Computers & Graphics 132 (2025) 104368

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on CAD/Graphics 2025

Explicit topology and connectivity constraints for 3D model repair

Jiantao Song [a], Wensong Wang [a], Rui Xu [b], Wenlong Meng [c], Shuangmin Chen [d,e,*], Shiqing Xin [a], Taku Komura [b], Changhe Tu [a], Wenping Wang [f]

[a] Shandong University, China
[b] The University of Hong Kong, China
[c] Harbin Institute of Technology, China
[d] Qingdao University of Science and Technology, China
[e] Shandong Key Laboratory of Deep Sea Equipment Intelligent Networkingy, China
[f] Texas A&M University, United States of America

ARTICLE INFO

Keywords:
Mesh repair
Integer programming
Topology
Connectivity

ABSTRACT

Designers often prioritize structural design over machinability, leading to various defects such as open boundaries, self-intersections, non-manifoldness, topological errors, and disconnected patches. This makes mesh repair a fundamental task in digital geometry processing. Despite significant progress in recent decades, topological and connectivity constraints continue to pose substantial challenges. In this paper, we propose a general integer programming framework to address the problem of mesh repair. We start by dividing the entire surface into topologically equivalent disk surface patches. Our formulation not only considers manifoldness and watertightness but also addresses connectivity constraints and user-specified topological constraints. First, topological constraints are defined based on Euler's characteristic formula. Second, we observe that in a connected graph with n nodes, a capacity configuration with $n-1$ '1's and one '$n-1$' leads to a non-vanishing flow plan, and thus allowing us to transform the connectivity requirements into a max-flow problem. In implementation, our method begins by identifying topological disk equivalent surface patches and assigning each surface patch a binary labeling variable to indicate whether it should be retained. To ensure the outcome closely resembles the original model, we also introduce the alpha-wrapping technique to extract the approximate outer layer, which helps define the adherence of surface patches. As the Fig. 1 demonstrates, our method can produce repaired models with genera of 0, 1, and 3, while maintaining a single component. We tested our method on a large set of defective models, demonstrating its advantages over state-of-the-art methods.

1. Introduction

Conceptual designs provide an initial, tangible visualization of the final product, allowing designers and stakeholders to assess its feasibility and make necessary adjustments before moving on to more detailed development phases. However, the transition from these designs to digital models can introduce various defects, such as open boundaries, self-intersections, non-manifold edges, topological issues, and disconnected patches. These defects can significantly complicate downstream tasks such as manufacturing, simulation, and rendering. Consequently, repairing digital models becomes a fundamental task in digital geometry processing.

Polygonal meshes, as the most prevalent and flexible representation, are extensively utilized across various industrial fields. The requirements for a valid polygonal model vary depending on the specific application field, but common requirements include manifoldness, watertightness, and connectedness. Additionally, there are scenarios where users must specify the genus of the target model. A significant body of literature focuses on the task of mesh repair; see the survey in [1,2]. For instance, Lazar et al. [3] studied how to control the genus and demonstrated its uses in reconstruction from cross-section slices and iso-surfacing an intensity volume. However, their algorithm requires the input data to be organized layer by layer, which limits its practical use. Moreover, Nan and Wonka [4] proposed a binary linear programming formulation to find an appropriate combination of fitting planes to ensure the output polygonal surface is manifold and watertight. In fields such as simulation and CAD, it is sometimes required to obtain high-quality hexahedral meshes [5,6] and quad mesh [7–9]. In summary, while the issues of manifoldness and watertightness have

* Corresponding author.

https://doi.org/10.1016/j.cag.2025.104368
Received 27 June 2025; Received in revised form 28 July 2025; Accepted 30 July 2025
Available online 8 August 2025

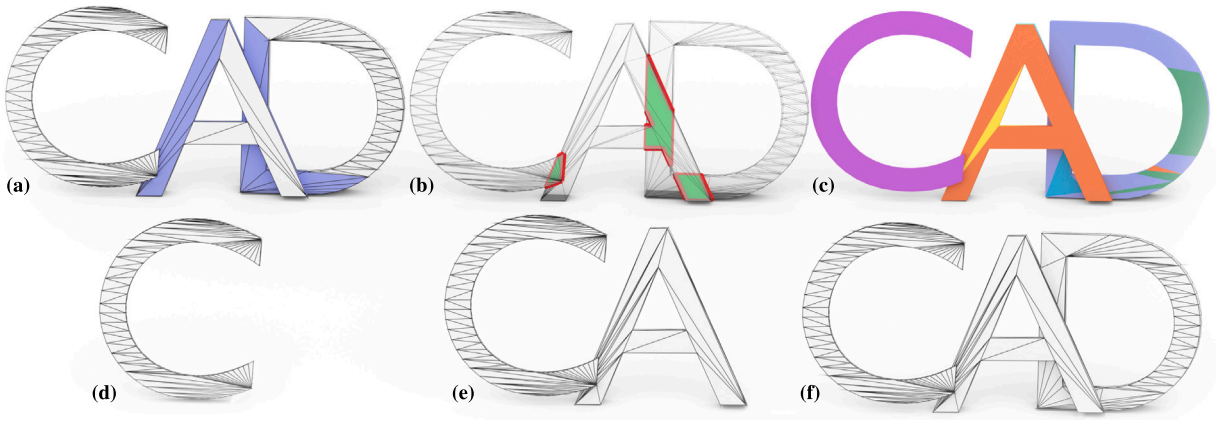

Fig. 1. In this paper, we propose a mixed integer programming framework to address the problem of mesh repair. Our formulation ensures manifoldness and watertightness and accounts for connectivity constraints and user-specified topological constraints. (a) illustrates the input model, which exhibits issues such as self-intersection. (b) presents the result after preprocessing, where some non-manifold structures remain. (c) depicts the patch segmentation derived from our method. And (d)–(f) respectively demonstrate the manifold, watertight, and simply connected outcomes produced by our method under the specified genus constraints.

been extensively discussed and addressed in past research, explicitly specifying topological and connectivity constraints continues to pose substantial challenges (see Fig. 1).

In this paper, we adopt the approach of integrating various requirements into a unified integer programming framework. In addition to the fundamental requirements of manifoldness and watertightness, we emphasize the explicit representation of topological and connectivity requirements. Our method begins by dividing the entire surface into topologically equivalent disk surface patches and organizing the surface into an abstract graph of vertices, edges (curves), and faces (surface patches). In this structure, a surface patch acts as a face, an edge is defined as the common boundary between two surface patches, and a vertex is the meeting point of at least three neighboring surface patches.

By assigning each surface patch a binary labeling variable to indicate whether it should be retained, we significantly reduce the number of operational units compared to using a triangle as a unit. We then formulate topological constraints using Euler's characteristic formula, allowing users to specify different genus values. Furthermore, we observe that in a connected graph with n nodes, a capacity configuration with $n-1$ '1' s and one '$n-1$' leads to a non-vanishing flow plan. This insight allows us to transform the connectivity requirements into a max-flow problem. Additionally, to ensure the outcome closely resembles the original model, we introduce the alpha-wrapping technique to extract the approximate outer layer, which helps define the adherence of surface patches. As the teaser figure demonstrates, our method can produce three repaired models with genera of 0, 2, and 3, while maintaining a single component.

Our contributions are three-fold:

- We integrate multiple mesh repair requirements into our integer programming framework. These include not only manifoldness and watertightness but also connectedness and the ability to specify different genera. As demonstrated in the teaser figure, users can select various genera to achieve diverse outcomes.
- We observe that a connected graph with n nodes can maintain a non-vanishing flow when equipped with a capacity configuration like $(n-1,1,1,\ldots)$, allowing us to frame the connectivity requirements as a max-flow problem.
- To ensure the final result closely resembles the original model, we utilize the alpha-wrapping technique to generate an approximate outer layer of the model. This technique helps in determining the adherence of surface patches.

2. Related work

2.1. 3D model repair

There is a substantial body of literature on the topic of mesh repairing; see [1,2] for a comprehensive survey. Some methods [10,11] directly operate on the mesh surface, typically requiring the decomposition of the mesh at singular points and non-manifold edges, followed by selecting faces that best adhere to the original model while respecting manifold constraints. Some works [12,13] addressed the issue of self-intersections by robustly subdividing intersecting faces using a mixed geometric kernel, producing a 2-manifold triangle mesh. Additionally, other methods focus on dividing the space into interior and exterior zones, extracting the dividing surface as the manifold output. These methods employ various space discretization techniques, including occupancy grid voxelization [14,15], BSP partitioning [16], and tetrahedral meshing of space [17–19]. Furthermore, Nan and Wonka [4] proposed a binary linear programming formulation to find an appropriate combination of fitting planes to ensure the output polygonal surface is manifold and watertight.

In summary, while the issues [20] of manifoldness and watertightness have been extensively discussed and addressed in past research, explicitly specifying topological and connectivity constraints continues to pose substantial challenges, which is the focus of this paper's research.

2.2. Topology and connectivity control

Topology control is challenging because the topology of a given polygonal surface is characterized by its overall structure, rather than solely by local connections. Early research on this topic typically required manual interaction [21–25]. Recent Works [26,27] proposed subdividing the space into units and developing a set of basic surface primitives for each unit, followed by solving a combinatorial optimization problem subject to the overall topological constraints. Lazar et al. [3] explored topology control and demonstrated its applications in reconstruction from cross-section slices and iso-surfacing an intensity volume. They also introduced a branch-and-bound technique to quickly filter out infeasible solutions. However, their algorithm requires the input data to be organized layer by layer, which limits its practical applicability.

With advancements in deep learning, an increasing number of researchers have begun to utilize learning-based techniques to address the problem of topology control. Zhou et al. [28] introduced the Topology Net, a network capable of fitting topological representations

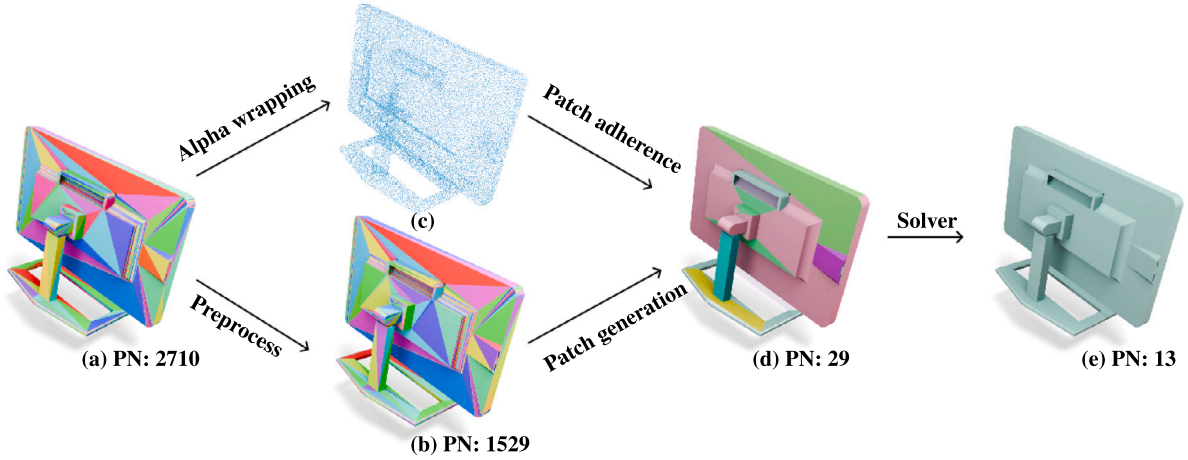

Fig. 2. Overview of our method. 'PN' denotes the number of units, with each unit visualized in a distinct color. (a) Input Polygonal Surfaces: We start with the given polygonal surfaces. (b) Processing Vertices and Faces: Coincident vertices and duplicate faces are made unique. New vertices and edges are added to address self-intersections. (c) Surface Sampling: By sampling the original surface and utilizing 3D alpha wrapping, we obtain an approximate representation of the outer layer. (d) Surface Decomposition: The surface is decomposed into patches that are topologically equivalent to disks. (e) MILP Formulation: Our Mixed Integer Linear Programming (MILP) formulation simultaneously considers manifoldness, watertightness, and connectedness, and specifies genera, leading to a desired repaired outcome.

from input point cloud data. Additionally, These works [28–31] have employed persistent homology diagrams (PHD) to capture and quantify the topological features of the underlying structures of trained models, thereby generating models with specified topologies. However, these approaches, lacking explicit control over topology, are applicable only to certain types of data and have very limited scalability.

Besides the challenge of topology control, controlling the number of connected components presents additional challenges, with few advancements thus far. In this paper, we focus on the explicit representation of topology and connectivity constraints.

2.3. Integer programming for stitching surface primitives

Optimization [32,33] is a common tool for obtaining 3D models. Due to the combinatorial nature inherent in mesh repair, integer programming is a commonly used formulation [34]. For example, some studies [3,4] assigned a binary variable to each candidate face to indicate whether it should be selected. Instead of using a triangle face as a unit, Jiang [35] first performed primitive fitting and then divided the fitted primitives into patches. This approach reduces the number of candidate faces, thereby improving the run-time performance and enabling the handling of more complex models.

The utilization of integer programming in mesh repair involves three main considerations. First, how to minimize the number of variables as much as possible. Second, how to define the objective function so that the repaired model closely resembles the original model while correcting various artifacts. Finally, how to translate various geometric requirements into algebraic constraints. The research in this paper addresses these three aspects.

3. Our method

3.1. Overview

Research goal. The theme of this paper is to repair a polygonal surface $S = (V, E, F)$ that may include various defects, such as open boundaries, self-intersections, non-manifoldness, topological errors, and disconnected patches. In addition to requirements for manifoldness and watertightness, we also consider connectivity constraints and user-specified topological constraints.

Algorithmic pipeline. Fig. 2 provides an overview of our method. Our approach begins with a preprocessing step. During this phase, we make coincident vertices unique, merge proximate vertices, and eliminate both duplicate and degenerate faces. Additionally, we introduce new vertices and edges at locations of self-intersections. Subsequently, we decompose the entire surface into a collection of surface patches, each of which is a topological homeomorphism to a disk. To this end, the surface is represented by $\mathcal{S} = (\widetilde{V}, \widetilde{E}, \widetilde{F})$, where $\widetilde{F}$ is a set of surface patches, $\widetilde{E}$ is the edge set with each edge being the common boundary between two adjacent surface patches, and $\widetilde{V}$ is the vertex set with each vertex being the common point among three surface patches. This decomposition allows us to treat each surface patch as a unit, significantly reducing the computational overhead. Simultaneously, we generate an approximate outer layer of the input model to gauge the fidelity of each surface patch to the original surface. Lastly, using our Mixed Integer Linear Programming (MILP) formulation, we simultaneously address manifoldness, watertightness, and connectedness, while specifying genera, thereby achieving the desired repaired outcome.

Formulation. We use $\mathcal{S}_{\text{wrap}}$ to denote the outer layer of the original model, with an alpha wrapping parameter α_{wrap}. Let $\widetilde{F} = \{\widetilde{f}_i\}_{i=1}^{N}$ represent the atlas set formed by clustering the triangles into surface patches. Our objective function for optimization is expressed as follows:

$$\begin{aligned} \text{Maximize} \quad & E = \sum_{i=1}^{N} l(\widetilde{f}_i) \frac{|\widetilde{f}_i \cap \mathcal{S}_{\text{wrap}}|}{|\widetilde{f}_i|^{1-\beta}} \\ \text{s.t.} \quad & \mathcal{S}_{\text{output}} \text{ is a 2-manifold;} \\ & \mathcal{S}_{\text{output}} \text{ has a genus of } g; \\ & \mathcal{S}_{\text{output}} \text{ forms one connected component,} \end{aligned} \tag{1}$$

where $|\cdot|$ defines the operation of taking the area of a surface or surface patch, $l(\cdot) \in \{0, 1\}$ defines a binary variable to indicate whether the corresponding element is selected, $\beta \in [0, 1]$ is a parameter to tune the degree to which the area of a surface patch is emphasized, and g the user-specified genus. Additionally, we use $l(\widetilde{v}_i) \in \{0, 1\}$, $l(\widetilde{e}_i) \in \{0, 1\}$, and $l(\widetilde{f}_i) \in \{0, 1\}$ as the variables to denote whether the corresponding element is retained in the final output of the repaired model. Details will be elaborated later.

3.2. Data preprocessing

Our method begins with a data processing step, with the primary task being to ensure the uniqueness of vertices and faces of S. First, we

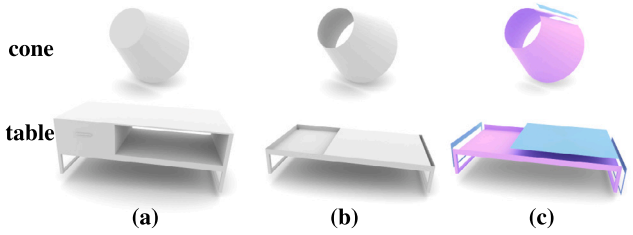

Fig. 3. Surface Decomposition into topologically equivalent disks: Top: The side face is decomposed into two patches because it is topologically homeomorphic to an annulus, rather than a disk. Bottom: The model has non-manifold edges where the surface will be decomposed. An overlay visualization is provided for clarity.

construct a kd-tree for the vertices in $\mathcal{S}$, which facilitates the search for the nearest neighbor or multiple neighbors. The subsequent tasks include the following items:

1. Merging Close Vertex Pairs: For a vertex v_i, we find the neighbors $\{v_j\}_{j=1}^{k}$ in a ϵ-size neighborhood, where ϵ is set to 1% of the average edge length of $\mathcal{S}$. We treat v_i and v_j as a pair of equivalent vertices. By union finding, we can finish merging all close vertex pairs, as well as removing vertex coincidence.
2. Removal of Duplicate Triangles: After the vertex merging process, the resulting triangles may become entirely superimposed. For any triangle represented as $f_i =\triangle\ v_{j_1}v_{j_2}v_{j_3}$, we arrange j_1, j_2, j_3 in increasing order. Assuming without loss of generality that j_1 is the smallest and j_3 the largest, we then map the identity of f_i to $j_1 + j_2 \times M + j_3 \times M^2$, where M is a sufficiently large integer. This mapping gives duplicate triangles the same identifier, facilitating their unique identification.
3. Elimination of Self-Intersections: We adopt the method described in [36] to eliminate both intersections and self-intersections from the model. The approach begins with detecting intersections between triangles to precisely locate each intersection point. In this context, all intersection points are represented using indirect offset predicates, and the necessary geometric predicates are defined. Based on these definitions, a splitting tree is constructed to facilitate rapid localization and querying of the intersection points. Finally, the intersection points are classified by type, and the intersecting triangles are re-triangulated accordingly.

3.3. Surface decomposition

As mentioned above, we decompose the entire surface into a collection of surface patches, each homeomorphic to a disk. Consequently, the surface can be abstractly represented by $\mathcal{S} = (\widetilde{V}, \widetilde{E}, \widetilde{F})$, where:

- $\widetilde{V} = \{\widetilde{v}_i\}$ denotes the vertices,
- $\widetilde{E} = \{\widetilde{e}_i\}$ represents the edges (curves), and
- $\widetilde{F} = \{\widetilde{f}_i\}$ from $i = 1$ to N corresponds to the faces (surface patches).

Topologically homeomorphic to a disk. Homeomorphisms define a continuous function between two topological spaces that has a continuous inverse. Both the function and its inverse need to be continuous to preserve the structure of the spaces. In our scenario, a surface patch is valid if it is akin to a disk:

- Being a simply connected region, which means there is only one boundary curve, and any curve drawn within the patch can be continuously shrunk to a point without leaving the patch.
- Having no holes, which distinguishes it from shapes like an annulus or a torus.

Flooding-based triangle clustering. Let $\widetilde{f}_i$ be the ith surface patch, consisting of a set of triangles. During the flooding process, we monitor changes in the boundary curve. Initially, $\widetilde{f}_i$ contains only one triangle, forming a single boundary composed of three line segments. Any segment on this boundary can direct the expansion of $\widetilde{f}_i$, generating a set of candidate triangles. To minimize memory usage, we utilize Breadth-first search (BFS) traversal during the flooding. The process terminates when the addition of any adjacent triangle would increase the number of boundary segments.

Although our method is applicable to any patch segmentation that meets the required conditions, we incorporate an additional strategy to enhance the regularity of the patches obtained through propagation. Specifically, during the propagation process, we compute the angle between the normals of two adjacent triangles; a smaller angle corresponds to a higher propagation priority. In the context of CAD models, if two adjacent triangular facets are perpendicular (i.e., the angle is 90°), the propagation terminates at that boundary. As shown in Fig. 3, we present the results of partitioning both a frustum and a table into patches, with all patches satisfying the aforementioned conditions. Note that the above propagation process depends on connectivity but is independent of orientations. Therefore, even when a Möbius torus is used as input, our method can still partition it into patches that satisfy the required conditions.

After the flooding process, we maintain $\widetilde{V} = \{\widetilde{v}_i\}$ and $\widetilde{E} = \{\widetilde{e}_i\}$, where a vertex is defined as the meeting point of at least three neighboring surface patches, and an edge is identified as the common boundary between two surface patches.

3.4. Adherence measurement

Generation of alpha-wrapping structure. It is generally expected that the repaired model should resemble the original defective model as closely as possible. This requires computing the outer layer of the original model. In our implementation, we first generate a dense set of sample points from $\mathcal{S}$ and then use the well-established Alpha-wrapping solver [37] to achieve this goal.

The choice of the alpha parameter plays a critical role in ensuring the reliability of the extracted outer layer. If the parameter α is set excessively large, the resulting α-wrap drifts far from the true surface, bridging distant vertices and even absorbing noise so that spurious elements are misclassified as part of the outer layer. Conversely, an α that is too small fragments the wrap into disconnected components, leaving portions of the genuine surface uncovered or inadvertently trimmed away.

Given that many input meshes are of only moderate quality, we set the wrapping parameter α_{wrap} to half the average edge length of the surface $\mathcal{S}$ to balance robustness and fidelity.

Although alternative methods can be used to extract the outer surface—such as [38]. But many implicit surface reconstruction techniques fail to faithfully capture model geometry, particularly in the presence of complex internal structures (e.g., self-intersections or occluded regions). To ensure general applicability across diverse datasets, we adopt alpha wrapping as our default approach.

Adherence area. For ease of computation, we convert both $\widetilde{f}_i$ and $\mathcal{S}_{\text{wrap}}$ into two sets of points. By abusing notations, we continue to use $\widetilde{f}_i$ and $\mathcal{S}_{\text{wrap}}$ to refer to these discrete point sets. For each point p in $\widetilde{f}_i$, we identify a corresponding point in $\mathcal{S}_{\text{wrap}}$ that is within a distance not exceeding α_{wrap}, and similarly, we exhaustively match close points in $\widetilde{f}_i$ for each point in $\mathcal{S}_{\text{wrap}}$. Consequently, we establish a correspondence graph $\mathcal{G}$ between subsets of $\widetilde{f}_i$ and $\mathcal{S}_{\text{wrap}}$, where $\mathcal{G}$ is structured as an unweighted bipartite graph. To determine one-to-one correspondences between the two subsets, we frame this as a maximal matching problem, which can be efficiently solved using the Hopcroft–Karp algorithm [39], with a computational complexity of $O(\sqrt{m_v}m_e)$, where m_v and m_e represent the numbers of nodes and edges in $\mathcal{G}$ respectively.

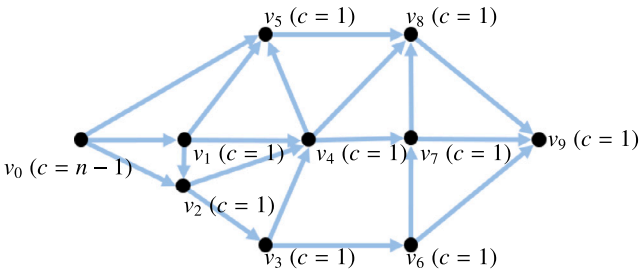

Fig. 4. Suppose the graph consists of a single connected component with n nodes. We arbitrarily select one node and assign it a capacity of $n-1$ units ($c=n-1$), while the remaining $n-1$ nodes are each assigned a capacity of one unit ($c=1$). By designating the highest-capacity node as the source, it is evident that there exists an integer-based flow plan to maximize the flow, allowing a flow of $n-1$ units. This observation motivates us to translate the connectivity constraint into a series of integer-based equations.

Adherence measurement. In this subsection, we define the degree to which a surface patch $\widetilde{f}_i$ adheres to the outer layer $\mathcal{S}_{\text{wrap}}$. For this purpose, we introduce the following ratio:

$$\frac{|\widetilde{f}_i \cap \mathcal{S}_{\text{wrap}}|}{|\widetilde{f}_i|^{1-\beta}}, \tag{2}$$

where $\widetilde{f}_i \cap \mathcal{S}_{\text{wrap}}$ represents the part of $\widetilde{f}_i$ that adheres to $\mathcal{S}_{\text{wrap}}$, β within the range $[0,1]$ is a parameter used to adjust the emphasis on the area of a surface patch. Here, $|\cdot|$ denotes the operation of taking the area of a surface or a surface patch. Specifically, setting $\beta=0$ encourages the selection of more surface patch units, giving priority to smaller patches, while $\beta=1$ focuses on maximizing the overall area of the output $\mathcal{S}_{\text{output}}$. We adopt the aforementioned term as the optimization objective, inspired by the work of [34,40], which seeks to produce a model that best preserves the desired visual effect. In this way, our method yields a result that not only closely aligns with the original appearance but also satisfies the specified constraints. Fig. 11 demonstrates the influence of the parameter β.

3.5. Various constraints

Let $\widetilde{V}=\{\widetilde{v}_i\}$, $\widetilde{E}=\{\widetilde{e}_i\}$, and $\widetilde{F}=\{\widetilde{f}_i\}$ represent the vertices, edges, and faces, respectively, with each surface patch being treated as a face. Concurrently, we use $l(\widetilde{v}_i)\in\{0,1\}$, $l(\widetilde{e}_i)\in\{0,1\}$, and $l(\widetilde{f}_i)\in\{0,1\}$ to denote whether the corresponding element is retained in the final output of the repaired model.

Edge manifoldness and watertightness. Like the manifoldness and watertightness requirements of a triangle mesh surface, we also require each edge to be incident to two neighboring faces for the surface patch atlas $\mathcal{S}=\{\widetilde{f}_i\}_{i=1}^{N}$. At the same time, each vertex is incident to a set of neighboring faces. It is a manifold vertex if the dual to the neighboring faces forms a single cycle.

In this paper, we consider only the 2-manifold edge constraints, leaving the study of the 2-manifold vertex constraints as future work. To this end, we write the manifoldness and watertightness constraints as follows:

$$\sum_{\widetilde{f}_j \text{ around } \widetilde{e}_i} l(\widetilde{f}_j) - 2\cdot l(\widetilde{e}_i) = 0, \quad \forall \widetilde{e}_i \in \widetilde{E}. \tag{3}$$

Genus constraints. The Euler characteristic is a topological invariant, meaning it remains unchanged under continuous deformations of the object. Although in our scenario the face becomes a curved surface patch, Euler's characteristic formula still holds:

$$\sum_i l(\widetilde{v}_i) + \sum_j l(\widetilde{f}_j) - \sum_k l(\widetilde{e}_k) = 2-2g, \tag{4}$$

where g is the user-specified genus.

Connectivity constraints. As Fig. 4 shows, we assume that the graph consists of a single connected component with n nodes. Each point v represents a patch resulting from the decomposition of the original model, and an edge between two points indicates that the corresponding patches are adjacent in the original geometry. Since the output model is required to be manifold and watertight, any patch included in the output must be reachable from every other included patch; in other words, the selected patches must form a connected component. We arbitrarily select one node and take it as the source, with the capacity set to $n-1$ units. At the same time, we set the capacity of the other nodes to one unit. Based on the capacity setting, we consider the max-flow plan. It can be proved that the maximum flow of $n-1$ units can be achieved; see [43] for the detailed discussion. In implementation, we take the vertex with the identity of 0 as the source and the other vertices as the destinations, and take each undirected edge as a pair of directed edges. Due to the property that the maximum flow of $n-1$ units can be achieved, we enforce the flow constraints for each of the vertices. For the source vertex, we require:

$$flow(\widetilde{e}_j) \le |\widetilde{V}|\ l(\widetilde{e}_j) \tag{5}$$

and

$$\sum_{\widetilde{e}_j \text{is outward}} flow(\widetilde{e}_j) = -(n-1) \tag{6}$$

In contrast, we require:

$$flow(\widetilde{e}_j) \le |\widetilde{V}|\ l(\widetilde{e}_j)\ \textit{and}$$
$$\sum_{\widetilde{e}_j \text{is inward}} flow(\widetilde{e}_j) - \sum_{\widetilde{e}_j \text{is outward}} flow(\widetilde{e}_j) = 1 \tag{7}$$

for the other vertices.

4. Evaluation

In this section, we showcase the results of our algorithm in repairing synthetic models. We analyze the computational performance of each step, demonstrating that our Mixed-Integer Linear Programming (MILP) approach can achieve results that meet constraints in a very short amount of time. Additionally, we discuss how the outcomes vary for the same model when different topological constraints are applied. It is worth noting that all results produced in the following experiments are guaranteed to be watertight and manifold, as these requirements are enforced as hard constraints in our formulation. Furthermore, we conducted ablation experiments to verify the effectiveness of the β parameter. Unless explicitly stated otherwise, the parameter β is 0.5.

4.1. Computational performance

In this section, we evaluate the efficiency of our algorithm by randomly selecting models from ModelNet and ShapeNet. We conducted all the experiments with an AMD Ryzen 9 7900X 12-core CPU and 64 GB of memory. Our implementation was fully developed in C++.

4.1.1. Preprocessing cost

Our preprocessing primarily involves mesh self-intersection detection and repair, as well as mesh patch division. Since we require the resulting patches to be topologically homomorphic to disks, our preprocessing time is relatively long. Generally, the preprocessing duration is directly proportional to the model's complexity. As shown in Table 1, models with a higher number of input faces and patches require longer preprocessing times. However, even for meshes containing tens of thousands of faces, our preprocessing can be completed within 12 s.

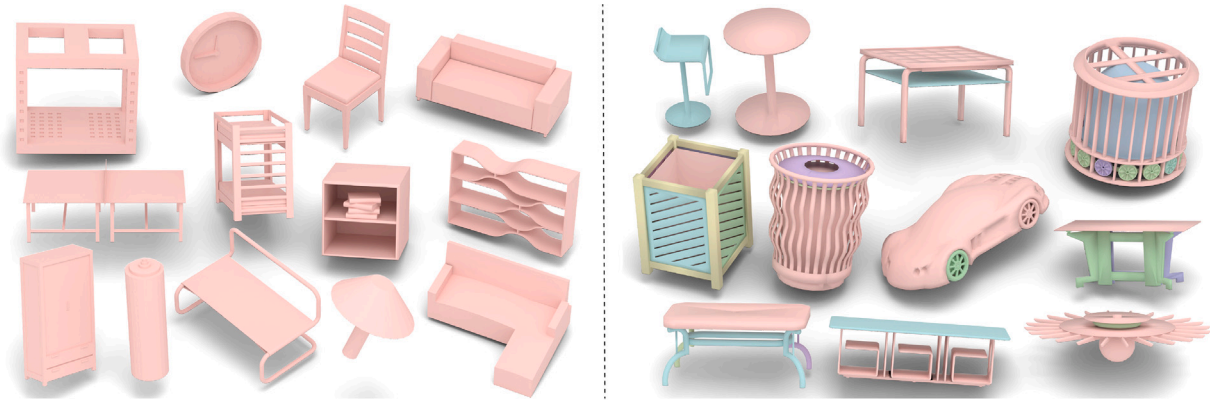

Fig. 5. More results by processing ModelNet [41] and ShapeNet [42] datasets. The left side shows results for one component with the number of genera that matches the appearance, and the right side shows the results obtained for all its connected branches.

Table 1
The table presents the experimental results for a subset of models we tested. For various models, under user-specified genus constraints, our method consistently completes the preprocessing and MILP within an exceptionally short timeframe.

#vertices	#faces	Preprocessing time	#patch	#Binary var	#Integer var	#genus	MILP time
5420	11 298	6.72991	26	163	83	3 6	0.007843 0.00818
650	1710	1.47584	94	327	151	0 9	0.021972 0.023029
106	286	0.478432	25	89	42	0 5	0.00891 0.006728

Fig. 6. The figure shows the repair results of a model with non-manifold edges. The numbers on the model edges are the number of non-manifold edges(NE) before repair.

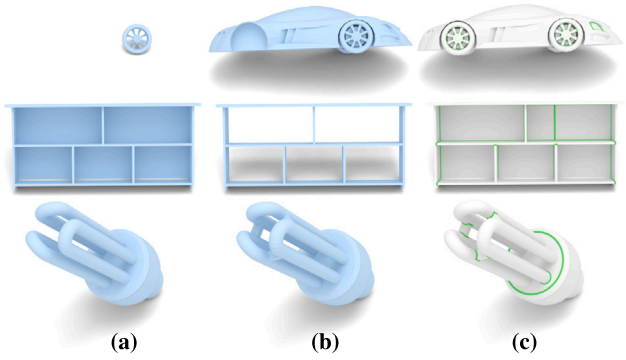

Fig. 7. The figure displays our experimental results. Panel (a) illustrates a connected component extracted by our algorithm that meets a specified genus. Panel (b) by specifying different genera, our method is able to extract results with the designated genus from the input model. In panel (c), shows the outcome following the initial optimization by [34], which results in open boundaries requiring post-processing for correction.

4.1.2. Efficiency of the MILP solver

We employ the Gurobi optimizer to solve the Mixed-Integer Linear Programming (MILP) problem. By simplifying the triangular surfaces into a collection of patches, we significantly reduce the number of variables. Additionally, by expressing manifold, genus, and connectivity constraints as first-order linear equations, our algorithm's optimization process is exceptionally fast. As demonstrated in Table 1, our models can typically obtain results with specified genus within just 1 ms.

4.2. Comparison

We compared our results with those obtained by others as mentioned in [34], who required additional processing to extract manifold surfaces after their initial optimization, due to the adjacency relationships among the mesh patches they obtained. We have incorporated manifold constraints directly into our integer programming approach, enabling us to obtain manifold watertight mesh surfaces through a single optimization process.

As shown in Fig. 5, our method achieves results comparable to those obtained through post-processing by others. By incorporating genus and connectivity constraints, our approach is capable of accomplishing more complex mesh repair tasks that meet specified topological conditions. We screened out models with defects from Thingi10K, ABC, ModelNet, ShapeNet and other datasets for testing. The most common defect is non-manifold edges. As shown in Fig. 6, our method can extract manifold and watertight results from models containing non-manifold edges. For example, as shown in Fig. 7, the bookcase can be processed by specifying the genus, which allows for the selective

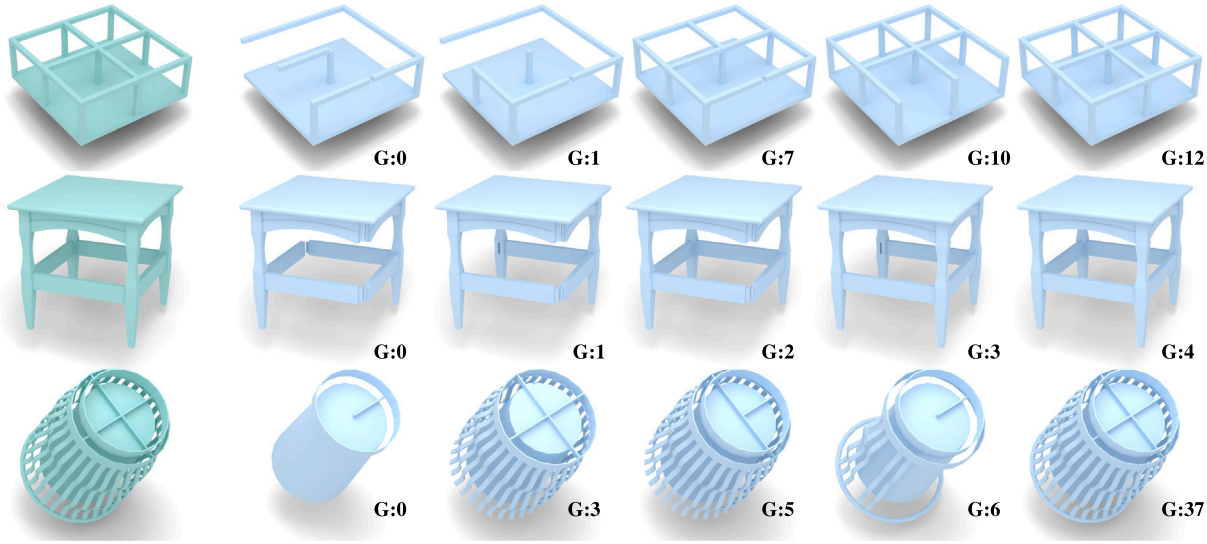

Fig. 8. In the image, the mesh on the far left is the one to be repaired, containing defects such as self-intersections and non-manifold edges. On the right are the reconstruction results with specified genera. G denotes the genus of the output model.

removal of the panels on the back of the bookcase. This approach enables the extraction of different repair results from the same input.

4.3. Topological control

We randomly selected a subset of models from the ModelNet and ShapeNet datasets to validate the effectiveness of our mesh repair framework in controlling model genus. As shown in Fig. 8, the left side presents an input mesh surface with defects. After preprocessing, the mesh is divided into a group of patches topologically homomorphic to disks, and the right side shows the results of mesh repair when the user specifies a genus. When different genera are input, the mixed-integer programming outputs a patch combination that satisfies the constraints and maximizes the objective function. For example, in the data shown at the bottom, when the genus is 0, it selects the largest combination of patches that can form a watertight manifold surface. As the genus increases, the selection of patches changes accordingly. We only display genus values where there are significant changes in the mesh structure in the figure. Beyond these values, our method can also produce outputs that meet the requirements when other genera are specified by the user. It is important to note that not all specified genera can yield results. Our method can only extract results with a specified genus when the input model's triangles can form the constraints for that genus.

4.4. Connected control

In the ModelNet and ShapeNet datasets, many models are fragmented into multiple pieces. We filtered out these models and randomly selected some for testing. As shown in Fig. 5 left, our method consistently ensures that the extracted mesh is a single connected component, regardless of the genus specified by the user. If the user desires results with multiple connected components, our method also supports further selection within the patches not chosen initially.

As depicted in Fig. 5 right, the pink connected component represents the result initially selected by the integer programming procedure. We then reran the program on the unselected patches to select new connected components, which are shown in other colors in the diagram.

4.5. Robustness

Broken mesh. In practical applications, input models often contain gaps, placing stringent demands on the robustness of any repair algorithm. For meshes with open boundaries, we first adopt the hole-filling strategy proposed in [44] to seal the openings, and then apply our method to extract a manifold, watertight mesh that satisfies the required properties. As illustrated in Fig. 9, the final result is obtained by first filling the holes and subsequently repairing the model using our algorithm.

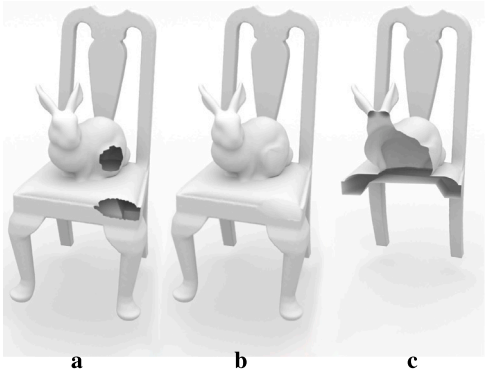

Fig. 9. (a) The input mesh contains several holes. (b) After conventional hole-filling, the mesh still exhibits geometric defects, such as self-intersections. (c) Our method produces a watertight, manifold mesh; the figure shows an interior cross-section of the final result.

Noise. The results produced by our method are guaranteed to be watertight and simply connected. In addition, the method exhibits strong robustness against noise, effectively filtering out extremely small, thin, or chaotically oriented patches, as well as those disconnected from the main structure. As Fig. 10 shown, spurious artifacts present in the input model do not propagate to the final output.

Other geometric defects. Our method is also robust to a wide range of geometric imperfections. By enforcing that the output mesh is both manifold and watertight, it can effectively repair input models containing self-intersections and non-manifold edges. Furthermore, since our approach does not rely on input normals, it naturally handles models with inconsistent or flipped normals by reorienting face normals consistently in the output.

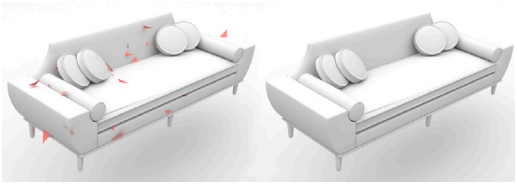

Fig. 10. The left shows the input mesh, where the red regions highlight noise composed of sparse, disconnected triangles. The right presents the output mesh, which is manifold, watertight, and has genus zero.

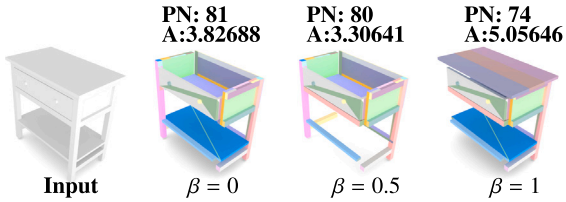

Fig. 11. The figure demonstrates the results obtained on the same input model when constraining the mesh genus to 0, under different values of β. Each combination of colored patches represents a selected patch. This visualization helps in comparing how the variation in β values influences the selection of patches and, consequently, the topology and geometry of the final mesh output. (PN means the number of selected patches and the A means the model's surface area).

Table 2
This table presents the detailed results corresponding to the three alpha values shown in Fig. 12. The Hausdorff distance reported here is the approximate one-sided Hausdorff distance.

alpha	#Verts	#Faces	Time (s)	Hausdorff distance
avg_EdgesLength/10	157 466	315 036	20.069	$4.260 * 10^{-2}$
avg_EdgesLength/2	8074	16 232	1.536	$4.859 * 10^{-2}$
avg_EdgesLength	3052	6116	0.648	$5.942 * 10^{-2}$

4.6. Ablation study of β

The parameter β is a critical variable in our objective function, quantifying the impact of the patch area on its weight. As β approaches 0, the influence of patch area on the selection outcome diminishes, and the model tends to choose as many patches as possible. Conversely, as β approaches 1, the impact of patch area on the selection results increases, and the model tends to select the outcome with the largest surface area. As illustrated in Fig. 11, when using the exact same inputs and constraints, a β value of 1 yields a model with the largest surface area, whereas a lower β results in a model utilizing the maximum number of patches.

4.7. Ablation study of *alpha*

The parameter alpha in alpha-wrapping controls the extent to which the output geometry preserves the shape characteristics of the input model in our method. To evaluate its impact, we conducted a controlled experiment analyzing how different alpha values influence the results of alpha wrapping. As shown in Fig. 12 and Table 2, a very small alpha value leads to significantly increased computational time, while a large alpha value results in excessive smoothing and the loss of important geometric details. Based on our observations, setting alpha to half of the average edge length of the input model provides a favorable balance between accuracy and efficiency. This choice enables the subsequent algorithm to operate on a reconstructed surface that closely adheres to the original input geometry.

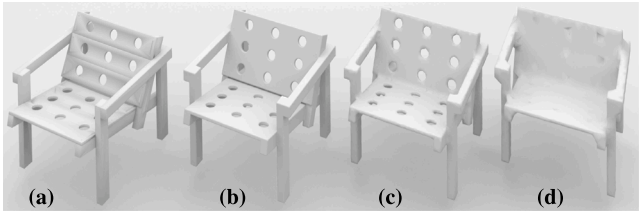

Fig. 12. This figure illustrates the influence of the alpha parameter on the resulting geometry when applying alpha wrapping to the input model. (a) shows the original input model. (b), (c), and (d) depict the wrapped results using alpha values set to one-tenth, one-half, and equal to the average edge length of the input mesh, respectively.

5. Conclusions and limitations

In this paper, we adopt the approach of integrating various requirements into a unified integer programming framework. In addition to the fundamental requirements of manifoldness and watertightness, we consider the explicit representation of topological and connectivity requirements. Specifically, we observe that in a connected graph with n nodes, a capacity configuration with $n-1$ '1' s and one '$n-1$' leads to a non-vanishing flow plan. This insight allows us to transform the connectivity requirements into a max-flow problem. Additionally, we introduce the alpha-wrapping technique to extract the approximate outer layer, which helps define the adherence of surface patches. As experimental results demonstrate, our method can produce repaired models with varying genera while maintaining a single component.

Limitations and future work. Currently, our method does not support users specifying the number of connected components. Additionally, as shown in the teaser, we can only extract results with genera of 2, 5, and 8. If other genera are specified, our method fails to produce results because no combination can form the specified genus. At the same time, when the input mesh is extremely large (e.g., containing millions of faces), our preprocessing typically decomposes it into dozens to hundreds of patches, making the resulting MIP problem tractable for the optimizer. However, for meshes with highly complex structures, the decomposition may yield thousands or even tens of thousands of patches—far exceeding the capabilities of current MIP solvers. In future research, we aim to generate multiple feasible solutions simultaneously, facilitating users in finding the desired surface.

CRediT authorship contribution statement

Jiantao Song: Writing – original draft, Methodology, Formal analysis, Data curation, Conceptualization. **Wensong Wang:** Writing – original draft, Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Rui Xu:** Writing – original draft, Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Wenlong Meng:** Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Shuangmin Chen:** Writing – original draft, Methodology, Formal analysis, Conceptualization. **Shiqing Xin:** Writing – original draft, Methodology, Formal analysis, Conceptualization. **Taku Komura:** Methodology, Formal analysis, Conceptualization. **Changhe Tu:** Methodology, Formal analysis, Conceptualization. **Wenping Wang:** Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Key R&D Program of China (2022YFB3303200) and the National Natural Science Foundation of China (U23A20312, 62272277).

Data availability

Data will be made available on request.

References

[1] Ju T. Fixing geometric errors on polygonal models: a survey. J Comput Sci Tech 2009;24(1):19–29.
[2] Attene M, Campen M, Kobbelt L. Polygon mesh repairing: An application perspective. ACM Comput Surv 2013;45(2):1–33.
[3] Lazar R, Dym N, Kushinsky Y, Huang Z, Ju T, Lipman Y. Robust optimization for topological surface reconstruction. ACM Trans Graph 2018;37(4):1–10.
[4] Nan L, Wonka P. Polyfit: Polygonal surface reconstruction from point clouds. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 2353–61.
[5] Zhang S, Xu G, Wu H, Gu R, Qi L, Pang Y. Medial hex-meshing: high-quality all-hexahedral mesh generation based on medial mesh. Eng Comput 2024;40(4):2537–57.
[6] Zhao Q, Xu G, Xiao Z, Wu H, Gu R, Liu Y, Pang Y. Bc-hexmatching: an improved hexahedral mesh matching approach based on base-complex structure. Eng Comput 2024;40(4):2209–26.
[7] Gao J, Xiao Z, Shen S, Xu C, Cai J, Xu G. Improved hexahedral mesh generation from quadrilateral surface meshes. Comput Struct 2025;307:107620.
[8] Zheng W, Wu H, Xu G, Ling R, Gu R. Feature-preserving quadrilateral mesh boolean operation with cross-field guided layout blending. Comput Aided Geom Design 2024;111:102324.
[9] Qi L, Qiu J, Xu G, Liu Y, Xu J, Gu R, Lu F, Pang Y. RFF-meshing: a parallel and anisotropic quad-dominant mesh generation framework based on Riemann frame field. Eng Comput 2024;1–23.
[10] Rossignac J, Cardoze D. Matchmaker: Manifold breps for non-manifold r-sets. In: Proceedings of the fifth ACM symposium on solid modeling and applications. 1999, p. 31–41.
[11] Guéziec A, Taubin G, Lazarus F, Hom B. Cutting and stitching: Converting sets of polygons to manifold surfaces. IEEE Trans Vis Comput Graphics 2001;7(2):136–51.
[12] Attene M. Direct repair of self-intersecting meshes. Graph Model 2014;76(6):658–68.
[13] Attene M. ImatiSTL-fast and reliable mesh processing with a hybrid kernel. Trans Comput Sci XXIX 2017;86–96.
[14] Ju T. Robust repair of polygonal models. ACM Trans Graph 2004;23(3):888–95.
[15] Nooruddin FS, Turk G. Simplification and repair of polygonal models using volumetric techniques. IEEE Trans Vis Comput Graphics 2003;9(2):191–205.
[16] Murali T, Funkhouser TA. Consistent solid and boundary representations from arbitrary polygonal data. In: Proceedings of the 1997 symposium on interactive 3D graphics. 1997, p. 155–ff.
[17] Hu Y, Zhou Q, Gao X, Jacobson A, Zorin D, Panozzo D. Tetrahedral meshing in the wild.. ACM Trans Graph 2018;37(4):60.
[18] Hu Y, Schneider T, Wang B, Zorin D, Panozzo D. Fast tetrahedral meshing in the wild. ACM Trans Graph 2020;39(4). http://dx.doi.org/10.1145/3386569.3392385.
[19] Diazzi L, Attene M. Convex polyhedral meshing for robust solid modeling. ACM Trans Graph 2021;40(6):1–16.
[20] Huang J, Zhou Y, Guibas L. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. 2020, arXiv preprint arXiv:2005.11621.
[21] Sharf A, Lewiner T, Shklarski G, Toledo S, Cohen-Or D. Interactive topology-aware surface reconstruction. ACM Trans Graph 2007;26(3):43–es.
[22] Yin K, Huang H, Zhang H, Gong M, Cohen-Or D, Chen B. Morfit: interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control.. ACM Trans Graph 2014;33(6). 202–1.
[23] Bazin P-L, Pham DL. Topology-preserving tissue classification of magnetic resonance brain images. IEEE Trans Med Imaging 2007;26(4):487–96.
[24] Zeng Y, Samaras D, Chen W, Peng Q. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N–d images. Comput Vis Image Underst 2008;112(1):81–90.
[25] Zhang Y, Wang Z, Zhao Z, Xu R, Chen S, Xin S, Wang W, Tu C. A hessian-based field deformer for real-time topology-aware shape editing. In: SIGGRAPH Asia 2023 conference papers. 2023, p. 1–11.
[26] Huang Z, Zou M, Carr N, Ju T. Topology-controlled reconstruction of multi-labelled domains from cross-sections. ACM Trans Graph 2017;36(4):1–12.
[27] Zou M, Holloway M, Carr N, Ju T. Topology-constrained surface reconstruction from cross-sections. ACM Trans Graph 2015;34(4):1–10.
[28] Zhou C, Dong Z, Lin H. Learning persistent homology of 3D point clouds. Comput Graph 2022.
[29] Hu J, Fei B, Xu B, Hou F, Yang W, Wang S, Lei N, Qian C, He Y. Topology-aware latent diffusion for 3D shape generation. 2024, arXiv preprint arXiv:2401.17603.
[30] Mezghanni M, Boulkenafed M, Lieutier A, Ovsjanikov M. Physically-aware generative network for 3d shape modeling. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 9330–41.
[31] Dong Z, Chen J, Lin H. Topology-controllable implicit surface reconstruction based on persistent homology. Computer- Aided Des 2022;150:103308.
[32] Xu R, Liu L, Wang N, Chen S, Xin S, Guo X, Zhong Z, Komura T, Wang W, Tu C. CWF: Consolidating weak features in high-quality mesh simplification. ACM Trans Graph 2024;43(4). http://dx.doi.org/10.1145/3658159, URL https://doi.org/10.1145/3658159.
[33] Ye Y, Wang Y, Cao J, Chen Z. Watertight surface reconstruction method for CAD models based on optimal transport. Comput Vis Media 2024;10(5):859–72.
[34] Chu L, Pan H, Liu Y, Wang W. Repairing man-made meshes via visual driven global optimization with minimum intrusion. ACM Trans Graph (SIGGRAPH ASIA) 2019;38(6):158:1–158:18. http://dx.doi.org/10.1145/3355089.3356507.
[35] Jiang J, Zhao M, Xin S, Yang Y, Wang H, Jia X, Yan D-M. Structure-aware surface reconstruction via primitive assembly. In: 2023 IEEE/CVF international conference on computer vision. ICCV, 2023, p. 14125–34. http://dx.doi.org/10.1109/ICCV51070.2023.01303.
[36] Guo J-P, Fu X-M. Exact and efficient intersection resolution for mesh arrangements. ACM Trans Graph 2024;43(6):1–14.
[37] Project TC. CGAL user and reference manual. 5.6.1. CGAL Editorial Board; 2024, URL https://doc.cgal.org/5.6.1/Manual/packages.html.
[38] Wen H, Wang L, Chen S, Xin S, Deng C, He Y, Wang W, Tu C. Ims: implicit shell for the sandwich-walled space surrounding polygonal meshes. Vis Comput 2025;1–14.
[39] Tassa T. Finding all maximally-matchable edges in a bipartite graph. Theoret Comput Sci 2012;423:50–8.
[40] Zheng Z, Gao X, Pan Z, Li W, Wang P-S, Wang G, Wu K. Visual-preserving mesh repair. IEEE Trans Vis Comput Graphics 2024;30(9):6586–97.
[41] Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J. 3D shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, p. 1912–20.
[42] Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, et al. Shapenet: An information-rich 3d model repository. 2015, arXiv preprint arXiv:1512.03012.
[43] Maksimović Z. A new mixed integer linear programming formulation for the maximum degree bounded connected subgraph problem. Publ de L' Inst Math 2016;99(113):99–108.
[44] Liepa P. Filling holes in meshes. In: Proceedings of the 2003 eurographics/ACM SIGGRAPH symposium on geometry processing. 2003, p. 200–5.