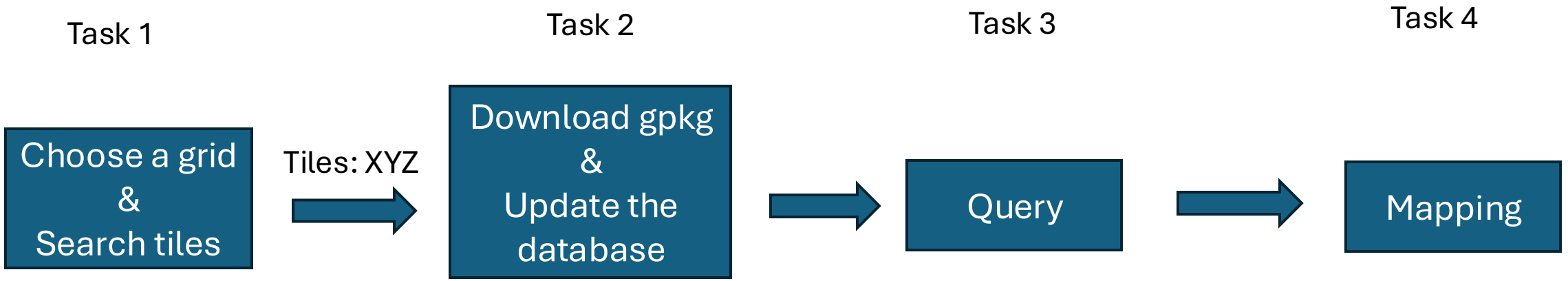
Meeting 10/22/2024

- Create grids in Pan-Arctic area

- Pick a grid (256*256km) generate coverage ratio/perimeter/length/width map
  - Validation

- Automatic the one pass progress
  - Validation

Building data pipeline in one grid



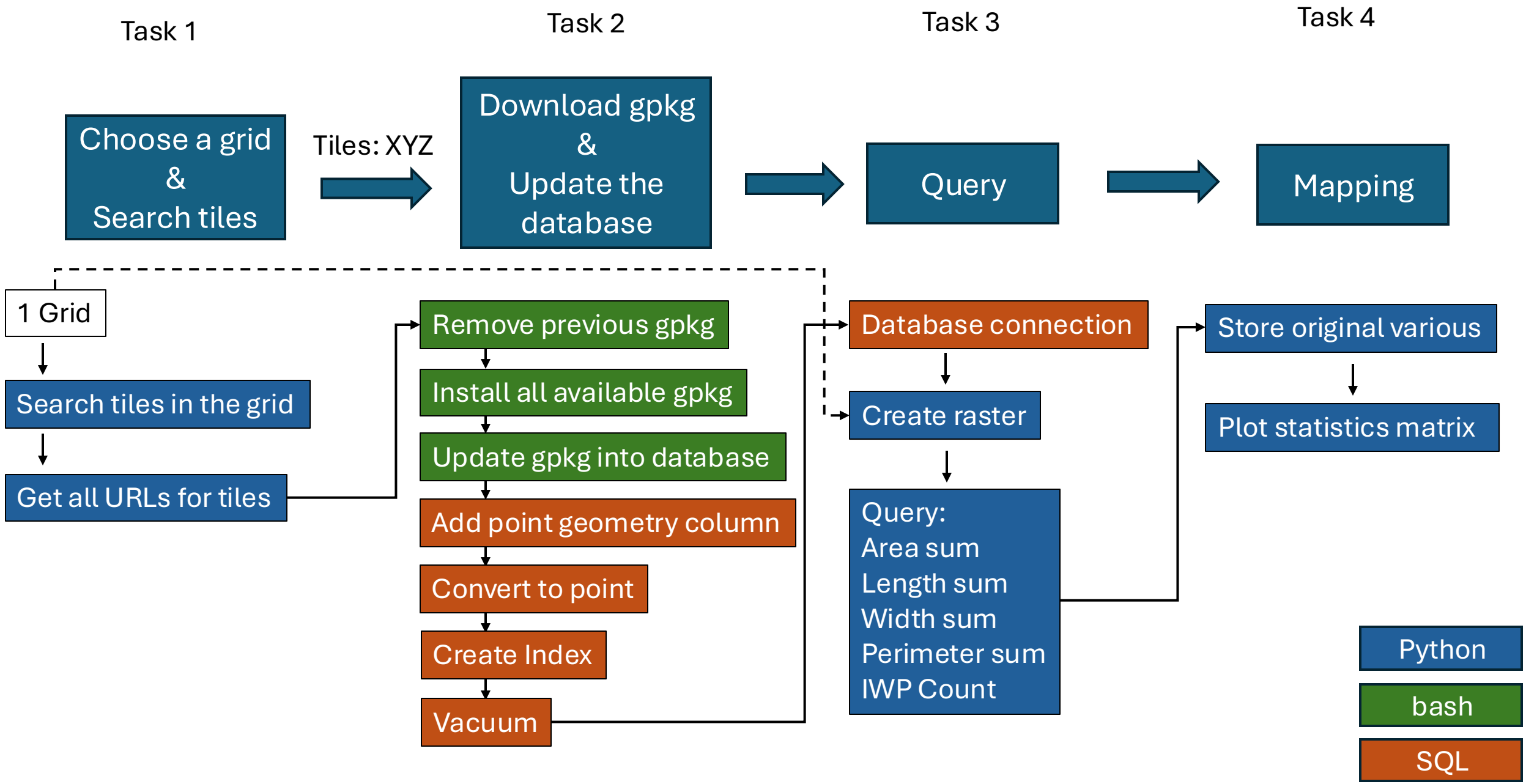
(Choose 1 grid and search for tiles in the grid)

(Download all gpkg files in those tiles)

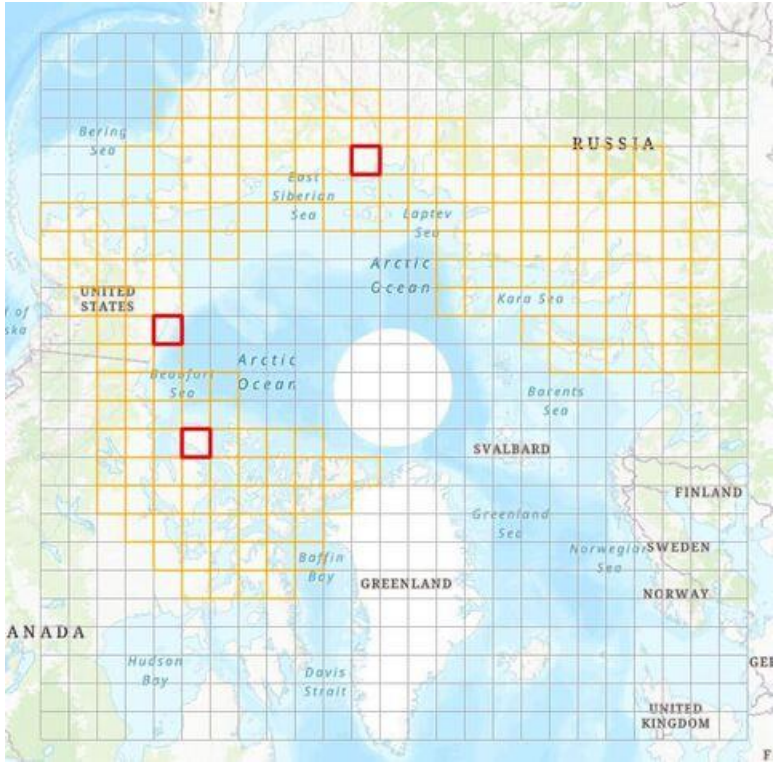
(Query all data from database within the grid we chose in Search step)

(Map the statistics information of data)

Building data pipeline in one grid



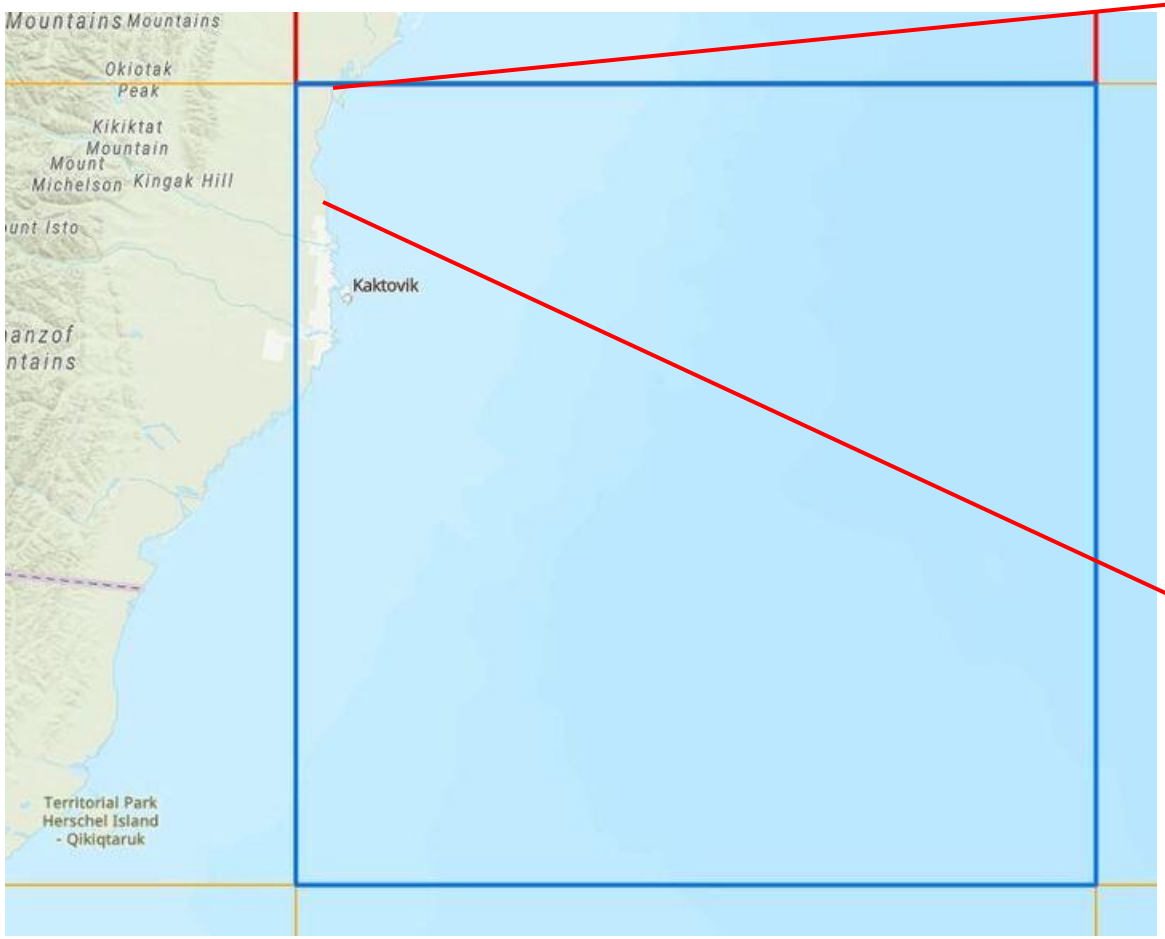
Create Grids (256km*256km)

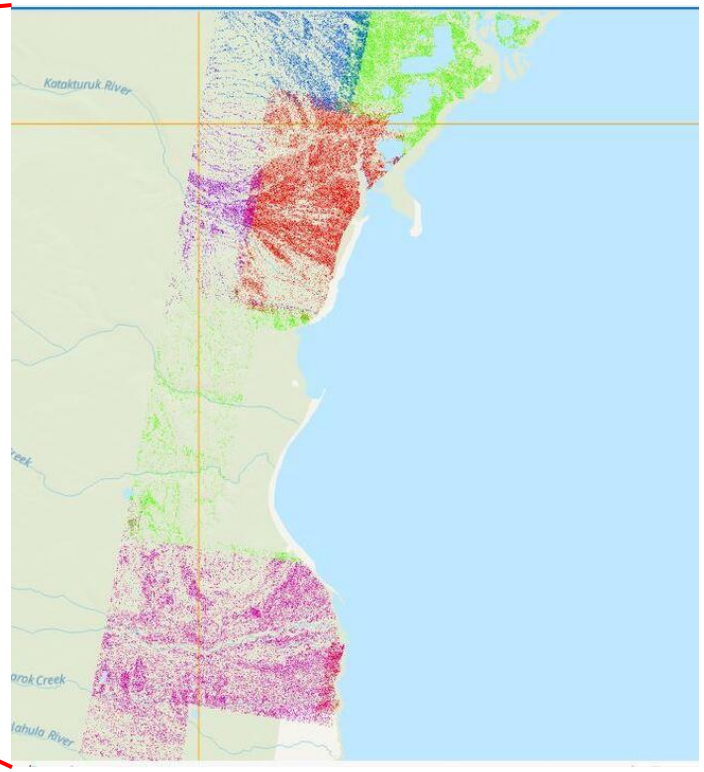


EPSG: 3413

EPSG: 4326

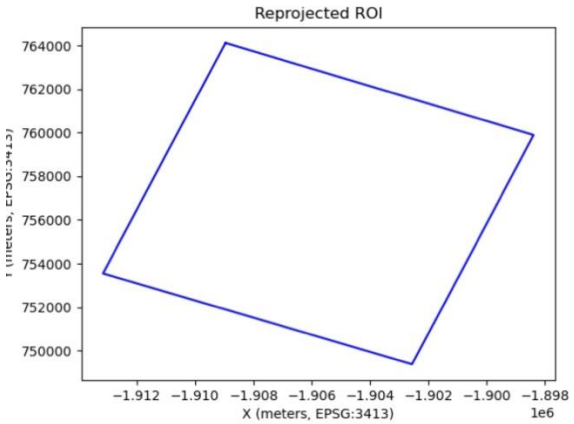
Select a grid to test

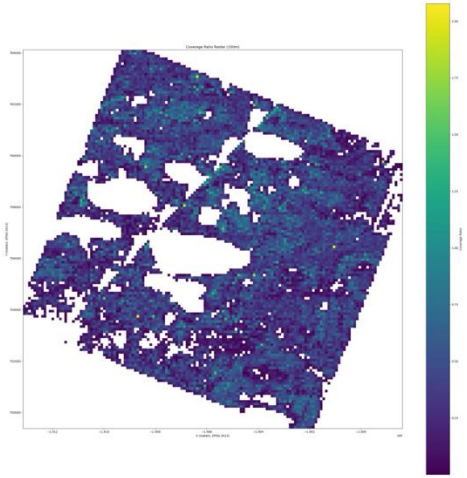




ROI projection

Previous:



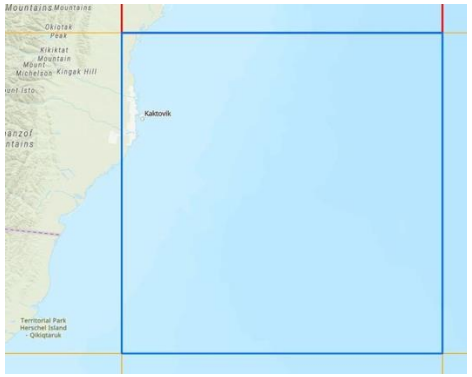


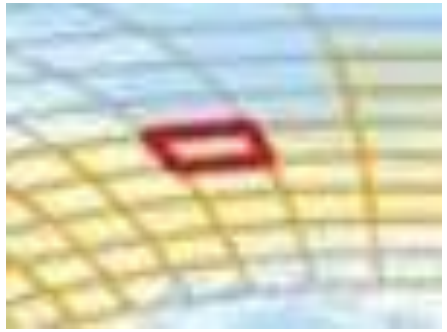
Search tiles: EPSG: 4326

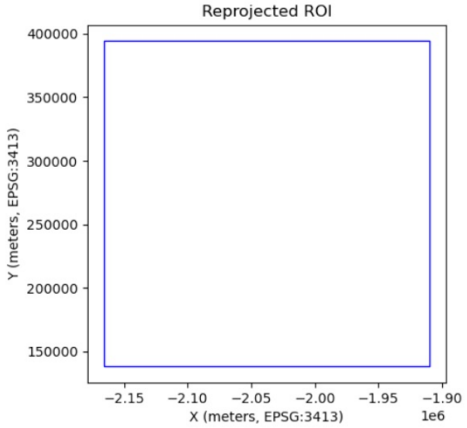
Query: EPSG: 3413

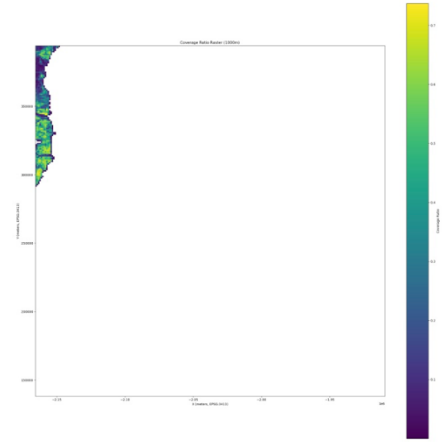
Visualization: EPSG: 3413

Current:









Create Grids: EPSG: 3413

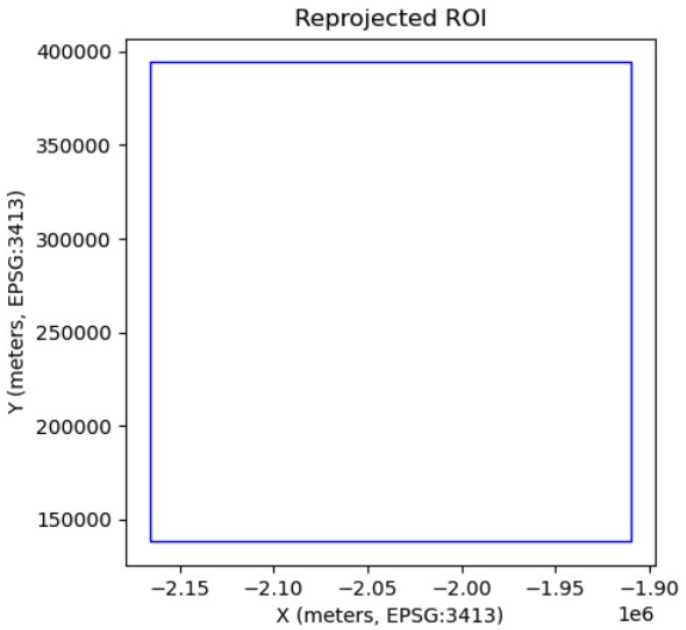
Search tiles: EPSG: 4326

Search IWPs: EPSG: 3413

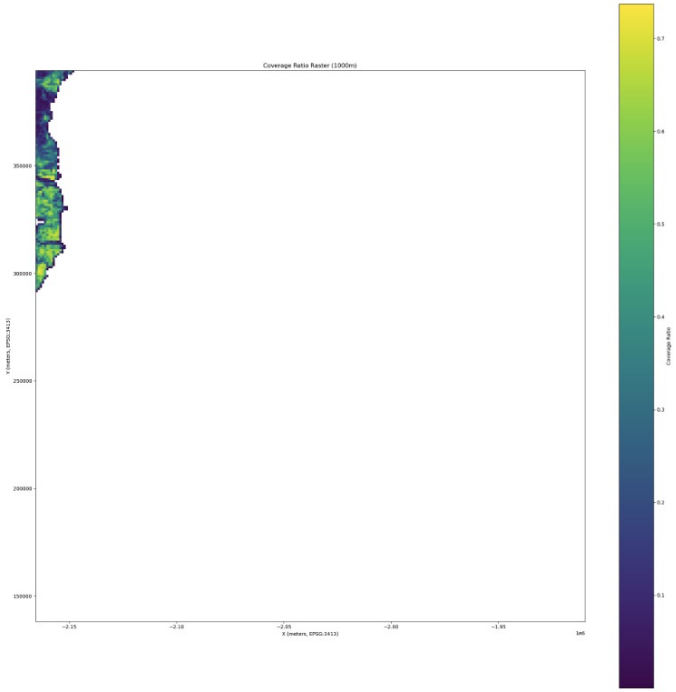
Visualization: EPSG: 3413

Validation: projection

EPSG: 3413



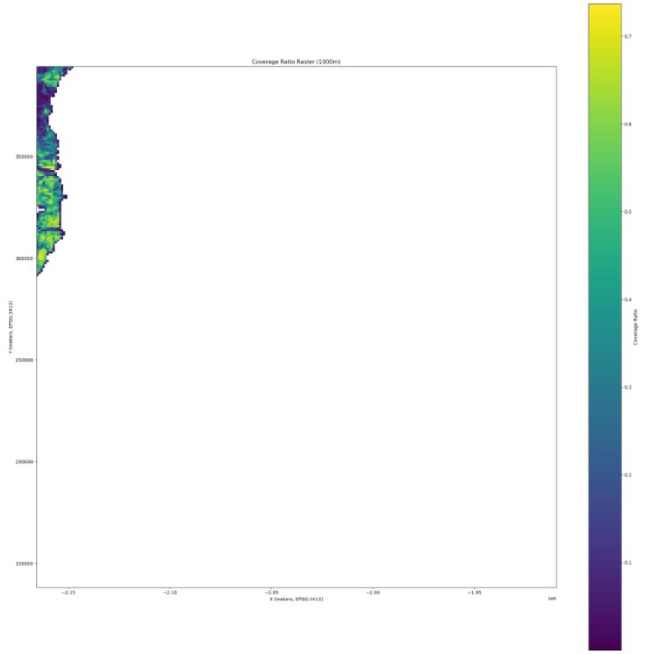
EPSG: 3413



EPSG: 3413

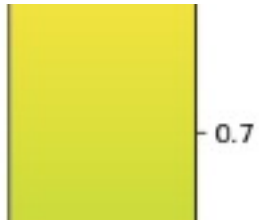
256*256 pixels

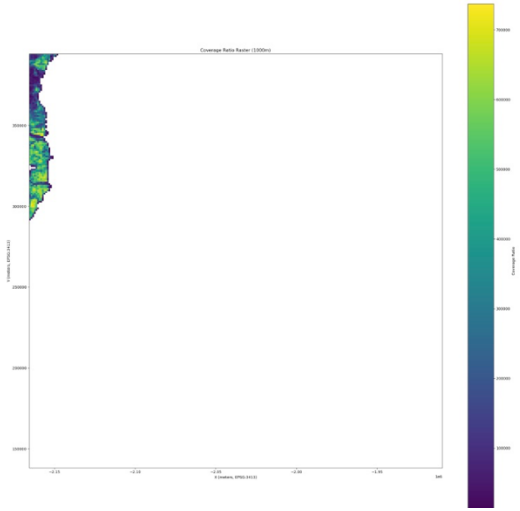
Validation: value



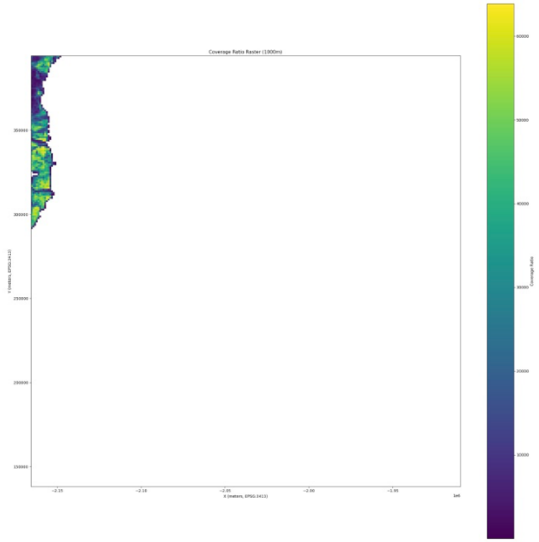
Coverage Ratio

Max





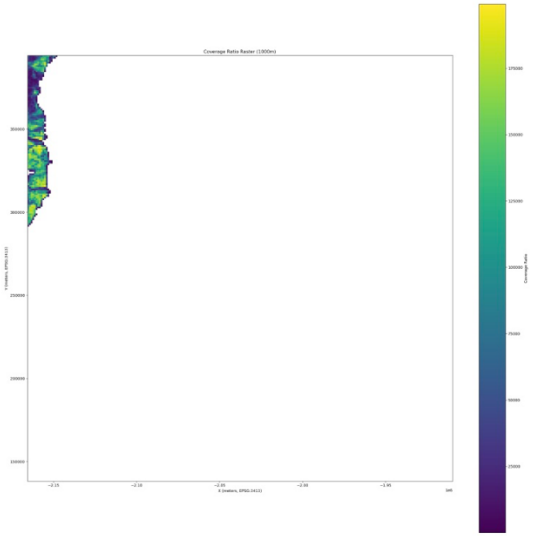
Coverage sum
Max: 700,000



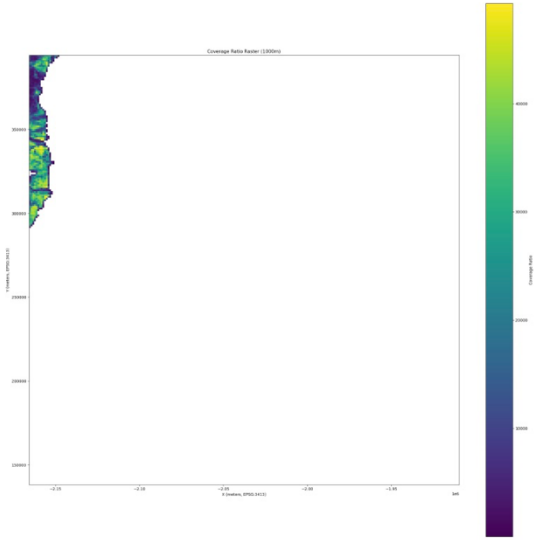
Length sum
Max: 60,000

Plot size: 25*25

IWP Count

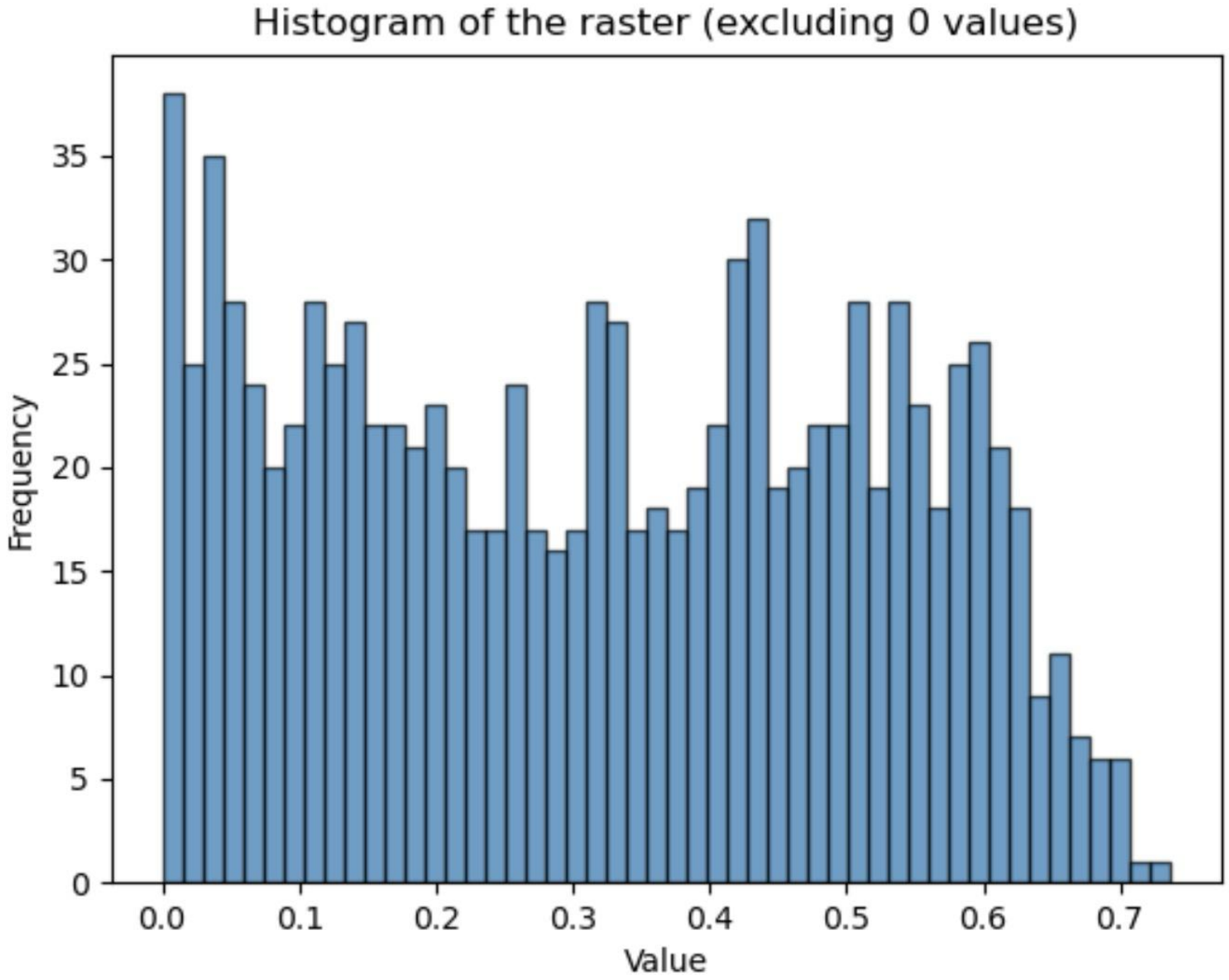


Perimeter sum
Max: 175,000

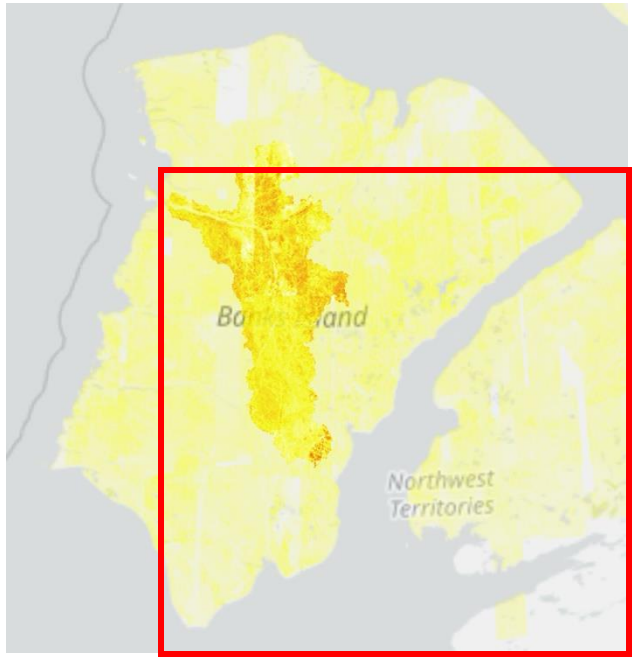


Width sum
Max: 40,000

Validation: value



Validation: value

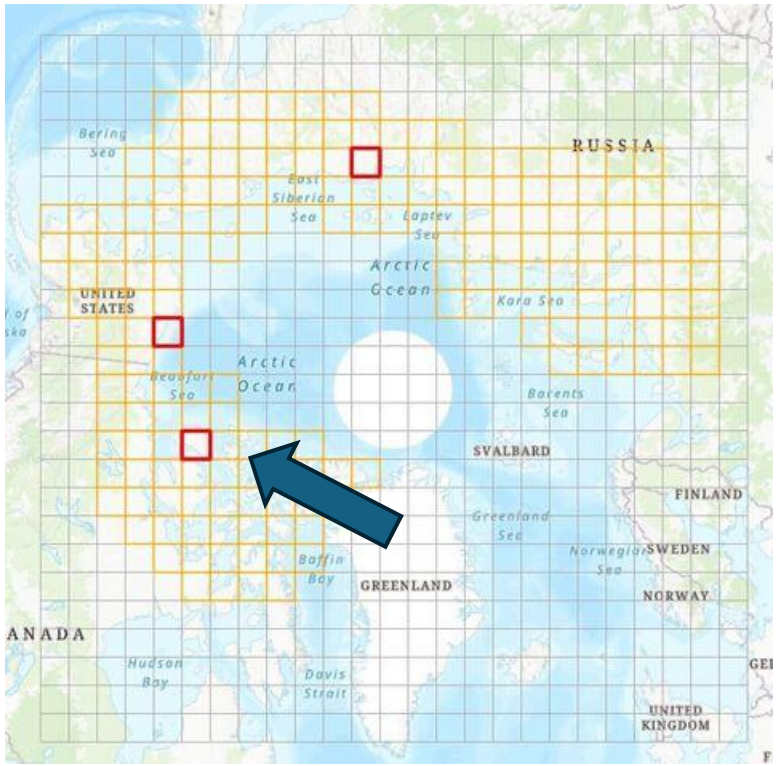


EPSG: 4326

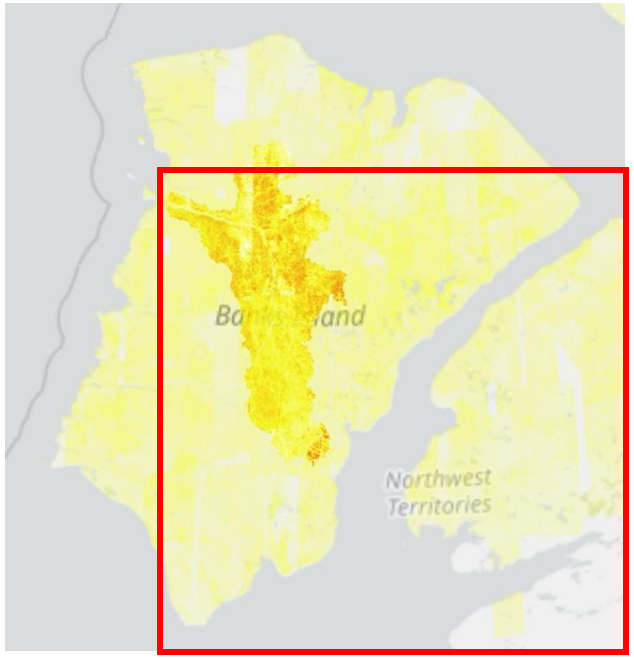
Remaining issues

~600,000 gpkg
~130~200G gpkg

1. Data storage for dense area



EPSG: 3413



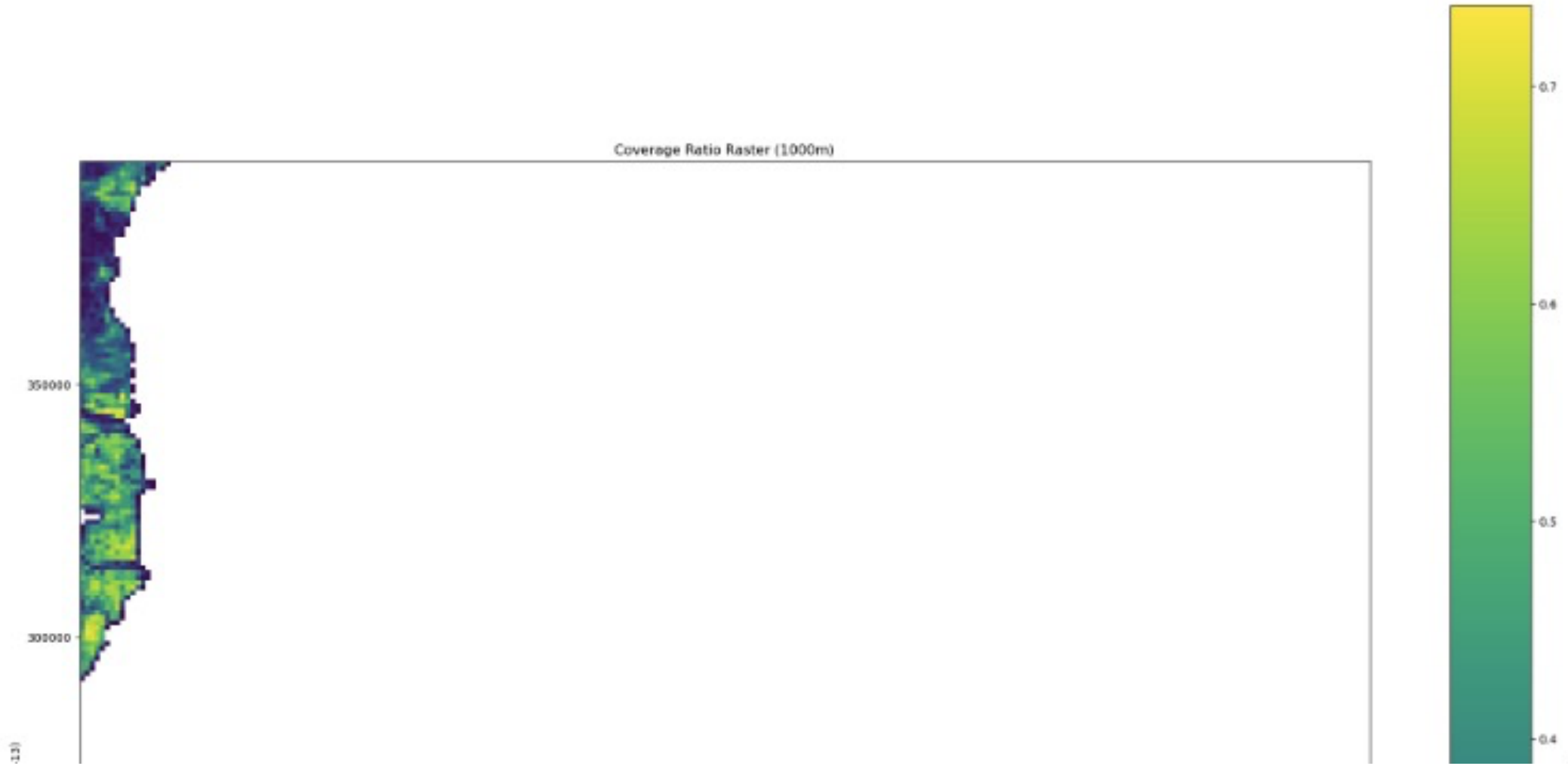
EPSG: 3413

EPSG: 4326

EPSG: 4326

Remaining issues

2. Consistent rendering maximum value



Meeting 11/5/2024

- Pick a grid (256*256km) generate coverage ratio/perimeter/length/width map
  - Validation
    1) Validation: BBOX : xmin, ymin, xmax, ymax

    2) Validation: GPKGs in database

Select 1km sub-grid (test)

Process validation

From 256*256 1km pixels, we will select the two pixels

In our workflow

Convert the 256km*256km grid into a 256*256 raster (with 1km as a pixel size)

256*256 1km pixels
Choose two of pixels

Query 1

Two pixels generated via ArcGIS tools

Code: to convert to query

Query 2

Compare query

get area sum in database

Overlapped GPKG in a pixel

get area sum in database

Compare data

Get area sum in ArcGIS

~708,460 m^2

BBOX coordinates check

Query 1: generated from our raster

(1, 13)

SELECT area, perimeter, length, width, geom_centroid FROM grid_gpkg_3413 polygons WHERE ST_within(polygons.geom_centroid, ST_GeomFromText('POLYGON((-2153757.915805478 393242.08426331764, -2153757.915805478 394242.08426331764, -2152757.915805478 394242.08426331764, -2152757.915805478 393242.08426331764, -2153757.915805478 393242.08426331764))', 3413)) AND staging_duplicated = false

→389180.4

(12, 4)

SELECT area, perimeter, length, width, geom_centroid FROM grid_gpkg_3413 polygons WHERE ST_within(polygons.geom_centroid, ST_GeomFromText('POLYGON((-2162757.9158000015 382242.0841999985, -2161757.9158000015 382242.0841999985, -2161757.9158000015 383242.0841999985, -2162757.9158000015 383242.0841999985, -2162757.9158000015 382242.0841999985))', 3413)) AND staging_duplicated = false

→319280.18

BBOX coordinates check

Query 2: generated from ArcGIS and converted via code

(1, 13)

SELECT area, perimeter, length, width, geom_centroid FROM grid_gpkg_3413 polygons WHERE ST_within(polygons.geom_centroid, ST_GeomFromText('POLYGON((-2153757.915805478 393242.08426331764, -2153757.915805478 394242.08426331764, -2152757.915805478 394242.08426331764, -2152757.915805478 393242.08426331764, -2153757.915805478 393242.08426331764))', 3413)) AND staging_duplicated = false

Number: 2785

→389180.408

(12, 4)

SELECT area, perimeter, length, width, geom_centroid FROM grid_gpkg_3413 polygons WHERE ST_within(polygons.geom_centroid, ST_GeomFromText('POLYGON((-2162757.915805478 382242.08426331764, -2162757.915805478 383242.08426331764, -2161757.915805478 383242.08426331764, -2161757.915805478 382242.08426331764, -2162757.915805478 382242.08426331764))', 3413)) AND staging_duplicated = false

Number: 1190

→319280.195

Total area sum:
389180.408+319280.195 = 708460.603

Data in database vs. geopackages data

Summarize: Center in

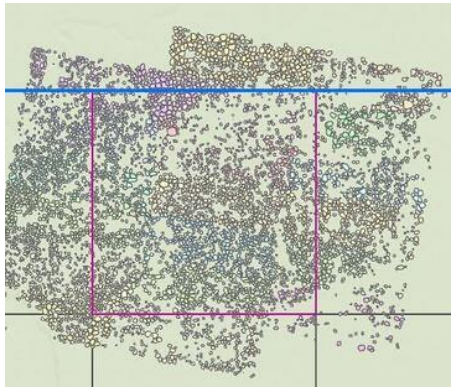
	G	H	I	J	K	L
1	Area	CentroidX	CentroidY	Perimeter	Length	Width
3955	214.572	-2161845.664	382414.794	65.013	19.589	14.234
3956	426.427	-2161900.34	382433.877	80.24	23.578	23.318
3957	139.862	-2161919.297	382436.698	46.198	15.687	11.582
3958	50.771	-2161867.973	382477.609	27.843	9.1	6.783
3959	103.209	-2161878.4	382481.406	41.173	13.143	9.514
3960	199.666	-2161874	382443.722	56.545	17.383	15.305
3961	319.623	-2161813.553	382441.414	67.553	21.407	18.992
3962	299.104	-2161785.53	382446.53	67.86	20.03	19.518
3963	200.192	-2161795.245	382431.044	55.033	16.133	15.263
3964	233.864	-2161792.776	382462.358	73.459	20.35	16.96
3965	212.293	-2161827.955	382427.084	57.983	19.057	14.508
3966	98.649	-2161798.697	382411.144	41.968	14.22	9.304
3967	156.698	-2161810.109	382417.886	51.354	16.96	12.295
3968	73.044	-2161824.257	382451.039	35.217	11.466	8.551
3969	78.831	-2161844.082	382459.405	34.676	10.599	9.514
3970	420.815	-2161842.919	382442.734	82.947	26.888	18.655
3971	87.25	-2161785.975	382405.293	36.73	10.341	10.175
3972	366.623	-2161873.119	382376.372	79.688	26.859	18.15
3973	481.845	-2161852.511	382368.733	86.491	29.98	20.651
3974	91.459	-2161828.946	382384.356	37.137	11.717	9.621
3975	205.102	-2161826.282	382369.619	55.378	16.54	15.324
3976	67.607	-2161842.797	382380.366	31.234	9.751	8.687
3977	708460.603			204080.118	64540.57	50296.361
3978						

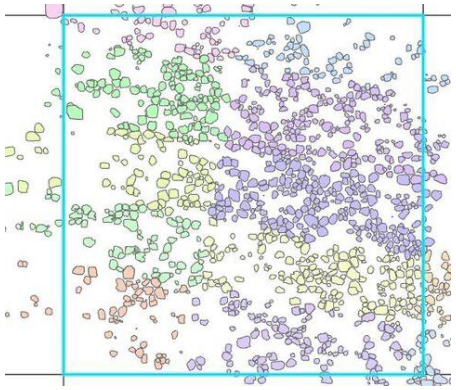
The previous results:
708460.603

127953.55
+76126.57
=204080.12

40786.008
+ 23754.56
=64540.568

31559.664
+18736.695
=50296.359

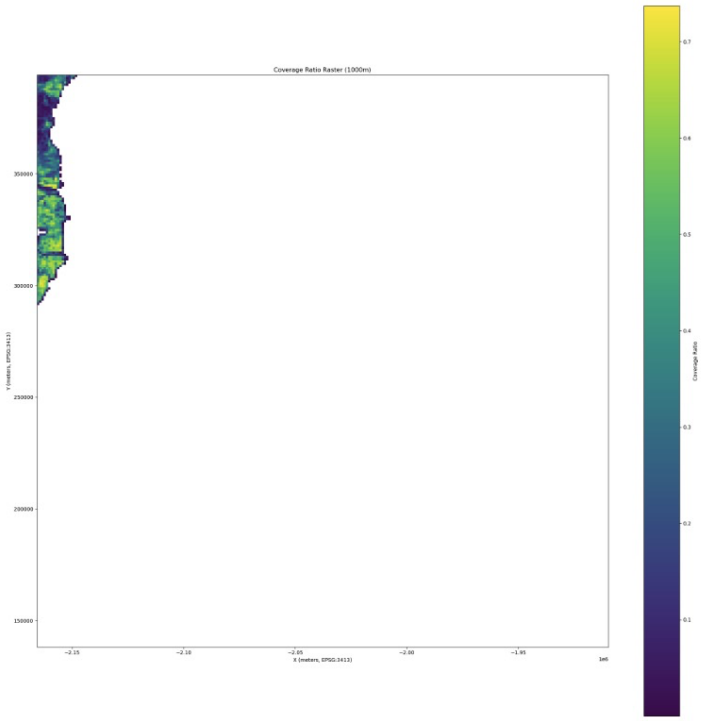


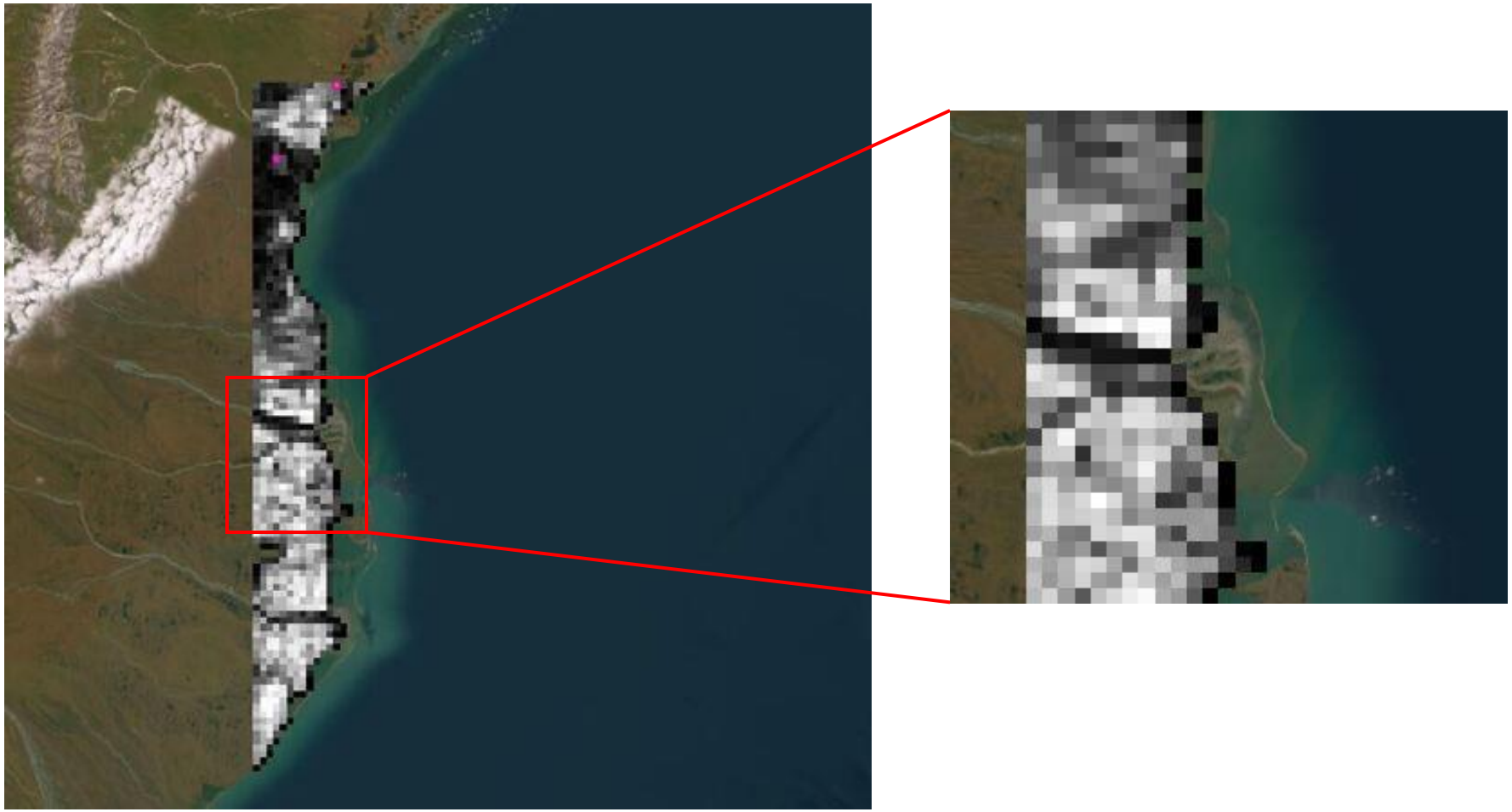


Meeting 11/12/2024

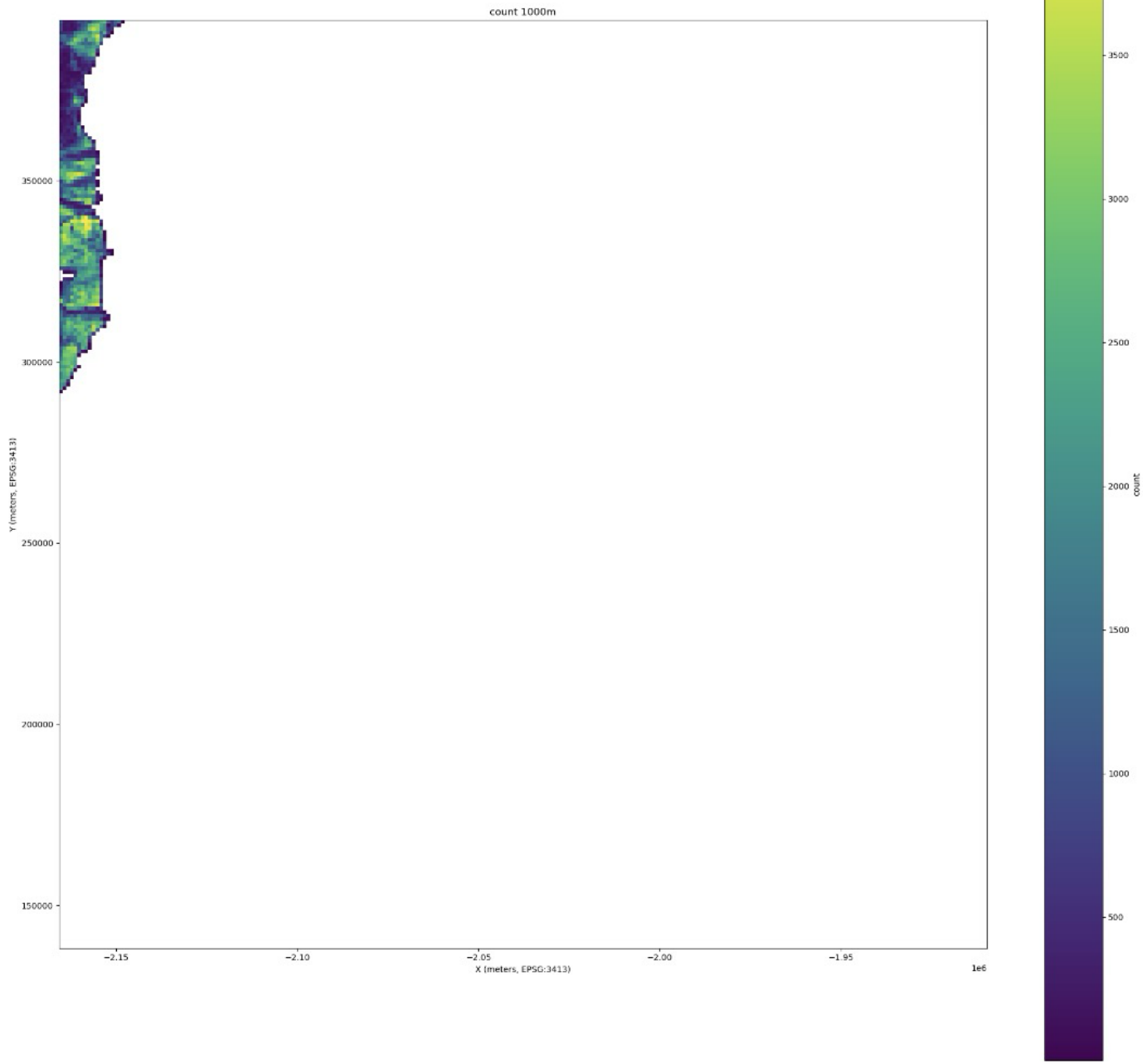
- Validate: Overlay with Base map
- Count of IWPs at each pixel
- Trough Width (Sum/Average)

Validate with basemap





Count of IWP map

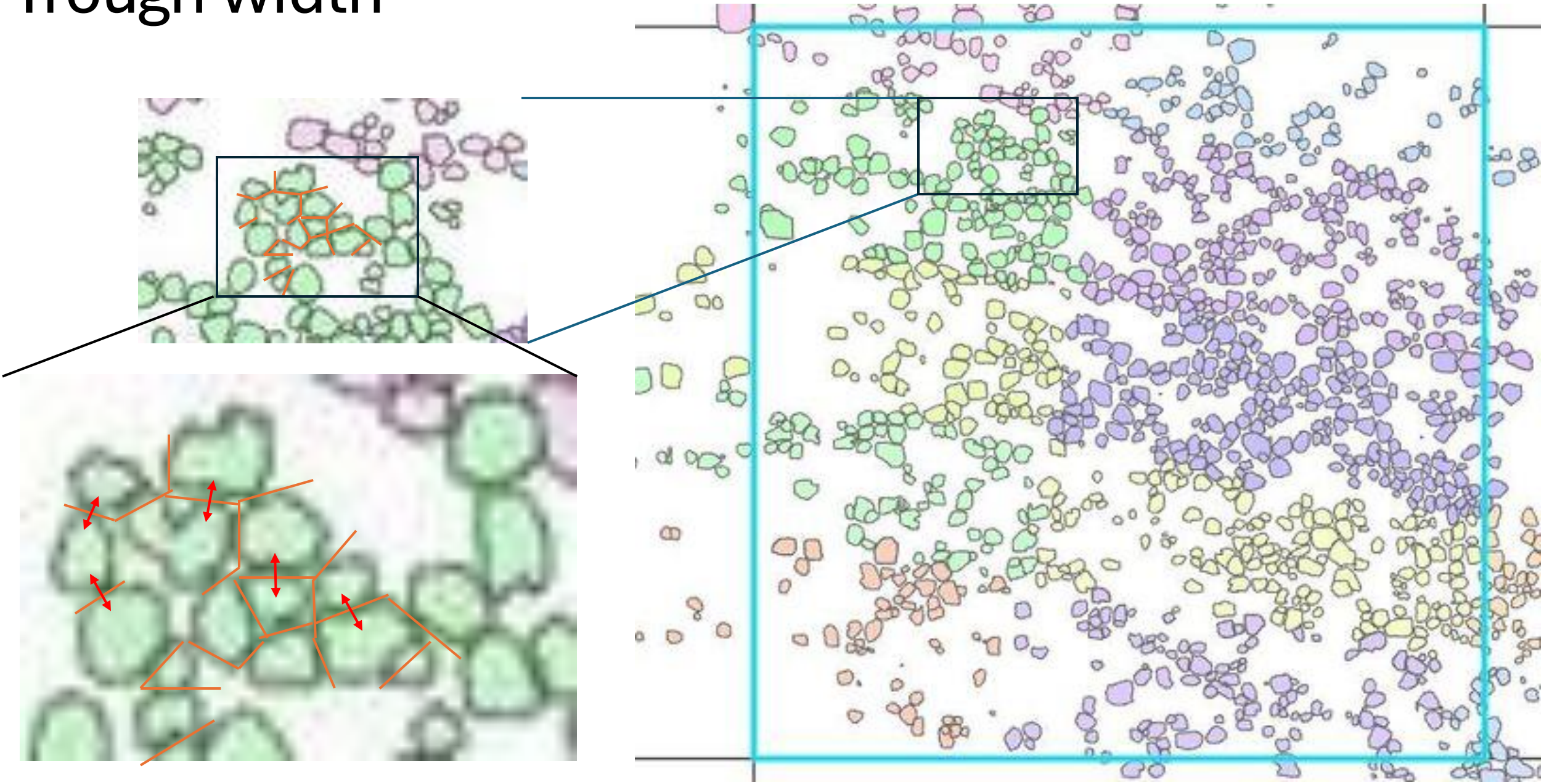


Trough width calculation

Centerline of IWPs → Remove duplicated centerlines if they overlapped with footprints → Calculate the entire length of centerline

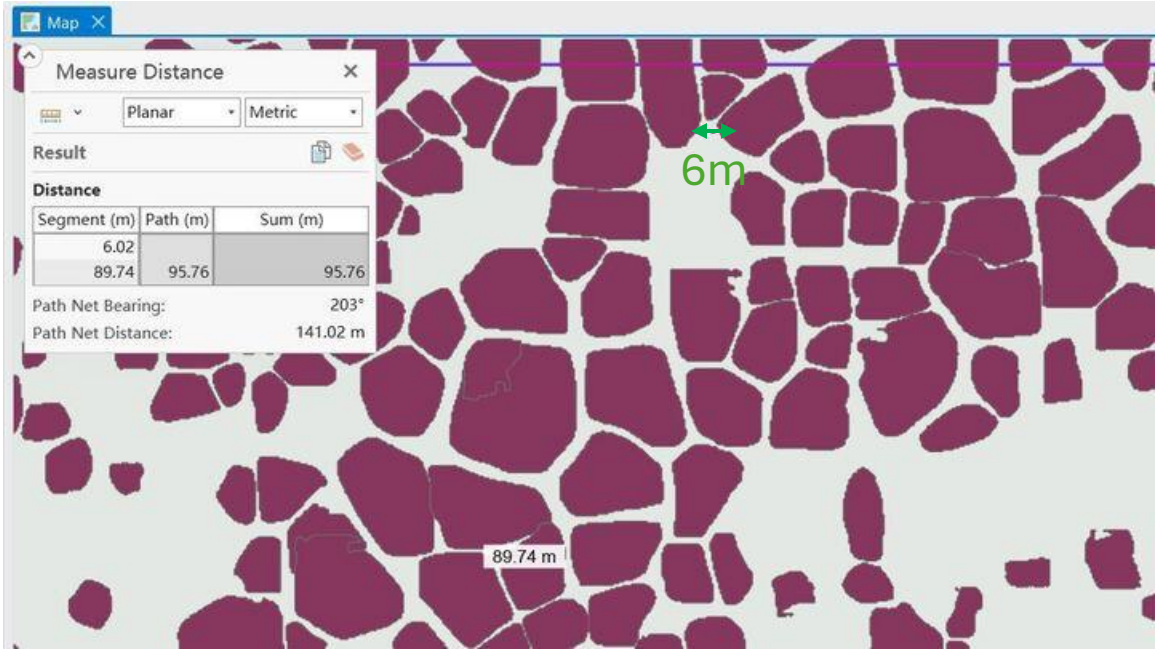
Footprints

Trough width

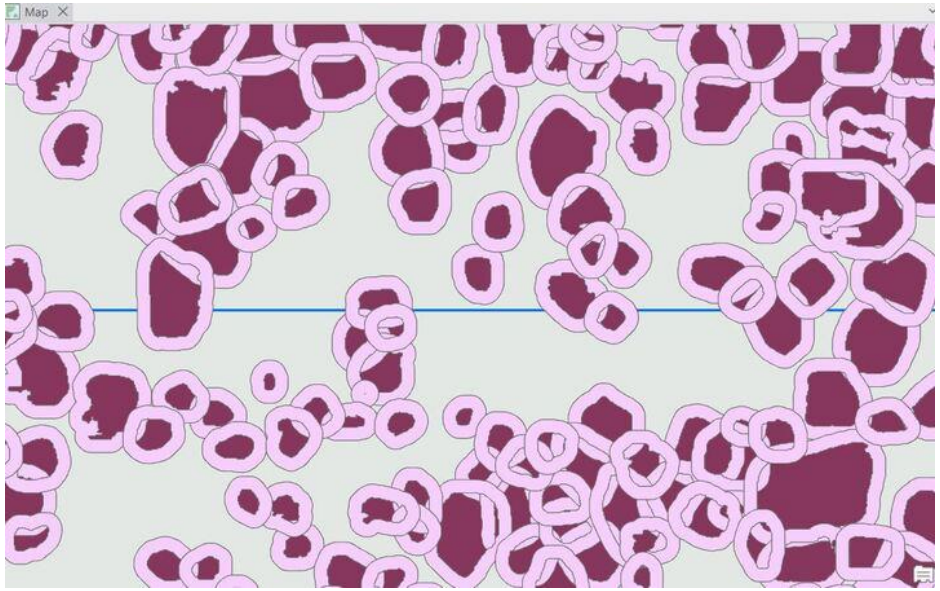


Centerline of IWPs → Remove duplicated centerlines if they overlapped with footprints → Calculate the entire length of centerline

1) Create 5m buffer
2) Convert buffer into raster
3) Skeletonize the raster
4) Convert the skeletonized raster to polyline → Centerline



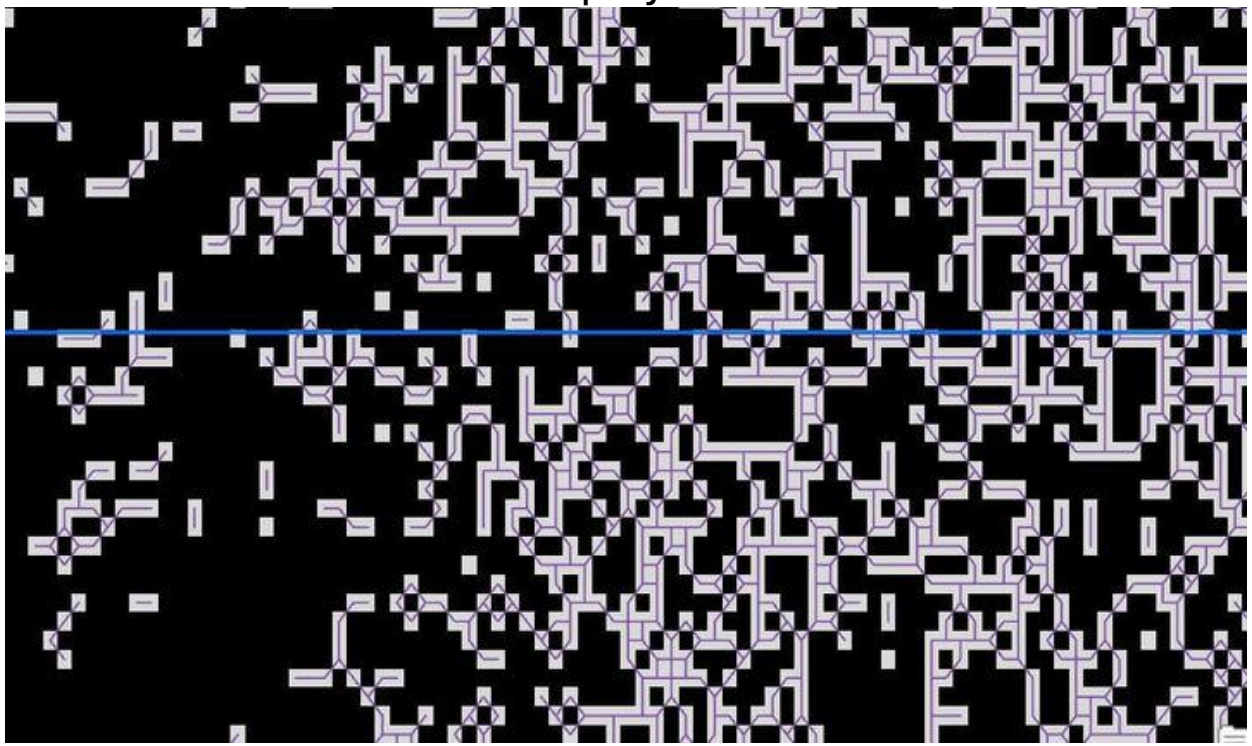
Buffer



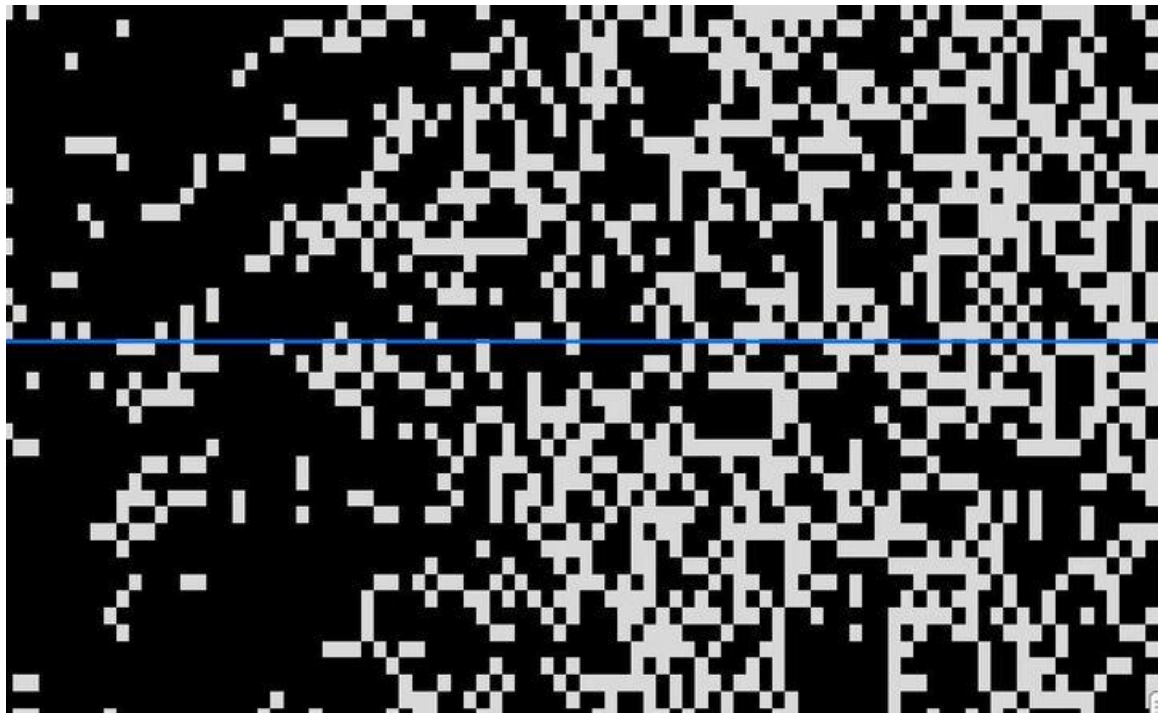
Centerline of IWPs → Remove duplicated centerlines if they overlapped with footprints → Calculate the entire length of centerline

1) Create 5m buffer
2) Convert buffer into raster
3) Skeletonize the raster
4) Convert the skeletonized raster to polyline → Centerline

Convert to polylines

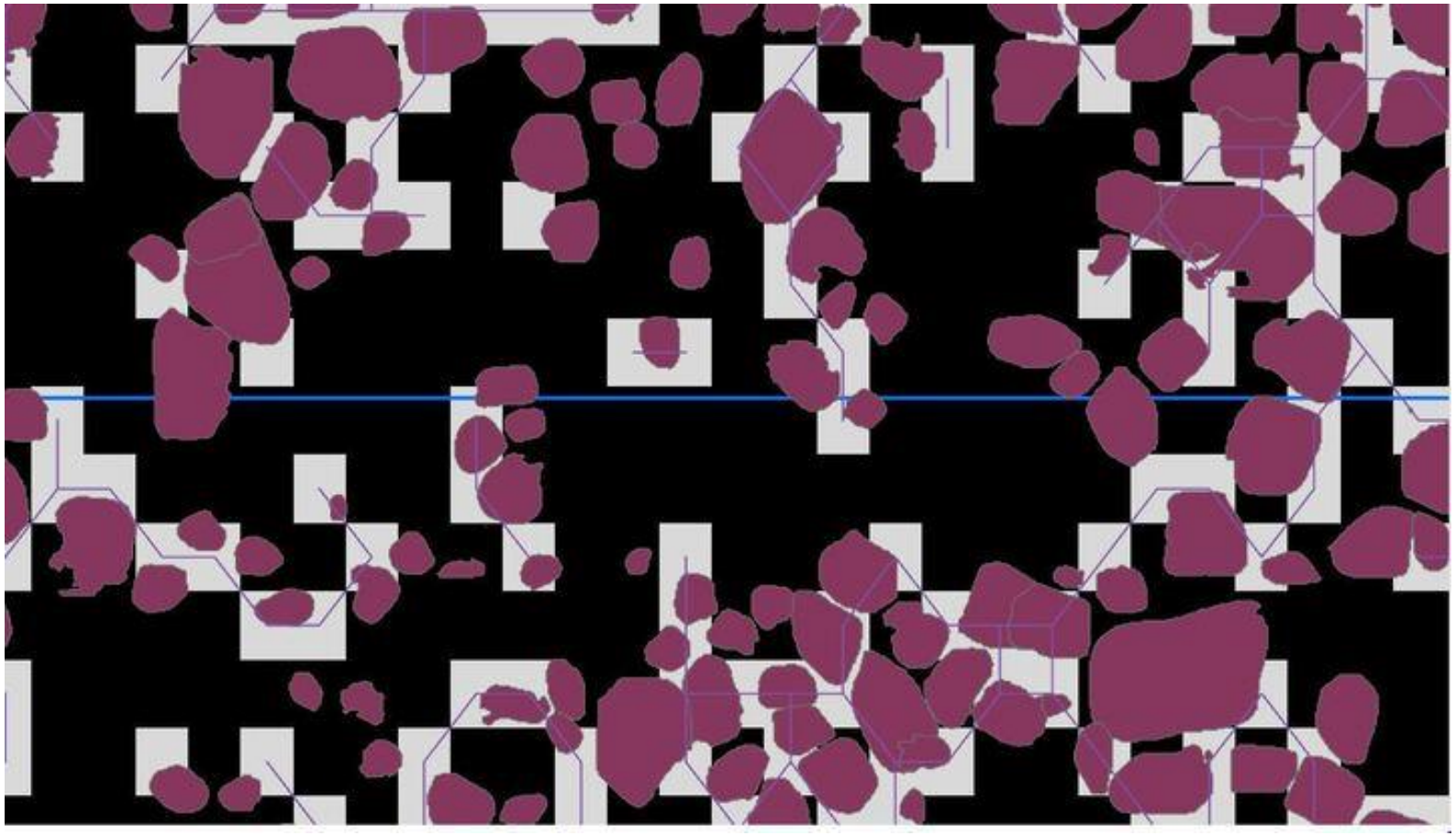


Skeletonized buffer raster



Centerline of IWPs → Remove duplicated centerlines if they overlapped with footprints → Calculate the entire length of centerline

1) Create 5m buffer
2) Convert buffer into raster
3) Skeletonize the raster
4) Convert the skeletonized raster to polyline → Centerline

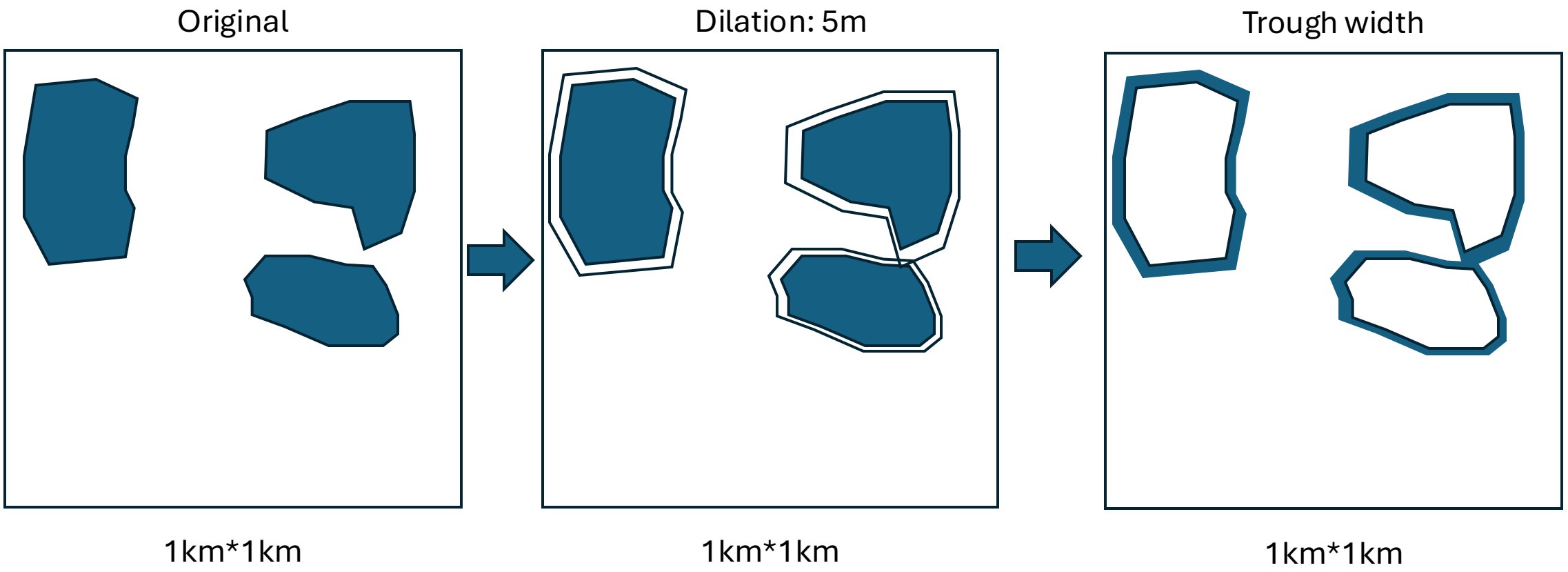


Meeting 11/19/2024

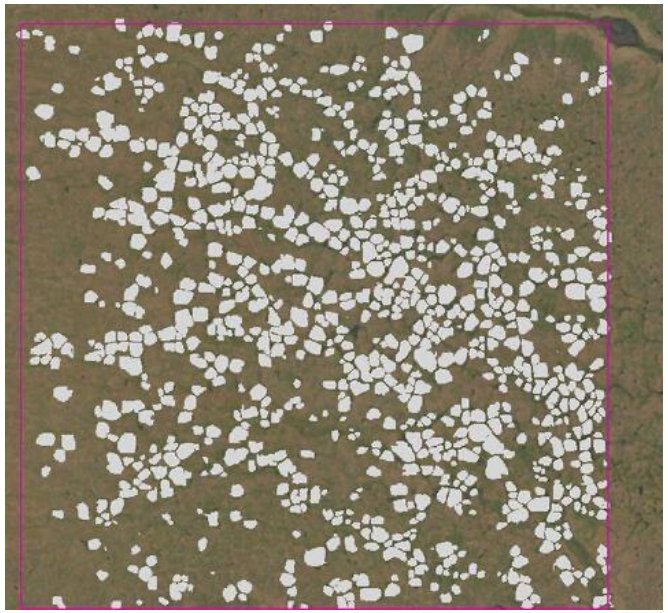
- Trough width
  - Dilation-> exclude 0 values area (cv approach)
  - Other area -> subtract IWP raster → Through area?

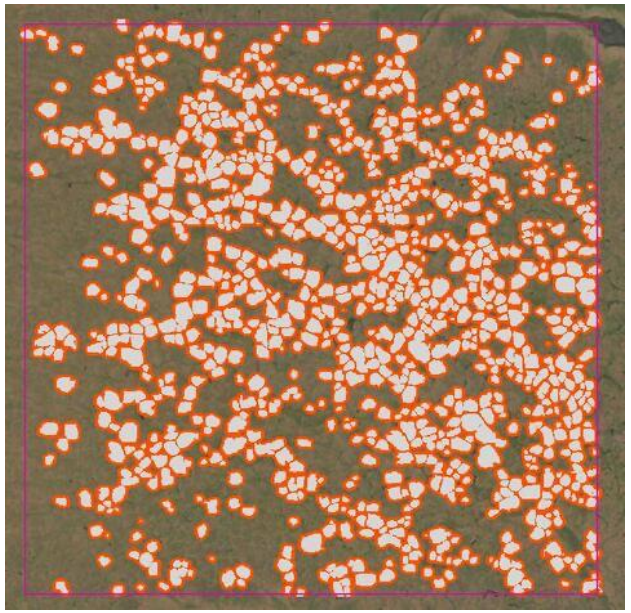
- Generate statistics maps on Northern Alaska

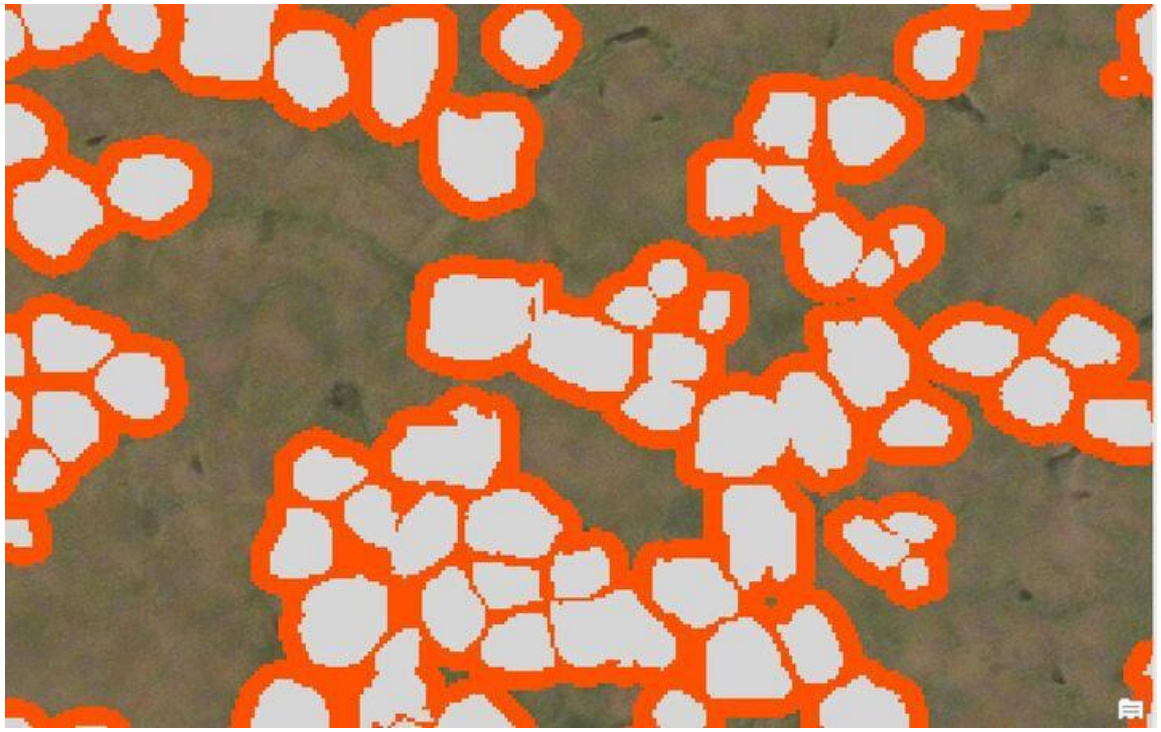
Trough width



Rasterize the dilated region (5m)

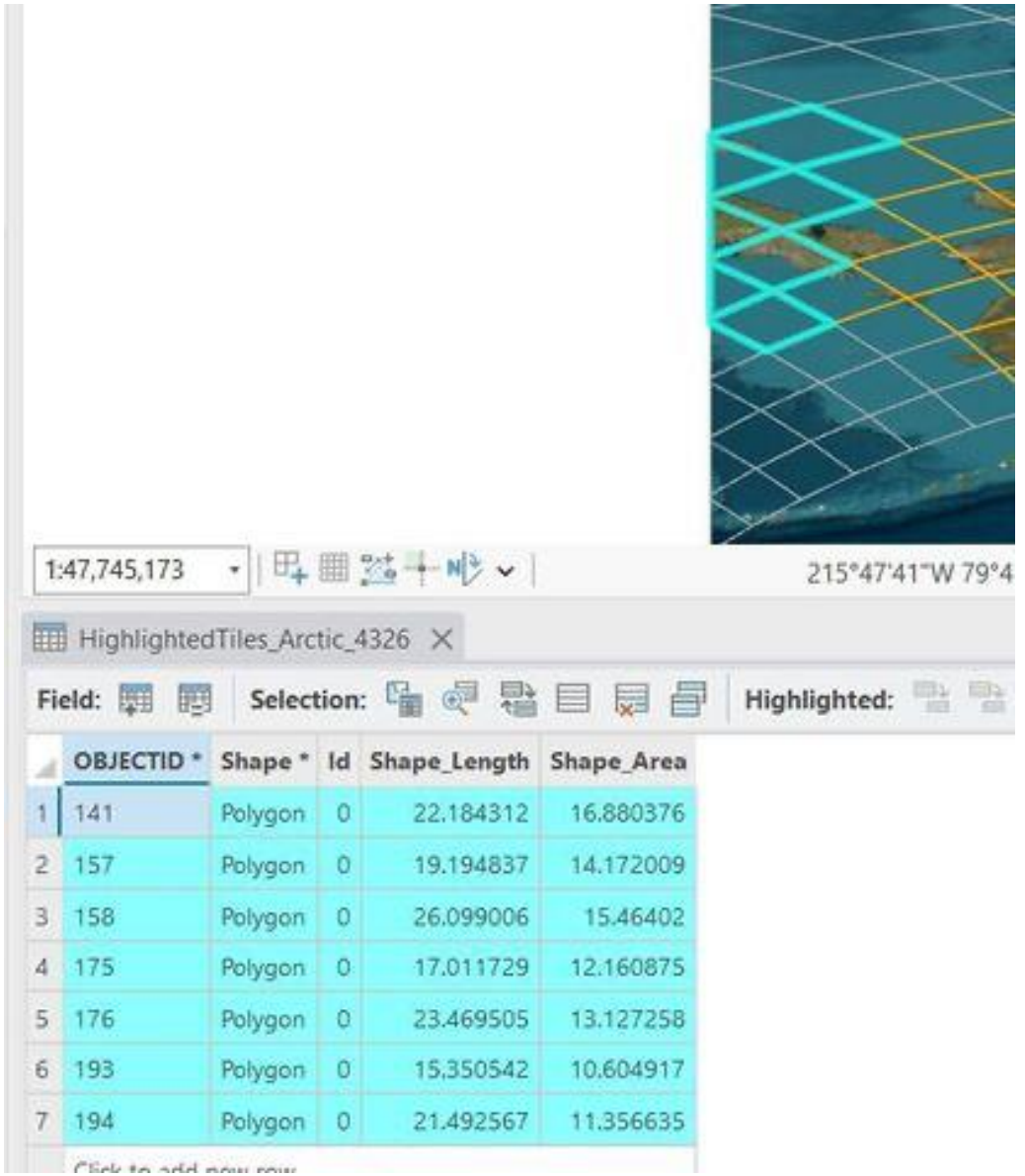






3*3 windows -> larger windows

Apply to the Pan-Arctic Region



	OBJECTID *	Shape *	Id	Shape_Length	Shape_Area
1	141	Polygon	0	22.184312	16.880376
2	157	Polygon	0	19.194837	14.172009
3	158	Polygon	0	26.099006	15.46402
4	175	Polygon	0	17.011729	12.160875
5	176	Polygon	0	23.469505	13.127258
6	193	Polygon	0	15.350542	10.604917
7	194	Polygon	0	21.492567	11.356635

Apply to the Pan-Arctic Region

- Debugging

```
/home/xchen/data/geopackage/Grid/: Scheme missing.
--2024-11-19 00:03:00--  https://arcticdata.io/data/10.18739/A2KW57K57/iwp_geopackage_high/WGS1984Quad/15/16385/3886.gpkg
/home/xchen/data/geopackage/Grid/: Scheme missing.
--2024-11-19 00:03:00--  https://arcticdata.io/data/10.18739/A2KW57K57/iwp_geopackage_high/WGS1984Quad/15/16380/3887.gpkg
/home/xchen/data/geopackage/Grid/: Scheme missing.
/home/xchen/data/geopackage/Grid/: Scheme missing.
```

```
--2024-11-19 00:03:00--  https://arcticdata.io/data/10.18739/A2KW57K57/iwp_geopackage_high/WGS1984Quad/15/16390/3890.gpkg
Resolving arcticdata.io (arcticdata.io)... 128.111.85.224
Connecting to arcticdata.io (arcticdata.io)|128.111.85.224|:443... 128.111.85.224
Resolving arcticdata.io (arcticdata.io)... Connecting to arcticdata.io (arcticdata.io)|128.111.85.224|:443... 128.111.85.224
Connecting to arcticdata.io (arcticdata.io)|128.111.85.224|:443... connected.
Resolving arcticdata.io (arcticdata.io)... Resolving arcticdata.io (arcticdata.io)... 128.111.85.224
128.111.85.224
Connecting to arcticdata.io (arcticdata.io)|128.111.85.224|:443... 128.111.85.224
Connecting to arcticdata.io (arcticdata.io)|128.111.85.224|:443... Resolving arcticdata.io (arcticdata.io)... Connecting to arcticdata.io (arcticdata.io)|128.111.85.224|:443... 128.111.85.224
Connecting to arcticdata.io (arcticdata.io)|128.111.85.224|:443... Resolving arcticdata.io (arcticdata.io)... Resolving arcticdata.io (arcticdata.io)... /home/xchen/data/geopackage/Grid/: Scheme missing.
```

```
failed: Connection refused.
Connecting to arcticdata.io (arcticdata.io)|64:ff9b::806f:55e0|:443... failed: Network is unreachable.
failed: Connection refused.
Connecting to arcticdata.io (arcticdata.io)|64:ff9b::806f:55e0|:443... failed: Network is unreachable.
failed: Connection refused.
Connecting to arcticdata.io (arcticdata.io)|64:ff9b::806f:55e0|:443... failed: Network is unreachable.
failed: Connection refused.
Connecting to arcticdata.io (arcticdata.io)|64:ff9b::806f:55e0|:443... failed: Network is unreachable.
```

Meeting 11/19/2024

- Apply to Pan-Arctic area (except for 7 grids)
- Trough length -> Trough width

Apply to Pan-Arctic (except for 7 grids)

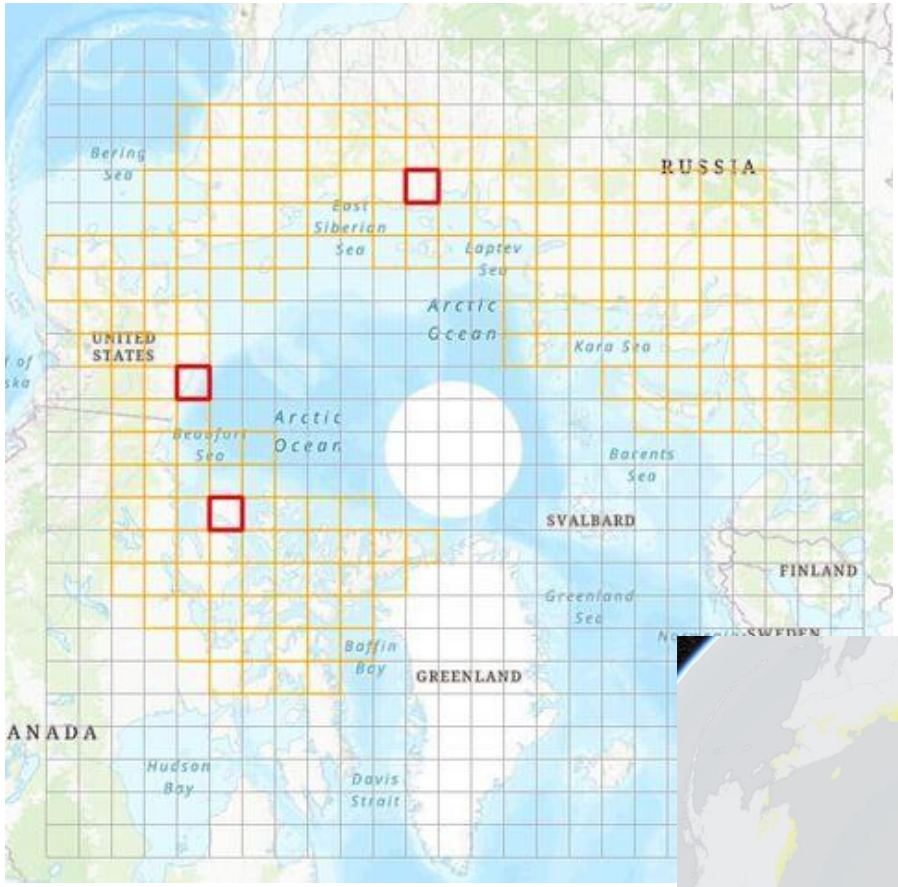
➔12/3 debug

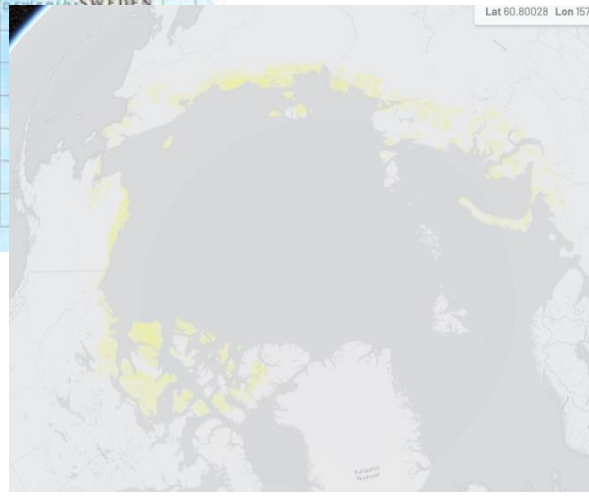
➔12/17 progress check

Total: 230 grids

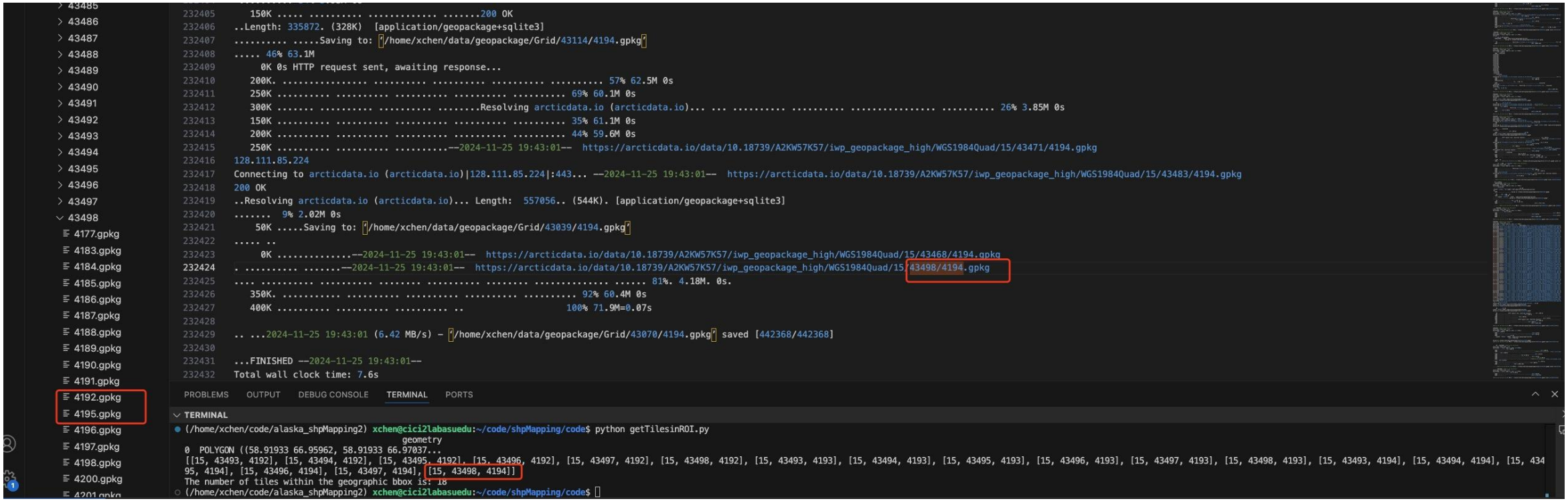
130 grids with data -> 5h -> 25d

uniqueID: XY (4 digit)





Validate on another grid



Trough length and width

→12/3 Test on one grid

Meeting 11/19/2024

- Debug
  - Data installation: Uninterrupted & Validate

Data installation

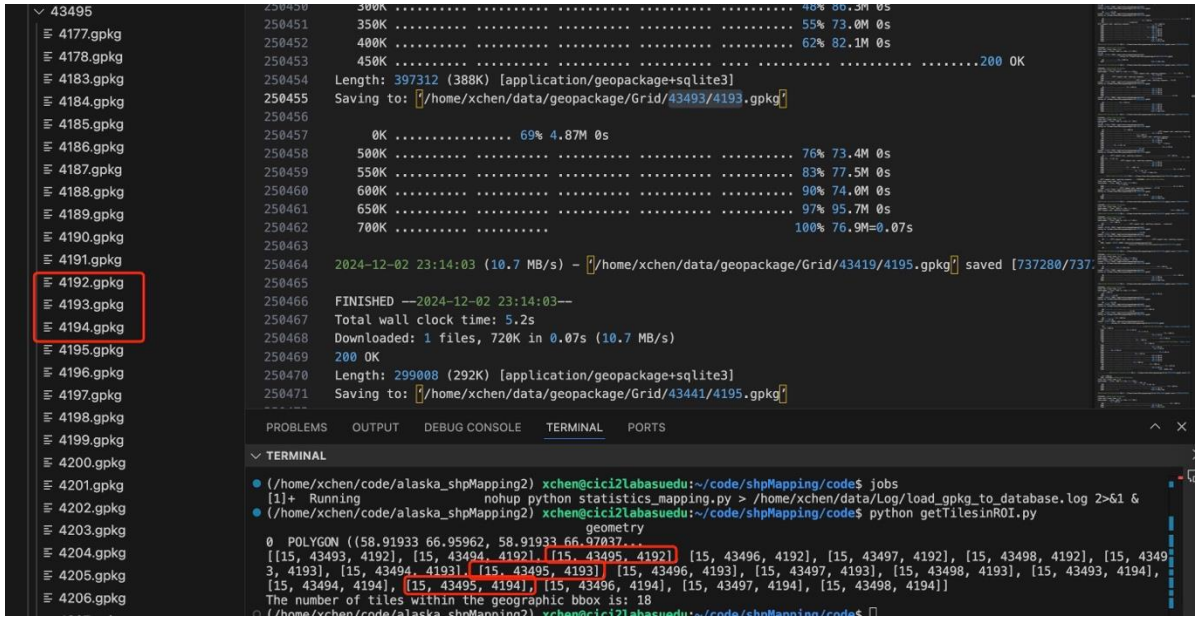
- Uninterrupted
  - Retry up to 30 times
  - Timeout: 120 seconds

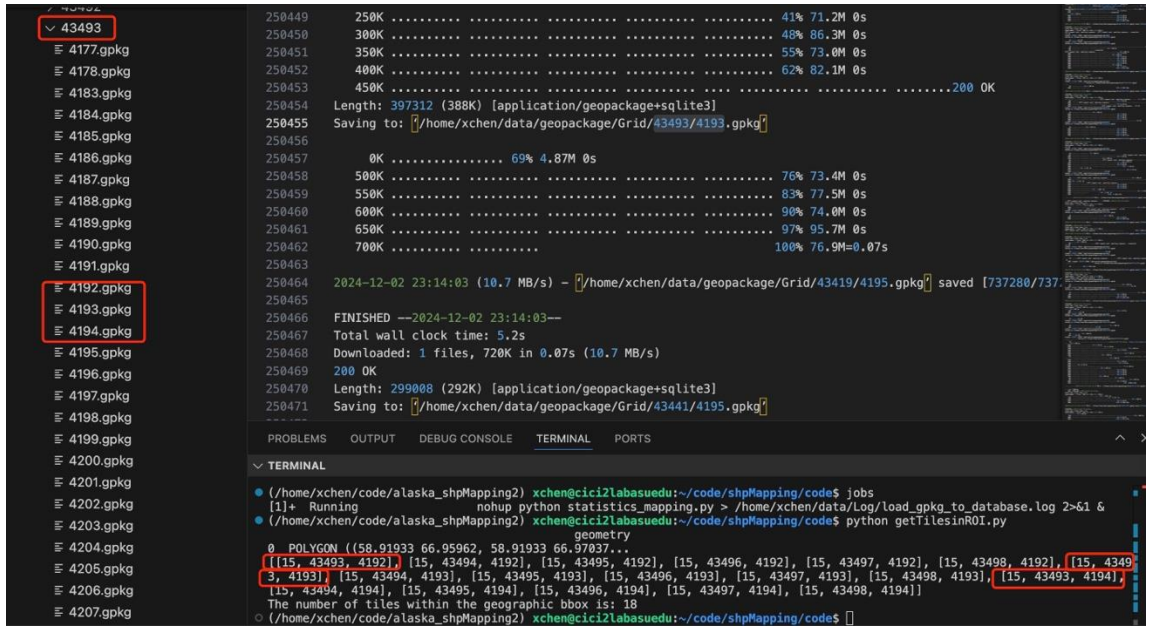
➢No fetch failures due to various network connection issues (Connection, timeout, SSL, etc.)

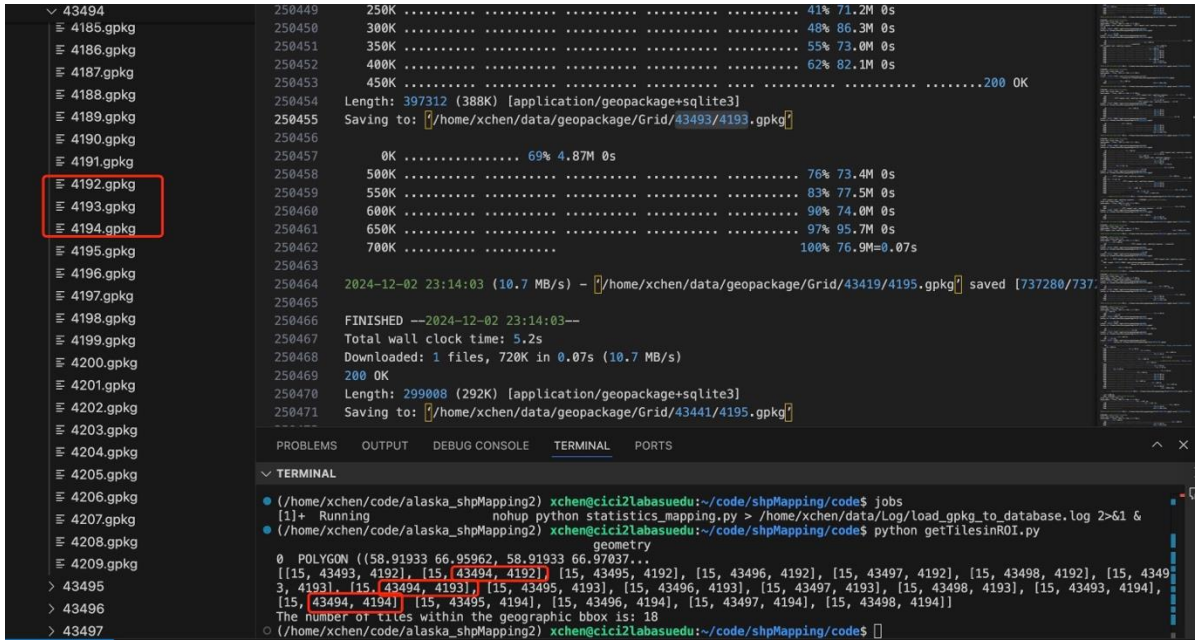
➢Total runtime for this grid: 5719 seconds (1.5h)

Data installation

- Validation
  - Randomly select a pixel: Compare the tiles within those pixels with installed GPKG



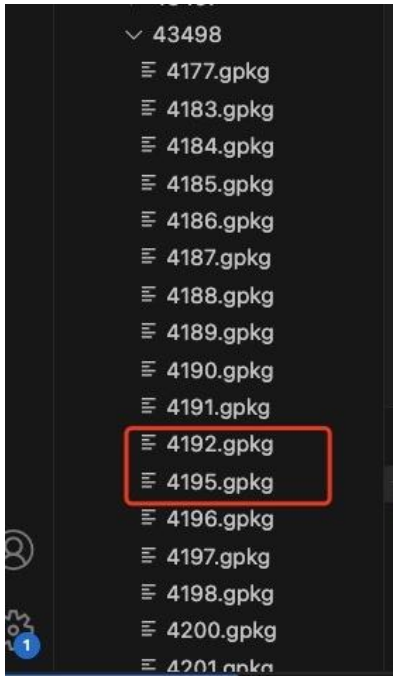




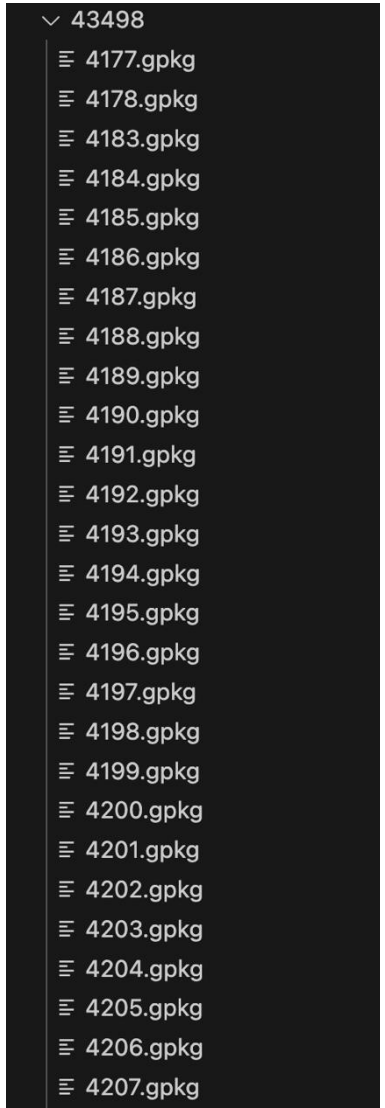
Data installation

- Validation
  - Randomly selected a Tile X: Check continuous tiles

Previous



Current



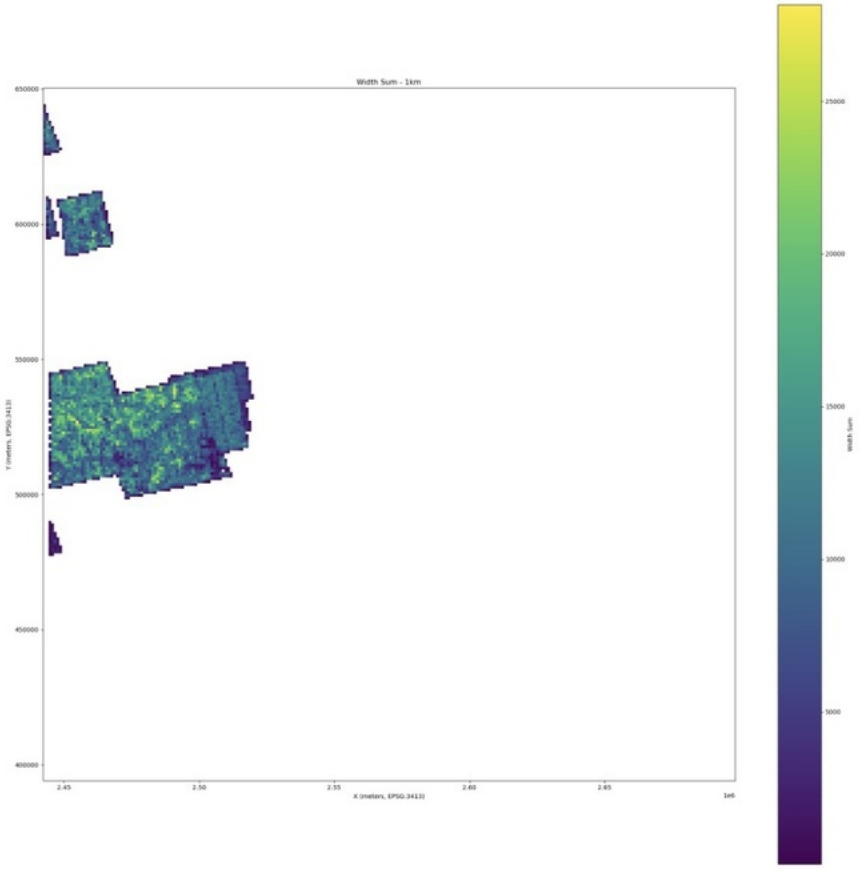
Data installation

- Validation
  - Randomly selected a Tile X: Check continuous tiles

Previous		Current
18,826 GPKG		22,157 GPKG

Total time in processing 1 grid

- Data installation: 1.5h
- Data loading into database: 1.5h
- Ploting data: 0.5~1h



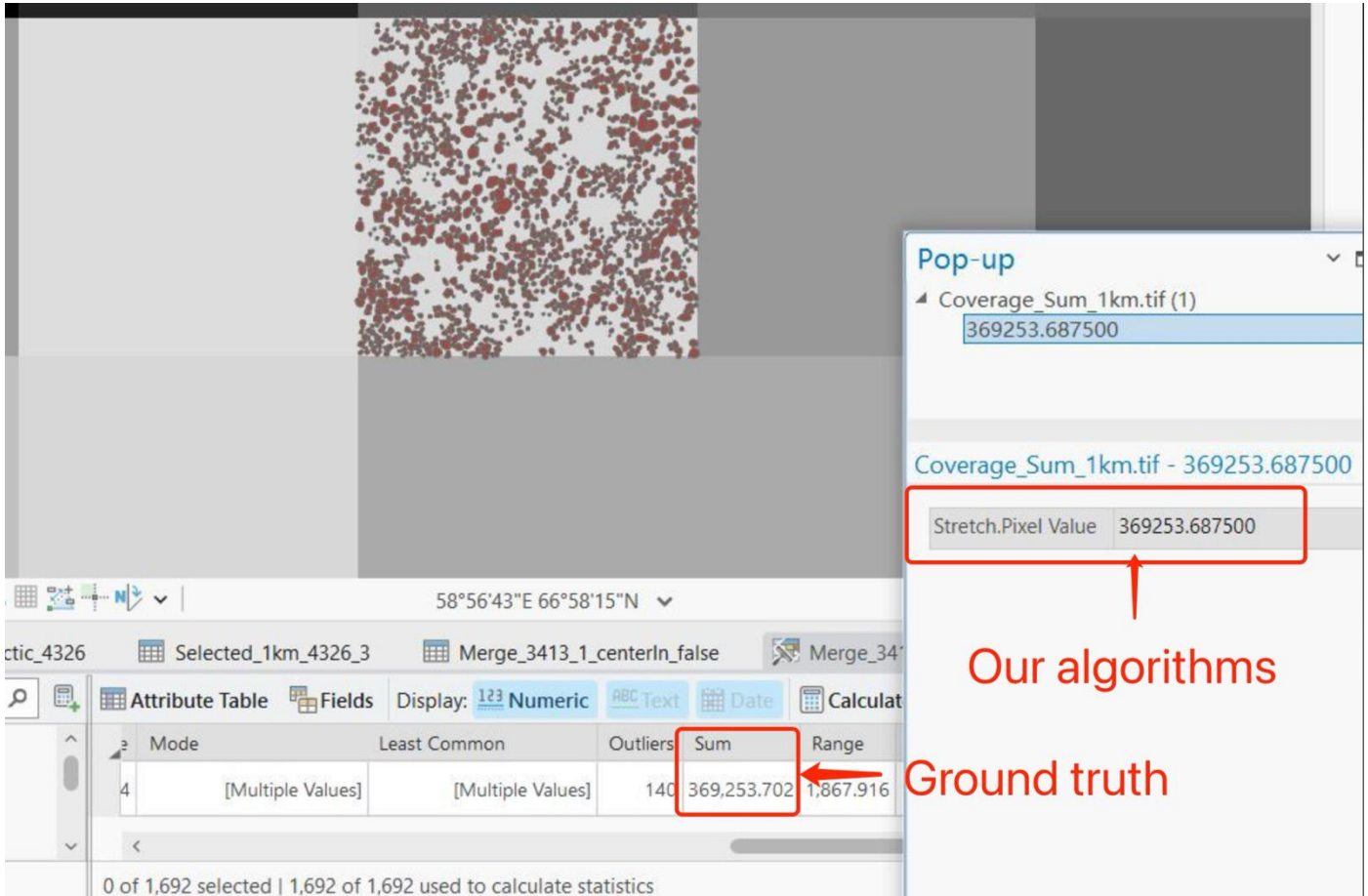
Meeting 12/10/2024

- Validation
  - Values within a pixel
  - Automatic and complete program

Validate

```
TERMINAL
(/home/xchen/code/alaska_shpMapping2) xchen@cici2labasuedu:~/code/shpMapping/code$ python getTilesinROI.py
                                        geometry
0  POLYGON ((58.91933 66.95962, 58.91933 66.97037...
[[15, 43493, 4192], [15, 43494, 4192], [15, 43495, 4192], [15, 43496, 4192], [15, 43497, 4192], [15, 43498, 4192], [15, 43493, 4193], [15, 43494, 4193], [15, 43495, 4193], [15, 43496, 4193], [15, 43497, 4193], [15, 43498, 4193], [15, 43493, 4194], [15, 43494, 4194], [15, 434
95, 4194], [15, 43496, 4194], [15, 43497, 4194], [15, 43498, 4194]]
The number of tiles within the geographic bbox is: 18
(/home/xchen/code/alaska_shpMapping2) xchen@cici2labasuedu:~/code/shpMapping/code$
```

1. Select one pixel
2. Get ground truth (coverage sum in that pixel)
3. Compare our values from the algorithms <-> Ground truth



Validate

Automatic and complete program

```
----------Processing Grid 1.....----------
Grid 1: 1.shp
The bounds of the grid 1 is: [-94.10902193  65.43618401 -85.89097807  68.65069617]
```

```
----------Processing Grid 10.....----------
Grid 10: 10.shp
The bounds of the grid 10 is: [-77.36239785  72.80682717 -65.84704217  75.91622142]
```

```
----------Processing Grid 100.....----------
Grid 100: 100.shp
The bounds of the grid 100 is: [53.31270628 64.77761297 59.90901987 67.44712869]
```

```
----------Processing Grid 101.....----------
Grid 101: 101.shp
The bounds of the grid 101 is: [52.60118202 62.58967968 58.54920873 65.20596327]
```

```
----------Processing Grid 102.....----------
Grid 102: 102.shp
The bounds of the grid 102 is: [52.0012997  60.41250662 57.41313139 62.9787845 ]
```

```
----------Processing Grid 103.....----------
Grid 103: 103.shp
The bounds of the grid 103 is: [-153.69752416   62.19197172 -147.49708883   64.95302996]
```

Meeting 12/17/2024

- Mark those grids that have been processed
  - Visualize several grids

- Trough length
  - At pixel level
- Trough width
  - Find clusters at pixel level

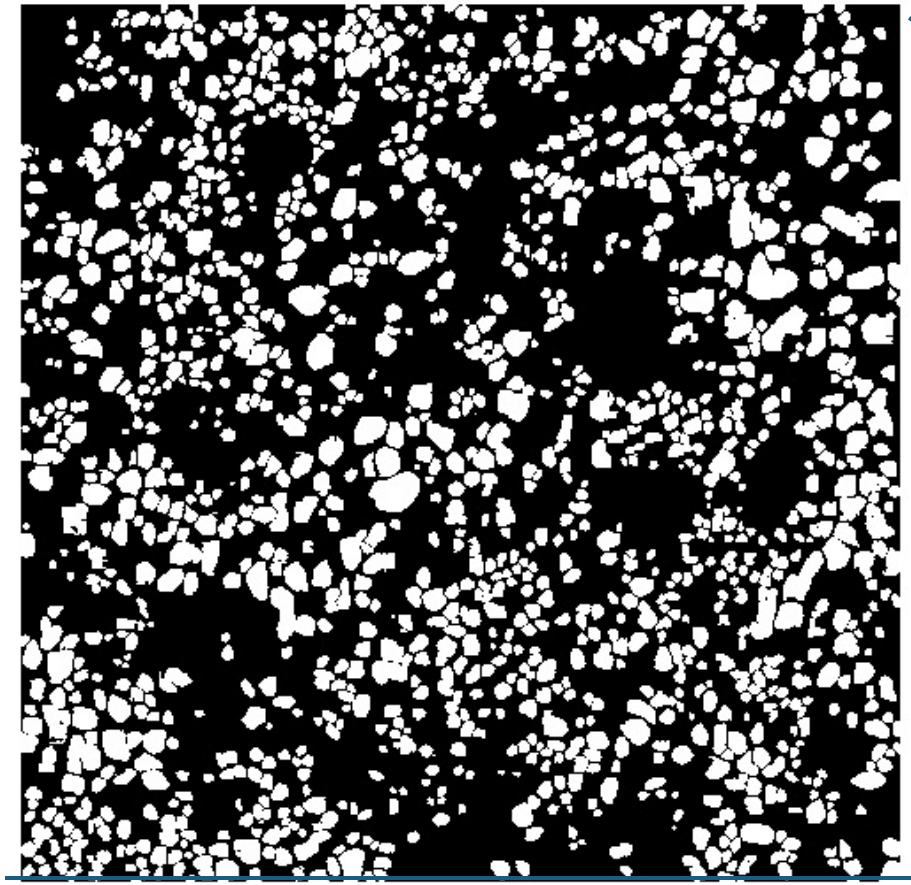
Trough length

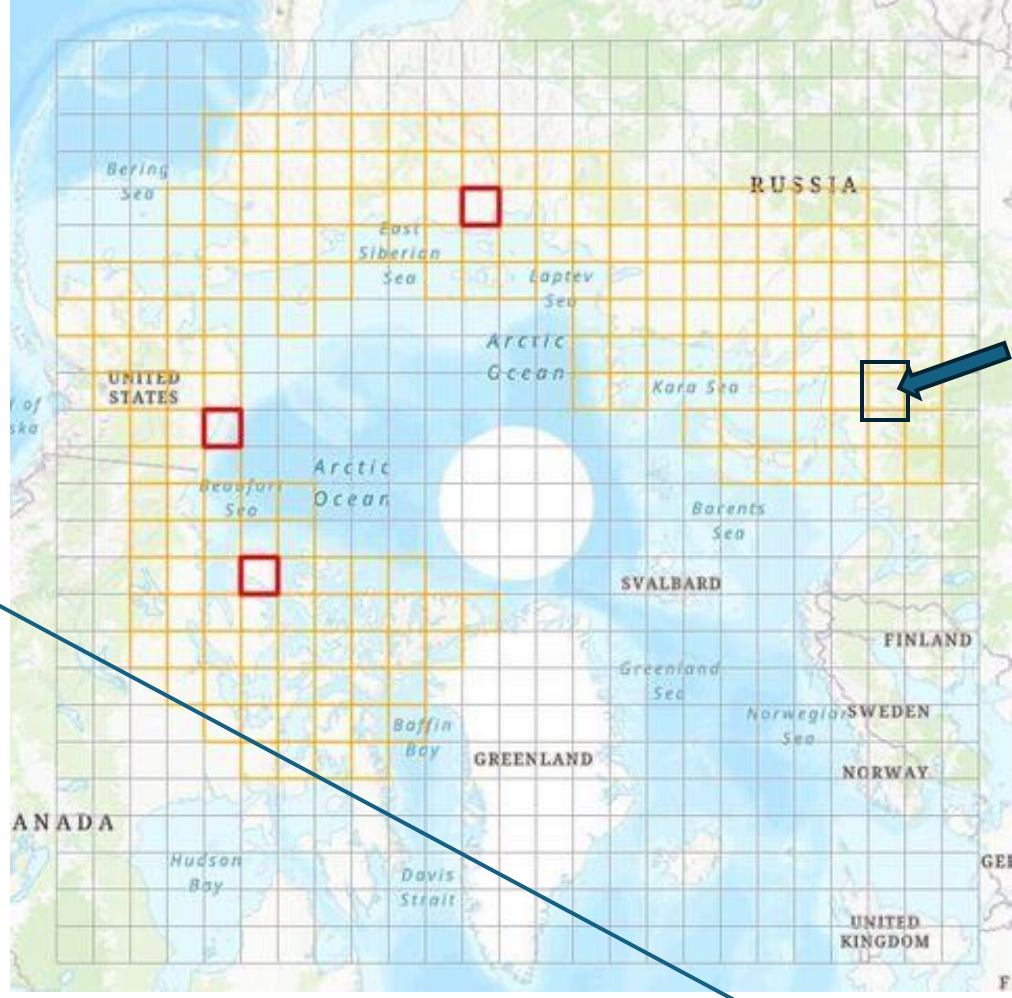
Test on 1 pixel within a grid

Width > IWP gap

Multiple Subset >5-20

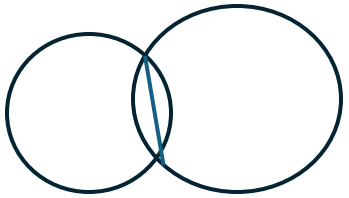
Centroid distance -> Intersect with polygons





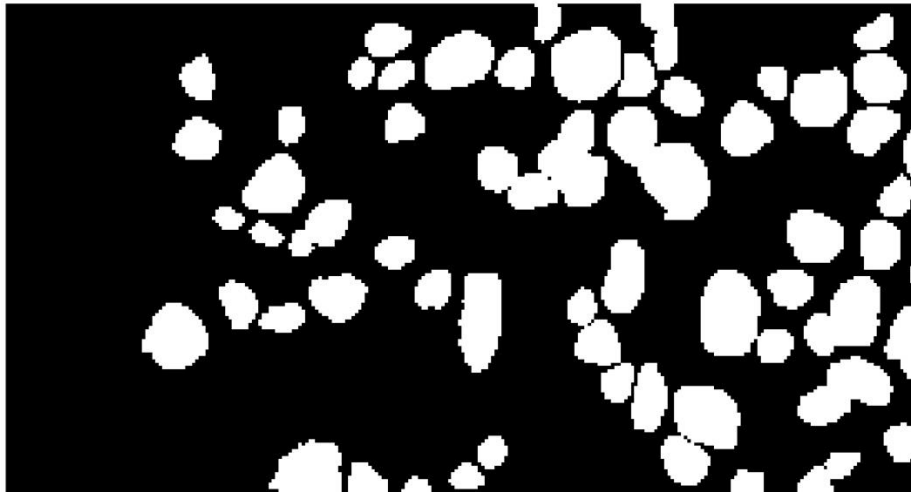
GDAL

Trough length



Test on 1 pixel within a grid

Original raster

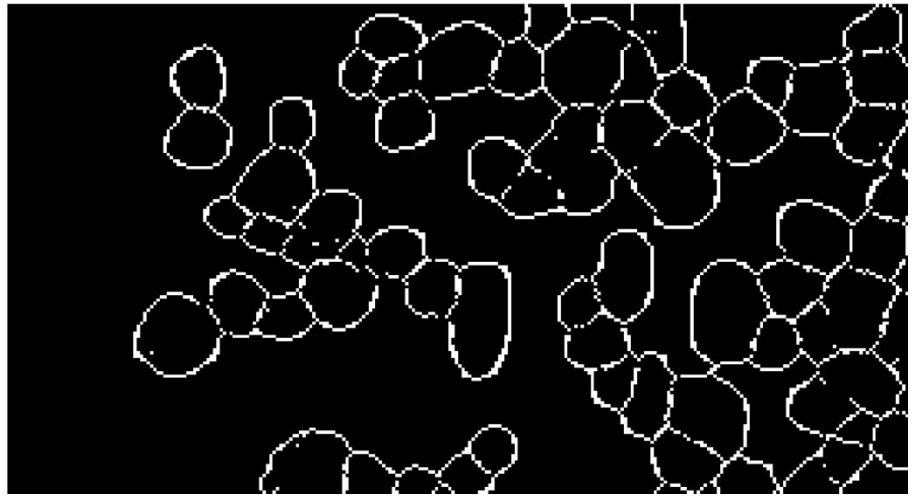


Buffer
-> dilate

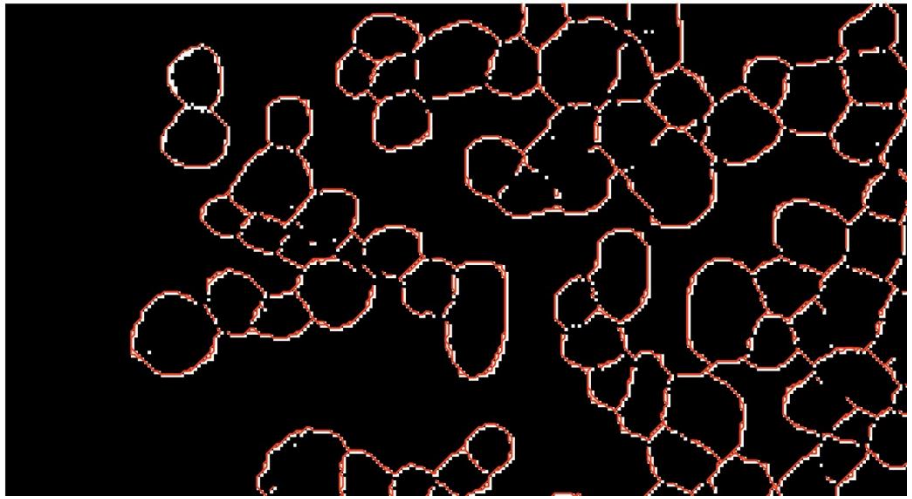
Difference
= Dilated raster – original raster

Skeletonized Difference → Transformation → Converted Polylines

Not consistent



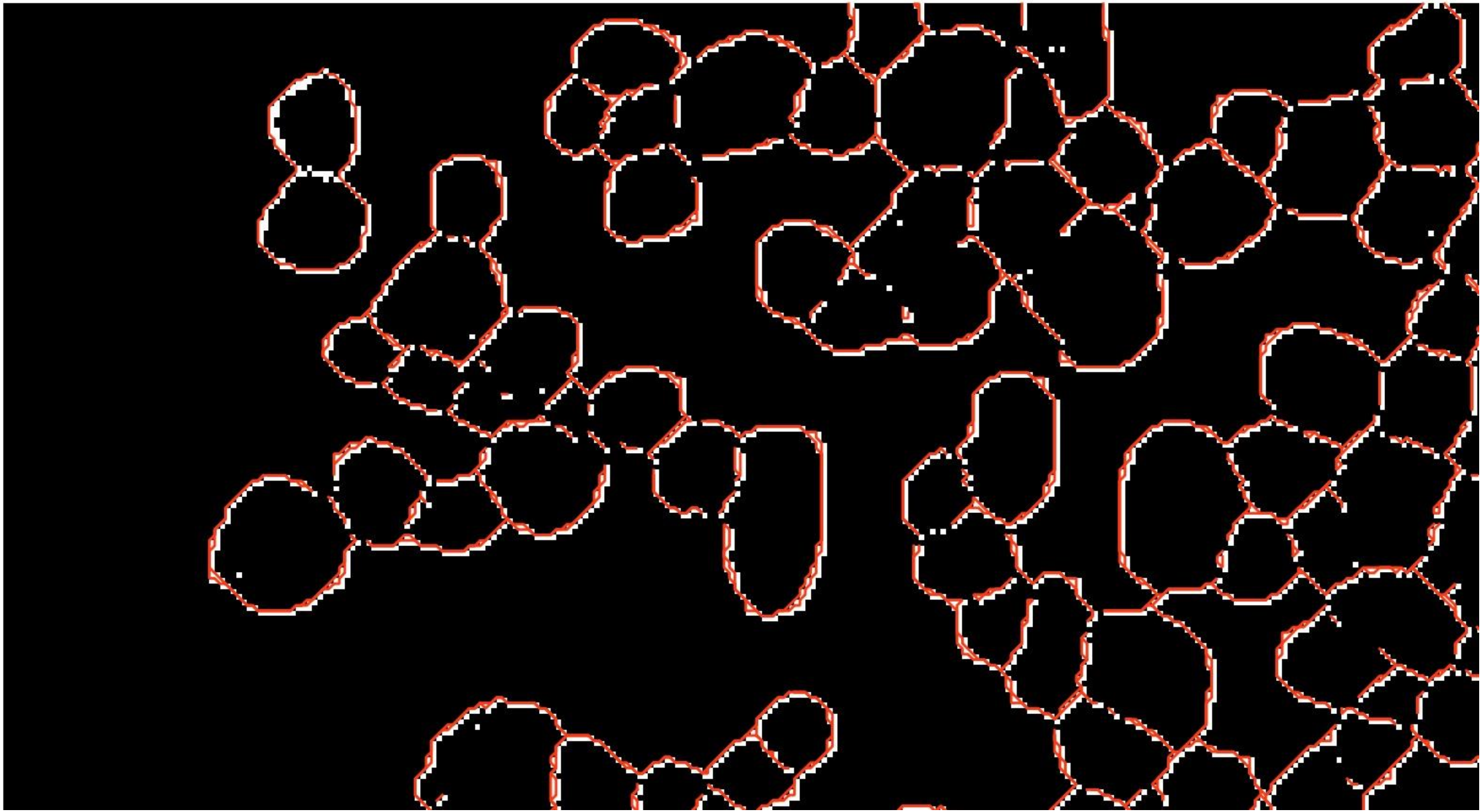
Contour point ->polylines



Trough length

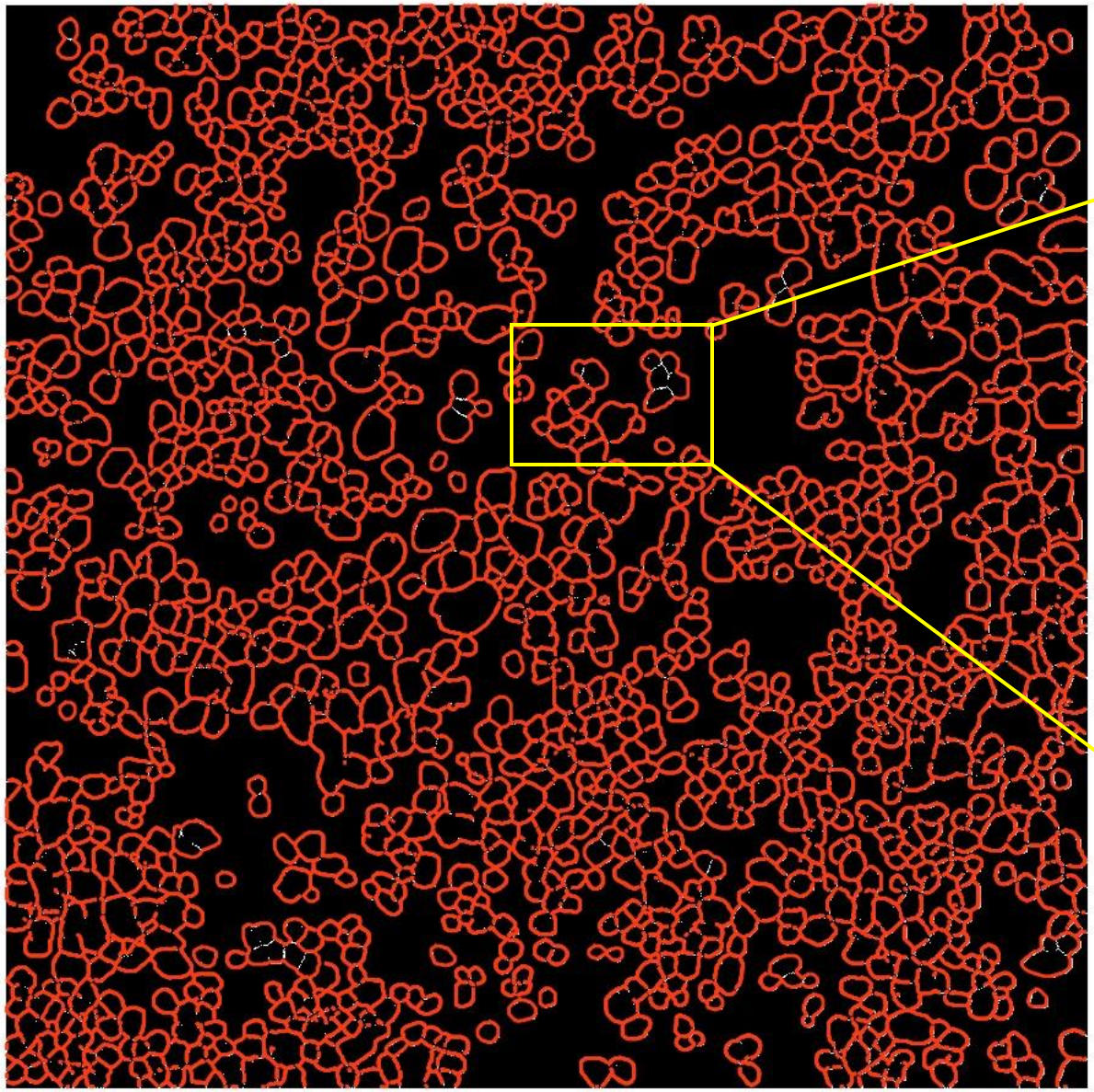
Vetorize: avoid duplicated lines

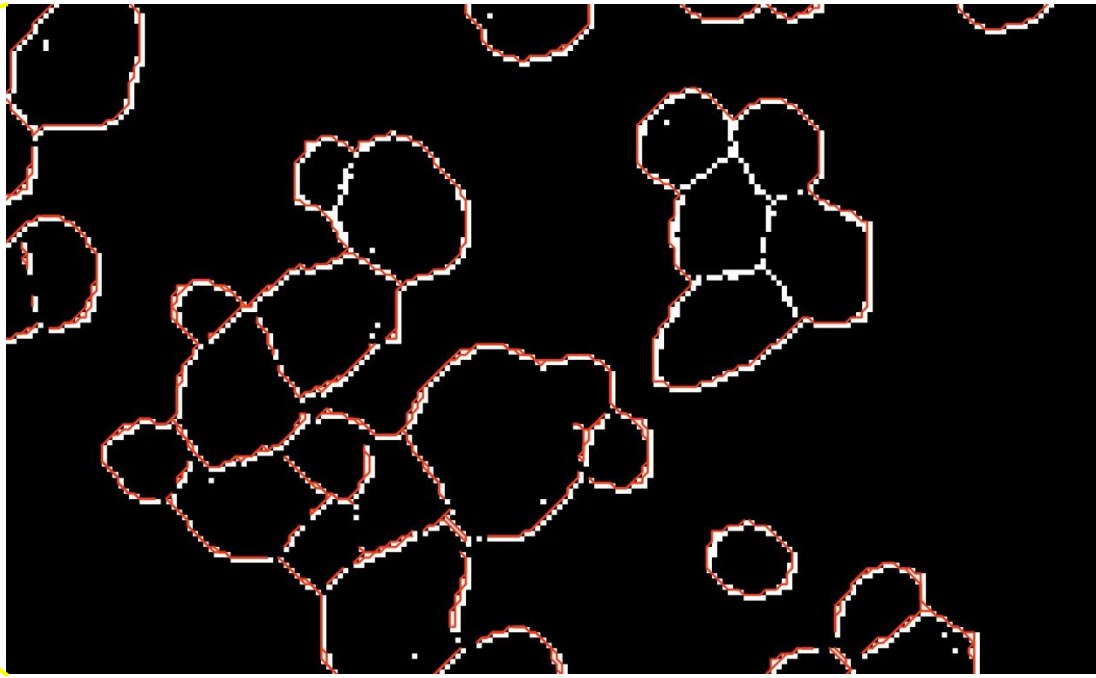
Test on 1 pixel within a grid



Trough length

Test on 1 pixel within a grid





Total length:
129,618.031 meters
129 km

Runtime:
45s

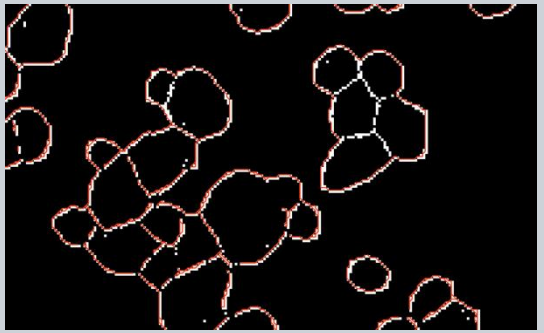
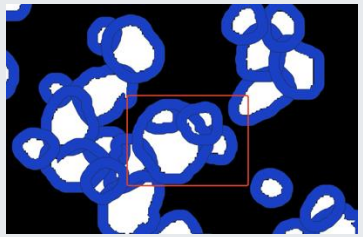
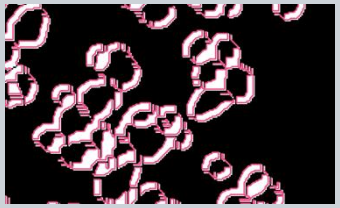
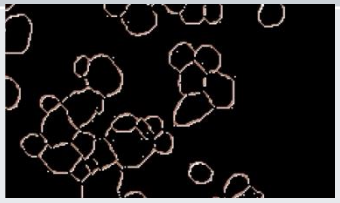
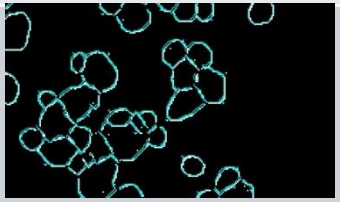
Meeting 1/7/2025

- Trough length refine
  - Capture trough between intersected IWPs
  - Reduce processing time
- Trough width

Intersected IWPs

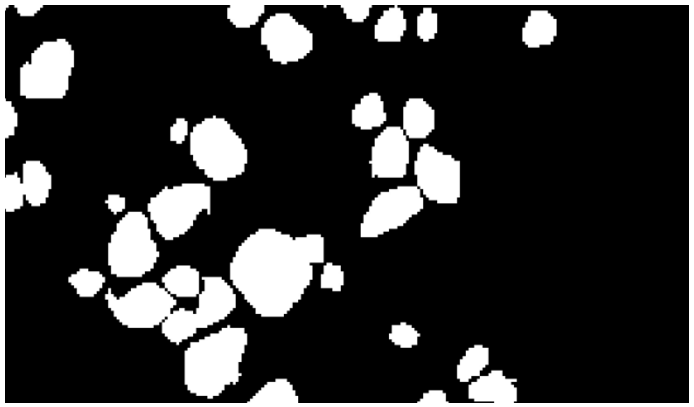
Trough length refine

(See details in the next slide)

methodology	Kernel size	Time	Results
Original method: *To ignore trough between intersected IWPs* ➔ Rasterize ➔ dilated ➔ difference ➔ skeletonize ➔ contour point ➔ transformation ➔ polyline	3	Transformation: 31.67s Total time: 32.05s	
Method 2: *To capture trough between intersected IWPs* ➔ vector ➔ buffer ➔ difference ➔ Rasterize ➔ Skeleton ➔ contour point ➔ transformation & conversion ➔ polyline 	1	More than 5 mins	
	2	Transformation: 64.00s Total time: 65.32s	
	3	Transformation: 49.99s Total time: 52.52s	
	5	Transformation: 39.16s Total time: 40.46s	

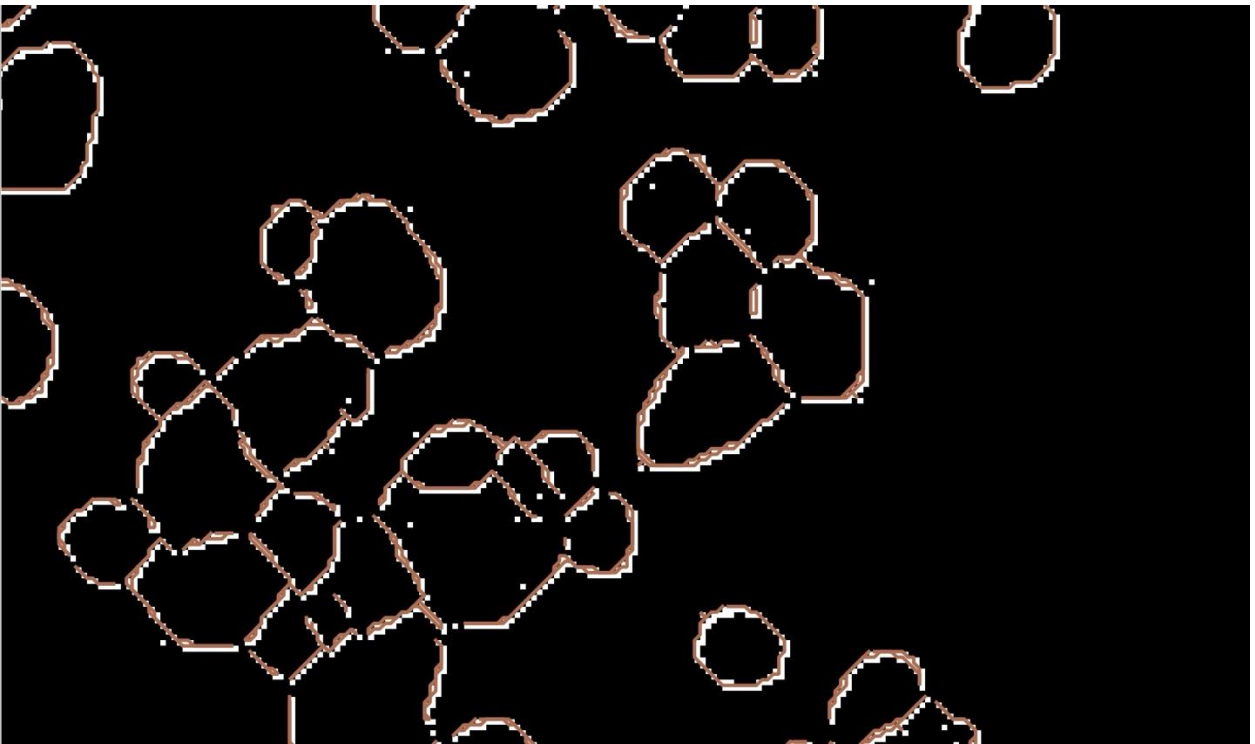
Trough length refine

Original IWPs

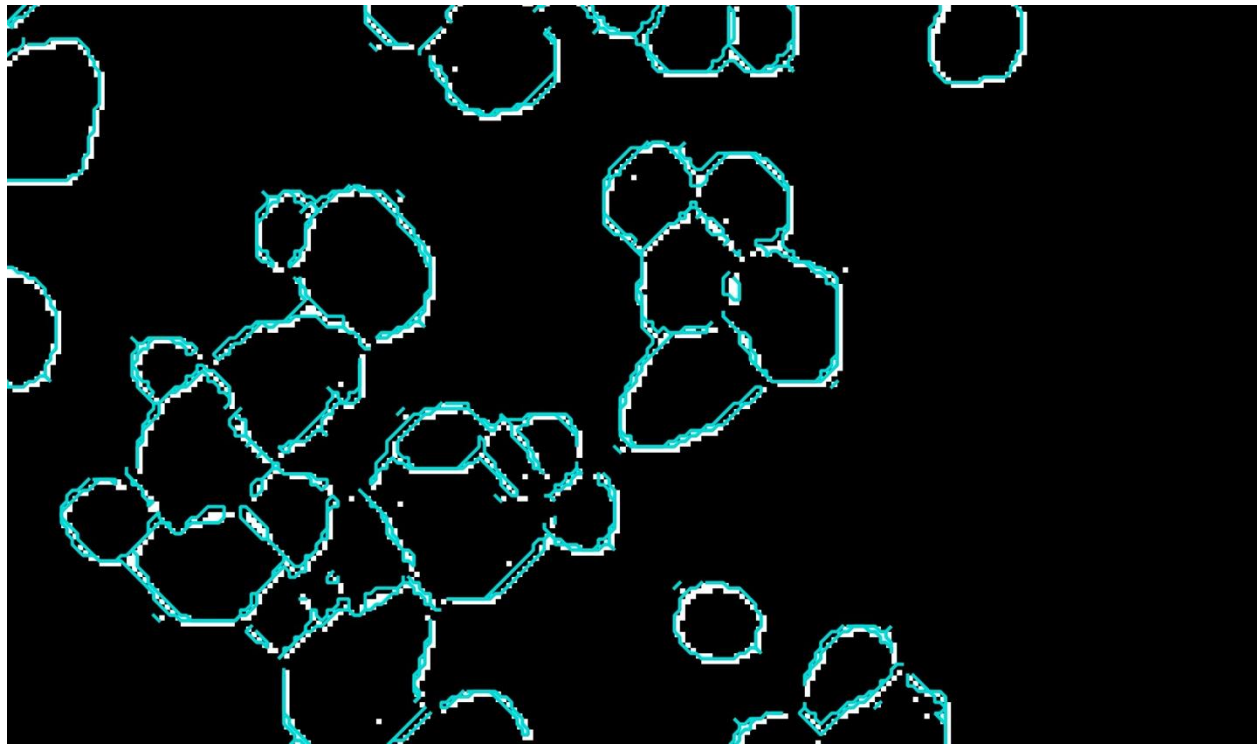


Kernel 2 (problematic)

Kernel 3



Kernel 5



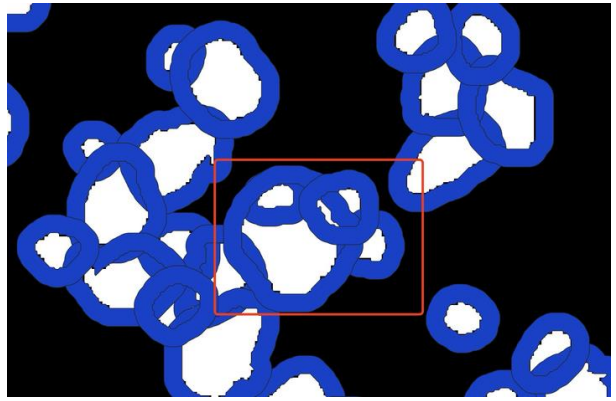
Meeting 1/14/2025

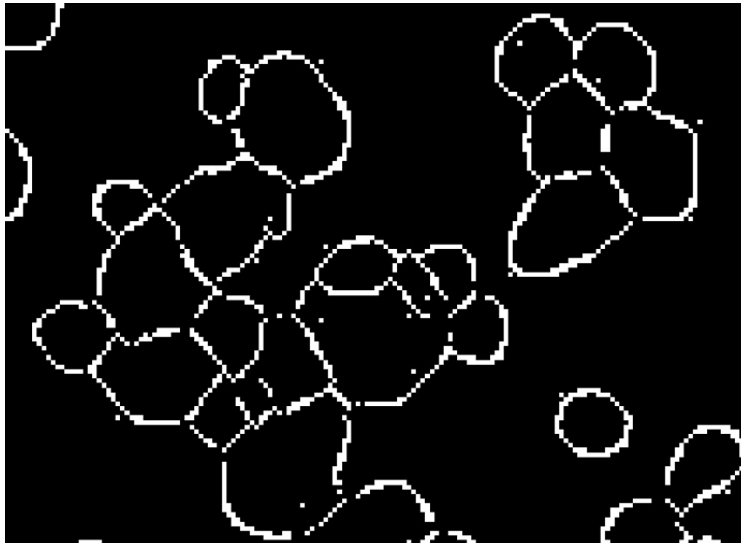
- Trough length
  - Identify trough area: Whether we should keep intersected IWPs
  - Skeletonization
  - Conversion: skeletonization -> polylines -> trough length
  - Conversion speed
- Trough width
  - Set different clustering parameters

Identify trough area: Keep the inner trough or not

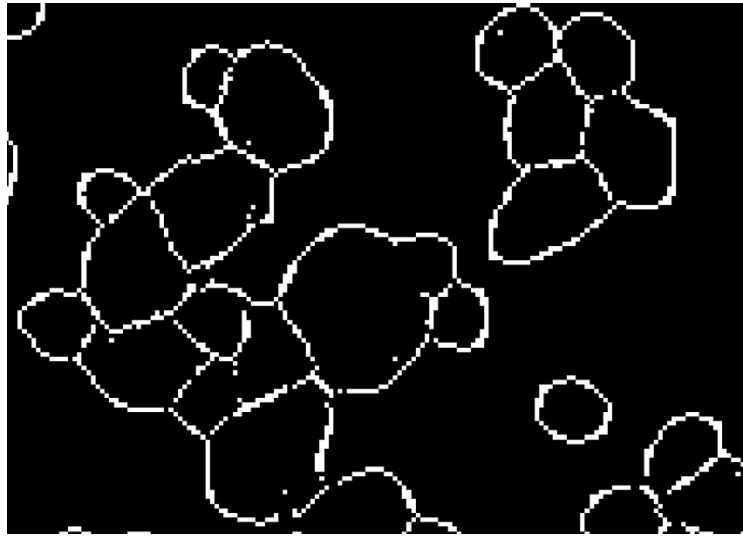
Original IWPs

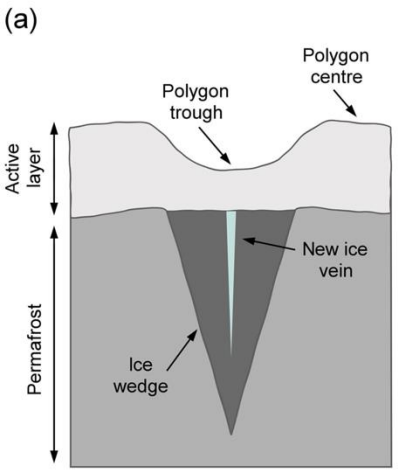
Keep

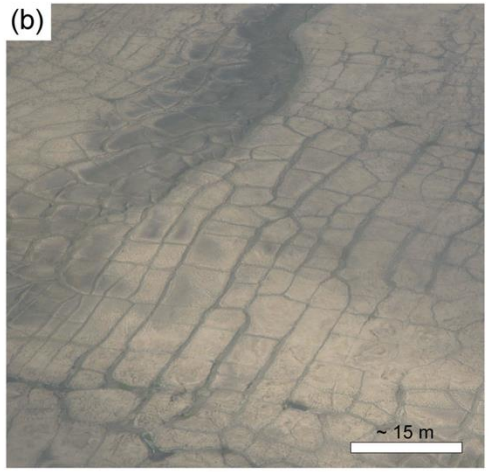




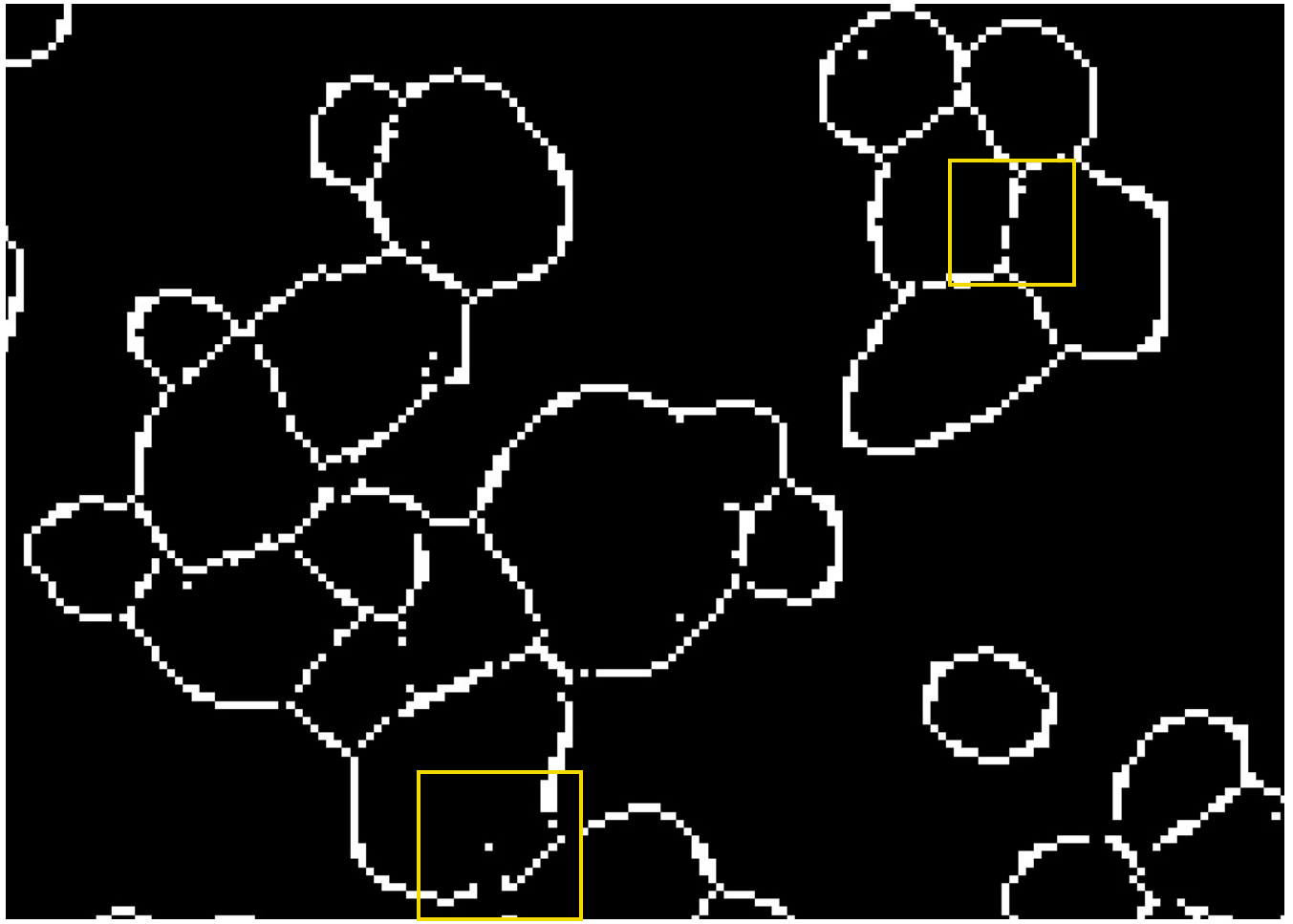
Ignore





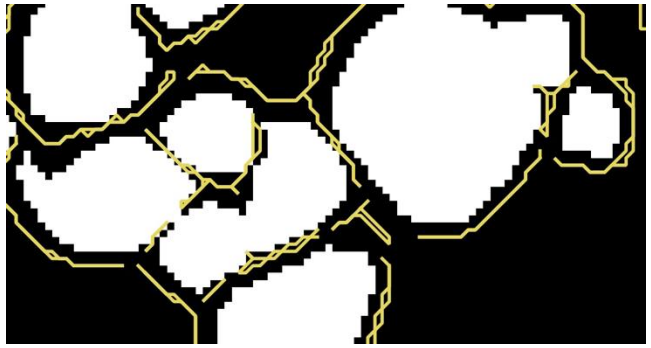


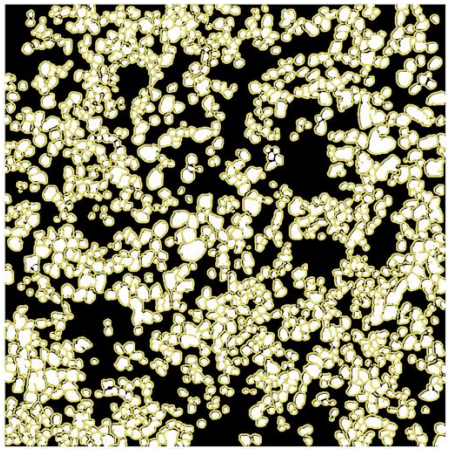
Skeleton image: not consistent



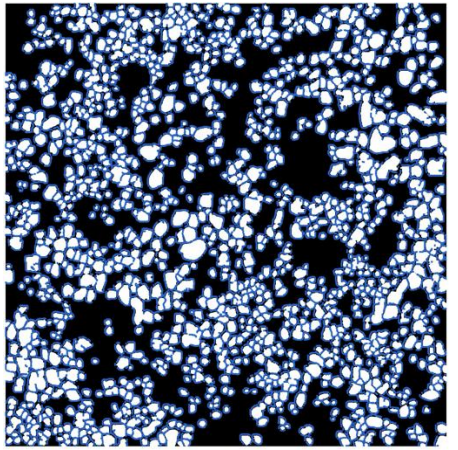
Skeleton image -> contour points -> Polylines:

1. Opencv.findcontours to find contour points



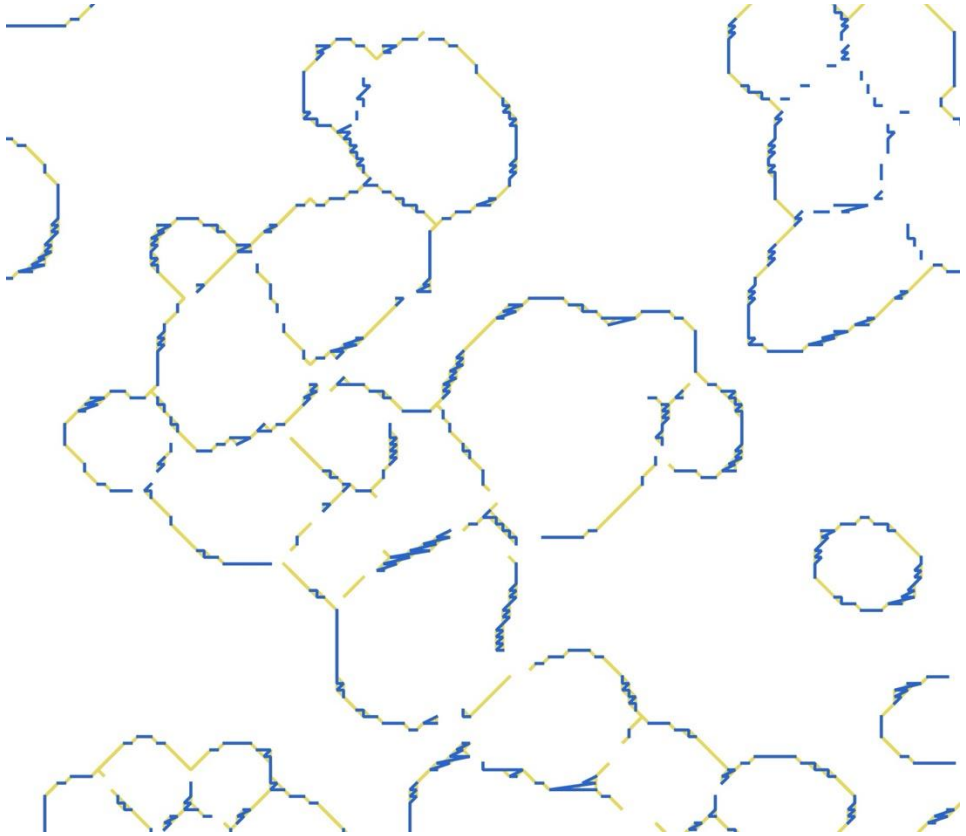


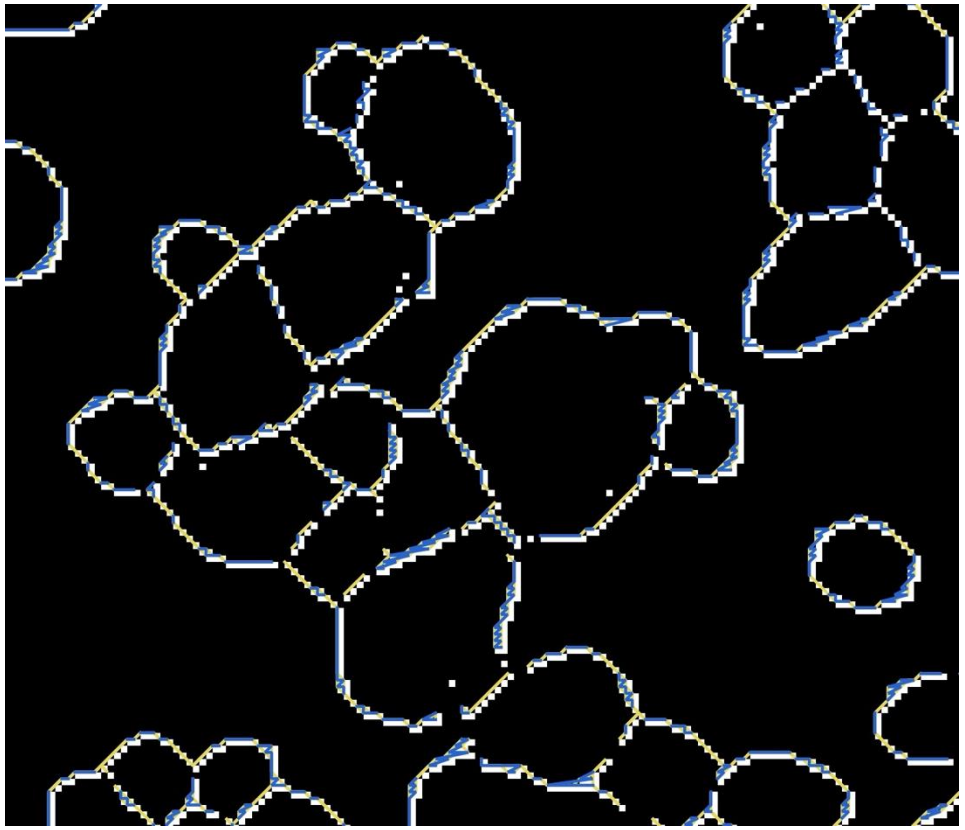
2. Scipy.label to find contour points



Original IWPs

Compare of different trough track





Yellow: original method-opencv

Blue: another method-scipy

Compare of Process Efficiency

CPU cores available on our server: 80 (20 recommended)

Parallel processing core	Processing time / pixel	Count of contours	Polylines
1	49 ~ 52 s	23,638	11,642
10	6 ~ 7 s		
50	4 ~ 6 s		
80	4 ~ 6 s		
100	4 ~ 6 s		

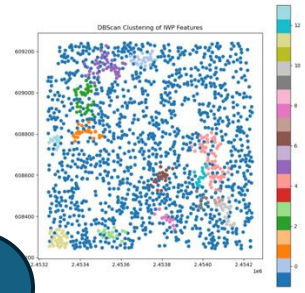
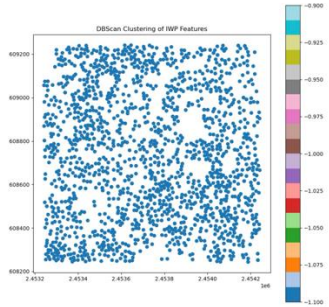
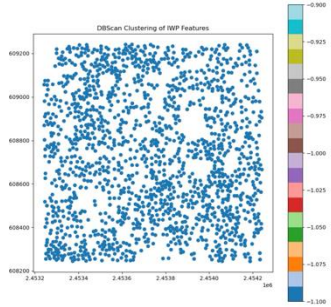
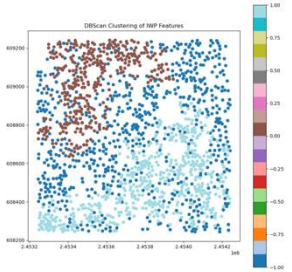
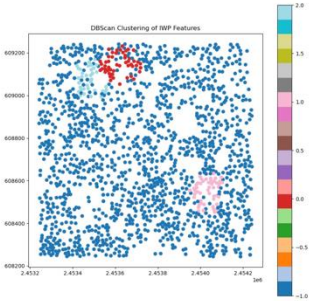
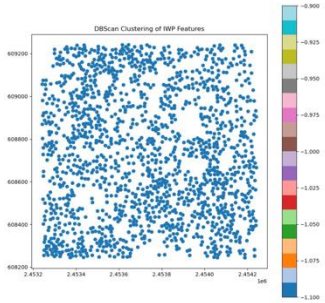
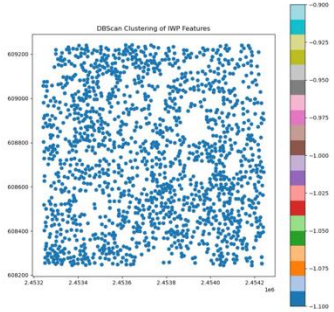
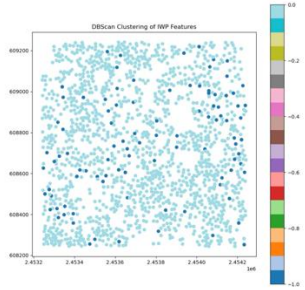
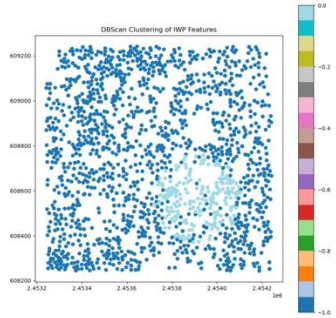
Total length of trough in this pixel: **55,554.81 meters**

Contours:

```
Y coordinates and X coordinates: (array([999, 999, 999]), array([773, 774, 775]))
Y coordinates and X coordinates: (array([999, 999]), array([778, 779]))
Y coordinates and X coordinates: (array([999, 999]), array([867, 868]))
Y coordinates and X coordinates: (array([999]), array([870]))
Y coordinates and X coordinates: (array([999, 999]), array([872, 873]))
Y coordinates and X coordinates: (array([999, 999]), array([980, 981]))
```

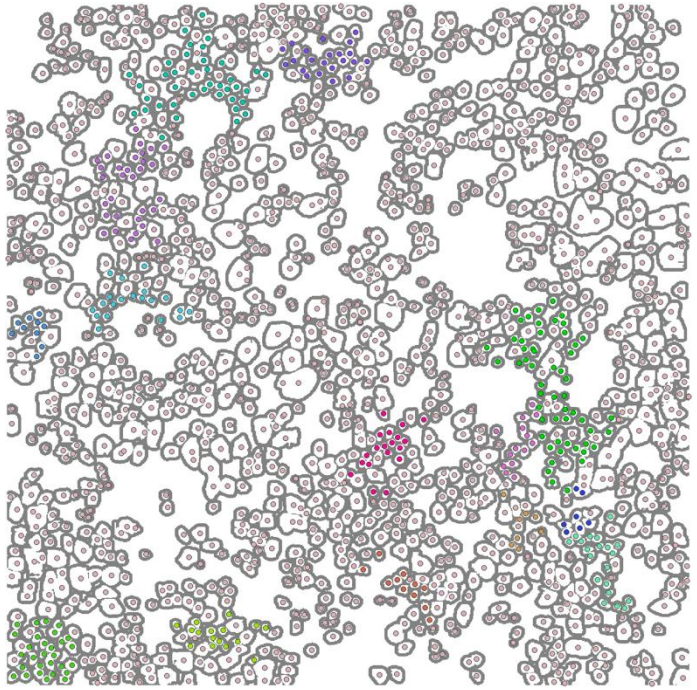
- Processing time / pixel: 5s
- Total count of pixels in a grid: 65, 536
- Maximum processing time of a grid: 90 ~ 110h

DBSCAN: finding IWP proxies

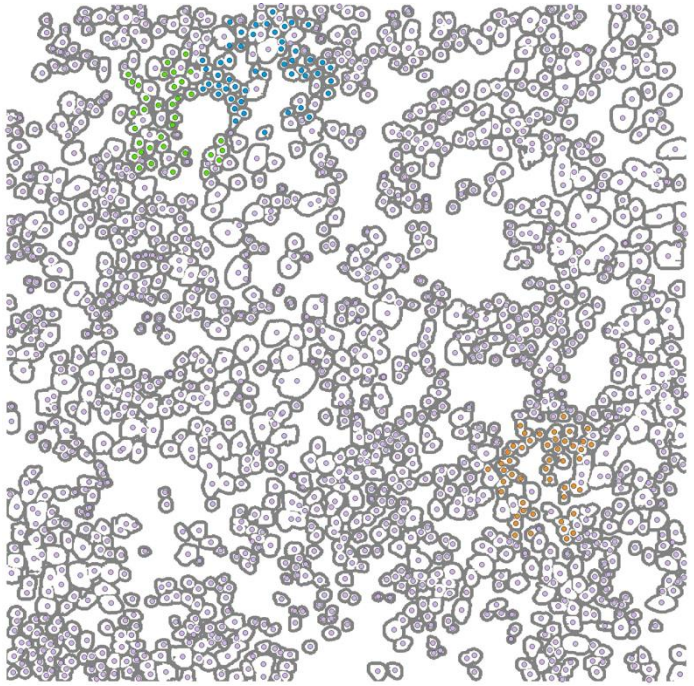
Min samples / eps	10	30	50	80	100	200
50m						
100m						
200m						

DBSCAN: finding IWP proxies

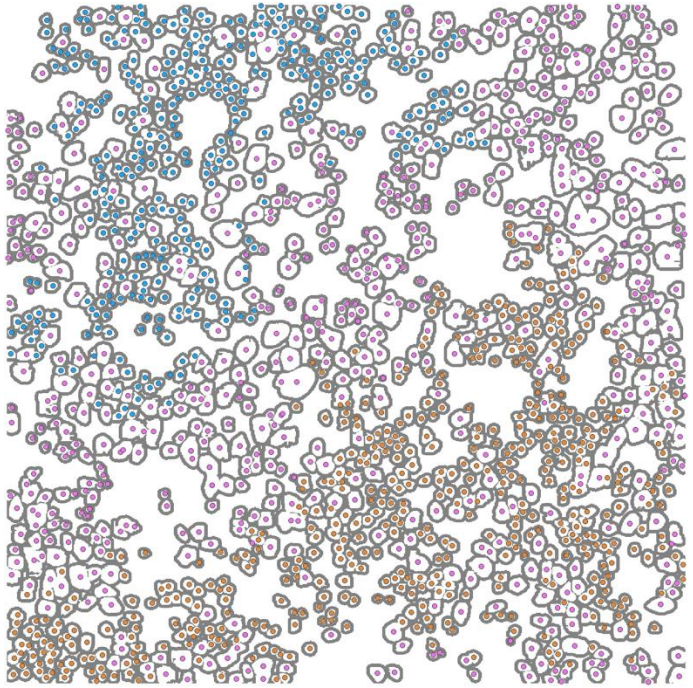
eps: 50m, min samples: 10



eps: 100m, min samples: 50



eps: 100m, min samples: 30



Add map scale

Conclusion

- Optimize skeletonized image, find better skeletonize method. Check arcpy
- Vetorize time: avoid vectorize to try other methods
- Include large IWP in calculating Width: exclude “area” as criteria, try smaller max distance

Meeting 1/21/2025

- The length of ice-wedge network
  - Identify trough area: Whether we should keep intersected IWPs
  - Skeletonization improvement
  - Faster conversion: skeletonization -> polylines -> trough length
  - Validation
- Trough width (Find proxies)
  - DBScan clustering algorithm parameters

Ice wedge polygon (IWP) & Trough: formation

- **Melting:**

1. The thermal equilibrium of the permafrost is disrupted (rising temperature/loss of vegetation) -> upper part of thermafrost: warm.

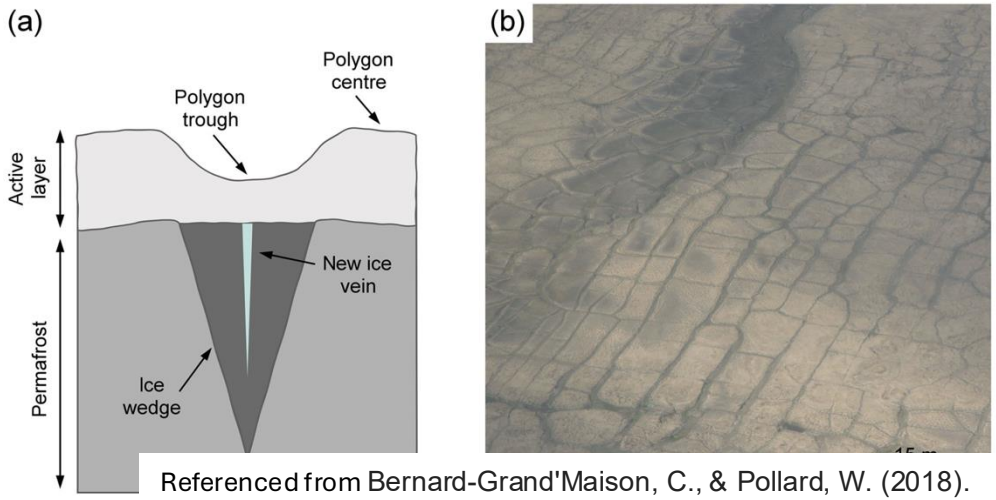
2. Ground ice: melt -> Ground reduction ->. Ice wedge Trough

- **Freezing:**

1. Ice: cooling -> cracks

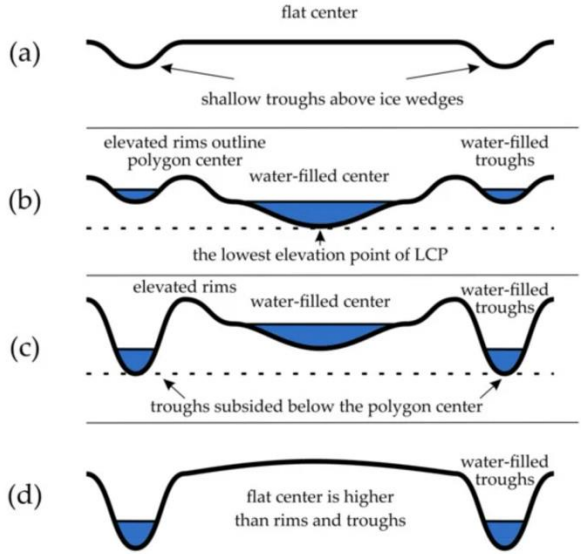
- **Melting:**

1. Cracks: full with meltwater
2. Ice vein formation: Penetrating deeper



Referenced from Bernard-Grand'Maison, C., & Pollard, W. (2018).

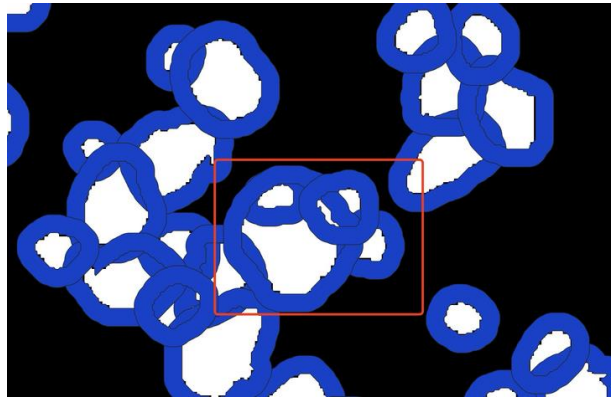
Schematic profiles of mapped polygon types

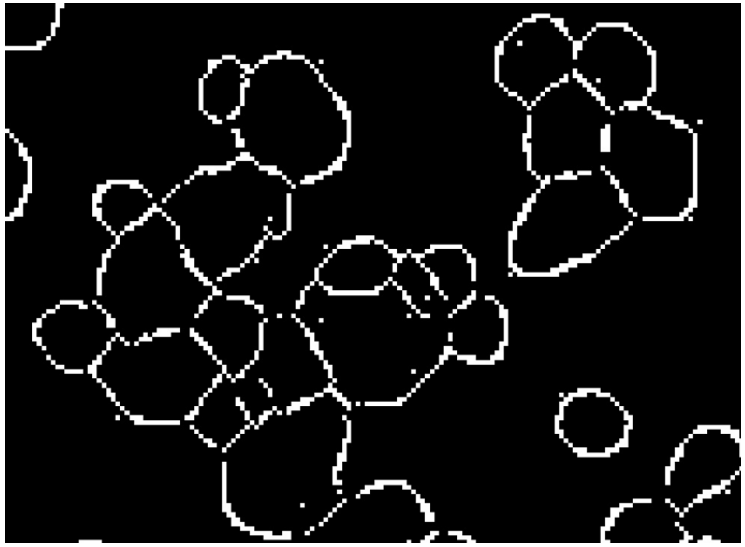


Identify trough area: Keep the inner trough or not

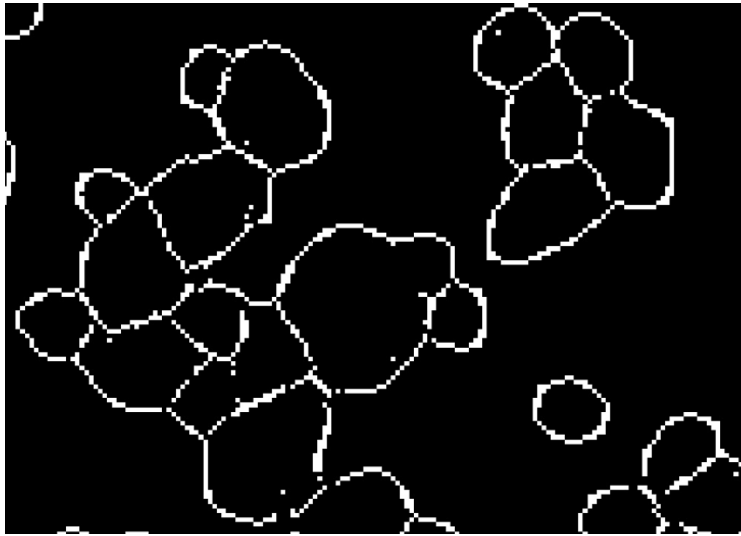
Original IWPs

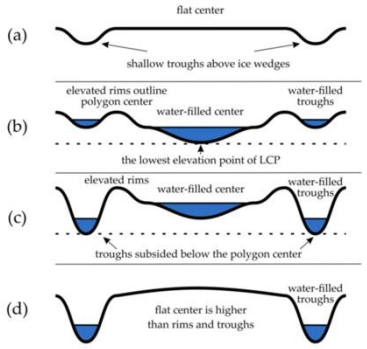
Keep





Ignore





Schematic profiles of mapped polygon types

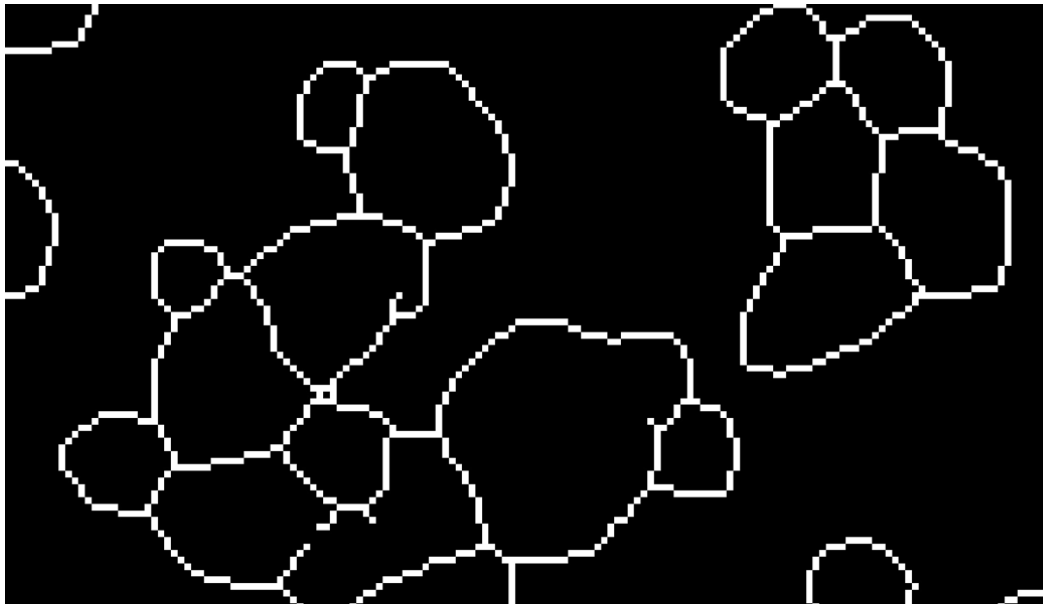
Skeletonization

Previous

Python:

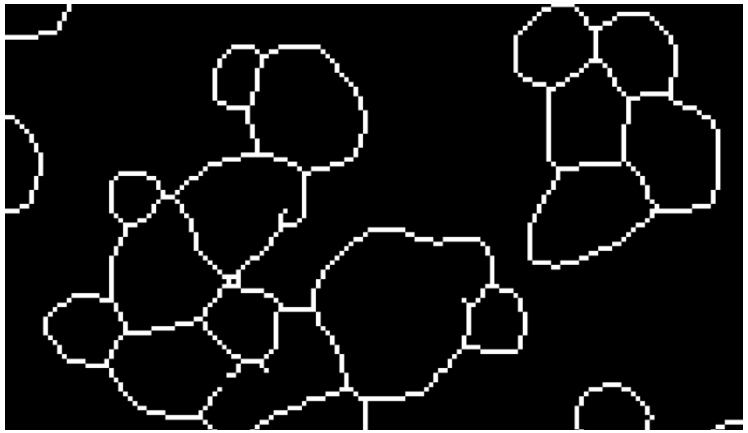
skimage.morphology
.skeletonize

Current

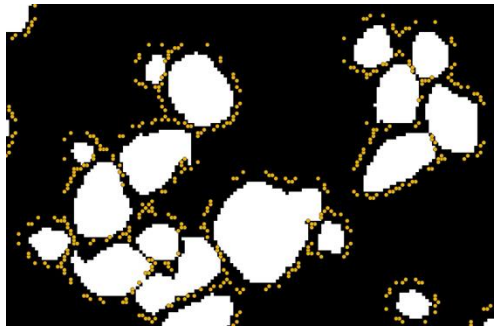


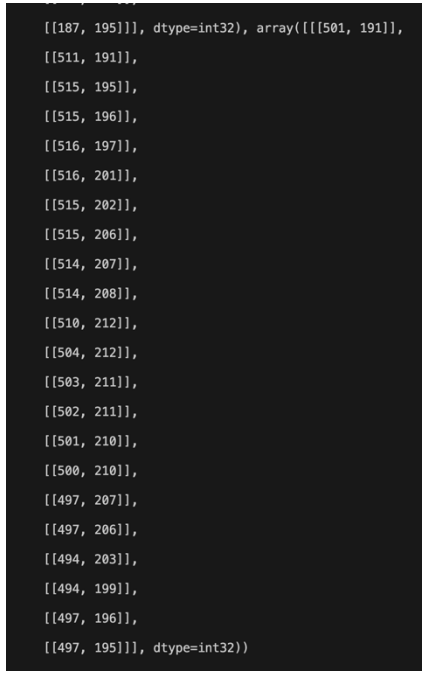
Conversion: skeleton -> contour points -> polylines

Skeleton image



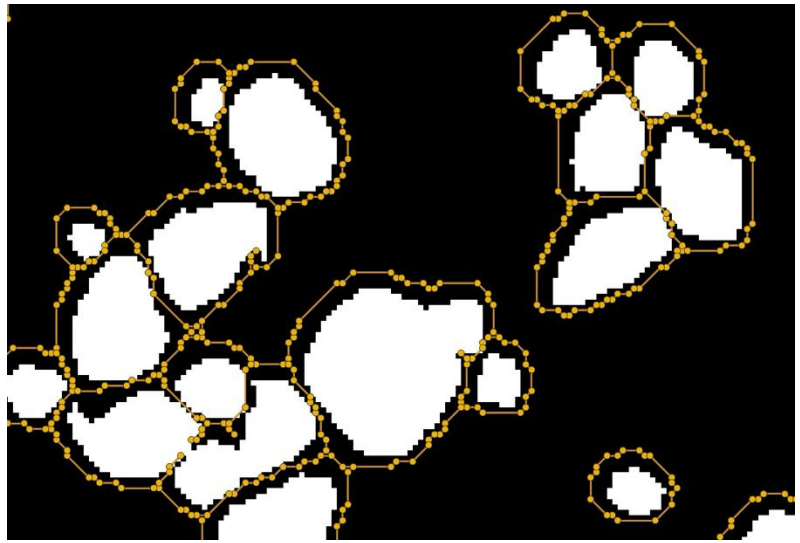
Contour points (1,612 pts)





Reprojection

Polylines 3413



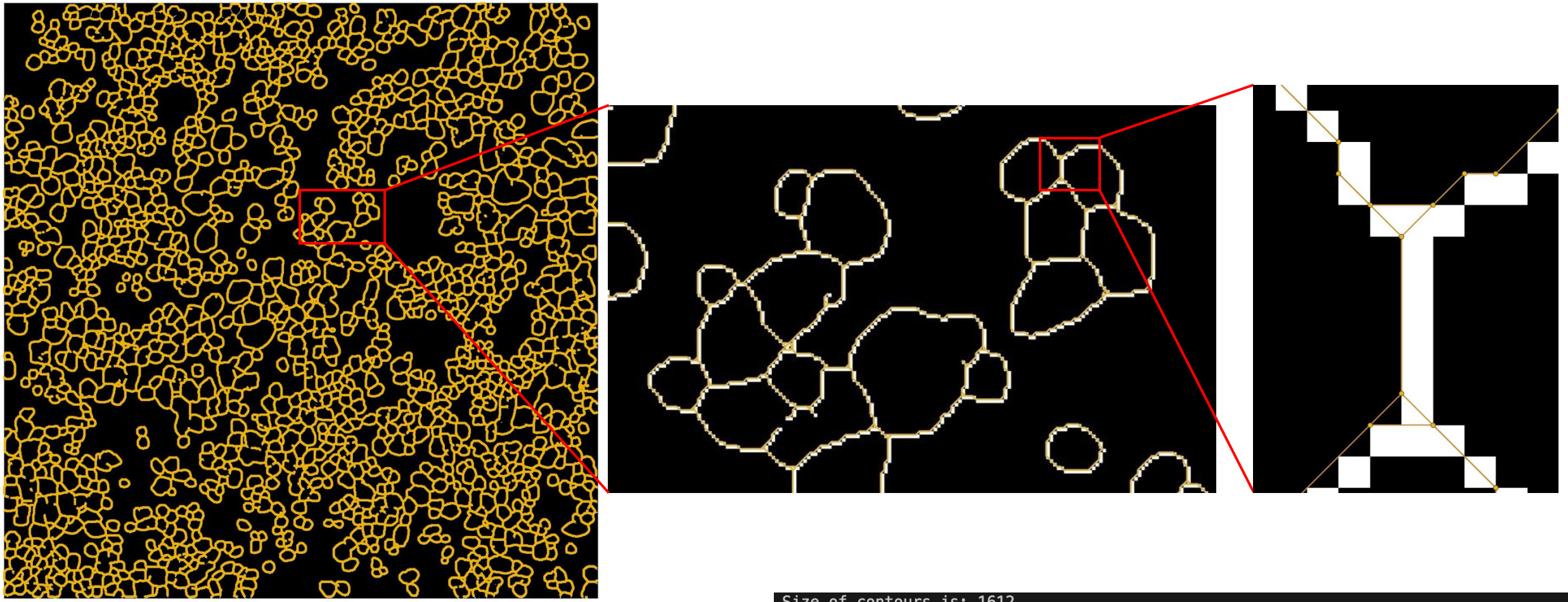
Process in parallel: 20 cores

Python package: cv2.findcontours

Processing time: ~1s

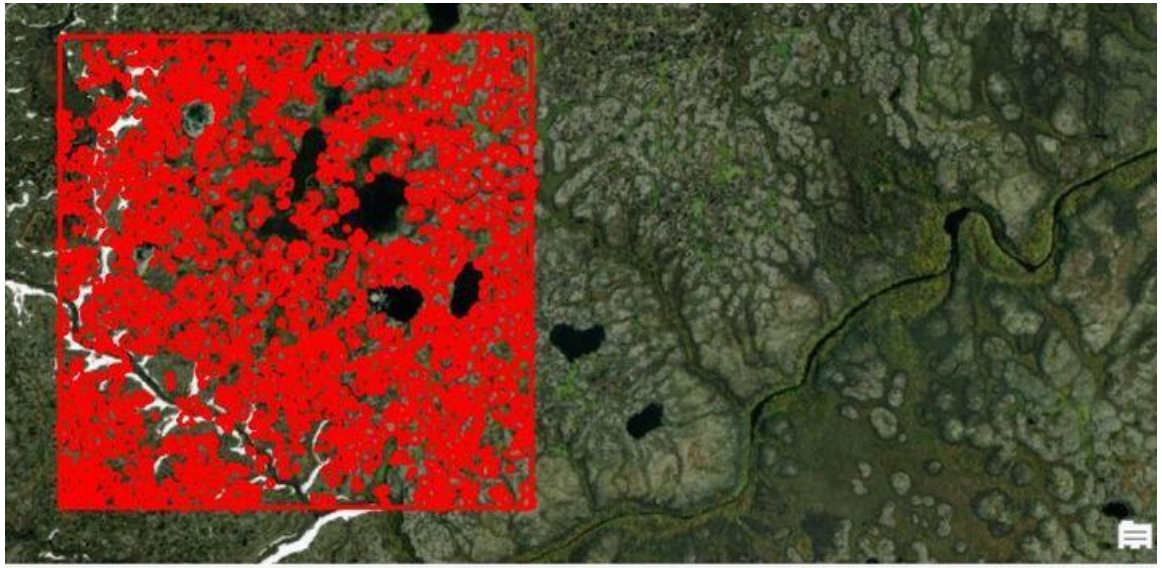
Extracted Trough network Results

Trough length: 146,881m



```
Size of contours is: 1612
chunks size is: 80
The last chunk size is: 29
Number of polylines: 1572
The runtime of converting contours to polylines within the pixel is 0.6643979549407959
Total length of polylines: 146881.97906464734 meters
The total runtime trough length within the pixel is 1.2237823009490967
```

The Length of IW network Validation




```
Size of contours is: 1612
chunks size is: 80
The last chunk size is: 29
Number of polylines: 1572
The runtime of converting contours to polylines within th
Total length of polylines: 146881.97906464734 meters
The total runtime trough length within the pixel is 1.223
```

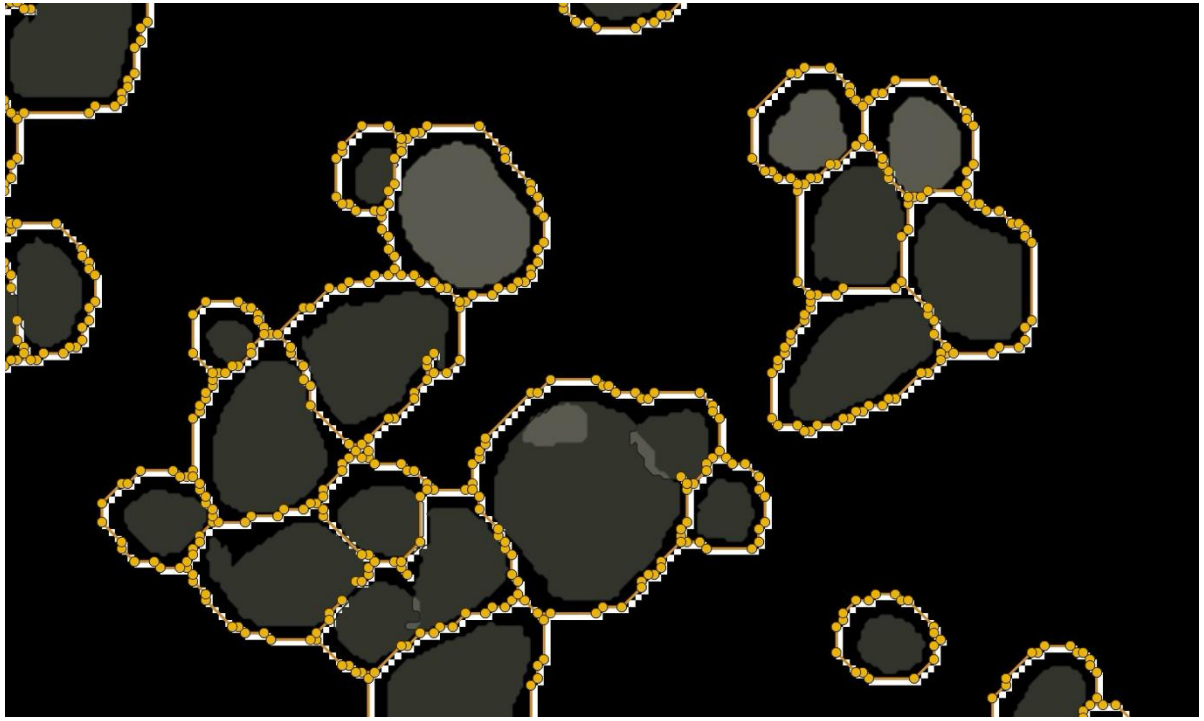
Area

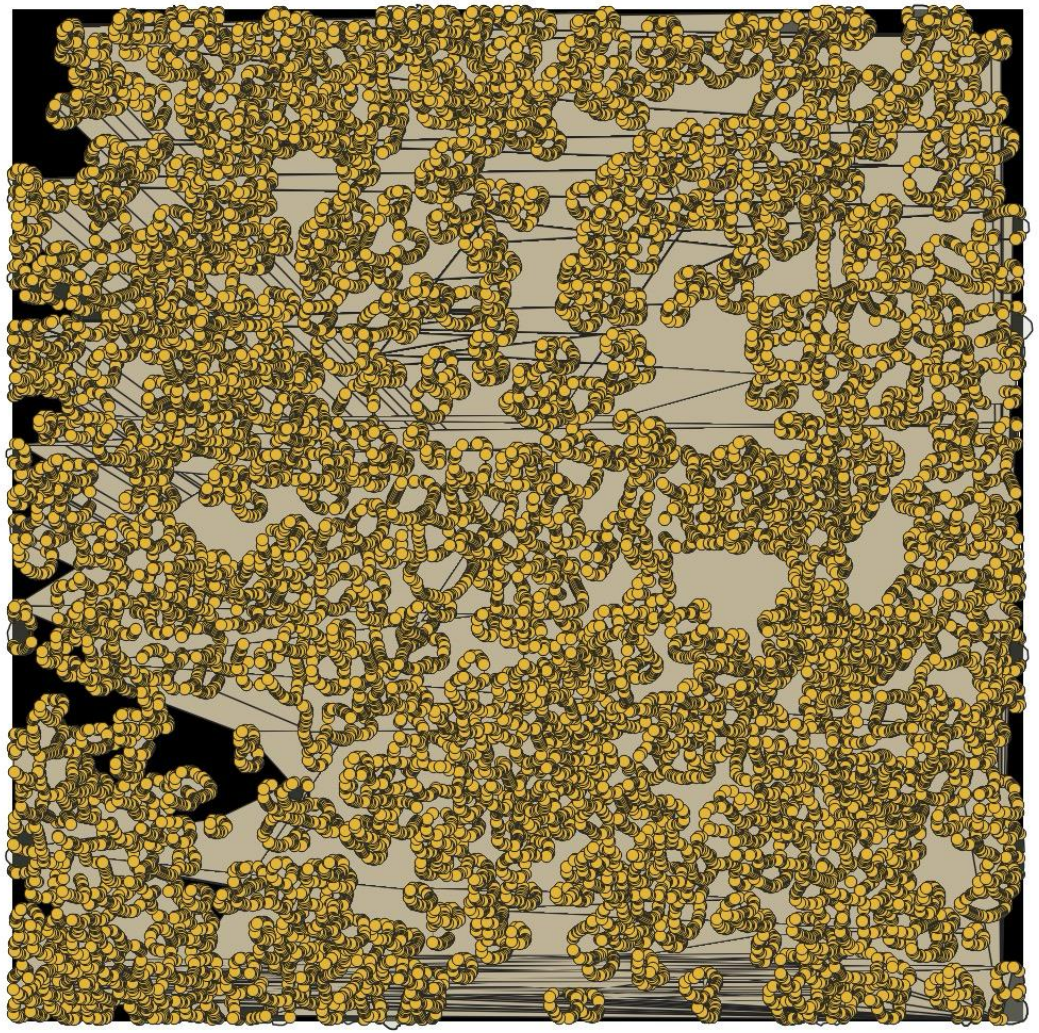
Perimeter

Length

Trough Length Validation

1. Duplicated IWPs
2. Disordered and duplicated contour points





Trough width: DBScan

- **eps**: Maximum distance to distinguish clusters
- **min_samples**: Minimum samples in a cluster, including the core point

High min_samples -> denser clusters.

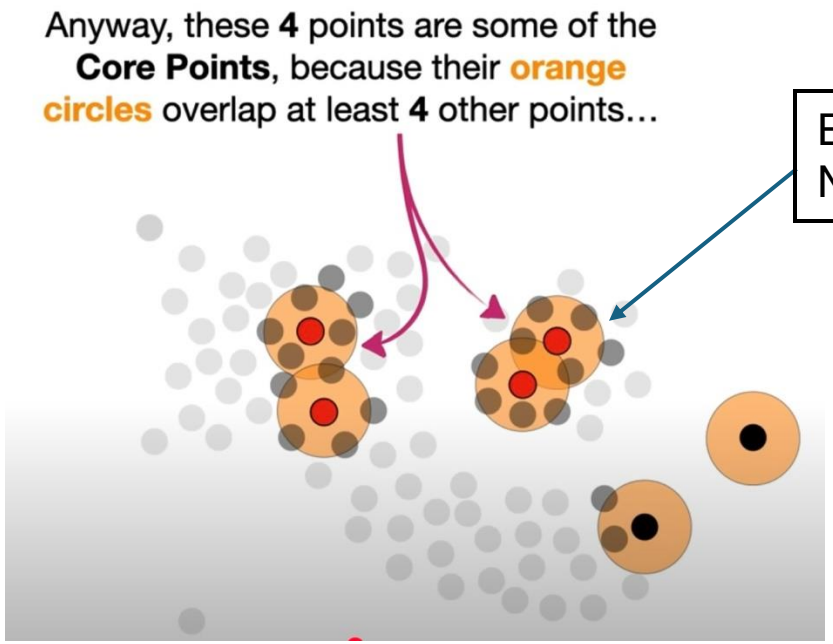
Low min_samples -> sparse clusters.

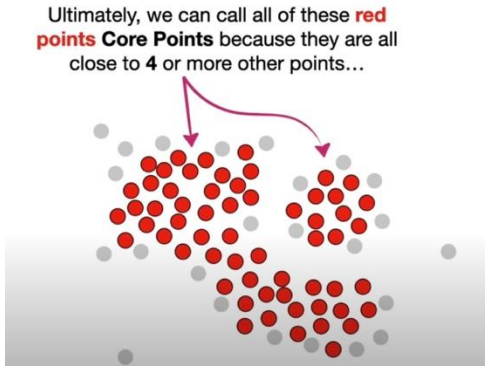
- **Metrics:** The metric to use when calculating distance between instances in a feature array
- **fit_predict**(*X*, *y=None*, *sample_weight=None*)

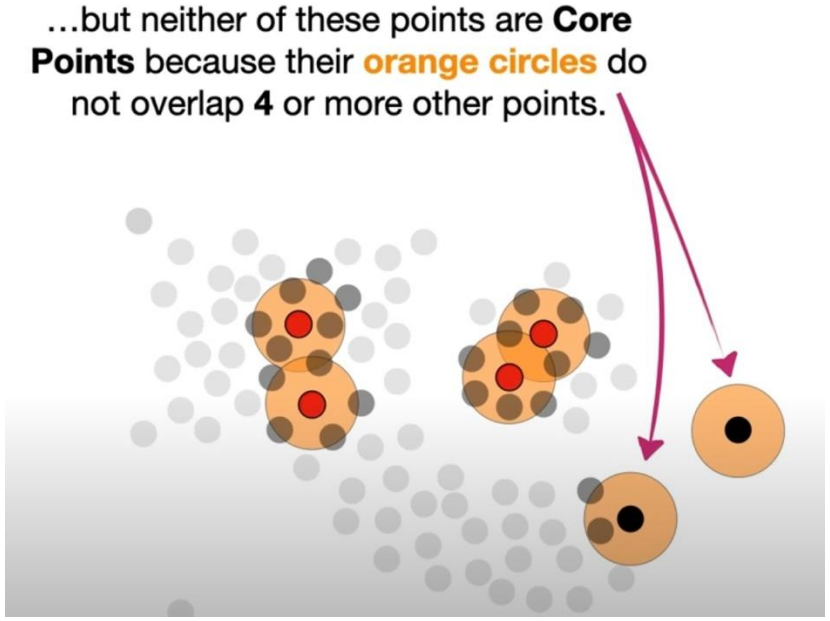
Trough width: DBScan

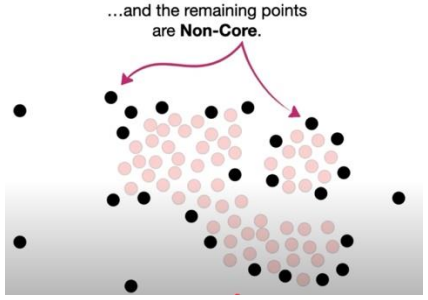
Step 1: Find Core points & Non-core points

- Core points -> Points that have minimum number of samples in a neighbor
- Non-core points: Points that have less than min number of samples in a neighbor





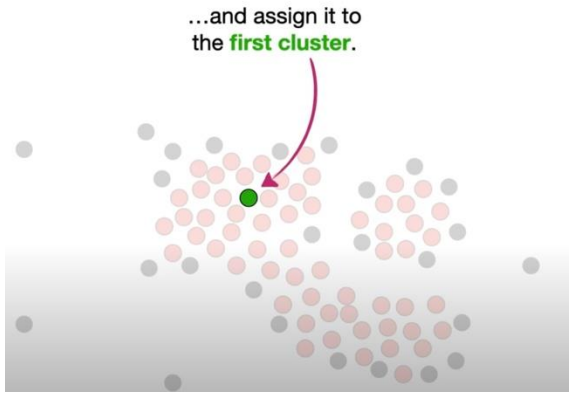


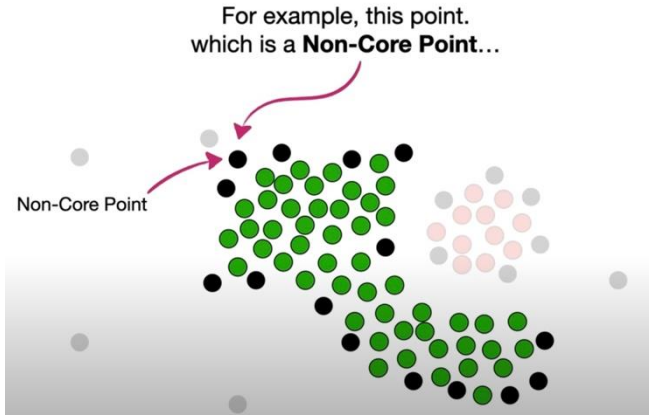


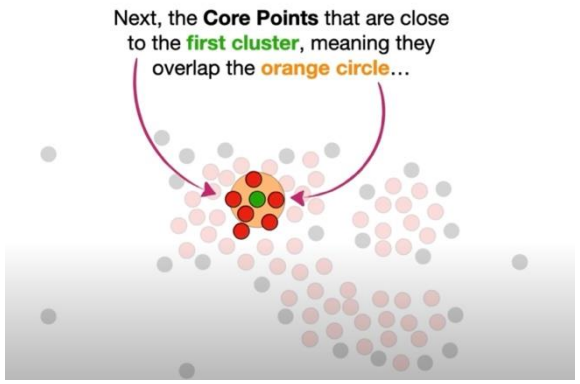
Trough width: DBScan

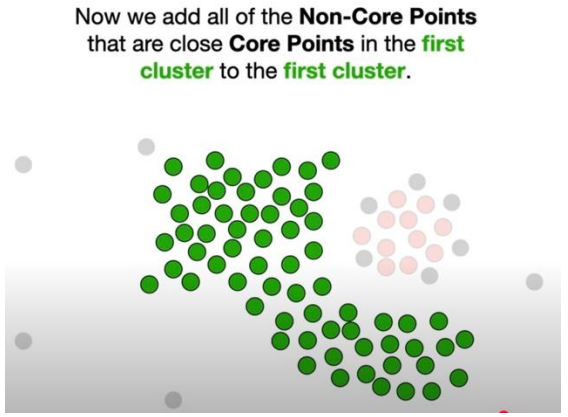
Step 2: Start clustering

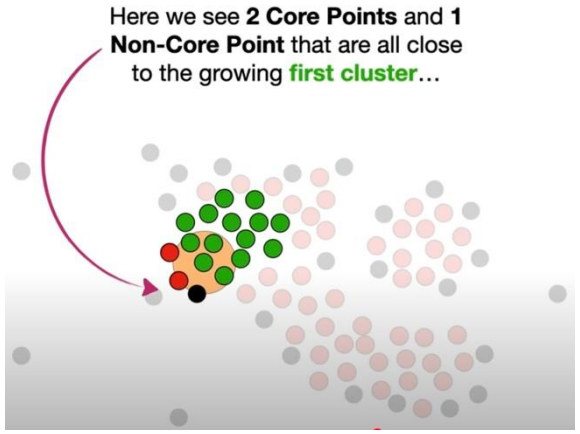
- Find a core point -> first cluster
- Expand the first cluster -> expand along with core points
- End -> stop at and include the non-core points

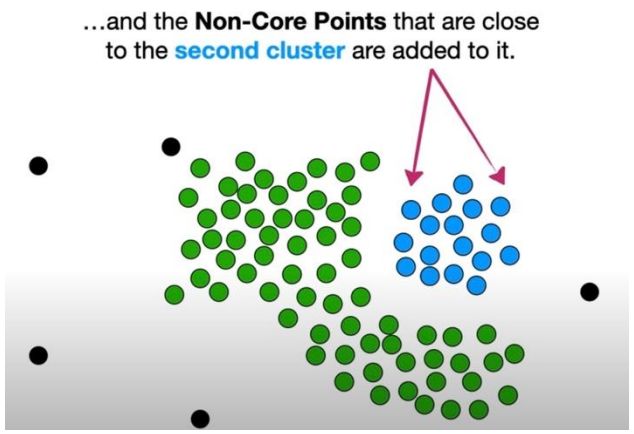








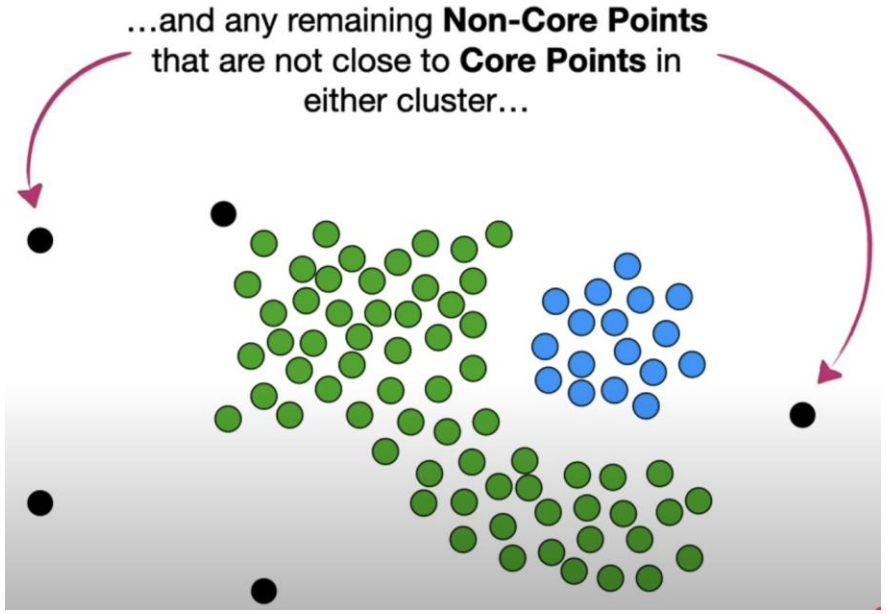




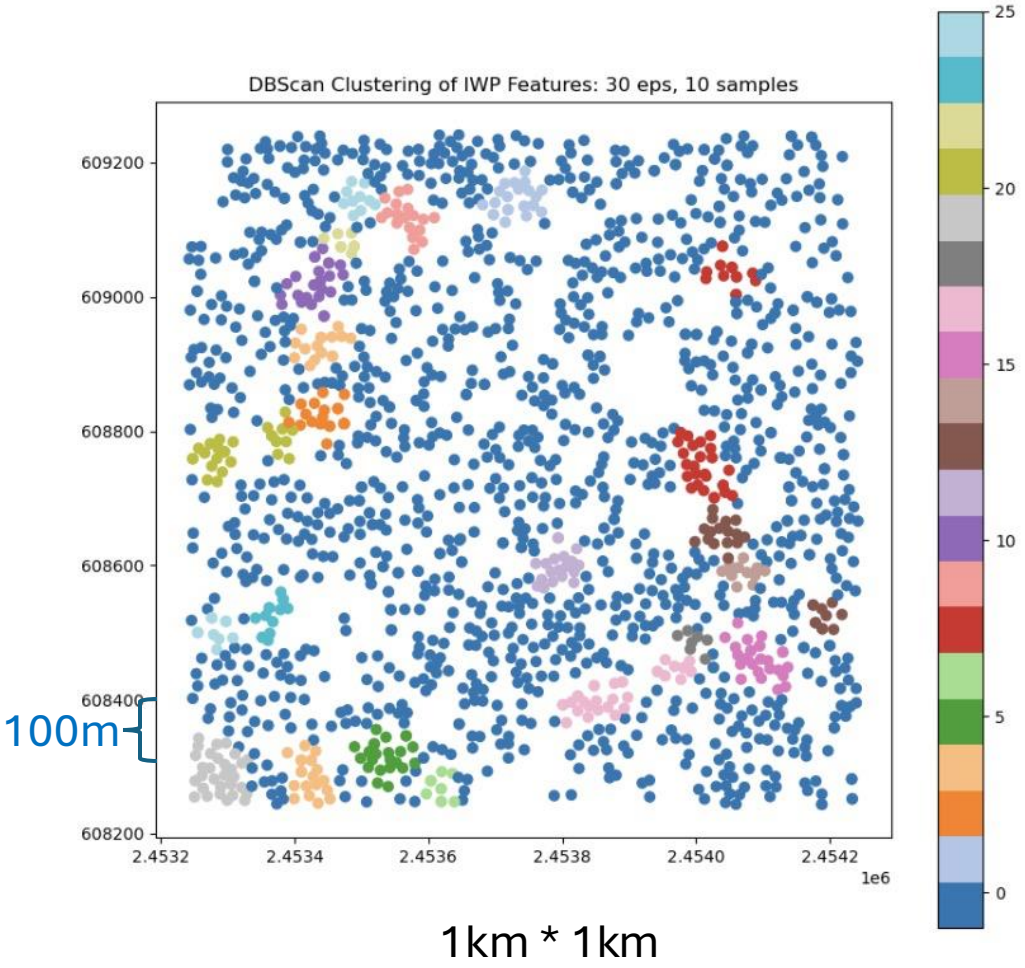
Trough width: DBScan

Step 3: Clusters & Outliers

- Outliers: Points that do not belong to any clusters

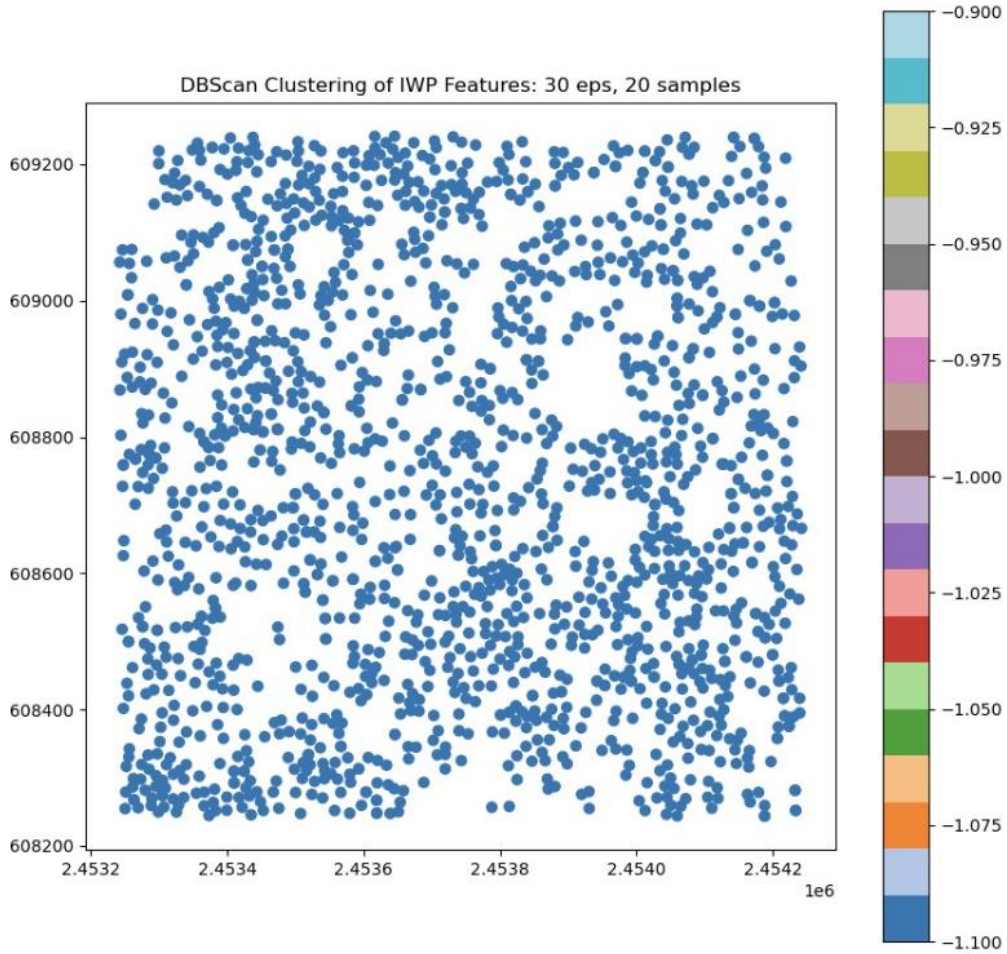


Trough width: clustering of IWP centroids – eps: 30m

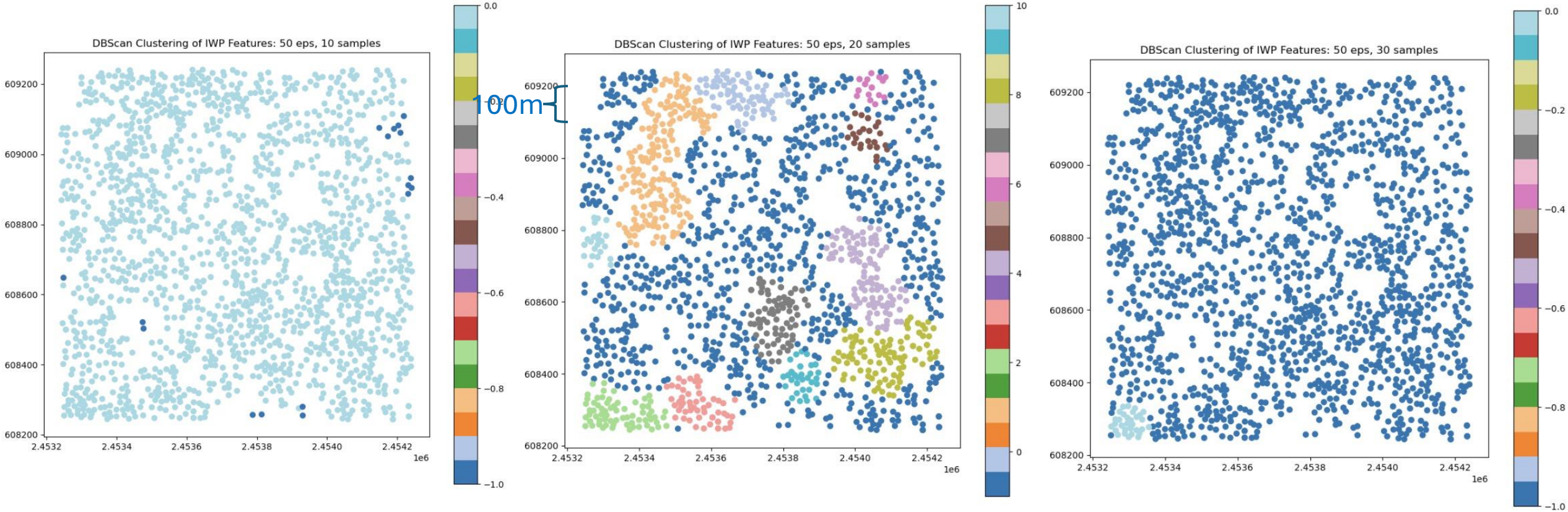


100m

1km * 1km

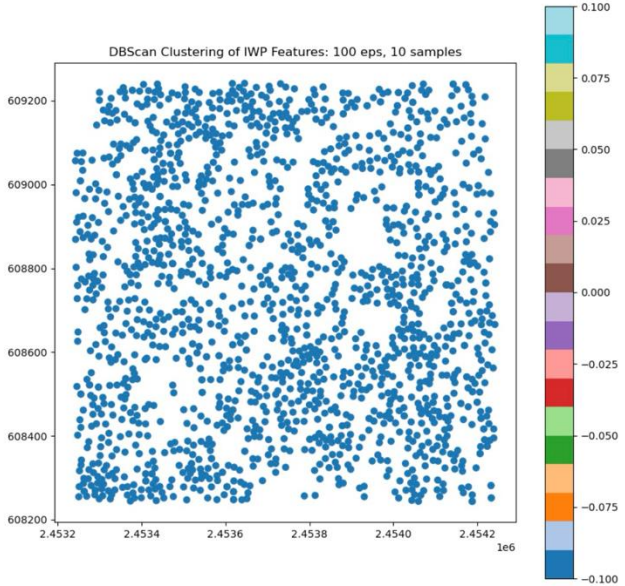


Trough width: clustering of IWP centroids – eps: 50m

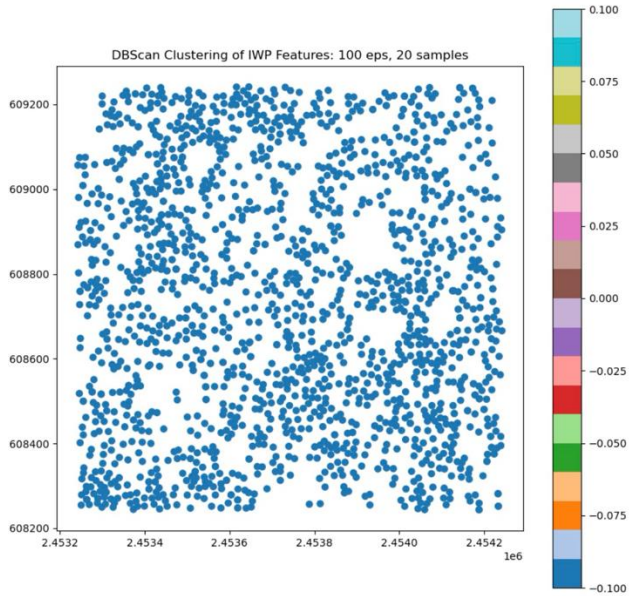


High min_samples -> dense clusters.
Low min_samples -> sparse clusters.

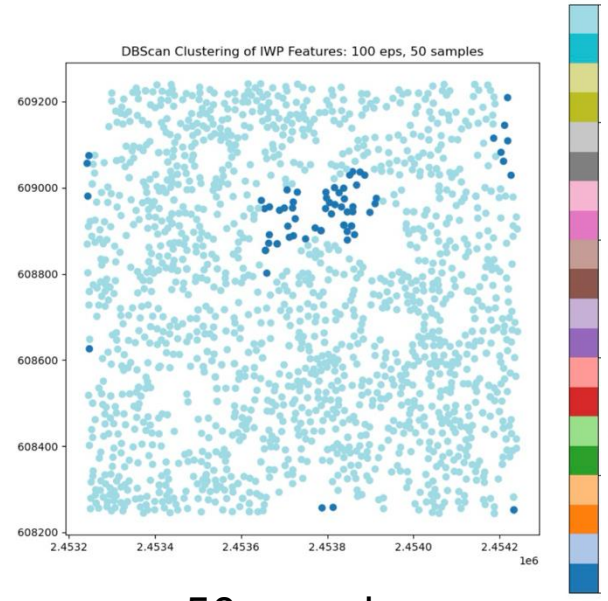
Trough width: clustering of IWP centroids – eps:100m



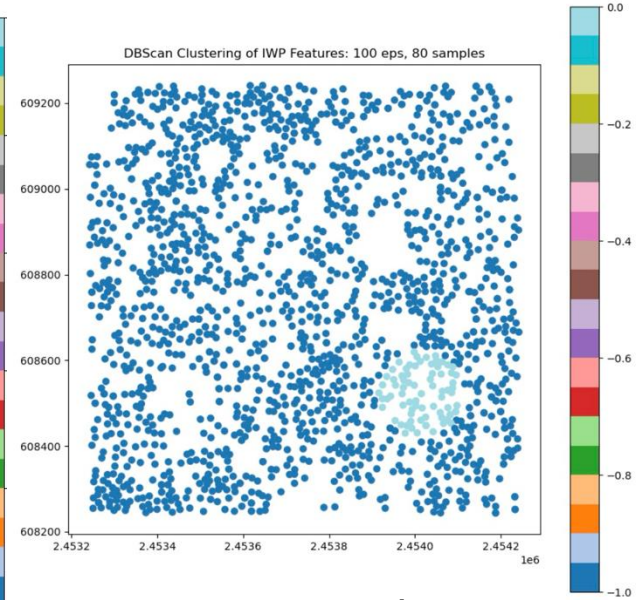
10 samples



20 samples



50 samples

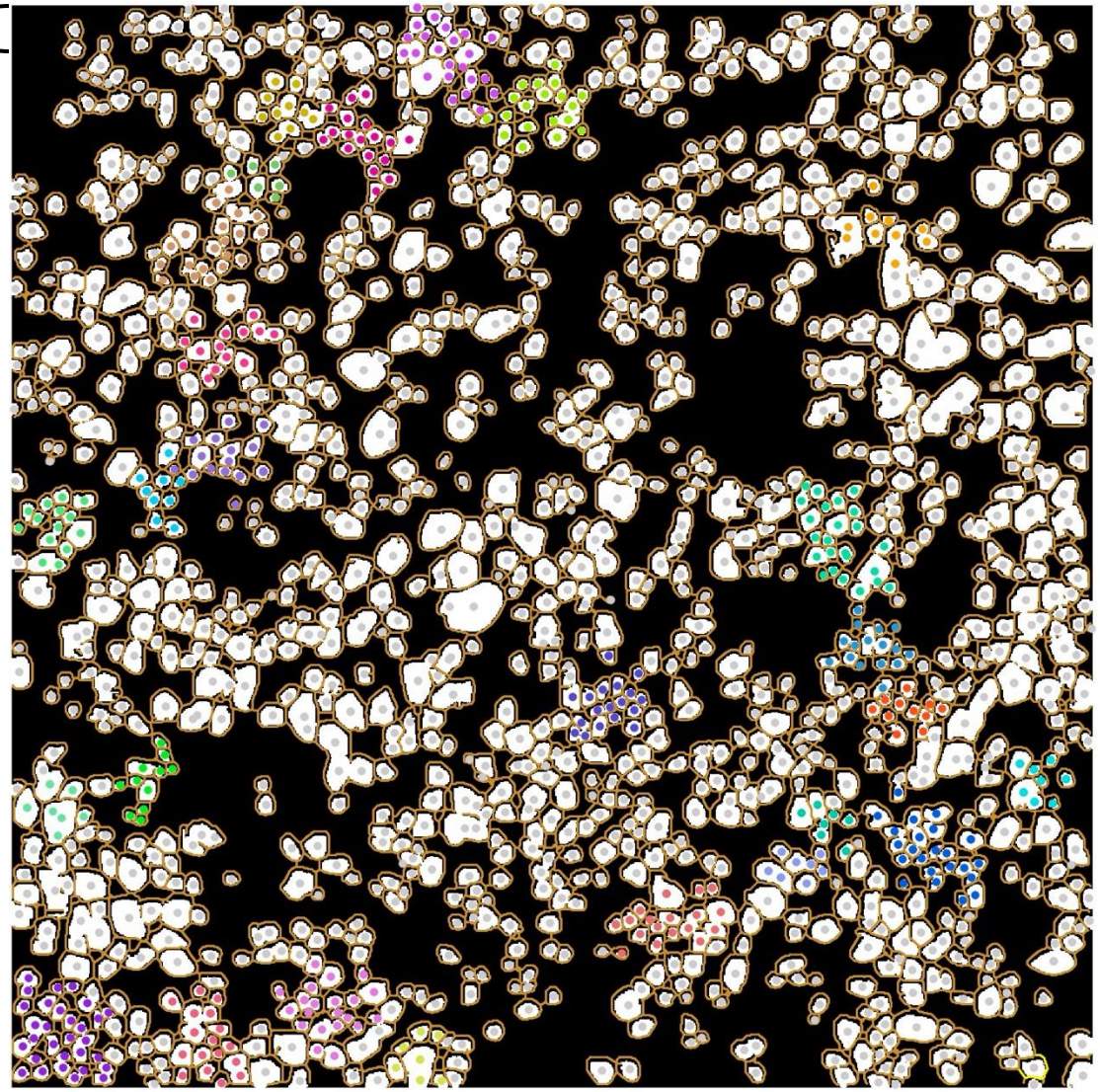


80 samples

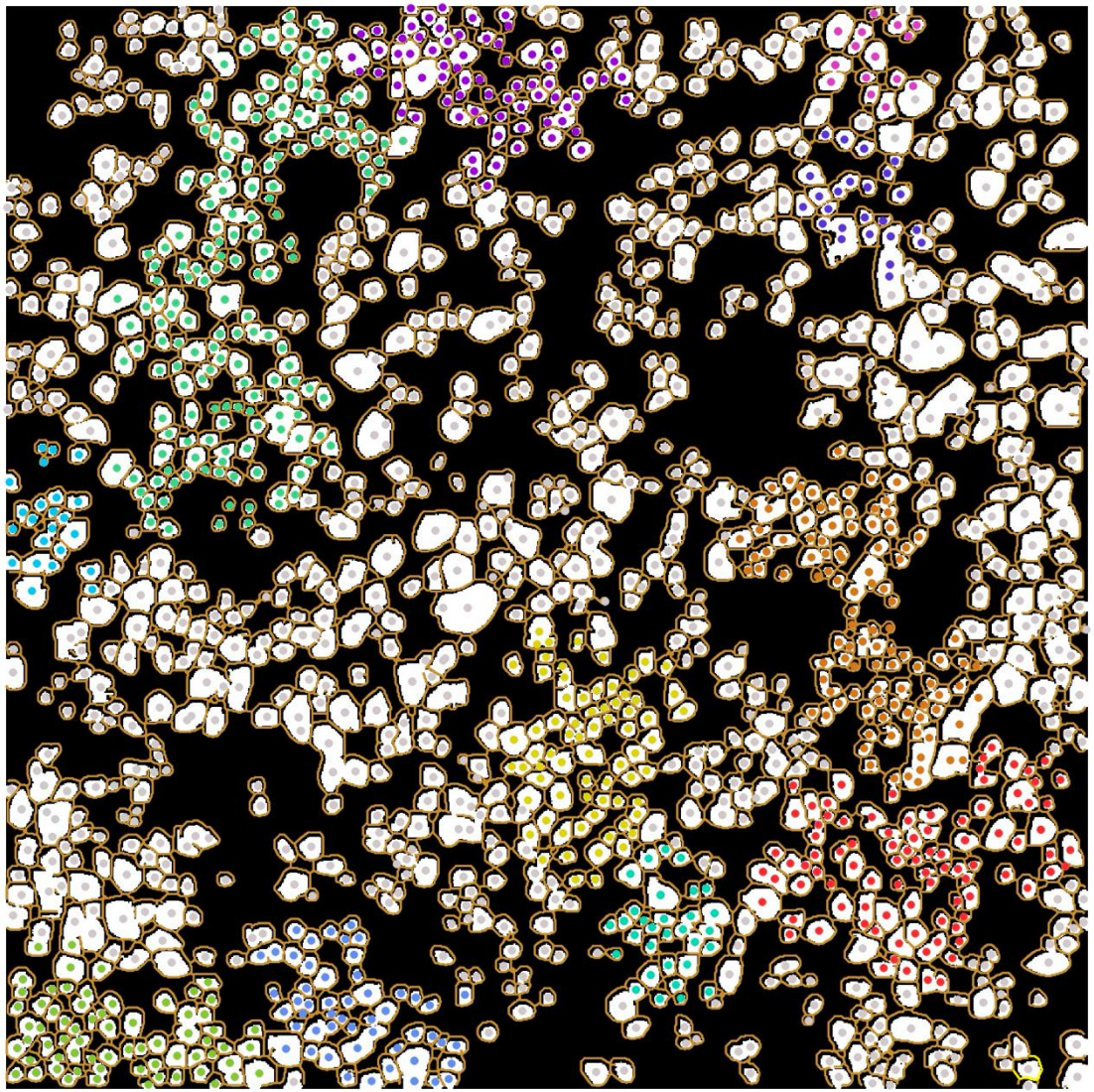
Clustering Centroids overlayed with IWPs

30 eps, 10 min samples

50m



50 eps, 20 min samples

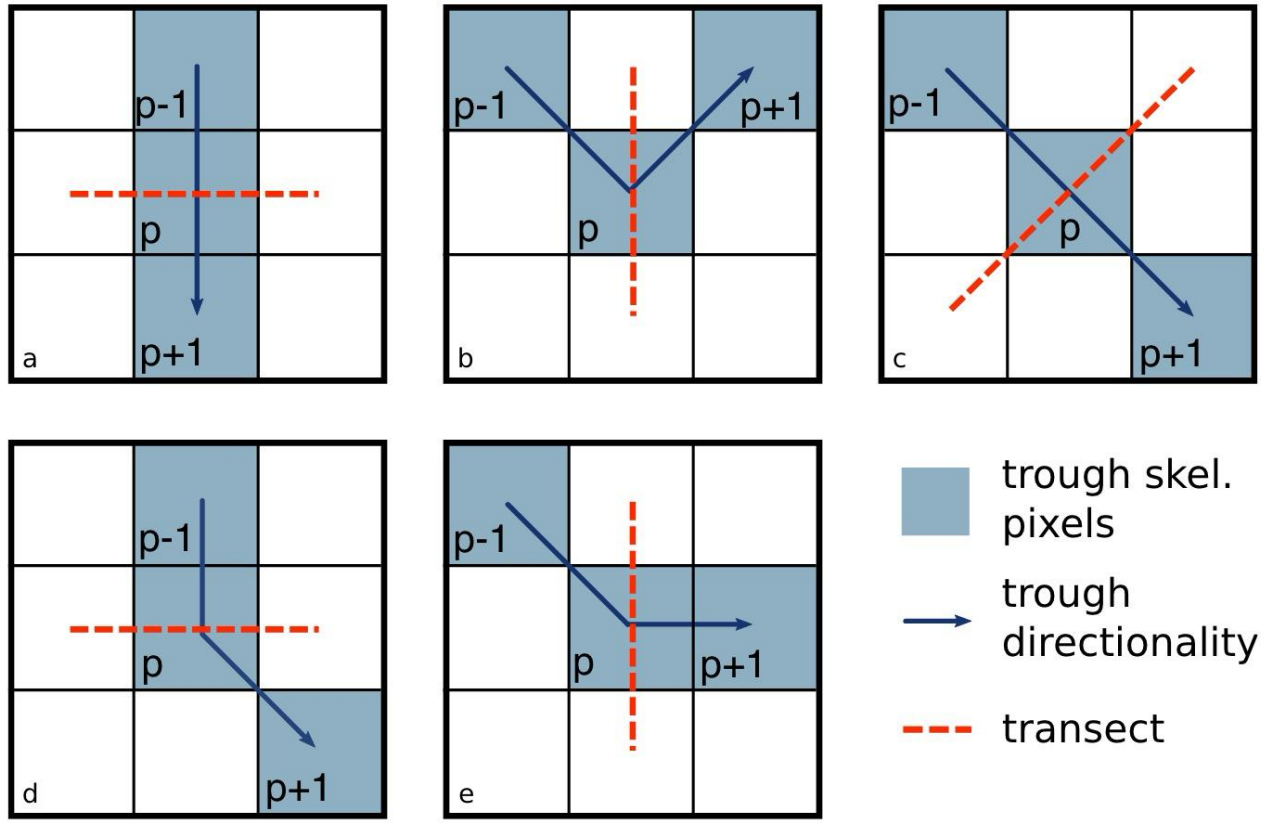


The paper: Trough depth & width

- Point clouds
- Rasterized DTM -> Detrend -> erosion -> binarization -> noise remove -> skeletonize
- Create transect and defined length of transect perpendicular to the trough's direction
- Transect profiles: 2-D points
- Gaussian curve fitting -> Maximum: depth, Full-Width-at-Half-Maximum: width
- Convert to graph structure: edges & nodes -> graph/network analysis

The paper: Trough depth & width

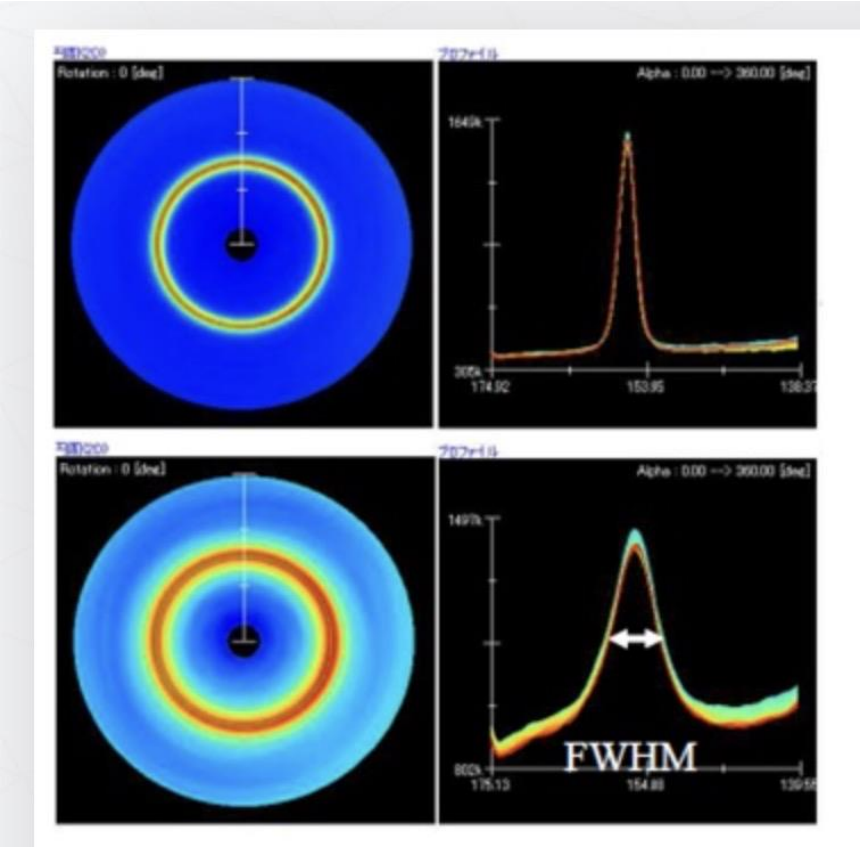
- Create transect and defined length of transect perpendicular to the trough's direction

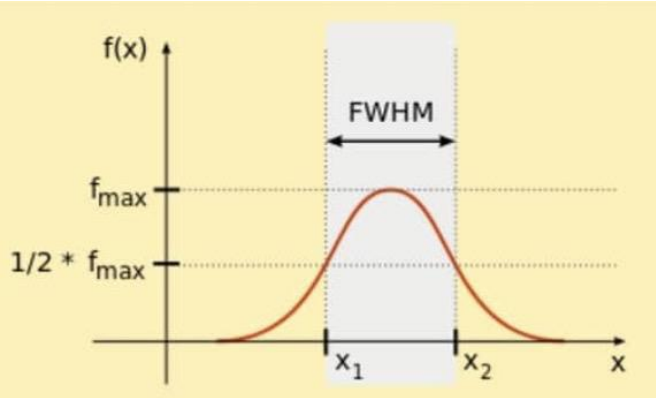


The paper: Trough depth & width

- Transect -> Transect profile -> elevation points

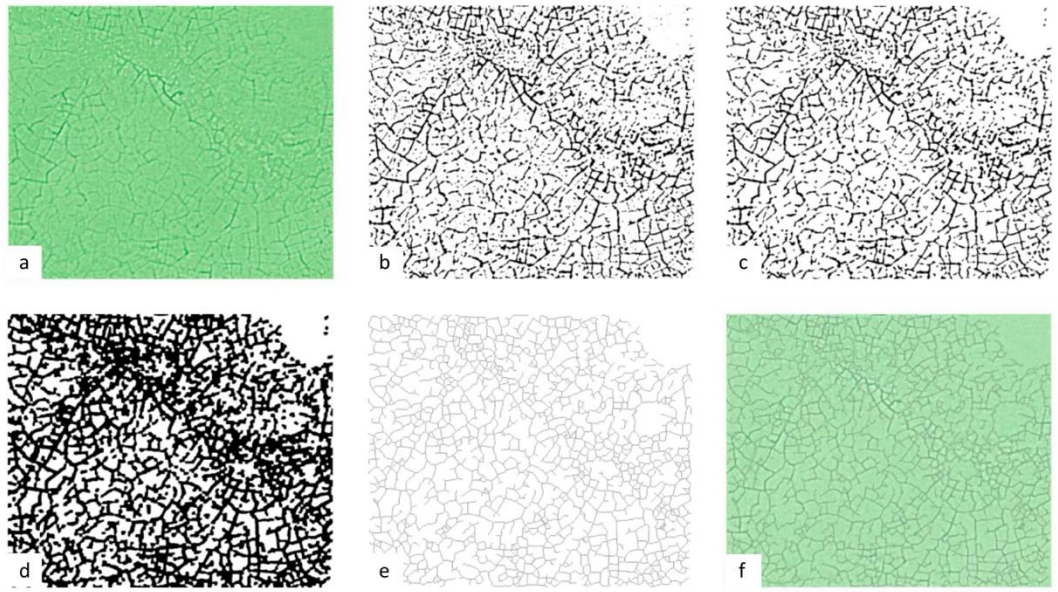
- Gaussian curve fitting: Maximum -> depth; Full-Width-at-Half-Maximum: width



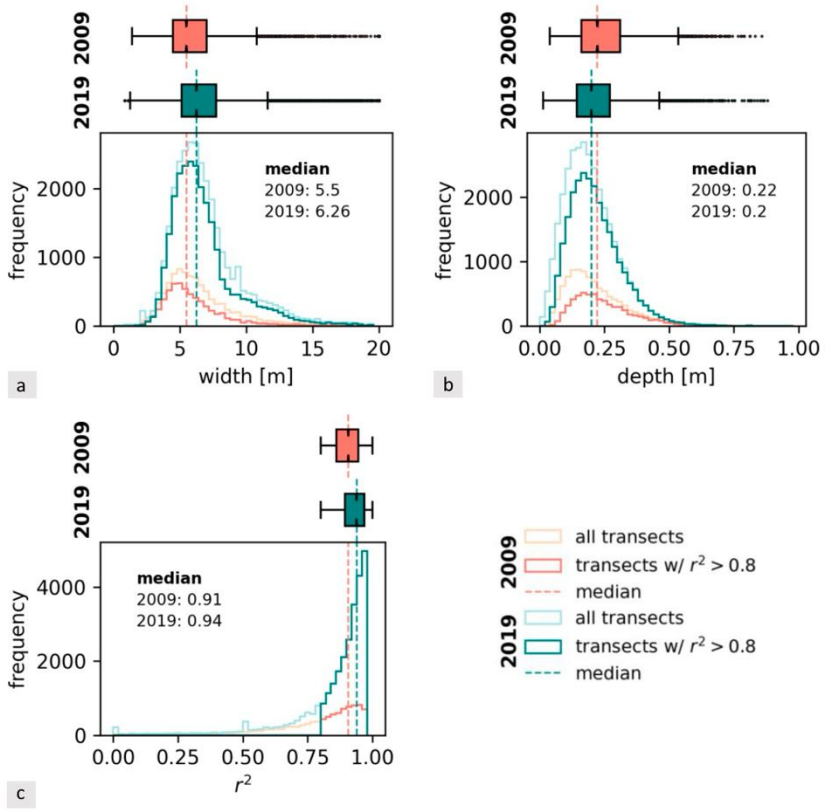


The paper: Trough height & width

- Convert to graph
- Graph/network analysis



re 3. Image pre-processing workflow for extracting a skeleton of troughs as the basis for the graph transform



Meeting 1/28/2025

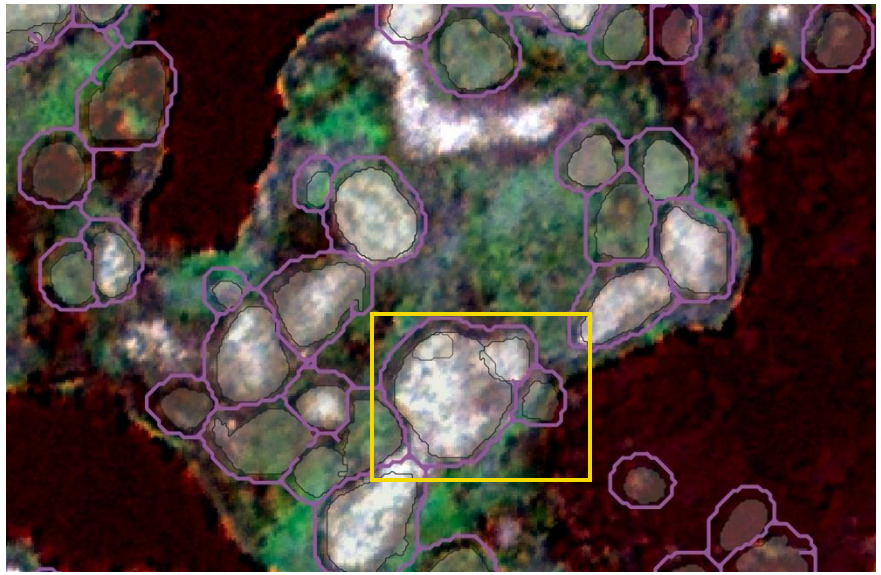
- Keep Inner trough network or not

- Deploy the code on datateam server to generate data products without downloading GPKG (Expected: 1.5h)
  - Find GPKG within a grid
  - Database connection AND operation
  - Draw statistics maps (Count, length sum, width sum, area sum, perimeter sum, coverage ratio, IWP density, area of centers/trough network, count of IWP types) Class 1: low-center

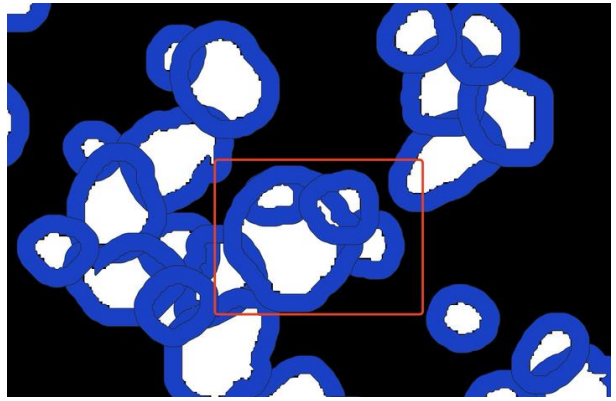
- Trough width

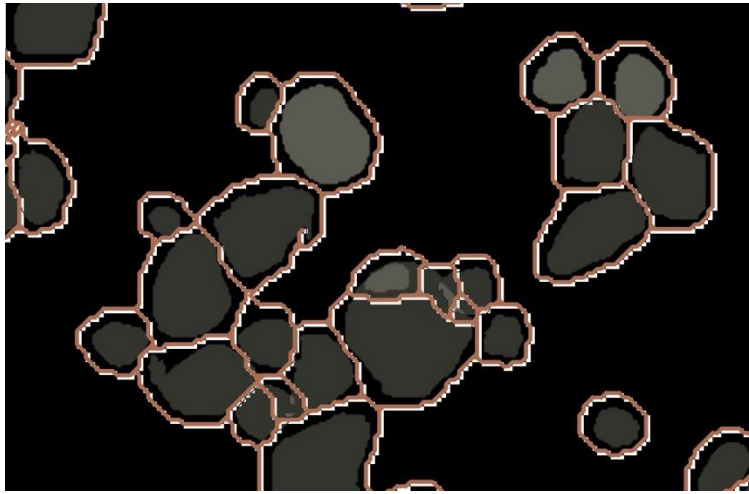
Identify trough area: Keep the inner trough or not

Original IWPs

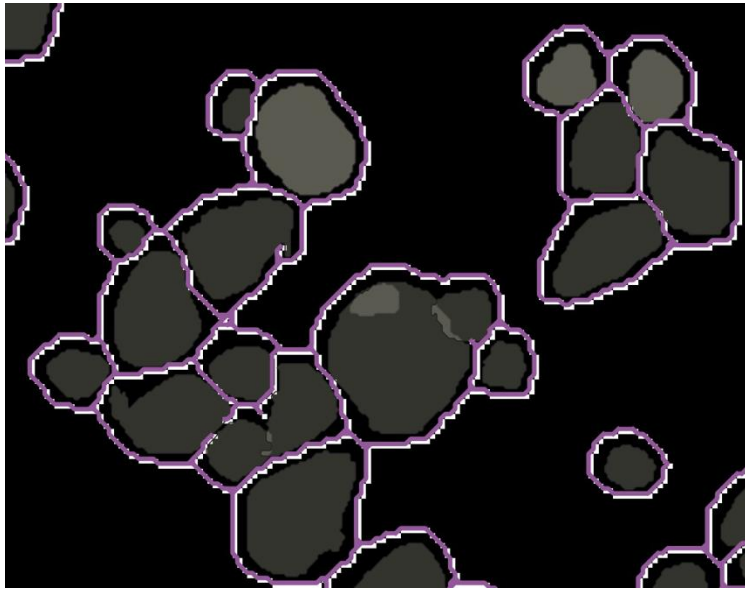


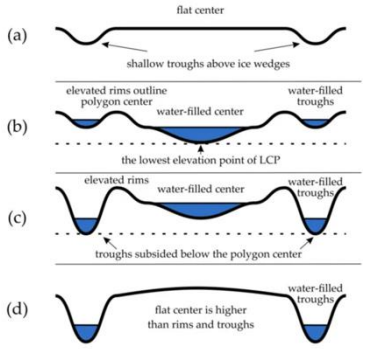
Keep





Ignore





Schematic profiles of mapped polygon types

Trough width: DBScan (Density-Based Spatial Clustering of Applications with Noise)

- **eps**: Maximum distance to distinguish clusters
- **min_samples**: Minimum samples in a neighborhood of a core point

High min_samples -> denser clusters.

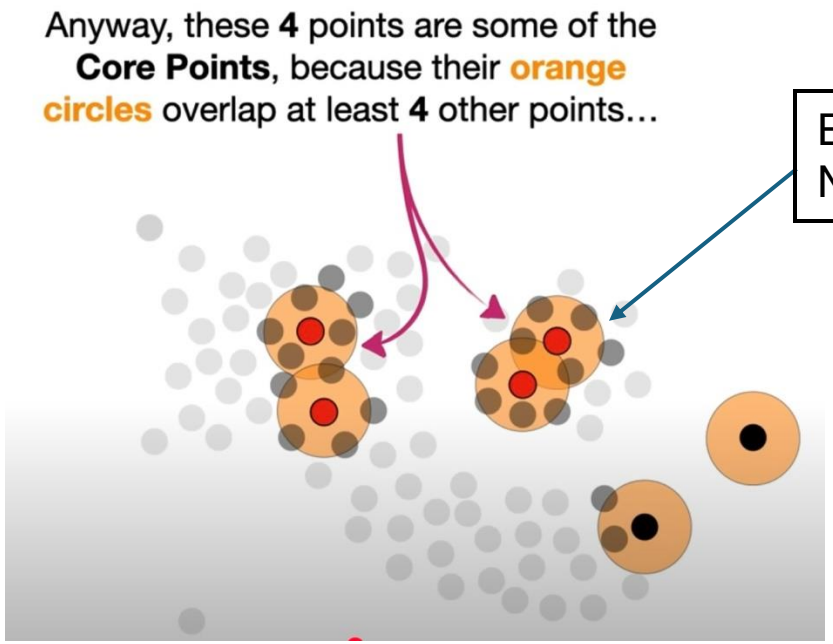
Low min_samples -> sparse clusters.

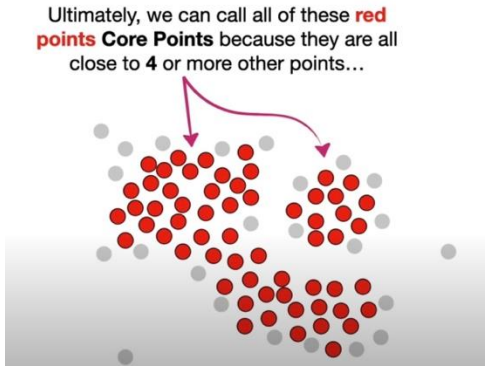
- **Metrics:** The metric to use when calculating distance between instances in a feature array
- **fit_predict**(*X*, *y=None*, *sample_weight=None*)

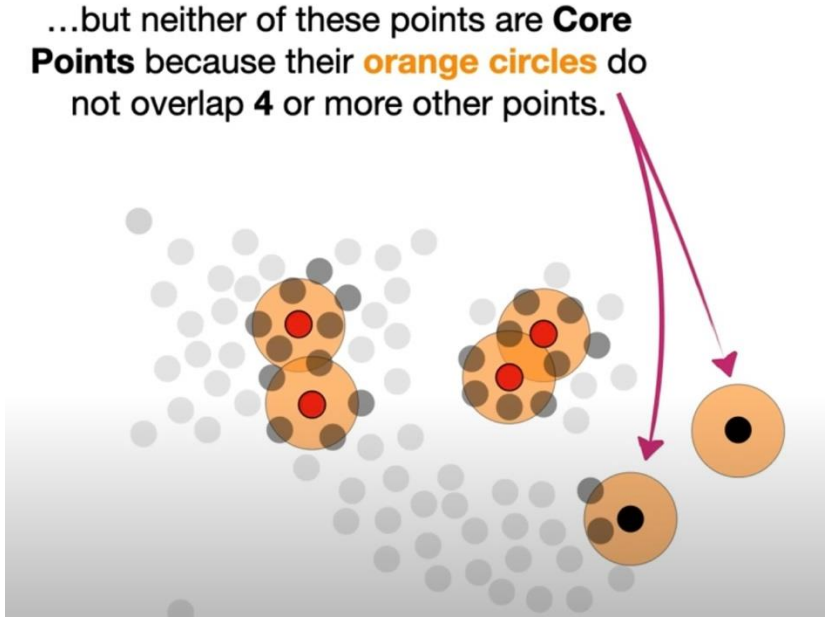
Trough width: DBScan

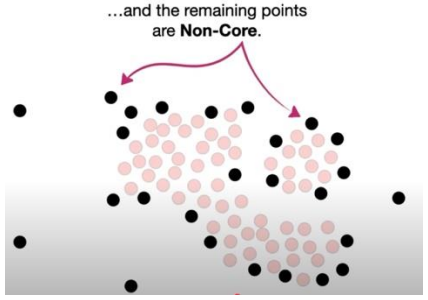
Step 1: Find Core points & Non-core points

- Core points -> Points that have minimum number of samples in a neighbor
- Non-core points: Points that have less than min number of samples in a neighbor





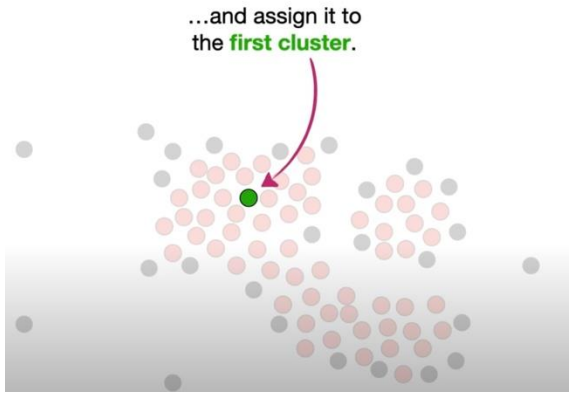


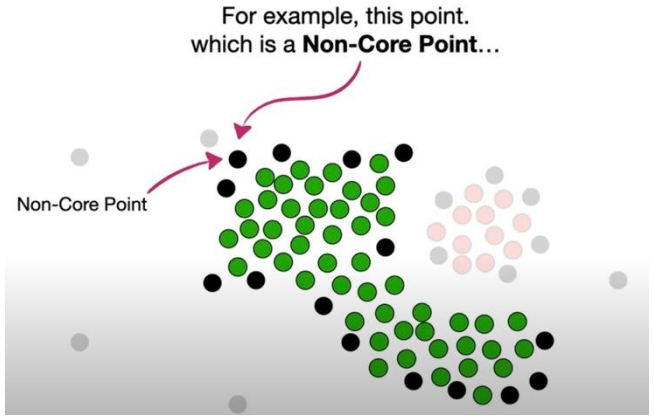


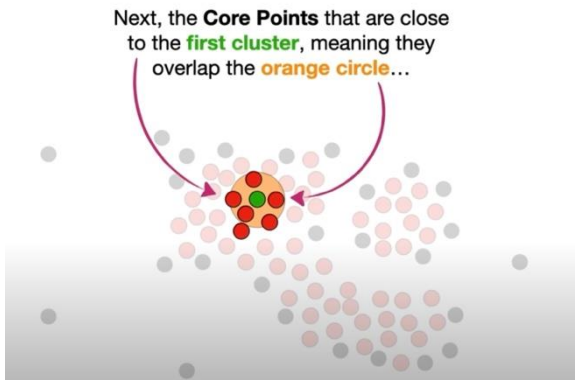
Trough width: DBScan

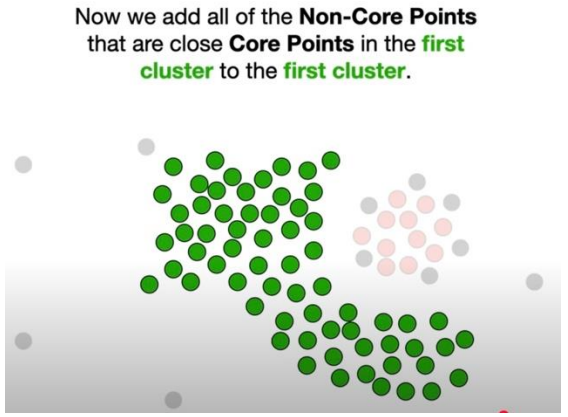
Step 2: Start clustering

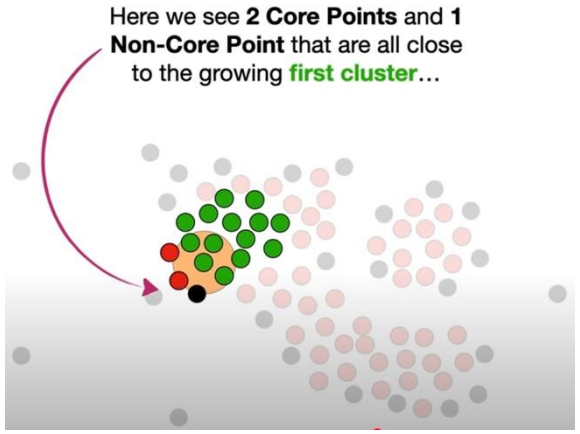
- Find a core point -> first cluster
- Expand the first cluster -> expand along with core points
- End -> stop at and include the non-core points

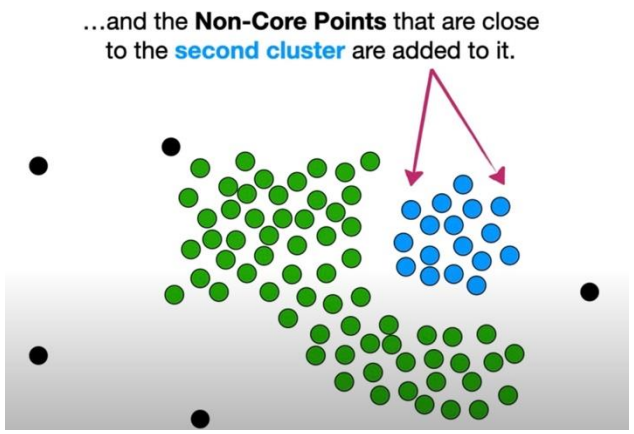








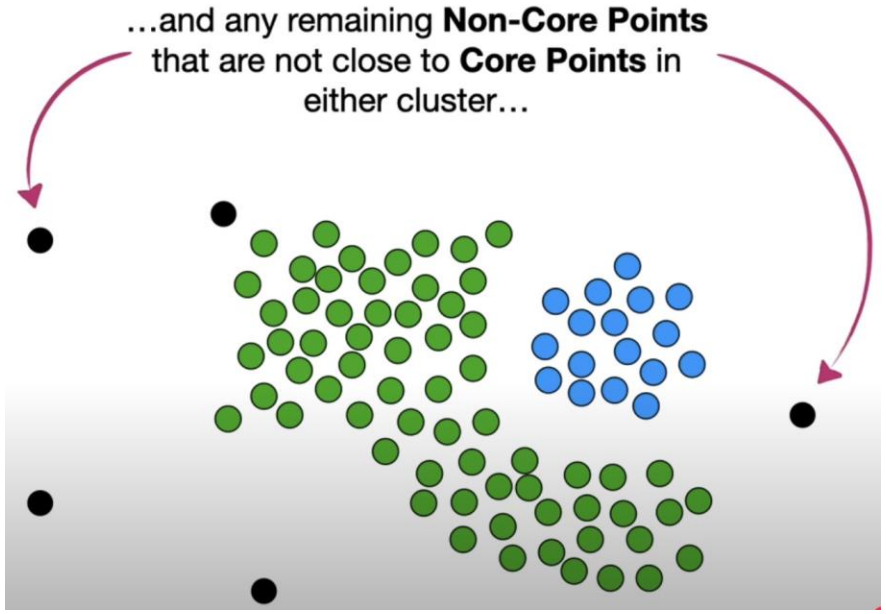




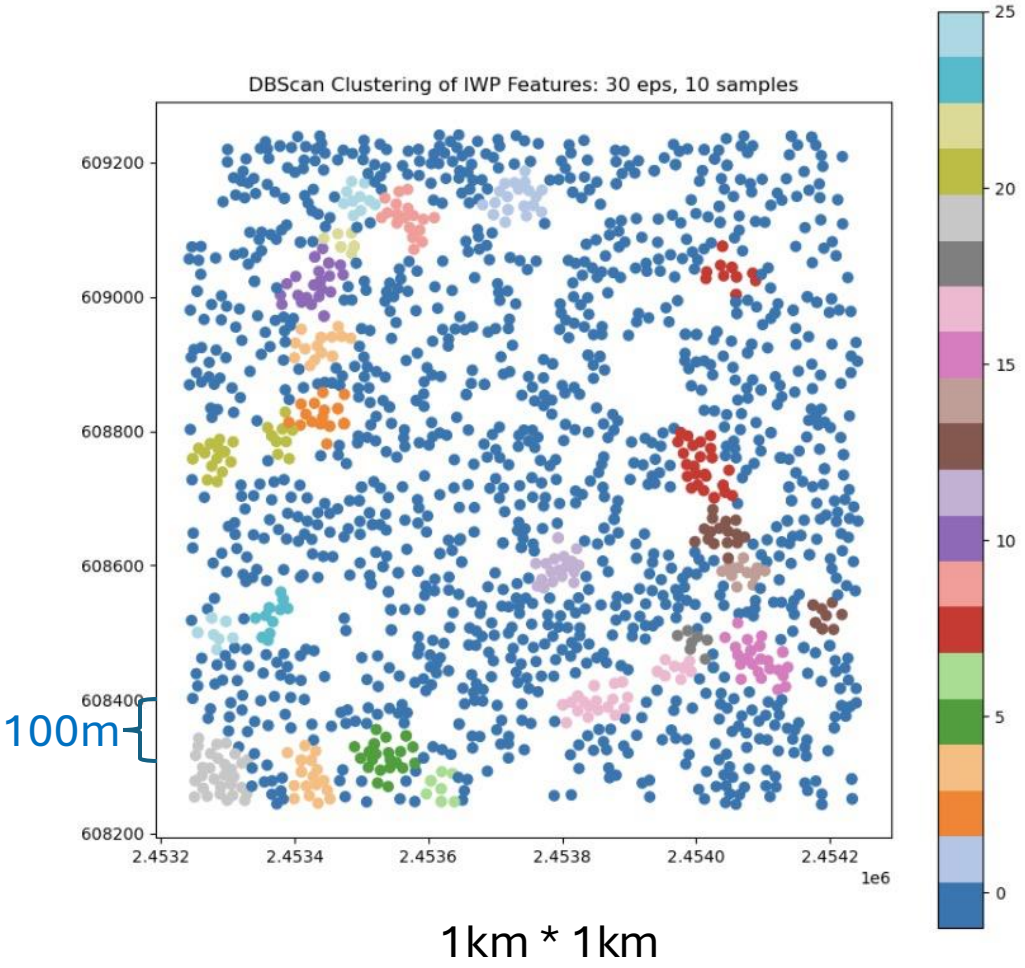
Trough width: DBScan

Step 3: Clusters & Outliers

- Outliers: Points that do not belong to any clusters

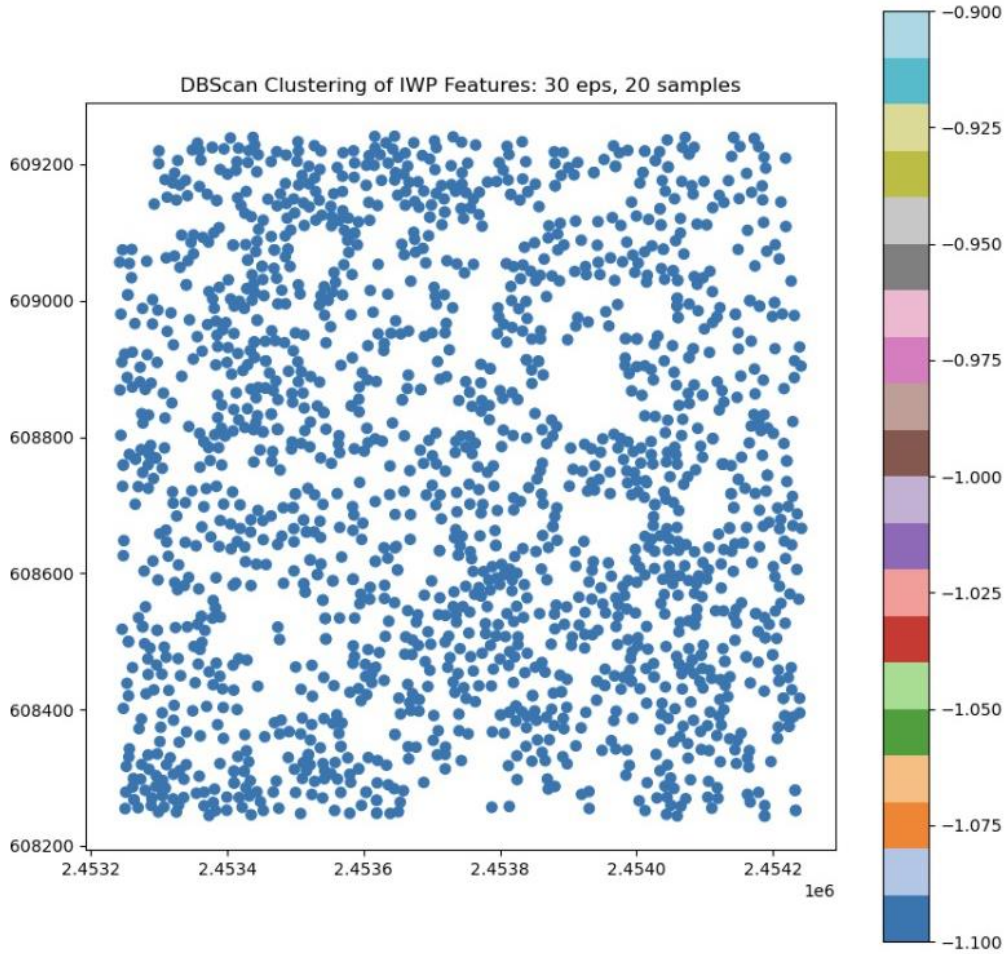


Trough width: clustering of IWP centroids – eps: 30m

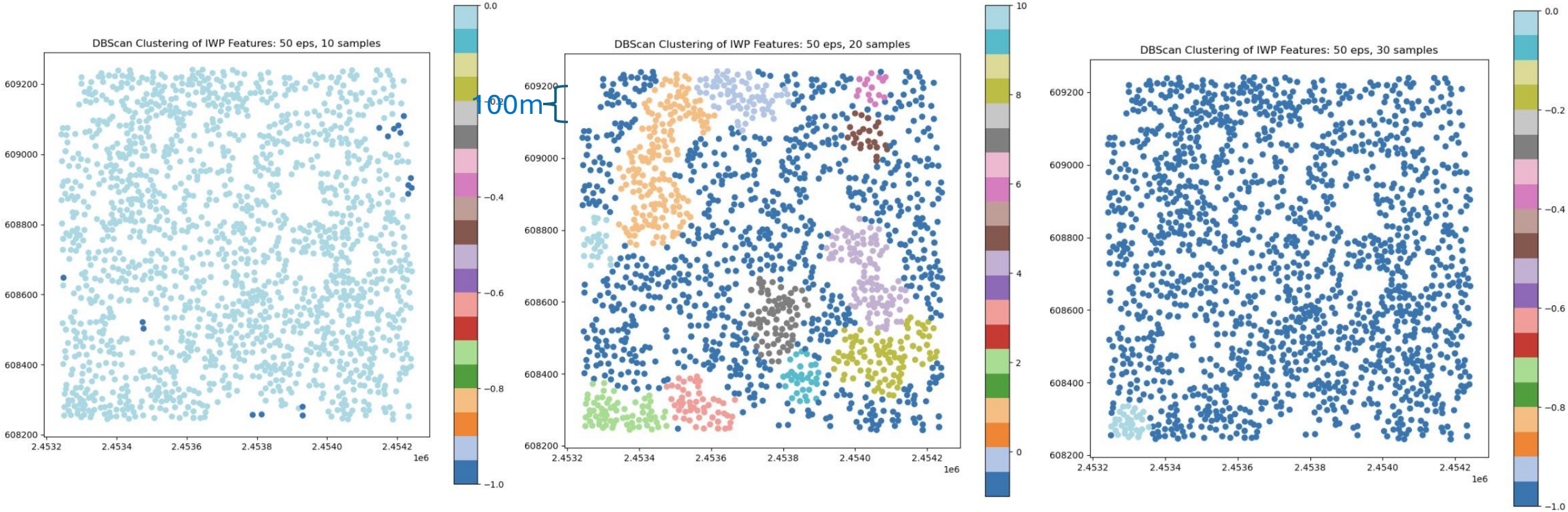


100m

1km * 1km

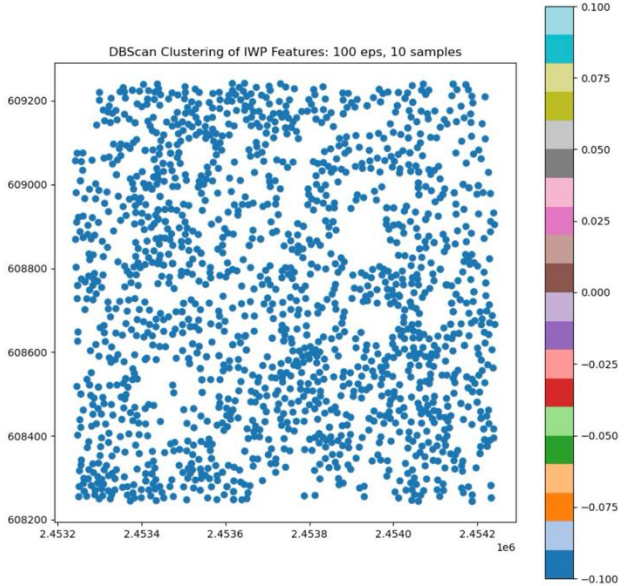


Trough width: clustering of IWP centroids – eps: 50m

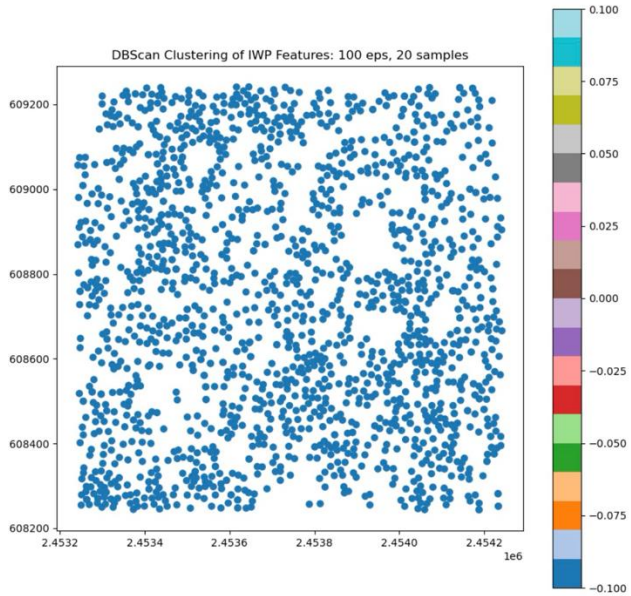


High min_samples -> dense clusters.
Low min_samples -> sparse clusters.

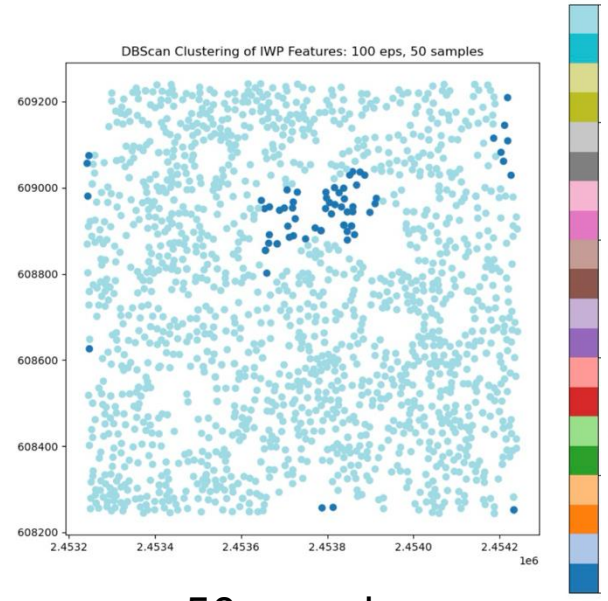
Trough width: clustering of IWP centroids – eps:100m



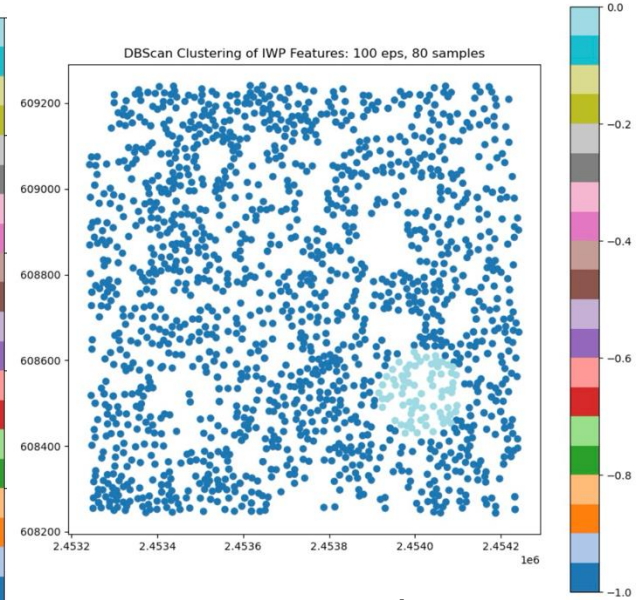
10 samples



20 samples



50 samples

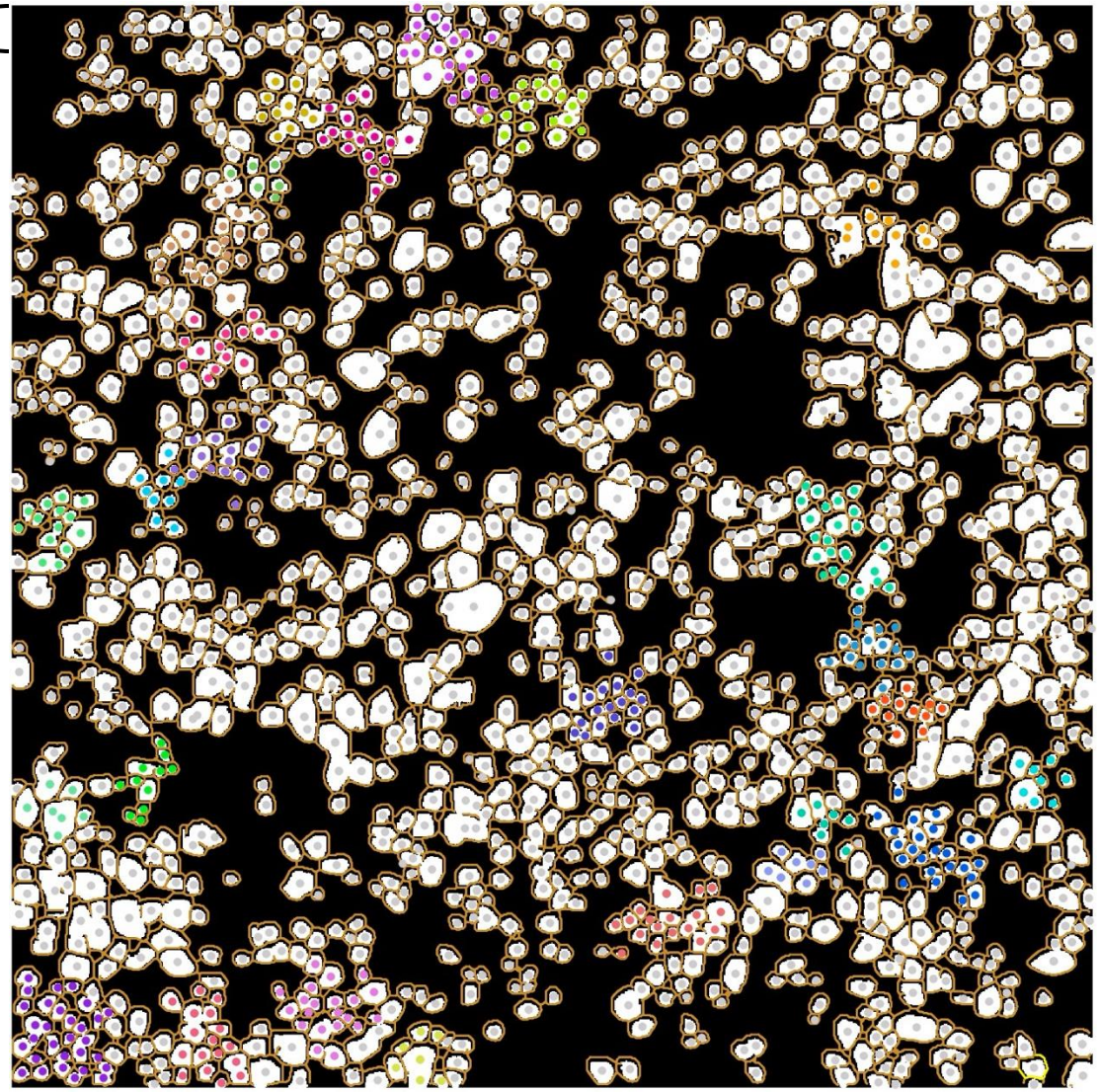


80 samples

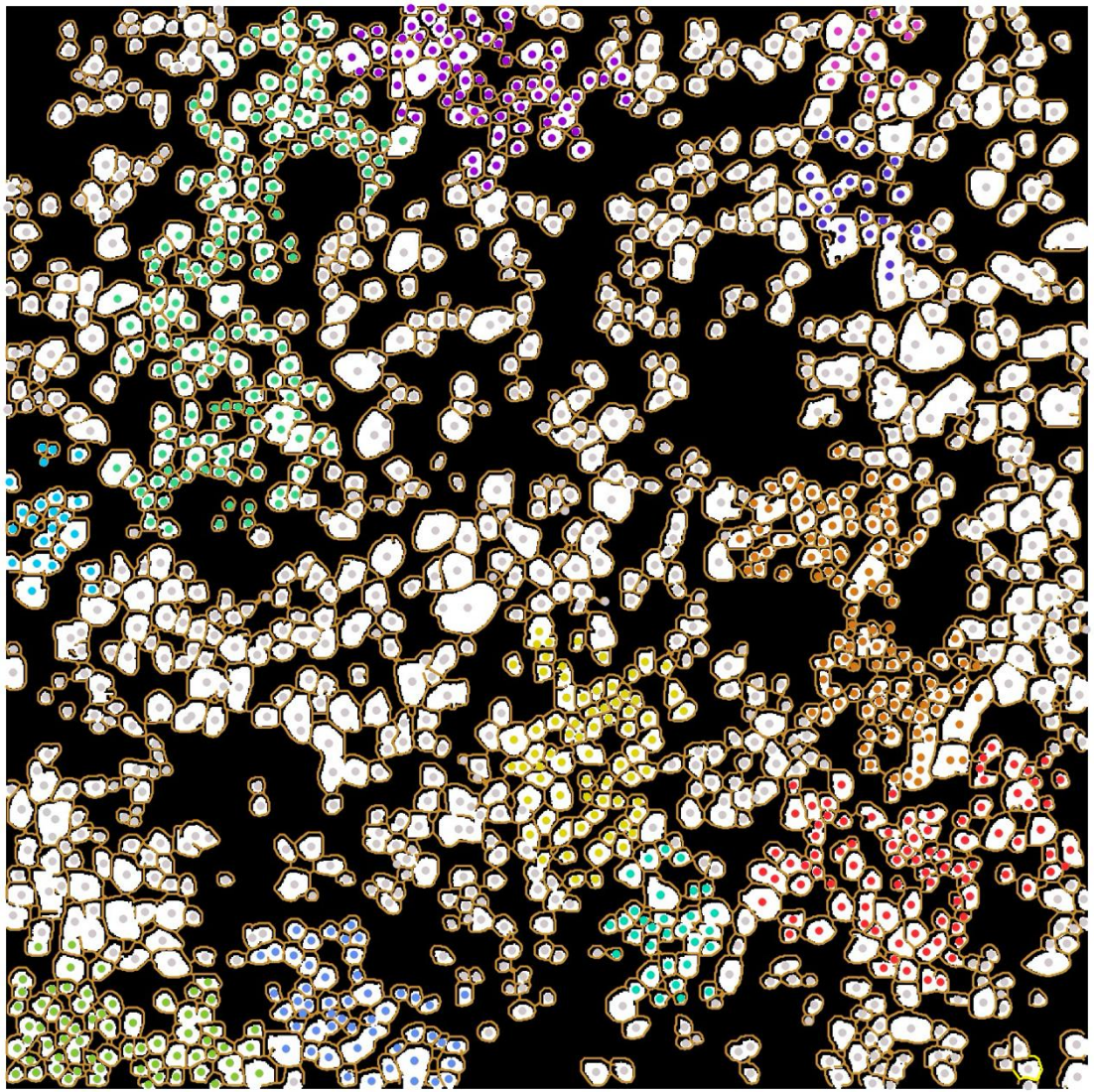
Clustering Centroids overlayed with IWPs

30 eps, 10 min samples

50m



50 eps, 20 min samples



Meeting 2/4/2025

- Data Transfer between data server and database server
- The width of ice wedge

Data transfer

- Data: datateam. Database: CICIlab

1. Create a database on datateam server
2. Mount data for CICIlab’s docker

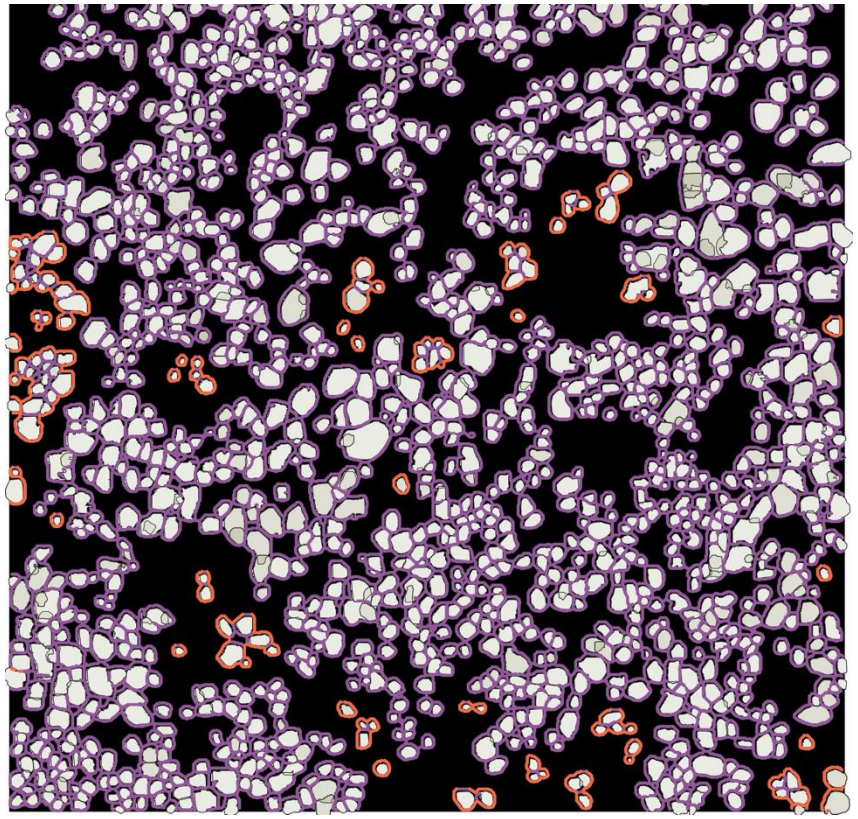
Width of Ice Wedge

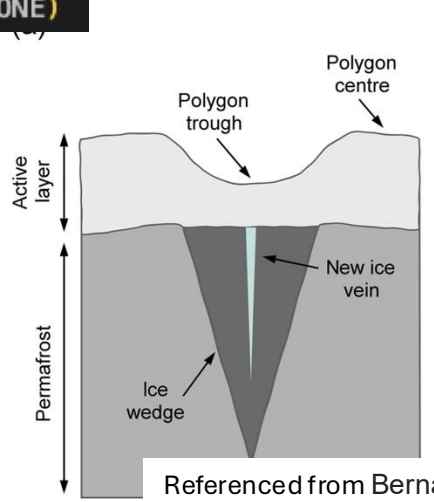
- EXTERNEL

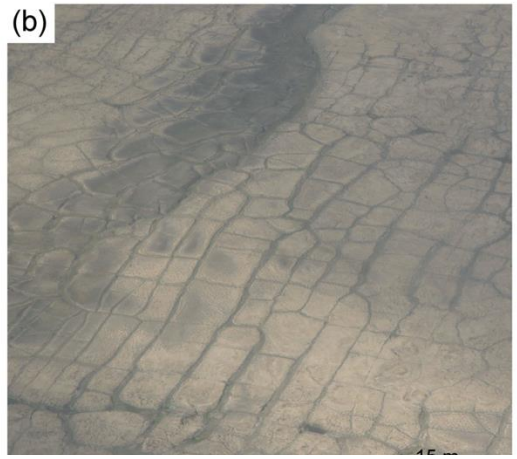
```
contours_ext, _ = cv2.findContours(skeleton_data, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

- TREE

```
contours_all, _ = cv2.findContours(skeleton_data, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

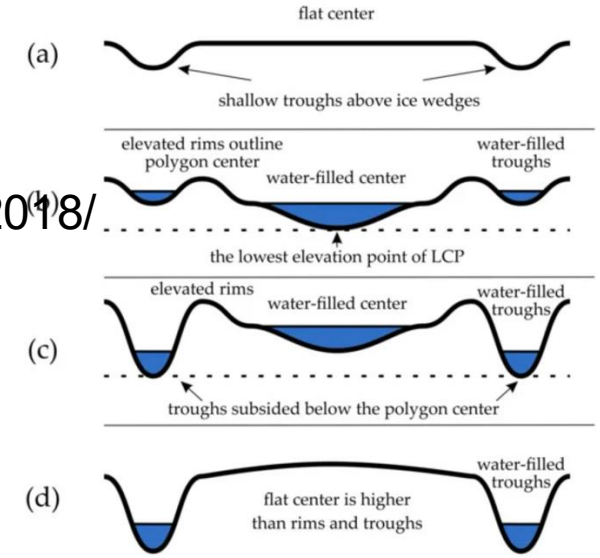






Referenced from Bernard-Grand'Maison, C., & Pollard, W. (2018).

Validate using images:
https://tc.copernicus.org/articles/12/3589/2018/



Opencv library:
https://docs.opencv.org/4.x/d9/d8b/tutorial_py_contours_hierarchy.html

Statistics: on Canada Bank island

- IWP Count
- IWP Density
- IWP coverage sum
- Perimeter sum
- Width sum
- Length sum
- Coverage ratio
- Length of IW
- LC vs others

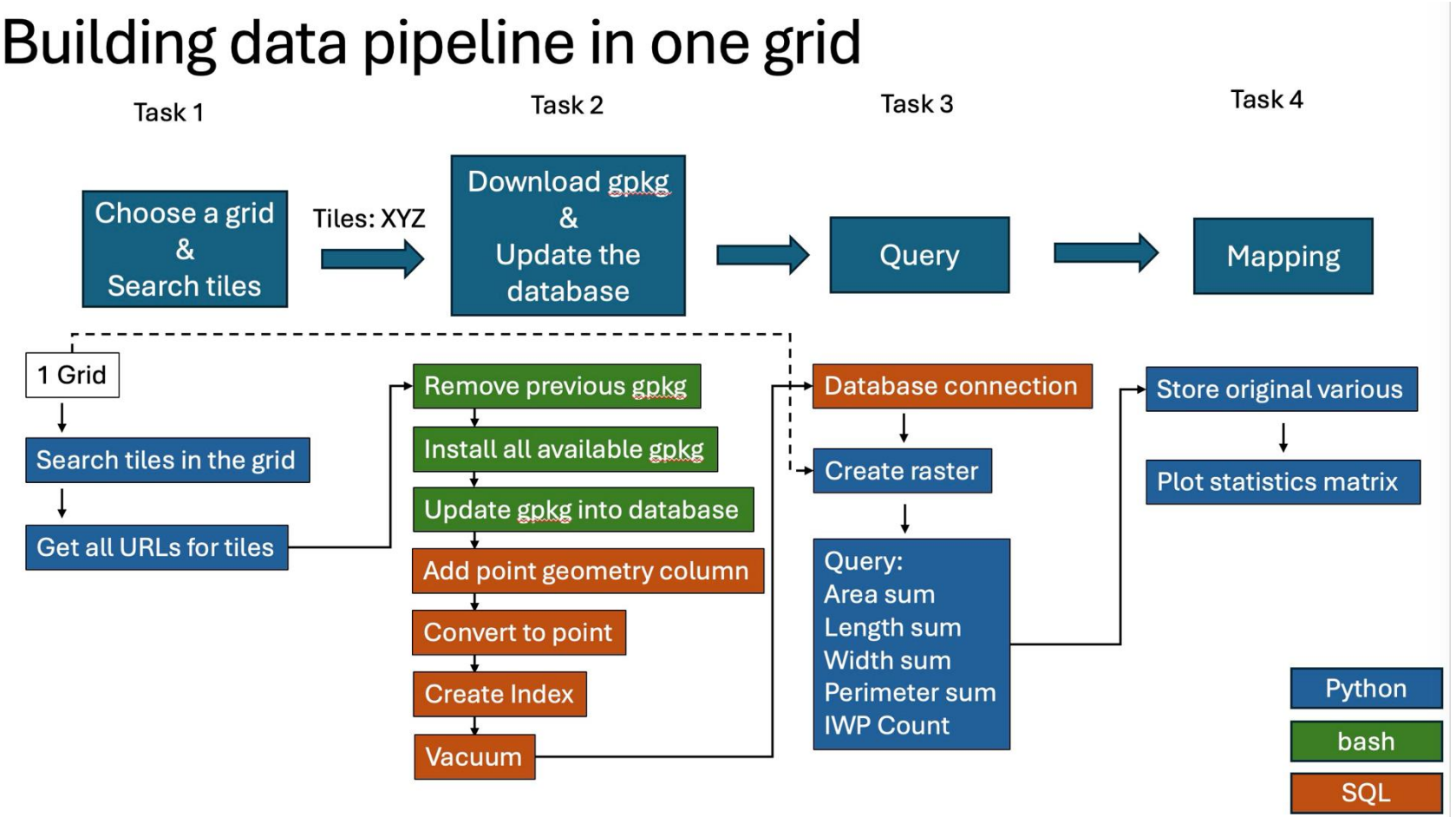
Scientific questions: Develop or degrade

Meeting 2/11/2025

- Production of data statistics in batch
  - Simplify the data pipeline
- Identify IWP network?
- Length of IW network at pixel level
- Width of IW (Trough width) at pixel level

Production of data statistics in batch: Simplify data pipeline

Original data pipeline



Production of data statistics in batch: Simplify data pipeline

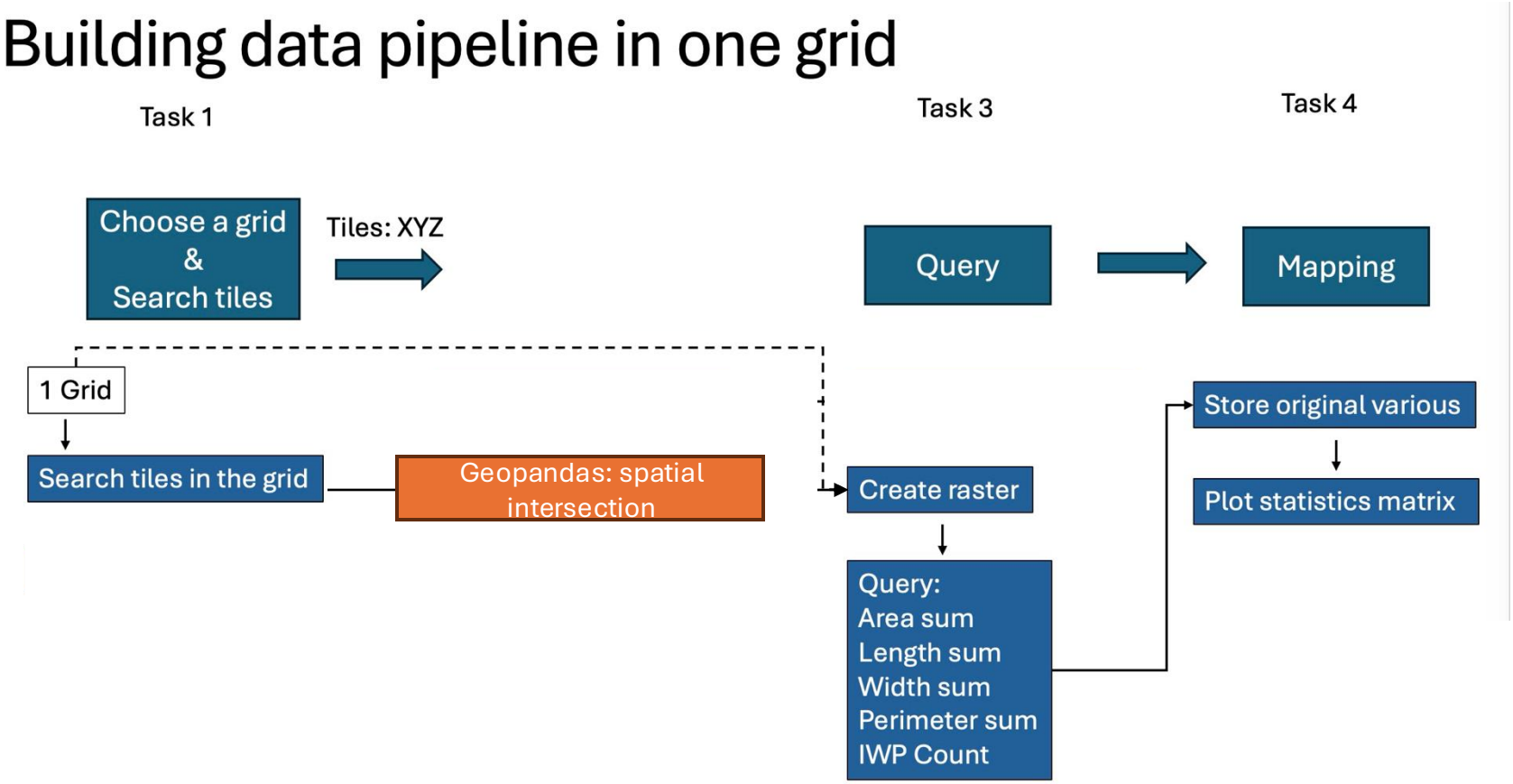
Current data pipeline

Advantage:

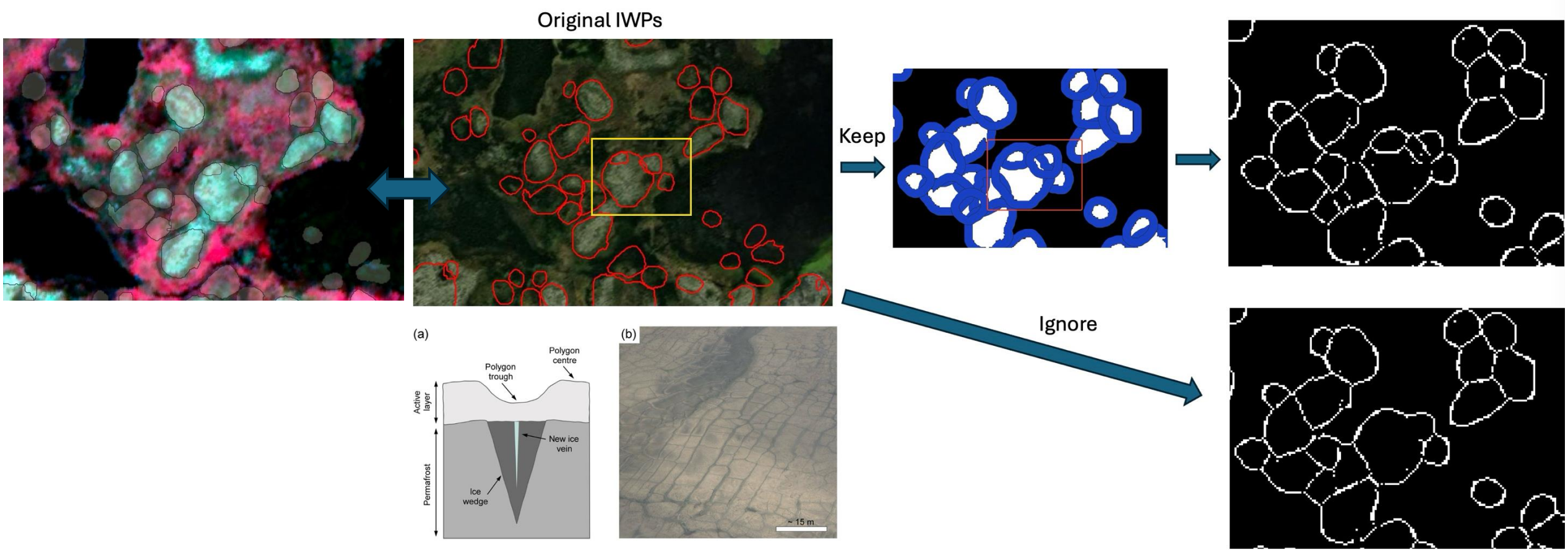
1) Delete interactions with database
2) Reduce the time to load data into database (1h+, depending on IWP count)

Disadvantage:

1) Increase time in iterating gpkg outside of database (originally, 0.5~1h, now it takes more than 10h+)



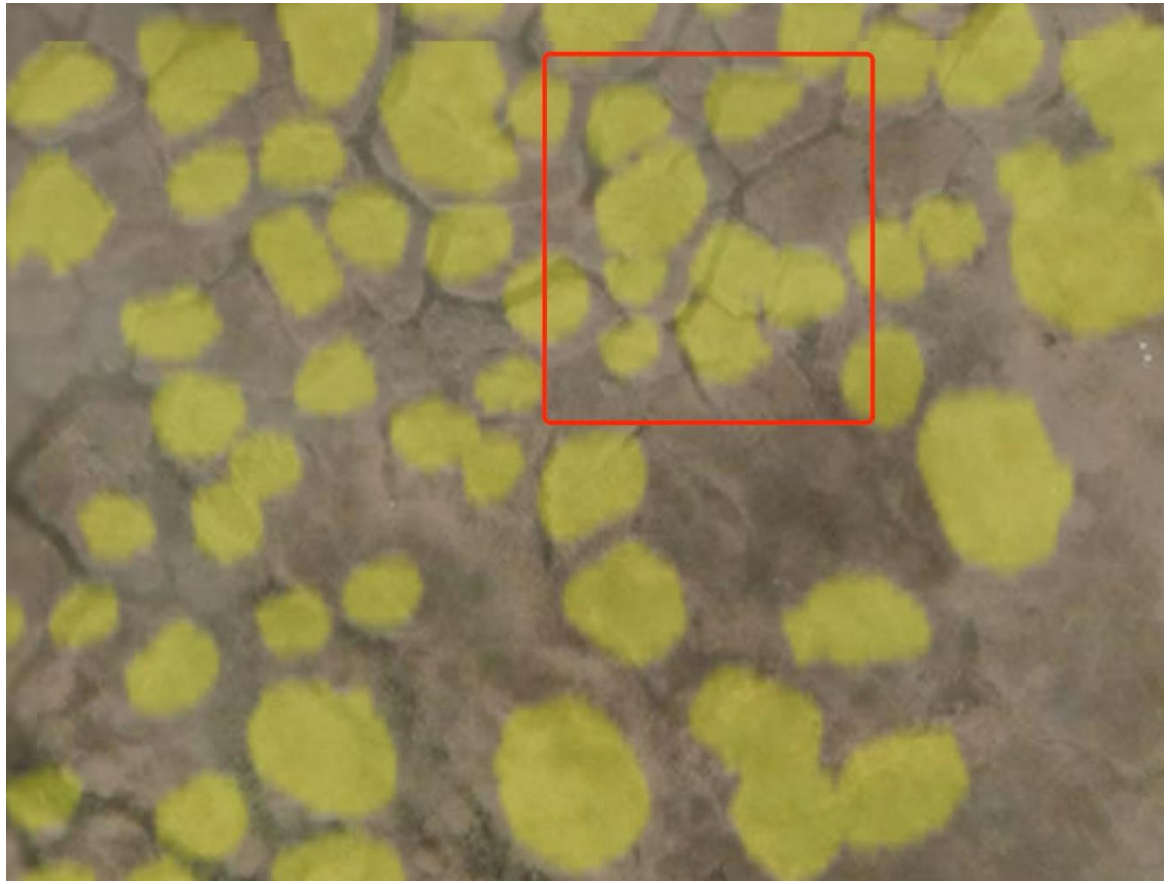
Identify IWP network



Identify IWP network

- Clearer IWPs on basemap:

https://arcticdata.io/catalog/portals/permafrost?lt=71.3087063039577 8&ln=-156.625009438525&ht=9095.857861049131&hd=3.053332494204976e-13&p=-89.9979991957138&r=0&el=iwp%2Cosm%2Cahri



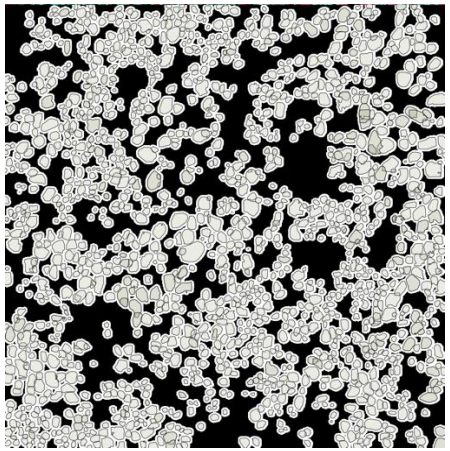
Area of inner IW network

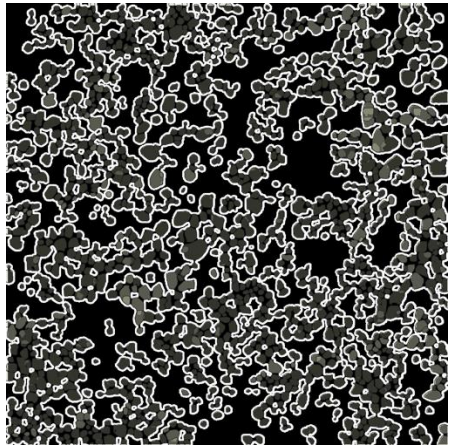
Dilated IWPs

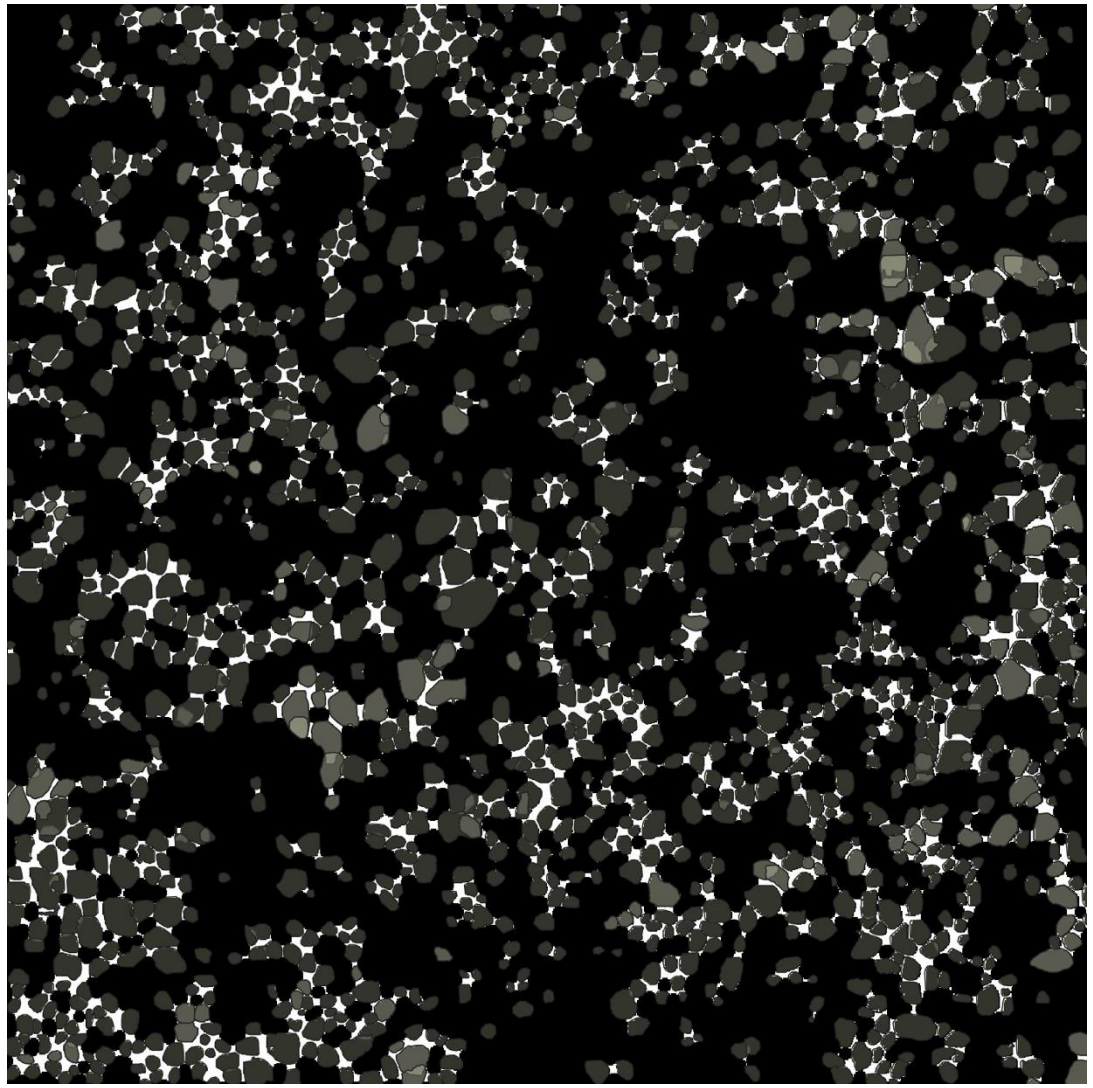
Do an inner buffer on dilated IWPs

External buffer

Area of inner IW network



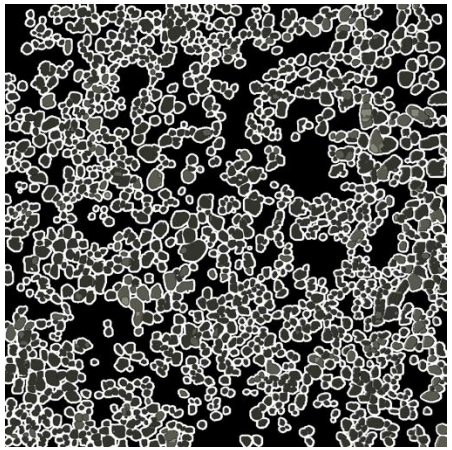




Subtract IWPs

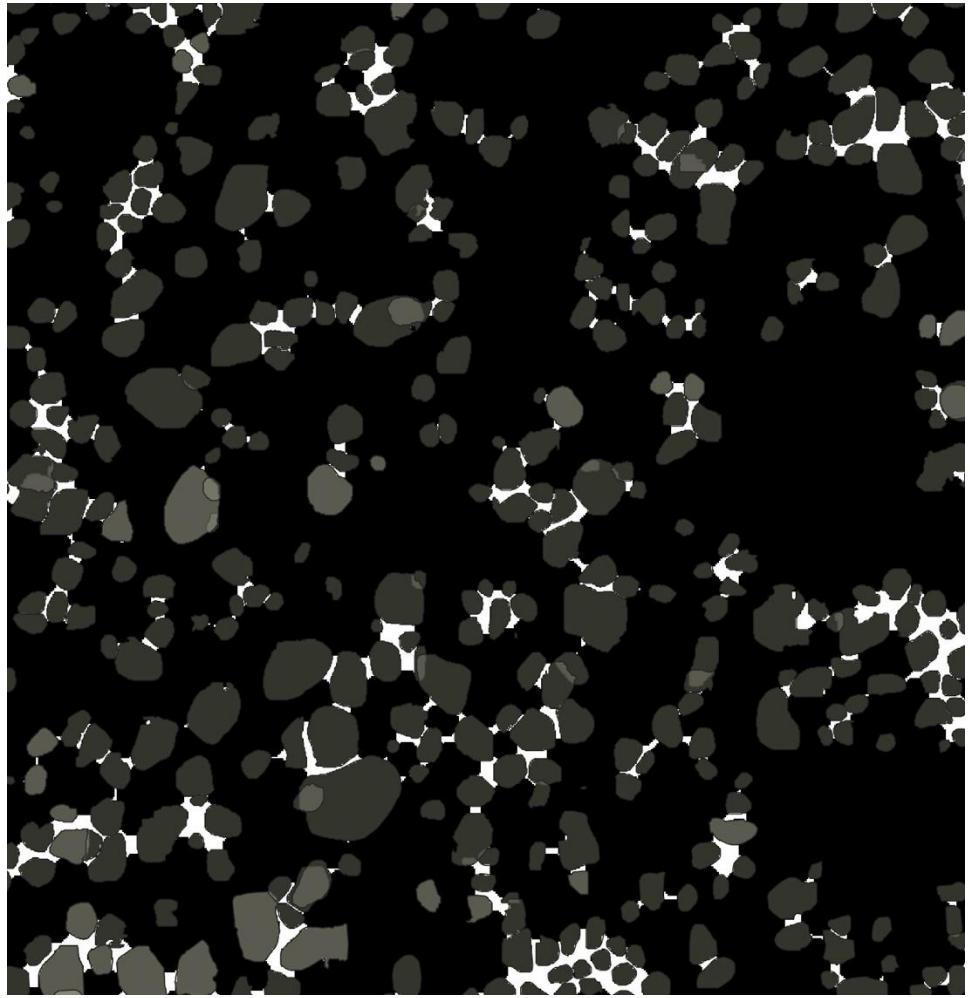
All buffer

subtract



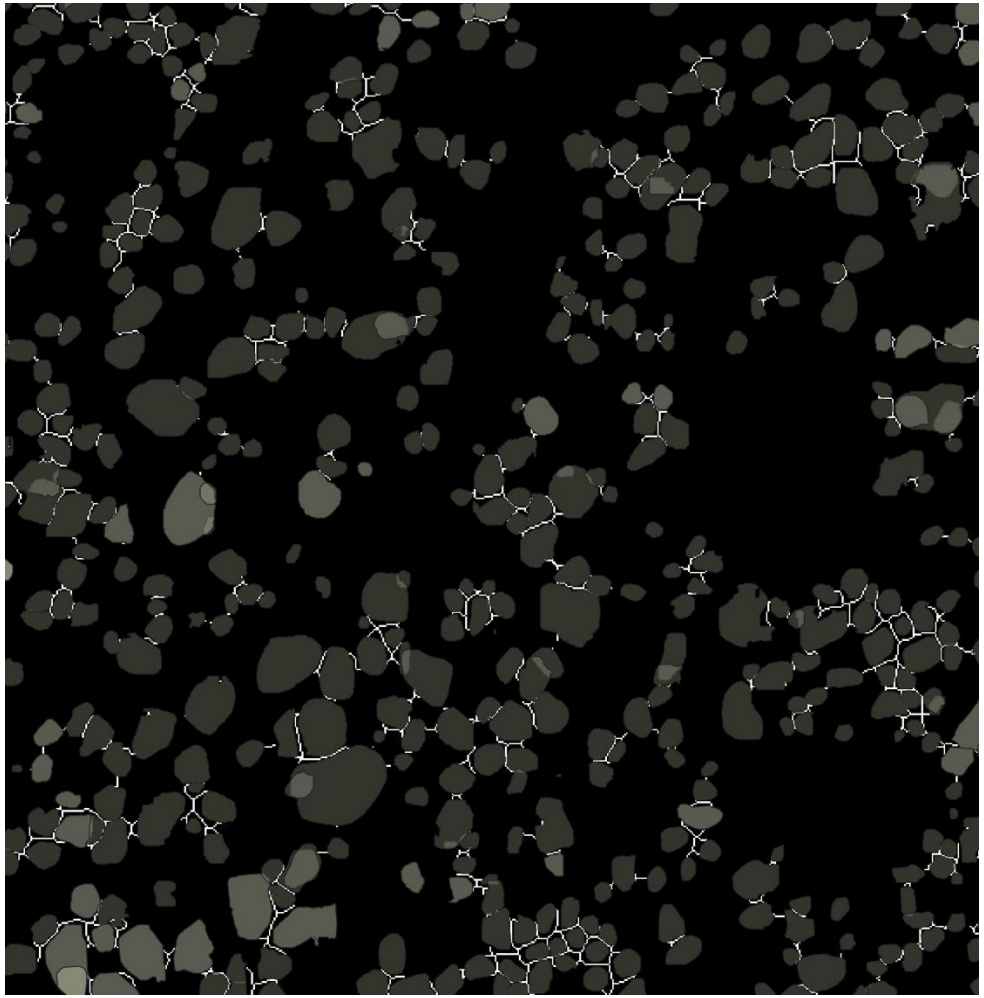
Length / Width of IW Network

Area of IW network



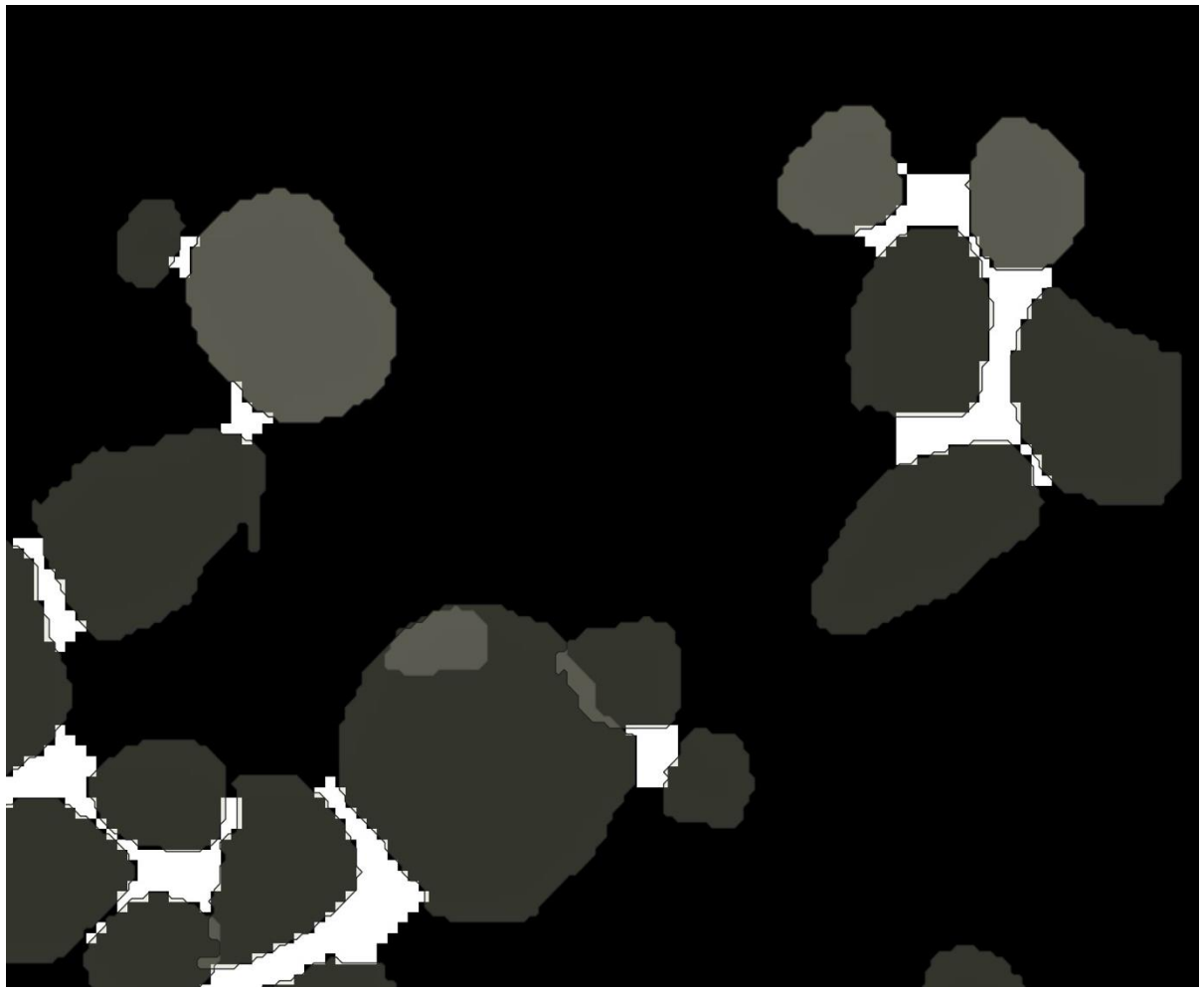
Skeletonize

(Length) of inner IW network

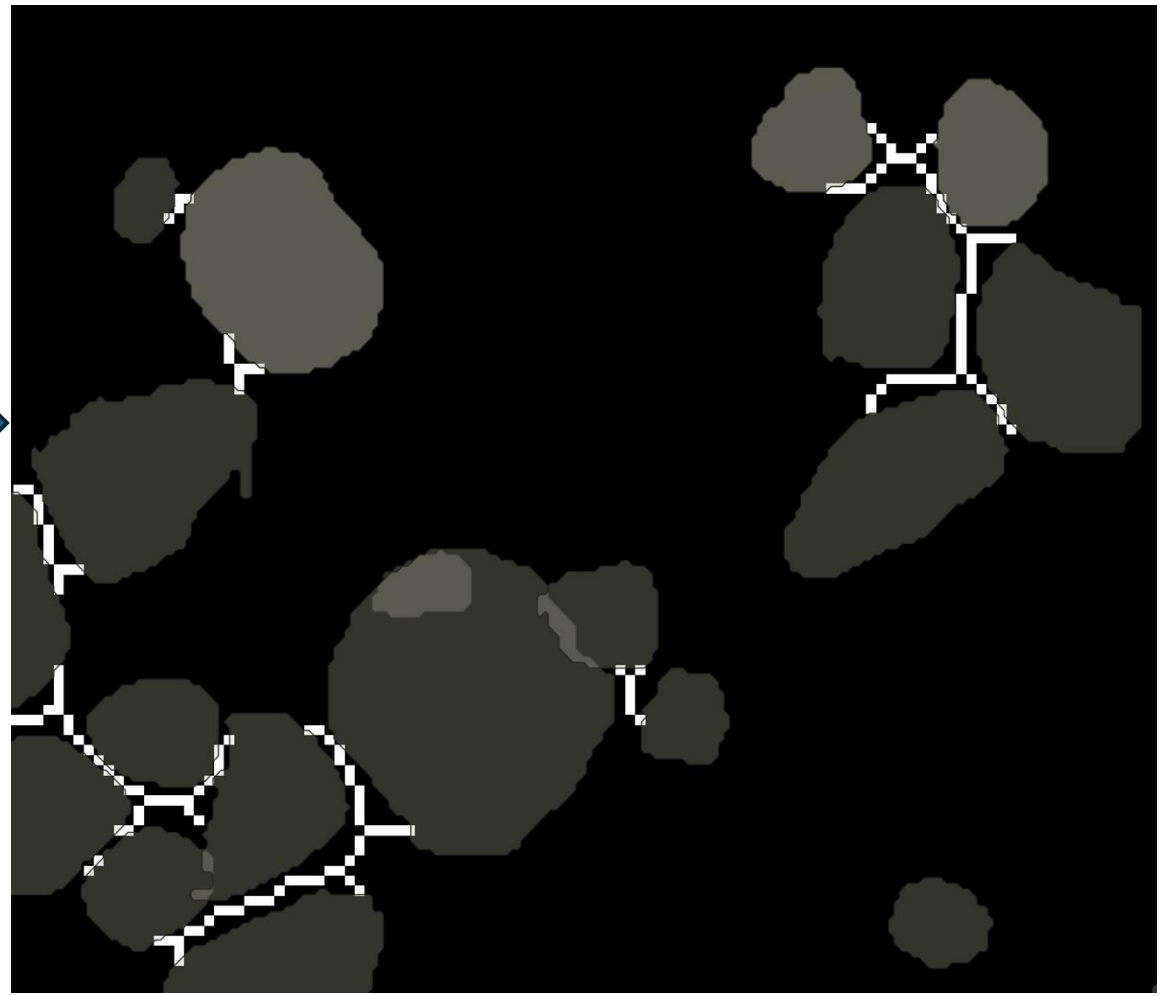


Length / Width of IW Network

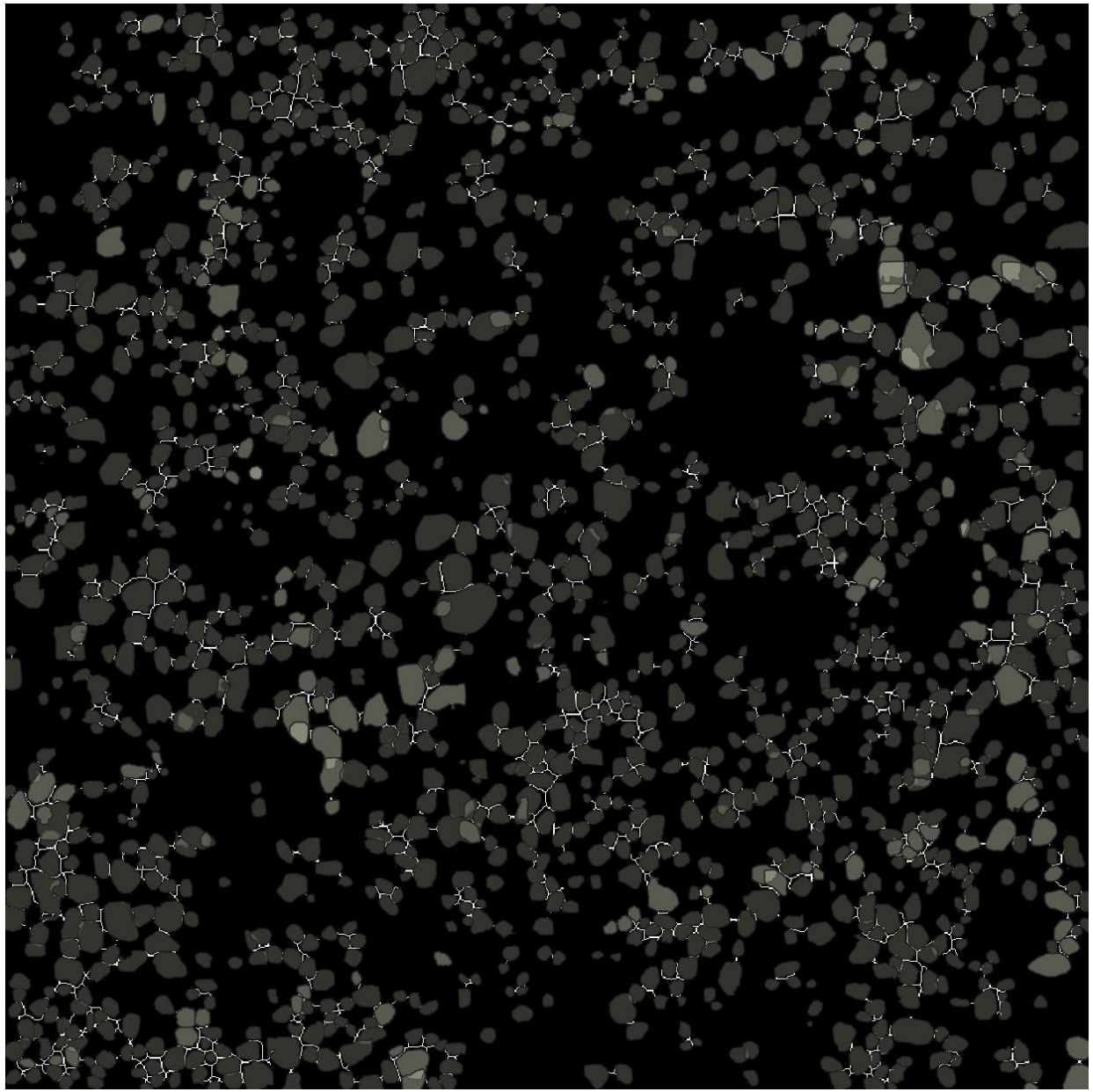
Area of IW network



(Length) of inner IW network



Length / Width of IW Network

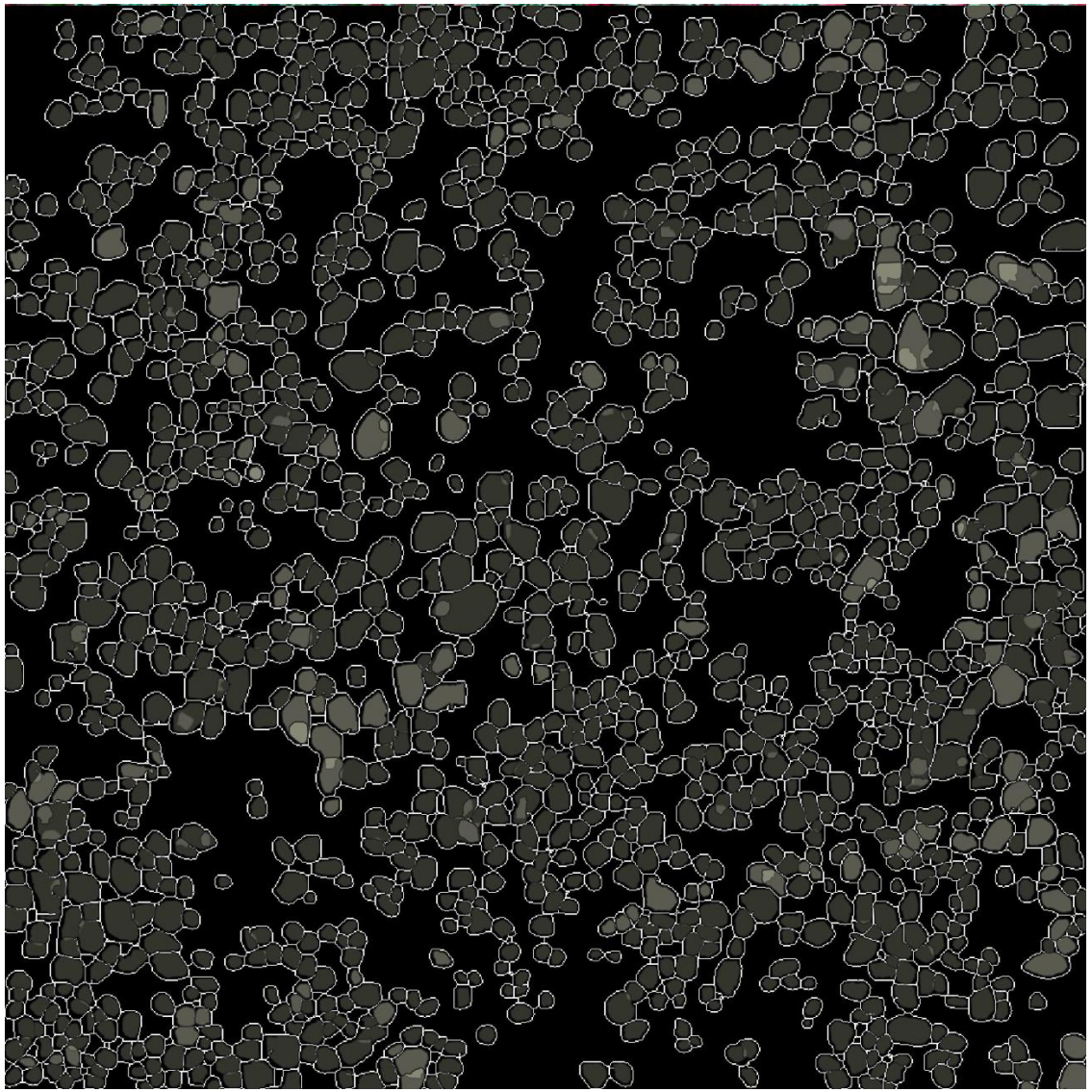


Within this pixel:

Length: 15814 m

Width: 8.73030226381687 m

Length of IW network

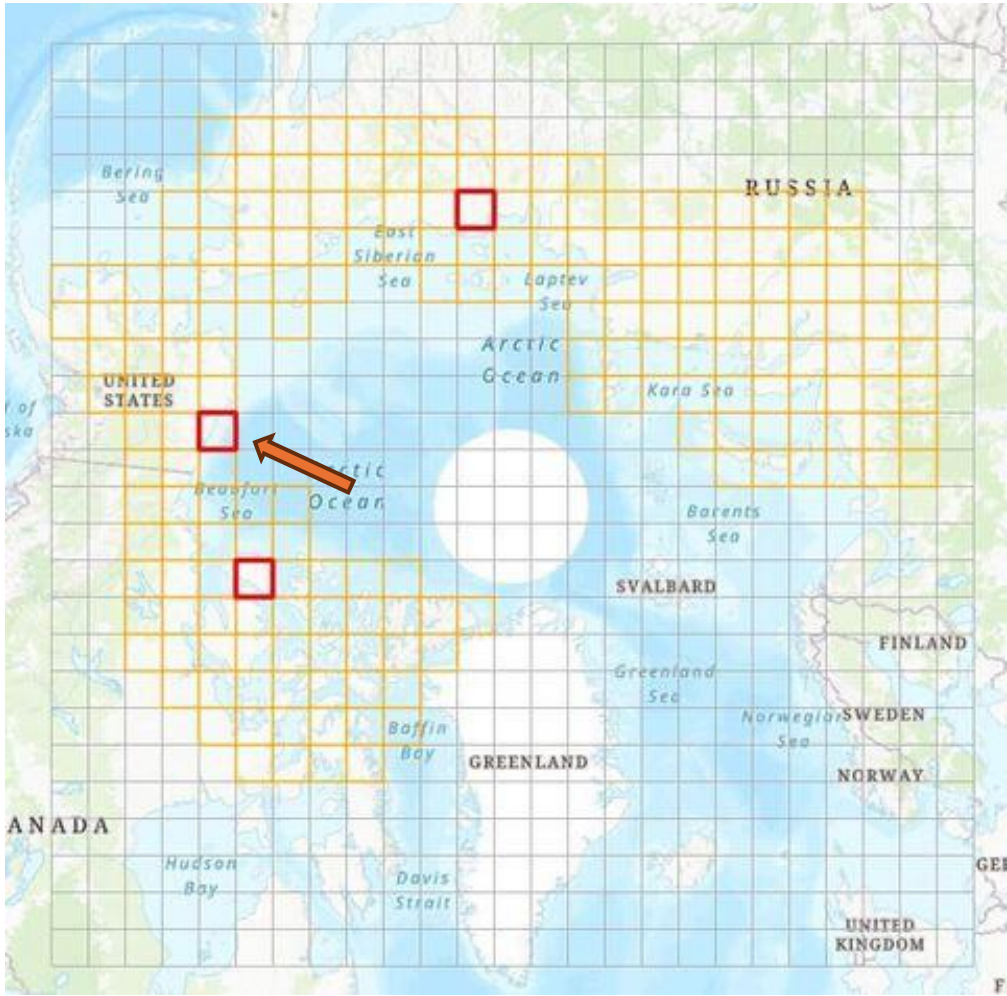


Within this pixel:

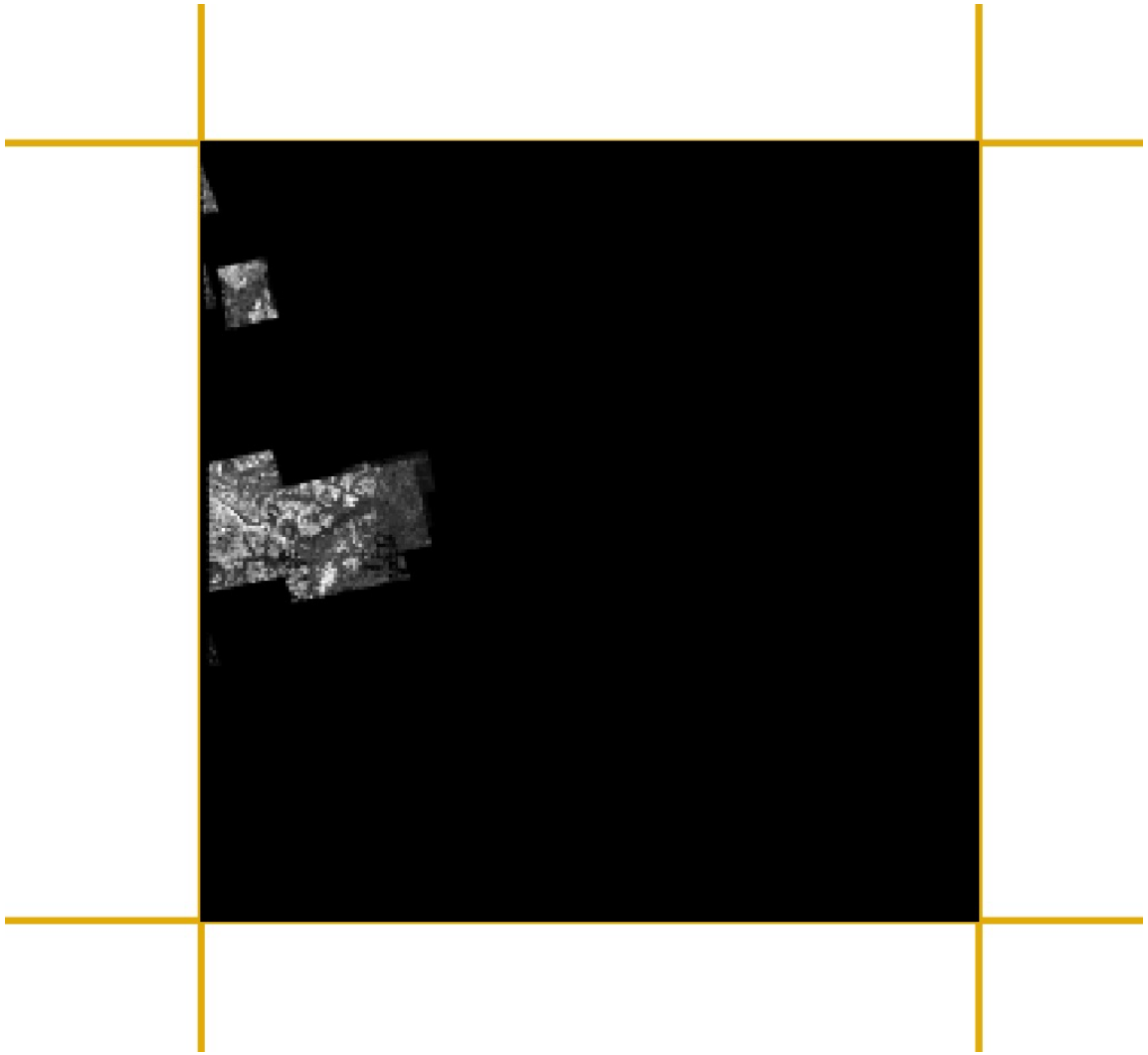
Length: 67, 685 m

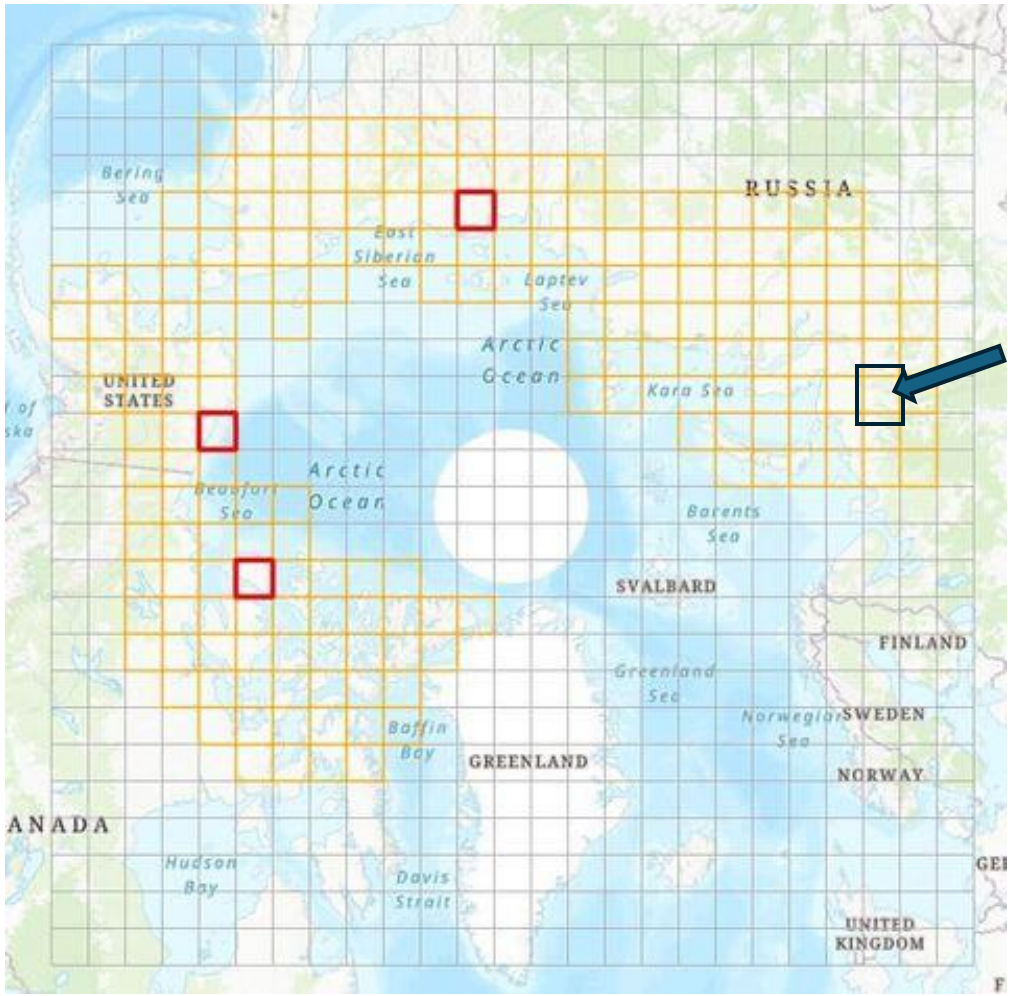
North Slope, Alaska

- 85,656,062 IWPs



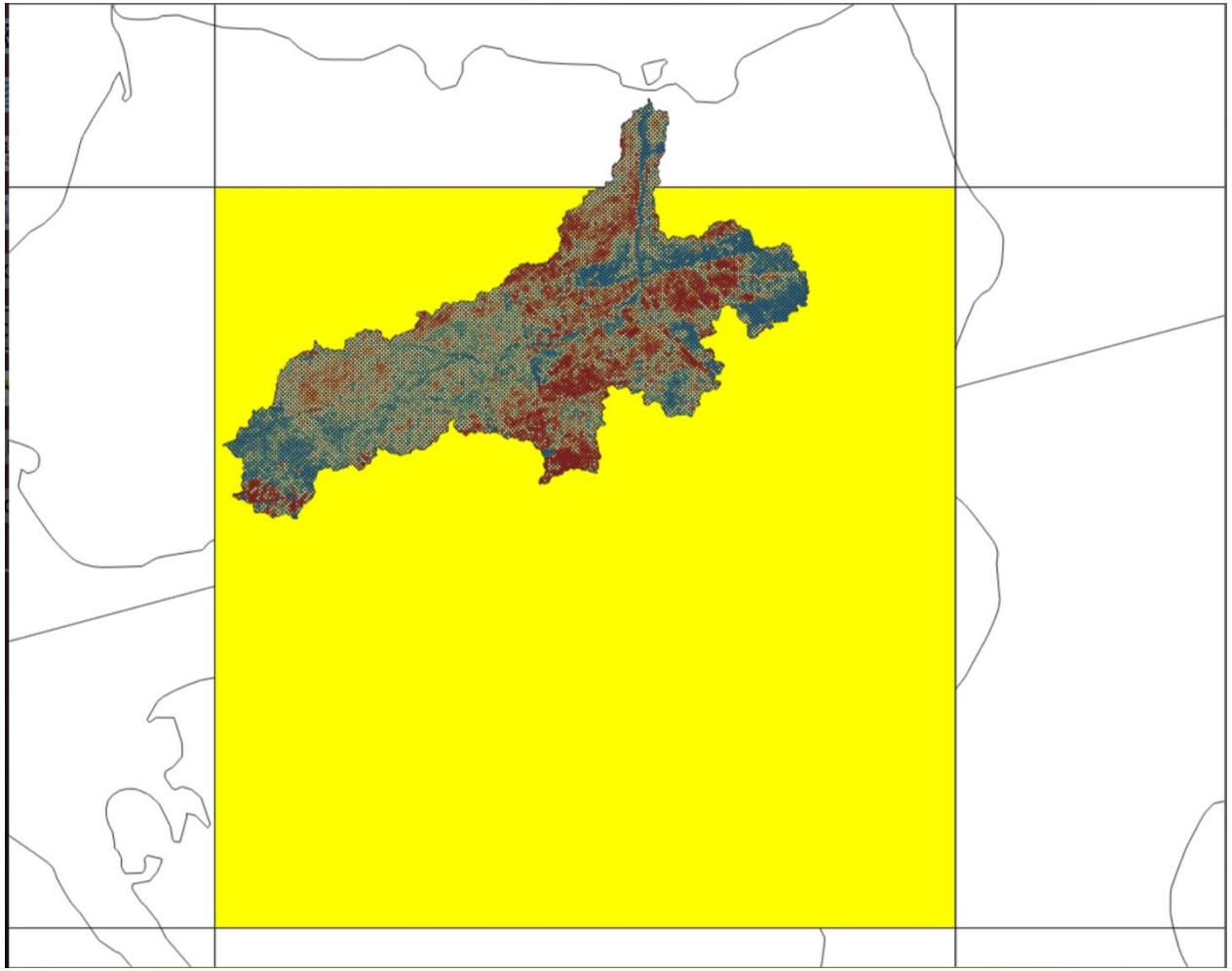
Processed grid





Meeting on 03/04/2025

- Cross validation on percentage coverage calculation with Elias' results
  - Yellow is one 256km by 256km at 1km resolution tile in our processing
  - The other is the watershed in Bank island that Elias processed.



Our results vs. Elias result on % coverage

- Computational efficiency improvement:
  - If we don’t consider data downloading time, each 256km^2 will take 20-30min on average.
  - So far, we are no longer using database
  - Processing the 240 tiles (a week)

Meeting 3/4/2025

- Results Evaluation
  - Accuracy evaluation: The degree of precision and correctness in predicting pipeline conditions or identifying anomalies.
  - Reliability evaluation: The consistency and stability of the results produced by the analytics techniques over multiple evaluations.
  - Scalability evaluation: The ability of the technique to handle large volumes of pipeline data and scale with increasing data sizes.
  - Computational efficiency: The speed and efficiency of the technique in processing and analyzing data within acceptable time frames.
  - Interpretability: The ease of understanding and interpreting the results generated by the analytics technique.

Accuracy evaluation

- IW network
  - Validate with external validation set (Elias work), which is generated outside of our algorithm-building process.

- Other statistics
  - Validate with Elias “coverage ratio” dataset

Reliability evaluation

- IW network:
  - Compared with hand-labelled IW network (randomly choose 1 pixels in Alaska, Canada, or Russia): relative error

- Other statistics
  - data downloading & querying: randomly choose 3× pixels in Alaska, Canada, and Russia. Compare results from our workflow and the manually downloaded gpkg files and calculated in ArcGIS.

Scalability & Computational Efficiency

- Compared parallel processing with non-parallel processing in data loading process in 1 grid
  - Speed up
  - # of concurrent processing
  - Focus on Alaska

Scientific validation

Their IWP dataset starts from 2001 to 2021, if our IWP dataset get trained on the images (2005-2008) within one of their 5 sites, then it should be fine. I'm not sure anything about their field sites now, so couldn't know if the datasets are consistent (edited)

Diameters of IWP (max) and total area of IWP. The smaller the IWP, the more ice wedges we have on ground.

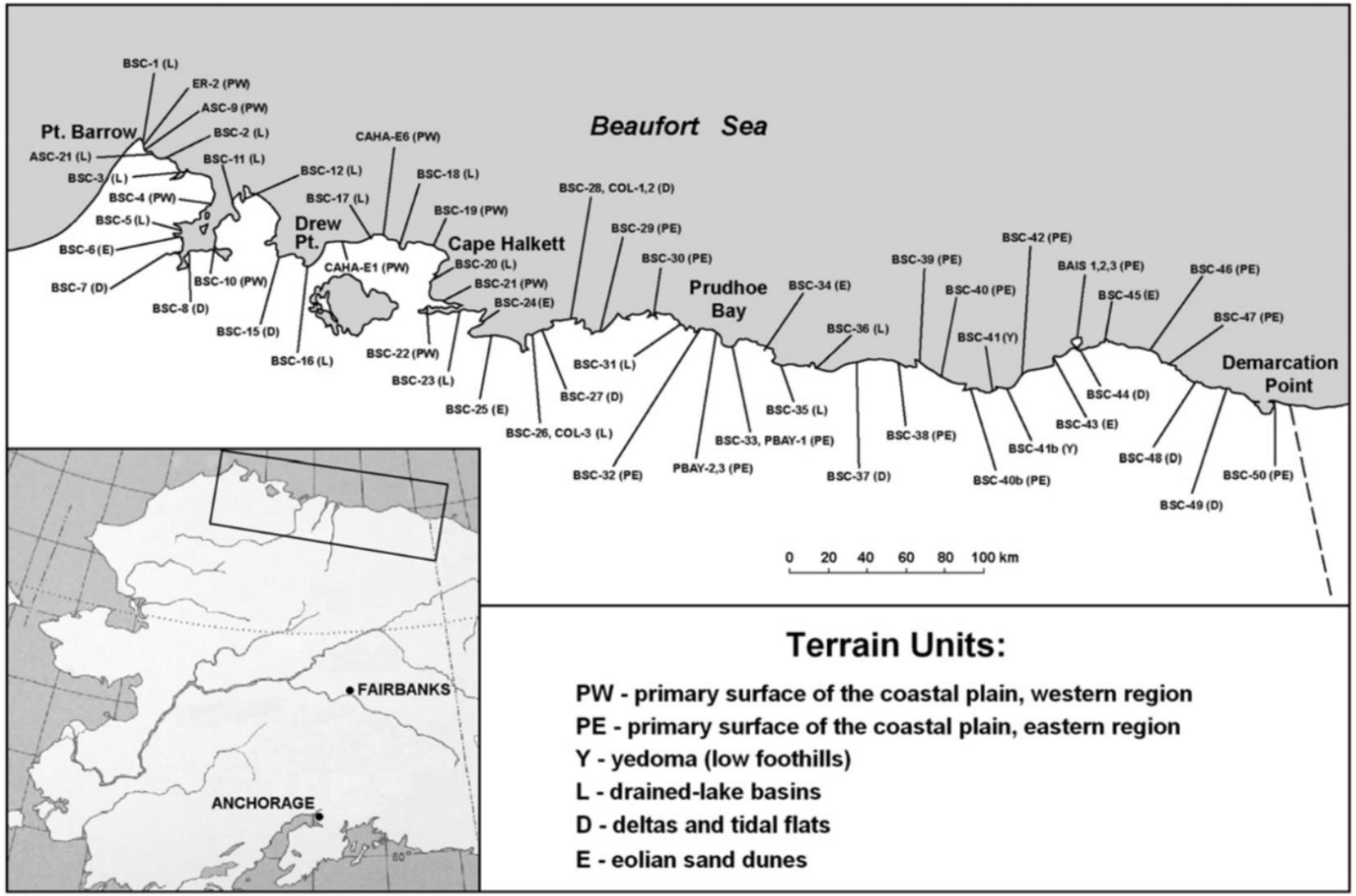
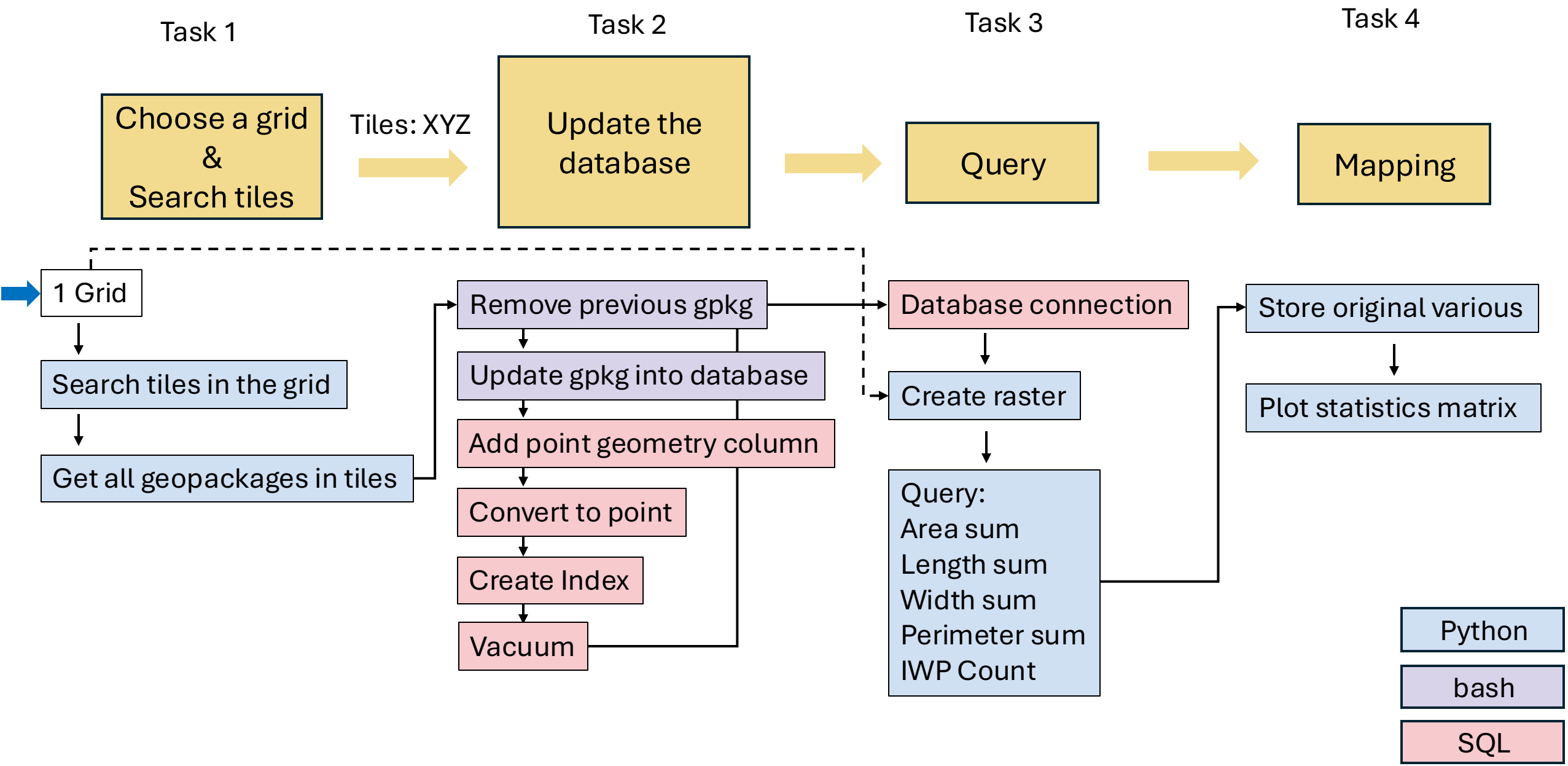


Fig. 1. Location of field sites studied in 2005–2008 along the Beaufort Sea coast from Point Barrow to the Canadian border.

https://www.sciencedirect.com/science/article/abs/pii/S0165232X12001644

Building data pipeline in one grid



Database parameters

	Inserting		Indexing			Querying + Calculating	
Concurrent files	20	20	1	20	20	1	20
Max_connection	200	30	200	30	30	200	30
Work_mem (default: 4MB)	64MB	Default	64MB	128MB	128MB	128MB	256MB
Maintanence_work_mem (default 64MB)	Default	Default	Default	1G	2G	Default	Default
Shared_buffer (25% RAM)	Default	Default	Default	Default	Default	Default	Default
effective_cache_size (4GB)	Default	Default	Default	Default	Default	Default	16GB
wal_level (replica)	Default	minimal	replica	replica	replica	replica	replica
Wal_buffers (-1)	Default	64MB	Default	Default	Default	Default	Default
Max_wal_size (1GB)	Default	4GB	Default	Default	Default	Default	Default
Fsync (on)	Default	off	Default	Default	Default	Default	Default
Synchronous_commit (on)	Default	off	Default	Default	Default	Default	Default
Time	13,521 s					4,074 s	

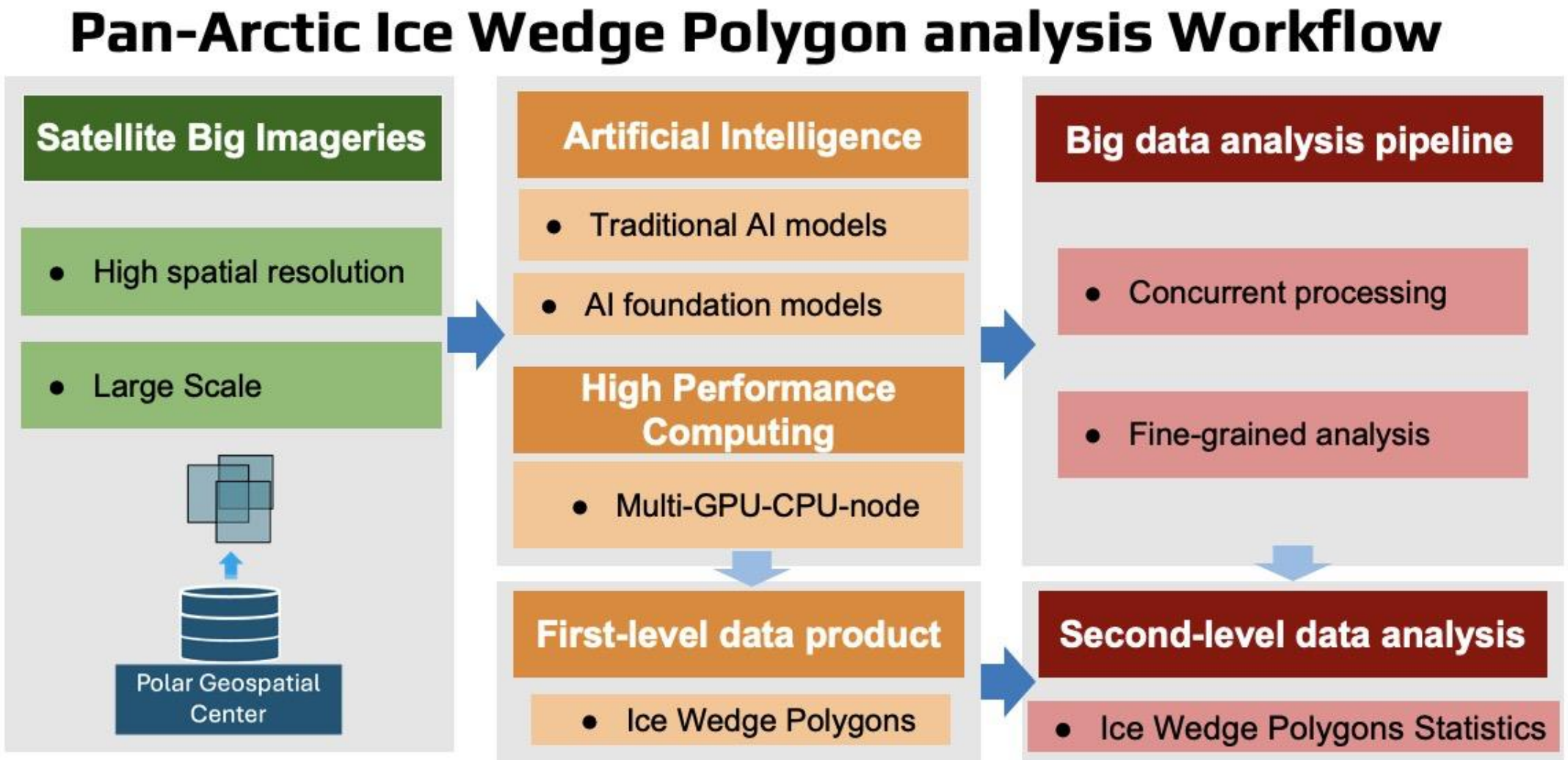
03/11/2025 - Evaluation

- Algorithm perspective
- Computational efficiency perspective
- Scientific assessment

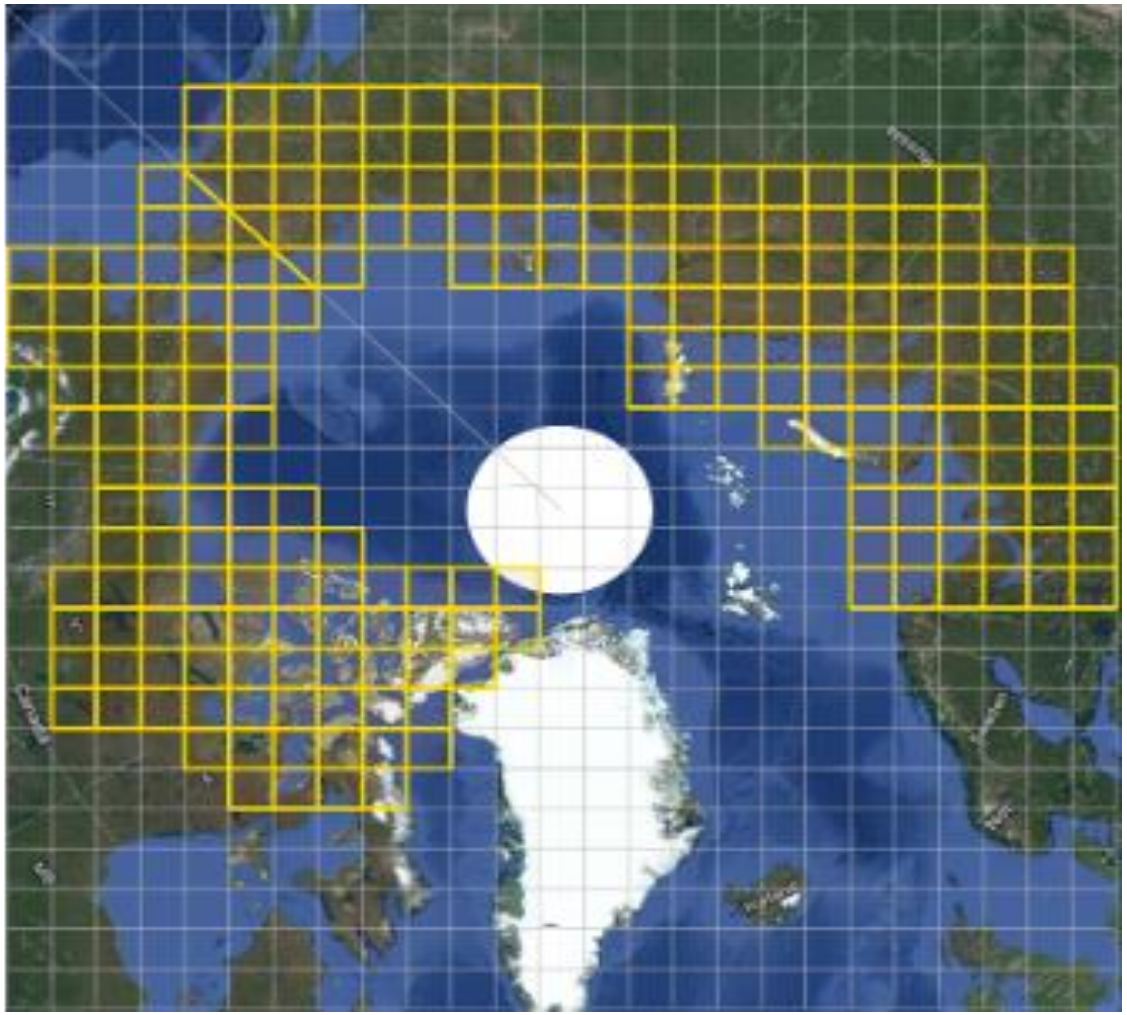
3/28/2025

- Mapping in Alaska

Workflow


Pan-Arctic Ice Wedge Polygon analysis Workflow
Satellite Big Imageries
High spatial resolution
Large Scale
Polar Geospatial Center
Artificial Intelligence
Traditional AI models
AI foundation models
High Performance Computing
Multi-GPU-CPU-node
First-level data product
Ice Wedge Polygons
Big data analysis pipeline
Concurrent processing
Fine-grained analysis
Second-level data analysis
Ice Wedge Polygons Statistics

Grids



Statistics mapping in Alaska