**Task1: dxf to geojson**

**Intro**

The main aim of Task1 is converting autoCAD file(dxf) to geojson file. In converting method automate cleaning method should be concerned. That means only the interested lines, such as wall and door should be extracted. To solve this problem 'ezdxf' library for Python is chosen. 'exdxf' is a Python package to create and modify dxf drawings. With this library all the elements from dxf file can be converted to python instance. According to other researches, the other researches were processed with pre-cleaned and pre-organized autoCAD file with autoCAD software manually.

**Step of process**

1. (clean uninterested elements on autoCAD)
2. Extract elements from autoCAD dxf file to Python instance with ezdxf library
3. Convert Custom Reference System to EPSG 32632
4. Divide door and wall layer
5. Get intersection of door and wall instance
6. Create each polygon of room and door
7. Create geojson file

**Issues/Todo**

1. With ezdxf, german letter [öüäß] cannot be used in layer name => change the layer name with autoCAD
2. With ezdxf, the arc element cannot be used => extract elements except arc
3. Variable layers name of wall layer => unify layer on autoCAD file or define new category with Python script
4. Multiple layers of wall layer => find most outer, inner walls
5. How to make polygon each room with door and wall elements => shapely poligonize, using buffer to combine door
6. Remove uninterested elements => using LISP with autoCAD, create cleaning function(LISP used in autoCAD is creating the function) / cleaning algorithm with Pyghon script
7. How to use property info in autoCAD file, such as room number, floor info, etc => Properties in aucoCAD file is wrote as text
8. Block in autoCAD file -> get coordinate of door block(start point is not on same position of each door block)/ explode each block and assign id for each door block
9. Get intersection of right door and wall=> most outer and inner wall with door rectangle
10. Add information of layer, category(door, wall…), door block info, elements id to each element
11. Leave only the rectangle wall shape which stick to wall
12. Remove null, short lines
13. Plan B -> get the Image from AutoCAD file + detect doors and lines from image of autoCAD file(**Open cv - vector->raster connect component adjust )**

**Pre-processing with autoCAD**

1. Remove letter of german letter (öüäß) with autoCAD => layer name
2. Remove unused part from each block(floor of Fassde block, vertical line of door block)

**Using Libraries**

Ezdxf

Geopandas

Shapely

json

geojson

openCV

AutoCAD modify

Explode blocks

라인, 도표, 평행, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**Task2: image processing with rescue plan image to geojson**

Symbol removal, line detection -> improve original code(binary2) / Edge detection with OpenCV(Canny Edge detection), line detection

Vectorize => current result is 'csv' format, change it to geojson/dwg format

Improve processing speed

Symbol removal

**Process with Window GUI with tkinter**

1. Rectification of the image
2. Creation of a binary image
3. Detect neighborhoods
4. Detecting lines of the outer edges (vectorize)
5. Filtering of the generated lines => final result: csv file

Task1: dxf(AutoCAD) file to geojson

Open dxf with AutoCAD

Delete uninterested layer with LAYDEL

Original dxf of 4OG

지도, 스크린샷이(가) 표시된 사진

자동 생성된 설명

After LAYDEL of 4OG

스크린샷이(가) 표시된 사진

자동 생성된 설명

before

스크린샷, 라인, 평행이(가) 표시된 사진

자동 생성된 설명

After deleting uninterested objects - Deleting arcs from the door blocks.

스크린샷, 라인, 평행, 지도이(가) 표시된 사진

자동 생성된 설명

Explode all duplicated blocks

Select all same blocks(QSELECT) -> explode(EXPLODE)

Assume that only the doors are block

Only interested layers

스크린샷, 그림, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

스케치, 그림, 흑백이(가) 표시된 사진

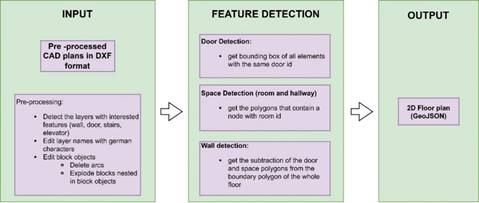
자동 생성된 설명

Keep only the real door part + make a block for each door

The aim was to generate a workflow that needs minimum manipulation of the DXF files. Nevertheless, some pre-adjustments are indispensable because of the complexity of CAD files and the lack of python libraries that can handle DXF files. Therefore, the workflow starts with the pre-processing of the DXF file, namely:

* Editing the layer names that contain German characters (ä, ö, ü, ß),
* Exploding blocks nested in another block feature,
* Deleting arcs from the door blocks.

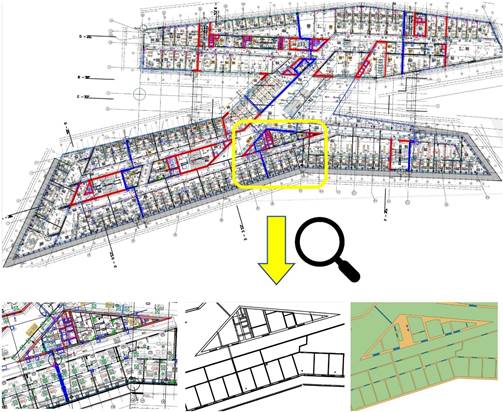
Once the pre-processing is completed, layers of interest that keep the wall, door, stairs, and elevator features are extracted using the python library EZDXF. Then, the extracted features are written into a GeoJSON file for the ease of further processing steps.



*Figure 1 High-level representation of the automatized workflow to generate 2D floor plans from CAD plans*

The detection of the door, spaces, and wall polygons is respectively applied. In DXF files, a door is often represented as a block feature, a group of features that make a door. Blocks are exploded during the extraction by keeping track of all elements of each door with a door id. Eventually, the bounding box around all elements sharing the same door id is generated and written into a separate GeoJSON file as door polygons of the final floor plan.

The whole extracted plan is polygonised before starting the space (rooms and hallways) detection. Then, two nodes, at around 41 cm (the highest door width value) offset from the centre of the earlier generated door polygons on both sides, are added with a room id. Then a spatial analysis is done to filter the polygons that contain nodes generated on both sides of the doors. These polygons are room and hallway polygons. So far, doors, rooms and hallways have been detected. Next, walls are detected as polygons with a spatial analysis, where the detected feature polygons are subtracted from a boundary polygon of the whole floor. The result is a polygonised 2D floor plan generated from a DXF file (Fig.2).



*Figure 2 The 2D floor plan generation from CAD plan. Above is the original CAD plan. Below is the zoomed view, from left to right; pre-processed plan, extracted features, and the final product - 2D floor plan*