

Graduation Plan

Master of Science Geomatics for the Built Environment

Semiautomatic extraction of loops and openings from 2D floor plans for 3D reconstruction

Name: Haoxiang Wu

Student number: 4325591

Graduation professor: Sisi Zlatanova

Daily supervisor: Liu Liu

Co-reader: Marianne Vries

Content

- Motivations
- Objective
- Research questions
- Research scope
- What is architectural floor plan?
- Challenges
- Related work
- Constrains
- Our proposed methods
- Results
- Conclusion & future work
- Reference

Motivations

- Needs for 3D models:
 - People spend more time indoor than outdoors
 - Indoor-related services or applications limited to 2D
 - An efficient tool to present indoor environment
- Manually create 3D models
 - Time-consuming and labor-intensive
- Photogrammetry and point cloud
 - Data resources limited and restrictive
- 2D architectural floor plans
 - Widespread, commonly available
 - Promising data source for 3D models

Objective

Thus, to automate the process of using 2D floor plans to reconstruct 3D models is of great significance.

- Objective:
 - *To propose a (semi-)automatic process to extract information from 2D CAD architectural floor plans needed for 3D reconstruction*

Research questions

- Main research question:
 - *To which extend is it possible to use 2D architectural floor plans as input data for automatic 3D reconstruction?*
- Three underlying questions:
 - *(1) What is the most basic information that can be extracted from 2D 2D architectural floor plans for 3D reconstruction?*
 - *(2) What characters should input floor plans have to facilitate an automatic extraction of these information?*
 - *(3) In what way can a floor plan be automatically processed for 3D reconstruction?*

Research scopes

- This thesis only deals with floor plans of a single floor. The superimposition of each floor to building a complete building will not be covered.
- The reconstruction algorithm. The aim of this thesis is to investigate the possibilities to extract useful information from 2D architectural floor plans for 3D reconstruction. The thesis will use algorithm developed by other researcher to perform 3D reconstruction with the extracted information.

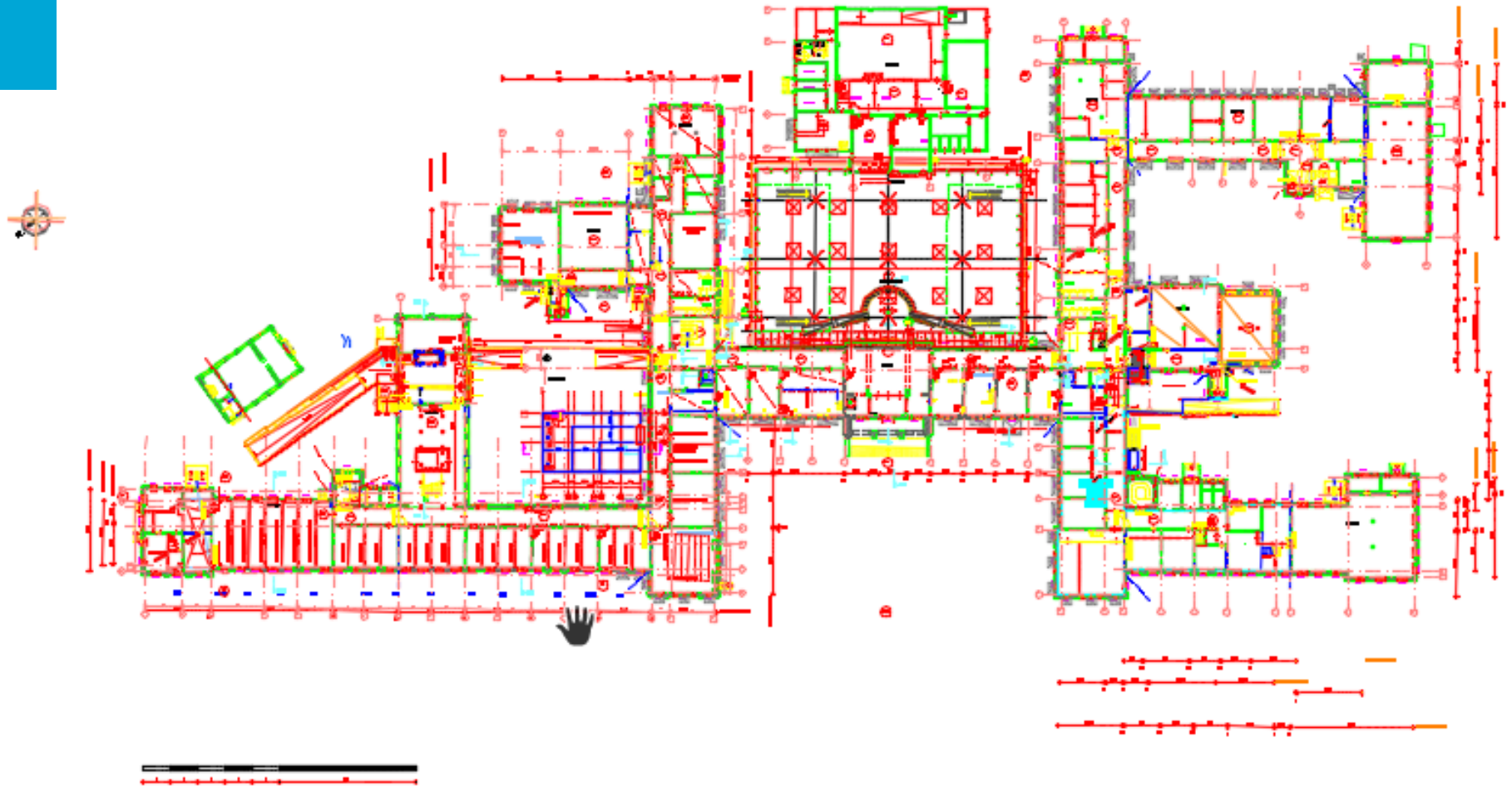
What is architectural floor plan?

- Architectural drawings
= **floor plans** + elevations + sections + details + ceiling plans
+ finished schedules + mechanical information +
- Among them, floor plan is the most importance one
- Technically, an **aerial plan view** that is horizontally cut approximately **4 feet** above the floor.
- Drawn to a scale, with different line weights and line types

Source: [1, 2]

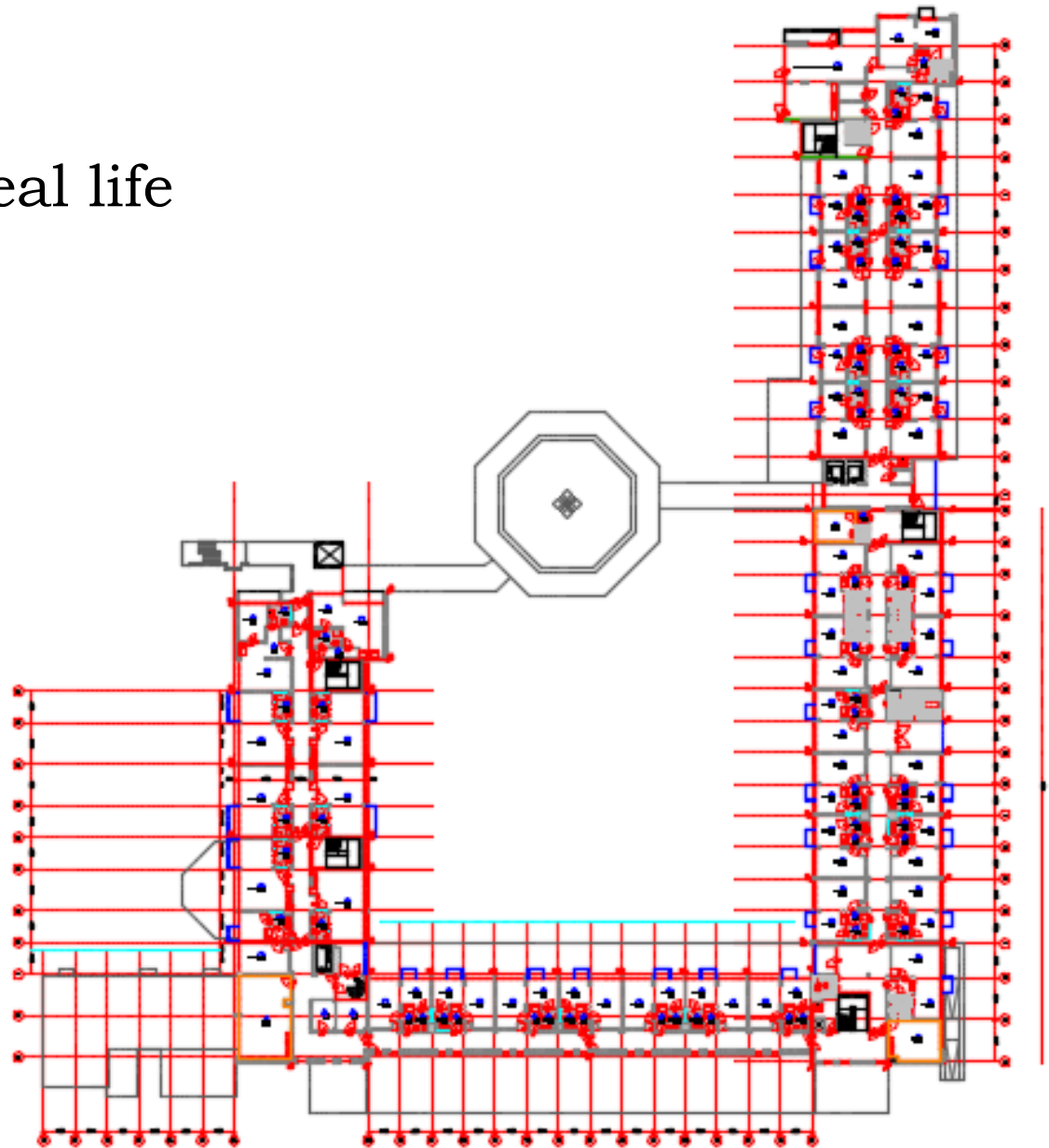
Challenges

Floor plans in real life



Challenges

Floor plans in real life



Challenges

Floor plans in real life

- Structural objects: walls and columns
- Openings: windows and doors
- Texture / cladding
- Dimensions / dimension lines
- Texts
- Annotations
- Furniture
- Sanitary outfits
- Scale
- North arrow
-

most basic

represent the overall
spatial subdivision of
the whole floor

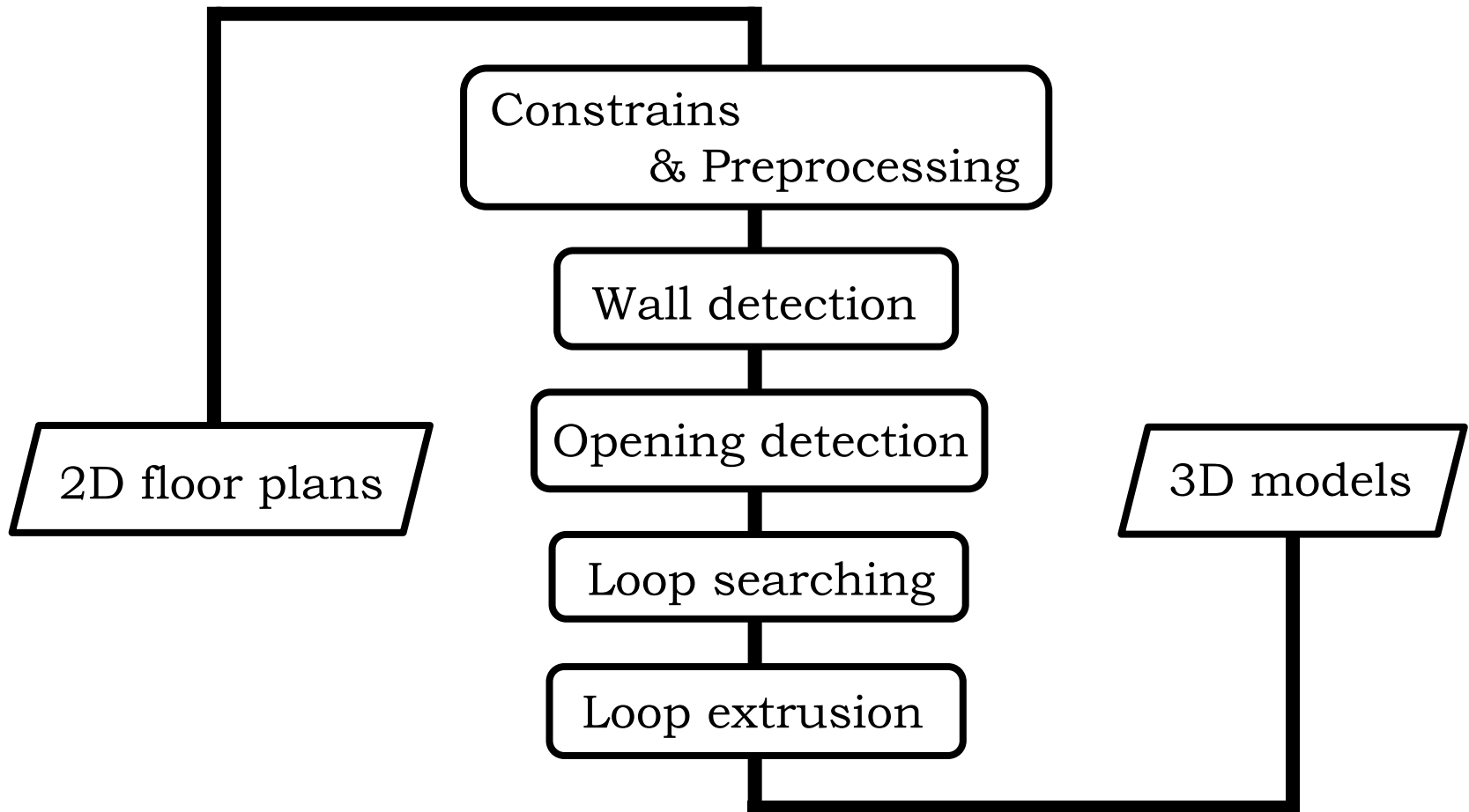
excessive information

disturbing primitives

Source: [2-5]

Related works

(1) General pipeline



Related works

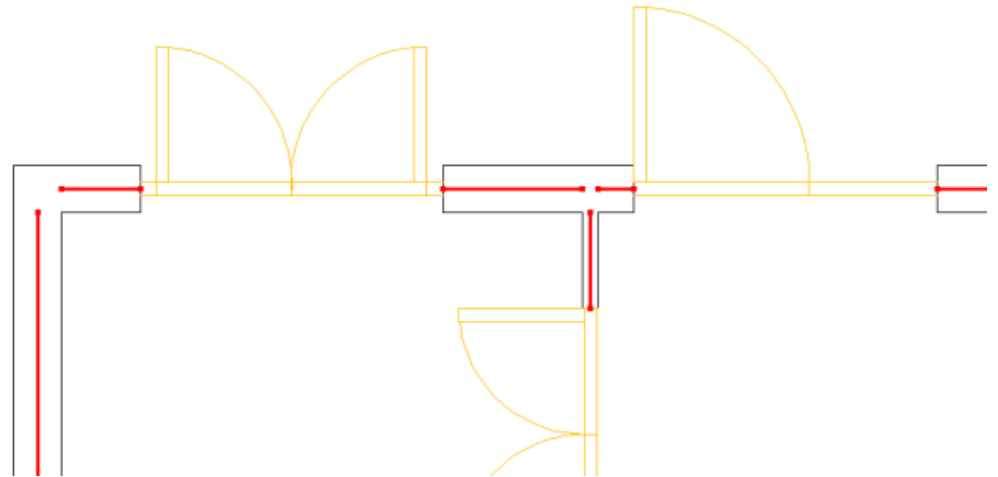
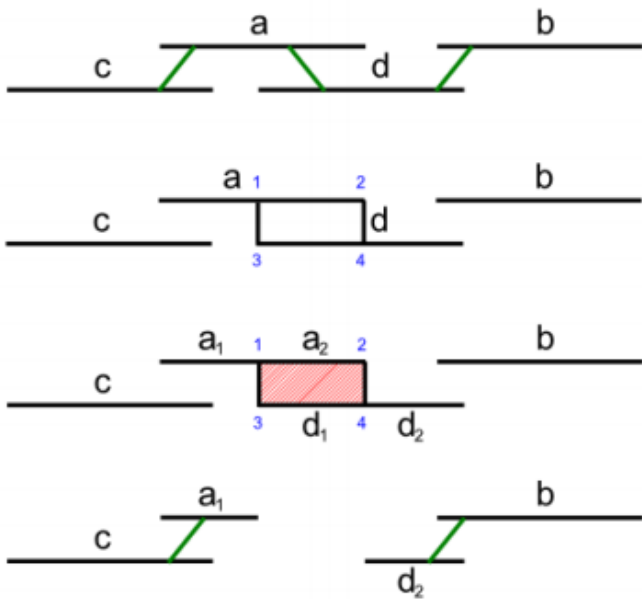
(2) Wall detection

Two directions:

- (1) start from walls, detect walls as parallel line pairs
 - Lu et al., 2007
 - Park and Kwon, 2003
 - Domínguez et al., 2012
- (2) start from openings, do not detect walls at the first step, but search for closed loop after opening is detected
 - Lewis and Séquin, 1998
 - Zhu et al., 2013

Related works

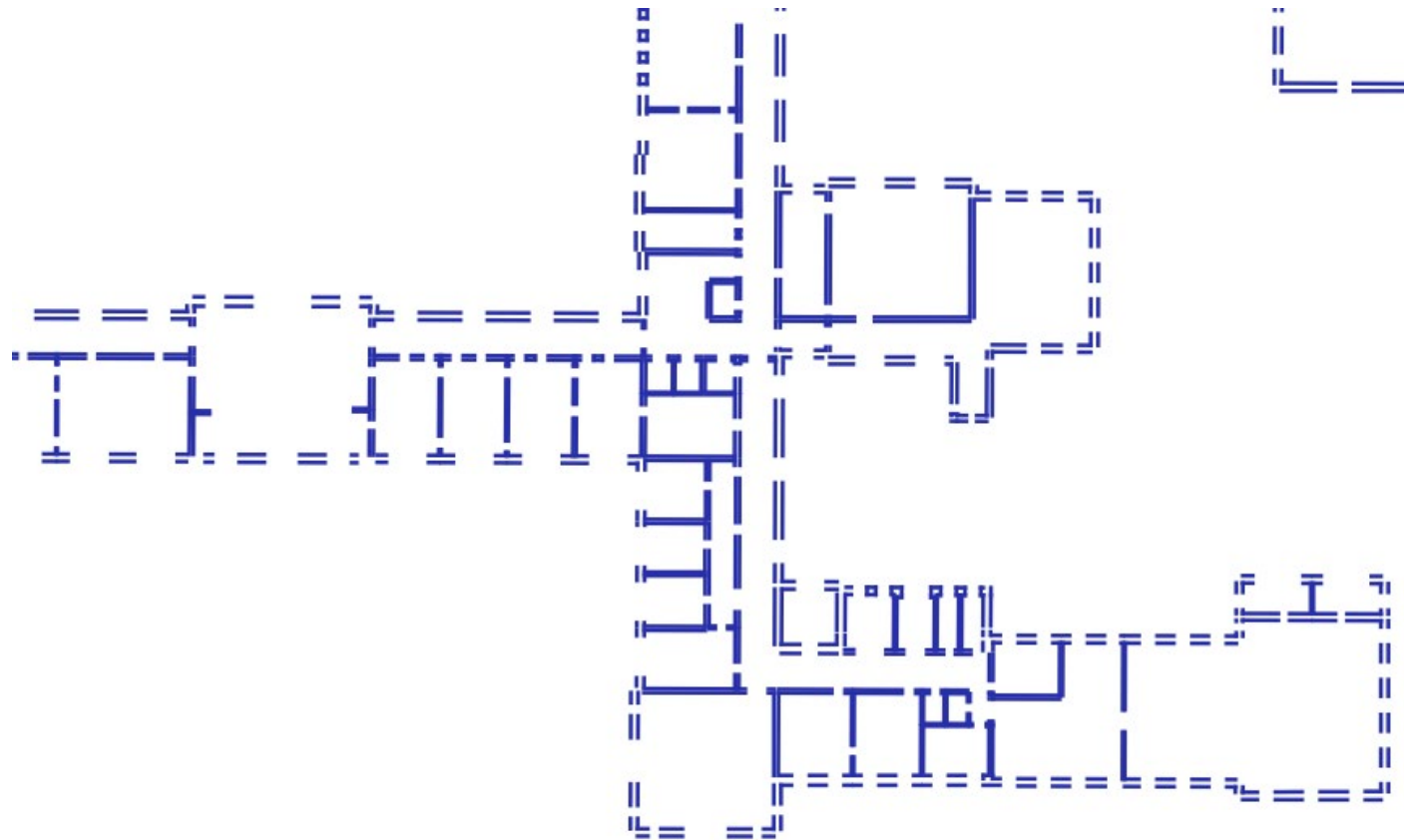
(2) Wall detection



Source: [8]

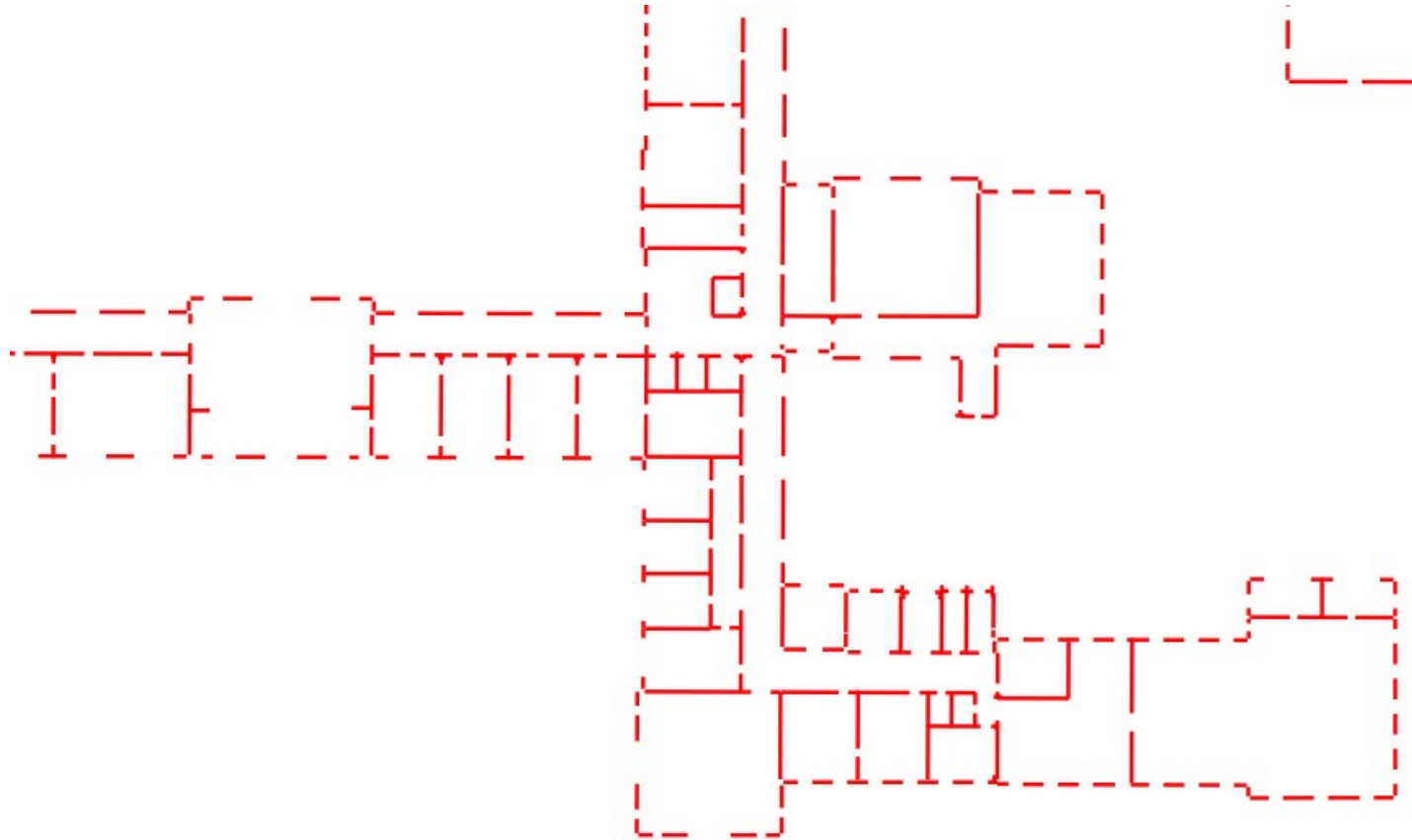
Related works

(2) Wall detection



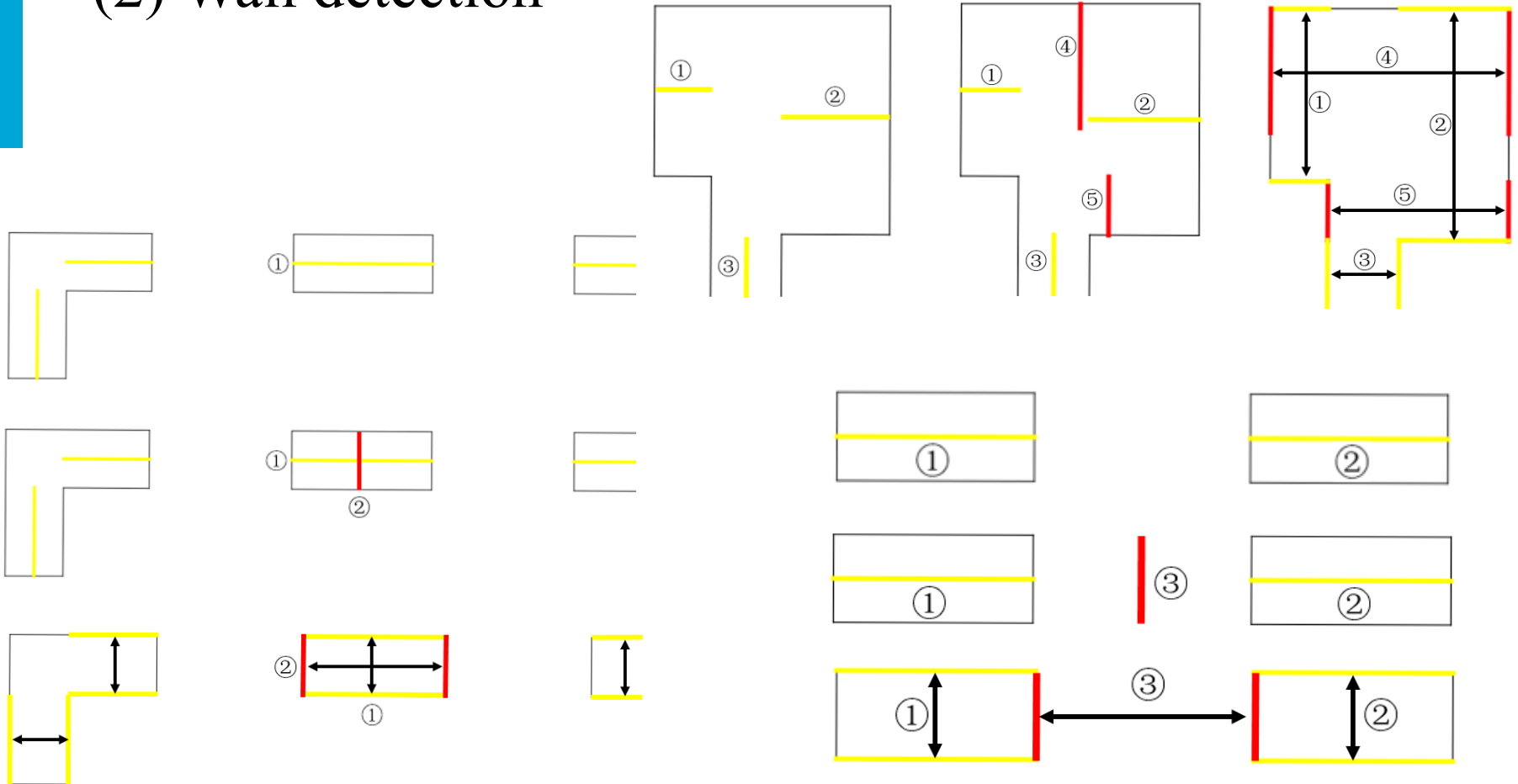
Related works

(2) Wall detection



Related works

(2) Wall detection



Related works

(3) Opening detection

Two directions:

- (1) Symbol recognition --- more generic but still “bottleneck”
 - Lewis and Séquin, 1998 --- Required to be simplified by the user to conform to pre-specified representation
 - Lu et al., 2007
 - Guo et al., 2012
 - Zhu et al., 2013

} Only considered limited geometrical relations
Only perform well in certain cases
- (2) Block bounding box
--- easier but human intervention needed

Related works

Based on the analysis described above, some conclusions are reached:

- Single lines cannot be trusted as basic elements to correctly represent walls because there might unavoidably be some falsely detected wall lines.
- In this thesis, we use polygons as basic elements of both walls and openings to reconstruction the spatial layout of the floor.
 - No need for wall detection
 - No need for loop searching
 - Loops can be easily retrieved from merged polygons
- Opening reconstruction will use block bounding box

Constrains

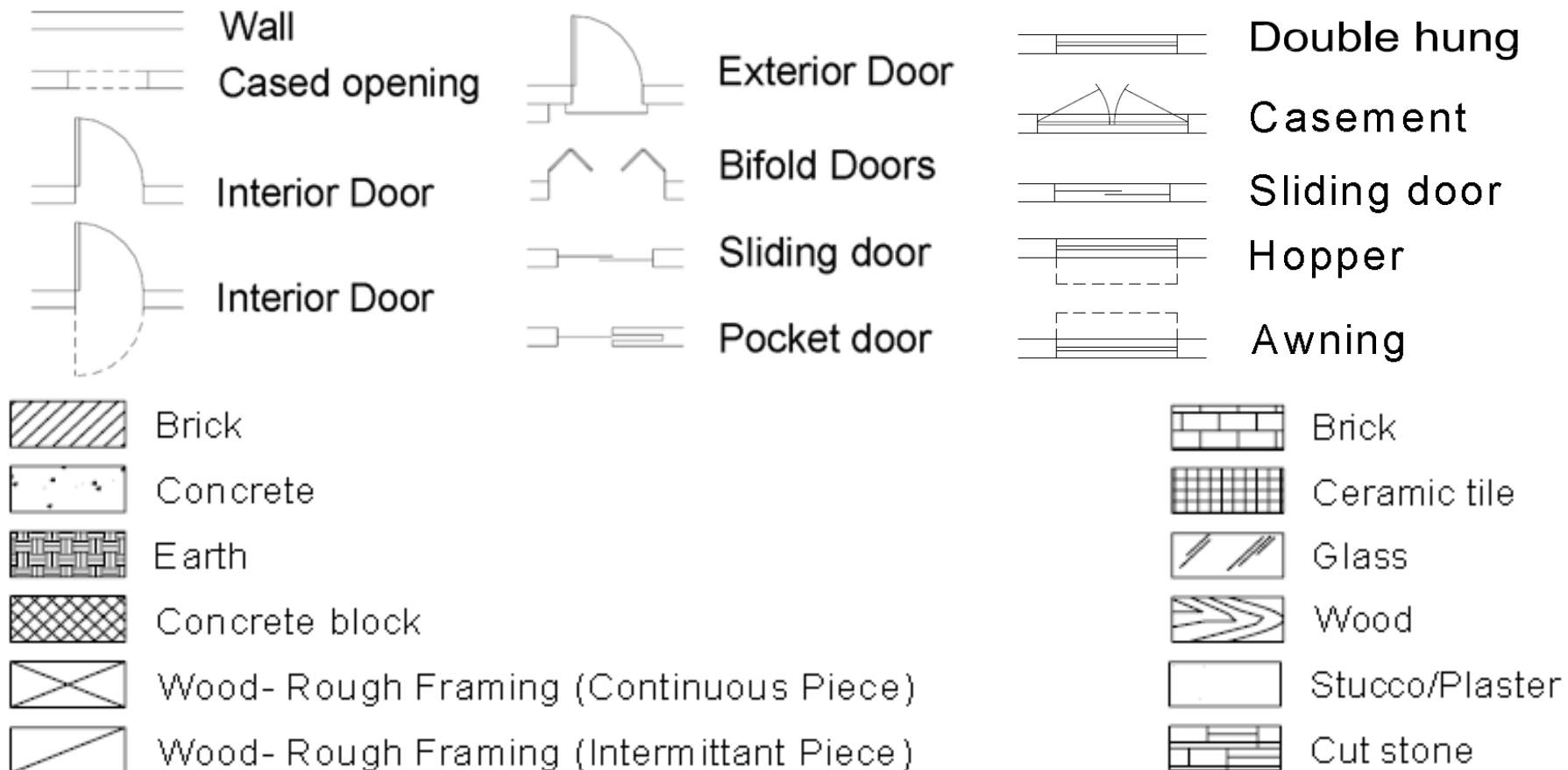
Constrain 1: CAD file will be converted into DXF format

- Vectorized floor plans vary in format
- DXF --- Drawing Exchange Format
- One of the most widely supported vector formats
- Open standard, specifications publicly published
- Support various object types
- ASCII version exists, easy format to parse
- Open-source libraries available (e.g. dxfgrabber, dxfwrite, ezdxf, SDXF)

Source: [11-14]

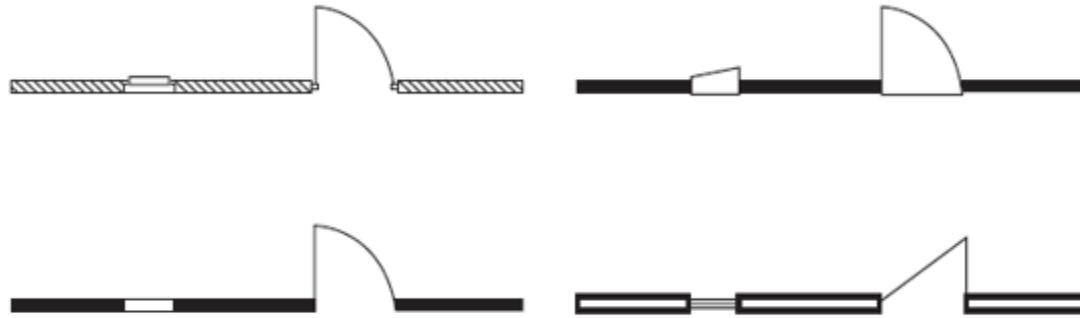
Constrains

Some common rules and standards exist [17-18]



Source: [2]

Constrains



- None of these standards are mandatory
- Differences exist between different standards and libraries [15]
- Designers can choose freely among them based on their purpose for the drawing
- Symbols subject to designers' drafting habit and artistic incline [16]
- Characters of a same symbol can change emphasizing on different aspects at different design stages [19]

Constrains

Constrain 2: Symbols of openings are saved as blocks.

- Blocks allow designers to easily instantiate repeating symbols (e.g. doors, windows and stairs)
- Primitives representing opening symbols are assumed to be blocked together by designer in the drawing process, or manually fulfilled by user.
- In this thesis, block bounding box will be used to reconstruct openings

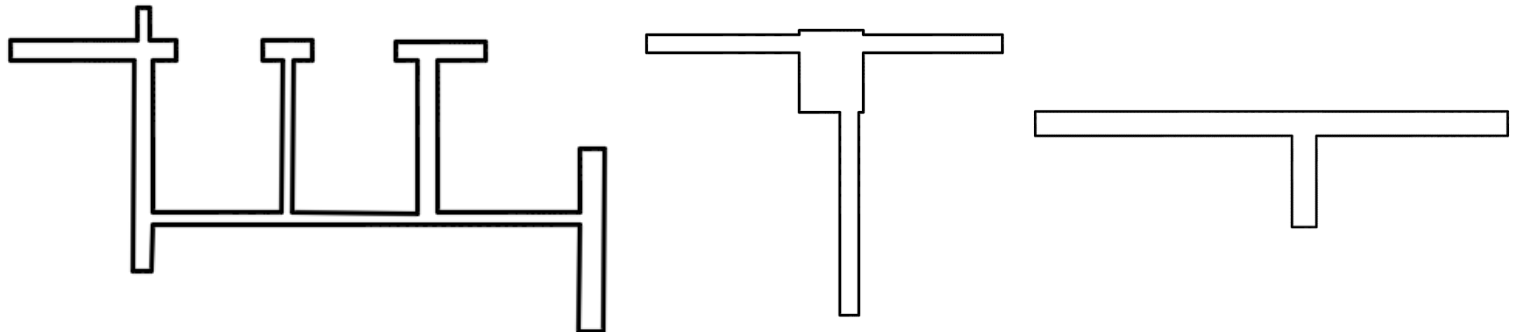
Constrains

Constrain 3:

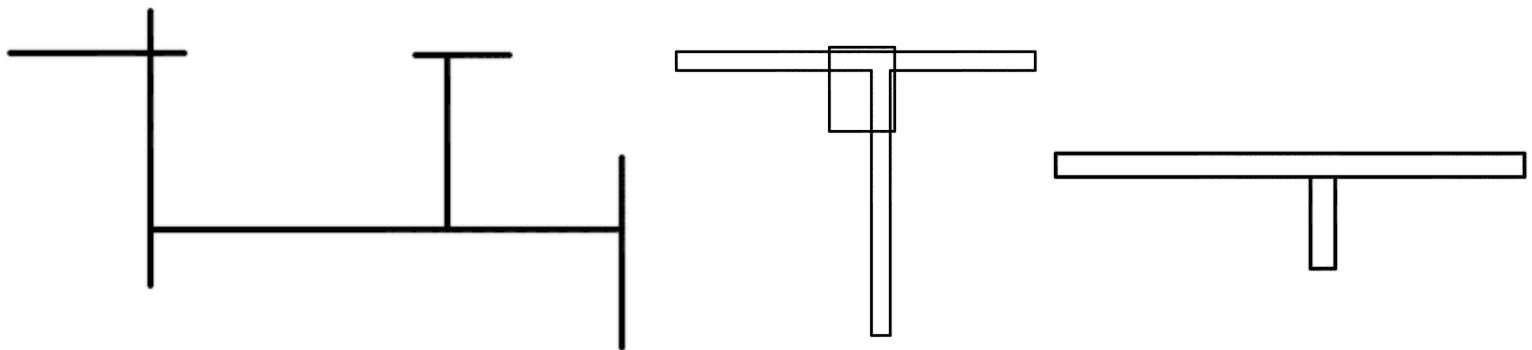
Structural objects are represented by polygons, within which walls are represented by parallel line pairs. Polygons representing structural objects do not intersect with each other. Otherwise, intersecting polygons will be joined into one polygon.



Expected



Ambiguity



Constrains

Constrain 4:

Contents of walls, windows and doors are separately stored in different layers. Name of each layer will be provided by user. Other information needs to be removed beforehand.

- Structural objects (walls and columns) and openings (windows and doors) are the most basic elements composing the network that subdivides the space of the whole floor
- Other information in floor plans also contributes to the semantics and topology of 3D models, but will not be included in this thesis

Preprocessing

- Manually generated input floor plans typically suffer from many drafting errors and redundancies [9]
- Visually imperceptible
- Cause unpredictable results for later algorithm
- Only affect wall layer
- Typical drafting errors:
 - Null-length lines
 - Duplicates
 - Disjoint vertices

Preprocessing

(1) Null-length lines

- Two types:
 - ‘LINE’ primitives whose start point and end point are both assigned to a same point (length is zero)
 - Lines that are shorter than a given threshold
- In this thesis, this threshold is set to be 5mm
- Will be excluded when data is read from DXF file

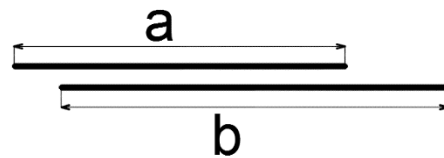
Preprocessing

(2) Duplicates

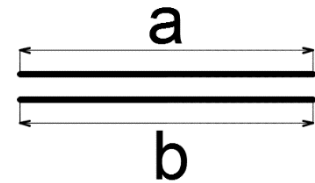
- Happen when a single edge of a polygon in the floor plans is mistakenly represented by multiple lines

- Five cases:

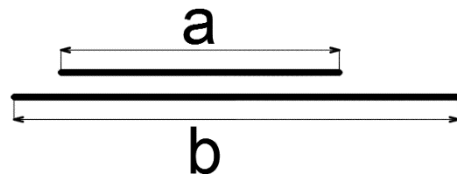
- Overlapping



- Identical

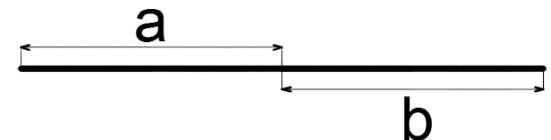


- Containing



- Contained

- Consecutive

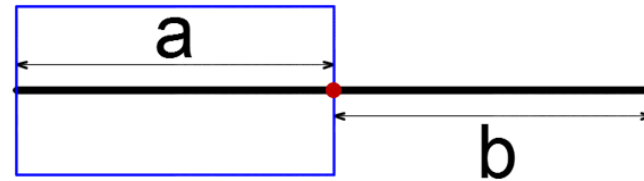
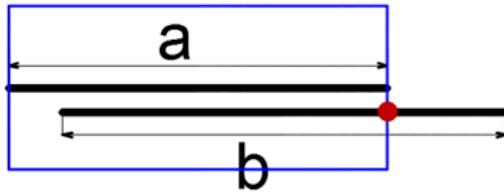


Preprocessing

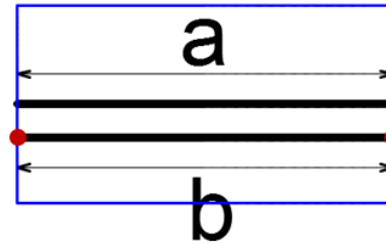
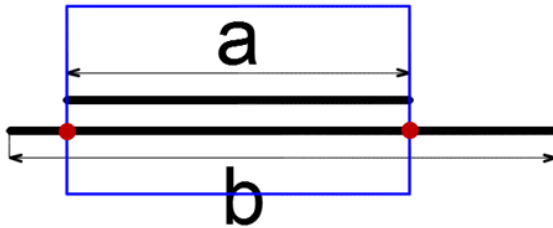
(2) Duplicates

Python package 'shapely' is used for basic geometric operations

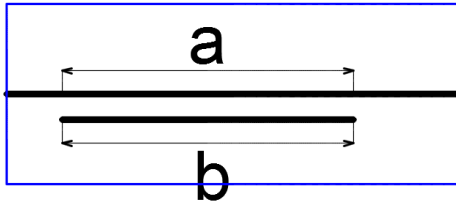
- ① 'Point' type — *a new line will be created to replace the old ones*



- ② 'MultiPoint' type — *only line b will be kept*

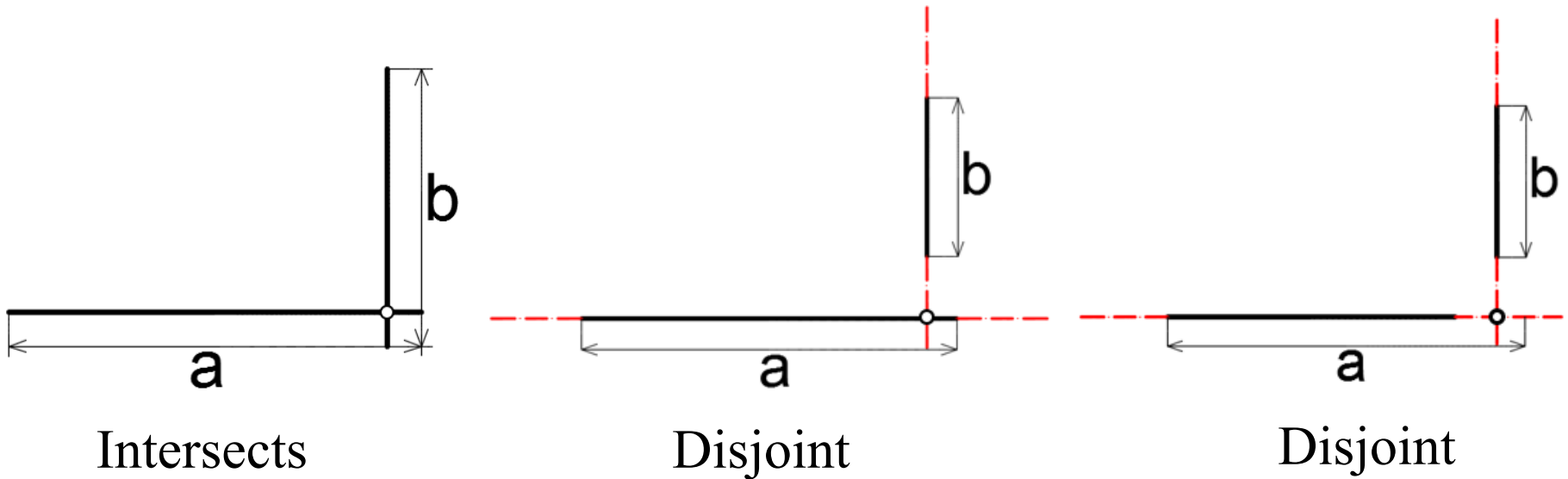


- ③ Null 'GeometryCollection' type — *only line a will be kept*



Preprocessing

(3) Disjoint vertices

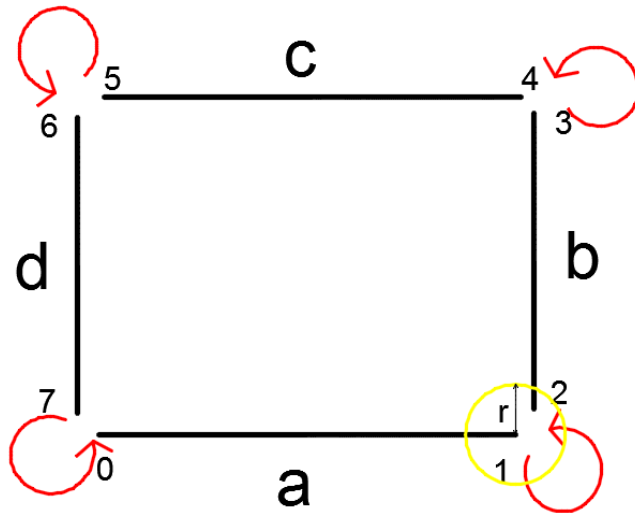


Preprocessing

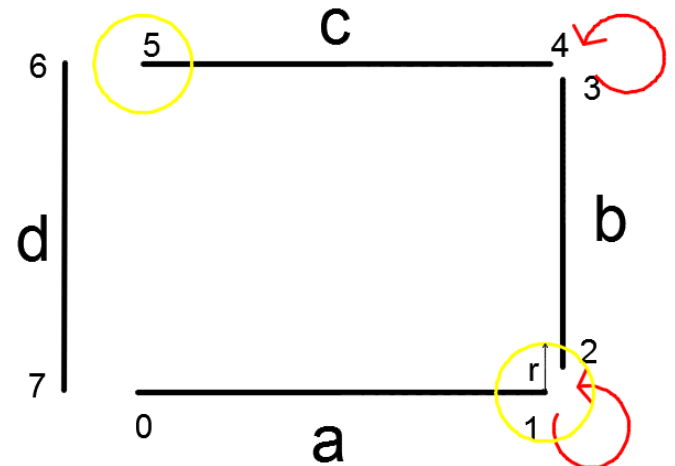
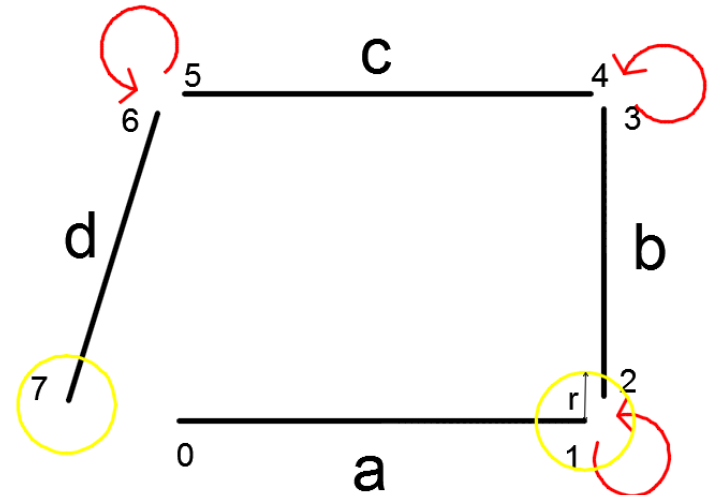
(3) Disjoint vertices

Line Grouping:

Lines belonging to a same polygon will be grouped together.



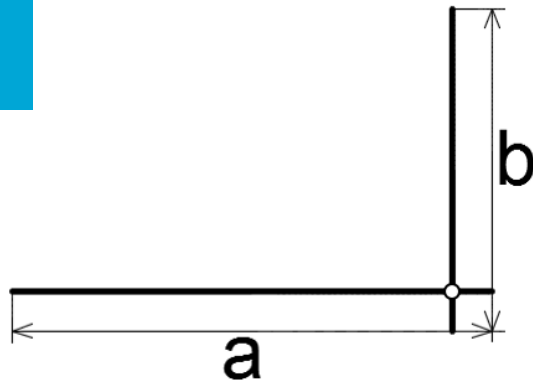
Closed chain



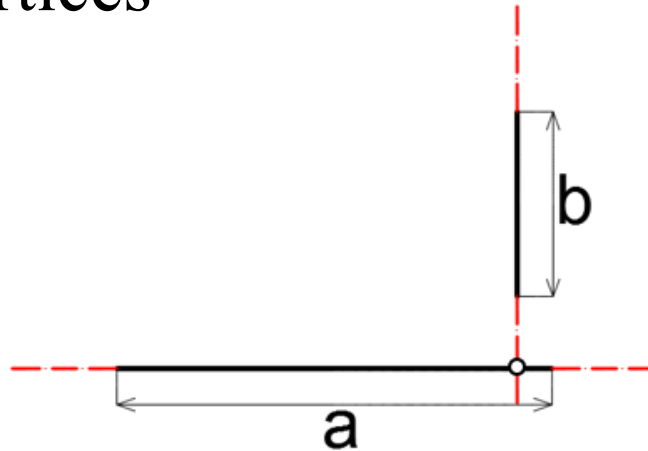
Break at the end or in the middle

Preprocessing

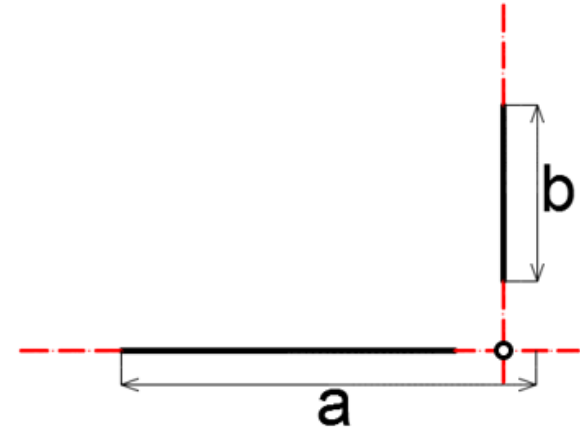
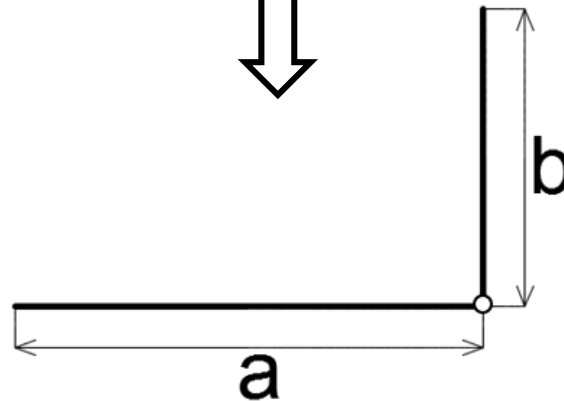
(3) Disjoint vertices



intersects



Disjoint



Disjoint



Wall polygons

- No wall detection needed
- Just make grouped and fixed lines into wall polygons
- To be merged with opening equivalent polygons created in next step

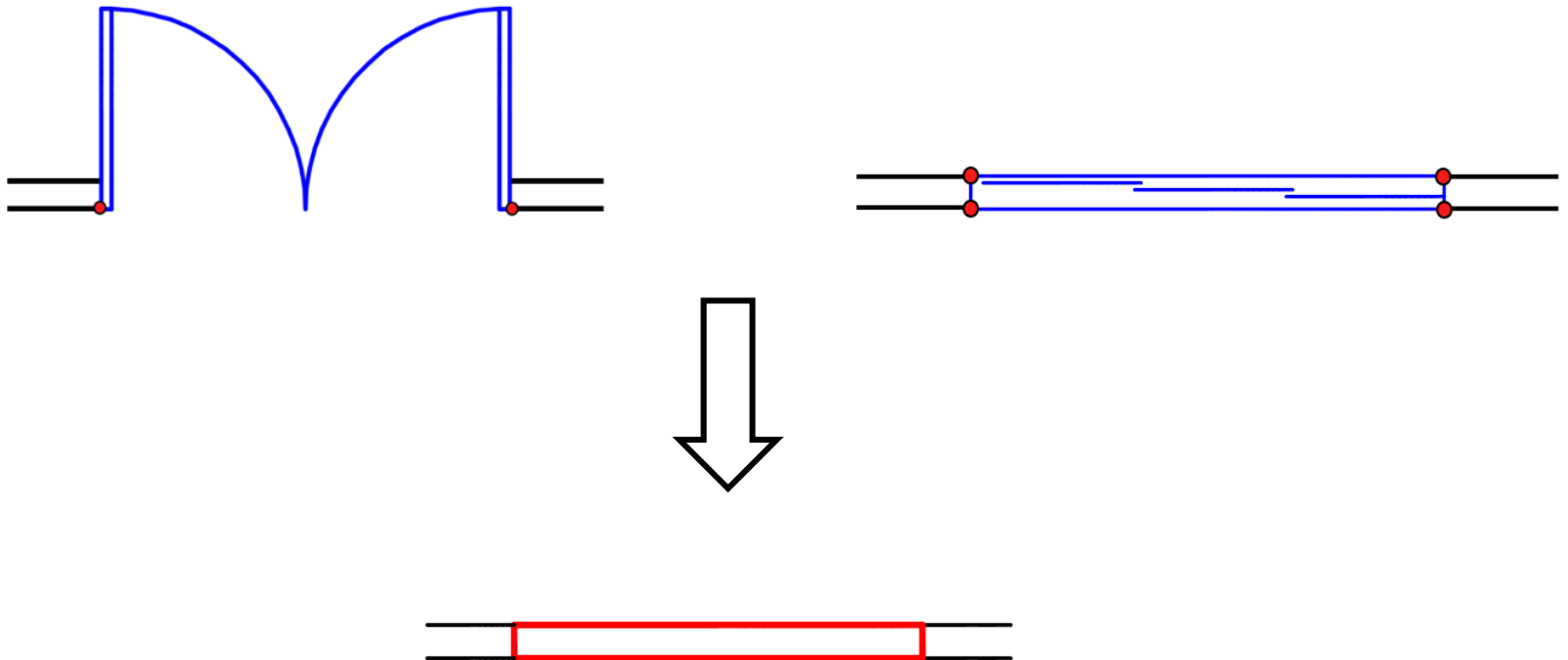


Opening equivalent polygons

- Our goal is to create opening equivalent polygons to replace the opening symbols in the floor plans
- After studying a set of floor plans, we found there are general three different placement of openings

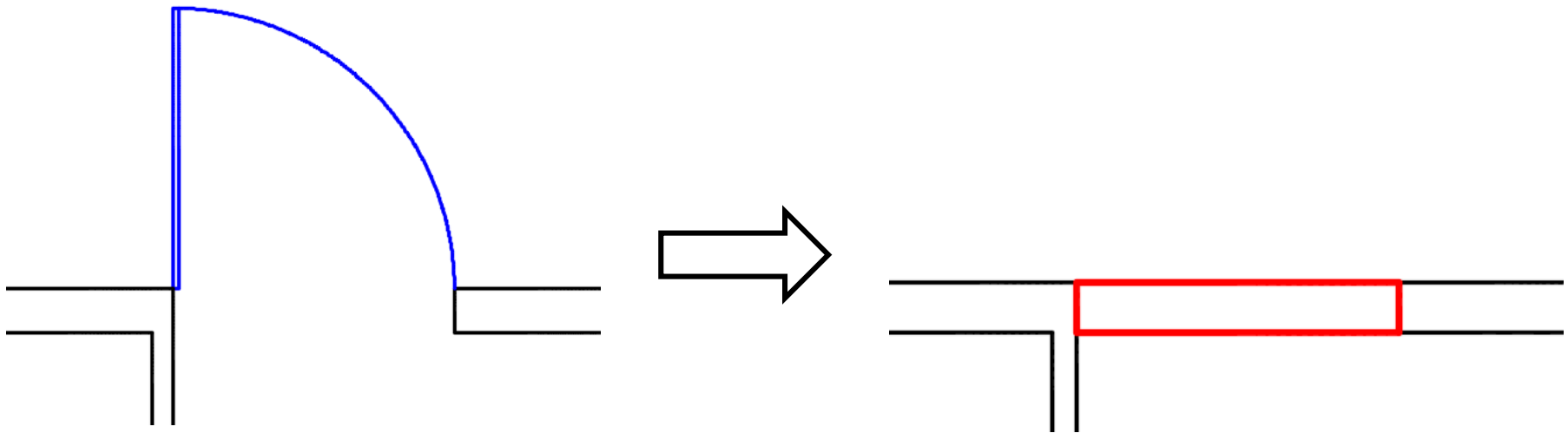
Opening equivalent polygons

Case 1: Two sides adjacent to end of wall



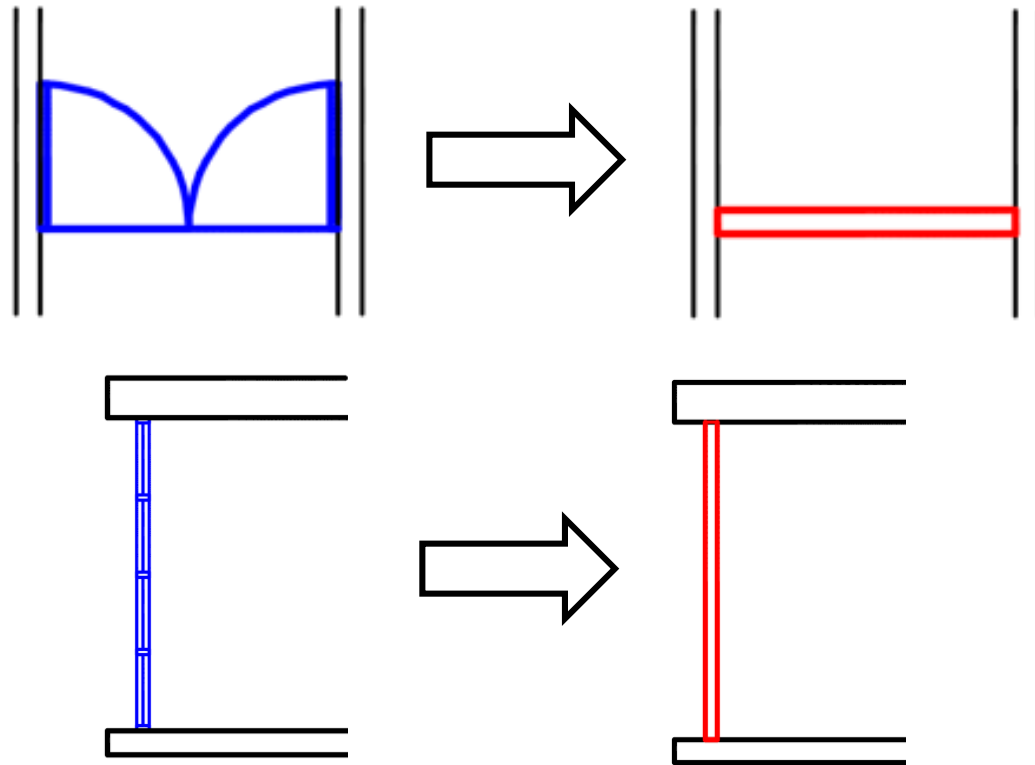
Opening equivalent polygons

Case 2: Only one side adjacent to end of wall



Opening equivalent polygons

Case 3: Openings perpendicularly exist in between two parallel walls



Opening equivalent polygons

- Constrain 2: openings symbols are saved as blocks.
- Primitives of each block are drawn in an independent coordinate system of their own
- Each block has a reference point in its local coordinate system
- All primitives in the block will be transferred from the original local coordinate system into the floor plan's coordinate system by translating, rotating and scaling, all with respect to the reference point

Opening equivalent polygons

- Each block is saved as an *insert* object in the floor plan
- There are four types of primitives that are most commonly used in blocks: line, polyline, arc, circle
 - Lines and polylines: most common and basic
 - Arcs: represent the trajectory of a hinged door or a casement window
 - Circles: represent the shaft of a door (indefinite)
- To calculate bounding box:
 - Traverse every primitives in block
 - Find minimal and maximal x and y coordinates
 - Lines and polylines: check every endpoints
 - Arcs: check center, start point and end point
 - Circles: ignore

Opening equivalent polygons

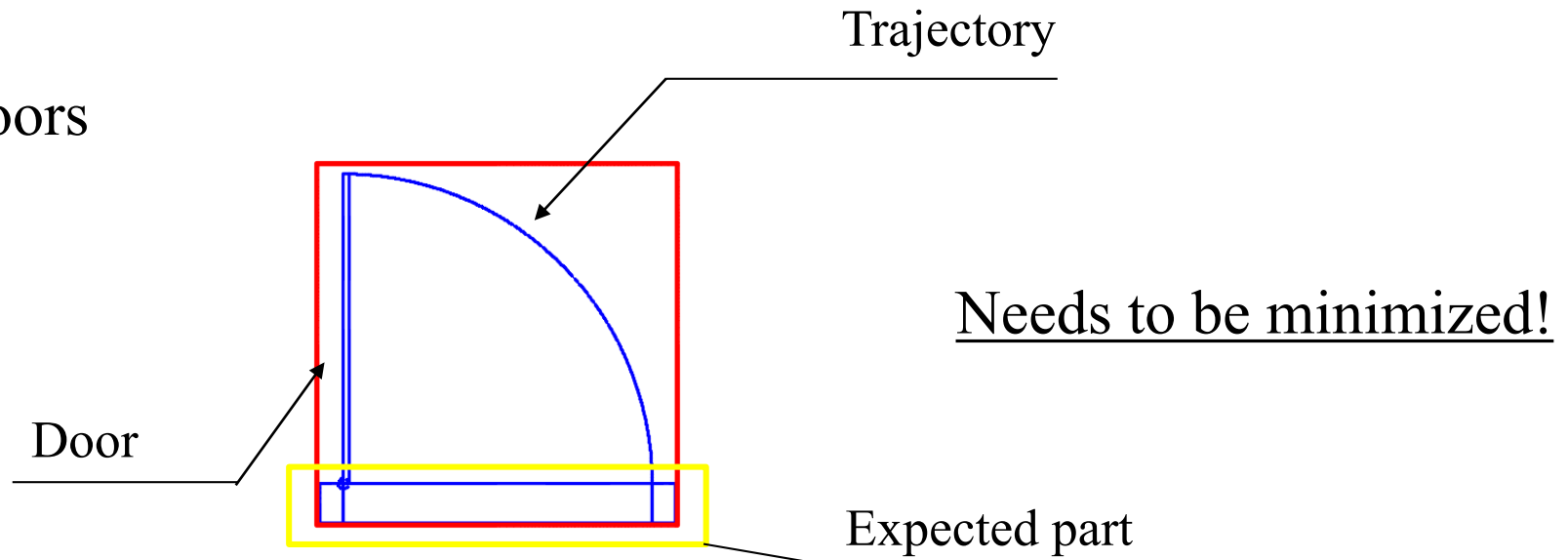
- Can we just use blocks' bounding box as their equivalent polygons?
- For windows,
 - Works fine
 - No trajectory in most window symbols
 - Clearly rectangular-shaped
- For doors,
 - Extra primitives representing the door and its trajectory in the block
 - Nearly square-shaped
 - Only part of the bounding box truly reflects the actual location of the opening

Opening equivalent polygons

- Windows



- Doors



Opening equivalent polygons

- Minimize the bounding box based on the location of the center of the trajectory arc
- First, the main direction of the bounding box has to be determined
 - Normally, opening is horizontally defined in its local system
 - If width $\geq a \cdot$ height, main direction is west-east
 - Else, main direction is north-south
 - a is set to be 0.8%
- Then, according to the center's location in the bounding box, its minimized bounding box (MBB) is different (given an overall opening thickness t)

Opening equivalent polygons

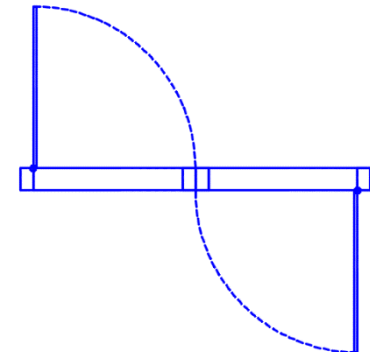
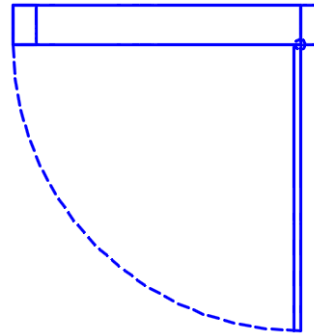
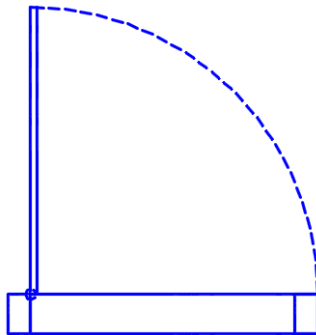
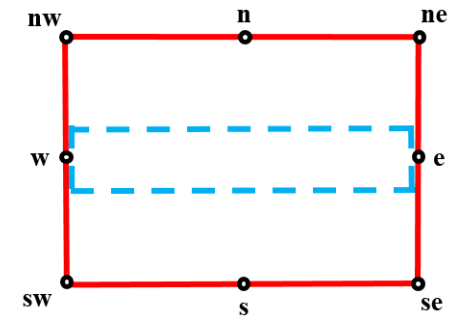
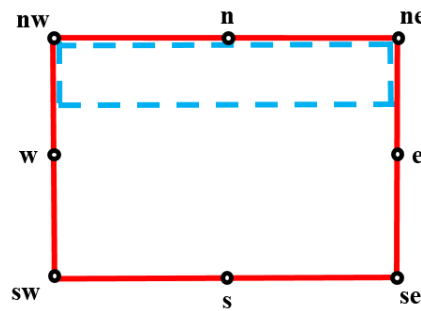
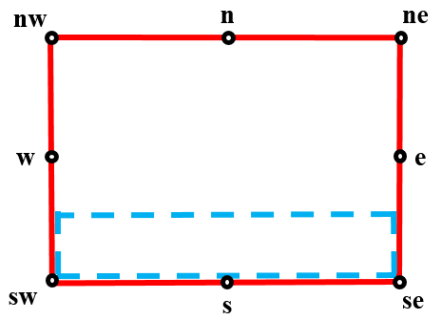
West-east direction

$$y_{max} = y_{min} + t$$

$$y_{min} = y_{max} - t$$

$$y_{max} = (y_{max} + y_{min})/2 + t/2$$

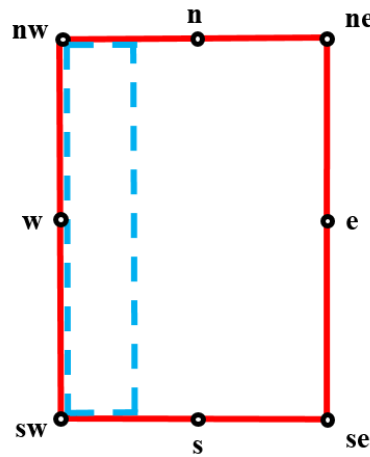
$$y_{min} = (y_{max} + y_{min})/2 - t/2$$



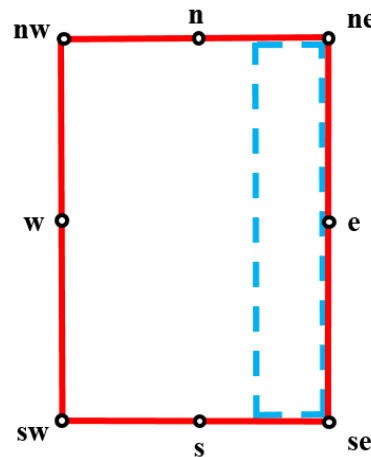
Opening equivalent polygons

North-south direction

$$x_{max} = x_{min} + t$$

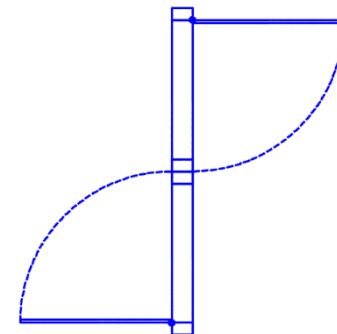
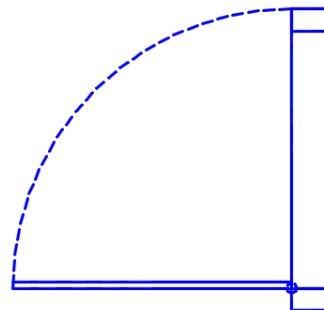
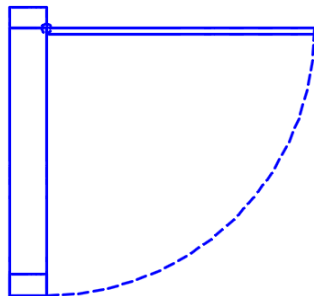
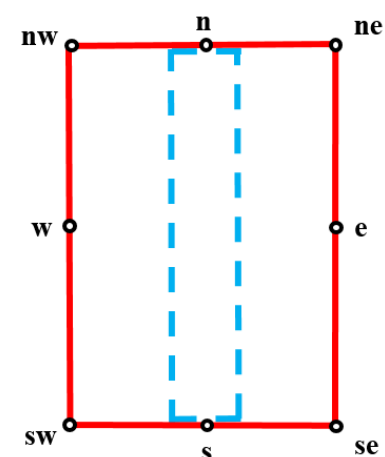


$$x_{min} = x_{max} - t$$



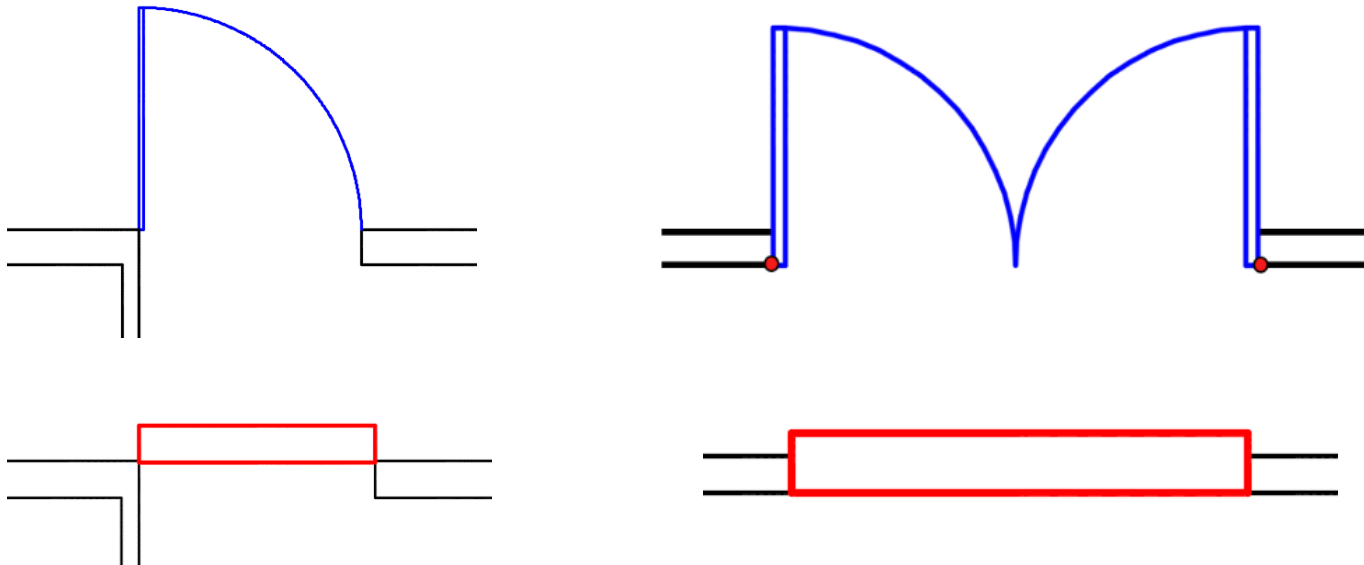
$$x_{max} = (x_{max} + x_{min}) / 2 + t / 2$$

$$x_{min} = (x_{max} + x_{min}) / 2 - t / 2$$



Opening equivalent polygons

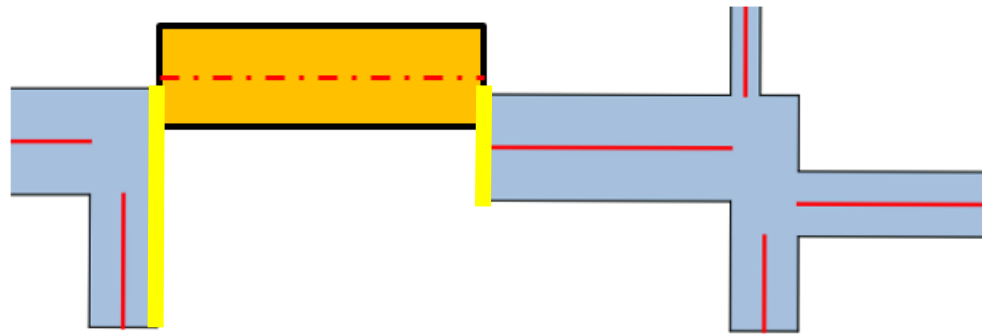
- MBB still doesn't equal to equivalent polygon, because



- Thus, needs to be further elaborated

Opening equivalent polygons

- Find two anchorage lines for each MBB on both sides
 - Anchorage lines: line intersects with MBB and perpendicular to the main direction of the MBB



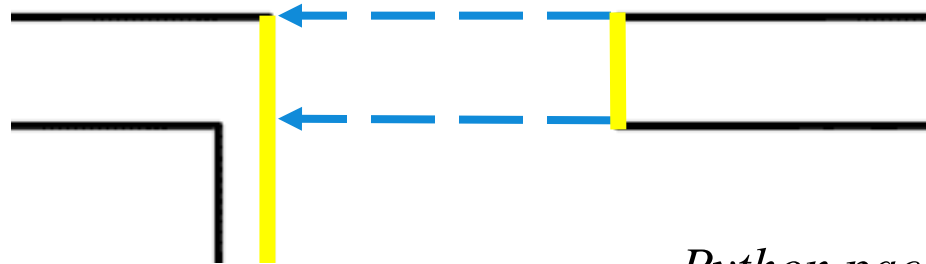
- Create opening equivalent polygons according to different cases
- Store every opening equivalent polygon's type (door or window), thickness and central point into DB

Opening equivalent polygons

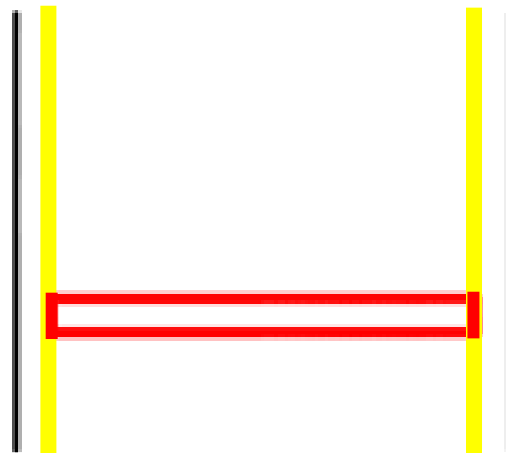
Case 1



Case 2



Case 3

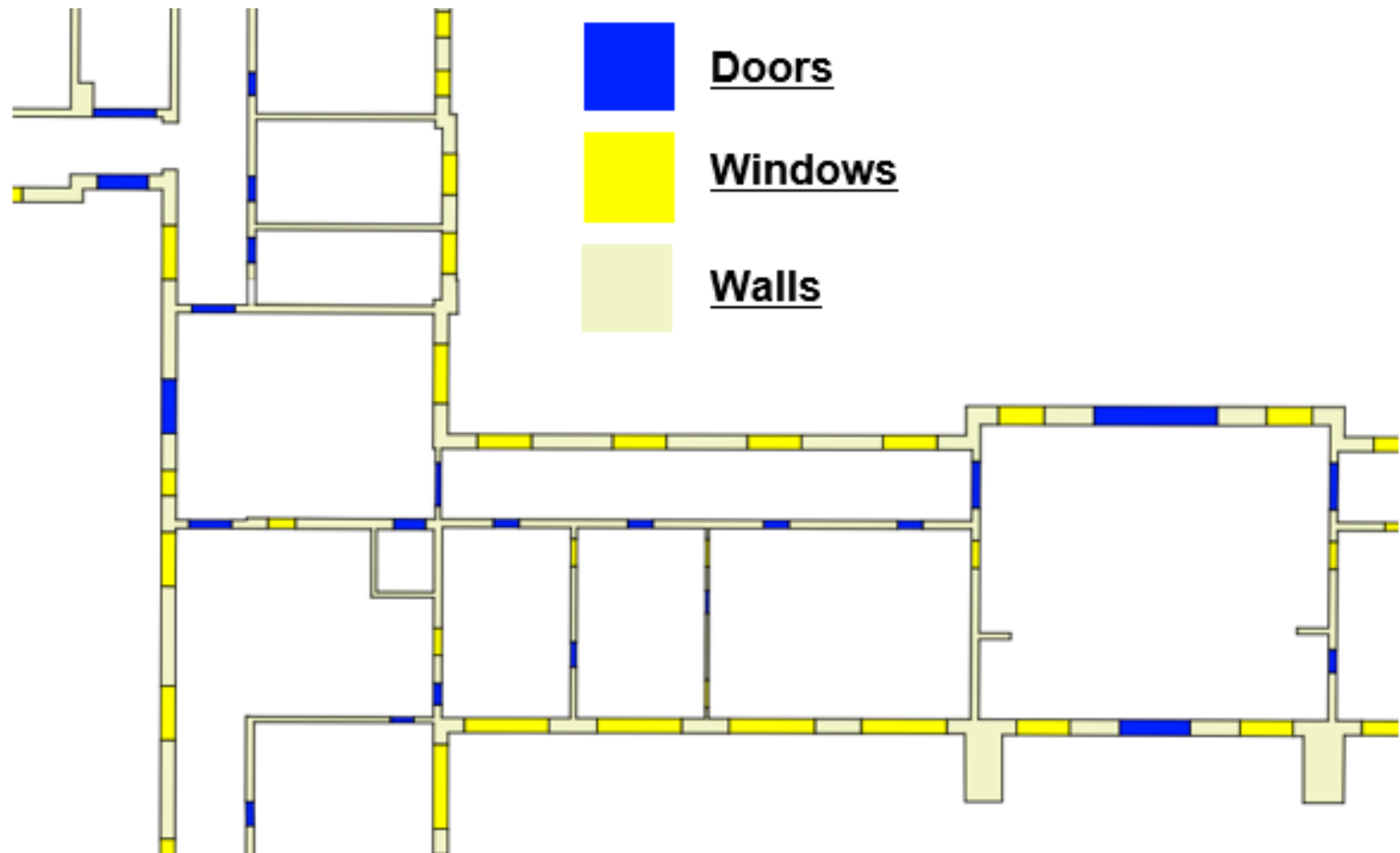


Python package 'dxfgrabber' is used for reading data from DXF file

Python package 'shapely' is used for basic geometric operations

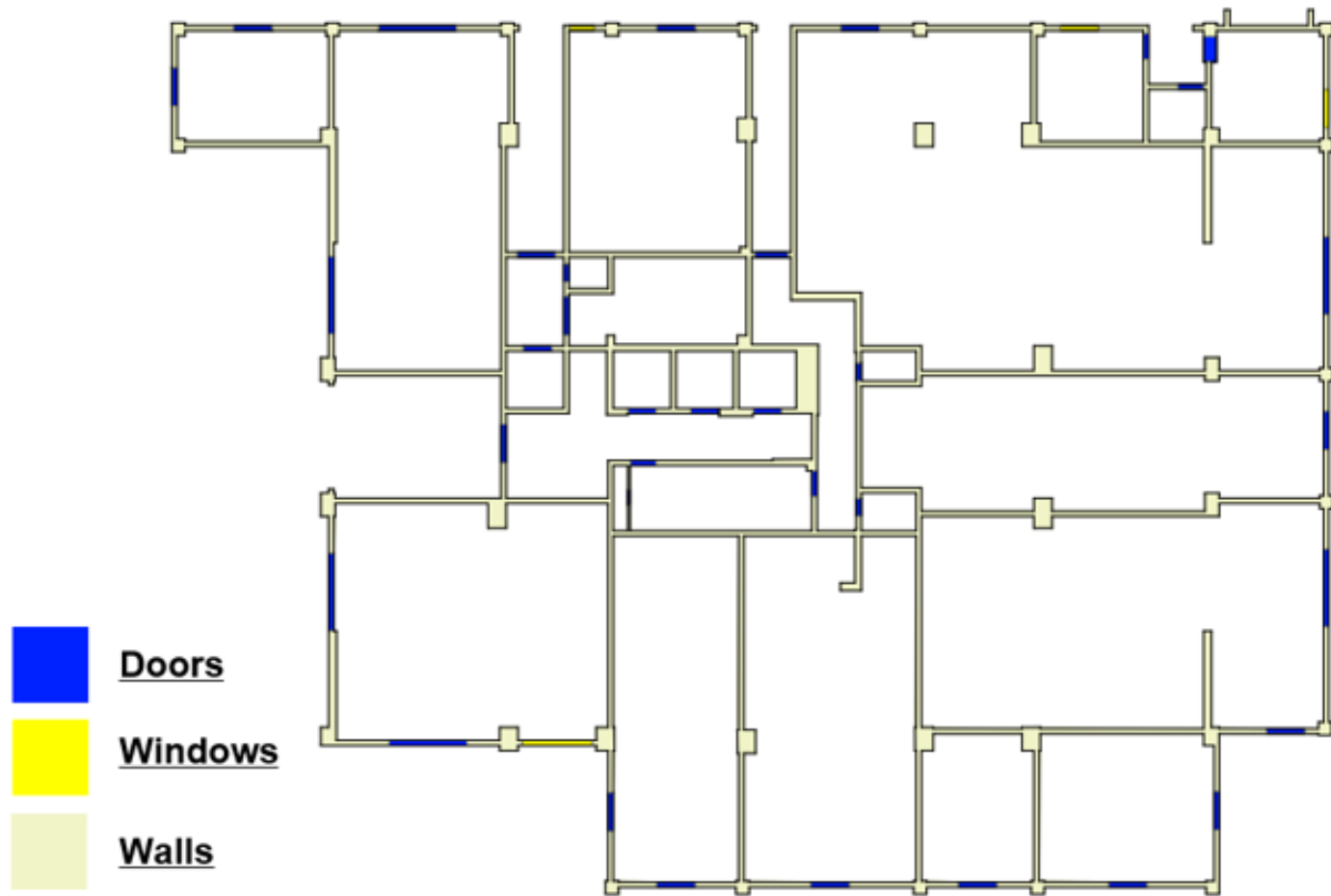
Opening equivalent polygons

Test floor plan 1



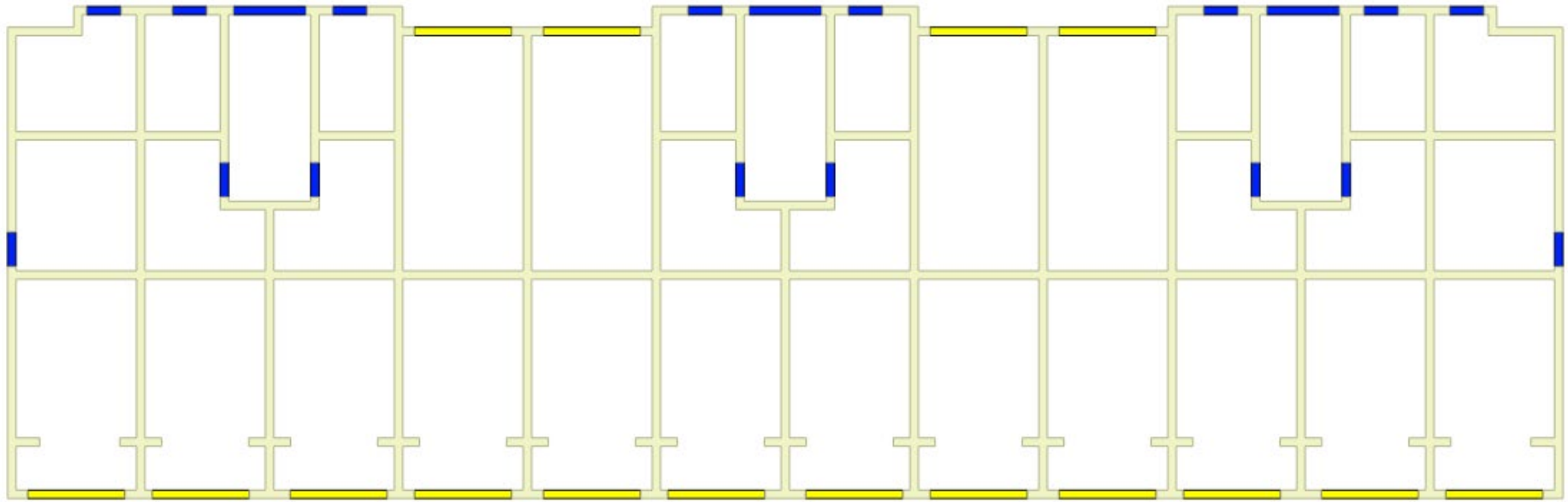
Opening equivalent polygons

Test floor plan 2



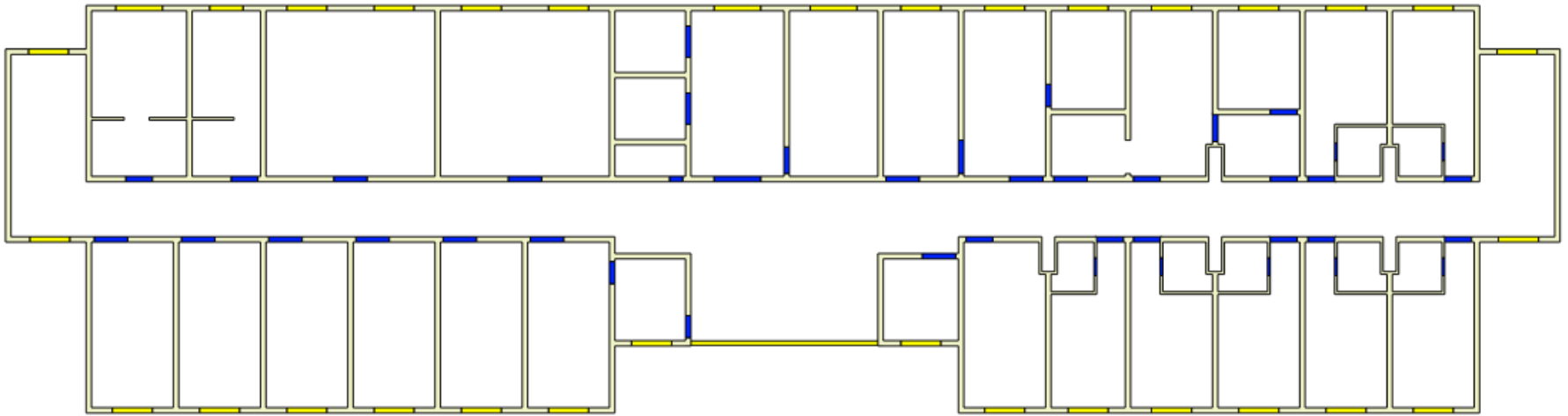
Opening equivalent polygons

Test floor plan 3



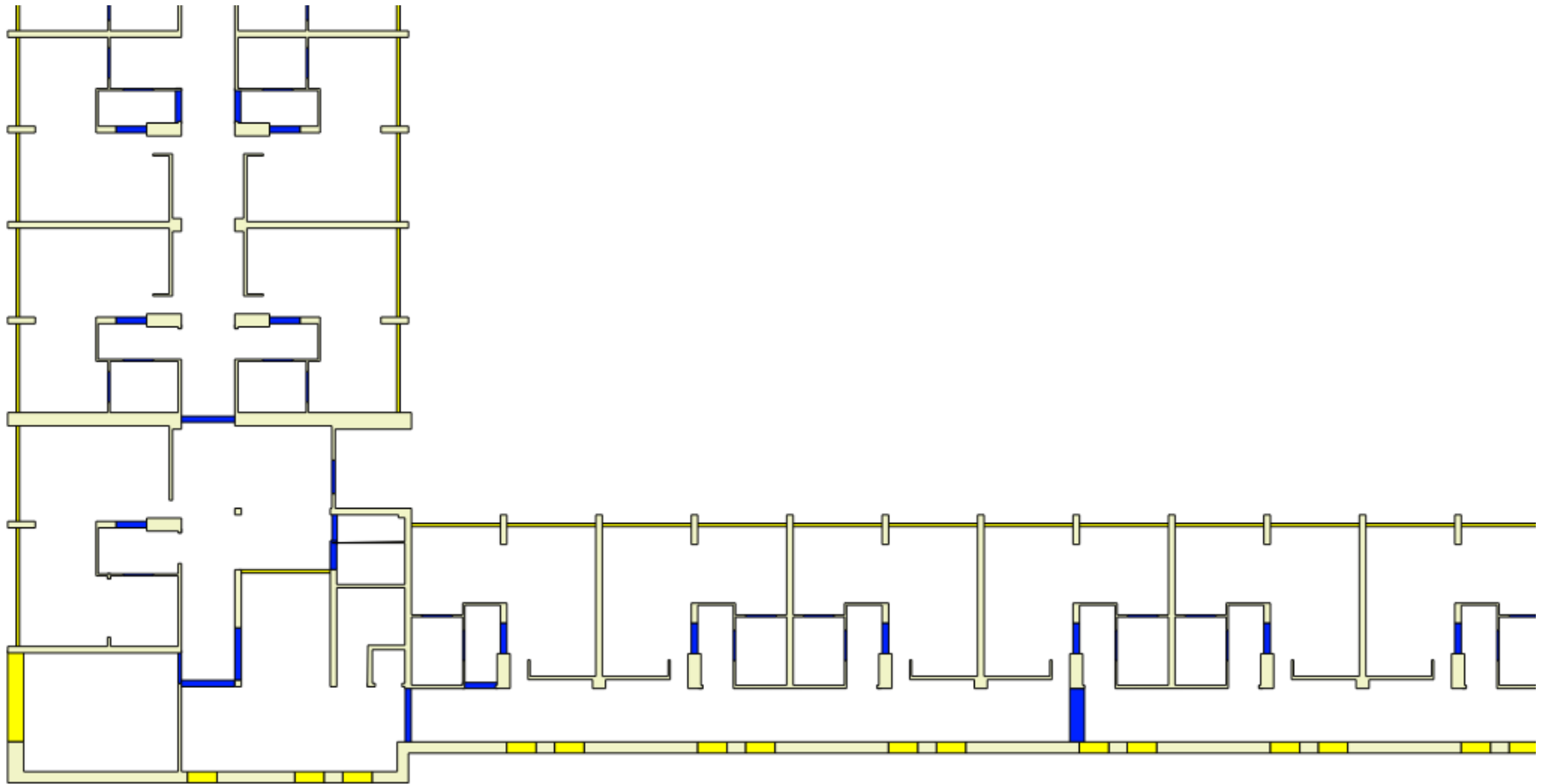
Opening equivalent polygons

Test floor plan 4



Opening equivalent polygons

Test floor plan 5



Merge polygons & loops extrusion

- To merge all polygons together
- Use function `.union()` in python package '*shapely*'
- No need for loop searching
- Loops can be easily retrieved from the merged polygon
 - Exterior boundary --- loops of level shell
 - Inner rings --- loops of rooms and corridors
- Extracted loops are stored into DB for extrusion

Results (1)

Preprocessing

Test floor plan	Null-length	Lines before cleaning	Contained/ Identical	Contains	Consecutive/ Overlapped	Lines after cleaning
1	25	2243	0	0	4	2230
2	3	627	0	0	13	609
3	1	422	0	0	10	408
4	0	542	0	0	0	542
5	1	1946	0	0	0	1946

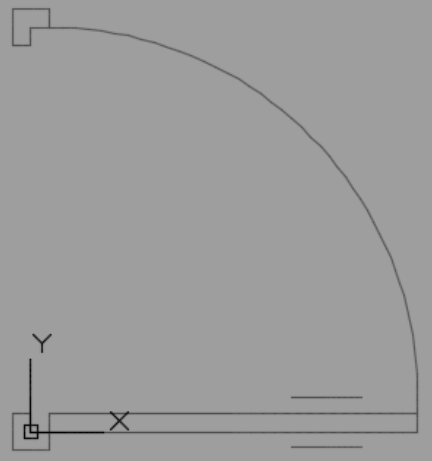
Results (2)

Line grouping

Test floor plan	r	Iterations	r_{max}	Found groups
1	5mm	3	15mm	331
2	5mm	1	5mm	14
3	5mm	1	5mm	1
4	5mm	1	5mm	38
5	5mm	9	45mm	207

Results (3)

Opening equivalent polygons

Test floor plan	a	t	Opening type	Total	Missing	Mistakes	Completeness	Correctness
1	0.8	0.65m	Doors		0	0	98.5%	100%
			Window			0	99.3%	100%
2	0.8	0.2m	Doors			3	85.7%	90.0%
			Window			0	100%	100%
3	0.8	0.25m	Doors			8	100%	57.8%
			Window			0	100%	100%
4	0.8	0.25m	Doors			0	100%	100%
			Window			0	100%	100%
5	0.8	0.2m	Doors			5	100%	97.3%
			Windows	125	0	2	100%	98.4

Test floor plan	a	t	Opening type	Total	Missing	Mistakes	Completeness	Correctness
3	1	0.25m	Doors	19	0	0	100%	100%
			Windows	16	0	0	100%	100%

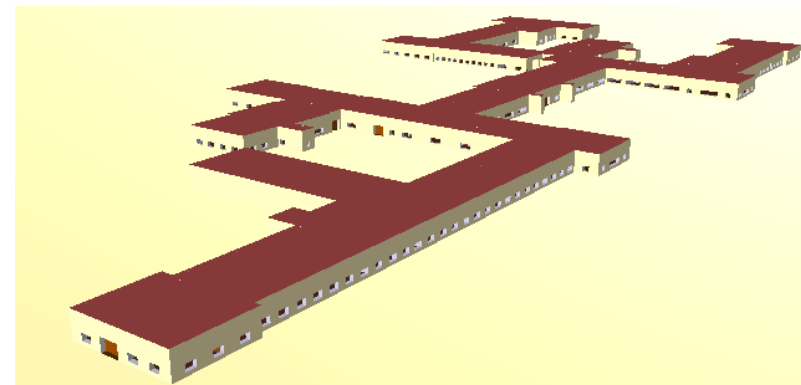
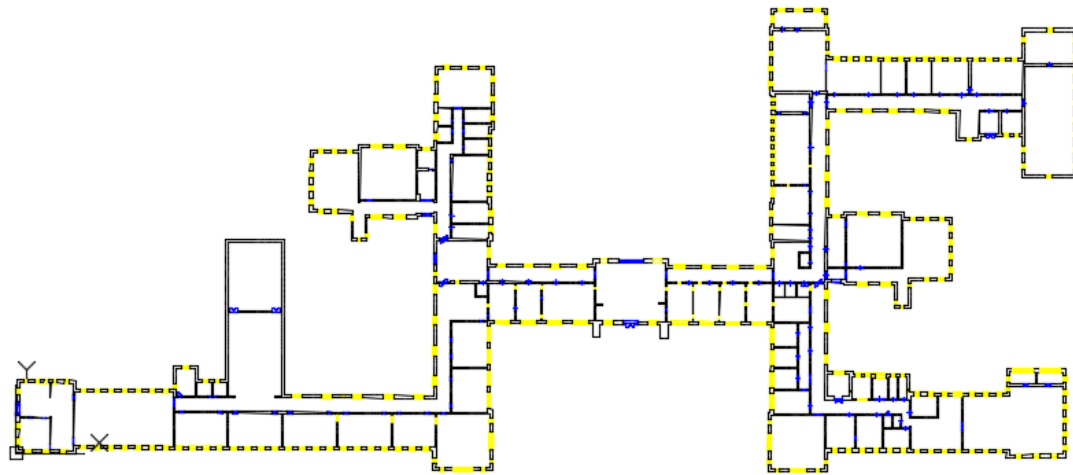
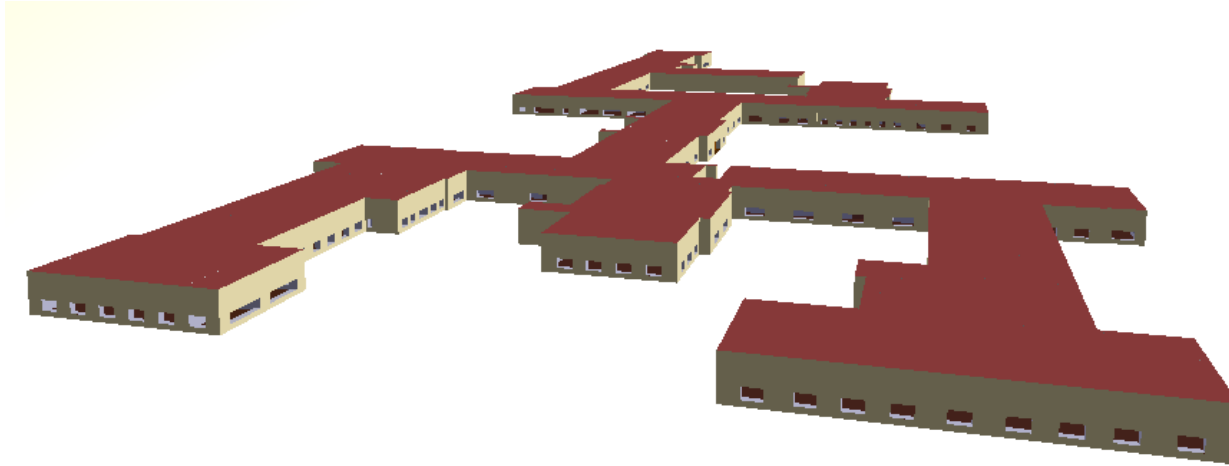
Results (4)

Loops retrieval

Test floor plan	<i>Loops</i>
1	85
2	20
3	36
4	40
5	70

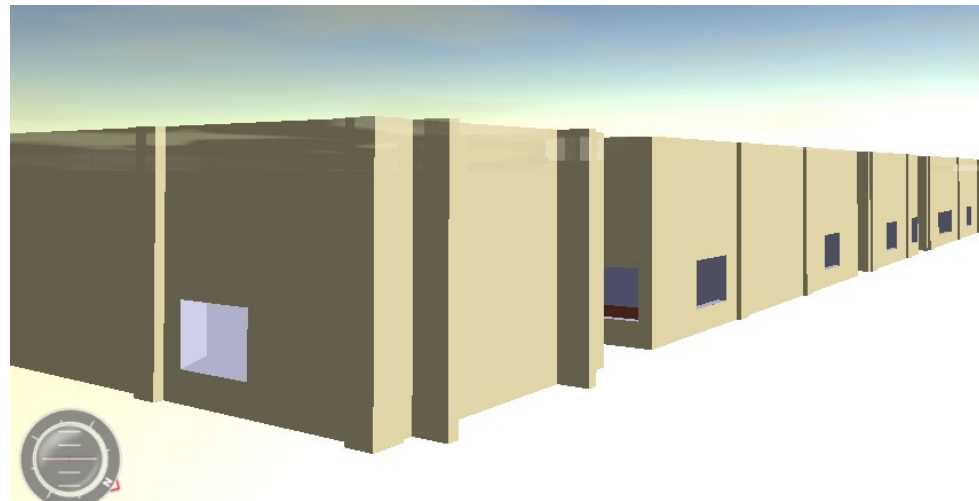
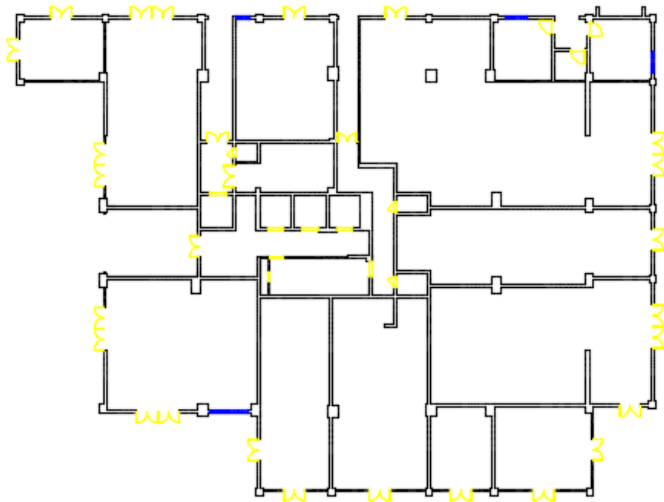
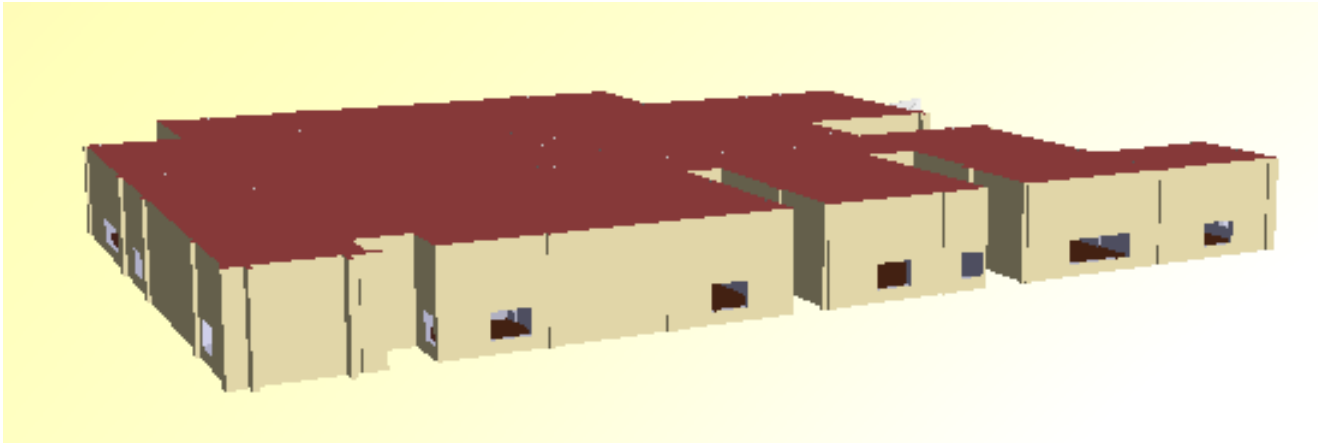
Results (5)

Reconstructed 3D models (1)



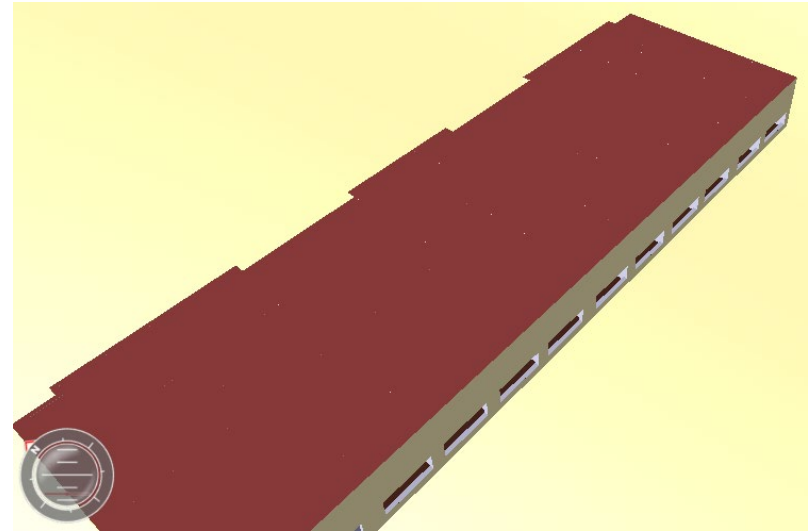
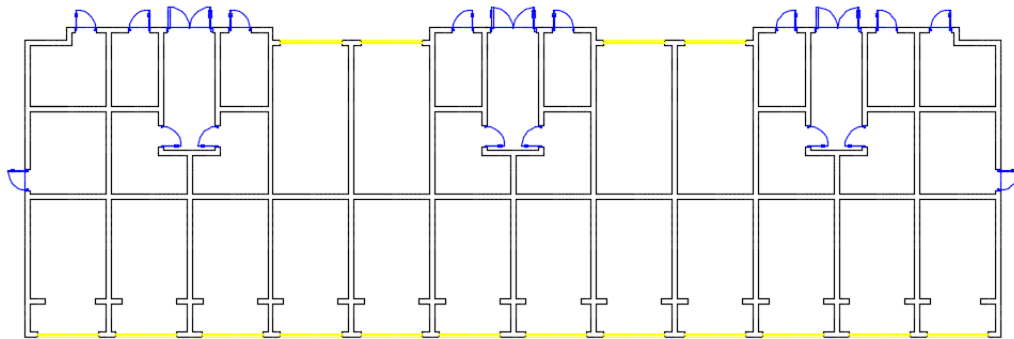
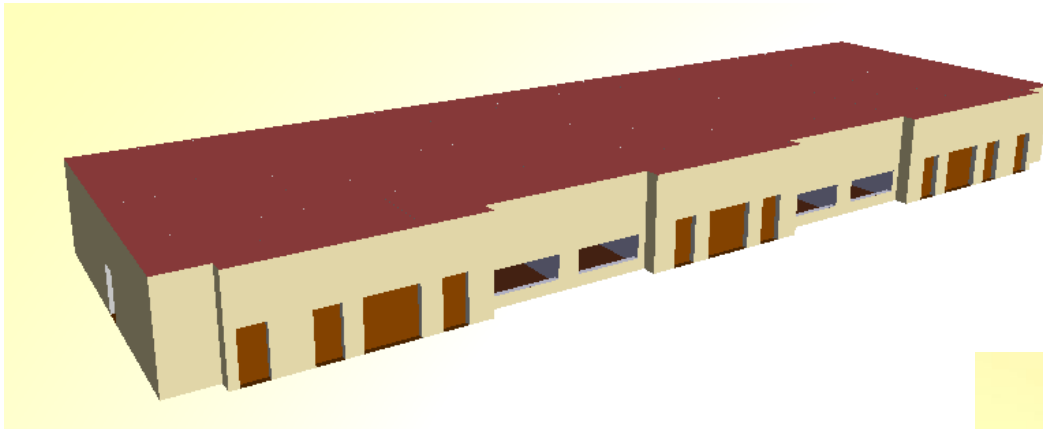
Results (5)

Reconstructed 3D models (2)



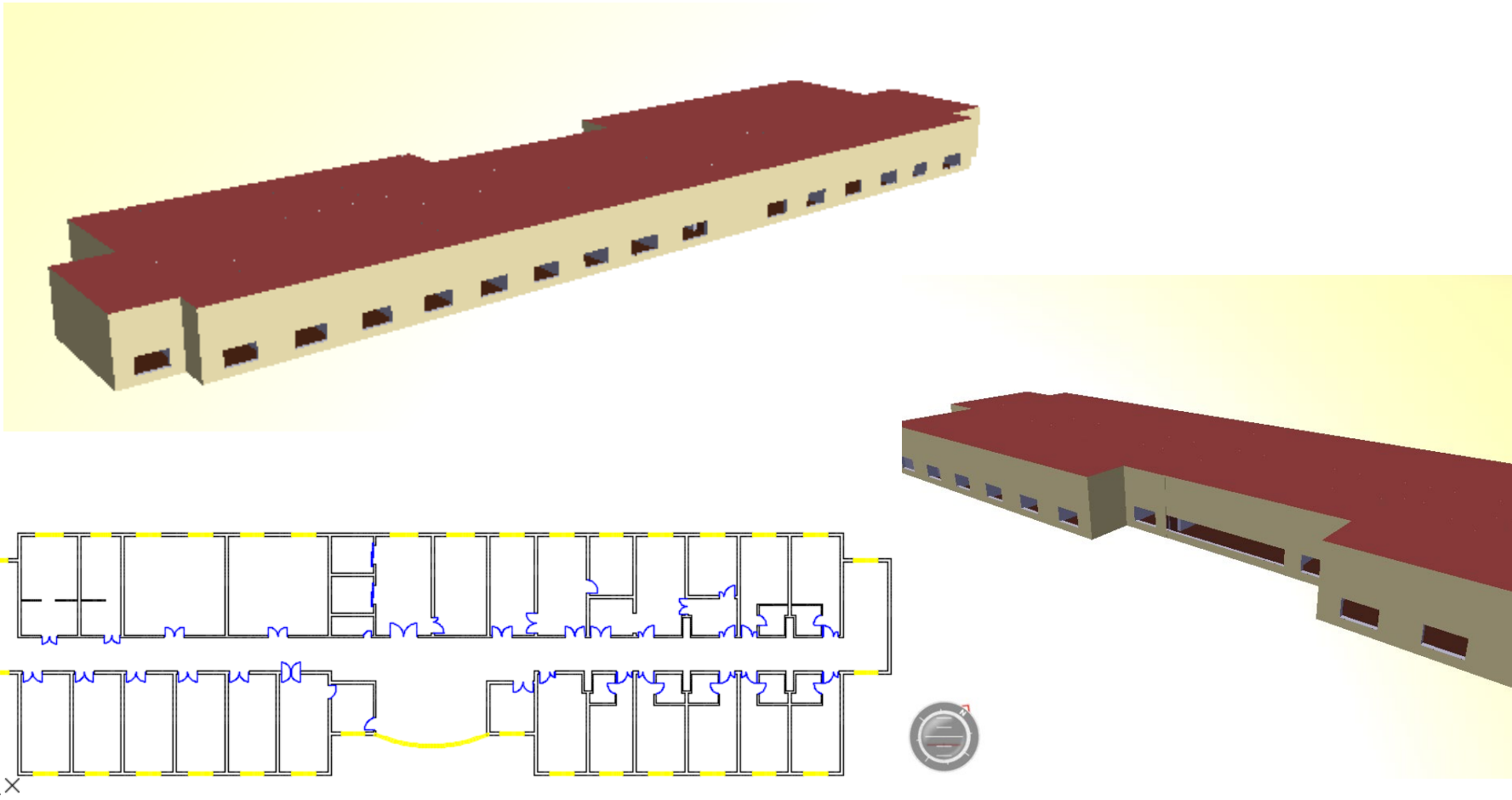
Results (5)

Reconstructed 3D models (3)



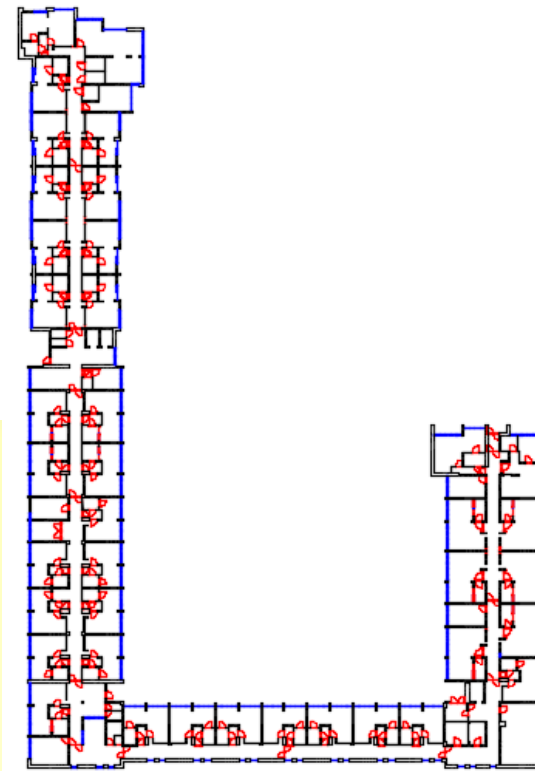
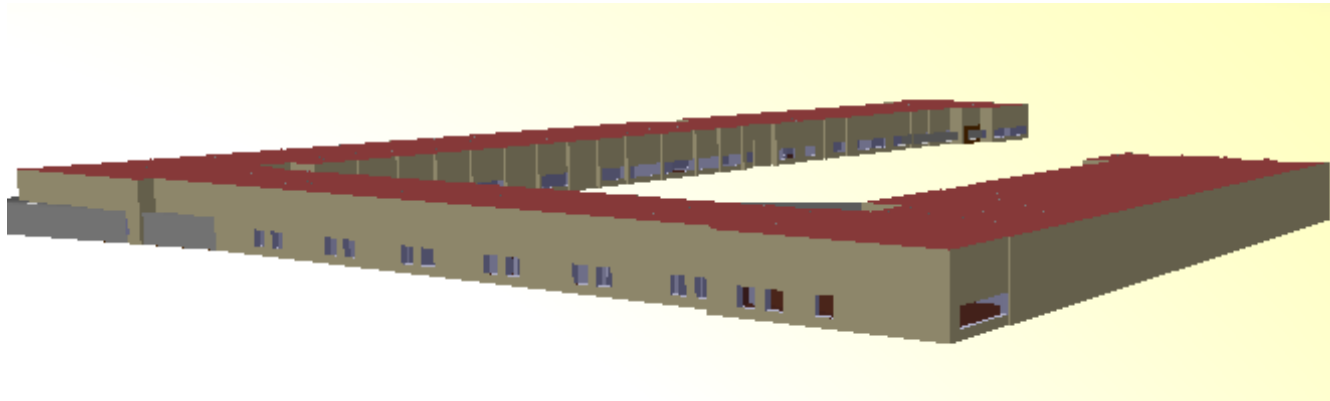
Results (5)

Reconstructed 3D models (4)



Results (5)

Reconstructed 3D models (5)



Conclusions

- 2D CAD architectural floor plans are a very promising data source for 3D reconstruction.
- Ambiguities and inconsistencies in real-life floor plans are the main obstructions for an automatic reconstruction process
- At present it is still very hard to fully automatically realize this with a raw floor plan from real life. Some trade-offs have to be made between designers of floor plans and the users of 3D models, or between the preprocessing and the reconstruction.
- Use polygons as basic elements to retrieve loops can save the work of wall detection and loop searching

Future work

- (1) Only deals with content of walls and openings. Models reconstructed in this way contain very limited semantics. How to include more information in the floor plans into the reconstruction process to enrich the semantics in the 3D models needs to be further studied.
- (2) Algorithms proposed in this thesis have multiple thresholds, which need to be provided by the user based on the specific scenario of a given floor plan .
- (3) Thus human intervention is needed to make the floor plans conform with the constraints, which is less interesting. More work needs to be invested into the automatic preprocessing of raw floor plans later.
- (4) Only addresses reconstruction problem for a single floor. Algorithm to automatically position each floor is required.

References

- [1] "Floor plan." [Online]. Available: http://en.wikipedia.org/wiki/Floor_plan. [Accessed: 18-Apr-2015].
- [2] Young S. Lee, "Interior Design Student Hand, Part 2: Basic Drafting Standards and Symbols", University of Minnesota, 2005.
- [3] United States National CAD standard, version 5, May 2011.
<http://www.nationalcadstandard.org/ncs5/content.php>
- [4] AEC (UK) Basic Layer Naming Handbook, version 3.0.1, August 2011.
<http://www.aec-uk.org>
- [5] ISO 13567: Organization and Naming of Layers for CAD, 1998. <http://www.iso.org>
- [6] T. Lu, H. Yang, R. Yang, and S. Cai, "Automatic analysis and integration of architectural drawings," *Int. J. Doc. Anal. Recognit.*, vol. 9, pp. 31–47, 2007.
- [7] J. Park and Y. Kwon, "A Main Wall Recognition of Architectural Drawings using Dimension Extension Line," *KIPS Trans.*, vol. 10B, pp. 837–846, 2003.
- [8] B. Domínguez, Á. L. García, and F. R. Feito, "Semiautomatic detection of floor topology from CAD architectural drawings," *Comput. Des.*, vol. 44, no. 5, pp. 367–378, May 2012.
- [9] R. Lewis and C. Séquin, "Generation of 3D building models from 2D architectural plans," *Computer-Aided Design*, vol. 30, no. 10, pp. 765–779, 1998.
- [10] J. Zhu, H. Zhang, and Y. Wen, "A new reconstruction method for 3D buildings from 2D vector floor plan," *Prog. Locat. Serv.*, vol. 2013, pp. 1–14, 2013.

References

- [11] Autodesk, “DXF Reference,” AutoCAD 2008, no. February 2010, p. 306, 2011.
- [12] "AutoCAD DXF." [Online]. Available: http://en.wikipedia.org/wiki/AutoCAD_DXF. [Accessed: 18-Apr-2015].
- [13] ".dwg." [Online]. Available: <http://en.wikipedia.org/wiki/.dwg>. [Accessed: 18-Apr-2015].
- [14] "About DWG." [Online]. Available: <http://www.autodesk.com/products/dwg>. [Accessed: 18-Apr-2015].
- [15] T. Guo, H. Zhang, and Y. Wen, “An improved example-driven symbol recognition approach in engineering drawings,” Comput. Graph., vol. 36, no. 7, pp. 835–845, Nov. 2012.
- [16] X. Yin, P. Wonka, and A. Razdan, “Generating 3D building models from architectural drawings: a survey.,” IEEE Comput. Graph. Appl., vol. 29, pp. 20–30, 2009.
- [17] Spence, W.P., "Architectural working drawings", NJ: Prentice Hall, 1993.
- [18] Ankerson, K.S., "Interior construction document", NY: Fairchild Books & Visuals, 2003.
- [19] Kilmer, W.O, & Kilmer, R., "Construction drawings and details for interiors: Basic skills", NJ: John Wiley & Sons, 2003.