# Multi-levels of detail terrain construction for navigation

Qiuxian Wei

Student number: 5801737

July 5, 2024

1st supervisor: Stelios Vitalis

2nd supervisor: Ken Arroyo Ohori

co-reader: Martijn Meijers

**Abstract:**
Web mapping is essential in many fields, including navigation, real estate, and tourism. With advances in mapping technology, 3D web mapping has been introduced by multiple geospatial platforms. Compared to 2D web mapping, 3D web mapping provides more details, better visual effects, and is crucial for applications like digital twins. In navigation, terrain is a fundamental element in 3D mapping. However, the vast amount of data from 3D models requires the use of multiple Levels of Detail (LoDs) to reduce data consumption and request time on the client side. Without further processing, different LoD models will have gaps at their boundaries. Such gaps can cause visual inconsistency and artifacts.
Locking boundaries is a commonly used method to solve the gaps in different LoDs problems, that is to maintain the vertices on the boundaries during simplification. While this method causes inefficiency in simplification.
Nanite (Nan) proposed a method using triangle clustering to address issues with Level-of-Detail (LoD) model gaps, inefficiencies in cluster simplifications, and visual inconsistencies caused by locked boundaries. This method can be applied to large-scale models, such as terrain data. However, this method was only applied to local data use cases and not to web mapping. Therefore, this thesis investigates adapting the triangle clustering method for web mapping.

Considering the fact that the Nanite triangle clustering method performs well on multi-LoD models (Nan), we believe this method could also be migrated to the area of large-scale terrain. We proposed a method to construct multi-level LoD terrain for navigation based on Nanite method. In our approach, large-scale terrain data is distributed using web tiles for navigation purposes.

***Keywords***— WebGIS; Web mapping; 3D map; Multi-LoD

**Acknowledgements:**

I want to express my gratitude to those who provided patience and help through my whole study process. Firstly, my supervisor both in TU Delft and TomTom, Stelios Vitalis, inspired me with frightening wisdom (;P), thank him for providing me with such a good opportunity to work with him and also the internship in TomTom, I learned a lot during this experience. Also, I would like to thank my supervisor, Ken Arroyo Ohori, for all his patience and guidance through my whole thesis. Besides, I want to thank Martijn Meijers and H.W. (Herman) de Wolff for their participation in the thesis and their kindness throughout the process.

My special thanks to my friends and family who knew about my hardships last year and helped me through my darkest time, especially my boyfriend who was always there by my side. I could never go through that time without you.

I want to express my deepest gratitude here to my mom, who was always proud of me and inspired me with her bravery and strength. I would not be who I am today without her care and love for 23 years, and the memory will always be my guidance in my life path.

# 1 Introduction

## 1.1 Research motivation

For location services such as navigation, as well as others that include web mapping, the service structure typically relies on a client-server (C/S) architecture. Specifically, unlike desktop applications that store and process data locally on the device (Mazzei and Quaroni, 2022), in web mapping, user requests are sent to the server, which processes these requests and delivers the data back to the user. With the development of web mapping (Veenendaal et al., 2017), multiple platforms now provide data in both 2D and 3D formats. The emergence of Google Earth, Microsoft Virtual Earth (now BING), Cesium, NASA World Wind, TomTom, and various other virtual earth platforms has made 3D maps accessible to everyday users (Veenendaal et al., 2017). Compared to 2D mapping, 3D mapping offers more detail and better visual effects. In the field of navigation, 3D mapping is also crucial for accurately representing complex urban environments.

Terrain is one of the most fundamental elements in web mapping. For navigation, elements attached to it such as buildings and roads also need consideration. There are two main ways to represent the terrain: Grid and Triangulated irregular network (TIN). The grid geometry has difficulty among regions with drastic change, e.g. cliffs, which results in distortion and redundancy in data. As to TIN-based methods, the geometry can theoretically produce a terrain with minimal complexity. Thus for 3D mapping, the TIN method is more suitable for models with accuracy needs.

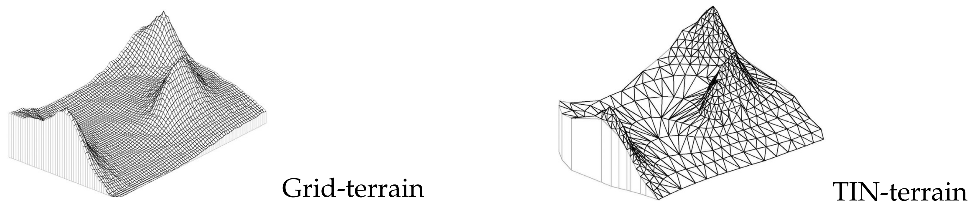


Figure 1: Two types of terrain(Gavran et al., 2017)

However, digital terrain datasets can easily contain several million vertices, while graphics hardware accelerators can only interactively render a fraction of this at 20 frames per second or more (Pajarola et al., 2002). Additionally, maintaining the mesh connectivity, hierarchies, and dependencies is costly (Zheng et al., 2017). To address these challenges, multiple levels of detail (LoD) based mesh simplification methods are used to reduce geometric complexity while preserving visualization quality.

Various methods have been proposed to tackle this problem. Currently, efficient processing and rendering of extensive datasets typically rely on two approaches: adaptive coarse-grained refinement from out-of-core multi-resolution data structures, such as Batched Dynamic Adaptive Meshes (BDAM) (Cignoni et al., 2003a), and in-core rendering from aggressively compressed pyramidal structures, like Geometry Clipmap (Losasso and Hoppe, 2004) (Gobbetti et al., 2006). Some methods utilize regular grid structures to form multi-LoD structures, such as QuadTIN (Pajarola et al., 2002), which generates a TIN through quadtree-based multiresolution triangulation.

However, most of these solutions are not designed for client-server-based architectures. While processing data on the user end is possible, real-time location services for navigation require requesting processed data instead of generating it on the user end. Therefore, most multi-LoD construction methods are not readily applicable to web mapping.

Hence, unlike desktop software solutions, web mapping addresses the bottleneck with multi-zoom level tiles. Web maps are divided into tiles at different zoom levels, each with a pre-constructed mesh of varying Levels of Detail (LoD). Each region is represented by a tile that can be downloaded separately as needed. The web map should seamlessly join different tiles, but gaps can occur between tiles with different LoDs at different zoom levels as shown in Figure 2.

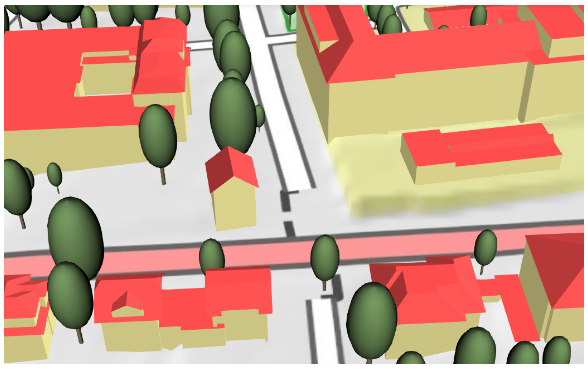


Figure 2: Gaps between different 3D map tiles(swi)

Gaps between different LoD tiles, known as the T-junction problem as shown in Figure 3, occur due to incoherent boundaries during LoD construction. As the model's geometry changes during simplification, points on the boundaries between different LoD tiles can be simplified differently. This can lead to height differences and gaps between tiles when these boundary points are not aligned.

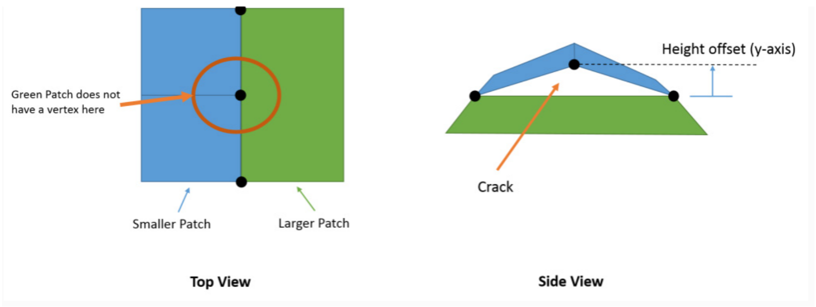


Figure 3: T-junction between different 3D map tiles(Tju)

A common solution to the T-junction problem involves locking the boundaries during simplification to maintain geometry continuity. This means that points on the boundaries between different tiles are "locked' and remain unchanged during simplification, while only points inside the boundaries can be removed. However, in models with multiple LoD levels, the boundaries can become complex due to dense triangle clusters, leading to inefficiencies in simplification because the points on the boundaries remain unchanged. Additionally, the complicated edges can result in artifacts in visualization.

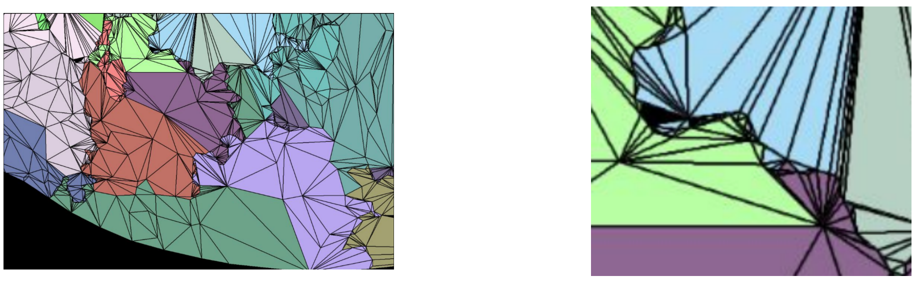


Figure 4: Complex boundaries(Nan)

Some methods provide structural multi-LoD construction to change the locked boundaries for each LoD, say, QuadTIN (Pajarola et al., 2002), which proposed a method based on quadtree structure. By adding extra points in the initial TIN, a quadtree structure is built. For each LoD, the locked boundaries change, and the density around the locked boundaries problem is eased. However, such methods result in a point number increase, compared to the original TIN, the structural could have twice the size of the data to reach the same TIN accuracy.

Nanite, the Unreal Engine 5's virtualized geometry system (Nan) proposed a triangle clustering technique, instead of inserting extra points to the models and constructing data structure, it clusters the triangles of the mesh for each LoD. The triangle cluster boundaries are locked and unlocked in the next triangle clustering process for the next LoD. Consequently, the complexity of the boundaries is solved.

However, this method also needs real-time processing on the device ends, and the data size of the models is costly either stored locally or transmitted through the web for web mapping (multiple LoDs of the same areas will be stored for visualization selection). For web mapping, the dissemination of the 3D models is based on the Tiled web map. A tiled web map is a map displayed in a web browser by seamlessly joining dozens of individually requested image or vector data files (Plex-Earth Support, 2023). The multiple LoD terrain models are assigned to different zoom-level tiles. Zoom level is the scale of the map. Each zoom level contains different LoDs terrain models inside. The zoom n contains $2^n$ tiles.

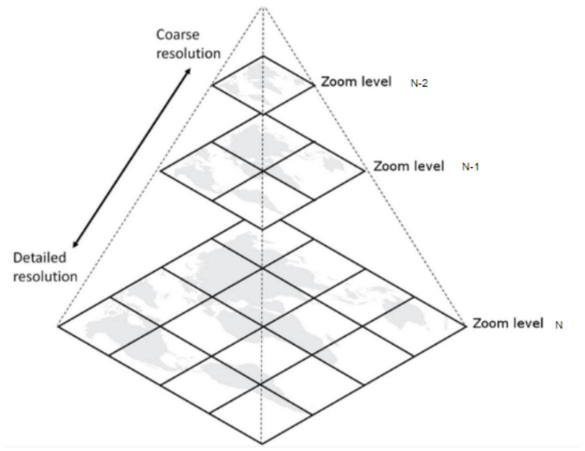


Figure 5: Tiled web map structure(Plex-Earth Support, 2023)

Based on the facts, this thesis investigated the adaption of the Nanite multi-LoDs construction method on the 3D web mapping navigation scenario. A method is proposed to construct visually seamless multi-LoD terrain with tiled web map structure and solve the problem of dense triangle cruft.

## 1.2 Research questions

The main research question of this thesis is: **How to construct seamless large-scale multi-LoD tiled terrain, with consideration of constraints in the process of simplification?**
This thesis focuses on investigating and adapting the Nanite triangle clustering method to construct multi-LoD seamless tiled terrain for web mapping. To achieve this goal, the following sub-questions are gradually studied:

- How to simplify the mesh to generate different LoD models?
  To generate multi-LoD terrain models, the original terrain data needs to be simplified with simplified algorithm. However, the simplified method and termination condition of the simplification needs to be investigated.
  - How to determine the LoDs simplification threshold? Based on the model errors after simplification or geometry features such as vertices/edges/faces number?
  - What kind of simplification methods? Edge-collapse? Greedy insertion? Point set simplification?

- How to partition the mesh into different clusters?
  The terrain model is simplified inside the clusters with the clusters boundaries locked, thus the 3D mesh needs to be partitioned into different clusters. Various aspects need further consideration as follows:
  - How does the number of clusters affect the simplification result?
  - How to partition the mesh so that the locked boundaries have less influence on the simplification efficiency?
- How to preserve boundaries of the clusters during simplification?
  The boundaries of the clusters need to remain unchanged during simplification, otherwise there would be cracks between different LoD models.
  - Set constraints to terrain and simplified them together?
- How to assign the clusters to the tiles?
  After simplifying the clusters, the clusters are assigned to different tiles to generated tiled web map. There are various solutions, such as based on the centroid point of the clusters, as long as the centroid point is inside the tile, the cluster is assigned to that tile, or as long as the cluster is cross the tile boundaries, the cluster is assigned to the crossed tiles.
- How to assess the quality of the simplification methods?
  The quality of the tiles need to be assessed after construction to evaluate the multi-LoD model construction. The following evaluation questions are considered:
  - Size of the mesh geometry (vertices, edges, faces) after simplification?
  - How to calculate the error of the simplified terrain?
  - How to ensure the connection of the different LoDs tiles is visually seamless?

## 1.3 Research scope

The partition of the triangles in the mesh is a graph partition problem, and the partition method has been implemented by METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering (Karypis and Kumar, 1999). The implementation of mesh simplification, which means to simplify the model with simplification algorithms, is focused on two simplification methods, edge collapse and greedy insertion. These two methods are introduced in more detail in Section 3.4. The implementation of these two algorithms is based on existing open library, specifically edge collapse is implemented by The Computational Geometry Algorithms Library (CGAL) (cga, 2021), and greedy insertion is implemented by tin-terrain (HERE Technologies, 2024).

Thus, this thesis focuses on selecting the proper partition options by METIS to experiment with the influence of different partitions on simplification results. Regarding simplification, this thesis investigates suitable algorithms that fit the triangle clustering method schema and implement constraint preservation. The main challenge is adapting the algorithm choices to the web tile schema.

Note that this thesis only investigates the build of seamless multi-LoD terrain, the elements attached to the terrain, such as roads and building footprints, are not within the scope of this thesis.

## 1.4 Research overview

The related work section provides an overview of the former research in the field, it also discusses the techniques and theory that are either used in the thesis or need adaption or improvement for this thesis research goal. Subsequently, the methodology gives more details on the techniques used in the thesis to prepare the test data, construct the multi-LoD structure, partition the clusters, simplify the mesh, and assign the clusters to the tiles. Also, this sector provides assessment methods to evaluate the simplification result and the seamless visualization of the generated tiles connection. Then, the implementation sector explains with more details about how the thesis is carried out. It explains the prototype implementation details with the implementation data format, data structure used and engineering decision. After that, the results and analysis sector gets into details about the experiment data and the evaluation of the result. Afterwards, the conclusion and future work sector sum up the conclusions of the thesis and proposes possible improvements in the future work.

# 2 Related work

## 2.1 Mesh simplification

Garl and Heckbert (1999) proposed efficient algorithms for approximating a height field using a piece-wise linear triangulated surface. It is a variant of greedy insertion. The method is both faster in theory and practice than many other methods: parallel greedy insertion algorithms and the algorithms before.

Garl and Heckbert (1997) proposed the greedy insertion method, which performs coarse-to-fine simplification. It starts with a very basic triangulation, e.g., the four corner points of the original raster to form two triangles as the initial mesh. They found the most effective way is to calculate the absolute height error added to the mesh at each iteration until all the points are within a user-defined tolerance.

Garland and Heckbert (1997) proposed the edge-collapse method, which is a fine-to-coarse simplification. At each iteration, it collapses the edge that would induce the smallest error. The operation collapses an edge by removing it, merging its adjacent vertices into one, and reconnecting the adjacent edges to the merged vertex.

Campos et al. (2020) experimented with three simplification methods: Greedy insertion, Edge-collapse simplification, and Point set simplification. Greedy insertion starts from a very basic triangulation, and the point inducing the largest error is added to the surface at each iteration until all the points are within a user-defined tolerance. Edge-collapse simplification creates an approximation by iteratively applying an edge-collapse operation, which, as the name suggests, merges the endpoints of an edge into a single point. The input of Point set simplification is a point cloud without connectivity, while the mesh will be reconstructed afterward.

These methods were chosen because they were believed to represent different relevant lines of research in state-of-the-art terrain/surface simplification. The result of the simplification showed that the greedy insertion and edge-simplification have better results than the point cloud method, therefore, only edge-collapse and greedy insertion simplification are experimented in the thesis.

ZHANG Na (2022) promoted a 3D Douglas-Peucker terrain simplification algorithm optimized with centroidal Voronoi diagram. The widely-used traditional 3D Douglas-Peucker algorithm simplifies terrain by setting distance threshold parameters. However, it tends to retain features with high variation, such as ridgelines and valleys, while areas with obvious local terrain undulations are not adequately considered in the simplified terrain.

The Centroidal Voronoi diagram, due to its characteristics, causes the seed points of the centroid Voronoi map to converge towards denser areas. The terrain factor can be used as the density function, driving the seed points to areas with large terrain undulations through iteration. The optimized algorithm reduces the simplification error by more than 13.6% at each simplification level.

## 2.2 Multiple LoD and seamless mesh construction

Hoppe (1998) specialized the view-dependent progressive mesh (VDPM) framework to the important case of terrain rendering. However, many approaches, on the contrary, impose a regular subdivision structure to speed up the transition between LODs.

Losasso and Hoppe (2004) proposed the clipmap method, which is based on the hierarchical organization and manipulation of a fixed grid whose center depends on the point of view. Additionally, there are methods using quadtree triangulations or triangle binary trees.

Livny et al. (2009) uses quadtree subdivision, and the terrain is divided into rectangular patches, where each patch is represented by four triangles.

Pajarola et al. (2002) proposed a quadtree-based hierarchical multiresolution triangulation method that takes advantage of the regular grid. Basic quadtree-based triangulation methods are applicable only to regular grid input datasets (Lario et al., 2003). QuadTIN applies the quadtree-based method by inserting extra points into the input data (generally less than 25% of the input data) and builds hierarchies of right triangles (HRT) (Evans et al., 2001). Figure 6 (Pajarola et al., 2002) shows that to achieve the same error, the terrain constructed by QuadTIN requires around twice the size of the points compared to terrain without structured partition.

Based on QuadTIN, Batched Dynamic Adaptive Meshes (BDAM) (Cignoni et al., 2003a) proposed a way to only lock the points on the longest boundary in right triangles. Thus, the locked boundaries can be changed during simplification and consequently reduce the complicated boundary problem. The figure below shows an example of BDAM. Each triangle represents a terrain patch composed of many triangles. Colors correspond to different errors, and the blending of the color inside each triangle corresponds to the smooth error variation inside each patch.

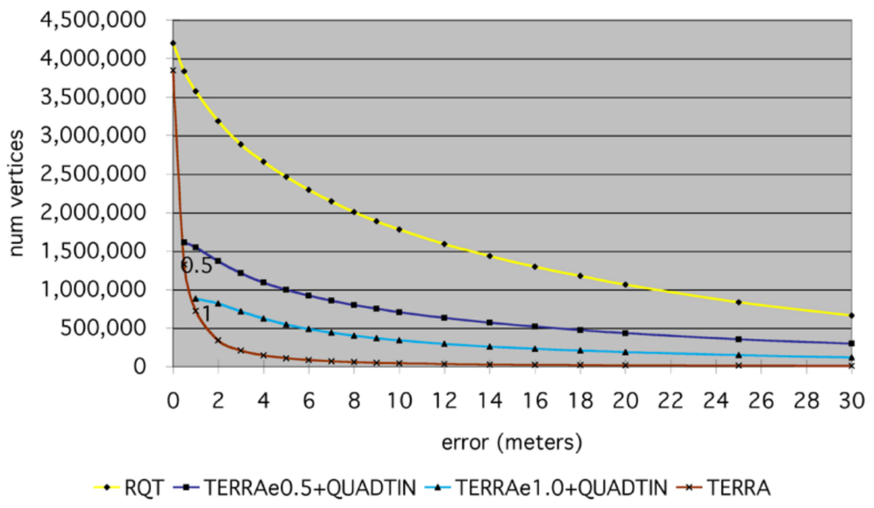


Figure 6: Points number for different TIN terrain construction (Pajarola et al., 2002)

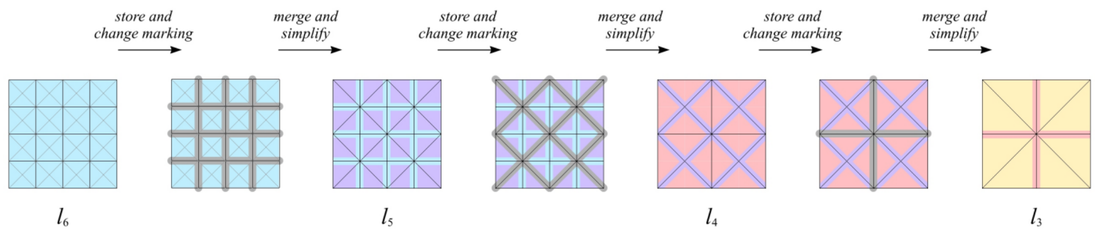


Figure 7: BDAM(Cignoni et al., 2003a)

Cignoni et al. (2003b) presented a planet-scale triangle binary trees subdivision based on Batched Dynamic Adaptive Meshes (BDAM).

Zheng et al. (2017) also proposed a structured hierarchical multiresolution triangulation. Similarly, it added extra corner and edge points to the tiles. The amount of extra data could be high if the mesh data was detailed and complicated.

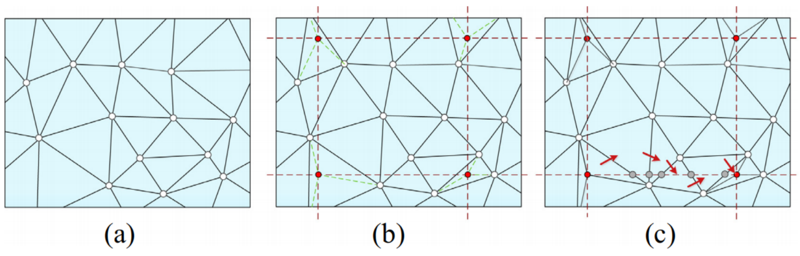


Figure 8: Tiled structure partition (Zheng et al., 2017): *(a) Original triangulation. (b) Insert tile corner points. (c) Quickly walk from one corner point to another corner point through the Delaunay edges and calculate the other border points.*

To address the inefficiency of structured hierarchical multiresolution triangulation, Nan proposed a triangle clustering method to partition the triangles into clusters without adding extra points. For better simplification results in partitioning the triangles, METIS (Karypis and Kumar, 1999) is used. METIS is a set of serial programs for partitioning graphs, meshes, etc.

Christen and Nebiker (2011) promoted a TIN-based visualization of a multi-resolution model from large-scale LIDAR elevation points. However, it focuses on real-time rendering and neglects the strict quality control of terrain data. Additionally, the solution to the cracks between terrains is not considered.

However, the main drawback of most of the approaches presented above for computing LoDs dynamically is that they require direct access to the whole multi-resolution structure to render the terrain in real-time (Campos et al., 2020). Keeping the entire structure in memory is a limitation for low-end hardware or high-resolution terrain. Thus, pyramid-structured multi-resolution methods are promoted. Campos et al. (2020) proposed pyramid tiling with multi-resolution terrain, in which terrain is separated by tiles and generated one by one. However, continuity between tiles is a problem in such a solution. The edge corners and points on edges are preserved for the connection among tiles. Zheng et al. (2017) proposed multi-resolution TIN construction with control over data quality standards to formalize each layer's level of detail generation. It addresses cracks by edge tessellation, adding boundary points from connected high-level of detail tiles to the boundary of low-level of detail tiles.

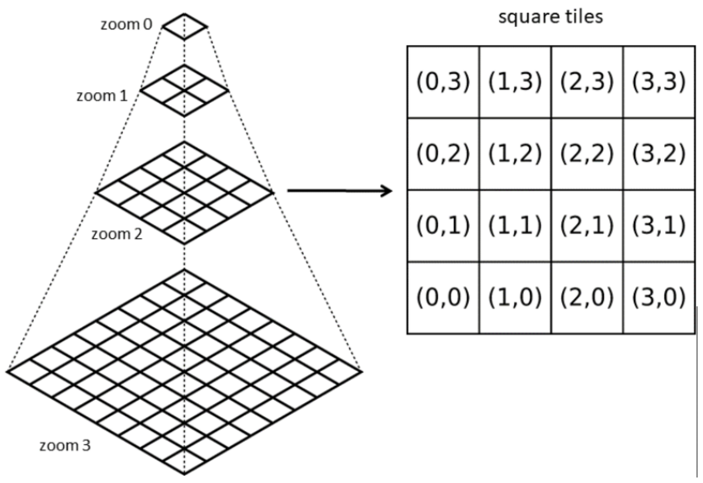


Figure 9: Multi-resolution pyramid (Campos et al., 2020)

Chen and Zhou (2013) Proposed a formula to establish data structure to match the demand from an application at a coarser scale. Based on cartographic principles, the relationship between the grid resolution $D_r^{'}$(local reference system, unit in meters) and the $E_{max}$ can be calculated :

$$\frac{E_{max}}{D_r^{'}} = \frac{4}{5}$$

The simplification will stop and generate TIN as one of the desired scales once the thresholds are met: the maximum elevation error $E_{max}$ and the root-mean-square error (RMSE). Chen and Zhou (2013) found that the two variables of RMSE and $E_{max}$ are highly correlated. Thus, only $E_{max}$ is used in this thesis to evaluate the error of terrain. However, it is a formula to generate multi-zoom-level tiles at a coarser scale, thus not suitable in navigation where map accuracy holds significant importance. According to the 2019 high precision maps industry research report by Tencent (Community, 2019), the industry aims to achieve 20 centimeters of $E_{max}$ with a good signal. Still, there lacks a consistent official standard, and specific accuracy metrics are typically determined by individual manufacturers based on the requirements of autonomous driving. Considering these facts, this study refrains from delving into the precise accuracy requirements for navigation.

The constraint of drainage features was conducted in Zheng et al. (2017) for the simplification of terrain that influences drainage features. They generalized the drainage features to incorporate them into each layer of terrain. For features on terrain, such as roads, similar methods should be carried out to maintain the topology and details.

Vaaraniemi et al. (2011) investigated techniques for integrating roads into high-resolution terrain. They adopted two methods for rendering cartographic roads: rounded caps were computed and added at the ends of road segments.

## 2.3 Summary

The former multi-LoD 3D model construction is usually based on keeping boundaries unchanged during simplification to avoid cracks between different LoDs. However, the density accumulation on the boundaries as simplification is carried out multiple times. The former research methods to change the locked boundaries during simplification usually need structural data while this leads to extra points insertion in the 3D model. Nanite proposed a triangle clustering method to cluster the triangles into clusters and simplify inside the clusters, thus no extra inserted points are needed and the locked boundaries can be changed during simplification. However, this method is based on local data storage and is not suitable for web mapping schema. In that case, this thesis proposes a solution to adapt triangle clustering techniques to the tiled web map.

# 3 Methodology

## 3.1 Overview

In this chapter, the triangle clustering based multi-LoD terrain construction is proposed and introduced in details. The method overview is as shown in Figure 17.

Firstly, the Digital Terrain Model (DTM) data in GeoTIFF format is transformed into Object File Format (OFF). The coverage of the data is more than 3 kilometers * 3 kilometers area to ensure the experiment area is large enough to test the connection between different LoD models.

Secondly, the terrain model is **partitioned** into different clusters with METIS Karypis and Kumar (1999), METIS is a set of serial programs for partitioning graphs, meshes, etc. Due to the possible density accumulation around boundaries area during multi-LoD models construction, the mesh is split into **clusters** and simplified inside the clusters with the clusters boundaries unchanged. The principle of the partition is to ensure the **locked boundaries** as fewer as possible, as the more more unlocked boundaries means more simplified options, and thus better simplification efficiency. The unchanged boundaries during simplification are referred as **reserved constraint**.

Thirdly, after simplification, the clusters are assigned to corresponding **tiles**. The solution to assign the clusters are based either on the centroids of the clusters or whether the clusters touch the tile. The tiles are grids with geographic content. And in this thesis, we use tiled web map structure. The different LoD clusters are assigned to different zoom level tiles. The **zoom level** is the pyramid structure with multiple floors of different LoD tiles. The lower zoom contains more details. The tiles are indexed by zoom level/row/column. For example, the tile in zoom 2, row 0, column 0 is Tile 2/0/0. The content of the tile is OFF file 3D terrain model.

Finally, the construction of the multi-LoD terrain is evaluated. The evaluation is based on the size of the generated data, the simplification quality, and whether the visual connection between different zoom level tiles is seamless.

### 3.1.1 Implementation Steps

The representation of the 3D terrain model is TIN mesh in the thesis. To generate seamless multi-LoD terrain, the TIN meshes are first partitioned into triangle clusters. Within each cluster, simplification is performed to achieve the desired LoD. Subsequently, the multi-LoD clusters are assigned to their respective tiles. This process involves the following steps:

- Pre-processing the data and constructing a multi-LoD hierarchical structure.
- Partitioning TIN into different clusters.
- Simplifying the TIN inside the clusters, locking the cluster boundaries, and preserving related features on the terrain.
- Assigning clusters to desired tiles to generate tiles.

## 3.2 Data Pre-processing

### 3.2.1 Generating TIN Data

Tin-terrain (HERE Technologies, 2024) is used to derive a TIN model from DEM data. It utilizes height maps in GeoTIFF format and uses the Terra method for greedy insertion simplification (Garland, Accessed: 2024). Meanwhile, when dealing with large-scale terrain data, parallel processing becomes necessary (Campos et al., 2020). In such cases, instead of generating a single large-scale, connected terrain model, the input DEM GeoTIFF models are tiled to create different TIN model tiles. These generated TIN tiles serve as the initial input data for the subsequent steps involving simplification and generating LoDs. Figure 10 shows an example of such transformation.

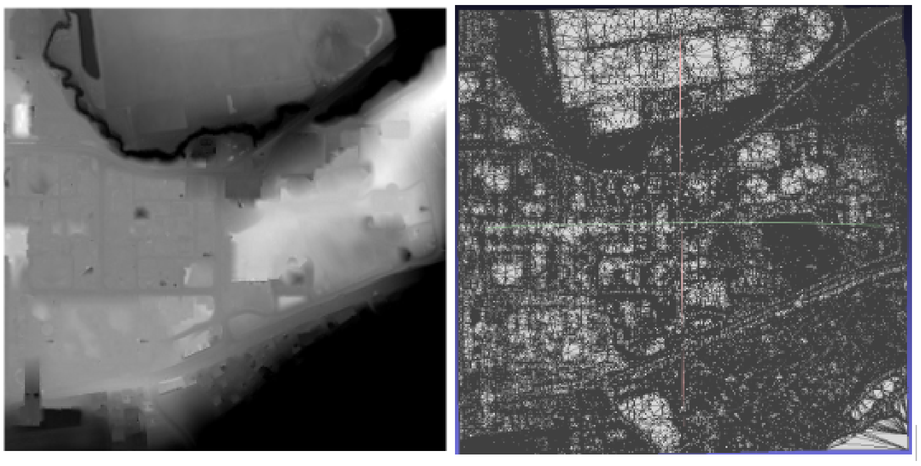


Figure 10: Transform DEM models to TIN models

### 3.2.2 Construction of Multi-zoom-level Tiles Hierarchy

As mentioned in the related work, due to the lack of consistent official standards for 3D navigation maps, this study refrains from delving into the precise accuracy requirements for navigation when constructing a hierarchy of multi-zoom-level tiles. Instead, we focus on implementing the triangle clustering method in web mapping and achieving seamless connections between tiles at different zoom levels. The accuracy of the zoom levels can be adjusted accordingly to cater to different needs.

To test the connection between tiles at different zoom levels, we establish a simple hierarchy with four zoom levels. Since the middle zoom level has both finer and coarser neighboring zoom levels, we can evaluate the connections by examining how the middle zoom levels interacts with the other two zoom levels and how they connect with each other. This setup allows us to effectively test the seamless connection between different zoom levels.

To determine the number of tiles at different zoom levels, we follow the web-tiled map schema. According to this schema, the tile size at coarser zoom levels is twice that of the neighboring finer zoom level. Consequently, the total number of tiles covering the same area is halved. Each zoom level's tile count is therefore set to the power of two. The zoom levels are labeled zoom 3, zoom 2, and zoom 1, and zoom 0. Zoom 3 representing the lowest zoom level with the finest details, and zoom 0 representing the top zoom level with the coarsest details. The structure is shown in Figure 11.

It is important to note that the tile count is established only for implementation and result analysis purposes. The minimum number of tiles required for testing connections between different zoom level tiles is set as follows. For zoom 0, we generate 1 * 1 tile. For zoom 1, one generated TIN tile covers a 2 * 2 area, equivalent to 4 tiles, facilitating connection testing between zoom 1 and zoom 2, as well as within zoom 1 itself. Similarly, for zoom 2, one TIN covers a 4 * 4 area, totaling 16 tiles, and for zoom 3, it covers an 8 * 8 area, totaling 64 tiles.

For naming different zoom levels: The zoom 3 tiles contain the most detailed TIN models, thus referred to as LoD 3; The next zoom level, zoom 2, contains simplified versions of the LoD 3 models, referred to as LoD 2; Similarly, zoom 0 tiles contain simplified versions of the LoD 1 models, referred to as LoD 0.

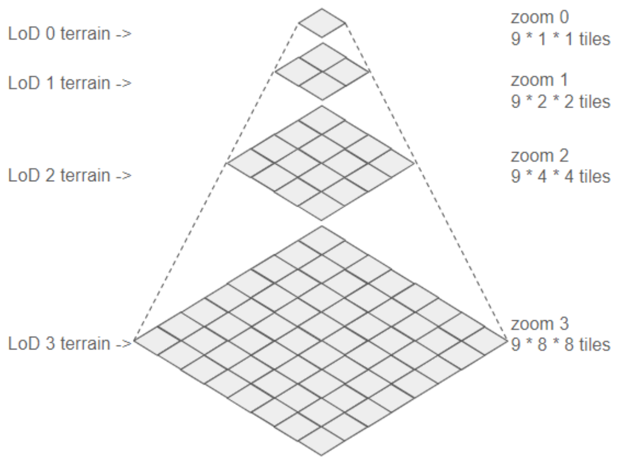


Figure 11: Multi-zoom-level tiles hierarchy

## 3.3 Partition TIN into different clusters

Previous research, such as that by Cignoni et al. (2003a) and Pajarola et al. (2002), utilized a structured hierarchy to partition the mesh. However, this approach requires additional points to be inserted to form structures, which can lead to inefficiencies in simplification since locked boundaries cannot be chosen strategically. Consequently, more points are needed to achieve specific mesh error thresholds.

To address these issues, the TIN (Triangulated Irregular Network) is partitioned without a structured hierarchy. Instead, the TIN mesh partitioning is treated as a graph partition problem, where the objective is to reduce a graph by dividing its nodes into smaller groups. In this context, each mesh face (triangle) is considered a node in the graph, representing the entire TIN mesh, and the connections between faces are treated as graph edges, as illustrated in Figure 13.

For better simplification results, minimizing the number of locked boundaries during simplification is crucial, as these boundaries remain unchanged and can cause simplification inefficiencies. Therefore, triangles that share the most boundaries are clustered together. As shown in Figure 12, the possible cluster boundary is represented by the blue line in the illustrated area.

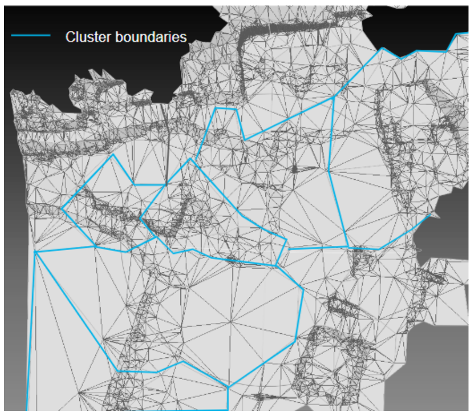


Figure 12: Cluster partition boundaries

To reduce the number of locked boundaries, we seek for a better partition method with minimized cuts between clusters. In detail, partitioning the TIN involves cutting the edges between the nodes, with the boundaries of the clusters being defined by these cuts. By minimizing the edge cuts, the faces that share common edges are more likely to be grouped into the same cluster. This results in more compact clusters with fewer locked boundaries, enhancing the efficiency of the simplification process.

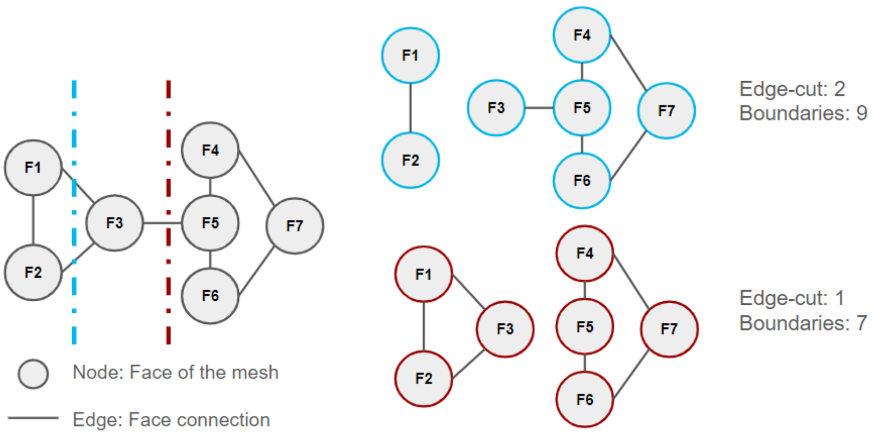


Figure 13: Edge-cut example in mesh partition

|  | Original |  | Greedy |  | Collapse |  |
|---|---|---|---|---|---|---|
|  | Vertices | Faces | Vertices | Faces | Vertices | Faces |
| Model1 | 63594 | 126976 | 24765 | 49406 | 24837 | 49483 |
| Model2 | 51016 | 101845 | 18923 | 37741 | 18905 | 37654 |
| Model3 | 66603 | 132972 | 27245 | 54351 | 27345 | 54479 |
| Model4 | 39226 | 78254 | 16121 | 32135 | 16113 | 32052 |
| Model5 | 60033 | 119710 | 32864 | 65505 | 32904 | 65485 |

Table 1: Vertices and faces number of the models

### 3.4 Simplify the mesh and preserve related features on the terrain

After partitioning the TIN mesh into different clusters, the clusters' boundaries are locked, and only geometry inside the clusters is simplified. For the next coarser LoD generation, the boundaries of the cluster are unlocked and the simplified TIN mesh is partitioned again, thus the locked boundaries change during the multi-LoD construction.

Regarding simplification methods, two popular techniques are experimented with:

- Greedy insertion: Performing coarse-to-fine simplification which is based on Garl and Heckbert (1997) and open-source implementation of greedy insertion TERRA (Garland, Accessed: 2024). During simplification, point with the biggest height error is inserted in every iteration. The stop condition is meeting the height error threshold.
- Edge-collapse simplification: This method is based on Garland and Heckbert (1997). During simplification, the edge with the smallest remove cost is simplified. For stop condition, we follow the implementation by cga (2021), which provides an edge-collapse algorithm in which the stop condition is determined by the ratio of the remaining edge numbers after simplification (cga).

The result of the models vertices, and faces numbers of original data (Original), after greedy insertion simplification (Greedy), and after edge collapse simplification (Collapse) is shown in Table 1.

The input data consists of four models with slight height differences (around 20 meters) and one model with a drastic gradient change on the ridge. Since the termination for the greedy insertion is conditioned on maximum height error threshold, while for edge-collapse it is the number of edges after simplification, the models are first simplified using greedy insertion with an error threshold of 1.0. Then, based on the number of edges after greedy insertion simplification, the original models are simplified using edge-collapse until they reach the same edge number. This ensures that the models have a similar number of vertices and faces after simplification. The simplification results are evaluated by calculating the Root Mean Square Error (RMSE) of the height differences between the original models and the simplified ones.

To calculate the height in each area, the TIN models are rasterized to a grid data with a resolution of 0.5 meters $\times$ 0.5 meters using interpolation, enabling the calculation of height per grid. The TIN models are interpolated using the Python library Scipy (Virtanen et al., 2020). The working process is shown in Figure 14.

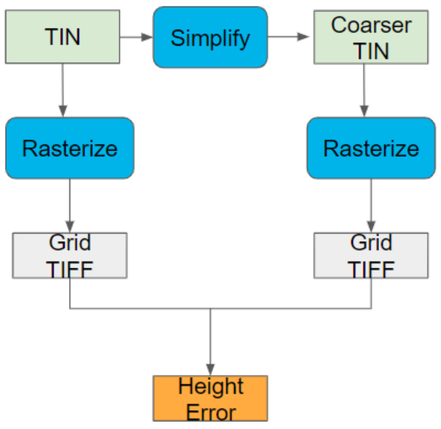


Figure 14: Height error calculation process

Based on the results shown in Figure 15, edge-collapse tends to outperform greedy insertion when the number of geometric elements, i.e., vertices and faces, is similar. This finding aligns with previous research by Campos et al. (2020), which indicates that greedy insertion is more prone to generating additional triangles in high-gradient areas. Consequently, with a similar number of triangles, greedy insertion is more likely to result in higher errors.

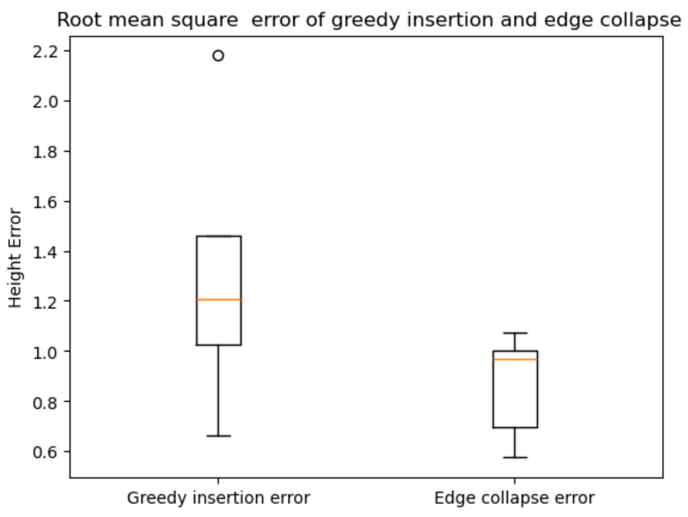


Figure 15: Root square error of greedy insertion and edge collapse

Additionally, edge-collapse better adheres to the locked boundary and related feature preservation because edges can be designated as constrained and not removed during simplification. Therefore, the edge-collapse method is used in this thesis.

The cluster boundaries that locked and need preservation during simplification are set as constraints. These constrained edges are considered not removable during edge-collapse simplification. As shown in Figure 16, both the boundaries of the clusters and the preserved constraints remain unchanged during the simplification process.

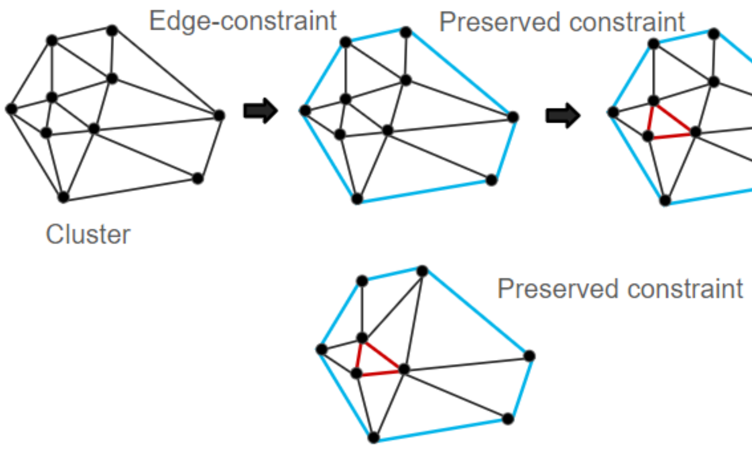


Figure 16: Simplification with boundaries and preserved constraints

The simplification algorithm is described in the following pseudo-code 1:

**Algorithm 1** Edge-collapse with edge constraint

```
Input: Triangle_mesh
Output: Simplified_Triangle_mesh
  while Edge in Triangle_mesh $\neq 0$ do
      if Edge is boundary then
          Edge_constrain_map $\leftarrow$ Edge
      else if N is constrained element then
          Edge_constrain_map $\leftarrow$ Edge
      else
          continue
      end if
  end while
  while Edge in Triangle_mesh has smallest collapse error do
      if Edge in Edge_constrain_map then
          continue
      else if  then
          Collapse Edge
      end if
  end while
```

## 3.5 Assign clusters to tiles and generate tiles

The initial input data is LoD 2 TIN terrain, which is partitioned into different triangle clusters using edge-cut minimization. These clusters are then assigned to corresponding tiles as Zoom 2 tiles based on the centroid point of the clusters. If the centroid point is inside a tile, the cluster is assigned to that tile. The partitioned cluster size is set to be smaller than the size of the tile, if possible, to ensure that each tile contains at least one cluster. To achieve this, the TIN is partitioned into clusters that are four times the number of tiles.

In cases where areas have very coarse triangles (flat areas), leading to unequally large clusters, the tiles are inspected after assigning the clusters. If a tile is found to be empty, the neighboring tiles are partitioned again to ensure each tile contains at least one cluster.

After assigning LoD 2 clusters to corresponding tiles to generate Zoom 2 tiles, edge-collapse simplification is conducted within the LoD 2 triangle clusters to generate the next zoom level tiles. This process produces LoD 1 clusters with the same boundaries as the LoD 2 clusters. The simplified clusters are then merged into an LoD 1 TIN, which is partitioned again to update the cluster boundaries.

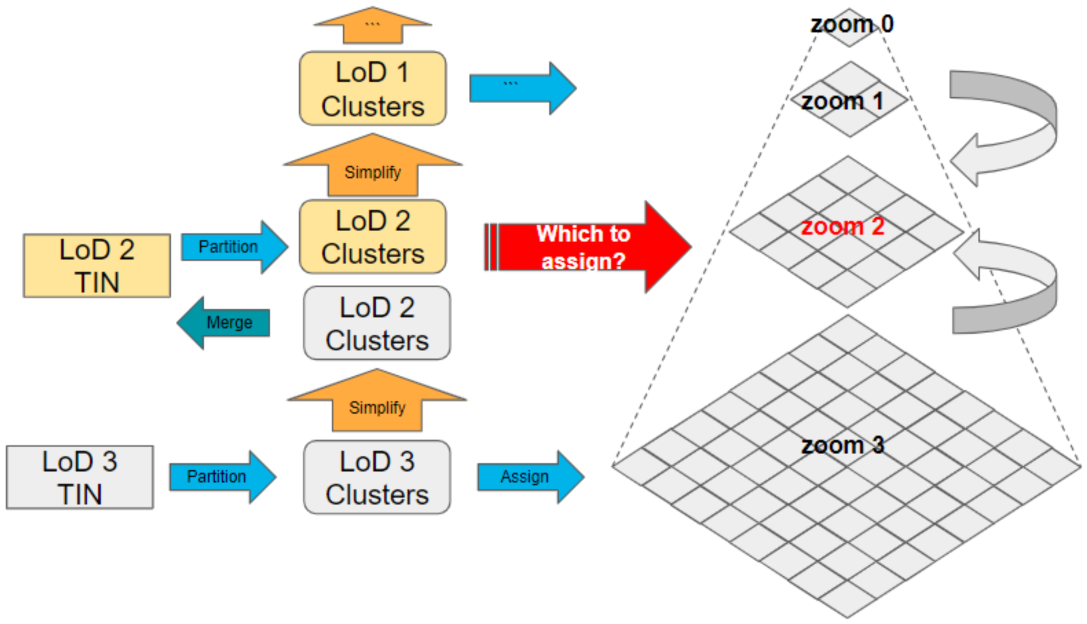


Figure 17: Working process

For tiles with neighboring zoom level tiles (i.e., a finer and coarser LoD tile), such as LoD 1 TIN in Zoom 1, there are two ways to cluster the triangles, as is shown in Figure 17). One method maintains the same boundaries as the LoD 2 clusters in Zoom 2, and the other aligns with the LoD 0 clusters in Zoom 0. The former ensures that the boundaries of Zoom 2 tiles are completely aligned, while the latter aligns with Zoom 0. This might bring overlaps and gaps between LoD clusters.

Specifically, a completely aligned boundary means that each vertex on the boundary of one cluster has a corresponding vertex on the boundary of the other cluster at the same location, allowing the two different LoD clusters to connect seamlessly without overlaps or gaps. Conversely, if clusters aligned with LoD 0 connect with Zoom 2 tiles, overlaps and gaps may occur due to the mismatched boundaries, as illustrated in Figure 20.

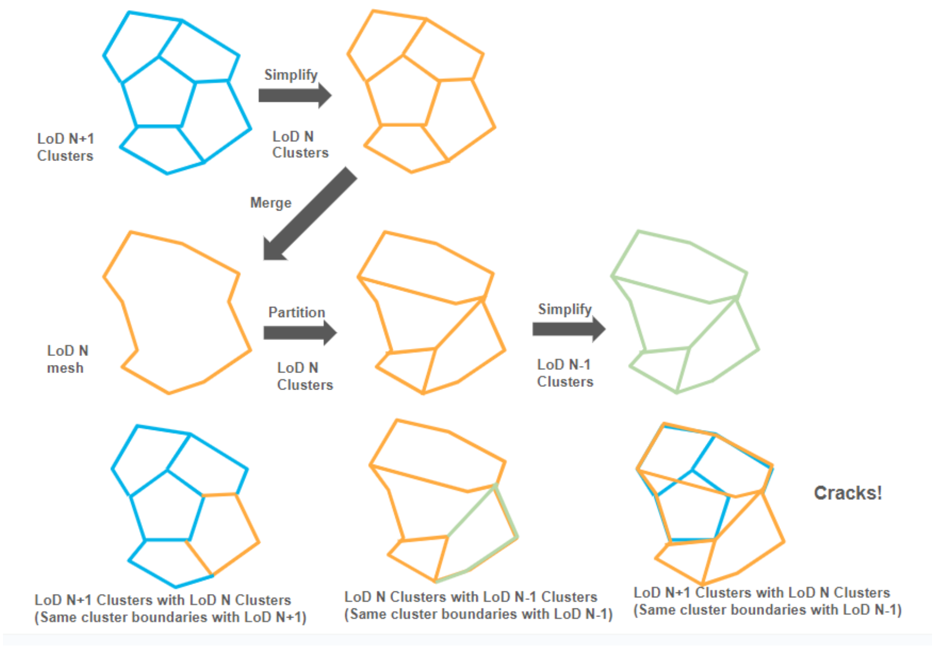

LoD N+1 Clusters
Simplify
LoD N Clusters
Merge
LoD N mesh
Partition
LoD N Clusters
Simplify
LoD N-1 Clusters
Cracks!
LoD N+1 Clusters with LoD N Clusters (Same cluster boundaries with LoD N+1)
LoD N Clusters with LoD N-1 Clusters (Same cluster boundaries with LoD N-1)
LoD N+1 Clusters with LoD N Clusters (Same cluster boundaries with LoD N-1)

Figure 18: Different clusters

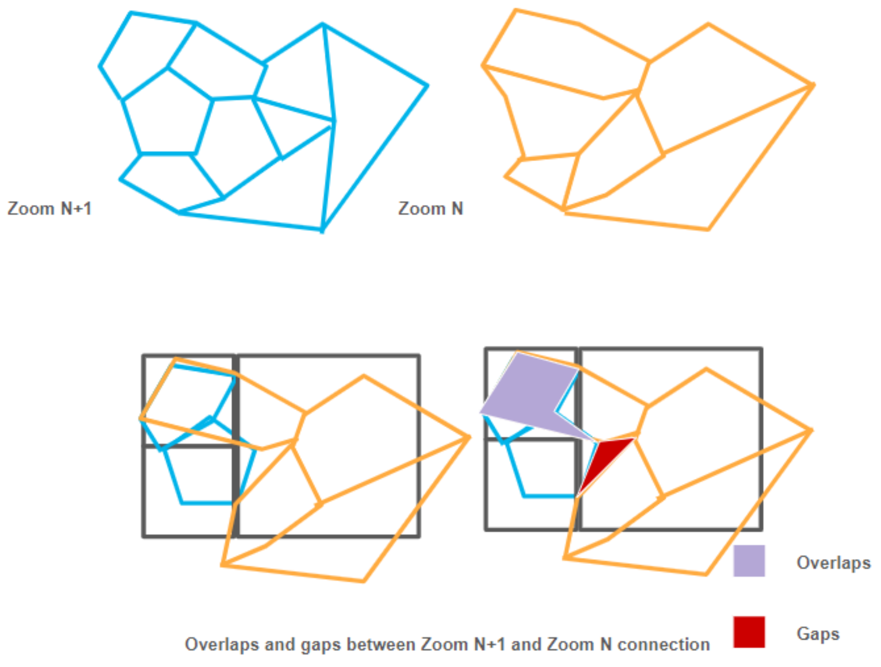


Figure 19: Overlaps and gaps between different zoom tiles

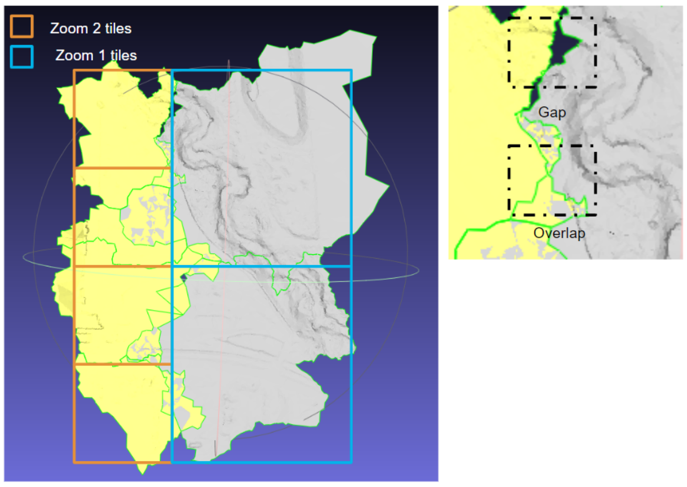


Figure 20: Examples of incoherent boundaries

There are several methods to address the above problem:

- **Double Zoom Tiles:** Store two different cluster boundaries at the same zoom level. Each boundary is assigned to separate zoom tiles.
- **Duplicate Clusters:** Store only one cluster boundary per zoom tile. If a cluster spans multiple tiles, include it in both tiles.
- **Boundaries Merge:** Store both cluster boundaries in the same tile.

More details of each of the method will be described in the following sections.

### 3.5.1 Double Zoom Tiles

Each LoD TIN that has both finer and coarser LoDs, for example, LoD 1, includes clusters that share the same boundaries with both LoD 2 and LoD 0. In this case, the clusters that share the same boundaries with LoD 2 are assigned to corresponding tiles to generate Zoom 1 tiles, referred to as Zoom 1-2 tiles. Similarly, the clusters with boundaries matching LoD 0 are assigned to corresponding tiles to generate another set of Zoom 1 tiles, referred to as Zoom 1-0 tiles.

When Zoom 1 tiles need to connect to both Zoom 0 and Zoom 2 tiles, the appropriate Zoom 1 tiles are retrieved. To connect to Zoom 2 tiles, Zoom 1-2 tiles are used, and to connect to Zoom 0 tiles, Zoom 1-0 tiles are used. As shown in Figure 21 and Figure 22, this approach ensures that tiles from different zoom levels align properly with each other.

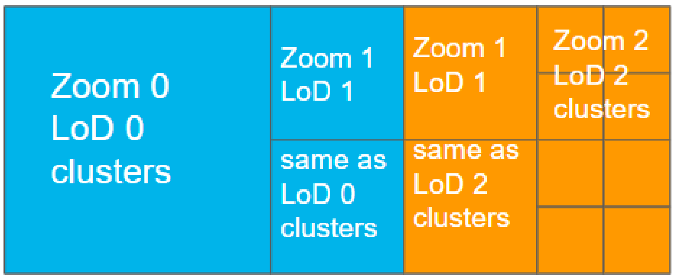
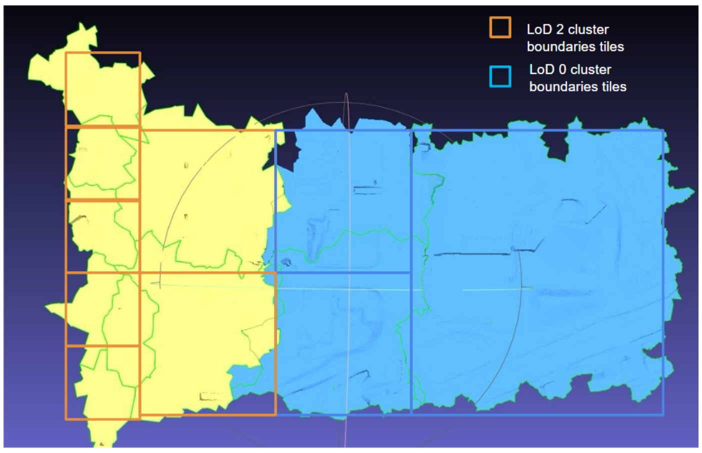


Figure 21: Different zoom tiles connection



LoD 2 cluster boundaries tiles
LoD 0 cluster boundaries tiles

Figure 22: Different zoom tiles connection example

However, this method is costly for web mapping. Similarly in navigation, If the user navigates back and forth within the same area, the same tiles will be retrieved from the server twice. For example, in one location, the user retrieves one Zoom 2 tile, one Zoom 1-2 tile, one Zoom 1-0 tile, and one Zoom 0 tile. When the user moves back, the same set of tiles—one new Zoom 2 tile, one Zoom 1-2 tile, one Zoom 1-0 tile, and one Zoom 0 tile—is retrieved again. Although the two Zoom 1 tiles contain the same LoD models, they are treated as different tiles.

If there were only one set of Zoom 1 tiles for each LoD, each Zoom 1 tile would only need to be retrieved once, avoiding redundant data retrieval. Due to this inefficiency, this solution is not suitable for web mapping purposes and is therefore not used in this thesis.

### 3.5.2 Duplicate clusters

To address the web mapping and navigation needs that mentioned above, this method uses a single tiling approach for each zoom level. The TIN is partitioned for each LoD, and the partitioned clusters are assigned to corresponding tiles. For instance, the input LoD 2 TIN is partitioned, and the LoD 2 clusters are assigned to Zoom 2 tiles. The clusters are then simplified and merged together to form the LoD 1 TIN. The LoD 1 TIN is partitioned again and assigned to Zoom 1 tiles.

Although the cluster boundaries inside different zoom tiles are not aligned, as shown in Figure 20, the overlaps are imperceptible in web map visualization. The gaps are caused by triangles in an area being partitioned into different clusters and assigned to neighboring tiles. To address these gaps, clusters are not assigned to tiles based on their centroid points. Instead, clusters are assigned to tiles if they cross the tiles' boundaries. This results in clusters in each Zoom tile being larger than the tile size, as shown in Figure 25.

With duplicate clusters, there are overlaps between different zoom tiles but no gaps. The overlaps do not cause cracks because the points on coarser tile boundaries have corresponding points in neighboring finer tiles, as they are simplified from the finer tiles. Similarly, the points on finer tile boundaries are locked during simplification, ensuring they have corresponding points in neighboring coarser tiles. Consequently, the T-junction problem does not exist anymore, and the tiles can still appear seamless in web map visualization with proper rendering.

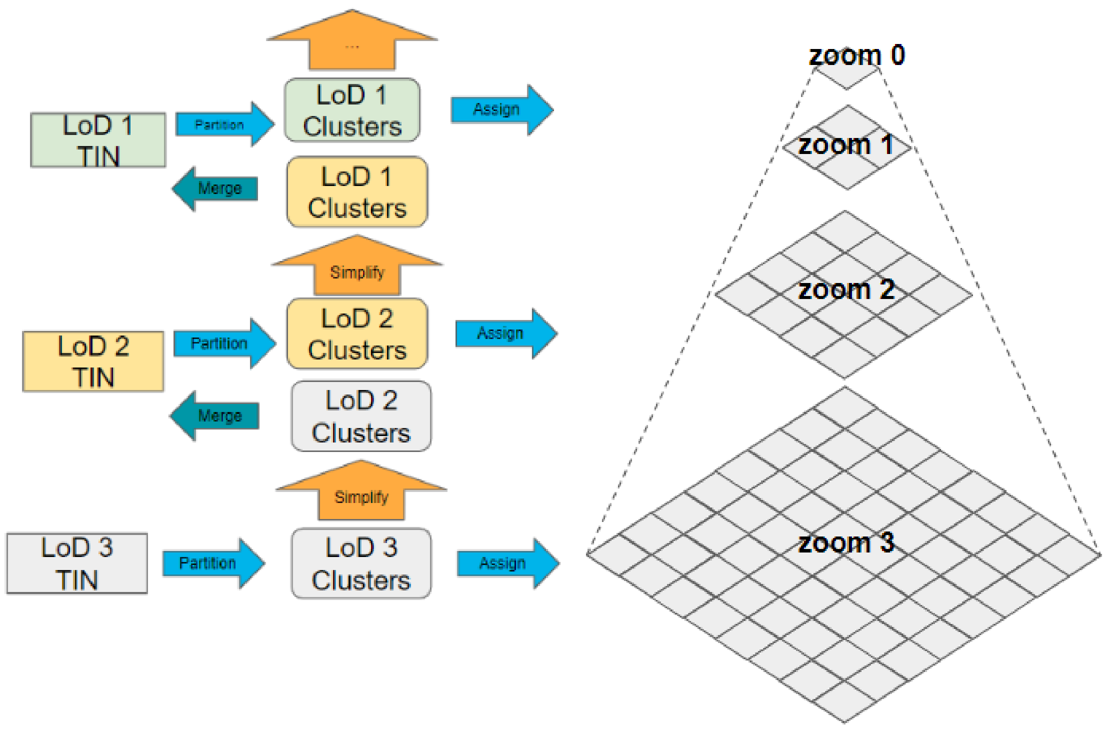


Figure 23: Duplicate clusters working process

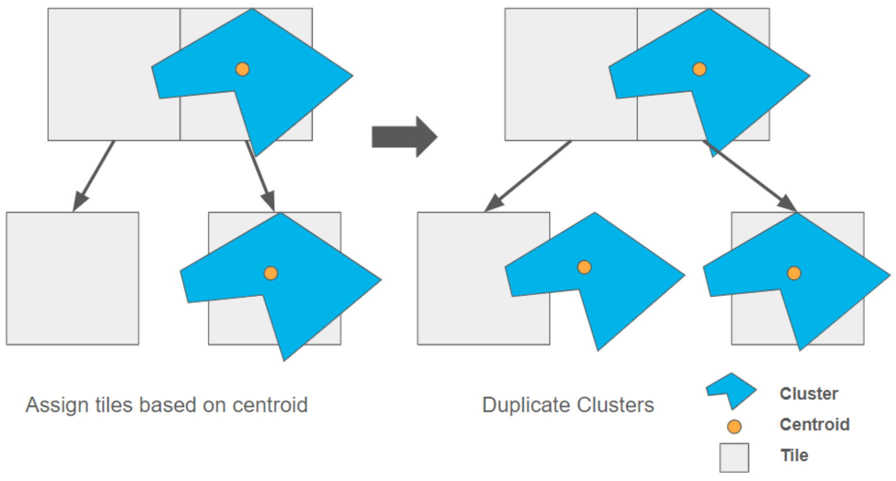


Figure 24: Duplicate clusters assigned to tiles

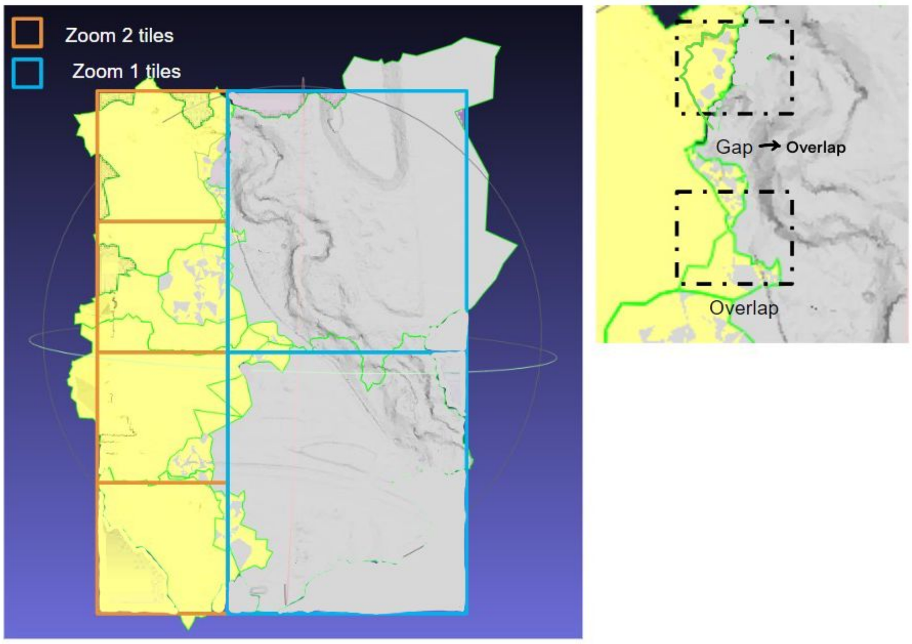


Figure 25: Duplicate clusters example

However, the size of the tiles with duplicate clusters needs to be considered. The redundancy of the data caused by these duplicate clusters is evaluated in Section 3.5.4.

### 3.5.3 Boundaries Merge

The idea behind this method is that for multi-LoD models, if the LoD 2 model is a pentagon and the LoD 0 model is a hexagon, then the LoD 1 model should incorporate the boundaries of both the pentagon and hexagon to connect seamlessly with both, as shown in Figure 26.

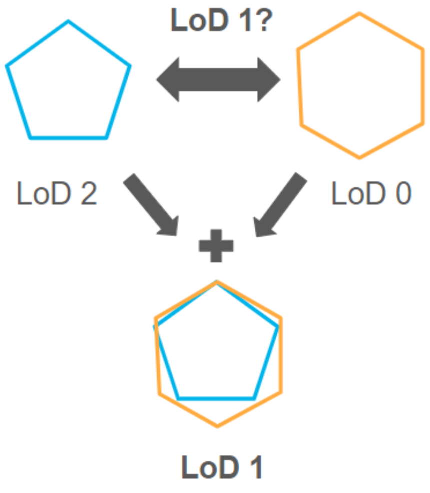


Figure 26: Pentagon and hexagon connection

Unlike the double zoom tiles method, which generates two tile sets for one zoom level, or the duplicate clusters method, which uses only one partitioned cluster for each zoom level, the boundary merging approach combines two different clusters within one tile. This method preserves both sets of cluster boundaries as the tile content.

For example, as shown in Figure Figure 27, in Zoom 1, clusters that share the same boundaries with Zoom 2 clusters (referred to as 1-2 clusters) and clusters that share the same boundaries with Zoom 0 clusters (referred to as 0-1 clusters) are assigned to the corresponding Zoom 1 tiles. These two clusters with different boundaries are merged to form Zoom 1 tiles. This merging process allows Zoom 1 tiles to connect seamlessly with both Zoom 2 and Zoom 0 tiles, with overlaps but no gaps. As mentioned earlier, these overlaps do not affect the seamless connection between different zoom level tiles in web map visualization.

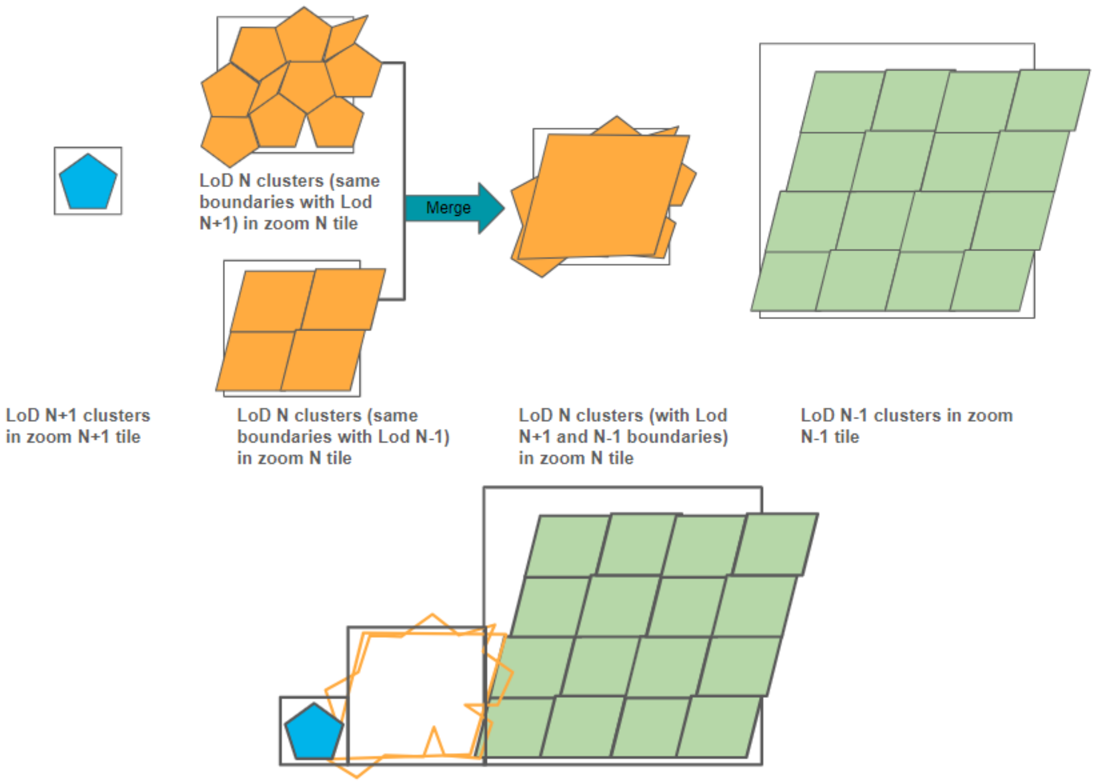

LoD N clusters (same boundaries with Lod N+1) in zoom N tile
Merge
LoD N+1 clusters in zoom N+1 tile
LoD N clusters (same boundaries with Lod N-1) in zoom N tile
LoD N clusters (with Lod N+1 and N-1 boundaries) in zoom N tile
LoD N-1 clusters in zoom N-1 tile

Figure 27: Merge two boundaries

The working process is as follows:

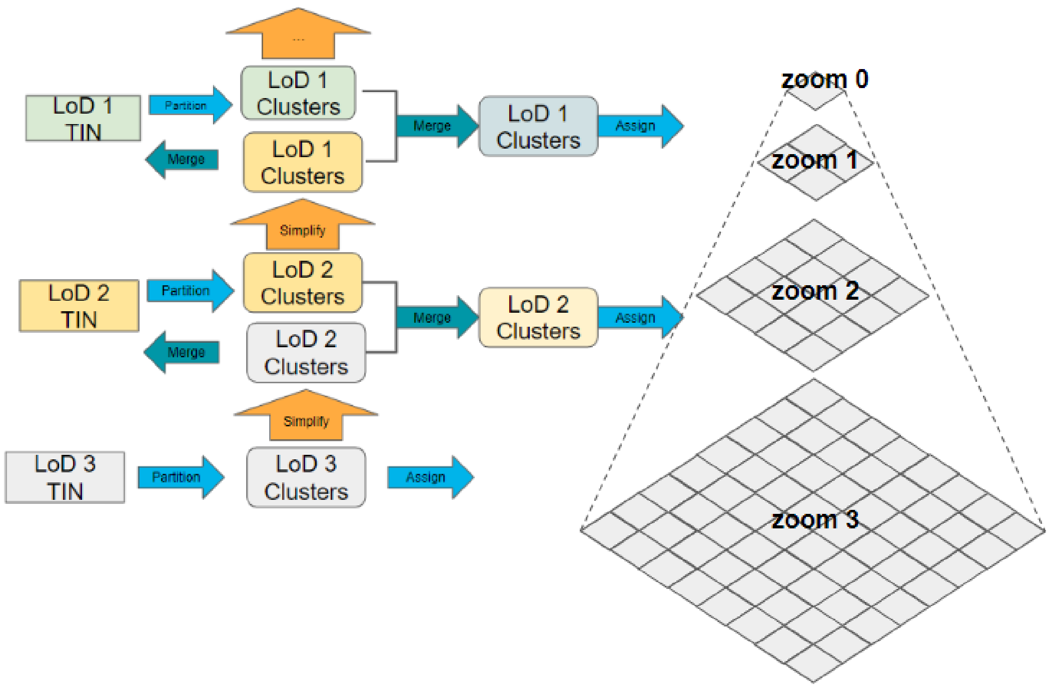


Figure 28: Merge two boundaries working process

Similar to the duplicate clusters method, the boundaries merge method also introduces redundancy of data. To better evaluate these two methods, we calculate the data redundancy resulted from each method, for which details will be discussed in the following section.

### 3.5.4 Evaluation of duplicate clusters and boundaries merge

**Redundancy data size comparison:**
To evaluate the two methods, we compare the sizes of the generated tiles produced by the duplicate clusters method and the boundaries merge method. The experiment utilizes datasets mentioned in Section 5.1. Specifically, 9 TIN models are tested, each containing 16 zoom 1 tiles. The partition number is set to 128 clusters per TIN model, ideally, each zoom 1 tile has 8 clusters.

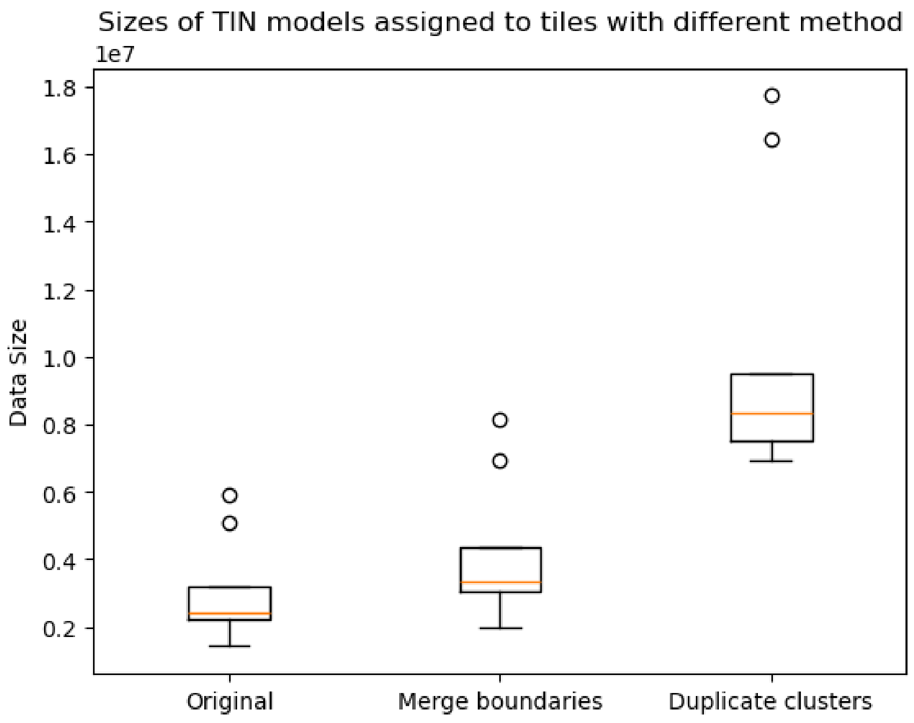


Figure 29: Sizes of TIN models assigned to tiles with different methods

The results in Figure 31 indicate that the merge boundaries method has significantly smaller data redundancy compared to the duplicate clusters method. The ”original size” refers to the TIN model’s size when clusters are assigned to tiles only once. Using the duplicate clusters method, the total size of the TIN model after being assigned to tiles is 3.3 times the original size. Besides, the data size produced by the duplicate clusters method is 2.42 times larger than that of the merge boundaries method.

In contrast, the merge boundaries method generates a data size that is 1.36 times larger than the original size. This method results in a smaller data size because the cluster boundaries of the two differently partitioned clusters are similar, as illustrated in Figure 30. This similarity is likely due to the minimal overall geometric face connection changes after one simplification process, leading to a similar edge-cut minimization result.

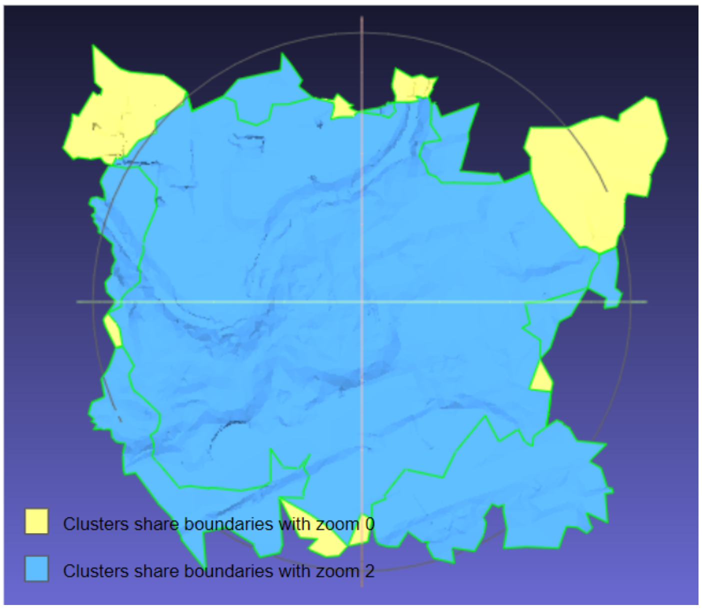

Clusters share boundaries with zoom 0
Clusters share boundaries with zoom 2

Figure 30: Different partition clusters share similar boundaries

However, considering the possibility that if there are smaller clusters around the tile boundaries, the redundancy of the duplicated clusters can be significantly reduced, we conduct experiment with smaller clusters and assign them to tiles.

**Smaller clusters assigned to tiles evaluation:**

To assess whether partitioning with smaller clusters reduces redundancy in the duplicate clusters method, we change the partition number to 256 clusters per TIN model, ideally, each zoom 1 tile has 16 clusters. Additionally, the simplification quality is evaluated to determine if smaller partitioned clusters influence the simplification result.
The result is shown in Figure 31. The size of the tiles generated by twice cluster numbers in general reduced by half compared to the former duplicate clusters result. However, it still has more redundancy than the merge boundaries result. Generally, the twice cluster number duplicate clusters method has 1.26 times the merge boundaries method file size. Technically, if the partitioned clusters are small enough, the duplicate clusters method redundancy can be reduced to as much as merge boundaries.

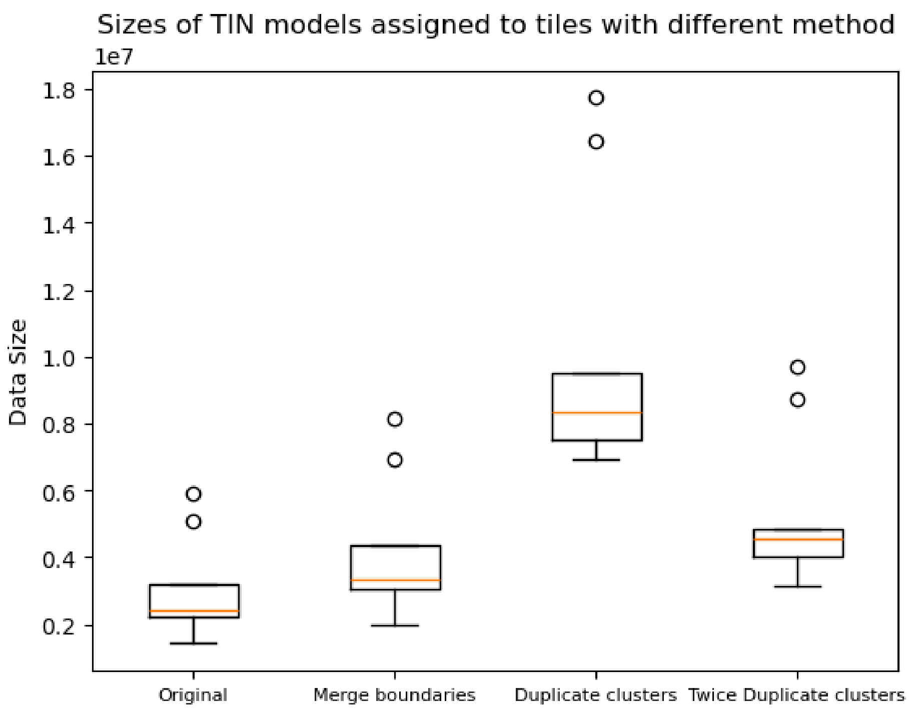


Figure 31: Sizes of TIN models assigned to tiles with different methods

In that case, we evaluated the simplification result of duplicate cluster methods with and without double cluster numbers. As shown in Figure 32, the double cluster simplification tends to have higher height error in general.

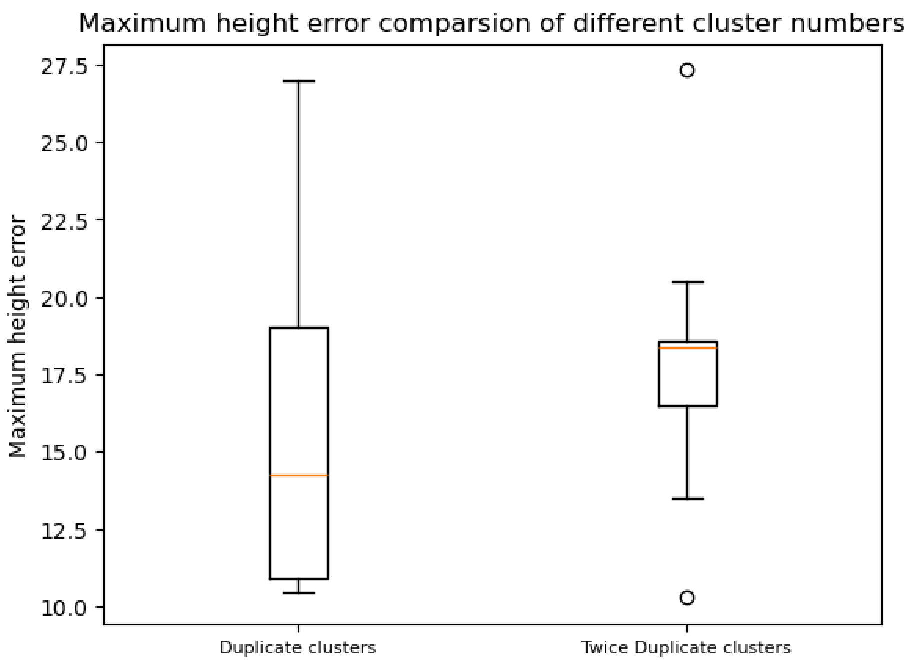


Figure 32: Simplification error with different cluster numbers

To compare with the merge boundaries method specifically, we partition the TIN models until the redundancy of the duplicate clusters when assigned to the tiles is similar to the former merge boundaries result (around 1.1 times the merge boundaries redundancy), and compare the simplification error.

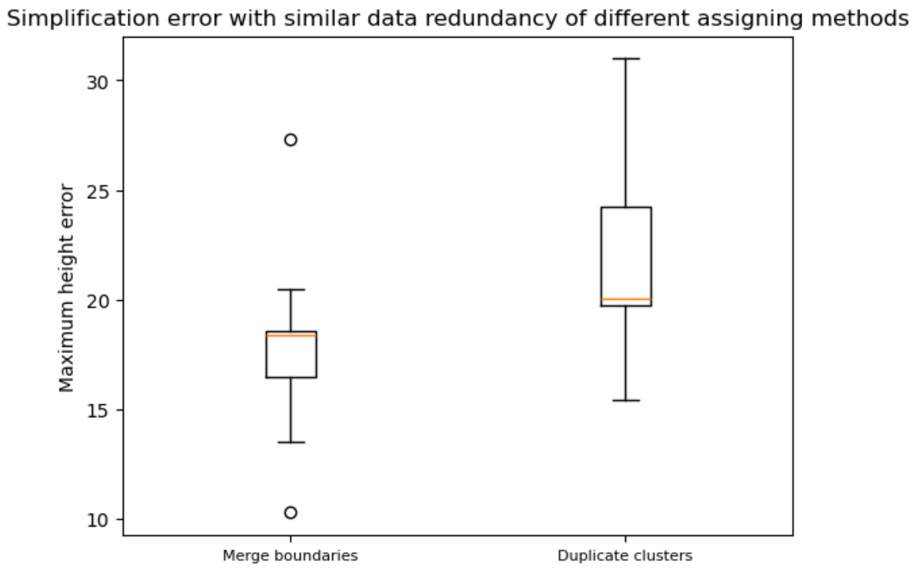


Figure 33: Simplification error of merge boundaries and duplicate clusters with similar redundancy

As we can conclude from Figure 33, when with similar redundancy, merge boundaries has smaller errors than duplicate clusters after simplification. This is because to decrease the data redundancy, the clusters are smaller and the locked boundaries amount increases. Consequently, the simplification efficiency is influenced. Thus, for this thesis, boundaries merge is used as the solution to assign clusters into tiles.

# 4 Implementation

To explore the feasibility of the proposed method for constructing seamless large-scaled multi-Level of Detail (LoD) tiled terrain, a prototype is developed using C++ as the programming language. The implementation utilizes the CGAL library (cga, 2021) and METIS (Karypis and Kumar, 1999).

To assess the quality of models after simplification, Python is utilized in conjunction with the Rasterio library (Gillies et al., 2013–2023) to compute the height different of the models between original data and afrer simplification to indicate the simplification error.

## 4.1 Data Details

### 4.1.1 Data Formats

The data used in this thesis include:

- **DEM data:** The original DEM data is provided in GeoTIFF format, representing a height map model. This DEM is transformed into a TIN using TIN-Terrain (HERE Technologies, 2024).
- **TIN data:** The 3D TIN models in this thesis are stored in the Object File Format (OFF). OFF format is chosen for its simple data structure, similar to Wavefront .obj (OBJ) format, but with more official standardization.

.

### 4.1.2 Data Structures

Several data structures are used in this thesis, and they are listed as follows:

- **TIN Mesh:** `CGAL::Surface_mesh`
  The TIN mesh is read into CGAL for processing in the `CGAL::Surface_mesh` structure (Botsch et al., 2024). This structure is a class from CGAL (cga, 2021) specifically designed for representing 3D meshes. It can be utilized as a halfedge (Kettner, 2024) data structure or a polyhedral surface. Note that a halfedge data structure is an edge-centered data structure capable of maintaining incidence information of vertices, edges, and faces.

- **Partition Information:** `Face_id_map`
  As a class derived from CGAL, it's a property map composed of `CGAL::Surface_mesh::Face_index`, which is the index of mesh faces (triangles), and an index to indicate the partition index. A property map is a `LvaluePropertyMap` (Boost Project, 2001–2023) which provides functions for accessing a reference to a value object.

- **Partition Information:** `Vertice_id_map`
  Similar to `Face_id_map`, but it stores the vertices index and partition index.

- **Partitioned Cluster:** `Filtered_graph`
  It's a struct inherited from `CGAL::Face_filtered_graph` (CGAL Project, 1996–2024), which is an adaptor that creates a filtered view of a graph by restricting it to a subset of faces. The partitioned cluster is filtered with partition index and `Face_id_map`.

- **Constrained Elements:** `is_constrained_edge_map`
  It's a struct inherited from the Property map. It is composed of a boolean value to indicate whether an edge is constrained and a reference to the edge.

- `halfedge_descriptor`
  It is an identifier defined by `graph_traits<Graph>` (Libraries, 2021). `graph_traits<Graph>` provides associated types for the graph, and `halfedge_descriptor` is used to identify a halfedge.

- `edge_descriptor`
  Similar to `halfedge_descriptor`, it is for the identification of an edge.

- **Tile**
  It's a class composed of four integers, X, Y, Z, and CRS. CRS is the coordinate reference system code of the tile. X and Y are the row and column index of the tile, and Z is the zoom level. This class takes partitioned clusters, calculates the centroid points of the clusters, and assigns them to corresponding tiles at a certain zoom level based on centroid points.

## 4.2 Engineering Decisions

Different choices of implementation tactics would lead to different outputs, and influence the efficiency of the program. In this section, several important decisions in implementation are presented and discussed. Note that these decisions are not necessarily the best options to achieve potential highest performance.

**Use halfedge structure to open TIN as surface mesh**

The halfedge structure is an edge-centered data structure that maintains the incidence information of vertices, edges, and faces. This structure supports the edge-collapse simplification method, making it ideal for storing partition index properties. It can reference the constrained_edge_map, ensuring that preserved features and cluster boundaries are not simplified.

**Use constrained_edge_map to preserve features**

To preserve important elements during simplification, such as roads and building footprints, these elements are added to the `constrained_edge_map`. This prevents their simplification, ensuring that they remain accurate and topologically correct for purposes like navigation. These elements are not generalized for each zoom level and are inserted into the terrain. Instead, their topology is carefully preserved and simplified along with the terrain.

**Choose random edges from halfedge structure as preserved constrained features**

Due to time constraints and the absence of real feature data that needs preservation during simplification, such as 3D road and building footprint data, the preserved constrained features are not based on actual data. Instead, for each cluster, a random set of connected faces is selected, and the edges of these faces are added to a constrained map to avoid simplification. These edges are stored in the property map, and the preservation of the constrained edges is validated using the `halfedge_descriptor`, as described in Section 4.1.2.

**Use data's local reference system to build tiles**

A tiled web map structure using CRS EPSG:3857, also known as Pseudo-Mercator, is built. This projected CRS is used for rendering maps like Google Maps and OpenStreetMap. Additionally, CRS EPSG:4326, the World Geodetic System tile structure, is also implemented. Besides these common tile structures, a pseudo-tiled web map structure based on the experimental data's CRS is created. This pseudo-tiled structure uses the experiment area as the map area, with the bottom-left corner of the experiment area serving as the origin. The zoom levels range from 2 (lowest) to 0 (highest). This setup facilitates the visualization and analysis of data assigned to tiles.

After partitioning the clusters and assigning them to the corresponding tiles, the clusters are merged internally. To ensure a seamless connection with tiles at other zoom levels, the outer boundaries must be locked. However, the internal cluster boundaries, which do not connect to other zoom-level tiles, can be merged and unlocked. Merging these internal locked boundaries improves the simplification results and reduces cluster sizes, as fewer locked boundaries result in better simplification. If not merged, duplicated cluster boundaries cause redundancy. As shown in Figure 34, the locked boundaries inside the clusters (indicated by yellow) are merged and unlocked, while the outer locked boundaries (indicated by red) are maintained.

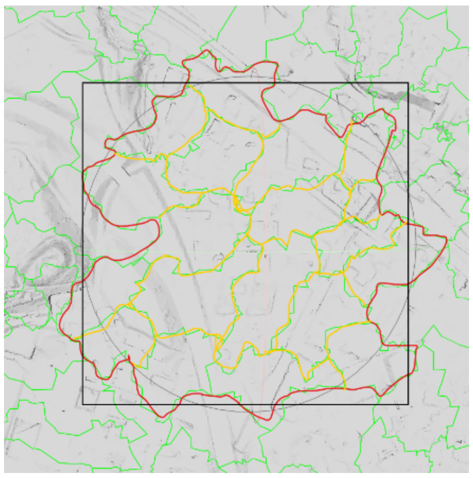


Figure 34: Merge clusters inside a tile

## 5 Results and Analysis

### 5.1 Datasets

An urban area with varied elevations is selected as the experimental dataset to evaluate whether the implemented prototype meets the research objectives. This choice is made for two main reasons. First, urban areas typically have a higher density of buildings and roads, which are crucial for navigation studies. Second, testing the connections between different Levels of Detail (LoD) in terrain models requires height variation; a flat terrain would not adequately highlight the advantages and disadvantages of the prototype.

Consequently, the experimental data is sourced from swissALTI3D (swi), which provides digital terrain model (DTM) data at a resolution of 0.5 meters in GeoTIFF format. Each DTM file is a 1 km $\times$ 1 km GeoTIFF tile. The GeoTIFF tiles are indexed with row and column numbers. The coordinate reference system (CRS) of the data is the current Swiss coordinate system: *LV95 LN02*, i.e., EPSG:2056.

To demonstrate the broad applicability of the methodology, we selected two areas as the experiment data. The experiment data selected for this thesis consists of 9 tiles covering a 3 km $\times$ 3 km area in Lausanne, Switzerland, and the other is 16 tiles covering a 4 km $\times$ 4 km area in Zurich, Switzerland. These two area are all located in the center of cities, characterized as an urban environment with multiple roads and significant topological variation (height differences). The maximum height in the Lausanne test area is 522.83 meters, while the minimum height is 367.68 meters, and for Zurich is 583.123 meters and 395.948 meters. Therefore, it is a suitable location to experiment with 3D terrain construction for navigation purposes. Each tile has an index for identification from swissALTI3D. The selected area in Lausanne spans from row 33 to row 35 and from column 52 to column 54, as shown in Figure 35, and in Zurich is from row 80 to row 83, and from column 47 to column 50.

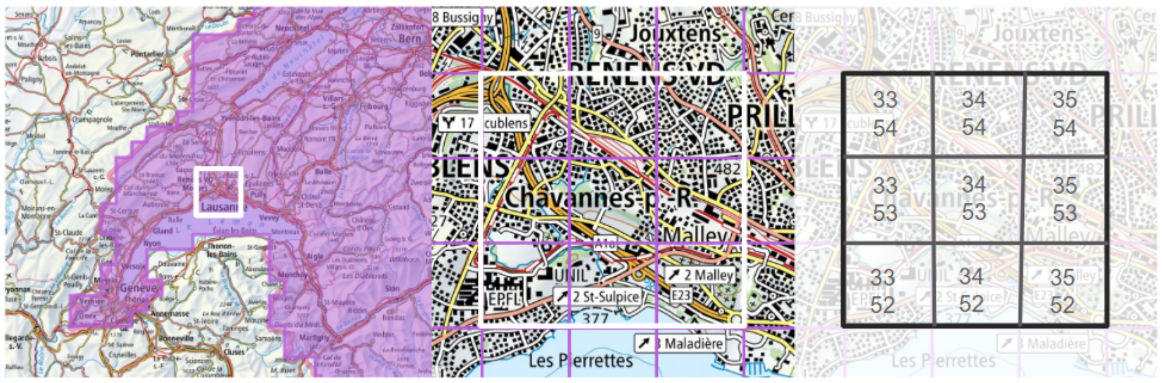


Figure 35: Experiment area

Table 2: Max height error and geometry number for Lausanne tiles after converting to TIN

| TIN Tile | 3352 | 3353 | 3354 | 3452 | 3453 | 3454 | 3552 | 3553 | 3554 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| Vertices number | 51016 | 63594 | 66603 | 39226 | 60116 | 80566 | 75219 | 93875 | 85022 | 68360 |
| Faces number | 101845 | 126976 | 132972 | 78254 | 119931 | 160849 | 150164 | 187405 | 169749 | 136460 |
| $E_{max}$ | 16.9836 | 10.2482 | 11.0498 | 12.6974 | 18.5229 | 27.2795 | 24.9407 | 8.01497 | 6.8 | 15.1708 |

As to Zurich, the 16 DEM tiles are combined due to time limitation, the vertices number is 1322370, and the faces number is 2643550.

Considering the elements attached to the terrain that need to be preserved during simplification for navigation purpose, such as roads and building footprints, they are not under consideration in this thesis. Due to time constraints and the absence of corresponding 3D road and building footprint data, real data was not utilized in this thesis. However, aside from the implementation of boundaries constraints, this thesis also randomly selects some edges as constraint edges in the terrain data to check whether it is possible to reserve elements inside the terrain during simplification, as mentioned in Section 4.2. For future applications, these random selected constraint edges can be altered with selections from actual data and can be one of the solutions to integrate roads and building in the 3D terrain.

## 5.2 Evaluation

### 5.2.1 Simplification result

The evaluation of the simplification result is based on the maximum height errors between the simplified TIN models and the original models.

Aside from considering the maximum height error of the simplified TIN models, the visualization effect is also taken into consideration as a factor for evaluating simplification. The connection between different tiles is visualized using MeshLab (Cignoni et al., 2008), an open-source system for processing, editing, and visualizing 3D triangular meshes. The locked boundaries are inspected in particular to evaluate the reduction of artifacts caused by complicated locked boundaries.

### 5.2.2 Preservation constraints

To evaluate whether the constrained features (i.e., essential elements on the terrain that need preservation) are maintained, the edges of these constrained features are checked to ensure they are still present after simplification. The `halfedge_descriptor` and property map mentioned in Section 4.1.2 are used to implement this function. The `halfedge_descriptor` serves as an identifier for identifying a halfedge, in this case, the constrained edges. Additionally, the property map provides a reference to a value object. Therefore, the constrained edges are first stored in a property map with references to them. Then, after simplification, the references of the constrained edges are checked to determine whether they are present in the simplified mesh using `halfedge_descriptor`.

The simplification algorithm is described in the following pseudo-code 2:

**Algorithm 2** Edge constraint preservation check

```
Input: Triangle_mesh, Constrained_edges
Output: True or False
  while Constrained_edges $\neq 0$ do
      Edge_constrain_map $\leftarrow$ Constrained_edge
  end while
  Simplify the mesh
  while Edge_constrain_map $\neq 0$ do
      if Edge in Edge_constrain_map And Edge in Triangle_mesh then
          Continue
      else if then
          Return False
      end if
  end while
  Return True
```

### 5.2.3 Visually seamless connection

As mentioned in Section 3.5.2, the duplicate area does not cause cracks in the visualization. To test the seamless connection between two different LoD tiles, the boundaries of the different tiles are inspected. As shown in Figure 36, as long as the zoom n tiles contain the zoom $n+1$ geometry connect boundaries vertices, and the zoom $n+1$ tiles contain the zoom $n$ geometry connect boundaries vertices, it means the two different zoom tiles boundaries vertices can find corresponding vertices in each other, and it demonstrates that two different zoom tiles can be visually seamlessly connected.

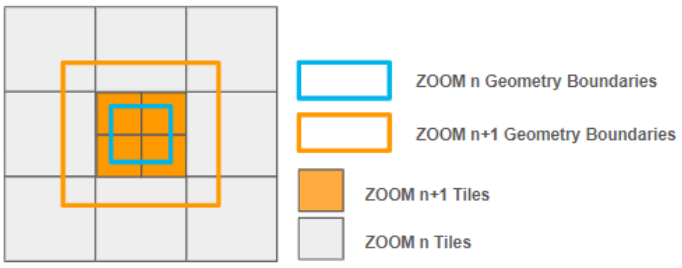


Figure 36: Connection of different zoom level tiles

## 5.3 Result and Analysis

The experiment area in Lausanne generates 576 zoom 3 tiles, with 24 rows and 24 columns, 144 zoom 2 tiles, with 12 rows and 12 columns, 36 zoom 1 tiles, with 6 rows and 6 columns, and 9 zoom 0 tiles. Each tile is identified with index zoom level/row/column. For example, the tile in zoom 2, row 0, column 0 is Tile 2/0/0. Similarly, the Zurich data contains 1024 zoom 3 tiles.

The connection between zoom 2, zoom 1, and zoom 0 tiles is visualized. The result shows that the tiles can be joined seamlessly together in visualization. As shown in Figure 37, 2 tile in zoom 0 (Tile 0/0/0, Tile 0/0/1), 2 tiles in zoom 1 (Tile 1/0/2, Tile 1/1/2), and 3 tiles in zoom 2 (Tile 2/0/6 - 2/2/6), 4 tiles in zoom 3 (Tile 3/0/13 - Tile 3/2/13, Tile 3/0/14) are joined together.

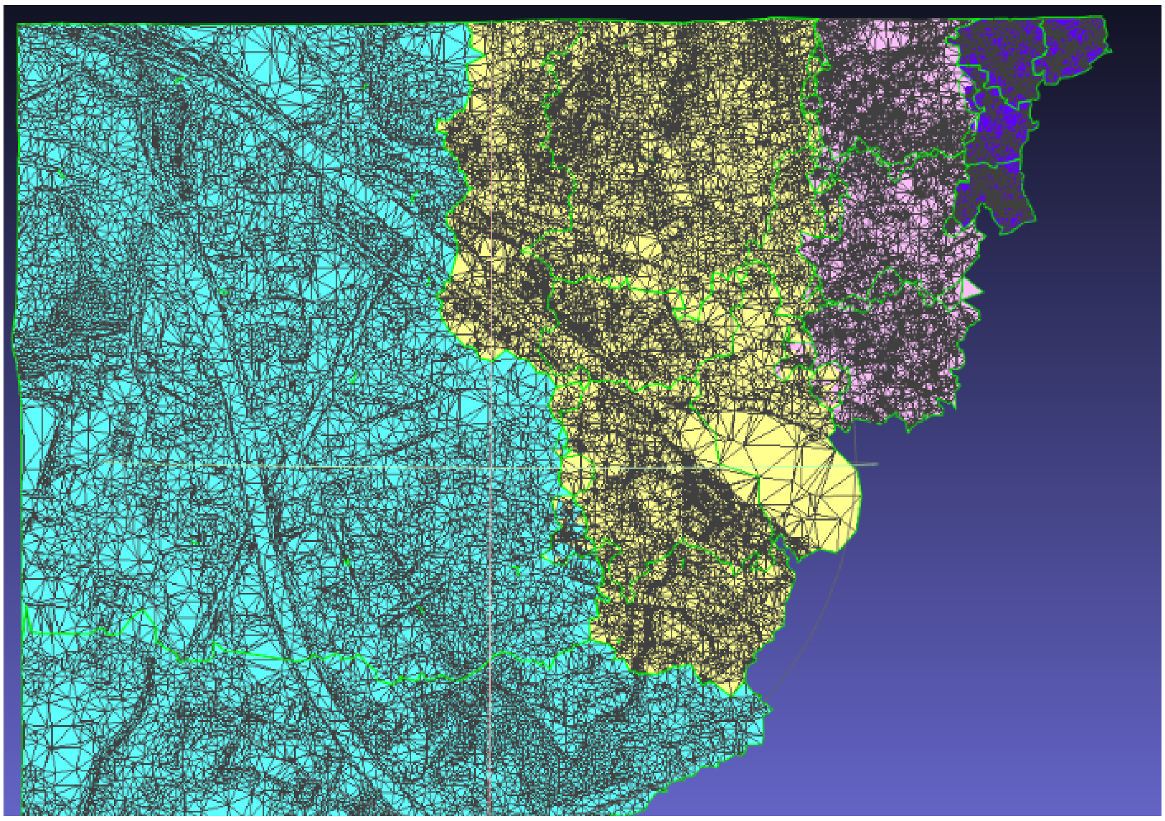
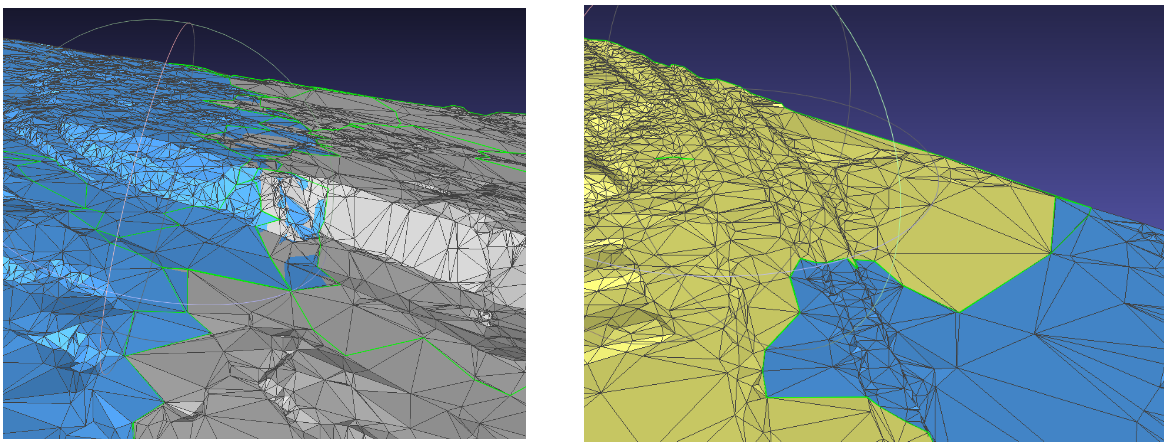


Figure 37: Connection of zoom 0, zoom 1, zoom 2 and zoom 3 tiles

As shown in Figure 38, the connection of the different zoom tiles would cause gaps but not cracks. This is because the vertices on the boundaries of the tiles can always have corresponding vertices on the other neighboring zoom tiles.

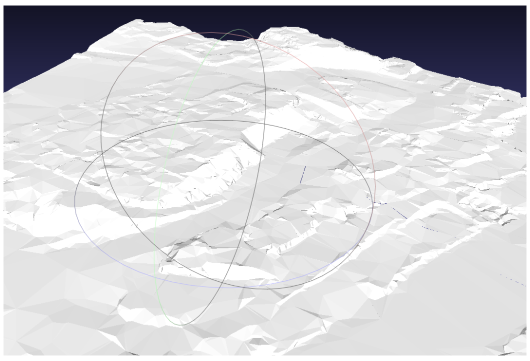


(a) Boundaries connection between zoom 2 and zoom 1

(b) Boundaries connection between zoom 0 and zoom 1

Figure 38: Comparison of boundaries connections at different zoom levels

However, as mentioned in the previous discussion, overlaps between different zoom tiles would still have seamless visualization in rendering, as shown in Figure 39.


(a) Seamless connection in overlap area

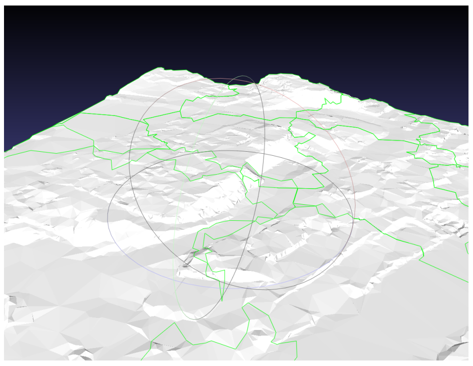

(b) Boundaries of the overlap areas

Figure 39: Rendering result in overlap areas

### 5.3.1 Triangle clustering analysis

The evaluation of the triangle clustering method is based on simplification result analysis and complicated boundary reduction analysis.

#### Simplification efficiency

To evaluate the effect and confirm the necessity of edge-cut minimization, we conducted experiments on the data. Each TIN model was partitioned into eight clusters using two different methods: one with edge-cut minimization and the other with random partitioning. After partitioning, we performed edge-collapse simplification within each of the cluster, while keeping the cluster boundaries locked during the process. The simplification was halted when the edge count of the simplified models reached 50% of the original models' edge count. The quality of the simplification was then evaluated, as described in Section 5.2, by measuring the maximum error of the simplified model compared to the original model.

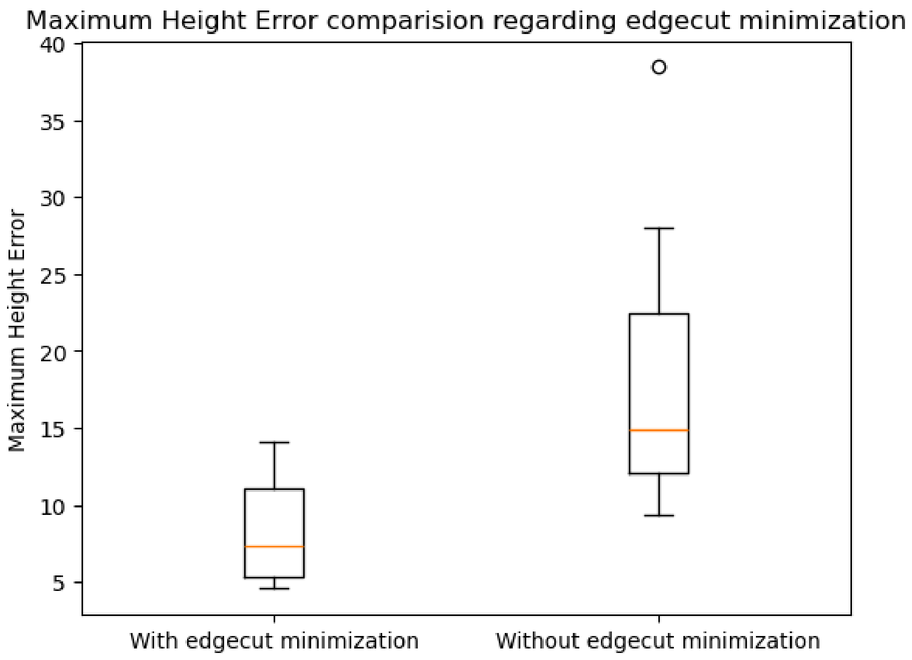


Figure 40: Maximum height error with edge-cut minimization partition and random partition simplification

As shown in Figure Figure 40, the TIN models partitioned with edge-cut minimization exhibit smaller maximum height errors compared to those partitioned randomly. The average maximum height error for the nine TIN models with edge-cut minimization is 8.5989 meters. In contrast, the average maximum height error for the randomly partitioned models is 35.5115 meters, which is about four times higher. However, in the case of terrain model 3552, the maximum height errors for both partitioning methods are similar. This could be due to the random choice of locked boundaries coincidentally resulting in a small edge cut, or the distribution of the TIN model being relatively uniform, thus causing the partition strategy less impactful on the simplification outcome. Despite this exception, we can conclude that edge-cut minimization generally enhances simplification efficiency, and is therefore used as the preferred method for partitioning TIN models.

To compare the influence of triangle clustering on simplification, we carried out an experiment on the 9 TIN models. The test data consisted of triangle clusters in zoom 1. In the first experiment, we re-partitioned the clusters to simplify the models, while in the second experiment, we simplified the clusters without re-partitioning them, serving as a control group.

The results, as shown in Figure 41, indicate that simplification results with re-partitioning the clusters generally have better outcomes compared to those without. However, 6 TIN models exhibited very similar errors. To better assess the effect of triangle clustering, we conducted further experiments.

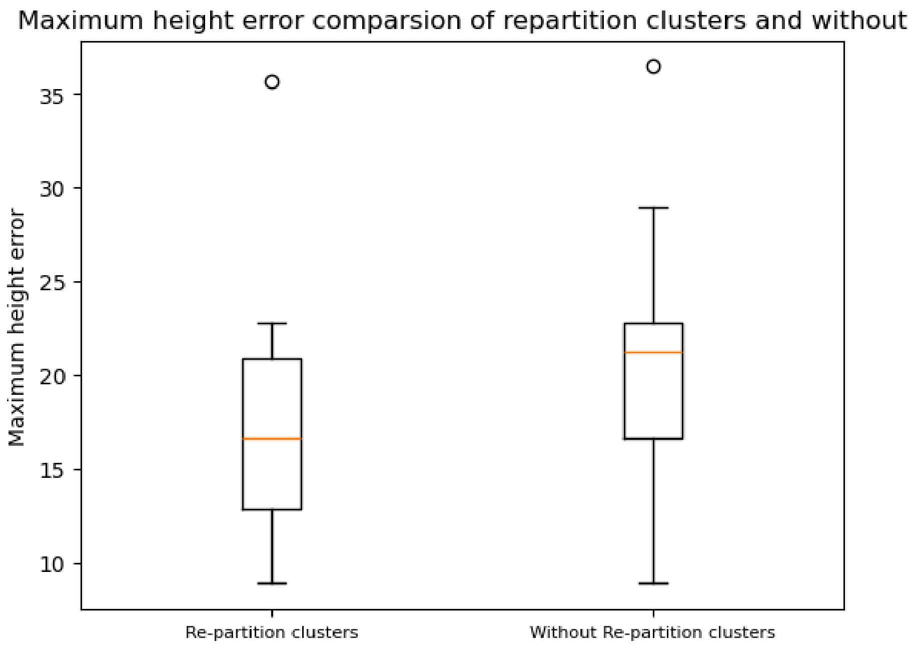


Figure 41: Maximum height error comparison of re-partition simplification and fixed locked boundaries comparison

We clustered the TIN model 3352 into 64 clusters and conducted simplification on it five times. Similarly, we performed one experiment with re-partitioning before simplification and a control group without re-partitioning. From the results shown in Figure 42, we can observe that the simplification errors caused by re-partitioning and changing the locked boundaries of the TIN model are much smaller than those without re-partitioning the clusters. Re-partitioning the clusters can improve the simplification result because the locked boundaries of the clusters can be unlocked and simplified when the TIN model is re-partitioned, and new boundaries are selected as locked.

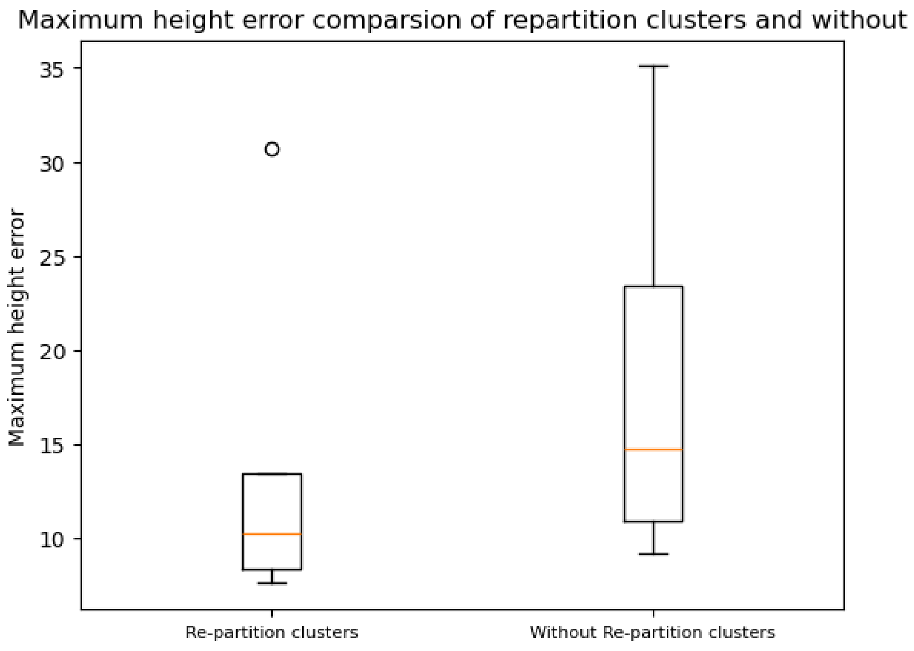
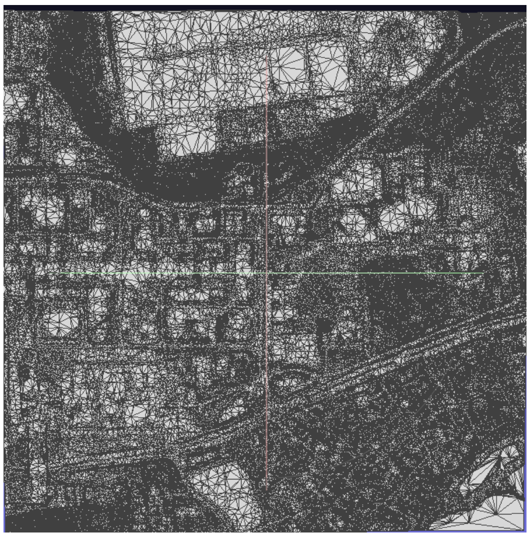
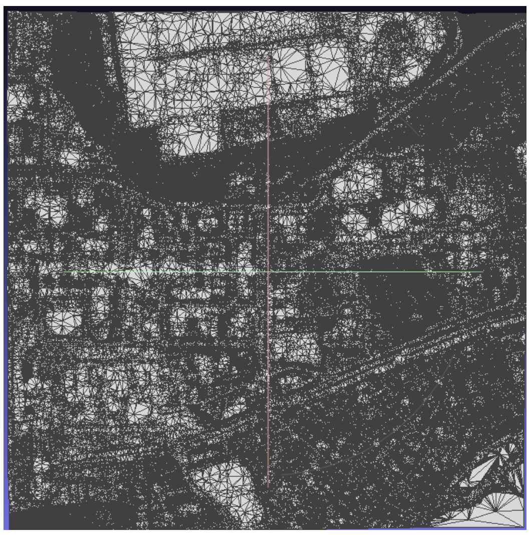


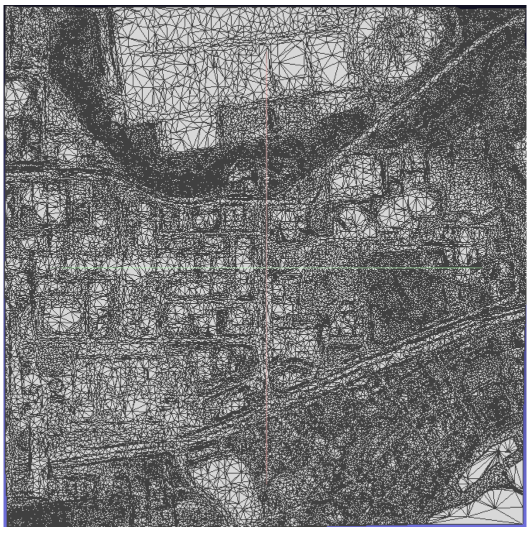
Figure 42: Error of 5 times simplification with and without re-partition

**Complicated edges caused by locked boundaries**

The simplified models are visualized to evaluate the complicated boundaries problem.


(a). Unchanged locked boundary simplification 1


(b). Changed locked boundary simplification 1


(c). Unchanged locked boundary simplification 2

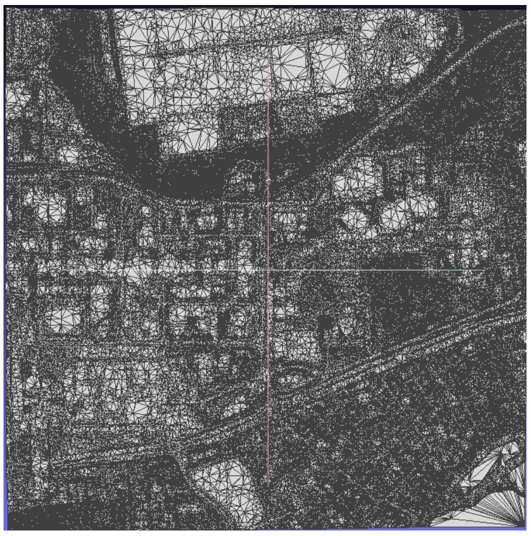
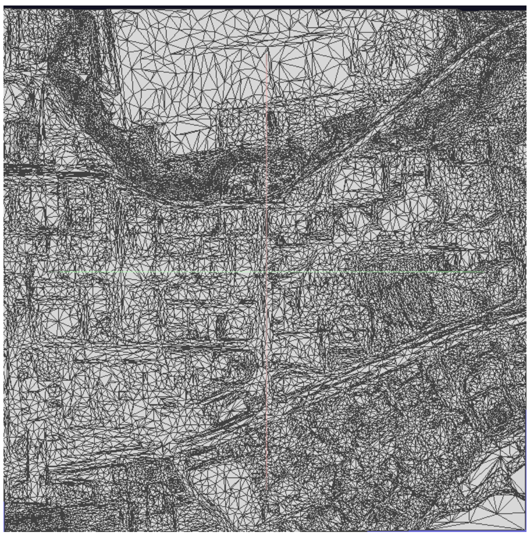
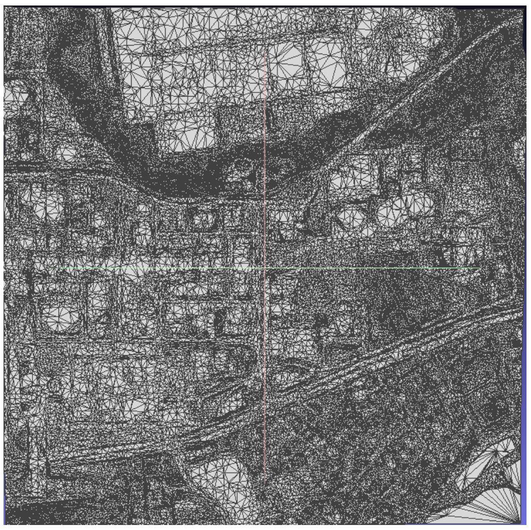
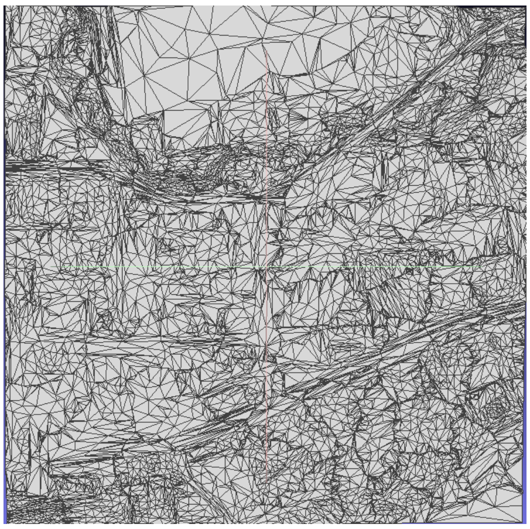
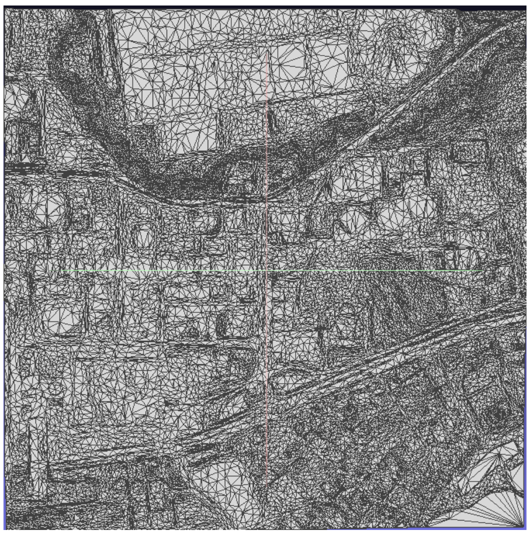

(d). Changed locked boundary simplification 2

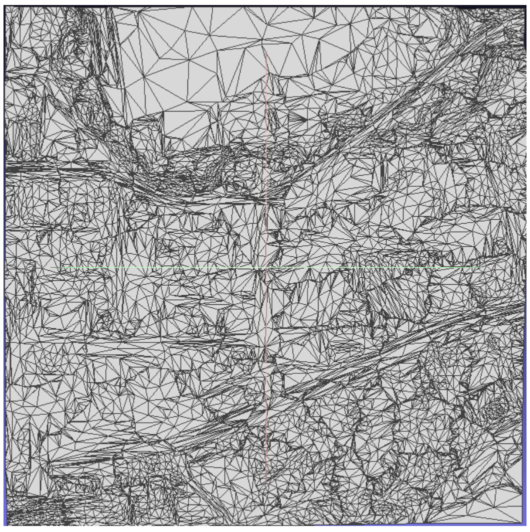
Figure 43: Comparison of the simplification results of complicated boundaries (Part 1)


(e) Unchanged locked boundary simplification 3


(f) Changed locked boundary simplification 3


(g) Unchanged locked boundary simplification 4


(h) Changed locked boundary simplification 4


(i) Unchanged locked boundary simplification 5

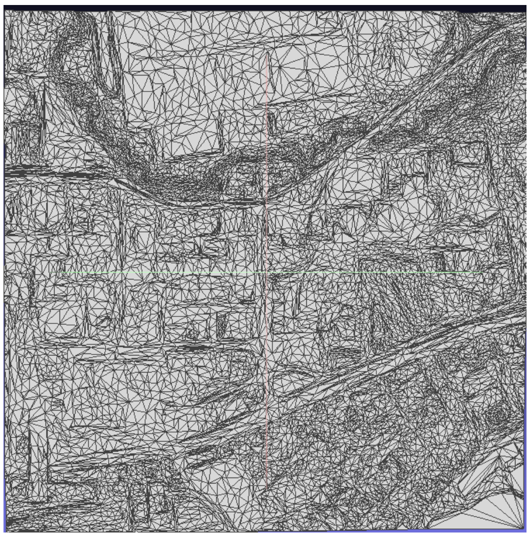
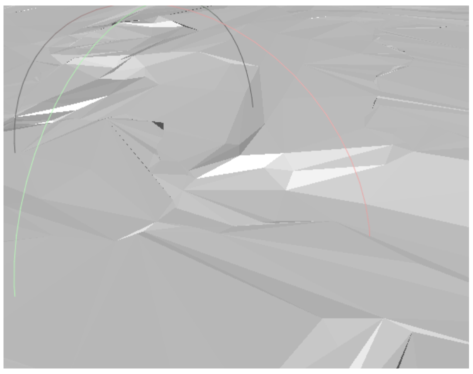

(j) Changed locked boundary simplification 5

Figure 43: Comparison of the simplification results of complicated boundaries (Part 2)

In Figure 37(g) and Figure 37(i), the complicated boundaries caused by unchanged boundaries are obvious. Due to the dense area around the locked boundaries, the other parts of the TIN model are relatively coarser because the edge-collapse simplification used in the experiment removes half of the edges each time. This uneven distribution of the edges not only leads to inefficiency in simplification and higher errors but also causes artifacts in visualization.


(a) Complicated boundaries artifacts

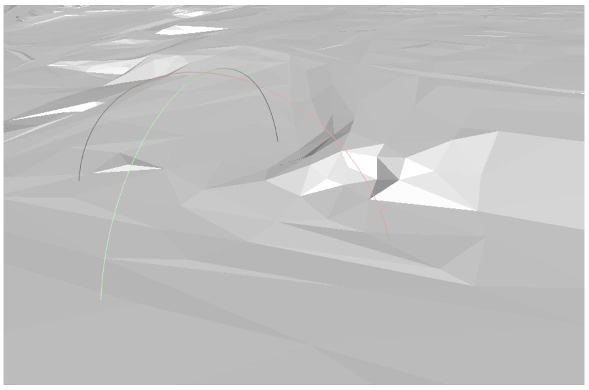

(b) Simplified boundaries artifacts

Figure 44: Artifacts on the boundaries

As shown in Figure 44, the simplified boundaries tend to have a smoother connection with the surrounding meshes compared to the complicated boundaries area.

# 6 Conclusion and Future Work

## 6.1 Conclusion

This thesis proposes a solution to build multi-LoD large-scale terrain for web mapping with Tiled web map. In the former research, during the process of building different LoD models and maintaining the seamless connection between them, the solution is to lock the boundaries of the different models. However, with the simplification of the models, the locked boundaries become complicated, leading to simplification inefficiency and artifacts in visualization. Inspired by the techniques of Nanite (Nan), the triangle clustering method is investigated to be applied in the scenario of tiled web maps to solve the problem caused by locked boundaries. Triangle clustering involves partitioning the model mesh into triangle clusters and simplifying the model inside the clusters without changing cluster boundaries. Subsequently, to adapt the triangle clustering to the tiled web map schema, different solutions to partition the terrain mesh and assign the clusters to tiles are proposed, implemented, and tested. The use case of the large-scale 3D terrain in web mapping is also considered, and a solution to preserve the topology of elements on the terrain, such as roads, is also proposed.

To partition the triangles into clusters, the edge-cut minimization method is used. Afterward, the models are simplified inside the clusters with the cluster boundaries unchanged. To preserve the constrained elements on the terrain and locked boundaries, the edges of these elements are stored in the `is_constrained_edge_map`, which is a property map that stores whether an edge is constrained or not. Furthermore, to assign the partitioned clusters to web tiles, three different assigning methods—double zoom tiles, duplicate clusters, and boundaries merge—are proposed and tested. The evaluation of these methods is based on the tiled web use case, data redundancy, and simplification quality. The evaluation shows that for the use of web map, the double zoom tiles method is not suitable. As to data redundancy, the boundaries merge method causes the smallest redundancy. However, although increasing the cluster number during triangle partition can lead to smaller clusters and consequently less redundancy in duplicate clusters, it also causes more locked boundaries and increases the simplification errors. When the merge boundaries and duplicate clusters have similar redundancy, the simplification error of the duplicate clusters is higher. Thus, for our research, the merge tiles solution is better suited.

The primary research questions, **How to construct seamless large-scale multi-LoD tiled terrain, with consideration of constraints in the process of simplification**, can be answered as follows:

- How to simplify the mesh to generate different LoD models?
  - In terms of simplification result, the greedy insertion simplification and edge-collapse simplification have similar error. However, the greedy insertion tends to have more vertices in areas with higher height variation. Also, the constrained edges need to be preserved in the simplification, which applies to the edge-collapse simplification process (removing the edges). Thus, considering the simplified model size and constraints preservation, edge-collapse simplification is used.
- How to partition the mesh into different clusters?
  - The more locked boundaries there are, the less efficient the simplification process becomes, as locked boundaries cannot be simplified. In this case, edge-cut minimization is used to partition the clusters. Edge-cut minimization aims to partition the triangles that share the most edges together, making the clusters more compact. Consequently, the clusters have fewer locked boundaries in general, and the simplification efficiency increases.
- How to preserve the boundaries of the clusters during simplification?
  - We set constraints on the terrain during simplification. The edges that need to be preserved are stored in the `is_constrained_edge_map` and are not removed during simplification.
- How to assign the clusters to the tiles?
  - The triangle clustering generates two cluster boundaries for each zoom level tile because the triangles are re-partitioned at every zoom level. Three different assigning methods—double zoom tiles, duplicate clusters, and boundaries merge—are proposed and tested. The double zoom tiles method is not suitable for web mapping navigation, as the user needs to retrieve double the data in the same area when moving in different directions. Both the duplicate clusters and boundaries merge methods cause redundancy in data storage. However, the duplicate clusters method tends to cause more data redundancy than the boundaries merge method.
- How to assess the quality of the simplification methods?
  - The quality of the simplification is assessed based on the maximum height error of the simplified models, as the maximum height error and root mean square error of the height are highly correlated (Chen and Zhou, 2013). The simplified TIN mesh is rasterized to calculate the height error.
  - Besides, the visualization effect is also taken into consideration. Boundaries of different LoDs are visualized and compared to evaluate the effect of the triangle clustering on the complicated locked boundaries.

The result implies that the triangle clustering techniques can improve the simplification efficiency. Besides, the visualization result of the locked edges indicates that, by changing the locked boundaries, the density of the triangles around the boundaries would be significantly decreased compared to maintaining the same locked boundaries. Consequently, the visualization of the triangle clustering simplified models shows it has a smoother transformation in the areas surrounding the locked boundaries. The preserved elements have been proven to remain during simplification after adding the edges in the constrained map. It needs to be noted that the number of constrained edges influences the efficiency of the simplification. That is to say, the more constrained edges there are, the higher the simplification errors in general.

The clusters are assigned to tiles to test the different zoom level tiles connection as an application of triangle clustering to tiled web mapping. For each zoom level tile, the triangles are re-partitioned, thus there are two cluster boundaries. Both cluster boundaries are preserved and assigned to tiles. This boundary merging method would cause an overlap, but in visualization the connection between tiles is seamless. The redundancy of the data in the test area is only 37% of the original area.

## 6.2 Potential Implications

The thesis provides insight into a method to construct a seamless large-scale 3D terrain model and preserve the features on it. The prototype implemented in this thesis also contributes to the application of triangle clustering techniques on web mapping. The findings gained from the thesis could be of assistance to the large-scale models of web dissemination, and an improvement for the structured multi-LoD model construction.

## 6.3 Reflection and Future Work

In the future, our research could be improved and extended as follows:

Firstly, the preservation of the features on the terrain does not involve real-life data. This is because of the time limitation, and also the lack of accurate road data to match the terrain. Thus, this thesis only investigates the preservation of random edges.

Secondly, the thesis does not carry out simplification on models with different geometry complexity. For example, the initial TIN models to be partitioned and simplified are set to be generated with error 0.5 meters with Tin-Terrain (HERE Technologies, 2024), while more experiments on triangle clustering could be carried out on models with different accuracy.

Thirdly, the conservation of the elements that are not completely attached to the terrain is not considered. The thesis only considers elements on the terrain, but elements like bridges and tunnels are not taken into consideration.

There are multiple directions for future works:

- Generating more zoom levels:
  - In the thesis we only test three zoom level connections. More zoom levels can be generated to test the triangle clustering, and also the redundancy of the data.
- Considering different simplification termination conditions:
  - The thesis sets the termination of the simplification to when the simplified edges are 50% of the original model, the simplification stops. This is set to 50% because the neighboring coarser zoom tiles size is twice the finer tiles size. However, the stop condition can be tested for different use cases. And for navigation, this stop condition can be set to maximum height error to control the accuracy of the map in different zoom levels.
- Experiment on various terrain areas:
  - Due to time limitations, the experiment data is 3 km $\times$ 3 km area data in Switzerland. Although they are tested with the same partition cluster numbers and the same termination condition, the height error varies. Although this is probably caused by different height varieties in different areas, in some cases, the model simplified with more locked edges has better simplification results. The relationship between the simplification errors, locked boundary numbers and simplified model size deserves further investigation.
- Evaluation of visualization effects
  - The thesis only evaluates the improvement of the complicated boundary area of the triangle clustering by visualizing it and comparing it with the unchanged locked boundary area. To better evaluate the result and quantify the visualization artifacts, the visualization of the models can be rasterized per pixel and error statistics can be computed from it.

## References

Brian Karis nanite a deep dive. https://advances.realtimerendering.com/s2021/Karis_Nanite_SIGGRAPH_Advances_2021_final.pdf. Accessed: 2024-05-02.

T-junction. https://victorbush.com/2015/01/tessellated-terrain/. Accessed: 2024-05-05.

CGAL surface mesh simplification: Edge count ratio stop predicate. https://doc.cgal.org/latest/Surface_mesh_simplification/classCGAL_1_1Surface__mesh__simplification_1_1Edge__count__ratio__stop__predicate.html. Accessed: May 13, 2024.

swisstopo. https://map.geo.admin.ch/?topic=swisstopo&bgLayer=ch.swisstopo.pixelkarte-farbe&lang=en&layers=ch.bav.haltestellen-oev,ch.swisstopo.swissnames3d&lon=7.45011&lat=46.93913&elevation=530&heading=360.000&pitch=-40.198. Accessed: 2024-05-05.

CGAL user and reference manual. https://doc.cgal.org/latest/Manual/packages.html, 2021. Version 5.2.1.

Boost Project. LvaluePropertyMap. https://www.boost.org/doc/libs/1_85_0/libs/property_map/doc/LvaluePropertyMap.html, 2001–2023.

M. Botsch, D. Sieger, P. Moeller, and A. Fabri. Surface mesh. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition, 2024. URL https://doc.cgal.org/5.6.1/Manual/packages.html#PkgSurfaceMesh.

R. Campos, J. Quintana, R. Garcia, T. Schmitt, G. Spoelstra, and D. M. A. Schaap. 3d simplification methods and large scale terrain tiling. *Remote Sensing*, 12(3), 2020. ISSN 2072-4292. doi: 10.3390/rs12030437. URL https://www.mdpi.com/2072-4292/12/3/437.

CGAL Project. CGAL::Face_filtered_graph. https://doc.cgal.org/latest/BGL/structCGAL_1_1Face__filtered__graph.html, 1996–2024.

Y. Chen and Q. Zhou. A scale-adaptive dem for multi-scale terrain analysis. *International Journal of Geographical Information Science*, 27(7):1329–1348, 2013. doi: 10.1080/13658816.2012.739690. URL https://doi.org/10.1080/13658816.2012.739690.

M. Christen and S. Nebiker. *Large Scale Constraint Delaunay Triangulation for Virtual Globe Rendering*, pages 57–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-12670-3. doi: 10.1007/978-3-642-12670-3_4. URL https://doi.org/10.1007/978-3-642-12670-3_4.

P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Bdam — batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3):505–514, 2003a. doi: https://doi.org/10.1111/1467-8659.00698. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00698.

P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Planet-sized batched dynamic adaptive meshes (p-bdam). pages 147–154, 11 2003b. doi: 10.1109/VISUAL.2003.1250366.

P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. https://www.meshlab.net/, 2008. Version 2020.12.

T. C. D. Community. High precision maps industry research report 2019, 2019. URL https://cloud.tencent.com/developer/news/537185. Accessed: 2024-05-17.

W. Evans, D. Kirkpatrick, and G. Townsend. Right-triangulated irregular networks. *Algorithmica*, 30:264–286, 06 2001. doi: 10.1007/s00453-001-0006-x.

M. Garl and P. Heckbert. Fast polygonal approximation of terrains and height fields. 05 1997.

M. Garl and P. Heckbert. Fast triangular approximation of terrains and height fields. 10 1999.

M. Garland. Terra. `https://mgarland.org/software/terra.html`, Accessed: 2024.

M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH '97*, page 209, 1997.

D. Gavran, S. Fric, V. Ilic, F. Trpevski, and S. Vranjevac. 3d control of obstacles in airport location studies. 04 2017.

S. Gillies et al. Rasterio: Geospatial raster i/o for python programmers. `https://rasterio.readthedocs.io/`, 2013–2023.

E. Gobbetti, F. Marton, P. Cignoni, M. Di Benedetto, and F. Ganovelli. C-bdam – compressed batched dynamic adaptive meshes for terrain rendering. *Computer Graphics Forum*, 25:333 – 342, 09 2006. doi: 10.1111/j.1467-8659.2006.00952.x.

HERE Technologies. tin-terrain: A library for creating TIN-based terrains from heightmaps. `https://github.com/heremaps/tin-terrain`, 2024.

H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 35–42, 1998. URL `https://api.semanticscholar.org/CorpusID:1231713`.

G. Karypis and V. Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

L. Kettner. Halfedge data structures. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6.1 edition, 2024. URL `https://doc.cgal.org/5.6.1/Manual/packages.html#PkgHalfedgeDS`.

R. Lario, R. Pajarola, and F. Tirado. Hyperblock-quadtin: Hyper-block quadtree based triangulated irregular networks. In *IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2003)*, pages 733–738. Citeseer, 2003.

B. C. Libraries. `graph_traits<Graph>`. `https://www.boost.org/doc/libs/1_79_0/libs/graph/doc/graph_traits.html`, 2021. Accessed: 2024-05-19.

Y. Livny, Z. Kogan, and J. El-Sana. Seamless patches for gpu-based terrain rendering. *The Visual Computer*, 25:197–208, 03 2009. doi: 10.1007/s00371-008-0214-3.

F. Losasso and H. Hoppe. Geometry clipmaps: Terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3):769–776, aug 2004. ISSN 0730-0301. doi: 10.1145/1015706.1015799. URL `https://doi.org/10.1145/1015706.1015799`.

M. Mazzei and D. Quaroni. Development of a 3d webgis application for the visualization of seismic risk on infrastructural work. *ISPRS International Journal of Geo-Information*, 11(1), 2022. ISSN 2220-9964. doi: 10.3390/ijgi11010022. URL `https://www.mdpi.com/2220-9964/11/1/22`.

R. Pajarola, M. Antonijuan, and R. Lario. Quadtin: quadtree based triangulated irregular networks. *IEEE Visualization, 2002. VIS 2002.*, pages 395–402, 2002. URL `https://api.semanticscholar.org/CorpusID:8001560`.

Plex-Earth Support. Understanding zoom level in maps and imagery, 2023. URL `https://support.plexearth.com/hc/en-us/articles/6325794324497-Understanding-Zoom-Level-in-Maps-and-Imagery`. Accessed: 2024-06-21.

M. Vaaraniemi, M. Treib, and R. Westermann. High-quality cartographic roads on high-resolution dems. *Journal of WSCG*, 19:41–48, 01 2011.

B. Veenendaal, M. A. Brovelli, and S. Li. Review of web mapping: Eras, trends and directions. *ISPRS International Journal of Geo-Information*, 6(10), 2017. ISSN 2220-9964. doi: 10.3390/ijgi6100317. URL `https://www.mdpi.com/2220-9964/6/10/317`.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, 2020. URL `https://scipy.org/`. Accessed: 2024-05-19.

Y. N. ZHANG Na, WANG Lei. A 3d douglas-peucker terrain simplification algorithm optimized using centroidal voronoi diagram. *Journal of Geo-information Science*, 24(7):1245, 2022. doi: 10.12082/dqxxkx.2020.210812. URL `https://www.dqxxkx.cn/CN/abstract/article_52063.shtml`.

X. Zheng, H. Xiong, J. Gong, and L. Yue. A morphologically preserved multi-resolution tin surface modeling and visualization method for virtual globes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 129:41–54, 2017. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2017.04.013. URL `https://www.sciencedirect.com/science/article/pii/S092427161730240X`.