Building Attribute Extraction and Street-Level Image Query

1.Introduction

The goal of the project is to be able to build a pipeline to process CityGML data and extract building footprints. The parsed data is also used in conjunction with Mapillary to retrieve street-level images around a given building.

Key Assumptions

- The file follows CityGML 1.0 specification standard with Level of Detail 2 (LoD 2)
- Ground surfaces accurately represent building footprints and building elements contains at least one ground surface.
- All coordinates within a ground surface element share the same elevation.

2. Tasks

2.1 Extract Building Footprints

The building footprint extraction was achieved through XML parsing of CityGML files using Python's ElementTree library. The parser systematically walked through the tree structure, identifying bldg:Building elements as parent nodes and then drilling down through child elements to locate bldg:GroundSurface nodes containing the geometric data. For each ground surface, the parser used the find() method to locate gml:Polygon elements and ultimately extract gml:posList elements containing 3D coordinate strings. These coordinates were converted to 2D footprints by extracting only the X and Y components, assuming uniform elevation across ground surfaces. The extracted geometries were then visualized using two approaches: interactive web maps generated as a .html file with GeoPandas, and static plots created using Matplotlib with custom polygon rendering.

2.2 Extract Building Attributes

Various attributes of the building were extracted from building metadata. Similar to the previous task, the parser was able to navigate through the hierarchical structure to retrieve various attributes of each building. Missing attributes were handled by implementing null value management. All extracted data was structured and exported to CSV format for better readability.

2.3 Retrieve Building Images from Mapillary

This task utilized Mapillary's street-view imagery service through their Python SDK to retrieve the street-level images with in a 50-meter radius of each building. To establish precise spatial reference points for each building, the geometric centroids was calculated from the ground surface polygons using Shapely's centroid computation method. The coordinates were transformed from the local EPSG:25832 coordinate reference system to WGS84 geographic coordinates suitable for the Mapillary API. These centroid coordinates was then utilized to execute imagery queries using the mapillary.interface module. The returned GeoJSON response was then filtered to display only the image ids from the feature property of the image metadata.

3. Challenges

The most significant challenge encountered was the coordinate system handling, which required careful analysis and transformation to ensure API compatibility. The Mapillary documentation clearly mentioned that the service operates exclusively on the WGS84 coordinate system (EPSG:4326), while the CityGML dataset used a different coordinate reference system.

To identify the appropriate CRS code for the input data, the EPSG.io spatial reference database was consulted, which revealed the corresponding code EPSG:25832 for the ETRS89 / UTM zone 32N coordinate system .GeoPandas CRS transformation capabilities were then employed using the to_crs() method to perform accurate coordinate conversion from the local projected coordinate system to the geographic WGS84 system required by Mapillary's API endpoints.

Data quality management was addressed through robust parsing functions that accommodated inconsistent attribute availability across buildings. The system implemented comprehensive null checking for optional elements and ensured graceful degradation when expected attributes were unavailable.

4. Libraries

- **Element Tree** module from the **xml** package
- **GeoPandas** – Geospatial Data Handling and Coordinate Reference System Transformation
- **Shapely** – Polygon geometry handling
- **Matplotlib** – Static Visualizations
- **Mapillary** – Street Level Image Retrieval