Feature Preserving Decimation of Urban Meshes

By

Vivek Kamra

A Thesis Submitted to
Saint Mary's University, Halifax, Nova Scotia
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Applied Science.

September 9, 2022, Halifax, Nova Scotia

ACKNOWLEDGEMENTS

I would like to express gratitude for my respected guide and mentor, Dr. Jiju Poovvancheri, for his assistance and dedicated involvement in every step throughout the process, this dissertation would have never been possible. I am grateful for his valuable suggestions, support, encouragement, understanding and good wishes over these past two years. I am thankful to the members of my supervisory committee, (Dr. Yasushi Akiyama and Dr. Khan Rahaman) for reviewing my work and giving their valuable suggestions. I would also like to thank Dr. Somaye ArabiNaree for proofreading and valuable suggestions. Thanks to FGSR and Saint Mary's University for giving me this opportunity. In addition, I would like to thank my parents and grandparents for their blessings and love. This work is dedicated to the memory of my late grandmother, who always believed in me. Lastly, I could not have completed this dissertation without the support of my friends, who provided stimulating discussions and happy distractions to keep me balanced during tough times.

Place: Halifax, Nova Scotia Vivek Kamra

Date: September 9, 2022

ABSTRACT

"Feature Preserving Decimation of Urban Meshes"

By Vivek Kamra

Commercial buildings as well as residential houses represent core structures of any modern day urban or semi-urban areas. Consequently, 3D models of urban buildings are of paramount importance to a majority of digital urban applications such as city planning, 3D mapping and navigation, video games and movies, among others. However, current studies suggest that existing 3D modeling approaches often involve high computational cost and large storage volumes for processing the geometric details of the buildings. Therefore, it is essential to generate concise digital representations of urban buildings from the 3D measurements or images, so that the acquired information can be efficiently utilized for various urban applications. Such concise representations, often referred to as "lightweight" models, strive to capture the details of the physical objects with less computational storage. Furthermore, lightweight models consume less bandwidth for online applications and facilitate accelerated visualizations. In this thesis, we provide an assessment study on state-of-the-art data structures for storing lightweight urban buildings. Then we propose a method to generate lightweight yet highly detailed 3D building models from LiDAR scans. The lightweight modeling pipeline comprises the following stages: mesh reconstruction, feature points detection and mesh decimation through gradient structure tensors. The gradient of each vertex of the reconstructed mesh is obtained by estimating the vertex confidence through eigen analysis and further encoded into a 3 X 3 structure tensor. We analyze the eigenvalues of structure tensor representing gradient variations and use it to classify vertices into various feature classes, e.g., edges, and corners. While decimating the mesh, feature points are preserved through a mean cost-based edge collapse operation. The experiments on different building facade models show that our method is effective in generating simplified models with a trade-off between simplification and accuracy.

Date: September 9, 2022

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **iii**

LIST OF TABLES **vi**

LIST OF FIGURES **1**

1 Introduction **2**

1.1 Introduction 2

1.2 Light-Weight Modeling 4

1.3 Motivation and Challenges 4

1.4 Contributions 7

1.5 Organization 8

2 Data Structures for Light Weight Models **10**

2.1 Polygonal Meshes 11

2.1.1 Indexed Mesh Storage 15

2.1.2 Winged-Edge Structure 16

2.1.3 Half-Edge Structure 17

2.2 Constructive Solid Geometry 18

2.3 Boundary Representation (B-rep) 22

2.4 Stellar Decomposition 26

2.5 Neural Radiance Fields 28

2.6 Assessment of Different Representations 29

3 Light-weight Modeling Algorithms **35**

3.1 Mesh Decimation 35

3.2 Geometric Abstractions 36

3.3 Deep Learning based Algorithms 37
3.4 Level of Detail Modeling 40
3.5 Summary and Remarks 41

4 Mesh Decimation Through Gradient Tensors **44**
4.1 Overview 44
4.2 Mesh Reconstruction 45
4.3 Feature Points Detection 48
4.4 Decimation 51
4.5 Results and Experiments 55
4.5.1 Qualitative Analysis 56
4.5.2 Quantitative Analysis 57
4.5.3 Comparison 58

5 Conclusion and Future Directions **62**
5.1 Summary 62
5.2 Future Directions 64
5.3 Publications based on this Thesis 64

LIST OF TABLES

1.1 *Statistics on storage size (in GBs) of different urban-scene based meshes reconstructed from [(Tan* et al.*, 2020)] and [(Hackel* et al.*, 2017)]* 6

2.1 *Space requirements for mesh structures.* 18

2.2 *Statistics related to different 3D representations.* 33

2.3 *Comparison between data structures.* 33

4.1 *Statistics on sizes and errors for the simplification results.* 57

4.2 *Comparison with other methods based on simplicity and fidelity statistics.* . . 60

LIST OF FIGURES

1.1 *A large-scale city scene with distinct buildings, Image Courtesy: [(Kuang* et al.*, 2013)]* 3

1.2 *(a) Triangular meshes based model vs (b) Light-weight 3D model, Image Courtesy: [(Li* et al.*, 2011)]* 3

1.3 *A sequence of approximations generated using edge-collapse with QEM, Image Courtesy: [(Garland and Heckbert, 1997)]* 5

1.4 *Surface simplification of a building block, Image Courtesy: [(Bouzas* et al.*, 2020)]* 6

1.5 *Simplified mesh of a building produced by our decimation method.* 8

2.1 *A closeup view of triangular meshes based (a) Lucy model; (b) David model; and (c) Buddha model* 11

2.2 *Quadrilateral based mesh model of a mechanical piece, Image Courtesy: www.geometryfactory.com.*

2.3 *A geodome constructed using a P-Hex mesh in the Eden Peoject in UK (left); P-Hex meshes based constant mean curvature (CMC) surfaces (right), Image Courtesy: [(Wang* et al.*, 2008)]* 13

2.4 *Manifold vs non-manifold meshes, Image Courtesy: [(Steve Marschner, 2014)]* 14

2.5 *A three triangle mesh with four vertices (a,b,c,d), represented with: separate triangles (left) and shared vertices (right).* 16

2.6 *References from an edge to the neighboring faces, edges and vertices in the winged-edge structure. Image Courtesy: [(Marschner and Shirley, 2018)]* . 17

2.7 *References from half-edge to the neighboring mesh components. Image Courtesy: [(Marschner and Shirley, 2018)]* 17

2.8 *An example of CSG boolean operations. Image Courtesy: [(Dr. C.-K. Shene, 2011)]* 19

2.9 *An example of CSG operations with expression. Image Courtesy: [(Commons, 2020)]* 20

2.10 *Common geometric primitives, Image Courtesy: [(Kaiser* et al.*, 2019)]* . . . 21

2.11 *A bezier curve and its 4 control points. Image Courtesy: www.scratchapixel.com* 24

2.12 *A bezier patch.* 24

2.13 *A few examples of CP complexes: (a) A pure simplicial 3-complex with 4 tetrahedra; (b) A CP complex with three top edges, three top triangles, two top quads and a top tetrahedron; (c) A 2-dimensional pseudo-manifold with 11 triangles, Image Courtesy: [(Fellegara* et al.*, 2021)]* 27

2.14 *Stellar decomposition encoding for triangles within a region r (red square), Image Courtesy: [(Fellegara* et al.*, 2021)]* 27

2.15 *An overview of neural radiance field scene representation, Image Courtesy: [(Mildenhall* et al.*, 2020)].* 28

2.16 *Classification of common modeling techniques and a few representation results.* 29

2.17 *(a) CAD models of sample buildings (downloaded from open-source 3D modeling platform Sketchfab - https://sketchfab.com/features/free-3d-models); (b) Equivalent triangular mesh models.* 32

2.18 *3D CAD models and their simplified version. (a) Empire State (triangular mesh), (b) Empire State (B-Rep), (c) Lans (triangular mesh) and (d) Lans (B-Rep).* 33

4.1 *Our mesh decimation pipeline.* 44

4.2 *Reconstructed meshes through screened poisson surface reconstruction.* . . 47

4.3 *Feature points representing critical facade elements of buildings.* 50

4.4 *Example point clouds of the Semantic3D dataset [(Hackel* et al.*, 2017)].* . . 53

4.5 *Mesh decimation preserving feature points through structure tensor. (a) An instance of birdfountain_station1 model (b) simplified version with faces retained at feature areas.* 54

4.6 *Efficient mesh simplification of birdfountain model. From (a) to (d), initial point cloud, reconstructed mesh, feature points extracted, final simplified model retaining feature points presenting facade elements of model.* 55

4.7 *Experimental results: (a) domfountain_station1 (2), (b) domfountain_station1 (1), (c) marketplacefeldkirch_station7 (2) and (d) stgallencathedral_station6.* 56

4.8 *Comparison on birdfountain_model. (a) Initial mesh model, (b) GH's method, (c) LT's method, (d) SLA's method and (e) our model.* 59

4.9 *Comparison on domfountain_model (1). (a) Initial mesh model, (b) GH's method, (c) LT's method, (d) SLA's method and (e) our model.* 59

4.10 *Comparison on domfountain_model (2). (a) Initial mesh model, (b) GH's method, (c) LT's method, (d) SLA's method and (e) our model.* 60

5.1 *Simplification of model with curved regions. (a) Original model, (b) Simplified model* 63

CHAPTER 1

Introduction

1.1 Introduction

In recent years, 3D digital objects are widely used in many applications such as architectural and industrial design, remote sensing, virtual reality, augmented reality, video games, movies and 3D navigation, among others. While there are different ways to create 3D digital models, most of them boil down to two basic methods: building a model in 3D modeling software, or taking an object from the real world and turning it into a digital model using a 3D scanner. Unlike CAD (Computer-Aided Design), polygonal modeling and digital sculpting that allows user to design a model completely from scratch, 3D scanning allows user to create a digital copy of a real world object by capturing its geometry in the form of a point cloud. A point cloud is a set of discrete measurements acquired by sensors such as LiDAR (Light Detection and Ranging) scanner or RGB-D (Red, Green, Blue, Depth) camera. Each measurement in the 3D scan is a point in three dimensional space with x, y and z coordinates representing the spatial location of the real world object being scanned. Each point can have additional information such as the RGB color of the scanned object. Due to the recent advancements in 3D scanning, point clouds have become one of the popular representations of 3D data. However, point clouds are normally converted to triangular meshes before being used in various applications.

The points can be joined together to form surfaces as polygon meshes, which can then be used to form a complete 3D model. Geometric modeling is the process of constructing representations of the 3D shape from point cloud data where the output representation can be either parametric or nonparametric, as well as surface-based or volumetric. These polygon meshes are collection of vertices, edges and faces that defines the shape of a 3D object. The faces consist of triangles, quadrilaterals or any simple convex polygons that can be used for rendering 3D model shape. Urban scene is a 3D object and hence, one of the representations is polygon mesh which has been extensively used for urban scene representation comprising a large number of buildings; however, it is a challenging task from storage, transmission speed and rendering performance point of view. The ongoing studies related to geometric modeling shows that mil-

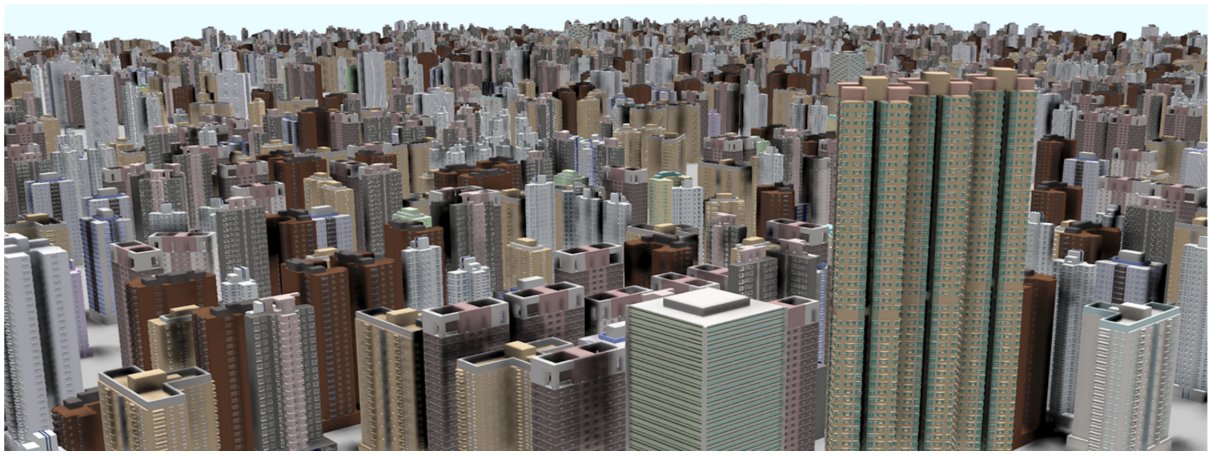


Figure 1.1: *A large-scale city scene with distinct buildings, Image Courtesy: [(Kuang* et al.*, 2013)]*

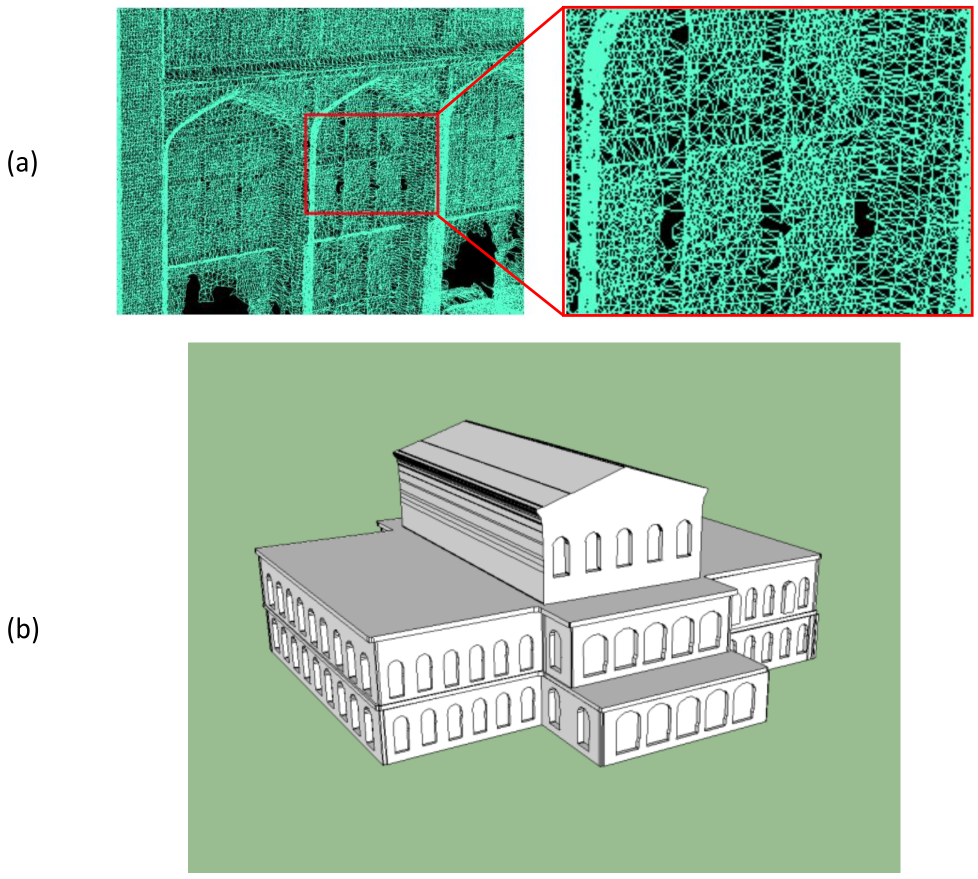


Figure 1.2: *(a) Triangular meshes based model vs (b) Light-weight 3D model, Image Courtesy: [(Li* et al.*, 2011)]*

lions of triangles are required to represent the details of the model. For instance, a city scene in Figure 1.1 with 5,061 distinct buildings based on meshes represents 671 million triangles that require 37GB of storage space. This can lead to burden on storage and performance for a large-scale urban scene model.

1.2 Light-Weight Modeling

Light-weight modeling refers to 3D representation of a model comprising minimum complexity such as less polygonal count resulting in less storage space for the model and aids in accelerated rendering and transmission of the model.

The triangular mesh model in Figure 1.2 (a), an instance of a building, for example, needs almost 400 MB of storage. Such huge meshes will slow down web-based applications. On the other hand, a corresponding light-weight model (Figure 1.2 (b)) with less vertices and faces would require less storage. These problems have been addressed before with different methods such as mesh simplification and polygonization which offers lightweight meshes with less complexity and memory requirements. However, such meshes lack the level of details required in real-world applications such as façade characteristics of a building in a 3D map. Therefore, this research investigates into generating lightweight 3D models that save memory size without missing information and retaining essential details.

1.3 Motivation and Challenges

There exist certain challenges at the very basic that needs to be addressed including the complex structures of 3D scans. Due to the irregularities in LiDAR scans, reconstructed meshes have defects. Some of the challenges in the case of LiDAR scans are the occlusions, noise and outliers as raw 3D point cloud directly obtained from laser scanners or image-based reconstruction methods are contaminated due to matching ambiguities or image imperfections such as lens distortion, sensor noise etc. This leads to the disturbing artifacts of reconstructed surfaces from the point cloud and thus, makes the modeling processes challenging.

With the advances in technology, industries now design models using computer-aided design (CAD) systems, resulting in complex, high detailed surfaces. 3D models created by surface reconstruction methods can often be very dense meshes and thus, become less suitable for web-based applications. In such cases, it is a tradeoff between the level of detail reconstructed on the surface and the amount of storage required. To achieve an acceptable level of detail in a model, we must substitute simpler approximations of the original model. Mesh simplification is a useful tool for tailoring high detailed models to the needs of individual applications and for producing more cost-effective surface models by reducing the polygon count of a large mesh using decimation algorithms. Most decimation algorithms are incremental and apply a topological operator on a mesh at each decimation step. An error metric computes the local changes that would appear in the geometry following an operator on a mesh. Therefore, an error value is associated with every possible operation that further allows to prioritize the operator with the lowest error value. One such popular mesh decimation algorithm combines the edge collapse operator with the Quadric Error Metric (QEM) [(Garland and Heckbert, 1997)] where an edge collapses (Figure 1.3) into a vertex minimizing the point-to-plane distance with regard to its local neighborhood at each decimation step.

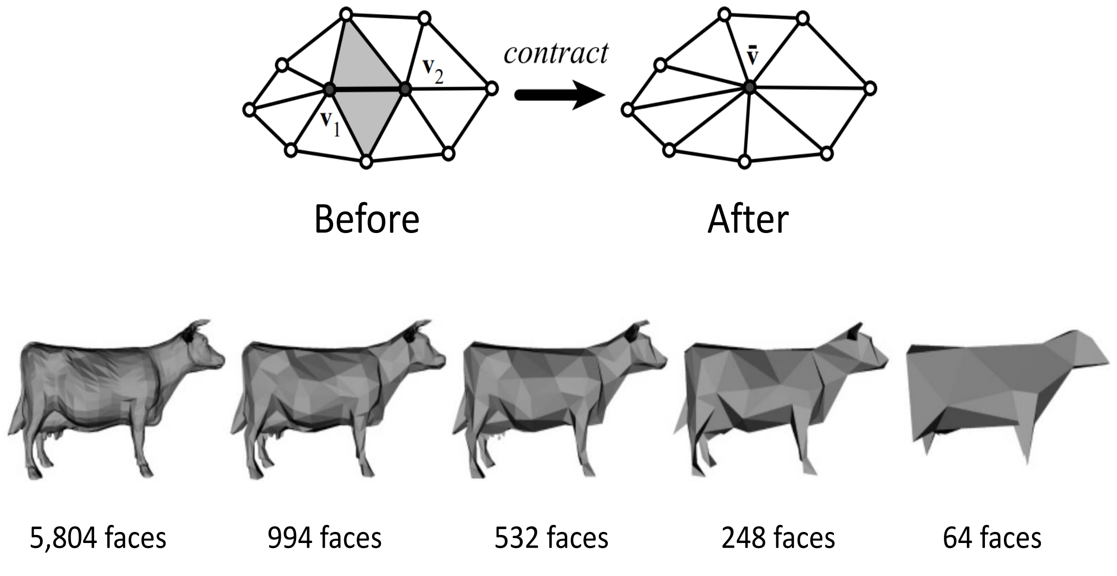


Figure 1.3: *A sequence of approximations generated using edge-collapse with QEM, Image Courtesy: [(Garland and Heckbert, 1997)]*

The approximations in Figure 1.3 show that features such as horns and hooves begin disappearing in low level models. Considering one such example of a 3D building model [(Bouzas *et al.*, 2020)] shown in Figure 1.4, the original mesh comprises 76636 faces with façade characteristics and the simplified model comprises 632 faces that retains the basic structure of the model but loses all the details on the façades. Simplified models are sufficient for some applications, e.g. city blueprint, however, for many modern applications based on interactive rendering, for instance, Google Earth, highly detailed models are mandatory. Although, polygonal meshes based 3D model preserves high level of detail through reconstruction yet they are expensive from storage and processing aspects, that makes them less suitable for web-based applications. Table 1.1 lists the number of points and faces along with storage size of different urban scene (Toronto, Canada and Central Europe) based 3D meshes reconstructed through screened poisson reconstruction method [(Kazhdan and Hoppe, 2013)] from large-scale point cloud sources. For instance, L003 from Toronto-3D: A large-scale LiDAR dataset [(Tan *et al.*, 2020)] (https://github.com/WeikaiTan/Toronto-3D) and birdfountain station1, neugasse station1 and sg27 station10 from Semantic3D [(Hackel *et al.*, 2017)] (https://www.semantic3d.net/).

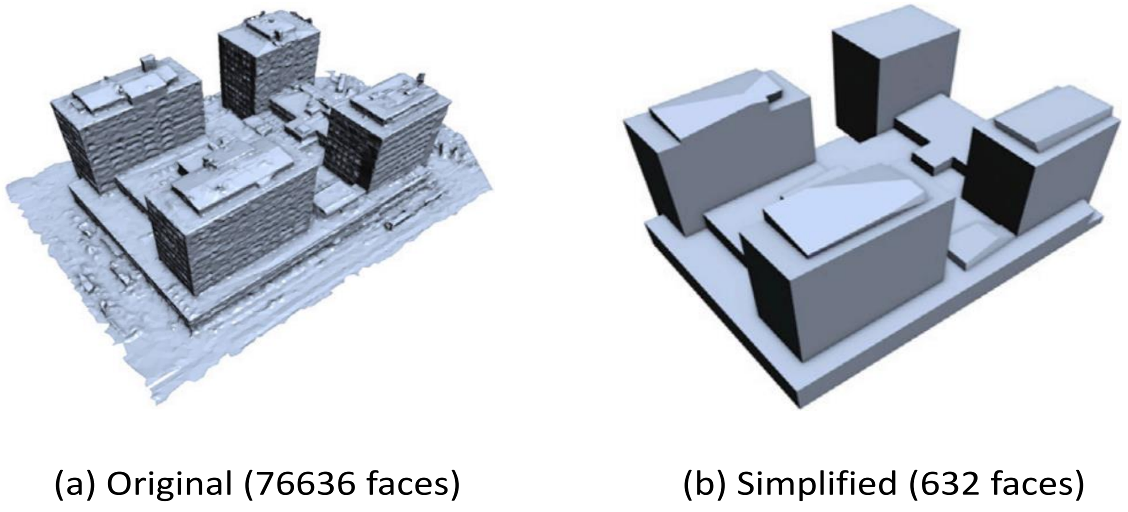


Figure 1.4: *Surface simplification of a building block, Image Courtesy: [(Bouzas* et al.*, 2020)]*

Table 1.1: *Statistics on storage size (in GBs) of different urban-scene based meshes reconstructed from [(Tan* et al.*, 2020)] and [(Hackel* et al.*, 2017)]*

Data	**Reconstruction algo**	**File format**	**Points**	**Faces**	**Storage**
L003	Screened Poisson	PLY	39,721,590	79,443,180	3.16
birdfountain station1	Screened Poisson	PLY	42,197,478	84,394,956	2.86
neugasse station1	Screened Poisson	PLY	49,890,815	99,781,630	3.24
sg27 station10	Screened Poisson	PLY	277,221,015	554,442,030	19.12

Motivated by these challenges associated with the mesh data structures, we investigate into a hybrid approach that offers simplification at dominant planar regions of the meshes but retain essential characteristics of the model for instance, façade elements of a building such as roof, balconies, windows etc. by preserving triangulations around theses areas in the mesh.

1.4 Contributions

In this thesis, we focus on efficiently decimating 3D meshes while preserving the geometry as well as other critical features of a model during the simplification process. We observed that in other simplification methods, major focus has always been given for the preservation of geometry, local smoothness and sharp contours of urban buildings to generate a compact representation. However, these models lack in terms of other fine architectural details such as complex façade element structure and level of detail. Therefore, we consider preserving additional details in the form of critical feature points obtained from a mesh as a set of vertices and preserving them in the simplified models (Figure 1.5). The method is not designed at processing with an entire urban scene at once, but to deal with single buildings in the scene one by one. Our pipeline consists of three major modules, namely mesh reconstruction, feature points detection through gradient structure tensors and mesh decimation (in accordance with points obtained in the previous step). The simplification pipeline can produce economic simplified 3D meshes with decent level of details retained in the simplified model for applications such as large-scale urban scene modeling.

The key contributions of this thesis are:

- **Mesh Decimation through Gradient Tensors:** After extracting critical feature points through gradient tensors, we analyze eigenvalue distribution from gradient tensors and associate it with quadric error metric [(Garland and Heckbert, 1997)] based mesh simplification method with the help of a custom cost penalty function that prevents decimation of extracted feature points in final simplified mesh model.

- **Empirical Study on Data Structures for light-weight modeling:** We analyzed our results and experiments with the help of empirical study in which we compared and evaluated (both qualitatively and quantitatively) our results with other state-of-the-art data

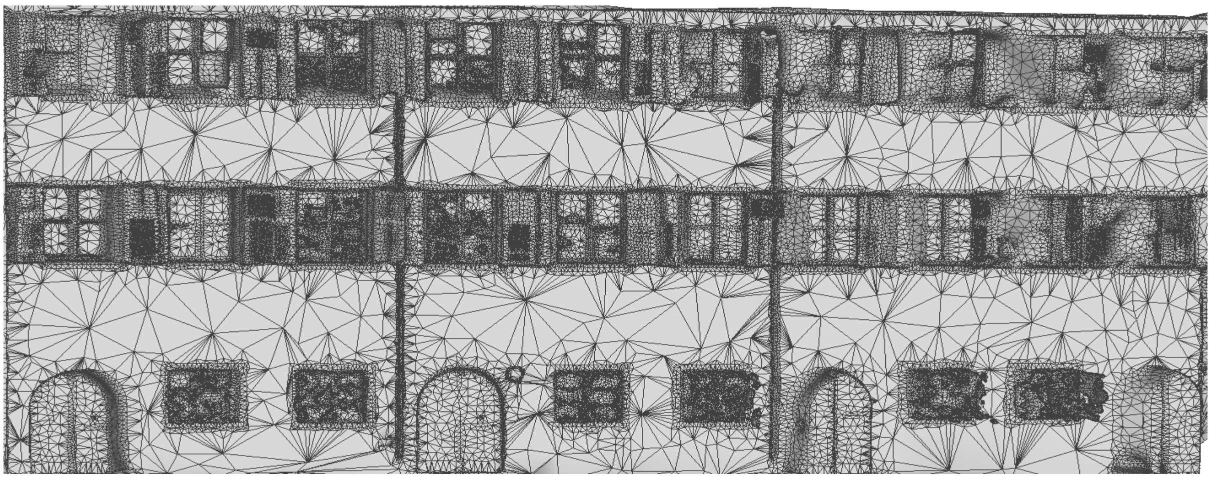


Figure 1.5: *Simplified mesh of a building produced by our decimation method.*

structures/representations of 3D urban scenes such as Polygonal Meshes, Constructive Solid Geometry, Boundary Representation, Stellar Decomposition and Neural Radiance Fields.

1.5 Organization

This thesis is organized into the following chapters:

- **Introduction:** Chapter 1 consists of an overview of light weight modeling and a brief introduction to the problem set. This chapter covers the problem introduction, challenges and an overview of topics needed to understand this thesis.
- **Data Structures for Light-weight Models:** Chapter 2 consists of a literature review and assessment of state of the art data representations of 3D urban scenes comprising polygonal (triangular, quadrilateral and hexagonal) meshes; constructive solid geometry; B-Rep (bezier patches, splines); stellar decomposition; neural radiance fields.
- **Light-weight Modeling Algorithms:** Chapter 3 covers traditional light-weight modeling approaches and their limitations in the context of urban scene. The algorithms include

mesh decimation, geometric abstractions, deep learning based methods and level of detail modeling.

- **Mesh Decimation Through Gradient Tensors:** Chapter 4 provides in-depth explanation of our 3D mesh decimation pipeline with results. We also discuss the experiments including qualitative and quantitative evaluations and comparisons with other methods.
- **Conclusions and Future Directions:** Finally, we conclude this thesis in the last chapter (Chapter 5) and list potential future directions of our method in this chapter.

CHAPTER 2

Data Structures for Light Weight Models

In graphics applications, three dimensional objects are represented using different geometric structures such as triangular meshes or constructive solid geometry in computer-aided design (CAD) systems. Geometric data structures processes the fundamental elements of a 3D object such as surfaces, space and scene structure. This chapter describes the most common data structures used to represent 3D objects including: polygonal (triangular/quadrilateral/hexagonal) meshes, constructive solid geometry, B-Rep, stellar decomposition and neural radiance fields. In the following sections, a brief explanation about each of these 3D models is given.

The representation of 3D models consists of topology and the geometry. Geometry of a 3D object includes surfaces, curves, and points. A 3D mesh is a geometric data structure to represent the surface of a 3D object by a set of polygons. Meshes (see section 2.1) are popular in computer graphics, to render surfaces, or in modeling, to approximate a continuous or implicit surface. A mesh is made up of vertices (or vertex), connected by edges making faces of polygonal shapes. When all faces are triangles, we speak of triangular meshing. Meshes are a great way to exploit the geometry of a point cloud, and often allows to reduce the number of points as vertices.

Both Constructive Solid Geometry (CSG) (see section 2.2) and Boundary Representation (B-rep) (see section 2.3) are solid modeling techniques. CSG allows to create a visually complex surface or object by using Boolean operators to combine primitive solid objects such as cuboids, cylinders, pyramids, spheres etc. Both the surface and the interior of an object can be defined using CSG, although implicitly. In 3D computer graphics and CAD (Computer-aided Design), constructive solid geometry is often used in parametric and/or procedural modeling (a set of techniques in computer graphics to create 3D models and textures from set of rules). In solid modeling and CAD, B-rep is the process of representing shapes using the limits and describes a solid as a collection of connected surface elements by detailing the points, edges and surfaces of a volume. Unlike CSG, B-rep describes only the oriented surface of a solid as a data structure. This oriented convention allows us to decide on which side of the surface the solid's interior is located [(Hoffmann, 1989)].

Stellar decomposition (see section 2.4) [(Fellegara *et al.*, 2021)] is a model for topological data structures that comprises a collection of regions indexing complex's vertices and cells in order to compactly represent arbitary complexes with a manifold or non-manifold domain. It offers efficient navigation of the topological connectivity of simplicial complexes such that it enables users to defer decisions about which topological connectivity relations to encode for the purpose of generating application-dependent local data structures at runtime.

Neural Radiance Field (NeRF) (see section 2.5) [(Mildenhall *et al.*, 2020)] refers to neural network based representation to generate input scene based on a fully connected network. This technique offers optimized results to render geometrically complex scenes with less storage memory in consumption but takes high rendering time.

In this chapter we will analyze different 3D data structures/representations for its suitability for light weight modeling.

2.1 Polygonal Meshes

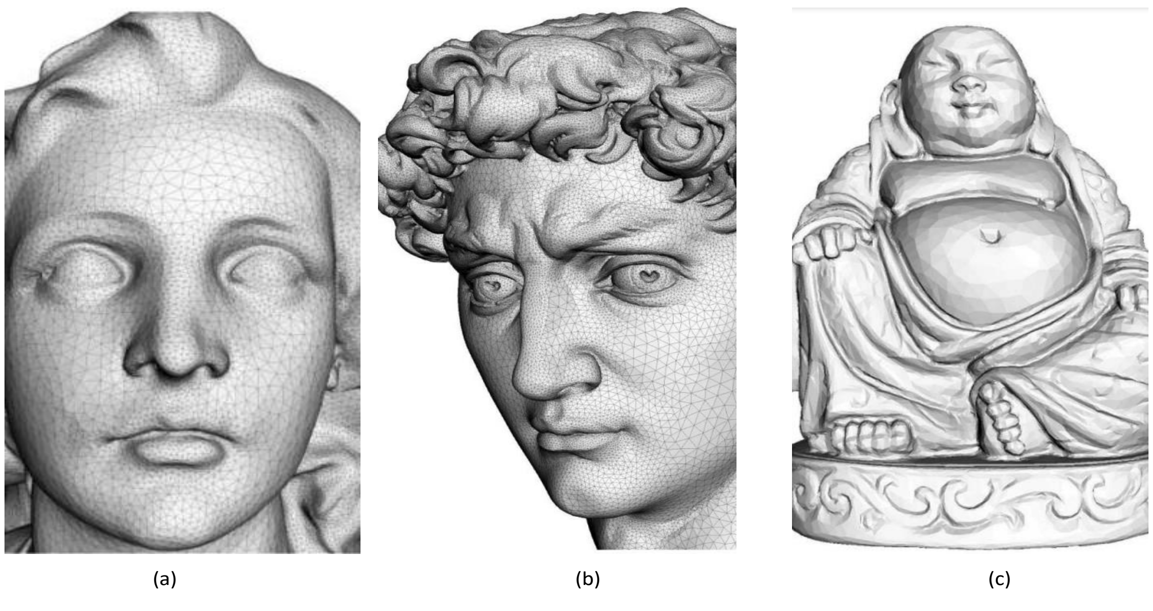


Figure 2.1: *A closeup view of triangular meshes based (a) Lucy model; (b) David model; and (c) Buddha model*

Polygonal meshes are useful representations with a large number of applications in computer graphics, geometric modeling, mechanical engineering, architecture etc. These representations are based on the concept of cell decomposition where a complex object is represented

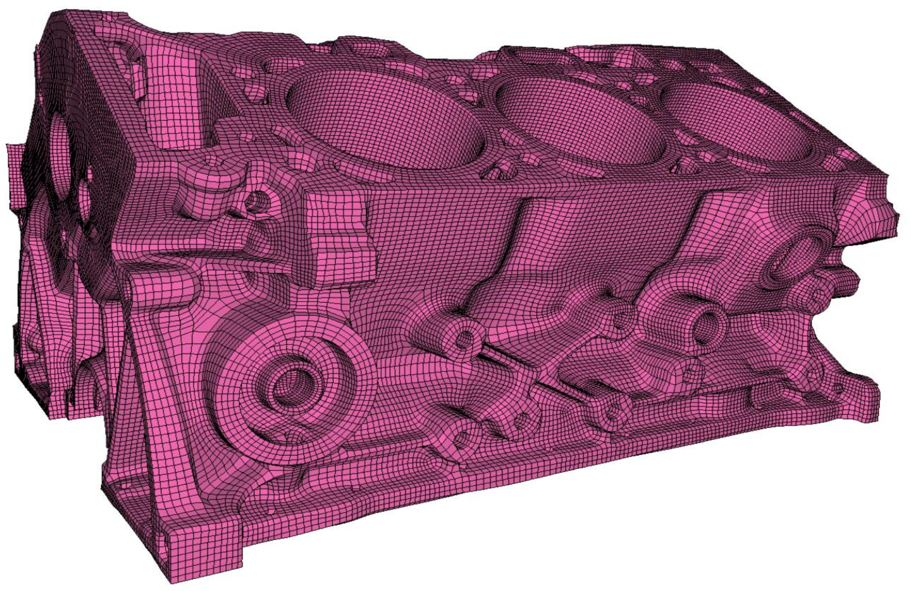


Figure 2.2: *Quadrilateral based mesh model of a mechanical piece, Image Courtesy: www.geometryfactory.com.*

with many simple polygonal cells. Although triangles and quadrilaterals are the most common polygonal types used for surface representation of real-world models, several conceptual architectural structures desire free-form meshes with planar hexagonal faces.

Most real-world models are composed of triangles with shared vertices, commonly known as triangle or triangular meshes. These triangular meshes (Figure 2.1) are used to represent surfaces with the help of a network that connects them through shared vertices and edges to form a single continuous surface. A triangle mesh $\mathcal{M}$ consists of a geometric and a topological component where the topology can be defined by a set of vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ and a set of edges $\mathcal{E} = \{e_1, \ldots, e_n\}, \quad e_i \in \mathcal{V} \times \mathcal{V}$, and triangular faces $\mathcal{F} = \{f_1, \ldots, f_n\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$ connecting them where each edge is defined by a pair of vertices from $\mathcal{V}$ and a face is defined by a triple of vertices from $\mathcal{V}$.

Quad meshes (Figure 2.2) on the other hand are meshes made entirely of quadrilaterals and are extensively in use in CAD and simulation for many years [(Bommes *et al.*, 2013)]. While triangle meshes are much more popular in computer graphics and geometric processing, quad meshes have their own advantages for real-time rendering of objects. For instance, objects designed for video-games, virtual reality and CG animation are represented as polygonal meshes using interactive modeling systems where line features and deformations of the represented shape is designed and processed more conveniently with quad meshes as compared to triangle meshes.

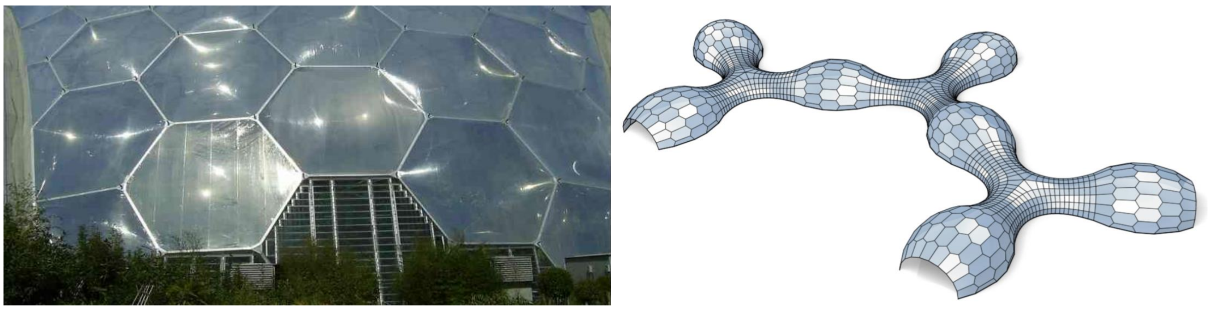


Figure 2.3: *A geodome constructed using a P-Hex mesh in the Eden Peoject in UK (left); P-Hex meshes based constant mean curvature (CMC) surfaces (right), Image Courtesy: [(Wang* et al.*, 2008)]*

Free-form meshes with planar hexagonal faces, also known as *P-Hex meshes* (Figure 2.3), are discrete polyhedral surfaces in 3D that offer important surface representation in discrete differential geometry such as simplicial complexes[1]. They are useful in architectural design for representing surfaces built with planar glass/metal panels [(Wang *et al.*, 2008)] or various special surfaces, such as minimal surfaces[2] or constant mean curvature surfaces[3] [(Bobenko *et al.*, 2006)] in discrete differential geometry. In architectural construction, glass panels represented through planar faces are framed by beams joined at junctions (also known as nodes) where all adjacent faces adjoined with each other. A triangle mesh with vertices of valence 6 meets the requirement of face planarity, but it offers complexity at junction level. Another disadvantage of triangle mesh is that it does not have exact definition of offset meshes [(Pottmann *et al.*, 2007)]. This leads to the requirement of free-form meshes with planar quadrilateral faces (*P-Quad meshes*) and mesh surfaces with planar hexagonal faces (*P-Hex meshes*), as extensions to plane tiling with squares and hexagons.

[1]A simplicial complex is a set composed of points, line segments, triangles, and their n-dimensional counterparts [(Wikipedia contributors, 2022*c*)]

[2]A minimal surface is a surface that locally minimizes its area. This is equivalent to having zero mean curvature [(Wikipedia contributors, 2022*b*)]

[3]Constant-mean-curvature (CMC) surfaces are surfaces with constant mean curvature [(Wikipedia contributors, 2021)]

We refer to [(Kettner, 1999)] for overview and comparison of different mesh data structures. In general, when choosing a data structure one has to take into account both topological as well as algorithmic considerations. An important topological description of a surface is whether it is a manifold or non-manifold. If manifold, whether comprise boundary edges or not.

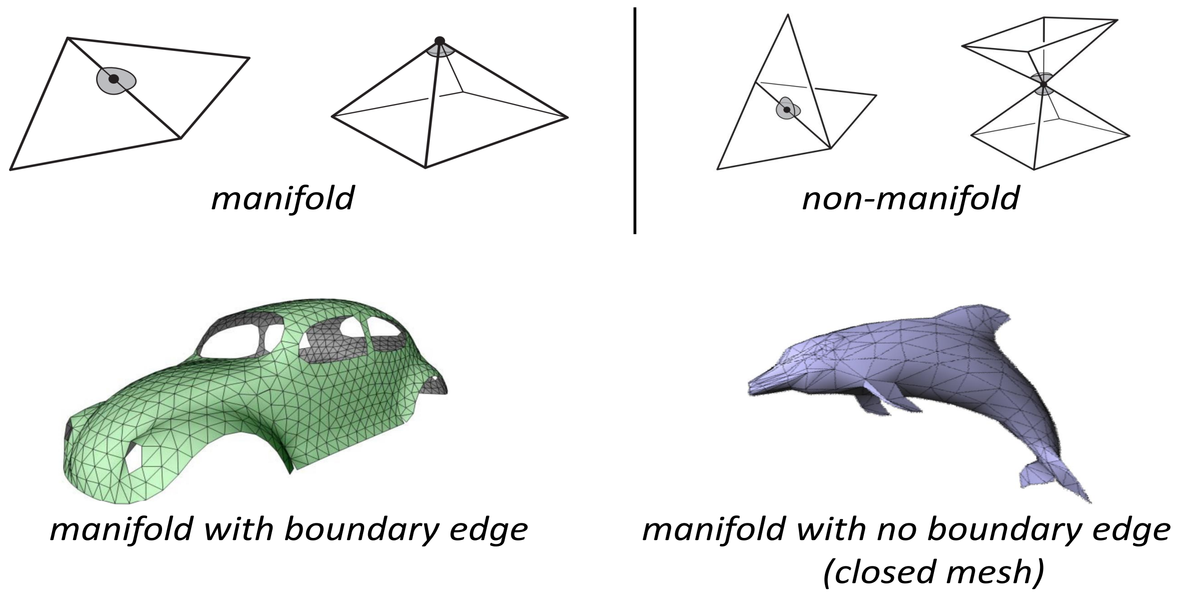


Figure 2.4: *Manifold vs non-manifold meshes, Image Courtesy: [(Steve Marschner, 2014)]*

Topological requirements. What types of meshes need to be represented by the data structure? Either we required boundaries or we can safely consider closed meshes? Either we need to represent complex edges or we can depend on a manifold mesh? Do we need to restrict our needs to pure triangular meshes or we might need to represent arbitrary polygonal meshes?

Algorithmic requirements. What type of algorithms will be operating on the data structures? Do we only need to render the mesh? Either we need to modify only the geometry of the mesh, or do we also have to change the topology? Is there any other additional information that needs to be be associated with vertices, edges and/or faces of the mesh?

The simplest representation for triangle meshes will only store a set of individual triangles. Consequently, it is not sufficient for most requirements: topological information cannot be accessed explicitly, and vertices and related data are replicated. The latter can be established by a shared vertex data structure which stores a table of vertices and encodes triangles as triples of indices in this table. However, without additional connectivity information it is still not efficient for major algorithms.

Some of the mesh operations that are frequently used by most algorithms:

- Access to individual vertices, edges, and faces.
- Oriented traversal of edges of a face that refers to finding out the next edge in a face.
- Access to the faces linked to an edge. Considering the orientation, it is either the left or right face in the manifold case. This also enables access to neighboring faces or in other words, traversal of faces.
- Access to starting and/or end vertex of a given edge.

These operations allow local and global traversal of the mesh and are also possible even for a shared vertex representation; however, the time complexity required will be high for searching vertices, edges and faces.

Several data structures have been developed that allow better traversal of meshes. Well known data structures to represent the meshes are indexed lists, winged edge [(Baumgart, 1972)] and half edge [(Mäntylä, 1987)] data structures.

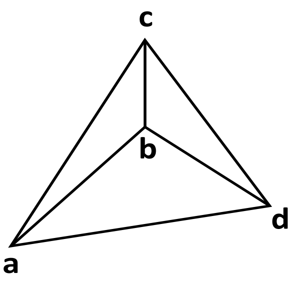
2.1.1 Indexed Mesh Storage

Consideing a simple triangular mesh as shown in Figure 2.5, one way of storing the triangles would be as independent entities in the form:

```
Triangle ()
vector3 vertexPosition [3]
```

The result is that the vertex *b* (Figure 2.5) is stored three times and the other vertices are stored twice. Therefore, nine vertices are stored while many of them are repetitive. The other way to represent the triangular meshes would be to arrange the common vertices and store only four vertices as a *shared-vertex mesh* [(Marschner and Shirley, 2018)]. The data structure has triangles which refer to vertices containing the vertex data:

```
Triangle ()
Vertex v[3]
Vertex
float position  //or other vertex data
```



separate triangles:

#	*vertex 0*	*vertex 1*	*vertex 2*
0	(a_x, a_y, a_z)	(b_x, b_y, b_z)	(c_x, c_y, c_z)
1	(a_x, a_y, a_z)	(b_x, b_y, b_z)	(c_x, c_y, c_z)
2	(a_x, a_y, a_z)	(b_x, b_y, b_z)	(c_x, c_y, c_z)

shared vertices:

triangles		vertices	
#	*vertices*	*#*	*position*
0	(0, 1, 2)	0	(a_x, a_y, a_z)
1	(1, 3, 2)	1	(b_x, b_y, b_z)
2	(0, 3, 1)	2	(c_x, c_y, c_z)
		3	(d_x, d_y, d_z)

Figure 2.5: *A three triangle mesh with four vertices (a,b,c,d), represented with: separate triangles (left) and shared vertices (right).*

During implementation, the vertices and triangles are stored in arrays with the triangle-to-vertex references processed by storing array indices:

```
IndexedMesh ()
int tInd [nt] [3]   //vertex indices
float verts [nv] [3]
```

The index of the *k*th vertex of the *i*th triangle is found in *tInd[i][k]*, and the position of that vertex is stored in the associated row of the verts array. This way of storing a shared vertex is called as indexed triangle mesh.

2.1.2 Winged-Edge Structure

One of the popular mesh data structures that store connectivity information at the edges instead of the faces is the *winged-edge* structure. Due to its edge-centric nature rather than face-centric, it also works for polygonal meshes. In a winged-edge mesh (Figure 2.6), each edge points to the two vertices it connects, i.e. head and tail vertices, the two faces it is part of (left and right faces) and the next and previous edges of its left and right faces in a counterclock-wise traversal manner. Each vertex and face also point to an edge that connects to it.

```
Edge {
        Edge lprev, lnext, rprev, rnext;
        Vertex head, tail;
        Face left, right;
}

Face {
       //    . . . per-face data . . .
      Edge e;       // any adjacent edge
}

Vertex {
       //   . . . per-vertex data . . .
       Edge e;     // any incident edge
}
```

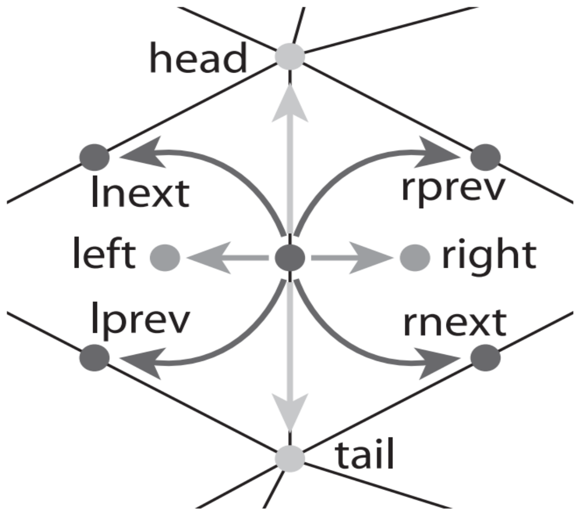


Figure 2.6: *References from an edge to the neighboring faces, edges and vertices in the winged-edge structure. Image Courtesy: [(Marschner and Shirley, 2018)]*

```
HEdge   {
      HEdge pair, next;
      Vertex v;
      Face f;
}

Face    {
    // . . . per-face data . . .
    HEdge h;    // any adjacent h-edge
}

Vertex    {
    // . . . per-vertex data . . .
    HEdge h;    // any incident h-edge
}
```

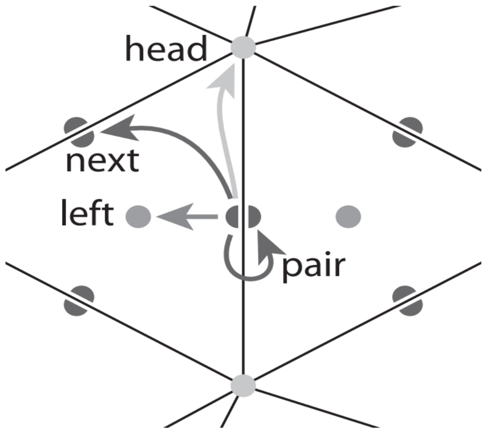


Figure 2.7: *References from half-edge to the neighboring mesh components. Image Courtesy: [(Marschner and Shirley, 2018)]*

2.1.3 Half-Edge Structure

One of the most convenient and flexible edge-based data structures in geometry processing is the half-edge data structure. The winged-edge structure also offers polygon mesh processing due to its edge-centric nature. However, it constantly requires to be checked if the edge is oriented before moving to the next edge: for instance, if the current edge is from the head or the tail. In a half-edge structure (Figure 2.7), we store data for each half-edge rather than storing data for each edge. Each edge is split into two opposing half-edges such that all half-edges are consistently oriented in counter-clockwise order around each face and along the boundary.

For each half-edge, a reference gets stored for: the vertex it points to; its adjacent faces (a zero pointer, in case of boundary half-edge); next half-edge of the face or boundary; its opposite half-edge; and the previous half-edge in the face. Further, references for each face to one of its adjacent half-edges and for each vertex to one of its outgoing half-edges get stored as well.

The space requirements for these structures are as follows:

Table 2.1: *Space requirements for mesh structures.*

Mesh Structure	**Vertex Position Array**	**Indices Array**	**Repr. Edge**	**Total Storage**
Indexed Mesh Storage	12 bytes/vert	24 bytes/vert	-	36 bytes/vert
Winged-Edge Structure	12 bytes/vert	96 bytes/vert	4 bytes/vert	112 bytes/vert
Half-Edge Structure	12 bytes/vert	96 bytes/vert	4 bytes/vert	112 bytes/vert

2.2 Constructive Solid Geometry

Constructive solid geometry (CSG) is a modeling technique used in CAD (computer-aided design) systems to create complex geometry based on combining primitives by parametric features and volumetric boolean set operations (such as union, intersection, and difference). The CAD system usually translates a CSG model into a B-rep (see section 2.3).

The standard CSG primitives comprise cube, cuboid, sphere, cylinder, cone, triangular prism and torus. These primitives (in normal or generic form) are instantiated by the user to be used in designing along with transformations such as scaling, rotation and translation to position manually. Then, two instantiated (and perhaps transformed) primitives can be combined into one with boolean operators such as union, intersection and difference. The CSG representation is an ordered binary tree where terminal nodes represent primitives and each internal node defines a boolean set operation applied to the left and right nodes, or children, it points to. This tree data structure is commonly referred to as CSG tree. Figure 2.8 illustrates a simple example, given two sets, *A* and *B* representing primitives, their union (a) consists of all points from either *A* or *B*; their intersection (b) consists of all points in both sets; and their difference written as *A* - *B* (c) consists of all points in *A* but not in *B* and vice versa (d).

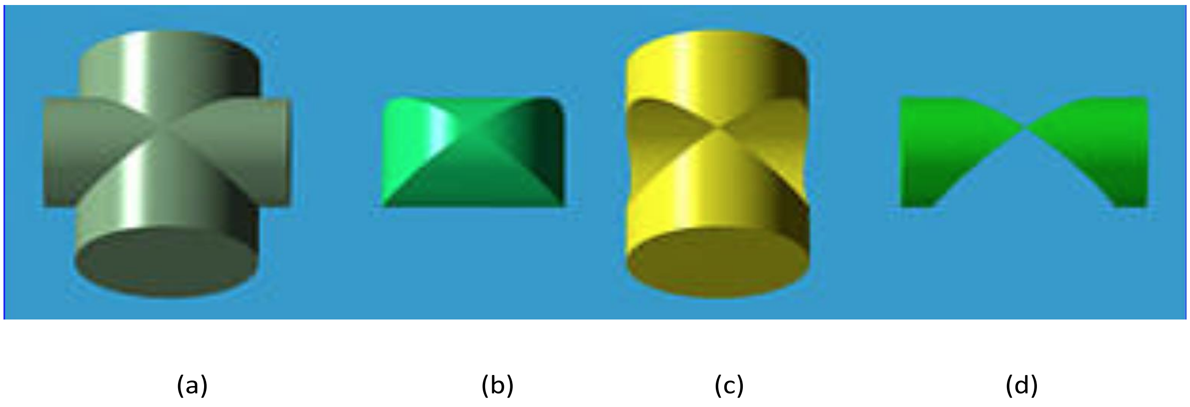


Figure 2.8: *An example of CSG boolean operations. Image Courtesy: [(Dr. C.-K. Shene, 2011)]*

Transformations such as scaling, rotation and translation can be applied at any node of the CSG tree. Applying a transformation or deformation at the head node would result in applying it to each individual primitive of the tree.

CSG Expressions. Figure 2.9 shows a simple example, to design a complex object by using boolean operations to combine simpler objects. We start with two instantiations of cylinder and one instantiation each for a cube and sphere. In one cycle, the two instantiated cylinders undergo union operators and in the other cycle, cube and sphere undergo intersection operator to create sampled solid objects. The final model will be obtained by computing the difference between the solid objects obtained in last step.

The design procedure of the above solid object can be written as an expression:

diff(union(Cylinder, union(Cylinder, Cylinder)), intersection(Cube, Sphere))

where *union(A,B)*, *intersection(A,B)* and *diff(A,B)* are the union, intersection and difference of *A* and *B*. The expression can be converted to an expression tree, the CSG expression of the design. Geometric primitives are special case of constructive solid geometry trees and are defined as simpler 3D geometric shapes with characteristics such as:

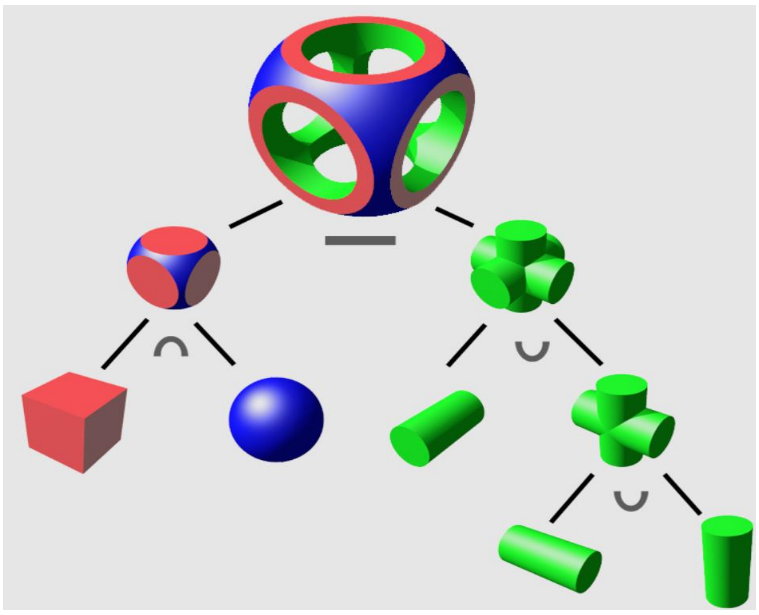


Figure 2.9: *An example of CSG operations with expression. Image Courtesy: [(Commons, 2020)]*

- convex structure (except for the torus);
- fixed and limited global intrinsic parameters that only define the global size, orientation and position of the shape;
- symmetry;
- basic shape that can be combined with other simpler shapes to form more complex shapes.

In constructive solid geometry, simple primitives are used to build complex shapes by applying Boolean exoressions. These primitives (Figure 2.10) can be classified into 4 categories [(Kaiser *et al.*, 2019)] which are as follows:

- plane;
- box and cuboid;
- sphere, cylinder and cone;
- ellipsoids, torus, non-rectangular parallelepipeds.

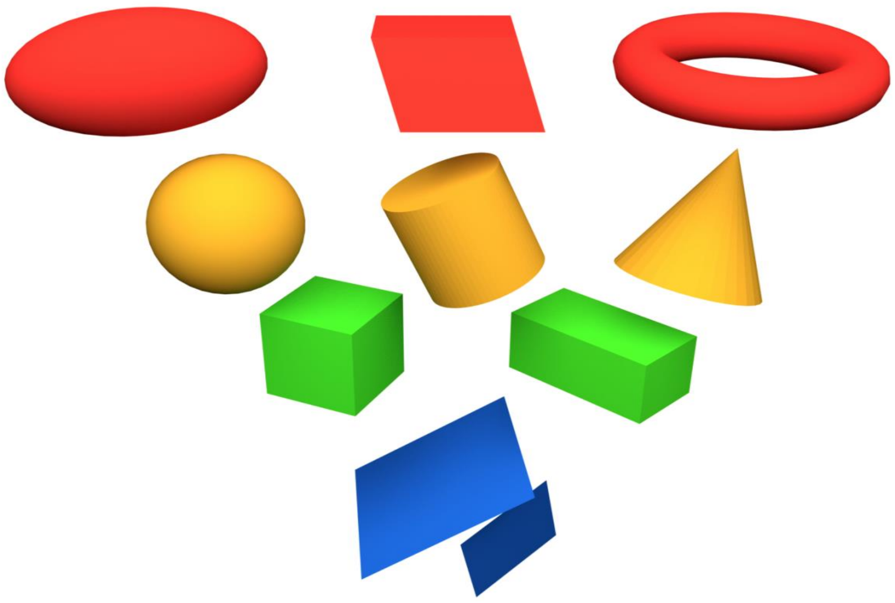


Figure 2.10: *Common geometric primitives, Image Courtesy: [(Kaiser* et al.*, 2019)]*

Plane. A *plane* is a basic isotropic shape which can be defined by a normal vector *n* and its distance $d \in \mathcal{IR}$ to the origin point of reference frame.

Box and Cuboid. A *box* and *cuboid* is defined by center *C*, orientation vector *n* with assembled orthogonal planes and three axis based dimensions. They are represented by their eight vectors or parameters of the planes forming them.

Sphere, Cylinder and Cone. A *sphere* is an isotropic shape with center *C* and a scalar $r \in \mathcal{IR}_+$ representing its radius.

A *cylinder* is parameterized with a point *C* of its axis, a radius $r \in \mathcal{IR}_+$ and vector *n* representing its orientation.

A *cone* can be parameterized by its center *C*, with an orientation vector *n* and an angle θ between its axis and surface.

Ellipsoid, Torus, Parallelepipeds. These are geometric shapes of higher complexity.

An *ellipsoid* is defined by its center *C*, axis *n* and three radii $x, y, z \in \mathcal{IR}_+$, also known as semi axes.

A *torus* is defined by its center *C*, axis *n*, minor and major axes *x* and *y* respectively.

A *parallelepiped* is defined by its center *C*, orientation vector *n*, three axis based dimensions and their interaxial angles.

Reconstruction of building roofs and facades from airborne laser scanning (ALS) point clouds has been an active research among computer graphics and remote sensing domains with usage of geometric primitives to represent models [(Yan *et al.*, 2014), (Verma *et al.*, 2006)]. The geometric elements of these primitives for instance, edges and vertices are used to reconstruct compact simpler building models with less complexity. However, for complex building rooftops these primitives based representations do not fit well. CSG modeling allows to divide complex modeling tasks into different subtasks and thus comes in handy. After designing relevant sub-building models, they can be adjoined to form a complex building model by applying Boolean operations using CSG technique [(Wang *et al.*, 2018)].

2.3 Boundary Representation (B-rep)

Boundary representation (often abbreviated as B-REP in CAD solid modeling) is a method for representing shapes using the limits. Here, a solid is represented as a collection of connected surface elements and defines the boundary between interior and exterior points [(Wikipedia contributors, 2022*a*)]. A boundary representation of a model describes only the geometry (such as surfaces, curves and points) of a solid as a data structure based on topology (vertices, edges and faces) [(Hoffmann, 1989)]. The topological and geometrical entities are intertwined in a way where:

- the face is a bounded portion of a character;
- an edge is an enclosed piece of curve; and
- a vertex lies at a point.

This way, topological entities allow making links between geometrical entities.

B-Rep has its own advantages and disadvantages, which are listed as bellow:

- This method is suitable for constructing solid models of unusual shapes.
- A B-rep model is relatively easy to convert to the wireframe model.

- B-rep uses only primitive objects and boolean operations to combine them, unlike constructive solid geometry (CSG).

- In addition to the boolean operations, B-Rep has additional actions such as extrusion (or sweeping), chamfer, blending, drafting, shelling, tweaking etc.

- B-rep is not suitable for applications like tool path generation.

- It requires large storage, especially if curved object are approximated with polyhedral models.

Surface Patch A surface patch is a fundamental building block for surface with curved bounded collection of points whose coordinates are given by continuous, two-parametric, single valued mathematical function of the form:

$$\bar{p}(u,w) = [x(u,w) \quad y(u,w) \quad z(u,w)]^T \tag{2.1}$$

The patch may be termed *biparametric*, as the two variables *u* and *w* vary across the patch. The parametric variables lie in the range of 0 to 1. Fixing the value of one of the parametric variables results in a curve on the patch in terms of the other variable (*Isoperimetric curve*).

Types of Surfaces:

i. Bezier Curve and Surface A Bezier curve [(scratchapixel.com)] is a parametric curve used in computer graphics to model smooth curves that can be scaled indefinitely and can be defined by a set of control points $\mathbf{P}_0$ through $\mathbf{P}_n$ where n is called the order of the curve. For instance, Figure 2.11 represents a simple bezier curve with 4 control points.

To create this curve, we need to compute it by combining these 4 points weighted by some coefficients.

$$P_{\text{curve}}(t) = P1 * k_1 + P2 * k_2 + P3 * k_3 + P4 * k_4 \tag{2.2}$$

where *P1*, *P2*, *P3*, *P4* are the bezier control points and k_1, k_2, k_3, k_4 are coefficients weighting the contribution of control points.

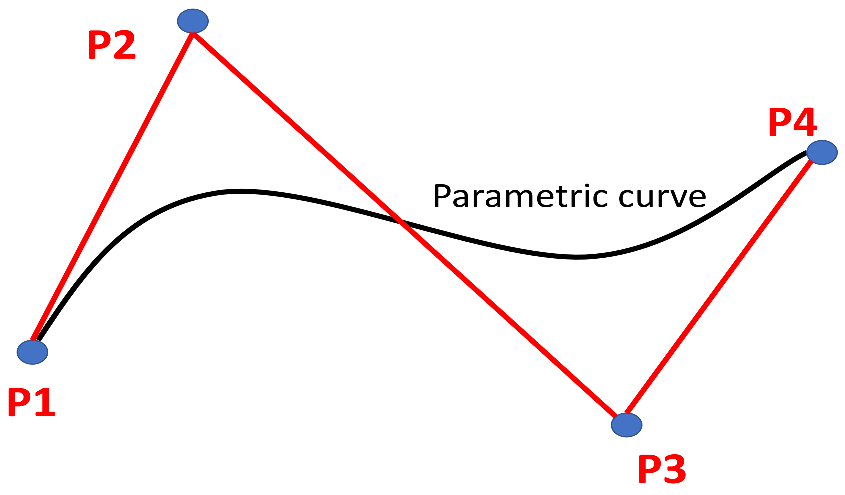


Figure 2.11: *A bezier curve and its 4 control points. Image Courtesy: www.scratchapixel.com*

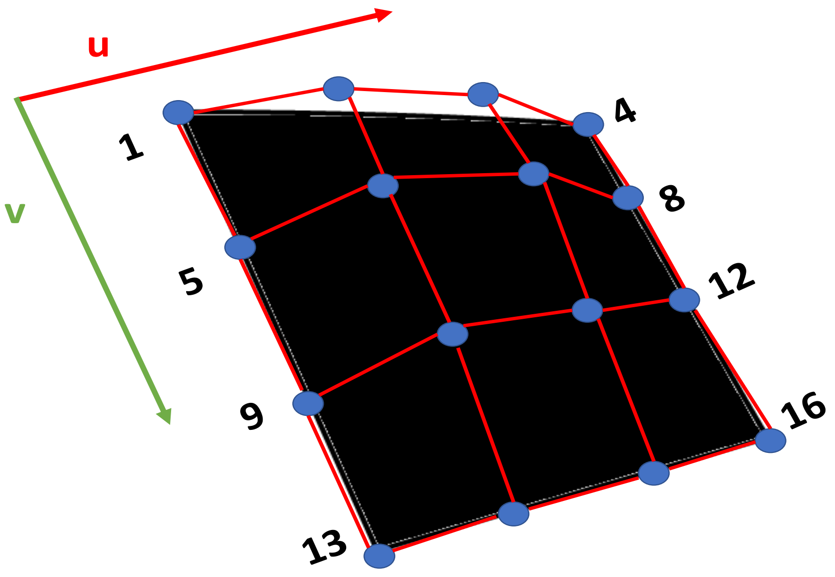


Figure 2.12: *A bezier patch.*

In a case of bezier surface, which is a family of bezier curves, we take for instance, 16 points (a grid of 4x4 control points) rather than 4 points to define the surface with parameters: u and v (Figure 2.12) where u and v belong to the range [0,1] that helps in remapping of the unit square into a smooth continuous surface. The equation 2.1 can now be defined as a double sum of control points and coefficients for bezier surface (which is a sum of bezier curves) as:

$$P(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) P_{ij} \tag{2.3}$$

ii. B-spline Curve and Surface Both bezier and b-spline curves are parametric; however, the bezier representation has two main disadvantages: first, the number of control points is directly related to degree. Thus, to increase the complexity of the shape of the curve by increasing control points requires increasing the degree of the curve or fulfilling the continuity conditions between consecutive segments of a composite curve. Second, changing any control point affects the entire curve or surface which makes the design of specific sections complicated. The concept of B-spline curve came to resolve the disadvantages of bezier curves where the control points impart local control over the curve-shape rather than the global control in the case of bezier curve. Due to this, only a specific segment of the curve-shape gets changes by the changing of the location of the control points.

A b-spline curve is defined as a linear combination of control points p_i and B-spline basis function $N_{i,k}(t)$ given by

$$\mathbf{r}(t) = \sum_{i=0}^{n} \mathbf{p}_i N_{i,k}(t), \quad n \geq k-1, \quad t \in [t_{k-1}, t_{n+1}] \tag{2.4}$$

where, p_i *: i = 0, 1, 2,..., n* are the control points, *k* is the order of the polynomial segments of the b-spline curve such that curve is made up of piecewise polynomial segments of degree *k - 1*, and the $N_{i,k}(t)$ are the normalized b-spline blending functions of order *k* and defined on a knot vector *T* = $(t_0, t_1, ..., t_{k-1}, t_k, t_{k+1}, ..., t_{n-1}, t_n, t_{n+1}, ..., t_{n+k})$ where there are *n + k + 1* elements.

The surface analogue of the b-spline curve is b-spline surface (patch) which is a tensor product surface defined by a rectangular set of control points $p_{i,j}, 0 \leq i \leq m, 0 \leq j \leq n$ and two knot vectors $U = (u_0, u_1, ..., u_{m+k})$ *and* $V = (v_0, v_1, ..., v_{n+l})$ associated with each parameter *u, v*. The corresponding integral B-spline surface is given by:

$$\mathbf{r}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{ij} N_{i,k}(u) N_{j,l}(v) \tag{2.5}$$

2.4 Stellar Decomposition

There has been numerous works in the area of topological mesh data structures, optimized data layouts and distributed mesh data structures. Mesh data structures are usually efficient to model simple problems on small low dimensional meshes for a broad range of mesh processing applications in computer graphics. However, in case of much larger meshes and in higher dimensions, these mesh structures does not perform well. Thus, flexibility is required to deal with complex meshes including irregularly connected cell types with the help of exploiting locality within the mesh.

The stellar decomposition [(Fellegara *et al.*, 2021)] is a modern day data structure that supports efficient navigation of the topologial connectivity of simplicial and of certain classes of cell complexes, for instance, complexes based on quadrilaterals, polygons, pyramids etc. also known as *Canonical Polytope (CP) complexes* (see Figure 2.13) with a manifold or non-manifold domain. A stellar decomposition of a complex is a collection of regions indexing the complex's vertices such that each vertex within a region has sufficient information to locally reconstruct the star of its vertices, i.e. the cells incident in the region's vertices. It is both scalable and flexible to support the generation of optimal local data-structures at runtime. A *Stellar tree* is an instance of the stellar decomposition model for spatially embeded complexes where the decomposition of vertices in the stellar tree is based on a hierarchical *n*-dimensional quadtree (or kD-tree) with a simple tuning parameter to assist balance in storage and performance needs. Stellar trees are designed to provide efficient processing on the regions of the mesh where users can generate optimized local topological data structures by defering which topological relations to encode.

A Stellar decomposition is defined as

$$S_D = (\Sigma, \Delta, \Phi),$$

where Σ is a CP complex; *decomposition* Δ constitute regions that cover the vertices Σ_V of complex Σ such that every vertex $v \in \Sigma_V$ belongs to one of the regions *r*; and *map* Φ from regions of Δ to entities of Σ.

Under mapping Φ, each region *r* in decomposition Δ maps to an array of vertices (r_V) and

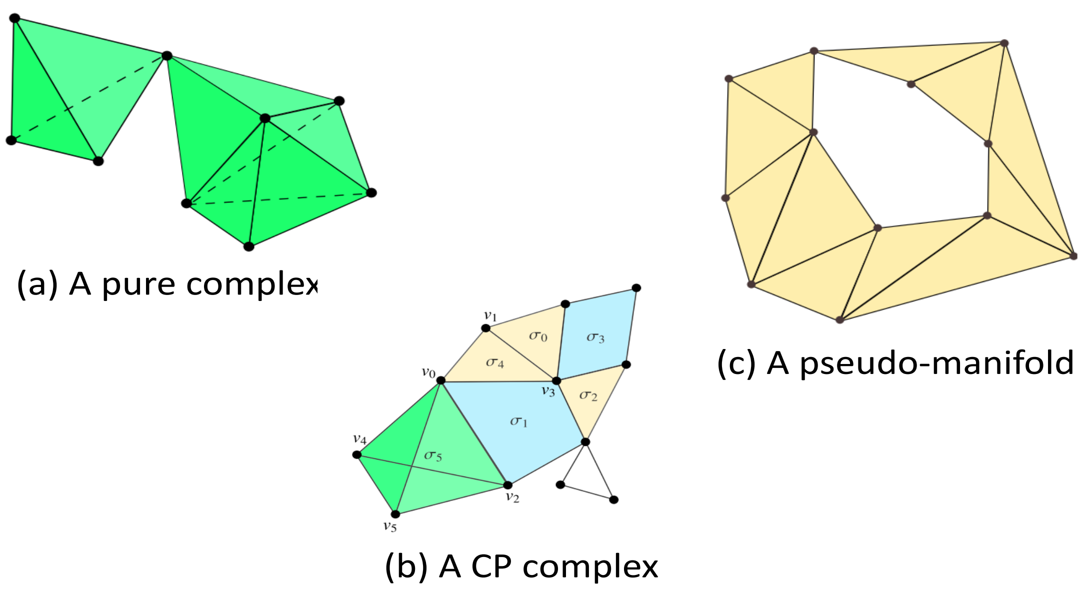
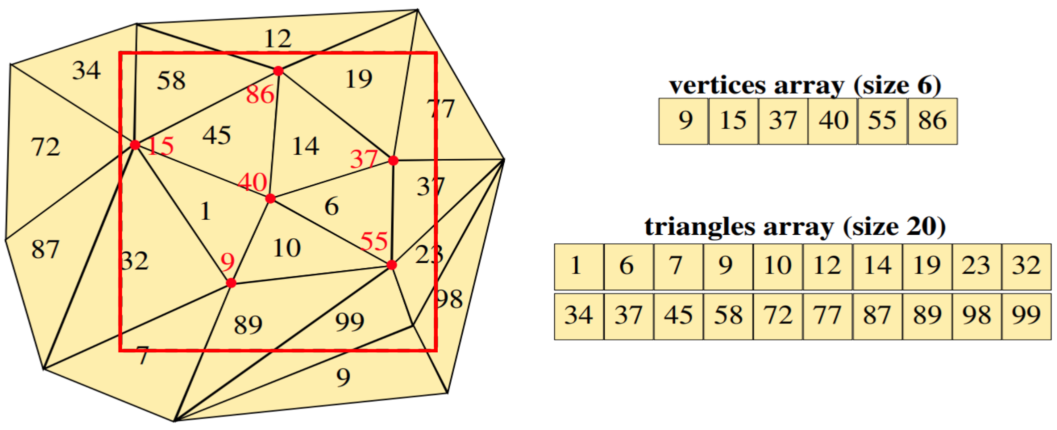


Figure 2.13: *A few examples of CP complexes: (a) A pure simplicial 3-complex with 4 tetrahedra; (b) A CP complex with three top edges, three top triangles, two top quads and a top tetrahedron; (c) A 2-dimensional pseudo-manifold with 11 triangles, Image Courtesy: [(Fellegara* et al.*, 2021)]*



vertices array (size 6)

9	15	37	40	55	86

triangles array (size 20)

1	6	7	9	10	12	14	19	23	32
34	37	45	58	72	77	87	89	98	99

Figure 2.14: *Stellar decomposition encoding for triangles within a region r (red square), Image Courtesy: [(Fellegara* et al.*, 2021)]*

top cells (r_T) from the complex Σ. Figure 2.14 refers to the Stellar decomposition encoding of arrays of vertices and top CP cells that explicitly list the associated elements, vertices and triangles for each region *r*.

2.5 Neural Radiance Fields

Neural radiance field [(Mildenhall *et al.*, 2020)] (Figure 2.15) is a neural based technique to generate a single input scene. It is a fully connected network which maps 5D input of a scene i.e. spatial location (x,y,z) and viewing direction (θ, ϕ) to 4D output which is view dependent RGB color and opacity in terms of volume density. Finally classical volume rendering technique is used to to differentiably render new 2D image views. As the process is differentiable, gradient descent can be used for model optimization. This error minimization through gradient descent results in prediction of a coherent model of the scene using assignment of high volume densities and accurate colors to the true scene-content locations.

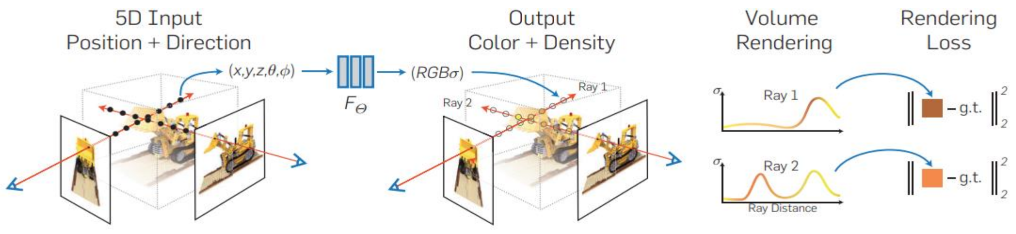


Figure 2.15: *An overview of neural radiance field scene representation, Image Courtesy: [(Mildenhall* et al.*, 2020)].*

Although this method uses volumetric representations to represent complex geometry and appearance, it overcomes the high strorage cost constraint of discretized voxel grids through hierarchical volume sampling. This technique provides effectively optimized results to render realistic views of the geometrically complex scenes. Furthermore, it requires lesser storage memory in comparison to the input image. However rendering time is high, as optimization of a single scene converges typically in 100 to 300k iterations on a single NVIDIA V100 GPU which takes around 1 to 2 days and per scene training takes atleast 12 hours.

2.6 Assessment of Different Representations

In this section, we critically analyze the methods discussed in Sections 2.1 - 2.5 based on parameters such as accurate representation of the model (comprising model being realistic and conforms to the geometry) and computational cost (gauged in terms of computer storage space) for different representation of 3D urban scenes. Based on that, we will identify the data structures/representations as prime candidates for light-weight urban modeling.

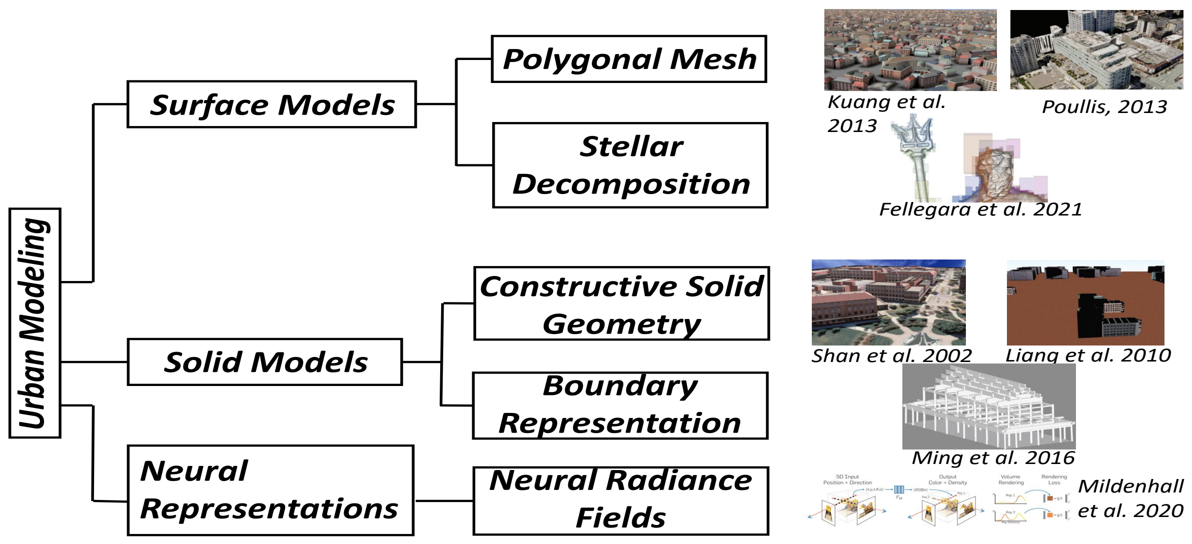


Figure 2.16: *Classification of common modeling techniques and a few representation results.*

A mesh model constructed from point clouds by a surface reconstruction algorithm [(Kazhdan *et al.*, 2006), (Guennebaud and Gross, 2007)] offers decent representation. Although LiDAR scans are complicatedly sparse with outliers, noise and occlusions that result in less accurate representations, dense meshes generated from airborne images have shown greater geometric accuracy and completeness [(Rouhani *et al.*, 2017)]. Some recent advances in shape scanning and modeling have allowed depth geometric information encoded into mesh with 3D textures resulting model being realistic and complying with the accurate geometry of the model. However, these meshes typically contain hundreds of millions of faces which create burden on the visualization, storage and rendering of real-world applications [(Li and Nan, 2021)]. For instance, an European-style city with 40,400 distinct buildings [(Kuang *et al.*, 2013)] comprises 1.36 billion triangles with a storage consumption of 61GB that makes using conventional meshes less suitable for web-based interactive applications. Thus, a lot of significance has been given in the area of mesh simplification [(Garland and Heckbert, 1997), (Li and

Nan, 2021)] and polygonization [(Bouzas *et al.*, 2020)] that makes meshes a prominent candidate for light-weight urban modeling. However, there are significant challenges with mesh simplification technique that we will address in Chapter 4.

Stellar tree based topological data structures are modern approaches for efficient traversal of the regions of arbitary complexes with manifold or non-manifold domain. These structures have been used in mesh processing based applications such as mesh simplification [(Fellegara *et al.*, 2020)] to geometrical and topological applications including local curvature estimation and mesh validation. Simplicial complexes that are not limited to triangle or tetrahedral meshes are complexes that are defined as collections of p-dimensional hyper-tetrahedra, also called as *p-simplices*. We refer to the experimental analysis and comparison from [(Fellegara *et al.*, 2021)], regarding storage comparison among Stellar-tree encodings such as EXPLICIT and COMPRESSED (most compact encoding). The tuning parameter, *bucketing threshold* k_v uniquely determines the decomposition for a given complex Σ. A block r in a PR-tree is considered full when it indexes more than k_v vertices. Smaller values of k_v yield deeper hierarchies with corresponding blocks index few vertices and top cells, while larger values of k_v yield shallower hierarchies with corresponding blocks indexing more vertices and top cells. By adjusting the tuning parameter k_v, the EXPLICIT and VERTEX-COMPRESSED trees yield reduction in memory requirements with upto *20-50 percentage*, for instance, NEPTUNE triangular dataset with storage requirement reduced from 32.0 MB to 26.2 MB for EXPLICIT trees while COMPRESSED trees reduced from 5.76 MB to 1.24 MB. While, Stellar-tree based data structures and simplicial complexes are widely used to discretize 3D shapes, there is not enough study on them regarding urban scene based representations. From our understanding, processing an urban model through stellar trees might not be the best choice as storage reduction offered in different encodings are not enough. However, the efficient nature of stellar trees in processing specific regions of mesh by encoding topological relations might come in handy for light-weight urban modeling. Due to different level of detail required in light-weight modeling, such as geometric primitives based simple compact representation while preservation of geometric structure, sharp features and facade elements, we believe stellar trees can be used as local correspondents to define topological relations around feature areas of a mesh. Thus, we consider stellar trees as a useful representation for light-weight urban modeling.

CAD system based solid modeling techniques such as CSG and B-Rep have vast background in modeling complex surfaces/objects as well as buildings [(Shan, 2002), (Ming *et al.*, 2016)]. [(Shan, 2002)] is based on generation of realistic urban elements such as various building models with three modeling approaches such as flat roof, triangulated irregular network and constructive solid geometry. To make the building models realistic, textures of building roofs and walls acquired through aerial and ground images are mapped to the sample model. This approach allows modeling complicated buildings and provides sufficient details, however, at a cost of a time-consuming modeling process as the mapping of textures to the models can be too slow. [(Ming *et al.*, 2016)] is based on a CSG-BRep (Constructive Solid Geometry-Boundary Representation) topological model where each topological element belongs to a shape (CSG-Shape) object and boolean operations can be performed on CSG-BRep model. Arbitrary topology object comprises three variables including topological location (CSG-Location), orientation (CSG-Orientation), and a subset of topological shapes (CSG-Subshape). At last, CSG-BRep based construction using LiDAR point cloud as data source is explained where improved RANSAC algorithm is used to fit parameters for simple point cloud while for complex point cloud, manual segmentation following by parameters fitting is performed to determine CSG voxel size. Geometry and topology data are fused together and then voxel model is created from simple element to complex element such as vertex to cylinder, cone, sphere or closed free-form surfaces. The following CSG-BRep model constructed and explained in the paper offers memory footprint 336 KB in comparison to an equivalent triangular mesh model (3991 points and 7475 faces) with memory footprint 1986 KB. This model further contains detailed topological relations and enables query and analysis of 3D spatial topological information. However, to represent complex architectural details from real-world examples such as, complex facade level details in european-style buildings this approach might not be suitable. Considering, generation of a structure through primitives as a time-consuming process, we believe a CAD system based solid modeling approach might not be the best choice.

To better assess the representation of a CAD model in comparison to a mesh model, we take 3D CAD models and their equivalent triangular mesh models (see Figure 2.17). These CAD models have memory footprints 14.4 MB and 2.94 MB in comparison to their equiavalent triangular mesh models 14.8 MB (116166 vertices and 162058 faces) and 3.46 MB (24287 vertices and 35423 faces). We observe that CAD based representations are useful when model exhibits dominant planar regions. However, in case of complex architectural details, a triangular mesh model would be more suitable to better represent the level of detail.

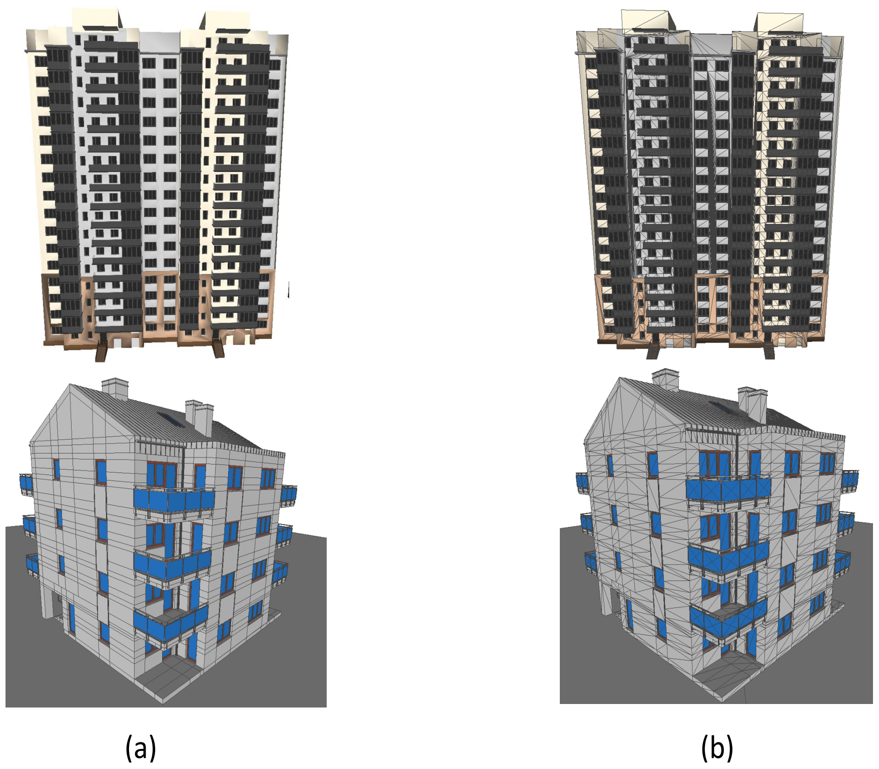


Figure 2.17: *(a) CAD models of sample buildings (downloaded from open-source 3D modeling platform Sketchfab - https://sketchfab.com/features/free-3d-models); (b) Equivalent triangular mesh models.*

To understand in more detail, Figure 2.18 represents buildings and their simplified version as triangular meshes and boundary representation. We observe that structural geometry can be preserved in boundary representation. However, this representation is not satisfactory to represent complex details such as facade elements in comparison to triangular meshes. Table 2.2 lists the statistics such as number of vertices, number of faces and total storage (in MB) required for both original as well as simplified 3D models. From the given data, it can be concluded that B-Rep offers light-weight modeling, however, such simpler representation is not enough for complex detail. On the other hand, polygonal meshes consume more storage yet allows representation of precise information due to more number of faces.

From our assessment for different representations, we can conclude that polygonal meshes and stellar trees are prominent candidates for light-weight modeling algorithms. As, our goal is to model highly detailed yet light-weight 3D buildings, we will focus more in the area of meshes based representation and available light-weight modeling techniques (see Chapter 3).

Table 2.3 lists the feature points of data representations for comparison. Polygonal mesh and stellar decomposition can be classified into surface modeling where geometric element is

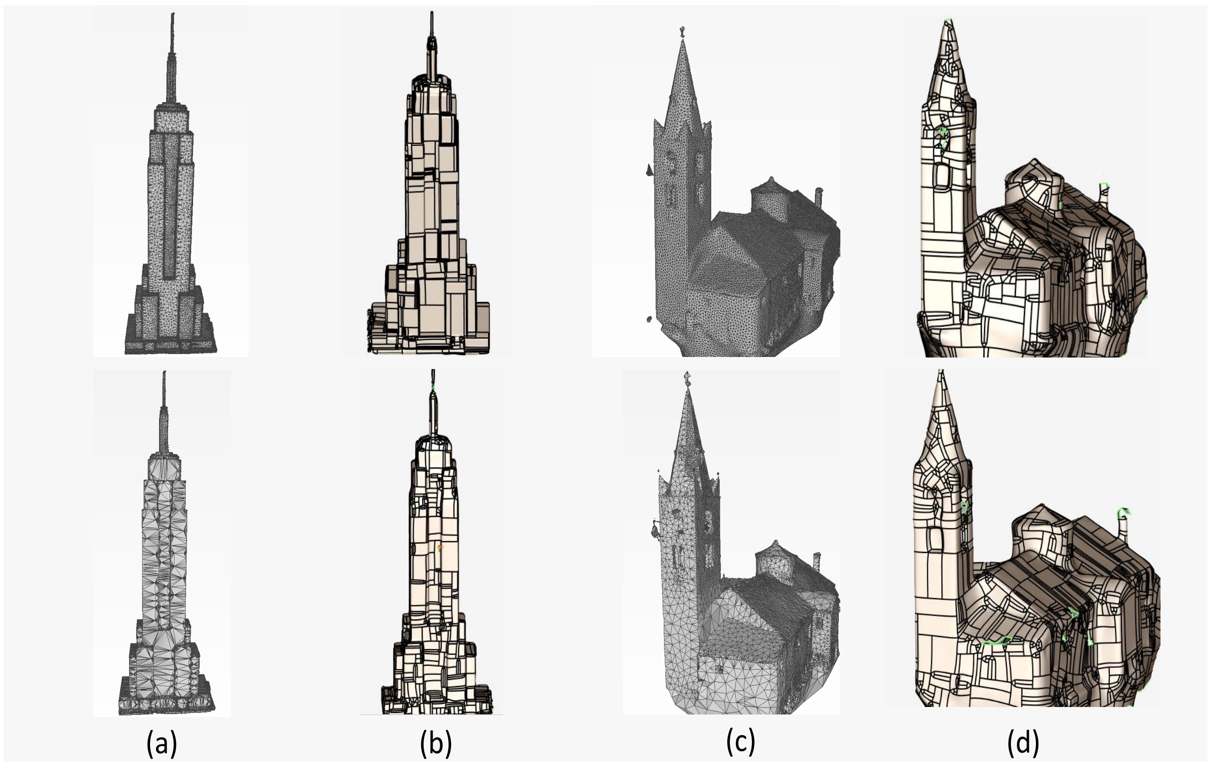


Figure 2.18: *3D CAD models and their simplified version. (a) Empire State (triangular mesh), (b) Empire State (B-Rep), (c) Lans (triangular mesh) and (d) Lans (B-Rep).*

Table 2.2: *Statistics related to different 3D representations.*

3d Models	**Data Structure**	**Models**	**# Vertices**	**# Faces**	**Storage**
Empire State	Polygonal Mesh (a)	Original	432,514	861,642	63.9
		Simplified	72,695	141,903	19.2
	Boundary Representation (b)	Original	2477	1675	27.2
		Simplified	1900	1263	21.5
Lans	Polygonal Mesh (c)	Original	25,994	51,426	13.48
		Simplified	18,543	31,598	11.48
	Boundary Representation (d)	Original	3534	2367	36.4
		Simplified	3349	2254	35.1

Table 2.3: *Comparison between data structures.*

Data Structure	**Topology**	**Geometry**	**Application**
Polygonal Mesh	vertices, edges and faces	surfaces	Continuous surfaces
Constructive Solid Geometry	geometric primitives	surfaces, curves and points	CAD models
Boundary Representation	vertices, edges and faces	surfaces, curves and points	CAD models
Stellar Decomposition	vertices, edges and faces	surfaces	Continuous surfaces

surface while CSG and B-Rep can be classified into solid modeling with geometric elements as surfaces, curves and points. From topological perspective, polygonal mesh, B-Rep and stellar decomposition rely on vertices, edges and faces. However, CSG-based representations do not store such topological relation and thus, only uses collection of geometric primitives to represent shapes. This implies that CAD based representations (such as CSG and B-Rep) are useful when model exhibits dominant planar regions. However, in case of complex architectural details, a triangular mesh model would be more suitable to better represent the level of detail.

CHAPTER 3

Light-weight Modeling Algorithms

In this chapter, we review various state-of-the-art light-weight modeling techniques including mesh decimation, geometric abstractions, deep learning based algorithms and level of detail (LOD) modeling.

3.1 Mesh Decimation

Mesh decimation deals with the reduction in complexity of mesh structures by reducing faces. In [(Bouzas *et al.*, 2020)], the authors proposed a novel approach for the polygonization of Multiview Stereo (MVS) meshes of buildings which results in compact and topologically valid models. For polygonization, the planar components of the input mesh along with their topology in the 3D space get detected based on region growing technique and then an initial set of candidate's faces to approximate the meshes are generated. Then, an optimization method constructs the simplified surface models considering sharp features through a building scaffold and faces through 2D arrangements. In [(González *et al.*, 2009)], a user-assisted mesh simplification method proposed that converts CAD models to triangle meshes and perform the simplification of each sub-object independently, at different level of details. For example, the user can desire a total number of triangles in the simplified model while some parts of the model are maintained or simplified to a definite percentage of simplification. Different levels of detail for different subobjects avoid the appearance of holes and preserve the boundaries between the sub-objects.

In [(Li and Nan, 2021)], an efficient mesh simplification method presented to reduce the complexity and storage size of the urban models while preserving the structure and sharp feature, for instance, sharp contours of building models. The method comprises filtering and simplification of 3D building mesh models to preserve piecewise planar structures. In [(Lescoat *et al.*, 2020)], authors proposed a method to simplify a mesh using edge collapses while targeting to preserve the input eigen vectors and eigen values through functional maps (a linear mapping between function spaces). This method supports the preservation of spectral properties and offers the similar storage size of simplified methods based on [(Garland and Heckbert, 1997)] but with a higher quality Laplacian. In [(Wang *et al.*, 2021*a*)], a topology-preserving mesh simplification method proposed for 3D building models. First, the method classifies building into different segments representing different components. Then, the vertices of the model get divided into different categories such as boundary vertices, hole vertices and other regular vertices. Then, cost related to edges is determined for instance, for a boundary edge, the angle between the edge and component (E-C angle) is introduced to estimate an error metric to skip the edge collapses at adjacent areas of building components. An improved quadratic error metric (QEM) is explained further to address the unexpected error in the case of hole vertices.

3.2 Geometric Abstractions

Due to increasing complexity of geometric information, meaningful and concise abstractions become helpful for higher level interaction and modeling. More specifically, geometric abstraction allows to reconstruct 3D models by using various primitives, resulting model being low-polygonal and light-weight. RANSAC [(Fischler and Bolles, 1981)] is a popular preprocessing technique for geometric abstraction. The method extracts shapes from point cloud data by randomly drawing minimal sets (minimum points that define a geometric primitive) and constructs respective primitive shapes [(Schnabel *et al.*, 2007)]. In [(Monszpart *et al.*, 2015)], a method proposed to reconstruct man-made scenes by extracting Regular Arrangement of Planes (RAP) from raw point cloud scans. First, the point set gets oversegmented with the help of region growing to group nearby points into planar patches. The method produces a set of candidate primitives with the help of local analysis and inter-primitive relations. Finally, a regular arrangement of planes gets extracted based on data fitting to point cloud and compact resulting arrangement. In [(Huang *et al.*, 2017)], a 3DLite framework proposed to reconstruct 3D environments using RGB-D sensors. This method computes a light-weight, low-polygonal geometric abstraction of the scanned geometry and produces high-resolution, sharp surface textures of the 3D models. In [(Chen *et al.*, 2021*b*)], a method presented that reconstructs compact, watertight, polygonal building 3D models from point clouds with the help of learnable implicit field using a deep neural network. An implicit field can be used to extract a smooth surface model of the object by learning directly from the point cloud. Then, a Markov random field

(MRF) extracts the compact surface of the building through combinatorial optimization.

In [(Xu *et al.*, 2022)], a method presented for building reconstruction based on geometrical segmentation of point clouds. First, points divided into voxels with the help of a clustering algorithm [(Lin *et al.*, 2018)]. The voxels get further assessed and classified based on quality and type of shape, for instance, curved shapes such as spheres, cylinders and cones. To separate the points into corresponding segments, a hybrid voting RANSAC approach is used. To improve the efficiency and robustness in complex scenes, voxel-based connectivity are analyzed. Furthermore, to deal with bad segmentation and spurious shapes, a graph-cut-based optimization is incorporated. In [(Xie *et al.*, 2021)], a method presented for efficient reconstruction of building models from photogrammetric point clouds by combining rule-based and hypothesis-based methods. Initially, the planar primitives and respective boundaries are extracted from the point cloud which get regularized to obtain abstracted building counters. Then, a two-stage reconstruction method generated 3D building models. In the first stage, to recover the topological relationship between different primitives, the regularity and adjacency of the building counters are used resulting an initial reconstruction model. In the next stage, an integer linear optimization problem solved to remove and reconstruct topologies related to ambiguous areas. In [(Zhang *et al.*, 2021)], a method presented for automatic reconstruction of a 3D building facade model from photogrammetric mesh model. First, the mesh model is divided into components based on contour line. Local contour trees exploited to find the segmented contour graphs based on analyzing the topological relationship between the contours. Through an iterative process, whole model segmented into diverse components from bottom to top. Next, the mesh model components approximated by minimum circumscribed cuboids in an iterative manner. Finally, to ensure the accuracy of the reconstructed facade model, the parameters of the cuboid model adjusted by means of a least square process.

3.3 Deep Learning based Algorithms

In the last decade, the ample accessibility of large training data sets has facilitated the researchers with the possibility of working on data-driven techniques to produce large 3D urban models using data priors. These learning-based models are capable of handling big datasets of urban scenes having lots of different objects such as buildings, roads, vegetation, parking area, etc. Many methods [(Li *et al.*, 2019; Bosch *et al.*, 2019; Liu and Ji, 2020; Fan *et al.*, 2021; Kelly *et al.*, 2018; Du *et al.*, 2020)] have addressed the urban reconstruction from multi-view images or satellite imagery; while point cloud based urban modeling [(Zhang and Zhang, 2017; Zhang *et al.*, 2018)] has been considered by fewer researchers.

The satellite images provide top view of urban areas but fail to provide scene information of buildings from other viewpoints. [(Li *et al.*, 2019)] utilizes high resolution satellite images along with Digital Terrain Model (DTM) and geographic tagged maps to reconstruct 3D model for complex urban scenes. This method uses two CNNs, first to classify land covers and another for building height estimation. Other scene components such as water area, road nets and building samples are estimated from the initial modalities provided to the system and then these are further processed and integrated to build virtual 3D urban scene at very high speed. This work uses lightweight CNN model with high acceleration rate which makes this approach convenient and flexible for urban simulation tools, although it is limited to the flat roof structures. CNN model discussed in [(Bosch *et al.*, 2019)] examines the utility of lightweight baseline architectures for scene and appearance changes in satellite images due to various seasons. This paper studies the necessity of authentic selection of image pair in multi-view 3D reconstruction. Multi-view reconstruction network RED-Net introduced in [(Liu and Ji, 2020)] achieves high efficiency and resolution in large scale reconstruction with lesser memory requirements. Also, a very recent web-based interactive platform [(Fan *et al.*, 2021)] VGI3D, works with VGI images to generate lightweight 3D building models using CNN. The platform realizes fast solution with lower time and labour costs. Also, CNN helps in detecting building façades of different and complex architectural style buildings.

Similar to CNNs, Generative Adversarial Network (GAN) is another deep learning approach which is becoming famous in image-based 3D reconstruction community. FrankenGAN [(Kelly *et al.*, 2018)] uses simplified modeling process to obtain realistically detailed mass models. Also, the memory requirement of GAN architecture is compensated by employing individual texturing of windows and super resolution GAN. An improvement to FrankenGAN is proposed in [(Du *et al.*, 2020)], which has replaced BicycleGAN [(Zhu *et al.*, 2017)] with StarGAN [(Choi *et al.*, 2018)] architecture to generate higher quality textures.

Point clouds carry 3D spatial information for efficient 3D reconstruction of objects or scenes. In a recent deep learning based framework [(Chen *et al.*, 2021*b*)], authors presents a method to reconstruct compact, watertight, polygonal building 3D models from point clouds with the help of learnable implicit field using a deep neural network. An implicit field can be used to extract a smooth surface model of the object by learning directly from the point cloud. Then, a Markov random field (MRF) extracts the compact surface of the building through combinatorial optimization. [(Zhang and Zhang, 2017; Zhang *et al.*, 2018)] uses 2.5D dual contouring method to produce lightweight 3D models from ALS point clouds of large residential areas. The deep reinforcement learning framework proposed in [(Zhang and Zhang, 2017)] and a rectified linear unit neural network proposed in [(Zhang *et al.*, 2018)] do not require large training data for network parameter learning for point cloud labelling which results in the significant reduction in the parameter tuning cost. Overall, both approaches are suitable for 3D modeling of irregular, complex roof components and small structures; however [(Zhang and Zhang, 2017)] sometimes struggles in learning discriminative features from smaller point clouds and [(Zhang *et al.*, 2018)] is limited by its high computation time. Authors in [(Bauchet and Lafarge, 2019)] reconstruct urban environments as compact polyhedral meshes. While most of the buildings can be reconstructed accurately, the proposed scheme needs improvement in dealing with free-form shapes and small structures of buildings. Furthermore, this method processes large data volumes in short time and hence this approach is sufficiently fast and scalable. In [(Lin *et al.*, 2013)], a supervised learning-based building modeling scheme is proposed for low-rise residential houses. For visual enhancement of reconstructed models, automatic billboard-tree approach is used which resulted in the rich, clean, and visually pleasing models of plant.

Another paper [(Xu *et al.*, 2020)] uses point clouds generated from multi-view satellite imagery for urban model reconstruction where deep learning helps in determining the roof shape in noisy and complex scenes. For efficient roof primitive segmentation and reconstruction, hierarchical RANSAC technique is used which shortens the computation time. Another promising direction to efficient 3D reconstruction is model synthesis which generates large 3D models based on compact example models, automatically. While [(Merrell and Manocha, 2008, 2010)] are classical model synthesis approaches generalized to higher dimension models, there are a smaller number of learning-based model synthesis techniques developed yet. Authors in [(Despine and Colleu, 2015)] uses adaptive procedural modeling approach to enhance texture quality for virtual city models. It extracts information from low resolution aerial images for roof texture reconstruction. The method relies on machine learning for analysing the images using pixel-based detection. Few works [(Wang *et al.*, 2021*b*; Ulyanov *et al.*, 2017)] have discussed CNN based texture synthesis on general dataset but model/ texture synthesis is still having lots of opportunity for urban modeling.

3.4 Level of Detail Modeling

3D city models and their level of detail (LOD) describe the fineness and complexity desirable for compact representation-based computer graphics applications. In [(Verdie *et al.*, 2015)], a method explained that reconstructs levels of detail (LODs) based on 3D urban scenes. Multi-view stereo systems based surface meshes are used as input that further proceeds into later stages of: classification, abstraction and reconstruction. First, the scene gets classified into four meaningful classes, that is, *ground, tree, facade* and *roof* with the help of geometric attributes and a set of semantic rules in association with a Markov random field. Then, in the next step planar structures on buildings get detected which leads to filter stage and simplifies LOD generation. The method pipeline feeds this regularized and filtered proxies to reconstruction submethod that generates watertight models. The next stages global regularization, LOD filtering and simplification generates light-weight polygonal meshes while ensuring the structure.

In [(Chen *et al.*, 2017)], a topologically aware 2.5D building rooftop reconstruction method explained from airborne laser scanning point clouds based on three steps pipeline: building rooftop primitive clustering; primitive boundary representation; and 2.5D building model reconstruction at multiple LoDs. In the first stage, rooftop primitives are clustered with the help of probability density clustering (PDC) technique based on topological consistency between primitives. In the next stage, subgraph of voronoi is used to extract primitive boundaries which further are divided into multiple linear segments to generate key points which are helpful in generating hybrid representation of the boundary. The hybrid key points-based model representation offers flexibility and accurate rooftop details to generate light-weight building models. Finally, the boundaries of primitives are assembled to reconstruct and render building models at multiple-level details.

In [(Zhu *et al.*, 2018)], a large-scale urban scene modeling framework explained comprising multi-view stereo (MVS) meshes as input and generates simplified models with different levels of detail (LODs) based semantics. First, the scene gets segmented into 4 different classes such as *ground, grass, tree* and *building* with the help of a Markov random field (MRF). Then, 2D orthograph images from 3D meshes used to further incorporate height and image features in the initial segmentation step. In the next step, building modeling is implemented by reconstructing buildings into LOD models. Different 2D line segments based on roof boundaries are detected and extruded with different faces to assign a roof plane label. Thus, produced models with different LODs.

In [(Han *et al.*, 2021)], a modeling framework explained for aerial images and textured 3D models based on large-scale urban scenes that generates compact polygonal models with semantics at different level of details. The framework comprises different stages such as scene segmentation, roof contour extraction and building modeling. First, by using deep neural network the scene is segmented into 3 classes: *ground, vegetation* and *building*. Then, the 2D line segments based on roof contours are detected that divide the ground into polygon cells. A roof plane gets assigned to each polygon cell with the help of Markov random field optimization technique. In the final stage, by extruding cells to different planes, building models with different LODs obtained.

3.5 Summary and Remarks

We summarize our review on various lightweight modeling algorithms based on their applicability and advantages and disadvantages in the context of urban scene modeling including man-made environments.

Geometric abstraction-based methods are reliable for understanding and analyzing urban scene layouts as well as indoor 3D environments. As major part of an urban scene comprises buildings with dominant planar regions, primitive based fitting methods appear suitable for approximating city scenes. Various geometric abstraction-based techniques have been employed in numerous works such as segmented aerial 3D point clouds differentiating buildings, ground-objects, and vegetation. Façade parsing and modeling is another application area where geometric abstraction-based techniques appear suitable. Once, simpler approximations are available, segmented point clouds can be substituted with CAD-based primitives in order to address lightweight aspect. However, these methods are only limited to simpler approximations with primitive types (plane, cylinder, cone etc.) and therefore, cannot be applied to complex objects/scenes in real-world examples.

Deep learning-based methods to generate 3D city scenes have been introduced recently with the availability of large training data sets of different objects such as buildings, roads, vegetation etc. Due to unorganized nature of point clouds, training neural network directly from point clouds is not an easy task. In a recent work [(Chen *et al.*, 2021*b*)], framework for reconstructing compact, watertight, polygonal building models from point clouds is presented in which learnable implicit fields are used to characterize 3D shape and surface extracted through Markov random field (MRF). Although this method generates valid models with accurate structural geometry; missing level of detail makes these models unsuitable for an interactive urban scene application. We believe that neural networks can be optimized to learn faster and better not only to approximate 3D shapes but also to render complex details from point clouds.

Level of Detail (LoD) modeling-based methods have shown prominent results, generating lightweight yet detailed representations. These methods have been used in various works that take aerial images and/or textured 3D models as input and generates compact polygonal models. To automatically reconstruct city scenes with different LoDs (level of details), method pipeline including stages such as scene segmentation and feature extraction appears appropriate at various occasions for efficient representation. Scene segmentation can help in categorizing objects such as buildings, vegetation and ground while feature exaction can help in obtaining rooftop contours. Some methods further incorporate mesh simplification to reduce polygons in order to maintain simplification. However, these methods require user assistance and are not suited well for reconstructing large scale city scenes.

There are numerous works on mesh decimation for reconstructing urban scenes as polygonal meshes from dense LiDAR-based point clouds offers detailed topological information. Though, point clouds comprise noise, outliers and other defects, techniques such as mesh filtering and denoising can smooth noisy and uneven density of shapes approximated with triangular meshes. By reducing faces, complexity of mesh structures can be reduced thereby making meshes as one of the prominent candidates for lightweight modeling. Various models based on mesh simplification and polygonization offer lightweight representation of the geometry, however these models lack in the level of detail required due to their compact nature. For instance, façade level details and complex architectural patterns are often omitted or ignored in various methods. Various works also talk about detailed simplified mesh models by incorporating various schemes such as using local correspondents, custom edge collapse operations with feature and local geometric error metrics etc. However, in the context of reconstructing detailed urban scene these techniques are not adequate. We provide our mesh decimation algorithm with experiments and comparative study in the next chapter (see chapter 4). The quality of our method is evaluated on various building models and the performance is validated by comparisons to other state-of-the-art techniques.

CHAPTER 4

Mesh Decimation Through Gradient Tensors

Efficient mesh decimation is an essential part of this light-weight modeling. In this chapter, an overview of our mesh simplification method is described followed by in depth discussion.

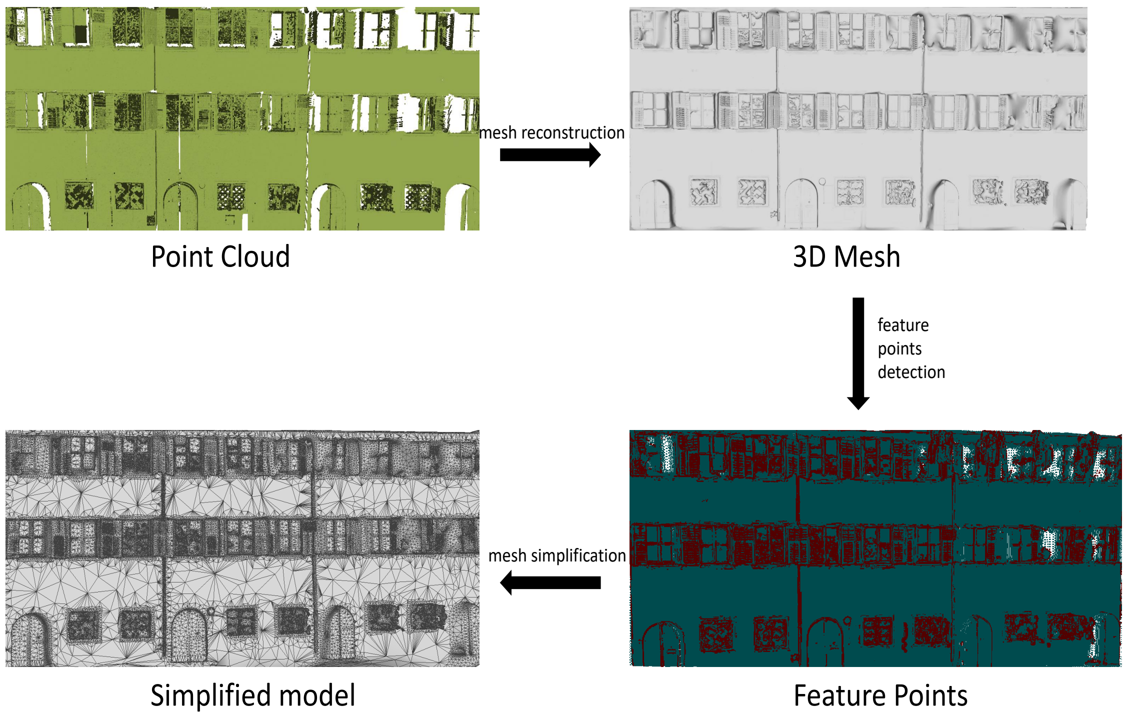


Figure 4.1: *Our mesh decimation pipeline.*

4.1 Overview

Urban mesh simplification based methods [(Garland and Heckbert, 1997), (Li and Nan, 2021), (Bouzas *et al.*, 2020)] showed great potential in last few years addressing the challenges of visualization, storage and processing of 3D models by reducing the faces to make models less complicated while preserving the essential geometry and sharp contours to generate compact representations. Considering, architectural variations in real-world urban scenes such as Manhattan and European style buildings with complex facade element structure, these methods do not perform well in terms of level of detail. Therefore, we consider preserving facade element details as part of critical features in the final simplified models. In this chapter, we review our mesh decimation pipeline (Figure 4.1) that takes 3D meshes of buildings reconstructed from dense point clouds with the help of screened poisson reconstruction technique [(Kazhdan and Hoppe, 2013)]. We take vertices of these meshes as topological data for feature points detection method that extracts feature points (such as corner, edge, boundary points) through gradient structure tensors. Then, our customized mesh simpification method decimates the input mesh without losing the extracted feature points in the final simplified model. The quality of our simplified results is evaluated on different building models and the effectiveness is validated by comparisons to the different state-of-the-art techniques. The key contributions of our proposed work are as follows:

- Mesh Simplification through Structure Tensor: Extracting feature points from a building mesh using gradient structure tensor and associate it with GH [(Garland and Heckbert, 1997)] based mesh simplification method with the help of custom cost penalty function.
- Empirical Study with Comparisons: the created simplified meshes keep consistent and good fidelity to the original meshes (in terms of Hausdorff distance and RMS error (see Table 4.1 and 4.2)) in comparison to other state-of-the-art techniques.
- Simplicity, Efficiency and Robustness: simple and efficient implementation of simplification with architectural variations retained for highly complex models with same fidelity and efficiency.

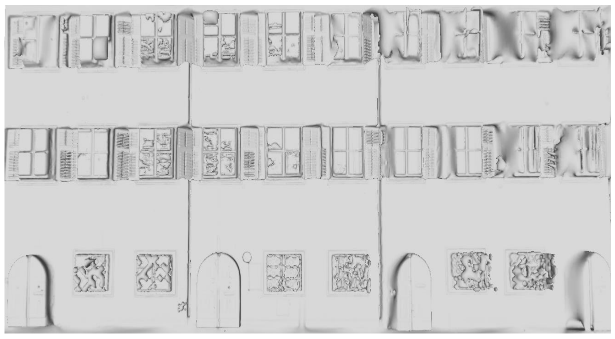
4.2 Mesh Reconstruction

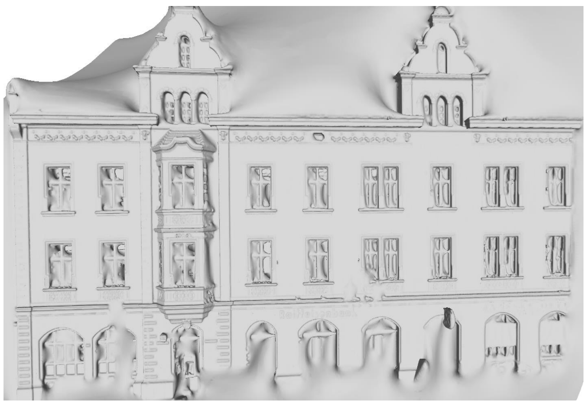
Although 3D point clouds are prominent structures to represent geometry for navigation and interaction applications, direct exploitation of geometry is not an easy task [(Bassier *et al.*, 2020)]. On the other hand, 3D meshes reconstructed from point clouds offer well defined topological information and are extensively used as an input for urban scene interpretation. Implicit function based surface reconstruction is one of the ways for approximating a surface and one such popular technique is poisson surface reconstruction [(Kazhdan *et al.*, 2006)] that creates watertight surfaces from point sets by solving a Poisson equation. An indicator function of a model is estimated by finding a function χ such that the difference between its gradient $\nabla\chi$ and vector field $\vec{V}$ is minimized, that is, $\min_\chi \|\nabla\chi - \vec{V}\|$. The gradient of the indicator function is a vector field which is equal to the inward surface normal. Therefore, the oriented point samples can be seen as samples of the gradient of the indicator function. The poisson surface reconstruction [(Kazhdan *et al.*, 2006)] minimizes the function:

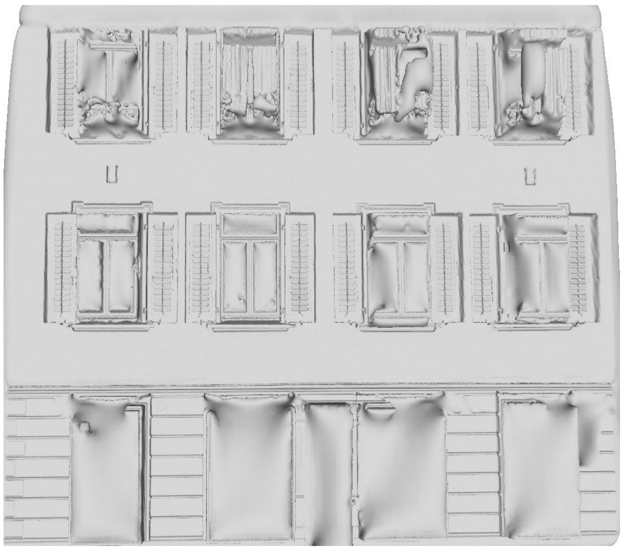
$$E(\chi) = \int \|\nabla\chi(p) - \vec{V}(p)\|^2 dp \tag{4.1}$$

However, in some situations, this method does not consider every points positions resulting model divering. Thus, we refer to [(Kazhdan and Hoppe, 2013)] for mesh reconstruction method in our pipeline that creates surfaces and further incorporates adaptions related to weighted positions of the points resulting final model faithful to the initial point cloud. Because, point clouds used for mesh reconstruction are raw, we first estimate the normals by using normal estimation method from [(Cignoni *et al.*, 2008)] that fits local planes and then uses them to estimate normals. Once normals are estimated, the underlying surface of geometry can be represented with the help of marching cubes algorithm [(Lorensen and Cline, 1987)] by building an octree data structure [(Wilhelms and Van Gelder, 1992)] and dividing the point cloud into a voxel grid that creates triangles by analyzing isosurface (reconstructed surface) of the model. There are different parameters in screened poisson reconstruction method that affect the reconstruction results:

- **Reconstruction Depth** : Maximum octree depth D used for surface reconstruction.
- **Samples Per Node** : Minimum number of sample points that should be contained in an octree node.
- **Point Weight** : Interpolation of point samples.
- **Scale Factor** : Ratio between the diameters of the cube used for reconstruction and the samples' bounding cube.
- **Normals** : Flag to output vertex normals estimated from the gradients of the implicit function.


(a) birdfountain_station1.ply : vertices 635k, faces 1271k, depth 10


(b) domfountain_station1.ply : vertices 558k, faces 1116k, depth 12


(c) stgallencathedral_station6.ply : vertices 2028k, faces 4056k, depth 12

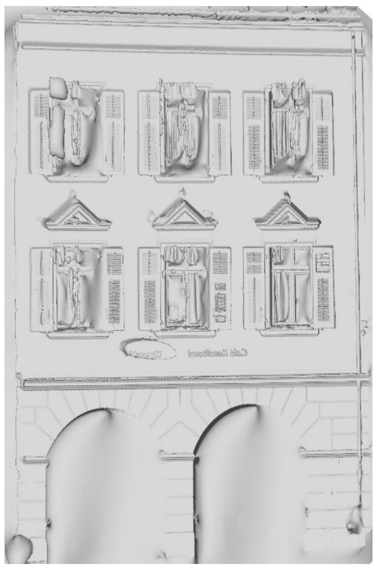

(d) marketplacefeldkirch_station7.ply : vertices 1132k, faces 2264k, depth 12

Figure 4.2: *Reconstructed meshes through screened poisson surface reconstruction.*

Parameters *reconstruction depth* and *samples per node* have major impact on the mesh reconstruction. The higher value of the depth results in the production of more detailed meshes as the voxels get finer due to the deeper hierarchy of octree. Similarly, a smaller value of samples per node parameter (sampling density) allows to put less points into the octree node resulting in generating high details. In case of noisy data, smaller depth value and higher sampling density value could be used to keep the surface reconstruction smooth. However, it will result in loss of detail. Figure 4.2 represents meshes reconstructed from Semantic3D point cloud dataset [(Hackel *et al.*, 2017)] which demonstrates buildings.

4.3 Feature Points Detection

After obtaining reconstructed meshes from point clouds, we take vertices of the meshes as topological information and obtain critical feature points (corner, edge or boundary points) related to urban buildings such as building facades by analyzing gradient structure tensors. We refer to [(Chen *et al.*, 2021a)] method for the detection of critical feature points to be preserved in mesh decimation stage. First, the confidence of vertex set is calculated to measure the local quality and to quantify the probability of vertex being a feature point. The confidence characteristic is then used in the meaningful interpretation of the gradient for each point. The gradient of vertex set is encoded into a 3 x 3 structure tensor, with eigenvalues indicating the gradient variations in the local regions. The feature point set representing the building facade, gets extracted by a dual-threshold method representing critical points by analyzing confidence and the gradient structure tensor of each point.

Confidence Estimation. To understand the local quality of vertex point set, it is essential to mesure the confidence of each point by analyzing the eigenvalue of covariance matrix of vertex v_i. Geometric attributes such as fitting quality C_f and sampling uniformity C_s are used to estimate the confidence where fitting quality represents the condition of the local tangent plane at vertex v_i and calculated by $C_f^i = \lambda_0^i / \left(\lambda_0^i + \lambda_1^i + \lambda_2^i\right)$. Here, $\lambda_0^i \leq \lambda_1^i \leq \lambda_2^i$ represents eigenvalues of the covariance matrix. If the value C_f^i tends to be 0, it means the vertex v_i fit the local tangent plane, however, if the value approaches 1 then the vertex v_i could be located at facade corner, facade edges and facade boundaries. While fitting quality deals with the quality of tangent plane, sampling uniformity describes the uniform distribution and calculated by $C_s^i = \lambda_1^i / \lambda_2^i$. If the value tends to be 0, the vertices are distributed linearly, however, if the value approaches 1 then the vertices are distributed uniformly. The quality of vertices are estimated by focusing on small neighborhood sphere centered at individual point. To keep the values accurate, different scales of neighborhood sphere with radius of 1.0, 1.5 and 2.0 times of mean point density of vertex set are used. The overall confidence measure $Conf^i \in [0, 1]$ of vertex v_i is calculated as follows:

$$Conf^i = 1 - \frac{1}{n} \sum_{k=1}^{n} \left(1 - 3C_f^i\right) \cdot C_s^i \tag{4.2}$$

where, n represents the number of scales. If the value of $Conf^i$ comes out to be 0, the corresponding vertices have high local fitting quality and could be located on wall or inside facade such as window glass. In case of value 1, the corresponding vertices are likely to be located on facade boundaries, corners and window frames.

Gradient Definition and Structure Tensor Generation. After calculating the confidence for each vertex, the gradient $G(x, y, z)$ for the vertex set can be defined as maximized directional derivative of a differentiable function $f(x, y, z)$ with direction vector $u = \cos(\alpha)i + \cos(\beta)j + \cos(\gamma)k$ in 3D space such that, $G(x, y, z) = max(D_u f)$ where angles α, β, and γ represents the angles between the direction vector *u* and positive three axes; and $D_u f = \lim_{t \to 0} \frac{f(x+t\cos(\alpha), y+t\cos(\beta), z+t\cos(\gamma)) - f(x,y,z)}{t}$. Considering the directional derivative to be maximum when the gradient of function $\nabla f(x, y, z)$ and direction vector *u* are in same direction, the gradient value $G(x, y, z)$ of vertex $v(x, y, z)$ can be defined as $\max_{j \in N} \left(\frac{C^i - C^j}{d_{ij}} \right)$, where C^i and C^j represent the estimated confidence of current vertex v_i and its neighborhood vertex v_j respectively and d_{ij} represents the euclidean distance from vertex v_i to vertex v_j.

Considering change of gradient $G'(\Delta x, \Delta y, \Delta z)$ of vertex set $v(x, y, z)$ as:

$$\sum_{v(x,y,z)} \left[G_{x+\Delta x, y+\Delta y, z+\Delta z} - G_{x,y,z} \right]^2 \tag{4.3}$$

using taylor expansion with $O\left(\Delta x^2 + \Delta y^2 + \Delta z^2\right)$, equation (4.2) can be interpreted as:

$$G'(\Delta x, \Delta y, \Delta z) = \sum_{v(x,y,z)} \left[\Delta x \cdot g_x + \Delta y \cdot g_y + \Delta z \cdot g_z + O\left(\Delta x^2 + \Delta y^2 + \Delta z^2\right) \right]^2 \tag{4.4}$$

which can be written as: $(\Delta x, \Delta y, \Delta z) M (\Delta x, \Delta y, \Delta z)^T$

where $M = \sum_{v(x,y,z)} \begin{bmatrix} g_x^2 & g_x g_y & g_x g_z \\ g_y g_x & g_y^2 & g_y g_z \\ g_z g_x & g_z g_y & g_z^2 \end{bmatrix}$ that is, gradient structure tensor of the local region of the vertex set where a point from the vertex set could either be a corner point (at the intersection of three non-parallel surfaces), edge point (belongs to the intersection edges) or boundary point (outer boundaries or inner holes).

Dual-Threshold Criterion. To detect feature points, i.e., vertices being critical points, a response function based on the three eigenvalues of smoothed gradient structure tensor M is

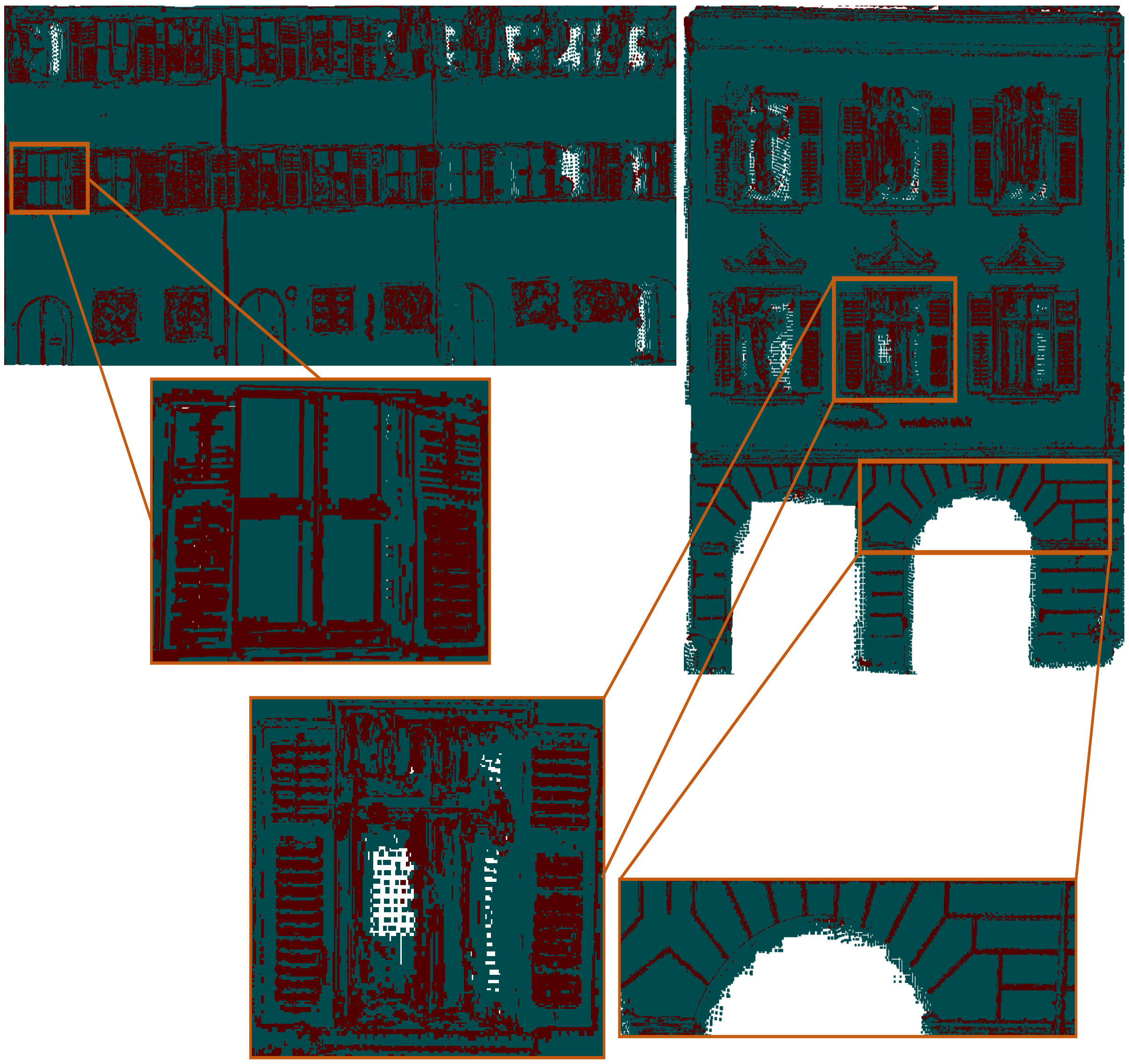


Figure 4.3: *Feature points representing critical facade elements of buildings.*

defined as:

$$R_F^{\bar{M}} = \mathrm{f}(\bar{M}) - k[\mathrm{g}(\bar{M})]^3$$

$$s.t. \begin{cases} \mathrm{f}(\bar{M}) = \lambda_0^{\bar{M}} \cdot \lambda_1^{\bar{M}} \cdot \lambda_2^{\bar{M}} \\ \mathrm{g}(\bar{M}) = \lambda_0^{\bar{M}} + \lambda_1^{\bar{M}} + \lambda_2^{\bar{M}} \end{cases}$$

The value of $R_F^{\bar{M}}$ indicates the status of point being critical feature point (that is, whether point being corner, edge or boundary) or general point with the help of dual-threshold criterion comprising combination of eigenvalues of gradient structure tensor $\bar{M}$ such that if value of $R_F^{\bar{M}}$ is greater than or equal to a predefined threshold $T_{\bar{M}}$; and second constraint as $G(x, y, z) \geq T_G$ here T_G is a predefined gradient threshold. Hence, for a point to be critical feature point $G(x, y, z) \geq T_G$ and $R_F^{\bar{M}} \geq T_{\bar{M}}$ conditions should be true. Figure 4.3 represents critical feature points of meshes obtained in previous step by setting 2.25 and 25 as threshold values for T_G and $T_{\bar{M}}$.

4.4 Decimation

After obtaining feature points from vertex set, we associate gradient structure tensor definition with GH [(Garland and Heckbert, 1997)] mesh decimation method to retain feature points in simplification stage. Our idea is to utilize the original error quadrics computed for each vertex through GH method and identify whether the corresponding vertex is a feature point or not, according to eigenvalue analysis from gradient structure tensor. If the corresponding vertex is a feature point, we preserve it during the simplification stage by adjusting the error computation. By not changing the placement of the collapsed vertex, efficiency of quadric error metric for shifting the collapsed vertices can be utilized.

Quadric Error Metric (QEM). By applying the edge collapse operation iteratively, two vertices v_1 and v_2 can be merged to a unique vertex $\widehat{v}$, i.e. $v_1, v_2 \to \widehat{v}$. For an edge e_{12} based on vertices v_1 and v_2, a cost $\Delta_{e_{12}}$ can be defined with an error metric to identify whether the two vertices should be collapsed or not. For each vertex *v* of an input mesh, a quadric Q_v can be defined as a 4x4 matrix such that $Q_v = PP^T$, where *P* = [a b c d]^T representing plane associated with vertex *v* and $\Delta_{qem}(\boldsymbol{v}) = \boldsymbol{v}^T Q_{\boldsymbol{v}} \boldsymbol{v}$ defines the error of the vertex *v* as sum of squared distances to its adjacent planes, i.e. $\sum \left(\boldsymbol{P}^\top \boldsymbol{v}\right)^2$. The error metric can be written as follows:

$$\begin{aligned}\Delta_{qem}(\boldsymbol{v}) &= \sum_{\boldsymbol{p} \in \text{ Planes } (\boldsymbol{v})} \left(\boldsymbol{v}^\top \boldsymbol{p}\right)\left(\boldsymbol{p}^\top \boldsymbol{v}\right) \\ &= \boldsymbol{v}^\top \left(\sum_{\boldsymbol{p} \in \text{ planes } (\boldsymbol{v})} \boldsymbol{Q}_{\boldsymbol{v}}\right) \boldsymbol{v}\end{aligned}$$

where $\boldsymbol{Q}_{\boldsymbol{v}} = \boldsymbol{P}\boldsymbol{P}^\top = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$

Hence, the error approximation $\overline{\boldsymbol{Q}}$ for $\widehat{v}$ becomes $\boldsymbol{Q}_1 + \boldsymbol{Q}_2$. For each valid pair (v_1, v_2), the estimated error value using QEM defines cost based on which all vertex pairs get placed in a priority queue data structure with the lowest cost at top ready for collapsing operation followed with an update on the costs of other valid pairs.

Algorithm overview. Our proposed method (see Figure 4.1) takes as input: a mesh M reconstructed from point cloud; a feature vertex set F_v representing extracted critical points from feature points detection stage; and outputs a simplified mesh M'' by analyzing gradient structure tensor M = $\begin{bmatrix} g_x^2 & g_x g_y & g_x g_z \\ g_y g_x & g_y^2 & g_y g_z \\ g_z g_x & g_z g_y & g_z^2 \end{bmatrix}$ of the local region of vertex set. For a feature vertex set F_v that also belongs to vertices V of mesh M, we analyze the corresponding eigenvalue distribution from gradient structure tensor definition and identify the status of vertex. In order to be a feature point, i.e., corner, edge or boundary point the corresponding vertex must reflect larger eigenvalues.

- Corner point: If the current vertex is at the intersection of three non-parallel surfaces (facades and rooftop planes in this case), all three corresponding eigenvalues will be large and the vertex could be a corner point.
- Edge point: If the vertex lies at the intersection edges, then two relative eigenvalues will be large and point will be considered as edge point.
- Boundary point: In case of one large eigenvalue, the vertex could be coming from the outer or inner boundaries (for instance, window frames) then the point will be considered as boundary point.

If $v(x, y, z)$ belongs to any of these three categories, we adjust the error computation by

$$\Delta_{qem}(\boldsymbol{v})_{\text{new}} = \Delta_{qem}(\boldsymbol{v})_{\text{old}} + \frac{1}{n}\sum_{i=1}^{n} \Delta_{qem}(\boldsymbol{v})_{\text{old}} \tag{4.5}$$

where $\frac{1}{n}\sum_{i=1}^{n} \Delta_{qem}(\boldsymbol{v})_{\text{old}}$ is the mean cost of n vertices of mesh M. Otherwise, the corresponding vertex will be treated with conventional QEM criterion.

We tested our mesh decimation approach on Semantic3D [(Hackel *et al.*, 2017)] dataset, a popular large-scale point cloud classification benchmark comprising terrestrial laser scans from Central Europe's urban and rural scenes. All scenes were captured through a surveying-grade

Algorithm 1: Structure tensor based mesh decimation

```
   Input: Feature vertex set $F_v : (v_1, v_2, ..., v_n) \in V$, mesh $M = (V, F)$
   Output: Simplified mesh $M" = (\overline{V}, \overline{F})$
1  function meshDecimation($V_{x,y,z}$)
2      $queue = \{\}$
3      forall vertices v in feature vertex set $F_v \in$ mesh M, identifying valid vertex pair
         based edges $(e_{ij})$ : $(v_i, v_j)$ is an edge or $\|\mathbf{v}_1 - \mathbf{v}_2\| < t$, where t is a threshold
         parameter do
4          insert $(e_{i,j})$ to queue
5          compute optimal collapsed target $\widehat{v}$
6          while queue not empty do
7              if eigenvalues$(\lambda_0^{\bar{M}}, \lambda_1^{\bar{M}}, \lambda_2^{\bar{M}}) \rightarrow max : T_G \geq 2.25$ and $T_{\bar{M}} \geq 25$ then
8                  update error with mean cost, i.e. $\Delta_v = \boldsymbol{v}^T Q_{\boldsymbol{v}} \boldsymbol{v} + \frac{1}{n}\sum_{i=1}^{n} \Delta(\boldsymbol{v})$
9              else
10                 compute error with $\Delta_v = \boldsymbol{v}^T Q_{\boldsymbol{v}} \boldsymbol{v}$
11             arrange pairs with cost (e,c) and collapse edge e with minimum cost
12         update costs of other valid pairs with respect to vertex $v_1$
13     return
```

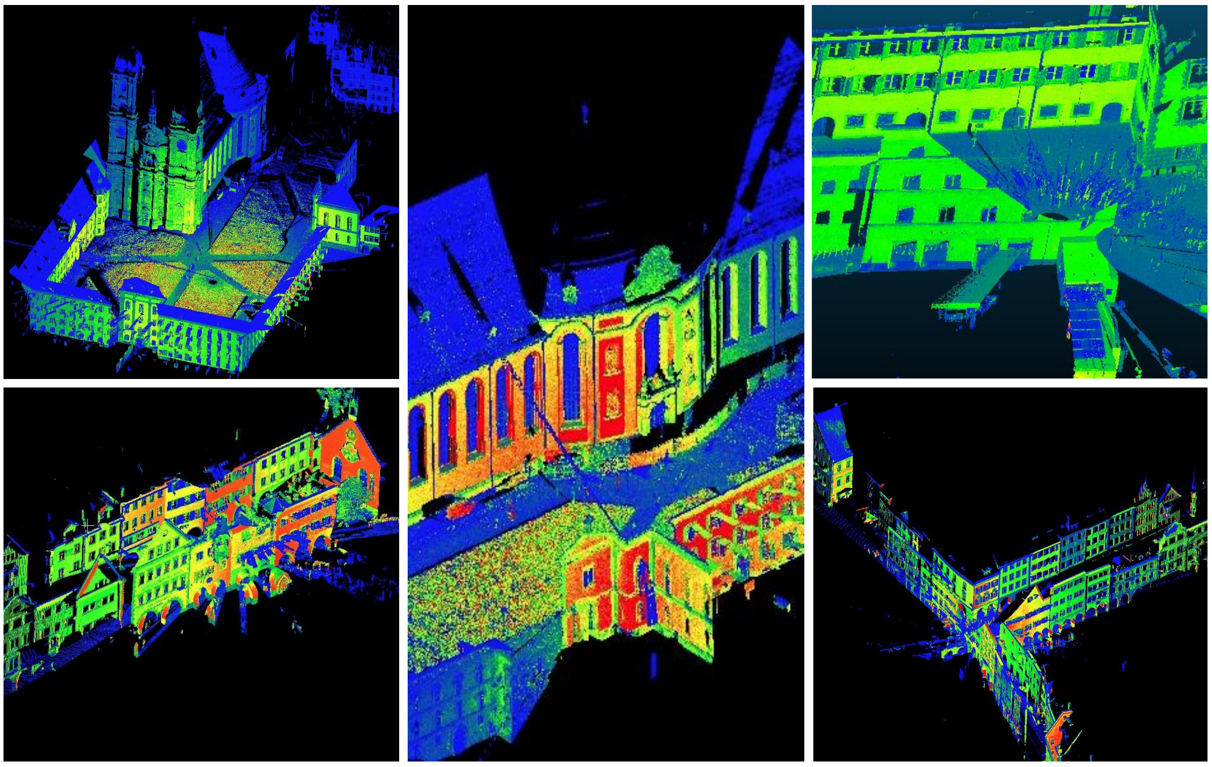


Figure 4.4: *Example point clouds of the Semantic3D dataset [(Hackel* et al.*, 2017)].*

laser scanner and describe typical european architecture. Figure 4.4 represents example point clouds from Semantic3D dataset.

For experimentation, we extracted various building samples from Semantic3D dataset com-

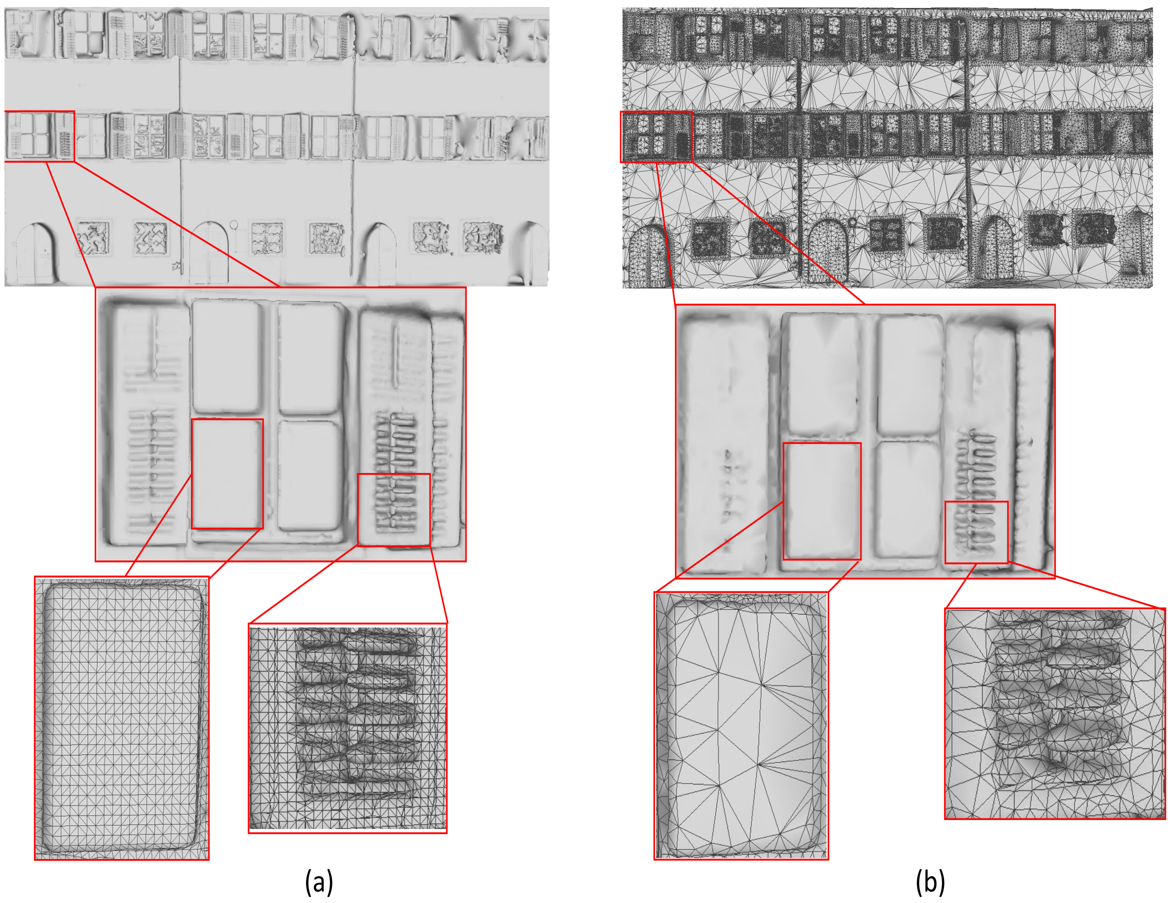


Figure 4.5: *Mesh decimation preserving feature points through structure tensor. (a) An instance of birdfountain_station1 model (b) simplified version with faces retained at feature areas.*

prising different architectural variations. Then, we reconstructed 3D meshes from extracted building point cloud samples through screened poisson reconstruction technique [(Kazhdan and Hoppe, 2013)]. After obtaining feature points from these reconstructed meshes, we analyze the eigenvalue distribution from gradient structure tensor and decimates according to the status of the vertex. If the current vertex belong to critical feature class, we preserve it by adjusting error approximation cost. However, for non-critical points, we follow general error approximation cost through GH [(Garland and Heckbert, 1997)] mesh decimation method. Figure 4.5 represents an instance of a building from birdfountain_station1 dataset from Semantic3D and our corresponding simplified model version. By keeping error approximation cost high for points that show higher eigenvalues on gradient structure tensors, we prevent the decimation of critical faces. As can be seen in Figure 4.5 (b), we have retained many feature points such as corner, edge and boundary points representing facade elements and necessary geometric structure in simplified version. However, we noticed that not all points were able to retained especially points lying very fine on the window frame. We will discuss some of these issues in next chapter addressing achievements and limitations.

4.5 Results and Experiments

We applied the proposed method to a variety of building mesh models with different noise levels and architectural variations for qualitative analysis. To validate the effectiveness, we quantify our simplified results in terms of compression ratio between final simplified faces to orignal faces. To measure the accuracy of geometry, we calculate the mean and root mean square (RMS) value of the hausdorff distances [(Guthe *et al.*, 2005)] between the original and simplified meshes. We also compare our simplified results with state-of-the-art methods available such as, GH method [(Garland and Heckbert, 1997)], LT method's [(Lindstrom and Turk, 1998)] and SLA method [(Salinas *et al.*, 2015)] for better evaluation.

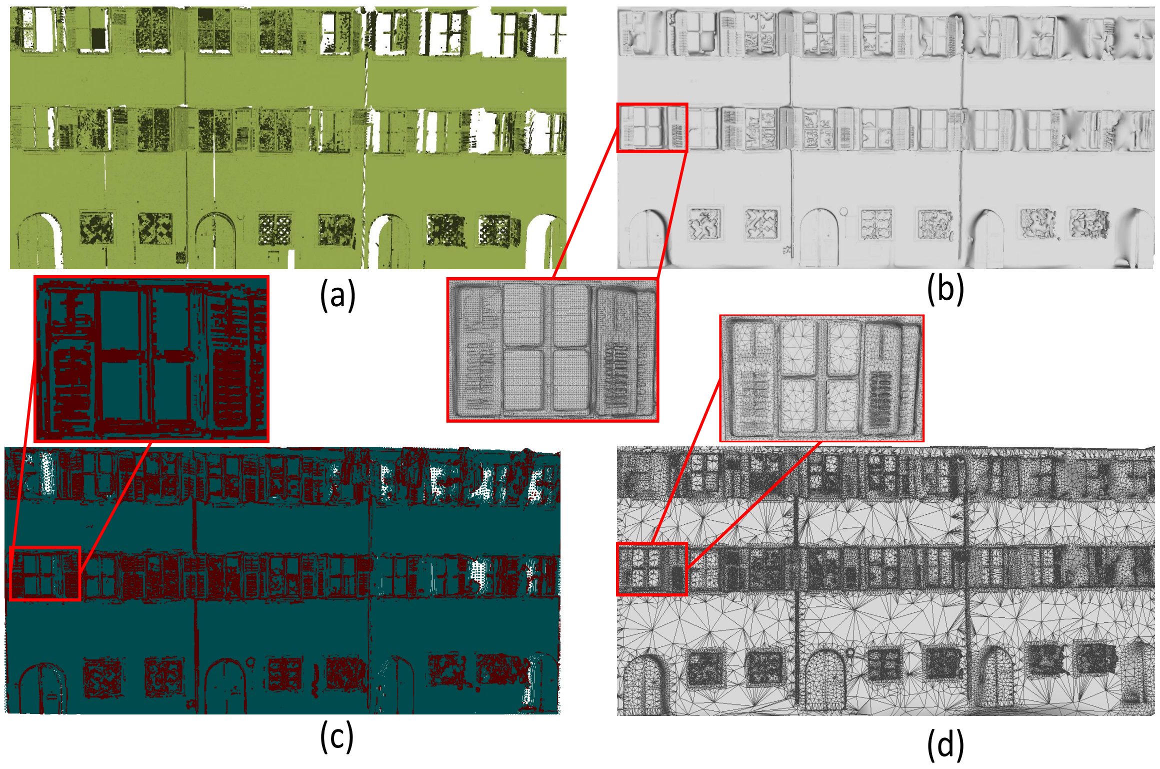


Figure 4.6: *Efficient mesh simplification of birdfountain model. From (a) to (d), initial point cloud, reconstructed mesh, feature points extracted, final simplified model retaining feature points presenting facade elements of model.*

4.5.1 Qualitative Analysis

Figure 4.6 represents our simplification method on *birdfountain model.* The proposed method efficiently extracts feature points representing facade elements such as window frames, boundary points and retains them in simplified model in the form of triangulations. While more triangles are being preserved to represent level of detail in simplified model, less triangles are kept in planar area such as wall to ensure the simplicity of model. We observed that points across the boundaries of the model are extracted in feature points stage and retained in simplified model resulting model being accurate in terms of geometric structure.

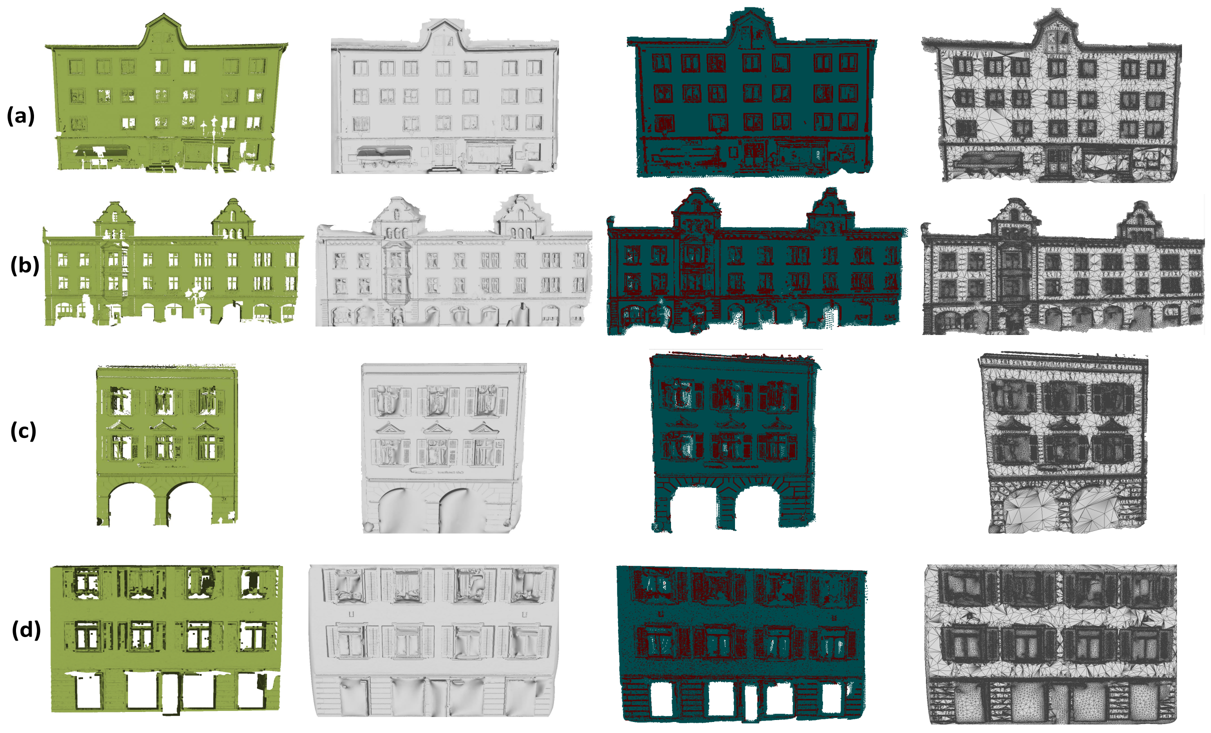


Figure 4.7: *Experimental results: (a) domfountain_station1 (2), (b) domfountain_station1 (1), (c) marketplacefeldkirch_station7 (2) and (d) stgallencathedral_station6.*

Figure 4.7 represents our experimental results on various building models comprising architectural variations. We observed that due to occlusion and noisy nature of initial point clouds, the reconstructed meshes generated contain inaccurate geometric elements such as blobs that does not relate to the original approximation. Although some of the faces representing blobs are retained because their vertices were extracted in feature points stage, our main focus here is to preserve the facade level definition of buildings such as roof area, windows, complex patterns over wall, soil pipes, doors and stairs in some cases. We also observed that due to free form simplification in planar areas or areas that do not contain any critical feature points, bad faces created however, we noticed that it does not affect the visualization of geometry. For well topological connectivity between facade elements (such as windows in this case) in final simplified model, we purposely restricted simplification to a certain threshold such that no holes should emerge in simplified version.

As can be seen in Figure 4.7, the proposed method efficiently extracted critical feature points and retained them in final simplified model while preserving the general structure of the buildings. This demonstrates that the proposed method can be applied to a variety of models with highly complicated architectural details.

4.5.2 Quantitative Analysis

To evaluate the effectiveness of proposed method for efficient mesh simplification, we calculated statistics for level of simplification and geometric accuracy. The simplification was calculated by compression ratio[1] between final simplified faces to the original faces. To evaluate the accuracy, we quantified the deviation based on the mean and root mean square (RMS) value of the hausdorff distances[2] indicating the distance between the vertices of original mesh to the faces of the simplified model. Table 4.1 lists the calculated statistics.

Table 4.1: *Statistics on sizes and errors for the simplification results.*

Model	**Original / Simplified face #**	**Compression ratio**	**Hausdorff distance**	
			Mean	**RMS**
birdfountain_station1	1334898 / 216182	16.19%	0.461	1.250
domfountain_station1 (2)	1694396 / 307719	18.16%	0.002	0.013
domfountain_station1 (1)	1116037 / 389596	34.90%	0.002	0.007
marketplacefeldkirch_station7	2264074 / 395018	17.44%	0.001	0.007
stgallencathedral_station6	4056124 / 608781	15.0%	0.005	0.019

As the compression ratio achieved by our proposed method is larger, we can conclude that due to preserving critical feature points, a large set of faces retained in final simplified meshes.

[1]Refers to a percentage or ratio expressing the difference between the size of a file before and after compression.

[2]Widely used thorough comparison technique for polygonal models among computer graphics community.

Complex architectural variations comprising various patterns and facade element details extracted and retained as feature points in simplified model. From hausdorff distance based error estimation, we can see how less our models are deviated from original meshes indicating simplified models are accurate and comply with the original structural geometry. Though mesh decimation based state-of-the-art techniques (see next section 4.5.3) offer good compression ratio through compact representations that preserve general structures, they do not offer much level of detail and thus, can not be considered appropriate for interactive applications. With our quantitative analysis, we have given insights related to the existing tradeoff between simplification and accuracy.

4.5.3 Comparison

We compared our simplified results with the GH method's [(Garland and Heckbert, 1997)], the LT method's [(Lindstrom and Turk, 1998)] and the SLA method's [(Salinas *et al.*, 2015)] results for better evaluation. Figures 4.8, 4.9 and 4.10 shows the comparison between our method and other decimation methods. Table 4.2 lists corresponding quantitative statistics. We observed that the simplified meshes generated by our method were more precise in representing the structure and facade level definition, considering an acceptable tradeoff, than the other methods.

Figure 4.8 represents a building instance of birdfountain_model with different simplified versions including state-of-the-art techniques and our simplified model. As can be seen, that original model comprises various facade element details including patterned window frame. Our method decimates meshes efficiently and retain most of the feature points in simplified version. Figure 4.9 represents a building instance of domfountain_model (1) along with simplified version from our method and other techniques. It is evident from the Figure that our method is effective in preserving complex architectural details such as pillar, european style window/balcony architecture etc. Similarly, our simplified version in Figure 4.10 represents best geometric detail of the facade regions, for instance, window inner boundaries and outside boundaries (window frame). Our method was able to preserve both of these boundaries as part of feature set. While our approach works for different models, other state-of-the-art techniques such as the GH method's [(Garland and Heckbert, 1997)], the LT method's [(Lindstrom and Turk, 1998)] and the SLA method's [(Salinas *et al.*, 2015)] did not generate acceptable models

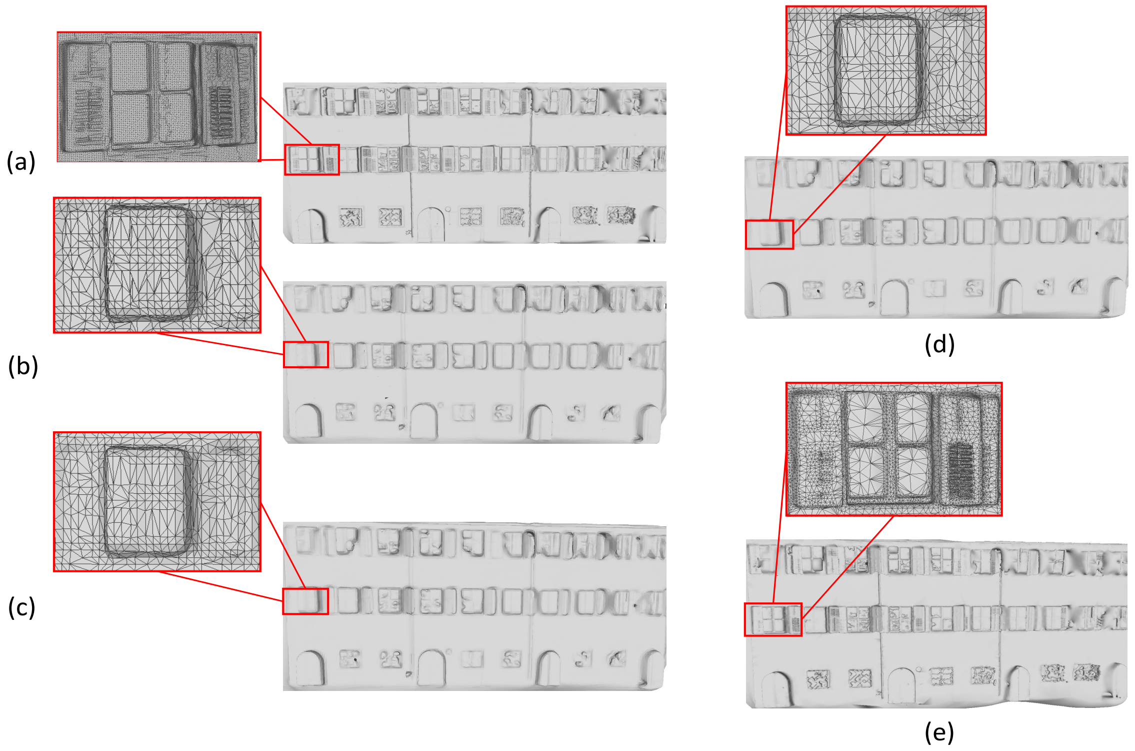


Figure 4.8: *Comparison on birdfountain_model. (a) Initial mesh model, (b) GH's method, (c) LT's method, (d) SLA's method and (e) our model.*

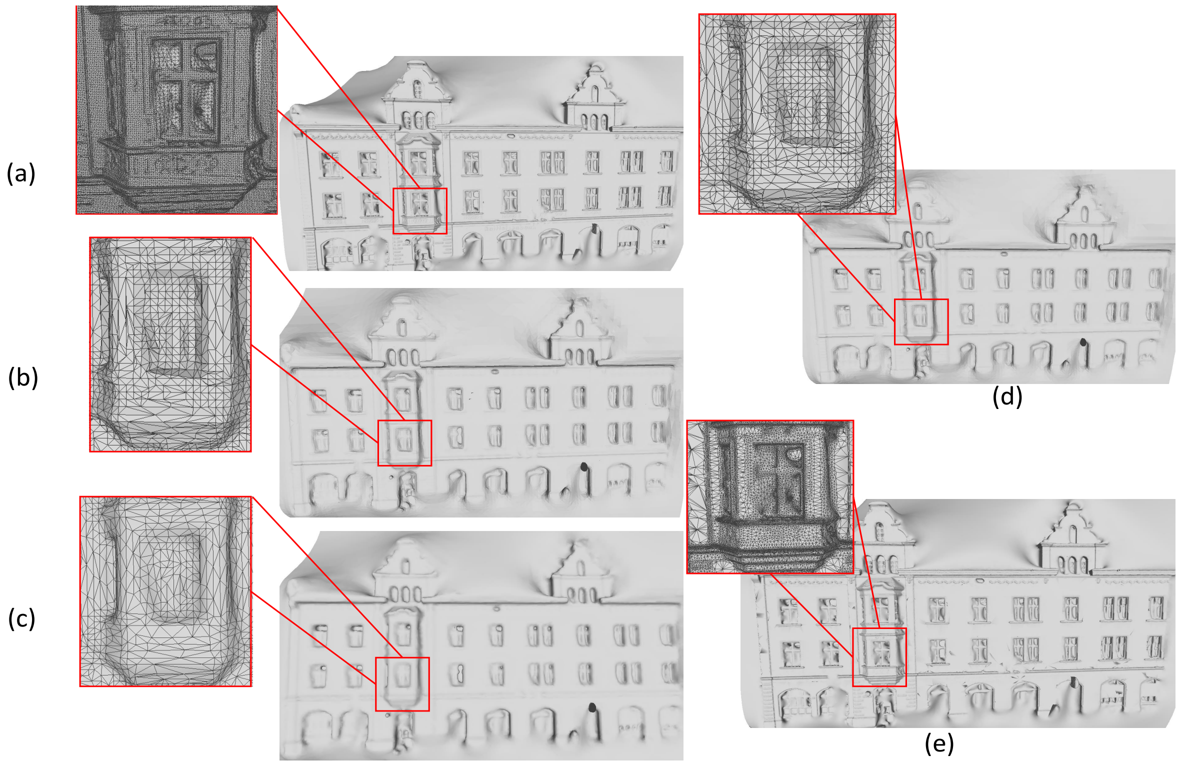


Figure 4.9: *Comparison on domfountain_model (1). (a) Initial mesh model, (b) GH's method, (c) LT's method, (d) SLA's method and (e) our model.*

in comparison to our models.

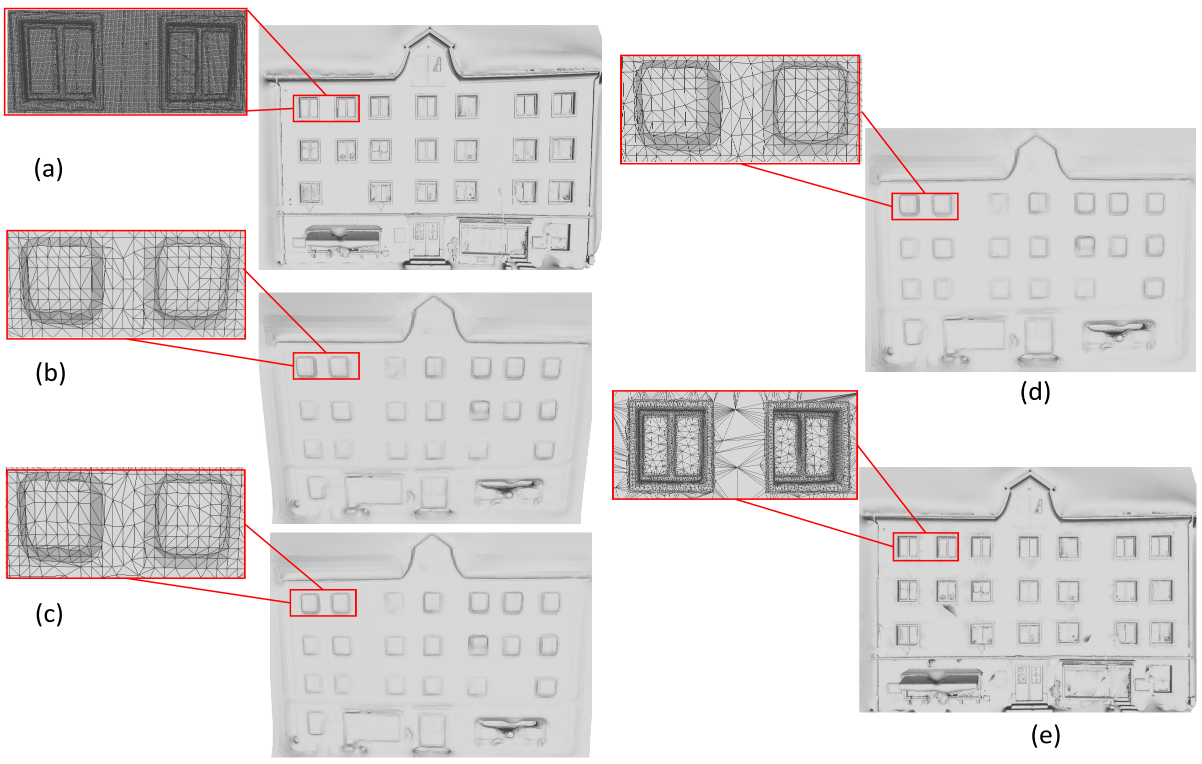


Figure 4.10: *Comparison on domfountain_model (2). (a) Initial mesh model, (b) GH's method, (c) LT's method, (d) SLA's method and (e) our model.*

Table 4.2: *Comparison with other methods based on simplicity and fidelity statistics.*

Model	**Compression ratio**	**Hausdorff distance**		**Method**
		Mean	**RMS**	
birdfountain_station1	3.08%	0.509	1.224	GH
	3.02%	0.503	1.217	LT
	3.02%	0.510	1.225	SLA
	16.19%	0.461	1.250	Ours
domfountain_station1 (1)	4.08%	0.558	1.530	GH
	3.94%	0.558	1.518	LT
	4.01%	0.554	1.514	SLA
	34.90%	0.002	0.007	Ours
domfountain_station1 (2)	1.25%	0.092	0.218	GH
	1.20%	0.092	0.218	LT
	1.15%	0.090	0.216	SLA
	18.16%	0.002	0.013	Ours

Table 4.2 lists statistics of the models discussed above, in terms of compression ratio and hausdorff distance (Mean and root mean square (RMS) values). Hausdorff distance explains

the error deviation of simplified model in comparison to the initial model. Although compression ratio offered in our approach is not satisfactory in comparison to other methods, due to preservation of faces. We noticed our simplified models show more fidelity as they are less deviated from original geometry.

CHAPTER 5

Conclusion and Future Directions

5.1 Summary

In this thesis, we analyze the latest approaches which apply decimation of 3D urban meshes. Then we introduce the decimation of 3D meshes using gradient structure tensors following an eigenvalue analysis from structure tensor, to preserve critical feature areas, i.e., geometric structure and facade elements. The method first reconstructs 3D meshes from point clouds representing buildings using screened poisson reconstruction technique [(Kazhdan and Hoppe, 2013)]. After mesh reconstruction, we extract vertices from meshes and provide them to feature points detection method proposed in [(Chen *et al.*, 2021*a*)] that extracts feature points, i.e., corner, edge, boundary points through gradient structure tensors. After feature points detection, an improved mesh decimation method is proposed to efficiently simplify the meshes while retaining the feature areas. We compute the status of vertex by analyzing the eigenvalue distribution of a structure tensor patch. If the eigenvalues are large, we consider the point as feature point and adjust its error approximation cost in order to prevent it from decimation. For all other general points, we follow conventional quadric error metric decimation scheme. As shown in experimental results, the proposed method generates decimated meshes without losing feature points and can be applied to various models with different architectural variations.

During our initial approach with mesh decimation through gradient structure tensor, we experimented with penalty cost value for extracted feature points. Although feature preserving metrics and functional maps [(Lescoat *et al.*, 2020)] based techniques allow to preserve descriptors and landmark correspondents. However, associating the functional basis with gradient structure tensor definition might get challenging and this might lead to unnecessary topology changes in mesh. Therefore, we restricted our experiments to different values of penalty cost by trial and error. We analyzed various simplified results generated based on different values tested, however, none of them gave accurate results. We assume error approximation cost for feature points were not adjusted enough to be preserved. By taking penalty cost as mean error value, we observed most of the feature points preserved and our method produce more detailed models.

The experiments we performed on a variety of building models explain that our method is effective in generating efficient simplified models in terms of simplicity and fidelity. We also demonstrated the validity of our experimental results by comparing against the state-of-the-art methods in urban mesh decimation. An ideal simplified model should be visually consistent to the initial model and must comprise limited complexities required to display the model without losing structures. However, for some complex structures such as facade elements across curved regions, feature points may not be detected, which can lead to vertex point being considered as non-criticial point, resulting excessive decimation and loosing facade level of detail. Figure 5.1 represents simplification of a model based on our proposed method. We noticed that across curved regions, our method does not extract feature points well, resulting model over-simplified in those critical regions. One way to overcome this would be to manually select correspondents from original mesh and apply remeshing. However, this would deviate us from our original goal that is, efficient mesh simplification. Hence, to overcome these scenarios in terms of mesh simplification, we propose future directions of our work to address current limitations.

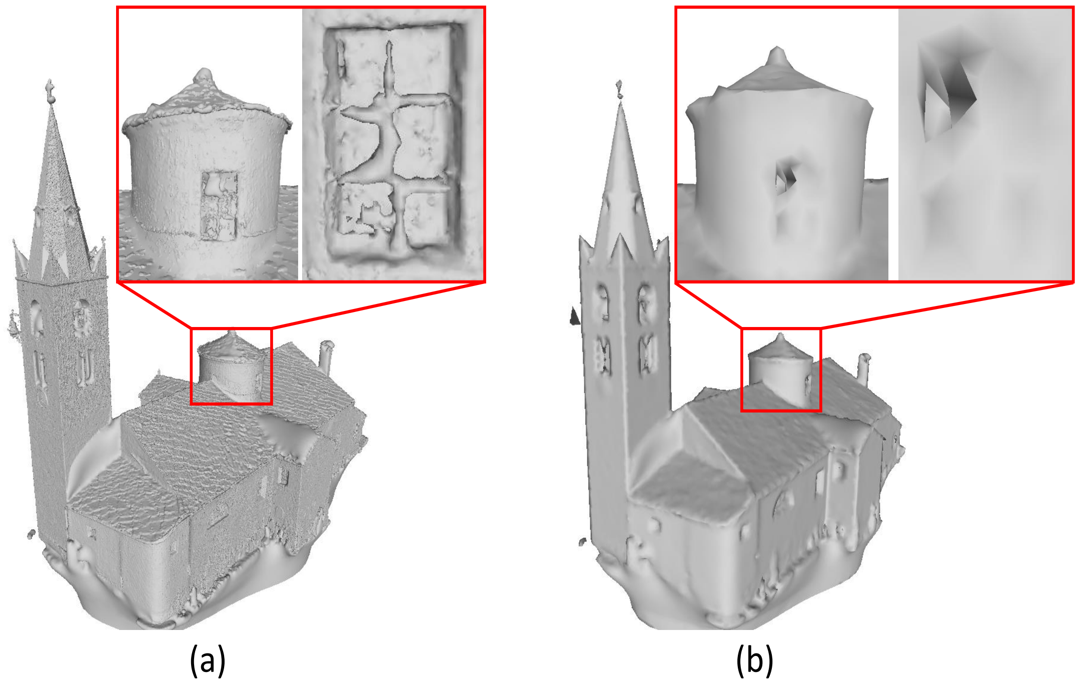


Figure 5.1: *Simplification of model with curved regions. (a) Original model, (b) Simplified model*

5.2 Future Directions

Further evaluations of our method by measuring the accuracies of all the methods [(Garland and Heckbert, 1997; Lindstrom and Turk, 1998; Salinas *et al.*, 2015)] under the same compression ratio is suggested. From our experiments, it is evident that each mesh representation is different and comprises specific elements that should or should not be preserved, subjected to user requirement in a 3D interactive application. By applying mesh segmentation and associating with structure graph topology, different mesh regions can be processed with a suitable scheme corresponding to given geometry. This would also be helpful to choose correspondents across curved regions and simplify accordingly. We notice that the level of simplification on planar regions in our results is not satisfactory, comparing with other mesh decimation techniques. Therefore, we also intend to investigate a better way to extract planar regions to reconstruct the simplified models by retaining feature points and associating planar primitives in order to generate compact representations of models.

5.3 Publications based on this Thesis

Kamra et al. "Lightweight Modeling of Urban Buildings: Data Structures, Algorithms and Future Directions", IEEE JSTARS (under review)

REFERENCES

1. **Bassier, M.**, **M. Vergauwen**, and **F. Poux** (2020). Point cloud vs. mesh features for building interior classification. *Remote Sensing*, **12**(14), 2224.

2. **Bauchet, J.-P.** and **F. Lafarge**, City reconstruction from airborne lidar: a computational geometry approach. *In 3D GeoInfo 2019-14thConference 3D GeoInfo*. 2019.

3. **Baumgart, B. G.** (1972). Winged edge polyhedron representation. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE.

4. **Bobenko, A. I.**, **T. Hoffmann**, and **B. A. Springborn** (2006). Minimal surfaces from circle patterns: Geometry from combinatorics. *Annals of Mathematics*, 231–264.

5. **Bommes, D.**, **B. Lévy**, **N. Pietroni**, **E. Puppo**, **C. Silva**, **M. Tarini**, and **D. Zorin**, Quad-mesh generation and processing: A survey. *In Computer Graphics Forum*, volume 32. Wiley Online Library, 2013.

6. **Bosch, M.**, **K. Foster**, **G. Christie**, **S. Wang**, **G. D. Hager**, and **M. Brown**, Semantic stereo for incidental satellite images. *In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019.

7. **Bouzas, V.**, **H. Ledoux**, and **L. Nan** (2020). Structure-aware building mesh polygonization. *ISPRS Journal of Photogrammetry and Remote Sensing*, **167**, 432–442.

8. **Chen, D.**, **J. Li**, **S. Di**, **J. Peethambaran**, **G. Xiang**, **L. Wan**, and **X. Li** (2021*a*). Critical points extraction from building façades by analyzing gradient structure tensor. *Remote Sensing*, **13**(16), 3146.

9. **Chen, D.**, **R. Wang**, and **J. Peethambaran** (2017). Topologically aware building rooftop reconstruction from airborne laser scanning point clouds. *IEEE Transactions on Geoscience and Remote Sensing*, **55**(12), 7032–7052.

10. **Chen, Z.**, **S. Khademi**, **H. Ledoux**, and **L. Nan** (2021*b*). Reconstructing compact building models from point clouds using deep implicit fields. *arXiv preprint arXiv:2112.13142*.

11. **Choi, Y.**, **M. Choi**, **M. Kim**, **J.-W. Ha**, **S. Kim**, and **J. Choo**, Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

12. **Cignoni, P.**, **M. Callieri**, **M. Corsini**, **M. Dellepiane**, **F. Ganovelli**, **G. Ranzuglia**, *et al.*, Meshlab: an open-source mesh processing tool. *In Eurographics Italian chapter conference*, volume 2008. Salerno, Italy, 2008.

13. **Commons, W.** (2020). File:csg tree.png — wikimedia commons, the free media repository. URL `https://commons.wikimedia.org/w/index.php?title=File:Csg_tree.png&oldid=485443649`. [Online; accessed 3-May-2022].

14. **Despine, G.** and **T. Colleu** (2015). Adaptive texture synthesis for large scale city modeling. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

15. **Dr. C.-K. Shene** (2011). Constructive Solid Geometry, CS3621 Introduction to Computing with Geometry Notes. `https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/model/csg.html`. [Accessed: 2022-09-16].

16. **Du, Z.**, **H. Shen**, **X. Li**, and **M. Wang** (2020). 3d building fabrication with geometry and texture coordination via hybrid gan. *Journal of Ambient Intelligence and Humanized Computing*, 1–12.

17. **Fan, H.**, **G. Kong**, and **C. Zhang** (2021). An interactive platform for low-cost 3d building modeling from vgi data using convolutional neural network. *Big Earth Data*, **5**(1), 49–65.

18. **Fellegara, R.**, **F. Iuricich**, **L. De Floriani**, and **U. Fugacci**, Efficient homology-preserving simplification of high-dimensional simplicial shapes. *In Computer Graphics Forum*, volume 39. Wiley Online Library, 2020.

19. **Fellegara, R.**, **K. Weiss**, and **L. De Floriani** (2021). The stellar decomposition: A compact representation for simplicial complexes and beyond. *Computers & Graphics*.

20. **Fischler, M. A.** and **R. C. Bolles** (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24**(6), 381–395.

21. **Garland, M.** and **P. S. Heckbert**, Surface simplification using quadric error metrics. *In Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 1997.

22. **González, C.**, **J. Gumbau**, **M. Chover**, **F. Ramos**, and **R. Quirós** (2009). User-assisted simplification method for triangle meshes preserving boundaries. *Computer-Aided Design*, **41**(12), 1095–1106.

23. **Guennebaud, G.** and **M. Gross**, Algebraic point set surfaces. *In ACM SIGGRAPH 2007 papers*. 2007, 23–es.

24. **Guthe, M.**, **P. Borodin**, and **R. Klein** (2005). Fast and accurate hausdorff distance calculation between meshes.

25. **Hackel, T.**, **N. Savinov**, **L. Ladicky**, **J. D. Wegner**, **K. Schindler**, and **M. Pollefeys** (2017). Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv preprint arXiv:1704.03847*.

26. **Han, J.**, **L. Zhu**, **X. Gao**, **Z. Hu**, **L. Zhou**, **H. Liu**, and **S. Shen** (2021). Urban scene lod vectorized modeling from photogrammetry meshes. *IEEE Transactions on Image Processing*, **30**, 7458–7471.

27. **Hoffmann, C. M.** (1989). Geometric and solid modeling.

28. **Huang, J.**, **A. Dai**, **L. J. Guibas**, and **M. Nießner** (2017). 3dlite: towards commodity 3d scanning for content creation. *ACM Trans. Graph.*, **36**(6), 203–1.

29. **Kaiser, A.**, **J. A. Ybanez Zepeda**, and **T. Boubekeur**, A survey of simple geometric primitives detection methods for captured 3d data. *In Computer Graphics Forum*, volume 38. Wiley Online Library, 2019.

30. **Kazhdan, M.**, **M. Bolitho**, and **H. Hoppe**, Poisson surface reconstruction. *In Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7. 2006.

31. **Kazhdan, M.** and **H. Hoppe** (2013). Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, **32**(3), 1–13.

32. **Kelly, T.**, **P. Guerrero**, **A. Steed**, **P. Wonka**, and **N. J. Mitra** (2018). Frankengan: guided detail synthesis for building mass-models using style-synchonized gans. *arXiv preprint arXiv:1806.07179*.

33. **Kettner, L.** (1999). Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry*, **13**(1), 65–90.

34. **Kuang, Z.**, **B. Chan**, **Y. Yu**, and **W. Wang** (2013). A compact random-access representation for urban modeling and rendering. *ACM Transactions on Graphics (TOG)*, **32**(6), 1–12.

35. **Lescoat, T.**, **H.-T. D. Liu**, **J.-M. Thiery**, **A. Jacobson**, **T. Boubekeur**, and **M. Ovsjanikov**, Spectral mesh simplification. *In Computer Graphics Forum*, volume 39. Wiley Online Library, 2020.

36. **Li, M.** and **L. Nan** (2021). Feature-preserving 3d mesh simplification for urban buildings. *ISPRS Journal of Photogrammetry and Remote Sensing*, **173**, 135–150.

37. **Li, S.**, **Z. Zhu**, **H. Wang**, and **F. Xu** (2019). 3d virtual urban scene reconstruction from a single optical remote sensing image. *IEEE Access*, **7**, 68305–68315.

38. **Li, W.**, **G. Wolberg**, and **S. Zokai**, Lightweight 3d modeling of urban buildings from range data. *In 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*. IEEE, 2011.

39. **Lin, H.**, **J. Gao**, **Y. Zhou**, **G. Lu**, **M. Ye**, **C. Zhang**, **L. Liu**, and **R. Yang** (2013). Semantic decomposition and reconstruction of residential scenes from lidar data. *ACM Transactions on Graphics (TOG)*, **32**(4), 1–10.

40. **Lin, Y.**, **C. Wang**, **D. Zhai**, **W. Li**, and **J. Li** (2018). Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS journal of photogrammetry and remote sensing*, **143**, 39–47.

41. **Lindstrom, P.** and **G. Turk**, Fast and memory efficient polygonal simplification. *In Proceedings Visualization'98 (Cat. No. 98CB36276)*. IEEE, 1998.

42. **Liu, J.** and **S. Ji**, A novel recurrent encoder-decoder structure for large-scale multi-view stereo reconstruction from an open aerial dataset. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

43. **Lorensen, W.** and **H. Cline** (1987). Marching cubes: A high resolution 3d surface reconstruction algorithm, acm computer graphics, 21 (4): 163-169.

44. **Mäntylä, M.**, *An introduction to solid modeling*. Computer Science Press, Inc., 1987.

45. **Marschner, S.** and **P. Shirley**, *Fundamentals of computer graphics*. CRC Press, 2018.

46. **Merrell, P.** and **D. Manocha**, Continuous model synthesis. *In ACM SIGGRAPH Asia 2008 papers*. 2008, 1–7.

47. **Merrell, P.** and **D. Manocha** (2010). Model synthesis: A general procedural modeling algorithm. *IEEE transactions on visualization and computer graphics*, **17**(6), 715–728.

48. **Mildenhall, B.**, **P. P. Srinivasan**, **M. Tancik**, **J. T. Barron**, **R. Ramamoorthi**, and **R. Ng**, Nerf: Representing scenes as neural radiance fields for view synthesis. *In European conference on computer vision*. Springer, 2020.

49. **Ming, H.**, **D. Yanzhu**, **Z. Jianguang**, and **Z. Yong** (2016). A topological enabled three-dimensional model based on constructive solid geometry and boundary representation. *Cluster Computing*, **19**(4), 2027–2037.

50. **Monszpart, A.**, **N. Mellado**, **G. J. Brostow**, and **N. J. Mitra** (2015). Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, **34**(4), 103–1.

51. **Pottmann, H.**, **Y. Liu**, **J. Wallner**, **A. Bobenko**, and **W. Wang**, Geometry of multi-layer freeform structures for architecture. *In ACM SIGGRAPH 2007 papers*. 2007, 65–es.

52. **Rouhani, M.**, **F. Lafarge**, and **P. Alliez** (2017). Semantic segmentation of 3d textured meshes for urban scene analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, **123**, 124–139.

53. **Salinas, D.**, **F. Lafarge**, and **P. Alliez**, Structure-aware mesh decimation. *In Computer Graphics Forum*, volume 34. Wiley Online Library, 2015.

54. **Schnabel, R.**, **R. Wahl**, and **R. Klein**, Efficient ransac for point-cloud shape detection. *In Computer graphics forum*, volume 26. Wiley Online Library, 2007.

55. **scratchapixel.com** (). Bézier Curves and Surfaces: the Utah Teapot. `https://www.scratchapixel.com/lessons/geometry/bezier-curve-rendering-utah-teapot`. [Accessed: 2022-09-16].

56. **Shan, E. S. J.** (2002). Building modeling and visualization for urban environment.

57. **Steve Marschner** (2014). Triangle Meshes I, Cornell CS4620 Lecture 2. `https://www.cs.cornell.edu/courses/cs4620/2014fa/lectures/02trimesh1.pdf`. [Accessed: 2022-09-16].

58. **Tan, W.**, **N. Qin**, **L. Ma**, **Y. Li**, **J. Du**, **G. Cai**, **K. Yang**, and **J. Li**, Toronto-3d: A large-scale mobile lidar dataset for semantic segmentation of urban roadways. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020.

59. **Ulyanov, D.**, **A. Vedaldi**, and **V. Lempitsky**, Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

60. **Verdie, Y.**, **F. Lafarge**, and **P. Alliez** (2015). Lod generation for urban scenes. *ACM Transactions on Graphics*, **34**(ARTICLE), 30.

61. **Verma, V.**, **R. Kumar**, and **S. Hsu**, 3d building detection and modeling from aerial lidar data. *In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2. IEEE, 2006.

62. **Wang, B.**, **G. Wu**, **Q. Zhao**, **Y. Li**, **Y. Gao**, and **J. She** (2021*a*). A topology-preserving simplification method for 3d building models. *ISPRS International Journal of Geo-Information*, **10**(6), 422.

63. **Wang, R.**, **J. Peethambaran**, and **D. Chen** (2018). Lidar point clouds to 3-d urban models : a review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **11**(2), 606–627.

64. **Wang, W.**, **Y. Liu**, **D. Yan**, **B. Chan**, **R. Ling**, and **F. Sun** (2008). Hexagonal meshes with planar faces. *Dept. of CS, HKU, Tech. Rep.*

65. **Wang, Z.-M.**, **M.-H. Li**, and **G.-S. Xia** (2021*b*). Conditional generative convnets for exemplar-based texture synthesis. *IEEE Transactions on Image Processing*, **30**, 2461–2475.

66. **Wikipedia contributors** (2021). Constant-mean-curvature surface — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Constant-mean-curvature_surface&oldid=1046232365`. [Online; accessed 3-May-2022].

67. **Wikipedia contributors** (2022*a*). Boundary representation — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Boundary_representation&oldid=1085580003`. [Online; accessed 3-May-2022].

68. **Wikipedia contributors** (2022*b*). Minimal surface — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Minimal_surface&oldid=1065201247`. [Online; accessed 3-May-2022].

69. **Wikipedia contributors** (2022*c*). Simplicial complex — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Simplicial_complex&oldid=1070796321`. [Online; accessed 3-May-2022].

70. **Wilhelms, J.** and **A. Van Gelder** (1992). Octrees for faster isosurface generation. *ACM Transactions on Graphics (TOG)*, **11**(3), 201–227.

71. **Xie, L.**, **H. Hu**, **Q. Zhu**, **X. Li**, **S. Tang**, **Y. Li**, **R. Guo**, **Y. Zhang**, and **W. Wang** (2021). Combined rule-based and hypothesis-based method for building model reconstruction from photogrammetric point clouds. *Remote Sensing*, **13**(6), 1107.

72. **Xu, B.**, **Z. Chen**, **Q. Zhu**, **X. Ge**, **S. Huang**, **Y. Zhang**, **T. Liu**, and **D. Wu** (2022). Geometrical segmentation of multi-shape point clouds based on adaptive shape prediction and hybrid voting ransac. *Remote Sensing*, **14**(9), 2024.

73. **Xu, B.**, **X. Zhang**, **Z. Li**, **M. Leotta**, **S.-F. Chang**, and **J. Shan** (2020). Deep learning guided building reconstruction from satellite imagery-derived point clouds. *arXiv preprint arXiv:2005.09223*.

74. **Yan, J.**, **K. Zhang**, **C. Zhang**, **S.-C. Chen**, and **G. Narasimhan** (2014). Automatic construction of 3-d building model from airborne lidar data through 2-d snake algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, **53**(1), 3–14.

75. **Zhang, L.**, **Z. Li**, **A. Li**, and **F. Liu** (2018). Large-scale urban point cloud labeling and reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, **138**, 86–100.

76. **Zhang, L.** and **L. Zhang** (2017). Deep learning-based classification and reconstruction of residential scenes from large-scale point clouds. *IEEE Transactions on Geoscience and Remote Sensing*, **56**(4), 1887–1897.

77. **Zhang, Y.**, **C. Zhang**, **S. Chen**, and **X. Chen** (2021). Automatic reconstruction of building façade model from photogrammetric mesh model. *Remote Sensing*, **13**(19), 3801.

78. **Zhu, J.-Y.**, **R. Zhang**, **D. Pathak**, **T. Darrell**, **A. A. Efros**, **O. Wang**, and **E. Shechtman** (2017). Toward multimodal image-to-image translation. *Advances in neural information processing systems*, **30**.

79. **Zhu, L.**, **S. Shen**, **X. Gao**, and **Z. Hu**, Large scale urban scene modeling from mvs meshes. *In Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.