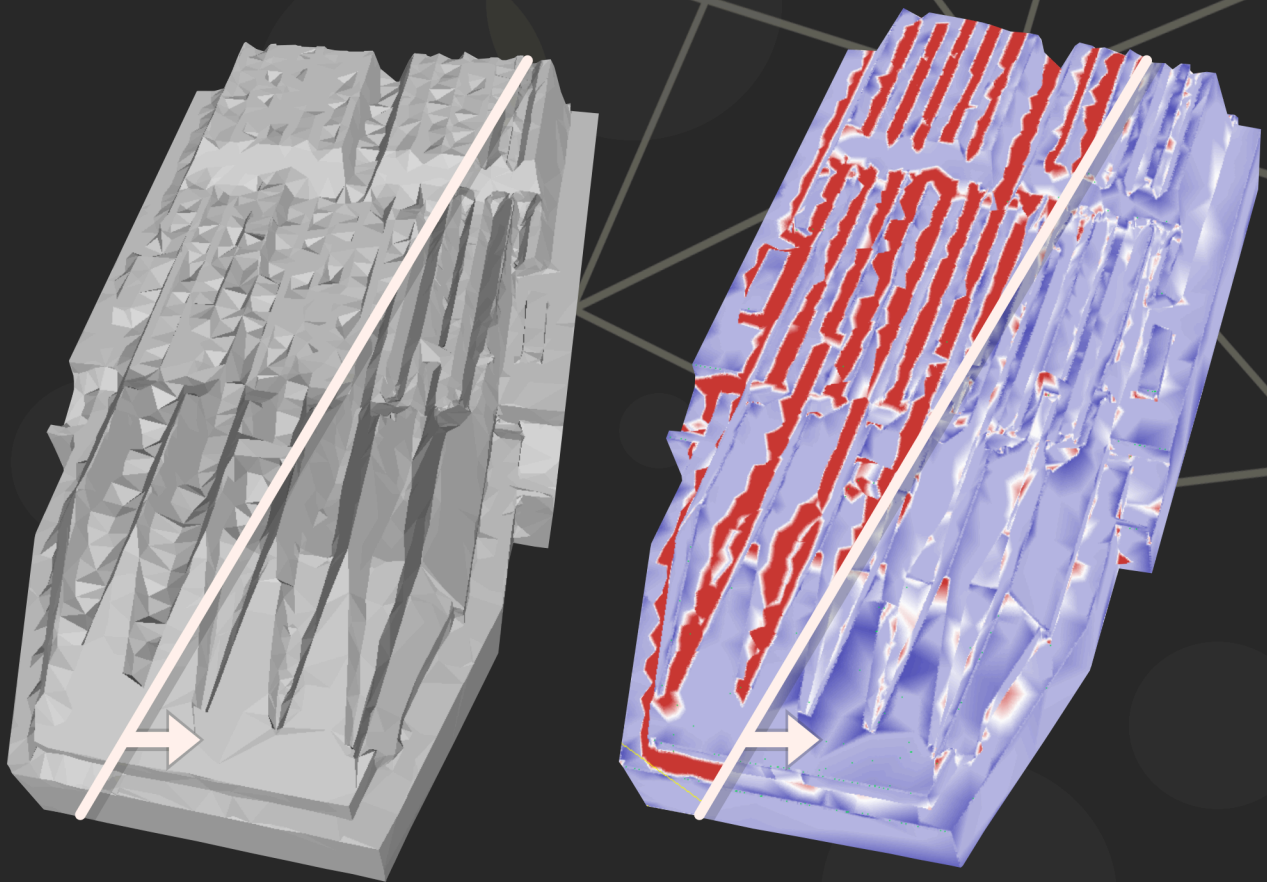


# Improving 3D alpha wrapping by preserving concave features and reducing mesh complexity

Michel Beeren - June 2026

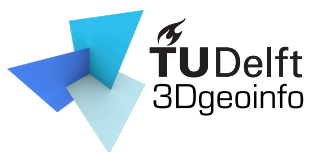


A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

Michel Beeren: *Improving 3D alpha wrapping by preserving concave features and reducing mesh complexity* (2026)

© This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

The work in this thesis was carried out in the:



3D geoinformation group  
Delft University of Technology

Supervisors: Dr. Clara García-Sánchez  
Dr. Hugo Ledoux

Co-reader: Dr. Ken Arroyo Ohori

# Acknowledgements

First of all, I would like to sincerely thank my supervisors, Clara García-Sánchez and Hugo Ledoux, for your guidance throughout this thesis. Our weekly meetings, your extensive feedback, and your continuous support have been extremely valuable. I am especially grateful that you gave me the freedom to really make this project my own, while at the same time always thinking along carefully whenever I came with questions or doubts. I did not expect to enjoy working on my thesis as much as I did, and I think your enthusiasm played a large part in that. It made the process much more enjoyable and helped me become even more enthusiastic about the topic myself.

I would also like to thank my co-reader, Ken Arroyo Ohori, and my delegate, Herman de Wolff, for your time, involvement, and for being part of the assessment of this thesis.

A big thank you goes to Anne and Julia, for showing up in the GeoLab day after day, and for making the writing process much more enjoyable. The rule that whoever arrived late had to buy coffee for the others may have been simple, but it definitely helped to keep us going.

I would also like to thank Joep and Erik for all the good food during this thesis period. I have eaten fantastically well. Joep, thank you also for the many cosy evenings, including but definitely not limited to watching Drag Race together. Erik, thank you for being a fellow Geomatics nerd, so I always had someone to happily overthink my thesis and all kinds of related topics with. Carla, thank you for all the fun walks and puzzle breaks. They were always really enjoyable, and also perfect for clearing my head a bit.

I am also grateful to Mark for the time in Finland. Despite the heavy snowstorm, the longest power outage in ten years, and the absence of running water, it was a very cosy and relaxing place to begin writing.

Finally, I would like to thank JJ and Wander for the many fun runs, bouldering sessions, and other small adventures. They made sure that, alongside working on a thesis I genuinely enjoyed, there was also plenty of other fun things.

# Contents

|                                                                                  |           |
|----------------------------------------------------------------------------------|-----------|
| <b>Part 1. General introduction</b>                                              | <b>3</b>  |
| Part 1.1 Thesis outline                                                          | 3         |
| Part 1.2 Introduction and motivation                                             | 3         |
| Part 1.3 Research objectives and scope                                           | 6         |
| <b>Part 2. Preliminary materials</b>                                             | <b>7</b>  |
| Part 2.1 Surface mesh validity and common defects                                | 7         |
| Part 2.2 Geometric basis of 3D Alpha Wrap                                        | 8         |
| Part 2.3 Why concave features are difficult to preserve in 3D Alpha Wrap         | 11        |
| Part 2.4 Design alternatives: spatially varying parameters and feature detection | 12        |
| Part 2.5 Guarantees: intuitive explanation                                       | 14        |
| Part 2.6 Evaluation metrics                                                      | 15        |
| <b>Part 3. Scientific article</b>                                                | <b>16</b> |
| <b>Part 4. Discussion</b>                                                        | <b>34</b> |
| Part 4.1 Link with MSc Geomatics for the Built Environment                       | 34        |
| Part 4.2 Reflection on the research objectives                                   | 34        |
| Part 4.3 Limitations and future work related to MSc Geomatics                    | 35        |
| <b>Bibliography</b>                                                              | <b>35</b> |
| <b>A Declaration of AI/LLM usage</b>                                             | <b>36</b> |
| <b>B Reproducibility self-assessment</b>                                         | <b>36</b> |



# Part 1. General introduction

## Part 1.1 Thesis outline

This thesis consists of four parts. Part 1 gives a general introduction to meshing, surface meshes, mesh quality, and mesh repair. It introduces the motivation of the research and explains why preserving sharp concave features while avoiding unnecessary mesh complexity is relevant for automatic 3D building modelling.

Part 2 provides the technical background needed to understand the scientific article in Part 3. It first explains surface mesh validity and common defects, because the method aims to produce valid repaired surface meshes from imperfect input geometry. It then introduces the geometric basis of 3D Alpha Wrap, including Delaunay tetrahedralization, Voronoi duality, inside/outside cell labelling, the offset surface, oracle queries, and the role of the  $\alpha$  and offset parameters. These concepts are used directly in the method described in Part 3. Part 2 also explains why concave features are difficult to preserve, discusses design alternatives such as spatially varying parameters and feature detection, gives an intuitive explanation of the guarantees that need to be preserved, and introduces the evaluation metrics used in the experiments.

Part 3 is written as a scientific article. It presents the proposed  $\tau$ -extended 3D Alpha Wrap method, including the new local refinement criterion, the adapted insertion strategy, the proof that the guarantees are preserved, and a comparison with real-world data with the original 3D Alpha Wrap algorithm. The article evaluates the methods in terms of approximation quality, output complexity, runtime, and validity.

Part 4 closes the thesis with a broader discussion. It reflects on the results of the research questions, relates the work to topics from the MSc Geomatics for the Built Environment, and discusses limitations and possible future work for built-environment applications.

## Part 1.2 Introduction and motivation

*Meshing and surface meshes.* Meshing is the process of discretizing a continuous geometry into smaller elements that can be stored, processed, visualized, or used in computations (*What Is Meshing, Really?*, n.d.). The resulting discrete representation is called a mesh. Depending on the application, 3D meshes can either be volumetric or surface-based. Volume meshes discretize a three-dimensional domain using elements such as tetrahedra, cubes, or polyhedra, and are commonly used, for example, to represent flow domains in computational fluid dynamics (Lintermann, 2021).

In this thesis, we focus on surface meshes: meshes that represent only the boundary of an object, rather than its full volume. A surface mesh consists of 2D planar elements, such as triangles, quadrilaterals, or more general polygons, embedded in 3D space. Examples are shown in Figure Figure 1.1. Throughout this thesis, the term mesh refers specifically to a surface mesh.

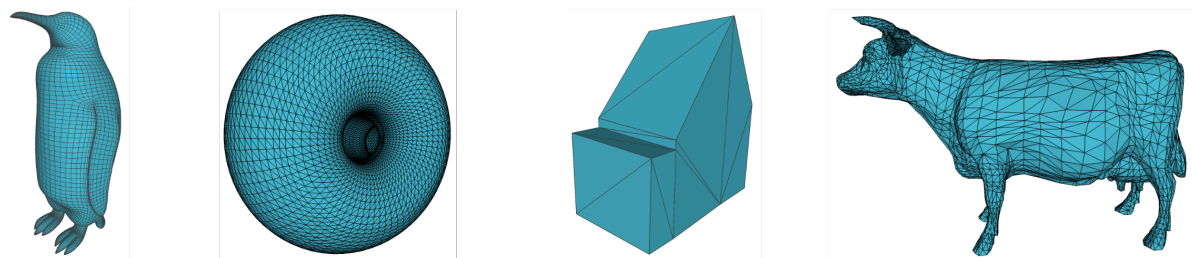


Figure 1.1: Example surface meshes.

A high-quality mesh is essential in many geometry-processing, reconstruction, and simulation workflows, but what “quality” means depends on the application (Attene et al., 2013). For visualization, it may be sufficient that the mesh looks close to the original object. For simulation, or geometric processing, stricter requirements are usually needed: the mesh should approximate the intended geometry well, be geometrically valid, and have a complexity that is appropriate for the level of detail needed (Saeedraashed & Benim, 2019). These three requirements form the focus of this thesis.

Approximation quality refers to how closely the mesh follows the intended input geometry. Relevant features such as roof shapes, façade details, sharp edges, and concave regions should not disappear or become overly rounded (Attene et al., 2013; Portaneri et al., 2022).

Geometric validity refers to whether the mesh satisfies the topological and geometric requirements needed for further processing. The ISO 19107 standard provides a spatial schema for geometric objects, including 3D solids, and is widely used in the GIS domain as a conceptual basis for defining and validating geometric validity (Arroyo Ohori et al., 2024; ISO 19107:2019, 2019; Ledoux, 2013; 2018). In practice, invalid geometries may contain gaps or missing faces, self-intersections, non-manifold edges, degenerate faces with zero area, or duplicate vertices and faces (Pađen et al., 2024).

Appropriate output complexity means that the mesh should contain enough faces to represent the relevant geometry, but not many more than needed. Adding faces can improve detail, but also increases file size, processing time, memory use, and downstream computation time (Pađen et al., 2024; Portaneri et al., 2022; Shumak, 2025). Extra faces are therefore only useful when they substantially improve the geometric approximation.

Meeting these requirements at the same time is difficult: many real-world 3D models are invalid, too coarse in detailed regions, unnecessarily dense in smooth regions, or a combination of these problems (Attene et al., 2013; Biljecki et al., 2016; Pađen et al., 2024).

*Mesh repair and reconstruction.* Mesh repair and reconstruction methods aim to transform defective or unsuitable input geometries into usable output meshes (Attene et al., 2013). Local repair methods correct specific defects directly, such as holes, non-manifold edges, or duplicated elements. This can preserve the original geometry well, but local methods often become unreliable when several types of defects occur simultaneously (Attene et al., 2013).

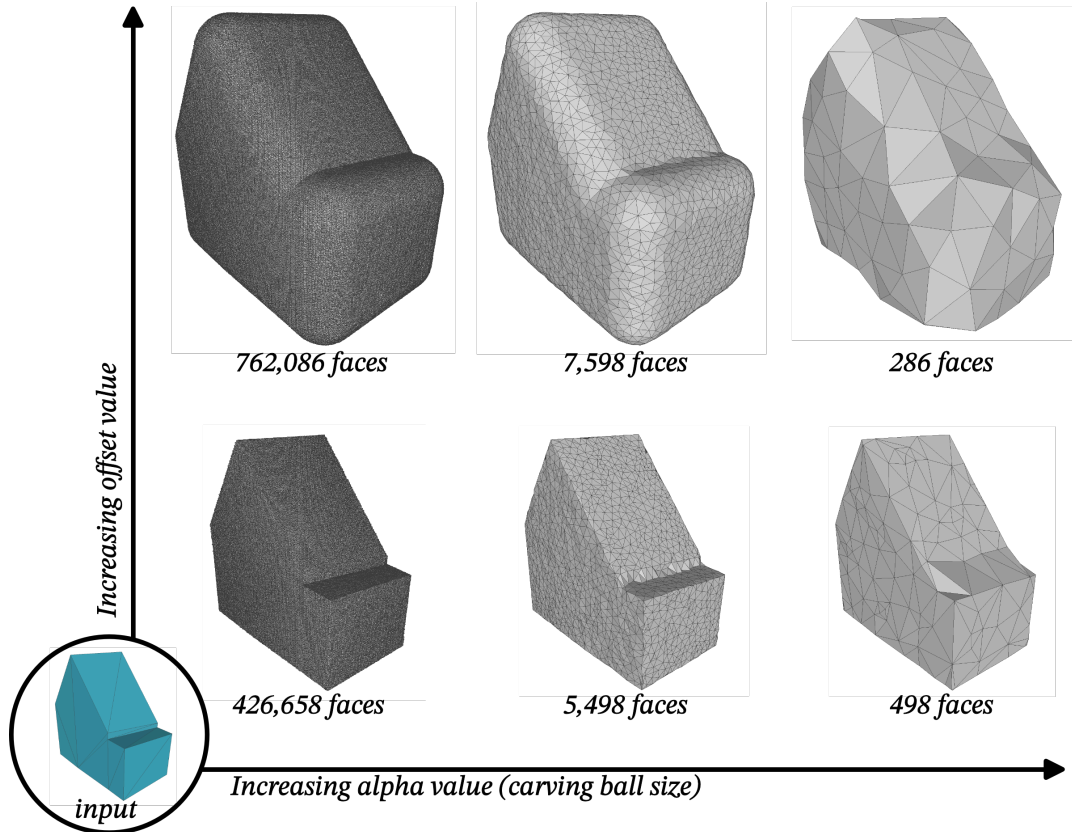
Global repair methods take a different approach: instead of fixing each error individually, they reconstruct or approximate the complete surface. This can make them more robust for defective inputs, but it can also smooth and/or remove important details or increase mesh complexity (Attene et al., 2013; Portaneri et al., 2022). Hybrid methods combine both ideas, for example by using a global reconstruction for robustness and then applying local projection or optimization to improve the geometric fit (Wen et al., 2025).

Although many mesh repair and reconstruction methods exist, each with its own strengths and limitations, there is no general “one-size-fits-all” solution (Attene et al., 2013). Substantial manual work is often still needed to obtain meshes that are both valid and sufficiently detailed for downstream use (Pađen et al., 2024; Saeedraashed & Benim, 2019).

*CGAL's 3D Alpha Wrap.* CGAL's 3D Alpha Wrap algorithm provides a robust global repair and approximation method for defective, overly complex, or otherwise difficult input geometries (Portaneri et al., 2022). The method can be understood as a shrink-wrapping procedure around an offset surface of the input. This offset surface lies at a user-defined distance from the input and determines where newly inserted vertices may be placed.

To perform the wrapping, the surrounding space is divided into tetrahedral cells using a 3D Delaunay triangulation. The algorithm starts with a loose enclosing surface mesh around the input. It then gradually moves this mesh inward towards the input geometry by “carving away” tetrahedral cells from the outside. The carving process is controlled by the parameter  $\alpha$ , which defines the radius of the carving ball. Smaller  $\alpha$ -values allow the wrap to pass through smaller cavities and openings, making the output follow the input more closely.

When a cell cannot be carved without crossing the input, the triangulation is refined instead. This is done by inserting an additional vertex on the offset surface, which locally refines both the Delaunay triangulation and the current surface mesh. The algorithm repeats these carving and refinement steps until the surface mesh can no longer move further inward according to the  $\alpha$ -criterion. The final output is guaranteed to strictly enclose the input and to be watertight, orientable, and manifold; the algorithm is also guaranteed to terminate (Portaneri et al., 2022). At the same time, the  $\alpha$  and offset parameters give the user control over the trade-off between approximation quality and output complexity: smaller values can improve the geometric fit, but usually lead to a denser output mesh. This trade-off is illustrated in the Figure 1.2, which shows how different combinations of  $\alpha$  and offset affect the wrapped output.



**Figure 1.2:** Effect of  $\alpha$  and offset on the 3D Alpha Wrap output. Smaller  $\alpha$ -values improve the approximation of concave regions but usually increase mesh density, while smaller offset values reduce rounding near convex features.

However, two limitations remain. First, capturing small features typically requires a small  $\alpha$ -value. This can improve the approximation of detailed regions, but since  $\alpha$  is global, the same small value also applies to smooth or nearly planar regions. As a result, the number of output faces increases throughout the mesh, including in regions where the additional faces do not substantially improve the approximation.

Second, sharp concave features tend to be rounded. The  $\alpha$ -ball cannot reach sharp concave regions as effectively as convex regions, because the wrap is constructed from the outside. A smaller  $\alpha$ -value allows the algorithm to carve deeper into concave regions, but sharp concave features may still remain rounded.

These limitations are especially relevant for inputs that combine large planar areas with sharp concave features, such as detailed building models used in flow simulation workflows, where geometric details and sharp edges can influence the simulation results (Paden et al., 2024). For such inputs, decreasing  $\alpha$  is not an ideal solution: it increases mesh complexity throughout the output, while sharp concave features may still remain rounded. This motivates a method that improves the approximation of sharp concave features without increasing mesh complexity globally.

## Part 1.3 Research objectives and scope

### Research objectives.

The main research question of this thesis is:

**Main:** *How can the 3D Alpha Wrap algorithm be extended to better preserve sharp concave features and reduce unnecessary mesh complexity, while retaining its guarantees?*

This main question is divided into the following sub-questions:

**RQ1:** *Can local refinement be applied and optimized to better preserve sharp concave features without increasing mesh complexity globally?*

**RQ2:** *How can the theoretical guarantees of the original 3D Alpha Wrap algorithm, namely termination, strict enclosure, watertightness, and manifoldness, be retained?*

**RQ3:** *Can the approach be controlled with a small and interpretable set of parameters?*

**RQ4:** *To what extent does the developed approach improve the trade-off between approximation quality and output complexity compared to the original algorithm, and how does this affect runtime and validity?*

### Scope.

This thesis focuses on extending the 3D Alpha Wrap algorithm within its existing carving-based framework. The goal is not to build a separate repair pipeline or user-facing tool, but to modify the algorithm itself while preserving its original guarantees. The implementation is developed as a prototype within the existing CGAL Alpha Wrap codebase and reuses the geometric query functions already available for the input geometry and offset surface.

The work focuses on surface mesh repair and approximation, with the goal of improving the preservation of sharp concave features while reducing unnecessary mesh complexity. The extension should remain controllable through a small and interpretable parameter set, so that the user can influence to what extent sharp concave features are refined. The method is evaluated on 3D BAG building data and compared to the original 3D Alpha Wrap algorithm in terms of approximation quality, output complexity, runtime, and validity.

The thesis does not aim to solve general hole filling, semantic reconstruction, post-processing simplification, or input cleaning. As in the original Alpha Wrap algorithm, large holes can cause internal structures to be included in the wrapped output (Portaneri et al., 2022). Because the output strictly encloses the input, significant outliers can also appear in the final wrap. The input itself may be defective, unoriented, or represented as a point cloud, triangle soup, or triangle mesh.

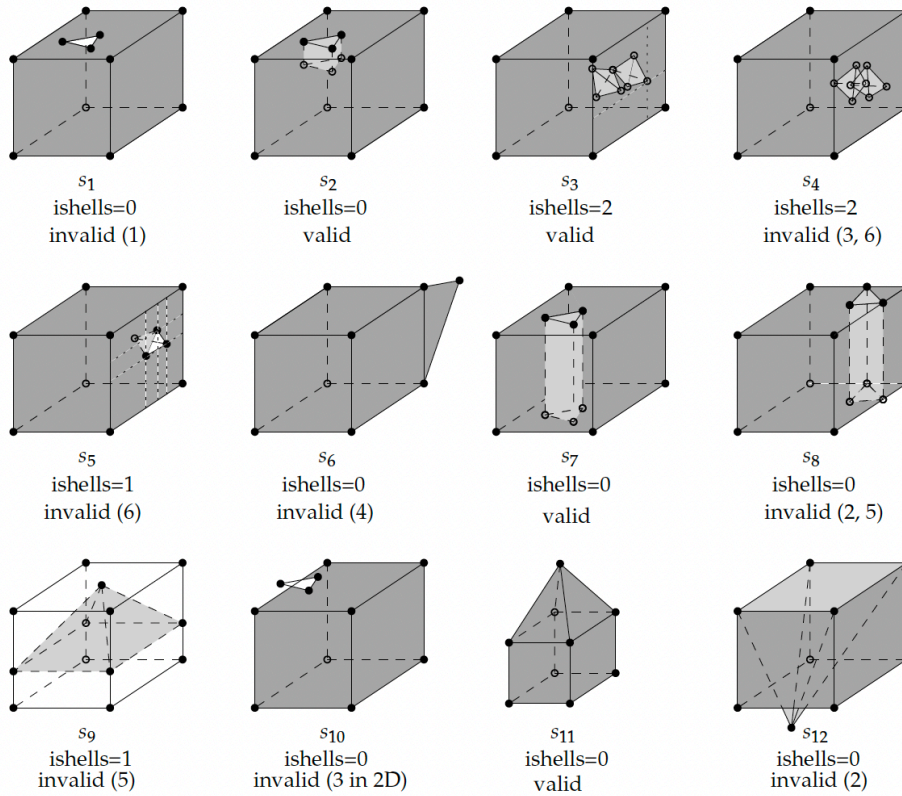
## Part 2. Preliminary materials

This part provides the preliminary materials needed to read the scientific article in Part 3. It introduces the main concepts used in the method and experiments, including surface mesh validity, Delaunay-based wrapping, 3D Alpha Wrap, the proposed tau-criterion, and the evaluation metrics.

### Part 2.1 Surface mesh validity and common defects

In this thesis, the output is a triangle surface mesh, consisting of vertices, edges, and triangular faces. In the context of 3D Alpha Wrap and Delaunay triangulations, these triangular faces are often called *facets*.

For many applications, a mesh should not only look like the intended object, but also be geometrically valid. For a 3D object to be valid, it must describe a well-defined solid. In the GIS domain, this is commonly related to ISO 19107 validity requirements, which consider both the geometry and topology of the object (Arroyo Ohori et al., 2024; ISO 19107:2019, 2019; Ledoux, 2013; 2018). A valid solid should be closed, should have simple 2-manifold shells, and should have a consistent interior and exterior. Figure 2.1 illustrates that invalidity can arise in several ways, such as holes in the boundary, dangling parts, invalid shell intersections, disconnected interiors, or geometrically invalid surfaces.



**Figure 2.1:** Examples of valid and invalid solids, adapted from (Arroyo Ohori et al., 2024). Solids  $s_2$ ,  $s_3$ ,  $s_7$ , and  $s_{11}$  are valid; the others violate one or more validity conditions. Solid  $s_1$  is not watertight. Solid  $s_4$  has intersecting interior shells, violating the shell-intersection and cavity conditions. Solid  $s_5$  is invalid because its cavity shares a surface with the exterior shell. Solid  $s_6$  contains a dangling part. Solid  $s_8$  has a non-simple exterior shell and a disconnected interior. Solid  $s_9$  has an interior shell that splits the solid into disconnected interior volumes. Solid  $s_{10}$  contains an invalid polygonal surface. Solid  $s_{12}$  is combinatorially similar to  $s_{11}$ , but geometrically invalid because its exterior shell is not simple. The numbers in parentheses refer to the violated OGC validity assertions.

3D Alpha Wrap generates an enclosing outer surface mesh around the input. If the input contains a sufficiently large hole, the wrap may still enter the object and include internal structures in the output. However, these structures then become part of the generated exterior shell, comparable to the valid tunnel-like solid  $s_7$  in Figure 2.1, rather than separately modelled interior shells. Therefore, validity issues related specifically to the correct modeling of interior shells or cavities are not the main focus here.



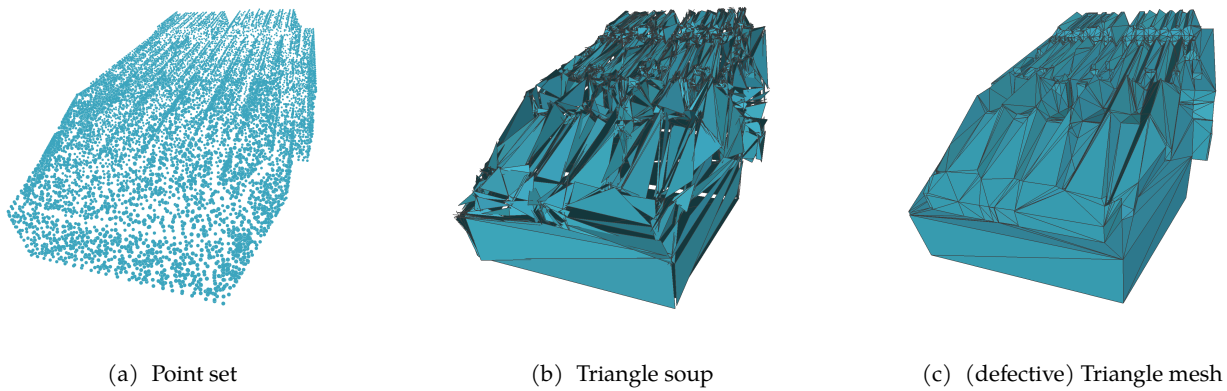
For this reason, the most relevant validity concepts in this thesis are watertightness, manifoldness, and orientability. These properties describe whether the generated wrapping surface can be interpreted as a clean and well-defined boundary. A watertight mesh forms a closed boundary without holes. A manifold mesh behaves locally like a clean surface: faces connect consistently along edges, without branching or dangling parts. Orientability means that the faces can be oriented consistently, so that the mesh has a clear inside and outside.

In addition to these validity-related properties, 3D Alpha Wrap also guarantees strict enclosure of the input (Portaneri et al., 2022). This means that the generated mesh lies around the input geometry and does not cut through it. Although strict enclosure is not a validity requirement by itself, it is important in the Alpha Wrap algorithm and is used in the reasoning behind the other guarantees.

Real-world input meshes often violate these requirements. They may contain holes or missing faces, self-intersections, non-manifold edges, degenerate faces with zero area, duplicate vertices or faces, or inconsistent face orientations (Pađen et al., 2024). Such defects can prevent the mesh from being interpreted as a valid surface or solid, which is why robust mesh repair methods are needed.

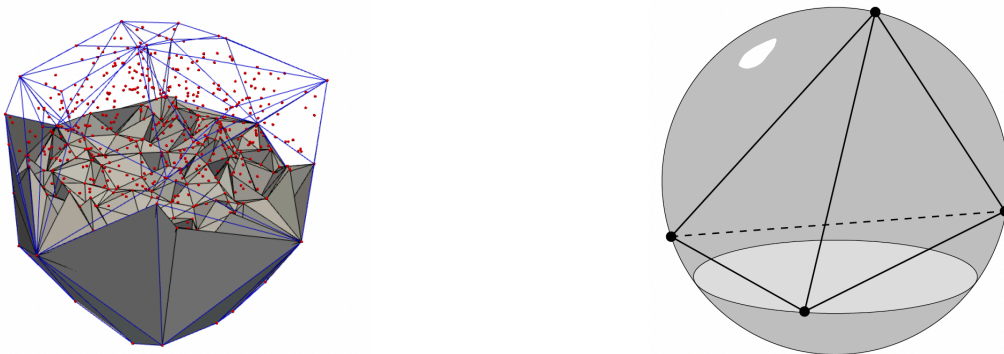
## Part 2.2 Geometric basis of 3D Alpha Wrap

CGAL's 3D Alpha Wrap algorithm is a global repair and approximation method that generates a watertight, orientable, manifold surface mesh that strictly encloses the input (Portaneri et al., 2022). The input can be represented as a point set (see Figure 2.2a), triangle soup (see Figure 2.2b), (defective) triangle mesh (see Figure 2.2c), or a mixed input. The method can be understood as a shrink-wrapping procedure around such input geometry.



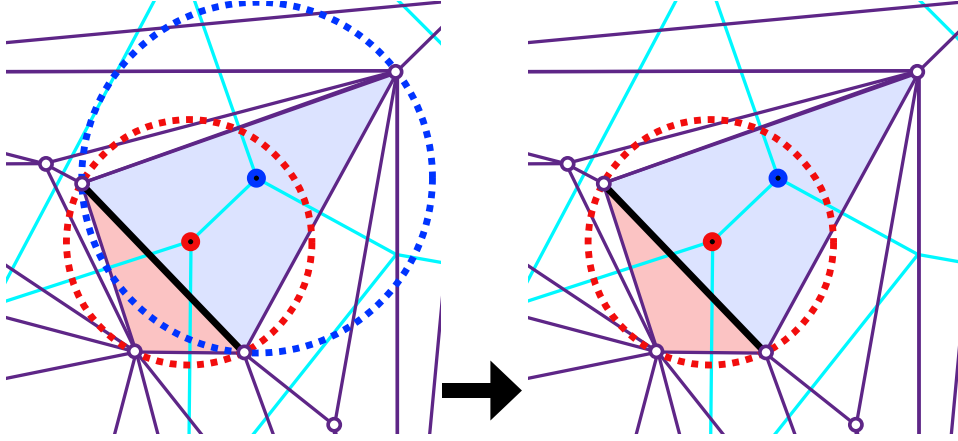
**Figure 2.2:** Possible input types of 3D Alpha Wrap.

Rather than directly deforming an existing surface, 3D Alpha Wrap constructs the wrap through a subdivision of the surrounding space. This space is represented by a 3D Delaunay tetrahedralization, which divides it into non-overlapping tetrahedra (see Figure 2.3a). These tetrahedra are formed by vertices of the input. In the Alpha Wrap algorithm, these tetrahedra are referred to as cells, and their triangular faces are referred to as facets. A key property of the Delaunay tetrahedralization is that each tetrahedral cell has an empty circumsphere (see Figure 2.3b): no other triangulation vertex lies inside this sphere (Arroyo Ohori et al., 2024).

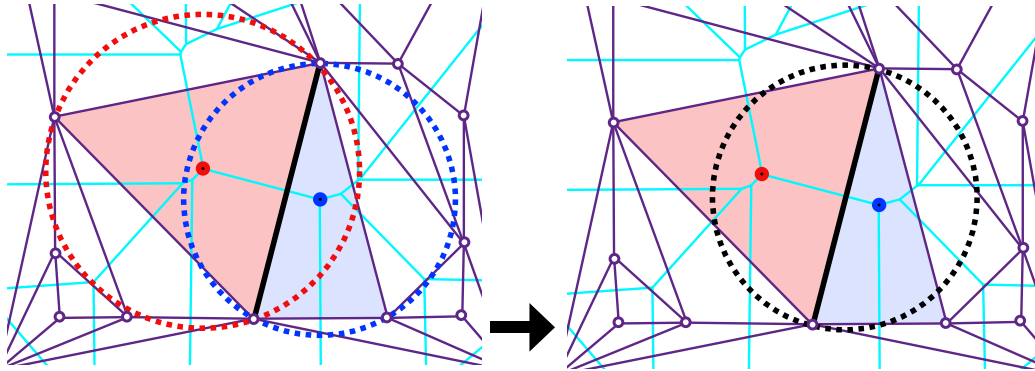


**Figure 2.3:** Tetrahedral space division with Delaunay properties.

In 3D Alpha Wrap, however, traversability is tested per surface facet rather than per tetrahedral cell. For this, the algorithm determines the minimum empty Delaunay ball associated with the facet. A surface facet is only processed further when this minimum empty Delaunay ball is larger than the  $\alpha$ -ball. If the circumcenters of the two neighboring cells lie on the same side of the facet, the minimum empty Delaunay ball is the circumsphere of the neighboring cell with the smaller circumradius (see Figure 2.4). If the circumcenters lie on opposite sides of the facet, the minimum empty Delaunay ball is the sphere through the three vertices of the facet itself (see Figure 2.5). The radius of this minimum empty Delaunay ball determines whether the  $\alpha$ -ball can pass through the facet, and therefore whether the wrapping surface can continue moving inward at that location.

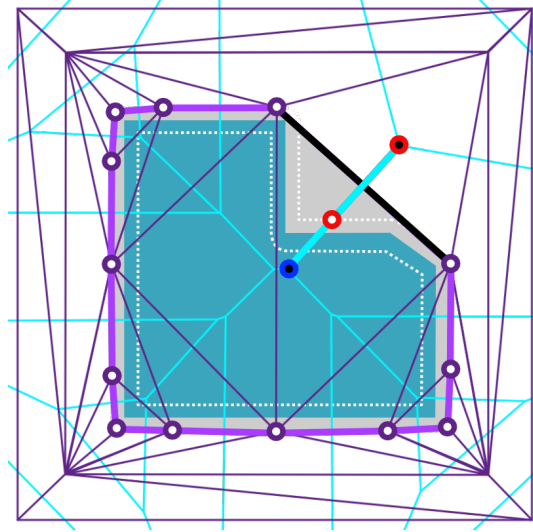


**Figure 2.4:** Schematic 2D illustration of the minimum empty Delaunay ball of a facet when the circumcenters of the two neighboring cells lie on the same side of the facet. In this case, the minimum empty Delaunay ball is the circumsphere of the neighboring cell with the smaller circumradius. The black line denotes the facet, the Delaunay triangulation is shown in purple, the Voronoi diagram in light blue, and the neighboring cells are highlighted in red and blue.



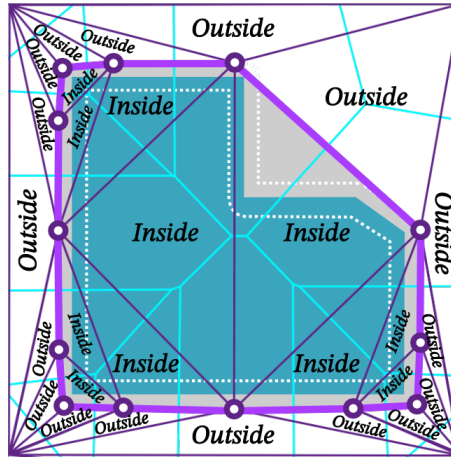
**Figure 2.5:** Schematic 2D illustration of the minimum empty Delaunay ball of a facet when the circumcenters of the two neighboring cells lie on opposite sides of the facet. In this case, the minimum empty Delaunay ball is the sphere defined by the three vertices of the facet itself. The black line denotes the facet, the Delaunay triangulation is shown in purple, the Voronoi diagram in light blue, and the neighboring cells are highlighted in red and blue.

The 3D Voronoi diagram partitions space according to the nearest triangulation vertex: each Voronoi cell contains the locations that are closer to one vertex than to all others (Arroyo Ohori et al., 2024). It is dual to the Delaunay tetrahedralization, meaning that corresponding elements exist in both structures. For example, a Delaunay facet shared by two cells has a dual Voronoi edge connecting the circumcenters of those cells. This edge is used in the original Alpha Wrap insertion rule R1, where a new vertex, called a Steiner point, is placed at its intersection with the offset surface (see Figure 2.6) (Portaneri et al., 2022).



**Figure 2.6:** Schematic 2D illustration of the original R1 insertion rule. The dual Voronoi edge of the processed surface facet intersects the offset surface, and a Steiner point is inserted at this intersection to locally refine the wrap.

The algorithm maintains inside and outside labels for the Delaunay cells. These labels are part of the wrapping process and do not require the input itself to be a valid solid. The current wrap is defined by the facets that separate inside cells from outside cells (see Figure 2.7). At the start, this surface is the boundary of a loose bounding box around the input. During carving, cells are relabelled from inside to outside, causing the wrapping surface to move inward towards the input.



**Figure 2.7:** Schematic 2D illustration of the inside/outside cell labelling used in 3D Alpha Wrap. The current wrapping surface is defined by the facets separating inside cells from outside cells. During carving, cells are relabelled from inside to outside, which moves the surface inward towards the input.

The offset surface can be understood as an implicit surface around the input. In implicit geometry, a surface does not have to be stored explicitly as a mesh, but can instead be defined indirectly by a condition or function (Arroyo Ohori et al., 2024). In this case, the offset surface consists of all locations whose distance to the input is equal to the offset distance  $\delta$ . New vertices inserted during refinement are placed on this offset surface rather than directly on the input. This gives the algorithm control over the distance between inserted vertices and the input, while preserving the strictly enclosing nature of the wrap (Portaneri et al., 2022).

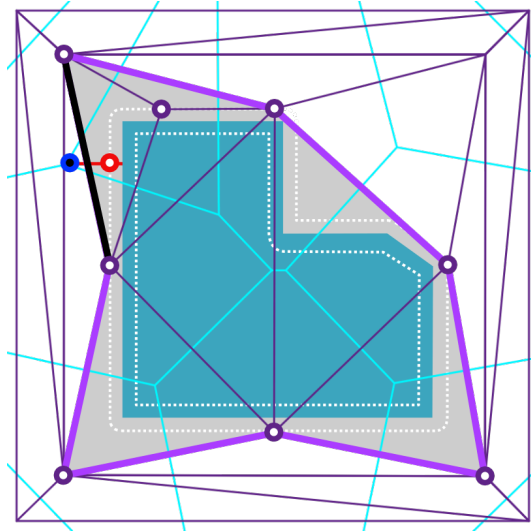
CGAL uses an abstract interface, called an oracle, to make 3D Alpha Wrap independent of the input representation. From the perspective of the wrapping algorithm, the oracle provides exactly three geometric queries (Portaneri et al., 2022):

1. a tetrahedron-input intersection test
2. a segment-offset surface intersection
3. a projection of a query point onto the input



Internally, these queries may involve different geometric primitives of the input, namely points, line segments, and triangle interiors. For example, even when the input is a triangle mesh, all three primitive types can contribute to the same projection, distance, or intersection query.

Each query has a specific role in the original algorithm. The tetrahedron-input intersection test is used to decide whether carving a cell is allowed. The segment-offset surface intersection is used in Rule R1 to place a Steiner point where the dual Voronoi edge intersects the offset surface. The projection query is used in Rule R2 to construct an alternative Steiner point (see Figure 2.8). Since the wrapping algorithm only depends on the oracle interface, the same procedure can be applied to point sets, segment and triangle soups, triangle meshes, and mixed inputs (Portaneri et al., 2022).

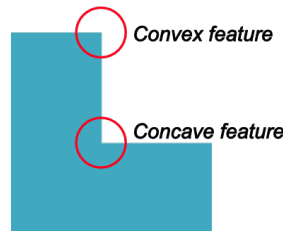


**Figure 2.8:** Schematic 2D illustration of the original R2 insertion rule. When R1 cannot be applied, the adjacent cell circumcenter is projected onto the input, and a Steiner point is inserted on the offset surface along this projection direction to locally refine the wrap.

The segment-offset intersection is more expensive than the other two queries because the offset surface is implicit. The algorithm does not intersect the segment with an explicitly meshed offset surface. Instead, it searches along the segment for the location where the distance to the input equals the offset value. This is done by dichotomy, which requires repeated distance queries to the input (Portaneri et al., 2022).

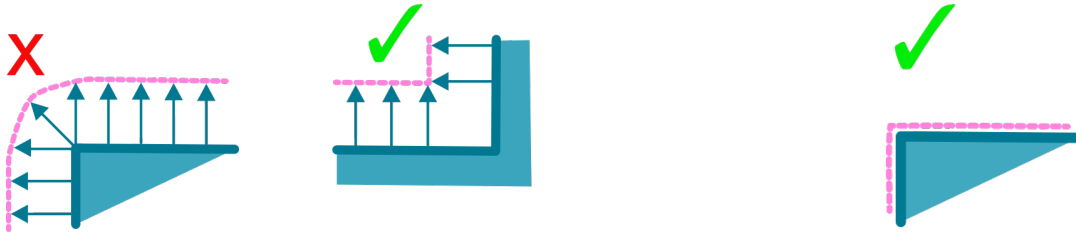
### Part 2.3 Why concave features are difficult to preserve in 3D Alpha Wrap

Sharp features can be related to the local curvature of a surface. Curvature describes how the gradient of a surface changes, and can be classified as convex, concave, or flat (Ledoux et al., 2025). In this thesis, this distinction is applied to surface meshes: a convex feature bends outward with respect to the exterior of the object, while a concave feature bends inward (see Figure 2.9).



**Figure 2.9:** Schematic illustration of convex and concave features. Convex features bend outward, while concave features bend inward with respect to the exterior of the object.

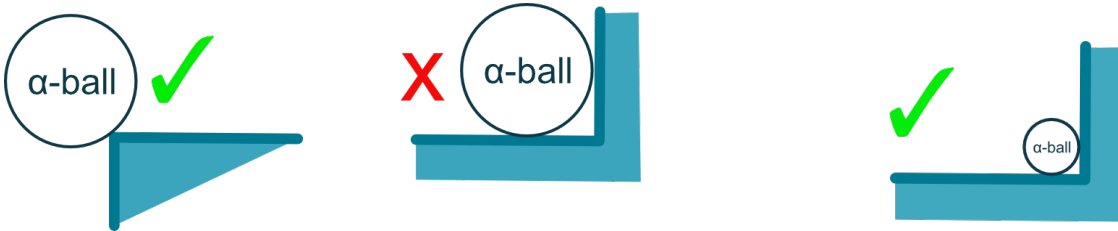
The offset parameter  $\delta$  mainly controls the distance between inserted vertices and the input. Because these vertices are placed on the offset surface, a large offset can round convex corners (see Figure 2.10a). However, this effect can usually be reduced by choosing a smaller offset value, so that the offset surface lies closer to the input (see Figure 2.10b). Convex features can often be preserved more sharply without requiring a much denser mesh.



(a) Effect of the offset surface: concave features remain sharp on the offset surface, while convex features are rounded. (b) A smaller offset distance  $\delta$  keeps the offset surface closer to convex features.

**Figure 2.10:** Schematic 2D illustration of the influence of the offset distance  $\delta$  on the implicit offset surface. Concave features remain sharp on the offset surface, whereas convex features are rounded. This rounding effect can be reduced by choosing a smaller  $\delta$ , so that the offset surface lies closer to the input geometry.

Concave features preservation is mainly controlled by  $\alpha$ . Since 3D Alpha Wrap constructs an outer wrap, convex features can be reached from the outside more directly (see Figure 2.11a), while concave features require the  $\alpha$ -ball to enter inward-facing regions. If  $\alpha$  is too large, the wrap stops too early and the concave feature becomes rounded. Decreasing  $\alpha$  allows the wrap to reach further into narrow concave regions (see Figure 2.11b) (Portaneri et al., 2022).



(a) A  $\alpha$ -ball can reach convex features more easily than concave features. (b) A smaller  $\alpha$ -ball can reach deeper into concave regions, compared to a larger  $\alpha$ -ball.

**Figure 2.11:** Schematic 2D illustration of the influence of  $\alpha$ . Convex features can be reached more easily by the  $\alpha$ -ball, whereas concave features are harder to access from the outside. Reducing  $\alpha$  allows the wrap to move further into narrow concave regions, but because  $\alpha$  is a global parameter, this also increases refinement elsewhere in the mesh.

However,  $\alpha$  is a global parameter. A smaller  $\alpha$ -value therefore increases refinement throughout the mesh, including in large planar regions where extra faces may not improve the approximation.

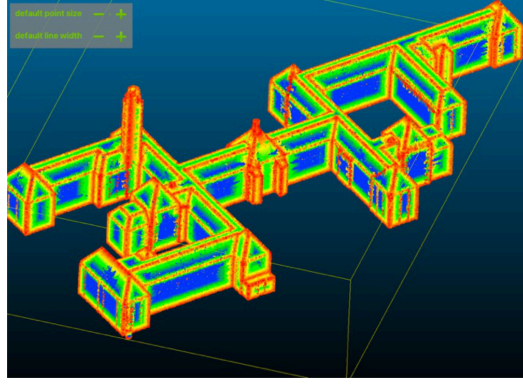
The combined influence of  $\alpha$  and offset is illustrated in Figure 1.2, which shows wrapped outputs generated with varying values of both parameters.

## Part 2.4 Design alternatives: spatially varying parameters and feature detection

One possible way to preserve sharp concave features without increasing the output complexity globally is to make the  $\alpha$ -parameter spatially varying. In such an approach, a smaller  $\alpha$ -value could be used near small or sharp concave features, while a larger  $\alpha$ -value could be used in flat or low-detail regions. This would allow the wrap to enter narrow concave regions where needed, without refining the entire mesh. This idea is also discussed by (Portaneri et al., 2022) as a possible direction for future work. However, they note that automatically defining such a spatially varying  $\alpha$ -field is difficult, because it requires understanding where the input is flat, where cavities are present, and where sharp concave features should be preserved.

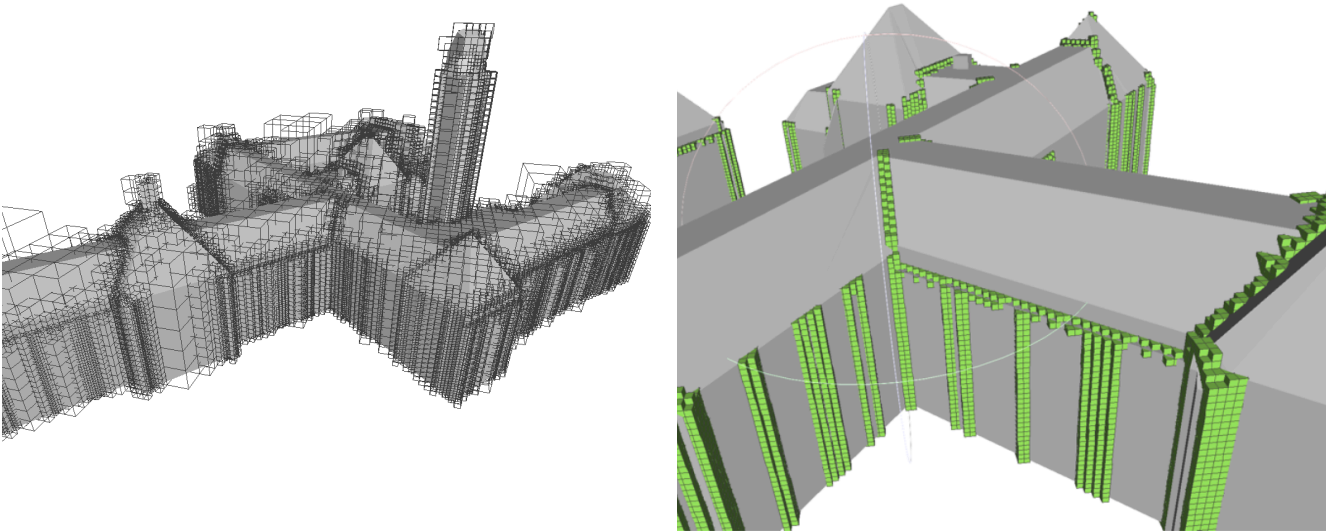
To explore this idea, two preliminary tests were performed during this thesis. Both tests aimed to obtain information about where local refinement might be useful, and to use this information to assign spatially varying alpha-values. The first test used the Medial Axis Transform to estimate a local feature-size field. The second test used adaptive octree refinement to detect candidate concave-feature regions.

The Medial Axis Transform represents a shape by a skeleton-like structure. The radii of medial balls can be used as an indication of local feature size: smaller values often indicate narrow regions or nearby surface features (Peters & Ledoux, 2016). In the explored approach, the input was sampled and an approximate feature-size field was computed using the MAT (see Figure 2.12). This field was then used to assign spatially varying  $\alpha$ -values during wrapping.



**Figure 2.12:** Approximate feature-size field derived from the Medial Axis Transform (MAT). Each sampled input point is assigned the radius of its corresponding medial ball.

Adaptive octree refinement takes a different approach. An octree recursively subdivides the space around the input into smaller axis-aligned cells. This makes it possible to create a spatially adaptive representation: regions that contain more geometric detail can be subdivided further, while simpler regions remain coarse (Bischoff et al., 2005; Zint et al., 2023). In the explored approach, a cell was split when the input planes inside the cell indicated the presence of a sharp concave feature, based on the angle between their normals. This process was repeated until a maximum refinement depth was reached (see Figure 2.13a). The smallest cells produced by this refinement were then used as candidate concave-feature locations (see Figure 2.13b), for the spatially varying  $\alpha$ -value.

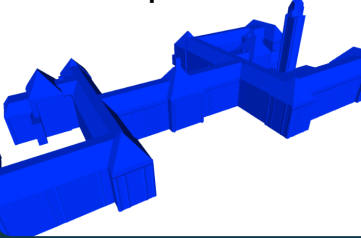
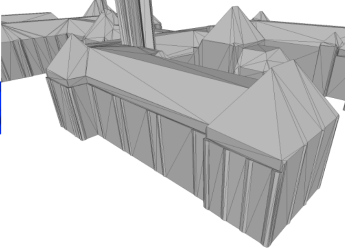
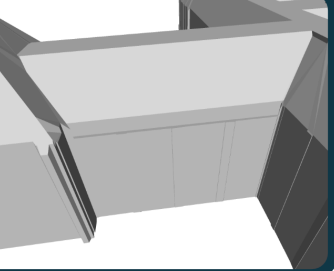
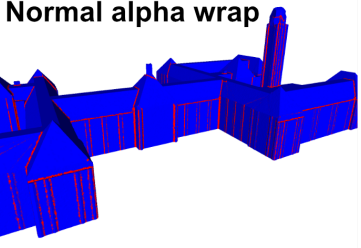
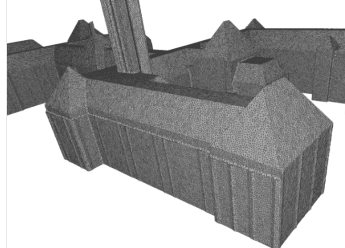
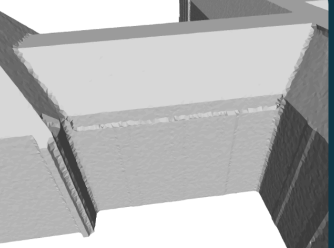
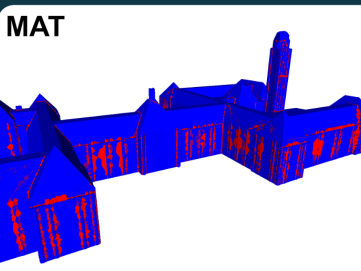
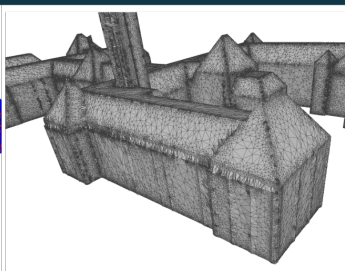
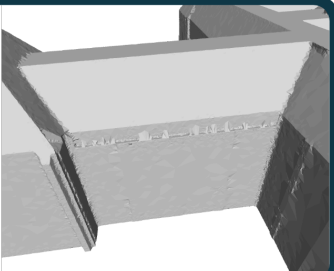
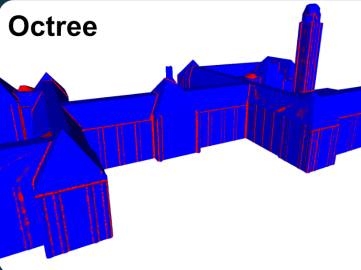
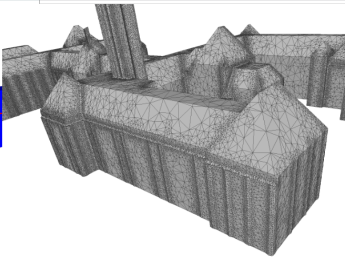
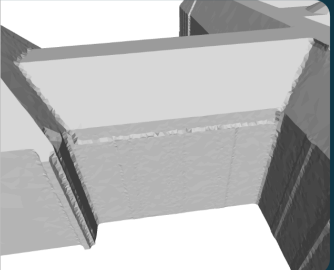


(a) Adaptive octree subdivision around the input geometry, with smaller cells near detected concave features. (b) Octree cells at the maximum refinement depth, used as candidate concave-feature locations.

**Figure 2.13:** Adaptive octree refinement used to identify candidate concave-feature regions. The full octree subdivision shows where the space around the input is refined, while the cells at the maximum refinement depth indicate the proximity of a concave feature and will be used to set the local varying  $\alpha$ .

Feature-detection-based methods can help define a spatially varying  $\alpha$ -value and can improve the output locally, as shown in Figure 2.14. However, the result depends on how well the relevant features are detected. The adaptive octree approach was able to identify many concave regions, but it requires additional input information, such as plane normals, and assumes that the input can be interpreted as planar regions. The MAT-based approach is more generic after sampling the input, but missed several small features.

For this reason, the final method avoids explicit feature detection. Instead, it stays within the original 3D Alpha Wrap framework and uses only the three oracle queries already implemented by CGAL: tetrahedron-input intersection, segment-offset surface intersection, and point projection onto the input (Portaneri et al., 2022), as discussed in Section Part 2.2.

| Distance to input                                                                                             | Triangle size                                                                       | Info                                                              | Visualization                                                                         |
|---------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| <b>Red</b> = triangle far from input<br><b>Blue</b> = triangle close to input                                 | Edges shown in black -><br>corresponds to local alpha size                          |                                                                   |                                                                                       |
| <b>3D BAG input</b><br>      |    | Filesize = 0.18 MB<br>Wrap t = ---<br>Valid? = <b>NO!</b>         |    |
| <b>Normal alpha wrap</b><br> |    | Filesize = 49.5 MB<br>Wrap t = 32.9 sec.<br>Valid? = <b>YES!</b>  |    |
| <b>MAT</b><br>              |   | Filesize = 54.8 MB<br>Wrap t = 100.5 sec.<br>Valid? = <b>YES!</b> |   |
| <b>Octree</b><br>          |  | Filesize = 16.0 MB<br>Wrap t = 15.4 sec.<br>Valid? = <b>YES!</b>  |  |

**Figure 2.14:** Comparison of output meshes generated with the original 3D Alpha Wrap algorithm and with spatially varying  $\alpha$ -values based on MAT and adaptive octree feature detection. All output meshes are valid. The MAT-based approach misses several small features. The adaptive octree approach preserves a proximity to the input comparable to the original algorithm, but with fewer output faces. However, the octree approach requires additional input information, such as plane normals, which makes it less general than the original oracle-based 3D Alpha Wrap framework.

## Part 2.5 Guarantees: intuitive explanation

One of the main strengths of 3D Alpha Wrap is that the algorithm does not only approximate the input, but also provides guarantees, as discussed in Section Part 2.2.

Most of these guarantees are inherited directly from the original algorithm. The proposed extension does not modify the parts of the algorithm on which strict enclosure, watertightness, orientability, and manifoldness are based. The wrapping surface is still constructed from the same inside/outside cell structure, and carving is still restricted by the same input-intersection tests.

Termination does require a separate discussion. The original termination proof relies on the way facets are selected for refinement and on where new Steiner points are inserted. Since both of these aspects are adapted in the

proposed method, the original proof cannot be used directly. Instead, the same underlying properties have to be shown again under the modified conditions.

The first property is that a newly inserted Steiner point must refine the intended local region. If a point were inserted somewhere else, the algorithm could keep trying to refine the same facet without actually changing the local configuration. The proof therefore has to show that each insertion affects the region that triggered the refinement.

The second property is that the algorithm cannot insert infinitely many points. This is shown by proving a positive lower bound on the insertion radius, which is the distance between a newly inserted Steiner point and the previously inserted vertices. If every new point must stay at least some minimum distance away from all previous points, then only finitely many points can be inserted on the finite offset surface.

Together, these two properties ensure that refinement makes local progress and cannot continue indefinitely. This preserves the termination guarantee for the extended algorithm.

## **Part 2.6 Evaluation metrics**

Approximation quality is evaluated using a directed Chamfer distance from the output mesh to the input geometry. Points are sampled on the output mesh, and for each sampled point the shortest distance to the input is computed. The average of these distances gives an indication of how closely the generated wrap follows the input.

A directed distance is used instead of a symmetric Chamfer distance. A symmetric Chamfer distance would also measure distances from the input to the output. This is less suitable here, because the input may contain internal structures or defective regions that are intentionally not reconstructed by the outer wrap. Measuring only from the output to the input therefore better reflects the approximation quality of the generated wrapping surface.

In addition to the average directed Chamfer distance, cumulative distance distributions are used to provide a more local view of the approximation error. These show what percentage of sampled output points lies within a certain distance from the input. This makes it possible to see whether larger deviations occur only locally or across a larger part of the output mesh. As before, distances are measured only from the output to the input, so that internal structures that are intentionally excluded from the wrap do not influence the evaluation. The cumulative plots are visualized as empirical cumulative distribution functions, where the y-value gives the probability that an observation is below a given x-value (The Matplotlib Development Team, 2026).

## **Part 3. Scientific article**

This page was intentionally left blank



# Improving 3D alpha wrapping by preserving concave features and reducing mesh complexity

Michel Beeren<sup>a,1</sup>, Clara García-Sánchez<sup>a</sup>, Hugo Ledoux<sup>a</sup>

<sup>a</sup> 3D Geoinformation Research Group, Delft University of Technology, Delft, The Netherlands

## Abstract

CGAL's 3D Alpha Wrap robustly produces valid enclosing meshes for defective input using two parameters:  $\alpha$ , which controls the carving-ball size, and  $\delta$ , an offset that controls the distance between the input and the output surface vertices. However, preserving sharp concave features often requires a small global  $\alpha$ -value, which also increases refinement in regions where additional vertices do not substantially improve the approximation. This paper proposes a  $\tau$ -guided extension that adds a local face-to-input distance criterion to the original algorithm. The new parameter  $\tau$  triggers additional refinement only where the current wrap remains locally too far from the input.

To make this refinement effective near concave features, we introduce R3 as an additional vertex-insertion strategy alongside the two insertion strategies of the original algorithm. R3 aims to place new vertices more directly near sharp concave features, whereas the original two insertion strategies may reach comparable vertex positions only after repeated refinement. The extension preserves the carving-based structure of the original algorithm, and we prove that the guarantees of termination, strict enclosure, watertightness, and manifoldness still hold. Experiments on 3D BAG building data show that the proposed method improves both average and local approximation quality, achieving lower directed Chamfer distances with fewer output faces than the original algorithm. Although R3 is less time efficient per insertion, it places vertices more effectively near concave features, so fewer insertions are needed to reach lower approximation errors. As a result, the proposed method can reduce runtime in the tested use case.

**Keywords:** 3D Alpha Wrap; mesh repair; mesh approximation; concave feature preservation; local refinement; geometric validity; computational geometry

## 1. Introduction

Surface meshes are essential in many geometry-processing, reconstruction, and simulation workflows (Portaneri et al., 2022). For such applications, the input mesh should not only approximate the intended geometry accurately, but also be valid: watertight, orientable, and manifold. In practice, however, existing models often contain defects or unnecessary complexity. Manually repairing such meshes can be time-consuming, while automatically repairing arbitrary combinations of defects remains challenging (Attene et al., 2013; Pađen et al., 2024; Saeedraashed & Benim, 2019).

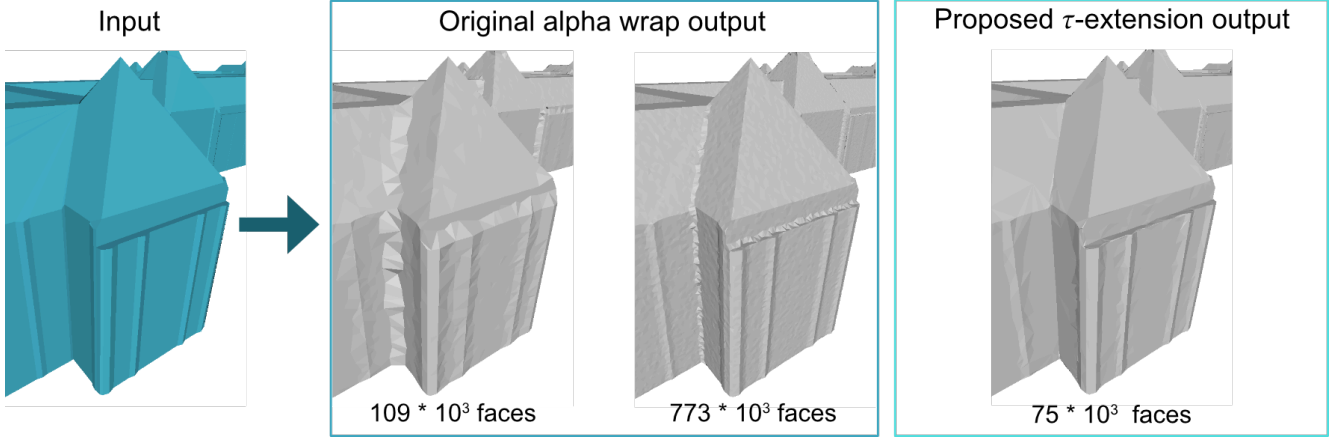
CGAL's 3D Alpha Wrap algorithm provides a robust global repair and approximation method for such inputs (Portaneri et al., 2022). The method wraps the input by carving a 3D Delaunay triangulation around an offset surface of the input. This carving process is controlled by the parameter  $\alpha$ , which defines the radius of the  $\alpha$ -ball: smaller  $\alpha$ -values allow the wrap to pass through smaller cavities and openings. When carving is not possible without exposing the input, the triangulation is refined by inserting additional vertices on the offset surface. The resulting mesh is guaranteed to terminate, strictly enclose the input, and be watertight, orientable, and manifold. This makes the method

---

<sup>1</sup>Corresponding author.  
(M. Beeren), (C. García-Sánchez), (H. Ledoux)

This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics.



**Figure 1.1:** Comparison between the original CGAL 3D Alpha Wrap algorithm and the proposed  $\tau$ -extended method for the Delft University of Technology Bouwkunde building from 3D BAG. The input is invalid and contains self-intersections and duplicated vertices, whereas the output meshes are watertight, manifold, and free of self-intersections. In the original method, sharp concave features are rounded; decreasing  $\alpha$  makes them more defined, but does not fully remove this effect and increases mesh complexity globally. The proposed method approximates the sharp concave features more accurately with fewer output faces.

especially suitable for defective, overly complex, or otherwise difficult input geometries.

However, two limitations remain (see also Figure 1.1):

1. *Resolution/complexity trade-off:* Capturing small features typically requires a small  $\alpha$ -value. This can improve the approximation of detailed regions. However, since  $\alpha$  is a global parameter, the same small value also applies to smooth or nearly planar regions. As a result, the number of output faces increases throughout the mesh, including in regions where the additional faces do not substantially improve the approximation.
2. *Rounding of sharp concave features:* The  $\alpha$ -ball cannot reach sharp concave regions as effectively as convex regions. Since the wrap is constructed from the outside, these concave features are therefore less well preserved and tend to appear rounded. A smaller  $\alpha$ -value allows the algorithm to carve deeper into concave regions and improves their approximation, but sharp concave features may still remain rounded.

These limitations are especially relevant for inputs that combine large planar areas with sharp concave features, such as detailed building models used in flow simulation workflows, where geometric details and sharp edges can influence the simulation results (Paden et al., 2024). For such inputs, decreasing  $\alpha$  is not an ideal solution: it increases mesh complexity throughout the output, while sharp concave features may still remain rounded. This motivates a method that improves the approximation of sharp concave features without increasing mesh complexity globally.

This paper presents a  $\tau$ -guided extension within the original 3D Alpha Wrap framework in CGAL. The main contributions are:

- A new precision parameter  $\tau$ , which defines a maximum allowed facet-centroid-to-input distance and triggers additional local refinement when this threshold is exceeded.
- A new vertex-insertion strategy, R3, used for facets refined by the  $\tau$ -criterion, designed to place refinement vertices more effectively near sharp concave features.

Additionally, we also provide a proof that the proposed extension retains the guarantees of the original method: termination, strict enclosure, watertightness, and manifoldness.

Together, these extensions aim to improve the geometric fidelity in difficult regions without requiring a globally smaller  $\alpha$ . In this way, the method reduces run time and unnecessary mesh complexity while preserving the robustness and guarantees of the original 3D Alpha Wrap algorithm.

The remainder of this paper is structured as follows. Section 2 discusses related work on mesh repair, mesh approximation, and feature-preserving methods. Section 3 reviews the original 3D Alpha Wrap algorithm and presents the proposed  $\tau$ -guided extension, including the new  $\tau$ -criterion, the R3 insertion strategy, the theoretical guarantees, and implementation details. Section 4 evaluates the method on 3D BAG data in terms of approximation quality, output complexity, runtime, and validity. Section 5 concludes with the main findings, limitations, and future work.

## 2. Related Work

Real-world meshes often contain errors such as holes, self-intersections, non-manifold elements, or inconsistent orientations (Paden et al., 2024). This is problematic because many applications and simulations require valid mesh input. Such meshes can be repaired or approximated, but it remains difficult to produce an output that is simultaneously valid, close to the input, and not overly refined (Attene et al., 2013).

This related work focuses on methods relevant to our setting: repair and approximation methods for defective input geometry, and feature-preserving methods that aim to improve sharp-feature approximation without unnecessarily increasing mesh complexity.

### 2.1. Repairing invalid surface meshes

Many *local* repair methods detect and correct specific defects directly. This can preserve the original geometry well, since only local parts of the mesh are changed. However, local repair methods often address only a subset of possible validity errors and can struggle when multiple defect types occur simultaneously, particularly for complex or heavily defective inputs (Attene et al., 2013). For this reason, this section focuses on global and hybrid repair methods.

*Global* repair methods reconstruct or approximate the complete surface rather than repairing each defect individually, which can make them more robust for defective inputs. Hybrid methods combine such global reconstruction with local projection or optimization, but these approaches still involve trade-offs between validity, approximation quality, feature preservation, and output complexity (Attene et al., 2013; Portaneri et al., 2022; Wen et al., 2025).

*Volumetric reconstruction.* Volumetric and implicit reconstruction methods convert the input to a grid, voxel, octree, or scalar-field representation and then extract a new surface from it (Berger et al., 2017). Such methods can be robust to local defects and holes (Bischoff et al., 2005). However, the output is resolution-dependent: coarse representations may smooth or remove sharp features, while fine representations increase computational cost and mesh complexity.

*ManifoldPlus.* ManifoldPlus follows a related hybrid strategy: it first extracts a surface from an occupied/empty volumetric representation and then applies projection-based optimization to better approximate the input (Huang et al., 2020). However, degenerate faces can occur, as also reported by (Portaneri et al., 2022), making the method less robust when it is used as a mesh repair method to obtain a valid output mesh.

*Heuristic shrink wrapping.* For 3D city models, (Zhao et al., 2013) propose HSW, a heuristic shrink-wrapping method for automatically repairing invalid solid-based CityGML LoD2 building models (Zhao et al., 2013). The method repairs and generalizes building models by shrink-wrapping a bounding surface towards the input. Their experiments show a strong reduction in invalid solids, although some repaired outputs remain invalid. Since the method is developed specifically for solid-based LoD2 building models, it is less directly applicable to more general input types such as triangle soups, point sets, or mixed defective geometry.

*3D alpha wrap.* CGAL’s 3D Alpha Wrap is a robust global repair and approximation method that produces a watertight, orientable surface mesh that strictly encloses the input, even when the input contains defects (Portaneri et al., 2022). These guarantees make the method especially suitable for difficult repair cases. The method refines and carves a 3D Delaunay triangulation on an offset surface of the input. The output is controlled by two parameters: offset, which controls the distance between inserted vertices and the input, and  $\alpha$ , which controls the carving scale.

However, as shown in Figure 1.1 for inputs with sharp concave features, two limitations remain relevant. First, since  $\alpha$  is a global parameter, improving detailed regions often increases mesh complexity throughout the output. Second, sharp concave features may still remain rounded, even for smaller  $\alpha$ -values.

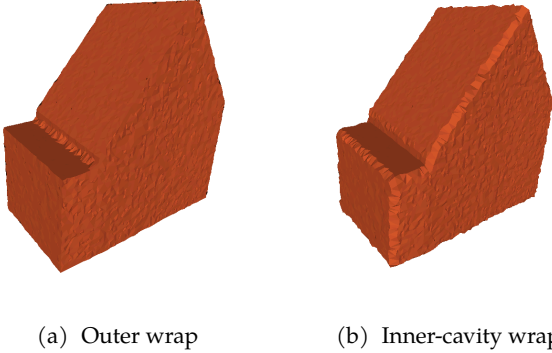
These limitations motivate the extension proposed in this work and are discussed in more detail in Section 3.

### 2.2. Feature-preserving methods

Besides validity, the repaired mesh should also preserve relevant geometric features. For building models, sharp edges and concave regions are important because rounding them changes the local shape of the object. Global repair methods may still smooth or round these features, even when they produce a valid output mesh. The following methods aim to improve sharp-feature preservation during or after repair.

*Alpha wrapping from a cavity.* CGAL’s 3D Alpha Wrap also supports wrapping from an inner cavity (Portaneri et al., 2022). In the standard outer wrap, convex features are generally preserved better than concave features, because concave regions are approached from the outside and may be rounded. When wrapping from an inner cavity, this behavior is reversed: the same concave regions are approached from the inside and can therefore be preserved more sharply. However, combining the inner and outer wraps in a way that keeps the best-preserved sharp features from both surfaces, while still producing one valid output mesh,

is not straightforward. The two wraps may not align or intersect in suitable locations, merging them may cause intersections with the input, and the inner wrap may include internal structures that would need to be detected and removed.



**Figure 2.1:** Comparison of outer and inner-cavity wrapping with CGAL’s 3D Alpha Wrap algorithm.

*Point optimization.* (Wen et al., 2025) propose a feature-preserving repair method that first applies CGAL’s 3D Alpha Wrap and then relocates sampled output vertices based on the underlying input geometry. The relocation step minimizes angular errors between neighboring vertices and tangent planes, which tends to pull vertices toward creases and corners. Although, this does improve the alignment of sharp features, especially in concave regions, it does not preserve the validity guarantees of the original Alpha Wrap. In particular, relocating vertices back toward the input may introduce self-intersections or invalid geometry, especially for thin or near-zero-volume structures. The method can also be computationally expensive, since the input must first be wrapped with a sufficiently small  $\alpha$ , followed by an additional optimization step, whose computation time strongly depends on the amount of sampled vertices used for this optimization step. Moreover, the method relies on input normal information, making it less suitable for inputs such as raw point clouds or triangle soups.

*Protection balls.* Some Delaunay-based meshing methods use protection balls around sharp features to prevent unwanted triangulation across edges and corners (Cheng et al., 2007). This can help preserve sharp features during refinement. However, these methods often rely on weighted Delaunay triangulations, which do not satisfy the same empty-sphere property as the ordinary Delaunay triangulation used in 3D Alpha Wrap. Therefore, directly integrating protection balls into the Alpha Wrap framework is not straightforward, since the termination guarantees of the original algorithm may no longer apply. In addition, the setting of Alpha Wrap is more difficult because refinement takes place on an offset surface. Sharp features would there-

fore not only need to be detected, but also projected to suitable offset locations, which is ambiguous for complex or defective input geometry (Portaner et al., 2022).

### 3. Methodology

#### 3.1. Original 3D Alpha Wrap algorithm

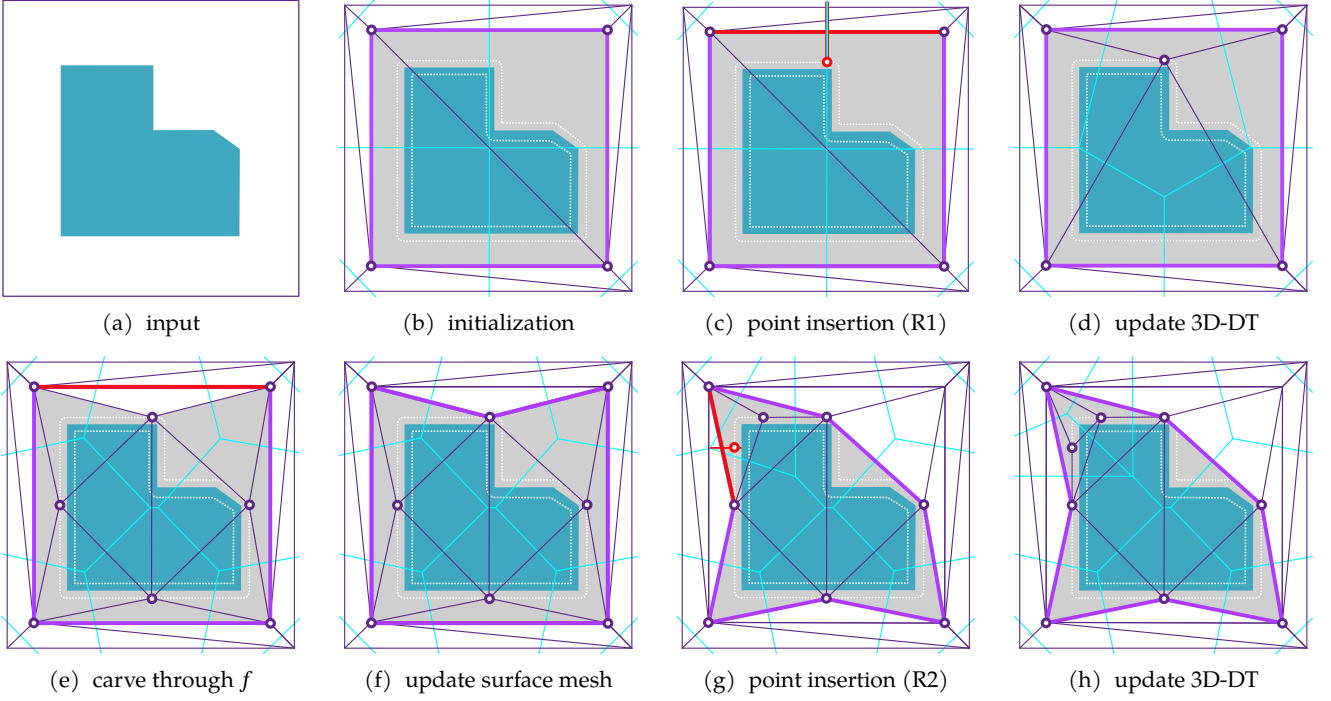
To clarify where the proposed extension adds to the original method, we first summarize the relevant concepts of CGAL’s 3D Alpha Wrap algorithm.

Given a 3D input geometry, the original 3D Alpha Wrap algorithm is a robust global mesh repair and approximation algorithm. It is guaranteed to terminate and to generate a surface mesh that is manifold, watertight, and strictly encloses the input. These guarantees also hold for inputs that are unoriented, defective, or represented only as a triangle soup, point set or mixed input. The algorithm can be interpreted as a shrink-wrapping procedure around an offset surface, where a carving ball is used to shrink the mesh towards the input.

*Parameters.* The original algorithm is controlled by two global parameters:  $\alpha$  and  $\delta$ . The parameter  $\alpha$  defines the radius of the carving ball: smaller  $\alpha$ -values allow the wrap to pass through smaller cavities and openings. The parameter  $\delta$  defines the distance of the implicit offset surface around the input. New vertices inserted during refinement are placed on this offset surface.

*Initialization.* A loose axis-aligned bounding box is first constructed around the input and its offset surface. This bounding box is tetrahedralized using a 3D Delaunay triangulation. The tetrahedra of this triangulation are referred to as cells. The cells that make up the bounding box are initially labelled `INSIDE`, while the infinite cells outside the bounding box are labelled `OUTSIDE`. The surface mesh is defined by the triangular facets separating an `INSIDE` cell from an `OUTSIDE` cell. At initialization, this surface is therefore the boundary of the loose bounding box, which strictly encloses the input (see Figure 3.1b).

*Alpha wrapping.* After initialization, the flood-filling procedure starts. The algorithm maintains a queue of facets that still need to be processed. Each facet in this queue is tested for  $\alpha$ -traversability. A facet is called  $\alpha$ -traversable when its minimal empty Delaunay ball is larger than the  $\alpha$ -ball. Intuitively, this means that the facet is still large enough, relative to  $\alpha$ , to be considered for further processing.



**Figure 3.1:** Schematic 2D illustration of the main steps of the original 3D Alpha Wrap algorithm, included to clarify where the proposed extension differs later in Figure 3.2 and Figure 3.3. For conciseness, only the main operations are shown. The input is shown in turquoise, the offset surface as a white dotted line, the current surface mesh in purple, the currently processed facet and inserted point in red, the Delaunay triangulation as thin dark-purple lines, and the Voronoi diagram in light blue. (b) A loose bounding box is initialized around the input. (c) For an  $\alpha$ -traversable facet  $f$ , Rule R1 inserts a Steiner point when the dual Voronoi edge intersects the offset surface. (d) The Delaunay triangulation is updated after insertion. (e) If no such intersection exists and the adjacent `INSIDE` cell does not intersect the input, the cell is carved. (f) The current surface mesh is updated after carving. (g) If the dual Voronoi edge does not intersect the offset surface but carving would expose the input, Rule R2 inserts a Steiner point instead. (h) The Delaunay triangulation is updated after insertion.

For each  $\alpha$ -traversable facet, the algorithm checks whether the adjacent `INSIDE` cell can be carved. Carving means relabelling this cell as `OUTSIDE` (see Figure 3.1e), thereby moving the current surface mesh inward (see Figure 3.1f). This is only allowed if the cell can be removed without crossing the input geometry. If carving is allowed, the surface mesh is updated and newly exposed surface facets are added to the queue. If not, the triangulation is refined by inserting a new vertex on the offset surface, called a Steiner point.

The original algorithm uses two insertion strategies to determine where a Steiner point is placed: Rule R1 and Rule R2. Rule R1 tries to insert a Steiner point at the intersection between the dual Voronoi edge of the processed facet and the offset surface (see Figure 3.1c). This generally leads to well-shaped facets. If R1 fails, Rule R2 first projects the circumcenter of the adjacent `INSIDE` cell onto the input. From this projected input point, a new point is then constructed on the offset surface (see Figure 3.1g). This strategy often performs well near convex features.

The algorithm continues alternating between carving and refinement until the queue is empty. At this point, no remaining surface facet is  $\alpha$ -traversable. The final

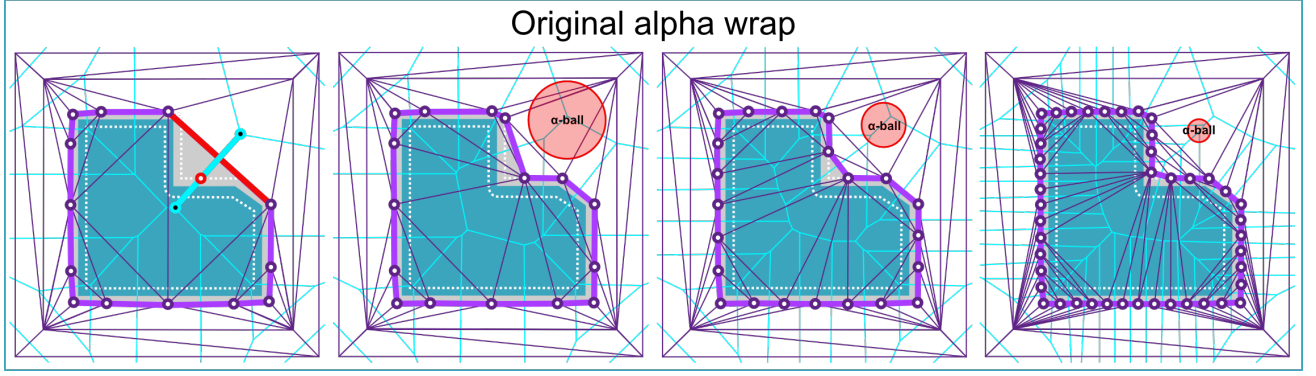
output mesh is formed exactly by those triangular facets that separate one `INSIDE` cell from one `OUTSIDE` cell

### 3.2. Proposed $\tau$ -extension for the 3D Alpha Wrap algorithm

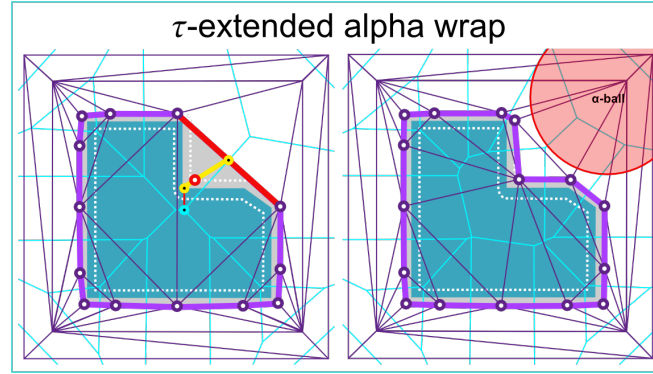
We propose an extension of CGAL’s 3D Alpha Wrap algorithm that preserves the original carving-based structure. The extension adds two components to the original algorithm: an additional  $\tau$ -traversability criterion and a new vertex-insertion strategy, denoted as Rule R3. The  $\tau$ -criterion is used to continue refinement when the current surface is still locally too far from the input, while R3 is designed to place Steiner points more effectively near sharp concave features. The main guarantees of the original method, namely termination, strict enclosure, watertightness, and manifoldness, are retained and discussed in Section 3.3.

*$\tau$ -traversability.* In the original algorithm, a surface facet is processed only when it is  $\alpha$ -traversable. Once no remaining surface facet satisfies this criterion, the algorithm terminates. However, near sharp concave features, the current surface can still be relatively far from the input even when the  $\alpha$ -criterion no longer triggers refinement.





**Figure 3.2:** 2D illustration of output meshes generated by the original 3D Alpha Wrap algorithm using only the  $\alpha$ -traversability criterion and insertion rules R1 and R2. The examples show decreasing  $\alpha$ -ball sizes: smaller  $\alpha$ -values improve the approximation of the concave feature, but also increase refinement in other regions. A zoomed-in view of the Steiner point insertion near the concave region is shown in Figure 3.6a.



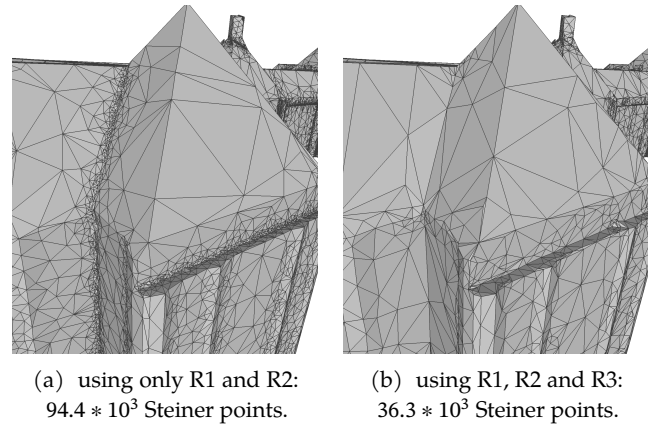
**Figure 3.3:** 2D illustration of output meshes generated by the proposed  $\tau$ -extended 3D Alpha Wrap algorithm. The method adds the  $\tau$ -traversability criterion and insertion rule R3, allowing additional refinement where the wrapped surface remains too far from the input. R3 places Steiner points more directly near the concave feature, so that the feature can be approximated more closely even with a larger  $\alpha$ -ball. A zoomed-in view of the R3 insertion near the concave region is shown in Figure 3.6b.

We therefore introduce an additional traversability criterion controlled by a new global parameter  $\tau$ . This parameter defines an allowed face-to-input distance relative to the offset distance  $\delta$ . For a facet  $f$ , with centroid  $g_f$ , we compute the shortest distance to the input geometry  $I$ , denoted as  $d(g_f, I)$ . If this distance exceeds the threshold  $\tau\delta$ , the facet is considered  $\tau$ -traversable: Since inserted vertices lie on the offset surface,  $\tau$  is chosen larger than 1, so that the threshold is larger than the Steiner-point-to-input distance.

In our extended algorithm, a facet can be processed in two cases. If the facet is  $\alpha$ -traversable, the original Alpha Wrap procedure is applied. If the facet is no longer  $\alpha$ -traversable but still exceeds the  $\tau$ -threshold, and thus is  $\tau$ -traversable, the original algorithm would stop at this location, whereas the extension continues refinement locally. This allows regions that remain too far from the input to be refined without decreasing  $\alpha$  globally (see Figure 3.2 vs Figure 3.3).

**Rule R3.** Adding the  $\tau$ -traversability criterion alone is not sufficient, because the original insertion rules R1 and R2 are not specifically designed to place Steiner points near sharp concave features. In such regions,

using only R1 and R2 may require many refinement steps before the surface becomes close enough to the input to satisfy the  $\tau$ -threshold. We therefore introduce a third insertion rule, R3, which is used only for facets that are  $\tau$ -traversable but not  $\alpha$ -traversable (see Figure 3.4a vs Figure 3.4b).



**Figure 3.4:** Effect of insertion rule R3 near concave features, shown for the Delft University of Technology Bouwkunde building using  $\alpha = 10$ , offset = 0.03, and  $\tau = 2$ .

The goal of R3 is to place a Steiner point on the outer offset surface close to a sharp concave feature. The un-



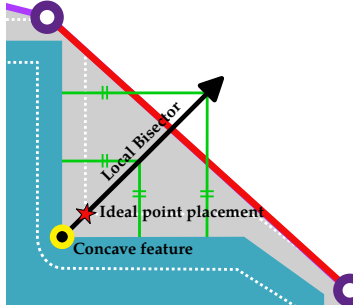
derlying idea of the construction is to first estimate the location of the relevant concave feature on the input, and then determine a direction from this point towards the outer offset surface, so that a Steiner point can be placed where this direction reaches the offset surface.

Let  $f$  be a surface facet that is  $\tau$ -traversable but not  $\alpha$ -traversable, and let  $c_n$  be the adjacent INSIDE cell. Let  $k_{c_n}$  be the circumcenter of this cell.

First, R3 tries to target the relevant concave feature on the input. To do this, the closest point on the input to  $k_{c_n}$  is computed and denoted by  $p_p$ . Although  $p_p$  is not guaranteed to lie exactly on a sharp edge, it is expected to lie on or near the relevant concave feature.

The next step is to approximate a direction from this input region towards the outer offset surface.

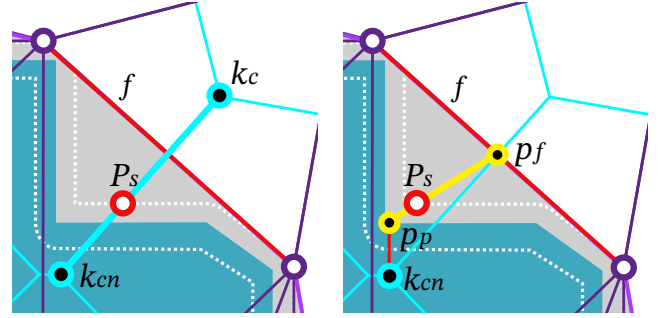
In an idealized setting, a sharp concave feature can be seen as the intersection of neighboring input planes. A suitable point on the outer offset surface near this feature would then lie in a direction that is approximately equidistant from the neighboring planes, corresponding to the local bisector direction of the concave feature. If reliable normals and connectivity were available, this direction could be estimated directly from the incident input faces (see Figure 3.5). However, the input may be unoriented, defective, or represented only as a triangle soup or point set. Therefore, R3 approximates this direction using only closest-point queries to the input and the currently processed surface facet.



**Figure 3.5:** Idealized point placement near a sharp concave feature. The desired point on the outer offset surface lies approximately along the local bisector direction of the neighboring input planes.

As shown in Figure 3.6, to approximate this direction, R3 first selects a helper point on the currently processed facet  $f$ . The edge midpoints of  $f$  are evaluated, and the midpoint with the largest distance to the input is selected. This midpoint is used as an approximation of the point on  $f$  that lies farthest from the input. Together with the approximated concave feature location  $p_p$ , it defines a direction that serves as an approximation of the local bisector direction towards the outer offset surface.

Before constructing the segment, the selected helper point is shifted towards the facet centroid  $g_f$  by 5% of the vector from the helper point to  $g_f$ , yielding  $P_f$ . We require  $d(P_f, I) > \delta$ , so that  $P_f$  lies outside the offset region. The candidate Steiner point is then constructed using the segment from  $P_f$  to  $p_p$ . Starting from  $P_f$ , the first intersection of this segment with the offset surface is selected. This intersection is expected to give a good approximation of the desired Steiner point position on the outer offset surface near the concave feature (see Figure 3.6a vs Figure 3.6b).



(a) Insertion using R1; (b) Insertion using R3; zoomed-in view of Figure 3.2. zoomed-in view of Figure 3.3.

**Figure 3.6:** Zoomed-in schematic comparison of the original R1 insertion rule and the proposed R3 insertion rule near a concave feature. R1 inserts a Steiner point at the intersection of the dual Voronoi edge with the offset surface, whereas R3 targets the concave region more directly using the segment from  $P_f$  to  $p_p$ .

Before insertion, the candidate point is tested against a lower bound on the insertion radius. The insertion radius is the shortest distance from the candidate Steiner point to the set of previously inserted Steiner points. As discussed in Section 3.3, the insertion radius must be bounded from below to prove termination. For R3, this lower bound is enforced explicitly: the candidate point is accepted only if its insertion radius is larger than the lower bound used for Steiner points inserted by R1 and R2,  $\min(\alpha, \delta, (\tau - 1)\delta)$ .

If this condition is not satisfied, R3 is rejected and the algorithm falls back to the original insertion rules R1 and R2. The 5% shift is applied to the construction point used to determine the R3 candidate. Moving the construction point towards  $g_f$  generally places it closer to the circumcenter of the neighboring cell. Because the Delaunay ball of that cell contains no previously inserted vertices, moving closer to its circumcenter generally increases the construction point's minimum distance to previously inserted Steiner points. Consequently, although the shift is not required for the algorithm to function, it makes R3 more likely to produce a candidate whose insertion radius meets the lower-bound requirement defined above, reducing the need to fall back to R1 or R2.

---

**Algorithm 1:  $\tau$ -Guided 3D Alpha Wrapping**

---

**Input:** Input geometry  $I$ , Parameter  $\alpha$ , Offset surface  $I_\delta$ ,  
Parameter  $\tau$   
**Output:** Output surface mesh  $\mathcal{O}$

Initialize Delaunay triangulation  $\mathcal{D}$  with the vertices of a loose bounding box of  $I_\delta$   
Tag OUTSIDE all infinite cells of  $\mathcal{D}$   
Tag INSIDE all finite cells of  $\mathcal{D}$   
Initialize priority queue  $\mathcal{Q}$  with convex hull facets of  $\mathcal{D}$  that are  $\alpha$ -traversable.  $\mathcal{Q}$  is sorted in decreasing order of min Delaunay ball radius  $\Phi(f)$

**while**  $\mathcal{Q}$  is not empty **do**  
  Gate  $f \leftarrow \mathcal{Q}.top()$   
   $\mathcal{Q}.pop()$   
  **if**  $\Phi(f) < \alpha$  **then**  
    Let  $f_{\text{centroid}}$  be the centroid of facet  $f$   
    Let  $d_f$  be the shortest distance from  $f_{\text{centroid}}$  to  $I$   
    **if**  $d_f \leq \tau * \delta$  **then**  
      **continue**  
    Let  $c_o$  be the cell incident to  $f$  tagged OUTSIDE  
    Let  $c_n$  be the other cell incident to  $f$   
    **if**  $c_n$  is tagged OUTSIDE **then**  
      **continue**  
    Let  $k_c$  be the dual Voronoi vertex of  $c_o$  (circumsphere center), or the circumcircle of  $f$  if  $c_o$  is infinite  
    Let  $k_{c_n}$  be the dual Voronoi vertex of  $c_n$   
    Let  $V_f$  be the oriented segment  $[k_c; k_{c_n}]$   
    Let  $P_s$  be a potential Steiner point  
    **if**  $\Phi(f) < \alpha$  **and**  $c_n$  intersects  $I$  **then**  
      Let  $P_p$  be the point closest from  $k_{c_n}$  on  $I$   
      Let  $P_f$  be the edge midpoint of  $f$  farthest from  $I$ , shifted 5% towards its centroid.  
      Let  $E_e$  be the oriented segment  $[P_f; P_p]$   
      Let  $\mathcal{P}$  be the set of all previously inserted points  
      Let  $P_{\text{test}}$  be a candidate Steiner point whose distance to  $\mathcal{P}$  is checked before insertion  
      **if**  $E_e$  intersects  $I_\delta$  **then**  
         $P_{\text{test}} \leftarrow$  first intersection point between  $E_e$  and  $I_\delta$   
        Let  $d_p$  be the shortest distance from  $P_{\text{test}}$  to  $\mathcal{P}$   
        Let  $d_{\min} = \min(\alpha, \delta, (\tau - 1)\delta)$   
        **if**  $d_p > d_{\min}$  **then**  
           $P_s \leftarrow P_{\text{test}}$   
      **if**  $V_f$  intersects  $I_\delta$  **and**  $P_s$  not yet assigned **then**  
         $P_s \leftarrow$  first intersection point between  $V_f$  and  $I_\delta$   
      **else if**  $c_n$  intersects  $I$  **then**  
        Let  $P_p$  be the point closest from  $k_{c_n}$  on  $I$   
        Let  $E_p$  be the oriented segment  $[k_{c_n}; P_p]$   
         $P_s \leftarrow$  closest point from  $k_{c_n}$  on  $I_\delta$  and along  $E_p$   
      **end**  
      **if**  $P_s$  exists **then**  
        Remove from  $\mathcal{Q}$  the gates that will be destroyed upon insertion of  $P_s$  into  $\mathcal{D}$   
        Insert  $P_s$  into  $\mathcal{D}$   
        Tag new finite cells INSIDE and new infinite cells OUTSIDE  
        Push new  $\alpha$ -traversable gates to  $\mathcal{Q}$   
      **else**  
        Tag  $c_n$  as OUTSIDE  
        Push all  $\alpha$ -traversable gates incident to  $c_n$  to  $\mathcal{Q}$   
      **end**  
    **end**  
   $\mathcal{O} \leftarrow$  surface triangle mesh represented facets separating INSIDE from OUTSIDE cells  
**return**  $\mathcal{O}$

---

**Algorithm 1:** Pseudocode of the  $\tau$ -extended algorithm. The original algorithm is shown in gray, while the extensions introduced in this work are marked by a green bar.

Taking the first intersection with the offset surface when moving from  $P_f$  towards  $P_p$  is important to avoid inserting Steiner points on inner structures. The point  $P_f$  lies on a surface facet, and is therefore incident to an OUTSIDE cell; it is consequently reachable from the exterior and not located on an inner structure. Since  $P_f$  is also chosen outside the offset region ( $d(P_f, I) > \delta$ ), the first intersection places the new Steiner point on the outer boundary of the offset region, before the segment can reach the input or possible inner structures. This is necessary because, in the original algorithm, R1 is tested first to prevent Steiner points from being inserted on inner structures. Since the extended algorithm may evaluate R3 before R1, R3 must provide the same protection.

### 3.3. Guarantees

The original 3D Alpha Wrap algorithm provides several theoretical guarantees: the algorithm terminates, and the output mesh is watertight, manifold, and strictly encloses the input (Portaneri et al., 2022). The modifications proposed in this work do not alter the parts of the algorithm on which the watertightness, manifoldness, and enclosure arguments rely. Hence, these guarantees are preserved. Only the proof of termination must be adapted, since both the traversability criterion and the Steiner-point insertion strategy have been modified.

*Termination.* In the original paper, termination follows from two key facts. First, a Steiner point inserted for refinement lies inside the Delaunay ball of the cell that is intended to be refined, so that the refinement step is effective and breaks that cell. Second, the insertion radius, defined as the shortest distance from the candidate Steiner point to the set of previously inserted Steiner points, is bounded from below, meaning that every newly inserted Steiner point is at least this fixed minimum distance from the previously inserted vertices. Together, these two facts give termination through the same packing argument as in the original proof.

These facts are proven in the original algorithm under the original  $\alpha$ -traversability criterion and the original Steiner-point insertion rules R1 and R2. Since the modified algorithm adds the  $\tau$ -based traversability criterion and introduces a new insertion strategy, the original termination proof does not directly apply anymore. Therefore, the same two facts need to be proven again for the modified setting. An adapted proof is given in Appendix Section A.

*Watertightness.* The output mesh is still defined as the set of facets separating one cell labelled INSIDE from one cell labelled OUTSIDE. Since this definition and the

cell-labelling procedure are unchanged, the output remains watertight.

*Manifoldness.* Each carving operation still removes an entire tetrahedron. Hence, the original manifoldness argument remains valid.

*Enclosure.* The modified algorithm preserves the two ingredients needed for strict enclosure: it starts from a bounding box that contains the input, and cells are carved only when they do not intersect the input. Since these parts are unchanged, the output mesh still strictly encloses the input.

### 3.4. Implementation

The extension is implemented in C++ directly within the existing CGAL (The CGAL Project, 2021) 3D Alpha Wrap codebase. The only algorithmic additions are the  $\tau$ -traversability criterion and the corresponding point-insertion strategy, denoted as Rule R3. These additions are integrated into the existing refinement framework rather than implemented as a separate geometry-processing pipeline.

For all operations that depend on the input geometry or the offset surface, the extension reuses query types already present in the original Alpha Wrap implementation. These include tetrahedron-input intersection tests, segment-offset-surface intersections, and projections of query points onto the input. Therefore, the extension does not introduce new types of input- or offset-surface queries.

This is important because the original Alpha Wrap implementation explicitly addresses practical correctness under floating-point arithmetic, filtered predicates, and approximate constructions. By reusing the same types of geometric operations, the extension remains within the numerical framework already considered by the original implementation.

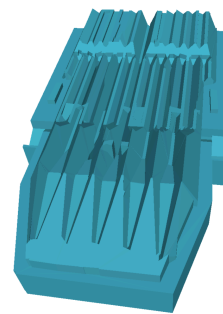
## 4. Experiments

All experiments are performed on a laptop running macOS 26.3.1, with an Apple M4 chip, 24 GB of memory, and internal solid-state storage. The code is compiled using Apple Clang 17.0.0.

A direct parameter-wise comparison with the original Alpha Wrap algorithm is difficult, because the proposed extension introduces the additional parameter  $\tau$ . Identical  $\alpha$  and offset values do not necessarily result in comparable behavior. Therefore, the methods are compared mainly in terms of approximation quality, output complexity, runtime, and validity. Output complexity is measured as the total number of output faces. Approximation quality is evaluated using the

directed Chamfer distance and cumulative distance distributions.

All approximation-quality and runtime experiments are performed on the Delft University of Technology Aula building from the 3D BAG dataset (see Figure 4.1). This input is invalid according to the validity checks, making it a relevant real-world repair case. It also represents the intended use case of the extension, since it contains large planar regions combined with several sharp concave features. Such inputs are challenging for the original Alpha Wrap algorithm: accurately approximating concave regions requires a smaller global  $\alpha$ , which also increases refinement in planar regions where this is not needed.



**Figure 4.1:** Delft University of Technology Aula building from the 3D BAG dataset, used for all approximation-quality and runtime experiments.

This targeted test case is chosen because it contains the type of geometry for which the extension is designed. The proposed method is not intended to change the behavior of Alpha Wrap in regions where the original wrap is already close enough to the input, meaning that the face-centroid-to-input distance does not exceed the  $\tau$ -threshold. Therefore, for inputs where the original algorithm already remains within this threshold, or when  $\tau$  is chosen sufficiently large, the extension follows the same refinement steps as the original algorithm and produces the same output mesh. In such cases, the only additional cost is evaluating the  $\tau$ -criterion.

The validation experiments use additional datasets to test robustness more broadly. These include all 3D BAG buildings in Delft (Peters et al., 2022), increasingly corrupted input geometry, and a complete 3D BAG tile of the Delft University of Technology campus.

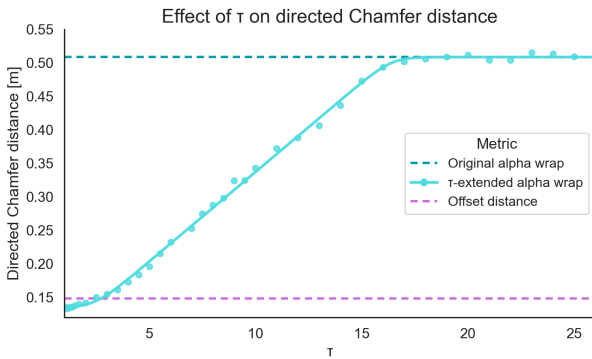
### 4.1. Approximation quality and output complexity

Since a fair comparison using identical parameters is not meaningful, the original and extended methods are compared based on the relation between approximation quality and output complexity. For the original Alpha Wrap algorithm, we vary the  $\alpha$ -value. For the

$\tau$ -extended method, we use a large fixed  $\alpha$ -value and vary  $\tau$ , so that the influence of the new criterion can be studied more directly. The offset distance is kept fixed, since it strongly affects the distance between the output mesh and the input.

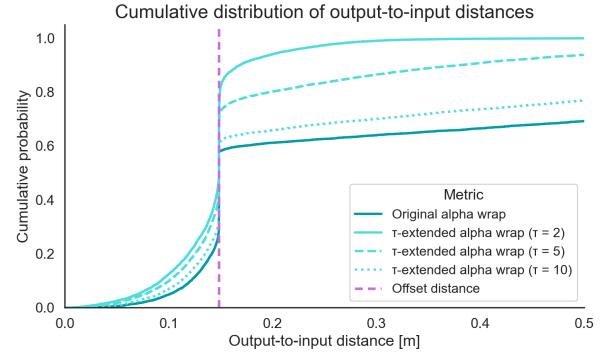
Approximation quality is measured using a directed Chamfer distance from the output mesh to the input, rather than a symmetric Chamfer distance. This avoids penalizing internal structures that may be present in the input but are intentionally not reconstructed in the output. For each measurement, 10,000 points are randomly sampled on the output mesh. For each sampled point, the shortest distance to the input mesh is computed. This process is repeated 100 times, after which the final reported value is obtained by averaging over all runs.

The effect of  $\tau$  is first evaluated by plotting the directed Chamfer distance against the  $\tau$ -value (see Figure 4.2). As  $\tau$  increases, the result approaches the original Alpha Wrap output. This is expected: when the  $\tau$ -threshold is sufficiently large, the additional refinement criterion is no longer triggered, and the method behaves like the original algorithm (see also Appendix Section E).



**Figure 4.2:** Directed Chamfer distance from the output mesh to the input for increasing  $\tau$ -values. Both methods use  $\alpha = 7.4$  and offset = 0.15. As  $\tau$  increases, the additional refinement criterion is triggered less often and the result approaches the original Alpha Wrap output.

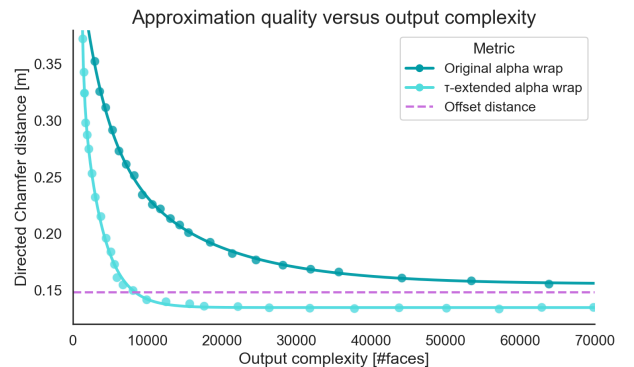
To study local approximation quality, we also use cumulative distance distributions (see Figure 4.3). The Chamfer distance provides a single average value, but does not show how errors are distributed over the mesh. The cumulative distribution shows that many sampled points lie close to the offset distance, which is expected since both methods insert points on the offset surface. However, lower  $\tau$ -values reduce the number of poorly approximated regions. For example, for  $\tau = 2$ , more than 90% of sampled points lie within 0.2m of the input, whereas this is only about 60% for the original algorithm.



**Figure 4.3:** Cumulative distribution of sampled output-to-input distances. All runs use  $\alpha = 7.4$  and offset = 0.15. Lower  $\tau$ -values place a larger fraction of the output surface close to the input, indicating improved local approximation quality.

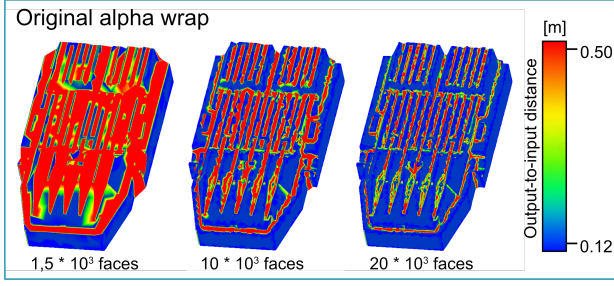
Finally, we compare directed Chamfer distance against output complexity (see Figure 4.4 and as visual results represented in Figure 4.5 and Figure 4.6). The original method is evaluated using a sweep over  $\alpha$ , while the extended method is evaluated using a sweep over  $\tau$  with a large fixed  $\alpha$ . Smaller  $\tau$ -values require more local refinement, but the new R3 insertion strategy aims to place refinement points more effectively near sharp concave regions. The results show that the  $\tau$ -extended method reaches low directed Chamfer distances with fewer output faces than the original method.

As shown in Figure 4.4, the directed Chamfer distance can fall below the offset distance at higher output complexities. Although Steiner points are inserted at the prescribed offset, the connecting facets are not constrained to lie at this distance. The  $\tau$ -criterion only triggers further refinement for facets that lie farther from the input than the prescribed offset. Consequently, facets that lie closer to the input remain unrefined and therefore lower the average directed Chamfer distance.

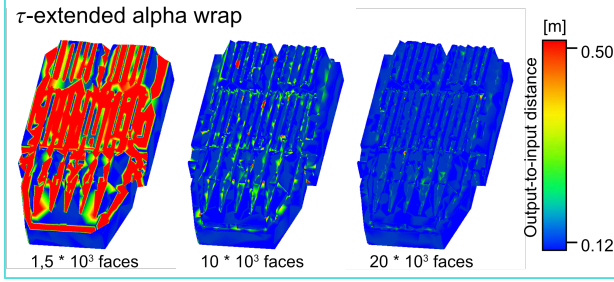


**Figure 4.4:** Directed Chamfer distance as a function of output complexity. The original Alpha Wrap method is evaluated by varying  $\alpha$ , while the  $\tau$ -extended method is evaluated by varying  $\tau$ . The offset is fixed at 0.15, and the  $\tau$ -extended method uses  $\alpha = 7.4$ . The  $\tau$ -extended method reaches lower directed Chamfer distances with fewer output faces.





**Figure 4.5:** Visual results of the original Alpha Wrap algorithm for increasing output complexity. The three outputs contain approximately  $1.5 \cdot 10^3$ ,  $10 \cdot 10^3$ , and  $20 \cdot 10^3$  faces.

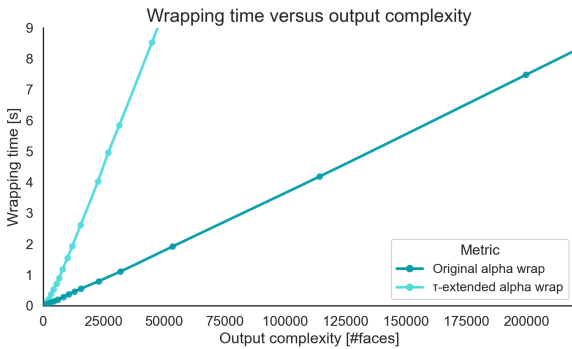


**Figure 4.6:** Visual results of the  $\tau$ -extended Alpha Wrap algorithm for increasing output complexity. The three outputs contain approximately  $1.5 \cdot 10^3$ ,  $10 \cdot 10^3$ , and  $20 \cdot 10^3$  faces.

## 4.2. Computational efficiency

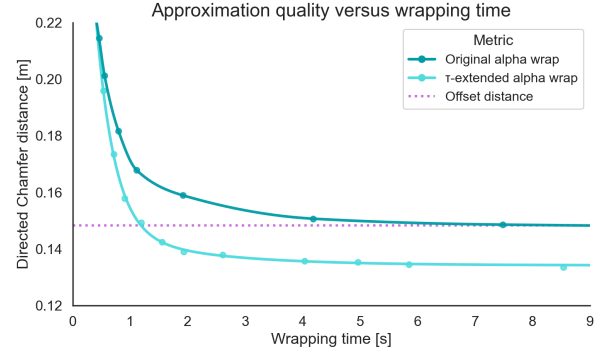
Runtime is measured by repeating each wrapping experiment five times and taking the average.

We first compare output complexity against wrapping time (see Figure 4.7). In this experiment, larger  $\alpha$ -values are used for the  $\tau$ -extended method, so that refinement is more often triggered by the  $\tau$ -criterion rather than by the original  $\alpha$ -criterion. As a result, the new R3 insertion strategy is used more frequently. The plot shows that, in these cases, output complexity increases more slowly with wrapping time. This suggests that R3 is computationally more expensive than the original R1 and R2 insertion strategies: fewer output faces are generated per unit of time. This is expected, since R3 requires additional distance queries to the input.



**Figure 4.7:** Wrapping time versus output complexity. The original Alpha Wrap varies  $\alpha$ , while the  $\tau$ -extended method varies  $\tau$  with  $\alpha = 7.4$ ; both use offset = 0.15. Its higher wrapping time at comparable output complexity suggests that R3 insertions are more expensive than R1 and R2.

We then compare the directed Chamfer distance against wrapping time. As suggested by Figure 4.7, R3 insertions are more expensive than the original R1 and R2 insertions. However, as shown in Figure 4.4, the  $\tau$ -extended method requires fewer output faces to achieve a similar approximation quality. As a result, it reaches lower directed Chamfer distances faster than the original method in the tested use case (see Figure 4.8). This indicates that the additional cost of R3 can be compensated by more effective local refinement.



**Figure 4.8:** Directed Chamfer distance as a function of wrapping time. The original Alpha Wrap method is evaluated by varying  $\alpha$ , with offset = 0.15. The  $\tau$ -extended method uses  $\alpha = 7.4$ , offset = 0.15, and varying  $\tau$ -values. Although R3 insertions are more expensive, the  $\tau$ -extended method reaches lower directed Chamfer distances faster in the tested use case.

## 4.3. Validation

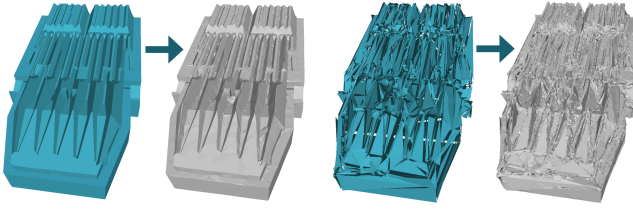
To evaluate whether the practical guarantees of the original Alpha Wrap algorithm are preserved, we test termination, watertightness and manifoldness. Termination is verified by whether the wrapping process finishes successfully. Watertightness and manifoldness are evaluated using val3dity (Ledoux, 2013; 2018).

Because some outputs may consist of multiple disconnected components, validity is evaluated per connected component. This avoids treating a valid collection of disconnected solids as one invalid solid. Each connected component is therefore tested individually with val3dity. In addition, possible intersections between disconnected components are checked using CGAL with exact predicates.

Three validation experiments are performed.

- **Test 1:** all 37,417 3D BAG buildings in Delft are wrapped individually using  $\alpha = 10m$ , offset =  $0.05m$ , and  $\tau = 2$ . This test evaluates whether the method consistently terminates and produces valid output across a large number of real-world building inputs, which may contain geometric defects. All buildings terminate successfully and pass the validity tests.

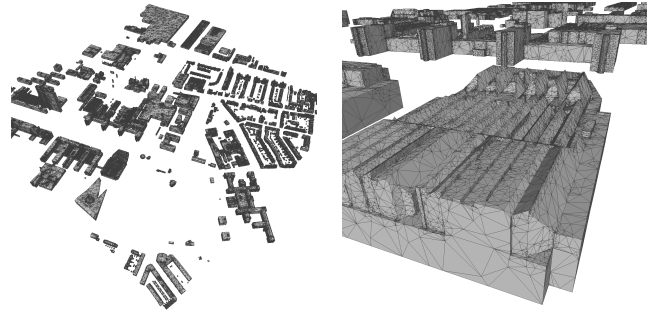
- **Test 2:** a robustness test is performed on increasingly corrupted input geometry. The input faces are first disconnected from each other and its vertices are then displaced in random directions with increasing displacement magnitudes. Even for a maximum displacement of 1m, the algorithm terminates and produces a valid output. However, at large displacements the input contains large openings, through which the wrap may enter the object and include internal structures. This is a limitation of both the original and extended methods. In the original Alpha Wrap algorithm, this behavior is mainly controlled by the  $\alpha$ -value: a smaller  $\alpha$ -ball can pass through smaller openings. In the  $\tau$ -extended method, this effect additionally depends on the  $\tau$ -threshold, since a smaller  $\tau$  can trigger further refinement near openings and thereby allow the wrap to enter regions that would otherwise remain unwrapped.



(a) No random vector shift      (b) Random vector shift of 1m

**Figure 4.9:** Robustness test on increasingly corrupted input geometry. The input faces are first disconnected from each other, after which their vertices are displaced in random directions. Both examples are wrapped using  $\alpha = 7.4$ , offset = 0.15, and  $\tau = 2$ .

- **Test 3:** a complete 3D BAG tile of the Delft University of Technology campus is wrapped in a single run. The tile contains many separate buildings, including objects that are close to each other, touching, or partially overlapping. The algorithm must therefore process multiple disconnected components within the same wrapping procedure, rather than handling each object in isolation. The dataset also contains many sharp concave regions. A large  $\alpha$ -value is combined with a small offset and small  $\tau$ -value to trigger the new refinement criterion frequently. The algorithm terminates successfully and produces a valid output.



(a) Complete wrapped tile      (b) Zoomed-in view

**Figure 4.10:** Wrapping result for the complete Delft University of Technology campus 3D BAG tile. The tile is wrapped as a single input using  $\alpha = 10$ , offset = 0.025, and  $\tau = 2$ .

## 5. Conclusions

This paper presents a  $\tau$ -guided extension of CGAL’s 3D Alpha Wrap algorithm—a simple and yet effective extension—designed to improve the preservation of sharp concave features without requiring a globally small  $\alpha$ -value. The method introduces an additional  $\tau$ -traversability criterion and a new vertex-insertion strategy, R3, while preserving the carving-based structure of the original algorithm.

A theoretical proof is provided showing that the main guarantees of the original 3D Alpha Wrap algorithm—termination, watertightness, manifoldness, and enclosure—are preserved. The arguments for watertightness, manifoldness, and enclosure carry over from the original method, since the relevant parts of the algorithm are unchanged. Only the termination proof is adapted to account for the new  $\tau$ -traversability criterion and the R3 insertion rule.

The experiments on the Delft University of Technology Aula building show that the  $\tau$ -extended method improves both the average and local approximation of the input. Compared to the original Alpha Wrap algorithm, it reaches lower directed Chamfer distances and, as shown by the cumulative distance distributions, has a larger fraction of sampled points close to the input, while using fewer output faces. This suggests that the  $\tau$ -criterion and R3 insertion strategy refine the mesh more selectively: additional vertices are placed mainly in regions where the original wrap remains too far from the input, instead of increasing refinement globally through a smaller  $\alpha$ .

The computational results show that the R3 insertion strategy is more expensive than the original R1 and R2 strategies. However, because the extension requires fewer output faces to reach a similar approximation quality, this additional cost can be compensated by more effective local refinement. As a result, the proposed method reaches lower directed Chamfer dis-



tances faster than the original method in the tested use case.

The validation experiments support the theoretical results in practice. The method terminates successfully and produces valid outputs for individual 3D BAG buildings in Delft, increasingly corrupted input geometry, and a complete 3D BAG tile of the Delft University of Technology campus.

Overall, the proposed extension is most useful for inputs that combine large planar regions with sharp concave features. For inputs where the original Alpha Wrap algorithm already performs well, or when  $\tau$  is chosen sufficiently large, the  $\tau$ -criterion is rarely triggered and the method behaves similarly to the original algorithm, apart from the additional cost of evaluating the criterion.

### 5.1. Limitations and future work

*$\tau$ -criterion.* The current  $\tau$ -criterion uses the face-centroid-to-input distance as an approximation of the local surface-to-input distance. This is simple and computationally efficient, and works well in the tested cases. However, it can miss cases where the centroid lies close to the input, while another part of the same face is still relatively far away. A more robust, but still efficient, way of determining whether a local mesh face is sufficiently close to the input would therefore be a useful improvement.

*R3 insertion strategy.* The R3 insertion strategy improves vertex placement near sharp concave features, but is computationally expensive due to the additional distance-to-input queries and insertion-radius check. Although, the output is guaranteed to be watertight and manifold, R3 can produce less regular triangles or thin surface regions. Future work could further improve the vertex-placement strategy, while preserving the guarantees of the original Alpha Wrap algorithm. For example by improving the local bisector direction approximation.

*Wrapping internal structures through input holes.* As in the original Alpha Wrap algorithm, the wrap may enter the input through large holes or openings and include internal structures. In the original method, this behavior is mainly controlled by the  $\alpha$ -value: smaller  $\alpha$ -balls can pass through smaller openings. In the  $\tau$ -extended method, it also depends on the  $\tau$ -threshold, since smaller  $\tau$ -values may trigger additional refinement near openings and increase the chance of wrapping internal structures.

*Output simplification.* The method may introduce more vertices than necessary near small or sharp concave features. This can happen when a vertex is inserted

close to, but not exactly at, an optimal location, followed by another nearby vertex insertion in a later refinement step. A post-processing simplification step could remove such redundant vertices, provided that the approximation quality, enclosure, validity and termination guarantees are preserved.

## Bibliography

- Attene, M., Campen, M., & Kobbelt, L. (2013). Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)*, 45. <https://doi.org/10.1145/2431211.2431214>
- Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A., & Silva, C. T. (2017). A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, 36(1), 301–329. <https://doi.org/10.1111/cgf.12802>
- Bischoff, S., Pavic, D., & Kobbelt, L. (2005). Automatic Restoration of Polygon Models. *ACM Trans. Graph.*, 24, 1332–1352. <https://doi.org/10.1145/1095878.1095883>
- The CGAL Project. (2021). *CGAL User and Reference Manual* (5.3). <https://doc.cgal.org/5.3/Manual/packages.html>
- Cheng, S.-W., Dey, T., & Levine, J. (2007). A Practical Delaunay Meshing Algorithm for a Large Class of Domains\*. *Proceedings of the 16th International Meshing Roundtable*, 477–494. [https://doi.org/10.1007/978-3-540-75103-8\\_27](https://doi.org/10.1007/978-3-540-75103-8_27)
- Huang, J., Zhou, Y., & Guibas, L. (2020). ManifoldPlus: A Robust and Scalable Watertight Manifold Surface Generation Method for Triangle Soups. *Arxiv*. <https://doi.org/10.48550/arXiv.2005.11621>
- Ledoux, H. (2013). On the Validation of Solids Represented with the International Standards for Geographic Information. *Computer-Aided Civil and Infrastructure Engineering*, 28(9), 693–706. <https://doi.org/10.1111/mice.12043>
- Ledoux, H. (2018). Val3dity: Validation of 3D GIS Primitives According to the International Standards. *Open Geospatial Data, Software and Standards*, 3(1), 1. <https://doi.org/10.1186/s40965-018-0043-x>
- Paden, I., Peters, R., García-Sánchez, C., & Ledoux, H. (2024). Automatic High-Detailed Building Reconstruction Workflow for Urban Microscale Simulations. *Building and Environment*, 265, 111978. <https://doi.org/10.1016/j.buildenv.2024.111978>
- Peters, R., Dukai, B., Vitalis, S., van Liempt, J., & Stoter, J. (2022). Automated 3D reconstruction of LoD2

and LoD1 models for all 10 million buildings of the Netherlands. *Photogrammetric Engineering and Remote Sensing*, 88(3), 165–170. <https://doi.org/10.14358/PERS.21-00032R2>

Portaneri, C., Rouxel-Labbé, M., Hemmer, M., Cohen-Steiner, D., & Alliez, P. (2022). Alpha Wrapping with an Offset. *ACM Trans. Graph.*, 41(4), 127:1–127:22. <https://doi.org/10.1145/3528223.3530152>

Saeedraashed, Y. S., & Benim, A. C. (2019). Validation Methods of Geometric 3D-CityGML Data for Urban Wind Simulations. *E3s Web of Conferences*, 128, 10006. <https://doi.org/10.1051/e3sconf/201912810006>

ShrinkWrap | Rhino 3-D Modeling. (n.d.). Retrieved January 17, 2026, from <https://docs.mcneel.com/rhino/8/help/en-us/commands/shrinkwrap.htm>

Wen, H., He, G., Xu, R., Chen, S., Xin, S., Shu, Z., Komura, T., Feng, J., Wang, W., & Tu, C. (2025). Feature-Preserving Mesh Repair via Restricted Power Diagram. *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers, SIGGRAPH Conference Papers '25*, 1–11. <https://doi.org/10.1145/3721238.3730671>

Zhao, Z., Ledoux, H., & Stoter, J. (2013). Automatic Repair of CityGML LOD2 Buildings using Shrink-Wrapping. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-2/W1, 309–317. <https://doi.org/10.5194/isprsannals-II-2-W1-309-2013>

## A Proof of termination

We adapt the proof of Appendix B of the 3D Alpha Wrap paper ((Portaneri et al., 2022)) to the extended traversability criterion and to the additional insertion rule R3, while keeping the notation of Appendix B. In particular we denote  $c$  and  $c_n$  as the two cells incident to a traversable facet  $f$ , where  $c$  is a cell labelled OUTSIDE and  $c_n$  is a cell labelled INSIDE. We also use the same notation of insertion radius: for a point  $p$  inserted in the current vertex set  $\mathcal{P}$ , its insertion radius  $r(p)$  is the Euclidean distance from  $p$  to  $\mathcal{P}$  before insertion.

Let  $f$  be a finite facet and let  $\Phi(f)$  denote the minimum radius of the Delaunay balls of  $f$ , as denoted in the original paper. In the extended algorithm,  $f$  is declared traversable if either  $\Phi(f) \geq \alpha$  or  $d(g_f, I) \geq \tau * \delta$ ,  $\tau > 1$ , where  $g_f$  is the centroid of  $f$ .

Furthermore let  $p_f$  be a point constructed in the interior of  $f$ , let  $k_{c_n}$  denote the circumcenter of the Delaunay ball of  $c_n$ , and let  $\Pi_{I(k_{c_n})}$  be the closest point on  $I$  to  $k_{c_n}$ .

We introduce the following additional insertion rule.

- Rule R3. If  $c_n$  intersects  $I$ ,  $d(g_f, I) \geq \tau * \delta$ ,  $\tau > 1$  and  $\Phi(f) < \alpha$ , insert the intersection between the segment  $[p_f; \Pi_{I(k_{c_n})}]$  and  $I_\delta$  closest to  $p_f$ , if it exists and satisfies  $r(p) \geq \min(\alpha, \delta, (\tau - 1) * \delta)$ . Otherwise, fall back to Rule R1 or Rule R2.

The next two lemmas are the extended counterparts of Lemmas 1 and 2 of Appendix B of the 3D Alpha Wrap paper.

**Lemma 1:** The insertion of a new point using Rule R3 breaks cell  $c_n$ .

PROOF. Let  $\mathcal{B}_{c_n}$  be the Delaunay ball of  $c_n$ , centered at  $k_{c_n}$ . Since  $c_n \subseteq \mathcal{B}_{c_n}$  and  $c_n \cap I \neq \emptyset$ , the closest point  $\Pi_{I(k_{c_n})}$  belongs to  $\mathcal{B}_{c_n}$ . Moreover,  $p_f$  is constructed in the interior of  $f$ . Since  $f$  is a facet of  $c_n$ , its vertices lie on the boundary of  $\mathcal{B}_{c_n}$ , and every interior point of  $f$  lies in the interior of  $\mathcal{B}_{c_n}$ . Hence  $p_f \in \text{int}(\mathcal{B}_{c_n})$ . By convexity of  $\mathcal{B}_{c_n}$ , the segment  $[p_f; \Pi_{I(k_{c_n})}]$  is contained in  $\mathcal{B}_{c_n}$ , and its open part is contained in  $\text{int}(\mathcal{B}_{c_n})$ . Therefore, any point inserted by rule R3 lies in  $\text{int}(\mathcal{B}_{c_n})$ , and thus breaks  $c_n$ .  $\square$

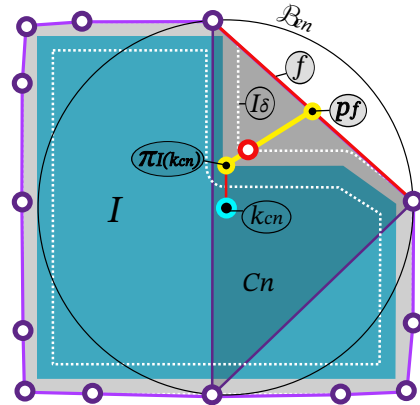


Figure A.1: Nomenclature for Lemma 1.

**Lemma 2:** The insertion radius of all inserted points is bounded from below:  $\forall p \in \mathcal{P}, r(p) \geq \min(\alpha, \delta, (\tau - 1) * \delta)$ .

PROOF. Let  $p$  be an inserted point. If  $p$  is inserted by Rule R1, then, exactly as in the original proof,  $r(p) \geq \Phi(f)$ , where  $f$  is the traversable facet that triggers the insertion. If  $f$  is traversable because  $\Phi(f) \geq \alpha$ , then immediately  $r(p) \geq \alpha$ .

Assume now that  $f$  is traversable because  $d(g_f, I) \geq \tau * \delta$ . Facets incident to the initial loose bounding box, will not get checked for traversability. Hence, when a facet is declared traversable, we may restrict to facets whose vertices are not bounding-box vertices. All vertices inserted after initialization are created on  $I_\delta$ . Therefore, if  $v_1, v_2, v_3$  denote vertices of  $f$ , then  $d(v_i, I) = \delta$ ,  $i = 1, 2, 3$ .

Moreover, moving a point can change its distance to  $I$  by at most the distance the point is moved. Hence, for any two points  $x$  and  $y$ ,  $|d(x, I) - d(y, I)| \leq \|x - y\|$ . Applying this with  $x = g_f$  and  $y = v_i$ , we obtain  $\|g_f - v_i\| \geq d(g_f, I) - d(v_i, I) \geq (\tau - 1) * \delta$ . Hence, the distance from  $g_f$  to each vertex of  $f$  is at least  $(\tau - 1) * \delta$ , which yields  $\frac{1}{3} \sum_{i=1}^3 \|v_i - g_f\|^2 \geq ((\tau - 1) * \delta)^2$ .

Let  $R_f$  be the circumradius of  $f$ , and let  $o_f$  be its circumcenter. Since  $\frac{1}{3} \sum_{i=1}^3 \|v_i - g_f\|^2 = R_f^2 - \|o_f - g_f\|^2 \leq R_f^2$ , it follows that  $R_f \geq (\tau - 1) * \delta$ . Since  $\Phi(f) \geq R_f$ , it follows that  $r(p) \geq \Phi(f) \geq (\tau - 1) * \delta$ .

If  $p$  is inserted by Rule R2, the original proof applies unchanged, and  $r(p) \geq \delta$ .

If  $p$  is inserted by Rule R3, it is accepted only if  $r(p) \geq \min(\alpha, \delta, (\tau - 1) * \delta)$ .

This proves the claim.  $\square$

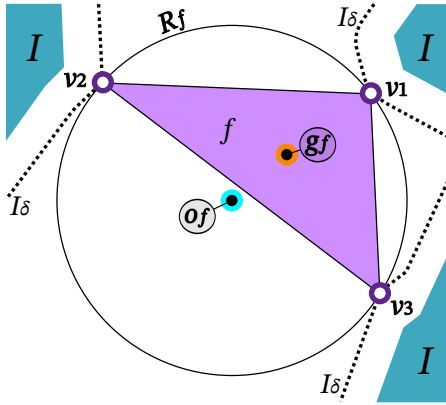


Figure A.2: Nomenclature for Lemma 2.

### Theorem 1

The extended algorithm terminates.

PROOF. By Lemma 2, every inserted point has insertion radius at least  $\rho = \min(\alpha, \delta, (\tau - 1) * \delta) > 0$ . The rest of the proof is identical to that of Appendix B of the 3D Alpha Wrap paper ((Portaneri et al., 2022)): a bounded region of  $\mathbb{R}^3$  can contain only finitely many pairwise disjoint open balls of fixed radius  $\frac{\rho}{2}$ . Hence only finitely many insertions are possible, and the algorithm terminates.  $\square$

## B Comparison Rhino ShrinkWrap

This appendix provides a qualitative comparison between the proposed  $\tau$ -extended 3D Alpha Wrap method and Rhino’s ShrinkWrap command. Rhino ShrinkWrap is included as a practical reference, since it can create closed meshes from defective or difficult-to-repair geometry (*ShrinkWrap | Rhino 3-D Modeling*, n.d.). Because the method is not open source, the comparison is intended as a visual reference rather than a direct algorithmic comparison.

The comparison is performed on the Delft University of Technology Aula building from 3D BAG. Rhino ShrinkWrap was run with a target edge length of 0.1m and polygon optimization set to 90%, resulting in 468,786 faces. The  $\tau$ -extended method was run with  $\alpha = 7.4$ , offset = 0.03, and  $\tau = 2$ , resulting in 37,420 output faces.

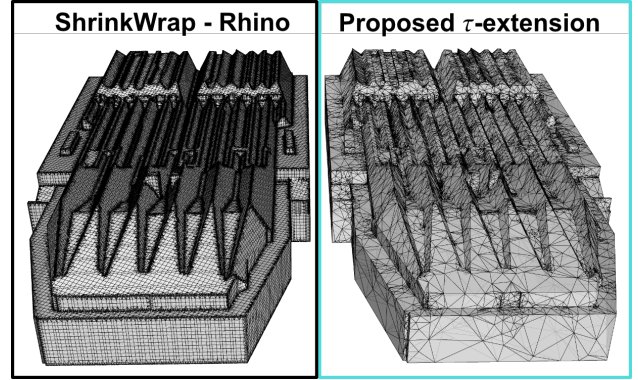


Figure B.1: Qualitative comparison between Rhino ShrinkWrap and the proposed  $\tau$ -extended 3D Alpha Wrap method for the Delft University of Technology Bouwkunde building. Despite producing substantially more faces, the ShrinkWrap result shows more rounded sharp features.

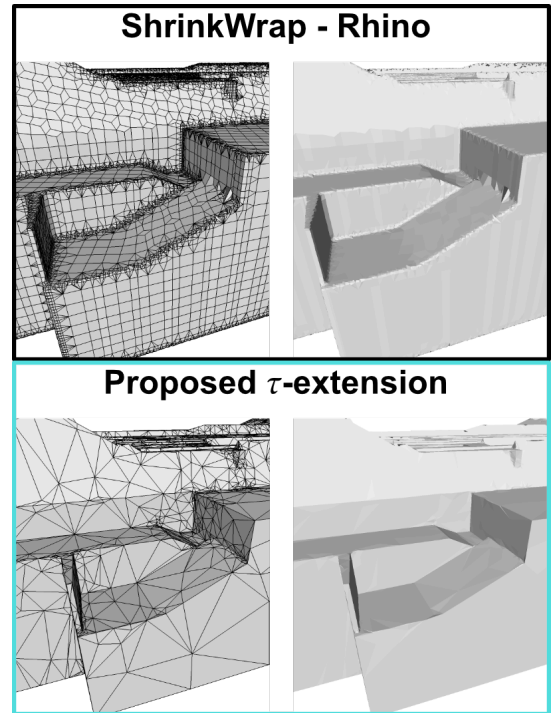
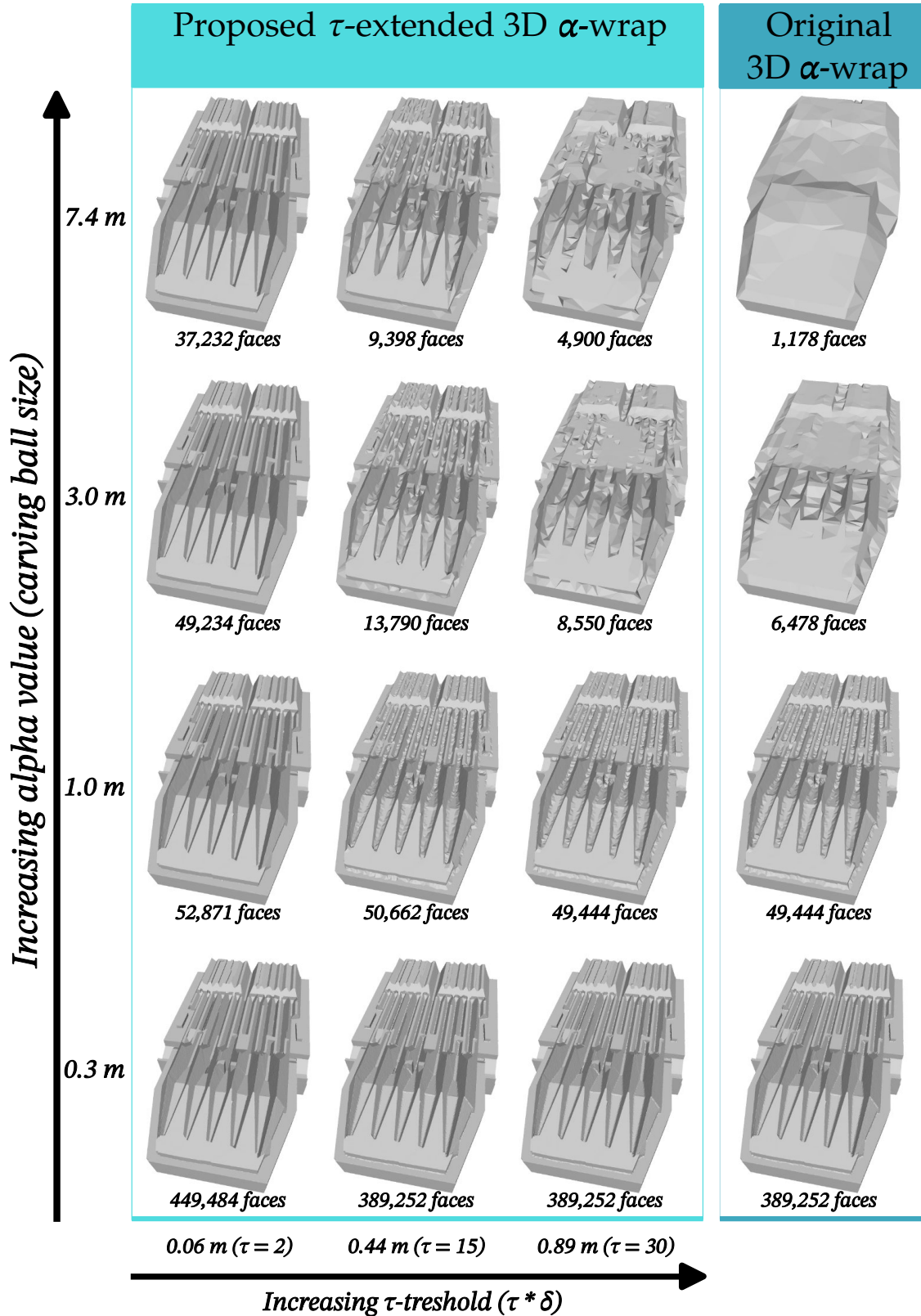


Figure B.2: Zoomed-in comparison shown with and without mesh edges. The edge views show the higher mesh density of Rhino ShrinkWrap, while the views without edges show that both convex and concave sharp features remain more rounded than in the  $\tau$ -extended result.

Rhino ShrinkWrap can produce a closed output mesh, but both convex and concave sharp features remain rounded even with a small target edge length and polygon optimization. The  $\tau$ -extended method preserves these features more clearly while using fewer output faces.

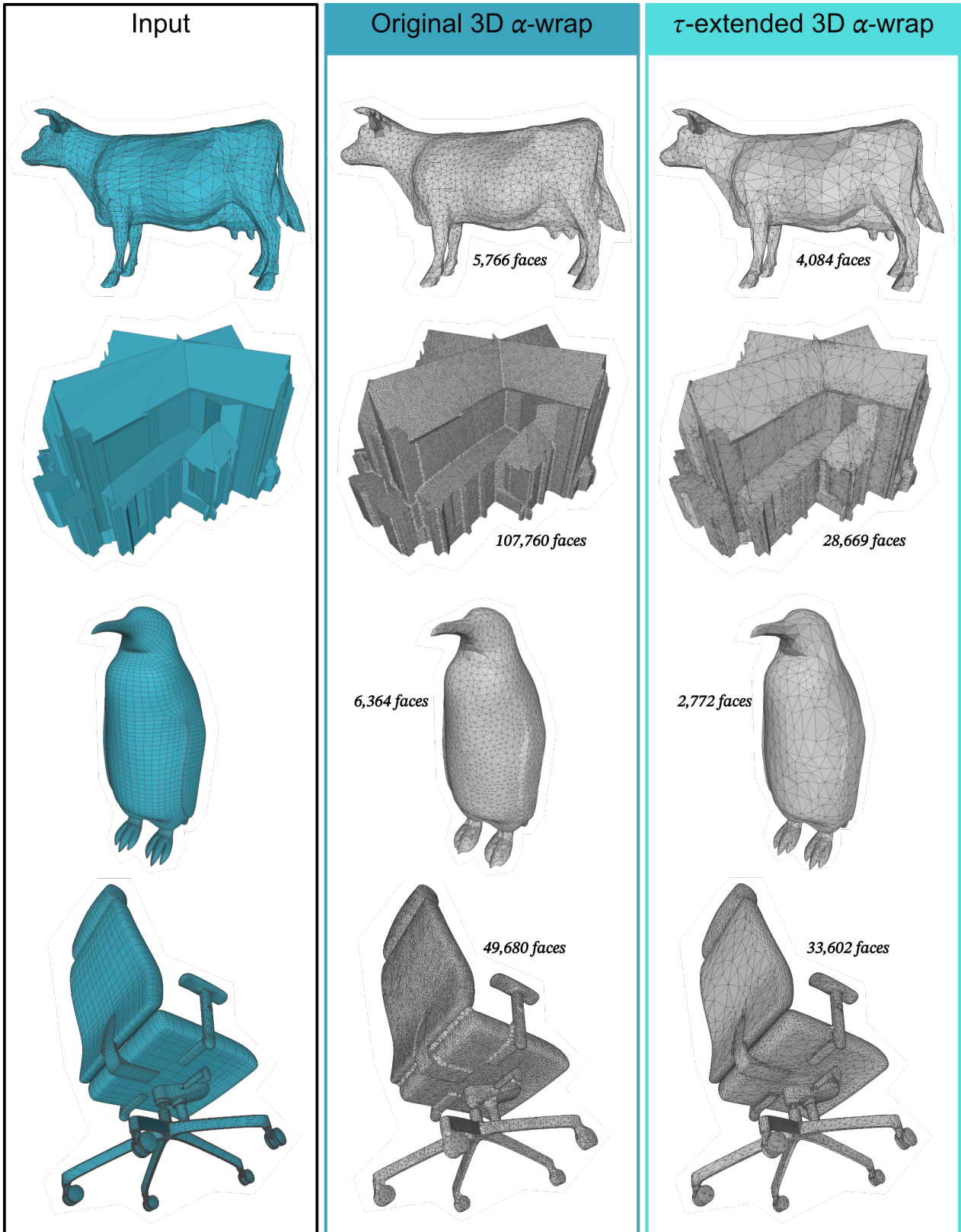


## E Qualitative effect of $\alpha$ and $\tau$



**Figure E.1:** Visual comparison of wrapped outputs for varying  $\alpha$ - and  $\tau$ -values on the Delft University of Technology Aula building from 3D BAG. The offset is fixed to  $0.03\text{m}$ . Both methods are shown for  $\alpha$ -values ranging from  $0.3\text{m}$  to  $7.4\text{m}$ . For the  $\tau$ -extended method,  $\tau$  is additionally varied from 2 to 30, corresponding to  $\tau$ -thresholds of  $0.06\text{m}$  to  $0.89\text{m}$ . The figure shows the global effect of decreasing  $\alpha$  and the additional local refinement introduced by the  $\tau$ -criterion.

## F Additional wrapping examples



**Figure F.1:** Additional qualitative examples of wrapped outputs generated with the original 3D Alpha Wrap algorithm and the proposed  $\tau$ -extended method. For each input geometry, the offset value is kept the same for both methods. The examples illustrate the type of meshes produced by both methods on different input geometries.



# Part 4. Discussion

## Part 4.1 Link with MSc Geomatics for the Built Environment

This thesis is closely connected to several core topics from the MSc Geomatics for the Built Environment. In Digital Terrain Modelling, concepts such as Delaunay triangulations, Voronoi diagrams, convex hulls, and alpha shapes are introduced as fundamental tools for representing and analysing spatial data. These concepts are also central to 3D Alpha Wrap: the algorithm relies on a 3D Delaunay triangulation, its dual Voronoi structure, and an alpha-based carving process, conceptually related to alpha shapes, to construct the output mesh.

The thesis also relates strongly to 3D Modelling of the Built Environment. In that course, 3D objects are studied through explicit and implicit representations, boundary representations, tetrahedralisations, isosurfaces, and validity concepts according to ISO 19107, such as manifoldness, orientability, and interior/exterior consistency. These ideas directly appear in this work: the goal is to generate a valid surface-mesh boundary representation that strictly encloses the input and is watertight, orientable, and manifold. The offset surface used in the algorithm can be interpreted as an implicit surface, and the algorithm uses the boundary mesh to separate space into inside and outside regions.

Finally, the work connects to wind modelling simulations in the urban environment, where the quality of building geometry is important for downstream simulation workflows. Although this thesis does not generate CFD volume meshes or evaluate simulation results, it focuses on a preceding geometric preparation step: obtaining valid and sufficiently detailed surface meshes of building geometry. This is relevant because automatic workflows for urban simulations depend on input geometries that are both geometrically valid and close enough to the intended building shape.

Together, these links show that the thesis sits at the intersection of computational geometry, 3D city modelling, and geometry preparation for built-environment applications. The work applies theoretical geometric concepts to a practical problem: automatically producing valid and detailed meshes from imperfect real-world input data.

## Part 4.2 Reflection on the research objectives

The main research question asked how CGAL's 3D Alpha Wrap algorithm can be extended to better preserve sharp concave features and reduce unnecessary mesh complexity, while retaining its guarantees. The developed method addresses this by adding a local  $\tau$ -criterion and a new insertion rule, R3, within the existing carving-based Alpha Wrap framework.

RQ1 asked whether local refinement can better preserve sharp concave features without increasing mesh complexity globally. The results show that this is possible for the tested use case. Instead of decreasing  $\alpha$  globally, the  $\tau$ -criterion triggers refinement only where the current wrap remains too far from the input. R3 then places new vertices more effectively near these concave regions, improving their approximation without refining the entire mesh.

RQ2 focused on retaining the theoretical guarantees of the original algorithm. The extension preserves the original carving structure, so the arguments for strict enclosure, watertightness, and manifoldness remain unchanged. Termination required an adapted proof, because both the traversability criterion and insertion strategy were modified. The proof shows two key properties: newly inserted vertices remain bounded from below in distance from previously inserted vertices, and each inserted vertex is placed in the cell that the algorithm intends to refine. Together, these properties ensure that only finitely many insertions are possible.

RQ3 asked whether the approach can be controlled with a small and interpretable set of parameters. The method keeps the original  $\alpha$  and offset parameters and adds only one new parameter,  $\tau$ . This parameter defines the maximum allowed local wrap-to-input distance relative to the offset distance. It therefore lets the user decide how closely the wrap should follow local geometric deviations before additional refinement is triggered. A smaller  $\tau$ -value triggers refinement earlier, while a larger  $\tau$ -value makes the method behave more like the original Alpha Wrap algorithm.

RQ4 considered the trade-off between approximation quality, output complexity, runtime, and validity. The experiments show that the  $\tau$ -extended method reaches lower directed Chamfer distances with fewer output faces than the original algorithm. Although R3 insertions are more expensive, fewer insertions are needed to reach lower

approximation errors, which reduced runtime in the tested use case. The validation experiments further show that the tested outputs remain valid.

Overall, the research questions are answered positively for the tested building datasets. The main remaining challenges are improving the  $\tau$ -criterion, reducing the cost of R3, and simplifying redundant output vertices without losing the original guarantees.

### Part 4.3 Limitations and future work related to MSc Geomatics

Several limitations remain for broader use in built-environment workflows. First, the method operates only on geometry. Semantic information, such as roof, wall, or ground labels, is not preserved automatically when a building model is wrapped. Future work could investigate how these attributes can be transferred to the output mesh or integrated into semantic reconstruction workflows.

Second, the method does not address input cleaning or general hole filling. Large holes can still cause internal structures to be included in the wrap, and outliers may also appear in the output because the mesh strictly encloses the input. Future workflows could therefore combine the method with preprocessing steps for outlier removal or hole handling.

Third, the output mesh could be simplified further after wrapping. The proposed method reduces unnecessary global refinement, but redundant vertices may still occur locally. A post-processing simplification step could make the output more compact, but should preserve approximation quality as well as the guarantees of 3D Alpha Wrap, including strict enclosure, watertightness, and manifoldness.

Finally, the ability of 3D Alpha Wrap to handle mixed input types could be explored further. Point clouds, triangle soups, and surface meshes could be combined in one input, for example to enrich plane-based building models with rounded or non-planar structures detected from point clouds. This could make the method useful for more detailed and flexible reconstruction workflows in the built environment.

## Bibliography

- Arroyo Otori, K., Ledoux, H., & Peters, R. (2024). *3D Modelling of the Built Environment*. <https://3d.bk.tudelft.nl/courses/geo1004/data/3dbook.pdf>
- Attene, M., Campen, M., & Kobbelt, L. (2013). Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)*, 45. <https://doi.org/10.1145/2431211.2431214>
- Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K. H., & Khoo, V. H. S. (2016). THE MOST COMMON GEOMETRIC AND SEMANTIC ERRORS IN CITYGMLDATASETS. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1, 13–22. <https://doi.org/10.5194/isprs-annals-IV-2-W1-13-2016>
- Bischoff, S., Pavic, D., & Kobbelt, L. (2005). Automatic Restoration of Polygon Models. *ACM Trans. Graph.*, 24, 1332–1352. <https://doi.org/10.1145/1095878.1095883>
- ISO 19107:2019. (2019,). <https://www.iso.org/standard/66175.html>
- Ledoux, H. (2013). On the Validation of Solids Represented with the International Standards for Geographic Information. *Computer-Aided Civil and Infrastructure Engineering*, 28(9), 693–706. <https://doi.org/10.1111/mice.12043>
- Ledoux, H. (2018). Val3dity: Validation of 3D GIS Primitives According to the International Standards. *Open Geospatial Data, Software and Standards*, 3(1), 1. <https://doi.org/10.1186/s40965-018-0043-x>
- Ledoux, H., Arroyo Otori, K., Peters, R., & Pronk, M. (2025). *Computational modelling of terrains*. Self-published. <https://doi.org/10.5281/zenodo.3992107>
- Lintermann, A. (2021). Computational Meshing for CFD Simulations. In K. Inthavong, N. Singh, E. Wong, & J. Tu (Eds.), *Clinical and Biomedical Engineering in the Human Nose: A Computational Fluid Dynamics Approach: Clinical and Biomedical Engineering in the Human Nose: A Computational Fluid Dynamics Approach* (pp. 85–115). Springer Singapore. [https://doi.org/10.1007/978-981-15-6716-2\\_6](https://doi.org/10.1007/978-981-15-6716-2_6)

- The Matplotlib Development Team. (2026,). *Cumulative Distributions*. [https://matplotlib.org/stable/gallery/statistics/histogram\\_cumulative.html](https://matplotlib.org/stable/gallery/statistics/histogram_cumulative.html)
- Pađen, I., Peters, R., García-Sánchez, C., & Ledoux, H. (2024). Automatic High-Detailed Building Reconstruction Workflow for Urban Microscale Simulations. *Building and Environment*, 265, 111978. <https://doi.org/10.1016/j.buildenv.2024.111978>
- Peters, R., & Ledoux, H. (2016). Robust Approximation of the Medial Axis Transform of LiDAR Point Clouds as a Tool for Visualisation. *Computers & Geosciences*, 90, 123–133. <https://doi.org/10.1016/j.cageo.2016.02.019>
- Peters, R., Dukai, B., Vitalis, S., van Liempt, J., & Stoter, J. (2022). Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands. *Photogrammetric Engineering and Remote Sensing*, 88(3), 165–170. <https://doi.org/10.14358/PERS.21-00032R2>
- Portaneri, C., Rouxel-Labbé, M., Hemmer, M., Cohen-Steiner, D., & Alliez, P. (2022). Alpha Wrapping with an Offset. *ACM Trans. Graph.*, 41(4), 127:1–127:22. <https://doi.org/10.1145/3528223.3530152>
- Saeedrashed, Y. S., & Benim, A. C. (2019). Validation Methods of Geometric 3D-CityGML Data for Urban Wind Simulations. *E3s Web of Conferences*, 128, 10006. <https://doi.org/10.1051/e3sconf/201912810006>
- Shumak, Y. (2025, March). *The Fundamentals of Mesh Quality in FEA*. <https://sdcverifier.com/structural-engineering-101/the-fundamentals-of-mesh-quality/>
- Wen, H., He, G., Xu, R., Chen, S., Xin, S., Shu, Z., Komura, T., Feng, J., Wang, W., & Tu, C. (2025). Feature-Preserving Mesh Repair via Restricted Power Diagram. *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers, SIGGRAPH Conference Papers '25*, 1–11. <https://doi.org/10.1145/3721238.3730671>
- What Is Meshing, Really?*. (n.d.). Retrieved January 17, 2026, from <https://gridpro.com/what-is-meshing>
- Zint, D., Maruani, N., Rouxel-Labbé, M., & Alliez, P. (2023). Feature-Preserving Offset Mesh Generation from Topology-Adapted Octrees. *Computer Graphics Forum*, 42(5), e14906. <https://doi.org/10.1111/cgf.14906>

## A Declaration of AI/LLM usage

AI/LLM tools, including ChatGPT and Codex, were used for this thesis to:

- **Improve report writing:** grammar checks, sentence structure, and word choice.
- **Explain concepts, papers, and code** that I did not fully understand.
- **Support code writing and debugging** during the implementation of my own ideas.

The main research ideas, results, analysis, and conclusions are my own.

## B Reproducibility self-assessment

The **data** used for this thesis is all publicly accessible;

- 3D BAG (Peters et al., 2022): <https://www.3dbag.nl/en/download>

All **code** is publically accessible through a GitHub repository: [https://github.com/michelbeeren/MSc\\_Geomatics\\_Thesis\\_3D\\_alpha\\_wrap](https://github.com/michelbeeren/MSc_Geomatics_Thesis_3D_alpha_wrap)

