HSW: Heuristic Shrink-wrapping for automatically repairing solid-based CityGML LOD2 building models

Junqiao Zhao[1], Hugo Ledoux[2], Jantien Stoter[2], and Tiantian Feng[1]

[1]Tongji University
[2]Delft University of Technology

Zhao, J., Ledoux, H., Stoter, J., and Feng, T. (2018). HSW: Heuristic shrink-wrapping for automatically repairing solid-based CityGML LOD2 building models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 146(289–304).
DOI: `http://dx.doi.org/10.1016/j.isprsjprs.2018.09.019`

The Level-of-Detail (LOD) 2 building models defined in CityGML are used widely in three-dimensional (3D) city applications. Many of these applications demand valid solid-based geometry (closed 2-manifold), which is crucial for analytical and computational purposes. However, this condition is often violated in practice because of the way LOD2 models are constructed and exchanged. Examples of the resulting errors include missing surfaces, intersecting building parts, and superfluous interior geometry. In this study, we present a heuristic shrink-wrapping algorithm for reconstructing valid solid-based LOD2 buildings by repairing and generalizing invalid input models. A single building model is first decomposed as intersection-free and reassembled by constrained tetrahedralization. The bounding membrane is then shrunk by incrementally carving the selected boundary tetrahedra and wrapping the expected shape of the building. In the algorithm, combinations of heuristics are proposed to guide the carving process. Topological and geometrical constraints are proposed to ensure the validity and exactness of the output model. The semantics of the input geometry are preserved and missing semantics are deduced based on pragmatic rules. We evaluated the performance of the algorithm using 3D building models, including CityGML datasets. The results showed that our method achieved state-of-the-art performance at repairing 3D building models.

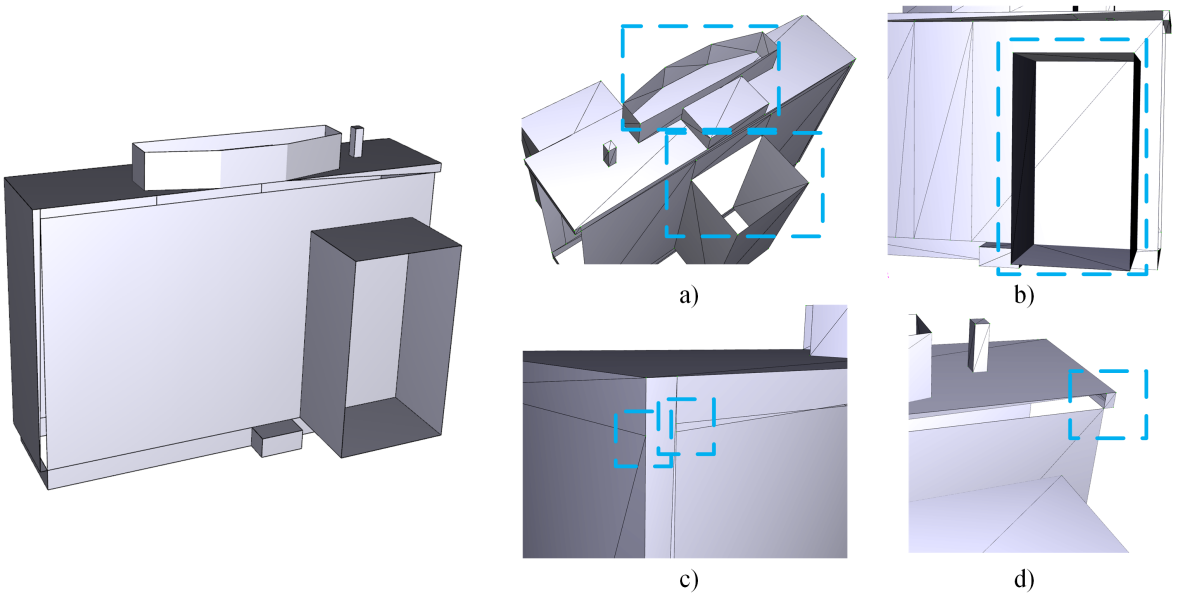


Figure 1: Geometrical and topological errors found in CityGML models: a) holes and unwanted interior geometry; b) intersecting components; c) collapsed faces; and d) non-2-manifolds.

1 Introduction

The three-dimensional (3D) Level-of-Detail (LOD) 2 building models defined in CityGML are used widely in GIS applications, such as city planning, navigation, and environmental analysis (Kolbe, 2008; Kolbe et al., 2008; Biljecki et al., 2015). Although not mandatory, the desired geometrical type for these models is solid without interior shells, and the boundary forms a closed 2-manifold, iethe *ComposedSurface* (Herring, 2005; Gröger et al., 2012). However, a LOD2 building model is often modeled as *multisurface* in practice due to the error-prone data acquisition or modeling processes (Wagner et al., 2013). As a result, geometrical and topological errors such as gaps and unwanted interior geometry (Fig. 1a), intersection between surfaces (Fig. 1b), degenerated edge and surface primitives (Fig. 1c), and non-manifold situations (Fig. 1d) are present, which violate the criteria for a valid solid-based LOD2 building (Zhao et al., 2014).

These are only a subset of the cases explored by Biljecki et al. (2016) regarding the quality of existing CityGML datasets. These flawed models can eventually lead to serious errors or even crash downstream applications. Therefore, they should be validated and repaired before use (Ledoux, 2013; Biljecki et al., 2016).

Repairing 3D models, especially those with a triangular mesh, is a popular research area in the fields of computer-aided design and computer graphics (Mezentsev and Woehler, 1999). However, most of the existing methods were proposed for continuous meshes and few can handle regular shaped 3D building models (Campen et al., 2012). In the GIS field, 3D repair has become an important topic in recent years because of the increasing availability of 3D city models worldwide. Great success has been achieved in the validation of 3D building models and several tools are now available (Kazar et al., 2008; Ledoux, 2013; Karki et al., 2010). However, the capabilities of the existing repair methods for 3D building models are still restricted (Bogdahn and Coors, 2010; Wagner et al., 2013; Biljecki et al., 2016; Mulder, 2015; Steuer et al., 2015).

In this study, we propose an automatic repair method based on the idea of shrink-wrapping (Sec. 3). This method simulates a process where an approximate membrane, iethe boundary

shell of the approximation, is shrunk and it finally wraps the exact shape of an object. Thus, a closed-2 manifold reconstruction of the object can be obtained. Moreover, the semantics attached to the surface of the original model are preserved and the missing semantics can be completed. This method is implemented by carving a tetrahedra-based representation heuristically while conforming to validity rules.

The proposed approach is a substensial extension of our previous paper (Zhao et al., 2013). Although both work share the shrink-wrapping framework for repairing, in this paper, we elaborated the newly introduced heuristics and geometrical constraints. Semantics defined in CityGML was also incorporated. We evaluated the proposed method using real CityGML datasets composed of thousands of buildings. The results obtained using the proposed approach were compared with those produced by state-of-the-art mesh-repair methods.

We describe the handling of intersections and the constrained tetrahedralization (CT) method in (Sec. 3.1). The heuristic carving process as well as the geometrical and topological constraints for guaranteeing the validity of the output and maintaining the shape of the building are explained in Sec. 3.2 and Sec. 4. Our experimental results (Sec. 5) demonstrate that typical errors such as gaps in the model, holes in the surface, intersections between building parts can all be repaired using the proposed method, and superfluous interior geometry either left by the automatic conversion from IFC (Industrial Foundation Class) models to CityGML or generated by the intersecting geometrical parts can also be removed.

2 Related work

In the following, we discuss two areas of related research: the geometrical repair of 3D models and the shrink-wrapping algorithm.

2.1 Geometrical repair of 3D models

As mentioned in Sec. 1, the repair of 3D city models is still in its early stages. In this section, we mainly consider the repair of generic 3D models, iemesh models. The existing methods can be classified into two broad categories: local and global approaches.

Local approaches deal with each of the defects locally on the mesh based on the intrinsic combinatorial structure, egfilling holes (Liepa, 2003), splitting non-manifold edges (Guéziec et al., 2001), and removing intersections (Campen and Kobbelt, 2010). This type of approach usually solves one or a few types of errors, but it may introduce new errors, as described by Campen et al. (2012), egfilling a hole might introduce intersections with another part of the mesh. Moreover, local approaches are vulnerable to unexpected error types in the input model and they may not function properly when these errors are present.

Global approaches usually employ volume-based representations, ievoxels, tetrahedra. These methods first try to repair the volumetric representation of the object and then reconstruct the surface, and thus they are robust. Nooruddin and Turk (2003) used voxels to represent the spaces occupied by the input mesh. Geometrical errors such as holes are healed by morphologic operators defined for voxels. The final mesh is derived using isosurface extraction. However, each voxel has to be classified correctly during repair by using an expensive multi-ray stabbing method, and unwanted discretizing artifacts are introduced into the result, even when smoothing and mesh optimization processes are applied subsequently. This is especially problematic for regular and planar surfaces such as those of 3D buildings.

Bischoff and Kobbelt (2005) proposed a structure-preserving method based on octrees, which helps to locally repair only the erroneous region in order to retain the sharp features. However, this method does not solve self-intersections. A similar method proposed by Bischoff et al. (2005) also repairs the model locally, and it employs an octree and BSP mixed structure to

represent the input. However, this method is complex to implement and local sampling still introduces artifacts in the results.

Further detail of mesh repair have been reviewed in previous studies (Campen et al., 2012; Attene et al., 2013). Steuer et al. (2015) proposed a volumetric repair method for 3D building models, where the exterior shell is extracted using the marching cube method, which causes discretization artifacts. Mulder (2015) evaluated two workflows for recovering the shape of a 3D building model based on its volumetric representation, iethe marching cube and dual contouring algorithms, and showed that the latter is more desirable because of its ability to retain sharp features. However, the repaired models still contain various artifacts. Recently, a robust rearrangement method for an arbitrary input mesh was proposed for robust constructive solid geometry (CSG) operations (Zhou et al., 2016). This method explores the arrangement of the geometry in its embedding space and extracts the desired boundary, as also did in our proposed method. However, it does not solve the repair problem.

2.2 Shrink-wrapping algorithm

The concept of shrink-wrapping is not new in geometry processing. Kobbelt et al. (1999) introduced a remeshing method that uses the initial mesh with the desired property as a membrane, and then a force is applied to each of the vertices to move the vertices accordingly. However, this method is not suitable for flawed models with holes or intersections. Koo et al. (2005) used a similar method to shrink wrap an unstructured point cloud to reconstruct the mesh model. Voxels are used to structure the points and thus smoothing has to be applied at the end. This approach cannot deal with surface models.

In order to simplify a polygonal mesh, Hagbi and El-Sana (2010) proposed a tetrahedron carving approach that generates several topologically simplified models. The tetrahedra are incrementally carved according to specific criteria. However, this method only accepts valid inputs, so it cannot deal with geometrical or topological errors. Furthermore, their carving operation may produce non-manifold results, which must be fixed using a stitching and cutting method (Guéziec et al., 2001).

A similar approach involves the approximation of an input polygon soup based on implicit surfaces (Shen et al., 2004). However, interpolation using an implicit surface cannot preserve the tessellation of the input surfaces and it also introduces a smoothing effect, which is not desirable for 3D buildings. Hétroy et al. (2011) also proposed a membrane shrinking-based method for repairing a mesh. However, this method requires manual guidance for shrinking, and it employs voxel-based morphological operations to shrink the model.

3 Repair by Heuristic Shrink Wrapping

In this study, we propose a repair approach based on the shrink-wrapping concept. This approach is a top-down method and it ensures that the intermediate output always has valid solid geometry. The geometrical and topological errors from the input building model are repaired in a uniform process, and a valid LOD2 building model is obtained.

To implement this idea, a "membrane" first has to be constructed for the input model, which should be valid and easy to build. A step-by-step shrinking process is then applied, where two options are possible. The first involves directly deforming the primitives of the membrane, i.e., vertices or faces, toward the correct positions for the original input model. However, the exact correspondences between the primitives of the membrane and the input model are difficult to build, especially for an input with non-zero genus, or even with holes or intersections (Kobbelt et al., 1999). The second option involves developing the process implicitly based on an auxiliary structure, such as voxels or tetrahedra. As discussed earlier, the voxelization of an input model has difficulty preserving either the exact shape of the model or its tessellation. The latter defect

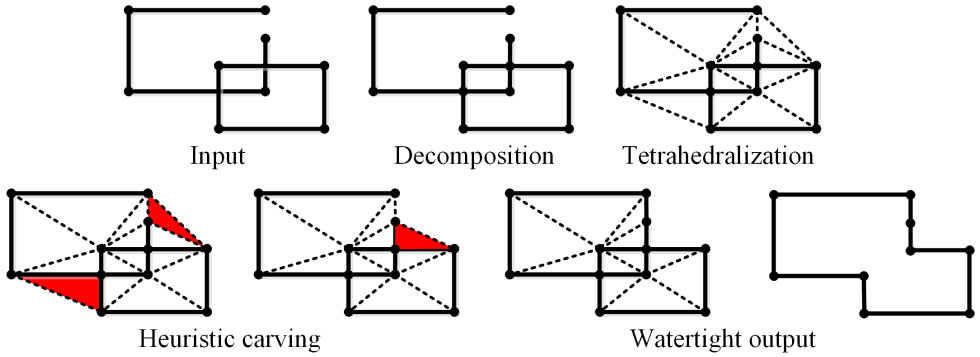


Figure 2: Work flow of our shrink-wrapping method.

is particularly unacceptable for the repair of an input model with properties attached to its surface primitives, such as a CityGML model. Therefore, CT is adopted in the second option for structuring the input model and to provide elements for the shrinking process.

Our shrinking process involves incrementally eliminating the excess tetrahedra while preserving the preferred tetrahedra, and finally wrapping the correct exterior shell of the model. This process can be implemented by a carving operation in a similar manner to the method used by Hagbi and El-Sana (2010). However, rather than removing details from a valid input, an appropriate carving strategy should be proposed for repair.

Fig. 2 demonstrates the work flow of our method in 2D. The steps comprise:

- step1: Decomposition and CT,
- step2: Heuristic carving,
- step3: Exterior shell extraction and semantic completion.

3.1 Decomposition and CT

Our method accepts any arbitrary input geometry, which means that geometries may intersect with each other and they may contain various types of defects. These defects cause the failure of the successive tetrahedralization step, and thus they should be cleaned up first by decomposition.

The decomposition step triangulates all the input polygons (Fig. 3a–b) and detects all of the different types of intersections between the triangles. The intersecting triangles are then subdivided at the intersection elements, iea vertex, an edge, or even a polygon. The result is a valid simplicial 2-complex (Fig. 3c), which can be further improved by coplanar merging, as shown in Fig. 3d.

CT reconstructs an intersection-free geometrical set into 3-simplices (tetrahedra) that are non-overlapping, and all the input *constraints*, i.e., triangle surfaces from the input model, are represented by facets in the tetrahedralization results.

3.2 Carving tetrahedra to obtain the 2-manifold

The carving operation is applied to the *candidate tetrahedra*, which are tetrahedra located on the membrane. They are referred to the *candidate triangles* and their detailed definition is given in Sec. 4.1.

After a carving operation, a *candidate tetrahedron* is removed and its adjacent tetrahedra are then converted into candidates if they are not on the boundary, as shown in Fig. 4 a–b. The membrane is then shrunk once.

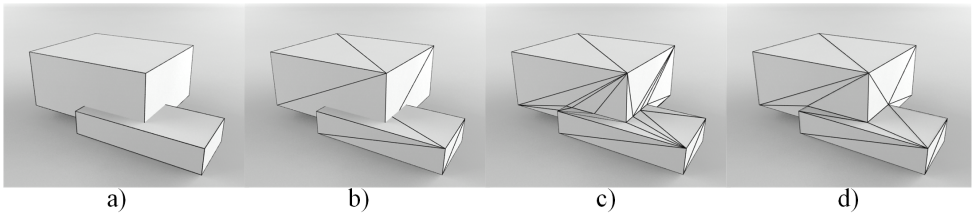


Figure 3: Decomposition of the intersected polygons.

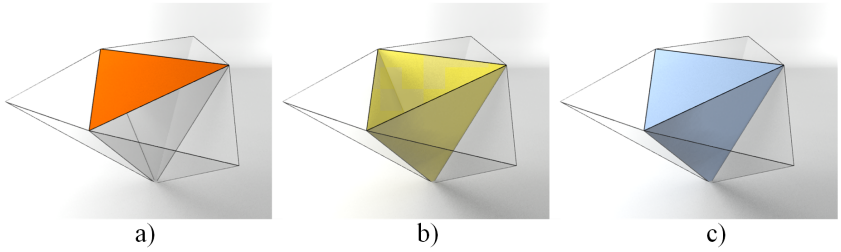


Figure 4: The carving operation where red indicates the *candidate triangle* for the selected *candidate tetrahedron* that needs to be carved, and yellow indicates the *candidate triangles* for the *candidate tetrahedra* after carving and whether the *candidate tetrahedron* in (a) should not be carved. All of the member triangles are tagged as *anchor/preserved* in blue. These colors are used consistently in this study).

Given that the aim is repair, we must be careful about the validity of carving because it may produce an invalid geometry (non-2-manifold) in the output. Therefore, validity rules are required to constrain the carving process, as explained in Sec. 4. If carving a *candidate tetrahedron* breaks any of the constraints, the tetrahedron is preserved or postponed for carving (Fig. 4c).

The validity of geometry, iethe 2-manifoldness, is defined based on its topological structure, so the validity rules are referred to as *topological constraints* in this study. Moreover, carving must also consider the shape of buildings, egtypical man-made objects such as buildings are rich in planar features. Therefore, we must preserve the tetrahedra that form these features during carving, so *geometrical constraints* are also proposed.

The carving operation only eliminates one tetrahedron each time, so an optimal sequence is required to guide the consecutive carving processes. It might be possible to formalize the carving costs using an optimization technique, as shown by Guercke et al. (2011). However, a heuristic method is introduced instead in this study, which is rapid and it yields promising results, as shown later.

Finally, the exterior shell is extracted from the remaining boundary tetrahedra.

3.3 Semantic completion

During carving, our method preserves the semantic information from the input and it also deduces the semantics for the newly generated surfaces. If we consider the exterior of a CityGML building model as an example, then the semantic information attached to the model comprises six types of boundary surface (RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface, and ClosureSurface) with two types of opening (Window and Door). In the decomposition stage, all of these semantics are retained with every input polygon and they are propagated to the decomposed intersection-free triangles.

After the carving process is finished, we first traverse all of the exterior triangles and validate the existing semantics based on normal vector-based rules. These rules for building boundaries have been employed in several studies on the generation of LOD2 models defined in CityGML,

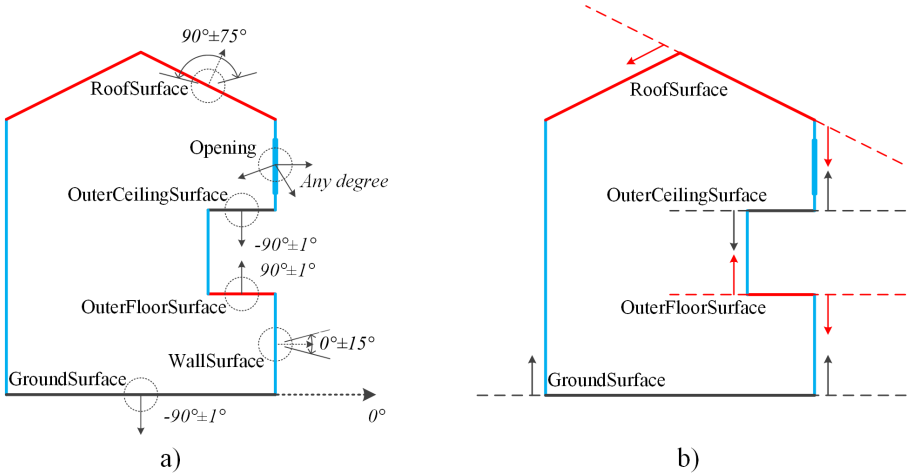


Figure 5: a) Rules based on normal directions for different types of boundary surfaces. b) Difference between an OuterCelingSurface (OuterFloorSurface) and GroundSurface (RoofSurface).

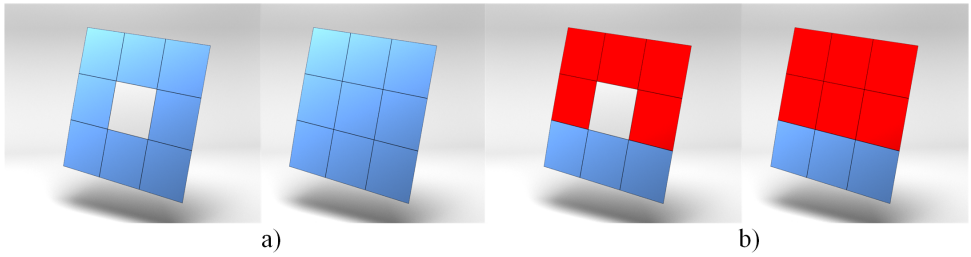


Figure 6: Semantic deduction based on neighboring faces: a) the undefined face has homogeneous coplanar neighbors; and b) the undefined face has heterogeneous coplanar neighbors.

and they are known to be effective (Donkers et al., 2016; Boeters, 2013). We consider a setting similar to that shown in Fig. 5a, with a vertical surface and its normal pointing to the horizontal direction is probably a WallSurface. A horizontal surface with its normal pointing downward should be a GroundSurface or an OuterCeilingSurface. By contrast, a horizontal surface with its normal pointing upward should be a RoofSurface or a OuterFloorSurface. The OuterCelingSurface and the OuterFloorSurface are distinguished by their neighbors, where they both have two neighboring WallSurfaces on different sides of the surface, but the GroundSurface or RoofSurface should have all their neighboring WallSurfaces on the same side of the surface, as shown in Fig. 5b. The remaining surfaces are all defined as RoofSurfaces.

Based on these rules, we deduce the semantics for the newly generated boundary surface according to the following situations (as illustrated in Fig. 6).

- If the newly generated surface has coplanar neighboring surfaces with the same semantics, then the semantics of the surface are defined as the same as those of its neighbors
- If the newly generated surface has coplanar neighboring surfaces with heterogeneous semantics, then the semantics of the surface are defined as the dominant semantics among its neighbors
- If the newly generated surface does not have coplanar neighboring surfaces, then its semantics are defined according to the normal vector-based rules.

4 Proposed heuristic carving rules

4.1 Categorizing triangles and tetrahedra

As mentioned earlier, the carving operation is applied to *candidate tetrahedra*. These tetrahedra are located on the boundary of the CT result and they are depicted based on the types of their member triangles. In our method, all of the triangles from the input model and its convex hull are classified into three categories: *anchor triangles*, *candidate triangles*, and *preserved triangles*.

The *anchor triangles* comprise all the triangles in the input model after decomposition. They are treated as the "anchors" in our method and retained during the carving process. We trust most of the input geometry, except some superfluous faces that may disturb the carving process, especially when the faces form dangling surfaces. These triangles are prevented by constraints described in Sec. 4.2.2.

The *candidate triangles* are newly emerged boundary triangles on the membrane. They also define the *candidate tetrahedra* that can be carved where possible. The *candidate tetrahedra* for carving are defined as tetrahedra that contain at least one *candidate triangle* member. After a *candidate tetrahedron* has been carved, its neighboring undefined triangle members are converted into *candidate triangles* on the fly (as shown in Fig. 4b).

The *preserved triangles* are the non-anchor member triangles of a tetrahedron that are preserved during the carving process. These triangles are also retained untouched during the subsequent carving processes. In contrast to the *anchor triangles*, they do not contain semantic information.

4.2 Carving constraints

The criteria for decisions regarding a carving operation are specified as constraints. We consider two basic constraints that are essential for the repair of building models, i.e., *topological constraints* and *geometrical constraints*. If carving a *candidate tetrahedron* breaks any of the *topological constraints*, the tetrahedron should be excluded from carving because the carving will introduce invalid non-2-manifold cases in the results. The *geometrical constraints* preserve the shape characteristics of a building model, such as maintaining *anchor triangles*, patching a surface in a nested hole, and retaining the consistent orientations of surfaces.

4.2.1 Topological constraints

During the carving process, it is essential to detect whether a non-2-manifold situation will be introduced. However, if a non-manifold situation occurs after carving, this does not necessarily mean that the final result will also be invalid because a temporarily invalid situation might be resolved in the subsequent carving operations. Therefore, we validate the carving operation according to *topological constraints* defined based on both the *candidate tetrahedron* and the neighboring tetrahedra affected by subsequent carvings.

Non-manifold situations can occur at facets, edges, or vertices (Botsch et al., 2007). Thus, the manifoldness of all the primitives of the neighbors of a *candidate tetrahedron* should be ensured.

Topological constraint for triangles For a triangle, the invalid case involves the triangle becoming a dangling triangle after carving, as shown in Fig. 7. Our method preserves the *anchor triangles*, so each of them should have at least one neighboring tetrahedron after carving. Therefore, the topological constraint for triangles is defined as follows.

Topological constraint I (Constraint I_t):

$$\exists\, f \in T \;\wedge\; f \in \textit{membrane} \;\wedge\; f \in \textit{anchor}\,(\textit{preserved})\;\textit{triangles} \Rightarrow\; T \in \textit{preserved tetrahedra} \quad (1)$$

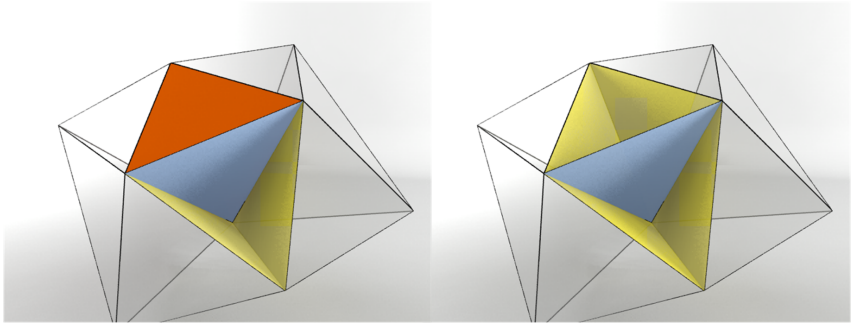


Figure 7: Topological constraint for triangles where carving the *candidate tetrahedron* in a) (in red) leaves the *anchor triangle* as a dangling triangle (in blue).

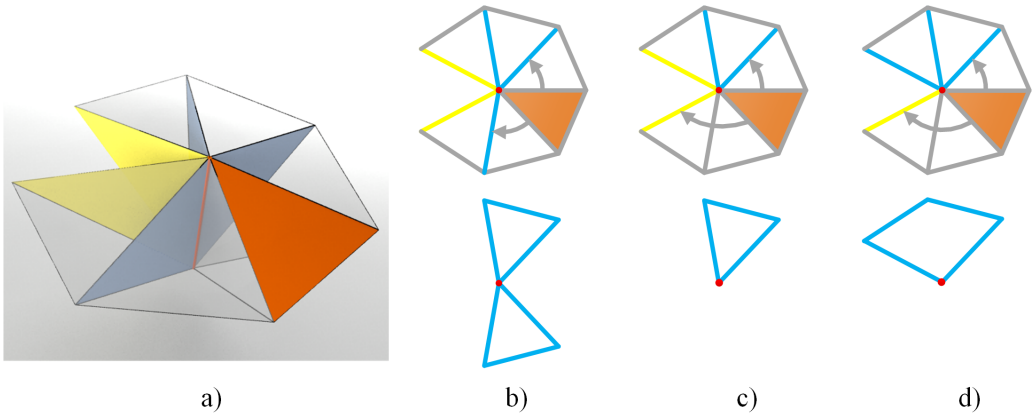


Figure 8: Topological constraint for edges where red indicates the *candidate triangle* that needs to be carved, yellow indicates the *candidate triangle* on the membrane, and blue indicates the *anchor/preserved triangle*.

where f is a triangle of the *candidate tetrahedron* T.

Topological constraint for edges For an edge, it is necessary to ensured that carving will not change the edge into a complex edge, ie, an edge shared by more than two triangles. To examine the validity of an edge, we traverse all of the incident triangles around the edge of a *candidate tetrahedron*, as shown by the red edge in the center of Fig. 8a.

Clearly, the validity has to be checked only when the edge of the *candidate tetrahedron* that needs to be carved is already on the membrane (as shown by the red edge where the two yellow *candidate triangles* join).

To avoid invalid cases, carving operations around this edge should finally result in a single connected component (where the boundary forms a 2-manifold), and thus if carving is permitted, the visited tetrahedra should be carved continually and simultaneously.

To better understand their relationships, we project the edge and its incident triangles into 2D (Fig. 8b–d), where the edge is mapped as a point in the center and all of the triangles are mapped as edges attaching to the point.

As shown in Fig. 8b, the carving operation should not break *anchor triangles* (represented by blue lines), so we eventually break the model into two components touching at one edge. However, as shown in Fig. 8c–d, carving in a clock-wise order will finally reach the existing boundary, so valid results can be obtained. Thus, we define this topological constraint as follows.

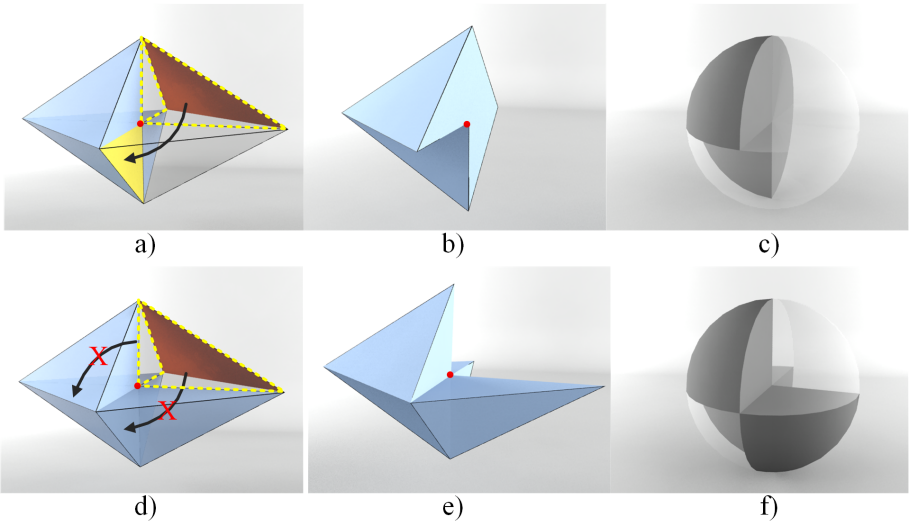


Figure 9: Topological constraint for vertices where red indicates the *candidate triangle* that needs to be carved, yellow indicates the *candidate triangle* on the membrane, and blue indicates the *anchor/preserved triangle*. The sphere maps in b) and e) are shown in c) and f), where the black half-spaces indicate the "occupied" volumes and the white half-spaces indicate the "void" volume (Granados et al., 2003).

Topological constraint II (Constraint II_t):

$$\exists\, e \in T \wedge e \in \textit{membrane} \wedge \exists\, \textit{anchor}\ f_1,\ f_2 \wedge e = f_2 \cap f_2 \wedge \\ f_1\ f_2\ \textit{are isolated by}\ T\ \textit{in star triangles of}\ e \Rightarrow T \in \textit{preserved tetrahedra} \quad (2)$$

where e is a hidden edge[1] of *candidate tetrahedron* T and f_i is a triangle.

Topological constraint for vertices The third topological constraint is defined for vertices. If a vertex is shared by more than one disconnected components, then it is called a *singular vertex*, which is a non-2-manifold case. Fig. 9 shows the *star tetrahedra* for a vertex of a *candidate tetrahedron*. Similar to the constraint for edges, because this vertex is already on the boundary of the membrane (as shown in Fig. 9b and e), its validity is checked by counting the number of incident disconnected components. This can be interpreted as determining whether a carving path made of homogeneous neighbors (the adjacent non-preserved tetrahedra) can be found that connects any one of the *candidate triangles* of the *candidate tetrahedron* with the membrane.

If no such path exists, the volume around the vertex will be divided into disconnected components after carving (as shown in Fig. 9d and e), which should be prevented. Otherwise, all of the tetrahedra along this path should be tested and carved so no more half-spaces are introduced into the neighborhood of the vertex (Fig. 9b). Therefore, the last topological constraint is defined as follows.

Topological constraint III (Constraint III_t):

$$\exists\, v \in T \wedge v \in \textit{membrane} \wedge \exists\, \textit{anchor}\ F = \{f_i | i = 1, ..., n\} \wedge v = \bigcap_{i=1}^{n} f_i \\ \wedge\ T\ \textit{is isolated from the opposite membrane by}\ F \Rightarrow T \in \textit{preserved tetrahedra} \quad (3)$$

[1] A hidden edge means that neither of its two incident triangles from the candidate tetrahedron presents on the membrane, except the edge itself.

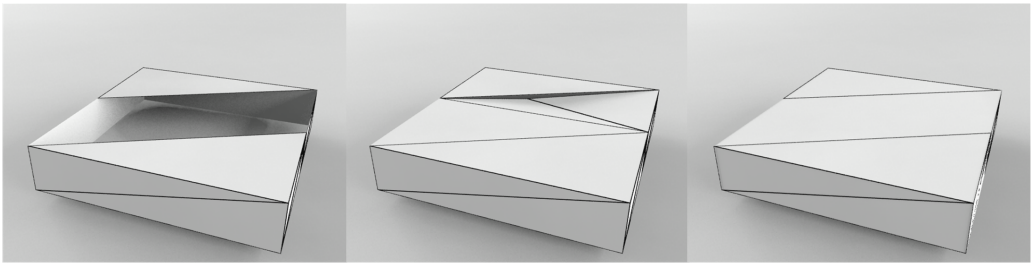


Figure 10: A box with holes repaired with and without geometrical constraints. a) shows the broken box with holes. b) shows the repaired solid without constraints, in which a concave surface is produced. c) shows the repaired solid with constraints, which is bounded by flat surfaces.

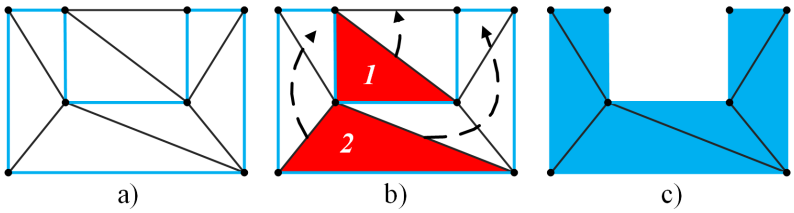


Figure 11: A U-shaped building model with a hole at bottom (a-c) where the carving process is projected onto 2D and the blue edges represent *anchor triangles*.

where v is a hidden vertex[2] of the *candidate tetrahedron* T and f_i is a triangle.

Multiple paths may exist so the shortest carving path around the vertex is found using Dijkstra's algorithm, thereby minimizing the differences before and after carving.

4.2.2 Geometrical constraints

Due to the *topological constraints*, each step of shrinking-wrapping will yield a closed 2-manifold. However, these constraints do not consider the geometrical aspects of the input model. For example, if holes are present in the input model (Fig. 10a), the carving process may excavate these areas on the membrane to yield a valid but undesired concave shape on the output surface, as shown in Fig. 10b. This is particularly undesirable for building models because these models are often formed of planar features. Thus, *geometrical constraints* are introduced concerning the shape characteristics of a building.

Geometrical constraint I The first geometrical constraint is proposed for filling holes. This constraint is based on the connected coplanar neighbors of a *candidate triangle*. If these triangles are bounded by an *open boundary* formed by *anchor triangles*, then this implies that these triangles can be a missed planar surface in the input model, as shown by the example in Fig. 11. A building model with a missing ground plane is a common error in practice. During repair, the coplanar triangles of candidate triangle 2 in Fig. 11b are bounded by the *anchor triangles* shown by blue edges. Therefore, carving should be prohibited inside this area, which results in the filled hole in Fig. 11c. Otherwise, if these triangles are not bounded (as shown by triangle 1 in Fig. 11b), they are interpreted as the auxiliary geometry generated from tetrahedralization and they can be carved (Fig. 11c). This constraint is referred to as *geometrical constraint I* in our

[2]The hidden vertex means that none of its three incident triangles from the candidate tetrahedron presents on the membrane, except the vertex itself.

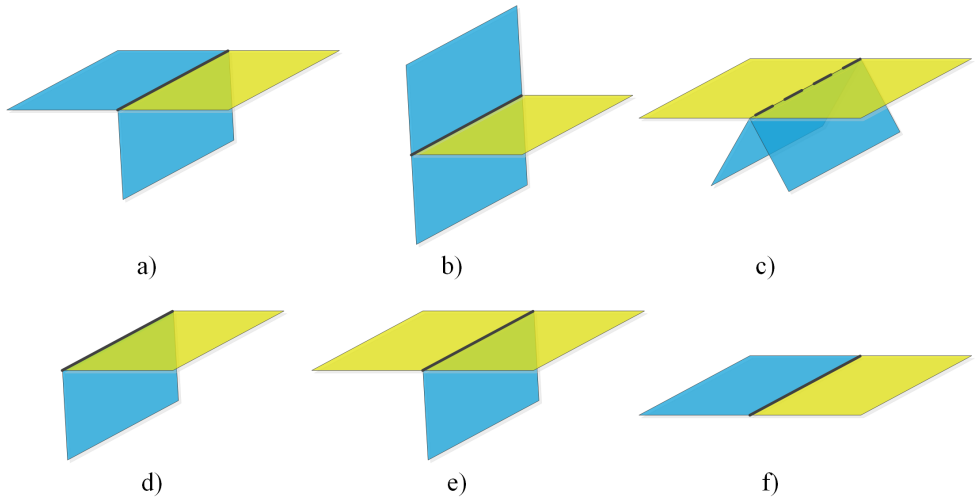


Figure 12: Definitions of open boundaries shown in black lines: (a–c) the *anchor triangles* already form 2-manifolds so they are not cases of open boundaries, but these cases can be treated as boundaries, egfor generalization; and (d–f) open boundaries of a hole.

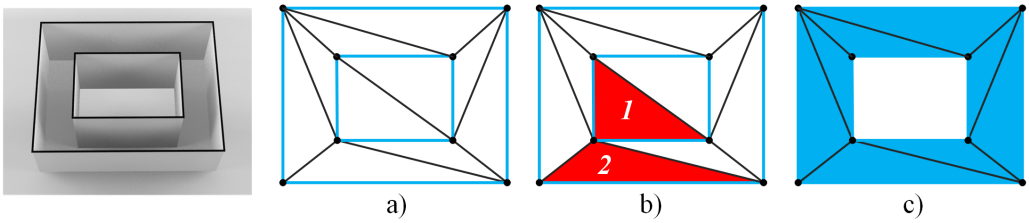


Figure 13: A tubular-shaped building model with a hole at bottom (a–c) where the carving process is projected onto 2D and the blue edges represent *anchor triangles*.

method. In the implementation, open boundaries should be detected with caution. As shown in Fig. 12, when *anchor/preserved triangles* already form 2-manifolds, the planar *candidate triangle* is not bounded. However, for other purposes, eggeneralization, these cases can be treated as bounded in order to fill concave shapes. It should also be noted that there is a special case when the connected coplanar neighbor triangles are bounded by nested anchor boundaries (shown in Fig. 13). In this case, the nested structure must be identified and the hollow space should not be preserved incorrectly (Fig. 13c).

Geometrical constraint II In addition to holes, other common geometrical flaws in the input model are ill-shaped triangles[3]. If these triangles form part of the boundary of a hole, the previous constraint might not be able to fill this hole because the co-planarity detection will be affected by the ill-shaped triangles. Moreover, these tiny triangles produce ill-shaped tetrahedra during CT, which can lead to unexpected consequences in the carving process. Thus, *Geometrical constraint II* is proposed for optimizing or discarding ill-shaped triangles. After an ill-shaped triangle is detected, we use a method similar to that proposed by Botsch and Kobbelt (2001) to flip the longest edge and collapse the triangle at the shortest edge.

Geometrical constraint III This constraint is optional and it is based on the assumption that most of the input geometry has correct orientations. This constraint protects tetrahedra inside a model from being carved. As shown in Fig. 14, we examine the anchor triangles of a

[3]Ill-shaped triangles have sharp and narrow angles, and the length of the normal vector is close to zero.

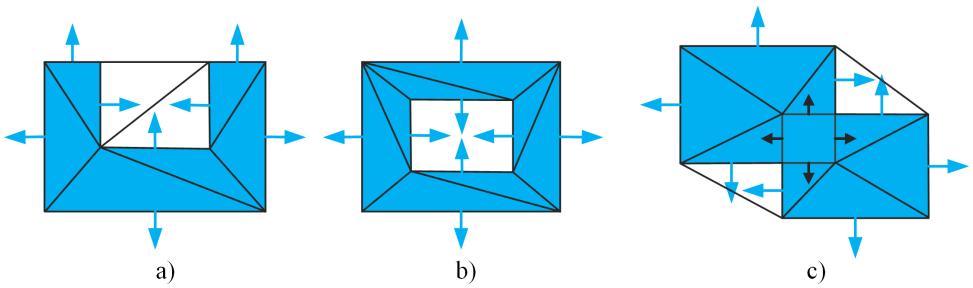


Figure 14: Geometrical constraint based on orientations illustrated in 2D, where the edges indicates anchor triangles and their normal vectors are shown by arrows.

Table 1: Topological (t) and geometrical (g) constraints imposed in various configurations of triangles of a *candidate tetrahedron*.

Configurations		Applied constraints	
Candidate triangle No.	*Anchor/preserved triangle No.*	*Topological*	*Geometrical*
4	0	Ø	II_g
3	1	I_t	II_g III_g
2	2	I_t II_t	II_g III_g
1	3	I_t II_t III_t	I_g II_g III_g
3	0	Ø	II_g
2	1	I_t II_t	II_g III_g
1	2	I_t II_t III_t	I_g II_g III_g
2	0	Ø	II_g
1	1	I_t II_t III_t	I_g II_g III_g
1	0	Ø	I_g II_g

given tetrahedron. If the orientation of one of these triangles points out the tetrahedron, the tetrahedron is inside the *anchor triangles*, and thus it should be preserved. However, if one of the *anchor triangles* points in, the tetrahedron is not necessarily outside the model, as shown by the black arrows in Fig. 14c. The tetrahedron is outside the model only if all its anchor triangle members are oriented inward.

4.2.3 Application of constraints

The previously introduced six constraints are implemented in our algorithm and they are applied to the *candidate tetrahedra* during carving. However, a tetrahedron does not require all six checks in every case because of the specific configuration of its member triangles. For example, if a *candidate tetrahedron* has at least one anchor triangle member on the membrane, then the tetrahedron should be preserved without checking the other constraints because of *topological constraint I* (I_t). By contrast, ill-shaped triangles can appear at any place on the input geometry, so *geometrical constraint II* (II_g) should be applied at the beginning of carving in all cases.

We examine all 10 combinations of the member triangles for a *candidate tetrahedron* and summarize the mandatory constraints that should be imposed for these cases in Table 1, where the first two columns show all the possible configurations of *candidate* and *anchor triangles* for a *candidate tetrahedron*. The third and fourth columns show the constraints that must be applied to the specific configuration, where Ø means that no constraint is needed.

First, it can be observed that there is no need for *topological constraints* in configurations when there are no *anchor/preserved triangles* because these constraints are only enabled for geometry from the input model or that preserved in the previous carving. Second, topological constraints II_t and III_t are applicable when the vertices and edges of the anchor triangle are inside the membrane. For example, the third and the sixth configurations do not need constraint III_t. Third, geometrical constraint I_g should only be applied to configurations when only one *can-*

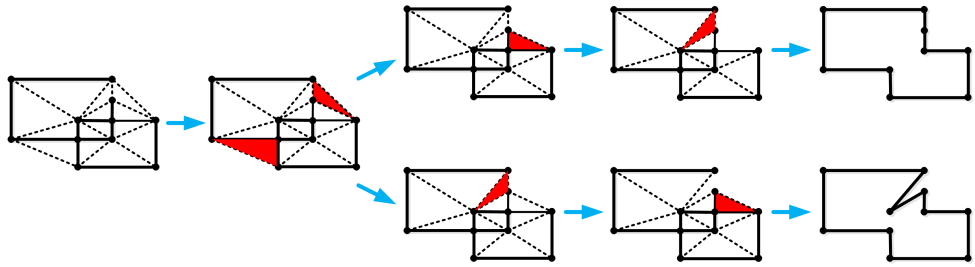


Figure 15: Two carving paths yield different repair results where the last carving process in the upper path is blocked by II_g, whereas the last carving process in the lower path is unblocked and terminated by I_t.

didate triangle is present on the membrane because two or three *candidate triangles* will form a non-coplanar surface[4]. Finally, geometrical constraint III_g is only enabled when *anchor triangles* are present.

All of the proposed constraints are used as pre-assertions. However, in the implementation, we also need to impose post-assertions because not all of the invalid cases can be detected before carving. For example, when a group of tetrahedra need to be carved continuously, as described in Sec. 4.2.1, the validity of the resulting geometry should be checked subsequently. Therefore, we postpone this carving by decreasing the priority of the *candidate tetrahedron*.

4.3 Carving priority

In addition to constraints, the step-by-step carving operations should be executed in an appropriate order. Fig. 15 shows that two different carving paths can result in different repaired results. Therefore, the problem is whether an optimal path can be discovered when carving *candidate tetrahedra*.

This can be considered as an optimization problem and dynamic programming methods are needed to obtain the solution. However, programming requires an objective function, which is difficult to define in the case of repair. It is well known that repair is an ill-posed problem, so our goal is to approximate the most likely shape of the input model and to fix all of the invalid cases. A pragmatic alternative is to adopt the ”best-first” strategy, which deals with the most prominent *candidate tetrahedron* first. This method is also known as the heuristic-based method. Each time, we select the most prominent candidate and check whether it can be carved according to constraints. Therefore, heuristics are required to evaluate the degree of prominence for a *candidate tetrahedron*. Identifying a suitable heuristic function is always a trial-and-error process. There are two different carving strategies. The first is analogous to ” sculpturing” where the carving processes spread around the membrane and shrink the membrane isotropically from the periphery toward the center. The second is analogous to ” region growing” where the membrane is shrunk intensively at some seed positions and carving processes are then propagated to the neighboring area. The first types of heuristics used to reflect the carving strategies described above are referred to as *global heuristics*.

Global heuristics As shown in Fig. 16a, we initialize each tetrahedron with a depth value (*Depth*). The boundary tetrahedra start from depth 1. During carving, the depth of the carved tetrahedron is propagated to its neighbors (Fig. 16a–b), as follows. If the depth value of a neighbor is less than or equal to the depth of the carved tetrahedron value, then its depth is set as the ”carved depth” incremented by 1; otherwise, its depth value remains unchanged (Fig. 16c–d).

[4]In the case where a tetrahedron becomes degenerate, planarity can still be found for these configurations. Thus, we apply geometrical constraint I_g to these tetrahedra in the implementation

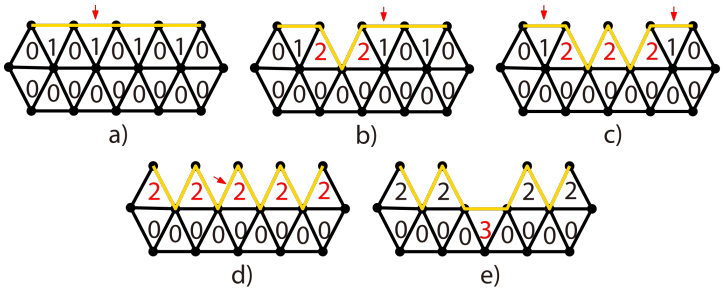


Figure 16: Depth heuristics for tetrahedra during carving (in 2D). The yellow line segments represent parts of the approximated membrane.

By sorting the *candidate tetrahedra* in ascending order of *Depth* (or in descending order of the negative value of *Depth*, i.e., the *NDepth*), we can distribute carving around the whole model. By contrast, carving in descending order of *Depth* (or in ascending order of *NDepth*) will focus carving on a certain location.

We also consider the distance from a candidate tetrahedron to the input model (*Dist*). During repair, if a distant candidate is carved earlier, the carving processes will also be distributed evenly, and vice versa. The distance is calculated from the center of the *candidate tetrahedron* to the nearest *anchor triangle*, which is accelerated by using the KD-tree (CGAL, 2013).

Finally, buildings are rich in parallel and perpendicular structures, and the main orientations can be extracted. The minimum deviation from the main orientations of a *candidate tetrahedron* is also treated as a global heuristic (*Dev*). *Dev* implies the conformity of the tetrahedron to a regular shape. If the deviation is greater, the tetrahedra are less likely to conform to the input building model. Carvings will then be guided to corners around the model, and vice versa.

Local heuristics In addition to *global heuristics*, we found that the local characteristics of a *candidate tetrahedron* can greatly influence the carving results. In this study, we consider the following five *local heuristics* for *candidate tetrahedra*.

1. The number of *candidate triangles* in the *candidate tetrahedron*, which we refer to as the degrees of freedom (DoF).
2. The total area of the member triangles on the *candidate tetrahedron*
3. The flatness of the *candidate tetrahedron* (defined by the ratio between the area of a candidate triangle and its distance to the opposite vertex).
4. The volume of the *candidate tetrahedron*.
5. The mean curvature of the *candidate triangle* on the *candidate tetrahedron*

As described by Zhao et al. (2013)), DoF indicates the degree of knowledge about the *candidate tetrahedron*. When the DoF is larger, more is revealed about the shape of the tetrahedron, as shown in Fig. 17. Thus, fewer constraints are required (Table 1), and thus there should be a higher priority for carving.

Local heuristics II to IV depict various geometrical characteristics of the *candidate tetrahedron*, where a larger value indicates greater size or a flatter shape. In general, these types of tetrahedra are more likely to be part of the auxiliary structure generated by CT.

Finally, the mean curvature of the *candidate triangle* in a *candidate tetrahedron* describes the local convexity. The initial membrane is convex so the convex part should be carved early.

We compared the performance of the global and local heuristics as well as their combinations with a few representative erroneous models (four typical examples are shown in Fig. 18). These

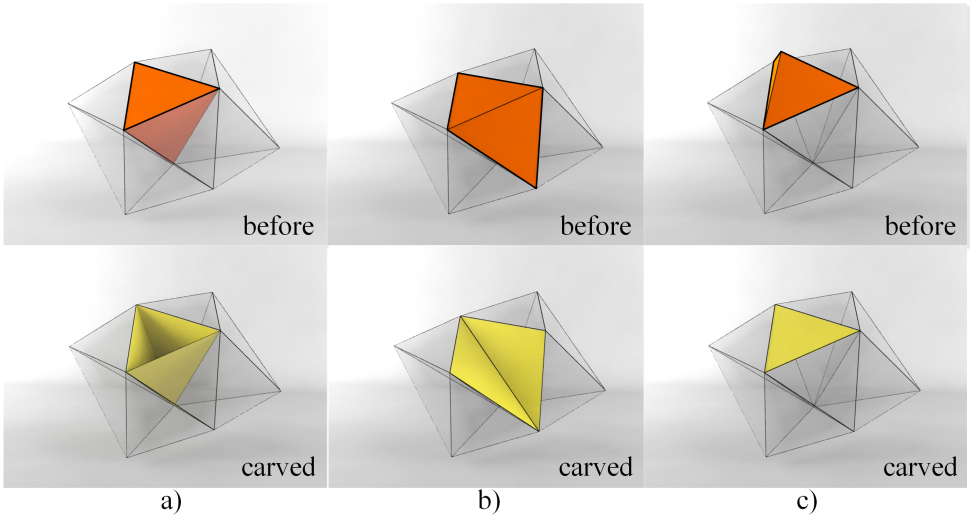


Figure 17: The possible DoF of a *candidate tetrahedron* and the carved results (a) DoF = 1, where only one *candidate triangle* is on the boundary; b) DoF = 2, where two *candidate triangles* are on the boundary; c) DoF = 3, where three *candidate triangles* are on the boundary.)

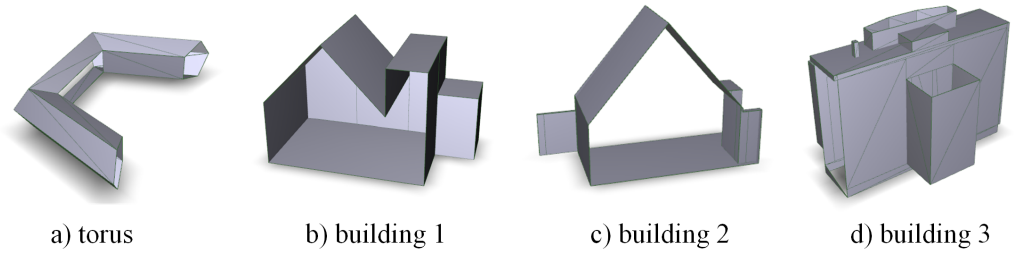


Figure 18: Four typical erroneous models used for selecting heuristics: a) synthetic erroneous model, b–c) simple erroneous building models, and d) a complex erroneous building model.

models are either created by hand or selected from existing datasets. Each of them contains various topological and geometrical errors, thus can be employed as the "touchstone". We inspected all the repaired models visually by a modeling software, ie3ds Max, and the results are shown in Table 2 and Table 3. The global heuristics were applied in both ascending and descending order, and the local heuristics were all applied in descending order.

Table 2 shows that the repair algorithm could not succeed with all four input geometries without using the heuristics or with only the global or local heuristics. In Table 3, all of the *candidate tetrahedra* are sorted first by the global heuristics and then by the local heuristics. We also tried to swap the order of the global and local heuristics, and found that the results were almost identical. The results showed that *the heuristic combination of NDepth in ascending order (or Depth in descending order) and DoF performed better than the other combinations* (some repair results are shown in Fig. 19 for comparison). This suggests that carving the corner tetrahedra (higher DoF) intensively at a specified area (higher Depth) is a better choice. Other combinations, egthe *NDepth* in descending order+*The mean curvature*, and the *Dev* in descending order+*DoF*, were also successful in some cases, but they could not pass all of the test models. Thus, the combination of the *Depth* and DoF heuristics was selected for our experiments.

The pseudo code of the proposed repair method is shown by Algorithm 1.

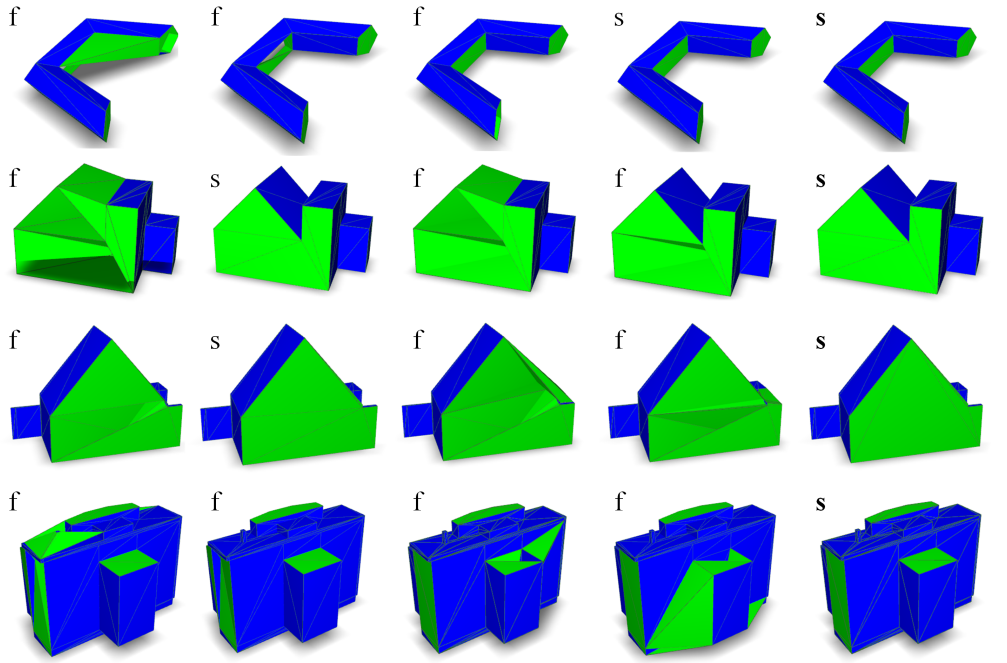
Table 2: Results based on purely global heuristics and local heuristics, where f denotes failure and s represents success.

Models	*Global heuristics descending*			*Global heuristics ascending*		
	NDepth	*Dev*	*Dist*	*NDepth*	*Dev*	*Dist*
Torus	f	s	f	s	f	s
building 1	f	s	f	f	f	s
building 2	f	f	f	f	f	f
building 3	f	s	f	f	f	f

Models	*Local heuristics*					***No heuristic***
	DoF	*Area*	*Flat*	*Volume*	*Curvature*	
Torus	f	s	s	f	f	**f**
building 1	s	f	f	f	f	**f**
building 2	s	f	f	f	f	**f**
building 3	f	f	f	s	s	**f**

Table 3: Comparisons of different combinations of global and local heuristics, where f denotes failure and s represents success.

descending	*NDepth*					*Dev*					*Dist*				
Models	*I*	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
Torus	f	s	f	f	s	f	s	f	f	s	f	s	f	f	s
building 1	s	f	f	f	s	s	f	f	f	s	f	f	f	f	s
building 2	f	f	f	f	f	s	f	f	f	f	f	f	f	f	f
building 3	s	f	s	s	s	s	f	s	s	s	s	f	s	s	s
ascending	***NDepth***					*Dev*					*Dist*				
Models	***I***	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*	*I*	*II*	*III*	*IV*	*V*
Torus	**s**	s	f	f	s	f	f	f	f	s	s	s	f	f	s
building 1	**s**	f	f	f	f	f	f	f	f	f	s	f	f	f	f
building 2	**s**	f	f	f	f	f	f	f	f	f	f	f	f	f	f
building 3	**s**	f	s	s	f	f	f	s	s	f	f	f	s	s	f



a) No heuristics b) DoF c) CarvingDetph (ascending) d) CarvingDetph (descending) **e) CarvingDetph (descending)+DoF**

Figure 19: Comparison of the repair results obtained using: a) no heuristic, b) only DoF in descending order, c) only *Depth* in ascending order, d) only *Depth* in descending order, and e) *Depth* in descending order combined with DoF in descending order.

Algorithm 1 HSW Repair of a building model M.

```
Intersection free M̃ ← Decomposition(M)
Tetrahedra T ← CT(M̃)
while Existed(T_candidate) do
    Sort(All T_candidate, descending) by Global heuristics Depth
    Sort(All T_candidate, descending) by Local heuristics DoF
    Tetrahedron t ← Pop(All T_candidate)
    if ViolateTopologicalConstraints(t) —— ViolateGeometricConstraints(t) then
        Preserve or Postpone t
    else
        Carve or Postpone t
    end if
    Update heuristics for All T_candidate
end while
Repaired M̂ ← ExtractBoundary(All T_candidate)
Restore orientations and semantics of M
return M̂
```

5 Results

We conducted several experiments to evaluate the proposed methods. First, we assessed the repair of geometrical defects by using building models created with 3D modeling software (egAutodesk 3ds Max and Maya). This type of model has been used widely in many 3D city projects throughout the world (Zhao et al., 2012). We then repaired the real 3D city datasets based on CityGML. As shown by Biljecki et al. (2016), the standardized models are often enriched in terms of their semantics, but they are not flawless. When repairing these models, we had to consider the semantics as well as the erroneous geometry. Finally, in Sec. 5.2, we present comparisons of the performance of the proposed repair method with those of the state-of-the-art mesh repair methods, i.e., MeshFix (Attene, 2010), PMP (Botsch et al., 2007), PolyMender (Ju, 2004), and ReMESH (Attene and Falcidieno, 2006).

In the implementation, we employed several third-party libraries. The Delaunay tessellation of polygons was based on the Triangle package (Shewchuk, 2002). The intersection detection method was built based on a fast triangle intersection detector (Möller, 1997). CT was conducted using Tetgen (Si, 2015).

5.1 Repair of building datasets

5.1.1 Repair of geometrical defects

We selected three invalid geometrical building models (shown in Fig. 20) from real 3D city datasets, where they contained the typical errors found in many building models, such as holes, self-intersections, and non-manifold cases (Fig. 21a). We applied the proposed method and automatically obtained the repaired results (Fig. 20). The detailed repaired results are demonstrated in (Fig. 21b), in which all of the errors were corrected and the results were closed 2-manifolds. Fig. 22 shows the carving sequence during the repair process. The membrane was shrunk gradually under guidance by the heuristics, and the topological and geometrical constraints ensured the correctness of the end results.

5.1.2 Repair of CityGML datasets

We repaired two real datasets comprising 4970 buildings (Fig. 23). The two real CityGML datasets were provided by the Rotterdam municipality and both were generated using photogrammetric methods. To quantitatively evaluate the validity of a CityGML dataset, iein both the topological and geometrical perspectives, the standard conforming Val3Dity tool (Ledoux,

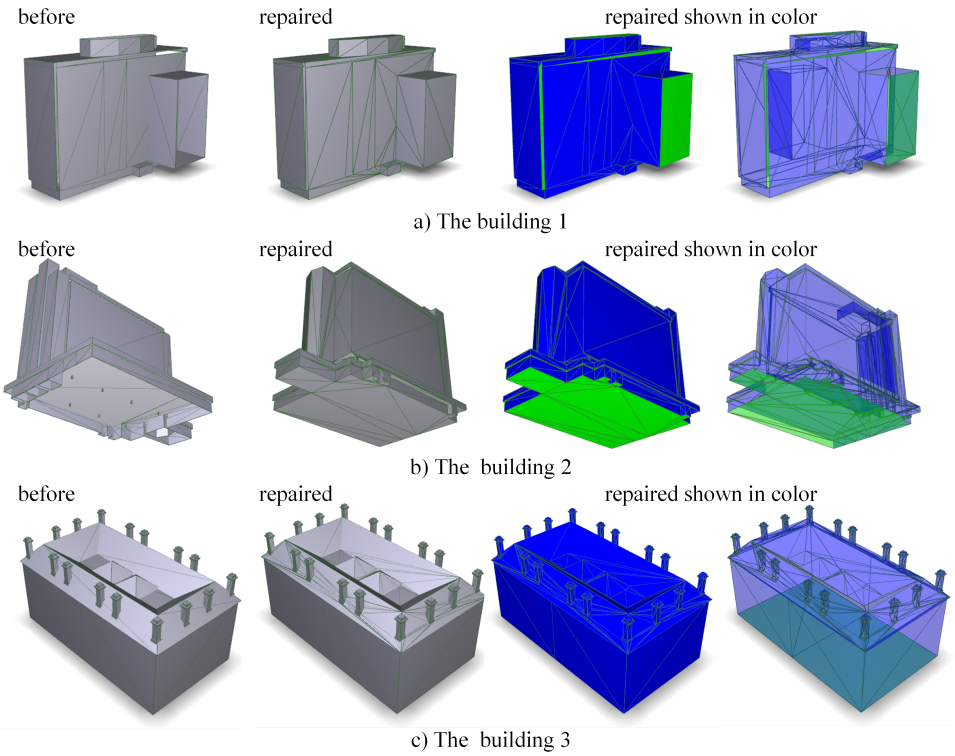


Figure 20: Selected erroneous building models and their repaired results, where blue indicates *anchor triangles* and green indicates newly generated *preserved triangles* during repair.

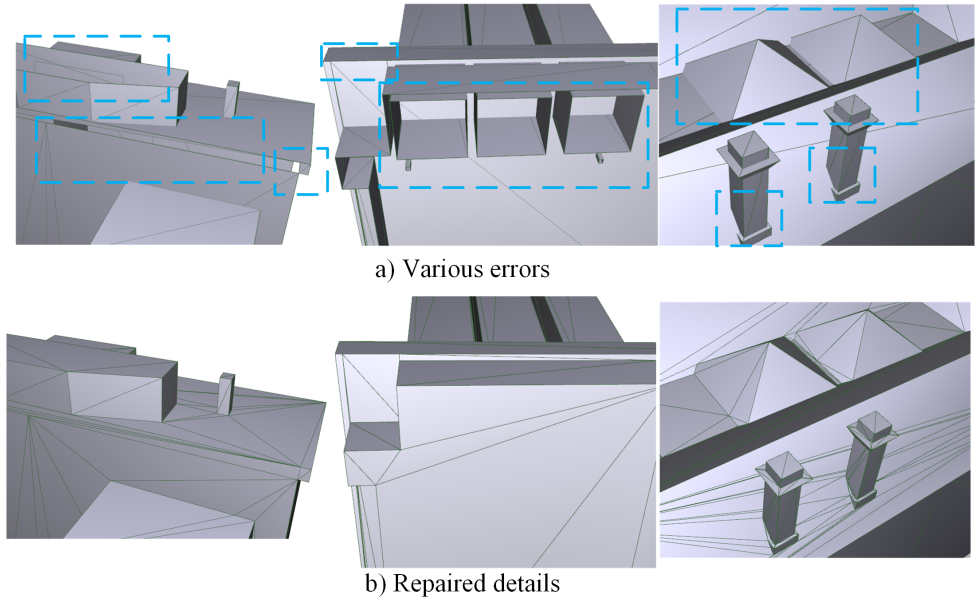


Figure 21: Details of the errors in the input geometrical models and the repaired results.

Table 4: Validation of repaired MultiSurfaces for two CityGML datasets. Note the repair process introduced extra MultiSurfaces. But both the ratio and the absolute number of invalid MultiSurfaces were drastically decreased after repairing.

	Original MultiSurfaces (invalid/total)	Repaired MultiSurfaces (invalid/total)
Witte Dorp	124/696 = 17.8%	2/7435 = 0.0002%
Nesselande	1424/12796 = 11.1%	4/163076 = 0.00002%

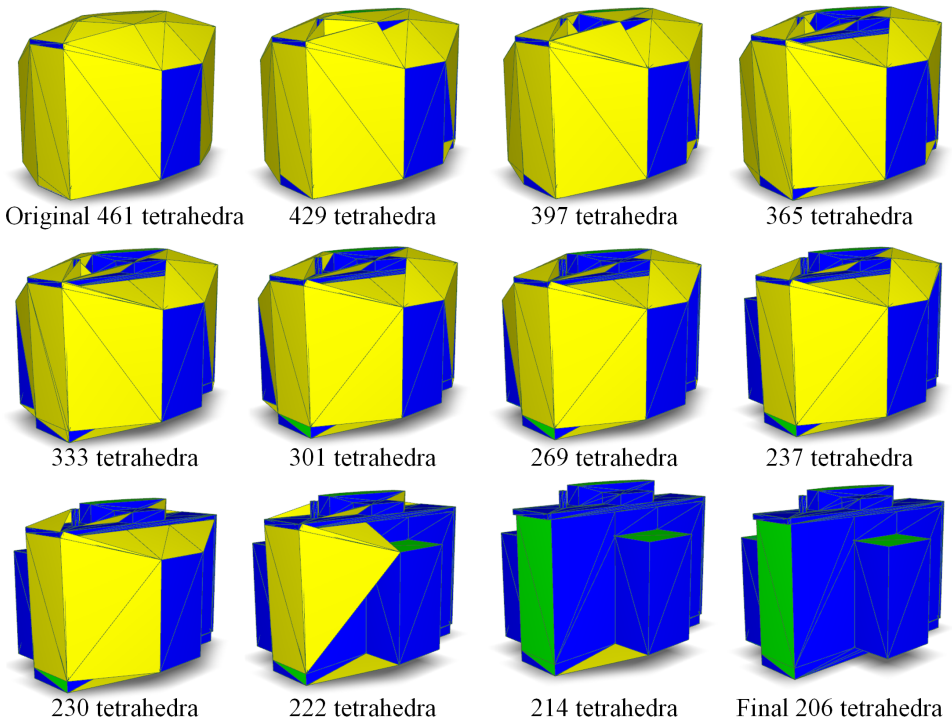


Figure 22: Illustration of the incremental shrink repair process where blue indicates *anchor triangles*, green indicates newly generated *preserved triangles* obtained during repairing, and yellow indicates *candidate triangles*.

a) WITTE DORP dataset b) NESSELANDE dataset

Figure 23: Two real LOD2 CityGML datasets adopted in the experiments.

Table 5: Validation of repaired solids for two CityGML datasets. The less the ratio, the better the repaired result. Note that there are some models cannot be processed by a certain algorithm, egindexing error of vertices or cases that cannot be handled .

	Origin Solids (invalid/total)	Repaired Solids (invalid/processed total)		
		HSW (Ours)	$PMP_{HoleFill}$	$PMP_{HoleFill}$ Decomposed
Witte Dorp	233/233 = 100%	5/232 = 2.15%	231/232 = 99.57%	72/231 = 31%
Nesselande	4273/4740 = 90.14%	200/4740 = 4.2%	3495/4733 = 73.84%	1338/4683 = 28.57%

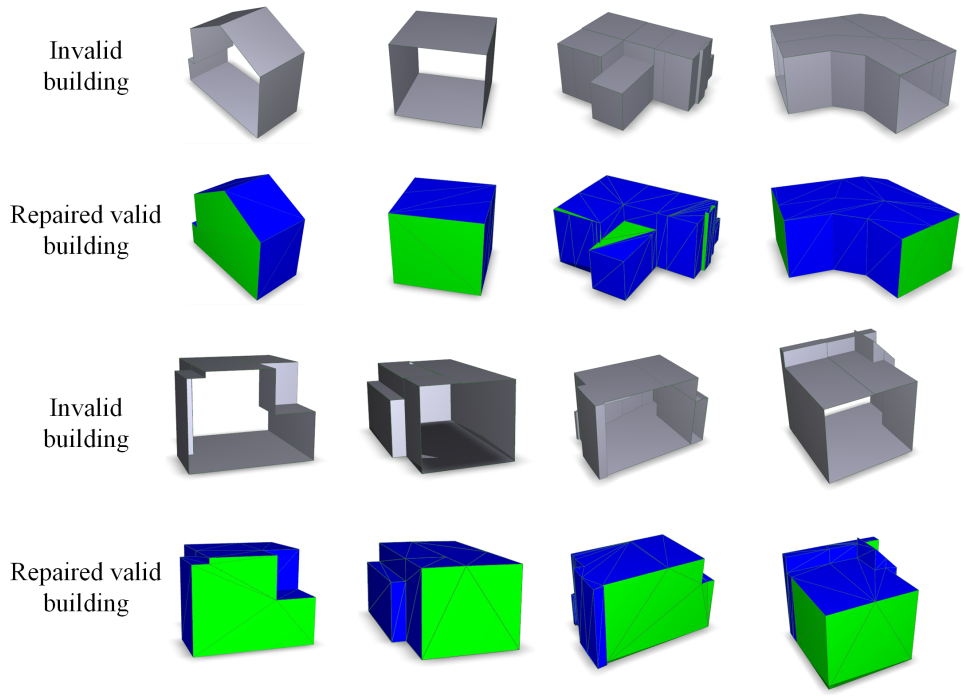


Figure 24: Selected repair results for two CityGML datasets, where blue indicates anchor triangles and green indicates newly generated preserved triangles during repair.

2013) is employed in this paper[5].

Both datasets were built based on MultiSurfaces. The invalid MultiSurfaces before and after repair are shown in Table 4. The aim of our repair method is to reconstruct solid-based representations, so we also compared the validation reports for each of the buildings and Val3Dity indicated that more than 97% of the buildings in Witte Dorp ($1 - 2.15\%$) and more than 95% of the buildings in Nesselande ($1 - 4.2\%$) were repaired (Table 5).

Fig. 24 compares the original buildings and those after repair. Holes were healed and self-intersections were removed. Each repaired building was a solid model and a closed 2-manifold. Several unsuccessful cases are also shown in Fig. 25. These failures occurred because too many faces were missing from these models, and thus the available information was insufficient for repair using automatic methods. We also visually inspected all the results and found that the method introduced artifacts in some topological corrected repaired models (Fig. 26). The introduction of tetrahedron surfaces was due to the formation of a complex edge when two neighboring solid shapes touched on the roof. In this case, the edge should be split in advance (Guéziec et al., 2001).

[5]https://github.com/tudelft3d/val3dity

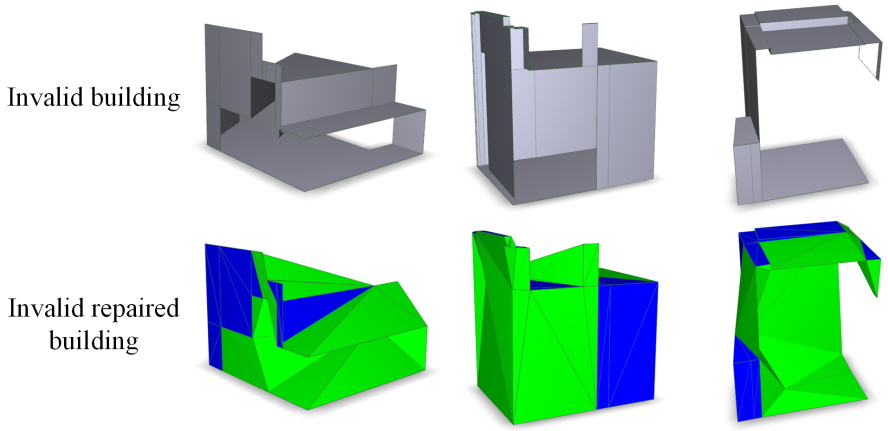


Figure 25: Several failed results after repair, where blue indicates anchor triangles and green indicates newly generated preserved triangles during repair.

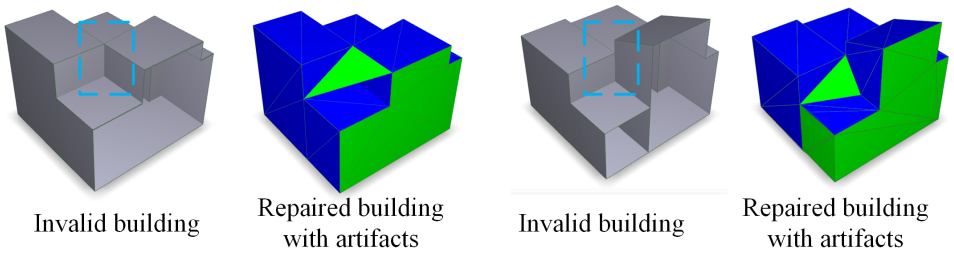


Figure 26: Introduction of artifacts due to the complex edge shown by dashed boxes, where blue indicates anchor triangles and green indicates newly generated preserved triangles during repair.

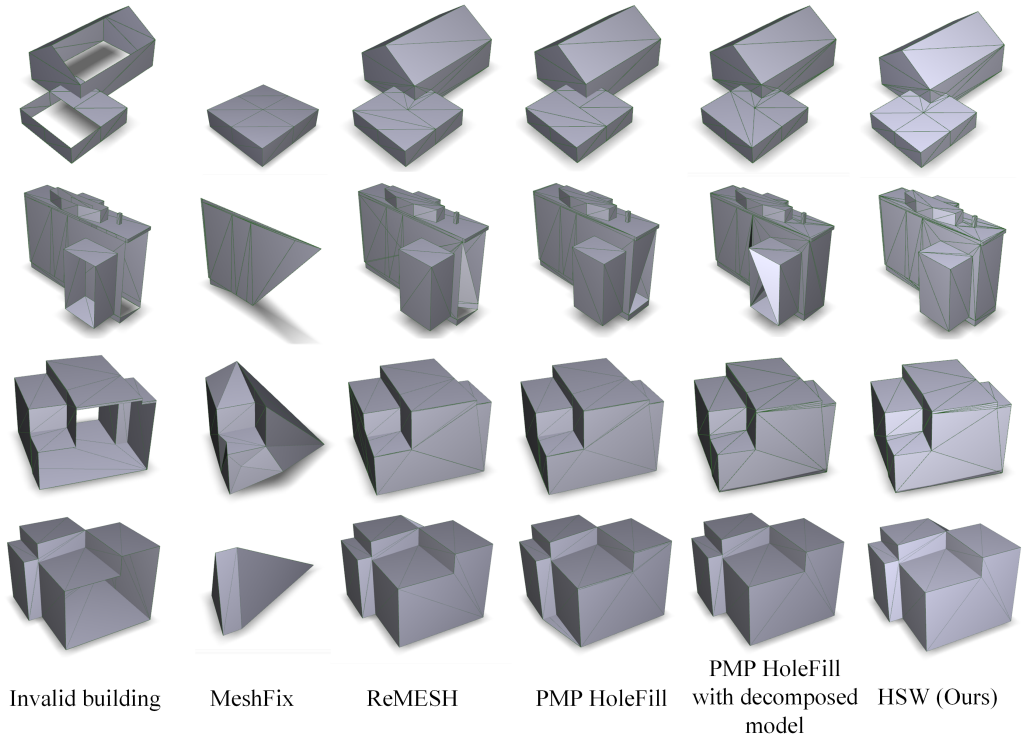


Figure 27: Illustration of the repaired results obtained by the methods compared.

5.2 Comparison

We compared our method with openly available automatic mesh repair methods, i.e., MeshFix (Attene, 2010), PMP (Botsch et al., 2007), PolyMender (Ju, 2004), and ReMESH (Attene and Falcidieno, 2006). We applied these methods with the default parameters to the models used in our previous experiments. The repaired results are shown in Fig. 27. The hole filling method in PMP ($PMP_{HoleFill}$) performed the best among these methods, but it is still inferior to our method.

We also compared PMP with our method using two CityGML datasets (Sec. 5.1.2). The PMP hole filling method lacks the capacity to resolve intersecting geometry, so we first decomposed the input model using our proposed method and then employed PMP to repair the intersection-free model. The Val3Dity reports for the repaired results are shown in Table 5, which demonstrates that the proposed method achieved state-of-the-art performance at repairing these datasets.

5.3 Discussion

Our experiments showed that the proposed method could fix most of the errors found in 3D city models. The method is robust because it employs a volumetric representation based on CT. The method is also accurate because the input geometry is preserved exactly in CT, and thus no discretization artifacts are introduced. The CT is conducted efficiently by using Tetgen (Si, 2015). According to Si (2015), the constraint tetrahedralization of a model of m vertices and r reflex edges results into $O(m + r^2)$ tetrahedra. Therefore, the complexity of our HSW method is $O((m + r^2)^2)$, because every tetrahedron is examined and sorted in each carving step. In comparison, the PMP method works on the mesh combinatorial structure directly, thus would have a much lower complexity. However, we considered the repair task to be insensitive to the computational time, since it can be conducted offline. The two real CityGML datasets adopted in the experiment can be repaired within a hour. This method can be further optimized and

accelerated by conducting a few batches of repair in parallel.

According to the experiments, we found that the proposed *topological constraints* are capable of maintaining the carving process as "2-manifold-bounded," which means that the repair process will always produce a solid model. The *geometrical constraints* were also found to perform efficiently at fixing the erroneous geometry of a building model. However, more *geometrical constraints* can be proposed for specific error types, such as non-planar holes. Although the proposed heuristics was derived by trial-and-error with a limited number of erroneous models, it generalizes well to large datasets.

A problem is that CT might introduce degenerate tetrahedra and this could affect our constraints. For example, $ConstraintI_8$ should be reinforced for configurations with more than one candidate triangle (as shown in Table 1) because the member triangles of a tetrahedron can be coplanar in these cases.

We did not expect to obtain accurate repair results for CityGML models with overhanging because overhangs are represented by surfaces that comprise topologically dangling faces. Auxiliary surfaces will remain and keep the dangling face as a part of the solid. Thus, our shrink-wrapping repair method should not be applied directly to these models, but instead it should only be used to fix models or building parts that are assumed to be solids (Zhao et al., 2014).

Given that the repair task is an ill-posed problem, our method can still be improved. We selected the heuristics using pragmatic methods and they can be extended. Backtracking of the carving process could be introduced to facilitate finding the best carving route by evaluating the cost of carving in an optimization process.

If complex edges or vertices are present, our method will introduce unwanted but "valid artifacts", which could be addressed by first splitting these geometries.

Finally, an efficient and fast data structure for tetrahedra should be formulated to allow rapid implementation of the proposed constraints. In addition, robust arithmetic kernels could also be used in the decomposition stage for extreme intersection cases (Attene, 2014).

6 Conclusion

In this study, we proposed a shrink-wrapping repair algorithm (HSW) for reconstructing valid solid-based LOD2 building models from invalid building models. This method employs CT and this makes it as robust as volume-based repair methods. Owing to the proposed topological constraints, any arbitrary input geometry can be accepted and the output is consistently a closed 2-manifold solid. The shape and semantics of the input building model are also preserved well, and few artifacts are introduced. We demonstrated the effectiveness of our method based on various experiments using 3D building models and CityGML datasets. The choices of different heuristics and their combinations were also investigated in detail. This method can be extended to the repair of regular-shaped mesh models where solids are the desired outputs. By introducing the proper constraints, the proposed method can be further adopted for generalization purpose. In future research, we will focus on implementing the overall repair framework for the CityGML model by treating building components separately according to their repair goals.

Acknowledgement

We would like to thanks Hang Si for his help with the Tetgen package. And we ppreciate the Rotterdam municipality for providingthe datasets. This work is supported by the National Key Research andDevelopment Program of China (No. 2017YFA0603104, No.2018YFB0105103), the National Natural Science Foundation of China(No. U1764261, No. 41801335, No. 41871370),

the European Research Council (ERC) under the European Union's Horizon 2020 research andinnovation programme (grant agreement No 677312 UMnD), the Natural Science Foundation of Shanghai (No. kz170020173571) and the Fundamental Research Funds for the Central Universities (No.22120180095).

References

Attene M (2010). A lightweight approach to repairing digitized polygon meshes. *The visual computer*, 26(11):1393–1406.

Attene M (2014). Direct repair of self-intersecting meshes. *Graphical Models*, 76(6):658–668. ISSN 1524-0703. doi:https://doi.org/10.1016/j.gmod.2014.09.002.

Attene M, Campen M, and Kobbelt L (2013). Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)*, 45(2):15.

Attene M and Falcidieno B (2006). Remesh: An interactive environment to edit and repair triangle meshes. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on*, pages 41–41. IEEE.

Biljecki F, Heuvelink GB, Ledoux H, and Stoter J (2015). Propagation of positional error in 3D GIS: estimation of the solar irradiation of building roofs. *International Journal of Geographical Information Science*, 29(12):2269–2294. doi:10.1080/13658816.2015.1073292.

Biljecki F, Ledoux H, Du X, Stoter J, Soon KH, and Khoo V (2016). The most common geometric and semantic errors in CityGML datasets. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4(2):13–22.

Bischoff S and Kobbelt L (2005). Structure preserving CAD model repair. *Computer Graphics Forum*, 24(3):527–536.

Bischoff S, Pavic D, and Kobbelt L (2005). Automatic restoration of polygon models. *ACM Transactions on Graphics (TOG)*, 24(4):1332–1352.

Boeters R (2013). Automatic enhancement of CityGML LoD2 models with interiors and its usability for net internal area determination. Delft University of Technology.

Bogdahn J and Coors V (2010). Towards an automated healing of 3D urban models. In Kolbe T, König G, and Claus N, editors, *5th International Conference on 3D Geoinformation*, 5th International Conference on 3D Geoinformation, pages 13–17. Shaker Verlag, Aachen, Germany.

Botsch M and Kobbelt L (2001). A robust procedure to eliminate degenerate faces from triangle meshes. In *Proc. of Vision, Modeling, and Visualization*, volume 1 of *Proc. of Vision, Modeling, and Visualization*, pages 283–289. Citeseer.

Botsch M, Pauly M, Kobbelt L, Alliez P, Lévy B, Bischoff S, and Röossl C (2007). Geometric modeling based on polygonal meshes. SIGGRAPH 2007 course.

Campen M, Attene M, and Kobbelt L (2012). A practical guide to polygon mesh repairing. In *Eurographics 2012*, Eurographics 2012, pages t4–undefined. The Eurographics Association, May 13-18 Cagliari, Italy. Eurographics 2012-Tutorials.

Campen M and Kobbelt L (2010). Exact and robust (self-) intersections for polygonal meshes. *Computer Graphics Forum*, 29:397–406. 2.

CGAL (2013). Computational geometry algorithms library, http://www.cgal.org.

Donkers S, Ledoux H, Zhao J, and Stoter J (2016). Automatic conversion of IFC datasets to geometrically and semantically correct CityGML lod3 buildings. *Transactions in GIS*, 20(4):547–569.

Granados M, Hachenberger P, Hert S, Kettner L, Mehlhorn K, and Seel M (2003). Boolean operations on 3D selective Nef complexes: Data structure, algorithms, and implementation. In Di Battista G and Zwick U, editors, *Algorithms–ESA 2003: 11th Annual European Symposium*, volume 2832, page 174–186. Springer, Budapest, Hugary, September 2003.

Gröger G, Kolbe TH, Nagel C, and Häfele KH (2012). *OpenGIS City Geography Markup Language (CityGML) Encoding Standard. Version 2.0.0*. OGC 12-019. Open Geospatial Consortium.

Guercke R, Gotzelmann T, Brenner C, and Sester M (2011). Aggregation of LoD 1 building models as an optimization problem. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(2):209–222.

Guéziec A, Taubin G, Lazarus F, and Hom B (2001). Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):136–151.

Hagbi N and El-Sana J (2010). Carving for topology simplification of polygonal meshes. *Computer-Aided Design*, 42(1):67–75.

Herring J (2005). *ISO 19107: 2005: Geographic information-Spatial schema*.

Hétroy F, Rey S, AndúJar C, Brunet P, and Vinacua A (2011). Mesh repair with user-friendly topology control. *Computer-Aided Design*, 43(1):101–113.

Ju T (2004). Robust repair of polygonal models. *ACM Transactions on Graphics (TOG)*, 23(3):888–895.

Karki S, Thompson R, and McDougall K (2010). Data validation in 3D cadastre. In Neutens T and Maeyer P, editors, *Developments in 3D Geo-Information Sciences*, Developments in 3D Geo-Information Sciences, pages 92–122. Springer Berlin Heidelberg.

Kazar BM, Kothuri R, Oosterom P, and Ravada S (2008). On valid and invalid three-dimensional geometries. In *Advances in 3D Geoinformation Systems*, Advances in 3D Geoinformation Systems, pages 19–46. Springer Berlin Heidelberg.

Kobbelt LP, Vorsatz J, and Labsik U (1999). A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18(3):119–130. 3.

Kolbe TH (2008). Representing and exchanging 3D city models with CityGML. In Zlatanova S and Lee J, editors, *3D Geo-Information Sciences*, 3D Geo-Information Sciences, pages 15–31. Springer Berlin Heidelberg.

Kolbe TH, Gröger G, and Plümer L (2008). CityGML-3D city models and their potential for emergency response. pages 257–274. Taylor & Francis, London.

Koo BK, Choi YK, Chu CW, Kim JC, and Choi BT (2005). Shrink-wrapped boundary face algorithm for mesh reconstruction from unorganized points. *ETRI journal*, 27(2):235–238.

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. doi:10.1111/mice.12043.

Liepa P (2003). Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 200–205. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. ISBN 1-58113-687-0.

Mezentsev AA and Woehler T (1999). Methods and algorithms of automated CAD repair for incremental surface meshing. In *IMR*, pages 299–309.

Möller T (1997). A fast triangle-triangle intersection test. *Journal of graphics tools*, 2(2):25–30.

Mulder DT (2015). *Automatic repair of geometrically invalid 3D City Building models using a voxel-based repair method*. Ph.D. thesis.

Nooruddin FS and Turk G (2003). Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205.

Shen C, O'Brien JF, and Shewchuk JR (2004). Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics (TOG)*, 23(3):896–904.

Shewchuk JR (2002). Delaunay refinement algorithms for triangular mesh generation. *Computational geometry*, 22(1):21–74.

Si H (2015). Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw*, 41(2):11:1–11:36. doi:10.1145/2629697.

Steuer H, Machl T, Sindram M, Liebel L, and Kolbe TH (2015). *Voluminator—Approximating the Volume of 3D Buildings to Overcome Topological Errors*, pages 343–362. Springer International Publishing. ISBN 978-3-319-16787-9. doi:10.1007/978-3-319-16787-9_20.

Wagner D, Wewetzer M, Bogdahn J, Alam N, Pries M, and Coors V (2013). Geometric-semantical consistency validation of CityGML models. In Pouliot J, Daniel S, Hubert F, and Zamyadi A, editors, *Progress and New Trends in 3D Geoinformation Sciences*, pages 171–192. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-29793-9. doi: 10.1007/978-3-642-29793-9_10.

Zhao J, Ledoux H, and Stoter J (2013). Automatic repair of CityGML LoD2 buildings using shrink-wrapping. In *Proceedings of the ISPRS 8th 3D GeoInfo conference & WG II/2 workshop*, volume II-2 W, pages 309–317. Istanbul.

Zhao J, Stoter J, and Ledoux H (2014). A framework for the automatic geometric repair of CityGML models. In Buchroithner M, Prechtel N, and Burghardt D, editors, *ICA conference 2013*, pages 187–202. Springer Berlin Heidelberg, Dresden.

Zhao J, Stoter J, Ledoux H, and Zhu Q (2012). Repair and generalization of hand-made 3D building models. In *Proceedings of the 15th workshop of the ICA commission on generalisation and multiple representation jointly organised with EuroSDR commission*, page 10.

Zhou Q, Grinspun E, Zorin D, and Jacobson A (2016). Mesh arrangements for solid geometry. *ACM Trans. Graph.*, 35(4):39:1–39:15. doi:10.1145/2897824.2925901.