CHEK

DIGITAL BUILDING PERMIT

Digital Transformation in Building Permits

Module 5 - 3D city modelling: background and standards

9th July 2025

Funded by
the European Union

About me

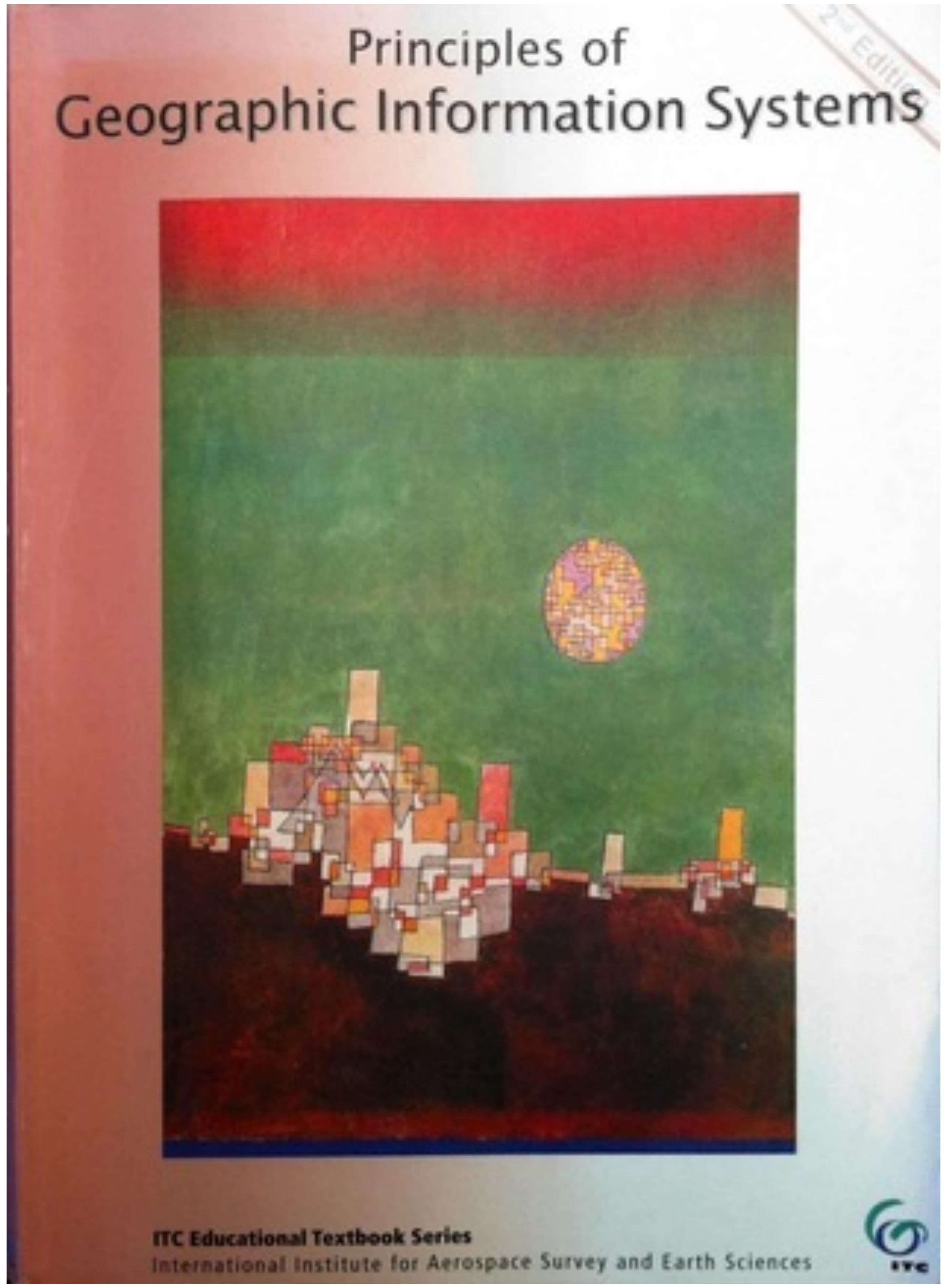
- Ken Arroyo Ohori
- Researcher and lecturer at TU Delft, the Netherlands
- k.ohori@tudelft.nl

Module 5 contents

- Foundations of 2D and 3D GIS
- Processing 2D and 3D geodata
- 3D city models

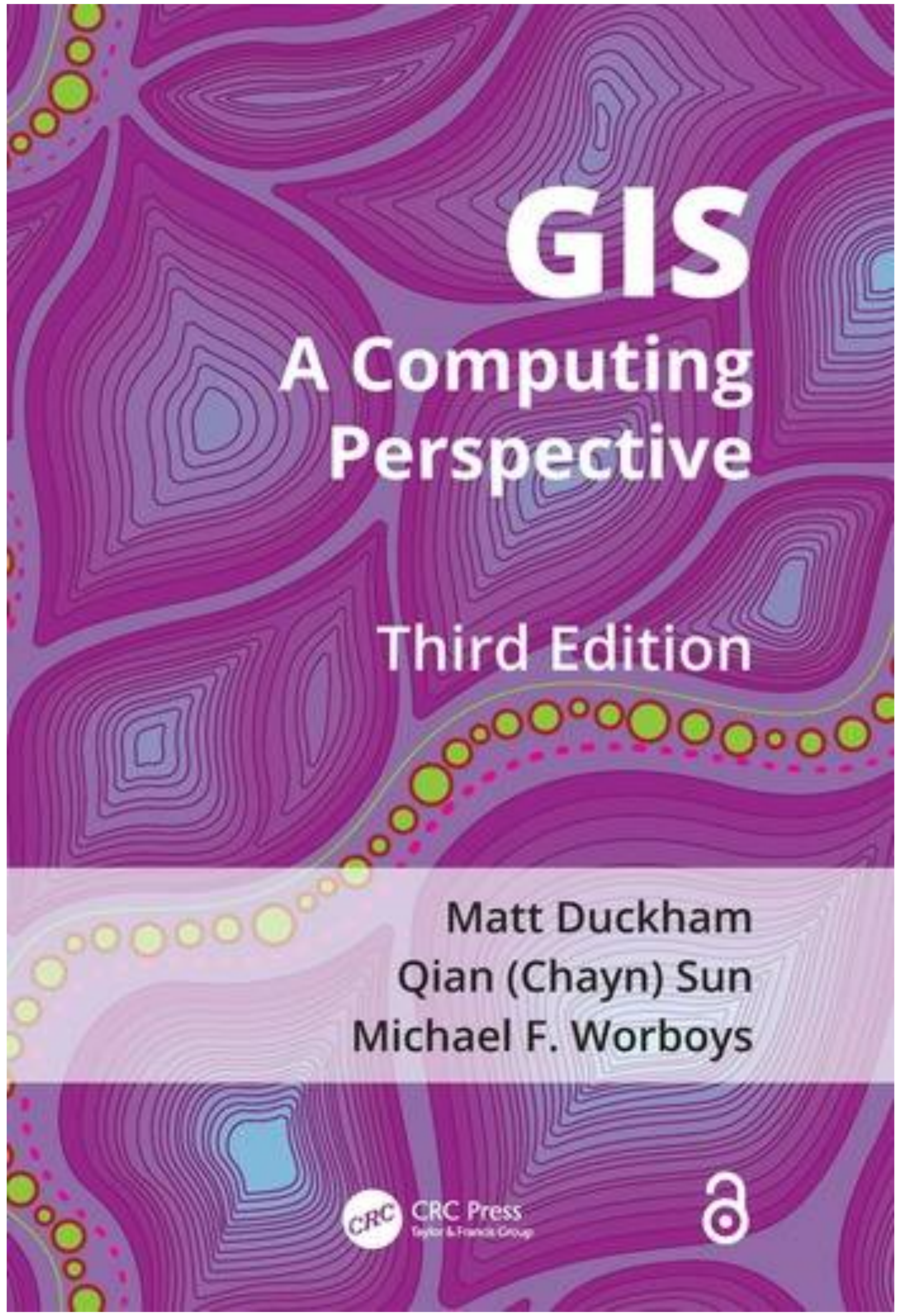
Recommended material

- Otto Huisman and Rolf A. de By. **Principles of Geographic Information Systems**. 4th Edition. 2009.
- https://archive.org/details/PrinciplesOfGeographicInformationSystemsBYOttoHuismanAndRolfA.De



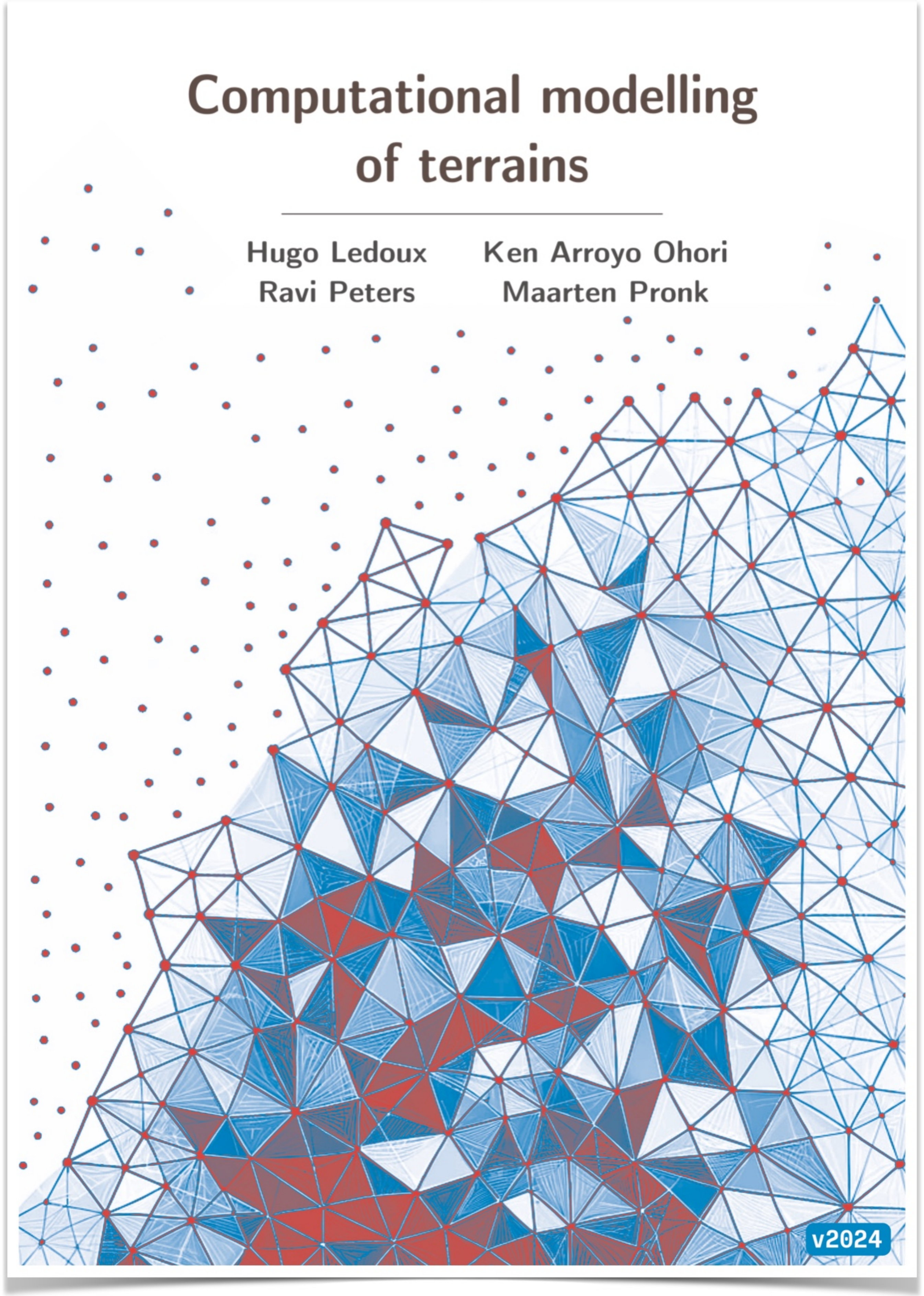
Recommended material

- Matt Duckham, Qian (Chayn) Sun and Michael F. Worboys. **GIS: A Computing Perspective**. 3rd Edition. 2024.



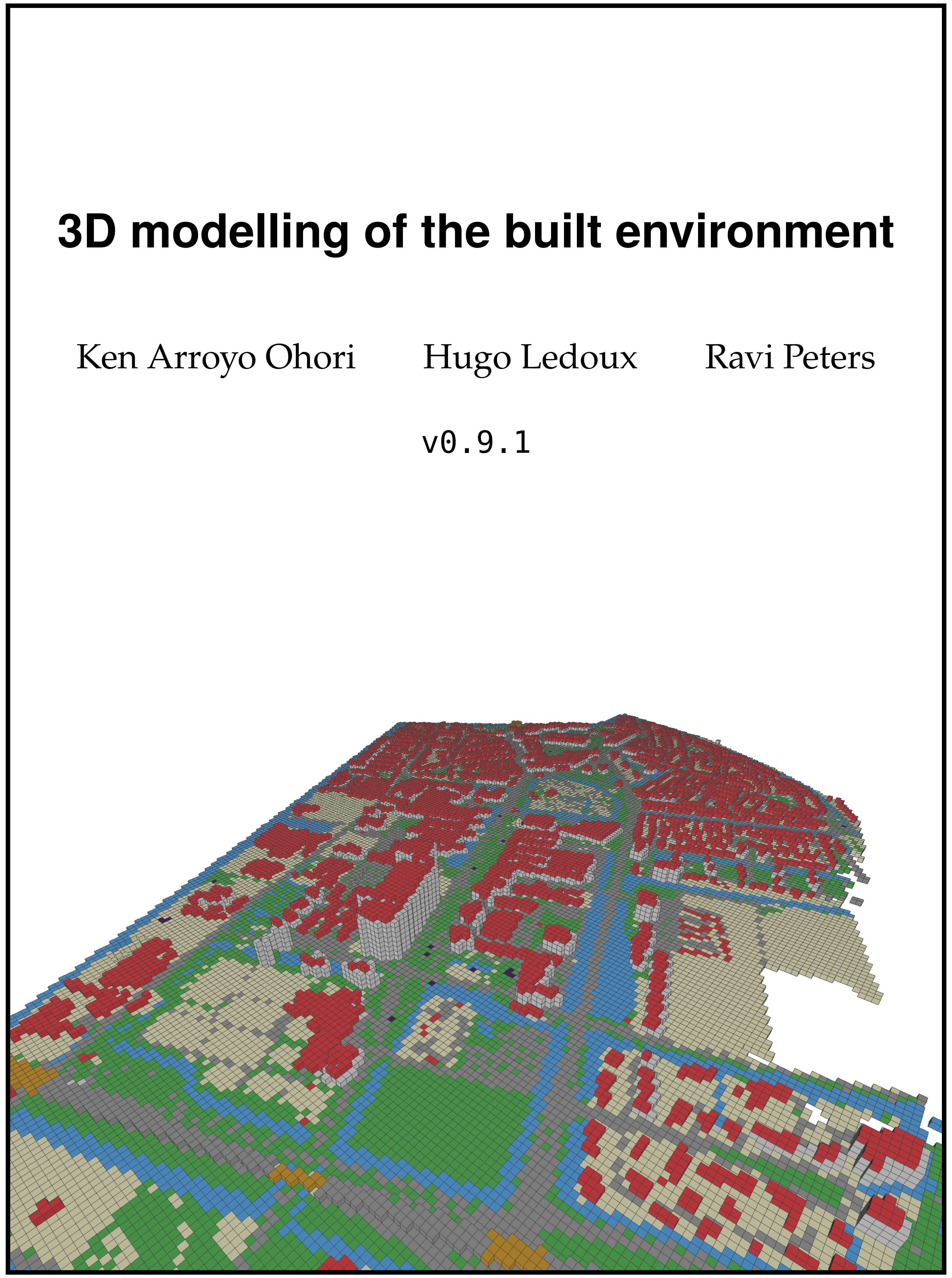
Recommended material

- Hugo Ledoux, Ken Arroyo Ohori, Ravi Peters and Maarten Pronk. **Computational modelling of terrains**.
- https://tudelft3d.github.io/terrainbook/



Recommended material

- Ken Arroyo Ohori, Hugo Ledoux and Ravi Peters. **3D modelling of the built environment**.
- https://github.com/tudelft3d/3dbook/releases/tag/v0.9.1



- Why a GIS? What is a GIS?
- Geographic phenomena: fields and objects
- Computer representations: vectors and rasters
- Georeferencing
- Practical session with QGIS

Why a GIS?

- An **urban planner** might want to assess the extent of urban fringe growth in their city, and quantify the population growth that some suburbs are witnessing. They might also like to understand why these particular suburbs are growing and others are not.

- A **mining engineer** could be interested in determining which prospective copper mines should be selected for future exploration, taking into account parameters such as extent, depth and quality of the ore body, amongst others.

- A **geoinformatics engineer** hired by a telecommunications company may want to determine the best sites for the company's cell phone towers, taking into account various cost factors such as land prices, shape of the terrain, tall buildings, etc.

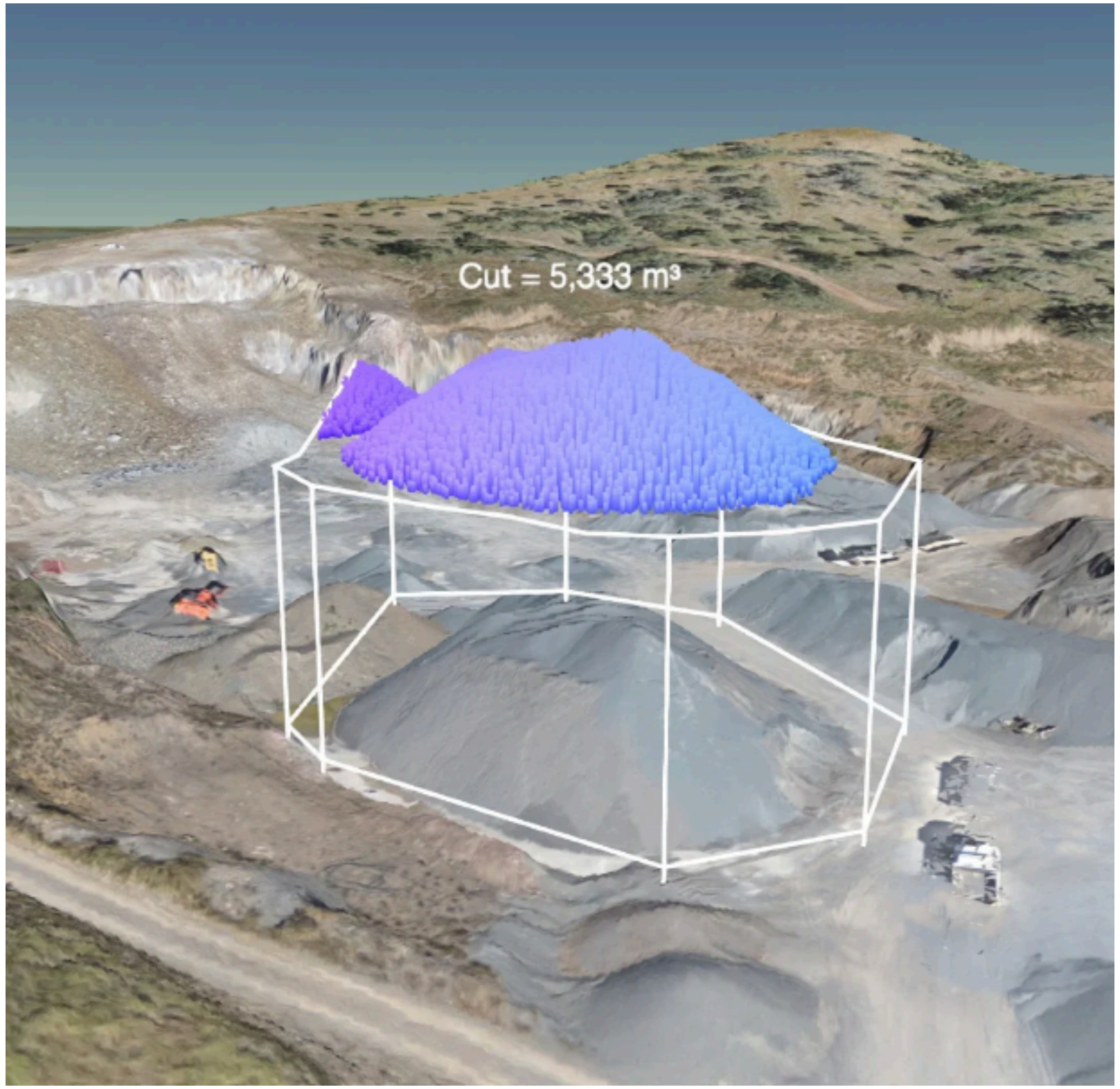
3D interface, aspern Seestadt in Vienna. Screenshot courtesy Wien3420

Why a GIS?

- An **urban planner** might want to assess the extent of urban fringe growth in their city, and quantify the population growth that some suburbs are witnessing. They might also like to understand why these particular suburbs are growing and others are not.

- A **mining engineer** could be interested in determining which prospective copper mines should be selected for future exploration, taking into account parameters such as extent, depth and quality of the ore body, amongst others.

- A **geoinformatics engineer** hired by a telecommunications company may want to determine the best sites for the company's cell phone towers, taking into account various cost factors such as land prices, shape of the terrain, tall buildings, etc.



Source: GIS Navigator

Why a GIS?

- An **urban planner** might want to assess the extent of urban fringe growth in their city, and quantify the population growth that some suburbs are witnessing. They might also like to understand why these particular suburbs are growing and others are not.

- A **mining engineer** could be interested in determining which prospective copper mines should be selected for future exploration, taking into account parameters such as extent, depth and quality of the ore body, amongst others.

- A **geoinformatics engineer** hired by a telecommunications company may want to determine the best sites for the company's cell phone towers, taking into account various cost factors such as land prices, shape of the terrain, tall buildings, etc.

Source: Urban Footprint

What is a GIS?

Short for: Geographic Information System

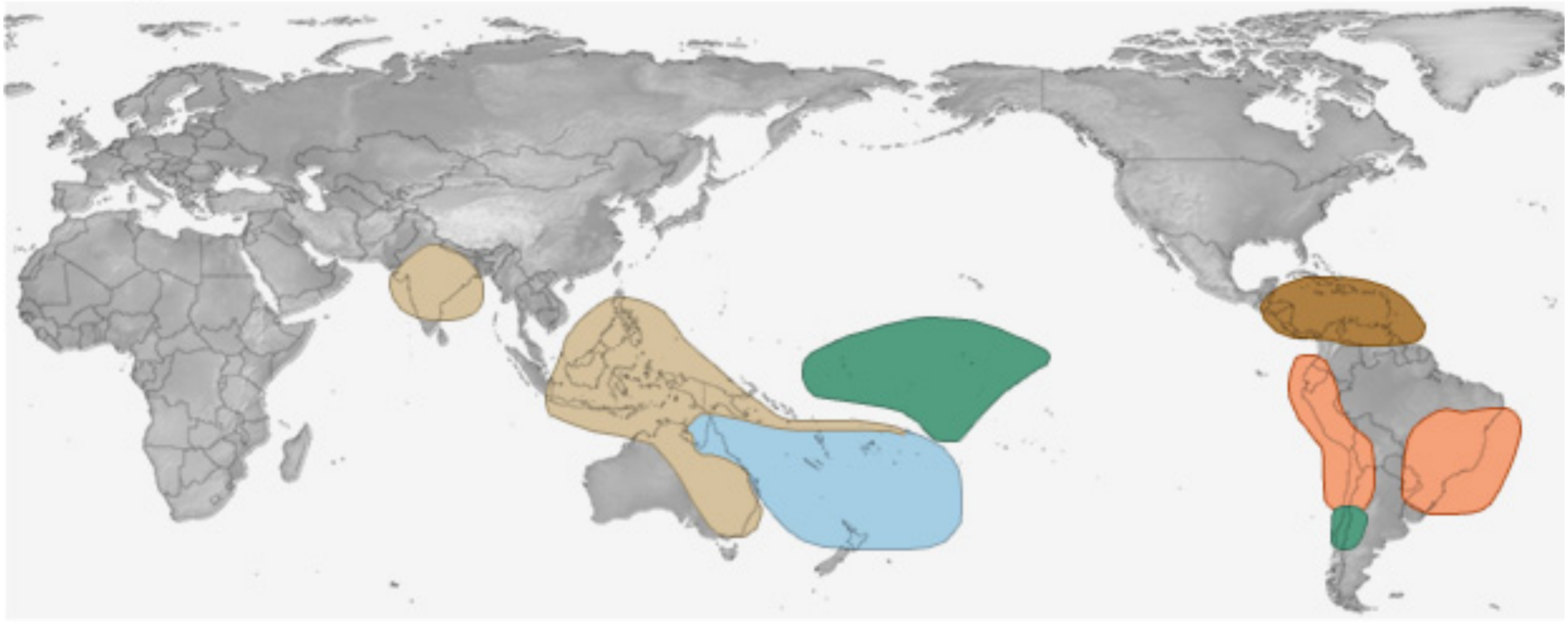
A GIS is a computer-based system that provides the following four sets of capabilities to handle georeferenced data:

- data capture and preparation,
- data management (storage and maintenance),
- data manipulation and analysis, and
- data presentation.

Image: macrovector in Freepik

Example: studying the El Niño phenomenon

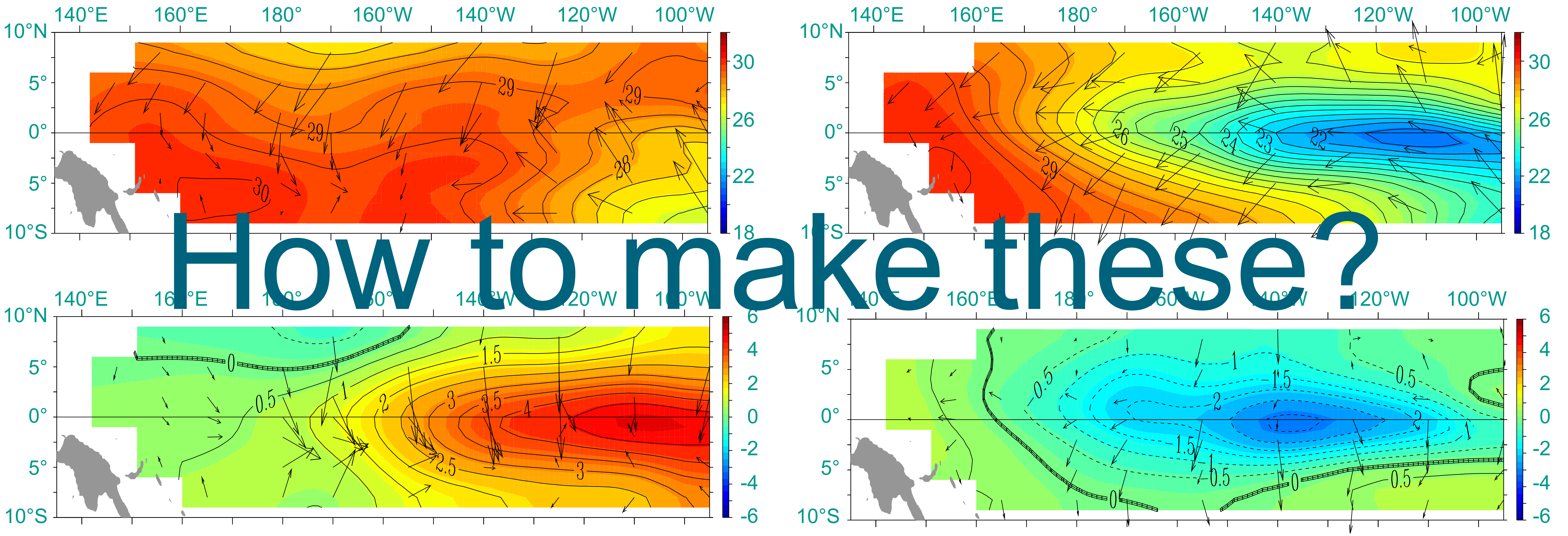
June-August



Cool	Wet	Cool and dry	Cool and Wet
Warm	Dry	Warm and dry	Warm and wet

Example: studying the El Niño phenomenon

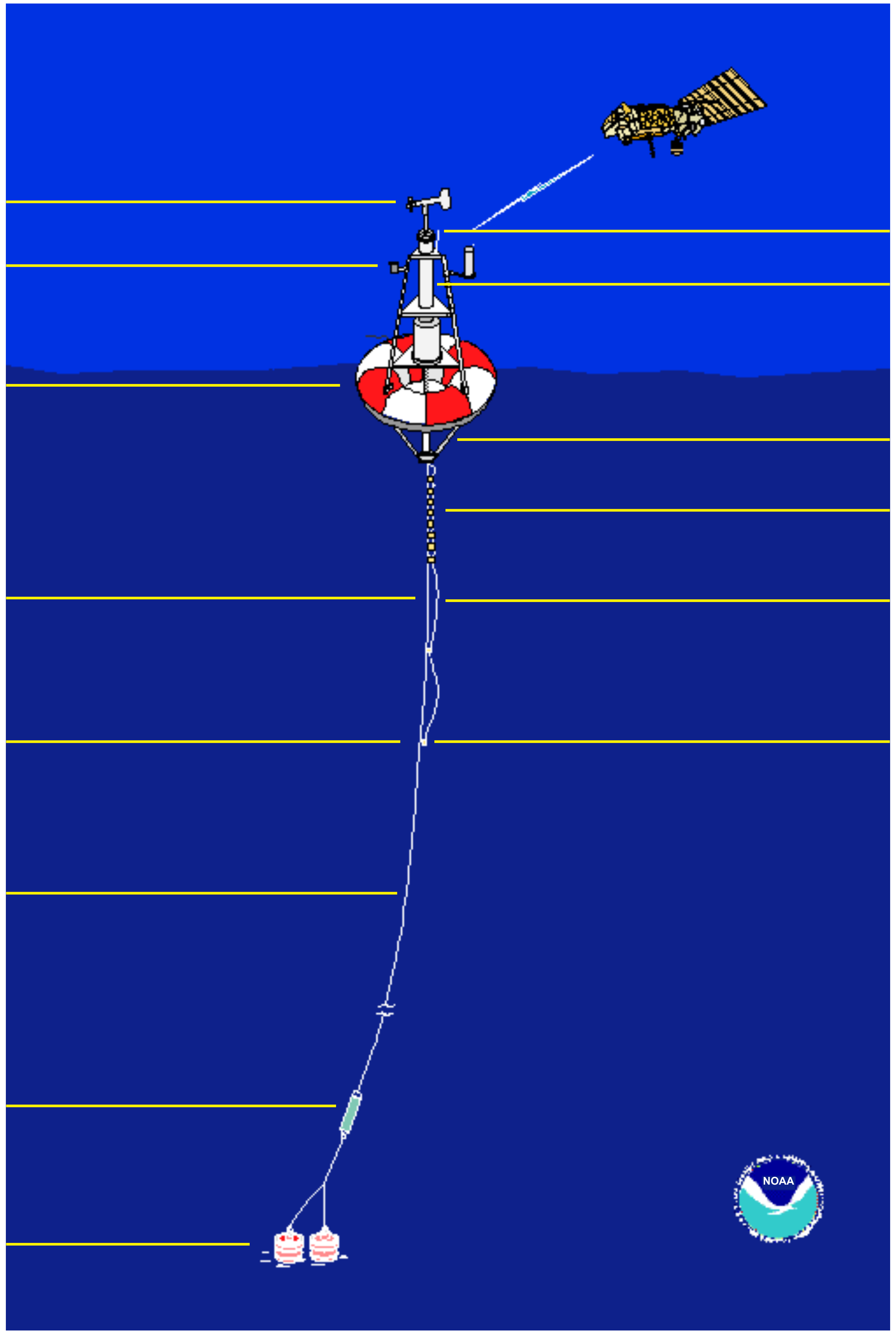
Upper figures: absolute values of average SST [°C] and WS [m/s]



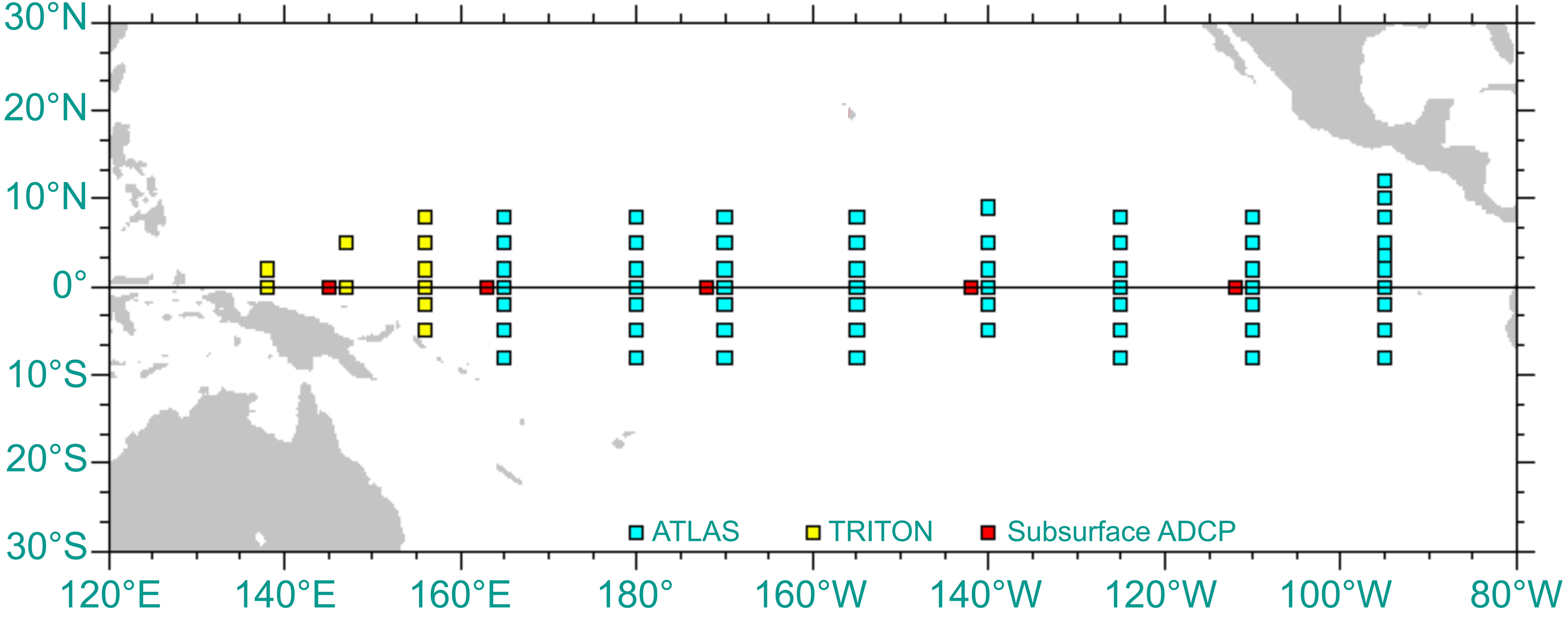
How to make these?

Lower figures: differences with normal situation

Step 1: collection of data



Step 2: data entry



Step 2: data entry

DayMeasurements

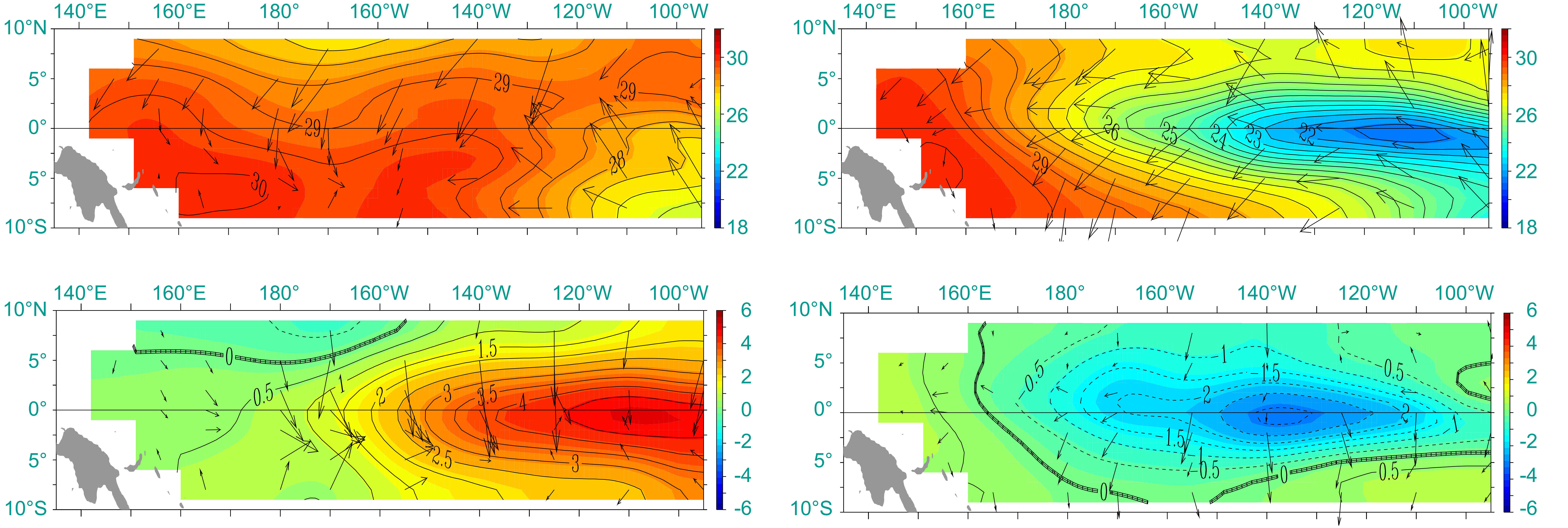
Buoy	*Date*	*SST*	*WS*	*Humid*	*Temp10*	...
B0749	1997/12/03	28.2 °C	NNW 4.2	72%	22.2 °C	...
B9204	1997/12/03	26.5 °C	NW 4.6	63%	20.8 °C	...
B1686	1997/12/03	27.8 °C	NNW 3.8	78%	22.8 °C	...
B0988	1997/12/03	27.4 °C	N 1.6	82%	23.8 °C	...
B3821	1997/12/03	27.5 °C	W 3.2	51%	20.8 °C	...
B6202	1997/12/03	26.5 °C	SW 4.3	67%	20.5 °C	...
B1536	1997/12/03	27.7 °C	SSW 4.8	58%	21.4 °C	...
B0138	1997/12/03	26.2 °C	W 1.9	62%	21.8 °C	...
B6823	1997/12/03	23.2 °C	S 3.6	61%	22.2 °C	...
...	...	...	...	...	...	...

Step 3: data analysis

Buoy	*Geographic position*	*Dec. 1997 avg. SST*
B0789	(165° E, 5° N)	28.02 °C
B7504	(180° E, 0° N)	27.34 °C
B1882	(110° W, 7°30' S)	25.28 °C
...	...	...

Step 4: data presentation

Upper figures: absolute values of average SST [°C] and WS [m/s]

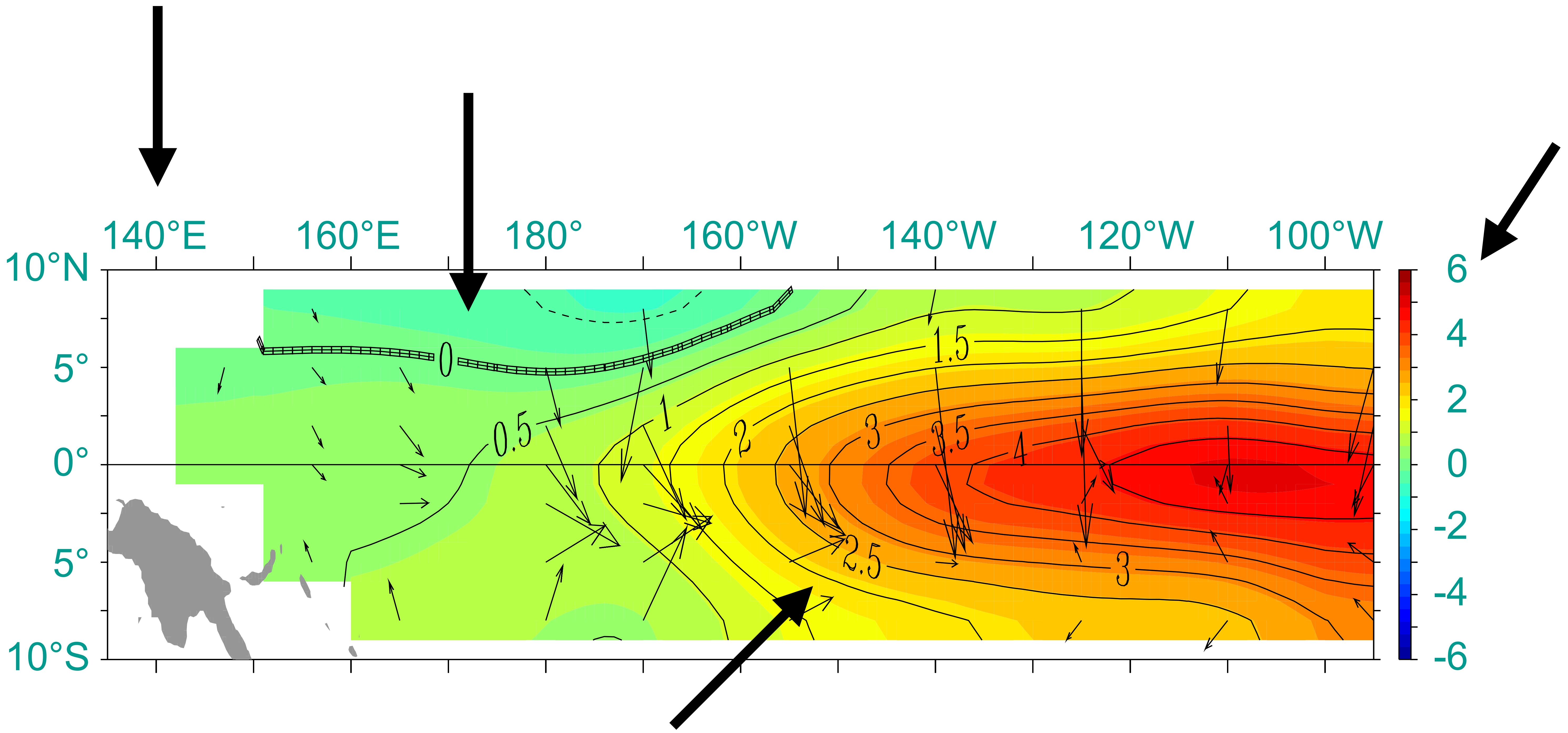


Lower figures: differences with normal situation

Step 4: data presentation

Some elements:

- contour lines
- colour selection
- coordinates
- legend



What is a GIS in practice?

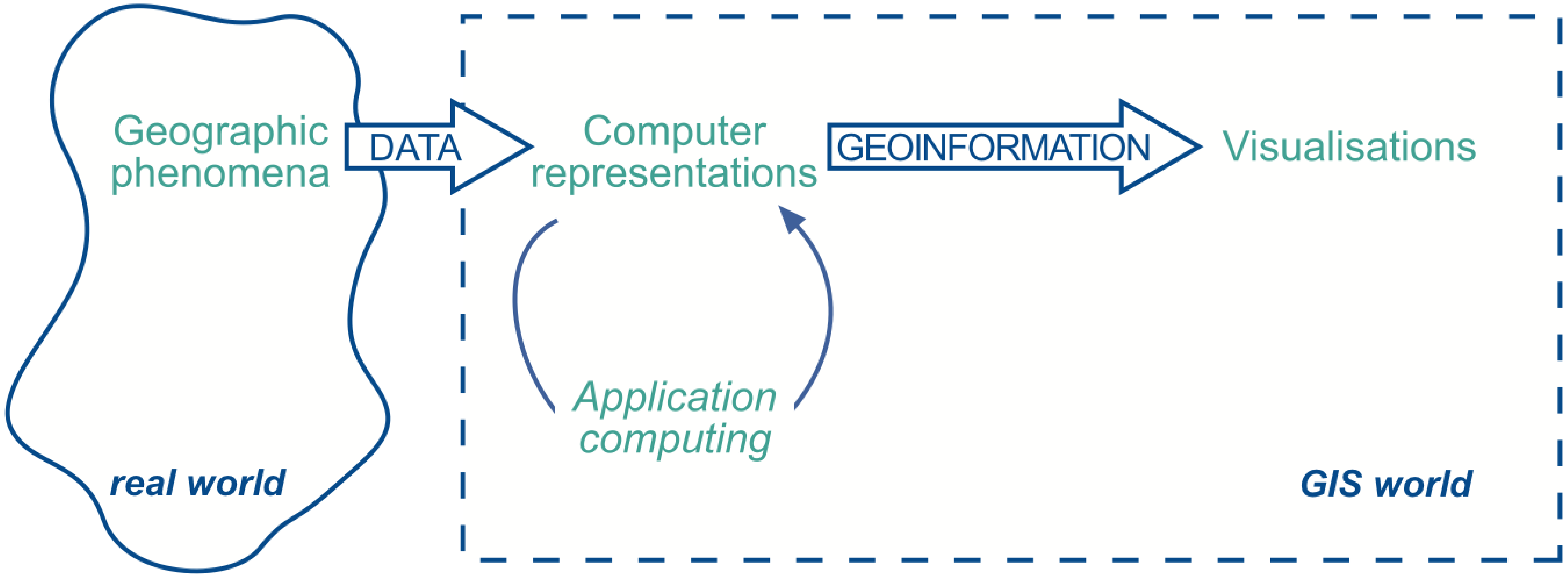
Depending on the interest of a particular application, a GIS can be considered to be:

- data storage / (geo)database,
- toolbox / a set of libraries,
- technology,
- data source,
- field of science.

Related terms

- Geographic information science (GIScience): discipline
- Geomatics: encompasses also acquisition of data and applications
- Geoinformatics: focus on computational methods

What is covered in a GIS?



- Why a GIS? What is a GIS?
- Geographic phenomena: fields and objects
- Computer representations: vectors and rasters
- Georeferencing
- QGIS demo & practical session

Geographic phenomena

- Entity or process that can be described, has a location in space and can be assigned a time at which it is present. There are two types:
  - **Fields**, which have a value for every point in the study area. Can be continuous or discrete.
  - **Objects**, which populate the study area and are are usually well-distinguished, discrete, and bounded entities. The space between them is potentially ‘empty’ or undetermined.

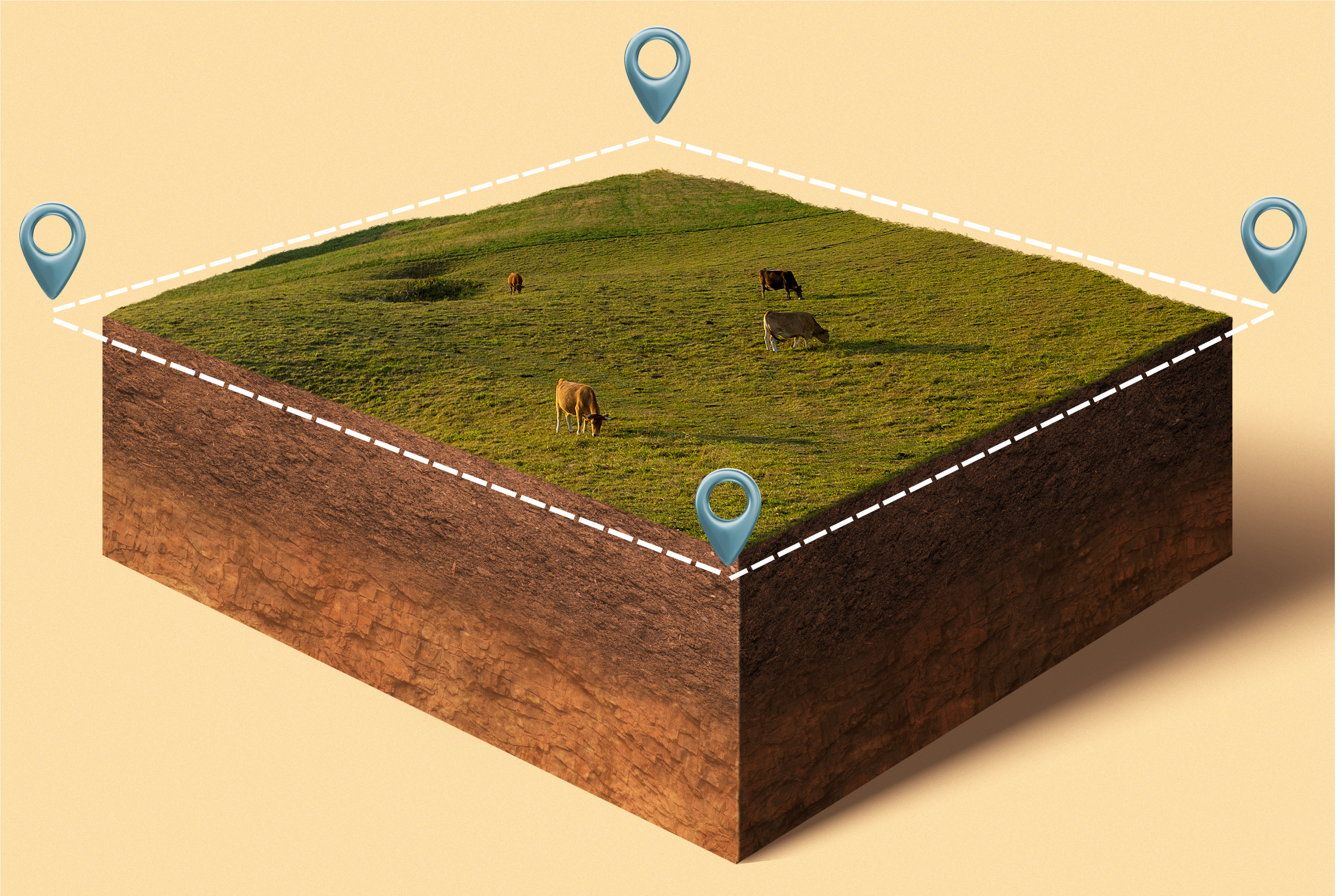
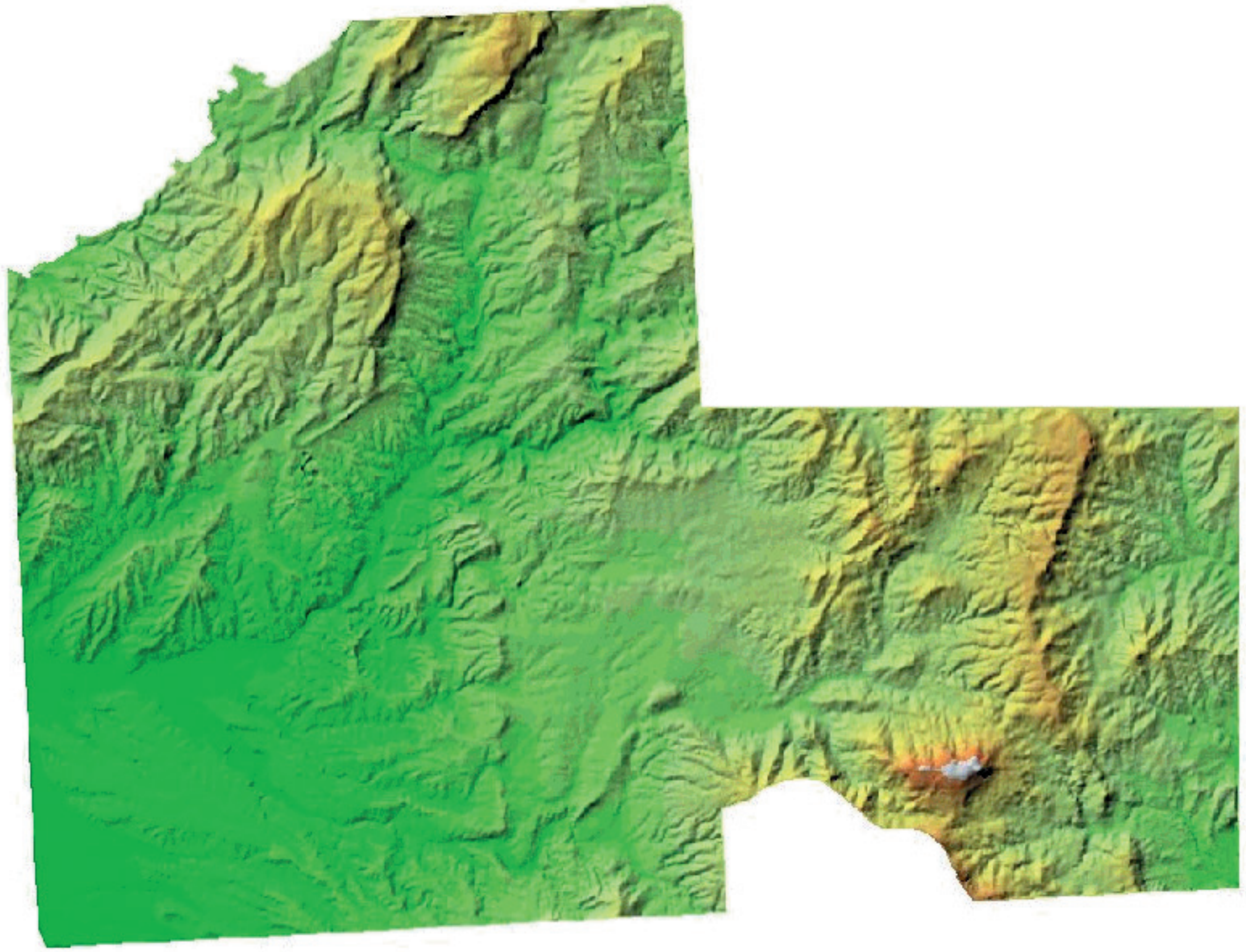


Image: Freepik

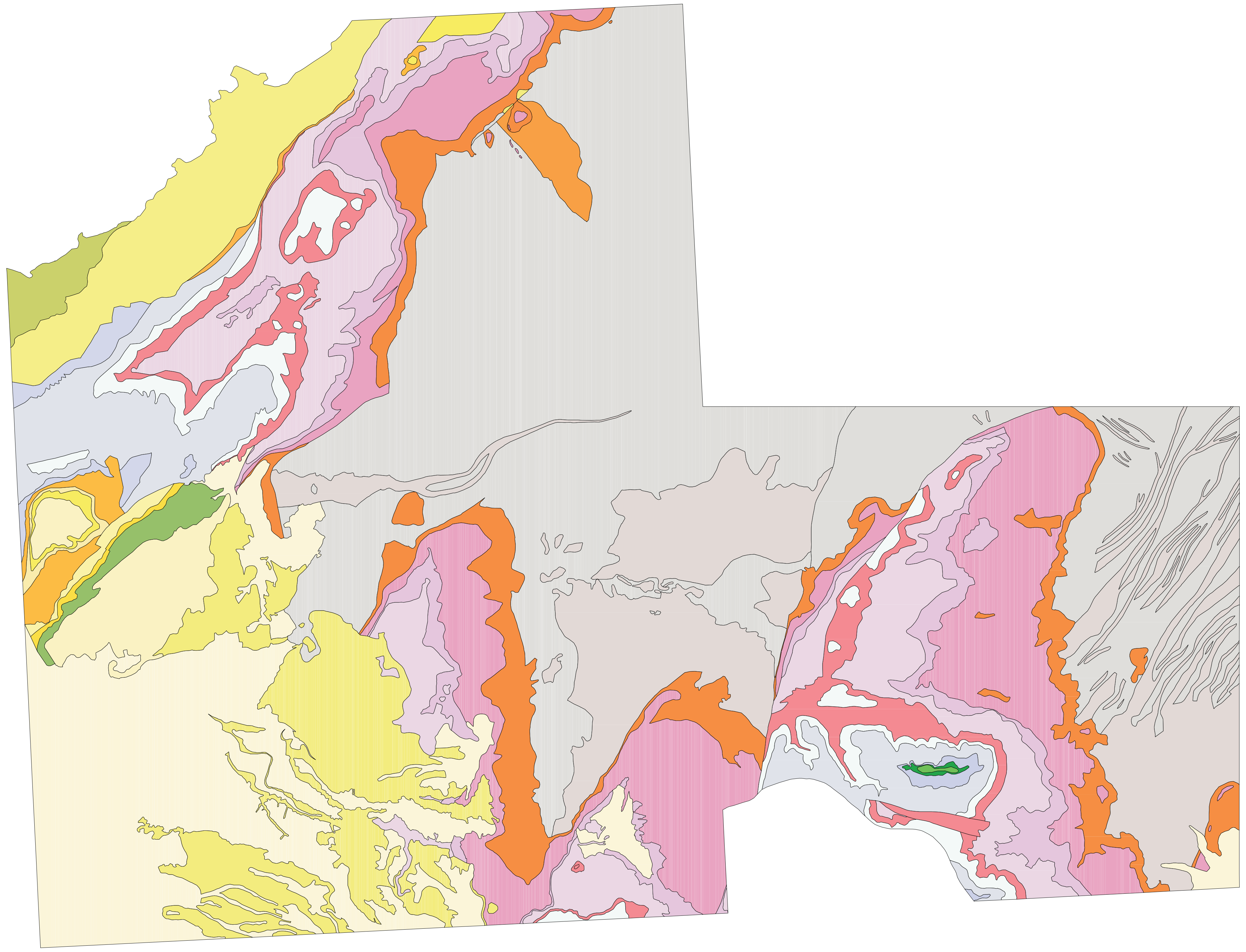
Geographic phenomena: fields (continuous)

- **In 2D:**
  - terrain elevation
  - soil salinity
- **Also in 3D:**
  - temperature
  - humidity
  - air pressure

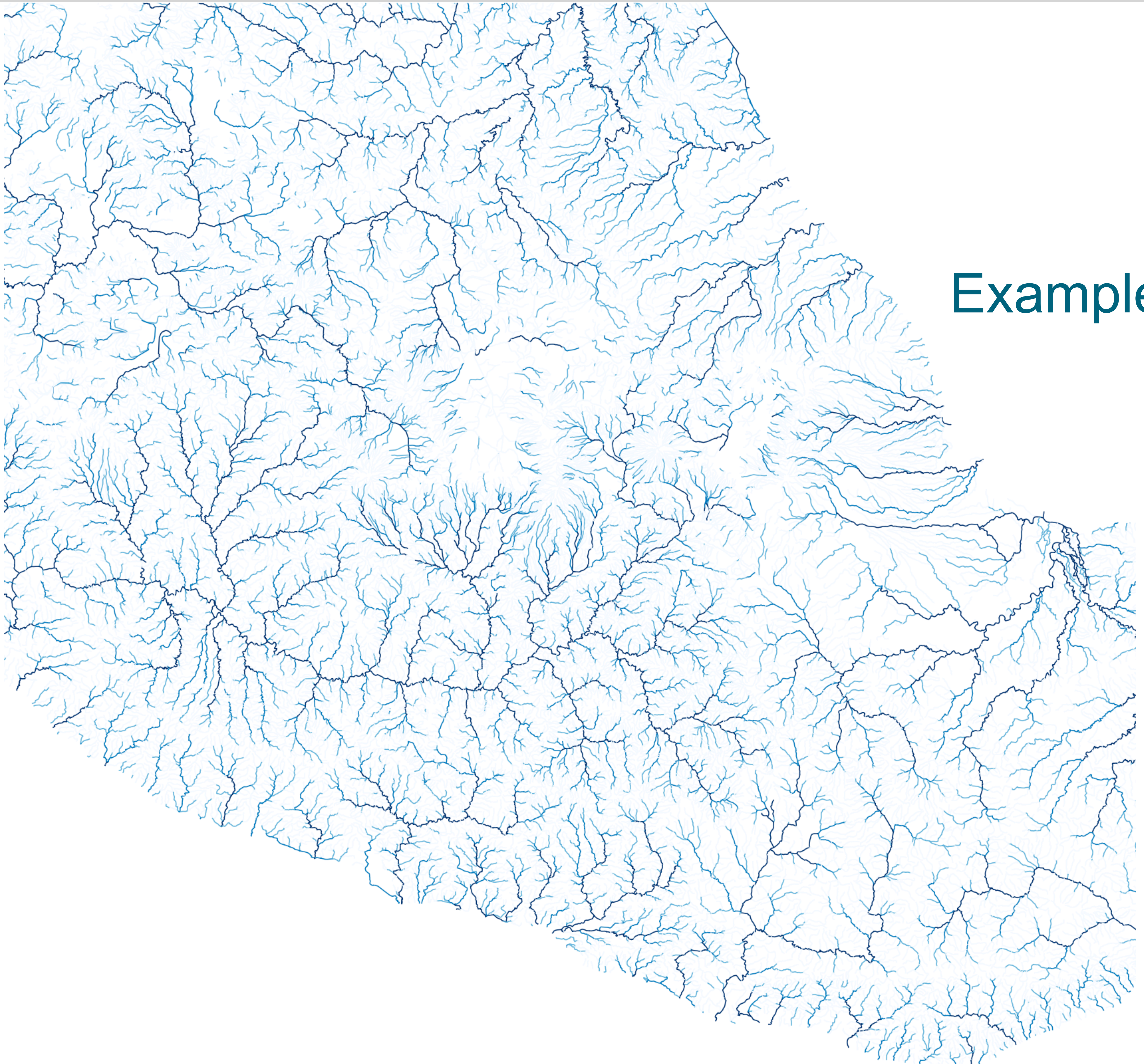


Geographic phenomena: fields (discrete)

- **In 2D:**
  - soil type
  - land use
  - vegetation type
- **Also in 3D:**
  - geological classes
  - building storeys



Geographic phenomena: objects



Examples:

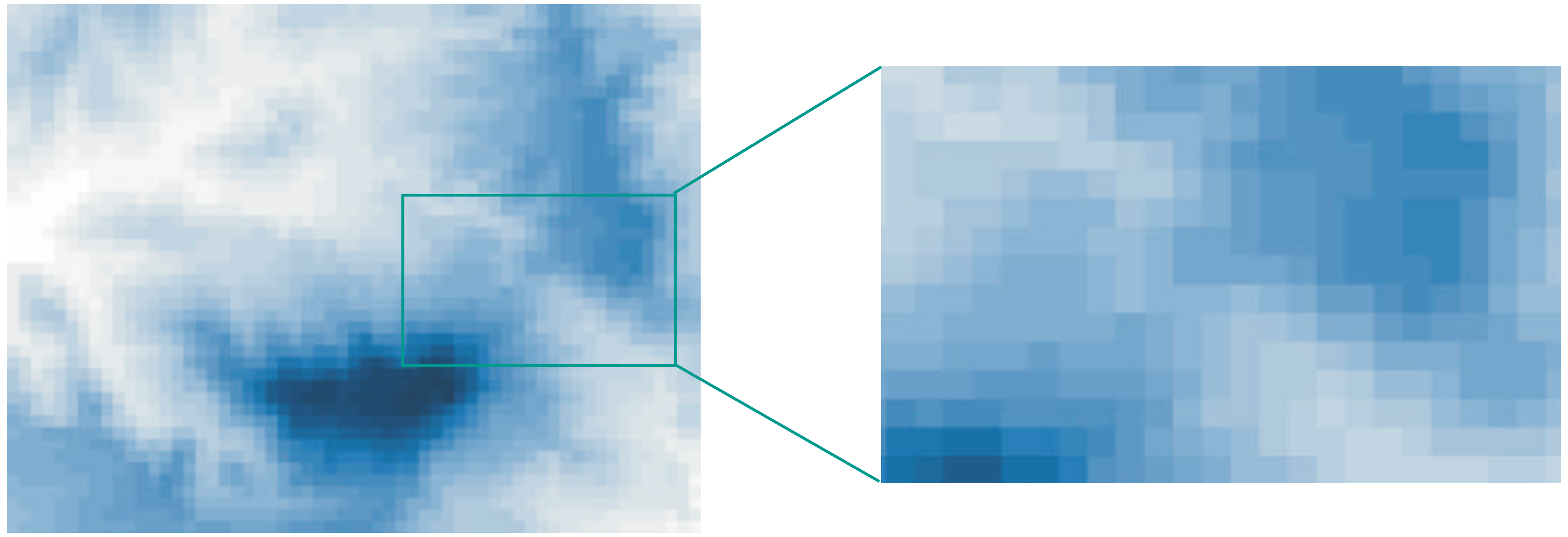
- cities (as points or areas)
- roads (as lines)
- buildings (in 3D)
- hydrological network

Computer representations: vector and raster

- Two ways to represent geographic phenomena in a computer:

rasters

have 2D or 3D grids of cells, each of which has a value;



vectors

are discrete shapes (e.g. points, lines, polygons and polyhedra) with precise boundaries.

- What is the relationship between fields/objects and rasters/vectors?

  - **fields** tend to be represented as **rasters**;

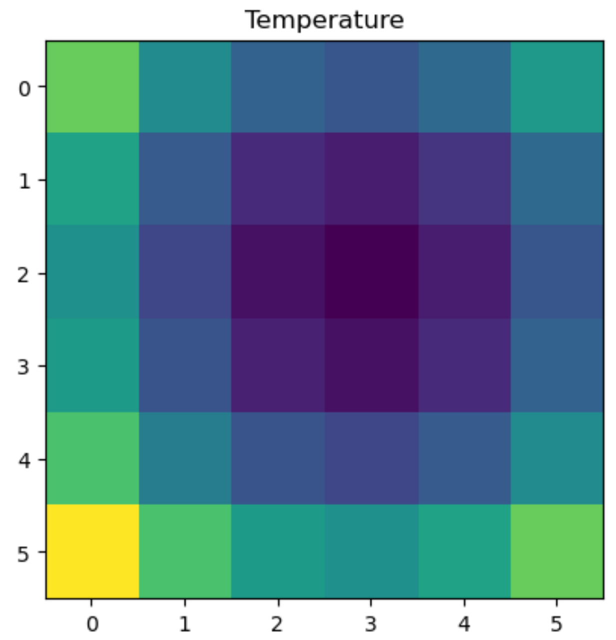
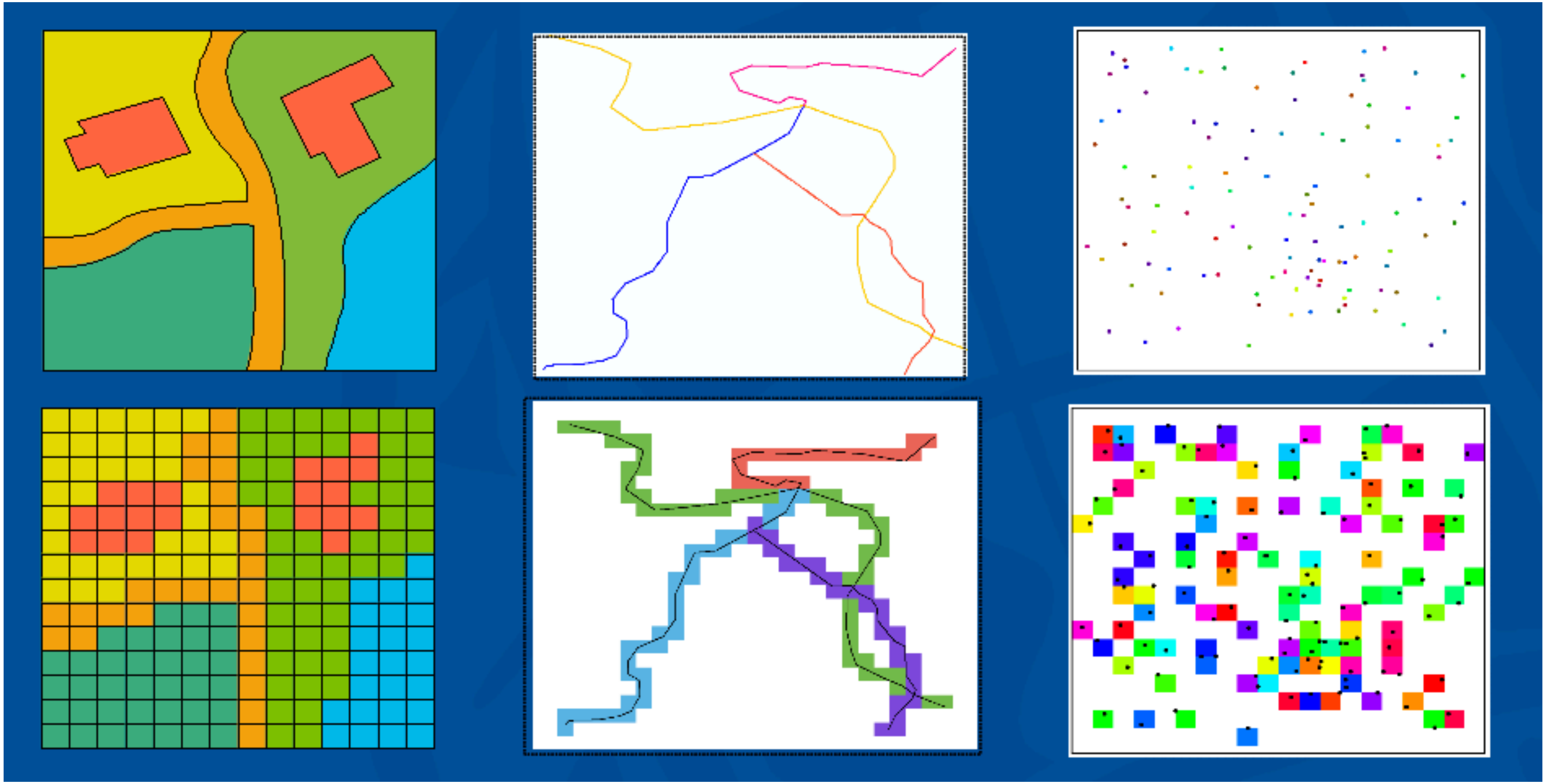


Image: pygis.io

  - **objects** tend to be represented as **vectors;**

- But there are many exceptions!

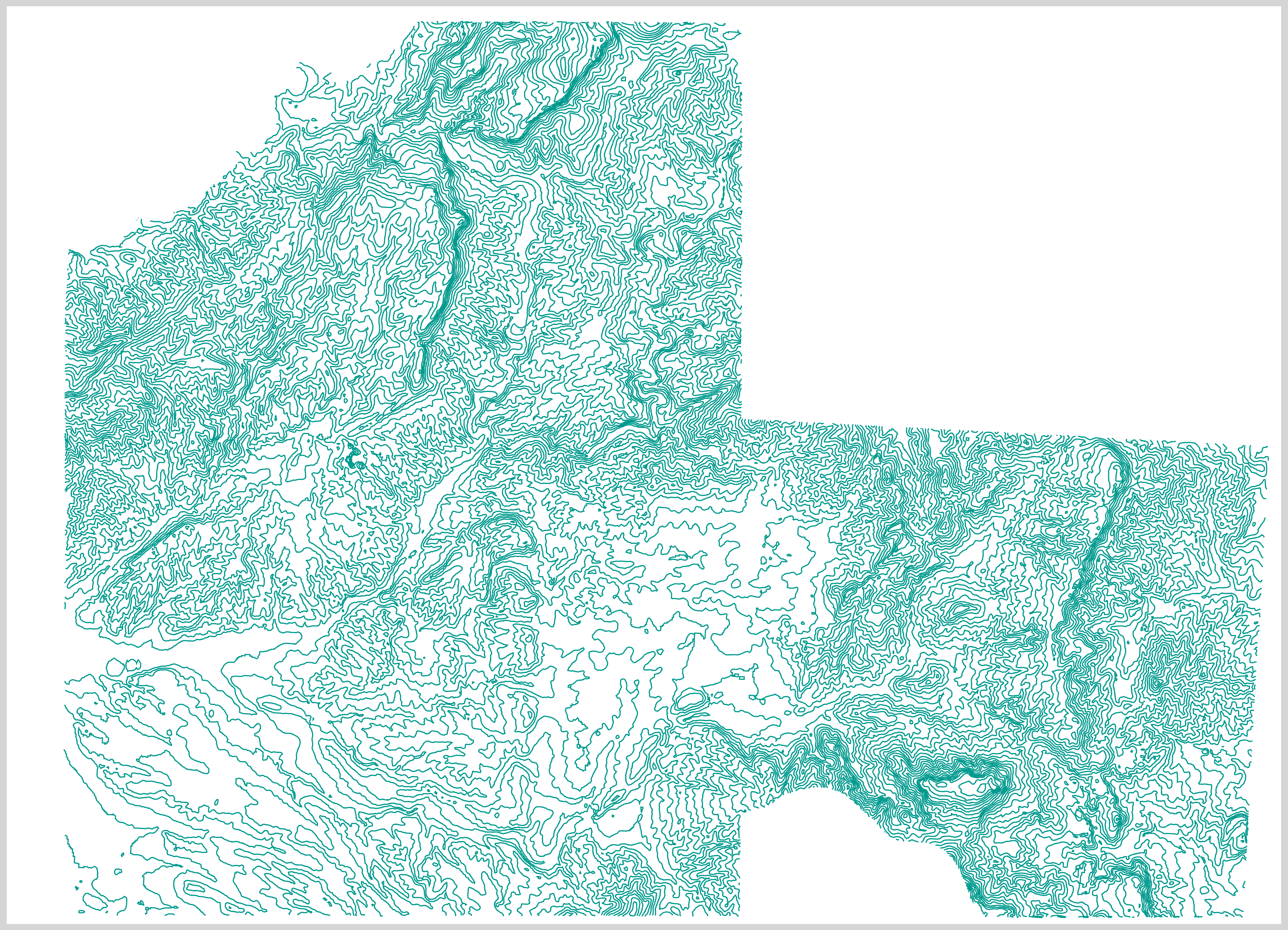
Computer representations: objects as vector and raster



Computer representations: object as raster

Computer representations: object as raster

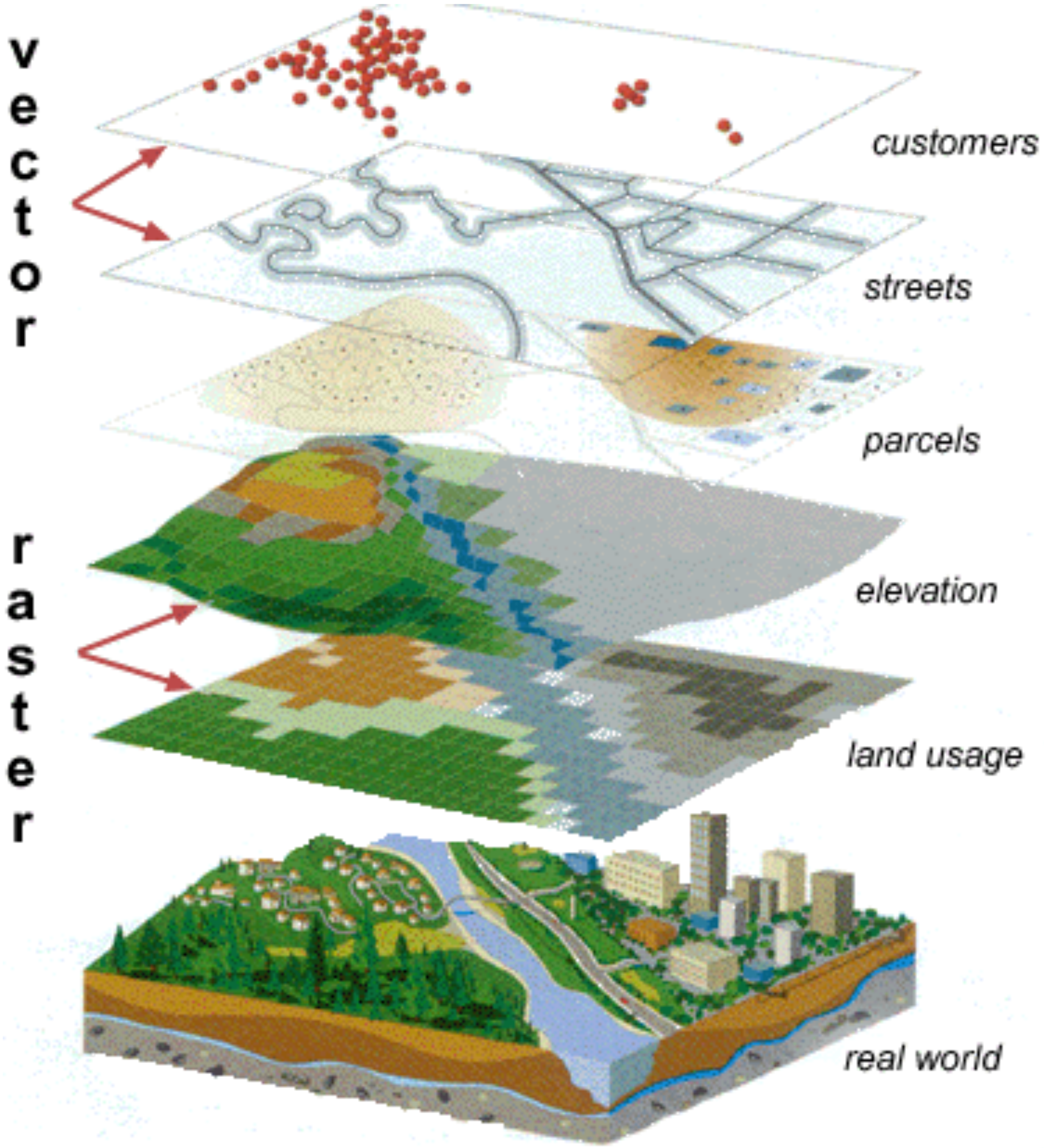
Computer representations: field as vector



Computer representations: vector and raster

Raster representation	*Vector representation*
advantages	
• simple data structure • simple implementation of overlays • efficient for image processing	• efficient representation of topology • adapts well to scale changes • allows representing networks • allows easy association with attribute data
disadvantages	
• less compact data structure • difficulties in representing topology • cell boundaries independent of feature boundaries	• complex data structure • overlay more difficult to implement • inefficient for image processing • more update-intensive

Computer representations: vector and raster



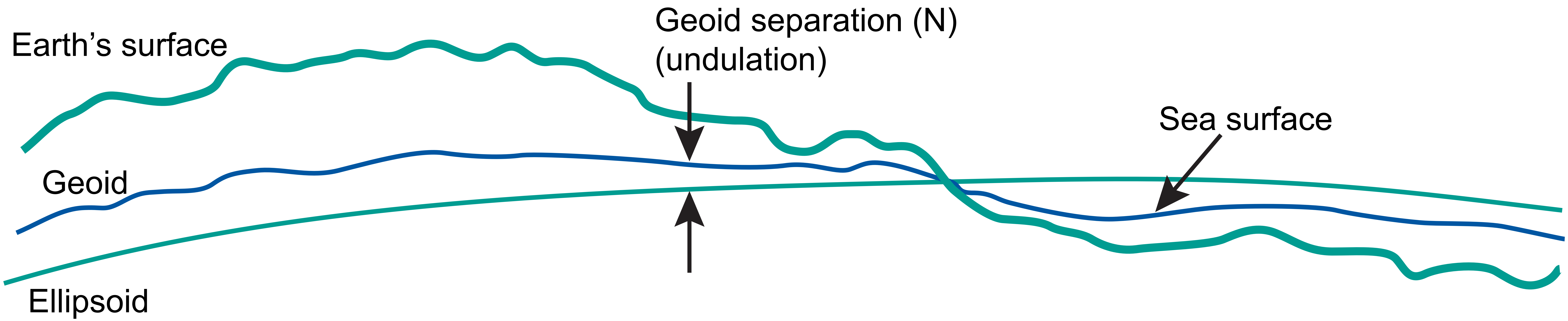
Computer representations: geodata

- Raster: GeoTIFF (.tif, .tiff), JPEG2000 (.jp2, .j2k), ERDAS Imagine (.img), Esri Grid, NetCDF (.nc), HDF (.hdf, .h5), ASCII grid (.asc, .dem), Cloud-optimised GeoTIFF (COG), MrSID (.sid), other image formats with World File (e.g. PNG or BMP), etc.

- Vector: Shapefile (.shp + others), GeoJSON (.geojson, .json), GeoPackage (.gpkg), KML/KMZ (.kml, .kmz), GML (.gml), Esri geodatabase (.gdb), OpenStreetMap (.osm, .pbf), SpatiaLite (.sqlite), CityJSON (.city.json), text-based formats with WKT geometry (e.g. CSV), etc.

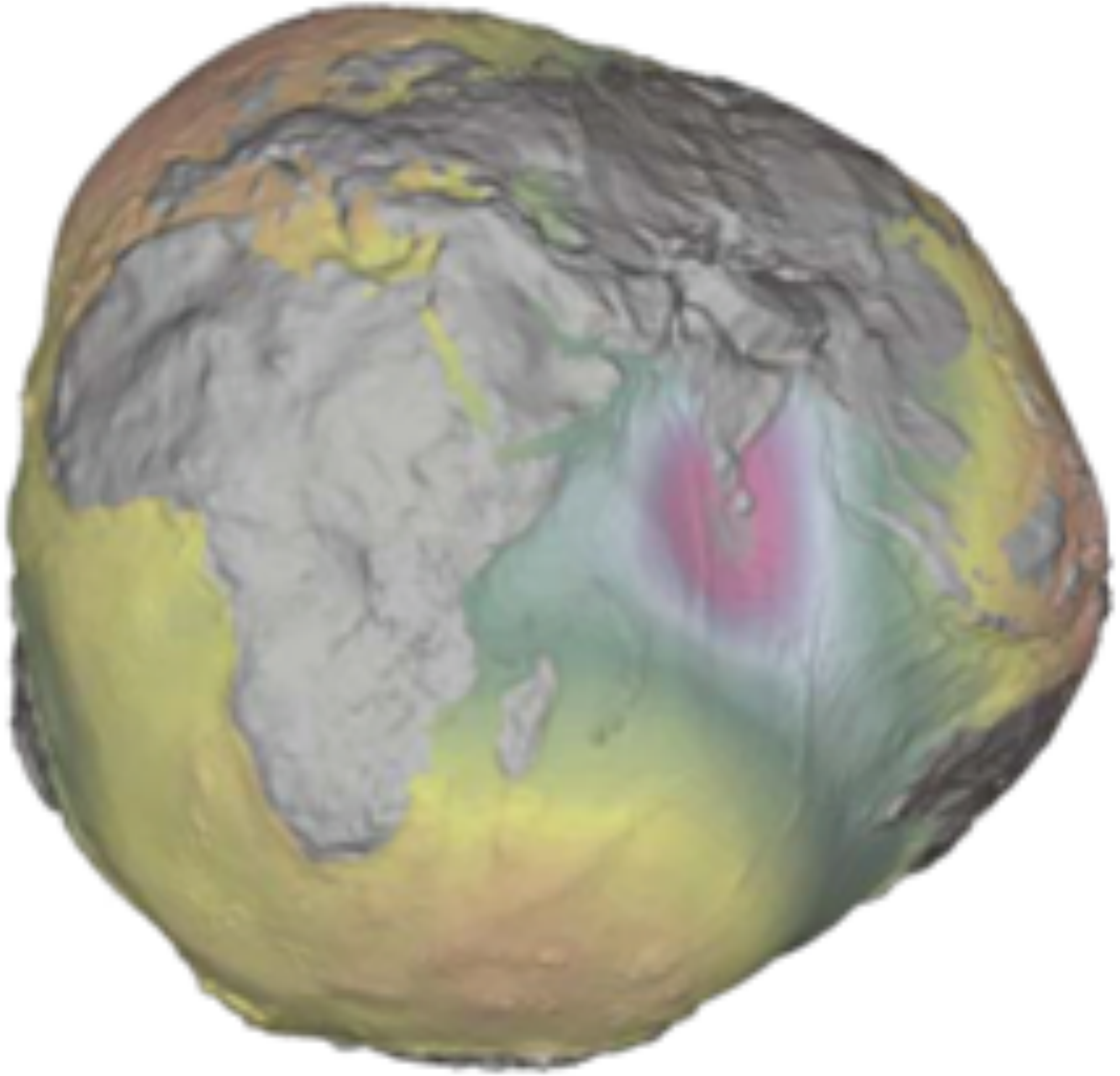
- Why a GIS? What is a GIS?
- Geographic phenomena: fields and objects
- Computer representations: vectors and rasters
- Georeferencing
- QGIS demo & practical session

Georeferencing: Earth’s surface, geoid and ellipsoid



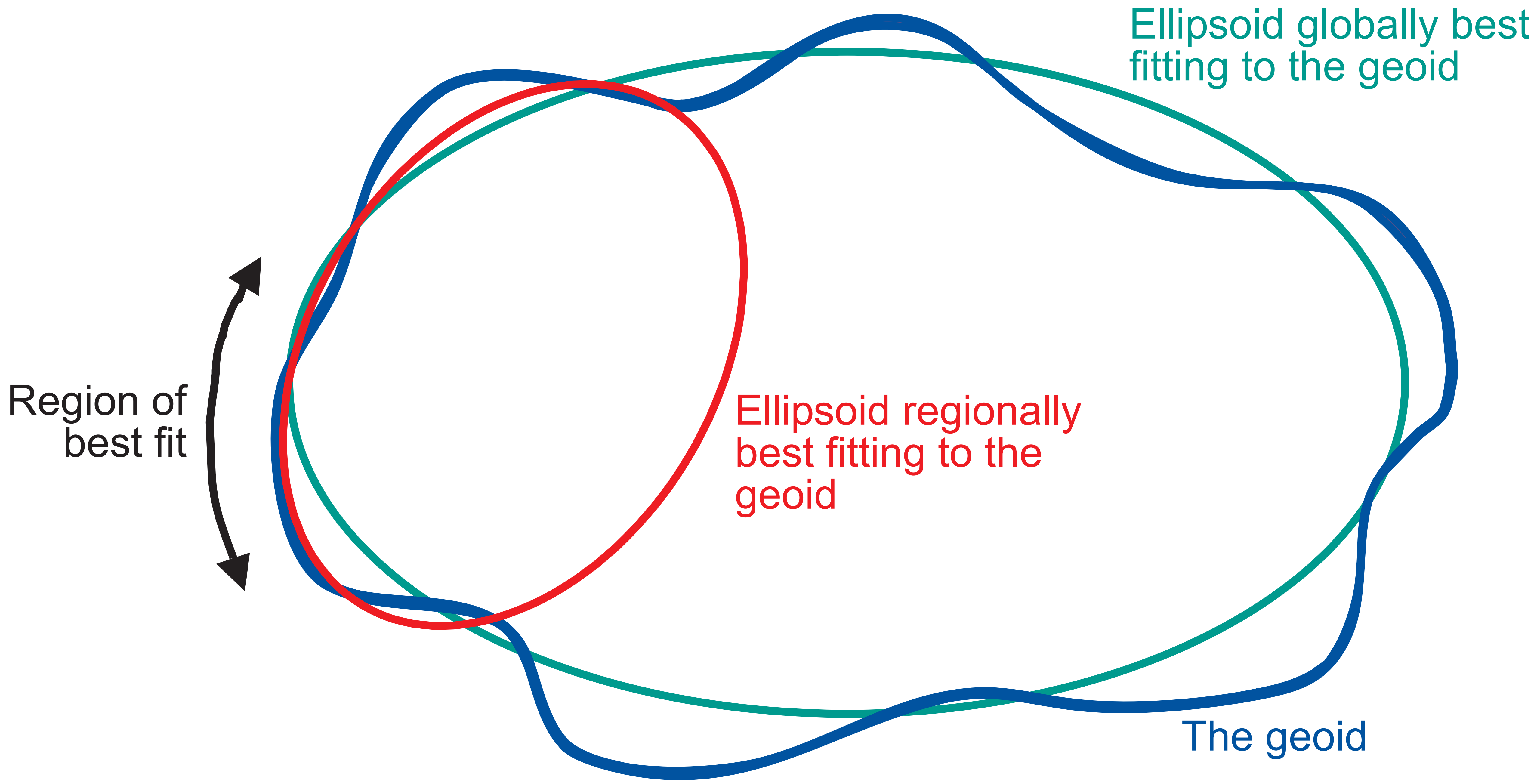
Georeferencing: geoid

- **Geoid**: equipotential surface of the Earth, i.e. shape of the Earth's sea level without tides, currents or wind if water could freely pass through continents. Typically used for *heights*.

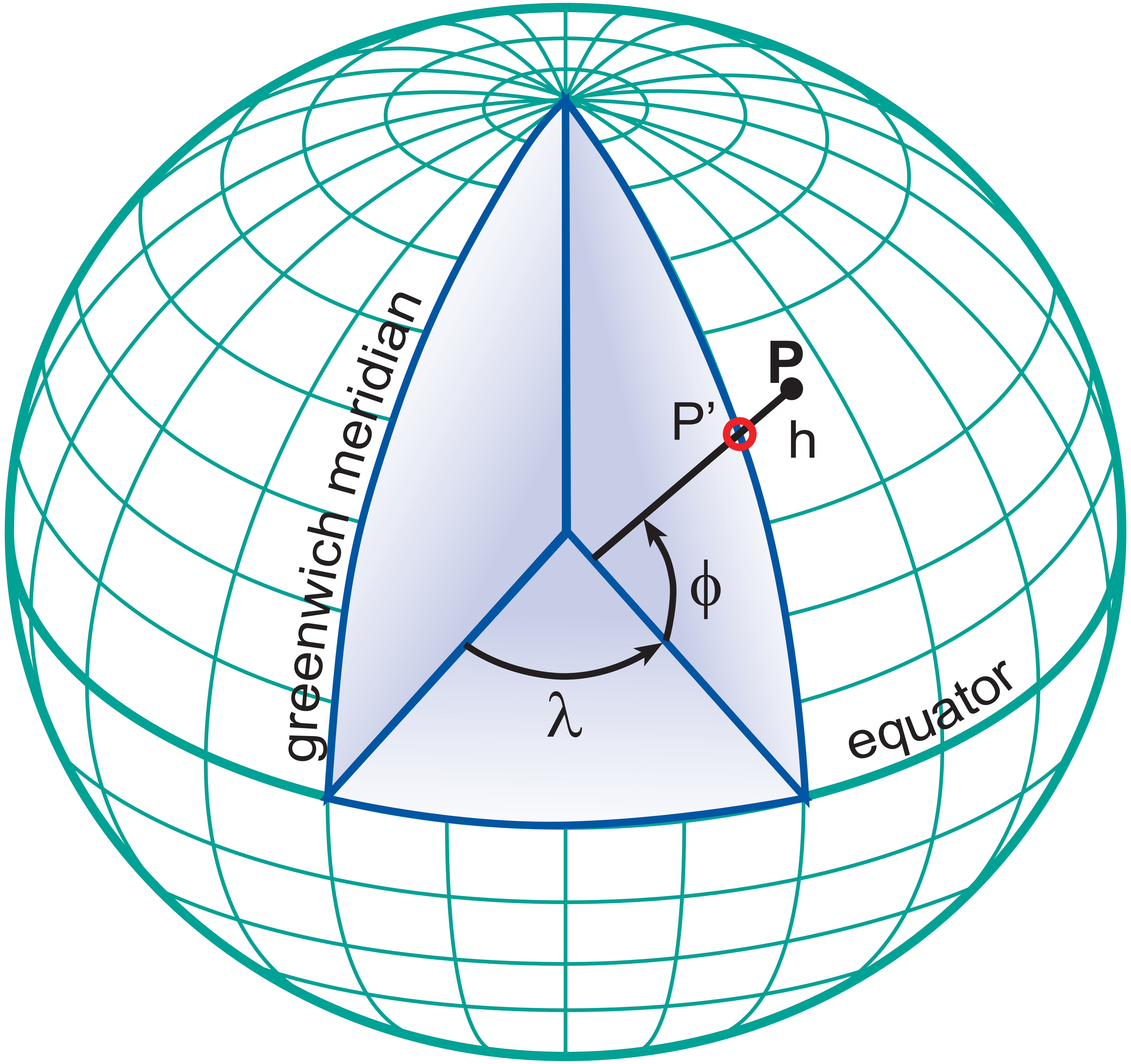


Georeferencing: ellipsoid

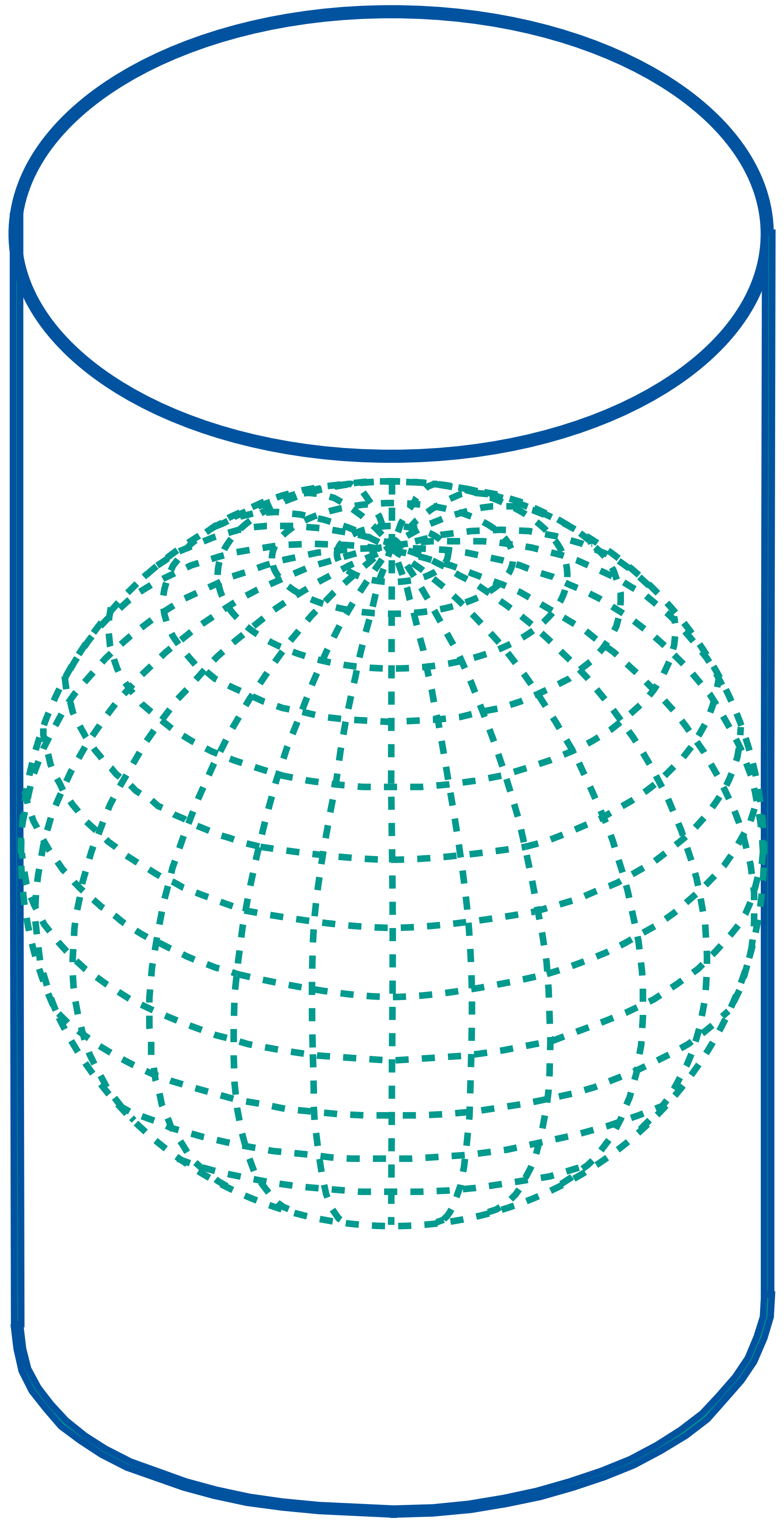
- **Ellipsoid**: most convenient mathematical object that resembles (part of) the geoid with reasonable accuracy. Typically used for *horizontal coordinates*.



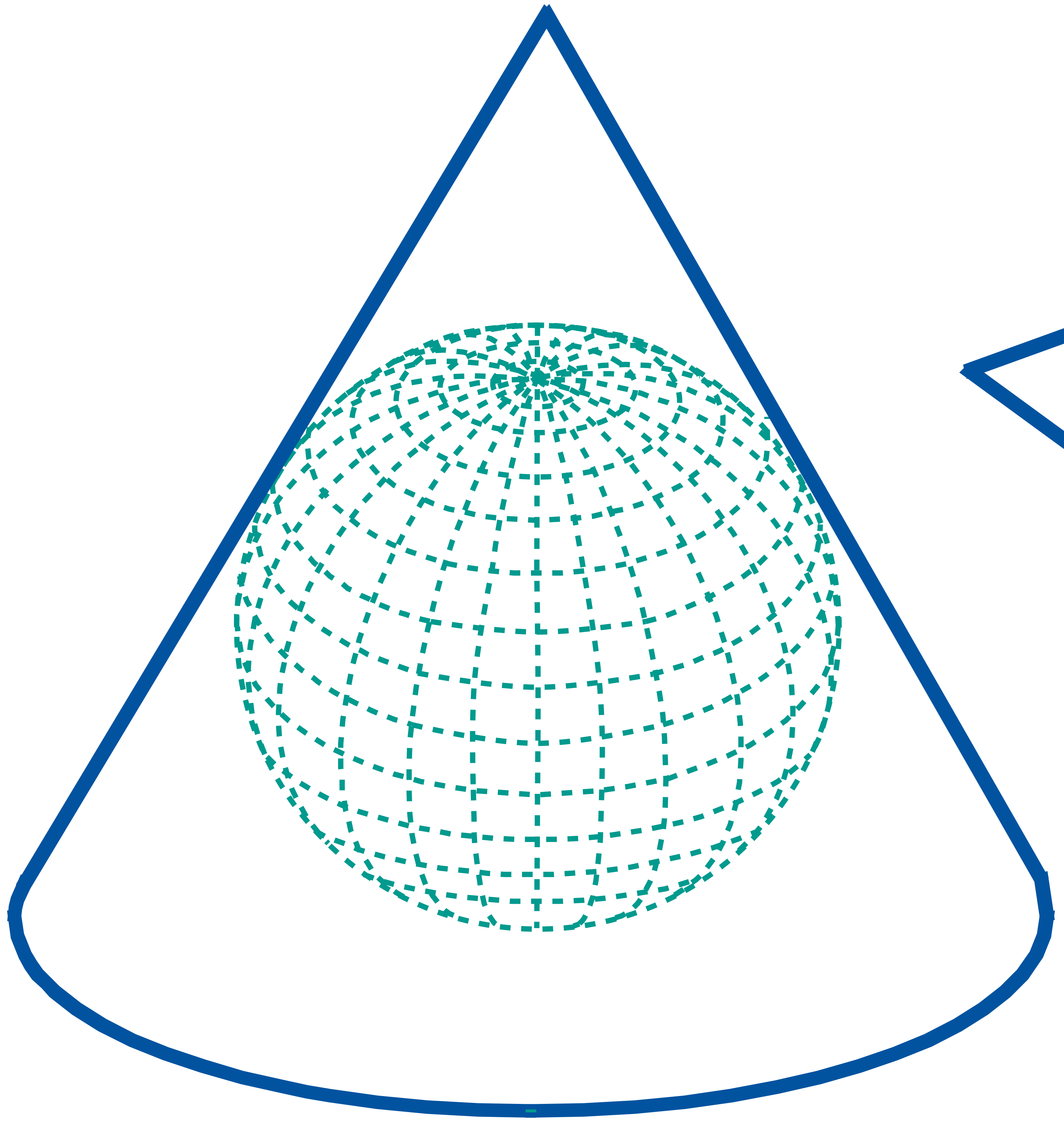
Georeferencing: 3D geographic coordinates



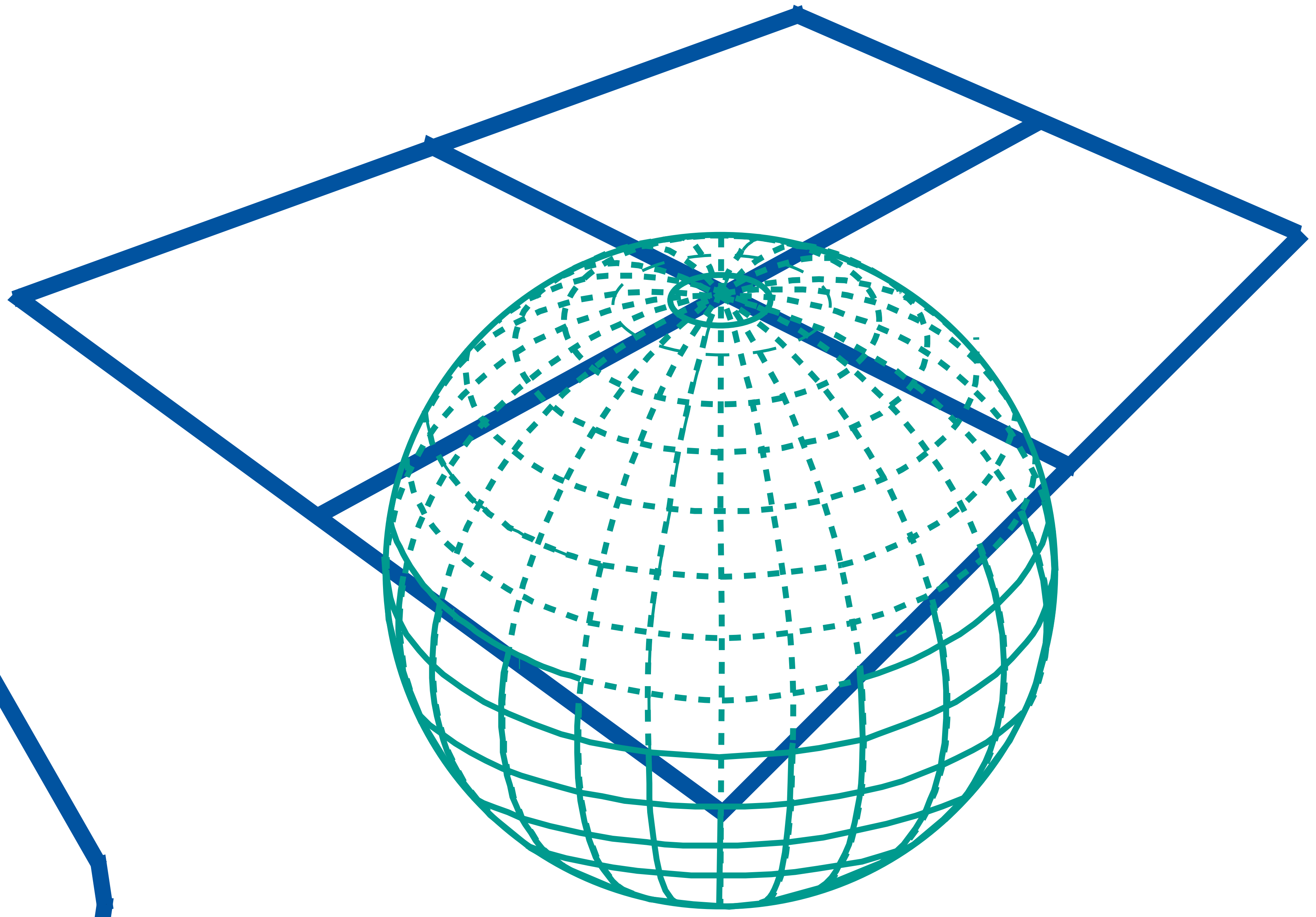
Georeferencing: from an ellipsoid to a surface



Cylindrical

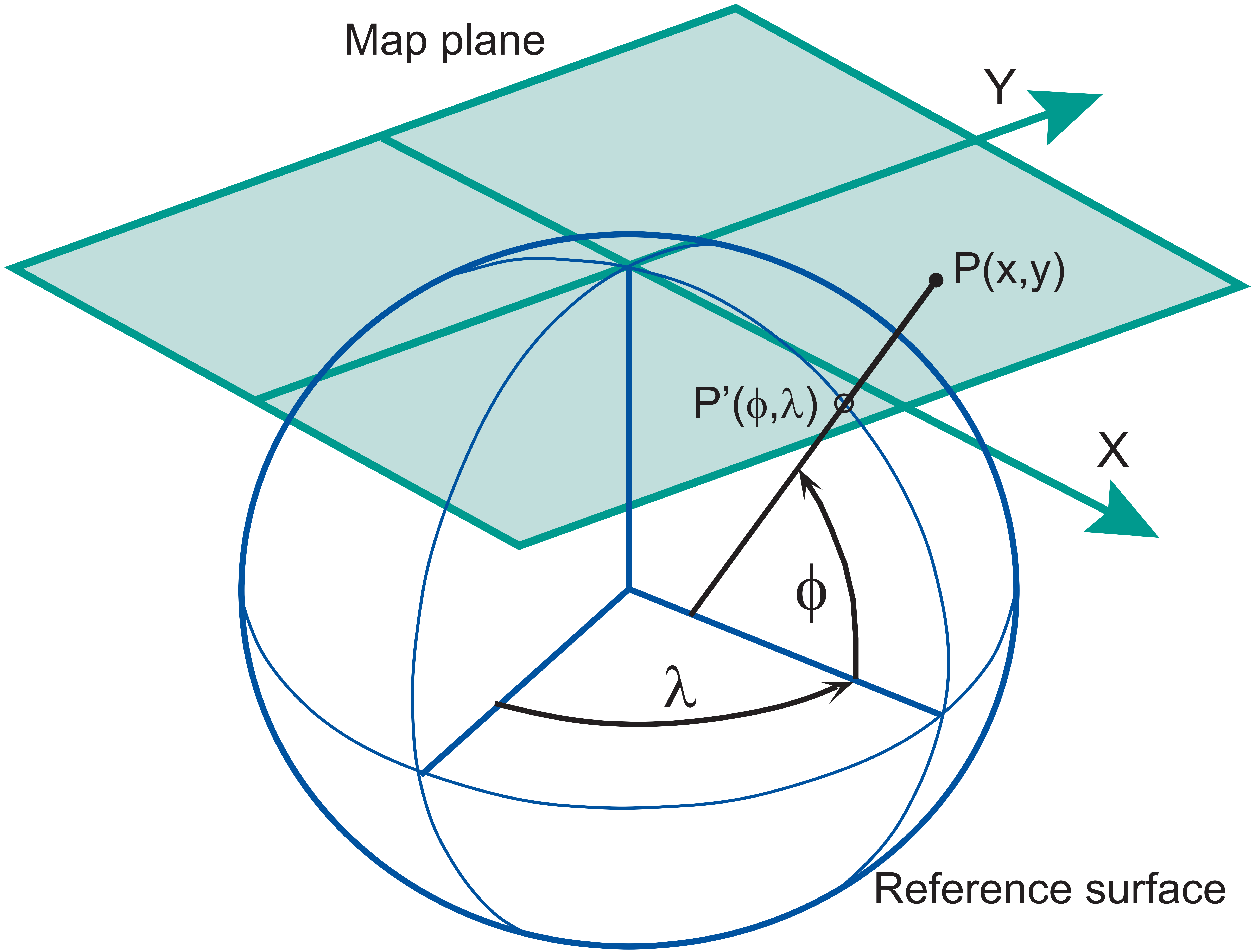


Conical

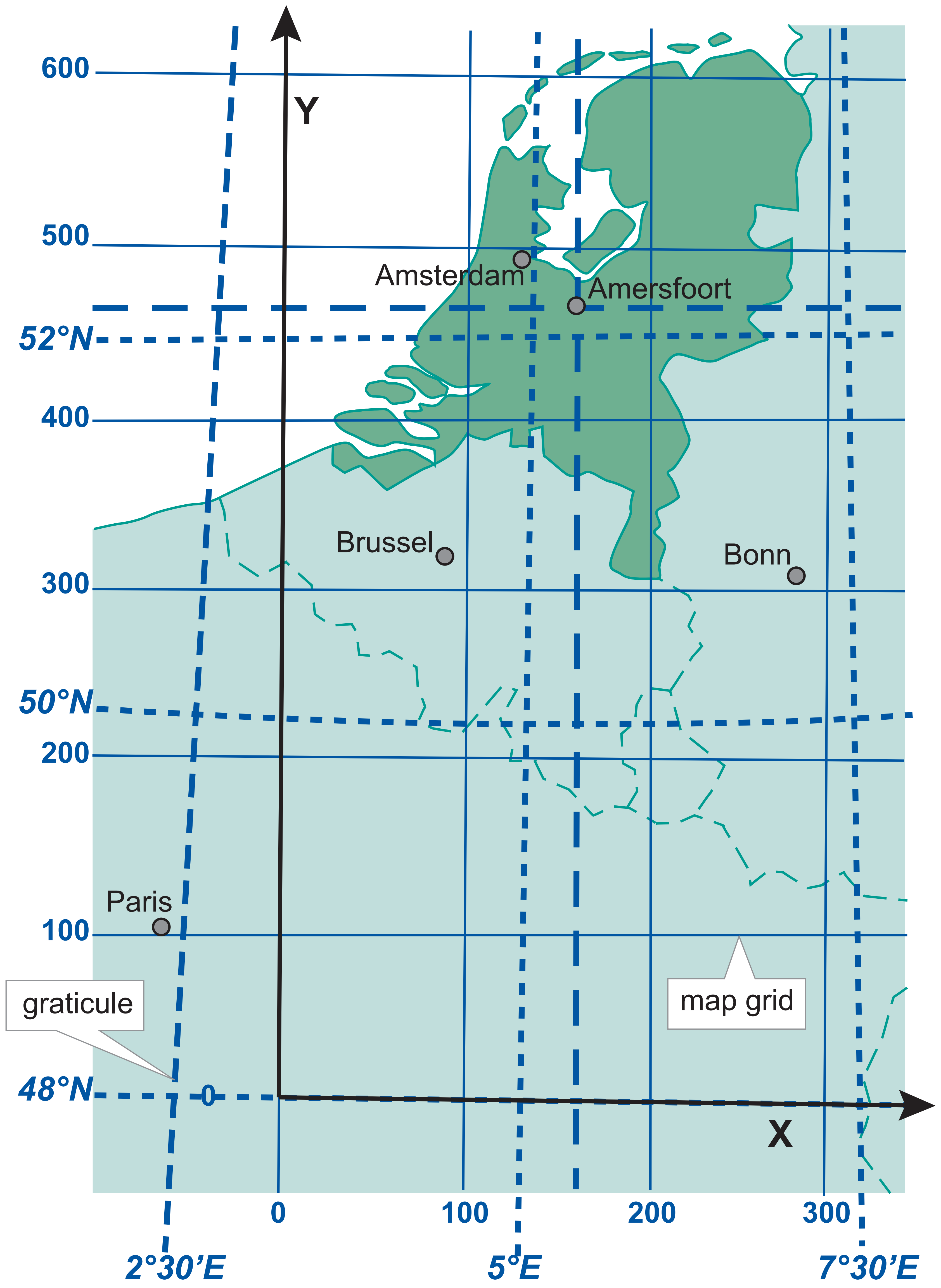


Azimuthal

Georeferencing: from 3D geographic to Cartesian coordinates



Georeferencing: from 3D geographic to Cartesian coordinates



Georeferencing: EPSG

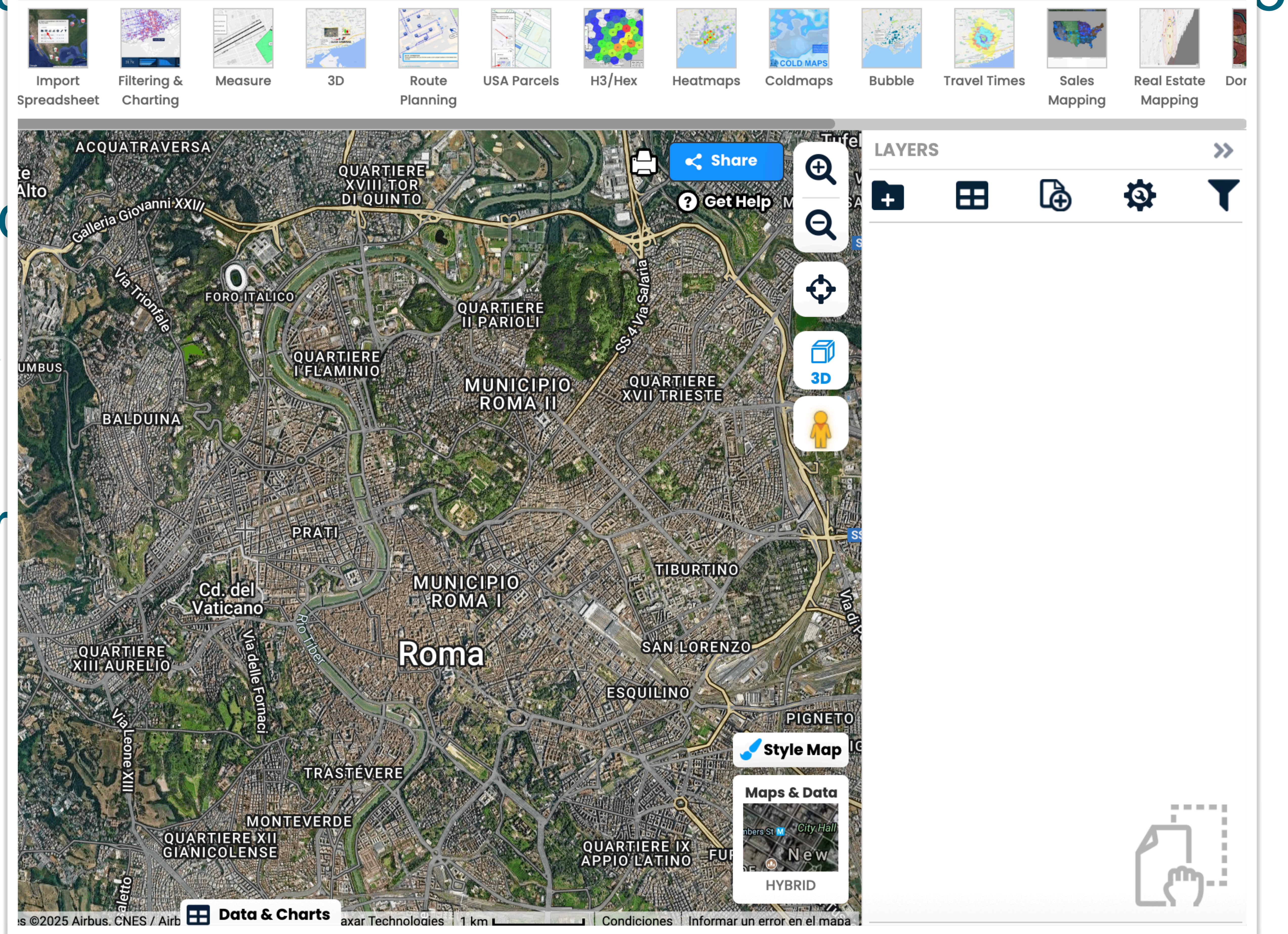
- Many parameters to store: ellipsoid (semi-major axis, semi-minor axis, Equatorial plane, prime meridian), projection surface, units, etc.

- Collectively known as a *geodetic datum*.

```
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]]
```

Georeferencing: EPSG

- All parameters can be stored conveniently using an EPSG code. Some examples:
  - 4326: WGS84 (latitu
  - 3857: Web Mercator
  - 3035: ETRS89 (EU-r
  - 4269: NAD83 (North
- Check: epsg.io



Questions?

- Why a GIS? What is a GIS?
- Geographic phenomena: fields and objects
- Computer representations: vectors and rasters
- Georeferencing
- QGIS demo & practical session

Time for a QGIS demo!

Practical session 1

- Create a map of a region of your interest (e.g. your hometown or a place where you want to travel to). Show some of its interesting features :-).
- Use 2-3 datasets. Some ideas: cities, land use, elevation, hydrology, roads.
- Some ideas for Portugal: dados.gov.pt or individual websites of Direção-Geral do Território (DGT), Instituto Nacional de Estatística (INE), SNIG (Sistema Nacional de Informação Geográfica), etc.
- Use a nice colour scheme, readable symbols, etc.

Module 5 contents

- Foundations of 2D and 3D GIS
- Processing 2D and 3D geodata
- 3D city models

Processing 2D and 3D geodata contents

- Data preparation
- Spatial analysis
- Practical session

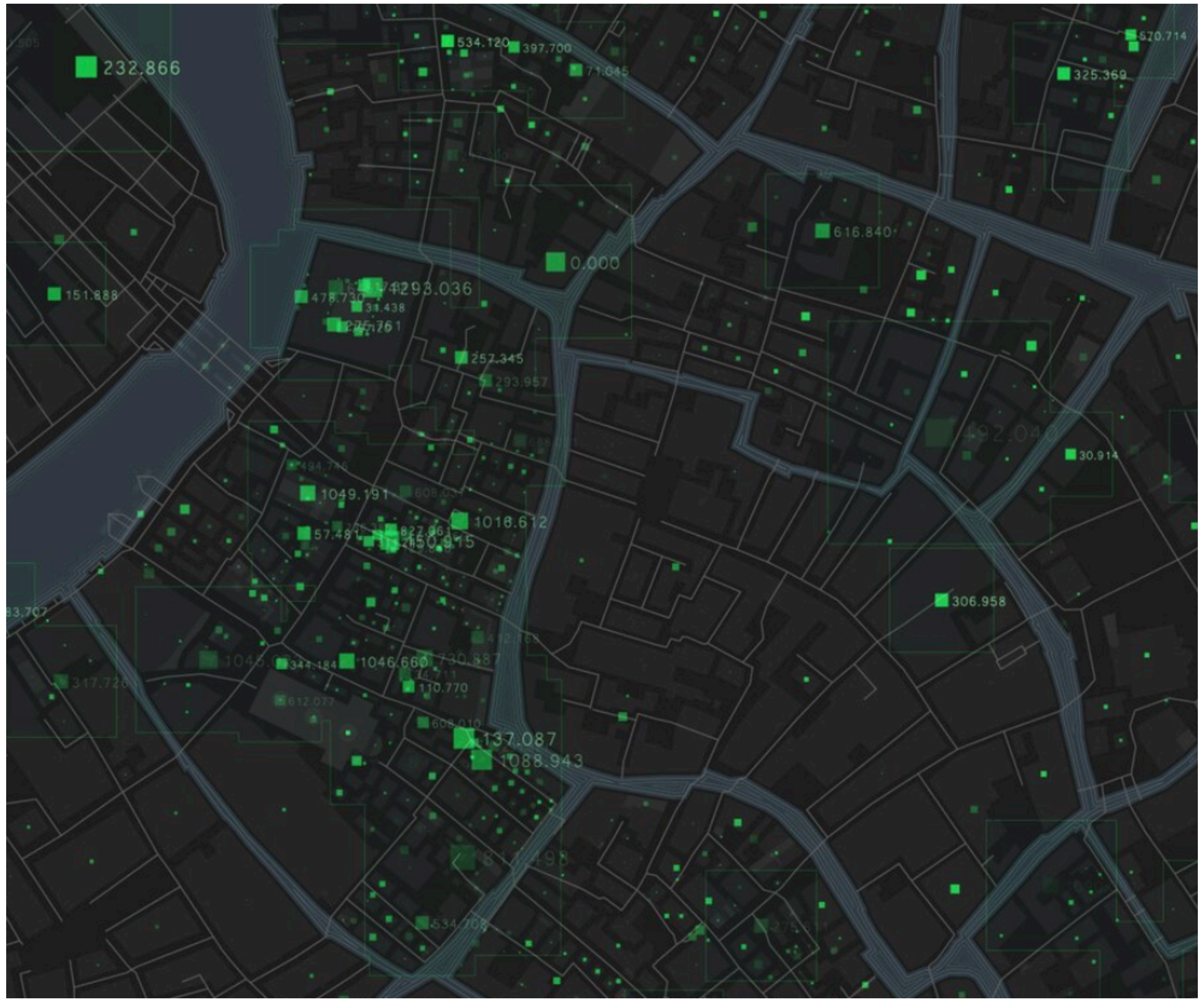
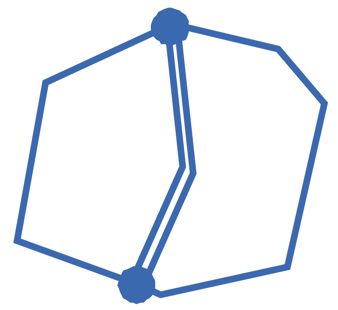
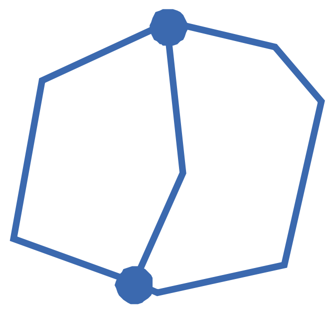
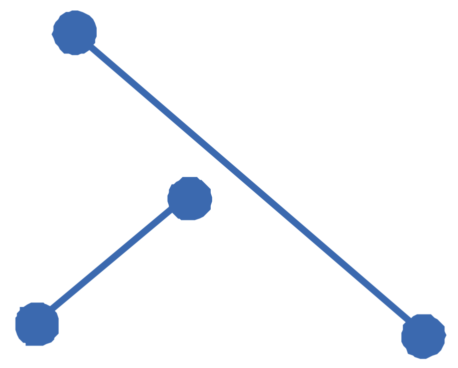
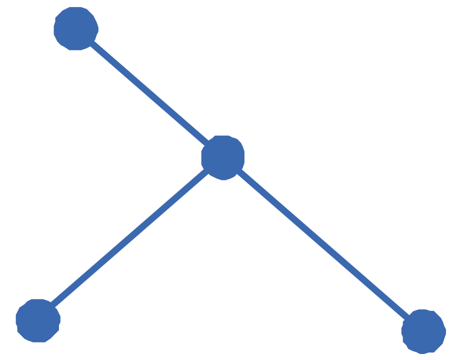
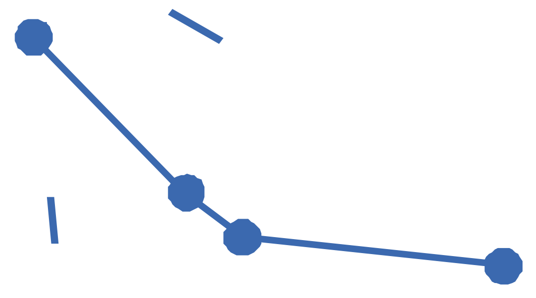
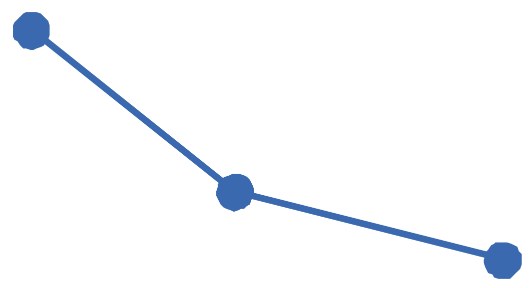
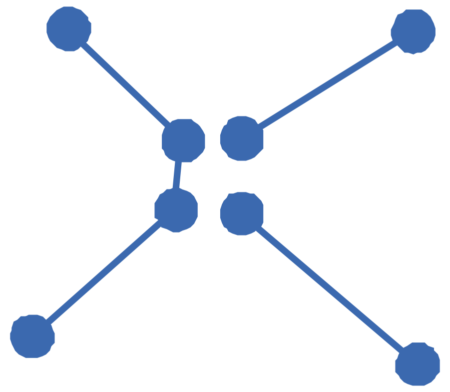
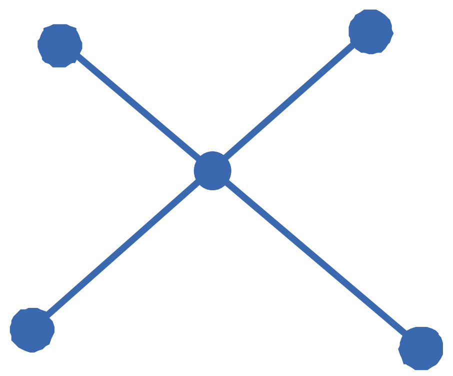
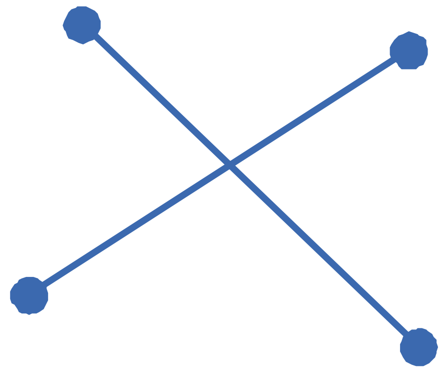
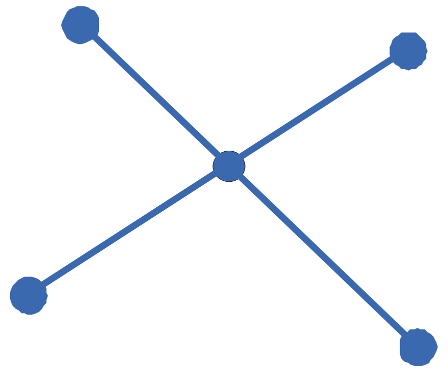
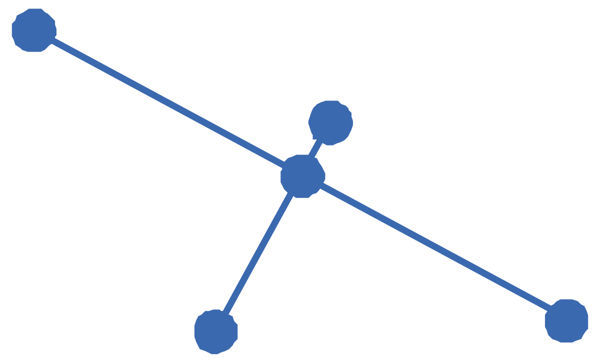
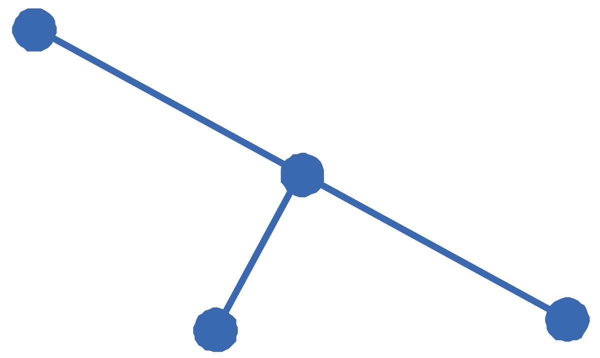
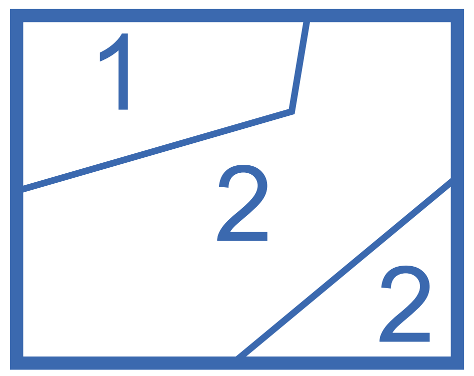
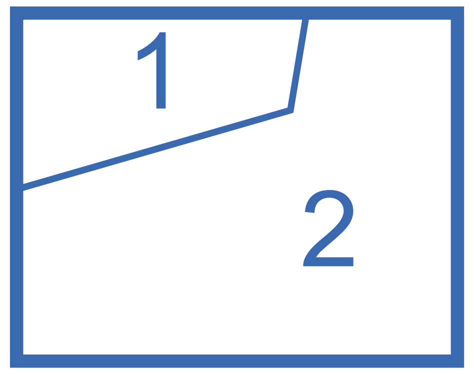
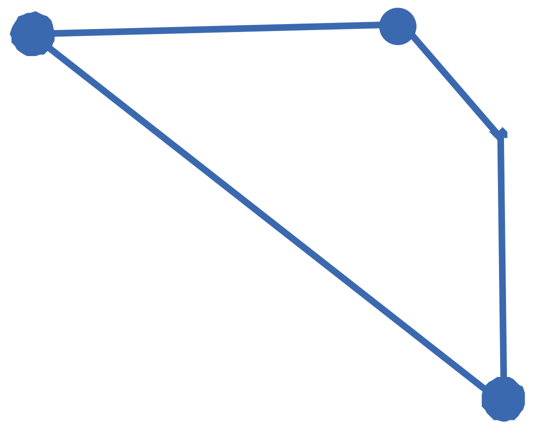
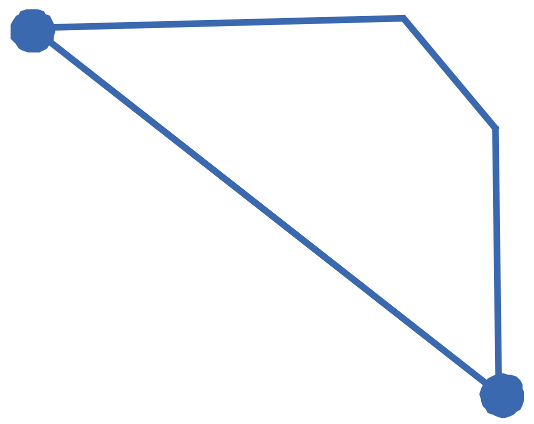


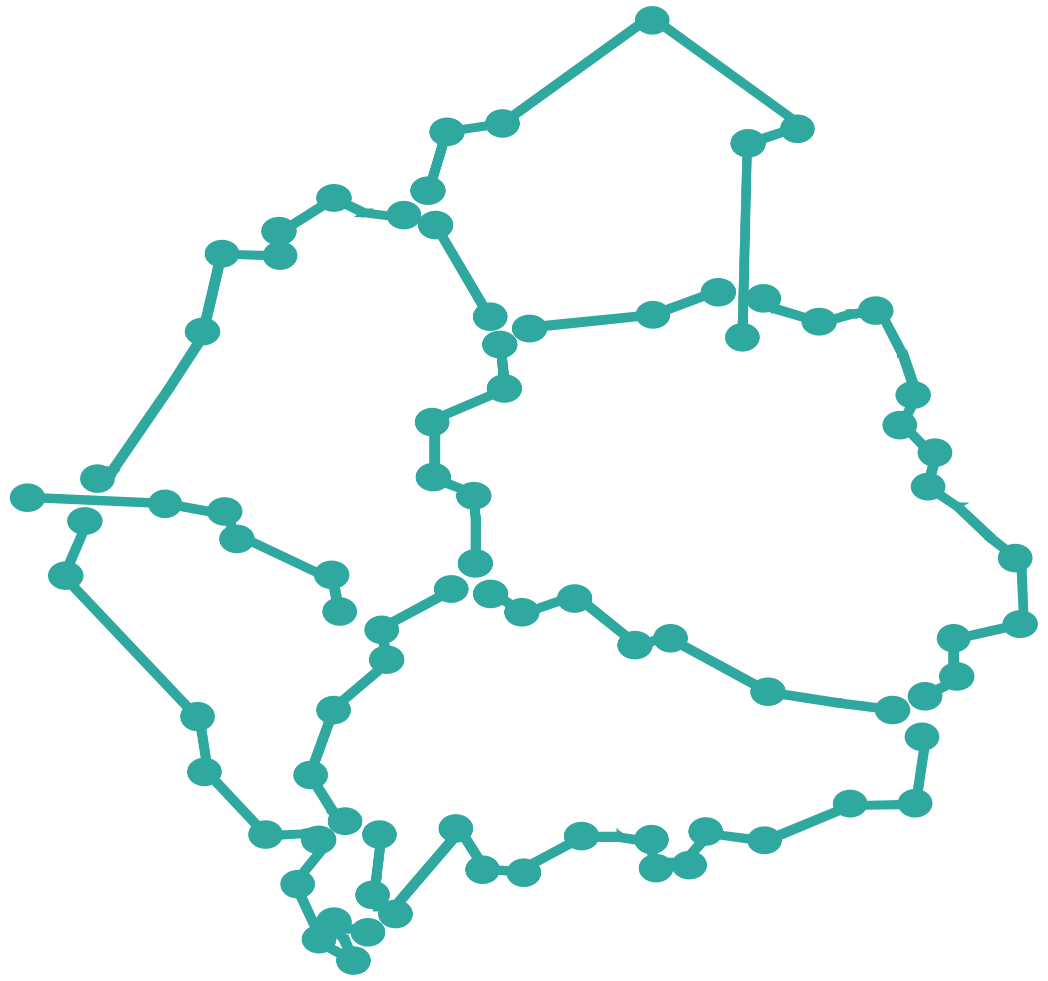
Image: Freepik

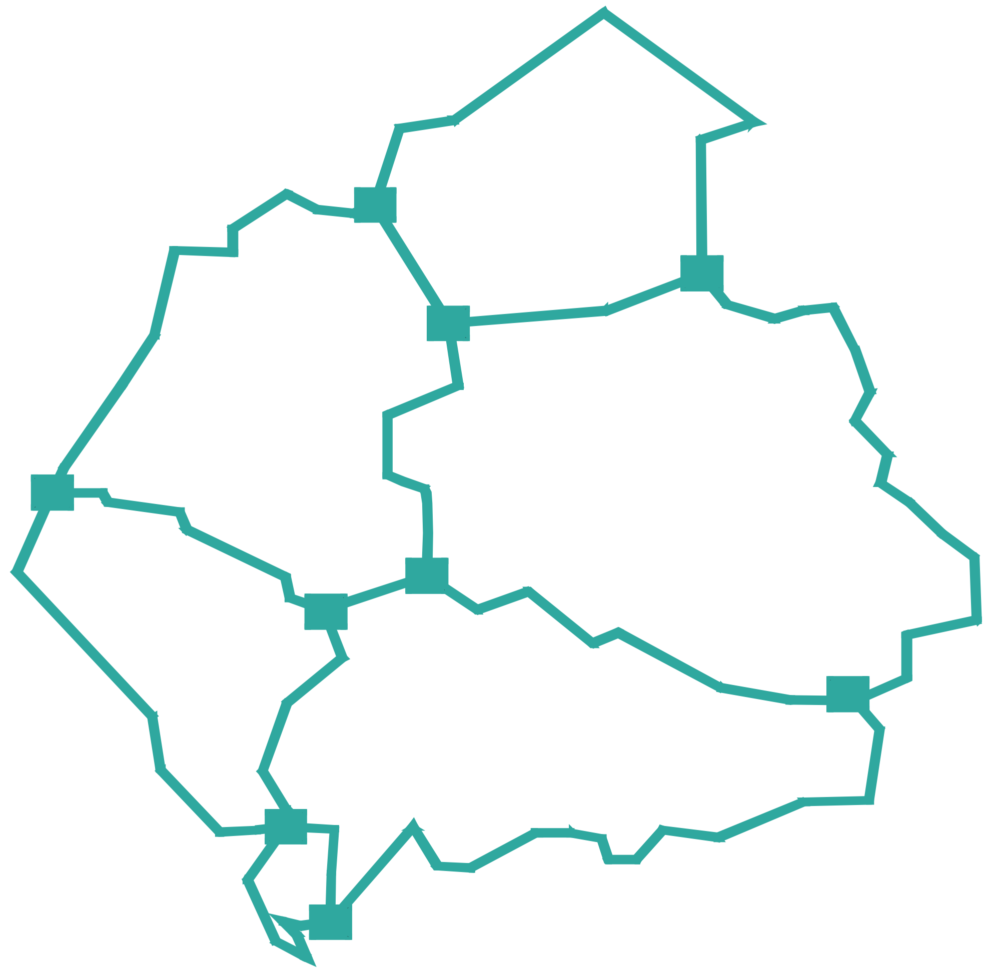
CHEK DIGITAL BUILDING PERMIT

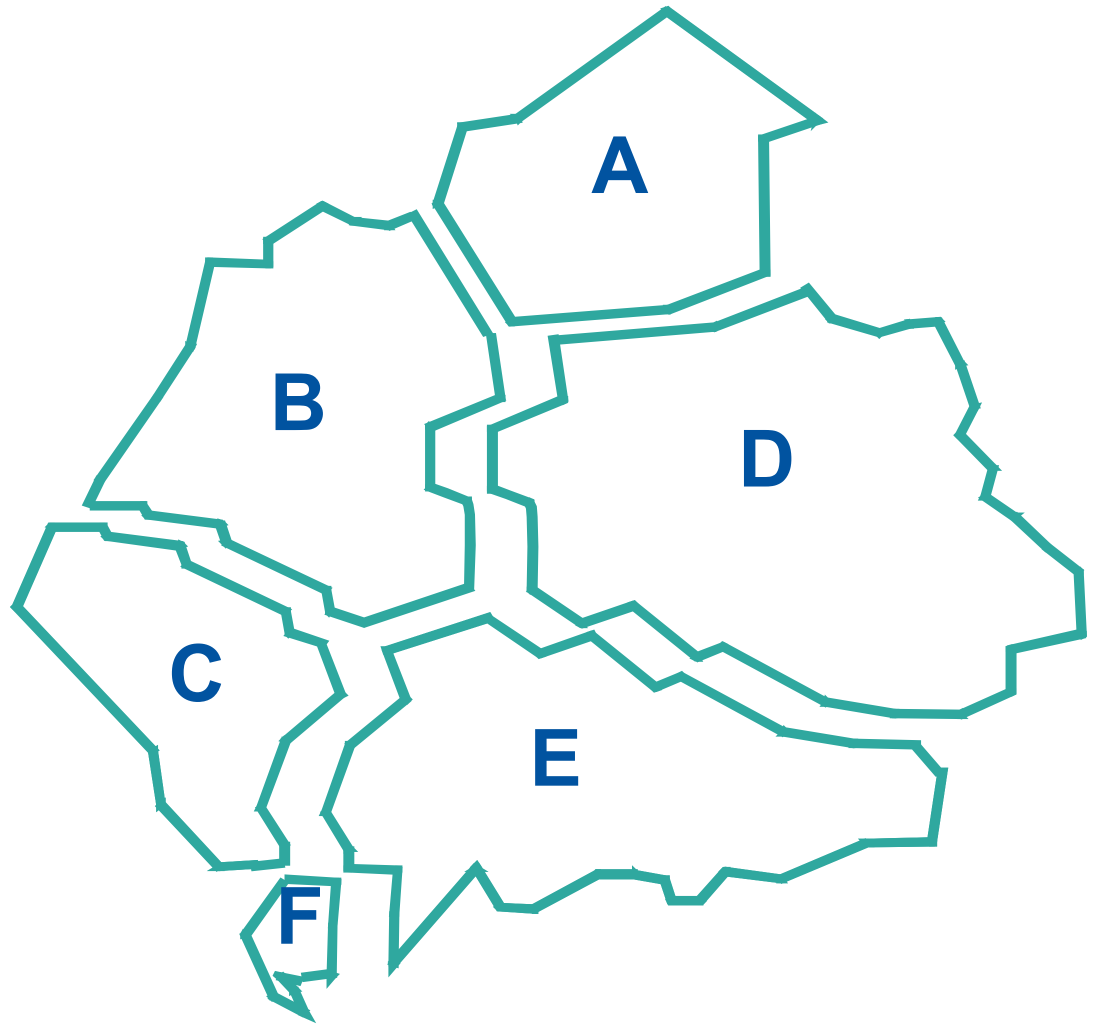
Data preparation: cleanup

Before cleanup	After cleanup	Description	Before cleanup	After cleanup	Description
		Erase duplicates or sliver lines			Extend undershoots
		Erase short objects			Snap clustered nodes
		Break crossing objects			Erase dangling objects or overshoots
		Dissolve polygons			Dissolve nodes into vertices

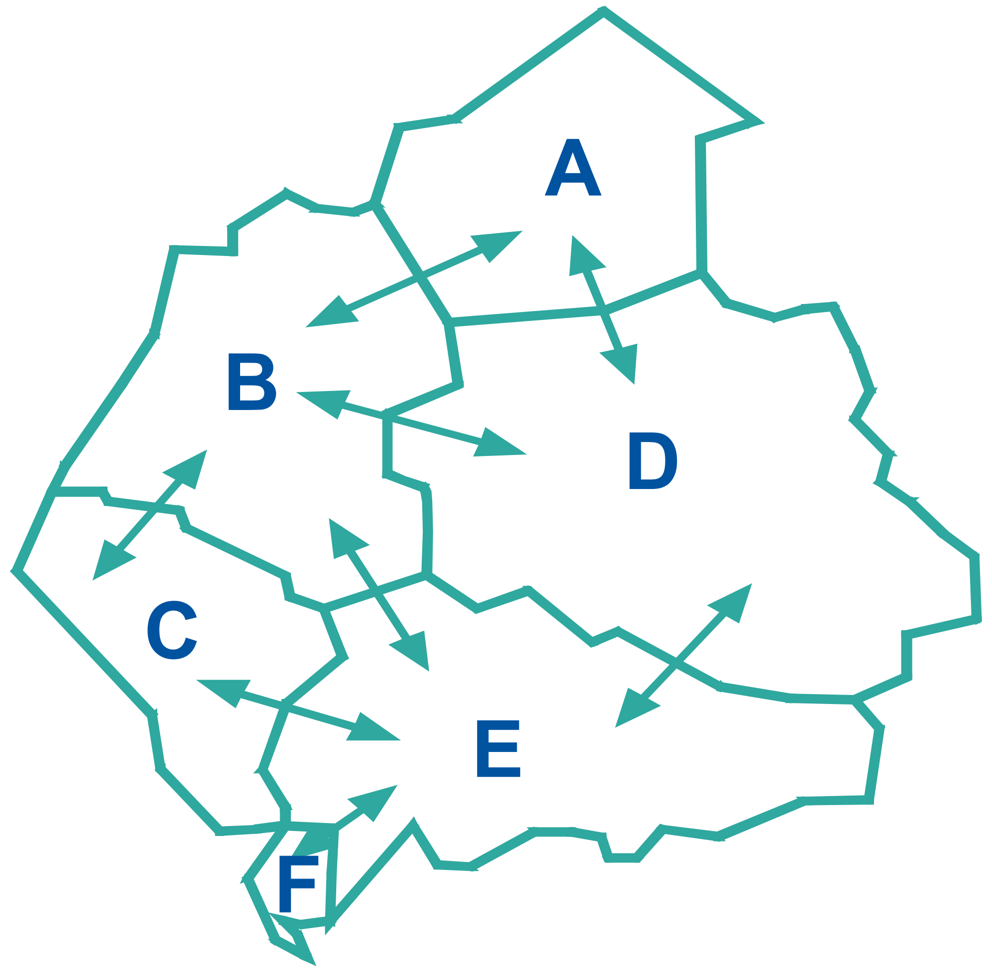
Data preparation: typical workflow


(a) Spaghetti data


(b) Spaghetti data (cleaned)

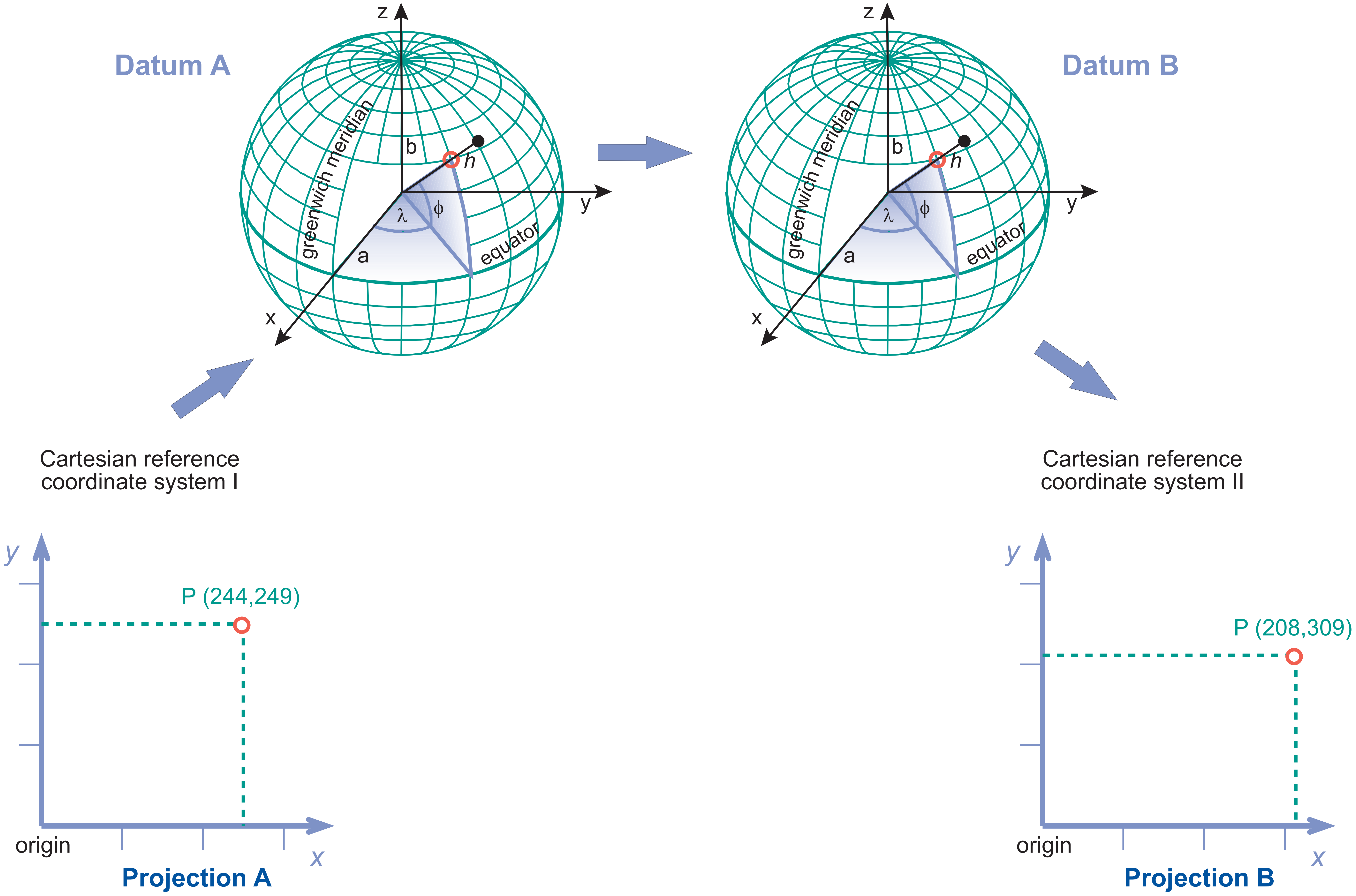


(c) Polygons

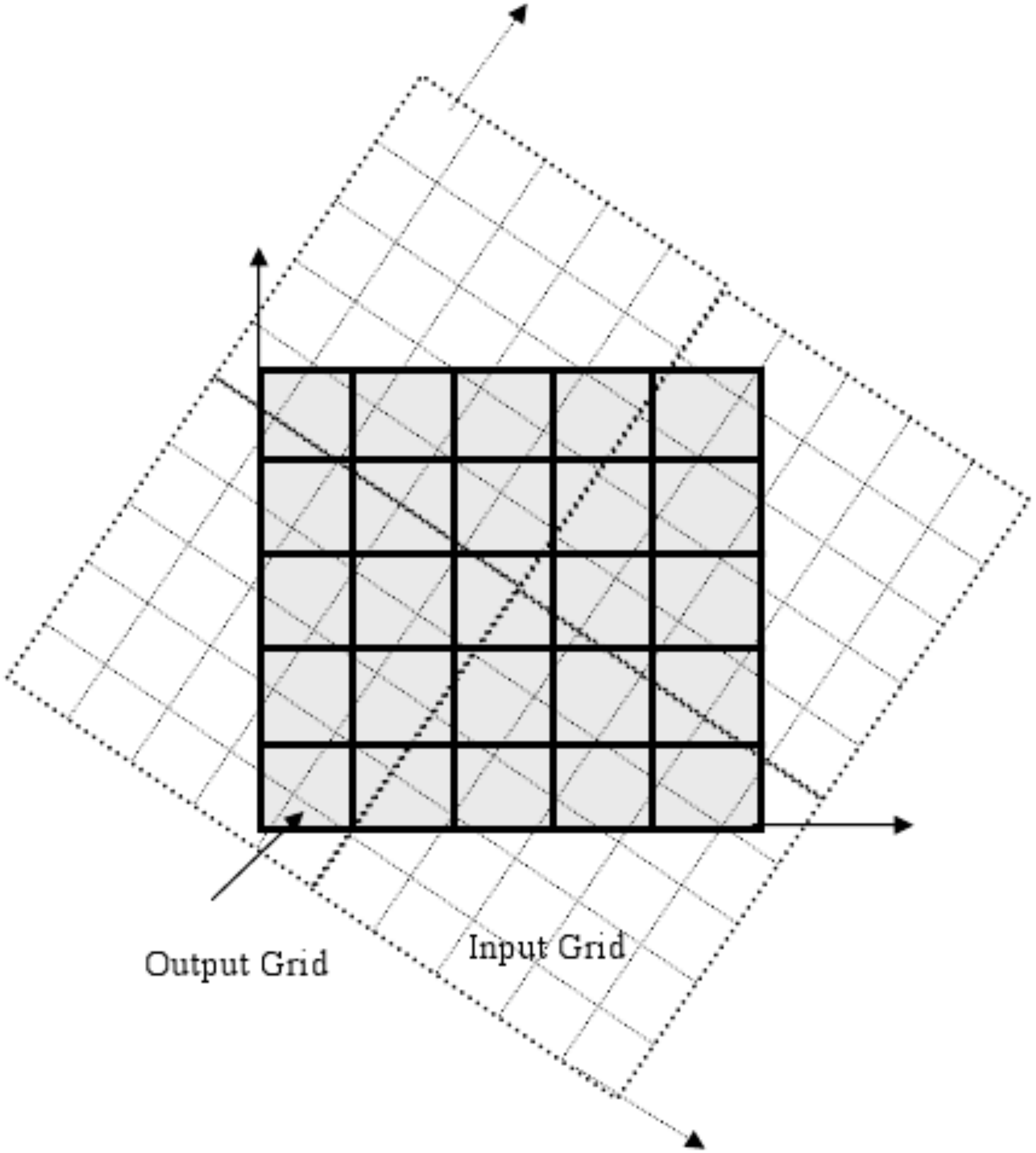


(d) Topology

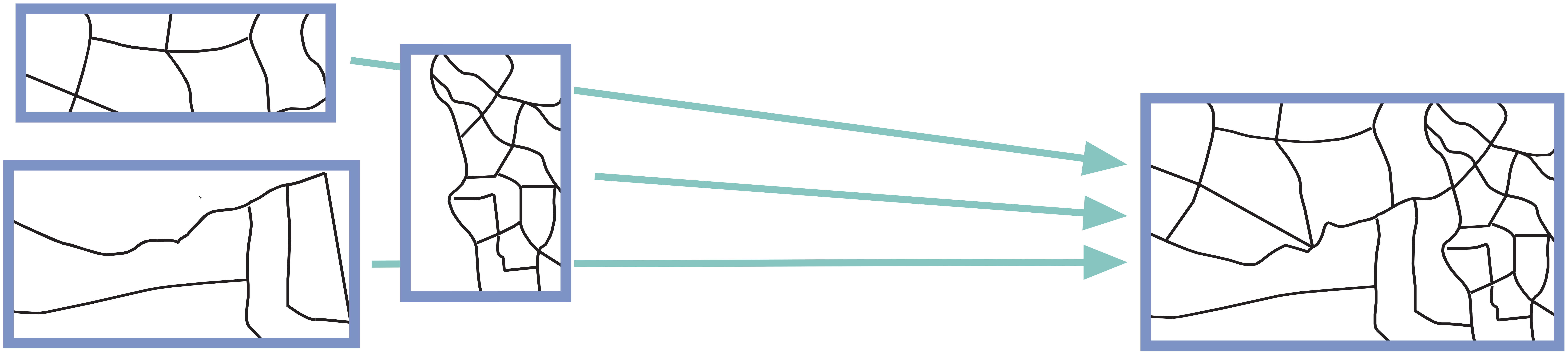
Data preparation: reprojection



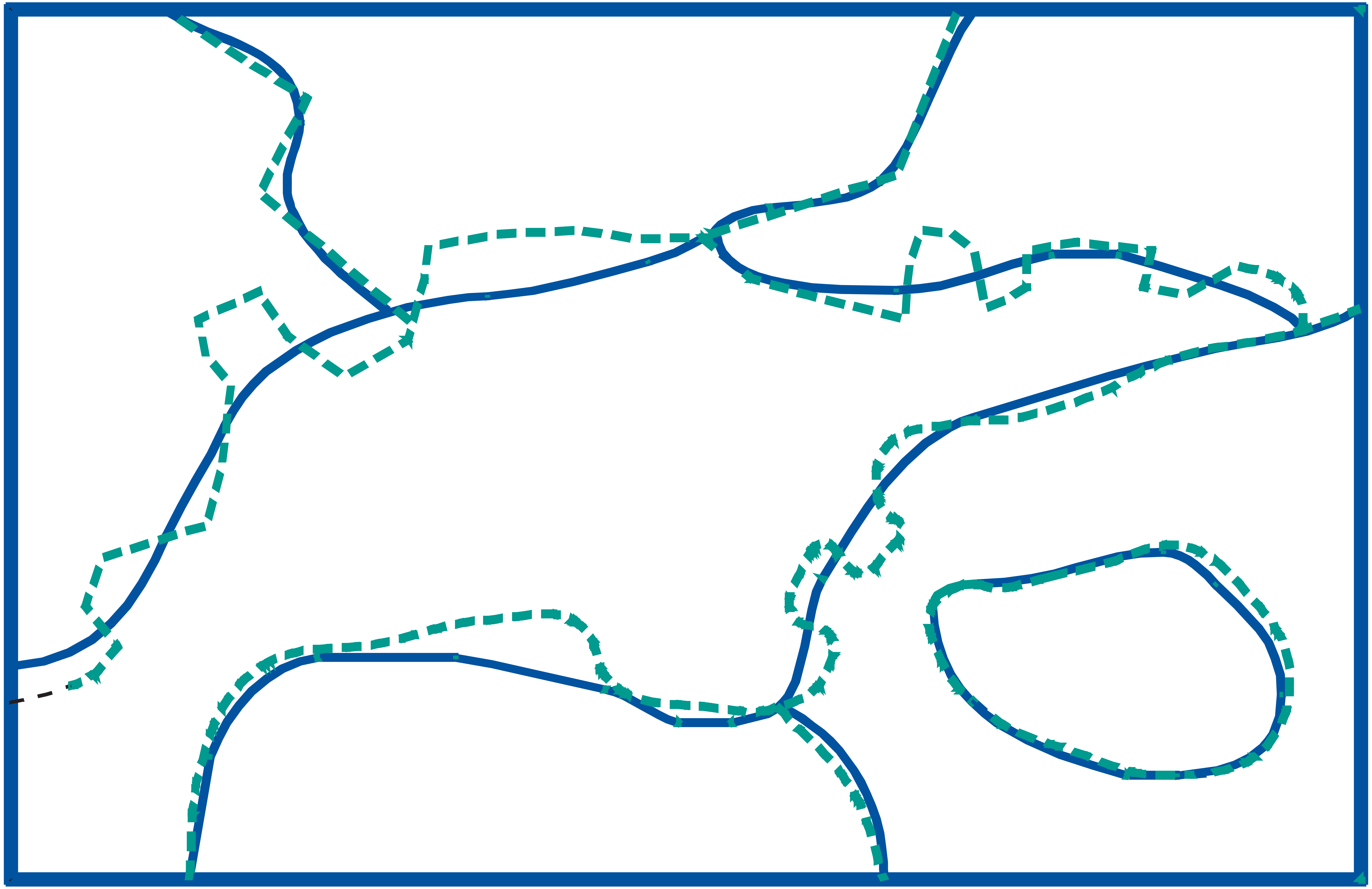
Data preparation: raster resampling



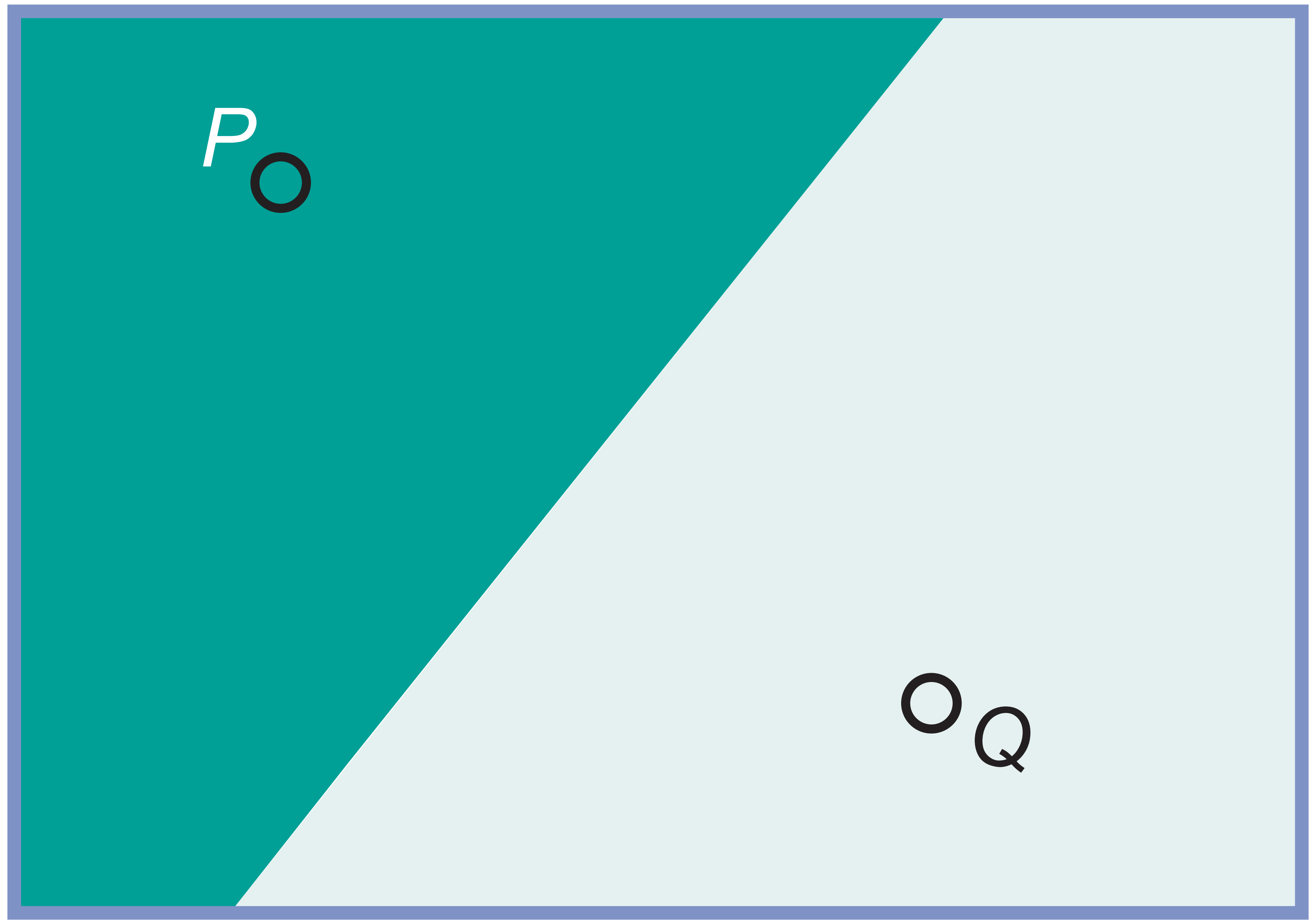
Data preparation: combining datasets

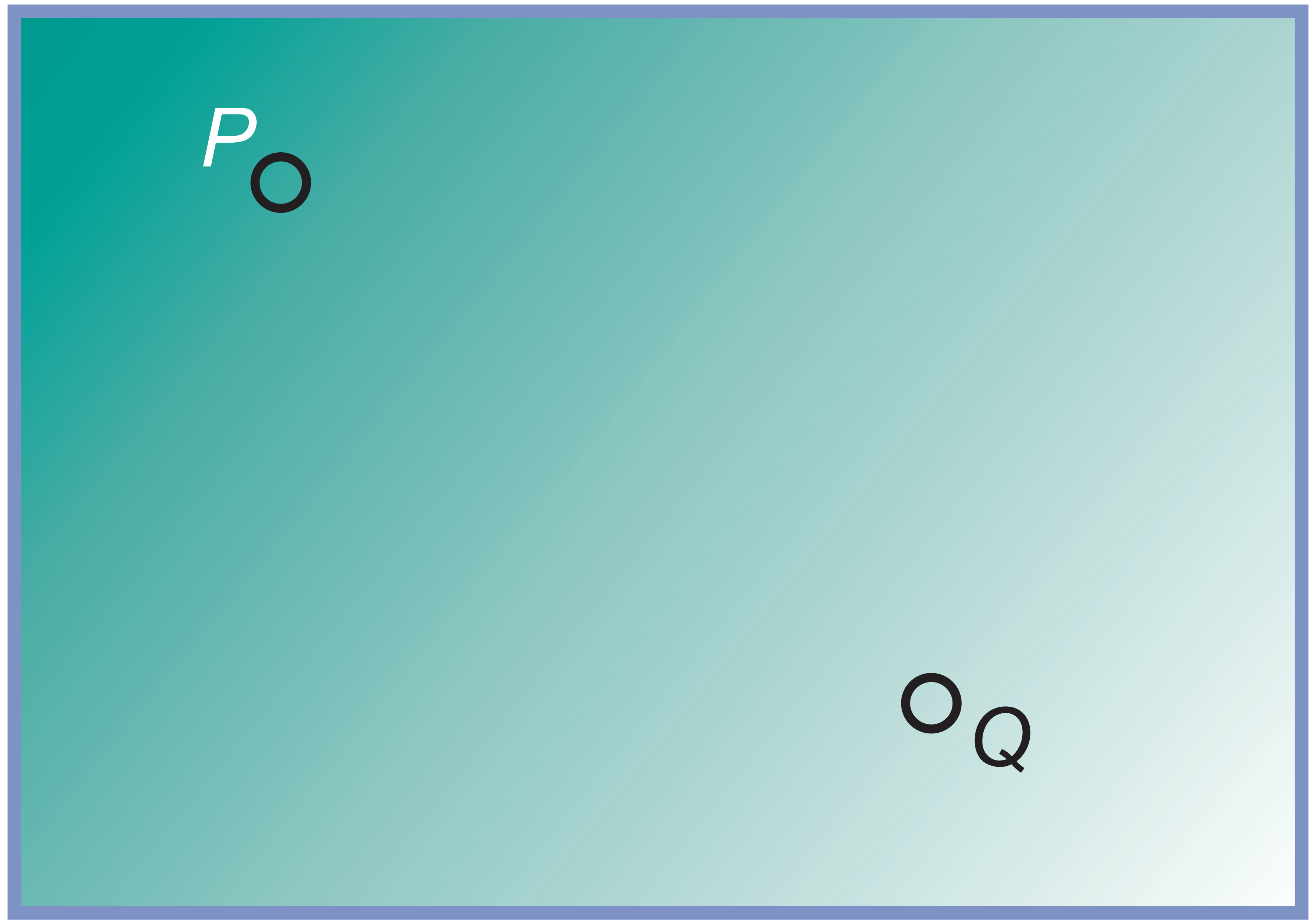


Data preparation: combining datasets

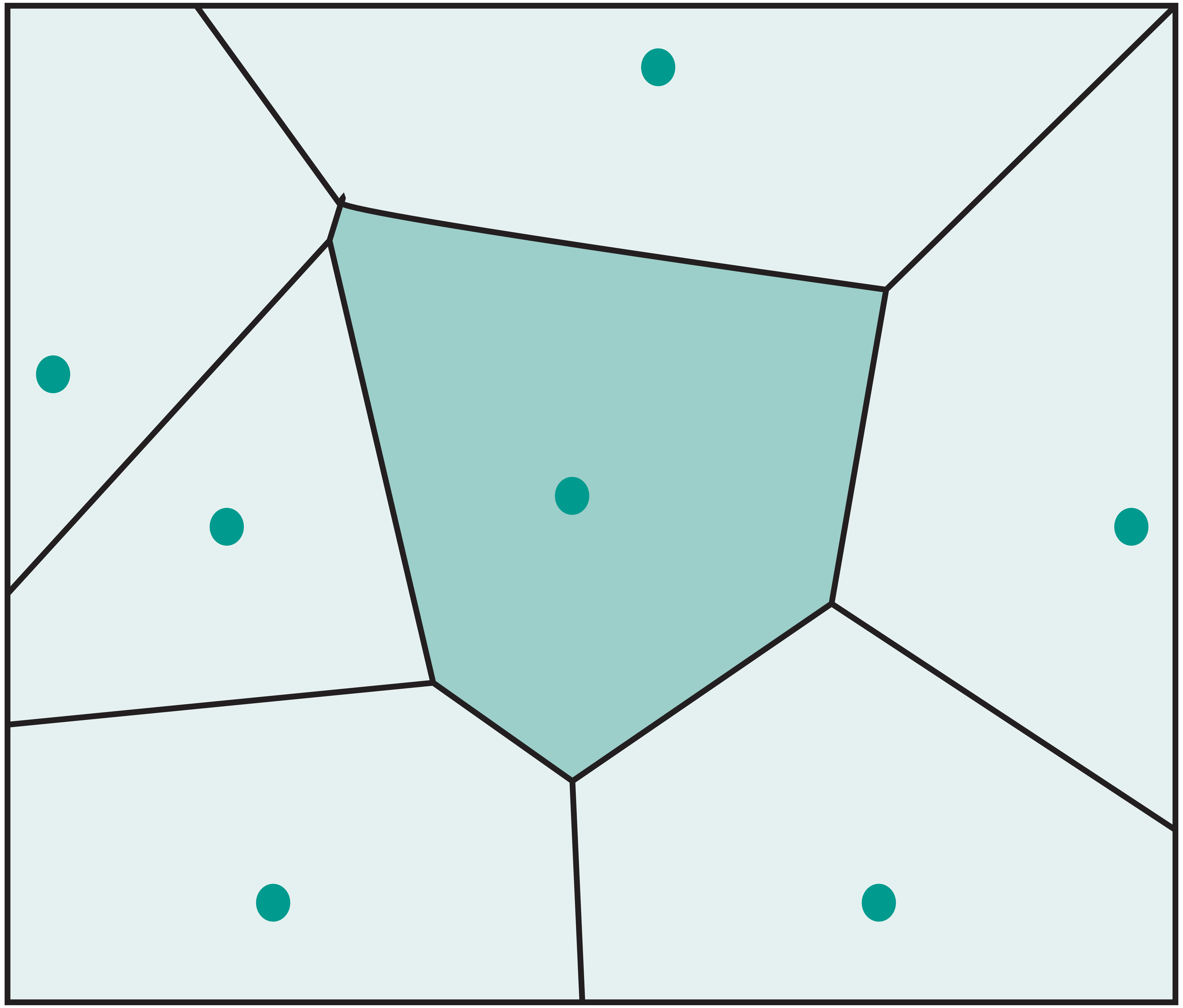


Data preparation: interpolation from points

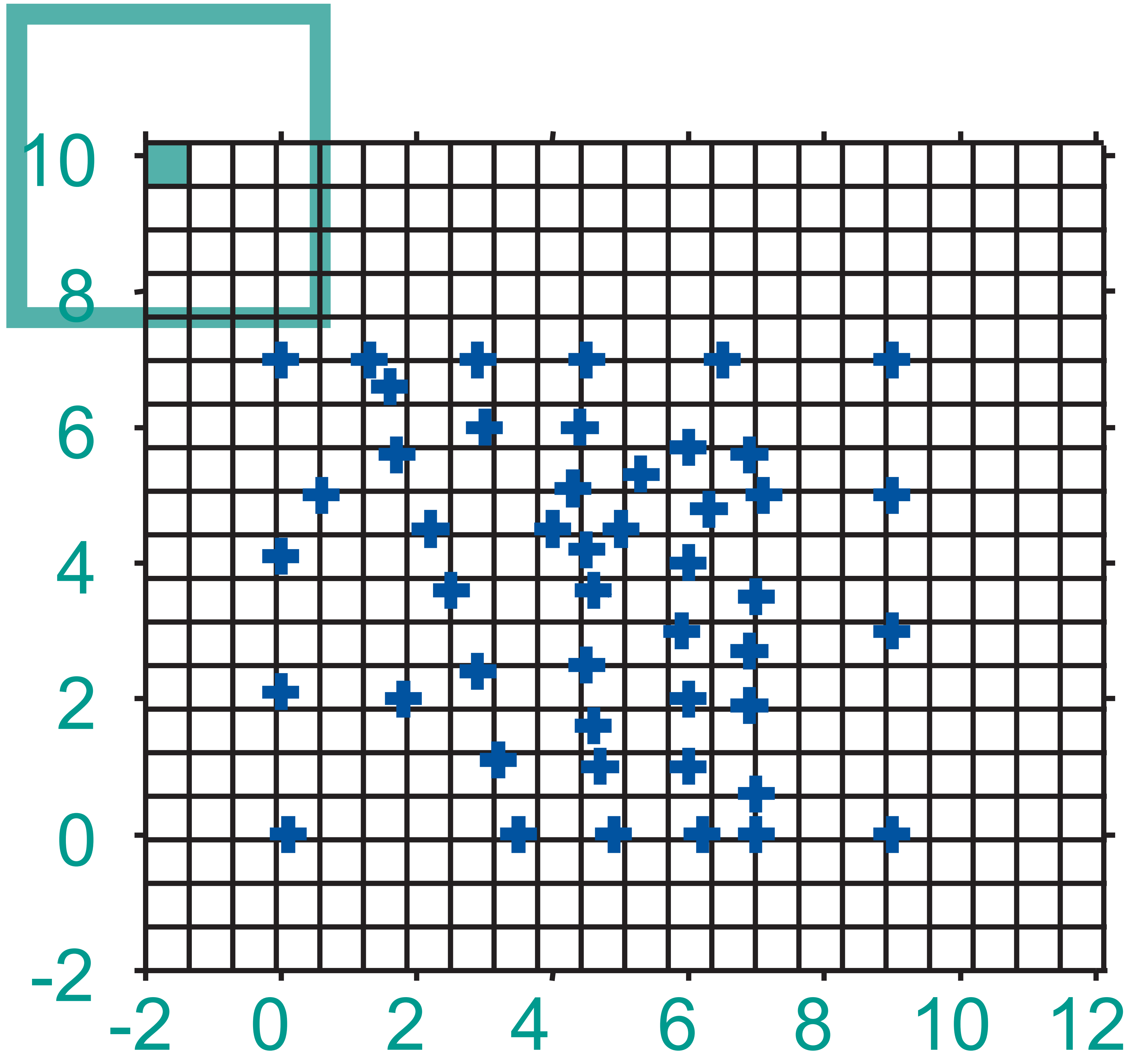


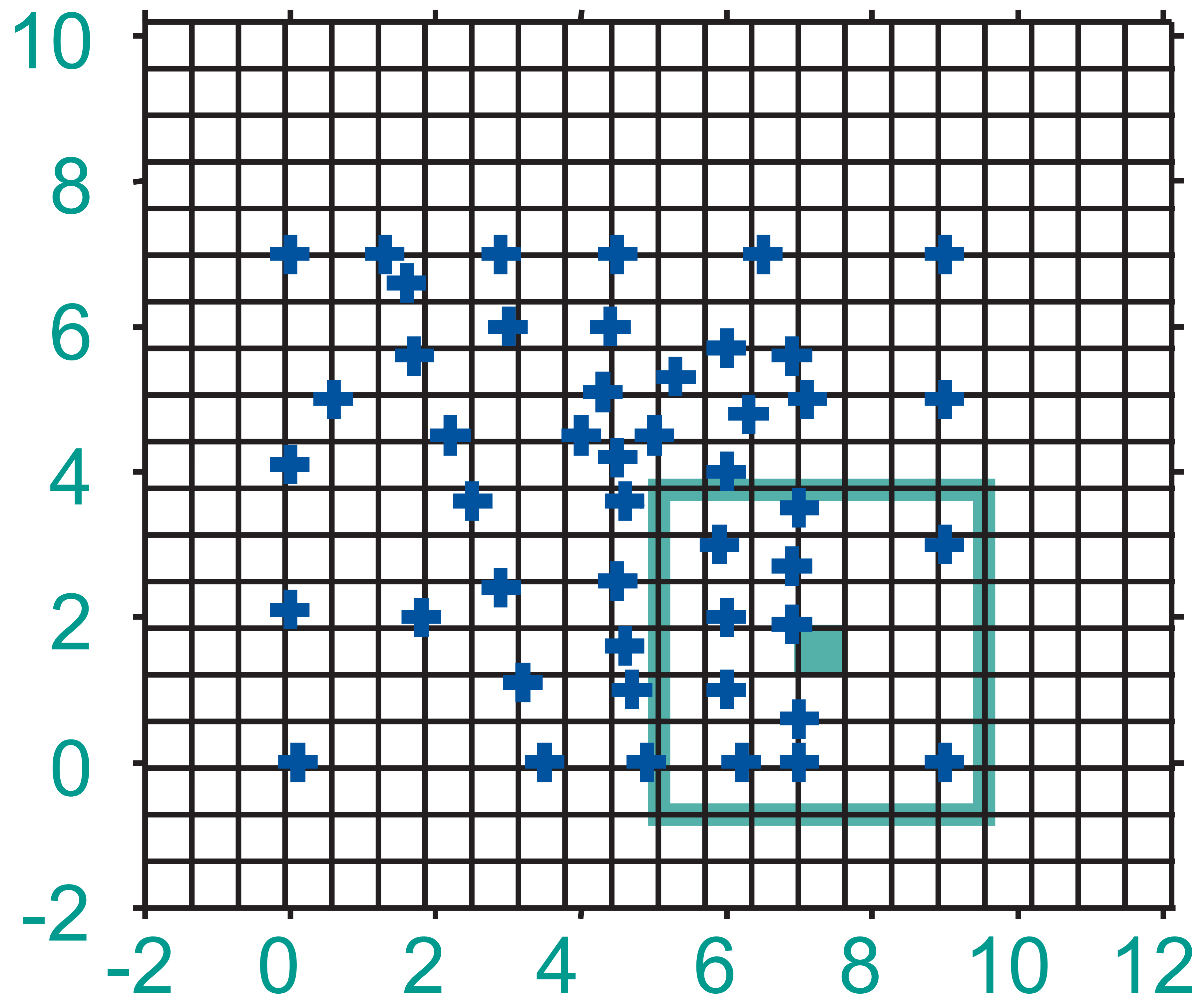


Data preparation: interpolation from points

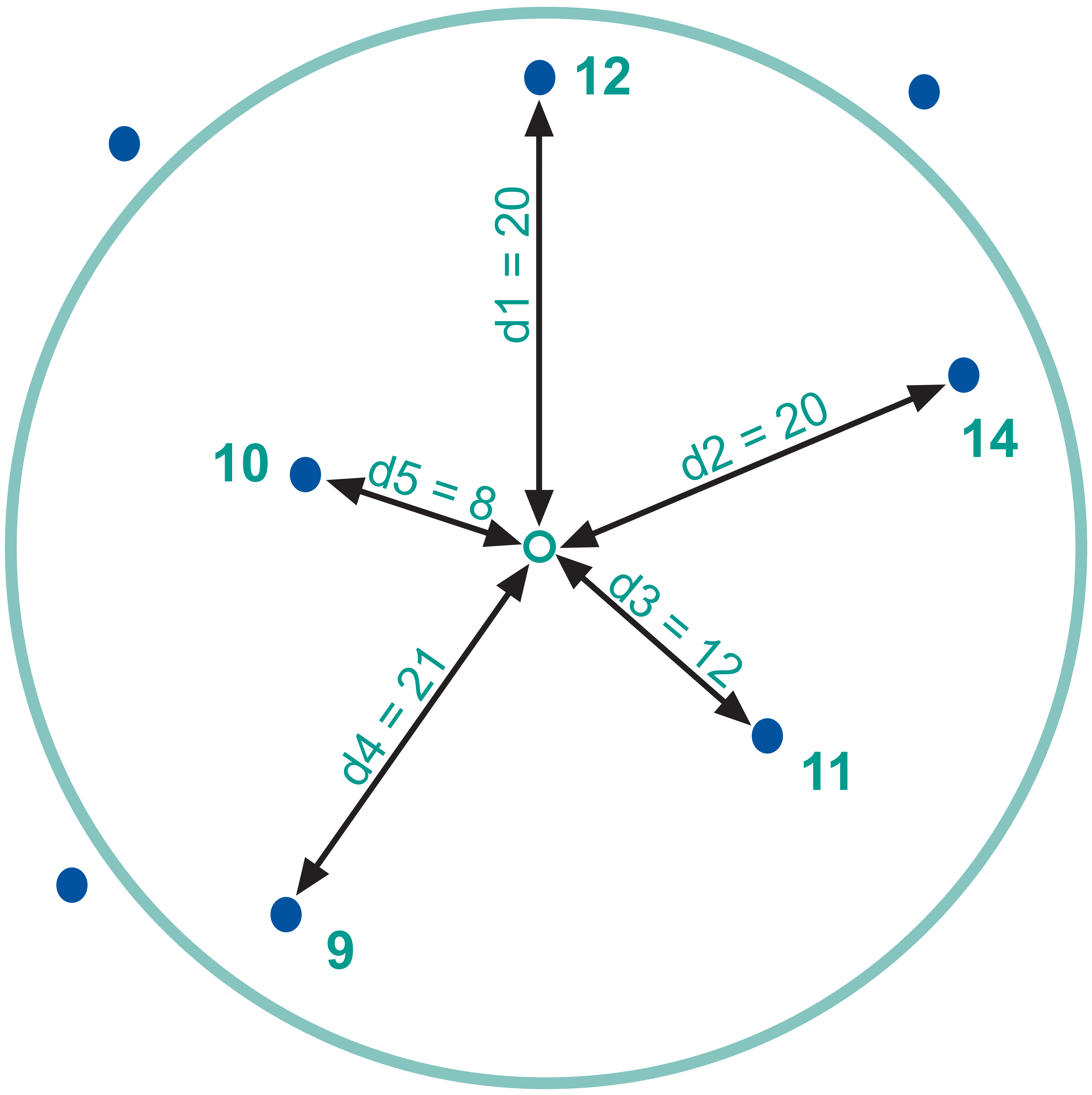


Data preparation: interpolation from points





Data preparation: interpolation from points



- Data preparation
- Spatial analysis
- Practical session

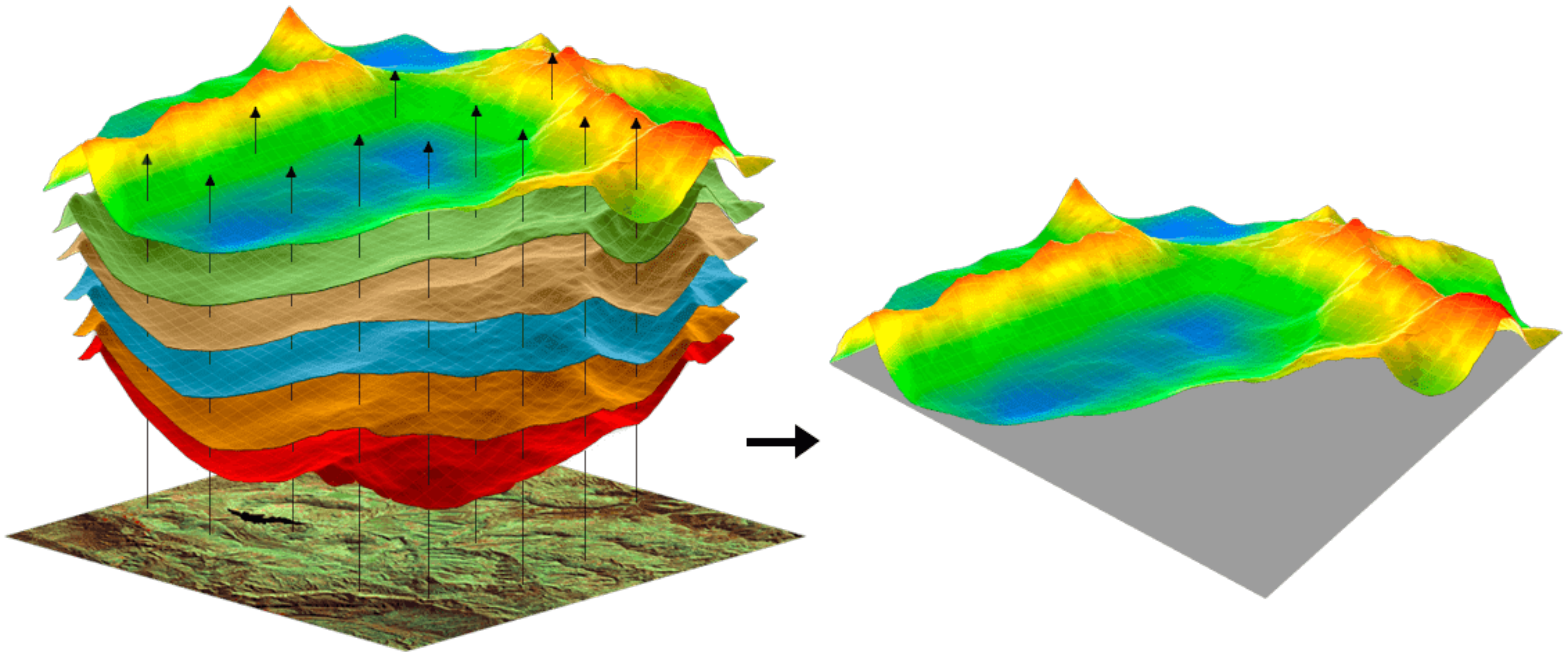


Image: EOS Data Analytics

Spatial analysis

Using one layer:

- classification
- retrieval
- generalisation
- measurement

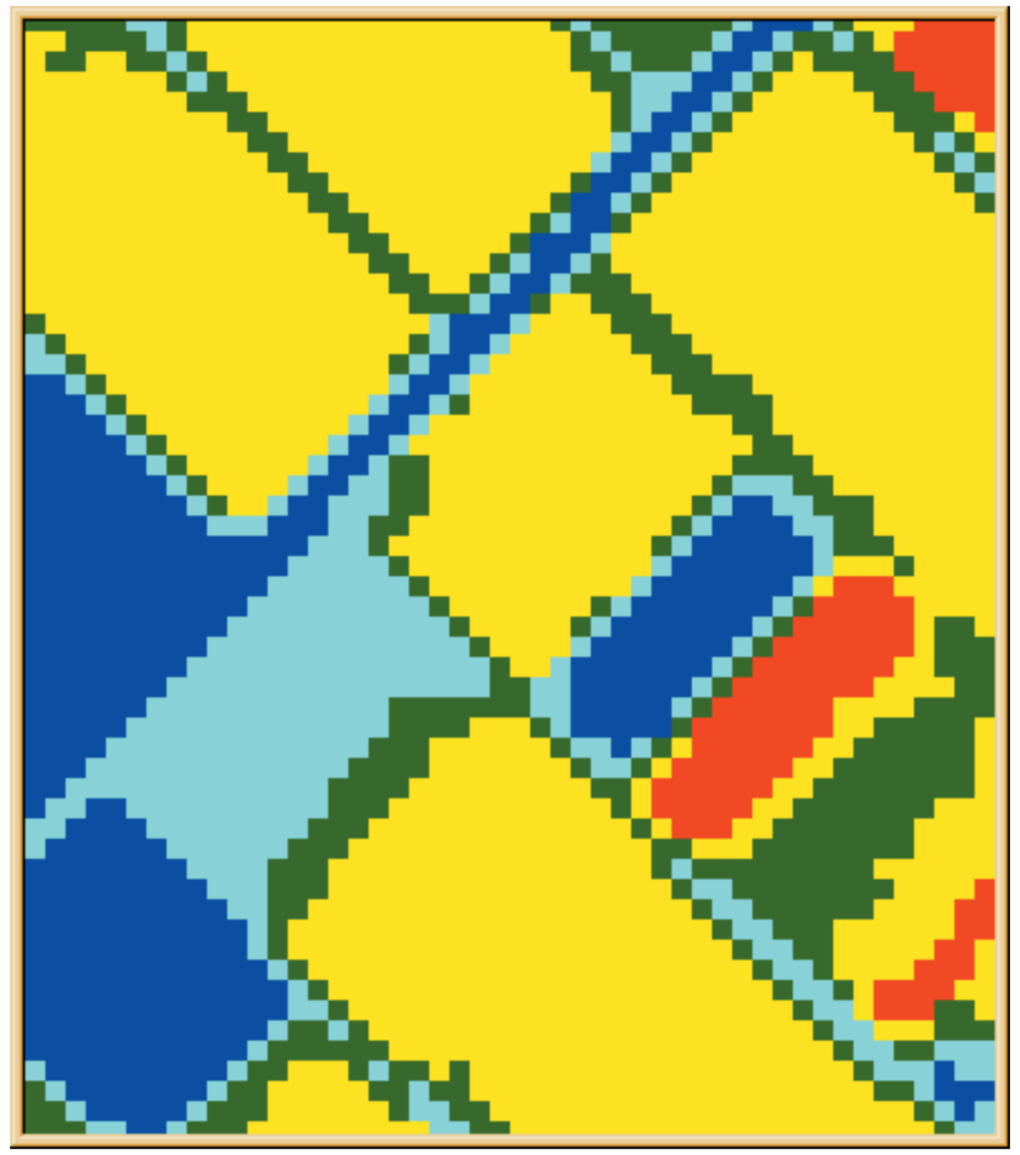
Using multiple layers:

- overlays
- neighbourhood
- connectivity

Spatial analysis: classification

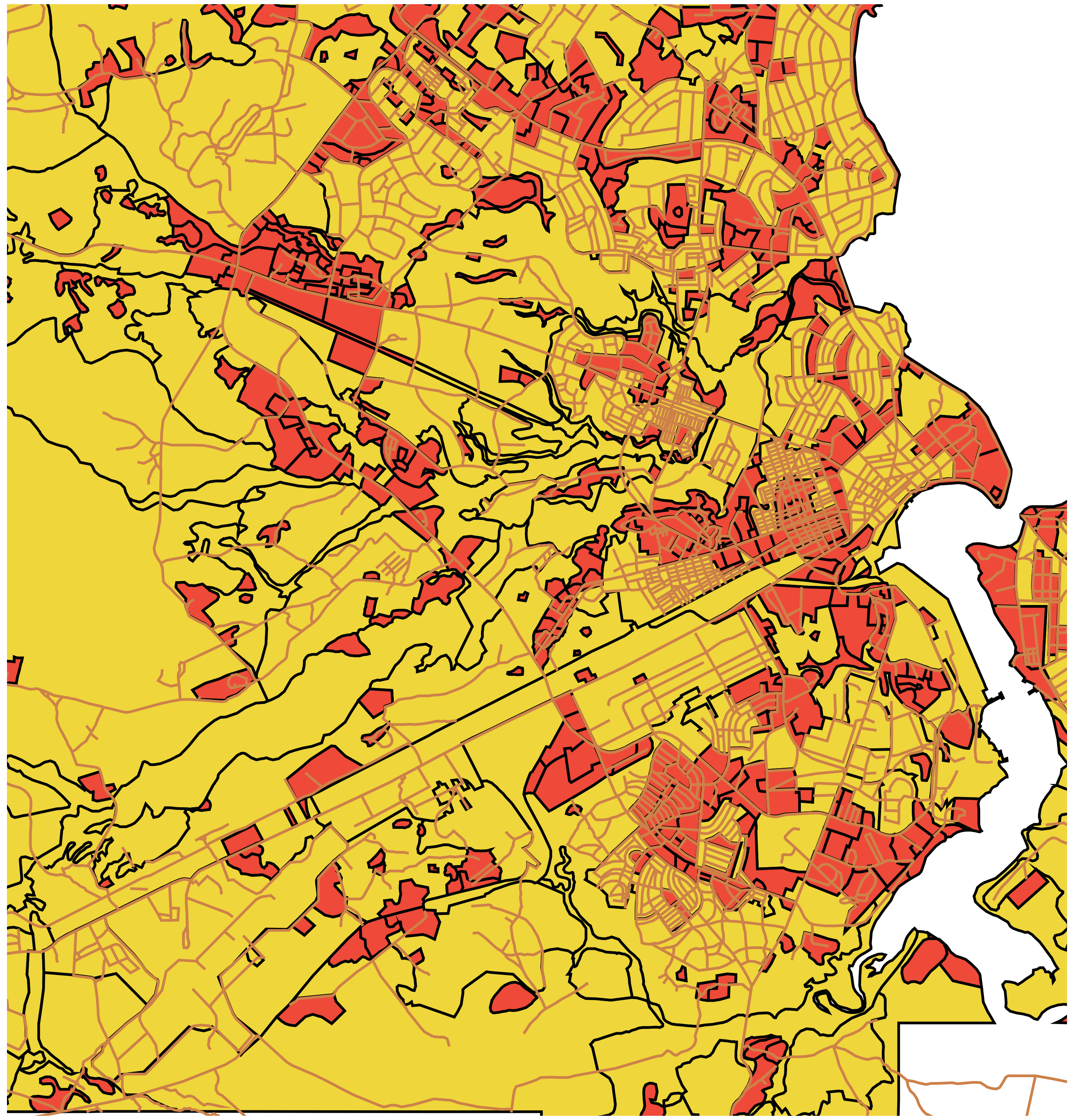
- **Classification** allows the assignment of features to a class on the basis of attribute values or attribute ranges (definition of data patterns).

- Example: on the basis of reflectance characteristics found in a raster, pixels may be classified as representing different crops, such as potato and maize.



Spatial analysis: retrieval

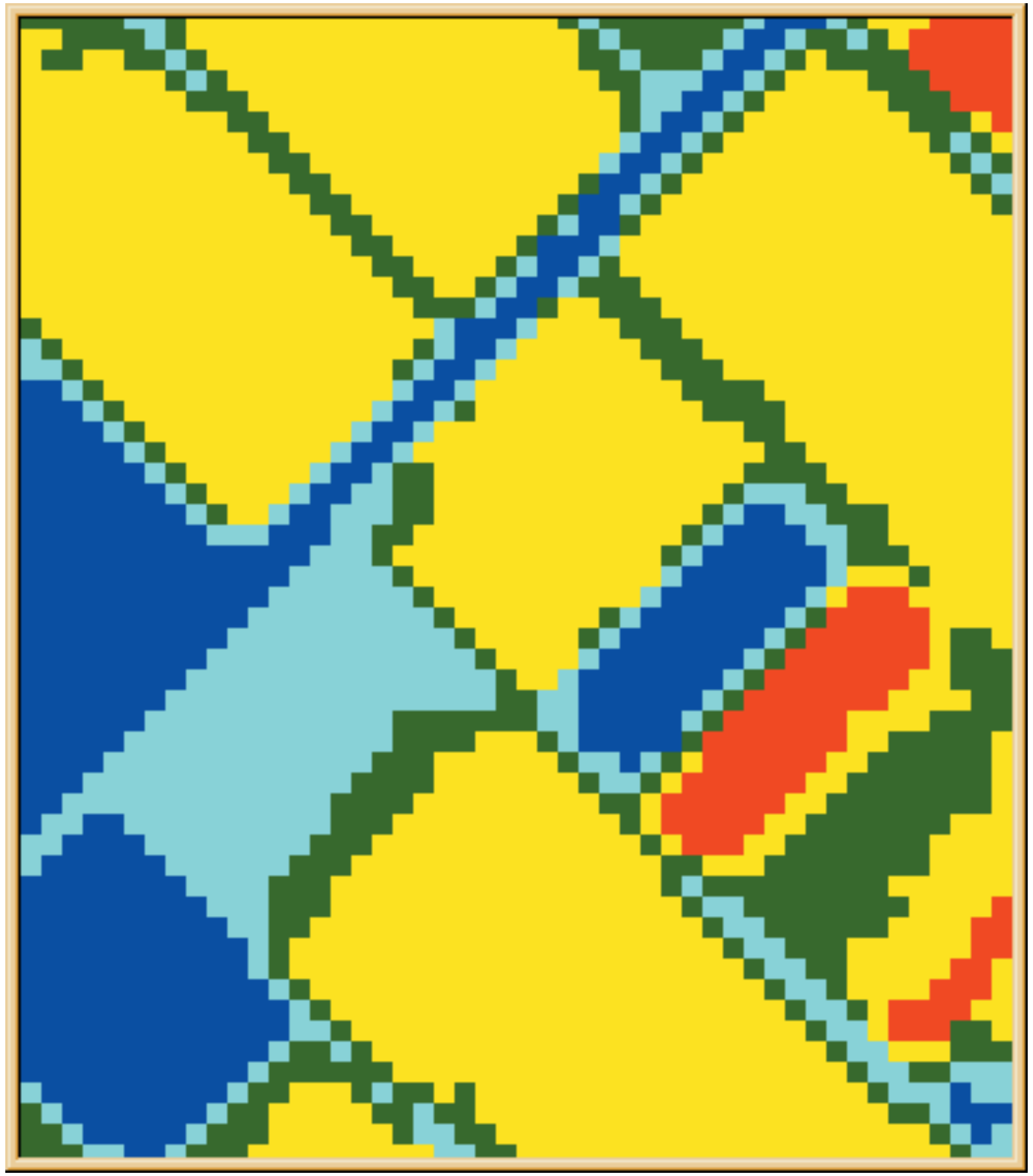
- **Retrieval** functions allow the selective search of data
- Example: select all polygons where Area < 400 000.



Area	IDs	LandUse
174308.70	2	30
2066475.00	3	70
214582.50	4	80
29313.86	5	80
73328.08	6	80
53303.30	7	80
614530.10	8	20
1637161.00	9	80
156357.40	10	70
59202.20	11	20
83289.59	12	80
225642.20	13	20
28377.33	14	40
228930.30	15	30
986242.30	16	70

Spatial analysis: generalisation

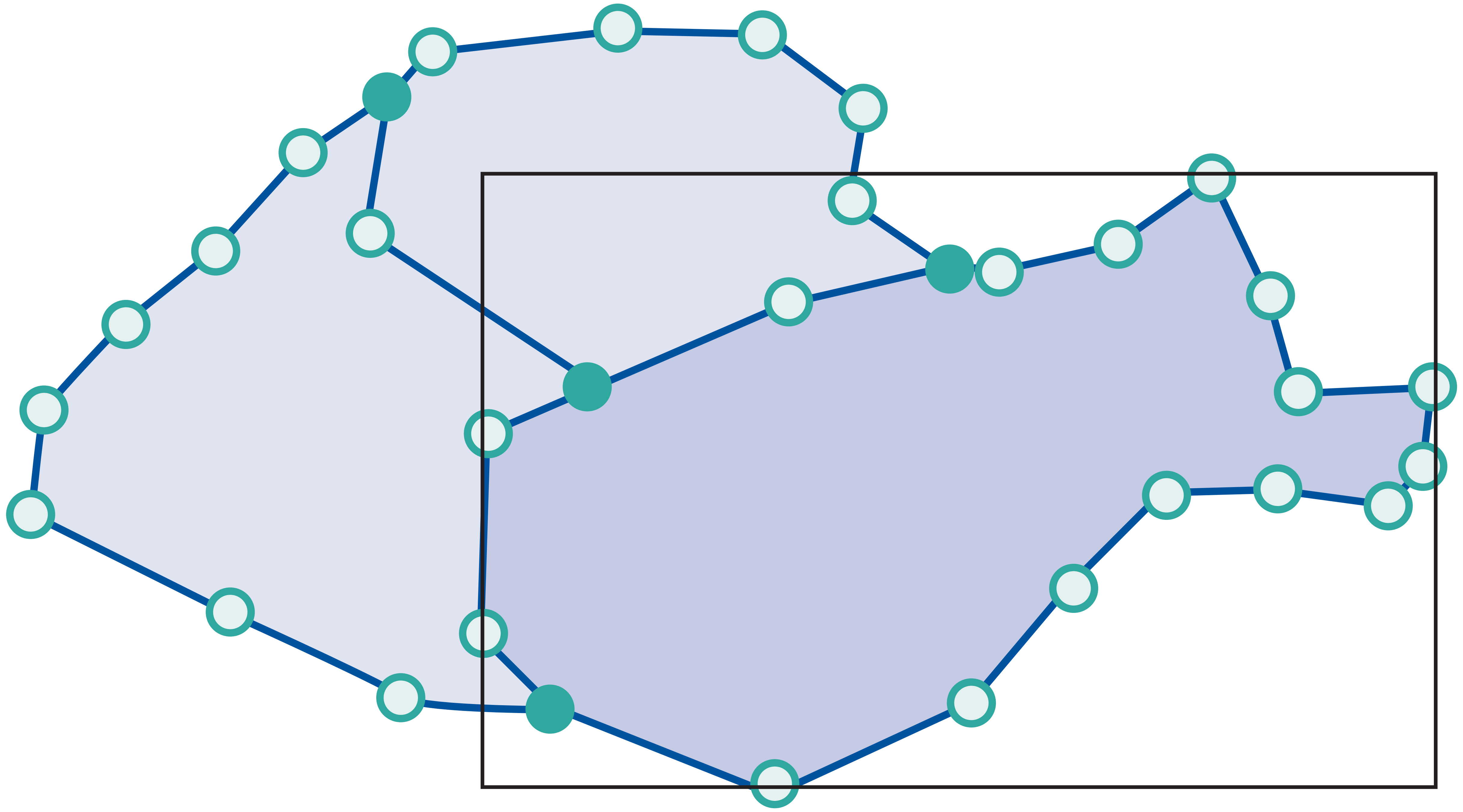
- **Generalisation** is a function that joins different classes of objects with common characteristics to a higher level (generalised) class
- Example: generalise fields where potato or maize are grown as ‘food production fields’.



Spatial analysis: measurement

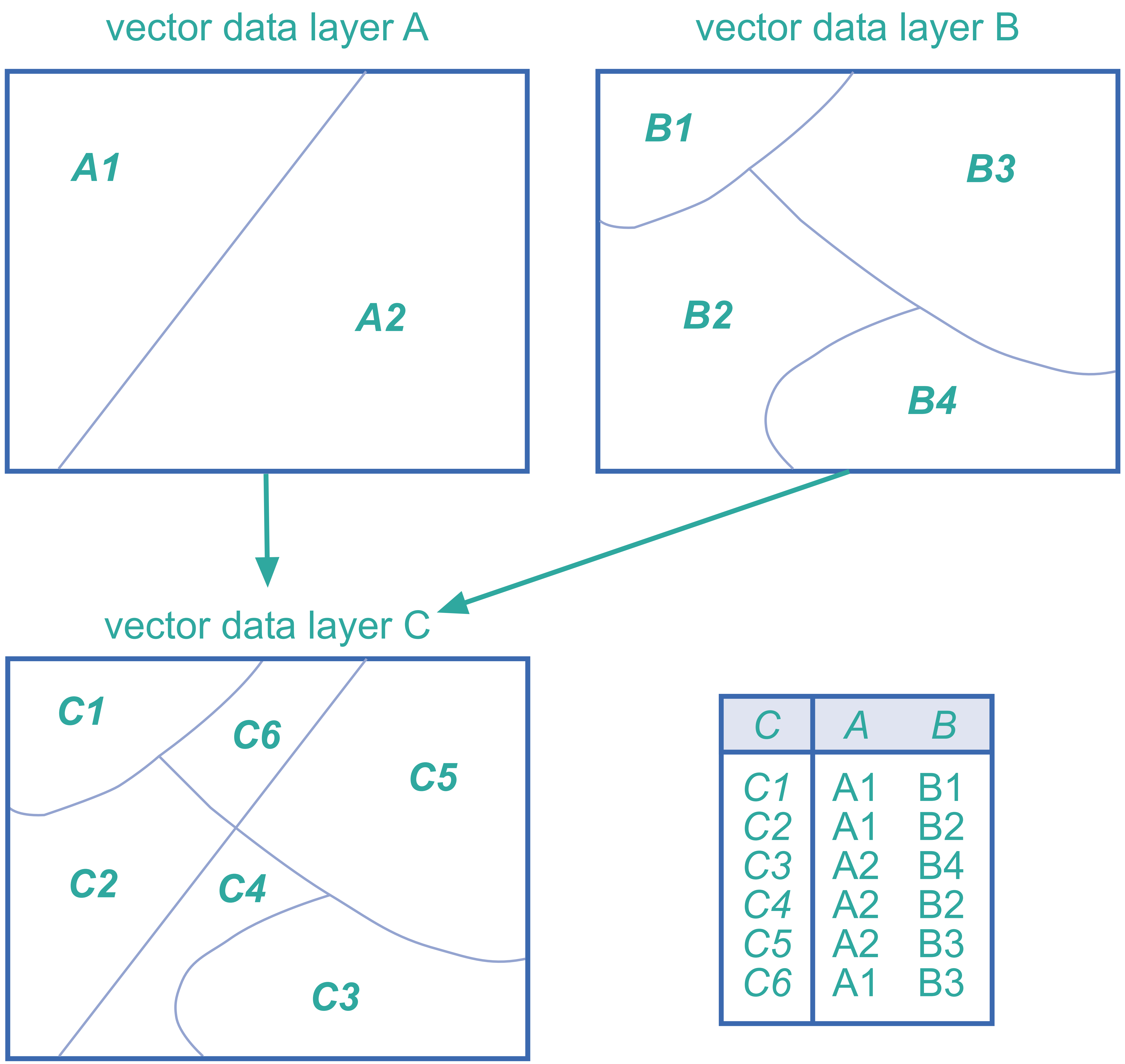
- **Measurement** functions allow the calculation of distances, lengths, or areas.

- Example: obtain the minimal bounding box of a polygon.



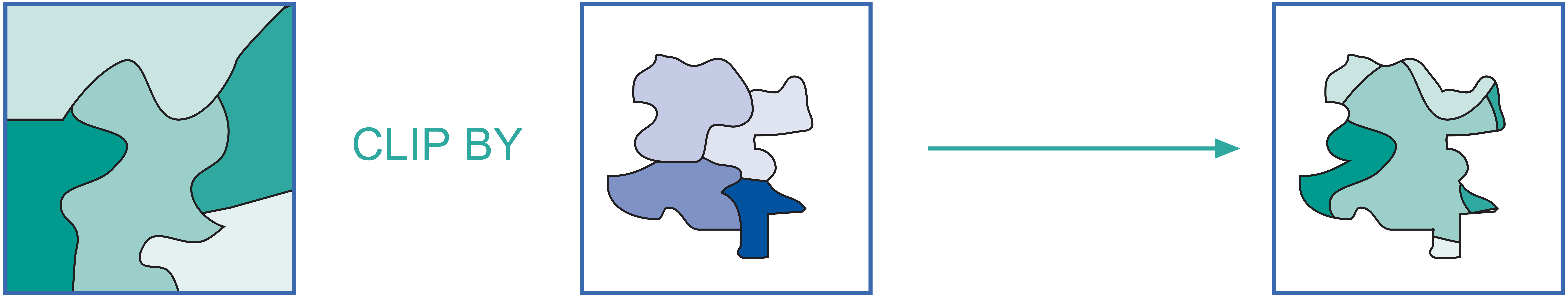
Spatial analysis: overlay

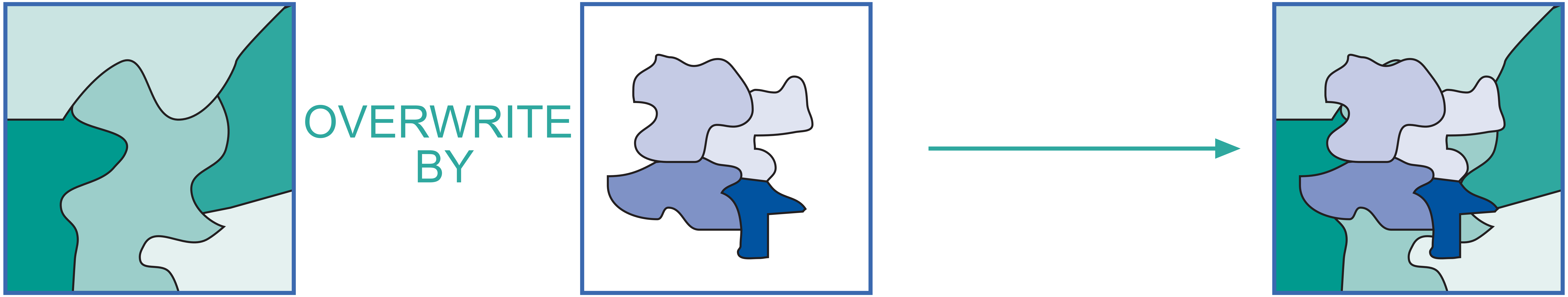
- **Overlays** allow the combination of two (or more) spatial data layers comparing them position by position, and treating areas of overlap—and of non-overlap—in distinct ways



C	A	B
C1	A1	B1
C2	A1	B2
C3	A2	B4
C4	A2	B2
C5	A2	B3
C6	A1	B3

Spatial analysis: overlay (operations)





Spatial analysis: overlay (map algebra)

A

5	5	2	2
5	5	5	2
6	2	2	2
6	6	6	6

B

4	4	8	8
4	4	4	8
1	1	1	8
1	1	8	8

C1 := A +10

C1

15	15	12	12
15	15	15	12
16	12	12	12
16	16	16	16

C2 := A + B

C2

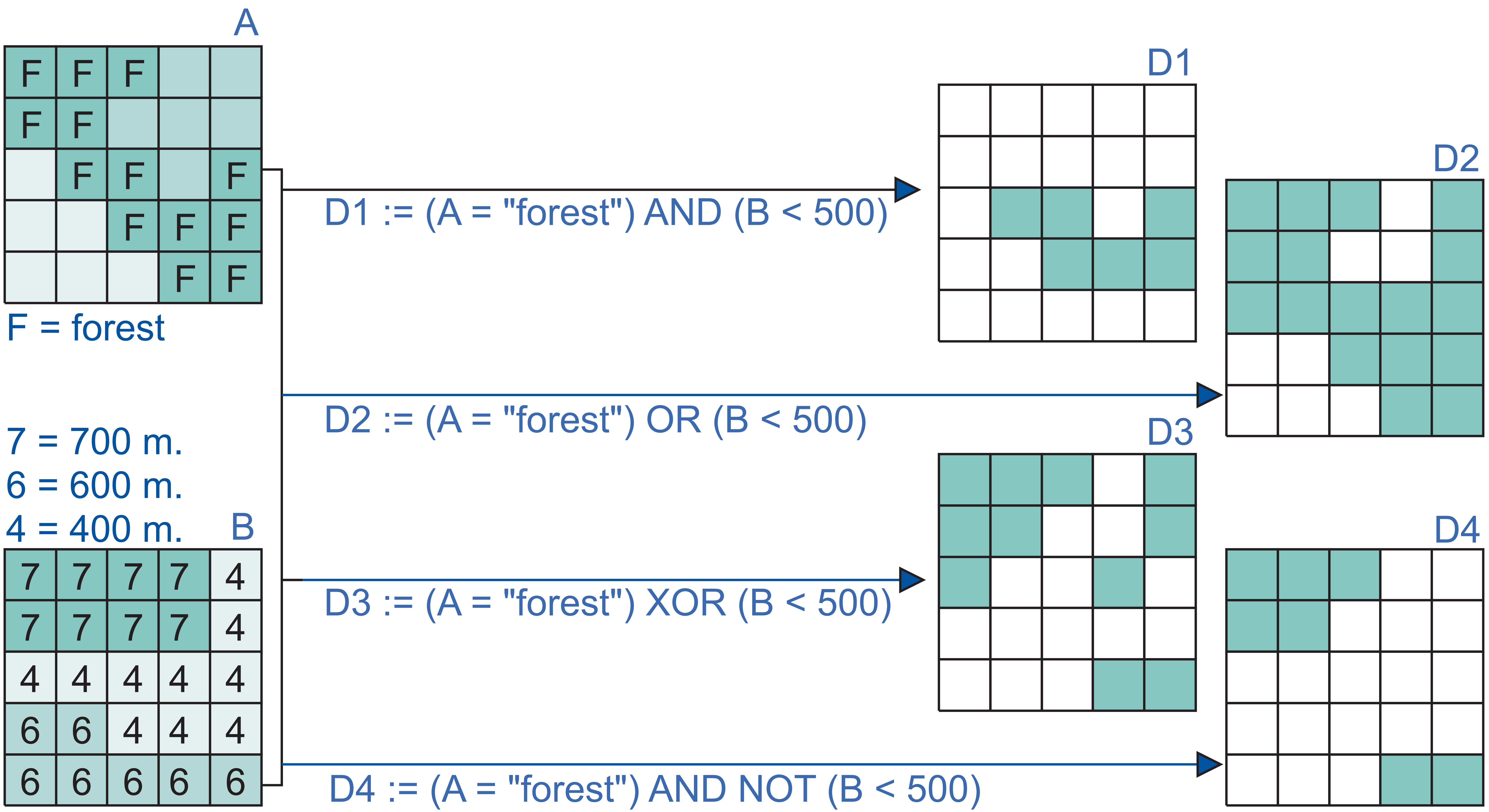
9	9	10	10
9	9	9	10
7	3	3	10
7	7	14	14

C3 := ((A - B)/(A + B))*100

C3

11	11	-60	-60
11	11	11	-60
71	33	33	-60
71	71	-14	-14

Spatial analysis: overlay (map algebra)



Spatial analysis: overlay (map algebra)

A

F	F	F		
F	F			
	F	F		F
		F	F	F
			F	F

F = forest

B

7	7	7	7	4
7	7	7	7	4
4	4	4	4	4
6	6	4	4	4
6	6	6	6	6

C1 := CON (A = "F", B, 0)

C1

7	7	7	0	0
7	7	0	0	0
0	4	4	0	4
0	0	4	4	4
0	0	0	6	6

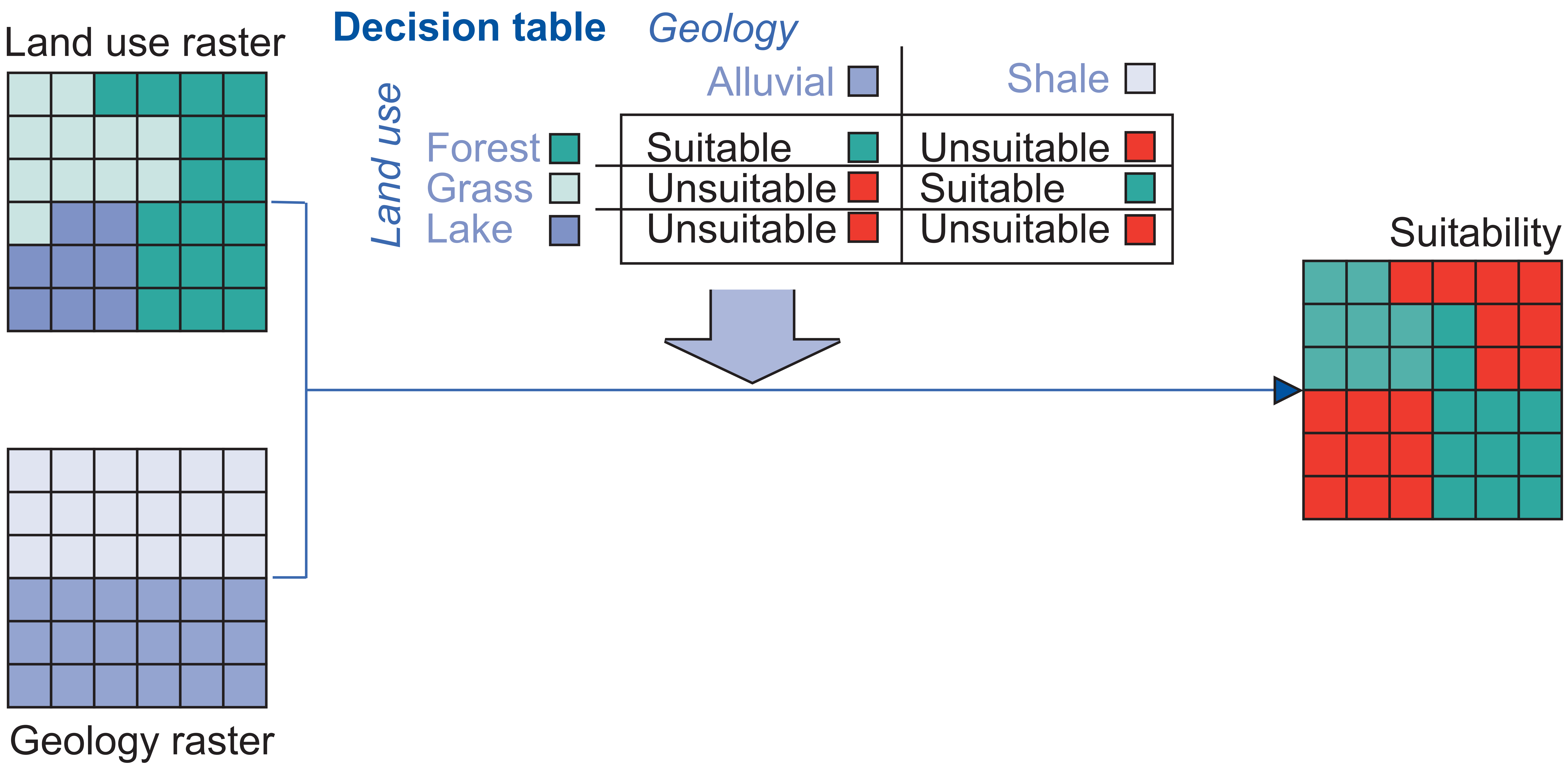
C2 := CON ((A = "F") AND (B = 7), 10, 0)

C2

10	10	10	0	0
10	10	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

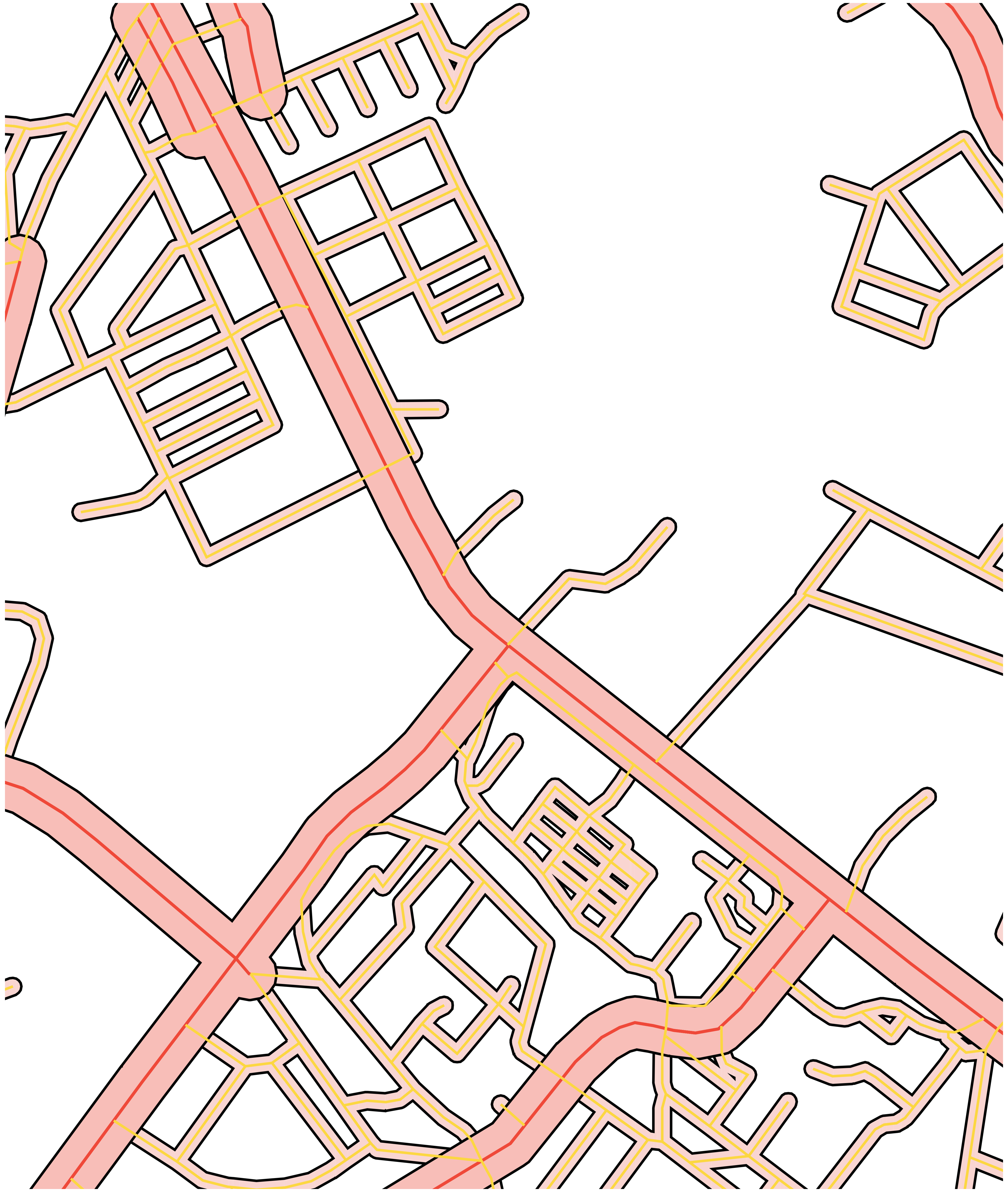
7 = 700 m.
6 = 600 m.
4 = 400 m.

Spatial analysis: overlay (map algebra)



Spatial analysis: neighbourhood

- **Neighbourhood** functions evaluate the characteristics of an area surrounding a feature's location.
- Example: Buffer zone generation using 25 metres for minor roads and 75 metres for main roads.



Spatial analysis: neighbourhood

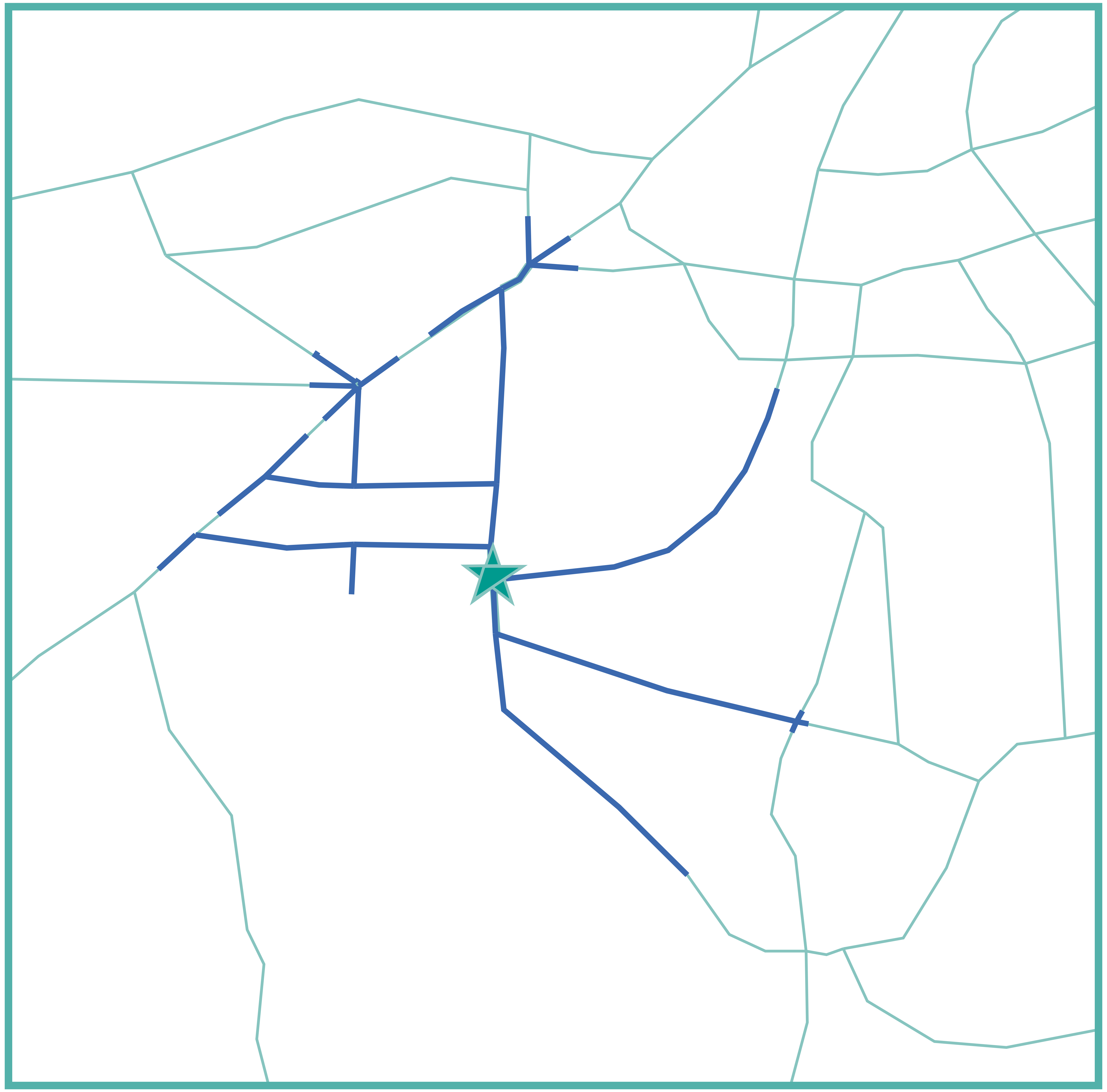
156	144	138	142	116	98
148	134	112	98	92	100
138	106	88	74	76	96
128	116	110	44	62	48
136	122	94	42	32	38
148	106	68	24	22	24

↘	↘	↘	↓	↓	↙
↘	↘	↘	↓	↓	↙
→	→	↘	↓	↙	↓
↗	↗	→	↘	↓	↙
↘	↘	→	↓	↓	↓
→	→	→	→	↓	←

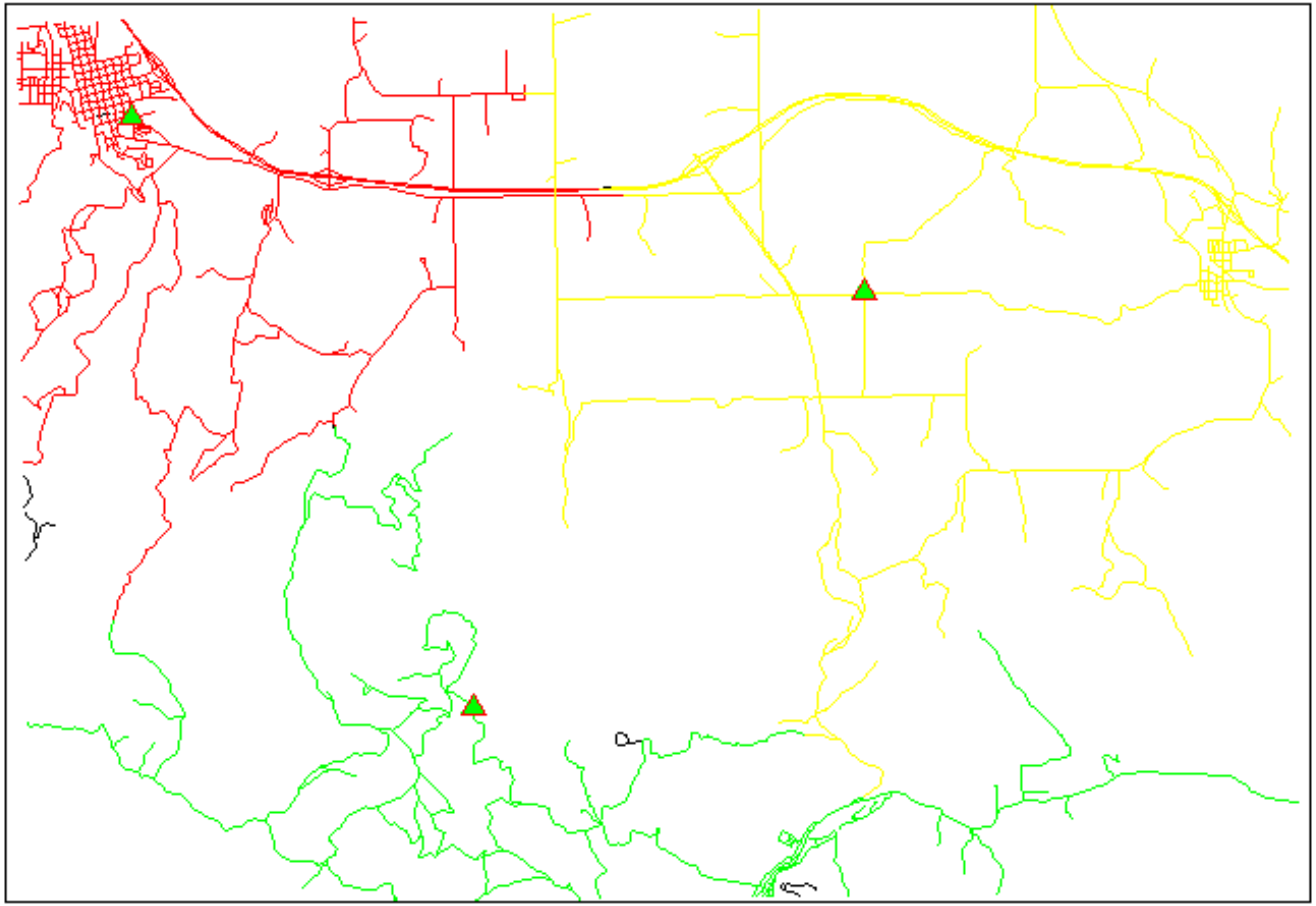
0	0	0	0	0	0
0	1	1	2	2	0
0	3	7	5	4	0
0	0	0	20	0	1
0	0	0	1	24	0
0	2	4	7	35	1

Spatial analysis: connectivity

- **Connectivity** functions work on the basis of networks, including road networks, water courses in coastal zones, and communication lines in mobile telephony.
- These networks represent spatial linkages between features.
- Example: Network allocation on a pupil/school assignment problem, where the street segments within 2 km of the school are identified.



Spatial analysis: connectivity



Questions?

Processing 2D and 3D geodata contents

- Data preparation
- Spatial analysis
- Practical session

Practical session 2

- Create a map of a region of your interest by applying spatial analysis methods.
- Some ideas:
  - Compute the population density based on an absolute population dataset.
  - Compute another density measure, e.g. the density of roads or railways.
  - Check if there's a correlation between two datasets, e.g. traffic and population, or conservation areas and land use.
- Look for a dataset from a national or regional agency. Alternatively, use a worldwide one like the Kontur population dataset.
- Use the print view to put map elements like scale, North arrow and legend.

3D city models

- Why 3D?
- 3D geometries in geoinformation (ISO19107)
- 3D city models
- Practical session

Image: Freepik

Why 3D?

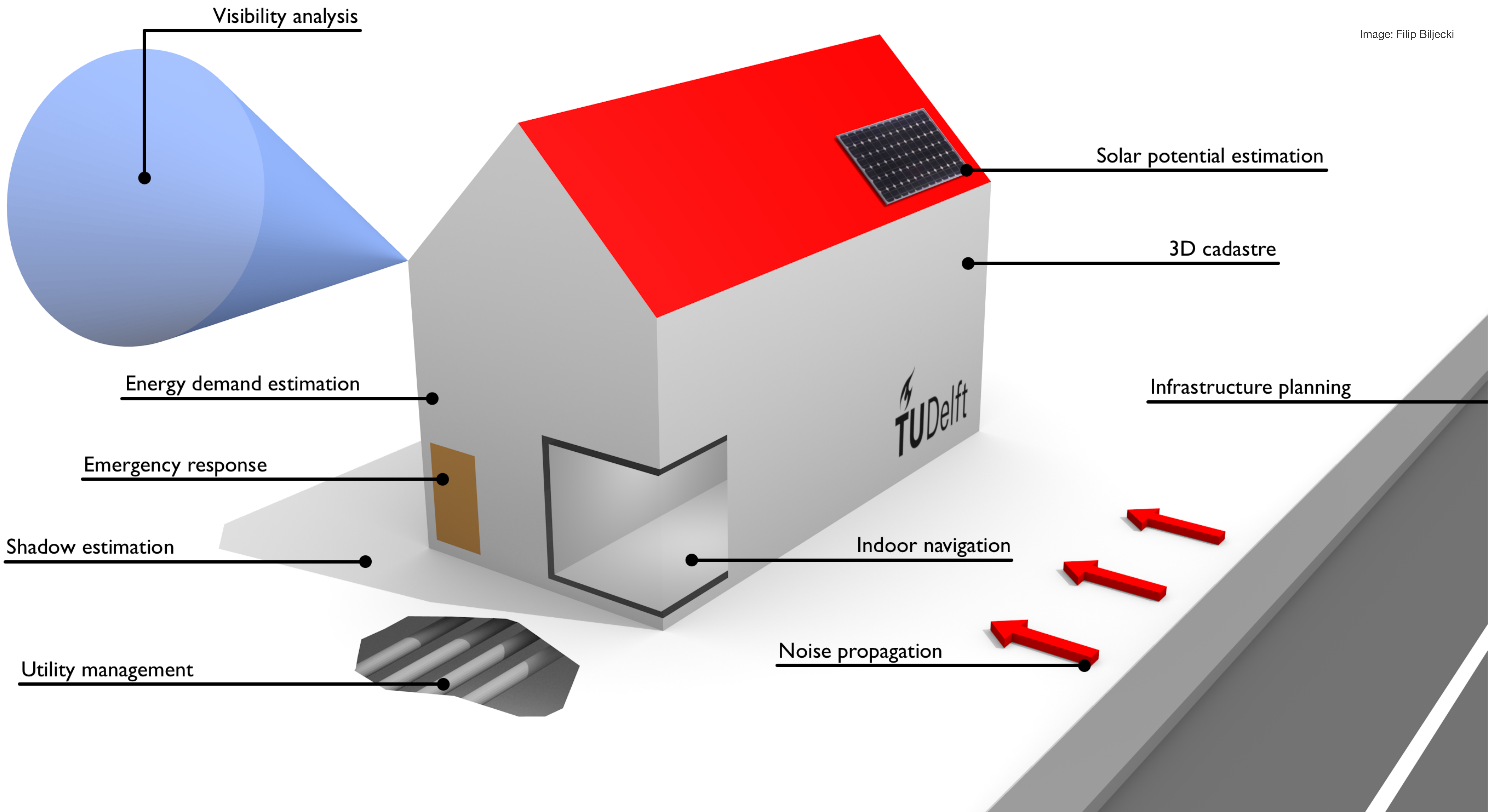


Image: Filip Biljecki

Why 3D?

Why 3D?

Why 3D?

Why 3D?

Why 3D?

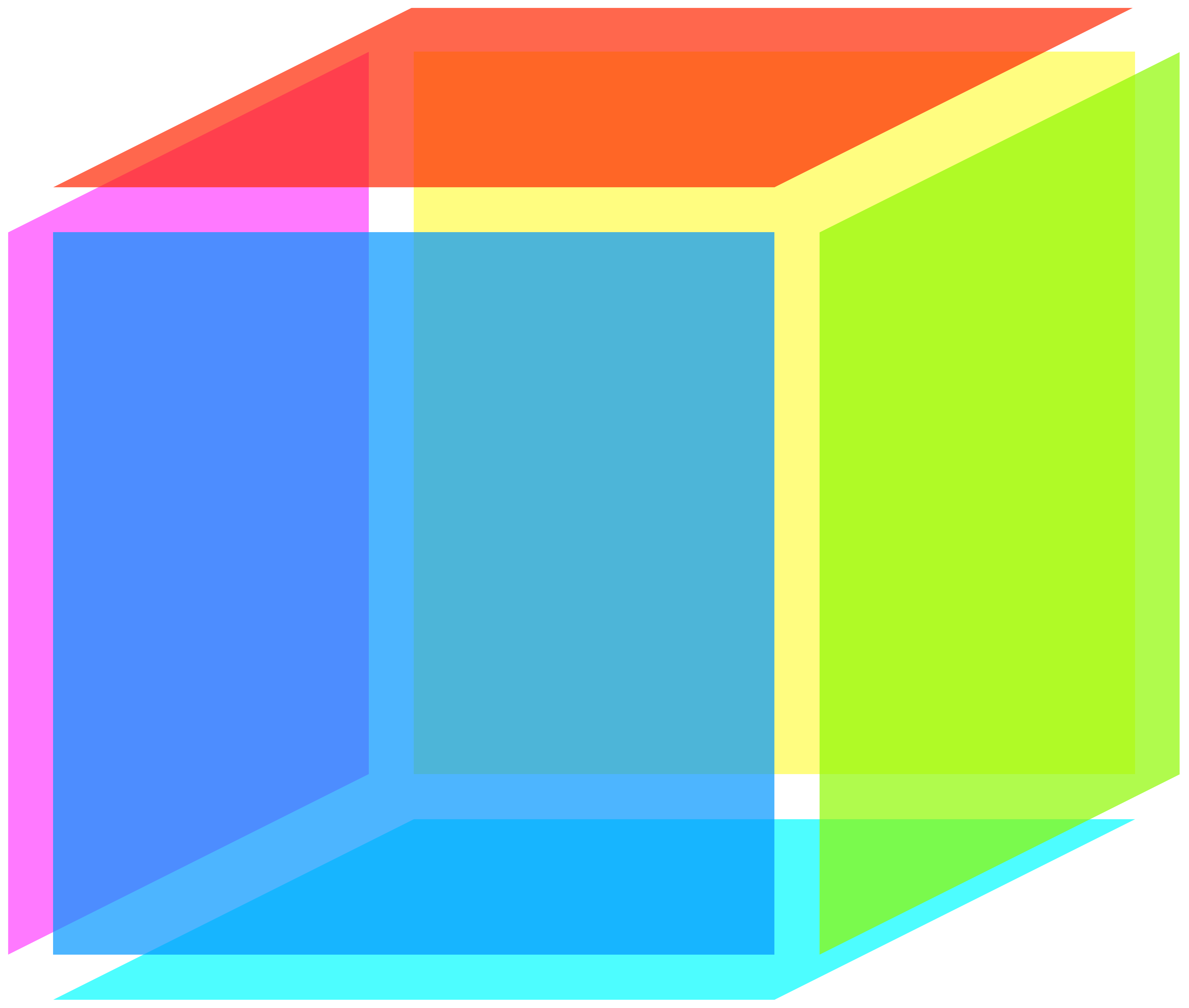
- Visualisation (e.g. for gaming, tourism, navigation, etc)
- Energy demand estimation (and potential for retrofitting)
- Computational fluid dynamics (e.g. for wind speeds, air quality, effects on buildings, etc)
- Shadow casting (e.g. for building permits, visibility analysis, improving energy demand/solar potential calculations, etc)

- Why 3D?
- 3D geometries in geoinformation (ISO19107)
- 3D city models
- Practical session

Image: Freepik

Boundary representation

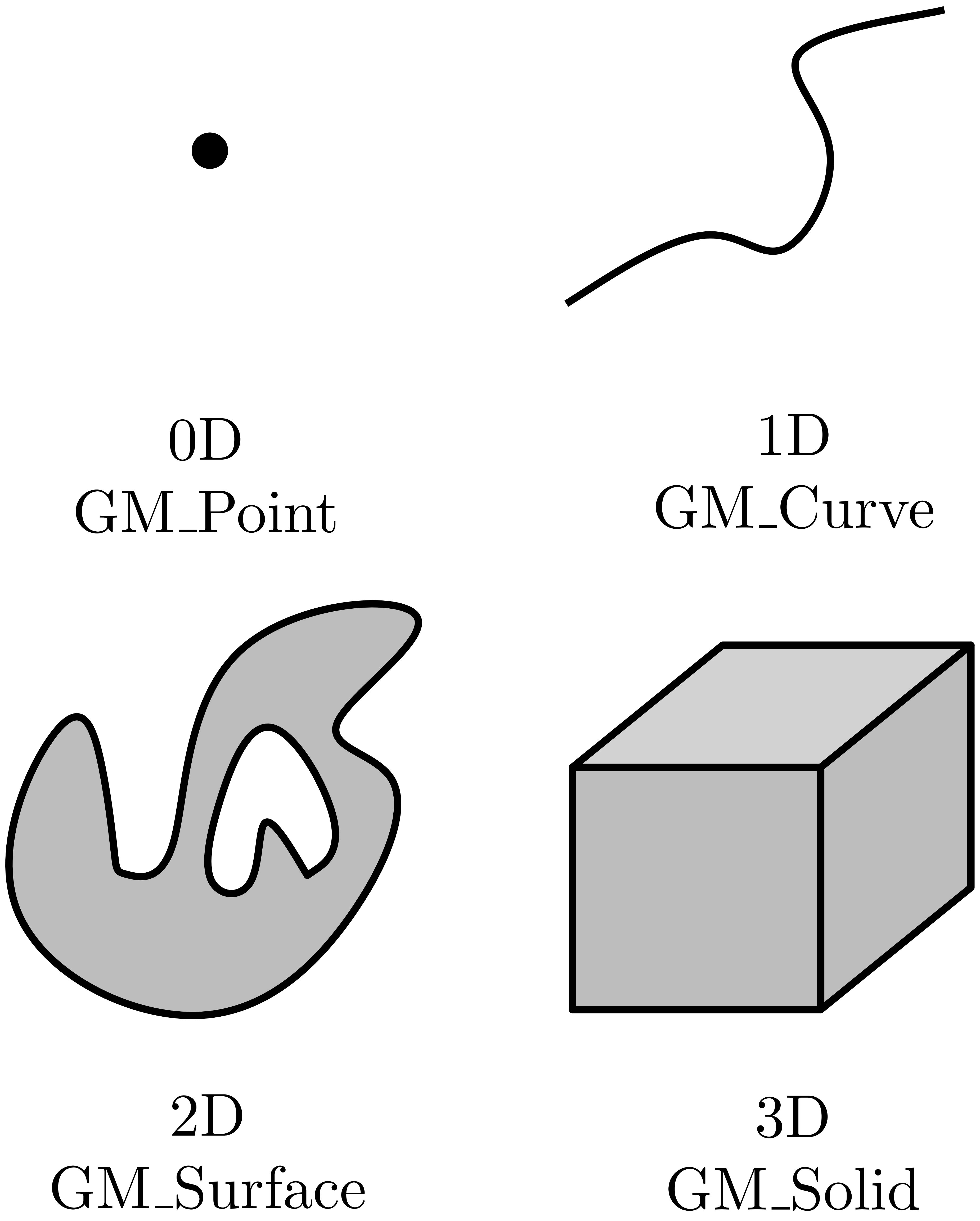
- Also known as b-rep or surface modelling
- Representing an n-dimensional object through its $(n-1)$-dimensional boundary
- Most of the time: a 3D object through its 2D boundary



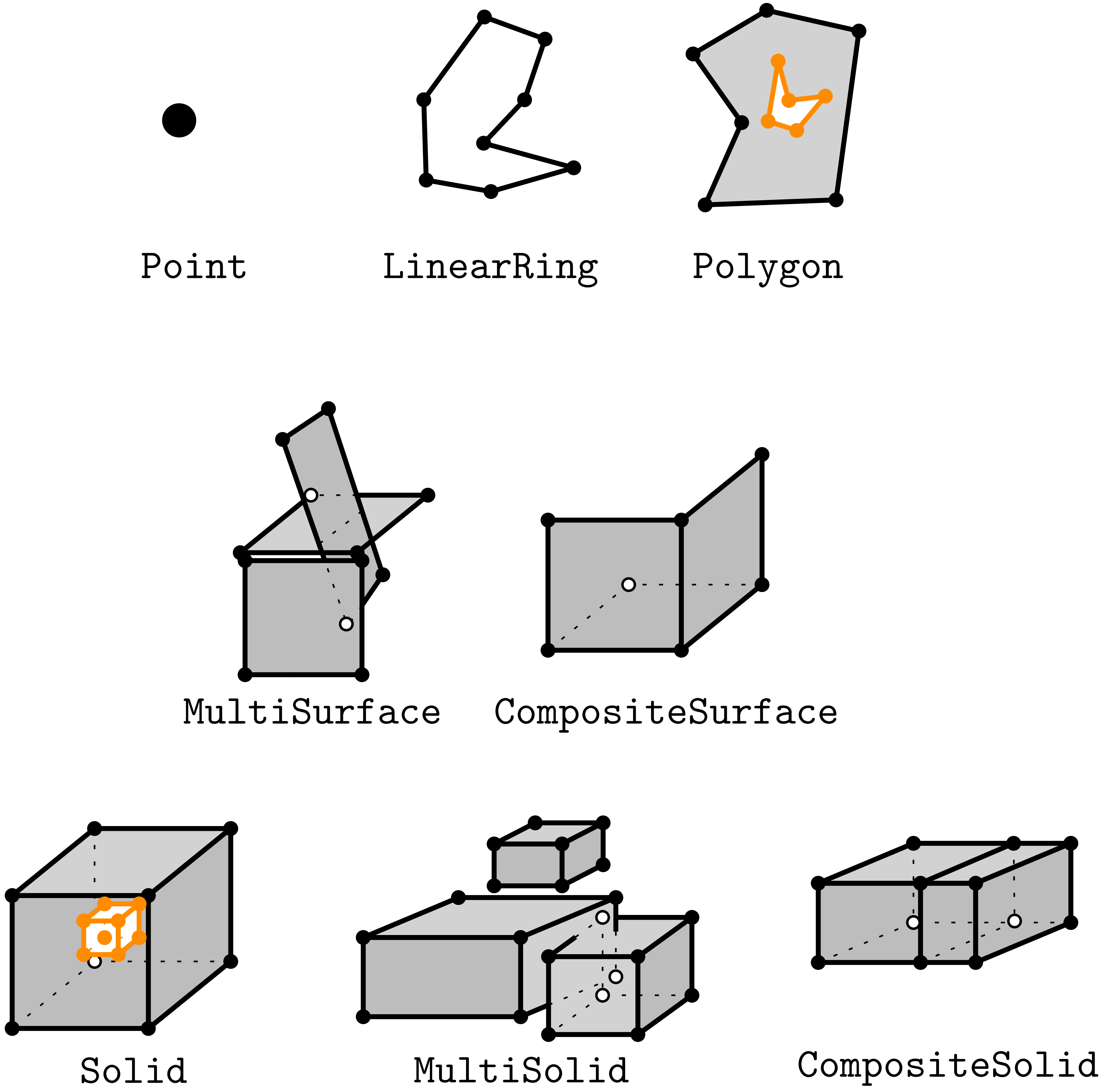
Boundary representation in 3D GIS

- Store 3D objects by storing their 2D boundary
- … which can be split into a set of surfaces (in GIS usually triangles or polygons)
- … which can be represented using a (2D) mesh, i.e. a repetitive arrangement of simpler elements
- note: not the same as a 3D mesh (e.g. TEN)

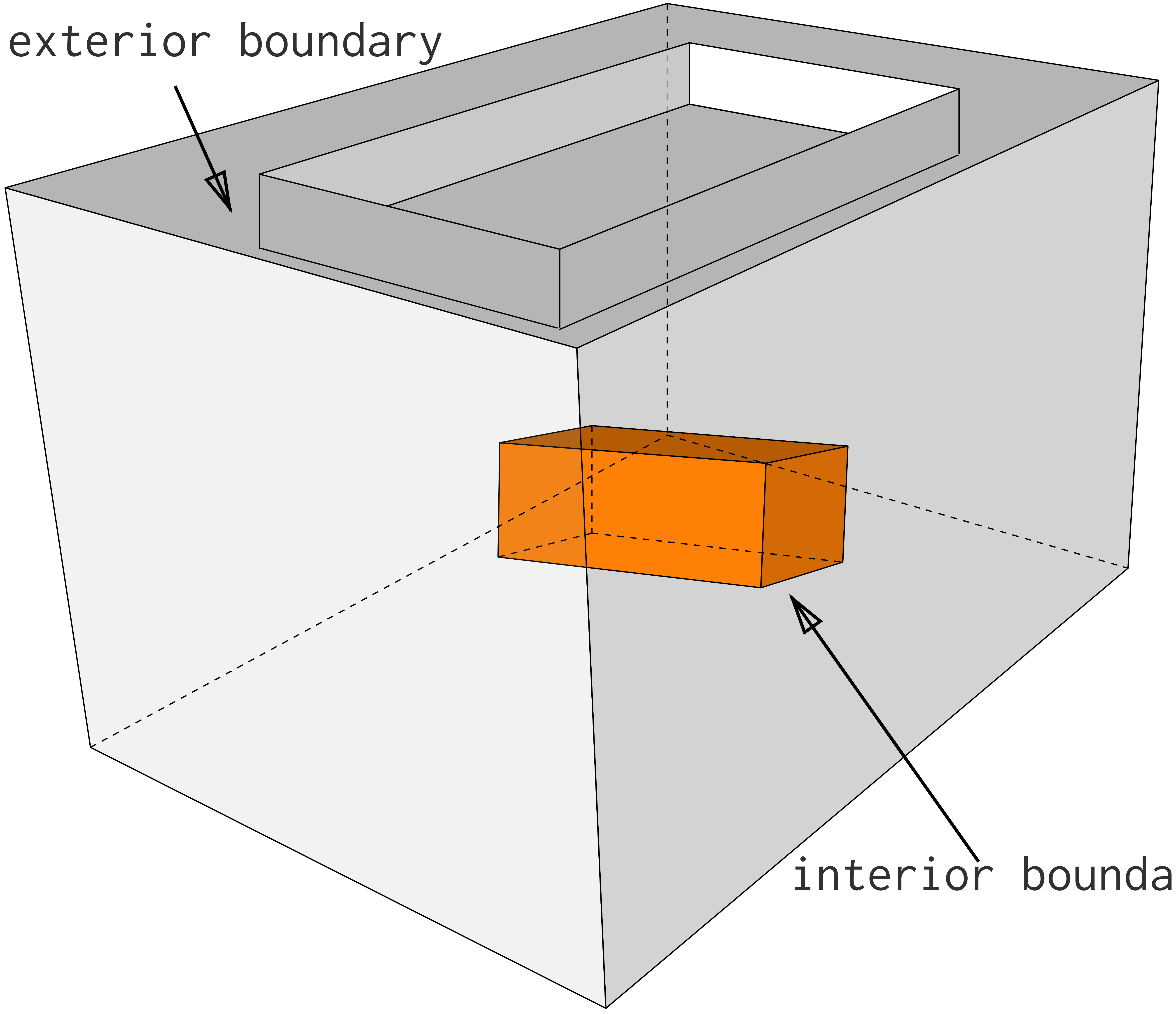
3D geometries in geoinformation

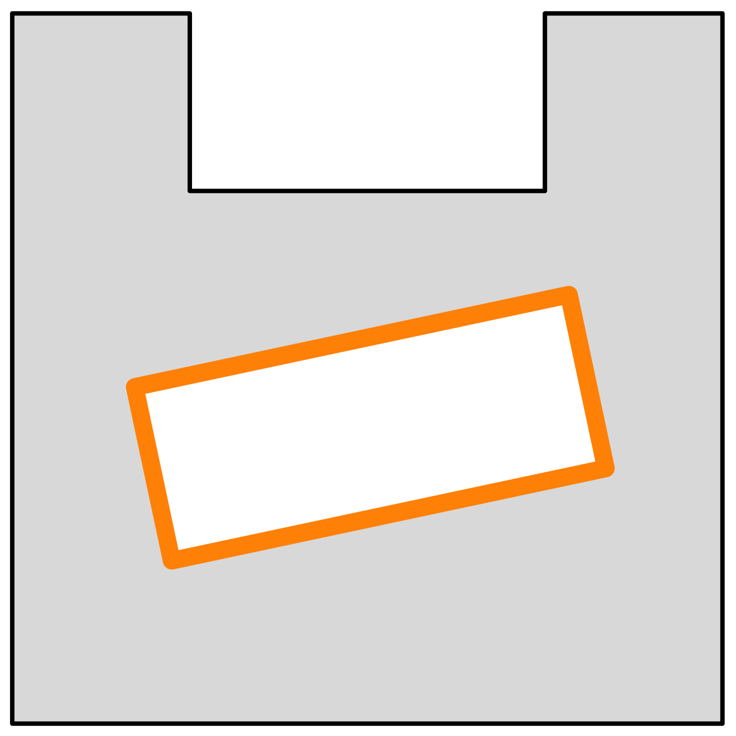


3D geometries in geoinformation

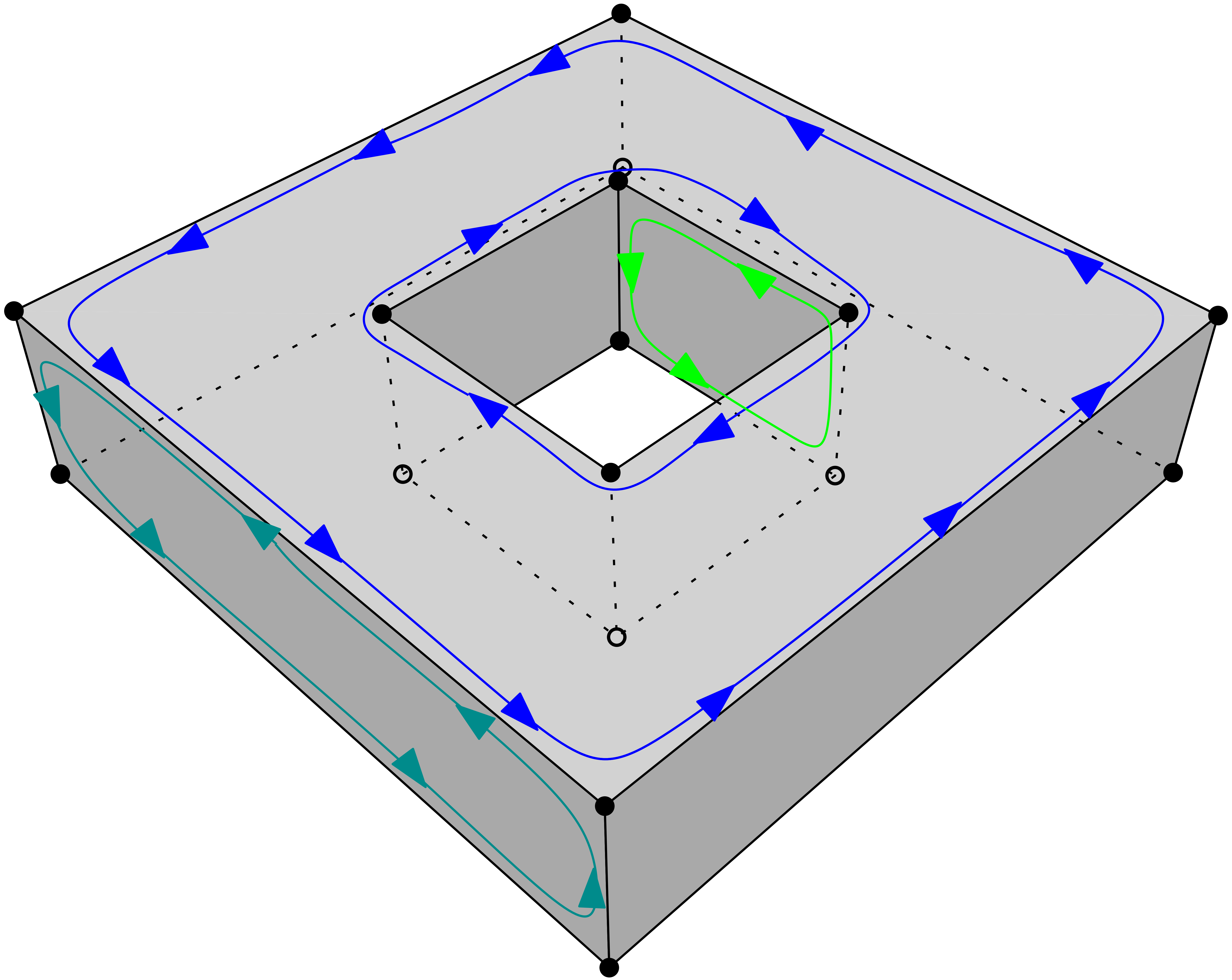


3D geometries in geoinformation





3D geometries in geoinformation



3D city models: contents

- Why 3D?
- 3D geometries in geoinformation (ISO19107)
- **3D city models**
- Practical session

CHEK DIGITAL BUILDING PERMIT

3D mesh

3D mesh

Difficult to answer:

- How many windows does the main façade of a given building have?
- How many floors does a given building have?
- Can the local park be seen from the second floor of a given building?

3D city models

3D city models

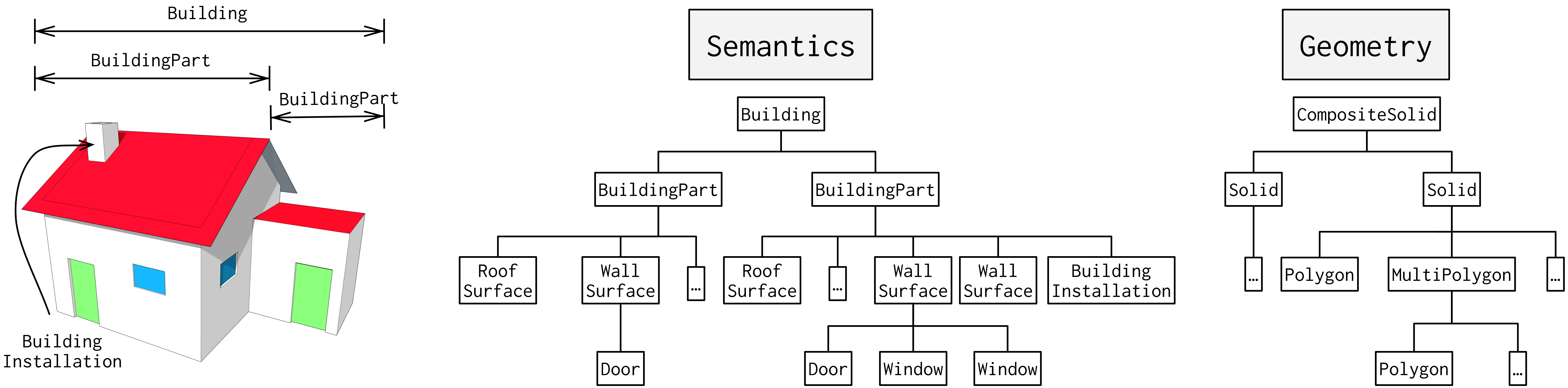
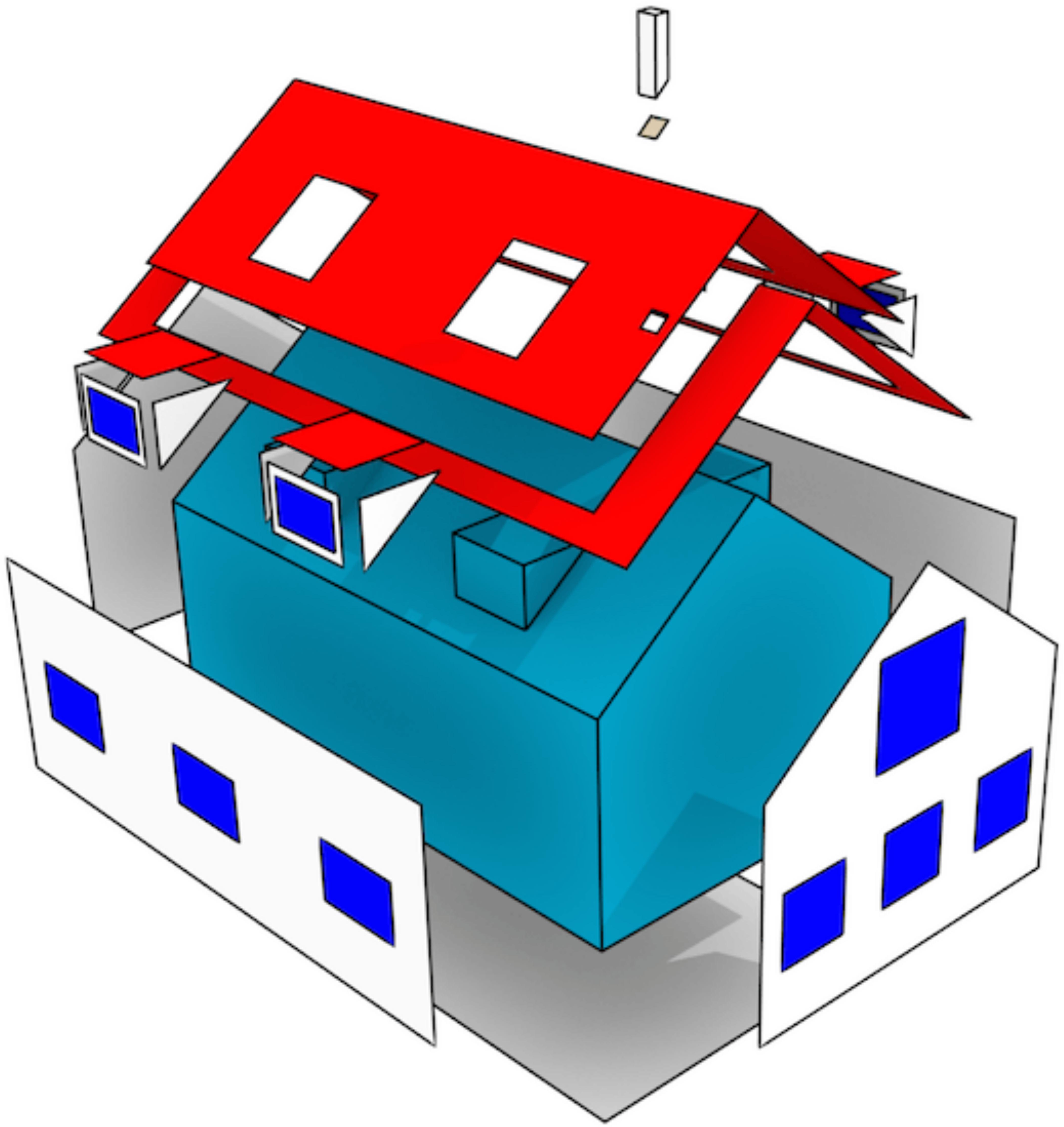
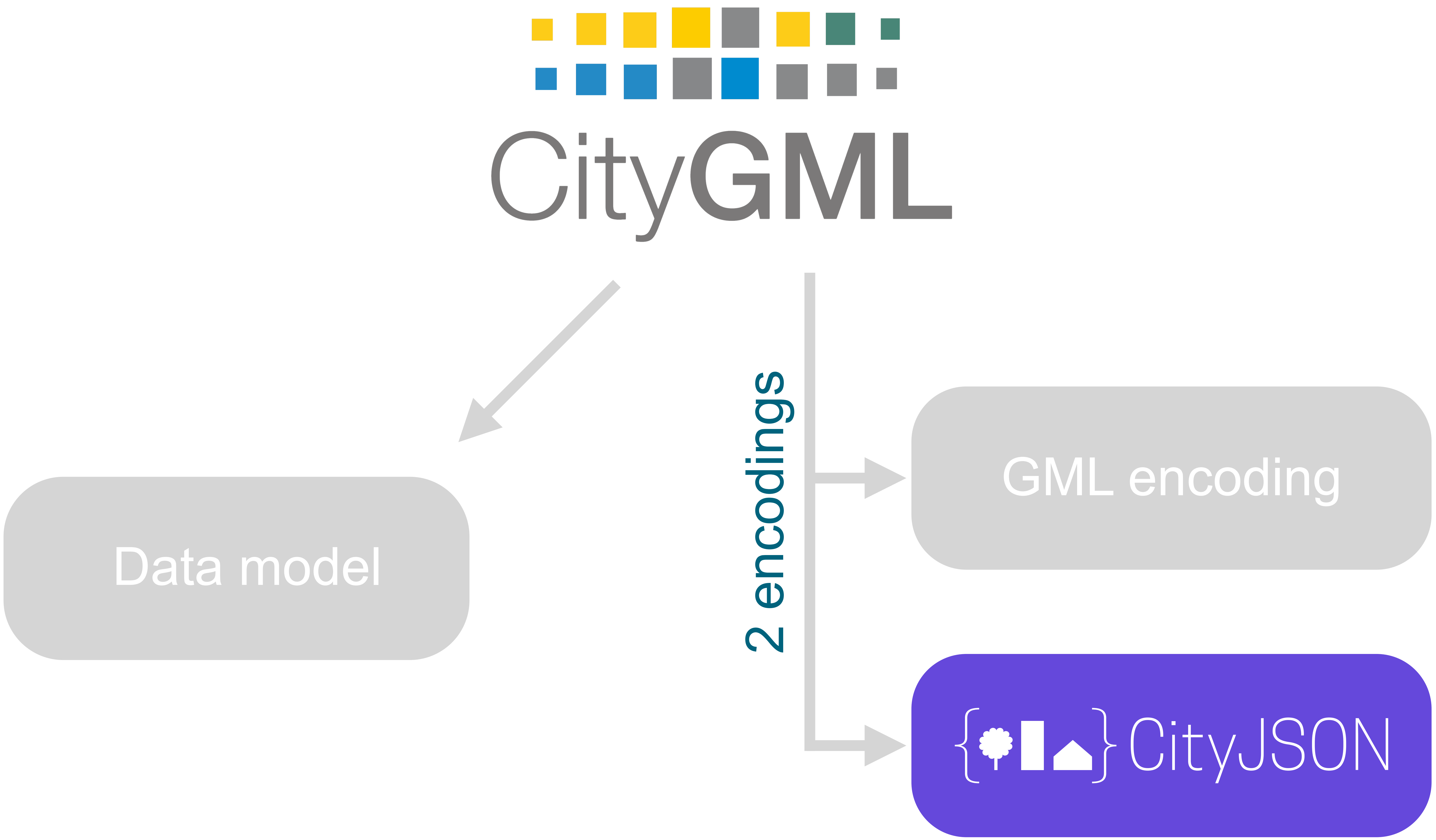


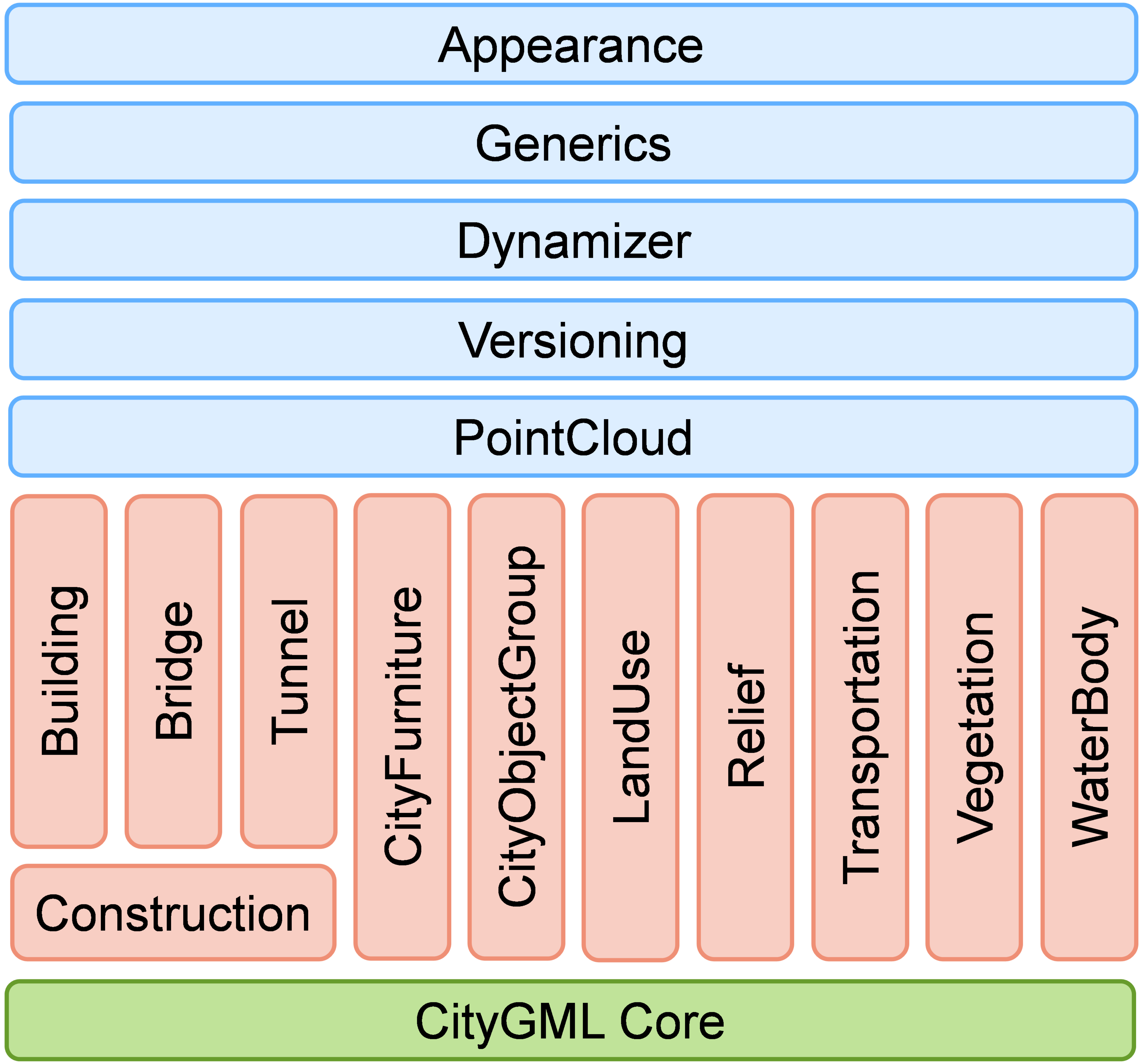
Image: Filip Biljecki



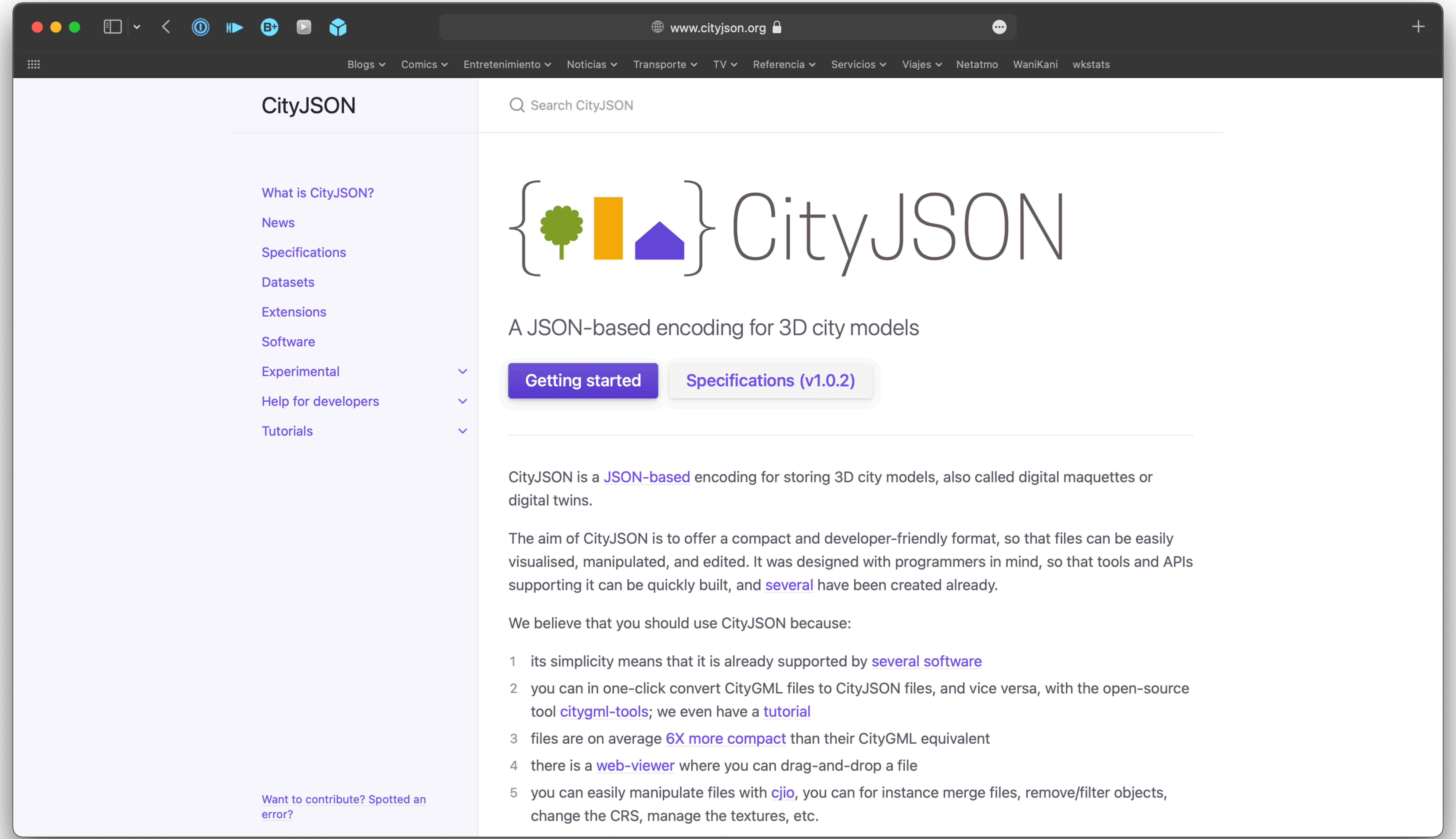
CityGML: data model and encodings



CityGML data model



CityJSON


www.cityjson.org
Blogs
Comics
Entretenimiento
Noticias
Transporte
TV
Referencia
Servicios
Viajes
Netatmo
WaniKani
wkstats
CityJSON
Search CityJSON
What is CityJSON?
News
Specifications
Datasets
Extensions
Software
Experimental
Help for developers
Tutorials
Want to contribute? Spotted an error?
CityJSON
A JSON-based encoding for 3D city models
Getting started
Specifications (v1.0.2)
CityJSON is a JSON-based encoding for storing 3D city models, also called digital maquettes or digital twins.
The aim of CityJSON is to offer a compact and developer-friendly format, so that files can be easily visualised, manipulated, and edited. It was designed with programmers in mind, so that tools and APIs supporting it can be quickly built, and several have been created already.
We believe that you should use CityJSON because:
1 its simplicity means that it is already supported by several software
2 you can in one-click convert CityGML files to CityJSON files, and vice versa, with the open-source tool citygml-tools; we even have a tutorial
3 files are on average 6X more compact than their CityGML equivalent
4 there is a web-viewer where you can drag-and-drop a file
5 you can easily manipulate files with cjio, you can for instance merge files, remove/filter objects, change the CRS, manage the textures, etc.

CityJSON for users

www.cityjson.org/software/

Blogs Comics Entretenimiento Noticias Transporte TV Referencia Servicios Viajes Netatmo WaniKani wkstats

CityJSON

- Datasets
- Extensions
- Software
- Schemas
- Specifications
- Experimental
- Help for developers
- Tutorials

Summary table

Software			View	Generate	Edit	Convert	Parse/API	Validate	Stor
3D City DB									●
3dfier				●					
Autoconverter			●			●			
azul			●						
citygml-tools						●			
citygml4j							●		
cjio						●	●	●	
cjval								●	
FME			●	●		●			
IFCCityJSON						●			
Measur3D			●						●
ninja			●		●				
ParaView reader			●						
QGIS plugin			●						
RhinoCity				●	●	●			
Schema Store							●		
Up3date			●		●				

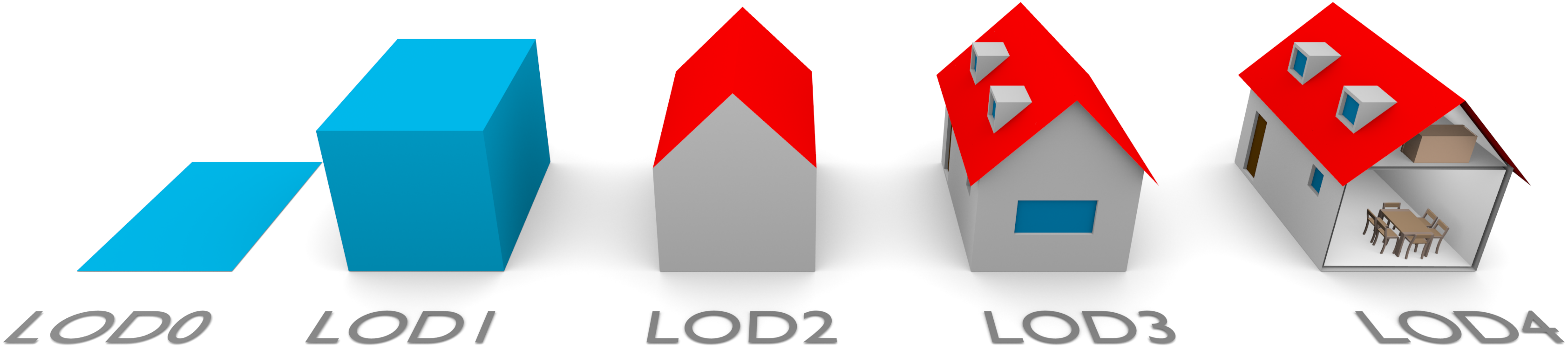
CityJSON for developers

```
import json

fin = open('mycity.json')
cm = json.loads(fin.read())

print "There are", len(cm['CityObjects']), "CityObjects"

# list all ids
for id in cm['CityObjects']:
    print "\t", id
```

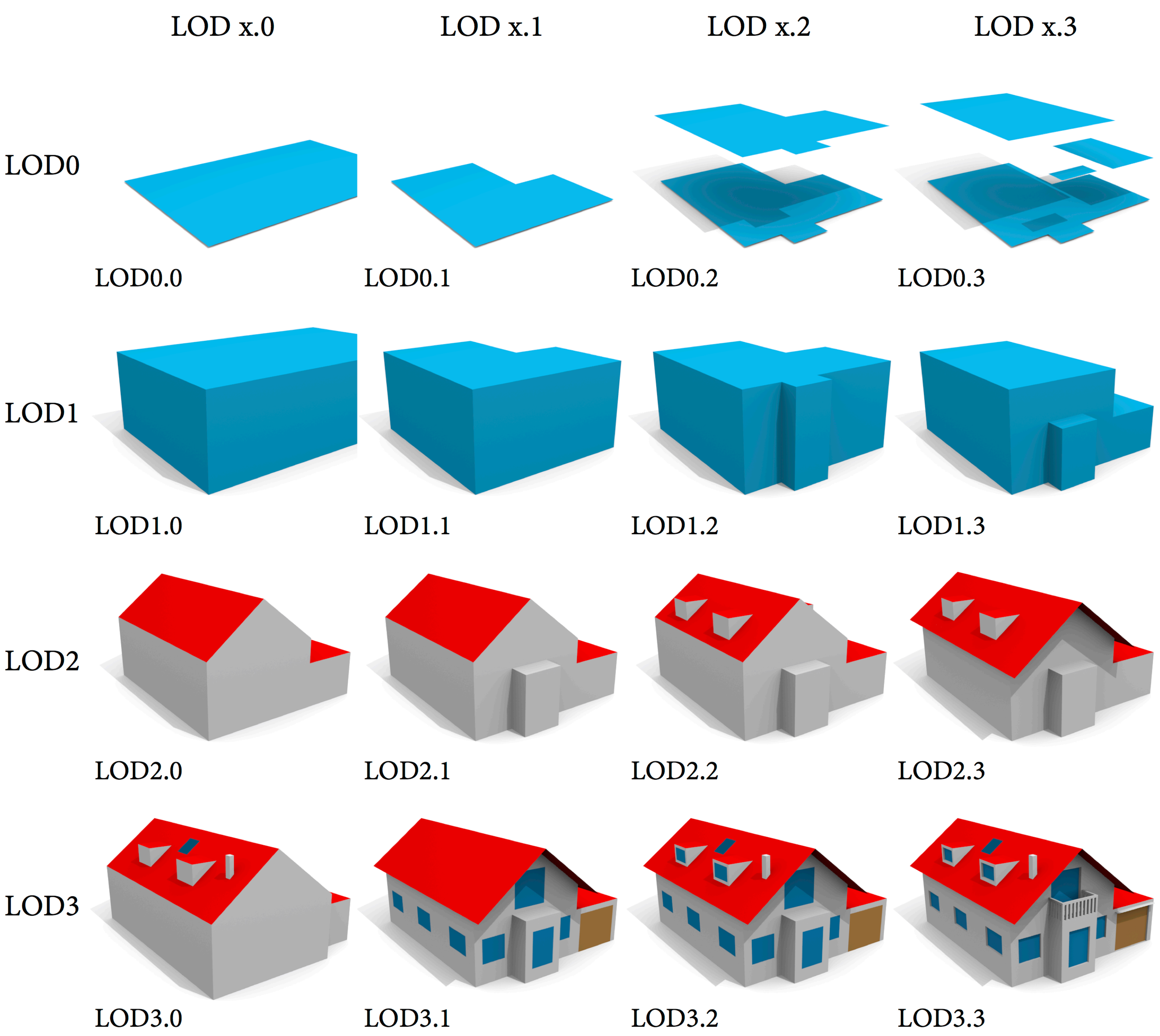
CityJSON

1st-level city objects	2nd-level city objects
Bridge	BridgePart, BridgeInstallation, BridgeConstructiveElement, BridgeRoom, BridgeFurniture
Building	BuildingPart, BuildingInstallation, BuildingConstructiveElement, BuildingFurniture, BuildingStorey, BuildingRoom, BuildingUnit
CityFurniture	
CityObjectGroup	
LandUse	
OtherConstruction	
PlantCover	
SolitaryVegetationObject	
TINRelief	
TransportationSquare	
Railway	
Road	
Tunnel	TunnelPart, TunnelInstallation, TunnelConstructiveElement, TunnelHollowSpace, TunnelFurniture
WaterBody	
Waterway	
+Extension	

CityGML LoDs



Extended LoDs



CityJSON: general structure

```
{
  "type": "CityJSON",
  "version": "2.0",
  "extensions": {},
  "transform": {
    "scale": [1.0, 1.0, 1.0],
    "translate": [0.0, 0.0, 0.0]
  },
  "metadata": {},
  "CityObjects": {},
  "vertices": [],
  "appearance": {},
  "geometry-templates": {}
}
```

CityJSON: city objects

```
"CityObjects": {
  "id-1": {
    "type": "Building",
    "attributes": {
      "roofType": "gabled roof"
    },
    "geographicalExtent": [ 84710.1, 446846.0, -5.3, 84757.1, 446944.0, 40.9 ],
    "children": ["id-56", "id-832", "mybalcony"]
  },
  "id-56": {
    "type": "BuildingPart",
    "parents": ["id-1"],
    ...
  },
  "mybalcony": {
    "type": "BuildingInstallation",
    "parents": ["id-1"],
    ...
  }
  ...
}
```

CityJSON: city objects

```
"CityObjects": {
  "id-1": {
    "type": "LandUse",
    "attributes": {
      "function": "Industry and Business",
      "area-parcel": {
        "value": 437,
        "uom": "m2"
      },
    },
    "geometry": [{...}]
  },
  "id-2": {
    "type": "WaterBody",
    "attributes": {
      "name": "Lake Black",
      "some-list": ["a", "b", "c"]
    },
    "geometry": [{...}]
  }
}
```

CityJSON: vertices

```
"vertices": [
  [102, 103, 1],
  [11, 910, 43],
  [25, 744, 22],
  ...
  [23, 88, 5],
  [8523, 487, 22]
]
```

CityJSON: MultiPoint and MultiLineString

```
{
  "type": "MultiPoint",
  "lod": "1",
  "boundaries": [2, 44, 0, 7]
}
```

```
{
  "type": "MultiLineString",
  "lod": "1",
  "boundaries": [
    [2, 3, 5], [77, 55, 212]
  ]
}
```

CityJSON: MultiSurface (same as CompositeSurface)

```
{
  "type": "MultiSurface",
  "lod": "2",
  "boundaries": [
    [[0, 3, 2, 1]], [[4, 5, 6, 7]], [[0, 1, 5, 4]]
  ]
}
```

CityJSON: Solid

```
{
  "type": "Solid",
  "lod": "2",
  "boundaries": [
    //-- exterior shell
    [ [[0, 3, 2, 1, 22]], [[4, 5, 6, 7]], [[0, 1, 5, 4]], [[1, 2, 6, 5]] ],
    //-- interior shell
    [ [[240, 243, 124]], [[244, 246, 724]], [[34, 414, 45]], [[111, 246, 5]] ]
  ]
}
```

CityJSON: CompositeSolid (same as MultiSolid)

```
{
  "type": "CompositeSolid",
  "lod": "3",
  "boundaries": [
    [ //-- 1st Solid
      [ [[0, 3, 2, 1, 22]], [[4, 5, 6, 7]], [[0, 1, 5, 4]], [[1, 2, 6, 5]] ],
      [ [[240, 243, 124]], [[244, 246, 724]], [[34, 414, 45]], [[111, 246, 5]] ]
    ],
    [ //-- 2nd Solid
      [ [[666, 667, 668]], [[74, 75, 76]], [[880, 881, 885]], [[111, 122, 226]] ]
    ]
  ]
}
```

CityJSON: semantic surfaces

```json
{
  "type": "RoofSurface",
  "slope": 16.4,
  "children": [2, 37],
  "solar-potential": 5
}
```

```json
{
  "type": "Window",
  "parent": 2,
  "type-glass": "HR++"
}
```

CHEK DIGITAL BUILDING PERMIT

CityJSON: semantic surfaces

```
{
  "type": "MultiSurface",
  "lod": "2",
  "boundaries": [
    [[0, 3, 2, 1]],
    [[4, 5, 6, 7]],
    [[0, 1, 5, 4]],
    [[0, 2, 3, 8]],
    [[10, 12, 23, 48]]
  ],
  "semantics": {
    "surfaces" : [
      {
        "type": "WallSurface",
        "slope": 33.4,
        "children": [2]
      },
      {
        "type": "RoofSurface",
        "slope": 66.6
      }
    ],
    "values": [0, 0, null, 1, 0]
  }
}
```

CHEK DIGITAL BUILDING PERMIT

CityJSON: semantic surfaces

```
{
   "type": "CompositeSolid",
   "lod": "2.2",
   "boundaries": [
     [ //-- 1st Solid
       [ [[0, 3, 2, 1, 22]], [[4, 5, 6, 7]], [[0, 1, 5, 4]], [[1, 2, 6, 5]] ]
     ],
     [ //-- 2nd Solid
       [ [[666, 667, 668]], [[74, 75, 76]], [[880, 881, 885]] ]
     ]
   ],
   "semantics": {
     "surfaces" : [
       {
         "type": "RoofSurface"
       },
       {
         "type": "WallSurface"
       }
     ],
     "values": [
       [ //-- 1st Solid
         [0, 1, 1, null]
       ],
       [ //-- 2nd Solid get all null values
         [null, null, null]
       ]
     ]
   }
 }
```

CityJSON: geometry templates


```json
"geometry-templates": {
  "templates": [
    {
      "type": "MultiSurface",
      "lod": "2.1",
      "boundaries": [
         [[0, 3, 2, 1]], [[4, 5, 6, 7]], [[0, 1, 5, 4]]
      ],
      "semantics": {
        "surfaces" : [
          {
            "type": "+Skylight",
          },
          {
            "type": "+PatioDoor",
          }
        ],
        "values": [0, 0, 1]
      }
    },
    ...
  ],
  "vertices-templates": [
    [0.0, 0.5, 0.0],
    ...
    [1.0, 1.0, 0.0],
    [0.0, 1.0, 0.0]
  ]
}
```

CityJSON: geometry templates


```
{
  "type": "SolitaryVegetationObject",
  "geometry": [
    {
      "type": "GeometryInstance",
      "template": 0,
      "boundaries": [372],
      "transformationMatrix": [
        2.0, 0.0, 0.0, 0.0,
        0.0, 2.0, 0.0, 0.0,
        0.0, 0.0, 2.0, 0.0,
        0.0, 0.0, 0.0, 1.0
      ]
    }
  ]
}
```

CityJSON: transform object

```
"transform": {
    "scale": [0.001, 0.001, 0.001],
    "translate": [442464.879, 5482614.692, 310.19]
}
```

Questions?

- Why 3D?
- 3D geometries in geoinformation (ISO19107)
- 3D city models
- Practical session

Cologne Image:cityframes.de/

Practical session 3

- Find a 3D city model that interests you in https://www.cityjson.org/datasets/
- Open it in CityJSON Ninja: https://ninja.cityjson.org/
- Analyse its main characteristics: which classes are included? what is its LoD or LoDs? which semantic surfaces are modelled?
- Try opening it in QGIS through the CityJSON plug-in. What happens? Check both the default view and the 3D Map.

References

- Otto Huisman and Rolf A. de By. **Principles of Geographic Information Systems**. 4th Edition. 2009.

- Matt Duckham, Qian (Chayn) Sun and Michael F. Worboys. **GIS: A Computing Perspective**. 3rd Edition. 2024.