User manual of Model3D.exe

This document is about the running of executable model3D.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).

To **start**, please click on ***./dist/model3D/model3D.exe***

Requirement

A **GPU** is required for user, if user wants to use the building detection and classification function.

Input data

Launching model3d.exe to start the building LoD-2 reconstruction workflow.

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

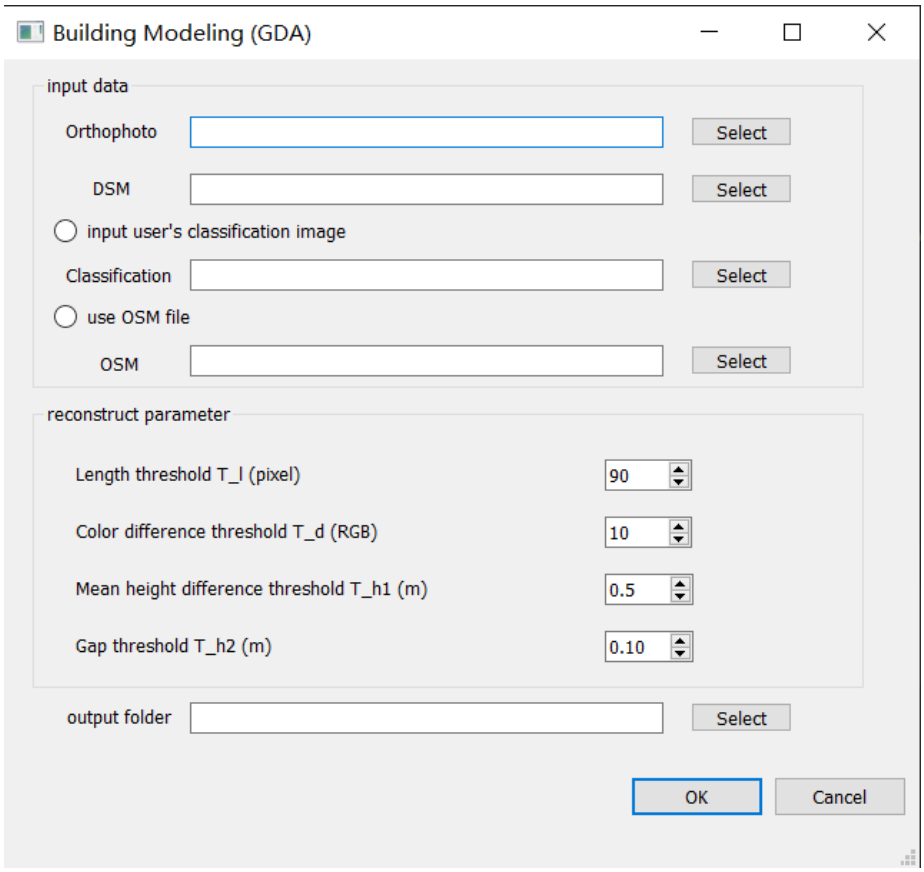

Figure 1. Main window of building LoD-2 reconstruction software

Required input data include: 1) *orthophoto* with RGB band (.png, .jpg, or .tif format); and 2) *DSM* (digital surface model, .tif format), and the orthophoto and DSM should have the same row and column.

Optional input data include: 1) *classification image*, building class is labeled as 255 (.png, .jpg, or .tif format); and 2) *OpenStreetMap shape file* (.shp format)

Building reconstruction parameters include: (please refer 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 ($\lvert\overline{C_1} - \overline{C_2}\rvert$) 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 ($\lvert\overline{H_1} - \overline{H_2}\rvert$) 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. Different from the method in the published paper, we adapt HRNet V2 as the building segmentation algorithm (training datasets are the same as paper), to extract 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 based on the building polygon 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) **Obj writing**: transform building model into an .obj format.

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, 413.490850434O1755, 178.90431610496617, 392.59890232820896,
189.26884881356713, 422.69169983412957, 198.88791864095697, 401.799751728321, 506, 0.41012734054149097, 1, 0.75, 0.9]]
```

[number of segment, number of rectangle in a segment, number of line in a segment, orientation, length of line].

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, 0.34680991300533777]
```

[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.