Conversion of LoD2.2 to LoD1.2 Building Models (PhD assignment)

The Aim of the Task

The goal of this task is to implement a software tool that converts a detailed LoD2.2 building model to a simplified LoD1.2 representation. This conversion reduces the complexity of the building geometry by removing detailed roof structures and flattening the roof surface, resulting in a block-like model useful for various urban analysis applications (e.g., energy estimation, noise simulation, and visualization).

To simplify the problem:

- Input and output are in the Wavefront `.OBJ` format.
- Each input file contains a single triangulated building.
- Semantics (e.g., building parts or functions) are not considered—only geometry is processed.

Definition of LoD1.2 and LoD2.2

According to Biljecki (2014):

- **LoD1.2**: A block-shaped building model with a single flat roof, including smaller building extensions (e.g., alcoves). The building is extruded to a single height.
- **LoD2.2**: A detailed building model with general roof shapes (as in LoD2.0 and LoD2.1), including roof superstructures larger than 2 meters in size or 2 m² in area (e.g., dormers, chimneys).

LoD1.2 models strike a balance between information richness and computational efficiency, making them suitable for simulations of shadows, energy demand, noise, flooding, and urban wind comfort.

Methodologies

Step0: Preprocessing

Before performing the LoD2.2 to LoD1.2 conversion, a minimal but essential preprocessing step was applied to ensure the robustness of the surface classification and height estimation steps.

The core challenges in this task revolve around:

- **Identifying ground surfaces** using only geometric properties, and
- **Determining an appropriate roof height** that represents the building's overall geometry without being biased by outliers.

However, the presence of **invalid or unrealistic geometry** can significantly impact both steps. For example, in the `bk.obj` model, there was a set of **spike-like faces protruding downward** from the building base—an artifact that does not correspond to any real-world structure.

To simplify the problem and ensure stable behavior of the algorithm, I **manually removed these erroneous faces** before running the pipeline. The figure below shows an example of such a spike that was eliminated.

This preprocessing step ensures a cleaner input mesh, particularly for buildings where outliers could distort the calculated ground level or dominate the roof height estimation. For the other models (`simple.obj`, `beeb.obj`, `otb.obj`), no such cleaning was necessary.

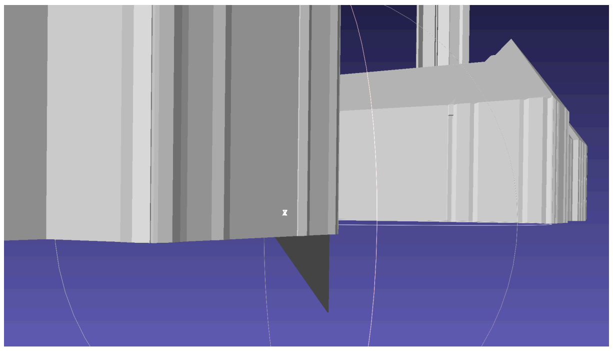


Figure: example of spike-like faces (BK)

Step 1: Surface Classification

- **Compute the lowest z-value** among all building vertices. This acts as a reference height for identifying the ground.
- Compute face normals to classify surfaces:
  1. **GroundSurface**:
     - Surfaces with normal vectors nearly vertical (dot product with z-axis ≈ ±1.0).
     - min z-values close (within 1.5 meters) to the building's minimum z-coordinate.
  2. **WallSurface**:
     - Vertical faces with normal vectors perpendicular to the z-axis (dot product ≈ 0.0).
  3. **RoofSurface**:
     - All other surfaces above the GroundSurface.

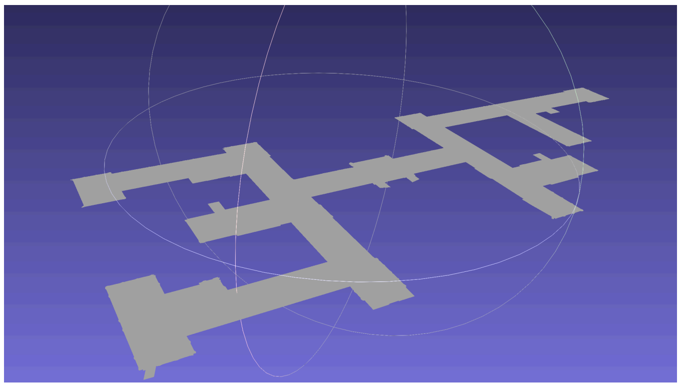


Figure: extracted ground surface

Step 2: LoD1.2 Height Calculation

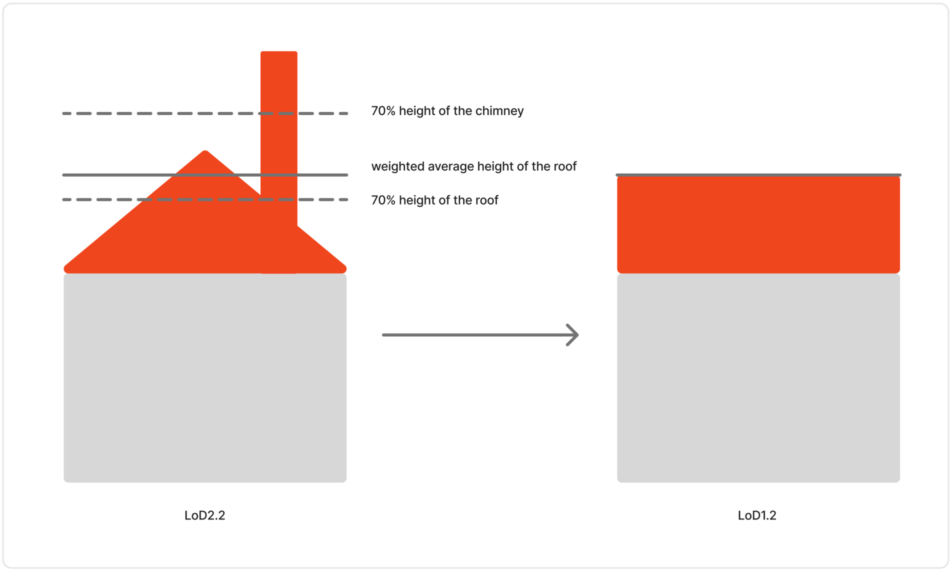
- Since many roof surfaces are sloped, we do not simply take the maximum height. Instead, for each roof face, we calculate 70% of the vertical span between its highest and lowest vertex to represent a typical height. The choice of the 70% threshold follows the 3DBAG project's approach We then compute the area-weighted average of these adjusted heights so that larger, more dominant roof surfaces contribute more significantly to the final building height.

$$H = \frac{1}{\sum_{i \in \mathcal{R}} A_i} \sum_{i \in \mathcal{R}} A_i \cdot \left(z_{\min}^{(i)} + (z_{\max}^{(i)} - z_{\min}^{(i)}) \cdot p \right)$$

where:

H	is the final roof height of the LoD1.2 model
$\mathcal{R}$	is the set of all roof faces
A_i	is the area of roof face i
$z_{\text{min}}^{(i)}$	is the minimum z-coordinate of face i
$z_{\text{max}}^{(i)}$	is the maximum z-coordinate of face i
p	is the roof height percentile (e.g., $p = 0.7$ for 70%)

Methodology


70% height of the chimney
weighted average height of the roof
70% height of the roof
LoD2.2
LoD1.2

Step 3: Footprint Extraction and Extrusion

- Identify boundary edges of the GroundSurface (edges used only once).
- Vertically extrude these edges to the calculated LoD1.2 height.
- Generate vertical faces (walls) connecting original and extruded ground vertices.
- Construct a flat horizontal roof at the LoD1.2 height using the extruded polygon.

Step 4: Model Construction

- Combine:
    - Original GroundSurface.
    - Newly created vertical WallSurfaces.
    - Newly created flat RoofSurface.
- Remove all original RoofSurfaces and WallSurfaces.

Discussions

How to classify ground surface correctly?

Option 1: Classification Based Solely on Normals

A straightforward approach is to classify surfaces using only their normal vectors. Surfaces with normals **pointing upward are labeled as roofs, those pointing downward as ground, and those perpendicular to the z-axis as walls**. While this method is fast and easy to implement, it assumes consistent surface orientation throughout the model. In practice, however, **inconsistencies in face orientation or the presence of below-ground structures can lead to misclassification**. This makes the approach unreliable, especially for models derived from automated reconstruction.

Option 2: Region-Growing via Adjacency Traversal

A more robust alternative is to perform region-growing starting from the lowest surface, which is assumed to be the ground. From this seed, the algorithm can traverse adjacent faces and classify them based on orientation relative to the up vector. This method is more resilient to variations in model geometry, particularly when normals are not reliable. However spike like ground surface collapse this approach.

Option 3 (Used): Minimum Height Anchoring with Normal Filtering

The chosen approach combines the advantages of the previous methods. It begins by identifying the minimum z-value in the building model to anchor the ground level. Then, each face is classified based on its orientation relative to the z-axis and its elevation relative to the minimum height. Horizontal surfaces near the minimum height are labeled as ground, vertical surfaces as walls, and remaining surfaces as roofs.

Why?

It's simple and more robust for dataset having invalid geometry

---

How to calculate roof height?

Option 1: Volume Preservation

A methodologies considered was to determine the extrusion height such that the resulting LoD1.2 model preserves the volume of the original LoD2.2 building. This method is more physically meaningful and beneficial for simulations that rely on accurate volumetric information (I honestly prefer this). However, computing the volume of an arbitrary triangulated mesh requires the model to be watertight and well-formed. Given the complexity of implementing volume repair and the time limitations of the task, this method was deemed impractical.

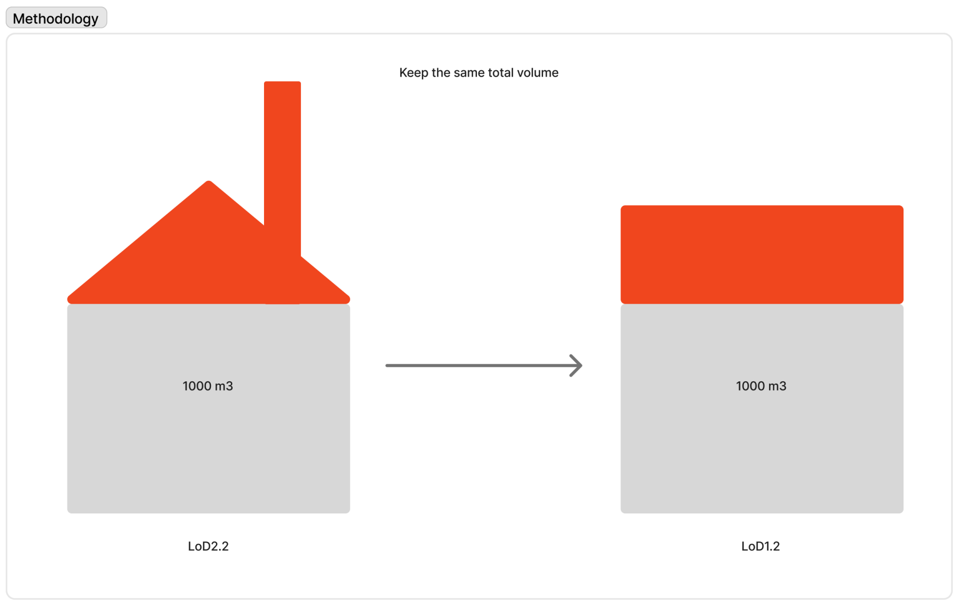


Figure: example of volume preservation approach

Option 2 (Used): Area-Weighted Percentile Height

This approached has been already explain in the section before.

Result

The implemented method was successfully applied to the provided LoD2.2 building models. As illustrated in the figures, the algorithm was able to extract simplified LoD1.2 representations that conform to the expected geometric abstraction.

Visual Intepretation

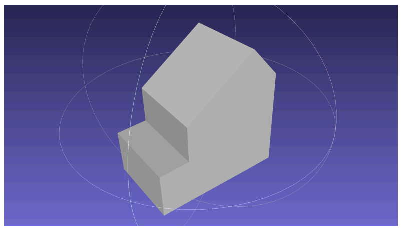
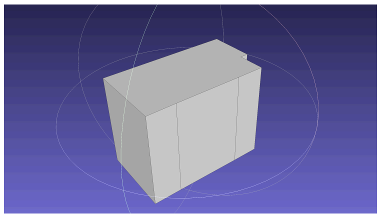
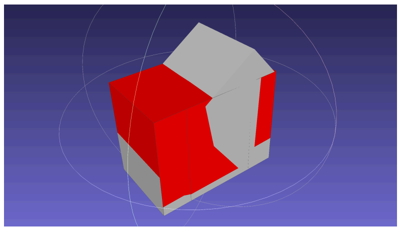
Each figure presents three visualizations:

1. The original **LoD2.2** model, with detailed roof structures;
2. The resulting **LoD1.2** model, characterized by a flat roof and vertical walls; and
3. An **overlay** of both models within the same scene for direct comparison—where the LoD1.2 model is rendered in red for clarity.

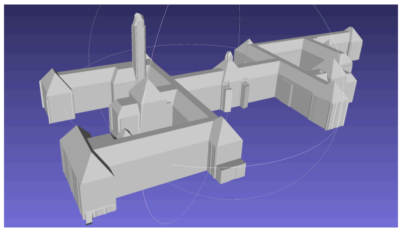
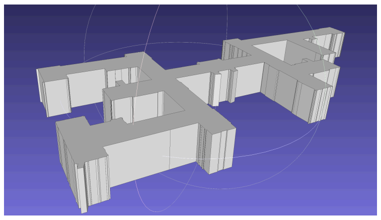
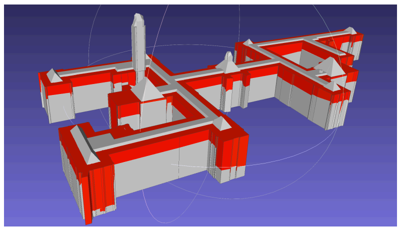
These visualizations confirm that the simplified models maintain the original footprint and vertical extent of the buildings while effectively removing fine roof details. The overlay demonstrates that the LoD1.2 geometry aligns well with the original model's envelope, with height adjustments reflecting dominant roof shapes rather than local outliers.

Overall, the results indicate that the approach is robust and suitable for generalized LoD simplification tasks, particularly in applications where computational efficiency and geometric abstraction are preferred over detailed roof fidelity.

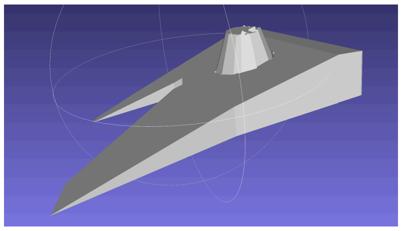
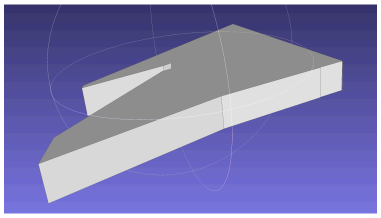
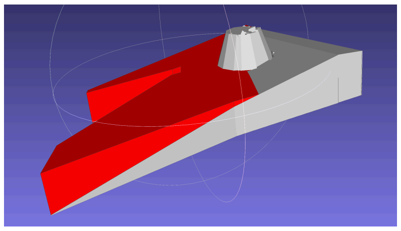
Simple.obj

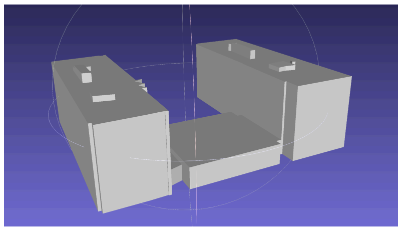
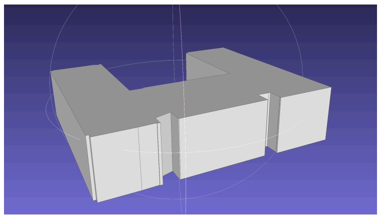
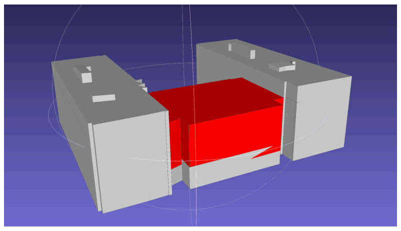
BK.obj

Beeb.obj

Otb.obj

Quantitative Analysis (Future)

At this stage, a quantitative evaluation has not yet been conducted. However, in the next phase of development, a set of standard geometric and data representation metrics will be used to systematically compare the LoD2.2 and LoD1.2 models.

Following the evaluation criteria proposed by **Biljecki et al. (2014)**, the analysis will include:

- **Average triangle count per building**
- **Total surface area**
- **Area of the WallSurface**
- **Volume of the corresponding solid**
- **Size in memory for each representation**

This evaluation will provide insight into how well the simplified LoD1.2 models preserve key physical and structural properties of the original geometry. It will also help to identify any systematic biases introduced during simplification and inform potential improvements to the pipeline.

Limitations and Disclaimer

While the proposed method performs well on the provided dataset, several assumptions and limitations should be acknowledged to contextualize its applicability and reliability.

Limited Dataset Testing

The program has only been evaluated on the four OBJ files provided with the task. These files are relatively clean and consistent in structure. As a result, the robustness of the method on larger or more diverse datasets remains untested. Generalization to different building typologies or real-world urban datasets is not guaranteed.

Assumption on Minimum Height (z-value)

The approach assumes that the **minimum z-value** among all vertices corresponds to a valid ground surface. However, models may contain geometric outliers or artifacts—such as downward-pointing spikes—that violate this assumption. In such cases, the resulting LoD1.2 model may be incorrectly grounded. For instance, in the `bk.obj` file, spike-like faces below the building distorted the ground detection and had to be manually removed.

Lack of Face Orientation Handling

The current implementation does not verify or correct the **orientation of face vertices**. Ideally, faces should follow the **right-hand rule**, with outward-pointing normals. This assumption may affect normal-based classification and rendering. When visualizing the output, it is recommended to enable double-sided face rendering to ensure visibility of all surfaces.

No Mesh Cleaning or Validation

The method does not account for **invalid or inconsistent geometry** such as:

- Non-manifold edges
- Self-intersecting faces
- Duplicate vertices
- Unconnected components

While the current models were processed successfully, robustness to such issues is not guaranteed without further mesh cleaning.

Limited Support for Complex Building Structures

The method assumes relatively regular building geometry. It does not yet support complex cases such as:

- Courtyards or interior voids
- Overhangs or cantilevered parts
- Multi-level terraces

These elements may not be preserved or represented meaningfully in the simplified LoD1.2 model.

Volume Preservation Not Guaranteed

The extrusion height is computed via a percentile-based approximation and does not attempt to preserve the original building **volume**. In many applications, this is sufficient. However, for energy modeling, flood simulation, or structural analysis—where volume consistency matters—this method may yield inaccurate results.

Only Tested on Buildings

The method has been developed and tested exclusively on **building-type objects**. It has not been evaluated on other City Objects such as bridges, tunnels, or installations. As such, its applicability to these other classes remains unknown.

Conclusion

This project presents a simplified yet effective method for converting LoD2.2 building models into LoD1.2 representations using only geometric information from triangulated OBJ files. By combining heuristic surface classification with a percentile-based height estimation approach inspired by the 3DBAG project, the method successfully generates block-shaped models that are visually consistent with the original building envelopes.

The implementation was tested on a small set of building models, demonstrating that the approach is capable of robustly simplifying roof structures while preserving the overall shape and footprint. While the current evaluation is limited to visual inspection, future work will focus on quantitative analysis to assess how well geometric and volumetric properties are maintained.

This work serves as a foundation for further refinement, with opportunities to enhance robustness, support more complex geometries, and evaluate performance across a wider range of urban data.

References

- 3D geoinformation research group. (n.d.). *Data Layers - 3DBAG*. https://docs.3dbag.nl/en/schema/layers/
- Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers Environment and Urban Systems*, *59*, 25–37. https://doi.org/10.1016/j.compenvurbsys.2016.04.005
- Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K. H., & Khoo, V. H. S. (2016). THE MOST COMMON GEOMETRIC AND SEMANTIC ERRORS IN CITYGML DATASETS. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-2/W1*, 13–22. https://doi.org/10.5194/isprs-annals-iv-2-w1-13-2016