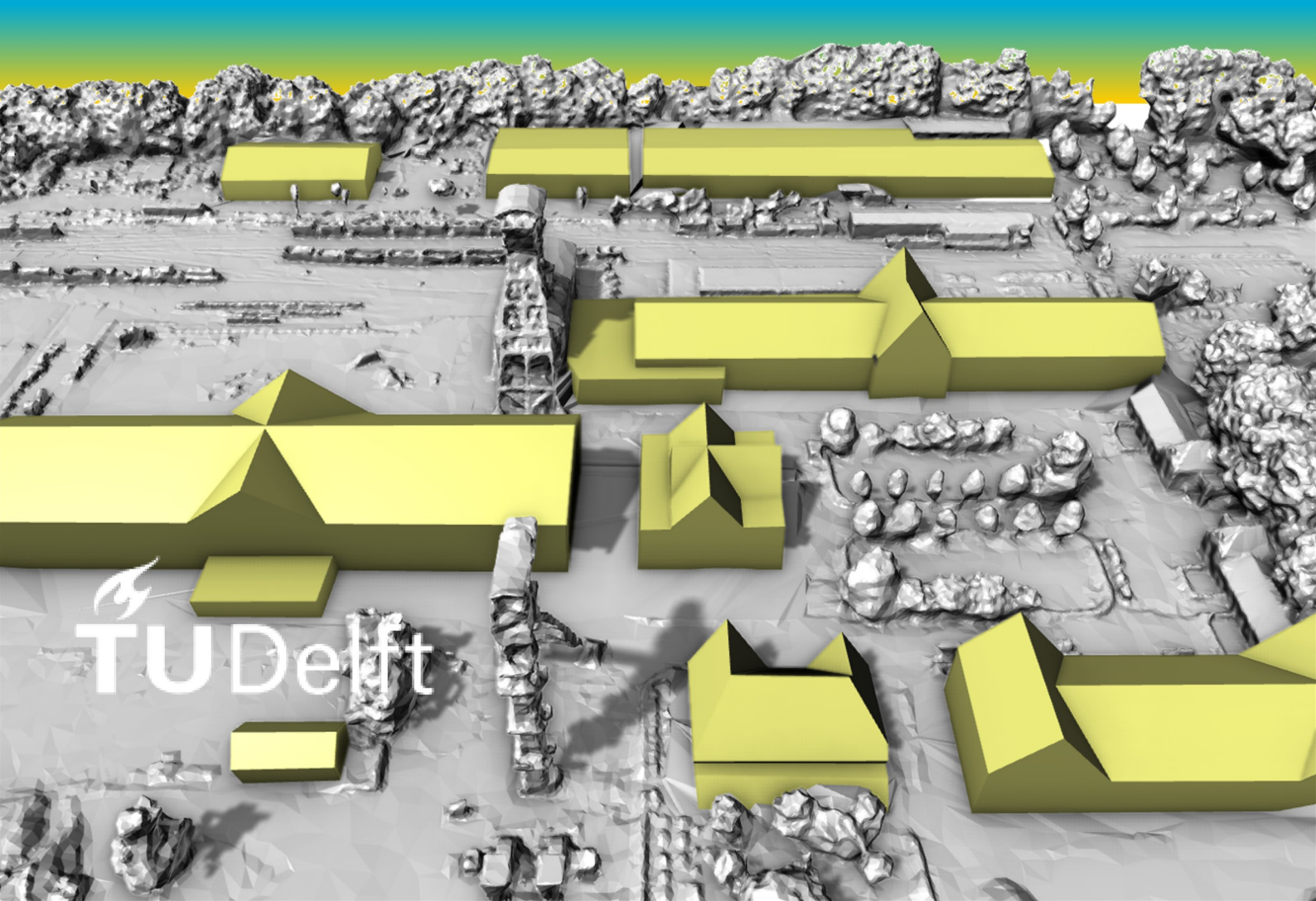
MSc thesis in Geomatics for the Built Environment

Structure-aware Building Mesh Simplification

Vasileios Bouzas

2019



STRUCTURE-AWARE BUILDING MESH SIMPLIFICATION

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Vasileios Bouzas

June 2019

The work in this thesis was made in the:

3D geoinformation group
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors: Dr. Liangliang Nan
Dr. Hugo Ledoux
Co-reader: Dr. Ravi Peters

ABSTRACT

In recent years, there is an ever-increasing demand — both by industry and academia — for 3D spatial information and especially, for 3D building models. One of the ways to acquire such models derives from the combination of massive point clouds with reconstruction techniques, such as Ball Pivoting and Poisson Reconstruction. The result of these techniques is the representation of individual buildings or entire urban scenes in the form of *surface meshes* — data structures consisting of vertices, edges and faces.

Despite their usefulness for visualization purposes, the high complexity of these meshes, along with various geometric and topological flaws, stands as an obstacle to their usage in further applications, such as simulations and urban planning. To address the issue of complexity, the production of lightweight building meshes can be achieved through *mesh simplification*, which reduces the amount of faces used in the original representations. Moreover, simplification methods focus on conforming the resulting mesh to the original one, in order to minimize the differences between the two of them. As a consequence, simple and accurate building models are possible to be acquired, whose geometric and topological validity is yet questionable.

In this thesis, we introduce a novel approach for the simplification of building models, which results into a more compact representation, free of topological defects. The main characteristic of our method is *structure awareness* — namely, the recovery and preservation, for the input mesh, of both its primitives and the interrelationships between them (their configuration in 3D space). This awareness asserts that the resulting mesh closely follows the original and at the same time, dictates the geometric operations needed for its construction in the first place — thus providing accuracy, along with computational efficiency.

Our proposed methodology is divided into three main stages: **(a)** primitive detection via *mesh segmentation*, **(b)** storage of primitive interrelationships in a *structure graph* and **(c)** simplification. In particular, simplification is accomplished here by approximating the primitive borders with a *building scaffold*, out of which a set of *candidate faces* is defined. The selection of faces from the candidate set to form the simplified mesh is achieved through the formulation of a linear binary programming problem, along with certain hard constraints to ensure that this mesh is both *manifold* and *watertight*.

Experimentation reveals that our simplification method is able to produce simpler representations for both *closed* and *open* building meshes, which highly conform to the initial structure and are ready to be used for spatial analysis. Additionally, a fairly good approximation of a given mesh is possible to be obtained within reasonable execution times, regardless of the initial noise level or topological invalidity. Finally, a comparative analysis shows that the accuracy of our method stands in parallel with that of other available simplification techniques.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my first mentor, Dr. Liangliang Nan, for his continuous and constant support during this graduation project. Special thanks also to Dr. Hugo Ledoux for his very helpful suggestions on the project and assistance with all sorts of university practicalities. Finally, many thanks to Dr. Ravi Peters and Dr. Marjolein Spaans for their comments.

During my studies, my family was always here with me, despite the distance between us. Given that, I honestly think that there is not enough space in this thesis to even slightly describe how important that fact was to me. As for my friends, old and new ones, I believe that thanking them is just a waste of my time, since I did warn them from the start what they were getting into — now, why they still consider me as their friend remains a great mystery to me...

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

CGAL	Computational Geometry Algorithms Library
HDS	Halfedge Data Structure
LiDAR	Light Detection And Ranging
LOD	Level Of Detail
MRF	Markov Random Field
MVS	MultiView Stereo
OGC	Open Geospatial Consortium
PCA	Principal Component Analysis
QEM	Quadric Error Metrics
RANSAC	RANdom SAmple Consensus
RMS	Root Mean Square
SAMD	Structure-Aware Mesh Decimation
SfM	Structure from Motion
UAV	Unmanned Aerial Vehicle
VSA	Variational Shape Approximation

1 INTRODUCTION

This chapter begins with a brief discussion on the motivation behind this graduation project, as well as its relation to the field of Geomatics. Based on these facts, the main research questions to be addressed in the project are defined. Finally, a general overview of the thesis is presented in the last section.

1.1 MOTIVATION

In recent decades, there is an ever-increasing demand — both by academia and industry — for 3D spatial information and especially, for 3D City models [Biljecki et al., 2015]. The main reason behind this demand is that 3D data either provides a much more enriched context for some applications — in comparison to the already available 2D data — or makes the conduction of others possible. Applications, which may benefit from the use of 3D data, vary from infrastructure planning, utility management and 3D cadastre to solar potential estimation and visibility analysis (see Figure 1.1).

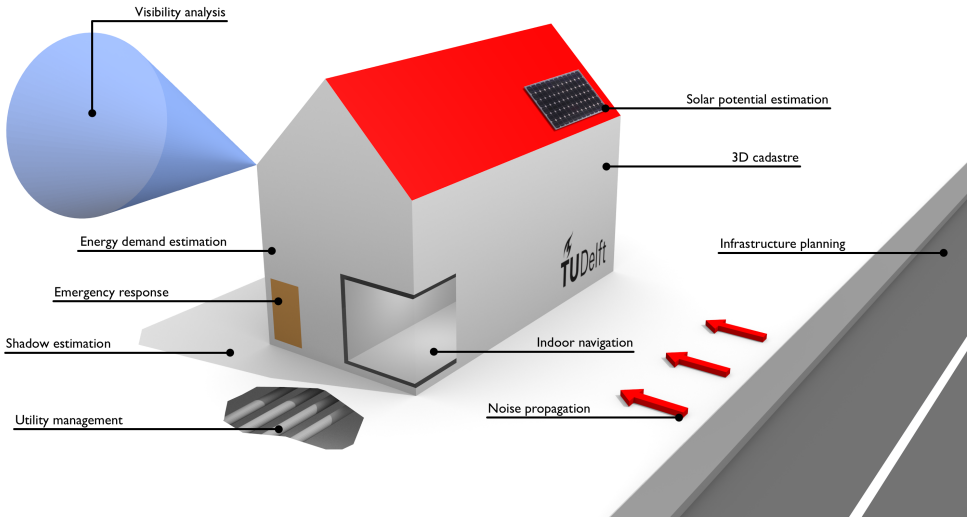


Figure 1.1: Applications of 3D city models [Biljecki et al., 2015]

As a consequence of the above, 3D City models have been an objective of extensive scientific research in all fields closely related to spatial information processing, Geomatics included [Lancelle and Fellner, 2010; Jain et al., 2013; Billen et al., 2014]. In fact, interest in this particular type of spatial information has further increased recently, since advances in modern technology allow the derivation of 3D models from different acquisition techniques. Apart from architectural models and drawings [Donkers et al., 2015; Yin et al., 2009; Lewis and Sequin, 1998], other methods — such as photogrammetry and laser scanning [Süveg, 2003; Haala and Kada, 2010; Veljanovski et al., 2011; Tomljenovic et al., 2015] — can also contribute to the pro-

duction of 3D models, based on the existence of any relevant 2D geoinformation or independently.

This variety of methods and techniques results to the construction of City models with a wide spectrum of different characteristics. These characteristics need to be taken into consideration in order to decide which model type is actually suitable for a certain kind of application. For example, in the case of spatial analyses such as the estimation of the gross volume or insolation of buildings, accuracy is often dependent on the Level Of Detail (LOD) of the model to be used [Biljecki et al., 2018]. The term LOD refers to the level of abstraction of a model, in reference to its real-world counterpart. According to the current standards by the Open Geospatial Consortium (OGC), five different levels of detail can be defined for a given building model (see Figure 1.2):

- LOD0: The 2D footprint of the building
- LOD1: A prismatic solid, resulting from the extrusion of the building footprint in 3D space
- LOD2: A prismatic solid, enhanced with a simplified roof shape
- LOD3: An architecturally detailed model of the outer shell of a building
- LOD4: A detailed model of the building, including indoor features

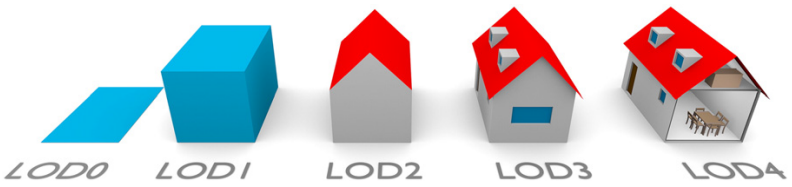


Figure 1.2: Levels of detail [Biljecki et al., 2016]

Since the required LOD highly depends on each individual application, the production of multiple LODs for the same building model is often needed. For the generation of individual buildings or entire urban scenes for different LODs, a lot of techniques are based on the integration of 2D data (such as building footprints) with 3D point clouds [Awrangjeb and Lu, 2014; Commandeur, 2012; Xiong et al., 2014]. Nowadays, it is possible to acquire these clouds either by means of Light Detection And Ranging (LiDAR) or aerial imagery, combined with Structure from Motion (SfM) and MultiView Stereo (MVS) methods [Furukawa and Hernández, 2015].

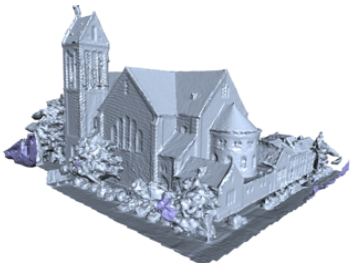
Apart from combining them with 2D spatial information to derive city models, point clouds have also been used separately to reconstruct building models and urban scenes (see Figure 1.3). Various methods have been developed in the last decades for the reconstruction of models from point clouds, two of them being Ball Pivoting [Bernardini et al., 1999] and Poisson Reconstruction [Kazhdan et al., 2006]. The reconstructed result is the representation of the original building in the form of a *surface mesh* (see Figure 1.4). A *surface mesh* is the representation of a surface consisting of vertices, faces and edges [Jonsson, 2016].

Although the quality of these meshes is sufficient for visualization purposes, still it is not enough for other applications, such as urban planning and simulations [Holzmann et al., 2017]. The main reasons for this are the following (see Figure 1.5):

- *Semantic information*: The most critical problem of reconstructed meshes is the lack of semantic information, which is necessary for the estimation of building energy demand, the cadastre or other applications. However, these meshes are so complex (e.g. composed of a high number of faces or inconsistently oriented components) that semantic information cannot be easily added to them; editing is almost impossible.

(a) Aerial imagery

(b) Point cloud



(c) Reconstructed model

(d) With texture

Figure 1.3: Building model reconstruction

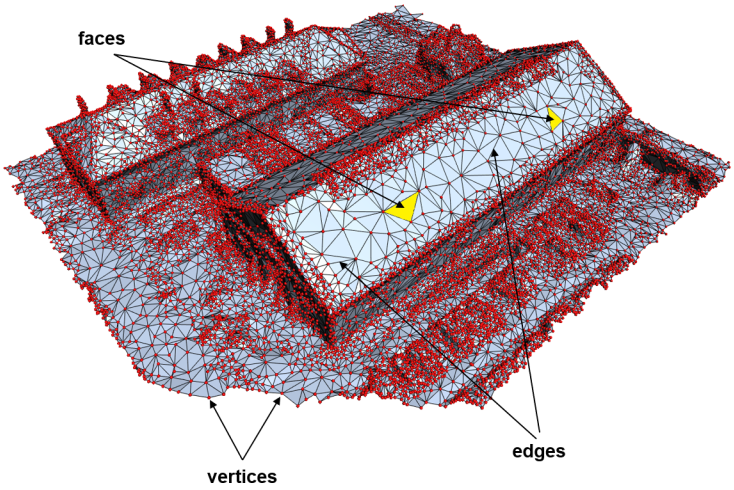
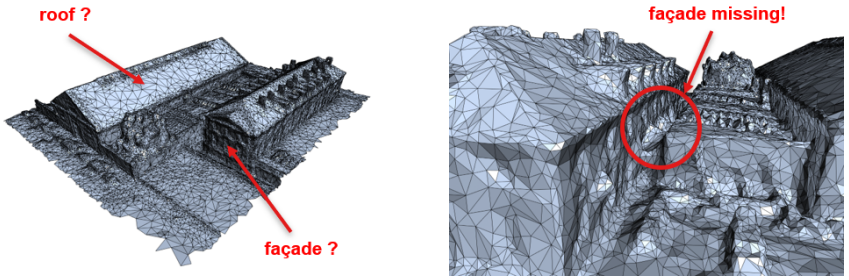


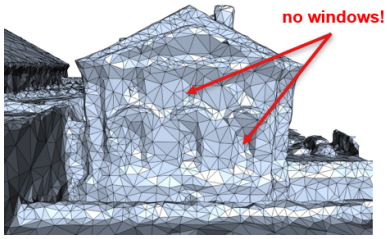
Figure 1.4: Surface mesh

- *Large memory size*: The main concept behind reconstruction techniques is the fitting of a surface over a point cloud. This surface usually consists of a high number of primitives, adding unnecessary complexity and redundant information. Both this complexity and extra information incommode any further processing of the resulting model.
- *Missing information*: There are also cases where parts of a model cannot be reconstructed, due to incompleteness of the original point cloud. A usual cause of this incompleteness is occlusion — a common phenomenon for urban scenes, where the view to parts of a given scene is blocked by high objects, such as buildings or vegetation.
- *Noise / Undesired structures*: Outliers in the original point cloud or flaws in the reconstruction method used can result to defects in the final mesh (e.g. self-intersecting parts or holes).



(a) Semantics

(b) Missing information

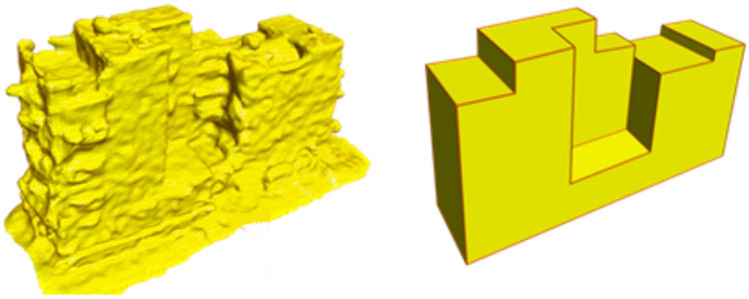


(c) Noise / undesired structures

Figure 1.5: Issues of building reconstruction

To address these problems in building reconstruction, there are mainly two options. The first one is to develop a mesh reconstruction method that deals with them during the creation of the original models. However, the main drawback of this approach is that in many cases, only the reconstructed model is publicly available (e.g. mesh models on Google Earth) and the original data (imagery or point cloud) is inaccessible.

The other option is *mesh simplification*. Surface mesh simplification is defined as the process of reducing the number of faces used for the representation of a given model [The Computational Geometry Algorithms Library (CGAL), 2018]. The result of this process is the production of meshes composed of a few number of simple geometric shapes, thus minimizing both complexity and memory requirements (see Figure 1.6).



(a) Original model

(b) Simplified model

Figure 1.6: Building model simplification [Nan and Wonka, 2017]

In respect to the above, the goal of this project is to develop a methodology that processes the *surface mesh* of a given *building* and transforms it into a *simpler, piecewise-planar* mesh — a mesh with a smaller number of vertices, edges and faces than the original one, which consists only of planar primitives.

1.2 RESEARCH QUESTIONS

Although a simpler presentation is much more efficient from a computational perspective, a dramatic alteration in the characteristics of the original building model (dimensions, volume etc.) will have a profound impact on the accuracy of any analysis. Hence, one of the main requirements for mesh simplification is that the simplified model should closely follow the *structure* of the original. Similar to the work of Mitra et al. [2013], the notion of *structure* is also defined here as the ensemble of the model *primitives* and their *interrelationships*:

- *Primitives*: This term refers to the individual components that compose a building model (see Figure 1.7). In reality, these primitives correspond to parts of the original building, such as walls, roof segments and other architectural details.
- *Interrelationships*: Each primitive is connected to others through *topological* relationships — in other words, their configuration in 3D space. For example, a wall might be adjacent to another but at the same time, it is not associated with the chimneys, lying on the roof.

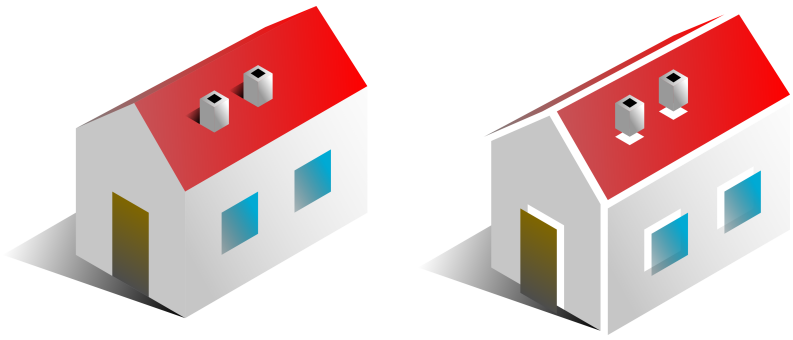


Figure 1.7: Building model structure

Based on the motivation behind this project and the aforementioned definition of structure, the main research question can be defined as follows:

Is it possible to obtain a simpler, compact representation of a building mesh, while preserving the initial structure?

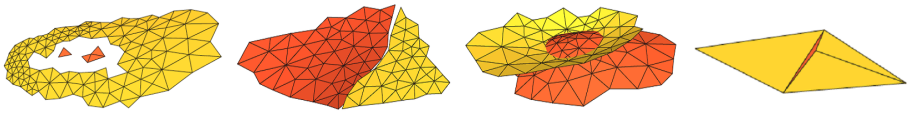
As it has already been mentioned in Section 1.1, reconstructed building models have been used up until now primarily for visualization purposes, due to their complexity. Complexity aside, geometric (Figure 1.8) and topological flaws (Figure 1.9) have also been a hindrance to the use of these models in applications, with simulations being an example of them. The main reason is that simulation software is usually developed on the assumption that the input model is free of these types of inconsistencies. With respect to that, we set as an additional condition for our simplified mesh that it should be free of any geometric and topological defects, so it can be later used for other applications.

All additional requirements can be briefly summarized in the next research sub-question:

- ***Can we ensure that the resulting mesh is free of topological defects?***

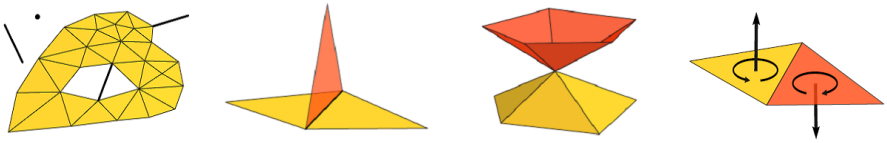
Contrary to other techniques, which might require additional information (images, point clouds etc.), we demand that the only input needed for our proposed methodology should be the mesh model of a given building. Furthermore, this project focuses on simplifying the geometry of a given model, without any use of texture; recovering any semantic information on the model is out of the scope of this research. Finally, our methodology will be based on the assumption that the input mesh comprises

only piecewise-planar objects, thus ignoring the simplification of any objects with different geometry (cones, cylinders etc.).



(a) Hole & islands **(b)** Gap **(c)** (Self-)intersection **(d)** (Near) degeneracy

Figure 1.8: Geometric defects [Attene et al., 2013]



(a) Isolated elements **(b)** Singular edge **(c)** Singular vertex **(d)** Variable orientation

Figure 1.9: Topological defects [Attene et al., 2013]

1.3 THESIS OVERVIEW

The main content of this thesis extends in five chapters. Chapter 2 is an elementary introduction to the scientific research related to this graduation project; in particular, mesh reconstruction and simplification for building models and urban scenes.

In Chapter 3, the methodology proposed to address the research questions of this project are overviewed and each individual processing step is analysed with the use of an example mesh. Further details on the implementation of our methodology in practice are provided in Chapter 4.

In Chapter 5, results are presented from the application of our proposed methodology to other exemplary meshes. These results are used to reflect on the methodology, first by examining them individually and afterwards, by comparing them with the respective ones from other available methods.

The thesis concludes in Chapter 6. There, the degree, in which the research questions have been addressed, is first discussed, followed by the contributions of our proposed methodology to the current state of art. Then, the limitations of our methodology are presented, which serve as the ground for the recommendation of some future work.

2 RELATED WORK

In this chapter, the scientific research related to this graduation project is reviewed. In particular, three different subjects are to be discussed in individuals sections: mesh reconstruction, mesh simplification and building mesh simplification.

2.1 MESH RECONSTRUCTION

As mentioned in Section 1.1, there are basically two approaches to address problems that emerge from building mesh reconstruction, with the use of point clouds. The first one is to develop a methodology that deals with these problems, either during the original reconstruction or in a post-processing step by *regularizing* the model. *Regularization* is the process of enforcing a mesh to conform to certain geometric or topological constraints. An example of such a constraint could be that the vertices belonging to a planar primitive of a given mesh should all lie on the same plane.

According to the above, it is possible to divide the various tecnhiques for building mesh reconstruction into three main categories: **(a)** *building mesh reconstruction*, **(b)** *building mesh regularization* and **(c)** *urban scene reconstruction*.

2.1.1 Building mesh reconstruction

An algorithm for polygonal surface reconstruction from SfM point clouds was introduced by Nan and Wonka [2017]. The algorithm starts with the detection of planar primitives in a given cloud, by means of RANdom SAmple Consensus (RANSAC) [Fischler and Bolles, 1981; Schnabel et al., 2007]. Afterwards, the pairwise intersections of theses planes are computed in order to produce a set of *candidate faces* — polygons that might be used later for the construction of the model. The selection of valid faces to form the resulting model is based on an energy optimization problem (see Figure 2.1).

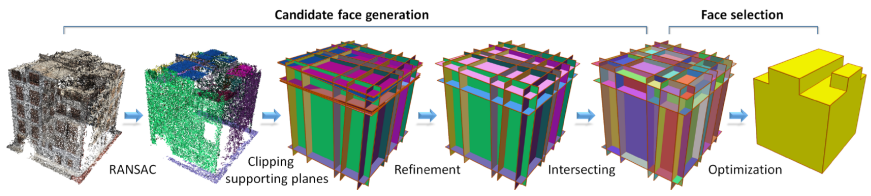


Figure 2.1: Polygonal surface reconstruction from point clouds [Nan and Wonka, 2017]

Holzmann et al. [2017] have also proposed a method for reconstructing plane-based regularized 3D meshes from point clouds. At first, planes are detected with RANSAC in the input cloud and the points are projected on them to minimize noise. After that, the cloud is subdivided into tetrahedra, which are labelled with an energy minimization problem as either inside or outside of the surface to be reconstructed. Finally, the surface mesh is computed as the interface between sets of inside and outside tetrahedra (see Figure 2.2).

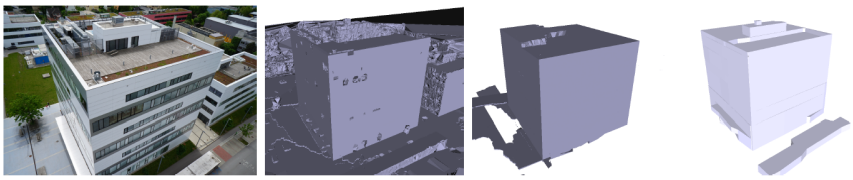


Figure 2.2: Plane-based surface regularization [Holzmann et al., 2017]

On the other hand, Li et al. [2016] have developed a method for the reconstruction of building models, using point clouds generated with Unmanned Aerial Vehicle (UAV) images. In this approach, the SfM input cloud is first segmented into clusters, representing objects of three classes: *ground*, *building* and *tree*. The clusters corresponding to the *building* class are then processed for modelling individual structures. Specifically, a set of contours describing the roof structure of buildings are extracted and then, regularized with the Douglas-Peucker algorithm. The reconstructed model is finally produced by extruding these contours in the third dimension, with the use of elevation information from the cloud (see Figure 2.3).

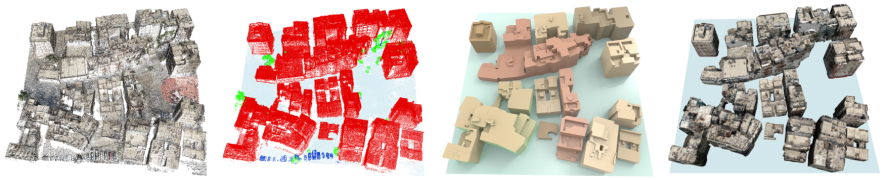


Figure 2.3: Building model reconstruction from UAV images [Li et al., 2016]

Another method for addressing issues of MVS meshes along the reconstruction step was proposed by Bódis-Szomorú et al. [2015]. Its main objective is the reconstruction of 3D meshes, by integrating a given SfM point cloud with single-view images. These images are part of the imagery used to create the cloud in the first place. In the first step, the cloud is decomposed in 3D point clusters in order to identify the individual components. Afterwards, 2D meshes are derived from the imagery, by means of Delaunay triangulation. To add the third dimension to these 2D meshes, an energy optimization problem is formulated, based on the relevant information from the 3D point clusters. A general overview of the methodology can be seen in Figure 2.4.

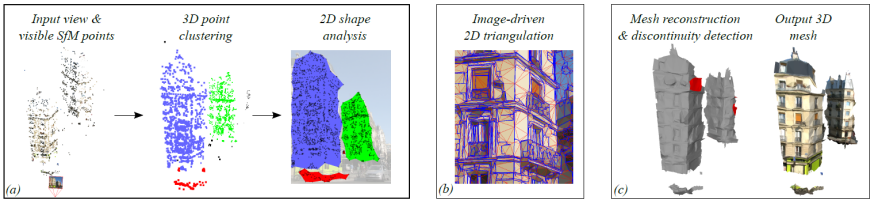


Figure 2.4: Surface reconstruction from single-view Images [Bódis-Szomorú et al., 2015]

2.1.2 Building mesh regularization

Contrary to the above, regularization methods focus on the reconstructed mesh, rather than the reconstruction process itself. An example of such a method was introduced by Jonsson [2016], where the input mesh is first decomposed into planar segments via the formulation of a Markov Random Field (MRF) problem. After that, the segmentation is gradually refined by merging different segments with a defined set of criteria. At the end, the mesh is regularized by projecting the vertices on their respective planar segment (see Figure 2.5)

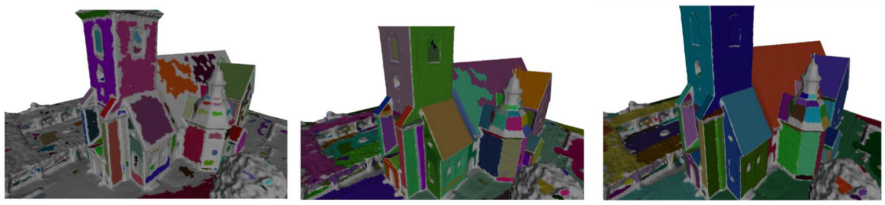


Figure 2.5: Detection of planar regions in MVS meshes [Jonsson, 2016]

Similar to the aforementioned methodology, Wang et al. [2016] have also developed an algorithm for regularization of MVS meshes after reconstruction. In this case, the original imagery is used to recover the sharp features of the given model with an edge detection algorithm. These edges are the basis for the construction of a *scaffold* that provides information about the *structure* of the original object. The regularized result is achieved by enforcing the reconstructed model to conform to the scaffold (see Figure 2.6).

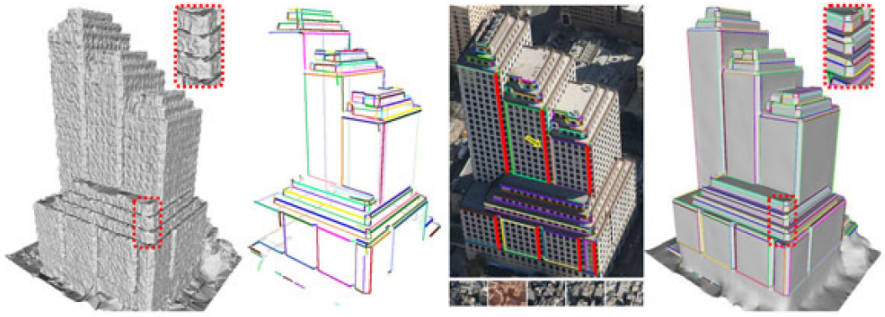


Figure 2.6: Image-based building regularization [Wang et al., 2016]

Except for the original imagery and point cloud, additional data might be used to regularize the reconstructed model. Kelly et al. [2017] have developed such a method, based on complementary data. In particular, their method exploits three different, available data sources: the reconstructed surface mesh of a building, the street-level imagery of its facade and its 2D footprint. By combining these sources, information describing the building geometry is extracted in three different forms: a set of planar segments representing parts of the facade, a collection of line segments forming profiles of the building and a group of edges consisting the ground plane. The final model is produced by fusing these forms via optimization (see Figure 2.7).

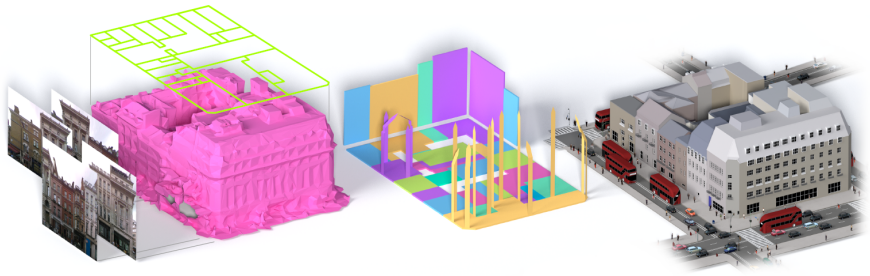


Figure 2.7: Large-scale structured urban reconstruction [Kelly et al., 2017]

2.1.3 Urban scene reconstruction

Different methods have also been developed to address the reconstruction of entire urban scenes for different LODs. Verdie et al. [2015] have demonstrated a method, where the only input required is the reconstructed scene. At first, the given scene is

semantically segmented via MRF into four classes: *ground, tree, facade* and *roof*. For all LODs, ground is represented by a Delaunay triangulation. For LOD2 and lower levels, buildings are approximated by detecting, regularizing and intersecting the planar components from which they are defined.

The addition of architectural details (windows, chimneys etc.) for LOD3 building models is based on two concepts: the construction of a *height map* and the process of *iconization*. A *height map* is the depiction of how far two given meshes are from each other — in literature, this distance is usually referred to as *Hausdorff distance*. On the other hand, *iconization* depends on the definition of a set of geometric objects (*icons*), whose dimensions are encoded into parameters. Using a *height map*, objects, not included in the corresponding LOD2 approximation, are detected in the original model. Then, the *icon*, whose geometry best describes the detected object, is adjusted and then, added to the LOD2 model (see Figure 2.8).

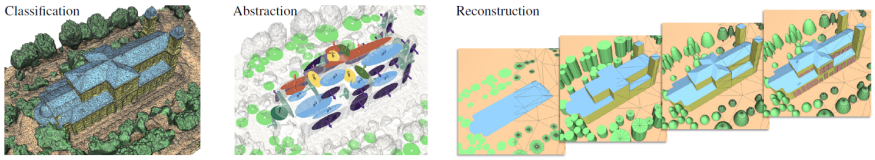


Figure 2.8: LOD generation for urban scenes [Verdie et al., 2015]

Zhu et al. [2018] followed a similar approach for the reconstruction of large urban scenes, for levels between LOD0 and LOD2. In the beginning, the semantic segmentation of the input scene is achieved by applying MRF on orthophotos, initially used to reconstruct the scene itself. Similar to Verdie et al., the scene is decomposed again into four different classes: *ground, grass, tree* and *building*. Each connected *building* is then isolated from the rest of the scene and modelled separately. Assuming that the building walls can be represented as vertical planar segments, *candidate* roof segments are detected both on the reconstructed model and the orthophotos. Finally, to model the roof, valid faces are selected via MRF formulation on a *dual graph* — a graph where each vertex corresponds to a roof segment and each edge connects two adjacent segments (see Figure 2.9).

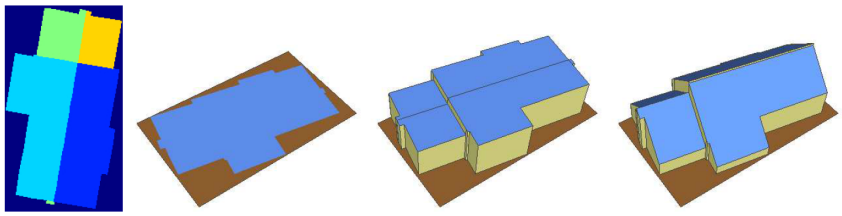


Figure 2.9: Urban scene modelling from MVS meshes [Zhu et al., 2018]

The aforementioned techniques, along with all others described in this section, allow the reconstruction of regularized MVS building models and urban scenes, the resulting meshes might still consist of a high number of faces, making their use in applications difficult. Moreover, some of them may also require additional data sources (building footprints, imagery etc.), when, in most cases, only the reconstructed mesh is accessible. Thus, to achieve a simpler, more compact representation for the same building, based only on its MVS mesh, simplification is the other solution available.

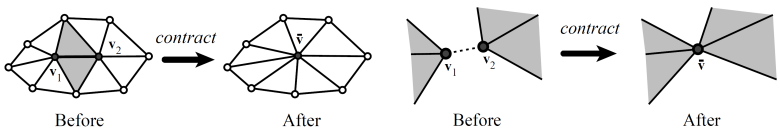
2.2 MESH SIMPLIFICATION

Except for mesh reconstruction, *mesh simplification* can be also applied to address the various issues in reconstructed models, as those discussed in Section 1.1. As mentioned also in the same section, *simplification* is the process of reducing the number of faces for a given mesh, while preserving its geometric, topological or other properties.

Up until now, simplification methods have been mostly applied to produce lightweight versions for any kind of model (building or not), mainly for visualization or animation purposes. In literature, the results of mesh reconstruction techniques — like the ones described in Section 2.1 — are often compared to three main simplification approaches: Quadric Error Metrics (QEM), Structure-Aware Mesh Decimation (SAMD) and Variational Shape Approximation (VSA). These two approaches will be discussed in this section.

2.2.1 Quadric error metrics

QEM is one of the most commonly used methods for simplification of polygonal models. It was first introduced by Garland and Heckbert in 1997. This method is based on pair contraction — namely, the selection and merge of vertex pairs. These pairs are formed, either by two vertices connected with an edge or in a distance smaller than a predefined threshold. In the former case, faces in the original mesh become degenerate and are removed during contraction. In the latter, the pair contraction forces two originally unrelated regions of the input mesh to be connected (see Figure 2.10).



(a) Edge contraction **(b)** Non-edge contraction

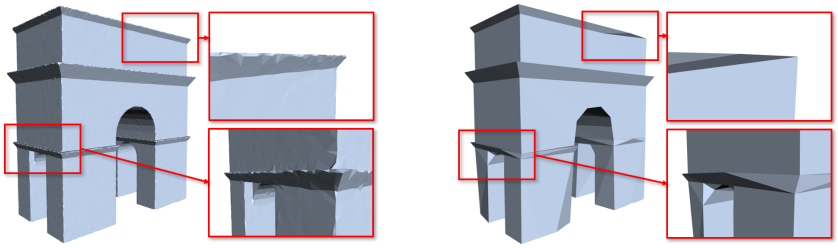
Figure 2.10: Vertex pair contraction [Garland and Heckbert, 1997]

Furthermore, each vertex is associated with a cost value in order to decide which vertex pairs should be contracted along the simplification process. This value is derived, by considering the vertex as the intersection of all its incident triangles and their corresponding planes. With the use of the parameters describing the incident planes, a matrix is formed, which approximates the error of a vertex and thus, its cost.

Based on the above, the main steps of QEM can be summarized as follows:

- Vertex pairs are detected in the original mesh, either by edge connection or a distance threshold
- The error matrix (cost) of each vertex is calculated and thus, the error of all vertex pairs
- The pair with the minimum cost is replaced by a new vertex and the costs of all vertices and pairs are re-calculated
- The whole process is repeated till the number of mesh simplexes is reduced, according to a threshold defined by the user

An example from the application of this method to a building model is shown in Figure 2.11.



(a) Original (27,258 faces) **(b)** Simplified (250 faces)

Figure 2.11: Quadric error metrics (QEM)

2.2.2 Structure-aware mesh decimation

Although QEM is useful for decreasing the initial complexity of a surface mesh, still some of its characteristics, needed to be preserved afterwards (such as dimensions or volume), may be severely altered, depending on the degree of simplification. With respect to these problems, Salinas et al. [2015] further extended the aforementioned algorithm to handle cases of oversimplification, by contributing the following elements:

- *Structure awareness*: To preserve the initial structure, the planar segments composing the model are detected by means of mesh segmentation techniques, such as region growing or RANSAC. Information on the primitives and their exact configuration is stored in a *graph of proxies*, where every primitive is represented with a vertex (see Figure 2.12). Additionally, two vertices of the graph are connected with an edge, if their corresponding primitives are adjacent to each other. Based on this graph, the simplification is then conducted in iterations; after each one, the result is compared with the graph, in order to verify that the number of primitives and their interrelationships are preserved. If not, the simplification process is terminated.

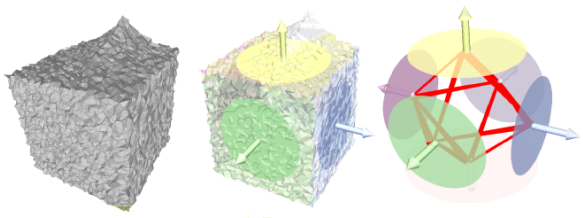
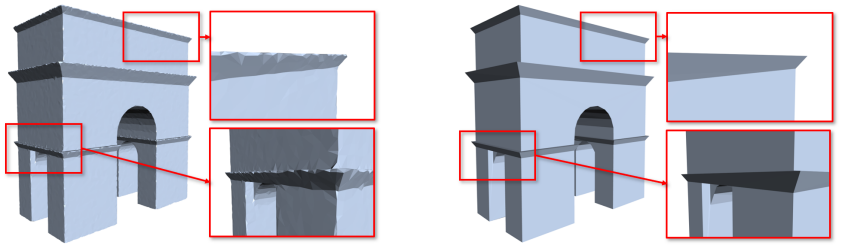


Figure 2.12: Graph of proxies [Salinas et al., 2015]

- *Boundary preservation*: Contrary to the approach of Garland and Heckbert, the method of Salinas et al. is based purely on edge, instead of pair, decimation, while the edge error is approximated by two individual matrices. The first defines the error of the triangles adjacent to a given edge, while the second the boundary error of the planar segment to which the edge belongs. In this way, the initial boundary of the model remains as intact as possible during simplification.

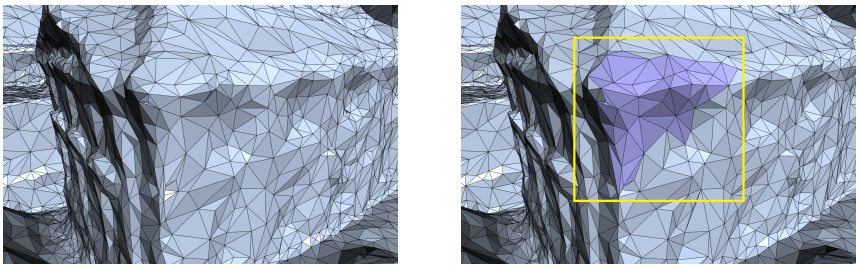
Despite the improvement in the preservation of the initial structure by the method of Salinas et al. (see Figure 2.13), some topological flaws may still be observed in the simplified model:

- *Non-consistent orientation*: For a surface model to be valid, all of its faces must have the same orientation; in other words, they must be defined by a sequence of vertices, either in a clockwise or counter-clockwise order, when viewed from the outside. Depending on the level of simplification, some wrongly oriented faces may still appear in the resulting mesh (see Figure 2.14).



(a) Original (27,258 faces) **(b)** Simplified (250 faces)

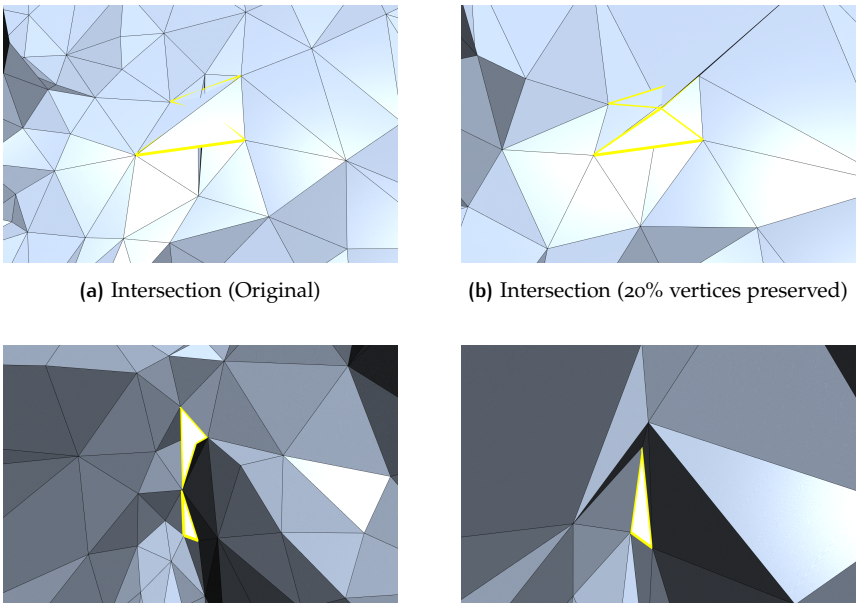
Figure 2.13: Structure-aware mesh decimation (SAMD)



(a) Original **(b)** Simplified (80% vertices preserved)

Figure 2.14: Non-consistent orientation

- *Topological defects*: Inconsistencies in the topology of the original mesh, such as self-intersecting parts or holes, are preserved in various simplification stages (see Figure 2.15).



(a) Intersection (Original) **(b)** Intersection (20% vertices preserved)

(c) Hole (Original) **(d)** Hole (20% vertices preserved)

Figure 2.15: Topological defects

2.2.3 Variational shape approximation

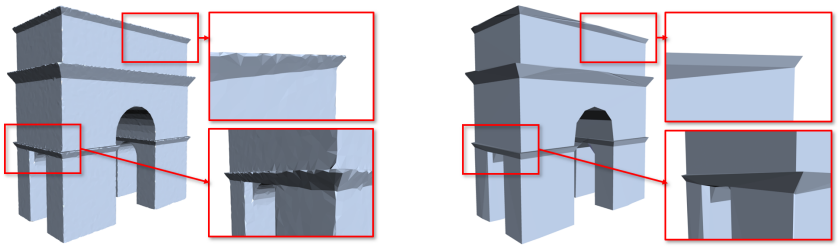
VSA has also become popular as a simplification technique, since it was first proposed by Cohen-Steiner et al. in 2004. Similar to other methods, this one also depends on first decomposing the initial mesh into planar segments. These segments are represented by their supporting plane and in particular, its origin (a point in 3D

space) and its normal (a 3D vector defining the orientation of the plane). Both the origin and the normal can be computed with the application of Principal Component Analysis (PCA) on the vertices belonging to each segment.

In a following step, the segments are approximated by a shape (*proxy*) in order to construct the simplified mesh. As a basic requirement for any simplification method, this set of proxies should be defined in a way that minimizes *distortion* — the difference between the original and the simplified model. To quantify the distortion for each proxy, several metrics have been formulated (like the error quadrics used in QEM), some of which are members of the so-called $\mathcal{L}$-metric family — the *Hausdorff distance* (mentioned in Section 2.1) is one of these metrics. Out of this family, the $\mathcal{L}^2$ metric is the one most commonly used, recording the difference between a given proxy and the projection of the original segment on that proxy.

However, VSA introduced a new $\mathcal{L}^{2,1}$ metric instead. This metric stands as an indicator of the difference between the normal (orientation) of a mesh segment and its corresponding proxy. This change in metrics can be justified, based on two different facts. First, curvature variations can be detected easily, since higher curvature translates into higher variation in local orientation. Furthermore, the amount of computations is significantly decreased, due to the fact that only the normals of segments are now required to evaluate the proxy error. So, rather than relying on PCA which is computationally expensive, the segment normals are computed by averaging the normals of the segment faces.

With the help of the $\mathcal{L}^{2,1}$ metric, the initial segmentation of the input mesh is iteratively optimized till the overall distortion is minimum. The result of applying VSA in a given mesh is shown in Figure 2.16. In general, this simplification method produces better results in comparison to approaches such as QEM, which rely on simplex removal (either vertices or edges). Nevertheless, the use of an iterative process acts as a hindrance to its scalability, since the required processing time might be three to twenty times longer [Cohen-Steiner et al., 2004].



(a) Original (27,258 faces) **(b)** Simplified (336 faces)

Figure 2.16: Variational shape approximation

2.3 BUILDING MESH SIMPLIFICATION

The various methods, discussed in Sections 2.1 and 2.2, have as their main objective either the regularization of MVS building meshes or the simplification of meshes, in general. Nonetheless, various issues may occur by their application for the production of building models.

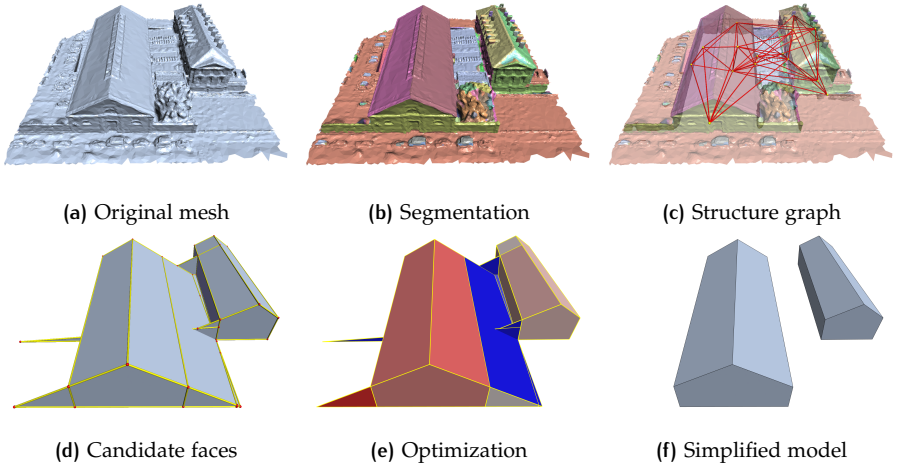
With respect to reconstruction methods, some of them result in a regularized version of the original model which still comprises a high number of faces, thus maintaining the initial unnecessary information and complexity. Others may require additional data (building footprints, imagery etc.), while this type of information is not always accessible.

On the other hand, simplification approaches usually focus on the production of lightweight meshes for general purposes (such as visualization and animation), without paying attention to structure preservation. Especially in cases of oversimplification, where a concise representation with fairly low complexity is required, alterations in structure might be significant. Furthermore, the resulting building models may not be usable for other applications (spatial analyses, simulations etc.), mainly due to geometric and topological distortions.

In this graduation project, we aim to the formulation of a methodology that produces a simpler and more compact representation for a building model, while preserving its initial structure as much as possible. At the same time, the resulting model should be ready to be used for spatial analyses and not for visualization alone. Finally, our approach should be also computationally efficient, allowing its integration in techniques for the reconstruction of 3D City models.

3 METHODOLOGY

The content of this chapter is the analysis of the methodology developed to address the research questions of this graduation project (see Figure 3.1). Based on our definition of *structure*, this methodology is divided into three main steps: **(a)** detection of primitives via *segmentation*, **(b)** storage of primitive interrelationships in a *structure graph* and **(c)** structure-aware *simplification*. The whole pipeline will be presented for an example model, where each step is to be discussed in separate sections.



(a) Original mesh **(b)** Segmentation **(c)** Structure graph

(d) Candidate faces **(e)** Optimization **(f)** Simplified model

Figure 3.1: Pipeline: The input MVS building mesh **(a)** is first decomposed into planar primitives **(b)** and its structure is defined **(c)**. Based on this definition, a set of *candidate faces* is produced **(d)**, which, by means of optimization **(e)**, is used to construct the simplified model **(f)**.

3.1 SEGMENTATION

In the first step of our methodology, the planar components composing the input mesh are identified with a new region growing algorithm, developed for the purposes of this project and based on related work.

3.1.1 Planarity

As already mentioned in Sections 1.1 and 1.2, our methodology is based on the assumption that the building mesh to be simplified is composed of planar components alone. This assumption is introduced mainly for reasons of simplicity and can be considered relatively valid for the majority of building types, encountered in a typical urban environment. However and contrary to some reconstruction methods, our methodology is not constrained by the *Manhattan World assumption* [Holzmann et al., 2017], which states that each planar component needs to be also parallel to one of the Cartesian axes. The main reason for this is that the embrace of such an extreme approach, which further simplifies the problem, eventually prevents the

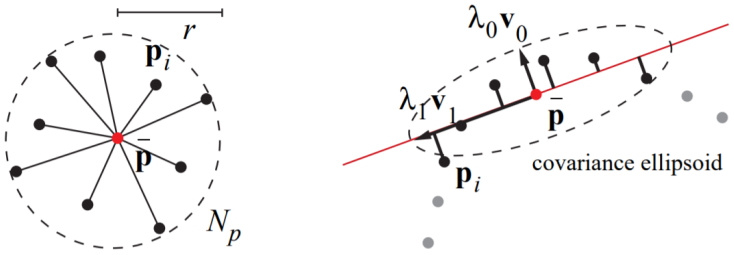
recovery of roof superstructures with inclined segments, often appearing in various models.

Based on the above, our simplification process starts with identifying planar components in the given mesh via *mesh segmentation*. As stated by Wu et al. [2017], the term *segmentation* refers to the partition of a mesh into meaningful subparts. Of course, what is considered as meaningful may differ from case to case. For our methodology, we characterize as significant segments those planar regions that correspond to three different classes of the original building parts: *floor*, *facade* and *roof*. Any architectural details, such as windows or chimneys, are ignored. Due to the limited resolution of MVS meshes, these objects are usually represented by a small amount of faces and subsequently, cannot be approximated with a set of planar components [Verdie et al., 2015].

Before proceeding to the segmentation itself, it is first needed to establish a parameter that stands as an indicator of *"flatness"*. This parameter should sufficiently describe the mesh curvature at a local level, thus allowing the detection of curvature variations which appear on the border of neighbouring planar segments. In this way, the boundaries of individuals segments can be identified, as well as the mesh simplexes (vertices, edges and faces) that belong to them.

In mesh segmentation, two indicators of "flatness" are most commonly used, which are, in fact, equivalent: *curvature* and *planarity* [Wu et al., 2017]. By definition, *curvature* is equal to 0 for a straight line (or a plane in 3D space), while it approximates 1 for a circle (or a sphere in 3D space) of infinitesimal radius — for the *planarity*, the values are exactly opposite in each case. Since these indicators basically express the same notion (the degree of fitting a plane to a given set of points), the selection between the two of them becomes trivial. In our case, *planarity* was chosen arbitrarily as the basis for segmentation.

With that given, it is also necessary now to establish a method for computing planarity. Either planarity or curvature, in meshes or point clouds, are usually computed by means of PCA [Pauly et al., 2002]. This method can be applied to a point cloud (or a mesh) by first defining a *local neighbourhood* for each point (or vertex). Then, the centroid of each neighbourhood and the position vectors of all points belonging to the neighbourhood are computed. Finally, a covariance matrix is formed, based on these position vectors, and its *eigenvalues* are calculated. For 3D space, this matrix has three unique eigenvalues, often denoted in literature as λ_0, λ_1 and λ_2 in *ascending* order (see Figure 3.2).



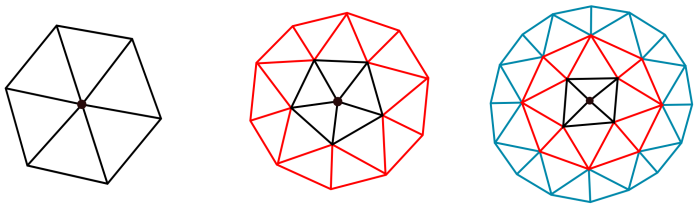
(a) Local neighbourhood **(b)** Covariance ellipsoid

Figure 3.2: Principal component analysis [Pauly et al., 2002]

Computation of planarity (or curvature) can be accomplished using these eigenvalues. There are different formulae and approximations for this task and for the purposes of our methodology, the planarity p of a given mesh vertex is defined with Equation 3.1:

$$p = 1 - \frac{\lambda_0}{\lambda_1},\ p \in [0, 1] \tag{3.1}$$

Furthermore, the concept of the *local neighbourhood* around a vertex also needs to be defined explicitly. In the case of a mesh, this concept can be related with the notion of the *k-ring neighbourhoods* [Gatzke and Grimm, 2006], both for vertices and faces. For example, the *1-ring neighbourhood* of a vertex consists of all vertices *adjacent* to it — in other words, those which are connected with it through an edge. This notion can be further extended to a *2-ring neighbourhood* (which also includes the vertices adjacent to those forming the 1-ring neighbourhood) and so forth (see Figure 3.3).



(a) 1-ring **(b)** 2-ring **(c)** 3-ring

Figure 3.3: *k*-ring neighbourhoods

It is possible to recover adjacent vertices by exploiting the Halfedge Data Structure (HDS), a topological structure commonly used for mesh representation [Campagna et al., 1998]. The main idea behind this representation is the concept of the *halfedge*, an oriented edge connecting two vertices (*originating* and *terminating* vertex), along with its double (*opposite edge*). For each halfedge, additional information is stored about the halfedge terminating at the originating vertex (*previous halfedge*) and the one originating from the terminating vertex (*next halfedge*). Every halfedge is also adjacent to a unique mesh face, thus establishing complete connectivity between all types of mesh simplexes (see Figure 3.4).

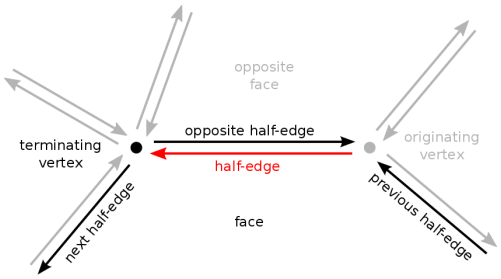
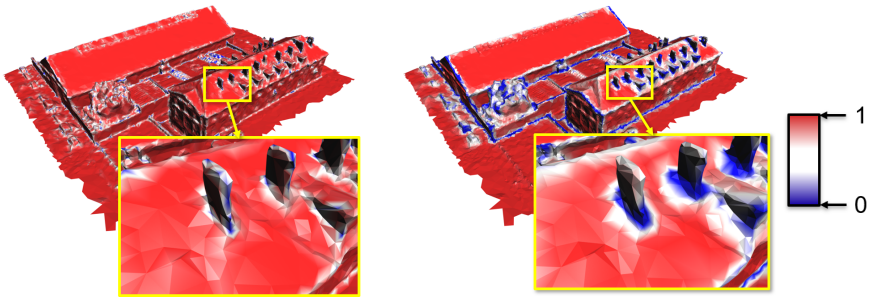


Figure 3.4: Halfedge data structure [The Point Cloud Library (PCL), 2018]

With the above, a pipeline is provided for the computation of planarity on all vertices of a given mesh and for different *k*-ring neighbourhoods, if needed (see Figure 3.5). By increasing the ring order and thus enlarging the neighbourhood of each vertex, "flatness" can be estimated for extended mesh regions. As a consequence of that, the borders between neighbouring planar regions are further enhanced, but the processing time also increases dramatically, since PCA is computationally expensive in general.

3.1.2 Region growing

Having established a method to estimate the mesh "flatness" at a local level, we now proceed to the analysis of the segmentation process itself, of which planarity is the cornerstone. As mentioned before, the purpose behind this segmentation process is the detection of planar regions in the original mesh, which serve later as the basis for the definition of the structure and simplification.

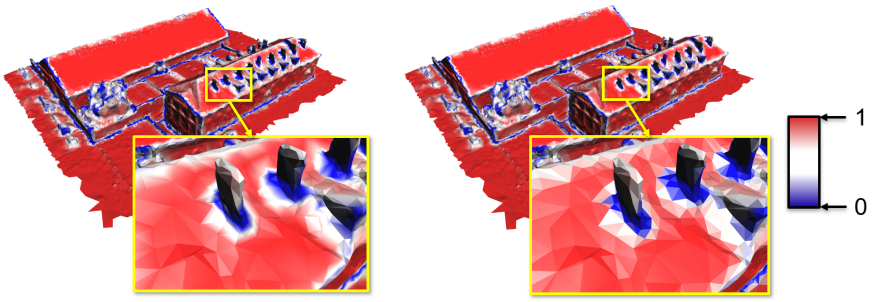


(a) 1-ring planarity **(b)** 3-ring planarity

Figure 3.5: Planarity: In this figure, vertices of low planarity (close to 0) are coloured in blue, while those of high planarity (close to 1) in red.

The first choice needed to be taken is the exact type of segmentation method to be used. One of the most popular group of segmentation techniques is the *region growing* algorithmic family [Wu et al., 2017; Fang et al., 2018; Vieira and Shimada, 2005]. These techniques are founded on the selection of mesh simplexes (either vertices or faces) which are referred to as *seeds*. These seeds are the basis for the development of *regions* — groups of simplexes that share some common characteristics. If a simplex — of the same type as of the seed — conforms to the criteria of a specific region, it is included in it and subsequently, each region *grows* till no more simplexes can be added to it.

In our methodology, it is decided that segmentation is to be performed over the mesh faces, not the vertices. The main reason behind this decision is that borders of planar regions comprise vertices and edges, thus faces are the only simplexes able to be classified definitely to one region or another. However, this approach causes a problem; since planarity is computed for *vertices* as mentioned in subsection 3.1.1, it is needed to transfer this property to the faces of the mesh in order to perform segmentation. To solve this inconsistency, it is chosen that each face should inherit as planarity the average planarity of its vertices. A comparison between vertex and face planarity, following this approach, is shown in Figure 3.6.

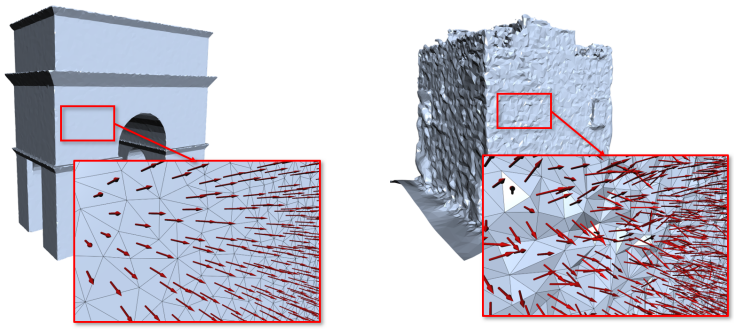


(a) Vertex planarity **(b)** Face planarity

Figure 3.6: Vertex and face planarity: As in Figure 3.6, simplexes of low planarity are coloured in blue, while those of high planarity in red (3-ring neighbourhood). Notice that any difference between the two states is minor and does not significantly alter the result.

In accordance to the above, our segmentation process starts with first detecting the face with the highest planarity. This choice is made due to the fact that the "flatness" of its surroundings indicates that this face is probably located at the centre of a planar region we wish to detect. Using that face as a *seed*, we then check its neighbouring faces in order to further expand the new region we initialized.

To decide if a face is to be included or not in a growing region, we need to define a set of criteria. A criterion that could be possibly used for this purpose is orientation, which, for a mesh face, is related to its *normal*. If the angle between the normal of a face and another normal — which stands as reference — is below a predefined threshold, then the face is included in the currently expanding region. However, the main disadvantage of this is that the segmentation becomes highly sensitive to orientation variations. While these variations are normally found on mesh regions with high curvature (borders between planar regions), they may also appear inside planar regions when the mesh is "noisy" (see Figure 3.7).



(a) "Clean" model **(b)** "Noisy" model

Figure 3.7: Face normals: Here are shown two models, one "clean" and one "noisy", and in each one, the face normals of a fairly planar region. While the face orientation is relatively consistent in **(a)**, instead it is highly irregular in **(b)**.

Instead of the orientation, the criterion used in our segmentation technique is the distance of the face vertices from the reference plane of each region (*supporting plane*). If the distance of *all* the vertices from the supporting plane is lower than a predefined threshold, then the face becomes a member of the respective region. For this plane to be defined, both an origin point and a normal vector are needed. Both of them can be computed, again by performing PCA on the face vertices currently belonging to a region. The centroid of these vertices is appointed as the origin of the plane, while the *eigenvector* corresponding to the smallest eigenvalue λ_0 of the covariance matrix stands as a good approximation of the plane normal.

To construct a reference plane when a region is first initialized (since the region consists of only one face in the beginning), the k-ring neighbours of the highest planarity face (*seed*) are collected, again with HDS. This initial plane should be representative of the supporting plane and at the same time, correspond to the estimation of planarity, performed during the pre-processing step of the segmentation. For these reasons, the order of the k-ring, used for the plane definition, is set equal to the one used for the planarity computation. After calculating the initial plane, the 1-ring neighbouring faces of the seed and of every other face, also assigned to the region, are checked till no more members can be added. As new faces are being added to the current region, the plane is regularly updated with better approximations.

The result of applying our segmentation technique in the example mesh is shown in Figure 3.8, while the complete pipeline of our segmentation algorithm can be summarized as follows (for more details, see Algorithm 1 in the Appendix A):

1. For all mesh vertices, their k-ring planarity is computed and then, inherited to all faces.
2. The face of the highest planarity (*seed*) is identified and with it, the first region is initialized.
3. The k-ring neighbours of the seed are collected and a reference plane is constructed via PCA.

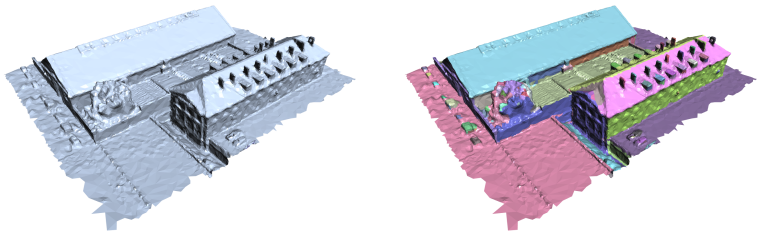
4. The 1-ring neighbouring faces of the seed are checked. If *all* the vertices of a face are within the distance threshold from the reference plane, this face is *marked* as a member of this region.

5. Using all the faces currently belonging to the region, the reference plane is updated.

6. The steps **(3)-(4)** are repeated again and again, for all the newly appointed members of the region. The expansion is terminated as soon as no more faces can be added.

7. As the search for a region is finalized, a new region is initialized by repeating the steps **(1)-(5)** to the remaining faces of the mesh.

8. The segmentation process stops when all mesh faces have been finally assigned to a region.



(a) Original model **(b)** Segmented model

Figure 3.8: Segmentation: In this figure, the planar regions of the example mesh, identified by our segmentation method, are shown in different colours.

3.2 STRUCTURE GRAPH

Having identified the planar components, out of which the input mesh is composed, we proceed to the definition of the model structure. In our methodology, the notion of structure is closely associated with a graph, to which we refer from now on by the name *"structure graph"*.

3.2.1 Refinement

Using our segmentation technique, developed for the purposes of this graduation project and discussed in Section 3.1, the planar components, out of which the input mesh is composed, are detected. The next step should be to define the *structure* of the mesh, since our simplification method aims to a more compact and simpler representation, while preserving this characteristic. However, as mentioned in Section 1.1, our notion of "structure" is not only related to the model *primitives*, which are approximated here by the planar regions of our mesh segmentation. More than that, there is also a need for the detection of their *interrelationships* — in other words, the topological relations between these primitives.

The main topological relationship, on which we primarily focus in the context of this project, is that of the *adjacency*. With respect to that, the task of recovering interrelationships between the model primitives translates into the detection of adjacencies between the planar regions representing them. In our methodology, two primitives are considered to be *adjacent* when their respective regions share a common border, formed out of vertices and edges. As a consequence of that, two regions are assumed to be adjacent when they share common vertices and edges.

From the above, it becomes clear that in order to discover all the adjacency relations in the input mesh, it is required to compare the border of each planar region with the respective borders of all others. However, it is possible for this task to become computationally demanding, as the number of regions increases. This may occur due to the complexity of the original model or because of *over-segmentation*, a term which refers to the detection of a much larger number of mesh subparts than the one needed. Of all these subparts, only some of them can be related to components of the original model and are meaningful for further processing. The rest can be considered as irrelevant and be discarded. So, the detection of all region adjacencies, without any further processing of the initial segmentation, may not only be computationally expensive, but also unnecessary.

To deal with the problem of over-segmentation, a *refinement* process is applied over the initial segmentation, similar to the work of Nan and Wonka [2017]. The objective of this refinement is to reduce the original number of planar segments, by iteratively merging them into new ones, still meaningful for our purposes. The whole process should terminate as soon as no more segments can be merged together. To decide which planar regions are suitable for merging, a set of rules needs to be defined, based on characteristics common to all segments and useful for detecting any similarities between them.

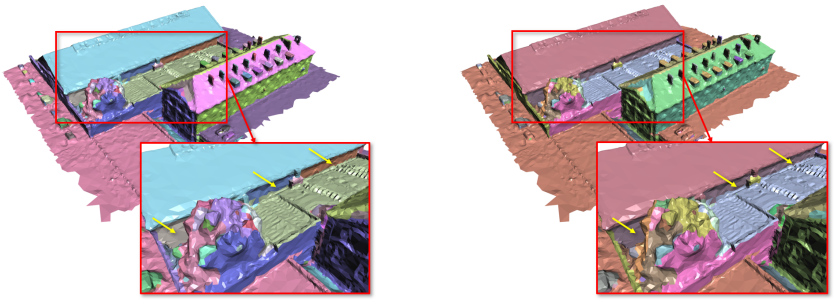
As mentioned in subsection 3.1.2, each planar region is associated with a reference plane (also referred to as *supporting plane*). During the segmentation process, a face is appended to a region, if the distance of its vertices from the supporting plane is lower than a predefined threshold. A similar approach can also be followed to merge regions during the refinement step. To merge two regions, we can simply check if the faces of the first lie on the supporting plane of the second and vice versa, while using the *same* distance threshold from the mesh segmentation. If this condition holds true, then a new segment is formed out of the faces from both regions and its supporting plane is computed. Afterwards, the old regions are discarded and the refinement process continues.

Since the objective of our refinement process is to reduce the number of planar segments as much as possible, the set of merge criteria we set are not that strict. Hence, two planar regions are considered to be similar, only if a few faces from *one of them* lie on the supporting plane of the other — specifically, at least *one fifth* (20%) of the total number of faces belonging to that region. Taking also into consideration that the end result should be planar segments still meaningful for our purposes, an additional criterion is used, based on the orientation of the supporting planes. According to it, two regions are candidate for merging, if only the angle between the normal vectors of their supporting planes is small — in other words, if these planes are almost *parallel*. A maximum normal angle of 10° seems to fit our purposes. The result of the segmentation refinement in the example mesh is shown in Figure 3.9.

3.2.2 Importance

Although the refinement procedure reduces the number of planar regions in the initial segmentation, there might be still segments which are not of use for our purposes and therefore, they must be discarded. One example of such segments are those that correspond to architectural details, like chimneys. Each of these building components is often represented by a single planar region, instead of one for each of its individual parts. Therefore, it is not possible for the structure of the component to be recovered (since a 3D object is simulated only by a plane). In such a case, the respective region only adds to the complexity of the segmentation (see Figure 3.10).

From the above, it becomes apparent that the formulation of a mechanism is needed in order to select only those segments, which are meaningful to us, and disregard the rest. This task actually translates into assigning a notion of *importance* to each one



(a) Initial segmentation **(b)** Post refinement

Figure 3.9: Segmentation refinement: In this figure, the result of our segmentation technique **(a)** is shown, in parallel with the one from our refinement procedure **(b)**. Notice the merge of the two ground segments in the initial segmentation or of the planar regions corresponding to the building wall, presented in the close-up view.

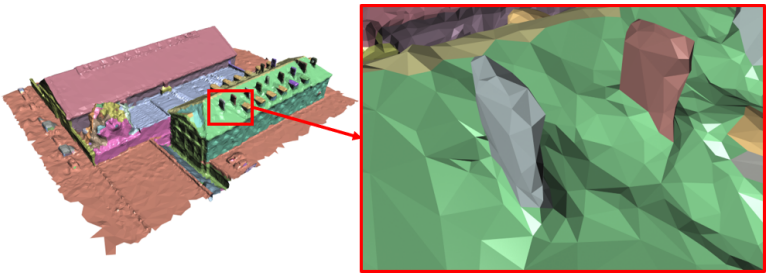


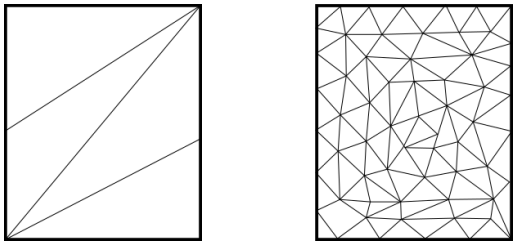
Figure 3.10: Segmentation and architectural details: Architectural details, such as chimneys, cannot be easily detected by segmentation techniques due to the limited resolution of MVS meshes. Even when found, they are approximated by a single planar region, thus any sense on the dimensionality of the object is diminished.

of the planar segments. According to this approach, segments of high importance are to be used in later processing steps, while those of low importance are to be ignored. As a consequence, it is necessary to discover which characteristic of the planar regions could be possibly associated with this notion of importance.

One option could be to assume that the importance of a planar segment is a function of the number of mesh faces it consists of — the larger this number, the higher the segment importance. Although this intuitive approach sounds as reasonable, still there is a major flaw in it. Consider the case of a planar segment that occupies a large part of a given mesh. If for some reason this mesh part comprises only a few faces, then the planar region should be regarded as one of small importance and therefore, discarded. In the opposite case, a planar segment could occupy a limited mesh area covered by a large number of faces and still be regarded as important (see Figure 3.11).

Another solution to this problem could be based on the fact that the main components of a building (walls, roof segments etc.) are those extending over a large *area* of it. Architectural details, despite being also parts of the entire structure, still occupy only a limited space on it. So the key to assigning importance to planar regions seems to be directly related with the *area* of these regions and particularly, their area as a *percentage* of the *surface area* of the entire mesh. Even more, this characteristic is not only independent of the face density for a given mesh, but also independent of the *model scale*. No matter the change in the mesh size, this proportion between the region area and the mesh area remains constant.

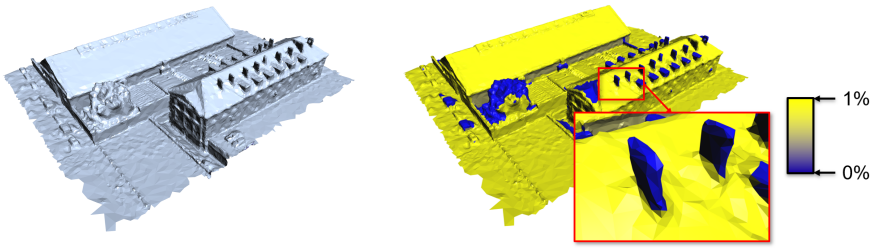
Based on the above, an importance threshold can be defined in order to select only those planar segments, which are meaningful and can be used for the simplification



(a) (b)

Figure 3.11: Region importance and face number: It is possible for a planar region of the same extent to be composed of a small **(a)** or large **(b)** number of mesh faces. Relating the region importance to the number of faces leads to a contradiction, since in the first case the region can be regarded as insignificant, while in the second as important.

of the input mesh in a later processing stage. This threshold is the minimum percentage of the mesh area that a segment should occupy, so it can be considered as important. The application of such a threshold in the example mesh is shown in Figure 3.12.



(a) Original mesh **(b)** Important regions

Figure 3.12: Importance: In this figure, planar regions in the example mesh that occupy more than 1% of the total surface area are coloured in yellow. These regions are considered as important and will be used later for simplification. Regions of lower importance (coloured in blue) such as chimneys are disregarded.

3.2.3 Graph construction

Both with the segmentation refinement and the importance threshold, discussed in subsections 3.2.1 and 3.2.2 respectively, a set of planar segments is formed, which can be considered representative of the model *primitives*. Hence, to complete the model *structure*, as we have defined it in the context of our project, the *interrelationships* of these primitives need to be detected. While these interrelationships are found, information, both on them and the model primitives, is recorded in the form of a *structure graph*, inspired by the *graph of proxies* in the work of Salinas et al. [2015].

Contrary to a mesh, a graph is a data structure, consisting only of vertices and edges [Trudeau, 2015]. Each vertex often corresponds to a certain object and two vertices are connected with an edge, if the respective objects are connected with some sort of "relation". In respect to that, the vertices of our structure graph actually correspond to the model primitives, while an edge is formed when the respective primitives of two vertices are connected with a *topological* relationship. Since, in our case, the only topological relationship we are interested in is that of the *adjacency*, two vertices are linked via an edge, if their primitives are *adjacent*.

Furthermore, an adjacency relation between two primitives holds true for both of its members — if a planar region is adjacent to another, then the opposite statement is also valid. As a consequence, the links of our graph should be bidirectional. A

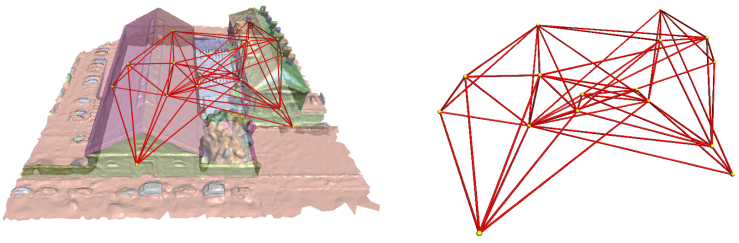
graph of this type is usually called *undirected*, since discovering the entirety of vertex relations depends on following both directions of every edge — from vertex A to vertex B and vice versa.

Based on the above, the construction of the structure graph starts with the definition of its vertices. The number of these vertices is equal to the size of the set of primitives, as it results from applying, in order, our segmentation technique, the refinement process and finally, an importance threshold on the mesh to be simplified. The only thing missing to complete the graph and thus, the model structure, is to also define the links, connecting the vertices with each other — in other words, find the primitive adjacencies.

As stated in subsection 3.2.1, to discover the primitive adjacencies, it is required to compare the border of each one with the borders of all others and identify common simplexes — vertices or edges. At the same time, it is necessary to clarify when two primitives can be considered as adjacent. One way to address this problem could be the statement that two primitives are adjacent, if their respective planar regions share *at least* a certain amount of common vertices or edges and therefore, an extended common border.

Although justified, this approach is slightly complicated, since it requires both a thorough research of the primitive borders and the use of an additional parameter, in order to define the minimum number of simplexes which guarantees the region adjacency. Not only that, it also ignores the case where two primitives may be connected with a common border, composed of a very small number of simplexes. As a result, we follow here a less strict approach, in which two primitives are considered adjacent, if the borders of their planar regions share, at least, one common vertex.

Of course, adopting such a moderate approach also entails the risk of detecting topological relationships between primitives, for which any connection is, in fact, meaningless — despite them sharing a common vertex. This might occur due to flaws in mesh segmentation, which, unfortunately, appear also in our method. However, these flaws also ensure that our graph cannot possibly describe the model structure *accurately*, even in an ideal situation. There will always be some invalid relations recorded in it, but an issue will arise only if the relations *required* to provide a basic notion of the model structure are not included (see Figure 3.13).



(a) (b)

Figure 3.13: Structure graph: In this figure, the structure graph is visualized, along with the segmentation **(a)** and separately in a close-up view **(b)**, by placing each vertex on the centroid of its respective planar region.

3.3 SIMPLIFICATION

With the structure of the input mesh recovered in the form of a graph, we now proceed to the simplification step itself. The main objective of our simplification technique is to produce a simpler, more compact representation of the same model, while recovering and preserving the structure as much as possible.

3.3.1 Building scaffold

Having defined successfully both the model primitives and their interrelationships that we need to preserve, we proceed to the final step of our methodology, which is the simplification itself. In this stage, the main goal is to obtain a mesh with less faces than the original one, which, at the same time, follows the initial structure as much as possible. To achieve that goal, it is necessary to exploit any information available, both in the original mesh and from our pre-processing stages of segmentation and structure definition.

Starting with the mesh segmentation, the primitives of the input mesh, that should be also present — in one form or another — in the simplified version, are known and represented in the form of planar regions — groups of mesh faces, lying on the same *supporting plane*. Furthermore, the borders and thus, the shapes of these regions are also known and well-defined. as sets of mesh vertices and edges. However, these borders are characterized by high complexity, usually consisting of a significant amount of simplexes. Therefore, the main task of our simplification should be the recovery of the *region shapes* in a much more concise form, while retaining any *region adjacencies* — meaning that if two regions are originally adjacent, their simplified versions need to share this type of relation too (see Figure 3.14).

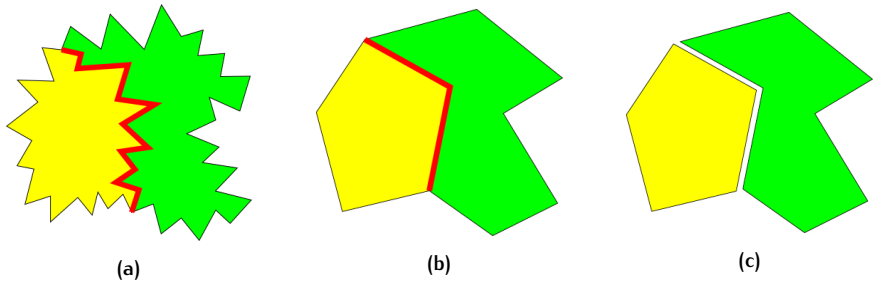


Figure 3.14: Adjacency preservation: The shapes of two adjacent planar regions in the original segmentation **(a)** need to be simplified as much as possible **(b)**, without compromising their connectivity **(c)**.

A possible approach to obtain simpler region shapes could be based on the simplification of their borders, by applying a curve decimation method like the Douglas-Peucker algorithm [Douglas and Peucker, 1973]. Since this algorithm operates on 2D polylines, its application would require to isolate each planar region from the rest and then, project it on its supporting plane, thus compromising the region connectivity. However, the connectivity could be restored with the identification, beforehand, of several "anchor" points, which are maintained during simplification and used afterwards in order to "stitch" the simplified borders together. Such points could be selected, for example, from the set of vertices forming the common borders of all regions.

Although different, our solution is still based on the concept of "anchor" points. Buildings, as man-made objects, are characterized by geometric regularity and clearly distinct features. Two types of such features are the *building corners* and the *building edges*. In a typical model, *building corners* can be defined as *points*, where *three* different building components come into contact. For example, a roof corner might actually be the connection between a roof segment and two walls. In the same sense, *building edges* can be similarly defined as *linear segments*, on which *two* building components meet each other (for example, a roof segment and a wall). These edges *link* the corners between them, resulting into the formation of a *scaffold* (see Figure 3.15), that resembles the *surface scaffold structure* of Wang et al. [2016].

Since the planar regions from the mesh segmentation correspond to the building components, our solution actually focuses on approximating the region borders with

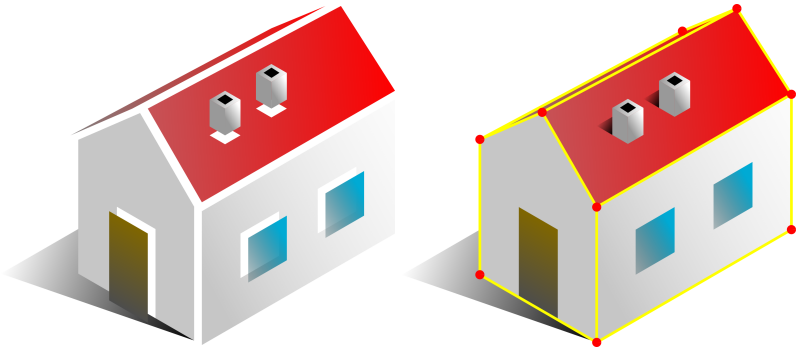


Figure 3.15: Building scaffold: In a typical model, edges (yellow segments) are located at the border of two individual building components while corners (red points) connect three of them. This set of vertices and edges defines a *scaffold* to which the building components conform.

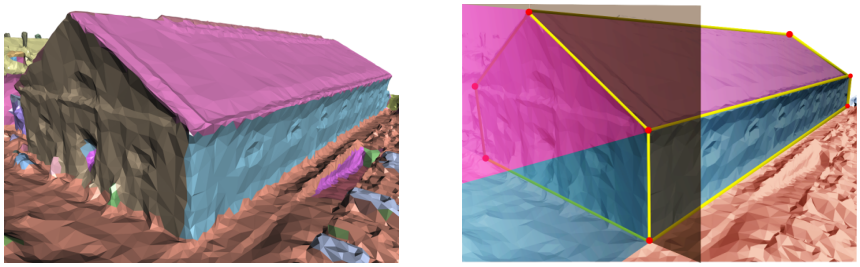


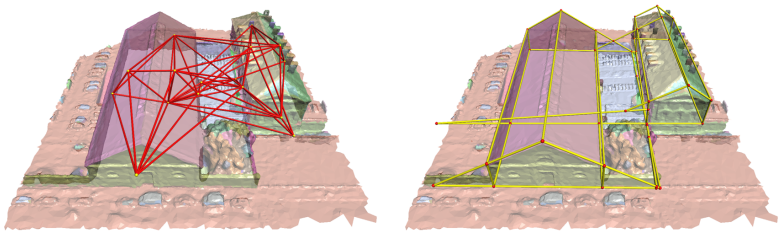
Figure 3.16: Building scaffold and supporting planes: Using the supporting planes of the planar regions from our mesh segmentation, the vertices and edges of the building scaffold can be constructed to define the shape of building components in the simplified version.

the construction of such a building scaffold. In particular, the scaffold *vertices* can be computed as the *intersecting points* of *three* planar regions, by using their *supporting planes*. In the same fashion, to restore the connectivity between the vertices, the scaffold *edges* can be constructed, based on the *intersecting lines* of *two* supporting planes (see Figure 3.16).

Of course, to define both the vertices and edges of the scaffold, it is necessary to know the region adjacencies. For this purpose, our structure graph, in which the topological relationships between primitives are encoded, can be used. To retrieve the part of the scaffold surrounding a certain planar region, first the graph vertex corresponding to that region is identified. By also identifying the *adjacent* vertices of that vertex, the adjacent segments of the original region can be recovered. Finally, the supporting planes of both the region in study and its adjacent regions are collected and their intersections are computed. In this way, the building scaffold is being gradually constructed, by traversing the structure graph and detecting the adjacent vertices of each vertex (see Figure 3.17).

3.3.2 Candidate faces

With the construction of the building scaffold, a simplified version of the original mesh is achieved, in the form of a *wireframe mesh* — a mesh consisting only of vertices and edges. Since the main objective of our methodology is to obtain a fully-defined mesh that represents the original building in a simpler way, the faces of this wireframe mesh have to be defined as well. In this way, the primitives of the input mesh will be defined in the simplified version as sets of *polygonal faces*, which later can be further subdivided into triangular faces, if necessary.



(a) Structure graph **(b)** Building scaffold

Figure 3.17: Structure graph and building scaffold: By traversing the structure graph **(a)**, the adjacent segments for each planar region can be recovered. Thus, the scaffold vertices and edges **(b)** are computed as the intersections of their supporting planes.

There are many different topological structures to represent meshes, but one of the most commonly used is the *face-vertex* convention [Smith, 2006]. In this structure, mesh faces are defined as *ordered* sequences of vertex indices — in other words, it is not enough to know only the vertices belonging to a face, but also the order in which they should be traversed in order to reconstruct it. Similarly, to define a certain face in our wireframe mesh, only the ordered set of its vertices is needed and not the exact geometry — which in this case, it is that of a 3D polygon.

However, before proceeding any further, it is also necessary to take another fact into consideration. As mentioned in subsection 3.2.3, it is not guaranteed that our structure graph captivates the building structure *precisely*. As a consequence of that, the wireframe mesh may also include a number of parts (vertices or edges), which, in fact, are completely *irrelevant* to the components of the original model. The use of these parts results into the production of facets that cannot, in any case, be related to any of the original primitives. Hence, apart from the face definition, it is also important to establish a method, by which only the faces corresponding to some primitive can be selected as valid and the rest are discarded.

Available information in the original mesh could possibly provide a means to check the faces, produced from the wireframe mesh — which, from now on, will be referred to as *candidate faces*. Since we demand that each of these faces should correspond to or, at least, be part of an original primitive, a comparison between them and their respective planar regions should suffice for evaluating their validity. It is also vital to remember that both scaffold vertices and edges, forming the border of a planar region, have resulted from the intersection of its supporting plane with the respective ones of the adjacent regions. Consequently, the entirety of the simplexes, from which a candidate face will be composed, lies *by definition* on the supporting plane — in other words, the face will be *planar*. In a similar fashion, the region faces can be enforced to lie on the same plane, by *projecting* their vertices on it (see Figure 3.18).

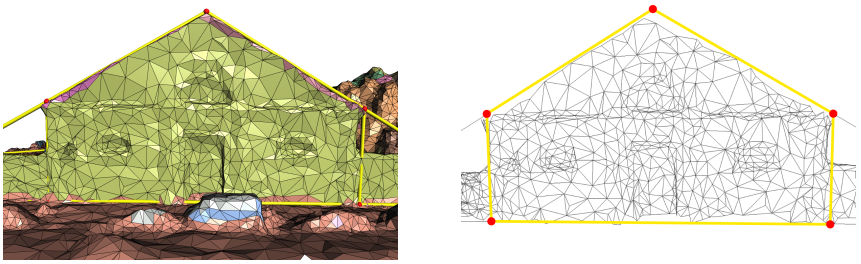


Figure 3.18: Face coverage: Using the scaffold vertices and edges belonging to a planar region (here coloured in green), a polygonal face can be defined. To check its validity, the faces of the region are also projected on the supporting plane. Then, a confidence level can be assigned based on the face coverage by the original region.

With both the candidate face and its corresponding region projected on the same plane, a comparison between them can be conducted in order to assign a level of *confidence* to the face. An intuitive approach, which is also followed here, is to define this confidence as the proportion between the total area of the face and its area, *covered* by the original region — the greater this proportion, the higher the face confidence. However, this task actually translates into computing the overlap between the candidate face and the region faces — therefore, the 2D projection of the face is needed in the form of a *2D polygon*. So, as mentioned earlier, the *3D geometry* of the face is not required for its *definition*, but its *2D projection* is needed for checking its *validity*.

Considering the above, both tasks of face *definition* and *validity* are combined into one process, consisting of the following steps:

- For each planar region, the scaffold edges (as *linear segments*), resulting from the intersection of its supporting plane with the respective ones of the adjacent regions, are collected and projected in 2D.
- This collection of linear segments forms a *2D arrangement* (see Figure 3.19) — a subdivision of the plane into vertices, edges and faces [Agarwal and Sharir, 2000]. The faces of this arrangement are used for:
  - The *definition* of the candidate faces, representing the original region in our simplified version
  - The recovery, for each face, of its 2D projection, later to be used for checking the face *validity*
- The original faces of the planar region are also projected into the supporting plane.
- For each candidate face, the overlap between its 2D projection and that of the original faces is computed. The overlapping area, as a proportion to the total area of the face, provides a *confidence level* for it.

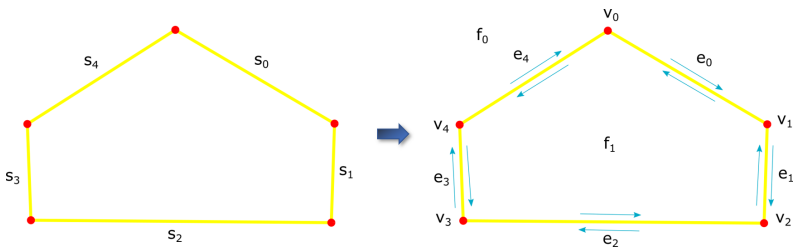


Figure 3.19: 2D arrangement: A 2D arrangement of linear segments **(s)** is the subdivision of the plane in vertices **(v)**, edges **(e)** and faces **(f)**.

The result of this process is an initial, simplified version (*proxy mesh*), which approximates the original model (see Figure 3.20).

3.3.3 Optimization

The proxy mesh, defined by the projected edges of the building scaffold, is just a set of candidate faces, that may correspond or *not* to parts of the original model. Therefore, the final step of our methodology should be the selection of faces in order to construct our simplified version for the input mesh. To accomplish this task, the face confidences, also computed during the previous step, are needed in order to assess their validity (see Figure 3.21).

At a first glance, a simple confidence threshold seems to be sufficient for selecting faces and forming our simplified mesh. However, as mentioned in Section 1.2,

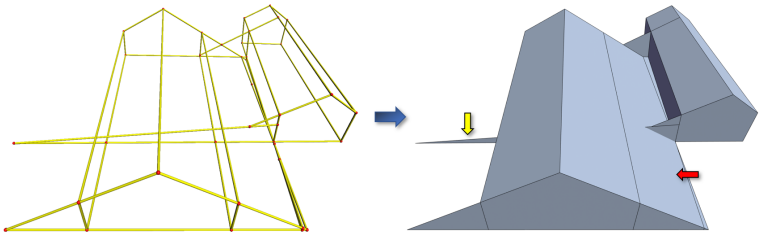
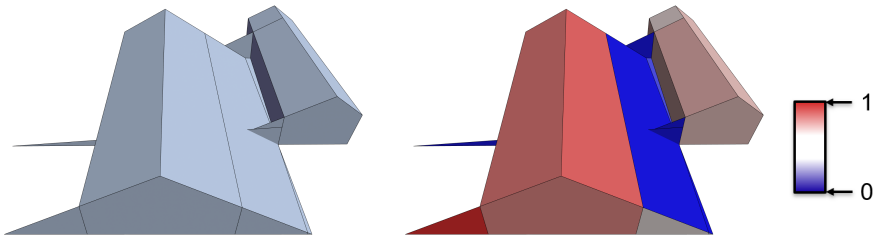


Figure 3.20: Proxy mesh: The 2D arrangements of the scaffold edges form a set of candidate faces for the definition of the proxy mesh. Notice that errors in the structure graph may cause the production of additional faces (red arrow) or self-intersections (yellow arrow).



(a) (b)

Figure 3.21: Face confidence: For each of the candidate faces **(a)**, a confidence level is assigned **(b)**, expressed as the percentage of its area covered by the original planar region.

an additional condition has been set, apart from a simpler representation of the original mesh; the simplification process should also result into a *topologically* valid mesh, which can be later used in applications. For our purposes, this condition of topological validity actually translates into the formation of a *2-manifold* and *watertight* polygonal surface — in other words, a mesh in which each edge is adjacent to exactly *two* faces.

To address this problem, we adopt the approach followed by Nan and Wonka [2017] for the reconstruction of polygonal surfaces from point clouds. Similar to that method, the selection of candidate face is also achieved here through optimization. This optimization process is based on the formulation of a *binary linear programming* problem — a method to achieve the optimal result for a mathematical model whose requirements are represented by linear relationships [Sierksma and Zwols, 2015]. In this problem, several hard constraints are also imposed in order to enforce the resulting mesh to be topologically valid. The solution of it is the assignment of a value to each of the candidate faces; if the face is to be selected, then this value is equal to one (otherwise, it is equal to zero).

The optimization starts with the formulation of an objective function for each of the candidate faces x_i, consisting of three *energy terms*: *face coverage*, *data fitting* and *model complexity*. The *face coverage* term is closely related to the confidence notion, as introduced before:

$$E_c = \frac{1}{area(M)} \sum_{i=1}^{N} x_i \cdot (area(f_i) - area(M_i^a)) \tag{3.2}$$

where $area(M)$ is the total surface area of the simplified mesh, $area(f_i)$ the area of the candidate face and $area(M_i^a)$ the face area, covered by the original region. This term favours faces with high coverage from the original mesh, which, at the same time, have the highest probability of corresponding to one of the original planar regions. To reduce complexity, the area of the simplified mesh is approximated

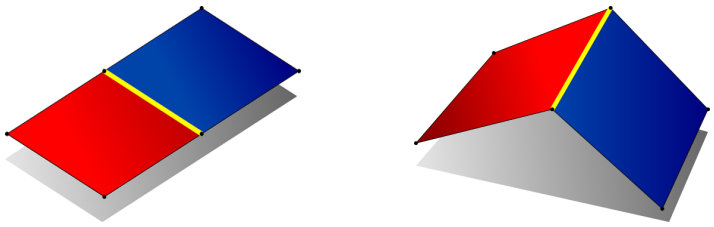
beforehand. Since the simplified version should conform to the original mesh as much as possible, its *bounding box* has to be similar in size with the original one too. Therefore, we assume that $area(M) \approx area(bbox)$.

Although coverage is probably sufficient by itself, it is still better to also include an additional method for further ensuring the validity of a candidate face. Hence, an additional term is introduced, which is directly related to the *fitting* of a face to the original model. According to this approach, the validity of a face is proportional not only to the *total area*, but also to the *total number* of original faces from which is covered. This is expressed through the following *data fitting* term:

$$E_f = 1 - \frac{1}{|F|} \sum_{i=1}^{N} x_i \cdot support(f_i) \tag{3.3}$$

where $|F|$ is the total number of the original faces and $support(f_i)$ 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.

However, the data-fitting term seems to comply to discontinuities in the original mesh (such as holes), which might be also reproduced in the simplified version. This need to avoid gaps in the final model and enforce the creation of large planar regions is addressed by introducing yet another energy term. This term is associated with the *model complexity* — in other words, the presence or not of distinct *features* (details). In general, theses features are produced by *sharp edges*, which connect non-planar faces (see Figure 3.22).



(a) Planar edge **(b)** Sharp edge

Figure 3.22: Sharp edges: Contrary to planar edges **(a)**, sharp edges **(b)** are used to define distinct features in meshes.

Subsequently, the model complexity can be assessed in our energy term, by evaluating the *ratio* of sharp edges over the total number of edges in the simplified version:

$$E_m = \frac{1}{|E|} \sum_{i=1}^{|E|} corner(e_i) \tag{3.4}$$

where $|E|$ is the total number of edges in the proxy mesh. On the other hand, $corner(e_i)$ 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. Otherwise, if the faces are non-planar (sharp edge), the function is equal to one.

The optimal set of candidate faces to form the simplified model is finally selected, by *minimizing* the weighted sum of the above energy terms. The whole optimization problem, along with the hard constraints to ensure that the resulting mesh is both *manifold* and *watertight*, are summarized in Equation 3.5.

$$\min_{x} \lambda_f \cdot E_f + \lambda_c \cdot E_c + \lambda_m \cdot E_m$$
$$\text{s.t.} \begin{cases} \sum_{j \in N(e_i)} x_j = 2 \text{ or } 0 \quad , \quad 1 \leq i \leq |E| \\ x_i \in \{0,1\} \quad , \quad 1 \leq i \leq N \end{cases} \tag{3.5}$$

The result of our optimization method is shown in Figure 3.23. With the optimization step, our method concludes, resulting into a final, simplified version of the input mesh. Both the original mesh and our simplified version are shown in parallel in Figure 3.24, while a thorough comparison between the two meshes will be provided in Chapter 5. There, different results will be presented in order to further support the effectiveness of our proposed methodology.

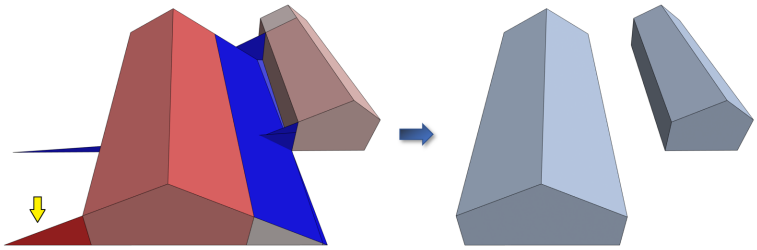


Figure 3.23: Face selection: The optimization results into the selection of candidate faces to form the simplified mesh. Notice that it is not necessary for a face (like the one indicated by the yellow arrow) to be chosen, despite its high confidence. This is due to the hard constraints used, in order to ensure that the simplified mesh will be manifold and watertight.

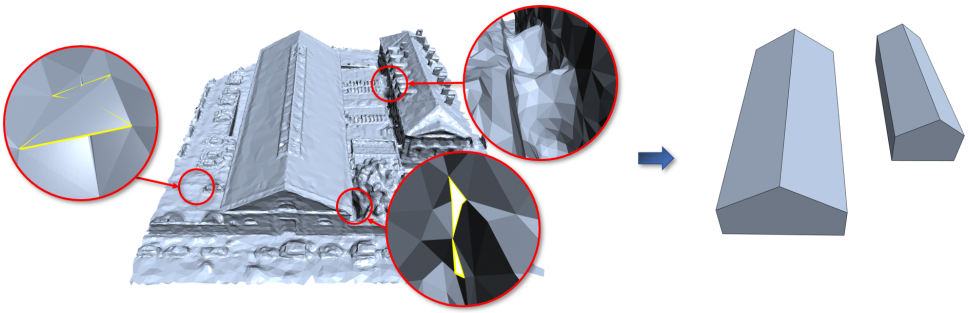


Figure 3.24: Simplified mesh: In this figure, the original building mesh and the result of our simplification process are shown in parallel. Even though various defects are present in the original mesh (holes, self-intersections, occlusions etc.), still our proposed method is able to produce a simplified version of the original.

4 IMPLEMENTATION

In this chapter, the implementation details of our methodology are to be discussed. These details can be divided into two parts: **(a)** programming specifics and **(b)** parameter tuning.

4.1 PROGRAMMING SPECIFICS

Here, a brief analysis is provided on the implementation of our methodology in a programming environment. For the purposes of this project, the methodology was implemented in C++, along with the use of Computational Geometry Algorithms Library (CGAL) [1]. Nevertheless, the following analysis focuses on details necessary for implementing the methodology *in general*, no matter the programming language or software to be used.

4.1.1 Segmentation

The implementation of our segmentation technique is straightforward and can be easily achieved by adopting our method of planarity inheritance from vertices to faces. The exact pipeline is presented in Algorithm 1 (Appendix A). As mentioned before, it is chosen to conduct the segmentation on the mesh faces and not the vertices. However, the algorithm is flexible and open to any modifications in order to decompose the mesh into vertex groups, if needed. Although there might be minor differences, both approaches are actually equivalent, since any analysis on the planar segments, such as the calculation of their supporting planes, is still performed at a vertex and not a face-level.

In our methodology, the computation and update of supporting planes during mesh segmentation is accomplished by means of PCA. The main disadvantage of this approach is that PCA is computationally expensive, so it is a good practice to limit its use as much as possible. Our segmentation algorithm tries to balance computational efficiency with accuracy by re-computing the supporting plane at certain intervals. In particular, the first update takes place after the 1-ring neighbours of the seed face are checked. For all of these faces finally included in the currently expanding region, their 1-ring neighbours are also collected, checked and used at the end for re-fitting yet again a new plane. In this way, the supporting plane is updated only after each k-ring neighbourhood around the seed face is inspected (see Figure 4.1).

Since the supporting planes are needed later during the refinement process and the simplification itself, it is preferred to store them as soon as a region is finalized, thus avoiding their re-computation. This procedure is also followed in the refinement step itself, when a new region is created out of the merge of two others. In this case, the supporting plane is computed using the faces from both regions and then, stored away. At the same time, the planes of the old regions are deleted to save memory space. During both the segmentation and refinement steps, establishing a connection between a plane and its respective region is of paramount importance. In this way,

1 The original code can be accessed in the following link:
https://github.com/VasileiosBouzas/MeshSimplification

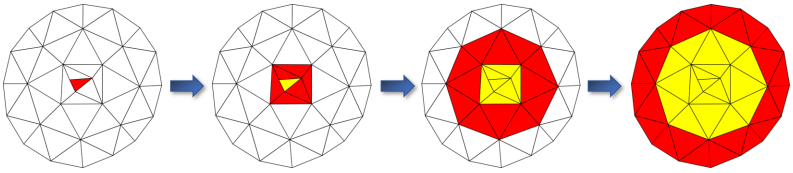


Figure 4.1: Plane update: Starting with a seed face and a supporting plane, a region is grown by first checking the ring neighbourhood of the seed. Before proceeding to the next ring, the initial plane is updated, using all the faces currently belonging to the region.

every time a region needs to be processed in a later stage, its plane can be easily accessed too.

4.1.2 Structure graph

To implement the refinement procedure, as described in subsection 3.2.1, an iterative process is chosen here. Since regions consisting of only a few faces are less likely to correspond to a primitive, they need to be merged first. For this reason, the refinement starts with the sorting of all regions in an *ascending* order, based on their number of faces. Then, the smallest region is checked according to the merging criteria we have defined, also in subsection 3.2.1. When a region suitable for merge is found, then the faces of both regions are marked as members of a new, larger segment. The older regions are deleted and the refinement starts *all over*. This whole process terminates when a region merging can no longer take place.

After the end of the refinement step, importance is assigned to the remaining segments by calculating their surface area. The area of a region, which is simply the total area of the faces belonging it, is then compared to the total surface area of the entire mesh. If the resulting importance is lower than the predefined threshold, the whole region is discarded to save memory space, since it is no longer needed for later stages. In this way, the number of primitives to be used for simplification is finally fixed and we can proceed to the definition of the model structure with our structure graph.

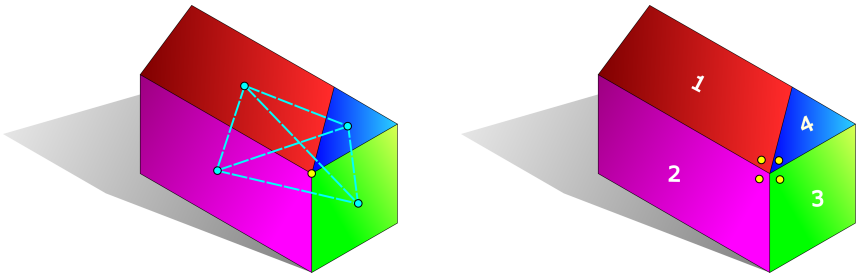
The structure graph is initialized by first appointing a vertex to each of the regions resulting from the segmentation, refinement and importance steps. With the primitives defined, the structure can be completed by adding the interrelationships between them. For this task, it is required to compare the border of each region with the border of all others and identify, at least, a common vertex. Here, this is accomplished by collecting all vertices belonging to a region and checking the neighbouring faces of each one. If a face is marked in a different region than the one in study, then a connection between these two regions is formed by linking their respective graph vertices with an edge.

Similar to the supporting planes, a connection is also established between a graph vertex and its respective region to facilitate later processing. Specifically, this connection is built to be bidirectional; when a region is to be processed, its graph vertex can be accessed and vice versa. The first connection type is used during the construction of the structure graph, when two regions need to be connected via an edge and their vertices have to be detected in the graph. The second type is helpful during the simplification process; while traversing the graph, the respective region of every vertex can be retrieved and consequently, its supporting plane.

4.1.3 Simplification

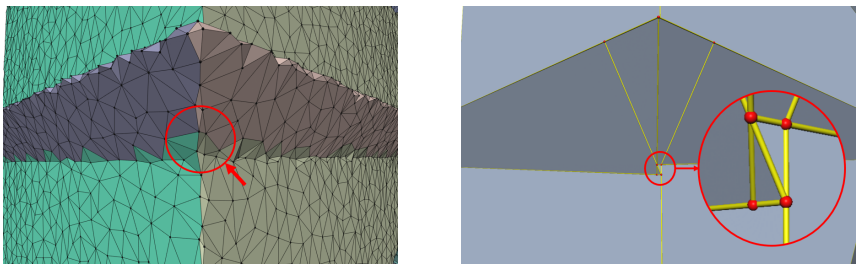
With the structure graph complete, we are able to proceed to the final stage of our methodology, the simplification itself. The first step in the simplification process is the construction of the building scaffold. For this purpose, the graph is traversed — with the help of the connections described above, so that the vertices and edges of the scaffold can be defined as intersections of supporting planes. Vertices are the first to be constructed as intersections of plane triplets. Furthermore, to deal with flaws in the definition of the model structure in our graph, any resulting vertex, not enclosed by the bounding box of the original mesh, is considered invalid and it is discarded. For the remaining set of vertices, their connectivity needs to be recovered in order to obtain the scaffold edges.

Due to arithmetic precision, individual vertices cannot be identified by their location, since the geometry of a vertex will slightly differ when computed as an intersecting point of various plane triplets (see Figure 4.2). Even more, it may occur that two vertices actually correspond to different building corners, despite being located at exactly the same point. To address these problems, each vertex is uniquely distinguished from others not by its location, but by the plane triplet from which it was originally constructed. Although this approach may result into multiple vertices, representing the same corner at the simplified model (see Figure 4.3), it still enables the recovery of the scaffold edges in a much more simpler way than actually trying to merge these vertices into one.



(a) **(b)**

Figure 4.2: Corners: There are cases where building corners connect more than three adjacent components **(a)**. Due to arithmetic precision, each triplet of supporting triplets results into a different point, corresponding to the same corner **(b)**.



(a) Original mesh **(b)** Simplified mesh

Figure 4.3: Simplified corners: The area corresponding to a building corner, where four or more planar regions meet **(a)**, is approximated in our simplified version with a group of faces **(b)**, instead of a unique vertex.

In particular, the scaffold edges are recovered by computing the intersections of adjacent plane pairs. If the intersection of such a pair is a line, any vertex, constructed from the intersection of this pair with a third plane, should lie on it. Hence, if two vertices share the same plane pair, they should be connected in the scaffold through an edge. If only two vertices were lying on each of the scaffold edges, then this

connection could be established directly. However, due to refinement, there are cases in which more than two points are lying on a given line (see Figure 4.4).

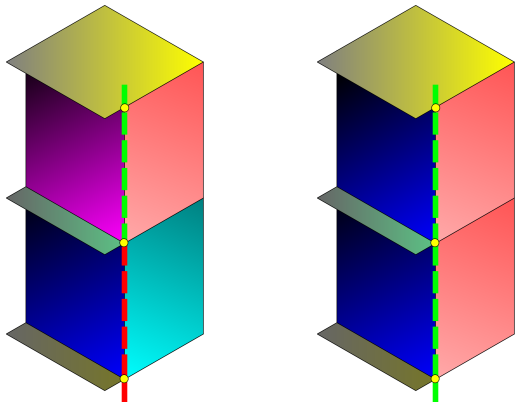
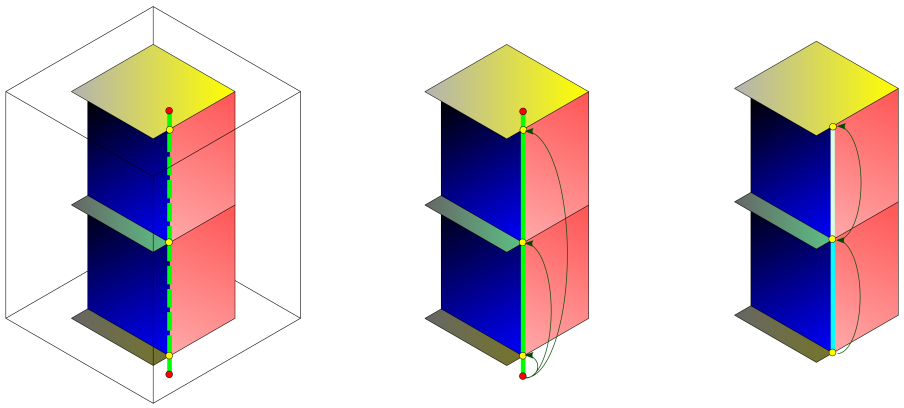


Figure 4.4: Scaffold edges: Without refinement **(a)**, only two vertices lie in any line produced by the intersection of plane pairs. However, due to refinement **(b)**, there are cases in which multiple vertices are located along the same line.

To solve this problem, the following approach has been devised; at first, each line is converted into a linear segment by clipping it with the bounding box of the original mesh. Afterwards, all the vertices, sharing two common planes with a given segment, are collected. Using one of the segment endpoints as a reference, the vertices are ordered according to their distance from it. Finally, the original segment is divided into sub-segments by linking each vertex with its next in order (see Figure 4.5).



(a) Line clipping **(b)** Distance sorting **(c)** Edge construction

Figure 4.5: Edge splitting: To construct scaffold edges, an intersecting line must be first clipped with the original bounding box **(a)**. Then, all the vertices lying on this line are sorted by distance from one of the endpoints **(b)**. Finally, edges are formed by linking each vertex with its next in order **(c)**.

With the aforementioned approach, the connectivity between the scaffold vertices can be fully restored. If a precise definition of the model structure was provided, then all edges should intersect with others only at their endpoints; namely, the scaffold vertices. Unfortunately, due to flaws in our structure graph, it might happen that two edges intersect in some other point in 3D space (see Figure 4.6). At a first glance, this inconsistency seems trivial, but it might affect the definition of candidate faces at a later processing stage. Therefore, any intersecting edges should be further split by defining a new vertex at the point they meet each other.

To split two scaffold edges, their intersecting point is required, but, due to arithmetic precision, its computation proves to be impossible most of the times. To solve the

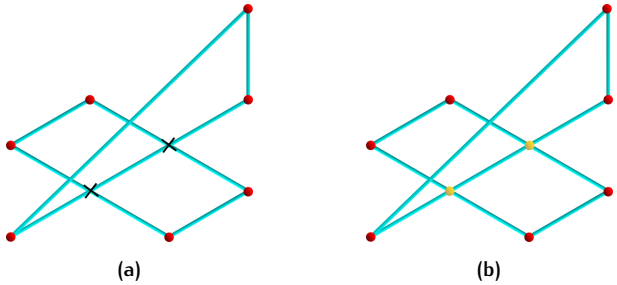


Figure 4.6: Edge intersection: There are cases where scaffold edges meet at a point other than their endpoints **(a)**. Since this inconsistency might affect the definition of candidate faces at a later stage, the intersecting edges should be further split by defining a new vertex at their meeting point **(b)**.

problem of edge intersection, we are able to adopt here an alternative solution. Since it is known for each edge from which plane pair it was originally constructed, we can check for intersections between all edges sharing one common plane — in other words, between edges lying on the same plane [2]. Then, the first edge is compared with the other plane of the second one. If its endpoints are located at different sides of the plane (one in the negative side and one in the positive), then we can deduce that the lines intersect. As a common point for both edges, we can assign the one resulting from the intersection of the first edge with the other plane of the second (see Figure 4.7).

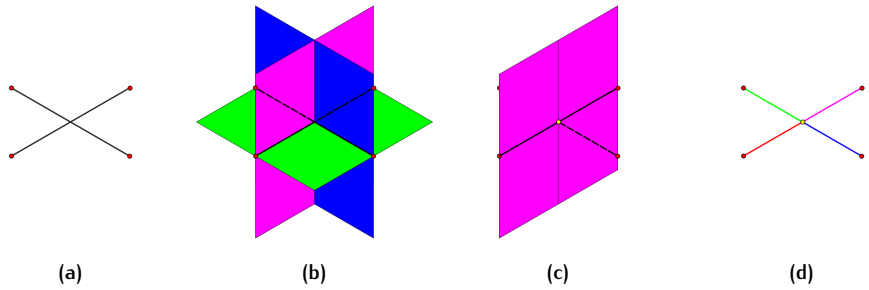


Figure 4.7: Edge cross-split: For two intersecting edges **(a)** lying at the same plane (green), the other two planes (blue & purple) are retrieved **(b)**. For each edge, the position of its endpoints, in reference to the plane of the other, is checked. If the endpoints are located at different sides **(c)**, the intersecting point of the edge with the plane is used to construct four new segments **(d)**.

This final addition of vertices and edges completes the construction of the building scaffold, resulting into a wireframe mesh whose faces need to be defined next. As mentioned in subsection 3.3.2, the candidate faces are produced by the 2D arrangements of scaffold edges belonging to the same plane. For this purpose, the structure graph is traversed and the supporting plane of each region is collected. Then, all edges, resulting from the intersection of this plane with others, are retrieved and projected on it to form the arrangement. Here, the formation of the arrangement can be accomplished with the respective package of CGAL or some other library of computational geometry. Finally, the arrangement faces are used to produce **(a)** the ordered sequence of vertices defining a candidate face and **(b)** the 2D projection of this face on the plane.

Before proceeding to the optimization, some additional information is necessary for each of the candidate faces; namely, **(a)** the total area of the face covered by the original region and **(b)** the number of original faces supporting the candidate

2 Edges sharing the same plane pair can be safely excluded, since they result from the same intersecting line and we have already ensured that they are connected at their endpoints (see Figure 4.5c for an example)

one. Hence, the original region is also projected on the plane, converting the whole process into a polygon-polygon intersection problem. The computation of such intersections are generally expensive, especially in our case where each candidate face has to be compared with hundreds, if not thousands, originals.

To achieve optimal performance, a candidate face is first compared to an original one, by examining the overlap of their bounding rectangles. If there is none, then the two faces are completely irrelevant. Otherwise, the positions of all the vertices in the original are then checked in reference to the candidate. When all vertices are inside the candidate, then the total area of the original covers it and there is no need for intersection computation [3]. If even one of them lies outside, then the interior vertices are collected, along with the intersecting points of the edges from both faces. Finally, the area of the convex hull, formed from these vertices, is added to the covered area of the candidate face (see Figure 4.8).

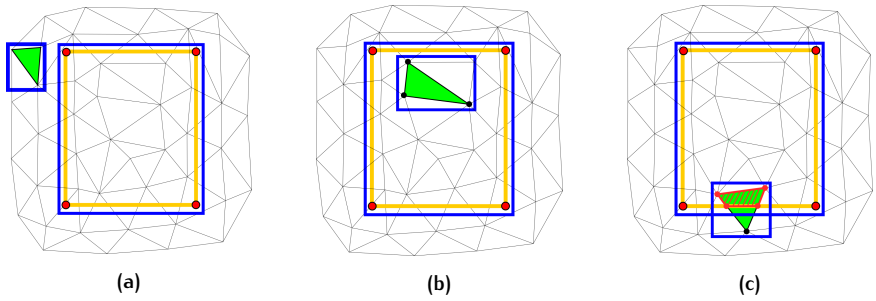


Figure 4.8: Polygon-polygon intersection: If the bounding rectangles of two polygons do not overlap **(a)**, then there is no intersection. When all the vertices of one are inside the other **(b)**, then the intersection is the polygon itself. If even one vertex lies outside **(c)**, the intersection is a polygon formed by the interior vertices and the intersecting point of the polygon edges.

The final stage in the simplification process is the optimization presented in subsection 3.3.3. In this step, faces from the candidate set are selected to form the final, simplified version of the original mesh. The binary linear problem, formulated for this purpose, is solved here with the Gurobi solver [4]. Notice that for the hard constraints used to ensure that the final mesh is manifold and watertight, information on the adjacent faces for each of the scaffold edges is required. This information is acquired along the construction of the candidate faces from the 2D arrangements of the edges. Based on this information, the constraints are set as follows; if an edge is adjacent to less than two faces, then these faces are disregarded. In the opposite case, only two of them are to be selected.

4.2 PARAMETER TUNING

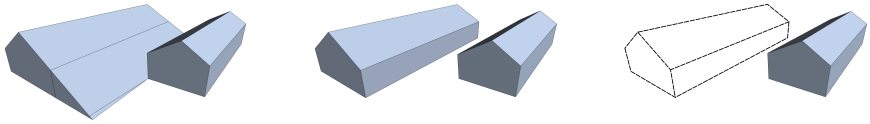
From the analysis of our methodology provided in Chapter 3, it can be deduced that the basic parameters of our simplification method are the following:

- The order of the k-ring neighbourhoods for the computation of planarity
- The distance threshold for our segmentation method
- The importance threshold for the selection of planar segments to be used for simplification
- The coefficients of the three energy terms in our optimization process

3 In our approach, only those faces are considered as supporting of the candidate one
4 http://www.gurobi.com/

Although these parameters could remain as user-defined, still we would like to make the
more automated approach — at least to some extent — is preferred here. To achieve this, some of the different parameters, appearing in our method, have been tuned through experimentation in order to apply to the majority of available models.

Starting with the coefficients of the energy terms in our optimization, their values can be set equal to those used by Nan and Wonka, who originally designed the method for the selection of the candidate faces. According to this method, the coefficient values, for most models available, are the following: $\lambda_f = 0.43$, $\lambda_c = 0.27$ and $\lambda_m = 0.30$. A wide range of these values is also able to produce the same results, with the exception of 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 Figure 4.9). 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.



(a) $\lambda_c = 0.14$, $\lambda_f = 0.56$ **(b)** $\lambda_c = 0.27$, $\lambda_f = 0.43$ **(c)** $\lambda_c = 0.56$, $\lambda_f = 0.14$

Figure 4.9: Energy coefficients: The data-fitting coefficient balances the effect of the coverage energy term over the simplified mesh **(b)**. However, if one of the coefficients is extremely favoured over the other, a larger **(a)** or smaller **(c)** amount of candidate faces will be preserved in the final result.

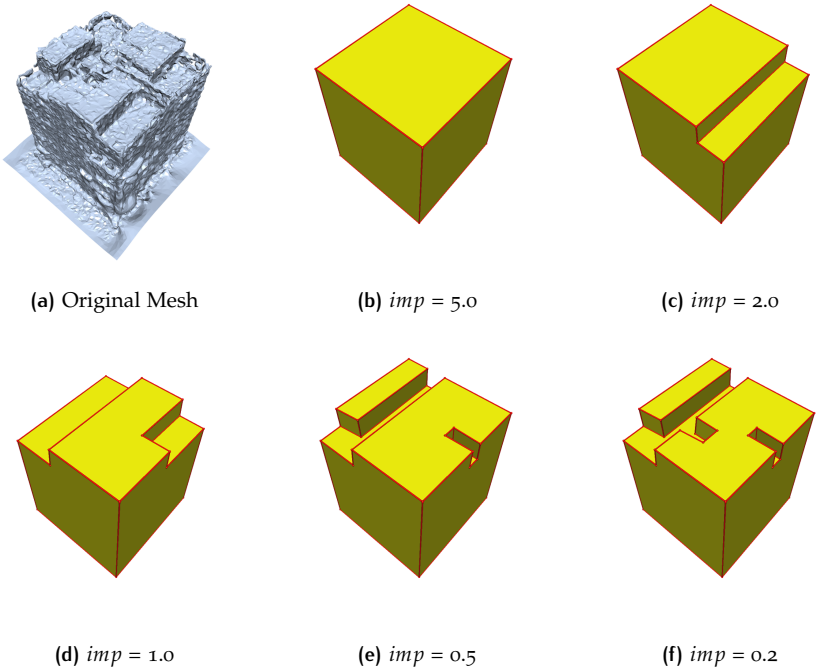
At this point, it is necessary to notice something important. As stated in subsection 3.3.3, the model complexity term allows the formation of large planar regions by limiting the number of sharp edges in the final model. However, in our methodology, the majority of edges in our building scaffold represents the boundaries between adjacent planar regions. Therefore, the scaffold consists, almost entirely, of *sharp* edges, rendering the use of the complexity term meaningless. Still, we choose to include this term in our method as a safety measure, instead of just setting it equal to zero. Even with this choice, the selection of candidate faces is highly dependent, for most models, on the proportion between the data-fitting and coverage coefficients.

A comparative analysis on the order of the k-ring neighbourhoods for the computation of planarity has already been conducted in subsection 3.1.1 (see Figure 3.5). There, it is shown that neighbourhoods of order 3 are more than sufficient for the detection of borders between adjacent planar regions. Any lower order is not sensitive enough for this task, while any higher order increases the computational time considerably. Furthermore, it is vital to take into consideration that the same ring is to be used also for the initial definition of supporting planes in our segmentation technique. While a close, first approximation of a region plane is needed, still its initialization should not be time-consuming.

With regard to the distance threshold, its automation proves to be difficult, mainly because it is also dependent, among other things, on the model scale. However, experimentation indicates that our segmentation technique closely follows the triangulation of the original mesh — that is to say, the threshold is comparable to the average edge length of this mesh. Therefore, the computation of this value provides the user, in most cases, with a pretty good suggestion on the distance threshold, needed for a meaningful segmentation of any given mesh.

With all that said, the only parameter left completely user-defined is the importance threshold. Nevertheless, still some suggestions can be also provided for the use of this parameter. As stated in subsection 3.2.2, the main purpose behind our notion of importance is the selection of the main building components to be used for simplification. Components (such as architectural details), which cannot be approximated via mesh segmentation, are discarded as of low importance. Hence, the simplification of model filled with architectural details requires a much higher importance threshold than those of low complexity.

Generally, a threshold of 1% seems to be sufficient for the simplification of any given model. Experimentation also reveals that this parameter can have a maximum range from 0.1% to 5%, in which it still produces simplified meshes, conforming adequately to the original structure (see Figure 4.10).



(a) Original Mesh **(b)** $imp = 5.0$ **(c)** $imp = 2.0$

(d) $imp = 1.0$ **(e)** $imp = 0.5$ **(f)** $imp = 0.2$

Figure 4.10: Importance influence: By adjusting the importance parameter, it is possible to obtain simplified versions of a given model for different levels of detail.

5 RESULTS & ANALYSIS

In this chapter, various results are presented from the application of our simplification method on a set of available MVS building meshes. For these meshes, analyses are conducted to assess their conformity to the goals we have defined for our project. Finally, a comparison is provided between other simplification techniques and ours.

5.1 RESULTS

The implementation of our methodology in a programming environment provides the possibility of testing it, in reference to the requirements we have defined for our project. As mentioned in Section 1.2, our main objective is the production of a mesh, which should be a simpler, more compact representation of the original model. This mesh should also follow the original structure of the input model (as defined for the purposes of this project), as much as possible. Furthermore, an additional condition of topological validity has been set to ensure that our simplification result is ready to be used in further applications.

As stated before, the only input needed for our simplification method is the MVS mesh of a given building — any information on the original point cloud or imagery is considered as unavailable. Moreover, this mesh can be either *closed* or *open*, depending on whether each of its edges is adjacent to two faces or not. Although our methods results into a closed mesh, still the input is allowed to be open — which is common, especially for building models extracted from urban scenes. However, In this case, a *ground plane* is required in order to complete the simplified model. As of right now, this plane must be provided from the original scene, out of which the model was extracted. However, an alternative implementation could possibly allow the user to import such a plane, according to their needs.

With respect to the above, our simplification method was applied in a set of available MVS building meshes. The results, shown in Figure 5.1, mostly refer to open meshes — with the exceptions of the models **(d)** and **(g)**. Furthermore, simplification is performed over models of individual buildings, again with the exception of the model **(a)** — also used in Chapter 3 for the analysis of our proposed methodology. These models consist only of planar components in various configurations, especially for 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 cloud.

At a first glance, our method seems to produce a well-defined, simplified version of the original model. Of course, it is also necessary to analyse these models according to the prerequisites we have established for them. In the following sections of this chapter, further details on the models are presented, which serve as a ground for a brief discussion on their different aspects.

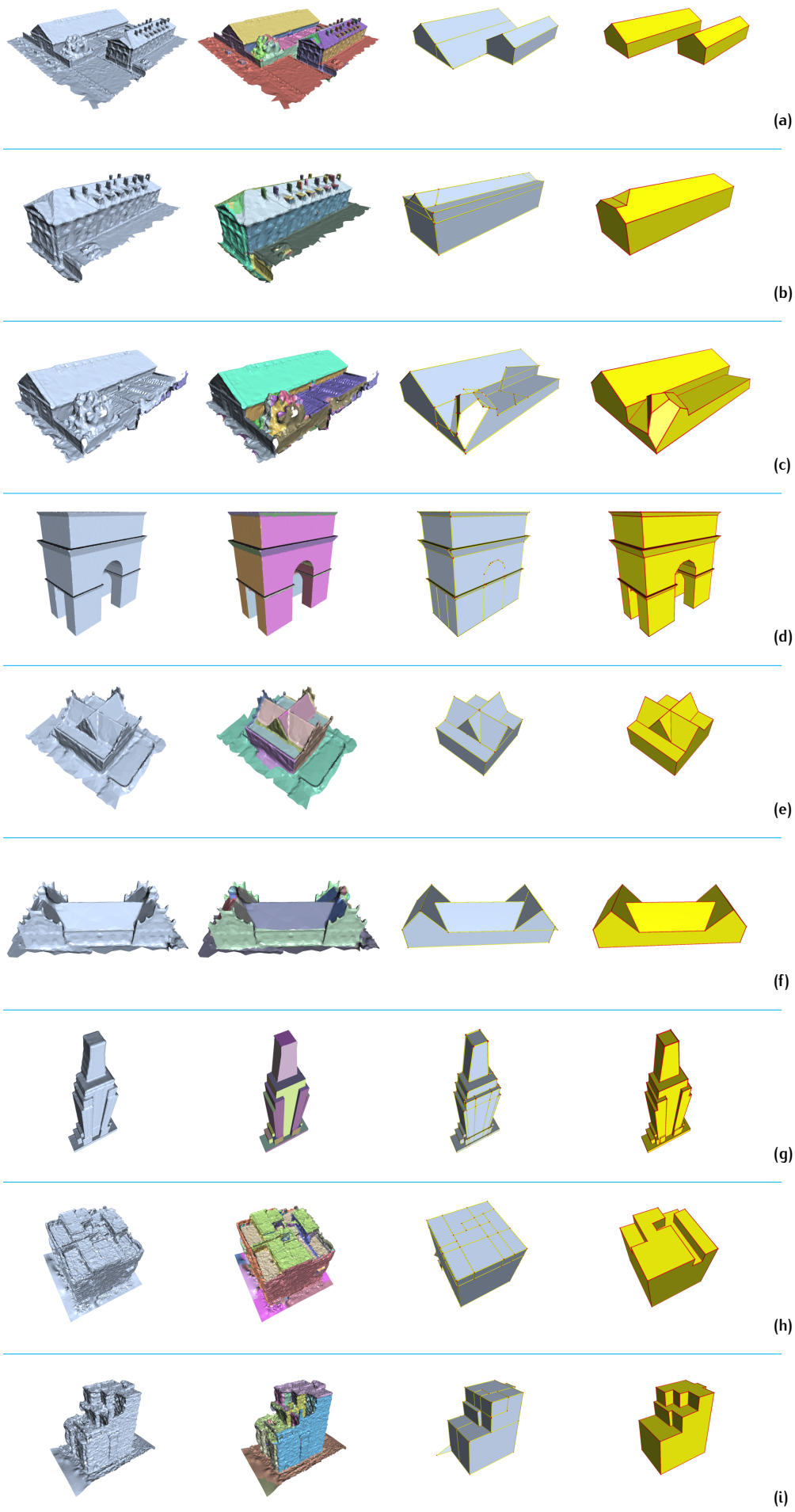
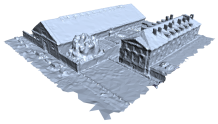
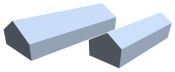
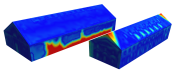
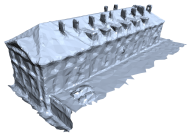
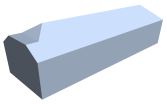
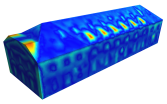
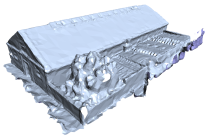
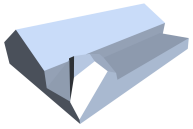
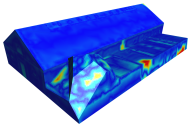
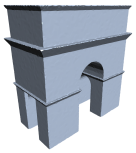
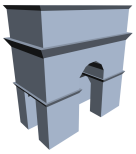
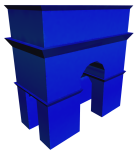
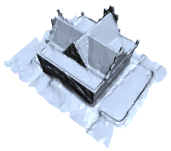
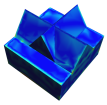
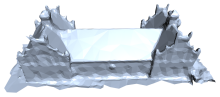
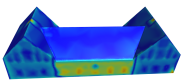
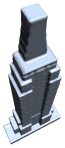
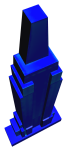
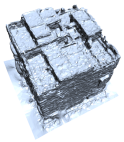
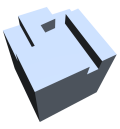
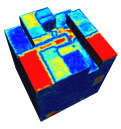
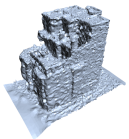
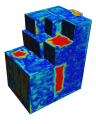


Figure 5.1: Simplification results: In this figure, results are presented from the application of our simplification method on a set of MVS building meshes. From left to right: original model, refined segmentation, candidate faces and simplified mesh.

Mesh	Error RMS (% BBox Diagonal)			
	Original	Simplified	Error	RMS
(a)				0.5%
(b)				0.4%
(c)				0.3%
(d)				0.1%
(e)				0.4%
(f)				0.5%
(g)				0.1%
(h)				1.0%
(i)				0.8%

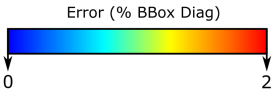


Table 5.1: Error analysis: From left to right, the original and simplified meshes are presented, along with a visualization of their Hausdorff distance. At the final column, the Root Mean Square (RMS) error of this distance is also provided, as a percentage of the bounding box diagonal.

5.2 ANALYSIS

In this section, further details on our results are presented and a discussion is conducted on five different aspects of them: topological validity, error analysis, memory reduction, execution time and comparisons with other available methods

5.2.1 Topological validity

We start our discussion with details on the topological validity of our results, since it is closely related with evidence, which supports our claim of achieving simplified versions of building models. As mentioned in Chapter 1, simplification is the process of reducing the amount of faces for any given mesh. That is to say, our simplification should result in a mesh with less faces than the original. In Table 5.2, topological information is presented, both on the input meshes and our simplified versions of them.

Mesh	Original				Simplified				
	V	E	F	M (kB)	V	E	F	M (kB)	Compression
(a)	23,811	47,030	70,910	1,542	20	48	32	1	0.1%
(b)	7,050	20,913	13,839	639	25	69	46	2	0.3%
(c)	11,026	32,498	21,454	1,003	57	165	110	3	0.3%
(d)	13,631	40,887	27,258	797	171	507	338	9	1.1%
(e)	3,131	9,308	6,172	282	33	93	62	2	0.7%
(f)	5,067	14,998	9,923	456	21	57	38	2	0.4%
(g)	19,524	58,566	39,044	1,891	144	426	284	8	0.4%
(h)	20,000	60,005	39,948	1,901	67	195	130	4	0.2%
(i)	18,721	56,005	37,269	1,773	52	150	100	3	0.2%

Table 5.2: Mesh statistics: For both the original and simplified mesh, the number of vertices **(V)**, edges **(E)** and faces **(F)** is presented. The memory of both meshes is also provided, along with the respective compression rate. Notice that for this comparison, the polygonal faces of the simplified meshes have been previously triangulated, similar to the originals.

A simple comparison between the two versions of each model reveals that the result of our method always consists of less faces than the original, thus simplifying it significantly. By inspecting the simplexes of the simplified version separately, it is also possible to deduce that it is topologically valid. The requirement that the result of our method should be manifold and watertight mathematically translates into having an object, which is topologically homeomorphic to a sphere. A sphere, as well as the surface of any convex polyhedron, has an *Euler characteristic* equal to two (which derives from the relation V-E+F) or a *genus* of zero.

To confirm these facts for each of the simplified meshes, their Euler characteristics can be first calculated with the help of Table 5.2. Notice that since the model **(a)** comprises two components and not one like the rest, its characteristic should be equal to four (the sum of the component characteristics). Computation of the genus

was also conducted both in Meshlab [1] and Val3dity [2]. These pieces of information can ensure, to some extend, that our simplified meshes are free of any geometric and topologically defects, such as self-intersections and holes. However, they do not assert that the mesh is also consistently oriented.

It is easy to observe that our methodology deals with the definition of a simplified mesh through the construction of polygonal faces, without paying attention to their orientation. This issue is, instead, addressed in a post-processing step, after the simplification. Due to our optimization, each edge of our simplified meshes is always adjacent to two faces. For the orientation to be consistent, an edge has to be also defined in both faces with *opposite* directions (see Figure 5.2). Thus, the face orientation can be corrected by first collecting all the face edges. Afterwards, a random face is selected as properly oriented and its edges are located, along with their duplicates. If a duplicate has a direction similar to that of the original edge, its face is reversed. This procedure continues for face after face, till all mesh edges have been traversed.

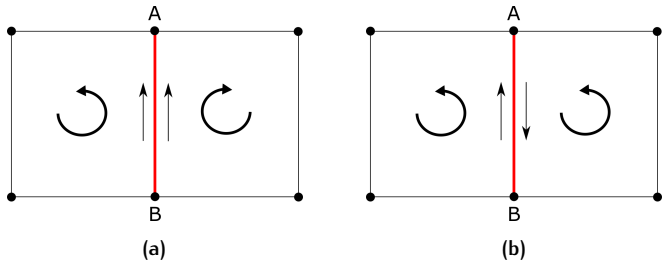


Figure 5.2: Consistent orientation: If two faces have opposite orientations **(a)**, their common edge is defined with the same direction in each of them. Instead, with the same orientation **(b)**, the edge appears in the mesh with both directions.

The aforementioned procedure corrects any flaws in the face orientation, thus solving this topological issue too. Nevertheless, the topological validity does not ascertain that our mesh can be used for further applications, due to the fact that it consists of polygonal, instead of triangular, faces. Problems might also be caused, since the mesh faces do not necessarily have the same shape (triangle, quad etc.) or are *concave*, instead of convex. Taking that into consideration, our implementation also triangulates all faces, before finally exporting the resulting mesh.

5.2.2 Error analysis

Apart from a simpler representation, it is also necessary to examine if the initial structure of the model has been also preserved. For this to be true, both the model primitives and their interrelationships, as we have detected them in the input mesh, need to remain intact. This can be also interpreted that the simplified mesh should conform to the original; namely, the distance between the two meshes should be minimum. Hence, the structure preservation can be assessed by computing the *Hausdorff distance* of the meshes.

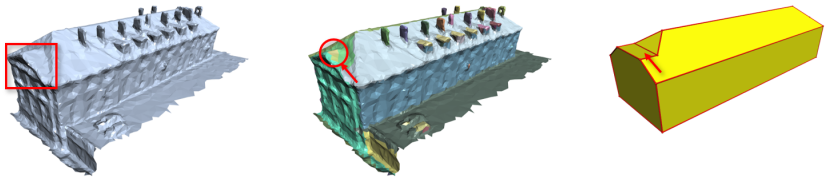
A common method for computing the Hausdorff distance is based on the construction of random points around the comparing meshes. For this purpose, the meshes need to be aligned at first. Then, the Euclidean distance of each point is calculated from both meshes, while the Hausdorff distance is finally estimated as the difference. Since this estimation gets better and better as the amount of sampling points rises, it is considered as the best practice, both for accuracy and computational efficiency, to have a sample equal, in size, to the vertices of the most detailed mesh; namely, the

1 http://www.meshlab.net/
2 https://github.com/tudelft3d/val3dity

original one. Based on these, the results of our error analysis, conducted in MeshLab, are shown in Table 5.1.

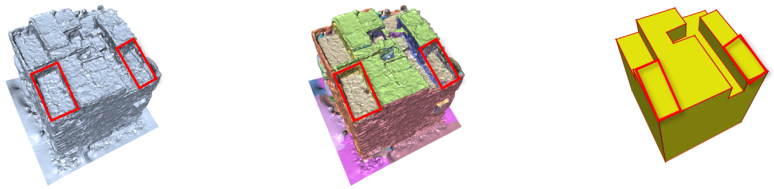
From the above table, it can be observed that for all cases, both closed and open meshes, the mean error is small. This indicates that our simplified versions closely follow the initial building models, especially when the original mesh is clean [**(d)**, **(g)**]. However, our simplification method performs rather well, also when the input model is quite noisy [**(h)**, **(i)**]. A comparison between model **(a)** and models **(b)**, **(c)** reveals that although it is possible for the method to be applied in an urban scene, still it is better to process each building model individually in order to achieve the optimal result.

Furthermore, our error analysis reveals certain inconsistencies that might be observed between the original models and our simplified versions. These inconsistencies are related to parts of structure that appear **(a)** with different geometry in each model or **(b)** only in one of the two models. The former error (see Figure 5.3) is associated with flaws due to segmentation — mainly, the detection of less planar components than the ones necessary to fully approximate a given model part. The latter one (see Figure 5.4) with the hard constraints imposed in our optimization process to ensure the manifoldness of the final result. If candidate faces in the building scaffold, with low confidence, do not entail a risk to the manifoldness of the simplified mesh, the hard constraints still attempt to incorporate them in the resulting model, one way or another.



(a) Original **(b)** Segmentation **(c)** Simplified

Figure 5.3: Structure errors (segmentation): In this example, the original model contains a roof segment, consisting of two inclined planes. However, our segmentation technique is able to detect — for the given distance threshold — a unique planar region. As a consequence, the segment is finally approximated by a single planar component in our simplified version.



(a) Original **(b)** Segmentation **(c)** Simplified

Figure 5.4: Structure errors (optimization): In this example, there are certain parts in our simplified version that cannot be identified in the original model. The main cause of this inconsistency is that the hard constraints of our optimization process enforce the construction of such parts, since they do not affect the manifoldness of the final result.

5.2.3 Memory & Execution time

Since the number of simplexes used for the definition of a model significantly decreases due to our simplification method, we should notice, at the same time, a similar reduction in memory space. Indeed, such a reduction is observed, as shown in Table 5.2. This decrease in memory space means that our simplified meshes are

more manageable than their originals, while they still conform successfully to the initial structure. This allows the performance of any spatial analysis on them, in a much simpler fashion. This fact further supports interoperability, due to the fact that the low memory size of our simplified meshes permits their fast transfer through the Web.

Apart from memory depletion, execution time is also an important factor in order to assess the efficiency of our methodology. Of course, this time highly depends on the given implementation in a programming environment, but it still provides us with some clues on the general performance and scalability of the method. For our implementation, the execution times [3] for the application of the method on the available dataset are presented in Table 5.3.

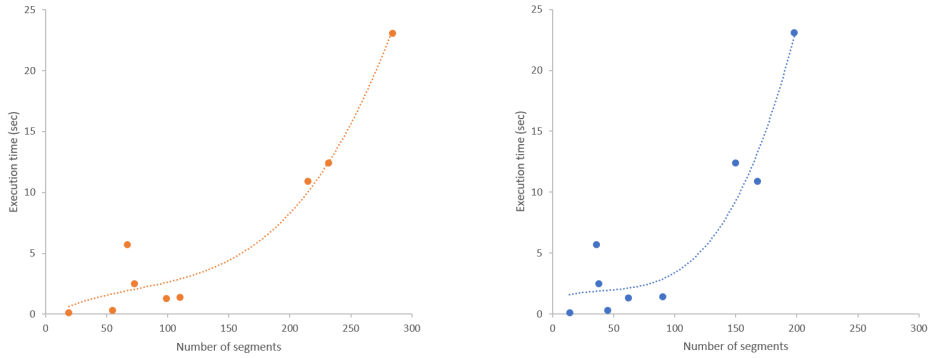
Mesh	Execution time (sec)				
	Planarity	**Segmentation**	**Structure graph**	**Simplification**	**Total**
(a)	0.9	12.4	0.5	0.2	14.0
(b)	0.3	1.3	0.1	0.1	1.8
(c)	0.4	1.4	0.2	0.2	2.2
(d)	0.5	2.5	0.2	0.6	3.8
(e)	0.1	0.1	0.1	0.1	0.4
(f)	0.2	0.3	0.1	0.1	0.7
(g)	0.7	5.7	0.4	0.9	7.7
(h)	0.8	23.1	0.5	0.3	24.7
(i)	0.7	10.9	0.4	0.2	12.2

Table 5.3: Execution time: From left to right are presented the execution times (in seconds) for the following four stages of our simplification method: planarity computation, segmentation (with refinement), structure graph and simplification.

As presented in the above table, the most demanding step of our method (in terms of processing time) is the mesh segmentation and mainly, its refinement. This is justified by the fact that the latter is the only method stage performed in iterations. As a consequence, the required time is highly dependent not only on the initial number of segments, but, most of all, on their configuration — which dictates if planar regions can actually merge together (see Figure 5.5). When a new region is formed, the whole process needs to start all over, in order to check for merges in the new available set of planar regions. The number of merges also increases in cases where the input mesh is over segmented, mainly due to noise **[(h), (i)]**.

Even with this, the method can still produce the simplified version of a given mesh in a matter of seconds. Again, a comparison between model **(a)** and models **(b), (c)** reveals that the method performs better on individual buildings, rather than entire urban scenes. The fast execution times in the latter models further proves that our method can be possibly incorporated in urban scene reconstructions, as long as buildings can be isolated from their surroundings with semantic segmentation.

3 Processor: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, RAM: 8.00 GB



(a) Initial segmentation **(b)** After refinement

Figure 5.5: Scalability: In this figure, the execution times (in secs) are shown for the models of the available dataset, with respect to the number of segments in the initial segmentation **(a)** and after refinement **(b)**.

5.2.4 Comparisons

At this point, we would like to compare our simplification method to similar techniques, presented in Section 2.2; namely, QEM [Garland and Heckbert, 1997], SAMD [Salinas et al., 2015] and VSA [Cohen-Steiner et al., 2004]. To assess the performance of all methods, we choose, out of the available dataset (see Table 5.1), the model **(a)**, since it is quite clean and contains only building components. The source code [4], used here for QEM and SAMD, is provided by Salinas et al., while a package on VSA (under the name "Triangulated Surface Mesh Approximation") has been introduced in CGAL v4.14 [5].

QEM / SAMD

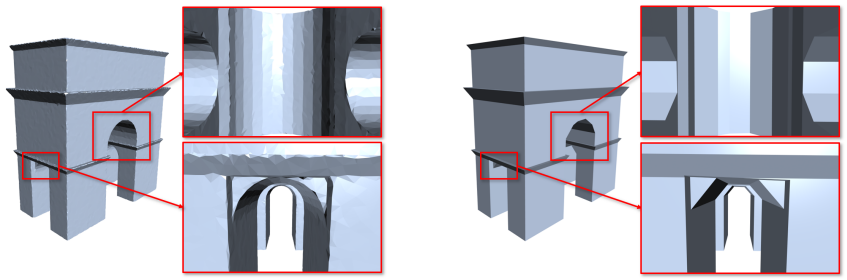
For a critical comparison between these methods and ours, we have applied all three of them in an example mesh to create a simplified version of it with the *same* number of faces (see Figure 5.6). The mean error of their results is also computed, with reference to the original model, in order to assess the degree on which they conform to the initial structure. Again, this error is estimated with the use of the Hausdorff distance, in a fashion similar to the one described in subsection 5.2.2. This analysis is presented in Figure 5.8.

At a first sight, it appears that our simplified version bears the greatest error out of the three methods, while SAMD is, by far, the best — mainly, because it is capable of approximating non-planar components of the original model much more efficiently. Still, there is an important fact we need to take into consideration. As mentioned before, our method approximates the original model *polygonal* faces, which are later replaced with co-planar triangles, so that the simplified mesh can be used in further applications. As a consequence of that, the original amount of faces for our simplified mesh is way smaller than the one we present here. In particular, our simplification for the example mesh used here initially comprises *134* faces, instead of the *338* faces after triangulation.

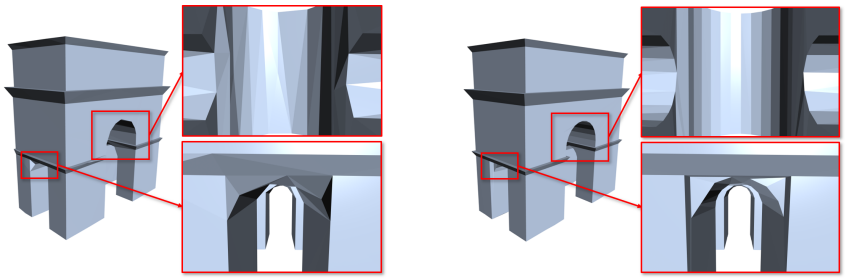
With respect to the above, the methods in comparison are applied again on the example mesh, but this time, for the original number of faces in our simplified mesh. The results are shown in Figure 5.7. As it can be observed from specific details of the example mesh in all three methods, our proposed methodology produces a far more accurate result for the same number of faces. This statement is also further supported by the error analysis, presented in Figure 5.9, where the mean error of our

4 The code is available at https://www-sop.inria.fr/members/Florent.Lafarge/codes.html
5 https://www.cgal.org

result is the smallest of all three. Furthermore, the analysis reveals the conformity error distributes evenly across the mesh surface in our result, while for the other two methods is located at specific features of the original model.

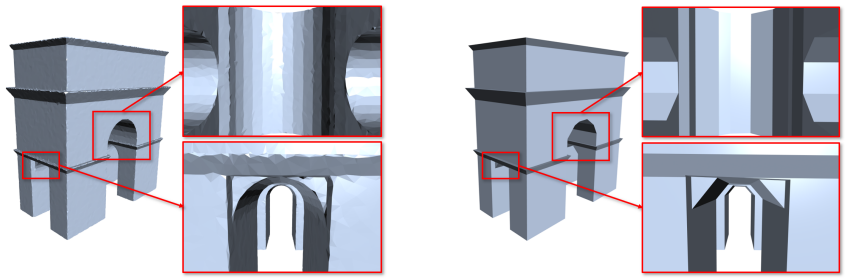


(a) Original **(b)** Ours

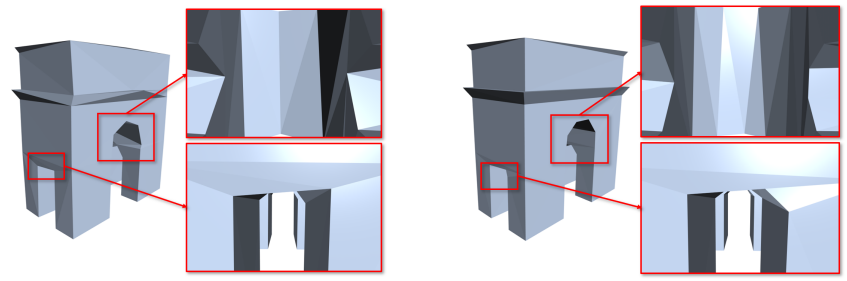


(c) QEM **(d)** SAMD

Figure 5.6: Method comparison (338 faces)



(a) Original **(b)** Ours

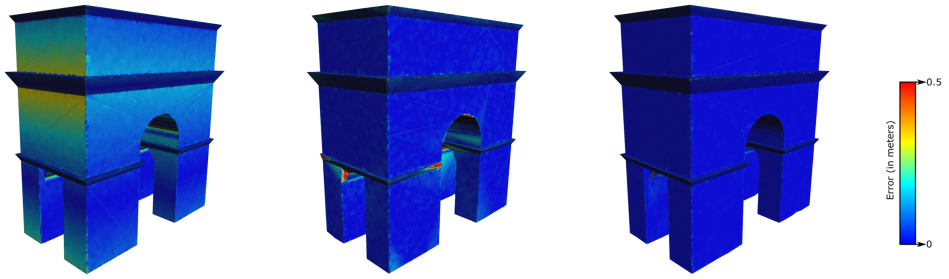


(c) QEM **(d)** SAMD

Figure 5.7: Method comparison (134 faces)

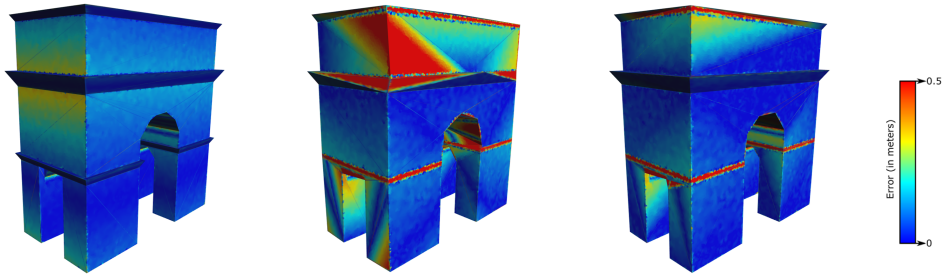
For both number of faces, the execution time of our method (4 secs) stands in the middle between the respective ones of QEM (2 secs) and SAMD (10 secs). Finally, it is important to notice that QEM and SAMD accomplish the gradual simplification of a given model, by allowing the user to specify the desired number of faces (or vertices) of the final result. On the contrary, our method is not that flexible and produces a unique, simplified version of a building mesh, corresponding to a level of detail equivalent to LOD2 (a prismatic solid with a simplified roof shape). However, the

level of model detail can be altered by adjusting the importance parameter (for an example of this, see Section 4.2).



(a) Ours (RMS=0.14) **(b)** QEM (RMS=0.09) **(c)** SAMD (RMS=0.03)

Figure 5.8: Error analysis (338 faces)

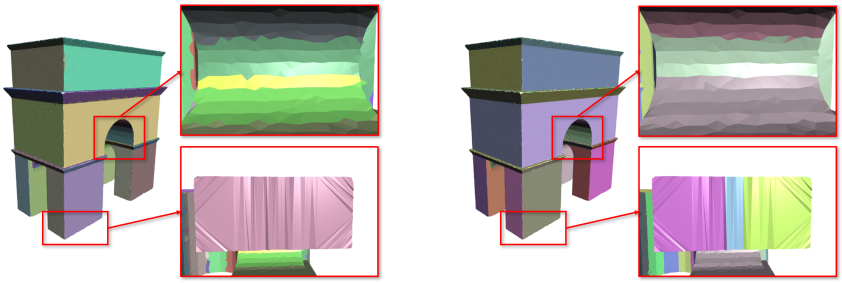


(a) Ours (RMS=0.14) **(b)** QEM (RMS=0.27) **(c)** SAMD (RMS=0.16)

Figure 5.9: Error analysis (134 faces)

VSA

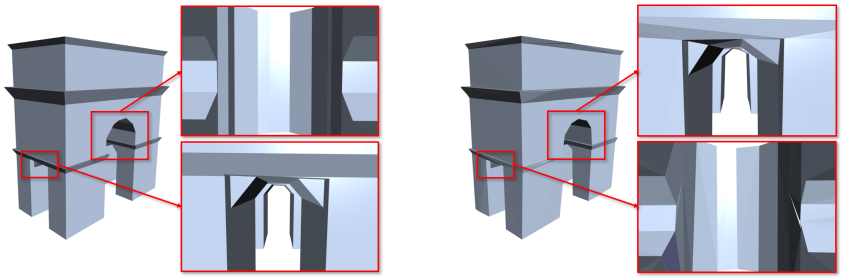
From all the simplification methods that we presented in Section 2.2, VSA is the one which is most closely associated with ours. Both our approach and VSA attempt to simplify a given model by approximating it with a set of planar components (*proxies*). Based on these proxies, an entirely new, more compact mesh is constructed which consists of *polygonal* faces, later decomposed into *triangles* if necessary.



(a) Ours **(b)** VSA

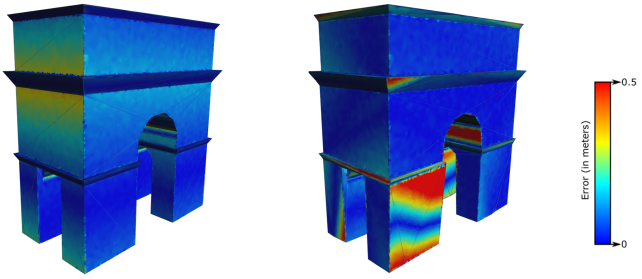
Figure 5.10: Proxy comparison

Similar to our method, VSA is also based on a mesh segmentation technique for the detection of planar components on the original mesh. Since this segmentation technique differs from ours, the resulting set of proxies for both methods might be also quite different (see Figure 5.10). Therefore, a possible comparison between the two methods can be conducted by analysing their respective results for the *same* number of proxies. Specifically, we apply VSA on the example mesh for a number of proxies equal to the amount of segments detected by our method *before*



(a) Ours **(b)** VSA

Figure 5.11: Method comparison (73 proxies)



(a) Ours (RMS=0.14) **(b)** QEM (RMS=0.18)

Figure 5.12: Error analysis (73 proxies)

the refinement step (see Figure 5.11). Finally, we conduct an error analysis for both results, again based on the Hausdorff distance (see Figure 5.12).

It can be deducted from our analysis that the error of VSA is slightly higher than that of our own approach. Furthermore, one of the main inputs of this method is the number of proxies needed to be detected by its segmentation technique — meaning that an accurate approximation for the original model can be only accomplished through a try-and-error process. Atlhough it does not ensure the topological validity of the simplified mesh, still it is applicable — unlike our approach – to building models which consist of both planar and not-planar primitives.

6 CONCLUSIONS

In this final chapter, the research questions of this graduation project are reviewed in order to assess the degree in which they have been finally addressed. Based on this, our contributions to the current state of the art are presented, along with the limitations of our proposed methodology. With respect to these limitations, some future work is also recommended.

6.1 RESEARCH OVERVIEW

At this point, we review the research questions we have defined for this graduation project in Section 1.2. For each question, a short answer is provided, based on the degree the question was finally addressed. At the same time, these answers are supported by evidence, which has been presented in previous chapters.

QUESTION:

Is it possible to obtain a simpler, compact representation of a building mesh, while preserving the initial structure?

ANSWER: **YES**

In this thesis, we have presented a methodology for the simplification of a given MVS building mesh, along with details on its implementation in a programming environment. This methodology is based on the assumption that information of any kind on the original building (imagery, point cloud etc.) is inaccessible, so the only input required is the building mesh itself (closed or open). The results, presented in subsection 5.2.1, reveal that our method produces a polygonal surface mesh, consisting of a less amount of faces than its original counterpart. With this, a simpler, compact representation of a building model is achieved in a reasonable execution time, which reduces the need for memory space significantly (see subsection 5.2.3).

Furthermore, we have offered an explicit definition for the structure of a building, purely for the purposes of this project. According to it, the structure can be possibly defined as the assembly of the building primitives (components), along with their interrelationships (configuration) in 3D space. This notion of structure is further incorporated in the context of our methodology with the form of a *structure graph*; an undirected graph whose vertices correspond to an individual building primitive, while each edge represents an adjacency relation between two primitives.

The structure graph serves as the cornerstone of our method and practically, dictates the rules by which the simplified mesh should be constructed in order to preserve the initial structure. In this way, the result of our simplification conforms as much as possible to the original model, thus minimizing distortion; namely, the distance between the input mesh and its simplified representation. In subsection 5.2.2, it is shown that the mean error of our simplified versions is small in general, while it stands in parallel with other methods, such as QEM and SAMD (see subsection 5.2.4).

QUESTION:

Can we ensure that the resulting mesh is free of topological defects?

ANSWER: **YES**

As an additional condition, we demanded that the result of our simplification method should be a compact representation of the original building mesh, which, at the same time, is topologically valid. This condition should hold true, even if any geometric and topological defects are present in the original state. This way, we ensure that the building model can be used in further applications, such as urban planning and simulations.

To achieve that, our methodology takes advantage of any available information from the input mesh, such as the coverage of a candidate face from the respective planar region. However, simplification is not accomplished here by reducing the number of simplexes in the input — an approach followed by techniques, such as QEM and SAMD. Instead, similar to VSA, a new mesh is constructed with the help of the structure graph, in which the primitives identified in the original are represented by simpler shapes. The main advantage of this is that any topological errors in the original model are not inherited in the simplified version — see Section 2.2 for examples.

To address any geometric or topological inconsistencies that may occur from flaws in our structure graph, a binary linear problem is formulated, originally introduced by Nan and Wonka [2017]. This problem imposes certain hard constraints that ensure that the final, simplified mesh is both manifold and watertight; in other words, each of its edges is adjacent to two faces. In this way, most defects (self-intersections, isolated edges etc.) are eliminated and the problem of consistent orientation remains to be solved in a post-processing step. Since the simplified mesh comprises polygonal faces of different shapes, it is finally converted into a triangular mesh to achieve uniformity.

Proof on the topological validity of our results can be found in subsection 5.2.1. There, both the Euler characteristics and the genus of the simplified meshes are examined to assert that they are topologically equivalent to a closed polyhedral surface.

6.2 DISCUSSION

6.2.1 Simplification or reconstruction?

In this thesis, we have proposed a methodology that results in a compact building model, given a MVS mesh as input. Although our method is presented along the thesis as a *simplification* technique, still its numerous similarities with *building model reconstruction* (see Section 2.1) may give rise to the following question: *Is this a* ***simplification*** *or* ***reconstruction*** *method?*

The main reason for this question to arise is that there is not a clear distinction between simplification and reconstruction. In general, simplification aims to a compact representation for a given mesh which consists of less faces than the original one, while the main objective of reconstruction techniques is the production of mesh representation from various sources (imagery, point cloud, mesh etc.), regardless of the complexity. Furthermore, simplification techniques, such as QEM and SAMD, often focus on the reduction of the simplexes forming the original mesh; on the contrary, reconstruction methods create an entirely new mesh based on information from the original.

Nonetheless, there are several techniques encountered in scientific literature which are clearly assigned to one category and at the same time, share several traits of the other. Such an example is the reconstruction technique introduced by Nan and Wonka [2017]; while its primary focus lies on the production of polygonal surface meshes from point clouds, these meshes also need to be lightweight, consisting of only a few faces. In contrast, VSA — which is closely related to our own method — simplifies a given mesh by approximating it with a set of planar proxies, finally used for the construction of an entirely new mesh.

From the above, it becomes apparent that whether our method could be categorized as a *simplification* or *reconstruction* technique is mostly a matter of choice. Here, the method is characterized as *simplification* since the primary goal of our project was the production of *simpler* versions for already-existing models and not the *construction* of a model for a given object in the first place. In any case, it is our honest belief that our proposed methodology is able to contribute to the current state of the art for both simplification and reconstruction.

6.2.2 Contributions

Along this project and in this thesis, several concepts have been presented for the formulation of our methodology. Although these concepts are not novel and they might be traced in the related work, it is still our belief that our approach on them can contribute to the current state of the art. Our most notable contributions are the following:

- **Structure definition**: Although various notions on structure can be found in scientific literature [Mitra et al., 2013], still a universally accepted definition has not been formulated yet. This task proves to be even more difficult for natural objects with no distinct geometry — for example, vegetation. In our case, however, we are dealing with buildings, which are generally characterized by geometric regularity. This allowed us to form our own definition of structure, in order to captivate this characteristic in the original mesh and preserve it, as much as possible, during simplification. Of course, it is possible for this definition to be adopted or even, further extended for other applications.

- **Mesh Segmentation**: For the purposes of this project, we have devised a region growing algorithm for the segmentation of a given mesh into planar components. This algorithm is not radically new, in comparison to similar approaches, and of course, is limited only to planar geometries (spheres, cylinders and cones are excluded). Nonetheless, several elements of it (for example, the inheritance of planarity from vertices to planes) can be further used in segmentation techniques or other applications. The set of criteria, used for the refinement of our segmentation, can also contribute to the research for the detection of planar shapes in meshes, especially for structural scales.

- **Structure graph**: One of the novelties introduced for this project is that of the *structure graph*. Similar data structures have appeared in literature (remember the *graph of proxies* in the work of Salinas et al. [2015]), but the way this graph is utilized here is completely different. Apart from the definition of structure in our method, the graph basically dictates also the formation of the simplified version for the input mesh, by indicating which pairwise intersections of the supporting planes should be computed or not. Contrary to other methods for the reconstruction of buildings, which rely on the pairwise intersection of *all* available planes, this approach dramatically reduces the amount of necessary computations, thus maximizing efficiency.

- **Importance**: Another novelty is the definition of an importance threshold, for the selection of planar components from the initial mesh segmentation. Similar to the structure graph, this concept does not appear for the first time in our method; however, its utilization differs here. Similar thresholds are defined also in other methods (examples can be found in the work of Verdie et al. [2015]), but they are used only to address over-segmentation and to avoid selecting segments, based only on the amount of their faces. Furthermore, these thresholds are set in reference to a specific region area, meaning that they are valid only for a certain model scale. Importance is, instead, assigned here to regions, according to the proportion between *their* area and the *total* surface area of the mesh. Not only that, the adjustment of this threshold can possibly dictate the level of detail for our simplified mesh.

- **Building Scaffold**: Similar structures have appeared in related work (remember the work of Wang et al. [2016]) for the regularization of meshes, but not their *construction*. Contrary to these methods, the scaffold is produced here with information available only in the original mesh and not other sources (such as imagery). In this way, both the borders of the initial planar regions and their adjacencies are preserved in the result of our simplification. Later on, the scaffold is also used to define the faces of the simplified mesh, with the help of 2D arrangements formed out of its edges.

- **Implementation**: For the implementation of our methodology in a programming environment, we have presented several techniques, especially to address issues related to arithmetic precision or problems of computational geometry. While these techniques might not be always applicable, they could still inspire the formulation of other approaches in order to solve such problems.

6.3 LIMITATIONS

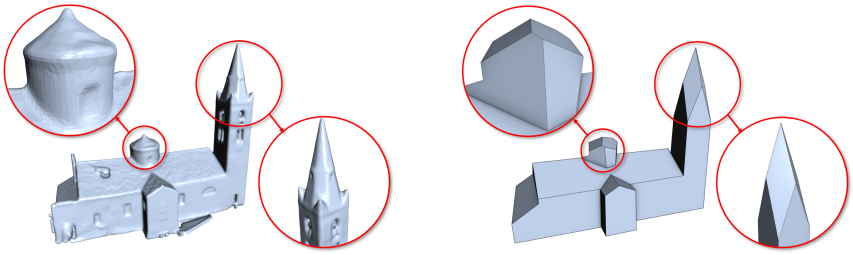
Apart from the contributions of our methodology in the current state of the art, there are also certain limitations of it that we would like to discuss in this section. These limitations actually reveal when our simplification technique is bound to produce an acceptable, simplified version of a given building mesh or where it fails. With this information at hand, it is possible to recommend some future work in order to address the various issues and improve the viability of our proposed framework.

6.3.1 Non-planar components

The first of these limitations is closely related with our assumption that the input mesh consists of only planar components; namely, the building primitives can be approximated by planes. While this assumption is valid for the majority of building types encountered in a typical urban environment, still there are cases where this fact no longer holds true. An example of this are components with curvature, such as cones and cylinders. Although these components can be approximated with a set of planes, still the resulting shape does not conform accurately to the original (see Figure 6.1).

6.3.2 Structure definition

The main issue in our simplification method currently stems from our attempt to combine two different properties for the resulting mesh: **(a)** *structure awareness* and **(b)** *topological validity*. In our proposed framework, the notion of structure is explicitly expressed through the concept of the *structure graph*, while the topological validity is



(a) Original mesh **(b)** Simplified mesh

Figure 6.1: Curved components: Although curves components of the original mesh can be approximated by plane sets, still the resulting shape does not conform to the original geometry.

imposed through the hard constraints, included as part of our optimization process. However, the combination of these properties does not always function properly.

As mentioned in Section 1.2, 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 relations. This requires that both these elements need to be recovered, so that the model structure is completely acquired. This is not always possible, mainly because of two reasons — both related to mesh segmentation (see Figure 6.2):

(a) The set of building primitives is not fully recovered. This occurs due to the defined distance threshold or when the input mesh is overall "smooth" — meaning that the curvature on the borders of planar regions changes gradually. As a consequence of that, 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 occurs due to the fact that the region extends in a limited area of the mesh. This means that it shares a common border (*is adjacent to*) with a number of regions, smaller than the one required to define a closed shape.

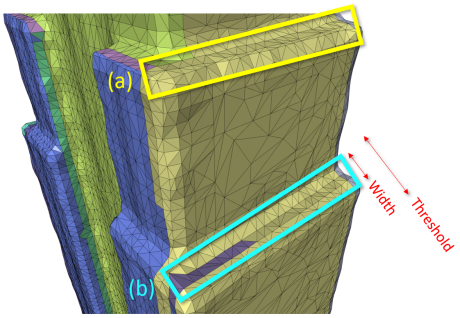
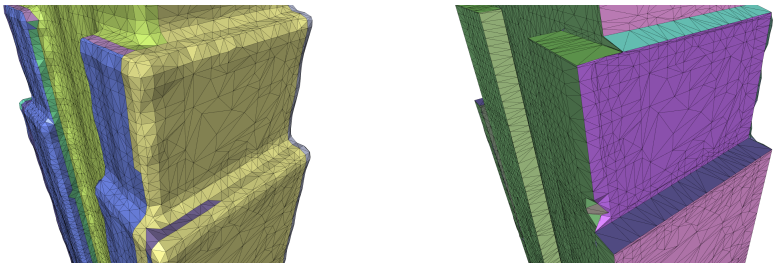


Figure 6.2: Incomplete structure: The structure of a building model might not be fully recovered, either because components cannot be detected via mesh segmentation **(a)** or the adjacency relations, necessary for the definition of their borders **(b)**, are not retrieved.

The above issues might be the cause for the presence of holes in the proxy mesh, constructed out of the initial set of candidate faces. Since the hard constraints of the optimization process enforce the simplified mesh to be manifold, these parts are even eventually excluded or even worse, they compromise the manifoldness of the final result in such a degree that in the end, no model is produced. With respect to that, the structure graph provides, on the one hand, a means to achieve computational efficiency and high conformity of our simplified version to the original

mesh. Unfortunately, at the same time, it does not guarantee that the proxy mesh would be enough by itself for the hard constraints to result into a topologically valid mesh.

Nevertheless, there is possibly a way to address this problem for a lot of cases, based on our experimentation. One common way to lower the level of noise in a given mesh depends on the detection of planar primitives in it and the projection of their faces to their respective *supporting plane*. It is also possible to incorporate this approach in our segmentation of the input mesh, by projecting a planar region to its plane after its definition, thus achieving the upcoming effects (see Figure 6.3):



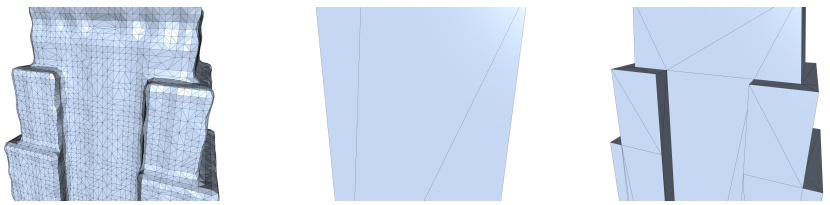
(a) Original segmentation **(b)** With region projection

Figure 6.3: Region projection: The projection of the region vertices on their supporting planes "stretches" the faces of narrow, elongated building components, thus facilitating their detection and increasing their importance.

(a) The projection of vertices on their reference plane increases the face area of narrow, elongated components, thus facilitating their detection during segmentation.

(b) This rise in the face area of a given region also increases its importance and as a result, its probability to be selected for simplification with the importance threshold.

The above effects enable the enrichment of the structure graph, with the addition of both primitives and adjacency relations, thus completing any information missing. This enrichment also has a profound effect to the final result of our simplification (see Figure 6.4). Regardless of the effectiveness of this approach, a revision on the extraction of adjacencies between planar regions is still required for our method to be further improved.



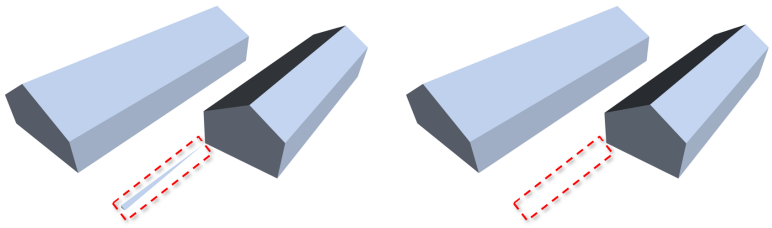
(a) Original mesh **(b)** Simplified mesh **(c)** With region projection

Figure 6.4: Region projection and simplification: For a given mesh **(a)**, finer parts of the initial parts cannot be retrieved, causing their omission from the simplified version **(b)**. However, the projection of regions to their supporting planes enhances these features, thus enabling their addition in the final result **(c)**.

6.3.3 Miscellaneous

The hard constraints to assert the manifoldness of our simplified mesh might also result into various by-products, for different segmentations of the input mesh (see Figure 6.5). The main cause for these by-products is the fact that the constraints

utilize the set of candidate faces to construct as many closed, polygonal surfaces as possible. As a consequence of that, there are cases where surfaces are still produced, despite the low confidence of their faces.



(a) $d = 0.80$, $imp = 1.0$ **(b)** $d = 0.85$, $imp = 1.0$

Figure 6.5: By-products: Different segmentations of the input mesh might result into closed, polygonal surfaces which cannot be related to building components.

Finally, some further limitations of our method have been already discussed in the context of previous chapters. One of them is the handling of corners, which serve as the intersecting points of four or more planar regions (for an example, see Figure 4.3). Although this approach facilitates the construction of the building scaffold, a representation of these corners with a unique vertex in our simplified mesh is still needed. Another limitation is related to the automatization of the method. As of right now, the method is semi-automatic, mainly requiring from the user to define the necessary importance threshold and possibly, the distance for the mesh segmentation. Nevertheless, a fully-automated process would be much more user-friendly and allow the incorporation of our methodology in further applications.

6.4 FUTURE WORK

Based on the limitations presented in Section 6.3, we would like to recommend some future work in this final section of this thesis. These recommendations aim to further improve our proposed methodology and allow its incorporation into other applications. For this purpose, some of our suggestions are the following:

- **Mesh Segmentation**: As mentioned in Section 6.3, many current limitations of our method are closely related to mesh segmentation. Although we introduced our own region growing algorithm to address the detection of planar regions in a given mesh, experimentation with other, available techniques could be proven useful. In this way, our segmentation method could be further improved or even, replaced in order to produce better results. For this purpose, it is necessary to study the combination of our simplification framework with different mesh segmentation techniques, especially those of the RANSAC family. Furthermore, the detection of geometries other than planar (spheres, cylinders, cones), via mesh segmentation, would allow the simplification of a wider variety of building types.

- **Structure Definition**: Our definition of building structure has proven sufficient for the purposes of our project, but some extensions to it are possible. As of right now, the interrelationships between primitives are limited to topological relations and specifically, only adjacencies. The addition of geometric relations, such as orthogonality (an attribute, relating the orientation of a primitive in reference to the z-axis), could facilitate the *regularization* of the simplified mesh. An example of that could be the construction of faces, exactly parallel to the z-axis, for the definition of building walls. Of course, this would require, at the

same time, the construction of multiple structure graphs, each one with the same amount of primitives, but different types of relations connecting them.

- **Structure Graph**: Even without redefining our structure definition, still some changes are necessary in order for the structure graph to be functional, especially in its construction. Currently, the adjacencies recorded in the graph are detected by locating common vertices in region borders. However, even this naive approach has proven to be quite demanding in several cases, thus forcing us to search for an alternative solution. A possible way to address this issue could be to regard as adjacent regions those, for which the minimum distance between their vertex sets is smaller than a predefined threshold — although this would lead to the addition of one extra parameter, highly dependent on the model scale.

- **Building Scaffold**: The construction of the building scaffold could be also revisited, especially for handling the intersections of four or more planar regions. To address this particular issue, this type of corners need to be located first in the structure graph. This can be accomplished by examining all graph *cliques* and isolating those of *order* higher than three (3). Then, the vertices, resulting from the intersection of plane triplets for a group of such primitives, could be merged into one, unique vertex.

- **Automatization**: Further automatization would make our simplification technique much more user-friendly and allow its incorporation in further applications, such as urban scene reconstruction. A fixed value could be probably assigned to the distance parameter, if the model scale in study is also fixed. Although the importance parameter remains user-defined in order to allow the level of detail for the simplified mesh, it could be still defined as an constant and then, further details could be obtained via *iconization*.

- **Level of Detail**: As of right now, our method produces simplified versions of building meshes for LOD2 — buildings with simple roof structure, without architectural details. However, as we have shown in Section 4.2, it is possible to alter the level of detail through the adjustment of the importance parameter, originally introduced to select the planar segments for simplification. For buildings with flat roofs, this allows the construction of LOD1 models, but still not of LOD0 (building footprint).

  To address this problem, a possible solution could be the addition of semantic information to our simplified models, through geometric regulations. In this case, the face orientation could be proven useful; for example, faces parallel to the z-axis could correspond to building walls, while those perpendicular to it and with low z-value could be assigned to the building floor. According to this approach, faces to form the simplified model are selected, according to their orientation, in a post-processing step, depending on the desired level.

BIBLIOGRAPHY

Agarwal, P. K. and Sharir, M. (2000). Arrangements and Their Applications. In *Handbook of Computational Geometry*, pages 49 – 119. North-Holland, Amsterdam.

Attene, M., Campen, M., and Kobbelt, L. (2013). Polygon mesh repairing: An application perspective. *ACM Comput. Surv.*, 45(2):15:1–15:33.

Awrangjeb, M. and Lu, G. (2014). Automatic Building Footprint Extraction and Regularisation from LIDAR Point Cloud Data. In *2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8.

Bernardini, F., Mittleman, J., Rushmeier, H. E., Silva, C. T., and Taubin, G. (1999). The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Trans. Vis. Comput. Graph.*, 5(4):349–359.

Biljecki, F., Heuvelink, G., Ledoux, H., and Stoter, J. (2018). The effect of acquisition error and level of detail on the accuracy of spatial analyses. *Cartography and Geographic Information Science*, 45(2):156–176.

Biljecki, F., Ledoux, H., and Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25 – 37.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.

Billen, R., Cutting-Decelle, A., Marina, O., Duarte de Almeida, J.-P., Caglioni, M., Falquet, G., Leduc, T., Métral, C., Moreau, G., Perret, J., Rabino, G., García, R., Yatskiv, I., and Zlatanova, S. (2014). *3D City Models and urban information: Current issues and perspectives*.

Bódis-Szomorú, A., Riemenschneider, H., and Gool, L. J. V. (2015). Superpixel meshes for fast edge-preserving surface reconstruction. In *CVPR*, pages 2011–2020. IEEE Computer Society.

Campagna, S., Kobbelt, L., and Seidel, H.-P. (1998). Directed Edges; A Scalable Representation for Triangle Meshes. *J. Graph. Tools*, 3(4):1–11.

Cohen-Steiner, D., Alliez, P., and Desbrun, M. (2004). Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914.

Commandeur, T. (2012). Footprint decomposition combined with point cloud segmentation for producing valid 3D models. Master's thesis, Delft University of Technology.

Donkers, S., Ledoux, H., Zhao, J., and Stoter, J. (2015). Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, 20.

Douglas, D. and Peucker, T. (1973). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

Fang, H., Lafarge, F., and Desbrun, M. (2018). Planar Shape Detection at Structural Scales. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, United States.

Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395.

Furukawa, Y. and Hernández, C. (2015). Multi-View Stereo: A Tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2):1–148.

Garland, M. and Heckbert, P. S. (1997). Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Gatzke, T. and Grimm, C. (2006). Estimating Curvature on Triangular Meshes. *International Journal of Shape Modeling*, 12:1–28.

Haala, N. and Kada, M. (2010). An update on automatic 3D building reconstruction.

Holzmann, T., Oswald, M., Pollefeys, M., Fraundorfer, F., and Bischof, H. (2017). Plane-based Surface Regularization for Urban 3D Reconstruction. In *28th British Machine Vision Conference*.

Jain, K., Gurjar, S. P., and Mandla, V. (2013). Virtual 3D City modeling: Techniques and Applications. volume XL-2/W2.

Jonsson, M. (2016). Make it Flat : Detection and Correction of Planar Regions in Triangle Meshes. Master's thesis, Linköping University, Computer Vision.

Kazhdan, M. M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Symposium on Geometry Processing*, volume 256 of *ACM International Conference Proceeding Series*, pages 61–70. Eurographics Association.

Kelly, T., Femiani, J., Wonka, P., and Mitra, N. J. (2017). BigSUR: large-scale structured urban reconstruction. *ACM Trans. Graph.*, 36(6):204:1–204:16.

Lancelle, M. and Fellner, D. (2010). Current issues on 3D city models. *Proceedings of the 25th International Conference in Image and Vision Computing*, pages 363—-369.

Lewis, R. and Sequin, C. (1998). Generation of 3D building models from 2D architectural plans. *Computer-Aided Design*, 30:765–779.

Li, M., Nan, L., Smith, N., and Wonka, P. (2016). Reconstructing building mass models from UAV images. *Computers & Graphics*, 54:84–93.

Mitra, N. J., Wand, M., Zhang, H. R., Cohen-Or, D., Kim, V. G., and Huang, Q. (2013). Structure-aware shape processing. In *SIGGRAPH Asia 2013, Hong Kong, China, November 19-22, 2013, Courses*, pages 1:1–1:20.

Nan, L. and Wonka, P. (2017). PolyFit: Polygonal Surface Reconstruction from Point Clouds. In *ICCV*, pages 2372–2380. IEEE Computer Society.

Pauly, M., Gross, M., and Kobbelt, L. P. (2002). Efficient Simplification of Point-sampled Surfaces. In *Proceedings of the Conference on Visualization '02*, VIS '02, pages 163–170, Washington, DC, USA. IEEE Computer Society.

Salinas, D., Lafarge, F., and Alliez, P. (2015). Structure-Aware Mesh Decimation. *Comput. Graph. Forum*, 34(6):211–227.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Comput. Graph. Forum*, 26(2):214–226.

Sierksma, G. and Zwols, Y. (2015). *Linear and integer optimization: Theory and practice*. Taylor & Francis Group, 3 edition.

Smith, C. (2006). *On Vertex-vertex Systems and Their Use in Geometric and Biological Modelling*. PhD thesis, Calgary, Alta., Canada, Canada. AAINR19574.

Süveg, I. (2003). *Reconstruction of 3D building models from aerial images and maps*. PhD thesis. Series: Netherlands Geodetic Commission NCG : Publications on Geodesy : New Series; 53. PhD Thesis Netherlands Geodetic Commission, summaries in English and Dutch.

The Computational Geometry Algorithms Library (CGAL) (2018). https://www.cgal.org/.

The Point Cloud Librarty (PCL) (2018).

Tomljenovic, I., Höfle, B., Tiede, D., and Blaschke, T. (2015). Building Extraction from Airborne Laser Scanning Data: An Analysis of the State of the Art. *Remote Sensing*, 7:3826–3862.

Trudeau, R. J. (2015). *Introduction to graph theory*.

Veljanovski, T., Kanjir, U., and Oštir, K. (2011). Object-based image analysis of remote sensing data. *Geodetski vestnik*, 55:641–664.

Verdie, Y., Lafarge, F., and Alliez, P. (2015). LOD Generation for Urban Scenes. *ACM Trans. Graph.*, 34(3):30:1–30:14.

Vieira, M. and Shimada, K. (2005). Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Design*, 22(8):771–792.

Wang, J., Fang, T., Su, Q., Zhu, S., Liu, J., Cai, S., Tai, C., and Quan, L. (2016). Image-Based Building Regularization Using Structural Linear Features. *IEEE Trans. Vis. Comput. Graph.*, 22(6):1760–1772.

Wu, K., Rashad, M., and Khamiss, M. (2017). A Review on Mesh Segmentation Techniques. *International Journal of Engineering and Innovative Technology (IJEIT)*, 6.

Xiong, B., Elberink, S. O., and Vosselman, G. (2014). Building Modeling from Noisy Photogrammetric Point Clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-3*.

Yin, X., Wonka, P., and Razdan, A. (2009). Generating 3D Building Models from Architectural Drawings: A Survey. *IEEE computer graphics and applications*, 29:20–30.

Zhu, L., Shen, S., Gao, X., and Hu, Z. (2018). Large Scale Urban Scene Modeling from MVS Meshes. In *ECCV (11)*, volume 11215 of *Lecture Notes in Computer Science*, pages 640–655. Springer.

A APPENDIX

Algorithm 1 Region growing with k-ring planarity

Input:

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

Output: Triangles assigned to segments

Initialize:

- Regions $\{R\} \leftarrow \emptyset$ // a list of list of integers (face indices)
- Available Faces $\{F\} \leftarrow \{1, 2, ..., m\}$ // a list of integers (face integers)

while $\{V\} \neq \emptyset$ **do**
- Current Region: $\{R_c\} \leftarrow \emptyset$ // face indices
- Current Seeds: $\{S_c\} \leftarrow \emptyset$ // face indices

Face with highest planarity $\{F\} \leftarrow f_{max}$
$\{S_c\} \leftarrow \{S_c\} \cup f_{max}$
$\{V\} \leftarrow \{V\} \setminus f_{max}$
Find k-ring neighbouring faces $\{B_c\} \leftarrow \Omega_k\{f_{max}\}$
Fit plane to Neighbors $plane \leftarrow \text{PCA}\{B_c\}$
while $\{S_c\} \neq \emptyset$ **do**
$\{B_c\} \leftarrow \emptyset$ // face indices
for s in $\{S_c\}$ **do**
$\{B_c\} \leftarrow \{B_c\} \cup \Omega_1\{s\}$ // 1-ring neighbors
$\{S_c\} \leftarrow \{S_c\} \setminus s$
end for
for B in $\{B_c\}$ **do**
$\{v_B\} \leftarrow$ vertices of B
if $B \in \{F\}$ and $dist(B, plane) \leq d^t$ **then**
$\{R_c\} \leftarrow \{R_c\} \cup B$
$\{V\} \leftarrow \{V_c\} \setminus \{v_B\}$
$\{S_c\} \leftarrow \{S_c\} \setminus \{v_B\}$
end if
end for
Re-fit plane to current region $plane \leftarrow \text{PCA}\{R_c\}$
end while
Add current region $\{R\} \leftarrow \{R_c\}$
end while

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the arsclassica package by Lorenzo Pantieri, which is an adaption of the original classicthesis package from André Miede.