

User manual of SAT2LoD2 software

This document is about the running of SAT2LoD2 (executable SAT2LoD2.exe), which is an open-source building LoD-2 reconstruction software corresponding to the paper “Automated LoD-2 model reconstruction from very-high-resolution satellite-derived digital surface model and orthophoto” (ISPRS Journal of Photogrammetry and Remote Sensing, 2021).

Requirement

A GPU with CUDA driver is required for user, if user wants to use building detection and classification function.

Input data

Launching model3d.exe to start the building LoD-2 reconstruction workflow. EXE file is in `./software/SAT2LoD2_CU10/SAT2LoD2.exe`. An example of input data is in `./software/example/input/` folder.

The main window is shown as following figure. If the input data are fulfilled, click on ‘OK’ button to start processing.

The screenshot shows the main window of the SAT2LoD2 software. The window has a title bar with the text "SAT2LOD2" and standard window controls (minimize, maximize, close). The main content area is divided into two sections. The "Input data" section contains four rows of input fields, each with a "Select" button: "Orthophoto", "DSM", "Input user's classification image (optional)" (with a radio button), and "Use OSM file (optional)" (with a radio button). The "Reconstruct parameter" section contains four rows of sliders: "Length threshold T_l (pixel) [45-150]" (set to 90), "Color difference threshold T_d (RGB) [6-20]" (set to 10), "Mean height difference threshold T_h1 (m) [0.5-1.5]" (set to 0.5), and "Gap threshold T_h2 (m) [0.1-0.3]" (set to 0.10). At the bottom of the window, there is an "Output folder" text box with a "Select" button, a "Contact Info" button, an "OK" button, and a "Cancel" button.

Figure 1. Main window of SAT2LoD2 software

Required input data include: 1) *orthophoto* with RGB band (.png, .jpg, or .tif format, 3 bands, 8 bit image); and 2) *DSM* (digital surface model, .tif format, 1 band, 16 bit int or 32 bit single image), and the orthophoto and DSM should have the same row and column. The maximum size of *orthophoto* and *DSM* is 5000 * 5000 for generating mesh in a few minutes.

Optional input data include: 1) *classification image*, building class is labeled as 1 or 255 (.png, .jpg, or .tif format with 8 bit single band, shown in Figure 2, other types of objects need to be set as 0, and other types of ground objects need to be set as zeros value); and 2) *OpenStreetMap shapefile* for street lines (.shp format, shown in Figure 3), and the corresponding .tfw file, which has the same name as *orthophoto*, and the projection of both *OSM shape file* and .tfw file need to be WGS84 UTM Zone.



Figure 2. Sample of building classification image



Figure 3. Sample of OpenStreetMap shape file

Building reconstruction parameters include: (please reference to the paper)

Parameter	Description	Range
Length threshold T_l (pixel)	A threshold of summed up length for determining building main orientations.	[45, 150]
Color difference threshold T_d (RGB)	A threshold of mean color differences ($ \overline{C_1} - \overline{C_2} $) of the two rectangles to decide whether to merge two nearby rectangles in building decomposition.	[6, 20]
Mean height difference threshold T_{h1} (m)	A threshold of mean height difference ($ \overline{H_1} - \overline{H_2} $) between two nearby rectangles to decide whether to merge two nearby rectangles in building decomposition.	[0.5, 1.5]
Gap threshold T_{h2} (m)	A threshold of dramatic height changes in a buffered region that cover the common edge between two nearby rectangles between two nearby rectangles.	[0.1, 0.3]

Output folder is the folder path that LoD-2 building model and all other intermediate results will be stored. If this path is empty, the output file will be generated to the path of .exe file.

Building reconstruction workflow

There are six main steps in building LoD-2 reconstruction software:

- 1) **Building detection and segmentation:** generate the building segments from input orthophoto, if the input user's classification image ratio button is not selected. We adapt HRNet V2 ([HRNet/HRNet-Semantic-Segmentation: The OCR approach is rephrased as Segmentation Transformer: https://arxiv.org/abs/1909.11065](https://arxiv.org/abs/1909.11065). This is an official implementation of semantic segmentation for HRNet. [https://arxiv.org/abs/1908.07919 \(github.com\)](https://arxiv.org/abs/1908.07919)) as the building segmentation algorithm, for extracting building segments with high accuracy and efficiency.
- 2) **Initial 2D building polygon extraction:** generate the building polygon (boundary) of each building segment, by following initial line extraction, line adjustment, and line refinement.
- 3) **Building rectangle decomposition:** generate the building rectangle base on the building polygon, by using a grid-based decomposition algorithm.
- 4) **Building rectangle orientation refinement:** refine the orientation of each building rectangle by combining OpenStreetMap line segments.
- 5) **3D model fitting:** fit and reconstruct the roof structure based on each building rectangle.
- 6) **Mesh generation:** transform building model into an .obj format. For irregular building (low IoU value with mask), directly generate mesh from DSM.

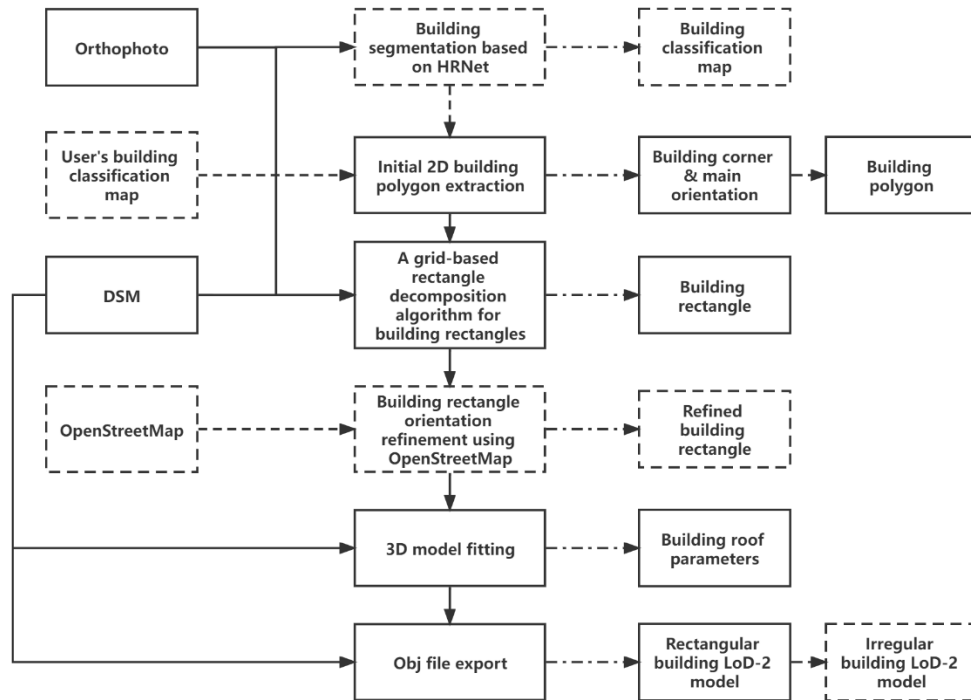


Figure 4. Workflow of the software

Output:

All output files will be at the **output folder** as selected in Main Window.

Patch folder: This folder includes the images and annotations for input orthophoto, which will be clipped as 512*512 images with 50% overlap.

class.png: Classification map for buildings, derived from orthophoto, by using HRNet as building segmentation method.

building_corner.json: The polygon nodes of each building, the data format of each node is:

```
[25, 1201.4596385681623, 483.15666353691796]
```

[number of segment, x, y].

building_polygon.json: The polygon lines of each building, the data format of each line is:

```
[89, 1071.6061133527417, 1074.5116773823315, 871.3054306969316, 840.2762669513455, 1, -1.47742880765086, 26.73013280924732]
```

[number of segment, x1, x2, y1, y2, number of line in a segment, orientation, length of line].

Main_ori.npy: The array of main orientation for each building segment.

boundary.png: Visualized building polygon (boundary) based on the orthophoto.

building_rectangle.json: The building rectangles of each building segment, the data format of each rectangle is:

```
[[16, 1, 169.28524627757633, 413.49085043401755, 178.90431610496617, 392.59890232820896, 189.26884881356713, 422.69169983412957, 198.88791864095697, 401.799751728321, 506, 0.41012734054149097]]
```

[number of segment, number of rectangle in a segment, x1, y1, x2, y2, x3, y3, x4, y4, orientation]

rectangle.png: Visualized building rectangle based on the orthophoto.

building_refinement.npy: The building rectangles after refinement, the data format of each rectangle is (n by 16):

[number of segment, number of rectangle in a segment, x value of center, y value of center, length, width, orientation, 0, x1, y1, x2, y2, x3, y3, x4, y4].

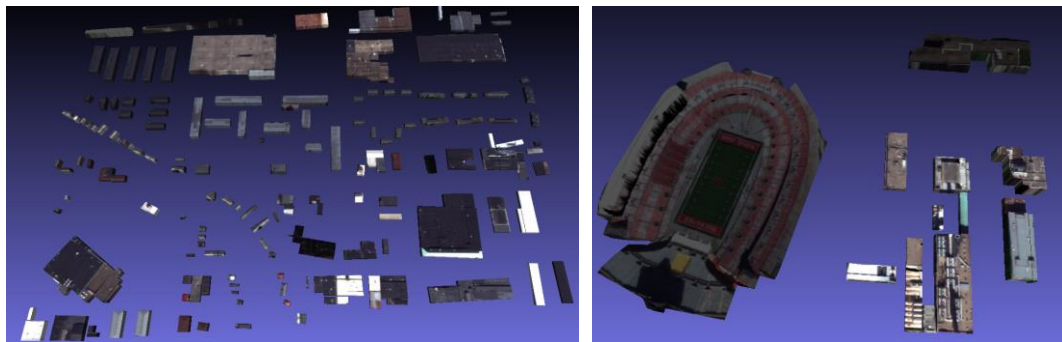
Shape2d.png: Visualized building rectangle after orientation refinement based on the orthophoto.

building_model3d.json: The building roof structure of each building rectangle, the data format is:

[5.0, 211.19618099084062, 212.59618099084062, 5.974999999999994, 5.974999999999994, 1.0, 0.3468099130053377]

[roof type (1: flat, 2: gable, 3: hip, 4: pyramid, 5: mansard), eave height, ridge height, length of hip 1, length of hip 2, hip direction, RMSE of original DSM].

building_model.obj, **building_model.mtl**, **model_texture.jpg**: LoD-2 building model with .obj format.



Notification: The points location of **building_corner.json**, **building_polygon.json**, **building_rectangle.json**, **building_refinement.npy** will have a (100, 100) offset than their original location in orthophoto and DSM, which means the point location (x, y) will be recorded as (x+100, y+100) in above files.

The example output files are shown in *./software/example/output/* folder. Typical processing time of example are about 3-4 minutes for 1500*1500 image, 10-12 minutes for 3000*3000 image.