Repairing Man-Made Meshes via Visual Driven Global Optimization with Minimum Intrusion

LEI CHU*, The University of Hong Kong and Microsoft Research Asia
HAO PAN† and YANG LIU, Microsoft Research Asia
WENPING WANG, The University of Hong Kong

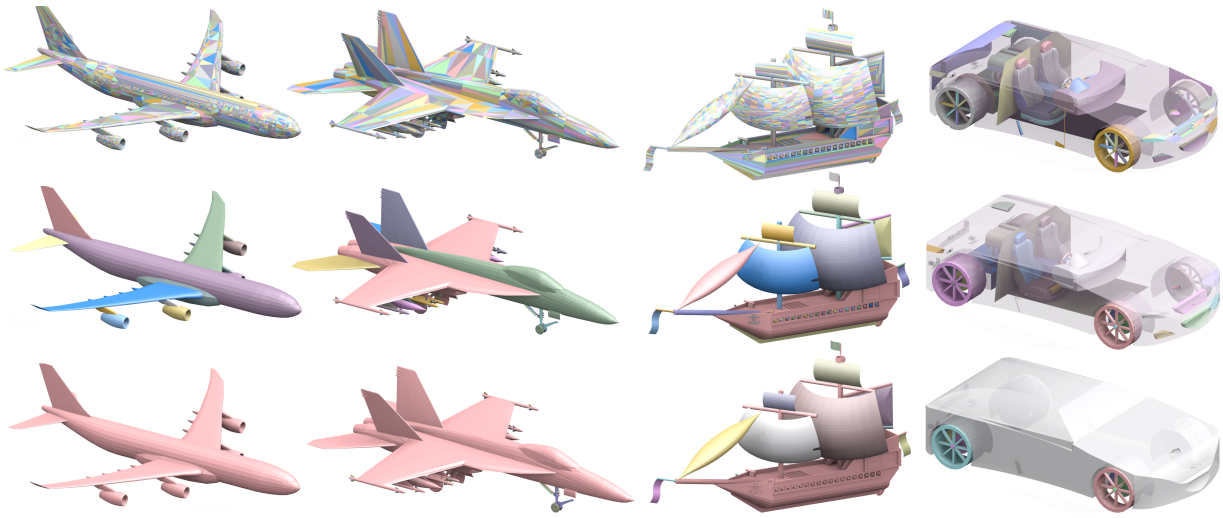


Fig. 1. Meshes from ModelNet [Wu et al. 2015] and ShapeNet [Chang et al. 2015] are repaired by our tool that produces large coherent manifold meshes preserving the input visual appearances and features. Each manifold surface patch is colored differently. From top to bottom: the input meshes with scattered and redundant patches, our results with meaningful components and the more concise results when hidden patches are removed (Sec. 6.2). The shell of the car model is rendered in transparency to show inner structures.

3D mesh models created by human users and shared through online platforms and datasets flourish recently. While the creators generally have spent large efforts in modeling the visually appealing shapes with both large scale structures and intricate details, a majority of the meshes are unfortunately flawed in terms of having duplicate faces, mis-oriented regions, disconnected patches, etc. , due to multiple factors involving both human errors and software inconsistencies. All these artifacts have severely limited the possible low-level and high-level processing tasks that can be applied to the rich datasets. In this work, we present a novel approach to fix these man-made meshes such that the outputs are guaranteed to be oriented manifold meshes that preserve the original structures, big and small, as much as possible. Our key observation is that the models all visually look meaningful, which leads to our strategy of repairing the flaws while always preserving the visual quality. We apply local refinements and removals only where necessary to achieve minimal intrusion of the original meshes, and global adjustments through robust optimization to ensure the outputs are valid manifold meshes with optimal connections. We test the approach on large-scale 3D datasets, and obtain quality meshes that are more readily usable for further geometry processing tasks.

CCS Concepts: • **Computing methodologies → Mesh models**.

Additional Key Words and Phrases: Mesh repair, ShapeNet, ModelNet, visual-driven, global optimization, output guarantee, minimal intrusion

ACM Reference Format:
Lei Chu, Hao Pan, Yang Liu, and Wenping Wang. 2019. Repairing Man-Made Meshes via Visual Driven Global Optimization with Minimum Intrusion. *ACM Trans. Graph.* 38, 6, Article 158 (November 2019), 18 pages. https://doi.org/10.1145/3355089.3356507

1 INTRODUCTION

The creation and processing of 3D digital models is a central problem in computer graphics. 3D models are most frequently represented by polygonal meshes, because of their flexibility and expressiveness. Alongside the flexibility, however, come the various artifacts that can be easily encoded by polygonal meshes but pose much difficulty for downstream applications [Attene et al. 2013]. For example, while a majority of geometry processing algorithms assume that the input

*This work was done when Lei Chu was an intern at Microsoft Research Asia.
†Corresponding author.

Lei Chu, The University of Hong Kong and Microsoft Research Asia, lchu@cs.hku.hk; Hao Pan, Yang Liu, {haopan,yangliu}@microsoft.com, Microsoft Research Asia; Wenping Wang, The University of Hong Kong, wenping@cs.hku.hk.

0730-0301/2019/11-ART158 $15.00
https://doi.org/10.1145/3355089.3356507

models are well-connected manifold surfaces with proper orientations, so that differential quantities, texture mappings, semantic segmentations and other advanced constructions can be applied on the surfaces, the meshes can easily be non-manifold or disconnected polygon soups with arbitrary and inconsistent orientations.

The problem becomes more urgent with the recent fast growing collections of man-made 3D models created by users with varying levels of modeling sophistication for diverse purposes; see [3DW 2018] for an example. On one hand, the data abundance raises hope for advanced 3D computational tasks, including shape analysis and synthesis, by also utilizing the maturing machine learning methods like deep learning that are data-hungry. To this end, various semantically labeled and organized 3D datasets are created, e.g. ModelNet [Wu et al. 2015] and ShapeNet [Chang et al. 2015]. On the other hand, these man-made meshes are frequently flawed with artifacts that prohibit further processing and hinder advanced applications built on clean data. For example, a majority of the models in ShapeNet are nonmanifold meshes with duplicated faces, extensive self-intersections, and fragmented and disconnected patches [Zhou and Jacobson 2016] (see also Figs. 1 and 3).

While previously many mesh repair methods have been developed [Attene et al. 2013], they mostly focus on fixing digitally acquired or man-made meshes representing solid objects, and are not suitable for processing meshes of these large-scale datasets that have visual plausibility but are far from representing valid 3D solids (see Sec. 7.4 for concrete evaluations). Indeed, an effective mesh repair procedure must fit for the characteristics of the source mesh models and the target applications. In this work, we present a novel tool designed for repairing these abundant meshes to enhance their usability for various downstream geometry processing tasks. Observing the mesh models extensively, we identify a mismatch between visual quality and structural quality, which guides the design of our approach. To be specific, we find the major characteristics of these man-made meshes are:

(1) visually, these models look meaningful in both large scale structures and fine level details, as they are crafted by the creators with great efforts and attention;
(2) structurally in terms of geometry and connectivity, however, the models frequently contain redundancy, mis-orientation and wrong tessellations, which can be attributed to both human errors and modeling software inconsistency.

As such, we take a visual perspective and fix the structural defects as much as possible while obeying the visual guidance and preserving the visual quality.

Our approach resolves the commonly found problems of duplications, mis-orientations and mis-connections with man-made 3D meshes by a pipeline consisting of two major steps. The first step applies a sequence of visual driven operations that measure, reorient, simplify, and intersect the local patches of the input mesh, to obtain a set of elementary manifold patches that have visually guided orientation preferences, potential connections and no redundancies; throughout this process, all delicate but important mesh structures are well preserved. The second step is a global combinatorial optimization that connects the isolated and mis-oriented elementary patches into coherent manifold surfaces while respecting the visual appearance of the mesh model as much as possible. We formulate the combinatorial optimization as an integer linear programming (ILP) that we solve efficiently with a novel approximate solver. Through our pipeline, the result mesh is guaranteed to be a clean manifold surface, has meaningful orientation and connections and minimum deviation from the input mesh, and is therefore more usable by subsequent geometry processing procedures.

We have applied our automatic mesh repair procedure to large scale 3D datasets, including ModelNet and ShapeNet (Sec. 7.2); downstream tasks, e.g. remeshing and texture mapping, critical for sophisticated applications are more readily carried out on the fixed models. Through comparison with previous methods, our tool is demonstrated to be more suitable for robustly processing these datasets and preserving high visual quality with meaningful structures (Sec. 7.4). While our tool does not fix problems like gaps and holes that can be well-defined only for solid objects, it complements other methods that target these defects by providing largely improved meshes as their input to work properly. The source code and processed datasets are open sourced[1] to facilitate future research.

2 RELATED WORK

Mesh repair is a practical problem that has been studied extensively for various situations. In particular, a large amount of previous works focus on repairing mesh models representing solid 3D objects for physical simulation and 3D printing, with the meshes either acquired from the real world by digitization or modeled manually [Attene et al. 2013; Ju 2009]. However, the particular challenge we address in this paper is a relatively new one, as it comes along with the abundance of diverse man-made 3D datasets which contain models mostly with visual plausibility rather than physical validity. Therefore, few existing mesh repair works are suitable for this problem; see Sec. 7.4 for detailed comparisons. Nonetheless, there are works with which our method shares similarities in design principles and methodology. In this section, we briefly review the related mesh repair works that feature global approaches with output guarantees, as well as the approaches that work directly on meshes for the sake of minimal intrusion; readers are referred to [Attene et al. 2013; Ju 2009] for extensive surveys of mesh repair methods. We also discuss the utilization of visual guidance and topology optimization in both mesh repair and other geometry processing tasks in general.

Global and direct mesh repair methods. As discussed in [Attene et al. 2013], mesh repair methods can be roughly classified into local approaches and global ones: while local approaches try to detect and fix a certain kind of defect, global methods generally fix a range of problems that are actually correlated and guarantee the output to be manifold surfaces bounding a 3D solid. Most global methods achieve such generality by using an intermediate volumetric 3D representation, e.g. an occupancy voxel grid [Ju 2004; Nooruddin and Turk 2003], a voxel grid encoding the distance function from boundary surface [Curless and Levoy 1996; Hornung and Kobbelt 2006; Kazhdan and Hoppe 2013], or irregular tessellations of the 3D volume using a BSP tree [Murali and Funkhouser 1997] or tetrahedra mesh [Hu et al. 2018]; the intermediate volumetric representation is then signed to distinguish interior from exterior, and the manifold

[1] See https://github.com/lei65537/visual_driven_mesh_repair.

boundary surface is finally extracted as the interface. While guaranteeing manifold output, a common problem with the global methods is that the input surfaces are generally resampled and remeshed, e.g. due to the scan conversion to regular grids or the re-tessellation by BSP trees. This may be undesirable, especially when the input meshes have sharp features, are hand-crafted with highly efficient polygons, or carry additional data like texture maps. In addition, these methods assume that the input surfaces enclose solid volumes, which however generally does not hold for the man-made 3D datasets we consider.

Different from the global methods that rely on an intermediate volumetric representation, there are tools that work on the polygonal meshes directly to avoid unnecessary changes of the input. For example, [Guéziec et al. 2001; Rossignac and Cardoze 1999] convert nonmanifold mesh models into manifold meshes by first decomposing them at singular vertices and edges and then greedily combining the parts while respecting manifoldness. MeshFix [Attene 2010] fixes digitized meshes bounding solid objects by detecting and repairing erroneous regions only. [Attene 2014] resolves all self-intersections of a mesh by subdividing the intersecting faces robustly using a hybrid geometric kernel [Attene 2017] to obtain a nonmanifold complex, and then extracts a manifold outer hull of the complex that consists of the most exterior faces by a repeated region growing procedure. [Attene 2016] takes a similar approach but focuses on making sure the outer hull bounds a valid volume for 3D printing.

Similar to the global methods, our approach guarantees the output mesh to be manifold, as we try to fix redundancies, inconsistent tessellations, flipped orientations and missing connections in one framework. On the other hand, different from the previous works, our method targets surface meshes that are frequently open rather than bounding 3D solids. To this end, our approach works with polygon meshes directly and makes the repair minimally intrusive, which shares similarity to [Attene 2014, 2016]. However, when processing the physically imprecise man-made models, it is common for the outer hulls to either miss the major components or apply uneven changes to them, while our visual guided approach works robustly and obtains meaningful structure preserving results (Sec. 7.4).

Visual guided geometry processing. Since generating visual content is fundamental to computer graphics, visual guidance is used in many geometry processing tasks. For example, mesh simplification has been driven by preserving the visual quality of 3D models while reducing the amount of mesh elements as much as possible [Garland and Heckbert 1997; Hoppe 1996; Lindstrom and Turk 2000; Zhang and Turk 2002]. Image-driven surface editing and geometric modeling takes one step further, as the user-specified visual difference is back-propagated to the object shape through inverse rendering procedures [Gingold and Zorin 2008; Liu et al. 2018].

Visual guidance is used in mesh repair methods mostly for the signing of volumetric representations. For example, in [Nooruddin and Turk 2003], an object is viewed in multiple directions and the patterns of ray-object intersection, defined either by parity count or simply by ray-stabbing, are used to determine which regions are inside the object. In space carving methods, distance fields are also signed according to the viewing directions of range scans [Curless and Levoy 1996; Furukawa et al. 2007]. Generalized winding number

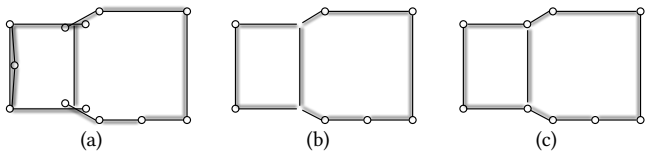


Fig. 2. 2D illustration of the repair pipeline. (a) depicts the input mesh whose faces are shown as line segments, with shadow showing backside of a face. The input mesh has flipped faces, redundant faces and self intersections. (b) shows the cleaned and reoriented face patches after the visual processing step (Sec. 4). (c) shows how the patches are further connected into large and coherent pieces by the manifold mesh reconstruction step (Sec. 5).

provides another way of signing volume [Jacobson et al. 2013; Zhou et al. 2016], as it integrates normal directions over the entire boundary surface. However, due to the reliance on normal directions, the generalized winding number cannot handle mis-orientations of input surfaces [Hu et al. 2018]. In contrast, the parity of collision is used by Bernstein and Wojtan [2013] to determine inside/outside when tracking the topological changes of interacting objects.

In our approach to the mesh repair problem, visual guidance is used extensively, to determine how mesh faces should be oriented, what parts are redundant (Sec. 4) and how different components should be connected for better visual regularity (Sec. 5).

Topology constrained integer optimization. Global mesh repair methods guarantee the output surfaces to be manifold, as a result of the conversion of the intermediate volumetric representation to the boundary representation. However, in our method, the manifoldness is guaranteed by breaking the input mesh into elementary manifold patches, and further searching the optimal connections among the patches inside the space of manifoldness, which is formulated as an ILP. Closely related is the problem of topology constrained surface reconstruction from basic surface patches, which can be defined by cross-sectional curves [Huang et al. 2017; Lazar et al. 2018; Zou et al. 2015] or detected by landmark triangulations [Huang et al. 2014]. Our enumeration of all manifold connections and the formulation as an ILP are inspired by [Lazar et al. 2018]; on the other hand, due to our particular task, we apply constraints on surface orientations rather than surface genus, and propose a novel rounding scheme that exploits the dual structure of the optimization problem and is highly efficient for finding close-to-optimal integer solutions (Sec. 5). Windheuser et al. [2011] formulate the nonrigid matching of two shapes as an ILP that searches in the product manifold of orientation preserving diffeomorphisms, and solve it by a sequence of relaxed LPs. In contrast, the ILP in this paper is solved efficiently by one relaxed LP and one global rounding.

3 METHOD OVERVIEW

We design a pipeline that translates the high visual quality of a 3D model into valid manifold structures. After a simple preprocessing of merging coinciding vertices and removing degenerate and duplicate faces, the pipeline consists of two major steps; see a 2D schematic illustration in Fig. 2 and a real example in Fig. 3. The first step focuses on visually measuring and cleaning the elementary patches of a mesh in a localized and feature-preserving manner (Sec. 4), to prepare them for the second step of manifold mesh reconstruction. To be specific, a sequence of operations measure the visibility and

Fig. 3. The visual driven global mesh repair pipeline applied on a bag model from ShapeNet. From left to right: (a) the input mesh, with its scattered manifold surface patches (Sec. 4.1) rendered in different colors; (b) after preprocessing that merges coinciding vertices and removes duplicate faces (Sec. 3); (c) after visual-driven processing (Sec. 4) that further reorients and merges scattered patches and removes visually redundant patches shown in (c̄); (d) after the global optimization of patch orientation and connection (Sec. 5), intersecting patches (e.g. around the rings) are connected into coherent and larger pieces; (e) the global optimization applied on the subset of patches excluding hidden ones shown in (ē), to enable more connections of different components (Sec. 6.2); (f) is (e) rendered with textures copied from the initial input, which are well preserved by our minimally intrusive approach. The number of manifold surface patches PN is shown for each mesh. For (a) and (b), we have used two-sided shading so that their many flipped faces can be shown.

orientation preferences of manifold patches in the mesh, reorient and merge them if possible, resolve intersections so that possible connections can be detected, and remove redundant patches which contribute nothing to the visual appearance of the whole model. Given these processed elementary manifold patches with possible connections and orientation preferences, the second step then tries to reorient and connect them into coherent and large manifold meshes through a global optimization (Sec. 5). This optimization problem is formulated as an ILP for which we find close-to-optimal solutions by relaxation to LP and efficient rounding. The finally extracted mesh is guaranteed to be manifold (Sec. 6) and preserves the original features and visual appearances.

Before going into details of the pipeline, we first introduce the notions required for subsequent presentation. A summary of important notations used throughout the paper is given in Table 2. In this paper we assume the input mesh has triangular faces, although the discussions can be easily extended to polygonal meshes.

We represent an input triangle mesh $M = (V, F)$ with a set of vertices $V = \{v_i\}$ and a set of triangle faces $F = \{f_i\}$, where each face $f_i = (v_{i0}, v_{i1}, v_{i2})$ is a tuple of vertices from V. Edges of the mesh can thus be collected as $E = \{e_i\}$, with each edge $e_i = \{v_{i0}, v_{i1}\}$ represented by its two endpoint vertices without order. The orientation of a face is encoded by the cyclic sequence of its vertices. An edge is *regular* if it has only one adjacent face or two adjacent faces that induce opposite directions for the shared edge, and *singular* otherwise; in addition, a singular edge adjacent to two faces with inconsistent orientations is called *orientation conflicting* (see Fig. 4). Let all the faces incident to a vertex v be v^*; we build a graph with these faces as nodes and a link between two nodes if their corresponding faces share an edge incident to v: the vertex v is called *regular* if the graph is a chain or cycle, and *singular* otherwise [Guéziec et al. 2001]. The mesh M is *manifold* if and only if it has no singular vertices.

Since our repairing pipeline works on the elementary manifold patches of a mesh, we also define the patches. A surface patch P of the mesh M is defined by a set of faces from F, i.e. $P = \{f_i \in F\}$. The patches are 2-chains in the language of chain complexes [Edelsbrunner and Harer 2010], based on which we can define the boundary curve of a patch P as the 1-chain $\partial P = \sum_i \partial f_i$, where

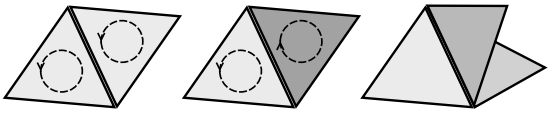


Fig. 4. Three types of edges. From left to right: a regular edge adjacent to two consistently oriented faces, an orientation conflicting edge adjacent to two faces with flipped orientations, and a singular edge adjacent to more than two faces.

$\partial f_i = a_{i0}e_{i0} + a_{i1}e_{i1} + a_{i2}e_{i2}$, with $a_{ij} = \pm 1$ denoting whether the face f_i induces positive or negative directions for its edges e_{ij}. The reference direction of an edge can be chosen arbitrarily but remains fixed. Note that while the patch boundary curve thus defined has coefficients for its constituent edges, in the following discussion for convenience we say an edge $e \in \partial P$ if e appears as a non-vanishing term in the curve 1-chain regardless of its specific coefficient.

The above definitions involve the topological structure of a mesh only. To give geometry to the mesh, each vertex v_i is assigned 3D coordinates $\mathbf{v}_i \in \mathbf{R}^3$. As preprocessing, we normalize an input mesh to make its bounding box centered at origin with a maximum side length equal to one. We then merge the almost coinciding vertices and remove combinatorial duplicates (two faces having the same set of vertices) and degeneracies (a face with less than three distinct vertices) by applying Alg. 1 with $\epsilon_m = 10^{-10}$.

4 VISUAL DRIVEN PROCESSING

The visual driven processing takes an input mesh and outputs a collection of manifold patches to be consumed by the global manifold mesh reconstruction step (Sec. 5). In this process, we apply a combination of three different operations:

(1) find the manifold patches, measure their preferred orientations through visibility counting and reorient, and merge them if possible (Sec. 4.1);
(2) measure the visual significance of each patch and remove unnecessary patches (Sec. 4.2);
(3) resolve self-intersections to detect connections (Sec. 4.3).

Each operation targets different problems with the man-made meshes. The first operation reorients the flipped patches that are visible to outside viewpoints and merge patches ready to be combined. The

Algorithm 1: Merge&Clean

Input: mesh $M = (V, F)$, coincidence threshold ϵ_m
Output: mesh $M' = (V', F')$
$V' = \{\}, F' = \{\}$;
for $i \leftarrow 1$ **to** $|V|$ **do**
  if $\exists v_j \in V', ||\mathbf{v}_i - \mathbf{v}_j|| \leq \epsilon_m$ **then**
    replace v_i with v_j for all f containing v_i;
  else
    $V' = V' \cup \{v_i\}$;
  end
end
for $i \leftarrow 1$ **to** $|F|$ **do**
  if $|\{v_{i0}, \cdots\}| < 3$ *or* $\exists f_j \in F', \{v_{i0}, \cdots\} = \{v_{j0}, \cdots\}$ **then**
    do nothing;
  else
    $F' = F' \cup \{f_i\}$;
  end
end

second operation removes visually redundant patches that cannot be found as face duplications. Finally, to handle a mesh model with self-intersections that frequently are intended to be meaningful connections, the third operation resolves the self-intersections to split mesh faces where necessary. The three operations are combined in a procedure summarized by Alg. 2. Next we present the details.

Algorithm 2: Visual driven mesh processing

Input: mesh M, significance threshold ϵ_{visual}
Output: updated mesh M with a collection of manifold patches and no redundancies
for $i \leftarrow 1$ **to** 2 **do**
  // Sec. 4.1
  detect all manifold surface patches $\{P_i \subset M\}$;
  compute patch orientation likelihood $S_+(P_i)$ and reorient;
  merge patches with consistent orientations ;
  // Sec. 4.2
  sample patches with points;
  compute visual significance score $\{VS(P_i)\}$ and visual affinity list L;
  while $\min\{VS(P_i)\} \leq \epsilon_{visual}$ **do**
    remove $P_{min} = \arg\min_{P_i}\{VS(P_i)\}$;
    update $VS(P_j)$ for $P_j \in L(P_{min})$ and L;
  end
  // Sec. 4.3
  resolve all self-intersections;
  if *no self-intersection* **then** break ;
  else Merge&Clean with $\epsilon_m = 0$;
end
apply the operation in Sec. 4.1 again;

4.1 Manifold patch detection, orientation and merging

Due to structural validity, orientable manifold surface patches are the basic components on which the subsequent visual driven processing and global optimization of orientation and connection are applied. An *orientable manifold patch* $P_i = \{f_j\}$ contains a set of faces connected by regular edges; singular edges adjacent to the faces are added to the patch boundary ∂P. We detect an orientable manifold patch by a flooding procedure that starts from a seed face and adds all neighboring faces crossing regular edges into the patch; all patches are detected by applying the flooding procedure on the remaining faces. Note that since singular edges can only appear at boundary, an unorientable surface like Möbius strip will be disconnected along orientation conflicting edges which become part of the boundary curve (see Fig. 7, can also be verified by the patch boundary computation (Sec. 3)); the handling of vertices on the singular edges is discussed in Sec. 6.1.

For two neighboring patches with shared boundary edges that are orientation conflicting, it is possible to combine the two patches by flipping the orientation of one patch. To resolve such cases, we first measure the orientation preferences of all surface patches using a visibility computation and reorient them accordingly, and then apply a greedy process to merge the patches as much as possible.

By observing the mesh model in various viewpoints, we can collect the chance of an exposed surface patch being positively oriented. In particular, for a surface patch P_i, let $S_+(P_i) \in [0, 1]$ be the likelihood of the patch orientation being aligned with visibility, and $S_-(P_i) = 1 - S_+(P_i)$ for being flipped against visibility. We compute the orientation likelihoods by counting positively and negatively exposed pixels, $Pix_+(P_i)$ and $Pix_-(P_i)$, for all patches:

(1) $Pix_+(P_i) = Pix_-(P_i) = 0$ for all i;
(2) with a customized shader of a standard graphics library, render the mesh model in $N = 42$ viewpoints $\{\mathbf{c}_i \in \mathbf{R}^3\}_{i=1,\cdots,N}$ evenly sampled on the bounding sphere of the model, to obtain N images of size 512×512, where each pixel records the patch it belongs to;
(3) for each image pixel under viewpoint $\mathbf{c}_i$ and its corresponding surface point $\mathbf{p}$ in patch P_j with normal $\mathbf{n}$, if $\langle \mathbf{n}, \mathbf{c}_i - \mathbf{p} \rangle > 0$, then $Pix_+(P_j) \leftarrow Pix_+(P_j) + 1$; otherwise if $\langle \mathbf{n}, \mathbf{c}_i - \mathbf{p} \rangle < 0$, then $Pix_-(P_j) \leftarrow Pix_-(P_j) + 1$;
(4) compute orientation likelihood as

$$S_+(P_i) = \begin{cases} \frac{Pix_+(P_i)}{Pix_+(P_i)+Pix_-(P_i)}, & \text{if } Pix_+(P_i) + Pix_-(P_i) > \epsilon_{pix} \\ 0.5, & \text{otherwise} \end{cases}$$

where $\epsilon_{pix} = 5$ is used to skip those patches with too few visible pixels to be informative. For each patch P_i, if $S_+(P_i) < 0.5$, it is flipped.

After each patch is reoriented according to visibility, we can use a simple flooding procedure to merge patches that previously were separated by *orientation conflicting* edges. The process starts from the patch P_i of largest surface area; for each neighboring patch P_j that is separated from P_i by orientation conflicting edges, if the two patches are now consistently oriented, they are merged. By applying the flooding procedure on remaining patches, all patches are merged to be as large as possible.

4.2 Visual significance scoring and iterated patch removal

Geometrically redundant but combinatorially different patches are prevalent in the man-made meshes but cannot be removed by preprocessing; see Fig. 3 (b) inset for such an example of two overlapping patches with different triangulations. The key property with such duplications is that after removing the redundant patches the overall mesh still looks almost the same. Similarly, the noisy small scale patches generated by modeling inaccuracies can also be safely removed without changing the mesh appearance. Based on this observation, we design a visual driven procedure for redundant patch removal.

We define the visual significance of a surface patch as the visual difference it introduces when it exists compared with when it is absent. To measure the visual differences, we first generate a set of sample points over each surface patch, through which we then render focused regions with and without the patch and compare the depth images for the quantified significance. Based on the visual significance scores, an iterated removal of the least significant patches is applied; see Alg. 2 for the process. Next, we focus on how to compute and update the visual significance scores.

The number of sample points generated for each patch depends on the patch area. In particular, for a patch P_i, we allocate $N_s(P_i) = \max\left(\frac{|P_i|}{diag^2} \cdot N_{tot}, N_{min}\right)$ points that are uniformly sampled on the patch surface. Here we use $N_{tot} = 1000, N_{min} = 20$ for a good coverage of all possible viewing directions, $|P_i|$ is the patch area, and $diag$ is the diagonal length of the mesh model's bounding box.

Through the densely sampled viewing points, we render the localized regions of the mesh model with and without each patch P_i being considered, to measure its visual significance $VS(P_i)$ by counting pixel differences, and to maintain a list L of visual affinity among the patches for fast updating of significance scores after each patch removal.

In particular, for each sampled viewpoint $\mathbf{c}_j$ with unit normal $\mathbf{n}_j$ on the patch P_i, by positioning the camera at $\{\mathbf{c}_j + \mathbf{n}_j, \mathbf{c}_j - \mathbf{n}_j\}$ and looking at the patch along normal, and for each camera position further switching the patch on and off, we render the model with orthogonal projection and a box viewing frustum, to obtain four depth maps $\mathbf{d}^+_{j,\text{on}}, \mathbf{d}^+_{j,\text{off}}, \mathbf{d}^-_{j,\text{on}}, \mathbf{d}^-_{j,\text{off}}$, and four patch id maps $\mathbf{id}^+_{j,\text{on}}, \mathbf{id}^+_{j,\text{off}}, \mathbf{id}^-_{j,\text{on}}, \mathbf{id}^-_{j,\text{off}}$ where each pixel records the visible patch it corresponds to, all of size 128×128. The box viewing frustum has the parameters $z_{near} = 1 - \epsilon_{cut}, z_{far} = 2.8$, with a clipping square of side length $\max(bbox(P_i), csl_{min})$, where $\epsilon_{cut} = 10^{-5}$ is used to accommodate depth inaccuracy in the rendering pipeline, $bbox(P_i)$ gives the side lengths of the bounding box of P_i so that we focus on how the current patch affects the visual appearance locally, and $csl_{min} = 0.08$ is a lower bound of field-of-view so that we do not zoom into very small scale patches that are probably noise due to modeling inaccuracies. To accelerate rendering, we only draw patches whose bounding boxes intersect the viewing frustum.

Given the rendered images, the visual significance score is computed as

$$VS(P_i) = \frac{\sum_{j=1}^{N_s(P_i)} \sum_{s\in\{+,-\}} \mathbf{m}^s_{j,\text{on}} \cdot \left(\mathbf{d}^s_{j,\text{off}} - \mathbf{d}^s_{j,\text{on}}\right)}{\sum_{j=1}^{N_s(P_i)} \sum_{s\in\{+,-\}} \left|\mathbf{m}^s_{j,\text{on}}\right|},$$

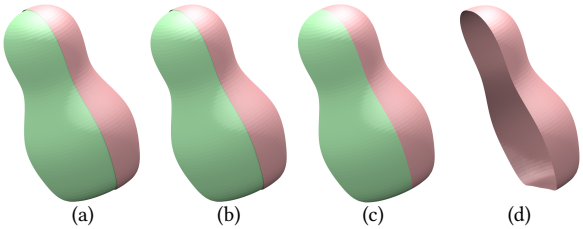


Fig. 5. Visual driven simplification of nesting doll-like models. (a) and (c) are the two input meshes, where the inner structure (green) in (a) has visible gap to the outer shell (pink, shown in half to reveal the inner structures), while in (c) there is no visible gap between the two layers. (b) and (d) are results after simplifying (a) and (c) respectively. (b) has two layers untouched. (d) only keeps the outer shell.

where $\mathbf{m}^{\pm}_{j,\text{on}} = \{\mathbf{id}^{\pm}_{j,\text{on}} = i\}$ are the binary masks for pixels covered by P_i. In case there is no pixel associated with the patch in all views, which can be due to the patch being entirely duplicated and hidden by other patches or it being too small to be visible, the visual significance score is set to zero.

After the scores are evaluated for all patches, we iteratively remove the least significant one and update the scores of other patches affected by the removal. The affected patches are given by the visual affinity list L: $P_j \in L(P_i)$ if and only if P_i appears in the locally rendered images of P_j when P_j is switched off, which implies that the removal of P_i can potentially change the significance score of P_j. The affinity list is built from the id maps; each time a patch P_i is removed, the existing patches in $L(P_i)$ are re-evaluated for visual significance and meanwhile their id maps lead to the update of corresponding affinity list entries.

The iterated patch removal process is continued until the least visual significance score is larger than ϵ_{visual}, which we set as 5.6×10^{-3} empirically for all experiments. The chosen threshold ensures the removal of truly redundant patches while preserving intricate but meaningful structures; see a test in Fig. 5. In addition, noticing that after removing a redundant patch, the remaining patches would only have increased significance, we skip updating patches with scores larger than ϵ_{visual} during the iterations for acceleration.

4.3 Self-intersection resolution

When creating the mesh models, people frequently place primitive components such that they have intersections and intend the components to be connected there. The self-intersections are defined as faces intersecting at places other than their shared edges or vertices; we detect such self-intersections and re-triangulate the faces involved, so that the components share conforming vertices and edges and can possibly be connected in subsequent steps. The robust detection and resolution of self-intersections of a mesh model rely on exact geometric kernels like rational numbers; in our implementation we use libigl [Jacobson et al. 2018; Zhou et al. 2016] which further uses [CGAL 2018] for exact constructions. Once the self-intersections are resolved, the mesh is processed to merge *exactly* coinciding vertices and remove combinatorially degenerate or duplicate faces (Alg. 1 with $\epsilon_m = 0$ applied to the rational number coordinates), and then fed to a second pass of visual processing to reorient, merge and simplify the newly generated patches (Alg. 2).

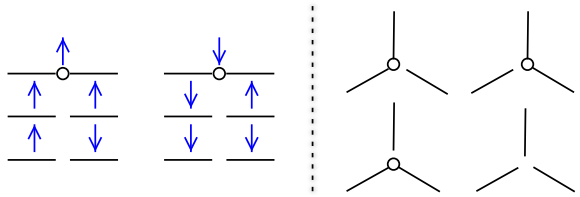


Fig. 6. A 2D illustration of the possible manifold configurations for singular curves of degree 2 (on the left) and 3 (on the right), where a segment denotes a patch, and a circle means a connection between two patches. For the degree 2 case, there are 6 configurations of connections; the blue arrows show the orientations of underlying patches. For the degree 3 case, there will be 20 configurations, 4 for each of the three cases with connected patches, and 8 for the lower right case with all three patches disconnected.

The reason for applying a first pass of the other two operations before resolving self-intersections, is to reduce early in the removal operation the occurrence of tangentially touching patches which are numerically unstable and may lead to overly fragmented intersection results. Simplifying meshes with very fragmented patches can be time consuming (Sec. 7.3).

5 MANIFOLD MESH RECONSTRUCTION THROUGH GLOBAL OPTIMIZATION

After the previous visual driven processing, we have a collection of non-redundant surface patches which are disconnected at singular edges. In this section, we present a global approach to reorient and connect the patches as much as possible, so that more complete and structurally meaningful manifold meshes are reconstructed from the scattered patches.

The manifold reconstruction problem is combinatorial in nature, so we formulate the global approach as an integer linear optimization, where the variables are the different connections at the patch separating curves, and the objective is to connect the patches as much as possible, while also maximizing the alignment with orientation likelihoods (Sec. 4.1), as well as the visual regularity of patches across connections. We present the problem setup in Sec. 5.1, the detailed formulation of optimization in Sec. 5.2, and show how to solve it efficiently with a novel rounding method in Sec. 5.3.

5.1 Enumerating and measuring the connections

The manifold patches are separated by polyline curves made of singular edges; the details for constructing these *singular curves* are given in Appendix A.1. For a singular curve C_i of degree N, denote the list of N patches sharing the curve as $Adj(C_i)$. Note that it is possible for a patch to occur multiple times in the adjacent patch list, which implies the patch connects to the curve in multiple ways.

Through a shared singular curve C_i, two patches $P_j, P_k \in Adj(C_i)$ can be reoriented consistently and connected into one larger patch: we call such a connection a *dual edge*, denoted as $de(P_j, P_k, C_i, o)$, where $o \in \{+, -\}$ signifies the orientation of P_j being preserved or flipped, and the orientation of P_k is induced accordingly. Note that when P_j and P_k are the same patch, the dual edge can be *invalid* by inducing conflicting orientations on the patch; we omit such invalid dual edges in computation. We additionally create a null patch P_{null} with arbitrary orientations and use $de(P_j, P_{null}, C_i, \pm)$ to denote the cases of P_j being disconnected at C_i.

The patches and the dual edges between patches form a *graph* $G = (\{P_i\}, \{de\})$. In addition, a manifold surface formed by the patches can be identified with the graph with a subset of dual edges. Therefore the manifold surface reconstruction problem can be regarded as a graph topology optimization problem, where a subset of graph edges are selected to maximize connectivity and visual regularity. However, in such a formulation extra care needs to be taken to ensure compatible dual edges are selected to imply manifoldness at the singular curves. To circumvent this problem, we change the variables to all *manifold configurations* instead.

For each singular curve, we can enumerate all possible configurations of manifold connections, or equivalently sets of dual edges, for its adjacent patches through the curve. For example, a degree 2 curve has 6 possible connection configurations, and a degree 3 curve has 20 configurations in total; see Fig. 6 for an illustration. The enumeration gives a list of configurations that grow exponentially with respect to the curve degree, though in practice curves with degrees larger than 4 are rare. In addition, a configuration may contain invalid or conflicting dual edges that induce opposite orientations for a patch; such self-conflicting configurations are discarded from consideration.

The connection of two patches induced by a dual edge can be evaluated for visual regularity, which is used subsequently for searching globally optimal connections among all patches. We define the visual regularity as a measurement of patch normal smoothness across the primal singular curve of the dual edge:

$$S_{reg}\left(de(P_i, P_j, C_k, o)\right) = \sum_{e \in C_k} w_e \left(1 - \frac{angle(P_i, P_j, e, o)}{\pi}\right),$$

where $e \in C_k$ is an edge in the singular curve, weight $w_e = \frac{|e|}{|C_k|}$ is the ratio of the edge length over the curve length, and $angle(P_i, P_j, e, o)$ is the angle between the normal vectors of the two patch faces incident to e, with the patch orientations induced by $o \in \{+, -\}$. Clearly, $S_{reg} \in [0, 1]$ and gets larger for patches with more aligned normal vectors at the singular curve, implying that they are more encouraged to be connected. When either patch of the dual edge is P_{null}, we set $S_{reg} = 0$ to discourage this disconnection.

In practice, we find that two patches may share a relatively tiny singular curve that is possibly noise due to modeling errors; it is undesirable for the patches to be connected through such noisy curves. To filter out such cases, if $\frac{|C_k|}{(|P_i|+|P_j|)^{1/2}} < \epsilon_{noise}$, we omit the dual edges $de(P_i, P_j, C_k, \pm)$. We set $\epsilon_{noise} = 0.03$ empirically.

Note that the visual regularity stands for a general notion of binary relationship between patches; as a result, other meaningful measurements like user preference or learned rules from annotated datasets can be used as well, which we leave for future work.

5.2 Manifold mesh reconstruction as an ILP

Let $[x_{i1}, x_{i2}, \ldots]$ be the indicator vector for selecting one configuration from all enumerated choices for a singular curve C_i; thus each x_{ij} corresponds to a set of dual edges. Further, a singular curve $C_j \subset \partial P_i$ if for all edges $e \in C_j$, there is $e \in \partial P_i$ (Sec. 3). Next we

present the constraints and the objective function for modeling the manifold mesh reconstruction problem as an ILP.

Selection constraint. We require a single configuration is selected for each singular curve, which means

$$\sum_j x_{ij} = 1, \forall i \tag{1a}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \tag{1b}$$

Orientation constraint. Orientation consistency requires that the selected orientation of a patch P_i is compatible with the selected configurations of all its boundary singular curves. Thus for the positive and negative orientations respectively, we have a chain of equations iterating over the boundary singular curves:

$$\begin{aligned} \sum_{x_{jl} \to +P_i} x_{jl} &= \sum_{x_{km} \to +P_i} x_{km}, \\ \sum_{x_{jn} \to -P_i} x_{jn} &= \sum_{x_{kp} \to -P_i} x_{kp}, \\ &\forall C_j, C_k \subset \partial P_i \end{aligned} \tag{2}$$

where $x \to +P_i$ means that the dual edges corresponding to x induce positive orientation of P_i, while $x \to -P_i$ induces negative (flipped) orientation. Note that $\{x_{jl}\} \cap \{x_{jn}\} = \varnothing$ because we discard self-conflicting configurations during enumeration.

Objective function. The linear objective function consists of two terms: the first term works on each patch, and measures the alignment of its orientation to visibility; the second term works on each dual edge, and measures the visual regularity of the induced connection. To be specific, we have

$$E(x) = \omega_1 E_{ori}(x) + \omega_2 E_{reg}(x). \tag{3}$$

The first term contains two symmetric parts, for positive and negative orientations respectively:

$$\begin{aligned} E_{ori}(x) &= \sum_{P_i} E^+_{ori}(P_i, x) + E^-_{ori}(P_i, x) \\ &= \frac{1}{\sum |P_j|} \sum_{P_i} \frac{|P_i| S_+(P_i)}{\sum_{C_k \subset \partial P_i} 1} \sum_{C_j \subset \partial P_i} \sum_{x_{jk} \to +P_i} x_{jk} \\ &\quad + \frac{1}{\sum |P_j|} \sum_{P_i} \frac{|P_i| S_-(P_i)}{\sum_{C_k \subset \partial P_i} 1} \sum_{C_j \subset \partial P_i} \sum_{x_{jl} \to -P_i} x_{jl}, \end{aligned}$$

where $S_\pm(P_i)$ give the likelihood of patch orientation being correct or flipped (Sec. 4.1). Intuitively, to maximize E_{ori} we should select the configurations that induce all the patch orientations with larger likelihood. Numerically, $E_{ori} \in [0, 1]$, because: 1) $\sum_{x_{jk} \to +P_i} x_{jk} \in \{0, 1\}$ due to (1a) and (1b), 2) the normalizations by curve numbers and patch areas, and 3) $0 \le S_\pm(P_i) \le 1$.

The second term simply sums up the visual regularity of all dual edges:

$$E_{reg}(x) = \sum_{de} E_{reg}(de, x) = \frac{1}{\sum |de^*|} \sum_{de} |de^*| S_{reg}(de) \sum_{x \to de} x,$$

where de^* denotes the primal singular curve of the dual edge, and $x \to de$ means that the configuration x contains the dual edge de. Note that due to the non-negativity of regularity scores S_{reg}, by maximizing E_{reg} we would select configurations that try to connect all patches as much as possible, and prioritize visually more regular connections over less regular ones. Numerically, $E_{reg} \in [0, 1]$, similarly because of the mutual exclusivity of configurations containing a dual edge, the normalization by singular curve lengths, and that $0 \le S_{reg} \le 1$.

Since E_{ori} and E_{reg} are numerically comparable, in all our experiments, we have used $\omega_1 = 5, \omega_2 = 1$ to put an emphasis on the visibility alignment of patches.

The ILP problem. We formulate the global manifold mesh reconstruction problem as an integer linear programming:

$$\begin{aligned} \max_x \ & E(x) \\ \text{s.t.} \ & (1), (2) \end{aligned}$$

The problem is guaranteed to have feasible solutions, because the trivial solution of disconnecting all manifold patches from singular curves always exists.

An ILP problem is NP-complete and therefore hard to solve for its optimal solution within a tractable computational budget. However, approximate solutions of high quality for many practical ILP problems can be found efficiently using carefully designed algorithms. Next we present such an algorithm for solving the manifold mesh reconstruction problem that exploits the duality of patch orientations and connections.

5.3 Solving the ILP

We proceed in two steps:

(1) The ILP is relaxed into a linear programming (LP), by replacing (1b) with $0 \le x_{ij} \le 1$. Let x^*_r be its optimal solution.
(2) Extract the integer solution x^*_b from x^*_r, using an efficient rounding scheme that first determines patch orientations and then selects connections.

The LP problem can be efficiently solved by standard solvers; we have used MOSEK [ApS 2018] in our experiments. We present details of the integer solution rounding method next.

Converting x^*_r to integer solutions is not straightforward, because of the constraints (1a) and (2). For example, a naive rounding of selecting the maximum x_{ij} to be 1 for each singular curve C_i would easily violate the orientation constraints. To circumvent this difficulty, we first round the orientation constraints by selecting for each patch the most likely orientation given the current connection probabilities x^*_r using a graph labeling procedure, and then choose the maximum from configurations that are compatible with the selected patch orientations.

The patch orientation selection can be viewed as a binary graph labeling problem on $G = (\{P_i\}, \{de\})$. In particular, we find the orientation $o_i \in \{+, -\}$ for P_i, such that the objective function adapted from (3) is maximized:

$$E_{graph}(o) = \omega_1 \sum_i E_{g\_ori}(P_i, o_i) + \omega_2 \sum_{i \le j} E_{g\_reg}(P_i, P_j, o_i, o_j),$$

where the unary term is

$$E_{g\_ori}(P_i, o_i) = \begin{cases} E^+_{ori}(P_i, x^*_r), & \text{if } o_i = + \\ E^-_{ori}(P_i, x^*_r), & \text{if } o_i = - \end{cases}$$

and the binary term is

$$E_{g\_reg}(P_i, P_j, o_i, o_j) = \sum_{de \to o_i, o_j} E_{reg}(de, x_r^*),$$

where $de \to o_i, o_j$ means that the dual edge de implies the orientations o_i, o_j for the patches P_i, P_j respectively. Notice that for the special case of $i = j$, meaning the patch connects to itself through singular curves, the binary term degenerates to a unary term.

Essentially, E_{graph} is a rearrangement of the terms in $E(x_r^*)$, depending on the discrete values of the patch orientations. Thus the orientations selected by maximizing E_{graph} are the optimal ones, given the optimal continuous connectivity likelihoods x_r^*. After obtaining the optimal orientations o^*, to determine the final connections, we only need to choose the configuration with $\max\{x_{r,ij}^* | x_{r,ij}^* \to o_k^*, \forall P_k \in Adj(C_i)\}$ for each curve C_i. As a result of this two-step process, the configurations selected are guaranteed to satisfy the constraints and therefore feasible.

The binary graph labeling problem can be solved efficiently with polynomial complexity algorithms under mild regularity conditions [Kolmogorov and Zabin 2004]. In our experiments, we have used [Kolmogorov 2006] to solve it.

In Sec. 7.3 we compare the two-step global optimization method with the alternative Branch-and-Bound and greedy solvers, and show empirical evidence of its effectiveness.

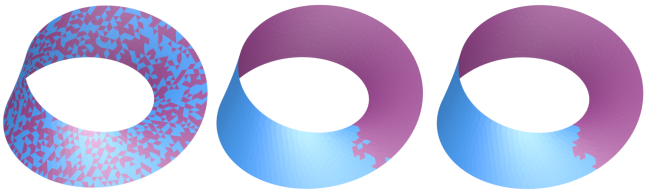
6 POSTPROCESSING

6.1 Extracting mesh from the solution

The solved optimal configurations encode the orientations of all faces and the regular adjacency relationships for faces that are incident to singular edges. After reorienting the faces as specified, we are only one step away from having the final well-connected manifold mesh. In this section we present the procedure to extract the output manifold mesh based on the solved adjacencies. The output manifoldness is proved here and empirically tested in Sec. 7.1.

We first apply a simple postprocessing to ensure successful manifold mesh extraction. We identify the singular curves which contain only single edges. For each single edge, we split the edge in the middle by inserting a new vertex; the adjacent faces incident to the edge are split into two as well. The connection solved for the single-edge curve is applied for the two newly created singular edges. The effect of this process is illustrated in Figs. 8 and 9, where the resulting edges of input one-edge singular curves may not have been encoded by their endpoint vertices if the singular curves were not split. This construction will also be utilized in the proof of result manifoldness in the following discussion.

The procedure to extract the manifold mesh is mainly about duplicating each singular vertex such that every manifold subgroup of its incident faces has a separate copy, thus satisfying the definition of manifoldness (Sec. 3). Let $M = (V, F)$ be the input mesh processed thus far, and $M' = (V', F')$ be the output manifold mesh to be extracted. We use the binary relation $\sim_E = \{\{f_i, f_j\}\}$ to denote all pairs of faces $f_i, f_j \in F$ that are adjacent because either their shared edge is regular in the input mesh or they are dictated so by the solved configuration; if a face f_i is adjacent to no face through its k-th edge, we abuse notation slightly and store $\{f_i, (k, null)\}$ in $\sim_E$.



(a) (b) (c)

Fig. 7. An unorientable Möbius strip. From left to right: (a) the input Möbius strip with faces randomly flipped (front in blue, back in purple), (b) result after visual processing that tries to reorient the faces into larger patches with coherent orientations, and (c) result after global manifold mesh reconstruction which further flips certain small patches to obtain a single oriented surface with a boundary curve separating the opposite facing triangles.

For a vertex $v \in V$ and its star of incident faces v^*, let $f_i, f_j \in v^*$ be two faces. We say $f_i \sim f_j$ if there exists a sequence of faces $f_i, \cdots, f_j \in v^*$ such that two consecutive faces in the sequence are adjacent by $\sim_E$; the relation is an equivalence that partitions v^* into $v^*/\sim$. We duplicate the vertex v into $|v^*/\sim|$ copies, and let faces in each equivalence class index their corresponding copy. The extended vertex set and the faces with possibly updated vertex references constitute the final output mesh.

Theorem. *The extracted mesh is manifold.*

Proof. See Appendix A.2. □

6.2 Optional hidden pieces removal

By solving the manifold mesh reconstruction problem, we are able to obtain large connected pieces from the scattered input patches, which generally correspond to meaningful decomposition or segmentation of the 3D models. However, for two components that are placed in intersecting positions, they may still be disconnected after the optimization, due to the fact that there are hidden marginal pieces that prevent the connection of the two major components. Depending on applications, we may regard the hidden marginal pieces as desirable or extraneous; if they are deemed irrelevant, a simple fix is to remove the hidden patches and their incident dual edges from the graph $G = (\{P_i\}, \{de\})$, on which the optimization is then applied. As discussed in Sec. 4.1, we define a patch as hidden if its visible pixels are less than ϵ_{pix}. For example, in Fig. 3 the removal of hidden pieces (ē) allows the attachments to be further connected with the bag handles, resulting in a more concise model. More examples without and with hidden pieces removed are shown in Figs. 1, 11, 13, 16 and 21.

7 RESULTS AND DISCUSSION

This section presents the results of processing meshes of diverse complexities with the proposed mesh repairing tool. We also analyze the computational performance of each step, demonstrating high efficiency of our novel ILP solver. Comparison with previous mesh fixing methods is conducted to show that our tool better fits the targeted man-made meshes and complements existing approaches. The sensitivity and impact of parameters are also discussed.

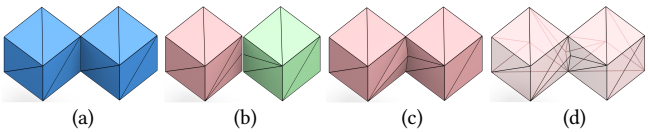


Fig. 8. Double cubes touching at a common edge, as shown in (a). Under our energy formulation, there are two equally optimal manifold solutions, (b) and (c). We slightly translated the two separate cubes in (b), and perturbed the geometrically coinciding vertices in (c) to better visualize the topology. (d) is a transparent rendering of (c) to show the connections clearly. Note that the one-edge singular curve in (a) is split by our post-processing (Sec. 6.1).

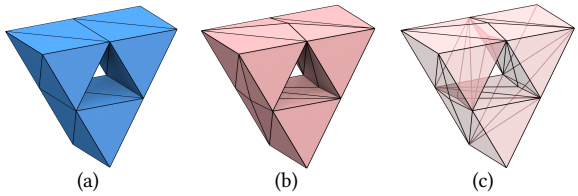


Fig. 9. Three prisms touching at three common edges. (a) the input nonmanifold mesh. (b) the result manifold mesh, with coinciding vertices slightly perturbed to better show the topology. (c) is (b) rendered in transparency to show the connections clearly. Note the three one-edge singular curves have been split by our post-processing (Sec. 6.1).

7.1 Validation

In this section we validate the presented algorithm with challenging synthetic cases. As shown in Fig. 7, our algorithm can handle an unorientable Möbius strip without problem. Given an input Möbius strip whose faces have been randomly flipped, the first step of our pipeline reorients the faces into improved coherency and merges them into larger patches (Fig. 7(b)), and the second step of global optimization further flips the three isolated small patches and connects all patches into a single surface. The final patch is oriented manifold as it differs from a Möbius strip with a boundary curve that cuts across the strip (Fig. 7(c)).

For two cubes touching at a common singular edge shown in Fig. 8, our manifold mesh reconstruction through global optimization actually has two equally optimal solutions, as they have the same orientation alignments and connection regularity scores (Eq. 3). One solution separates the two cubes, shown in Fig. 8(b). The other solution turns the shared edge into a tunnel, shown in Fig. 8(c) and (d) where the geometrically coinciding vertices on the tunnel edges are perturbed manually to show the structure. Note these vertices are created by the postprocessing (Sec. 6.1); without them the tunnel edges would not be distinguishable by endpoint vertices.

A similar case to the two cubes is shown in Fig. 9, where three prisms touch each other through singular edges. There is only one optimal solution for this case, with the three singular edges all turned into tunnels. Again the vertices on tunnel edges are perturbed in Fig. 9(b) and (c) to show the structure. All these test cases demonstrate the guaranteed output manifoldness of our algorithm.

7.2 Processing large 3D datasets

The proposed mesh repair method is applied on significant portions of the popular large scale datasets of ModelNet [Wu et al. 2015] and ShapeNet [Chang et al. 2015]. In particular, by setting a maximum

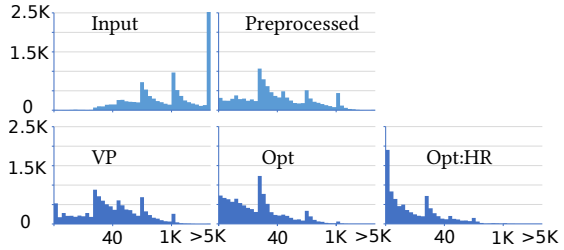


Fig. 10. Counting the number of patches on 10k processed meshes from ModelNet and ShapeNet. The horizontal axis denotes the number of surface patches and is piecewise linear. VP means the results after our visual driven processing step, Opt means results after global optimization, and Opt:HR means results when hidden patches are removed. Clearly the surface patches are greatly reduced by our tool, signifying more structured result meshes.

running time limit of 10 hours per model, our fully automatic tool has processed 4891 out of 4899 models from ModelNet10 (the other 8 models shown in supplemental material exceeded time limit), and 5536 out of 55341 models from ShapeNetCore, thus around 10k in total. Our approach is robust enough to work on all these inputs without failures. However, for complex models with large amounts of surface patches, the running time for the redundancy removal operation (Sec. 4.2) of the visual driven processing step can be quite long; see Sec. 7.3 for more discussions. The models before and after processing are counted for their numbers of manifold patches, which is summarized in Fig. 10. Example models are shown in Figs. 1, 11, 12, 13, 15 and 21; several mesh files can be found in the supplemental material.

The benefits of repairing the meshes with our method can be shown in immediate downstream applications. For example, on the large connected manifold patches, we can safely apply remeshing tools that work by iterated edge split/collapse and vertex smoothing, while also preserving sharp features of the input models. As shown in Fig. 12, we detect the feature edges of the input models by using a simple threshold on the dihedral angles, and apply the anisotropic remeshing tool by [Fu et al. 2014] on our fixed meshes to obtain these high quality and feature preserving meshes. On the other hand, remeshing directly on the input meshes with either scattered triangles or nonmanifold edges can be quite challenging. Many advanced downstream deep learning methods that require or benefit from manifold meshes can then be applied on these meshes of high quality, including [Maron et al. 2017; Masci et al. 2015; Poulenard and Ovsjanikov 2018; Qi et al. 2017]. Manual labeling of semantic parts requires manifold components as well [Mo et al. 2019].

As another example, due to the minimal intrusion nature of our algorithm, the processed models naturally preserve the textures of input models: for faces that exist in the input, the texture maps are simply carried over; for the newly generated faces by resolution of self-intersection and singular edge subdivision, their texture maps are deduced from the texture coordinates of their embedding faces in the input by barycentric interpolation. Figs. 3 (f), 13 show textured results. On these textured meshes with proper orientations and no redundant faces, it is more robust to compute ray-triangle intersection for ray tracing and other advanced rendering. In addition, it is noted that only on the meaningfully connected components

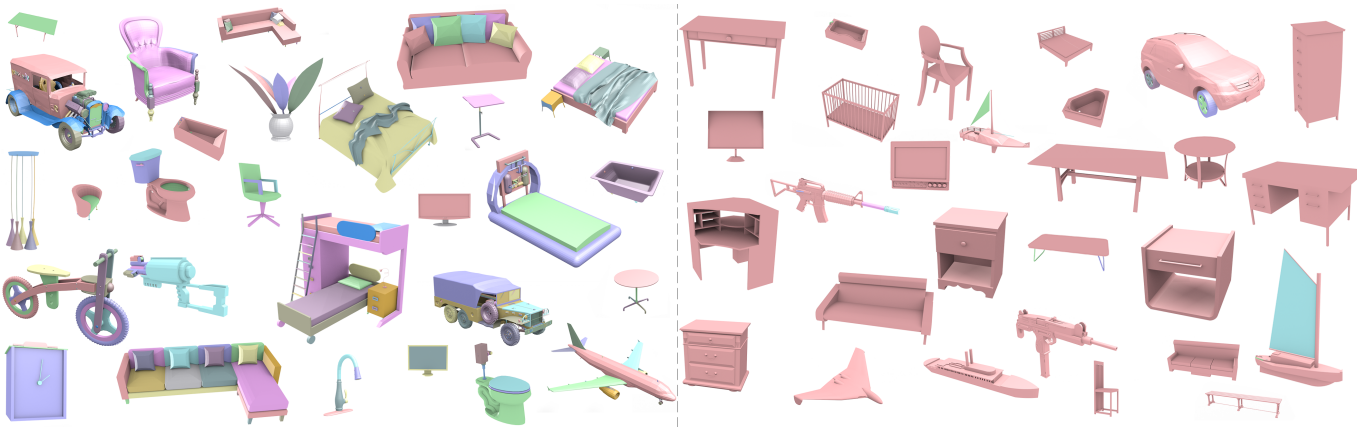


Fig. 11. More results by processing ModelNet and ShapeNet datasets. The left side shows results without hidden patch removal, and the right side more concise models with hidden patch removal.

Fig. 12. Remeshing on the repaired models. For each pair, the left is the input model, and the right is our fixed model with each patch further remeshed by local operations like edge flipping and vertex smoothing [Fu et al. 2014]. Hidden patches are not removed here. Note that sharp curves on the input models are detected as feature lines and well preserved by remeshing.

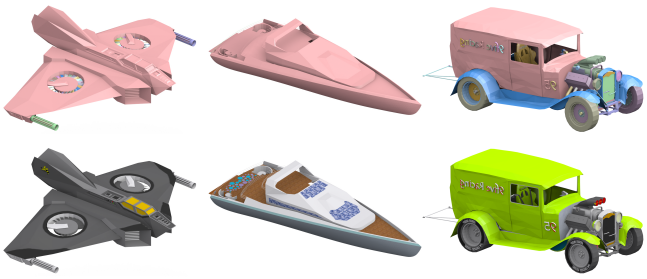


Fig. 13. Texturing the repaired meshes. Top row shows the repaired meshes that have large manifold meshes with all sorts of delicate structures preserved. Bottom row are the rendered meshes with textures copied from input meshes. Note how the textures are well transferred. The left two meshes are results with hidden pieces removed, and the right one without.

can new texture maps be properly applied for advanced appearance synthesis applications as demonstrated in [Park et al. 2018].

7.3 Computational performance

In this section, we focus on the computational performances of the proposed pipeline, by measuring on randomly selected models from ModelNet and ShapeNet. We also evaluate the efficiency of our approximate global optimization solver by comparing with alternative methods. All the evaluations are done on a desktop PC with Intel i5-3570 processor and integrated GPU.

7.3.1 Runtime cost. The simple preprocessing is fast, taking seconds even for large models. The time used by visual driven processing is positively related to the number of manifold patches; in particular, the first and third operations (Sec. 4) are efficient non-repeating passes over the basic patches, while the patch removal operation is an iterative process that has no upper bound on its running time. Therefore in Fig. 14, we sort a set of 30 test meshes according to the number of patches generated after preprocessing and self-intersection resolution, and plot their time costs in the visual processing step. We can see that this step is indeed time consuming: for very complex models it can take hours. Two of the most complex models are shown in Fig. 15: we see that in addition to the patch orientations being adjusted, a large number of cluttered and redundant patches have been removed during the simplification step, while the delicate but meaningful patches are well preserved and finally connected into clean components that strongly correspond to semantics.

For the 8 models of ModelNet that are not finished within 10 hours, we notice even larger numbers of patches and more convolved spatial relations among them (see Fig. 16). The overly cluttered structures with these cases, however, suggest that we may take an aggressive approach of simplification, i.e., we can apply the hidden patch removal (Sec. 6.2) early before the iterated removal operation in Alg. 2, so that a large number of these hidden and cluttered patches can be quickly removed. Indeed, using this aggressive strategy, all the eight models can be finished within 50 minutes. As shown in Fig. 16 (c) (d) for the chair model, the result is a clean mesh which may have lost some subtle inner structures alongside the cluttered redundancies.

The global optimization step is generally very efficient, taking around one second for the most complex models. As shown in

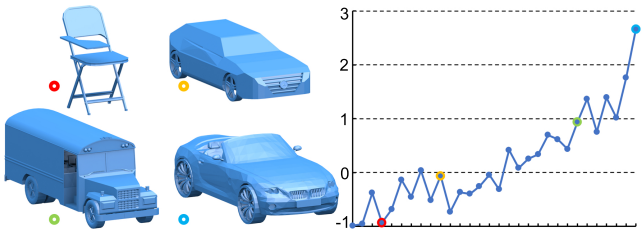


Fig. 14. Runtime plot of the visual driven processing step. The horizontal axis marks 30 test cases sorted by the number of patches. The vertical axis is the logarithm of minutes used. Test samples of various complexity are shown on the left: for the more detailed models, a large number of redundant patches are detected and removed during simplification (see Fig. 15).

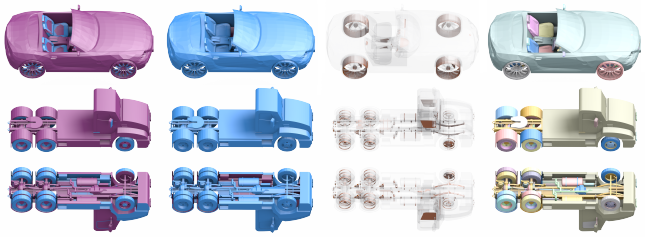


Fig. 15. Two test models with long visual processing running time. From left to right: the input mesh, the mesh after visual processing, the redundant pieces removed by simplification, and the final optimized result. Face backside is rendered in purple. After the processing, the number of patches have been reduced from 6618 to 323 and 3802 to 715 for the car and truck models, taking 407 and 51 minutes, respectively.

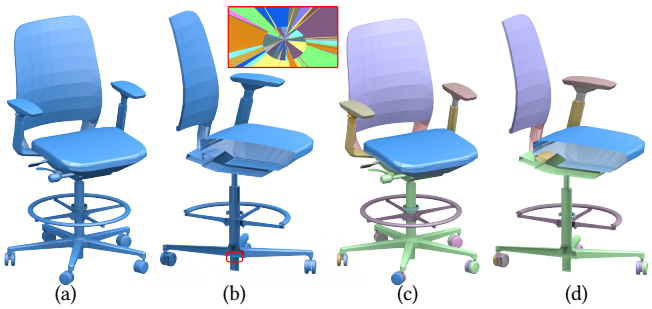


Fig. 16. The timed-out chair model has very fragmented patches, but is finished within 3 minutes using the aggressive simplification strategy that removes hidden patches first. (a) and (b) show the input mesh and its cutoff view, where highlighted is a small region which contains more than 84k cluttered patches (shown in different colors) after self-intersection resolution. (c) and (d) show the result and cutoff view by using aggressive simplification.

Fig. 17 (a), the time cost for global optimization is mainly influenced by the number of variables.

7.3.2 Efficiency of the approximate ILP solver. We evaluate our two-step global solver for reconstructing manifold meshes from scattered patches (Sec. 5.3) and compare it against alternative methods, i.e. the global optimal solvers (Branch-and-Bound (BnB)) and the greedy solver (Appendix A.3), extensively on 1k test models. We have considered two variants of the BnB solver, one with our rounding that guarantees solution feasibility and the other with naive maximal rounding. Thus the "BnB+our rounding" solver gives the upper

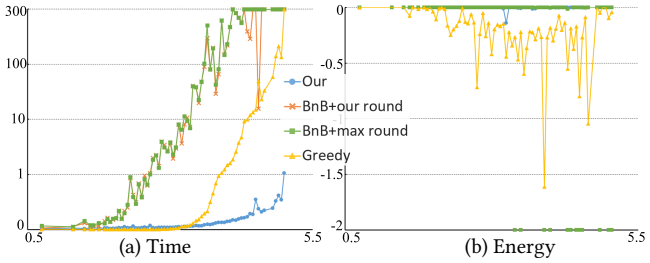


Fig. 17. Time and energy plot of four optimization algorithms for solving the manifold mesh reconstruction problem. The horizontal axes give the log scale of variable numbers for the test cases. In the time plot (a), the vertical axis is piecewise linear with second as the unit; a limit of 300 seconds is imposed. The energy plot (b) shows energy differences (scaled by ×100) from the upper bounds obtained by "BnB+our round"; the data points for "BnB+max round" on horizontal axis imply failure to find feasible solutions. In contrast, our solutions tightly approximate the upper bounds.

Table 1. The performance of four ILP solvers on representative test cases. #var is the number of variables, E_1/t_1, E_2/t_2 and E_{gr}/t_{gr} resp. give the optimized energy and time cost (in seconds) of "BnB+our rounding", "BnB+maximal rounding" and the greedy solver. For the 946 and 983 samples, BnB+maximal rounding fails to find feasible solutions within 300 seconds.

id	#var	E_{our}	t_{our}	E_1	t_1	E_2	t_2	E_{gr}	t_{gr}
34	40	4.94548	0.02	4.94548	0.05	4.94548	0.06	4.92471	0.01
505	1004	5.07029	0.02	5.07029	0.55	5.07029	0.56	5.06554	0.02
692	2192	3.07301	0.07	3.07301	24.9	3.07301	37.9	3.06658	0.08
918	7888	3.51262	0.14	3.51262	53.1	3.51262	300	3.50518	1.38
946	10434	3.65282	0.15	3.65333	300	-	300	3.646	2.58
983	15164	2.71965	0.16	2.71987	300	-	300	2.71852	5.28

bound energy values for all four methods. Time costs and energy differences from the upper bounds are plotted in Fig. 17.

Overall, our solver is capable of handling all test cases and finding close-to-optimal solutions fast. This is in contrast to the global optimal solvers whose time cost quickly grows as the variable number increases (Fig. 17(a)), and to the greedy solver which generally finds less optimal solutions (Fig. 17(b)) and takes longer time for moderately complex models. These are also confirmed by numeric data (Table 1) and visualization (Fig. 18) for concrete examples. Additionally, it is worth noting that compared to our rounding with guaranteed feasibility, "BnB+maximal rounding" fails to find feasible solutions for many big models (946 and 983 in Table 1 and exceptional data points in Fig. 17(b)).

7.4 Comparison

We compare our approach with two types of previous methods: the first type contains the global repair methods that rely on volumetric intermediate representations, in particular, PolyMender [Ju 2004] and TetWild [Hu et al. 2018]; the second type represents methods that work directly with polygonal meshes, including MeshFix [Attene 2010] and outer hulls [Attene 2016]. Notice that as reviewed before, these comparing methods do target differently than our method, as they all assume that the input mesh bounds a valid solid volume, which is however frequently not the case for meshes in the popular datasets. Nonetheless, we try to compare with them

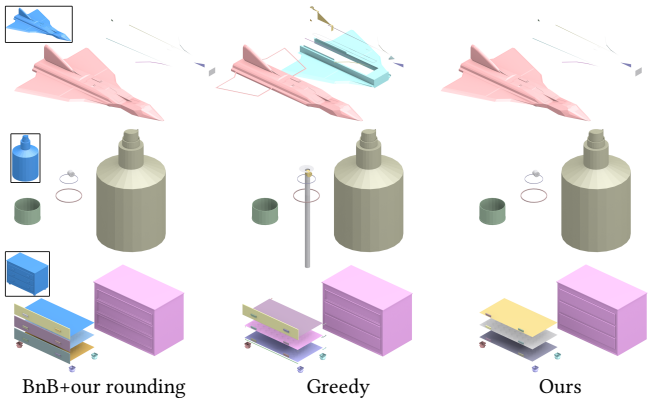


Fig. 18. Result manifold components by different solvers without hidden patch removal. The input of each test case is shown at upper left corner. The components are rearranged manually to better show the structures. Both BnB and our results are structurally regular and similar, while greedy solver results are more scattered.

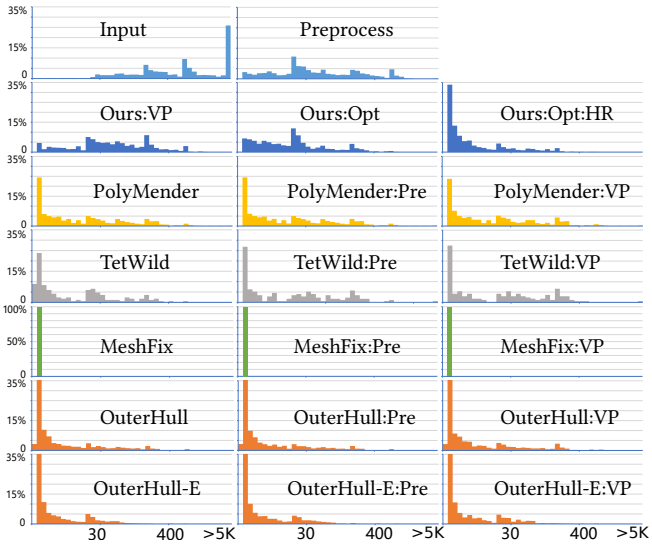


Fig. 19. The distributions of patch numbers for comparing mesh repair methods. For our method, VP stands for results after visual processing, Opt means after global optimization, HR further means the hidden pieces are removed. For other methods, for each test case the method is applied to the initial input, the preprocessed mesh and the visually processed mesh, respectively. Note the horizontal axis is piecewise linear.

on processing a set of 1092 randomly selected models from ModelNet and ShapeNet, to discuss the strengths and weaknesses of each method. The distributions in terms of patch numbers and Hausdorff distances from inputs are measured on the test set and shown in Fig. 19 and Fig. 20: in general, the patch numbers are greatly reduced after our processing while the results remain very close to the input meshes. Specific examples are further visualized in Fig. 21. Overall, our method is shown to be more suitable for repairing these meshes, as it robustly removes redundancies and conflicts and

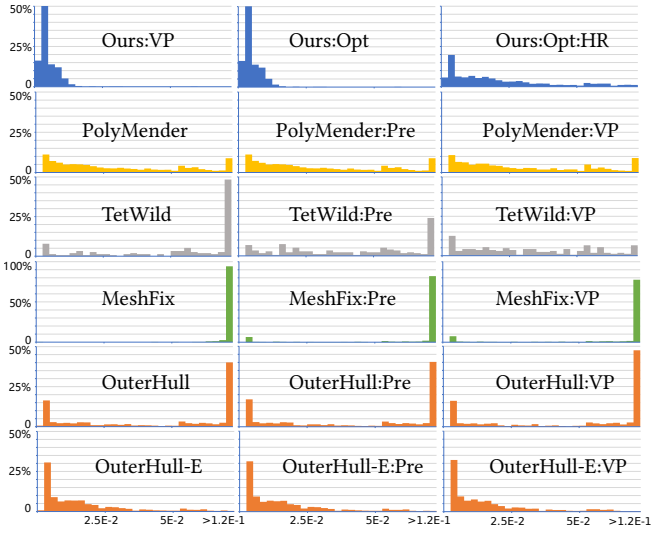


Fig. 20. The Hausdorff distance distributions of result meshes from input for comparing repair methods. See caption of Fig. 19 for explanation of notations. Note the horizontal axis is piecewise linear.

recovers manifold meshes that preserve the input structures with minimal intrusion. Next we compare with each method in detail.

With PolyMender [Ju 2004]. PolyMender takes a global repair approach by scan converting the input mesh into a volumetric grid representation first, and then sign the grid points consistently by patching singular boundaries to obtain a solid, from which a dual contouring surface mesh is finally extracted. The approach has the advantage of being very robust and generic, as it can handle meshes with many gaps and randomly flipped faces. On the other hand, an intrinsic problem with the volumetric grid conversion is the loss of original mesh structures and features.

For example, as shown in Fig. 21 (a1), there is strong aliasing of sharp features at the airplane wing edges due to the regular sampling by volumetric grids. In addition, because of limited grid resolution, small scale or thin features that are critical may not be properly captured by the volumetric representation, as shown through the zoomed-in wing tips in (a1) that are severely distorted, and the drawer handle rings in (a2) that are mostly lost. What's more, PolyMender fails to sign the volume properly to recover proper surfaces for the front side of table in (a2), because there are multiple layers of intricate inner structures that interfere with the signing process. In contrast, our approach of working on the meshes directly faithfully preserves all these important structures and features and robustly handles complex structures. The difference is also shown in the contrasting errors measured by Hausdorff distance from input meshes between PolyMender's results and ours (Fig. 20): our errors are on average an order of magnitude smaller than theirs.

With TetWild [Hu et al. 2018]. TetWild is a robust tetrahedral meshing method that can handle all kinds of input boundary meshes and generate high quality and feature sensitive tetrahedral meshes that conform to the boundary within a given threshold. From the mesh repair perspective, TetWild can also be regarded as a global

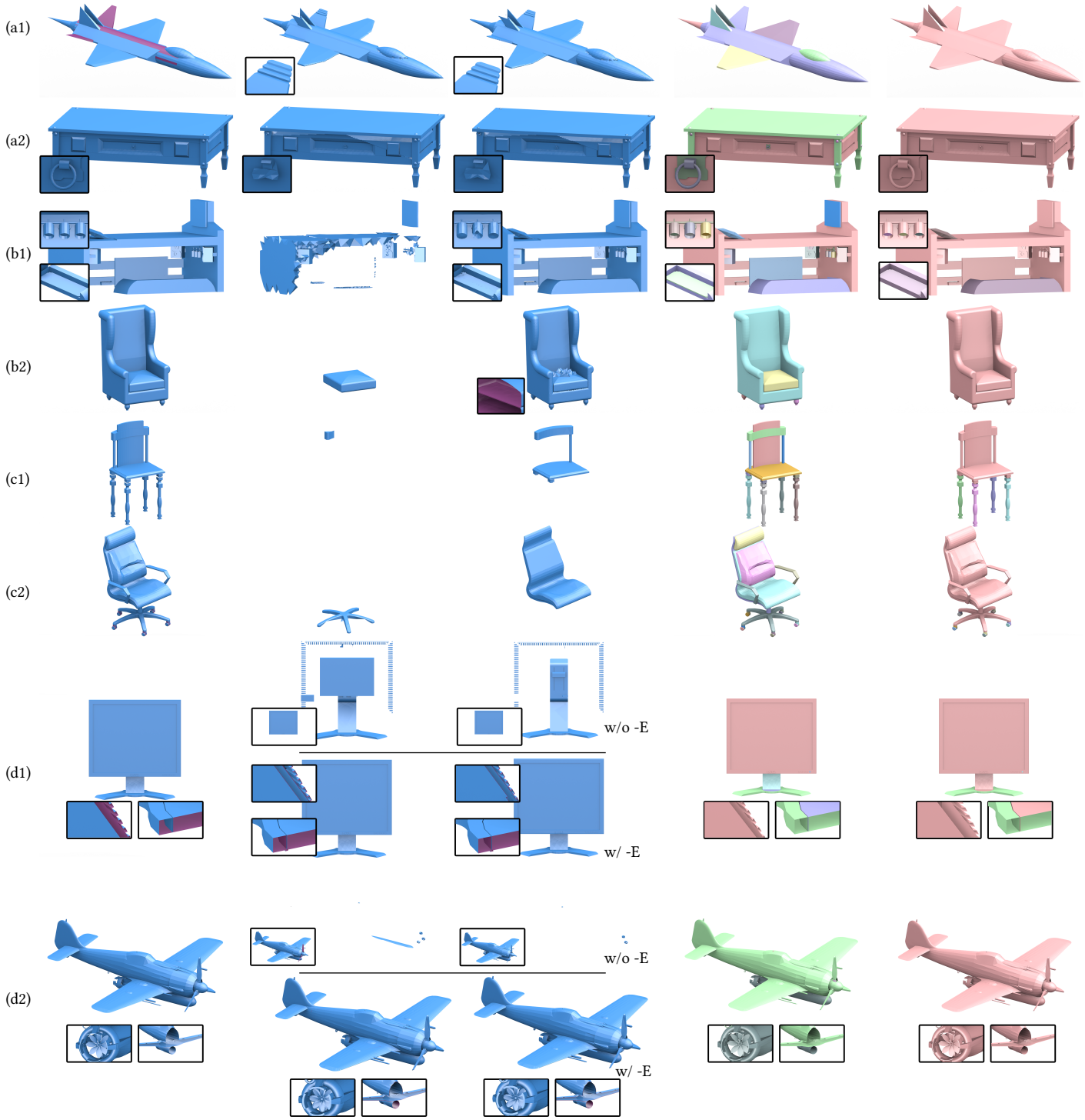
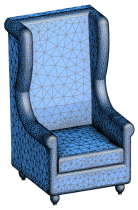


Fig. 21. Comparison with other methods through examples. For meshes other than our results, front sides are rendered in blue and back sides in purple. (a) compared with PolyMender [Ju 2004]; (b) with TetWild [Hu et al. 2018]; (c) with MeshFix [Attene 2010]; (d) with outer hulls by [Attene 2016], without and with its dangling surface extrusion option (-E), where the insets show the dangling surfaces and cutaway views of result models, respectively. In each row, from left to right, are the input mesh, the result of comparing method on the input, the result on the visually processed mesh by our first step, our result, and our result with hidden pieces removed from input. Zoom in to see the structure differences, and refer to text for detailed analysis. The mesh models are included in supplemental materials.

repair method that uses the tetrahedral mesh as the intermediate representation, from which the boundary surface is extracted by signing the tetrahedra using generalized winding numbers [Jacobson et al. 2013]. The advantage of TetWild is again its unconditional stability in generating tetrahedral meshes from arbitrary input meshes. Yet when used for repairing and extracting surface meshes, the signing method relies heavily on the assumption that the surface should be properly oriented and bound a valid solid, which is hardly the case for many meshes in our targeted datasets.

Indeed, as shown in Fig. 21 (b1), since there are hidden duplicated patches with flipped orientations in the input, TetWild cannot properly extract major components of the input. This problem is largely reduced when TetWild is applied on the mesh processed by our first step, as shown in (b1) third column, where the major problem becomes the loss of features defined by open surfaces. Similarly for the sofa in (b2), TetWild captures a small part when applied on the original input, and recovers more on the processed mesh by our first step. However, as shown in the inset cutoff view of the sofa cushion, there are inner structures with random orientations that still cause difficulty for its detection of solid, and lead to artifacts in the extracted surface. In contrast, our algorithm processes these open surfaces and inner structures robustly and recovers manifold meshes that well capture the meaningful components. The differences are also confirmed by Hausdorff distances from input meshes: as shown in Fig. 20, TetWild is very sensitive to the flipped orientations and redundancies in the input meshes, and obtains much improved results on our visually processed meshes. Notice that due to the long running time of TetWild, we have managed to run it on 167 test samples out of the 1k without surface extraction failures, on which the distributions of patch numbers and Hausdorff distances are reported.

On the other hand, meshes generated by our pipeline can be used as fixed boundary surfaces for TetWild to work properly. For example, by taking our fixed sofa (Fig. 21 (b2), fifth column) as input, TetWild generates a remeshed model with high mesh quality, as shown in inset.



With MeshFix [Attene 2010]. MeshFix is a comprehensive mesh repair tool that tries to detect and repair holes and self intersections of digitized meshes locally, to avoid unnecessary intrusions of the input. As is acknowledged in [Attene 2010], the tool is not very suitable for processing the man-made CAD models that we target, because the artifacts with them are quite different from that of the digitally acquired meshes. As shown in Fig. 21 (c), MeshFix cannot handle the original inputs with pervasive redundancies and self-intersections robustly; on the visually processed meshes, the results are slightly improved. This is also reflected in the large Hausdorff distances from input meshes shown in Fig. 20. In fact, for the 1k test samples, MeshFix fails to finish properly on 658 input meshes, 38 preprocessed meshes, and 59 visually processed meshes; the distributions of patch numbers and Hausdorff distances are reported on the finished test samples.

With outer hulls [Attene 2016]. [Attene 2016] processes a mesh by first resolving all its self-intersections and subdividing faces where necessary, and then extracting the outer hull that consists of the most exterior faces of the nonmanifold simplicial complex. To generate the outer hull, a greedy algorithm starts from a seeding exterior face, and grows along the manifold patches until reaching nonmanifold edges, where the algorithm checks the fan of surrounding faces incident to the edge and picks the next exterior face to be the one closest to the previous exterior face. Through this process, both the surfaces bounding solids and the outer sheets made of dangling faces outside the solids are extracted.

While good for extracting printable solids, such a strategy relies on the precision and quality of the input models, and does not fit for the targeted man-made 3D datasets with diverse quality levels. As shown in Fig. 21 (d) (without -E option), the extracted solids frequently do not capture the major shape components which, despite being visually quite meaningful, may simply not be closed. Therefore, large parts of the input meshes are cast into the incomplete outer sheets (shown as insets). As a result, when we measure the Hausdorff distances of the fixed solid surfaces from the input meshes, we see large deviations shown in Fig. 20. Note that there are 35 test cases whose solid outer hulls are simply empty, which are excluded from computing the statistics. In comparison, although our method directly works on meshes as well, due to the visually guided approach and global optimization of all connections that do not assume closed boundaries, our method robustly captures the critical structures, large and small.

To preserve the dangling pieces into the final solid, [Attene 2016] provides the option to extrude them into thin shells and apply the above procedure again (denoted -E in Figs. 19, 20, 21). As shown by the reduced Hausdorff distances in Fig. 20 and the examples in Fig. 21 (d), the result shapes become more complete. On the other hand, the result meshes have intricate and unpredictable changes and complexities caused by the extrusion. For example, through the cutoff views, we see that the back parts of the monitor model now have double layers, while the single layer monitor base has newly created subtle geometry. For the plane model as well, the cutoff views show that some surface regions become double layers. In addition, the missile propeller fans due to the relatively small scale are significantly thickened and undesirably joined with the bounding circles after extrusion. In contrast, our minimum intrusion approach cleans the input structural defects without introducing new artifacts. Note that with the extrusion option enabled, more complicated self-intersections are created, therefore the outer hull approach exceeds the 10-hour limit for 8 models from the input or the preprocessed, and 21 models from the visually processed; the statistics are reported on those finished properly.

7.5 Sensitivity and impact of parameters

The algorithm uses several parameters, mostly involved in the visual driven processing step. The parameters can be classified into two types: tunable parameters that may be adapted to different datasets and repair targets, and other parameters that are derived from algorithmic design and need no change. We have chosen the default values of the tunable parameters based on 12 randomly selected models, such that according to our subjective judgments the results preserve both large structures and subtle details. In this section we test the sensitivity of two tunable parameters, ϵ_{visual} and csl_{min},

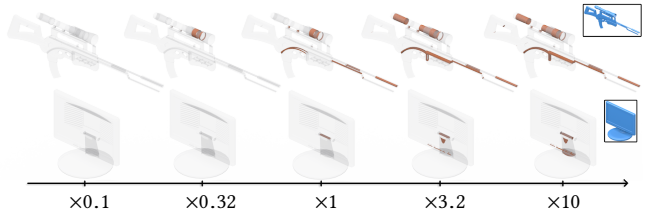


Fig. 22. Patches removed (solid color) by visual processing with various ϵ_{visual}. The default value is in the middle, and scaled by specified factors to both directions. Larger values imply more patches are considered insignificant, but the changes are small within a large range.

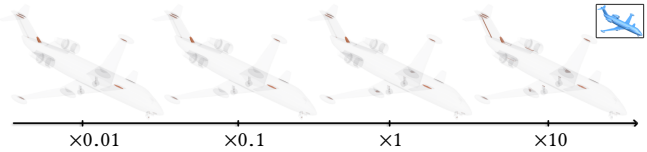


Fig. 23. Patches removed (solid color) by visual processing with various csl_{min}. It limits the maximum zoom-in factor for assessing visual significance and affects the micro patches only. Larger values imply more micro patches are removed, but the changes are marginal within a large range.

and discuss the impact of their variations. The other parameters are discussed in the supplemental material.

Varying ϵ_{visual} slightly changes the existence of the most redundant structures, with major structures defining the object appearances unaltered. We scaled ϵ_{visual} by $[0.1, 0.32, 3.2, 10]$ and tested on the 30 models used for timing visual processing (Sec. 7.3.1); the surface area changes from the default configuration results are $[3.2\%, 2.3\%, 4.1\%, 8.3\%]$ on average, i.e., the differences are generally small even for large variations. Meanwhile, larger values of ϵ_{visual} make more structures considered insignificant and removed; so in a sense, ϵ_{visual} measures the persistence of structures in terms of visual significance. Concrete examples are shown in Fig. 22.

Changing csl_{min} around its default value influences the occurrence of the micro patches only, as it specifies the maximum zoom-in factor for visual significance computation (Sec. 4.2). Tested on the same 30 models with csl_{min} scaled by $[0.01, 0.1, 10]$, the result surface area changes relative to default results are $[0.07\%, 0.07\%, 0.9\%]$ on average, which are indeed marginal as the changes are about micro patches. See Fig. 23 for examples.

8 CONCLUSION

We have presented a method designed for repairing man-made meshes found in popular large scale 3D datasets. The method fully exploits the mismatch between the high visual quality of the meshes and their low structural quality, and uses a two-step pipeline that works directly on the meshes to find the elementary manifold patches that are not visually redundant and connect them into large coherent manifold surfaces using global optimization. Compared with previous mesh repair methods that generally assume the meshes bound solid volumes, our approach is shown to work robustly on the imprecise meshes that are frequently open or have complex inner structures that interfere with solid volume detection. We have applied our method automatically on ModelNet and ShapeNet, and obtained quality results more usable for downstream processing. Our tool can be useful for data cleaning in the era of big data and machine learning for advanced 3D applications.

Limitations and future work. By assuming open surfaces not bounding solid volumes and working on them directly, our approach cannot handle gaps between proximate components that may be intended to be closed (see e.g. the separated components of chairs in Fig. 21). This is in contrast to the previous global repair methods that generally can close gaps by enforcing the volumetric assumption. However, if the assumption indeed holds, our repair can still be used as a preprocessing to improve the quality condition for subsequent application of other global repair methods that handle gaps, as is demonstrated in the TetWild case (Sec. 7.4).

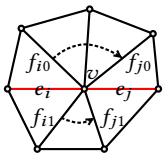
The parameters and thresholds of our algorithm are selected based on randomly sampled experiments on the available datasets, and thus may not be the best choices for all test cases or other datasets. In addition, while we believe the general pipeline of first cleaning elementary patches through localized operations and then combining the patches into coherent manifold meshes is suitable for repairing and cleaning a wide range of man-made meshes with diverse problems, the specific operations and heuristic measurements can be improved: for example, currently the scoring rules for patch significance and regularity rely on pure geometry without considering textures, face groups and other semantic information [Mo et al. 2019], and thus can be limited. In the future, it is worthwhile to consider learning the parameters, the scoring rules and even the operations from annotated datasets with rich input features, in order to be adaptive to dataset characteristics and semantics.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for suggestions, the ShapeNet and ModelNet datasets, and Jianzhao Zhang for help with rendering the models.

A APPENDIX

A.1 Singular curve construction



A singular curve is made of consecutive singular edges. The problem is to decide whether two singular edges should be connected into one curve. Suppose for two consecutive singular edges e_i, e_j sharing a common vertex v, each has a set of adjacent faces $F_i = \{f_{i0}, f_{i1}, ...\}$ and $F_j = \{f_{j0}, f_{j1}, ...\}$ respectively (see inset for illustration). We connect the two edges into a singular curve, if and only if the two sets of faces have a one-to-one correspondence as defined next: for two faces f_{i0}, f_{j0} sharing a common vertex v, a correspondence $f_{i0} \sim_v f_{j0}$ exists if and only if there is a path of faces sharing v that connects f_{i0} and f_{j0} and does not cross a singular edge. Note that the correspondence is purely combinatorial and robust against geometrical degeneracies.

A.2 Proof of output manifoldness

The proof contains two parts. First, we show that there is a one-to-one correspondence between edges in the output mesh M' and the

solved face-pair adjacency relation $\sim_E$. Second, we show that each vertex in M' is regular under the adjacency relation induced by $\sim_E$.

Denote the origin of vertex $v' \in V'$ as $pre(v') \in V$ which is duplicated $|pre(v')^*/\sim|$ times to create v' and others. Conversely, denote the duplicated vertices of v as $img(v)$. Further note that $f'_i \in F'$ differs from $f_i \in F$ only by the updated references to the newly created vertices if any. Therefore, we can pull back the binary relation $\sim_E$ to F' as $\sim'_E$, by $\{f'_i, f'_j\} \in \sim'_E$ iff $\{f_i, f_j\} \in \sim_E$.

We show the map from $\sim_E$ (and thus $\sim'_E$) to edges in M' is one-to-one and onto. For a pair $\{f_i, f_j\} \in \sim_E$, if their shared edge is regular in the input mesh, the edge is preserved in the output. Otherwise when the shared edge is singular, denote the edge as $v_l v_m$. Without loss of generality, we assume v_m is the vertex incident to another singular edge $v_m v_n$ that belongs to the same singular curve as $v_l v_m$; the existence of $v_m v_n$ is guaranteed by our postprocessing (Sec. 6.1). Due to the fan-shaped structure around the singular curve passing through v_m (A.1), the face pairs in $\sim_E$ that share the singular edge $v_l v_m$ (or $v_m v_n$) identify with the face equivalence classes $v_m^*/\sim$ induced by $\sim_E$. Thus by the duplication procedure, for each face pair sharing $v_l v_m$ there is a unique corresponding edge with an endpoint $v'_m \in img(v_m)$ in M'. This shows the one-to-one property. For any edge $v'_i v'_j$ in M', it is adjacent to two faces f'_l, f'_m that must belong to a common equivalence class for $pre(v'_i)^*$ (and $pre(v'_j)^*$), which by definition corresponds to a pair in $\sim_E$. This shows the onto property.

The second part is rather straightforward. By definition (Sec. 6.1), for each vertex v' in the output mesh, it has a star of faces v'^* such that for each face $f' \in v'^*$, its adjacent face through an edge incident to v' is unique and given by $\sim'_E$, and for every two faces $f'_i, f'_j \in v'^*$, they are connected by a path $f'_i, \cdots, f'_j \in v'^*$ such that two consecutive faces are adjacent by $\sim'_E$. Thus by treating each face $f' \in v'^*$ as a node, and linking two nodes $f'_i, f'_j \in v'^*$ if they are adjacent through $\sim'_E$, we obtain a single connected graph where each node has degree at most two, i.e. a cycle or chain. Because in the first part we have identified the edges of M' with $\sim'_E$, the output mesh is manifold.

A.3 Branch-and-Bound and greedy solvers

The Branch-and-Bound algorithm is an approach for finding globally optimal solutions of integer programmings by pruning the search space. We follow [Lazar et al. 2018] and design the BnB algorithm shown in Alg. 3 for solving the manifold mesh reconstruction ILP problem. The variable $conf$ is a vector of size equal to the number of singular curves, and indicates the selected configuration for each singular curve. The rounding to integer solution can be the naive maximal rounding, or our two-step rounding using graph labeling; when our rounding scheme is used, $\bar{x}$ is always feasible.

The greedy solver summarized in Alg. 4 iterates over the singular curves, and in each iteration searches among all remaining candidate configurations and selects the feasible one with maximal objective function value. While the greedy solver always finds feasible solutions, the solutions are mostly suboptimal because of the nontrivial patch orientation constraint (2) (Sec. 5.2). Intuitively, the greedy selection of configurations may fix suboptimal patch orientations that lead to difficulties for later junctions, while our global optimization achieves better coordination.

Algorithm 3: Branch and Bound

Input: the ILP problem
Output: the global optimal solution x^*
$UB = +\infty, conf = \mathbf{0}_{\#curves}, x^* = NaN$, list = $\{conf\}$;
while list *isn't empty* **do**
  $conf$ = list.pop();
  solve relaxed LP while fixing configurations specified in $conf$ to obtain x;
  $LB = -E(x)$;
  $\bar{x}$ = round(x);
  if *($\bar{x}$ is feasible) and ($-E(\bar{x}) < UB$)* **then**
    $UB = -E(\bar{x})$;
    $x^* = \bar{x}$;
  end
  if $LB \geq UB$ **then**
    do nothing;
  else
    find curve C_i for which x_i has maximal entropy $-\sum_j x_{ij} \log x_{ij}$;
    for $j \leftarrow 1$ **to** #Config(C_i) **do**
      $conf[i] = j$;
      list.push($conf$);
    end
  end
end

Algorithm 4: Greedy solver

Input: the ILP problem
Output: the greedy solution x
remain = $\{C_i\}$, visited = $\{\}$;
while remain *isn't empty* **do**
  $i^*, j^* =$ $\arg\max_{i,j} \{E(x_{ij}) | C_i \in$ remain, $1 \leq j \leq$ #Config(C_i), x_{ij} satisfy (2) against visited $\}$;
  remain.remove(C_{i^*}), visited.add(C_{i^*});
  $x_{i^*j^*} = 1, x_{i^*j \neq j^*} = 0$;
end

REFERENCES

2018. 3D Warehouse. https://3dwarehouse.sketchup.com. Accessed: 2018-12-14.

MOSEK ApS. 2018. *The MOSEK Optimizer API for C.* https://docs.mosek.com/8.1/capi/api-reference.html

Marco Attene. 2010. A Lightweight Approach to Repairing Digitized Polygon Meshes. *Vis. Comput.* 26, 11 (Nov. 2010), 14.

Marco Attene. 2014. Direct Repair of Self-intersecting Meshes. *Graph. Models* 76, 6 (Nov. 2014), 658–668.

Marco Attene. 2016. As-exact-as-possible repair of unprintable STL files. *Rapid Prototyping Journal* (05 2016).

Marco Attene. 2017. ImatiSTL - Fast and Reliable Mesh Processing with a Hybrid Kernel. In *LNCS on Transactions on Computational Science*. Springer, 86–96.

Table 2. Summary of notations.

Notation	Summary description
$M = (V, F)$	mesh with vertices V and faces F
$v_i \in V$	mesh vertex
$f_i = (v_{i0}, v_{i1}, v_{i2}) \in F$	mesh face
$e_i = \{v_{i0}, v_{i1}\}$	mesh edge
$v^* \subset F$	the faces adjacent to vertex v
$P = \{f_i \in F\}$	a mesh patch
∂P	boundary curves of patch P
$\mathbf{v}_i \in \mathbf{R}^3$	the 3D coordinates of v_i
$\lvert \cdot \rvert$	length/area of a geometric object, cardinality of a discrete set
$S_{\pm}(P_i) \in [0, 1]$	likelihood of P_i oriented along(+) /against(−) visibility
$VS(P_i) \in [0, 1]$	visual significance score
$\epsilon_{visual} > 0$	threshold for removing redundant patches
$Adj(C)$	patches adjacent to a singular curve C
$de(P_j, P_k, C_i, o)$	dual edge connecting P_j, P_k through C_i with P_j's orientation specified by o
$G = (\{P_i\}, \{de\})$	graph of patches and connections
$S_{reg}(de) \in [0, 1]$	connection regularity for a dual edge
$[x_{i1}, x_{i2}, \dots]$	indicator vector for configurations
$x \rightarrow \pm P$	configuration x induces positive(+) /negative(−) orientation of P
$x \rightarrow de$	configuration x contains dual edge de
$E_{ori} \in [0, 1]$	energy for orientation alignment
$E_{reg} \in [0, 1]$	energy for connection regularity
ω_1, ω_2	weights resp. for E_{ori}, E_{reg}
$\sim_E \subset F \times F$	the face pairs encoded by solution
$f_i \sim f_j$	face equivalence induced by $\sim_E$

Marco Attene, Marcel Campen, and Leif Kobbelt. 2013. Polygon Mesh Repairing: An Application Perspective. *ACM Comput. Surv.* 45, 2, Article 15 (March 2013).

Gilbert Louis Bernstein and Chris Wojtan. 2013. Putting Holes in Holey Geometry: Topology Change for Arbitrary Surfaces. *ACM Trans. Graph. (SIGGRAPH)* 32, 4, Article 34 (July 2013), 12 pages.

CGAL. 2018. CGAL, Computational Geometry Algorithms Library. https://www.cgal.org/.

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *CoRR* abs/1512.03012 (2015). arXiv:1512.03012 http://arxiv.org/abs/1512.03012

Brian Curless and Marc Levoy. 1996. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH '96*. ACM.

H. Edelsbrunner and J. Harer. 2010. *Computational Topology: An Introduction.* American Mathematical Society.

Xiao-Ming Fu, Yang Liu, John Snyder, and Baining Guo. 2014. Anisotropic Simplicial Meshing Using Local Convex Functions. *ACM Trans. Graph. (SIGGRAPH ASIA)* 33, 6, Article 182 (Nov. 2014), 11 pages.

Ryo Furukawa, Tomoya Itano, Akihiko Morisaka, and Hiroshi Kawasaki. 2007. Improved Space Carving Method for Merging and Interpolating Multiple Range Images Using Information of Light Sources of Active Stereo. In *ACCV*. 206–216.

Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *SIGGRAPH '97*. 209–216.

Yotam Gingold and Denis Zorin. 2008. Shading-based Surface Editing. *ACM Trans. Graph. (SIGGRAPH)* 27, 3, Article 95 (Aug. 2008), 9 pages.

André Guéziec, Gabriel Taubin, Francis Lazarus, and Bill Horn. 2001. Cutting and Stitching: Converting Sets of Polygons to Manifold Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 7, 2 (April 2001), 136–151.

Hugues Hoppe. 1996. Progressive Meshes. In *SIGGRAPH '96*. ACM, 99–108.

Alexander Hornung and Leif Kobbelt. 2006. Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds Without Normal Information. In *SGP*.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph. (SIGGRAPH)* 37, 4, Article 60 (July 2018).

Qixing Huang, Leonidas J. Guibas, and Niloy J. Mitra. 2014. Near-Regular Structure Discovery Using Linear Programming. *ACM Trans. Graph.* 33, 3, Article 23 (June 2014), 17 pages.

Zhiyang Huang, Ming Zou, Nathan Carr, and Tao Ju. 2017. Topology-controlled Reconstruction of Multi-labelled Domains from Cross-sections. *ACM Trans. Graph. (SIGGRAPH)* 36, 4, Article 76 (July 2017), 12 pages.

Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. 2013. Robust Inside-Outside Segmentation using Generalized Winding Numbers. *ACM Trans. Graph. (SIGGRAPH)* 32, 4 (2013), 33:1–33:12.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Tao Ju. 2004. Robust Repair of Polygonal Models. *ACM Trans. Graph. (SIGGRAPH)* 23, 3 (2004).

Tao Ju. 2009. Fixing Geometric Errors on Polygonal Models: A Survey. *Journal of Computer Science and Technology* 24, 1 (01 Jan 2009).

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* 32, 3, Article 29 (July 2013), 13 pages.

V. Kolmogorov. 2006. Convergent Tree-Reweighted Message Passing for Energy Minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 10 (Oct 2006).

V. Kolmogorov and R. Zabin. 2004. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 2 (Feb 2004).

Roee Lazar, Nadav Dym, Yam Kushinsky, Zhiyang Huang, Tao Ju, and Yaron Lipman. 2018. Robust Optimization for Topological Surface Reconstruction. *ACM Trans. Graph. (SIGGRAPH)* 37, 4, Article 46 (July 2018), 10 pages.

Peter Lindstrom and Greg Turk. 2000. Image-driven Simplification. *ACM Trans. Graph.* 19, 3 (July 2000), 204–241.

Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. 2018. Paparazzi: Surface Editing by way of Multi-View Image Processing. *ACM Trans. Graph. (SIGGRAPH ASIA)* (2018).

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G. Kim, and Yaron Lipman. 2017. Convolutional Neural Networks on Surfaces via Seamless Toric Covers. *ACM Trans. Graph. (SIGGRAPH)* 36, 4, Article 71 (July 2017), 10 pages.

Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. 2015. Geodesic Convolutional Neural Networks on Riemannian Manifolds. *IEEE ICCV* (2015), 832–840.

Kaichun Mo, Shilin Zhu, Angel Chang, Li Yi, Subarna Tripathi, Leonidas Guibas, and Hao Su. 2019. PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. (2019).

T. M. Murali and Thomas A. Funkhouser. 1997. Consistent Solid and Boundary Representations from Arbitrary Polygonal Data. In *1997 Symposium on Interactive 3D Graphics*. 155–162.

Fakir S. Nooruddin and Greg Turk. 2003. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Transactions on Visualization and Computer Graphics* 9, 2 (April 2003).

Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven Mm Seitz. 2018. PhotoShape: Photorealistic Materials for Large-Scale Shape Collections. *ACM Trans. Graph. (SIGGRAPH ASIA)* 37, 6, Article 192 (Nov. 2018).

Adrien Poulenard and Maks Ovsjanikov. 2018. Multi-directional Geodesic Neural Networks via Equivariant Convolution. *ACM Trans. Graph. (SIGGRAPH ASIA)* 37, 6, Article 236 (Dec. 2018), 14 pages.

Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NIPS*.

Jarek Rossignac and David Cardoze. 1999. Matchmaker: Manifold BReps for Non-manifold R-sets. In *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications (SMA '99)*. ACM, 31–41.

Thomas Windheuser, Ulrich Schlickewei, Frank R. Schmidt, and Daniel Cremers. 2011. Large-Scale Integer Linear Programming for Orientation Preserving 3D Shape Matching. *Computer Graphics Forum* (2011).

Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE CVPR*.

Eugene Zhang and Greg Turk. 2002. Visibility-guided Simplification. In *Proceedings of the Conference on Visualization '02 (VIS '02)*. IEEE Computer Society, 8.

Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh Arrangements for Solid Geometry. *ACM Trans. Graph. (SIGGRAPH)* 35, 4 (2016).

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10, 000 3D-Printing Models. *CoRR* abs/1605.04797 (2016). arXiv:1605.04797 http://arxiv.org/abs/1605.04797

Ming Zou, Michelle Holloway, Nathan Carr, and Tao Ju. 2015. Topology-constrained Surface Reconstruction from Cross-sections. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 128 (July 2015), 10 pages.