Structure-aware Building Mesh Polygonization
Vasileios Bouzas, Hugo Ledoux, Liangliang Nan

3D Geoinformation Research Group, Faculty of Architecture and the Built Environment, Delft University of Technology, 2628 BL Delft, The Netherlands

A R T I C L E   I N F O
Keywords:
MVS meshes
Structure awareness
Polygonization
Simplification

A B S T R A C T
We introduce a novel approach for the polygonization of Multi-view Stereo (MVS) meshes of buildings, which results in compact and topologically valid models. The main characteristic of our method is structure awareness, i.e., the recovery and preservation of the initial mesh primitives and their adjacencies. Our proposed methodology consists of three main stages: (a) primitive detection via mesh segmentation, (b) encoding of primitive adjacencies into a graph, and (c) polygonization. Polygonization is based on the approximation of the original mesh with a candidate set of planar polygonal faces. On this candidate set, we apply a binary labelling formulation to select and assemble an optimal set of faces under hard constraints that ensure that the final model is both manifold and watertight. Experiments on various building models demonstrate that our simplification method can produce simpler representations for both closed and open building meshes. Furthermore, these representations highly conform to the initial structure and are ready to be used for spatial analysis. The source code of this work is freely available at https://github.com/VasileiosBouzas/MeshPolygonization.

1. Introduction
In recent decades, there is an ever-increasing demand, both by academia and industry, for 3D spatial information and 3D City models (Biljecki et al., 2015). In contrast to 2D data, 3D data either provides a much more enriched context for some applications or makes others possible. Applications that benefit from the use of 3D data vary from infrastructure planning, utility management, and 3D cadastre to solar potential estimation and visibility analysis.

One common practice to obtain 3D models of buildings and urban scenes is the acquisition of massive point clouds through Airborne Laser Scanning (ALS), Structure from Motion (SfM), or Multi-view Stereo (MVS) (Furukawa and Hernández, 2015). These point clouds, combined with reconstruction techniques (Bernardini et al., 1999; Kazhdan et al., 2006), enable the representation of buildings in the form of surface meshes. Although the quality of these meshes is sufficient for visualization purposes, it is still not enough for other applications, such as urban planning and simulations (Holzmann et al., 2017), due to:

- Large memory size: The number of faces also increases the requirements of these meshes in memory space.
- Missing information: The incompleteness of the original point cloud, due to occlusion or other causes, often prevents the reconstruction of a given scene in its entirety.
- Noisy and undesired structures: Flaws both in the original point cloud and the reconstruction method lead to defects in the final mesh (e.g., self-intersecting parts or holes).

These problems are often addressed with mesh simplification and polygonization. Despite the similarities of these two approaches, simplification relates to the removal of redundant faces for representing the original mesh (Garland and Heckbert, 1997). On the other hand, the objective of polygonization is the approximation of the original mesh with a set of polygonal surfaces. The existing simplification and polygonization techniques (Garland and Heckbert, 1997; Salinas et al., 2015; Cohen-Steiner et al., 2004) succeed in the production of lightweight meshes with less complexity and memory requirements than their original counterparts. However, these meshes often lack the topological validity necessary for their use in real-world applications such as simulations.

In this work, we propose a novel approach for producing simpler, more compact representations for MVS building meshes through polygonization (see Fig. 1). Our inputs are building mesh models acquired from aerial and terrestrial imagery, for which we assume that the building instances are extracted from urban scenes via semantic segmentation (Landrieu and Simonovsky, 2018; Verdier et al., 2015; Rouhani et al., 2017; Zhu et al., 2018; Valentijn et al., 2013). Also, we mainly target buildings whose geometry can be represented by closed
polyhedra without dangling faces. We assess the effectiveness of our simplification method with several criteria. Apart from a lightweight representation of the original model, the resulting mesh should be also manifold, watertight, and free of geometric errors. This allows us to use it as input in software for different applications (e.g., energy estimation, wind simulation, cadastral, and solar potential). Furthermore, the outcome of this method should be independent of any imperfections in the original mesh (e.g., geometric or topological defects, noise, and undesired structures). Finally, simplification results should be accomplished within reasonable execution times to allow the processing of large-scale environments.

To achieve polygonization, we first detect the planar components (geometry) of the input mesh along with their configuration (topology) in the 3D space. Based on this information, we form an initial set of candidate faces to approximate the mesh. To construct the simplified model, we select candidate faces through a constrained optimization process that ensures the final result is both manifold and watertight. Our contributions to the current state of the art are the following:

- A novel mesh segmentation technique based on region growing for the detection of planar components in surface meshes;
- An optimization-based method for the construction of the simplified surface models based on the definition of sharp features through a building scaffold and of faces through 2D arrangements.

2. Related work

There is a large volume of research on mesh polygonization. In this section, we only review the most relevant work in the scientific literature for topics directly related to ours.

**Planar shape detection/abstraction.** Contrary to natural objects, man-made constructions conform to clearly defined geometries, thus allowing their approximation via an assembly of planar shapes. This approximation stands as the basis for several mesh simplification/polygonization techniques and therefore, there has been an extensive literature on the detection and abstraction of planar shapes both in point clouds and meshes. Several approaches to this problem (region growing/RANSAC (Schnabel et al., 2007)) attempt to decompose the input cloud or mesh into planes in one go, based on one or more attributes (normal orientation, curvature, planarity, etc.). Others use this decomposition as input for an optimization process during which the initial planar set needs to conform to a predefined metric (Monszpart et al., 2015; Oesau et al., 2015; Fang et al., 2018; Jonsson, 2016). Following these work, we contribute to the current state of the art with our proposed mesh segmentation technique (see Section 3.1).

**Plane assemblies.** The detection and assembly of planar shapes for the construction of compact polygon meshes constitute the core of our method. As in other existing methods (Nan and Wonka, 2017; Chauve et al., 2010; Chen and Chen, 2007; Bautz et al., 2019; Fang, 2019), the main problems to be addressed in this approach is the completeness of the final result despite imperfections in the input data (geometric/topological flaws, holes, and noise, etc.). While a holistic approach (i.e., pairwise intersections between all available planes) proves to be enough for tackling this problem, it often increases execution time considerably. To balance this trade-off between completeness and computational efficiency, it is therefore necessary to reduce the number of computations but not at the expense of the quality of the result. In our method, this is achieved with the introduction of our structure graph.

**Urban reconstruction.** The various methods currently existing for the reconstruction of urban environments are divided into three main categories: (1) building mesh reconstruction (Holzmann et al., 2017; Nan and Wonka, 2017; Li et al., 2016; Bódis-Szomorí et al., 2015), (2) building mesh regularization (Jonsson, 2016; Wang et al., 2016; Kelly et al., 2017), and (3) urban scene reconstruction (Verdie et al., 2015; Zhu et al., 2018). The first two categories produce 3D models for individual buildings, while the third one focuses on recreating entire urban scenes. Despite their effectiveness, many of these methods depend on multiple sources of information that are not always available (e.g., aerial imagery, point clouds, GIS data) (Kelly et al., 2017; Zhu et al., 2018). Others conform to the already reconstructed models to certain geometric regulations (e.g., vertex projection to planar primitives, orthogonality, perpendicularity), but still maintain the initial number of meshes faces (Bódis-Szomorí et al., 2015; Jonsson, 2016; Wang et al., 2016). Our method satisfies both the requirements of geometric regularity and simple representation, with only the reconstructed mesh as an input.

**Mesh simplification/polygonization.** In computer graphics, various methods exist for the simplification of 3D meshes, mainly for visualization and animation purposes. For example, Quadric Error metrics (QEM) (Garland and Heckbert, 1997) or Structure-Aware Mesh Decimation (SAMD) (Salinas et al., 2015) simplifies meshes by iteratively reducing the number of their simplexes. A characteristic example of polygonization techniques is Variational Shape Approximation (VSA) (Cohen-Steiner et al., 2004), which constructs an entirely new mesh that approximates the original model and consists of planar shapes (proxies). In either case, the topological validity of the simplified mesh, essential for its usage in further applications, is not always ensured. In this work, we combine simple representation with topological validity, resulting in building models ready for spatial analysis.

**Structure awareness.** Although the concept of structure still lacks any universally accepted definition, there has been some research on structure-aware shape processing over the last decades (Mitra et al., 2013; Salinas et al., 2015). Regardless of their unique characteristics, most structure-aware approaches define the structure of an object as the assembly of two elements: (a) the parts composing the object and (b) the interrelationships between these object parts. In return, shape processing also consists of two separate procedures: structure detection and processing according to the detected structure.

In this paper, we assume that the majority of real-world buildings demonstrate piecewise planar structures. We define the structure of a building by detecting the planar primitives of the input model along with their adjacency relations. Furthermore, any information on the structure is encoded into a structure graph, an undirected graph whose vertices correspond to the detected primitives while the edges denote adjacency relations between primitive pairs. The main advantages of structure awareness, in the form of the structure graph, for the simplification process are the following:

---

Fig. 1. An urban scene with its buildings simplified using our method. Each building was simplified individually after the semantic segmentation of the scene.
3. Methodology

As shown in Fig. 2, our methodology consists of three main processing stages: (a) detection of primitives via segmentation, (b) encoding of the primitive interrelationships in a structure graph, and (c) structure-aware polygonization.

3.1. Segmentation

To define the structure, we first identify the primitives of the input mesh via segmentation. We presume that the input mesh is composed only of planar components, thus ignoring any spherical, cylindrical, and conical geometries. The planar regions we wish to detect correspond to floor, façade, and roof segments, while any architectural details (e.g., windows, chimneys) are ignored. These details are usually represented by a small number of faces due to the limited resolution of MVS meshes, therefore their approximation with planar components is difficult (Verdie et al., 2015).

We detect planar segments via a region-growing algorithm, based on the computation of planarity for the k-ring neighbourhood of each mesh vertex (Gatzke and Grimm, 2006). Specifically, the 1-ring neighbourhood of a vertex consists of all vertices directly connected to it through an edge. This notion of a neighbourhood can be defined for extended mesh regions (for example, a 2-ring neighbourhood includes also the vertices adjacent to those forming the 1-ring neighbourhood), as well as for mesh faces. Hence, the k-ring planarity of a vertex describes the degree of fitting a plane on its k-ring neighbourhood (Pauly et al., 2002).

Our segmentation algorithm can be summarized as follows (for further details, see Algorithm 1):

(a) We compute the k-ring planarity of all the mesh vertices, while the planarity of each face is set equal to the average planarity of its vertices.
(b) With the highest planarity face (seed), we initialize the first planar region and collect its k-ring neighbours to define a reference plane via Principal Components Analysis (PCA) (Pauly et al., 2002).
(c) By examining the k-ring neighbourhoods of the seed, we append faces in the region as long as their vertices are within a distance threshold from the reference plane.
(d) We repeat the whole process till the entirety of the mesh has been decomposed into planar regions.
3.3. Polygonization

With the structure of the input model fully defined, we move to the polygonization itself. Our polygonization process is divided into three separate stages: (1) the construction of a building scaffold, (2) the generation of candidate faces, and (3) the selection of candidate faces through optimization to form the simplified model.

3.3.1. Building scaffold

Similar to Variational Shape Approximation (VSA) (Cohen-Steiner et al., 2004), our method also approximates the original model with a set of planar shapes (proxies). Each of these proxies corresponds to a primitive we detected via segmentation. To form the simplified mesh, we first define the boundaries of these proxies. These boundaries should connect proxies whose primitives are also adjacent in the original model, thus preserving the adjacencies recorded in our structure graph.

Here, we determine the borders of the proxies with the construction of the building scaffold. This scaffold is a graph consisting of any sharp features detected in the original model. We specifically focus on sharp features of two types; corners (formed out of three adjacent primitives) and non-planar edges (formed out of two adjacent primitives). To detect corners, we identify all triplets of adjacent regions in the structure graph and compute the intersections of their supporting planes. In the same way, we compute the intersections for pairs of adjacent regions to detect non-planar edges (see Fig. 5).

3.3.2. Candidate faces

With the construction of the building scaffold, we first approximate the original mesh with a wireframe mesh consisting only of vertices and edges. We also define the faces of this mesh (candidate faces) through the following procedure:

- For each planar region, we collect its scaffold edges and project them on the supporting plane of that region.
- The projections of the edges form a 2D arrangement (see Fig. 6), a subdivision of the plane into vertices, edges, and faces (Agarwal and Sharir, 2000). The faces of this arrangement define the candidate faces representing the planar region in our simplified mesh.

The outcome of this procedure is the formation of a proxy mesh (see Fig. 7). However, several adjacencies recorded in the structure graph might be incorrect due to segmentation errors, which results in candidate faces that do not correspond to any of the primitives from the input model. To reliably eliminate these redundant faces, the construction of the final, simplified mesh is achieved via an optimization process.

3.3.3. Optimization

We adopt and adapt here the optimization process developed by Nan et al. (Nan and Wonka, 2017) for the reconstruction of polygonal surfaces from point clouds. This optimization is based on the formulation of a binary linear programming problem (Papadimitriou and Steiglitz, 1982; Williams, 2009). For this type of problem, each of the unknowns is represented by a variable whose value can be either 0 (not chosen) or 1 (chosen). These variables are connected through (a) an energy function that maximizes the expectation of some reconstruction objectives and (b) a set of constraints that ensure the resulting mesh is both manifold and watertight. For the binary variable \( x_i \) of each candidate face, our objective function consists of three energy terms: face coverage, data fitting, and model complexity.

Face coverage. The face coverage term is related to the area of a candidate face covered by the faces of the original mesh (see Fig. 8),

\[
E_c = \frac{1}{A(M)} \sum_{i=1}^{N} x_i \cdot (A(f_i) - A(M^c_i))
\]

where \( A(M) \) is the total surface area of the simplified mesh \( M, A(f_i) \) the area of the candidate face \( f_i \), and \( A(M^c_i) \) the face area covered by the original region. This term favours choosing faces with high coverage from the original mesh. Computing the \( A(M) \) term is not possible as the area of the simplified mesh is unknown. However, we expect that the simplified version should conform to the original mesh. This allows us to use the surface area of its bounding box instead, i.e., \( A(M) \approx A(bbox) \).
Fig. 5. By traversing the structure graph (a), the adjacency of the planar segments can be recovered. Thus, the scaffold vertices and edges (b) are computed as the intersections of their supporting planes.

Fig. 6. A 2D arrangement of the projected scaffold edges.

Fig. 7. Proxy mesh. The 2D arrangements of the scaffold edges (left) form a set of candidate faces (right) for the proxy mesh. Notice that errors in the structure graph may cause the production of additional faces (red arrow) or self-intersections (yellow arrow). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 8. Two examples of face coverage. The border of the candidate face are coloured in black, while the faces of the input mesh in yellow. The value below each figure indicates the coverage ratio of each face. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
Data fitting. Apart from the area coverage, we also consider the number of faces from the input mesh covering a candidate face. This is expressed through the data fitting term

\[ E_f = 1 - \frac{1}{|F|} \sum_{i=1}^{N} x_i \cdot \text{st}(f_i), \]  

(2)

where \( |F| \) is the total number of the original faces and \( \text{st}(f_i) \) the number of those faces covering the candidate face. As a consequence, faces with a great amount of supporting faces have a higher chance to be selected in the final solution.

Model complexity. The data-fitting term complies with discontinuities in the original mesh (such as holes). To avoid such gaps in the final model and enforce the creation of large planar regions, we introduce a model complexity term, related to the number of model features (details). These features are represented by edges which are incident to faces from different supporting planes (see Fig. 9). To this end, we define the model complexity term to evaluate the ratio of non-planar edges over the total number of edges in the simplified model

\[ E_m = \frac{1}{|E|} \sum_{e \in E} c(e), \]  

(3)

where \( |E| \) is the total number of edges in the proxy mesh. On the other hand, \( c(e_i) \) is an indicator function, whose value is determined by the configuration of the candidate faces adjacent to an edge and selected in the final optimization solution. If the faces are co-planar, the function has a value of zero (Fig. 9(a)). Otherwise, if the faces are not co-planar forming a sharp edge, the function has a value of one (Fig. 9(b)).

We select the optimal subset of candidate faces to form the simplified model by minimizing the weighted sum of these energies. The complete objective function, along with the hard constraints to ensure that the resulting mesh is both manifold and watertight (meaning that each edge is adjacent to exactly two faces), is given in Eq. (4).

\[
\begin{align*}
\min_x \lambda_f \cdot E_f + \lambda_s \cdot E_s + \lambda_m \cdot E_m \\
\text{s.t.} \quad & \sum_{j \in N(e_i)} x_j = 2 \text{ or } 0, \quad 1 \leq i \leq |E| \\
& x_i \in \{0, 1\}, \quad 1 \leq i \leq N
\end{align*}
\]

(4)

This optimization is bound to produce a simplified, topologically valid representation, despite the various geometric or topological defects in the input MVS mesh (see Fig. 10).

3.4. Implementation details

We have implemented our method in C++ using the CGAL library. Through experimentation, we performed the computation of planarity for our segmentation technique over 3-ring neighbourhoods for both mesh vertices and faces. The distance threshold varied according to both the minimum width of a building component we wish to detect (see Fig. 11) and the respective scale of the input mesh. To eliminate architectural details from the simplification procedure, we used an importance threshold of 1% for all the available models. Finally, the weights of the energy terms in our optimization were those defined by Nan and Wonka (2017), i.e., \( \lambda_f = 0.43, \lambda_s = 0.27, \) and \( \lambda_m = 0.30. \)

4. Results & analysis

We applied our method over a set of building models. The results are shown in Fig. 12 where most of the examples refer to open meshes, with the exceptions of models (d) and (g). Furthermore, we performed polygonization over individual building models, again with the exception of model (a) which contains two separate buildings. The available meshes consist only of planar components in various configurations, with roof superstructures varying from simple, flat roofs [(h), (i)] to more complicated assemblies [(e), (f)]. The level of noise in those models is also variable, from clean [(d), (g)] to more distorted ones [(h), (i)], and dependent on the noise of the original point clouds.

Although the result of our method is always a closed mesh, the input is still allowed to be open which is the case with most building models extracted from urban scenes. To complete the simplified model, our implementation exploits the ground plane of the original urban scene from which the input model was extracted. However, an alternative implementation could allow the user to import such a plane, according to their needs.

We assess the conformity of our simplified version to the original model by computing the Hausdorff distance between the two meshes (Guthe et al., 2005). From Table 1, we observe that the RMSE error is small for both closed and open meshes, which indicates that our simplified versions closely follow the initial building models, especially when the original mesh is clean [(d), (g)]. Our simplification method performs rather well, also when the input model is quite noisy [(h), (i)].

A comparison between model (a) and models (b), (c) reveals that the method can be applied to an urban scene consisting of multiple buildings. Nevertheless, processing each building model individually results in much more detailed models of higher geometric accuracy. This is because the area of building primitives remains constant while their importance changes as the mesh surface area increases. This leads eventually to their exclusion from the simplification process.

Parameters. We have conducted a quantitative analysis of the effect of the parameters on the final results. This analysis shows that ring neighbourhoods of order 3 are more than sufficient for the detection of borders between adjacent planar regions (see Fig. 3). Any lower order is not sensitive enough for this task, while any higher order increases the computational time considerably.

The distance threshold is highly dependent on the scale of the input model. Nevertheless, our experiments have shown that the mean edge length of the input model is a good indicator of the distance threshold that achieves the best results for all the tested data.

Furthermore, the importance threshold of 1% is sufficient for the polygonization of all our tested models. Experimentation reveals that this parameter has a maximum range from 0.1% to 5%, in which it produces simplified meshes conforming adequately to the original structure (see Fig. 13).

As for the weights of the energy terms in the optimization process, a wide range of their values can produce the same results, except for cases where one of the data-fitting or coverage terms is extremely favoured over the other (in a proportion greater than four to one). A small coverage coefficient allows the selection of a larger amount of faces, while a high value of it reduces the candidate faces to only a few (see Fig. 14). In general, the data-fitting coefficient should always be slightly higher since coverage is a much stricter indicator of face validity, thus disregarding most of the candidate set.
Robustness. Even though various defects are present in the original mesh (holes, self-intersections, occlusions etc.), our proposed method is guaranteed to produce a simplified version of the original model.

By altering the distance threshold of our segmentation method, we are able to detect or ignore building components (here depicted with different colours).

Statistics on the simplified meshes. Notice that for this comparison, the polygonal faces of the simplified meshes have been previously triangulated for the visualization purpose.

<table>
<thead>
<tr>
<th>Mesh (Fig. 13)</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
<th>(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td># faces (original)</td>
<td>70,910</td>
<td>13,389</td>
<td>21,454</td>
<td>27,258</td>
<td>6,172</td>
<td>9,923</td>
<td>39,044</td>
<td>39,948</td>
<td>37,269</td>
</tr>
<tr>
<td># faces (simplified)</td>
<td>32</td>
<td>46</td>
<td>110</td>
<td>338</td>
<td>62</td>
<td>36</td>
<td>284</td>
<td>130</td>
<td>100</td>
</tr>
<tr>
<td>Planarity (s)</td>
<td>0.9</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
<td>0.7</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>Segmentation (s)</td>
<td>12.4</td>
<td>1.3</td>
<td>1.4</td>
<td>2.5</td>
<td>0.1</td>
<td>0.3</td>
<td>5.7</td>
<td>23.1</td>
<td>10.9</td>
</tr>
<tr>
<td>Struct. graph (s)</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>Simplification (s)</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.6</td>
<td>0.1</td>
<td>0.1</td>
<td>0.9</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>RMSE (% BBox Diagonal)</td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>0.1</td>
<td>1.0</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Structural accuracy. Certain inconsistencies might be observed between the original models and our simplified versions. These inconsistencies are related to parts of the structure that appear (a) with different geometry in each model or (b) only in one of the two models. The former error (see Fig. 15(a)) is associated with flaws due to segmentation, i.e., the detection of less planar components than the ones necessary to fully approximate a given model. The latter one (see Fig. 15(b)) with the hard constraints imposed in our optimization process to ensure the manifoldness of the final result. Specifically, the hard constraints will include candidate faces in the resulting model regardless of their face coverage, if the manifoldness of the final mesh remains unaffected.

Comparisons. Fig. 16 presents a comparison between our method and other available simplification and polygonization methods. In this comparison, we have simplified a model to have the same number of faces for each method. We observe that the error of our method is smaller than those from all the competing techniques. Additionally, the error on our results is distributed uniformly along the mesh surface contrary to the other methods where the error is located on specific model features. Furthermore, our approach is the only one to produce lightweight models, valid to be used for further applications, despite the topological and geometric defects of the original mesh.

We have also compared the performance of our method to the plane assembly technique PolyFit. We applied both methods to the building block model shown in Fig. 17 for the same number of initially detected planes. Our method took less than two seconds while PolyFit could not finish the optimization within two hours (see Table 2 for more details). This comparison proved that our approach is computationally more efficient than PolyFit.

Limitations.

Our definition of structure is based on the detection of the building primitives, along with their interrelationships, which, in our case, are limited only to adjacency relationships. This requires that both of these elements need to be recovered so that the model structure is completely acquired. This requirement may not be satisfied due to two reasons, both related to mesh segmentation (see Fig. 18):

(a) The set of building primitives is not fully recovered. This may occur when the distance threshold is too big or when the input mesh is “smooth”, meaning that the curvature on the borders of planar regions changes gradually. As a result, certain components cannot be represented with a planar region and therefore, are ignored during the construction of the building scaffold.

(b) The topological relationships, necessary to recover the border of a planar region in the building scaffold (as vertices and edges), are not included in the structure graph. This may occur when the region extends in a limited area of the mesh, i.e., it shares a common border with some of the adjacent regions, smaller than the one required to define a closed shape.
Fig. 12. Simplification results. From left to right: original model, refined segmentation, candidate faces, simplified mesh, and the visualization of the Hausdorff distance defined between the input model and the result.
Fig. 13. The effect of the importance threshold.

Fig. 14. The effect of the energy coefficients.

Fig. 15. Two types of structure inaccuracy.

Table 2
Comparison between our method and PolyFit. The execution of PolyFit was terminated after two hours.

<table>
<thead>
<tr>
<th></th>
<th>#planes</th>
<th>#candidate</th>
<th>#variables</th>
<th>#constraints</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>144</td>
<td>454</td>
<td>2100</td>
<td>804</td>
<td>1.7 s</td>
</tr>
<tr>
<td>PolyFit</td>
<td>144</td>
<td>43655</td>
<td>133701</td>
<td>278183</td>
<td>&gt;2 h</td>
</tr>
</tbody>
</table>
Another limitation of our method is that it is mostly developed for reconstructing individual buildings. If two buildings that are very close or adjacent, it is possible that the resulting volumes intersect, which would be due to elongated and/or unwanted features. Processing these two buildings together would be one way to avoid this (although these buildings could be merged into one instance).

It is also theoretically possible that, for one input model, two or more faces would intersect and be selected for the final model (see as an example the candidate faces where the yellow arrow points in Fig. 7). However, during our tests, we have never encountered such a case. This is because those faces get assigned a low confidence value before the optimization, and are therefore never selected (the data fitting term forbids choosing the ones with a low confidence value). In theory, this means that corners and edges in the resulting simplified model are not adjusted to feature orthogonality, and neither are the façades forced to be vertical. In other words, our method is generic. However, such a regularization step could be applied to our output as a post-processing step.

Applications. Our approach can be included as an individual part of a more general procedure for the reconstruction of entire urban scenes. Utilizing semantic segmentation (Landrieu and Simonovsky, 2018; Zhu et al., 2018), the buildings of a given urban scene can be isolated, simplified separately with our proposed technique and then, recombined with the rest of the scene (as shown in Fig. 1).

Future work. We would like to further refine our means for recovering the structure of the original model, thus improving its robustness. In addition to the planar primitives, we would like to incorporate additional primitive geometries (such as cylinders, spheres, and cones), to handle a wider range of building structures.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

See Algorithm 1.

Algorithm 1 Region growing with k-ring planarity

Input:
  • Triangle Surface Mesh $\mathcal{M}$ with faces $F$
  • $k$-ring Planarity Estimates $(p)$
  • $k$-ring Neighbouring Face Finding Function $\Omega_k(.)$
  • Distance Threshold $d'$

Output: Triangles assigned to segments

Initialize:
  • Regions $(R) \rightarrow \emptyset$ // a list of integers (face indices)
  • Available Faces $(F) \rightarrow (1, 2, ..., m)$ // a list of integers (face indices)

while $(F) \neq \emptyset$ do
  • Current Region: $(R) \rightarrow \emptyset$ // face indices
  • Current Seed: $(s) \rightarrow \emptyset$ // face indices
  • Face with highest planarity $(F) \rightarrow f_{max}(F) \leftarrow \Omega_k(p_{max})$
  • $F' \leftarrow F \setminus f_{max}$
  • Final $k$-ring neighbouring face $(B_c) \rightarrow B_c(f_{max})$
  • Fit plane to Neighbours $plane \leftarrow \text{PCA}(R_c)$
  • while $(S_c) \neq \emptyset$ do
    • $(B_c) \rightarrow \emptyset$ // face indices
    • for $v \in (S_c) \rightarrow (v) \leftarrow \text{vertices of } S_c$
    • if $\forall B \in (F) \text{ and } d(v, B) \leq d'$ then
      • $(B_c) \leftarrow (B_c) \cup (v)$
      • $(V) \leftarrow (V) \setminus (v)$
      • $(S_c) \leftarrow (S_c) \setminus (v)$
    end if
  end for
  • Add current region $(R) \leftarrow (R_c)$
end while
References


