Instructions for scripts

Initial setup

Test data is provided in the form of a set of LAS files, which all contain data for a building used for testing (building_349). The test data needs to be extracted from a zip package that is located in the folder "Data/Test_segmented_data/".

To use the scripts, LOD2 data for the target buildings is required. This was created by splitting the up-to-date LOD2 model of the entire Sendai test area into separate .obj files, each of which contains the information of a single building. This collection of building models needs to be available alongside the LiDAR dataset.

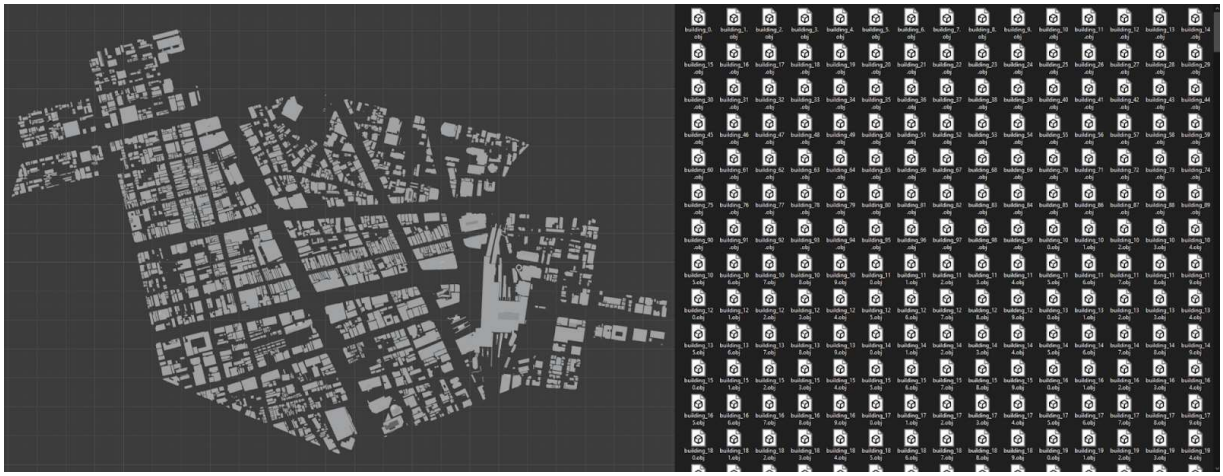

Image 1. LOD2 model of the city center, split into 2406 files.

The scripts have been developed to run on Windows PC, and might require some changes, if moved to another operating system.

Supported environments

Scripts were developed in a Windows 10 environment, but should be usable on other operating systems as well, possibly with minor changes required. The exception to this is the iPSR algorithm, which is compiled from source code for Windows only. To get it to run on other platforms, it needs to be recompiled, code available at: https://github.com/houfei0801/ipsr.

Create building footprints

Script name:	*"01_create_footprint_polygons.py"*
Usage:	*>>python 01_create_footprint_polygons.py*

Dependencies: *pyproj*

```
input_path          = "C:/Sendai/LOD2/Buildings/"
output_filename     = "C:/Sendai/LOD2/footprint_polygons_0m_dilation.txt"
dilation_amount     = 0.0
visualize_results   = True
```

Image 2. 01_create_footprint_polygons.py settings.

The script computes the horizontal 2D footprint polygon for every .obj model in the input path, and outputs the polygon node listings to a file, where they can be read from very quickly, when needed.

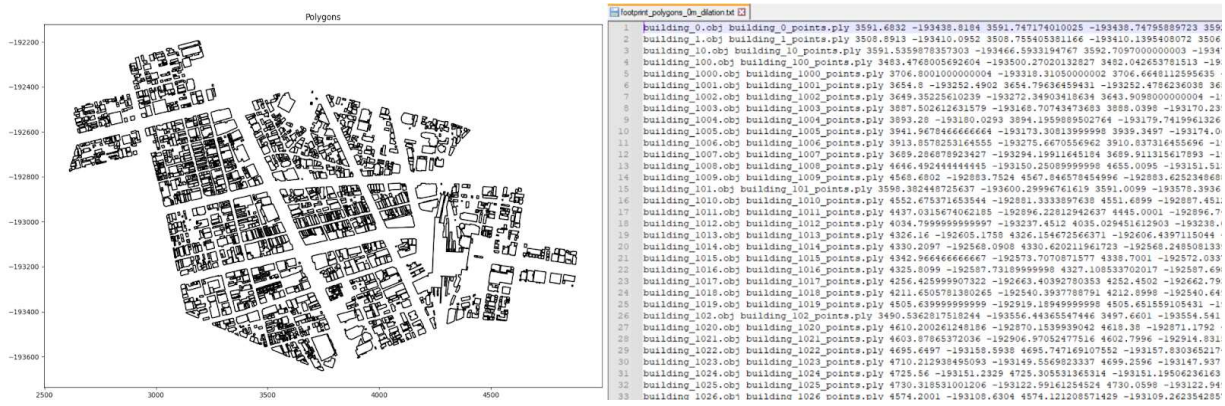

Image 3. 01_create_footprint_polygons.py output.

Expand footprint polygons

Script name: *“02_polygon_expansion.py”*
Script usage: *>>python 02_polygon_expansion.py*
Dependencies *matplotlib, numpy, shapely*

```
process_all_buildings       = True # if 'True' process all buildings in input file. If 'false', process only a single building defined by 'target_building'
target_building             = "building_349.obj"
input_filename              = "C:/Sendai/LOD2/footprint_polygons_0m_dilation.txt"
output_filename             = "C:/Sendai/LOD2/footprint_polygons_1_5m_dilation.txt"
neighbor_distance_threshold = 1.5
dilation_amount             = 1.5
visualize_results           = True
output_results_to_file      = True
```

Image 4. 02_polygon_expansion.py settings.

The script increases the size of the polygons in the input file, while taking into account all the other buildings nearby, so the resulting expanded polygon does not intersect with them.

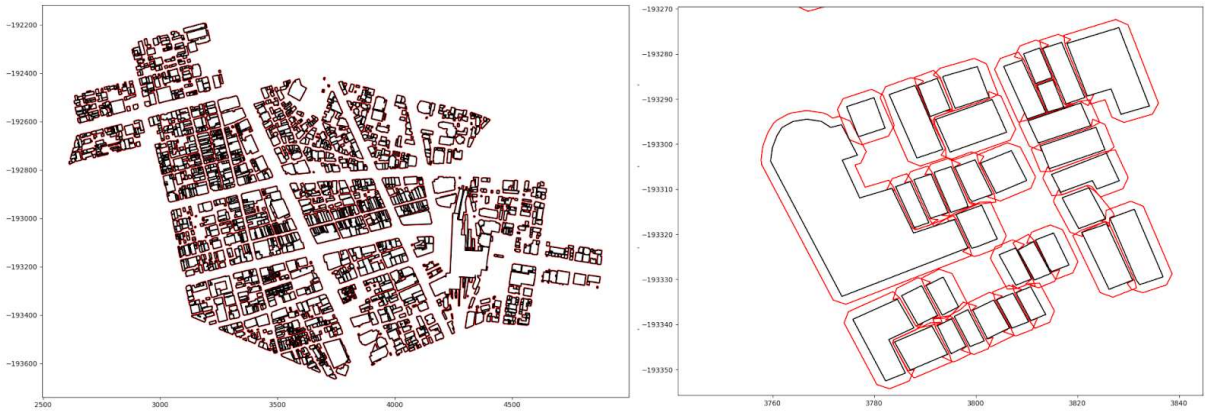

Image 5. 02_polygon_expansion.py output.

Define target area

These scripts are convenient, if multiple buildings are processed simultaneously in a pre-defined region of the city. To process a single building, continue to the next section.

Use any available online coordinate picker tool (for example: https://www.latlong.net/) to define a polygon on the map, the buildings in which should be processed.

Image 6. Use a coordinate picking tool to get latitude/longitude points of the target area.

One by one, feed the latitude and longitude coordinates to the script “epsg_converter.py” to get the matching coordinates in the coordinate system used by the LiDAR dataset.

Script name: *“epsg_converter.py”*
Script usage: *>>python epsg_converter.py*
Dependencies: *pyproj*

Image 7. Input the latitude and longitude coordinates into the script "epsg_converter.py" to get the same coordinates in the correct coordinate system.

The converter coordinates are then input to another script, "03_get_building_list_in_polygon.py" as nodes of the input list "polygon_points". The script will check the dataset against the polygon, and collect a list of all buildings inside it, which is then saved to a file.

Script name: *"03_get_building_list_in_polygon.py"*
Script usage: *>>python 03_get_building_list_in_polygon.py*
Dependencies: *matplotlib*

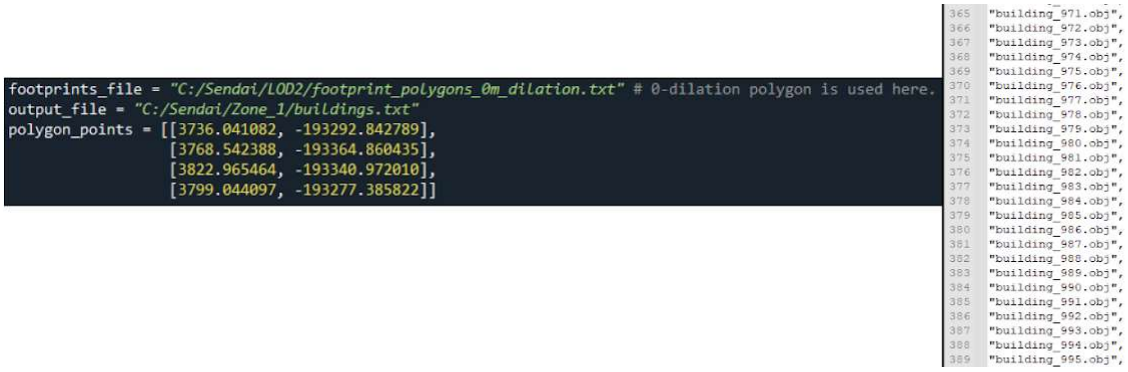

Image 8. Input the converted coordinates to the script "03_get_building_list_in_polygon.py", which will then generate a list of buildings within that area.

Another script, "create_dataset_bounding_box_list.py", can be used to create a *bounds* file, which is used to access the potentially very large LiDAR dataset faster. It will create a table of all LAS files' min/max bounds in the world, so potential data for each area can be recovered faster, and its output is required by some of the scripts in the pipeline.

Extract buildings and align data

The script "04_split_dataset_into_buildings_with_realignment.py" goes through all the point clouds in the dataset, extracting points that fall within the building's expanded footprint polygon. The point clouds are aligned against each other, and combined into a single LAS file. Each point in the LAS file has a UserData field, which is used to store the IDs of the points' originating point clouds in case further adjustments are needed.

In the script, in case of processing a single building, set the name of the target building object file in the list 'input_building_list' (e.g. *["building_1316.obj"]*). Multiple object names can be inserted, separated by a comma (*e.g. ["building_1316.obj", "building_1317.obj"]*). A list of these building object names can be produced by the scripts defined in the previous section.

Script name: *"04_split_dataset_into_buildings_with_realignment.py"*

Usage: *>>python 04_split_dataset_into_buildings_with_realignment.py*
Dependencies: *open3d, laspy, numpy, matplotlib*

```
building_polygons_file    = "C:/Sendai/LOD2/footprint_polygons_1_5m_dilation.txt"
bounds_file               = "C:/Sendai/Segmented_dataset/las_bounds.txt"
input_dataset_path        = "C:/Sendai/Segmented_dataset/las/"
input_building_list_file  = "C:/Sendai/Zone_1/buildings.txt"
aligned_output_path       = "C:/Sendai/Zone_1/Aligned_point_clouds/"
unaligned_output_path     = "C:/Sendai/Zone_1/Unaligned_point_clouds/"

visualize_alignment_steps = False
radius_normal             = 0.4
matching_resolution       = 0.02
force_las_output          = True

input_building_list       = ["building_997.obj"]
```

Image 9. Settings for the script "04_split_dataset_into_buildings_with_realignment.py".

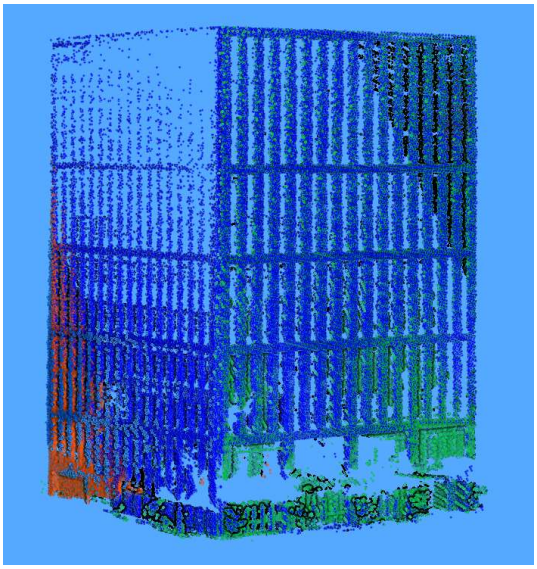

Image 10. Output of "04_split_dataset_into_buildings_with_realignment.py", with all partial point clouds color coded based on their ID.

Combine LiDAR data with LOD2 data

The script "05_combined_LOD2_with_point_cloud.py" attempts to use LOD2 data to patch in holes in LiDAR data coverage of a building by measuring distances to nearest LiDAR data points from each LOD2 data point.

It needs to be noted that this script does not work well, when the LOD2 model dimensions are not accurate. It is, for example, quite common that a wall in LOD2 data is located even

several meters away from the LiDAR data positions, in which case the script output can be very bad.

Script name:	*“05_combine_LOD2_and_point_cloud.py”*
Usage:	*>>python 05_combine_LOD2_and_point_cloud.py*
Dependencies:	*open3d, matplotlib, numpy, trimesh, laspy*

```
input_point_cloud_path        = "C:/Sendai/Zone_1/Aligned_point_clouds/"
input_lod2_obj_path           = "C:/Sendai/LOD2/Buildings/"
output_point_cloud_path       = "C:/Sendai/Zone_1/Combined_point_clouds/"
output_mesh_path              = "C:/Sendai/Zone_1/Meshes/"

# Process single file
process_entire_directory      = False # If False, only a single building, defined below, will be processed from the input paths
building_number               = 349
input_point_cloud_filename    = "aligned_building_"  + str(building_number) + "_points.las"
input_lod2_obj_filename       = "building_"          + str(building_number) + ".obj"
combined_point_cloud_filename = "combined_building_" + str(building_number) + "_points.ply" # Output in .ply format, because the iPSR implementation uses it.
```

Image 11. Settings for the script “05_combine_LOD2_and_point_cloud.py”. Input and output paths need to be set, and in case the ‘process_entire_directory’ flag is set to False, the number of the building needs to be defined.

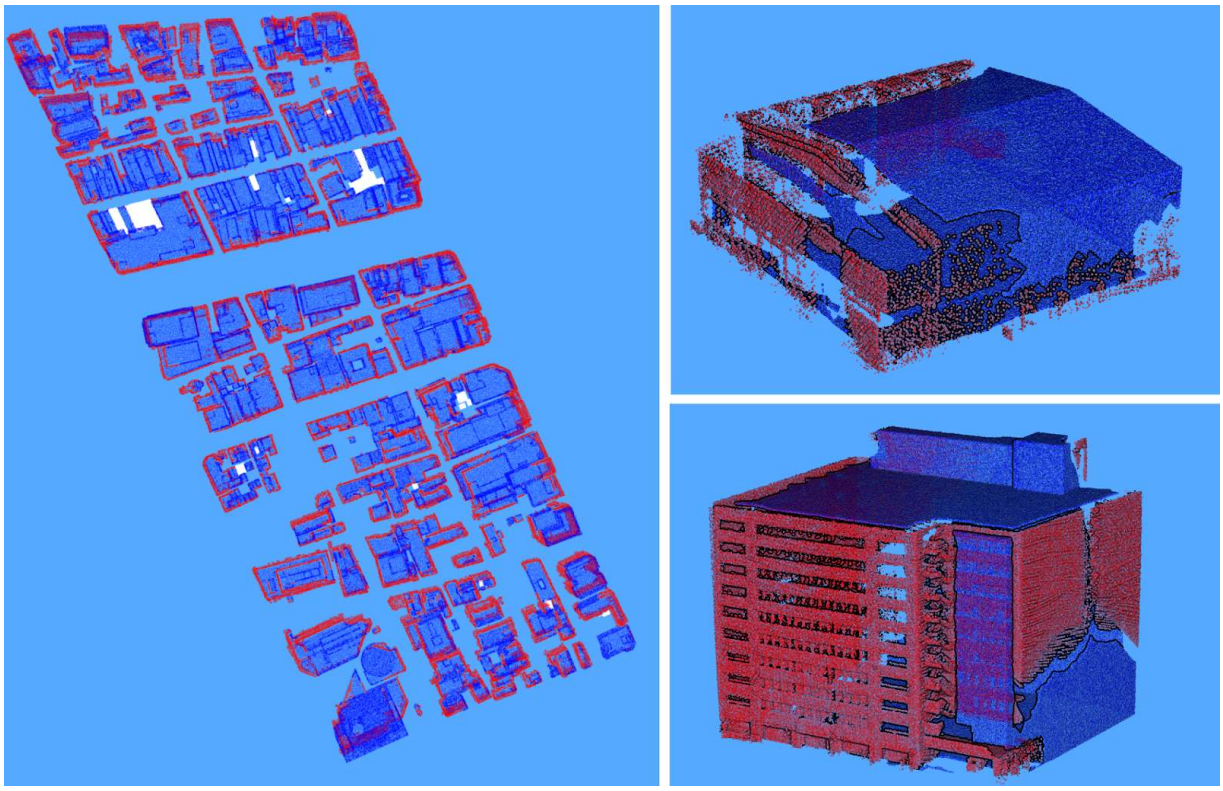

Image 12. Examples of combined LiDAR and LOD2 data. Red points originate from LiDAR data, blue points from LOD2 data.

Create a mesh using iPSR algorithm

Script name:	*“06_run_iPSR.py”*
Usage:	*>>python 06_run_iPSR.py*
Dependencies:	*laspy, open3d, numpy*

```
process_entire_folder = True

input_path = "C:/Sendai/Zone_1/Combined_point_clouds/"
output_path = "C:/Sendai/Zone_1/Meshes/"

# Process single file
building_number = 349
input_filename = "combined_building_" + str(building_number) + ".ply"
output_filename = "mesh_" + str(building_number) + ".ply"
```

Image 13. Settings for running the "06_run_iPSR.py" script. If 'process_entire_folder' is set to False, the 'building_number' variable is used to set input and output files.

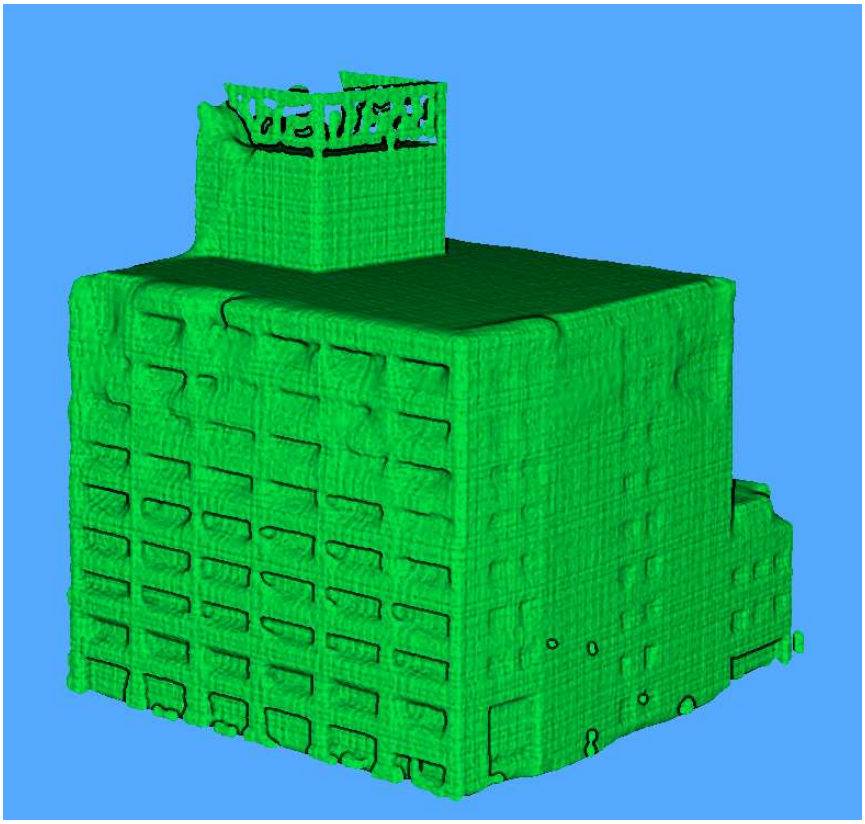

Image 14. Output of the iPSR algorithm, when LiDAR and LOD2 data complement each other well.