Restricted Delaunay Triangulation for Explicit Surface Reconstruction

PENGFEI WANG, Shandong University, China
ZIXIONG WANG, Shandong University, China
SHIQING XIN*, Shandong University, China
XIFENG GAO, Tencent America, United States
WENPING WANG, The University of Hong Kong, China
CHANGHE TU*, Shandong University, China

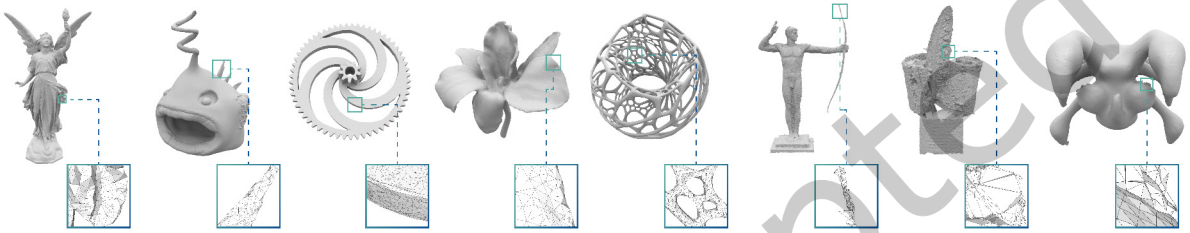


Fig. 1. Our algorithm can robustly process point clouds with high geometric and topologic complexities, such as thin plates/tubes, sharp features, rich details, and small topological holes. It generates the corresponding restricted Delaunay meshes that are watertight, manifold and interpolate the given point clouds. From left to right, No. 1-4 point clouds are simulated real scans using blensor, No. 5 point cloud is a direct sampling on a triangle mesh, and No. 6-7 point clouds are real scans (courtesy of LGG and EPFL), the rightmost point cloud is a real scan using 3D Laser Scanner. Note that our algorithm allows automatic feature alignment for models with sharp features, such as the Gear model (No. 3).

The task of explicit surface reconstruction is to generate a surface mesh by interpolating a given point cloud. Explicit surface reconstruction is necessary when the point cloud is required to appear exactly on the surface. However, for a non-perfect input, e.g. lack of normals, low density, irregular distribution, thin and tiny parts, high genus, etc, a robust explicit reconstruction method that can generate a high-quality manifold triangulation is missing.

We propose a robust explicit surface reconstruction method that starts from an initial simple surface mesh, alternately performs a Filmsticking step and a Sculpting step of the initial mesh and converges when the surface mesh interpolates all input points (except outliers) and remains stable. The Filmsticking is to minimize the geometric distance between the surface mesh and the point cloud through iteratively performing a restricted Voronoi diagram technique on the surface mesh, while

*Co-corresponding authors: Shiqing Xin (xinshiqing@sdu.edu.cn) and Changhe Tu (chtu@sdu.edu.cn).

Authors' addresses: Pengfei Wang, Shandong University, Qingdao, China, 8144756@qq.com; Zixiong Wang, Shandong University, Qingdao, China, zixiong_wang@outlook.com; Shiqing Xin, Shandong University, Qingdao, China, xinshiqing@sdu.edu.cn; Xifeng Gao, Tencent America, Seattle, United States, xifgao@tencent.com; Wenping Wang, The University of Hong Kong, Hong Kong, China, wenping@cs.hku.hk; Changhe Tu, Shandong University, Qingdao, China, chtu@sdu.edu.cn.

0730-0301/2022/5-ART $15.00
https://doi.org/10.1145/3533768

the Sculpting is to bootstrap the Filmsticking iteration from local minima by applying appropriate geometric and topological changes of the surface mesh.

Our algorithm is fully automatic and produces high-quality surface meshes for non-perfect inputs that are typically considered to be challenging for prior state-of-the-art. We conducted extensive experiments on simulated scans and real scans to validate the effectiveness of our approach.

Additional Key Words and Phrases: surface reconstruction, restricted Voronoi diagram, watertight manifold, winding number

1 INTRODUCTION

Surface reconstruction [Berger et al. 2017] refers to restoring the underlying geometry (typically represented as a mesh) from partial information of the unknown surface (CT images, point clouds, etc.). Reconstruction methods can be roughly divided into implicit reconstruction approaches [Carr et al. 2001; Kazhdan and Hoppe 2013] and explicit ones [Bernardini et al. 1999; Cohen-Steiner and Da 2004; Digne et al. 2011]. Implicit approaches often need to infer a certain implicit function and then extract the zero-level iso-surface as the reconstructed result, while explicit approaches aim at directly connecting the points into a triangle mesh. Either group of approaches has its advantages. Explicit approaches are useful for maintaining high fidelity and inheriting as-much-as-possible information from the raw data.

The surface reconstruction problem is fundamentally challenging when the input point cloud has poor quality (lack of normals, low point density, irregular point distribution, etc.) and the represented shape has thin plates/tubes, sharp features, rich details, or small topological holes. Reconstructing such geometrically and topologically complicated shapes remains to be challenging yet fascinating so far. The theme of this paper is explicit surface reconstruction, assuming that the input point cloud is not equipped with normal vectors and has various defects such as low/irregular point density, noise, missing data, and outliers.

The goal of this paper is to develop a robust and efficient solver to obtain a watertight manifold triangle surface that can well manifest the real geometry. The difficulties are two-fold. First, it is not easy to guarantee a watertight manifold, especially for an unorganized and poor-quality point cloud due to the high combinatorial complexity. Second, accurately predicting the underlying geometry is highly non-trivial. The majority of existing explicit methods employ Delaunay triangulation or Voronoi diagram based techniques, like alpha shape [Edelsbrunner and Mücke 1994], power crust [Amenta et al. 1998], triangular sculpting [Boissonnat 1984], mesh growing [Li et al. 2009] and some improved versions [Amenta et al. 2000, 2001; Attali 1998; Guo et al. 1997; Veltkamp 1995; Yang et al. 2010]. However, the existing approaches are weak to deal with poor-quality inputs.

In this paper, we introduce a guiding surface that is initially simple, e.g., a spherical bounding surface, and keep evolving the surface through an iterative procedure until it interpolates all the points (except outliers) and well represents the underlying shape. Within each iteration, our algorithm consists of a Filmsticking step and a Sculpting step. The Filmsticking step incrementally augments the guiding surface to interpolate the given points based on a technique of restricted Voronoi diagram, while the Sculpting step is to chip away redundant solids between the surface and the target geometry represented by the point cloud. The key difficulty of the proposed framework is to strictly guarantee the surface to be a watertight manifold throughout the whole surface evolution. Figure 1 shows several examples to exhibit the capability of our algorithm for handling challenging point clouds; See the attached video for demonstrating how the guiding surface evolves, step by step, from a simple spherical surface to the final stage. An intuitive explanation to account for the alternative refinement technique is based on Gestalt psychology [Köhler 1967]: among several geometrically possible configurations, the one that possesses the best, simplest, and most stable shape will actually occur.

Our main contributions are as follows:

- With the support of a guiding surface, we alternately perform the Filmsticking step and the Sculpting step until the surface well interpolates the point cloud and accurately manifests the real geometry. The

Filmsticking step aims at connecting the points into a triangle mesh based on proximity while the Sculpting step modifies the overall shape based on simplicity priors.

- During the Filmsticking step, we use the technique of restricted Voronoi diagram (RVD) to incrementally augment the vertex set of the guiding surface. We give a set of strategies to guarantee that the guiding surface is a watertight manifold during the surface augmentation.
- During the Sculpting step, we tetrahedralize the volume enclosed by the guiding surface and delete redundant tetrahedral elements by more accurately distinguishing the interior from the exterior based on winding number. The Sculpting operation makes the resulting surface (serving as the input of the next iteration) closer to the underlying geometry, while maintaining the surface to be manifold.

2 RELATED WORK

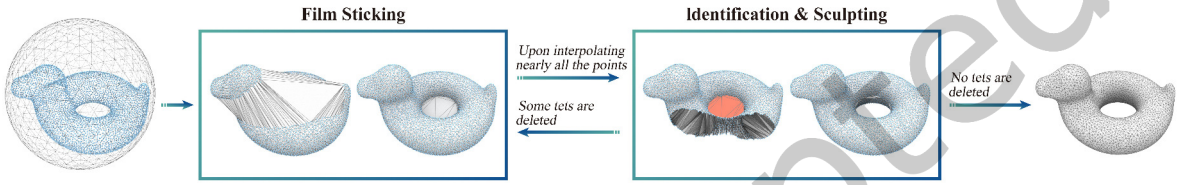


Fig. 2. The pipeline of our algorithm. From left to right: A point cloud and the bounding sphere as an initial (guiding) surface; A Filmsticking step to make the guiding surface interpolate nearly all the points; A Sculpting step to chip off tetrahedral elements that are identified as unnecessary; Return the guiding surface as the result if it remains unchanged in the Sculpting step.

Surface reconstruction from 3D point clouds is a fundamental geometry processing task in the field of computer graphics. Numerous algorithms [You et al. 2020] have been reported in the last four decades. In this section, we briefly review the existing algorithms.

Approximation methods. Large body of literature [Berger et al. 2013] that focuses on inferring the implicit equation and then extracts the zero level set surface. For example, radial basis functions (RBFs) [Carr et al. 2001; Nagai et al. 2010] are a useful tool to infer the implicit equation of the underlying surface. Signed-distance fields (SDF) are also helpful in approximating the underlying field. For instance, Poisson reconstruction [Kazhdan et al. 2006] and screened Poisson reconstruction [Fuhrmann and Goesele 2014; Kazhdan and Hoppe 2013] formulate the inside-outside indicator field (like an SDF) as the solution to the Poisson equation. When the given point cloud is very sparse, Ladicky et al. [Ladicky et al. 2017] proposed to learn and predict surfaces based on regression forest.

There are also some new techniques to deal with challenges in specific surface reconstruction scenarios, e.g., urban buildings or indoor scenes [Le and Li 2018; Wang et al. 2017; Yi et al. 2017]. For instance, Le and Li [2018] proposed a 3D global matching algorithm to handle insufficient temporal sampling or fast camera movement. Schertler et al. [2017] gave a field-aligned online surface reconstruction method by sidestepping the signed-distance computation of classical reconstruction techniques, enabling an efficient output-driven interactive scanning and reconstruction workflow. For a comprehensive survey, we refer readers to [Berger et al. 2017].

In addition, there are some other approximation-based surface reconstruction approaches. For example, one can approximate the underlying surface using a set of surface primitives [Nan and Wonka 2017; Ohtake et al. 2005]. The most significant advantage of approximation methods lies in their ability to report a smooth manifold triangle surface, but they have some disadvantages. First, it does not interpolate the given points and thus is hard to fully retain the original geometry information (tiny geometric features/details, thin-plate models, or

tubular shapes). Some carefully devised techniques such as adaptive subdivision [Aroudj et al. 2017; Fuhrmann and Goesele 2014; Ummenhofer and Brox 2015] or persistence diagrams [Gabrielsson et al. 2020] have to be utilized to improve the geometric fidelity. Second, most of the approximation methods cannot work without normal vectors.

Interpolation methods. The basic requirement of interpolation methods is to interpolate all the input points. Existing approaches in this category can be further divided into four main types: tangent-plane approaches, restricted Delaunay based approaches, sculpting approaches, and region-growing approaches. Tangent-plane approaches map points in a small range onto the tangent plane [Gopi et al. 2000] and then infer the connections between points. However, it is hard to output a watertight manifold since stitching the piece-wise connections is a highly non-trivial task. Delaunay based approaches such as the classic Crust [Amenta et al. 1998] and Cocone [Amenta et al. 2000] inherit the spirit of Delaunay triangulation and aim at generating high-quality triangles. They work well on the condition that the points are dense enough to encode the local feature size, but the criterion is hard to meet in practice. Sculpting algorithms [Peethambaran and Muthuganapathy 2015a] build tetrahedralization in the convex hull and then distinguish interior tetrahedral elements from exterior tetrahedral elements. In [Kuo and Yau 2005], instead, a greedy region-growing technique is used based on some criteria, e.g., the empty-ball principle. However, the principle is weak to deal with sparse points, geometric features, and thin plates. Ball Pivoting [Bernardini et al. 1999], also in a region-growing fashion, works by repeatedly rolling a ball with a radius of α around the front edges, and when there are three points touched by the ball, they are connected into a triangle. The intrinsic property driven (IPD) algorithm [Lin et al. 2004] improves the original ball-pivoting algorithm by determining the target point by the weighted least length criterion. Later, Li et al. [2009] put forward a direct region-growing algorithm to deal with relatively flat regions in advance, followed by sharp regions. To summarize, point insufficiency is the biggest challenge for explicit approaches. Existing point connection schemes cannot work well in complicated situations.

Learning based methods. Diverse learning-based reconstruction methods have sprung up in recent years. For a survey on geometric deep learning, we refer the reader to [Bronstein et al. 2017; Xiao et al. 2020]. An early approach [Dai and Nießner 2018] developed a graph-based formulation to generate triangles in a mesh. However, the method specializes in particular shape categories and does not guarantee to produce manifold meshes. Deep Geometric Prior (DGP) [Williams et al. 2019] partitions the point cloud into different patches and then trains different MLPs for each patch. It has to use Poisson reconstruction as post-processing. Similar research works aiming at predicting implicit shape representations also include [Erler et al. 2020] and [Gropp et al. 2020].

Shrinking the wrapping surface is an important technique for surface reconstruction. For example, Point2Mesh [Hanocka et al. 2020] uses MeshCNN [Hanocka et al. 2019], an edge-based CNN with weight-sharing convolutions, to learn a self-prior for surface reconstruction. It works by repeatedly tuning the initial surface until it sufficiently approaches the real shape. Although it shows great potential in surface reconstruction, the disadvantages are very obvious. On one hand, it requires thousands of iterations (about 3 hours) to obtain a reconstructed result. On the other hand, it neither interpolates the points nor guarantees high-quality triangulation. Some researchers consider the surface reconstruction assuming that the input is an image. Pixel2Mesh [Wang et al. 2018] progressively deforms an ellipsoid based on a graph-based convolutional neural network to infer a visually appealing and physically accurate geometry for a 0-genus model. [Pan et al. 2020] further suggest deforms a single genus-0 template mesh to the target surface, the genus-0 mesh is evolved by alternately performing mesh deformation and topology modification, where the deformation network predicts the per-vertex displacements between the reconstructed mesh and the ground truth, and the topology modification network is used to modify topology by prune the error-prone faces. However, these methods train a network for each class of objects and are not easy to support cross-category surface reconstruction.

Deep learning techniques are also used in the implicit reconstruction of a large scene. [Peng et al. 2020] learns a scalable implicit representation by using a convolutional decoder to predict the occupancy values by local voxel crop. In this way, it can aggregate local and global information and get the reconstruction of large-scale scenes. [Jiang et al. 2020] introduces Local Implicit Grid Representations, a new 3D shape representation designed for scalability and generality, to reconstruct 3D objects from partial or noisy data based on the prior knowledge that most 3D surfaces share geometric details at a scale smaller than an entire object and larger than a small patch.

Deep learning-based approaches for interpolating a point cloud are quite a few. Very recently, Sharp and Ovsjanikov [2020] proposed a deep learning-based explicit surface reconstruction approach, named *PointTriNet*. PointTriNet enables point set triangulation as a layer in the 3D learning pipeline. It consists of two neural networks, i.e., a classification network that predicts whether a candidate triangle should appear in the triangulation and a proposal network that suggests additional candidates. PointTriNet is weak to deal with poor-quality point clouds - the output is not a manifold mesh.

Restricted Delaunay triangulation in surface reconstruction. Restricted Delaunay triangulation (RDT) is found to be helpful in mesh extraction [Boltcheva and Lévy 2017; Khoury and Shewchuk 2016; Pellerin et al. 2014; Yan et al. 2009] due to its potential of directly producing high-quality triangulation from unorganized points. It was originally used to improve the meshing quality [Yan et al. 2009]. [Pellerin et al. 2014] proposed to remesh the surfaces of 3D sealed geological structural models to align user-specified contact lines. [Khoury and Shewchuk 2016] observed that if the input points, sampled from a given mesh surface, meet the local-feature-size (LFS) standard, it is able to connect the points into a triangle mesh that is another triangulation of the manifold. However, the LFS-sampling condition cannot be satisfied in practice. Therefore, we develop a set of strategies to tackle this issue in this paper. [Boltcheva and Lévy 2017] suggested taking each point as a disk, orthogonal to the estimated normal direction, and then connecting the points into a surface based on the proximity between disks. However, it may produce many non-manifold edges, requiring a tedious post-processing step.

3 ALGORITHM

The research theme of this paper is explicit surface reconstruction, i.e., to construct a triangle mesh interpolating the given point cloud $\mathbf{P} \in \mathrm{R}^3$. Our goal is to output a watertight manifold triangle mesh, denoted as $\mathbf{G} \in \mathrm{R}^3$, that accurately represents the underlying shape, even when there are imperfections of input data.

As illustrated in Figure 2, our algorithm begins with a simple guiding surface $\mathbf{G}$ that is manifold, watertight, and encloses the input point cloud. We use a spherical surface for initializing $\mathbf{G}$ in our experiments. Then, we iterate the following two steps, *Filmsticking* and *Sculpting*, to update $\mathbf{G}$ until it converges. Note that, during the entire process, $\mathbf{G}$ is maintained to be a manifold.

Filmsticking is to attract $\mathbf{G}$ from its original shape that does not necessarily interpolate any points of $\mathbf{P}$ to a state that interpolates as many as possible points of $\mathbf{P}$ through computing Voronoi diagrams restricted on the guiding surface (Figure 3); See the details in Section 4). We need to ensure the manifoldness of $\mathbf{G}$ by introducing a set of new strategies. Note that the output of this step may be with a geometric and topological configuration much different from the desired surface.

Sculpting takes the output from Filmsticking and generates a new version of $\mathbf{G}$ by re-identifying the interior and the exterior based on winding number. At the end of the step, the resulting surface is closer to the real shape in both geometry and topology (Figure 8); See Section 5 for details.

Our algorithm operates by alternately performing the Filmsticking step and the Sculpting step, and terminates when the Sculpting step makes no changes to $\mathbf{G}$. The evolution of the guiding surface generally requires less than 10 iterations. The pseudo-code is given in Algorithm 1. The biggest technical challenge lies in enforcing every intermediate state of the guiding surface to be manifold throughout the evolution.

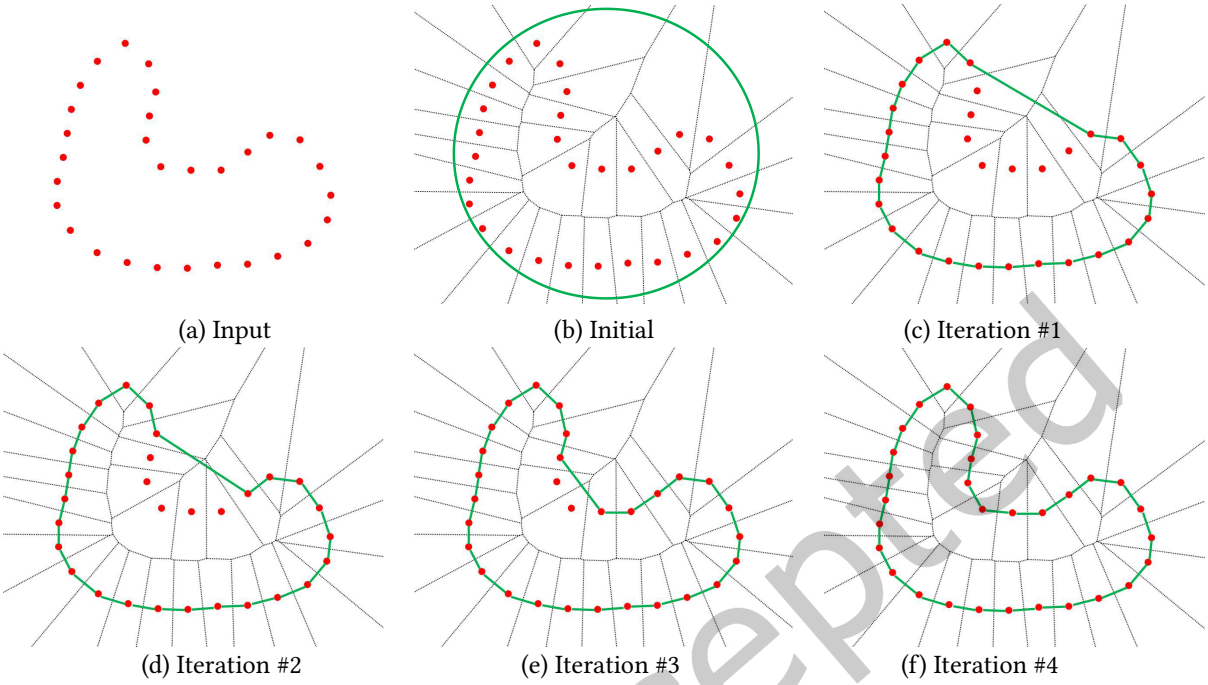


Fig. 3. A simple 2D example for illustrating how the Filmsticking works.

ALGORITHM 1: Explicit Surface Reconstruction

Input: Point set **P** (without normal vectors).
Output: Triangulated watertight manifold **M**
. Initialize the guiding surface G_0 to be the bounding sphere of **P**;
$i := 1$;
repeat
  $\mathbf{G_i}$ = Repeat **Filmsticking**($\mathbf{G_{i-1}}$); //See Section 4;
  $\mathbf{G_{i+1}}$ = **Sculpting**($\mathbf{G_i}$); //See Section 5;
  $i := i + 2$;
until $\mathbf{G_i} == \mathbf{G_{i+1}}$;
output **M** := **G**.

4 FILMSTICKING

Starting from a simple spherical surface **G**, the goal of this step is to iteratively evolve **G** to a surface that interpolates **P** while ensuring the manifoldness of **G**. The evolution of **G** in the Filmsticking step is to perform the following two steps: 1) computing a **G**-restricted Voronoi diagram (abbreviated as **G**-RVD) for **P**, and 2) replacing **G** by the dual of the **G**-RVD.

4.1 G-RVD and G-RDT

Consider each point $p_i \in \mathbf{P}$ as a site, the Voronoi diagram of $\mathbf{P}$ is composed of $|\mathbf{P}|$ Voronoi cells, where each point p_i has a corresponding Voronoi cell

$$C_i = \{x \in \mathrm{R}^3 \mid ||p_i - x|| \leq ||p_j - x||, \forall i \neq j\}.$$

The **G**-restricted Voronoi diagram is defined as the intersection between **G** and the Voronoi diagram of **P**. In other words, **G**-RVD is composed of a set of **G**-restricted surface cells $R_i \triangleq C_i \cap \mathbf{G}$.

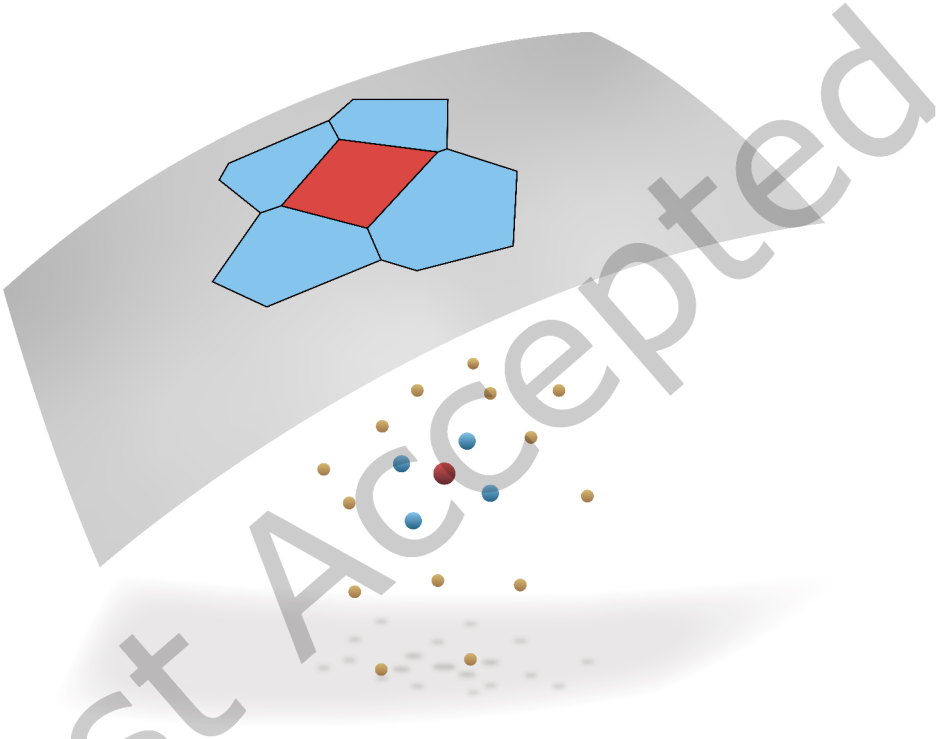


Fig. 4. **G**-restricted Voronoi diagram. Different from the traditional restricted Voronoi diagram, the sites (red, blue, yellow dots) are not located on the base surface (colored in gray). The red region is dominated by the red site, which is given by intersecting the base surface with the corresponding Voronoi cell.

Figure 4 shows an example of **G**-RVD. The main difference between **G**-RVD and the traditional RVD lies in that **P** may not be lying on **G**. If a point $p_i \in \mathbf{P}$ is at a large distance from **G**, it doesn't contribute to the **G**-RVD, making the cell $R_i = C_i \cap \mathbf{G}$ being empty. Suppose that p_i's cell and p_j's cell are non-empty and they share a common boundary. We connect p_i and p_j using a straight-line segment. The resulting triangulation is named **G**-restricted Delaunay triangulation (**G**-RDT), which shall be used to update **G**.

Both [Leibon and Letscher 2000] and [Khoury and Shewchuk 2016] show that when **P** is a sufficiently dense sample set of a certain surface, the restricted Delaunay triangulation leads to a manifold mesh. The property doesn't hold in our scenario due to the two aspects: (1) **P** may not be on **G**, and (2) **P** may be very sparse and irregular. Therefore, it is necessary to check and fix the manifoldness of **G**-RDT w.r.t. **P**.

4.2 Manifoldness checking rules

Suppose that p_1, p_2, p_3 determine a **G**-RVD vertex, then p_1, p_2, p_3 are connected into a triangle. Based on [Bischoff et al. 2005; Edelsbrunner and Shah 1997], we give the manifoldness conditions as follows, i.e., the triangulation induced by **G**-RDT is 2-manifold if the following conditions are satisfied at the same time:

(1) Any cell of **G**-RVD must be homeomorphic to a 2D disk;
(2) Any two neighboring cells of **G**-RVD must have one common edge (rather than multiple segments).
(3) Any cell of **G**-RVD has at least three neighbors.

Condition #2 implies that no non-manifold edge exists. Satisfying both Condition #1 and Condition #2 implies that **G**-RDT contains neither non-manifold vertices nor non-manifold edges. Condition #1 and Condition #2 are naturally guaranteed if the input points meet the LFS standard. In our problem, however, they are likely to be violated due to (1) **P** may not be on **G**, and (2) **P** may be very sparse and irregular. Condition #3 is to avoid the occurrence of two coinciding triangles that share the same vertex set but with different orientations. For example, one can imagine the situation that three points are located on the equator of a spherical surface, yielding a degenerate configuration with three vertices, three edges, and a couple of triangle faces.

In the following, we enumerate all the four possible cases that can violate Condition #1,#2,#3 based on the adjacency between **G**-RVD cells (see Figure 5):

Case (a): a point dominates disconnected Voronoi cells.
Case (b): a **G**-RVD cell has two or more boundary curves.
Case (c): a point dominates the whole surface, or its **G**-RVD cell has only one or two neighboring **G**-RVD cells.
Case (d): the common boundary between two adjacent **G**-RVD cells consists of two or more curved segments.

It is easy to know that Case (a,b,c,d) exhaust all the violation cases. To be more detailed,

- Violating Condition #1 must produce either Case (a) or Case (b).
- Violating Condition #2 must produce Case (d).
- Violating Condition #3 must produce Case (c).

Remark. Based on the above discussion, if we can eliminate Case (a,b,c,d), then the induced **G**-RDT must be a manifold mesh that has the possibility of being topologically different from **G**. Imagine that **G** is a thin-sheet model with many topological holes punched into the surface. If the points are too sparse to encode the topological holes, it is hard to interpolate the points into a polygonal mesh that is topologically identical with the high-genus thin-sheet surface. We ignore the topological inconsistency issue caused by point insufficiency in this paper.

4.3 Manifoldness fixing strategies

Given a **G**-RVD that possibly violates Condition #1 (Figure 5(a-b)), Condition #2 (Figure 5(d)), and Condition #3 (Figure 5(c)), we need to correct the dual to **G**-RVD so as to produce a manifold triangle mesh. In the following, we first describe the correction strategy for each violation case, followed by composing these strategies into an algorithm. Finally, we prove that our algorithm will produce a manifold triangle mesh (or an empty mesh in the worst case).

Strategy I. When a point dominates multiple disconnected regions (Figure 5(a)), we keep only the nearest connected component for the point, leaving the other connected components *ownerless*; See the red region of Figure 5(a). The ownerless regions need to be re-partitioned by the local Voronoi diagram w.r.t. its neighboring sites. See [Wang et al. 2020] for details.

Strategy II. If a **G**-RVD cell, belonging to p, is connected but not homeomorphic to 2D disk (Figure 5(b)) or the number of its neighboring cells is less than 3 (Figure 5(c)), we remove p temporarily and re-partition p's Voronoi cell to its neighboring sites based on the local Voronoi diagram.

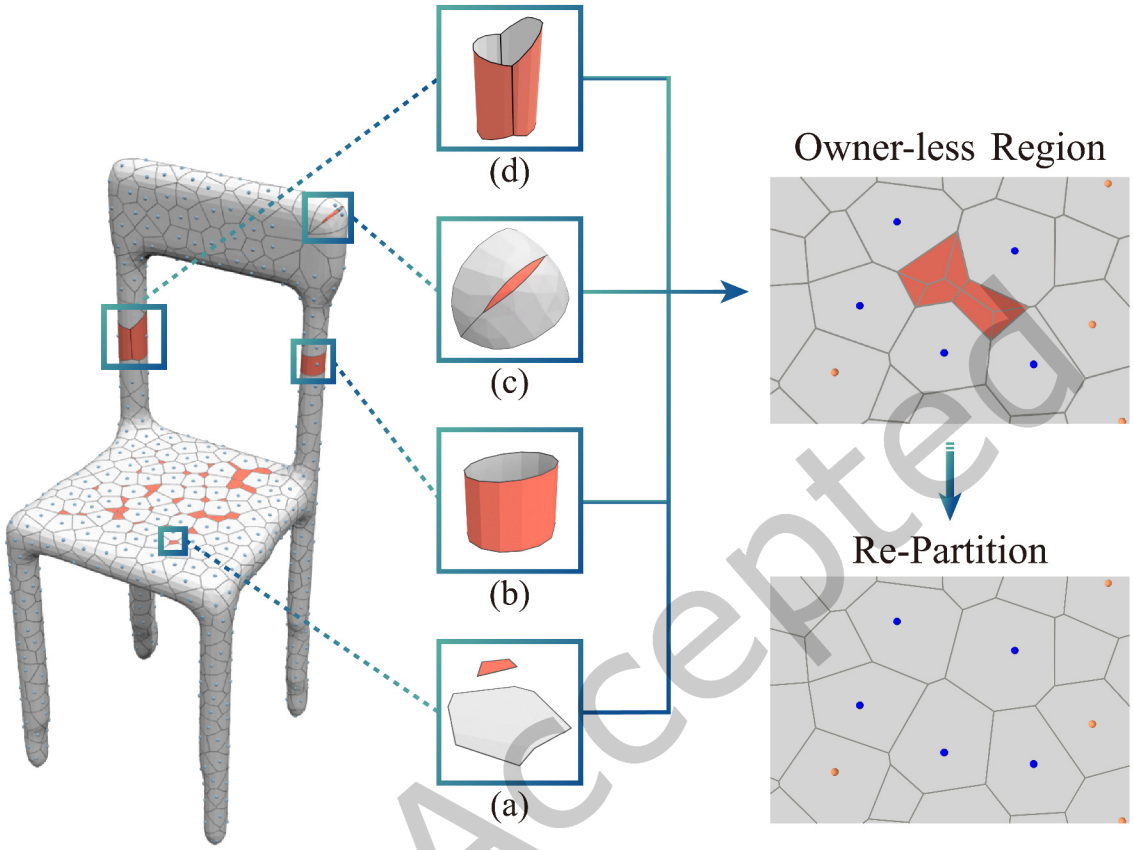


Fig. 5. The **G**-restricted Voronoi diagram w.r.t. the input point cloud **P** may not lead to a manifold surface due to the violation of Condition #1,#2,#3. There are four violation cases in total: (a) a point dominates disconnected regions, (b) the region belonging to a point is connected but has two or more boundary curves, (c) the number of neighbors is less than 3, and (d) the common boundary consists of two or more curved segments.

Strategy III. As Figure 5(d) shows, there are at least 2 shared edges between p_i's RVD cell and p_j's RVD cell. We remove p_i and p_j at the same time temporarily and re-partition their Voronoi cells by the neighboring sites based on the local Voronoi diagram. It's worth noting that this strategy is incurred only if there are no degree-2 sites.

Priority for using Strategy I/II/III. Our task is to eliminate Case (a,b,c,d), using Strategy I/II/III, so as to meet Condition #1, #2, #3 at the same time. As mentioned above, Strategy I is used to fixing Case (a), Strategy II is used to fix Case (b,c) and Strategy III is used to fix Case (d). In the implementation, the priority for calling Strategy I/II/III is

$$\text{Priority(Strategy I)} > \text{Priority(Strategy II)} > \text{Priority(Strategy III)}.$$

It's worth noting that Strategy I does not change the number of contributing sites, Strategy II removes one point and Strategy III removes two points. The main consideration for defining such a priority is to tune the connections

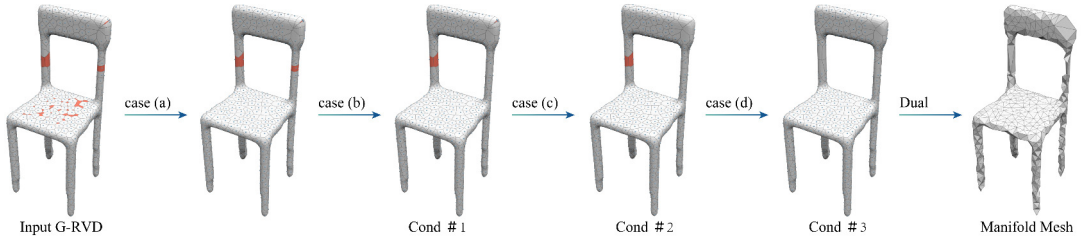


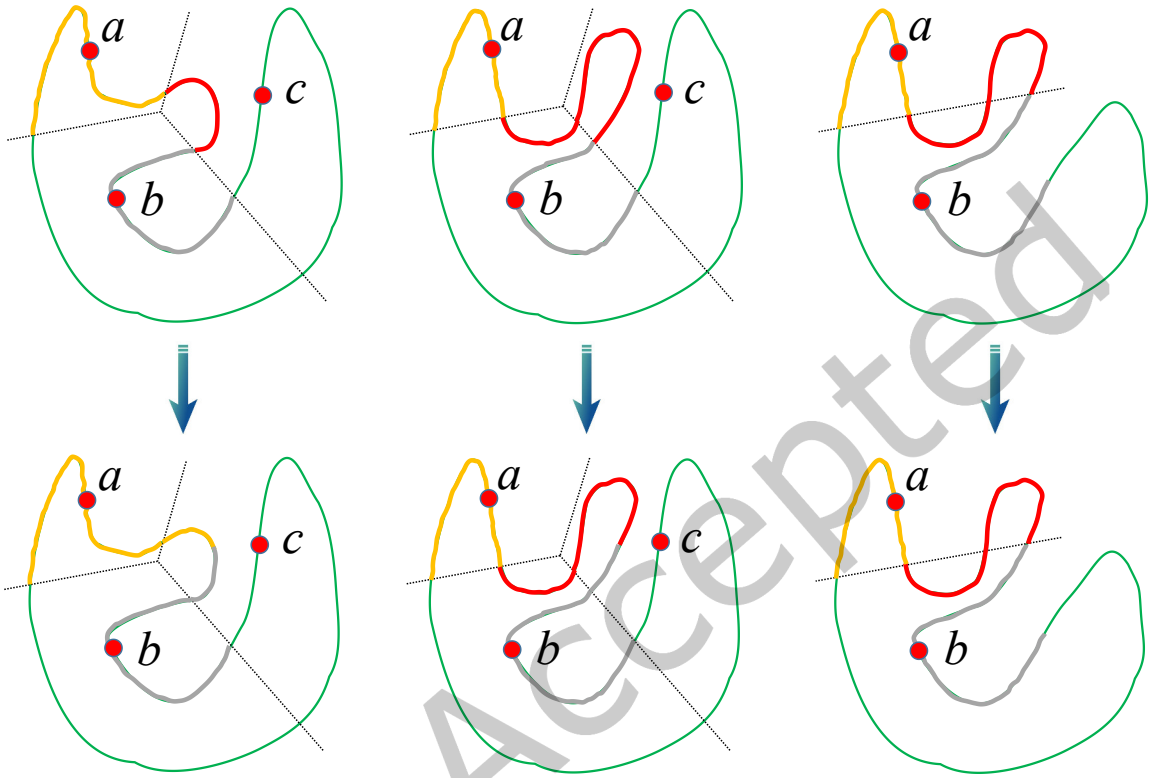
Fig. 6. We run Algorithm 2 on the Chair model with 500 sample points. The non-manifoldness issues are fixed step by step, resulting in a 2-manifold mesh (meeting Condition #1, #2, #3).

ALGORITHM 2: Filmsticking

Input: Point cloud **P** and guiding surface **G**.
Output: A new guiding surface that interpolates more points of **P**.
repeat
  Compute RVD $\{\Omega_i\}_{i=1}^n$ according to **P** and **G**;
  Step 1: Apply Strategy-I to identify and correct those RVD regions that violate singly-connected property (Condition 1 violated by Case (a) of Figure 5);
  Step 2: Apply Strategy-II to identify and eliminate those RVD regions that is singly-connected but not be homeomorphic to 2D disk (Condition 1 violated by Case (b));
  if *Case (a) occurs* **then**
    Goto Step 1;
  end
  Step 3: Apply Strategy-II to eliminate Case (c) (guarantee Condition 3);
  if *Case (a) occurs* **then**
    Goto Step 1;
  end
  if *Case (b) occurs* **then**
    Goto Step 2;
  end
  Step 4: Apply Strategy-III to eliminate Case (d) (guarantee Condition 2);
  if *Case (a) occurs* **then**
    Goto Step 1;
  end
  if *Case (b) occurs* **then**
    Goto Step 2;
  end
  if *Case (c) occurs* **then**
    Goto Step 3;
  end
  G = Dual of RVD;
until G *remains unchanged*;
Output **G**;

to be a manifold while removing as-few-as-possible points. We summarize the pseudo-code in Algorithm 2. In Figure 6, we extract 500 sample points from the Chair model. With the original surface being the guiding surface,

it can be observed that all the issues of Case (a,b,c,d) exist. These issues can be completely eliminated, step by step, following Algorithm 2. We shall give the proof later.



(a) Situations #1 (b) Situations #2 (c) Situations #3

Fig. 7. Three situations when one partitions ownerless regions, where the segment colored in red is the ownerless region waiting for partitioning, the segment colored in yellow is dominated by a, and the segment colored in grey is dominated by b. The re-partitioning of an ownerless region is based on a local Voronoi diagram w.r.t the nearby sites. (a) The point a and the point b carve up the ownerless region given by the point c (note that in this example c dominates two disconnected regions), making the ownerless region vanishing. (b) b's region is augmented, and the ownerless region is shrunk but does not vanish. (c) The ownerless region cannot be shrunk anymore since the augmentation of a's region or b's region will violate the connectivity principle.

Re-partitioning ownerless regions. When Case (a) happens, there exists a site p that dominates multiple disconnected pieces. We assign the nearest patch to p while labeling the other patches ownerless. The basic strategy is to re-partition the ownerless regions by the nearby sites with the help of the local Voronoi diagram. (In fact, Case (b,c,d) also possibly lead to ownerless regions upon site removal, which can be handled likewise.)

There are three situations after re-partitioning an ownerless region:

- Situations #1: the ownerless region vanishes (Figure 7(a)).
- Situations #2: the ownerless region shrinks but does not vanish (Figure 7(b)).
- Situations #3: the ownerless region remains unchanged (Figure 7(c)).

It's worth noting that under the circumstance that no site is deleted, it is impossible that the re-partitioning operation makes the ownerless region become larger. Therefore, the above three situations exhaust all the possibilities.

For Situations #2, we continue partitioning the remaining ownerless regions. If Situations #3 never occurs, the ownerless regions can be completely eliminated within finitely many iterations. For Situations #3, we randomly remove one of the nearby sites, say, q, and then label q's dominating region with "ownerless".

Convergence analysis. We discuss the convergence of Algorithm 2 in two steps. First, we shall show that by repeatedly re-partitioning ownerless regions, Case (a) can be totally eliminated. Second, we shall show Algorithm 2 is bound to terminate after a limited number of region-partitioning or site-removal operations.

Lemma 4.1. *Our strategy for handling Case (a), can completely eliminate ownerless regions.*

Proof. After re-partitioning ownerless regions, if Situations #1 or Situations #2 happens, then the total area of ownerless regions is reduced. Each time when Situations #3 happens, the total area of ownerless regions increases but the number of sites is reduced. However, Situations #3 cannot always happen since the total number of sites is limited (not more than the total number of points in the point cloud). The worst case is that only one point remains, and the point dominates the whole surface. Therefore, our strategy cannot lead to an endless loop. □

Lemma 4.2. *Algorithm 2 (Filmsticking) can terminate with finitely many iterations. The output is either a manifold triangle mesh or empty.*

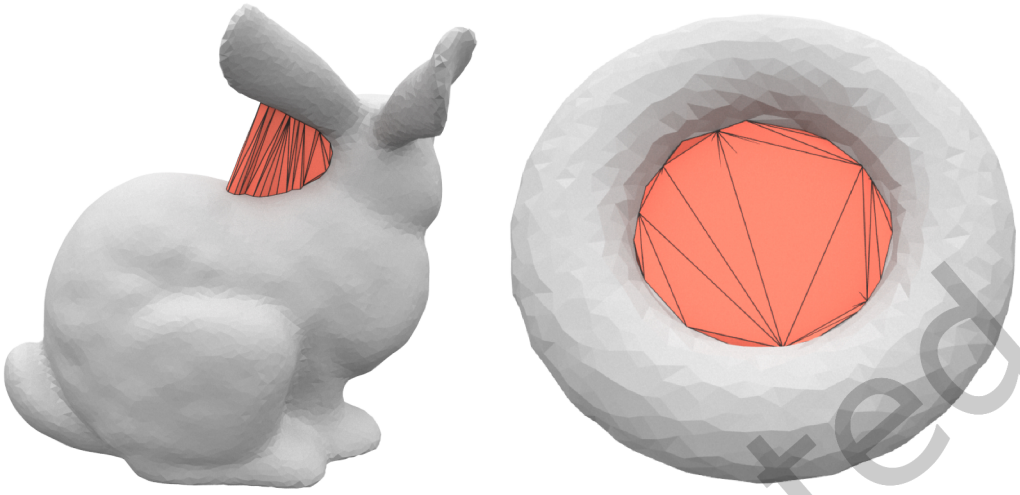
Proof. Recall that we use Strategy II/III to deal with Case (b,c,d). Both Strategy II and Strategy III will remove at least one site, which implies that during the execution of the Algorithm 2, Case (b,c,d) cannot always occur (the total number of sites is limited). Lemma 4.1 points out that Case (a) can be eliminated within finitely many iterations. Therefore, Algorithm 2 can terminate with finitely many iterations (it is possible that all the sites are removed, producing an empty mesh). □

Difference from [Wang et al. 2020]. As mentioned above, Case (a,b,c,d) exhaust all the cases that violate the manifoldness condition. [Wang et al. 2020] considers only Case (a) and Case (b), but ignores Case (c) and Case (d). In fact, [Wang et al. 2020] assumes that the base surface is known and all the points are sitting on the base surface but in this paper the assumption does not hold any more - some points may be far away from the guiding surface. Empirical evidence shows that there is a high occurrence of Case (c,d) in our scenario. Furthermore, the strategy for eliminating Case (a) proposed in [Wang et al. 2020] is flawed because Situation #2 (Figure 7(b)) and Situation #3 (Figure 7(c)) are not taken into account. To summarize, in this paper, we systematically discuss all the cases that violate the manifoldness condition, which is of great importance in both theory and practice.

5 SCULPTING

The Filmsticking step outputs a manifold triangle mesh $\mathbf{G}$ that interpolates part of $\mathbf{P}$ except just a few points that violate the manifoldness conditions[1]. However, $\mathbf{G}$ may be much different from the desired output due to *highly concave parts or topological holes*; See Figure 8. In this section, we introduce a step of Sculpting to further tune $\mathbf{G}$ such that the interior of $\mathbf{G}$ can be better distinguished from the exterior to generate a new surface $\mathbf{G}_1$ that can better manifest the real geometry. The workflow consists of two sub-steps: (1) build the Delaunay triangulation of the interior of $\mathbf{G}$ with tetgen [Si 2015], resulting in a collection of tetrahedral elements $\mathbf{T} = \{T_i\}_{i=1}^{m}$; and (2) find a subset of $\mathbf{T}$ whose boundary surface is manifold (meeting Condition #1, #2, #3) based on analysis of winding number [Barill et al. 2018]. The pseudo-code of Sculpting is shown in Algorithm 3.

[1]There are also some complicated cases where $\mathbf{G}$ interpolates only part of $\mathbf{P}$, which shall be discussed in Section 6.



(a) Concave geometry (b) Topological holes

Fig. 8. Removal of dangling parts from the guiding surface. (a) Dangling parts due to highly concave geometry. (b) Dangling parts due to topological holes.

ALGORITHM 3: Sculpting

Input: Point set **P** and guiding surface **G**.
Output: Manifold triangle mesh **M**.
Tetrahedralize the interior volume defined by **G**;
if *there are tetrahedral elements deemed as exterior based on winding number* **then**
    Delete these tetrahedral elements;
    Return **M** = Shell of the tetrahedral mesh;
end
Once the resulting mesh is non-manifold, we restore some tetrahedral elements such that the surface of the tetrahedral mesh is a manifold; See Section 5;
Return **M** = Shell of the tetrahedral mesh;

Geometric hints for sculpting. Sculpting is an important technique to distinguish the interior from the exterior. A typical way is to tetrahedralize the convex hull of **P** and then peel the convex hull, tetrahedron by tetrahedron and from outer to inner until no tetrahedral element can be deleted. Many sculpting algorithms depend on different geometric hints. [Boissonnat and Jean-Daniel 1984] proposes to delete T_i if the maximum distance between the faces of a boundary tetrahedron and the associated parts of the circumsphere of the tetrahedron exceeds a tolerance. [Chaine 2003] checks each triangle f on the up-to-date boundary surface to see if its circumscribed sphere contains some point q that is not on the boundary surface, and update the triangle by three new triangles with q being a vertex. [Veltkamp 1994] considers the ratio of the circumcircle radius of f to the circumsphere radius of the corresponding tetrahedron. Hybrid sculpting [Attene and Spagnuolo 2010] identifies the exterior tetrahedra based on the magnitude of the longest edge of a boundary tetrahedron. [Peethambaran and Muthuganapathy 2015b] prioritizes removing those tetrahedra with a large circumsphere radius. However,

these procedural approaches mainly consider the shape of a single tetrahedron and lack knowledge about the overall shape. The disadvantages are two-fold: (1) they heavily depend on the point density and cannot work on low-density points, and (2) if one checks the manifoldness conditions each time a tetrahedron is deleted, it is hard to produce a high-genus model (because of this, [Boissonnat and Jean-Daniel 1984] and [Peethambaran and Muthuganapathy 2015b] can only deal with genus-0 models).

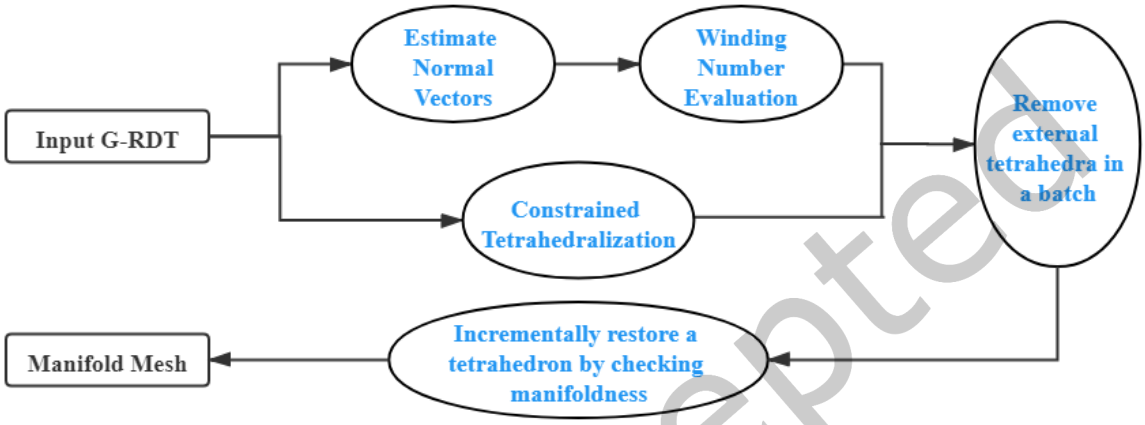


Fig. 9. The workflow of winding number based Sculpting.

Winding number based sculpting. In this paper, we use a different strategy that is aware of the overall shape, i.e., using the winding number to determine if a tetrahedron lies inside. The workflow is shown in Figure 9. First, we build tetrahedralization for the interior $\mathbf{T}$ of $\mathbf{G}$ with tetgen: $\mathbf{T} = \{T_i\}_{i=1}^{m}$, and compute the normal vector for each vertex of $\mathbf{G}$. Second, we evaluate the winding number $W(\cdot)$ for each $T_i \in \mathbf{T}$:

$$W(T_i) = \sum_{j=1}^{n} a_j \frac{(p_j - t_i) \cdot \mathbf{n}_j}{4\pi \|(p_j - t_i)\|^3}, \tag{1}$$

where t_i is the center point of T_i, $\mathbf{n}_j$ is the normal vector at p_j, n is the total number of vertices of $\mathbf{G}$, and a_j is the influence area of p_j, computed by projecting the neighboring points onto the best-fit plane and querying the area of its 2D Voronoi cell. After that, we temporarily erase

$$\mathbf{T}^{\text{out}} \subset \mathbf{T} = \{T_i \in \mathbf{T} \,|\, W(T_i) \leq \tau\}$$

from $\mathbf{T}$ ($W(T_i) \leq \tau$ implies that T_i tends to be located outside). If the surface of $\mathbf{T} \backslash \mathbf{T}^{\text{out}}$ is manifold (meeting Condition #1, #2, #3), the Sculpting step is finished. Otherwise, we identify the non-manifold vertices and the non-manifold edges and fix them. τ is set to 0.3 by default.

Classification of non-manifold issues. The non-manifold issues consist of two types: non-manifold vertices and non-manifold edges. Suppose that the boundary surface of $\mathbf{T}$ is 2-manifold, i.e., non-manifold vertices/edges do not exist. Based on connectivity (two tetrahedra are in one cluster if they share one triangle face), $\mathbf{T}$ has the following properties:

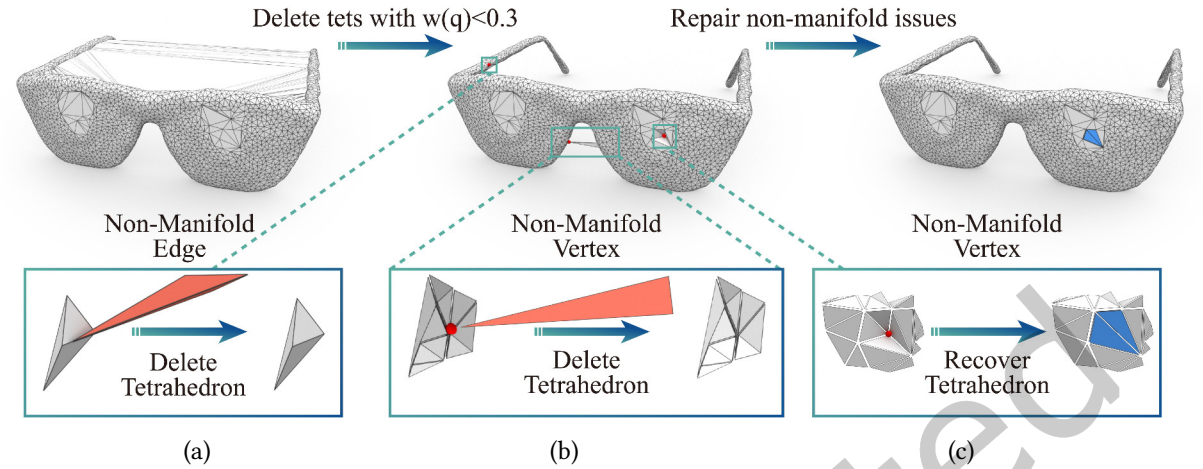


Fig. 10. Fixing non-manifold issues in the Sculpting step. (a) A non-manifold edge whose surrounding tetrahedra consist of multiple connected components (two tetrahedra are in one cluster if they share one triangle face). (b) A non-manifold vertex whose neighboring tetrahedral elements consist of multiple components based on adjacency. (c) A non-manifold vertex whose neighboring tetrahedral elements have only one component.

- The tetrahedra surrounding any edge e consist of only one connected component. Their dual is a closed fan (for an interior edge) or an open fan (for an edge on the boundary surface of $\mathbf{T}$).
- The tetrahedra neighboring to any vertex v consist of only one connected component.

Similar properties can be found in [Attene et al. 2009]. Based on this, all the possible non-manifold issues fall into the following three types (as shown in Figure 10):

Type I: A non-manifold edge whose surrounding tetrahedra cannot form an open or closed fan. See Figure 10(a).
Type II: A non-manifold vertex whose neighboring tetrahedra cannot induce a connected component. See Figure 10(b).
Type III: A non-manifold vertex whose neighboring tetrahedra are connected but the boundary triangles incident to the vertex cannot define a connected component. See Figure 10(c).

Fixing non-manifold vertices/edges. For Type I, i.e., a non-manifold edge e whose surrounding tetrahedra can be clustered into two or more components, we keep the largest cluster of tetrahedra and remove the other tetrahedra. For Type II, among the multiple components that surround a non-manifold vertex, we keep only the cluster with the largest average winding number (more possibly lying inside). For Type III, we restore the tetrahedra surrounding the non-manifold vertex (in the constrained Delaunay triangulation of the interior of $\mathbf{G}$, the dual of the tetrahedra neighboring to any vertex v is homeomorphic to a 2D disk). We can repeatedly handle these issues until all the non-manifold issues can be finally eliminated. In the final state, we extract the boundary surface of $\mathbf{T}$ as the new guiding surface.

The handling of one non-manifold issue may lead to the occurrence of another non-manifold issue although this rarely happens. To avoid an endless loop, we enforce a mandatory rule - any restored tetrahedra cannot be deleted anymore.

Lemma 5.1. *Our Sculpting strategy can guarantee that the output is a 2-manifold surface or empty with finitely many iterations, in whatever order the three types of issues are addressed.*

Proof. When Type-I or Type-II issues occur, there are some tetrahedra deleted, reducing the number of tetrahedra in **T**. When Type-III issues occur, the number of tetrahedra in **T** is increased. However, our algorithm specifies a mandatory rule - any restored tetrahedra cannot be deleted anymore, and thus any tetrahedral element cannot be deleted twice, which guarantees that our algorithm can terminate with finitely many iterations. □

At the termination, there are possibly three situations:

Situation #1: the output is a non-empty manifold mesh but different from the input guiding surface (at the beginning of the Sculpting step).

Situation #2: the output is empty since all the tetrahedra are deleted.

Situation #3: the output is the same as the input guiding surface (being a manifold surface) since all the deleted tetrahedra are restored or no one tetrahedra are deleted.

Termination condition. The purpose of the Filmsticking step is to predict the connections between points with the help of the guiding surface while the purpose of the Sculpting step is to make the resulting guiding surface closer to the underlying shape in topology and geometry. Due to the alternative execution between the Filmsticking operation and the Sculpting operation, the guiding surface becomes closer and closer to the real geometry, and the prediction of winding numbers becomes more and more accurate. Generally, there are more and more points participating in the triangulation. Therefore, we simply compare the number of points contained in the guiding surface between this iteration and the previous iteration, enforce a termination if the number of points does not increase.

6 FURTHER IMPROVEMENTS

By alternately performing the Filmsticking step and the Sculpting step, the guiding surface **G** can finally approximate the underlying shape when it remains unchanged. The algorithm works well in most cases. In this section, we propose to further improve it from two aspects: (1) how to deal with a shape with a narrow-mouthed point cloud, and (2) how to preserve feature lines as much as possible.

6.1 Nested guiding surfaces

It must be pointed out that although our algorithm works well for most point clouds, it cannot deal with a shape with an inward opening. As Figure 11(a) shows, if the inner wall of the point cloud is shaped like "V", then even without the Sculpting step, the Filmsticking step suffices to make the guiding surface stretch to the inner wall. However, when the opening of the point cloud curves inward, the guiding surface cannot stretch to the inner wall (see the top figure of Figure 11(b)) even if we alternatively run the Filmsticking step and the Sculpting step. Therefore, we have to introduce a nested guiding surface to attract the remaining points (see the bottom figure of Figure 11(c)).

We explain our insight as follows. As Figure 11(b) shows, there are still many points located in the internal space enclosed by the existing guiding surface but they cannot help estimate winding number due to lack of normal vectors. Because of this, the Sculpting step, based on winding-number estimation, cannot make the guiding surface stretch into the inner wall. If we introduce a nested guiding surface to enclose those "benchwarmers", it can better guess which tetrahedral element is inside and which is outside. As the top figure of Figure 11(c) shows, with the help of the nested guiding surface, the Sculpting step is able to fuse the two layers of guiding surfaces, push the guiding surface into the mouth and make the final guiding surface approach the real shape.

In implementation, we introduce a nested guiding surface in a conservative way, i.e., the nested guiding surface is introduced only if (1) both Filmsticking and Sculpting cannot increase the number of vertices in the guiding surface, and (2) the number of points inside the guiding surface is larger than or equal to 4. If the newly introduced nested guiding surface is able to help increase the number of helpful vertices (count the vertices of the guiding

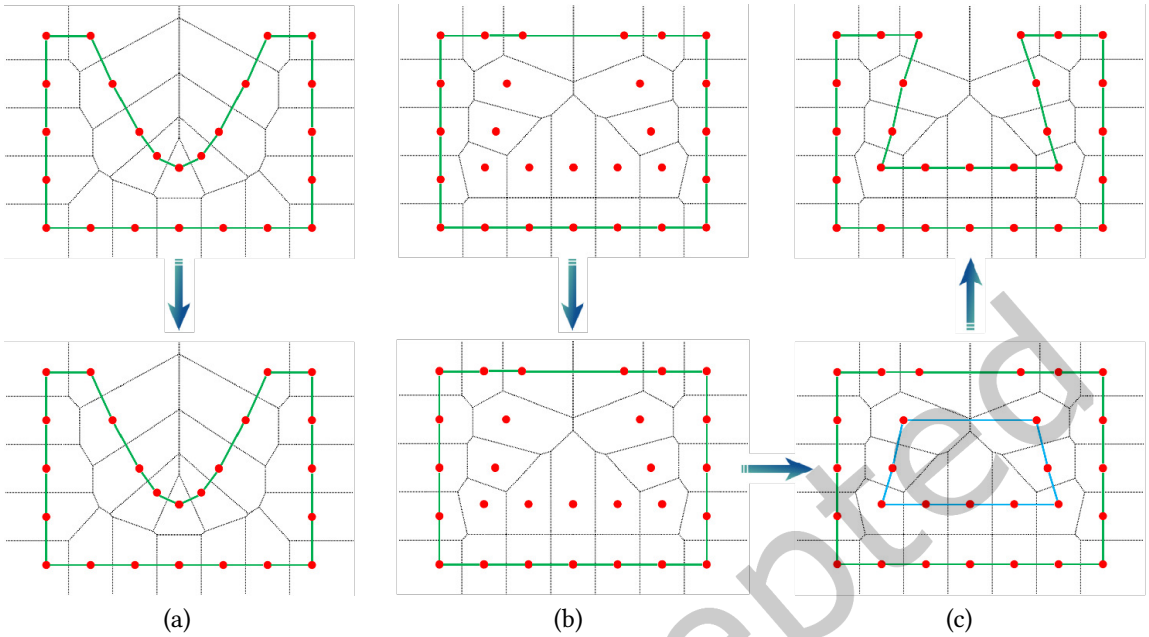


Fig. 11. The necessity of adding a nested guiding surface. (a) Even without the Sculpting step, only the Filmsticking step can report a well-shaped surface for the point cloud with a V-like inner wall. (b) For the point cloud with an inward opening, the Filmsticking step cannot make the guiding surface stretch to the inner wall (see the top figure); Even with the Sculpting step, the guiding surface remains unchanged, failing to attract the points on the inner wall (see the bottom figure). (c) Therefore, we introduce a nested guiding surface to attract the remaining points (see the bottom figure). With the help of the nested guiding surface, the Sculpting step can break through the mouth and produce a new guiding surface that can better manifest the real shape (see the top figure).

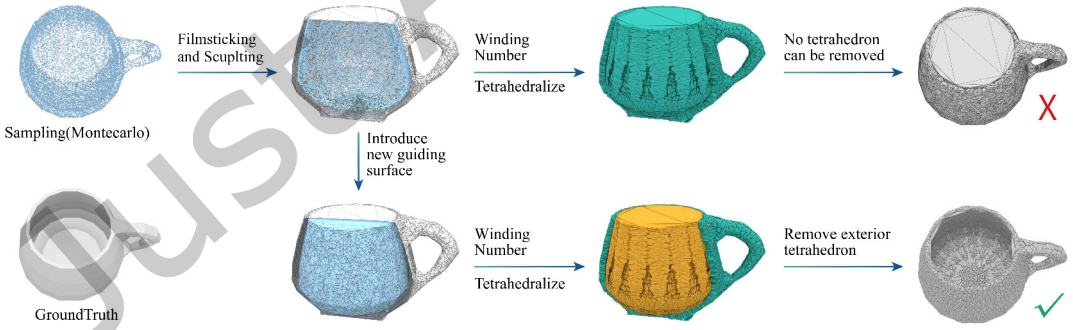


Fig. 12. If there are still points that cannot be attracted by the existing guiding surface, we introduce a new guiding surface to interpolate the remaining points.

surface at the end of the Sculpting step), we continue Filmsticking and Sculpting. Otherwise, the algorithm terminates.

As Figure 12 shows, when the first Filmsticking operation is performed, we get one guiding surface $\mathbf{G}$, but the points on the inner wall are remaining (the number of remaining points is larger than 4). Starting from a

simple spherical surface that is manifold, watertight, and encloses the remaining points, we perform another Filmsticking operation and get one more guiding surface $\mathbf{G}'$. We set $\mathbf{G}$ and $\mathbf{G}'$ to be in different orientations, facilitating the computation of winding number. It can be explained from 3D boolean operations: the volume of the underlying shape equal to $\mathbf{G}$'s interior minus $\mathbf{G}'$'s interior. In the implementation, we tetrahedralize the interior of $\mathbf{G}$ by taking each point in $\mathbf{P}$ as the tetrahedral vertex. The purpose of introducing $\mathbf{G}'$ is to help provide a normal vector for those points not contained in $\mathbf{G}$ such that the winding number can be accurately estimated.

Although $\mathbf{G}$ and $\mathbf{G}'$ are separate in connectivity, they are generally fused into one connected surface at the end of the Sculpting step. The fused guiding surface serves as the input to the next Filmsticking step. (If the input points represent the inner layer and the outer layer of a hollowed sphere, $\mathbf{G}$ and $\mathbf{G}'$ are still separate after Sculpting; In this case, $\mathbf{G}'$ does not help increase the number of vertices on $\mathbf{G}$ and thus we directly return $\mathbf{G}$ as the output.)

Remark: The nesting depth can be larger - a larger depth generally leads to fewer iterations of Filmsticking and Sculpting. But the nesting depth of 2 (two guiding surfaces $\mathbf{G}$ and $\mathbf{G}'$) is enough in practice based on our tests. First, the purpose of the nested guiding surface is to augment the point set that contributes to the triangulation - the remaining points still have a chance to participate in the triangulation in the next turn. Second, if the nesting depth is too large, the prediction of the winding number becomes less accurate, which further decreases the accuracy of Sculpting.

6.2 Feature-line alignment

Feature alignment is required especially when the input point cloud represents a CAD model with sharp creases. Therefore, we provide an *optional* step for users to achieve this purpose. As pointed out in [Stein et al. 2018] for a point on the surface, the small neighborhood of an off-feature-line point can be roughly viewed as a planar region, while the small neighborhood of an on-feature-line point can be considered as a folded planar region. Therefore, if the vertex v_i is on the flat area, the normal vectors of the incident triangles have a small deviation from v_i's normal vector. We evaluate the deviation by

$$\xi_1(v_i) = \sum_{f_j} \theta_j \|\mathbf{n}_j - \overline{\mathbf{n}_i}\|^2, \tag{2}$$

where $\overline{\mathbf{n}_i}$ is the average normal at v_i, f_j is a triangle incident to v_i, $\mathbf{n}_j$ is the normal vector at f_j, and θ_j is the v_i-incident angle in f_j. If v_i is located near a feature line, we hope the following score is as small as possible [Stein et al. 2018]:

$$\xi_2(v_i) = \sum_{e_j} \frac{\theta_j^1 + \theta_j^2}{2} \| < \overline{\mathbf{n}_i}, \mathbf{n}_j^1 > - < \overline{\mathbf{n}_i}, \mathbf{n}_j^2 > \|^2, \tag{3}$$

where e_j is a v_i-incident edge, θ_j^1, θ_j^2 are the v_i-incident two angles in the two triangles incident to e_j, and $\mathbf{n}_j^1, \mathbf{n}_j^2$ are respectively the normal vectors of the two triangles incident to e_j.

Finally, we find the optimal edge configuration that can minimize the overall score

$$\sum_i \xi_1(v_i)\xi_2(v_i). \tag{4}$$

In the implementation, rather than borrow an existing numerical optimization solver, we introduce a priority queue to maintain the order in which the edges are processed. Obviously, one edge flipping operation brings a change to four vertices, as Figure 13 shows. For each edge, its priority is measured by the overall decrease of the scores of the related 4 vertices. Each edge flipping operation may cause a score decrease. Throughout the algorithm, the overall score keeps decreasing, which implies that the algorithm can terminate after finitely many edge flipping operations. See an example for feature alignment edge flipping in Figure 13. In addition, we

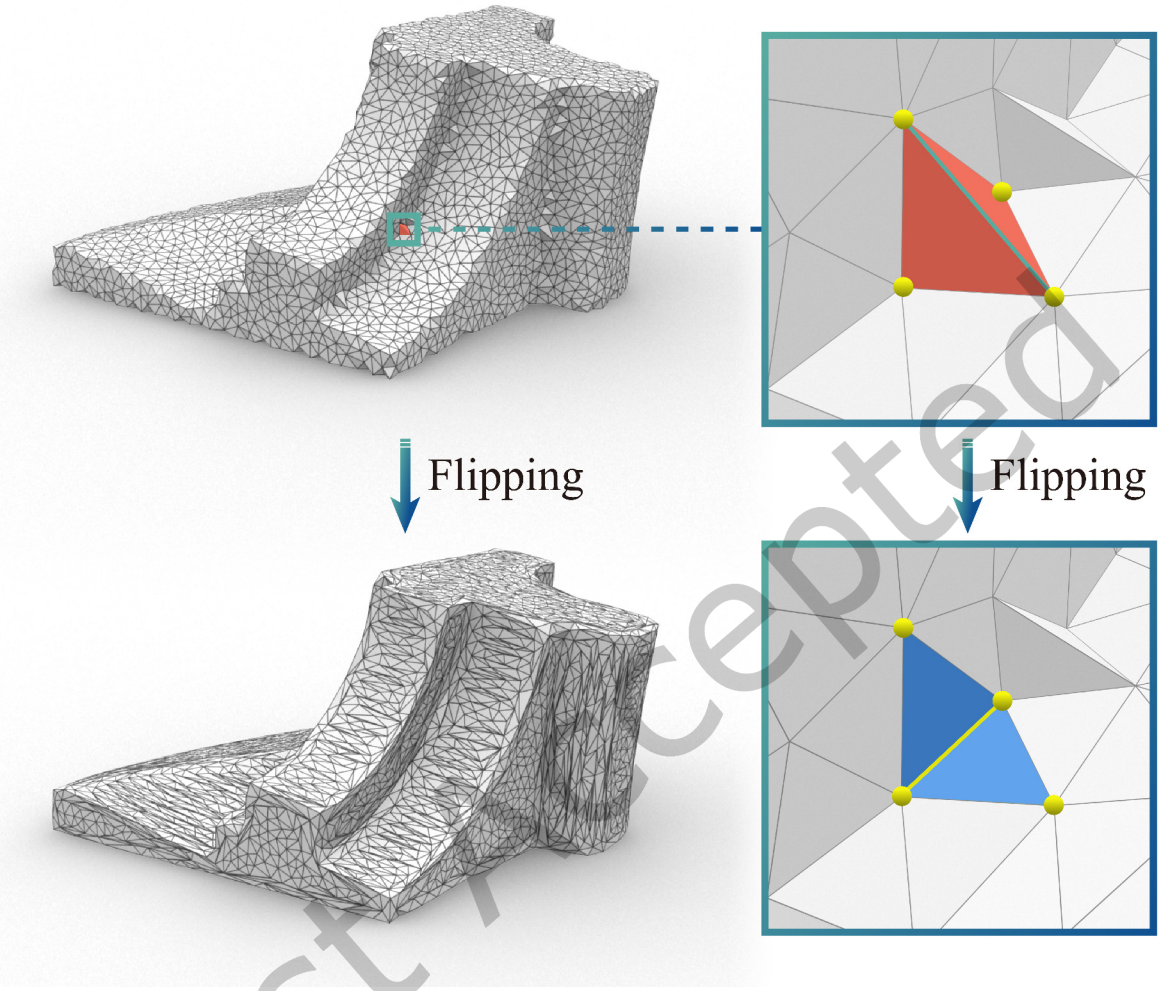


Fig. 13. Edge flipping for feature alignment. Each edge flipping operation makes the overall score (see Eq. (4)) keep decreasing. After finitely many flipping operations, our algorithm reports a feature-alignment connection configuration (still a watertight manifold).

suppress the edge flipping operation if it leads to non-manifold edges/vertices, degenerate triangles, or surface self-intersection.

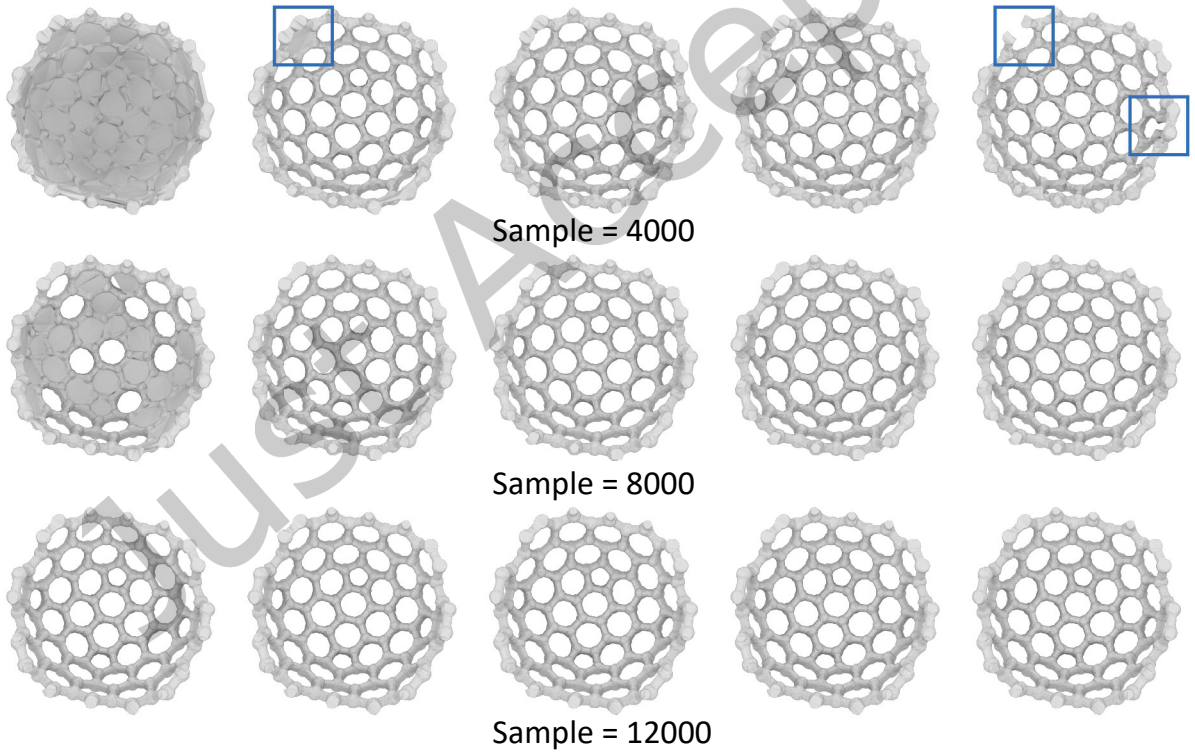
7 EXPERIMENTS

In this section, we evaluate the proposed approach on synthetic and real scanned point clouds. The experiments were all conducted on a PC with Intel(R) Core(TM) i9-9900k CPU 3.60 GHz and 32 GB memory and RTX 3090.

7.1 Settings

Existing approaches. We compare our approach with 7 existing surface reconstruction approaches to demonstrate the superiority of our approach. Some are interpolation based, e.g., Greedy [Cohen-Steiner and Da 2004], Ball Pivoting (BP) [Bernardini et al. 1999], and Scale Space (SS) [Digne et al. 2011]. Some are approximation based, such as Screened Poisson (SPR) [Kazhdan and Hoppe 2013], Point2Mesh (P2M) [Hanocka et al. 2020], Point2Surface (P2S) [Erler et al. 2020]. In fact, P2M and P2S are learning-based methods. We also include PointTriNet (PTN) [Sharp and Ovsjanikov 2020] for comparison, which is learning-based but interpolates the given point cloud. It's worth noting that Screened Poisson require normals of the point cloud. We employ [Xu et al. 2018] for estimating normal vectors to facilitate the execution of screened Poisson. For our algorithm, the switch of feature alignment edge flipping is turned off if without any specification. Furthermore, we also discuss the difference between our method and [Khoury and Shewchuk 2016].

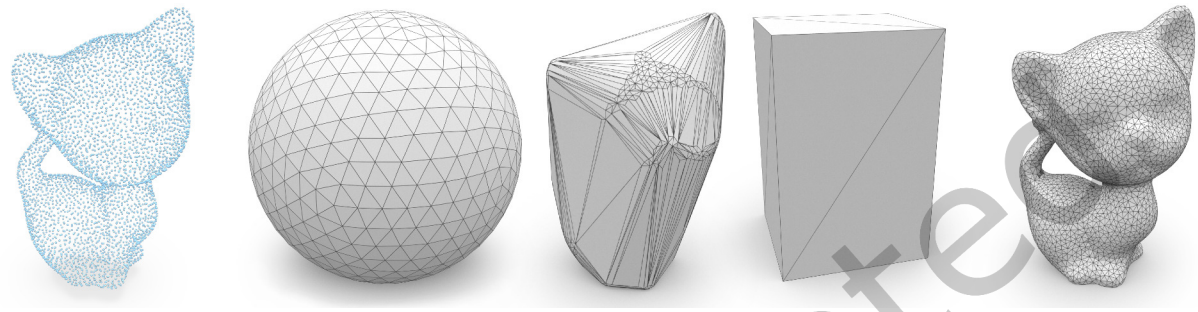
Datasets. We used diverse datasets for the test. The first kind of data is obtained by direct sampling on meshes in Thingi10K [Zhou and Jacobson 2016] and D-Faust [Bogo et al. 2017]. The second kind is simulated scan by BlenSor [Gschwandtner et al. 2011] on models from Thingi10K. The last kind is real scanned data, some are courtesy of LGG, EPFL (http://lgg.epfl.ch/statues.php) and large-size dataset [Aanæs et al. 2016] while some are obtained by the Go!SCAN 3D G2 white light scanner.



(a) w = 0.1 (b) w = 0.2 (c) w = 0.3 (d) w = 0.4 (e) w = 0.5

Fig. 14. A smaller winding-number tolerance tends to encourage a larger interior space; See the highlighted windows in the first row. Furthermore, the tolerance becomes more insensitive for denser point data. We set the tolerance to 0.3 by default.

Parameter setting. Our algorithm includes a single parameter, i.e., the winding-number threshold w, which is used to distinguish the interior volume from the exterior volume. Figure 14 shows the influence of w, where the number of samples ranges from 4K to 12K and w is set to $0.1, 0.2, 0.3, 0.4, 0.5$ respectively. It can be seen that in the case of sparse sampling (4k), if w is too small or too large, it is easy to confuse interior and exterior. We set w to be 0.3 by default in our experiments. Furthermore, we observe that when the number of points is large, the choice of the winding-number threshold becomes insensitive.



(a) Input (b) Initial guiding surface (c) Result

Fig. 15. We test a bounding sphere, a convex hull, and a bounding box respectively as the initial guiding surface and obtain the same reconstructed result.

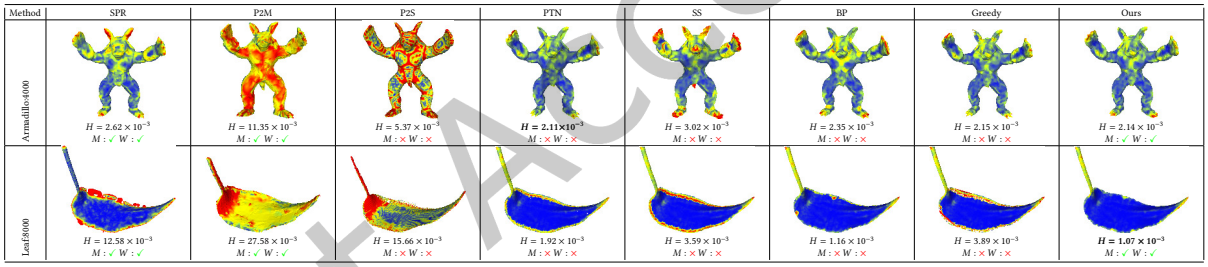


Table 1. Accuracy statistics. H: reconstruction error. M: manifold. W: watertight. Warm color represents large deviation from the ground truth.

Guiding surface initialization. As Figure 15 shows, we test a bounding sphere, a convex hull, and a bounding box respectively as the initial guiding surface and obtain the same reconstructed result. Furthermore, even if the initial guiding surface is inside the volume or stays away from the given point cloud, the result remains unchanged but may need more iterations.

(a) Input (b) SPR (c) P2M (d) P2S (e) PTN (f) SS (g) BP (h) Greedy (i) Ours

Fig. 16. Constructing a thin-plate surface from sparse point cloud (2K points).

Indicators for evaluating a reconstruction approach. We shall evaluate our reconstruction approach from the following ten aspects. The indicators include:

- *Whether to guarantee a watertight manifold output.*
- *Reconstruction accuracy against the ground truth.*
- *The ability to deal with sparse and irregularly distributed points.*
- *The ability to deal with high-genus tubular structures.*
- *The ability to deal with thin-plate models.*
- *The ability to deal with feature lines and sharp features.*
- *The ability to recover geometric details.*
- *Triangulation quality.*
- *Robustness to noise, outliers, and missing data.*
- *Run-time performance.*

7.2 Accuracy

Let $\mathbf{M}$ be the ground truth mesh and $\mathbf{M}'$ the reconstructed one. The symmetric Hausdorff distance between $\mathbf{M}$ and $\mathbf{M}'$ can be defined as follows:

$$d(\mathbf{M}, \mathbf{M}') = \max\left(\frac{\sum_{i=1}^{n} A(v_i) d(v_i, \mathbf{M}')}{\sum_{i=1}^{n} A(v_i)}, \frac{\sum_{i=1}^{n'} A(v_i') d(v_i', \mathbf{M})}{\sum_{i=1}^{n'} A(v_i')}\right),$$

where n, n' are respectively the number of vertices contained in $\mathbf{M}$ and $\mathbf{M}'$, and $A(\cdot)$ is to take the influence area of a vertex.

In Table 1, we compare our algorithm with the other 7 approaches on two simulated scans of the Armadillo model (4K) and the Leaf model (8K). Generally speaking, the interpolation based approaches (PTN, SS, BP, Greedy, and ours) are more accurate than the approximation based approaches. Our approach, also interpolation based, is in the first level as far as the accuracy is concerned. For the Armadillo model, PTN is comparable to ours in terms of accuracy but its output is far from being a watertight manifold. For the Leaf model, our method has obvious advantages over the other approaches. SPR heavily depends on the normal vectors. However, it is hard to compute reliable normal vectors on a thin plate model, making SPR difficult to yield a faithful reconstruction result. P2M uses MeshCNN to predict the displacement for an initial mesh and can gradually approach the underlying shape. In spite of being an explicit approach, P2M cannot interpolate the given point cloud, which accounts for the fact that P2M is less accurate than the interpolation based approaches. P2S directly predicts the signed distance at a query point from the latent codes. It has good generalizability and adapts to different noise scales. However, it cannot guarantee normal consistency and may produce bumpy surfaces with ripples. BP triangulate every three points that fit into a ball of radius r without containing any further point inside the ball, but it may cause misconnections of points if the distance between neighboring points is larger than the thickness of the model. Greedy starts from a seed facet and augments the surface by adding Delaunay triangles one by one. However, Greedy is hard to distinguish the points on opposite sides of a thin plate model.

7.3 Manifoldness

In Table 1, we use "M" and "W" respectively to indicate whether the result is manifold or watertight. The requirement of being manifold and watertight seems trivial for those approximation-based approaches that extract the zero level set of a scalar field. However, the requirement becomes extremely hard for the interpolation-based approaches due to the inherent terrible combinatorial complexity. As reported in Table 1, only ours can guarantee the manifoldness among the interpolation-based approaches. For the Leaf model, PTN produces 196 non-manifold edges and 25 non-manifold vertices and 141 open-boundary edges, SS produces 8050 non-manifold

edges and 3 non-manifold vertices, BP produces 63 non-manifold vertices and 274 open-boundary edges, and Greedy produces 157 open-boundary edges. It is worth mentioning that Greedy uses a greedy triangle selection strategy to guess the next "best" connection, but the best guess depends on the local positional configuration of points, lacking the knowledge about the global shape. Therefore, Greedy cannot ensure a closed mesh if the point cloud has a low quality (e.g. far from the LFS sampling condition). To summarize, our algorithm accurately interpolates the point cloud and can guarantee a watertight manifold, which distinguishes itself from other approaches.

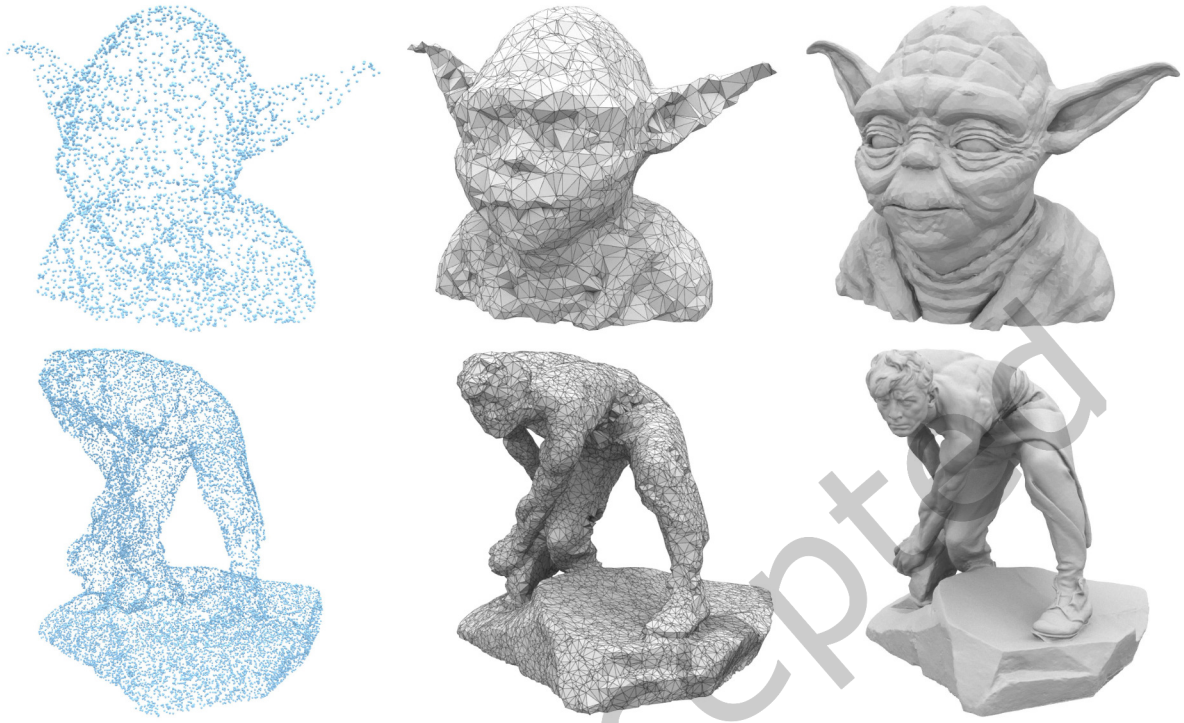
7.4 Irregular and sparse distribution of point cloud

A point cloud is said to have a regular distribution if there is an almost even gap between points. It's known that both CVT and blue-noise sampling can generate a set of points with a regular distribution. The problem of surface reconstruction becomes difficult if the input points have an irregular and sparse distribution.

Sparse point cloud. Figure 16 shows a point cloud sampled from the Cup model with a thin wall. It includes 2K points and is generated by Blensor. Generally, a set of points is said to be sparse if the gap between adjacent points is bigger than the thickness of the wall. In this case, the influence of the points on one side of the wall easily penetrates the other side, thus producing misconnections of points. Our algorithm alternately performs the Filmsticking step and the Sculpting step. Its spirit is to find the simplest connection configuration among a huge number of possibilities. Since the points on the inner wall of the Cup model (Figure 16) are too sparse, even if the normal vectors are very accurate, SPR still fails to get a faithful result due to the lack of a reliable gradient field. P2M has two disadvantages in dealing with sparse point clouds. First, it is hard to accurately predict the position for each moveable vertex. Second, it does not have a high triangulation quality. P2S and PTN cannot control the overall topology of the target model and are likely to produce unwanted topological holes. SS requires a pair of parameters, i.e., the radius of the neighborhood ball and the number iterations of increasing the scale, to define a good indication of which points are near each other. Even with fine-tuned parameters, the inner wall and the outer wall of the Cup model cannot be separated due to the fact that SS operates locally. It's also hard to tune the ball-radius parameter for BP. Greedy seems to be able to get a good connection but the resulting mesh contains gaps.

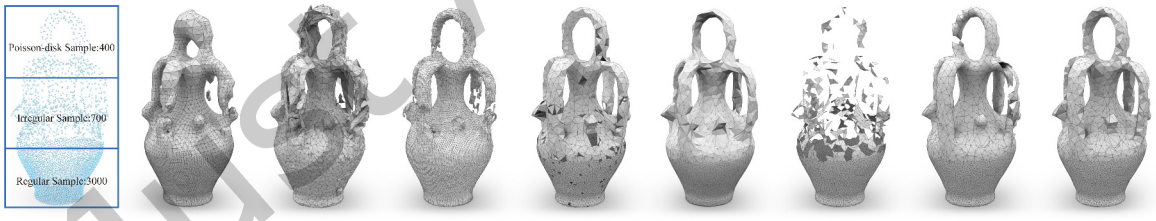
Irregular distribution. In Figure 17, we generate 4K, 20K irregularly distributed points on the 2 models using Monte-Carlo sampling. When the point distribution becomes irregular, there exists ambiguity on how to predict the desirable connection configuration, it is hard to use a local geometric rule to determine the connections between points. In fact, both the Filmsticking step and the Sculpting step of our algorithm are to seek for a connected watertight manifold surface, which is prior knowledge about how people perceive a 3D shape.

Diverse point distributions. In Figure 18(a), we synthesize a sampling result by employing Poisson disk, Monte-Carlo sampling, and regular recursive sampling in the upper, middle, and bottom parts, and the point density varies much from top to bottom. Since the Vase model contains thin tubes/plates, we use the model to test various reconstruction approaches to see if they can handle a point cloud with the sparse, irregular, and non-uniform distribution. It can be seen that our algorithm can recover the underlying geometry for this challenging case and outputs a watertight manifold mesh. However, the results yielded by the other algorithms have various defects. Due to the poor quality of point distribution, it is hard to estimate the normal vectors very accurately, making SPR produce a bumpy surface with broken handles. P2M takes SPR's resulting surface as the input and thus inherits such defects. The sign propagation mechanism of P2S is sensitive to the distribution of input points and thus cannot get a good result for a poor-quality point cloud. PTN suffers from point sparsity and easily leads to a non-manifold and non-watertight output. It's hard for SS to tune the scale parameter. One has to use a large scale to adapt sparse point clouds. However, some points are missed on the final output mesh, as Figure 18(f) shows.



(a) Input (b) Ours (c) GT

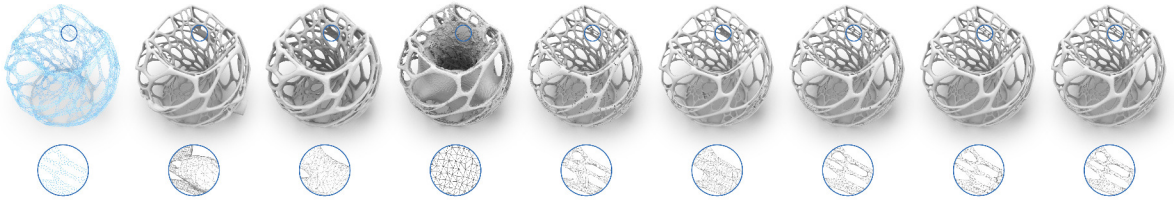
Fig. 17. We generate irregularly distributed points using Monte-Carlo sampling. From top to bottom: the numbers of samples are respectively 4K, 20K. From left to right: sampled point cloud, our reconstructed results, ground-truth.



(a) Input (b) SPR (c) P2M (d) P2S (e) PTN (f) SS (g) BP (h) Greedy (i) Ours

Fig. 18. We synthesize a sampling result by employing Poisson disk, Monte-Carlo sampling, and regular recursive sampling in the upper, middle, and bottom parts. Most of the existing approaches fail due to the poor-quality input whereas our algorithm works well.

BP requires a global ball-radius parameter but it is hard to find a suitable value on the point cloud with unequal point densities. Greedy cannot predict a correct connection between points at the thin parts, causing broken handles.



(a) Input (b) SPR (c) P2M (d) P2S (e) PTN (f) SS (g) BP (h) Greedy (i) Ours

Fig. 19. Comparing the ability to reconstruct the complicated tubular network.

7.5 High-genus, thin plates, sharp features, and geometric details

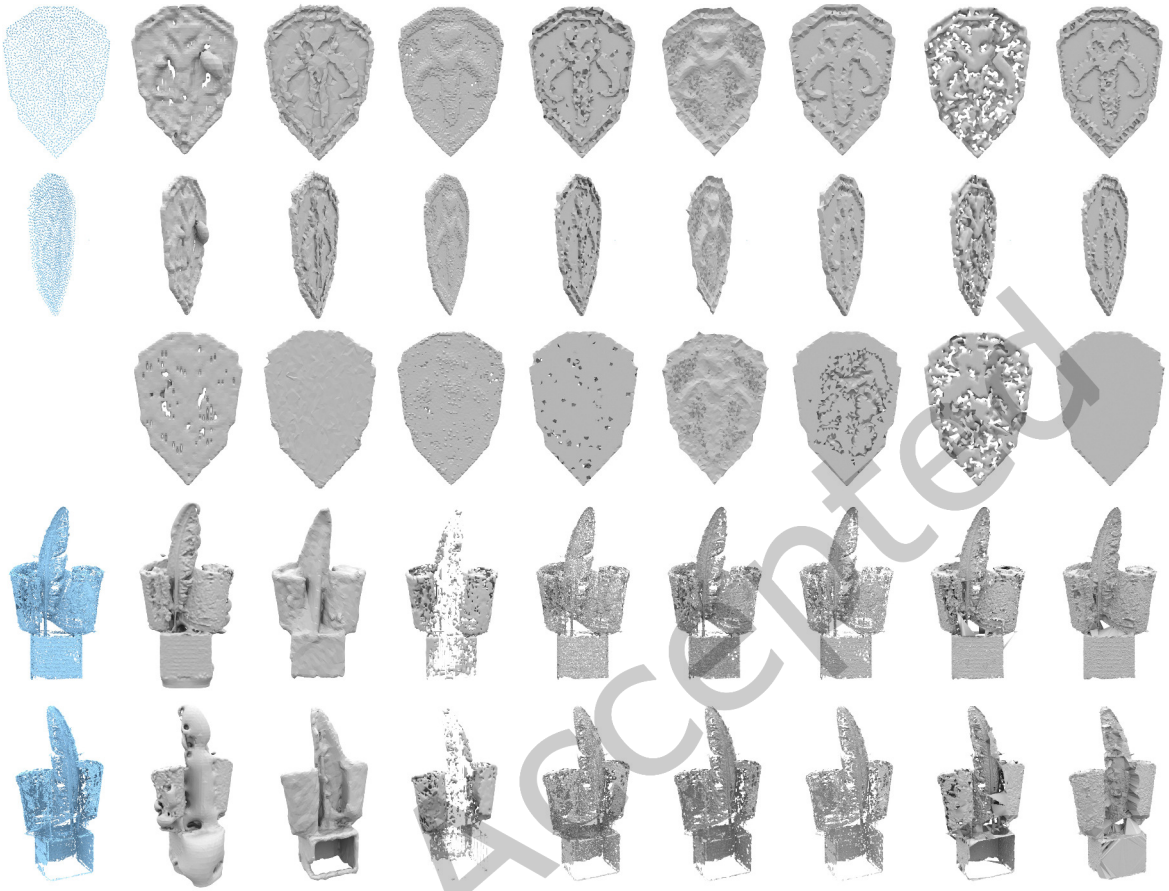
High-genus. In Figure 19, we test various algorithms on a high-genus model from the Thingi10K dataset. We sample 50K points using the Poisson disk method. Those implicit reconstruction approaches easily produce unwanted bumps while the existing interpolation based approaches easily produce misconnections between gaps or broken branches at super-thin parts. By contrast, our algorithm interpolates the point cloud, guarantees a watertight manifold, and well approximates the real geometry.

Thin plates, geometric details and missing data. In Figure 20, we use two models to test various algorithms to see if they can handle thin plates, geometric details, and missing data. The tests on the Shield model (3K points, simulated scan by Blensor) show that our algorithm can distinguish the front side from the backside and preserve geometric details as well, unlike SPR, P2S, PTN, and Greedy that generate a large number of small unexpected topological holes. The tests on the Gogol model (a real scan courtesy of LGG, EPFL), show that most of the existing algorithms fail to deal with data missing. By contrast, our algorithm is able to bridge the missing parts, yielding a watertight manifold mesh, which is due to the fact that our algorithm suppresses unnecessary topological holes as far as possible. See Figure 21 for more data missing examples. We must point out that if data missing is too severe such that the underlying shape represented by the point cloud is far from being a closed surface, our algorithm may report an empty result as the output. For example, we take samples from a spherical cap with a height of $R/2$ (R is the radius), the output is a closed mesh; But when the cap height amounts to $R/4$, the output becomes empty. We further use Figure 23 to show a gallery of reconstructed results to give an in-depth comparison between the existing approaches and ours. Some of the point clouds (the top three rows of Figure 23) are simulated scans using Blensor while some (the bottom two rows of Figure 23) are real scans courtesy of LGG and EPFL.

Preserving feature lines. At least for CAD models, the near-feature regions and the off-feature regions should be treated differently. One has to encourage feature alignment at the cost of a possible decrease in the meshing quality. Recall that our algorithm includes an optional step of edge flipping to encourage feature alignment around the near-feature area while suppressing fluctuation in the off-feature area. Most of the existing algorithms do not take feature alignment into account. We activate the feature alignment on the Gear (No. 3 model in Figure 1) and Fandisk (Figure 13) model to achieve feature aligned triangulations.

7.6 Triangulation quality

For approximation-based approaches, there is enough freedom to optimize vertex placement such that the resulting mesh has a high triangulation quality. For interpolation-based approaches, however, the triangulation quality mainly depends on the original point distribution, although there does limited freedom to flip the edges



(a) Input (b) SPR (c) P2M (c) P2S (d) PTN (e) SS (f) BP (g) Greedy (h) Ours

Fig. 20. The thin-plate model has many geometric details on the front while remaining flat on the back. We give three separate views for each result. The Gogol model, courtesy of LGG and EPFL, has severe data missing. One of the nice features of our algorithm lies in that it is able to bridge the missing parts, yielding a watertight manifold mesh.

(like Delaunay triangulation). We use the following commonly used formula to measure the quality of a triangle t:

$$Q(t) = \frac{6}{\sqrt{3}} \frac{S}{pl}, \tag{5}$$

where S, p, l are respectively the triangle area, half of the perimeter, and the length of the longest edge. $Q(t) = 1$ when t is a regular triangle and $Q(t) < 1$ otherwise. Let $\theta(t)$ denote the minima angle of t. By visiting all the triangles, we keep Q_i and θ_i for the i-th triangle. In this way, we obtain the average values Q_{ave}, θ_{ave}, as well as the worst values Q_{min}, θ_{min}. From the statistics shown in Figure 22, our algorithm has a conspicuous advantage over the other algorithms. The rationale behind this lies in that our generated mesh is restricted Delaunay. We must point out that the Greedy method is also based on Delaunay, thus producing equally good triangulation quality.

Fig. 21. Our method is able to deal with data missing. From left to right: simulated scans using Blensor, real scan using 3D Laser Scanner, real scan courtesy of LGG and EPFL, a large-size real scan point cloud [Aanæs et al. 2016].

7.7 Real scans

For real scans, there are various defects such as noise, outliers, and data missing. In the situation of severe noise, it is hard to accurately estimate normal directions, which greatly diminishes the quality of the underlying scalar field for those implicit surface reconstruction approaches. Our algorithm, however, does not require the input of normal vectors. It gradually infers the real normal vectors while refining the guiding surface. The prediction of normal vectors may become inaccurate when severe noise exists, but it does not have a significant influence on the computation of winding number. In addition, the tetrahedral elements incident to outliers (or noisy points at a large distance to the underlying surface) are identified as in the exterior and thus are deleted in the Sculpting step; See Figure 24. At the same time, for this example, the data is missing on the top of the head and at the nose part. SPR naturally fills the region of missing data with a smoothly interpolated surface. P2M also has the feature of hole filling due to the manifold and watertight prior. P2S and PTN, in spite of using soft penalties to increase the ability to produce a manifold and watertight surface, are still weak in handling the defect of data missing. SS and BP have to tune a parameter to adapt large gaps between points. The gap cannot be filled if there is a large portion of data missing. Greedy fills the empty regions with large triangles, but it can neither guarantee a manifold nor deal with outliers. Our algorithm is able to handle diverse defects at the same time.

Accuracy statistics on real scans. We used the Go!SCAN 3D G2 white light scanner to obtain 8 real scans. Each noisy scan contains about 10-100K points. The reconstructed models are visualized in Figure 25. It can be seen that our algorithm can reconstruct the geometrically challenging objects with high fidelity. It is worth noting that there are some points on the ground included in the given point clouds. Our algorithm has a nice feature of automatically filtering out the points located on the ground base, which is due to the following interesting observation: during the iterations, the dangling tetrahedra are gradually deleted and thus the tetrahedral elements contributed by the ground points are isolated from the main body. Our algorithm keeps only the largest component

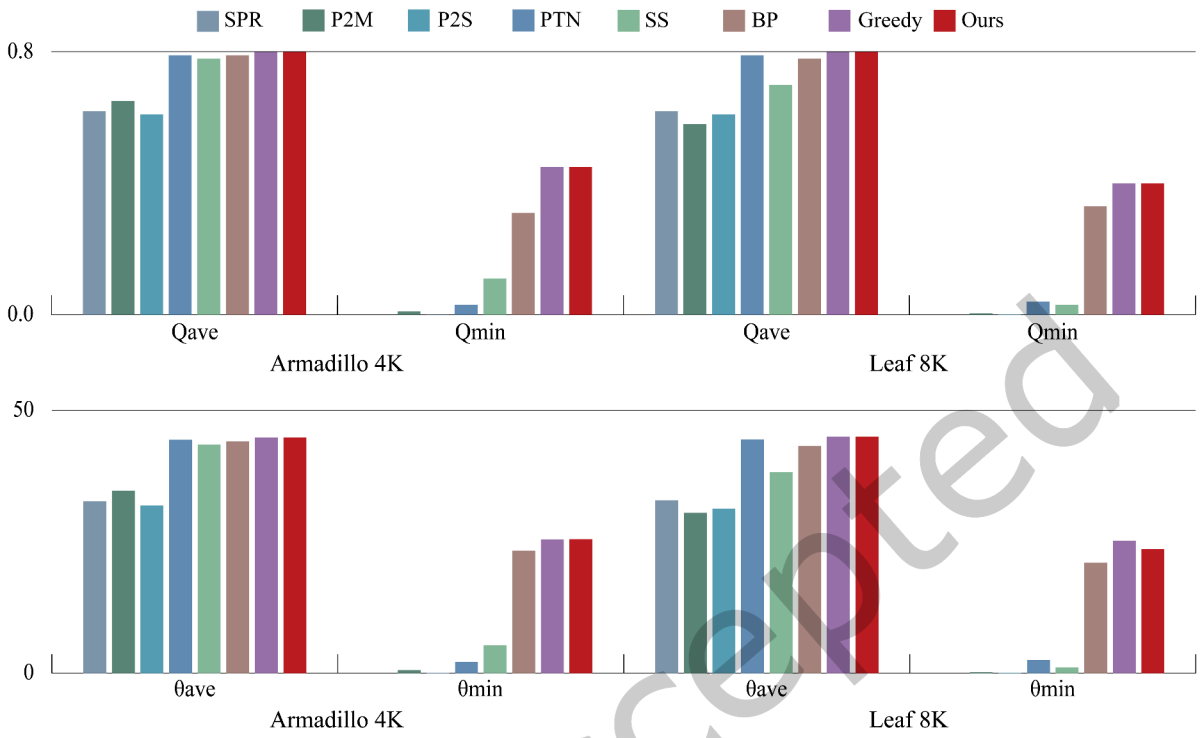
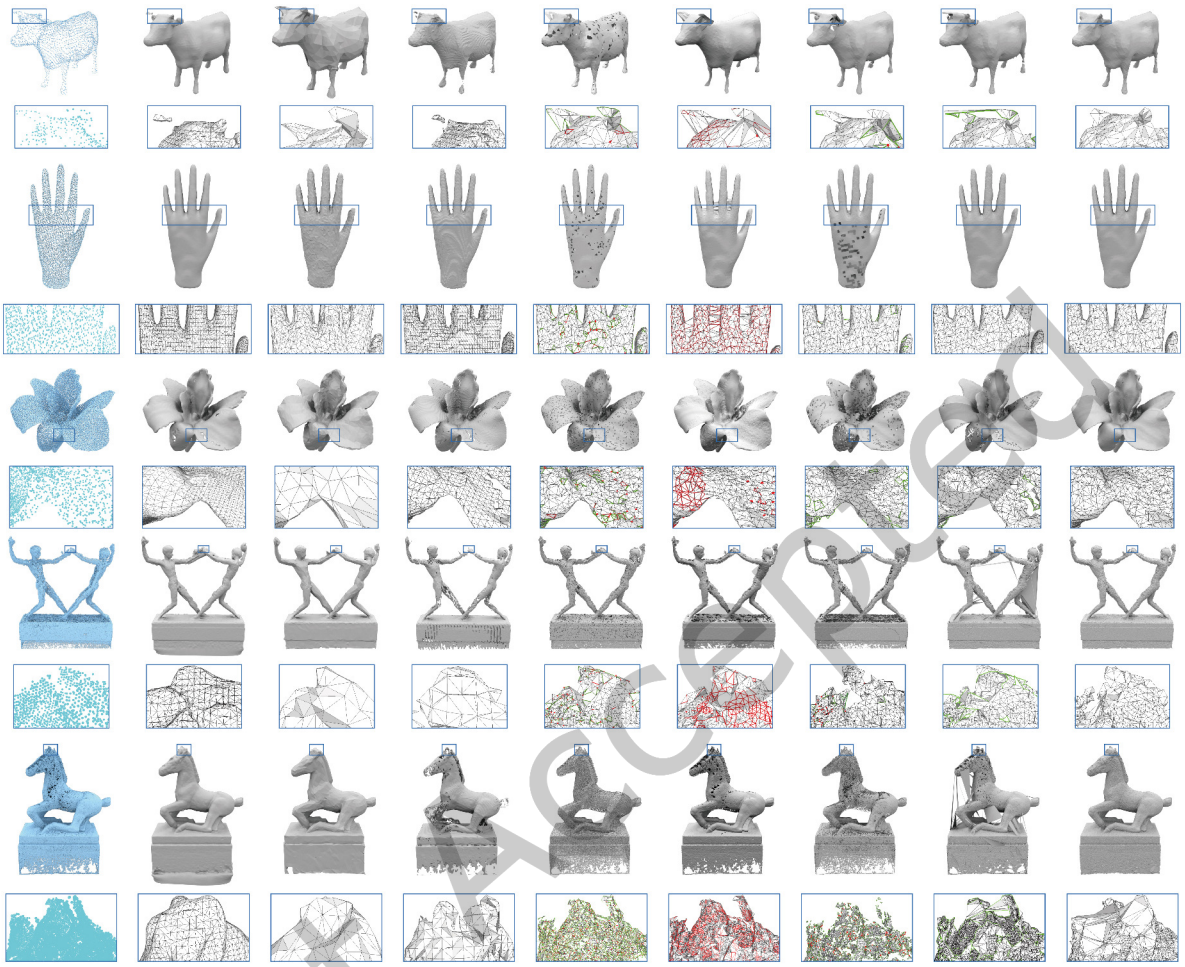


Fig. 22. Triangle quality statistics on the Armadillo model and the leaf model.

and thus cleverly avoids the ground points dirtying the main-body surface. We plot the accuracy statistics of the above-mentioned 7 approaches, as well as ours, in Figure 26. The horizontal axis is the Hausdorff error between the original model and the reconstructed counterpart while the vertical axis denotes the run-time performance. More comparisons on real scans are available in the supplemental material.

7.8 Run-time performance statistics and analysis

Suppose that the input point cloud $\mathbf{P}$ contains n points and finally produces a triangle mesh of about $2n$ triangle faces. For efficient query of the distance from a point to $\mathbf{P}$, we pre-build a kd-tree of $\mathbf{P}$ in $O(n \log n)$ time [Arya and Mount 1997] and pre-compute the Delaunay triangulation w.r.t. $\mathbf{P}$ in $O(n \log n)$ time in general condition [Attali et al. 2003]. Throughout the algorithm, the guiding surface $\mathbf{G}$ has a complexity of not exceeding $2n$ faces. Our algorithm generally requires only a few iterations, typically 5 to 10. During each iteration, $1 \sim 3$ RVD decompositions and $1 \sim 2$ Sculpting operations are needed. Considering that either computing RVD once on $\mathbf{G}$ or Sculpting the interior of $\mathbf{G}$ once requires about $O(n)$ time, the empirical time complexity can be estimated by $O(n \log n) + kO(n)$, where k equals to the number of computing RVDs plus the number of Sculpting. Empirical evidence shows that k is smaller than 18. The timing of processing the models in Figure 23 is listed in Table 2. It's worth noting that if the number of points is too large, both the Filmsticking step and the Sculpting will be very time-consuming. Therefore, when the number of point clouds is over 100K, we down-sample the point cloud to 100K that is still dense enough to encode the rough shape. After reconstructing a surface for the 100K points, we



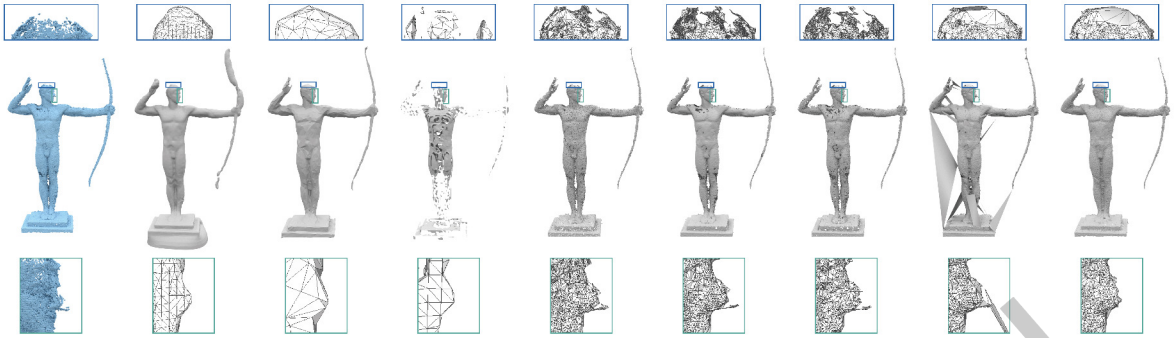
(a) Input (b) SPR (c) P2M (c) P2S (d) PTN (e) SS (f) BP (g) Greedy (h) Ours

Fig. 23. Comparing the reconstruction approaches in terms of the ability to deal with high-genus topology, sharp features, and geometric details. Red dots, red edges, and greed edges respectively denote non-manifold vertices, non-manifold edges, and boundary edges. Cattle(4K), Hand(8K), Flower(35K) are simulated scans using Blensor. Kids(200K) and Horse(1200K) are real scans courtesy of LGG and EPFL.

run the Filmsticking step once to pull all the points onto the surface. In this way, the computation gets a great speed-up for a large-size point cloud.

7.9 Tests on multiple datasets

Thingi10K. We randomly selected 200 models from Thingi10K [Zhou and Jacobson 2016] and generated 10K points by Blensor. We run our algorithm on the simulated scans and no failure case was found - all the outputs are watertight manifold. The average Hausdorff error for the 200 models is 0.97×10^{-3} (the whole model is normalized to [0,1]×[0,1]×[0,1]). The average running time is 4.76 seconds. More results can be found in the supplemental material.



(a) Input (b) SPR (c) P2M (c) P2S (d) PTN (e) SS (f) BP (g) Greedy (h) Ours

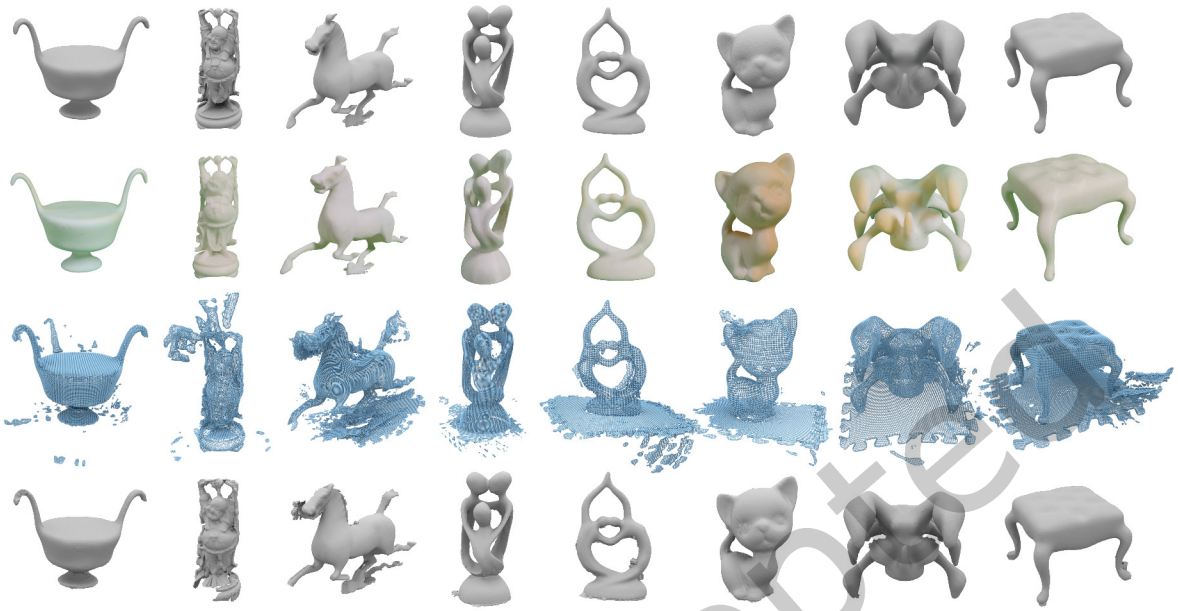
Fig. 24. Test robustness to noise, outliers, and missing data. Our algorithm is much better than the other 7 methods. Courtesy of LGG and EPFL.

Method	Cattle:4K	Hand:8K	Flower:35K	Kids:200k	Horse:1200K
SPR	0.682	1.537	2.646	5.463	10.825
P2M	3451.982	3293.732	3384.013	4924.681	7350.864
P2S	317.771	333.139	363.030	372.048	546.837
PTN	12.664	31.664	130.448	4007.943	27227.101
SS	0.169	0.437	1.773	8.674	50.814
BP	0.079	0.142	2.426	35.089	4374.711
Greedy	0.088	0.243	0.97	9.468	119.44
Ours	0.793	2.827	17.863	78.755	48.783+47

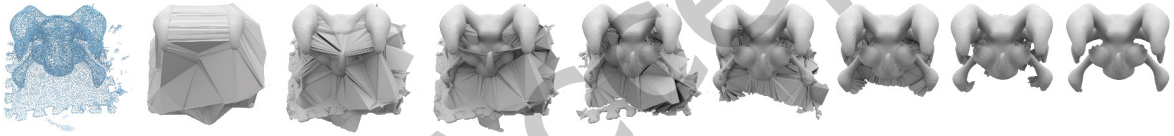
Table 2. Run-time performance comparison. For the Horse model, we down-sample the original point cloud from 1200K points to 100K points. After reconstructing a surface for the 100K points, we run the Filmsticking step once to pull all the points onto the surface.

Large-size point clouds. There is a dataset [Aanæs et al. 2016] that contains large-size point clouds (scanned from 40 real objects). It is necessary to test if our algorithm can handle large-size data. For each model, we select a single view stereo scan for the test, whose size ranges from 1000K to 4000K. If we directly run our algorithm on the large-size point cloud, it needs about 3.5 hours to finish the reconstruction. Part of the results is shown in Figure 27. More results can be found in the supplemental material. As mentioned above, we can down-sample the original point cloud to 100K points, and attract all the points to the guiding surface by one Filmsticking operation. In this way, the total running time can be reduced to about one hour but achieves almost the same result. Our algorithm, in its current form, is unoptimized and has a great potential to be further speeded up.

Human data. We also tested our algorithm on the human dataset [Bogo et al. 2017]. We used Monte-Carlo sampling to generate 40K points with an irregular distribution. Part of the results is visualized in Figure 28. The average Hausdorff distance is 0.72×10^{-3} for the dataset. More results can be found in the supplemental material.



(a) Our result for real scans.



(b) Process of our algorithm.

Fig. 25. (a) Reconstruction from real scans. The first row shows the ground truth digital models. The second row shows the snapshots of the 3D-printed objects. The third row shows the real scans by the Go!SCAN 3D G2 white light scanner. The last row shows our reconstructed results. (b) By alternately performing the Filmsticking step and the Sculpting step, our algorithm gradually eliminates noise.

7.10 Comparison between our method and [Khoury and Shewchuk 2016]

The contributions of [Khoury and Shewchuk 2016] are mainly on the theoretical side. It proves that if the points, sampled from the base surface $\mathbf{G}$, are sufficiently dense, then the dual of $\mathbf{G}$-RVD gives another manifold triangulation of $\mathbf{G}$. In implementation, [Khoury and Shewchuk 2016] suggests directly computing all the triangles that might be in a fixed-point triangulation, but it still heavily depends on the point density. To our best knowledge, nearly all the existing interpolation based reconstruction approaches depend on the point density. The main contribution of our paper lies in eliminating the unreasonable requirement. In Figure 29, we test [Khoury and Shewchuk 2016] by tuning the sampling density. In the case of 500 sample points, their algorithm misses most of target triangles. If we increase the number of sample points to 7000, there are still some triangles missing, which is due to the fact that it is hard to meet the LFS sampling condition on a thin-sheet model. More results can be found in the supplemental material.

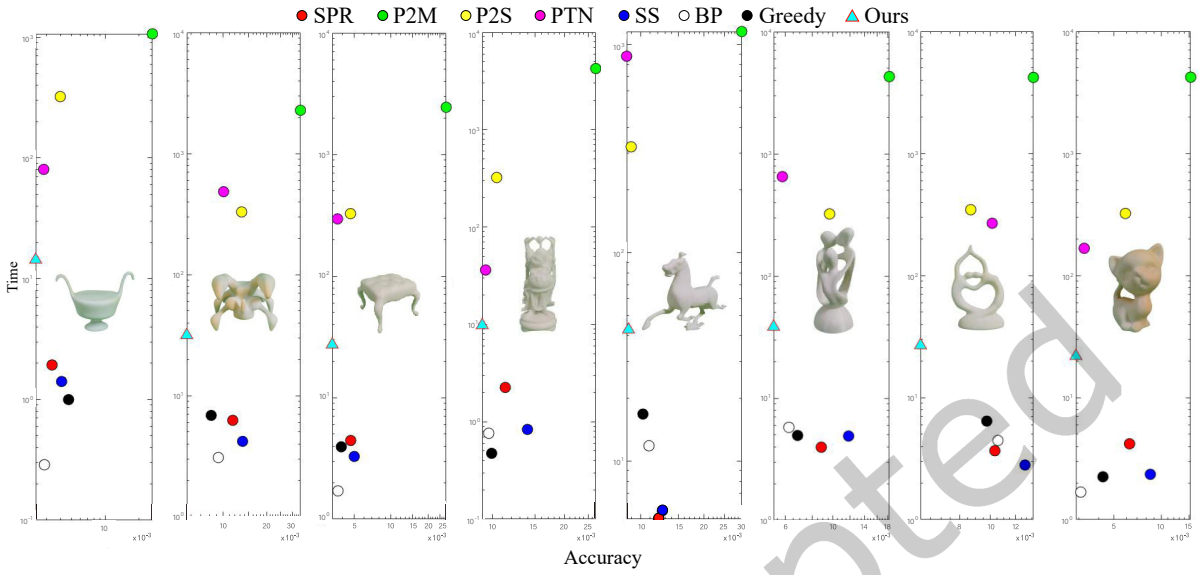
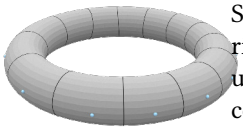


Fig. 26. Real scan for Scatter diagram. We test 8 models from the real scan, X axis is the Hausdorff error from the original model, and the axis Y is the runtime.

8 LIMITATIONS

Our algorithm, in its current form, has three limitations. First, in some rare cases, our algorithm may remove all the points in Filmsticking or delete all tetrahedral elements in Sculpting, resulting an empty mesh. This may occur when all the points are co-planar or the points are far from a reasonable sampling; See the inset figure.



Second, currently, we distinguish the interior of the guiding surface from the exterior based on winding number. An overly severe noisy point cloud may lead to an unwanted surface. Third, for large-size point clouds (e.g. more than 1000K points), the computational cost is relatively high. We shall further develop speedup strategies in the future.

9 CONCLUSION AND FUTURE WORK

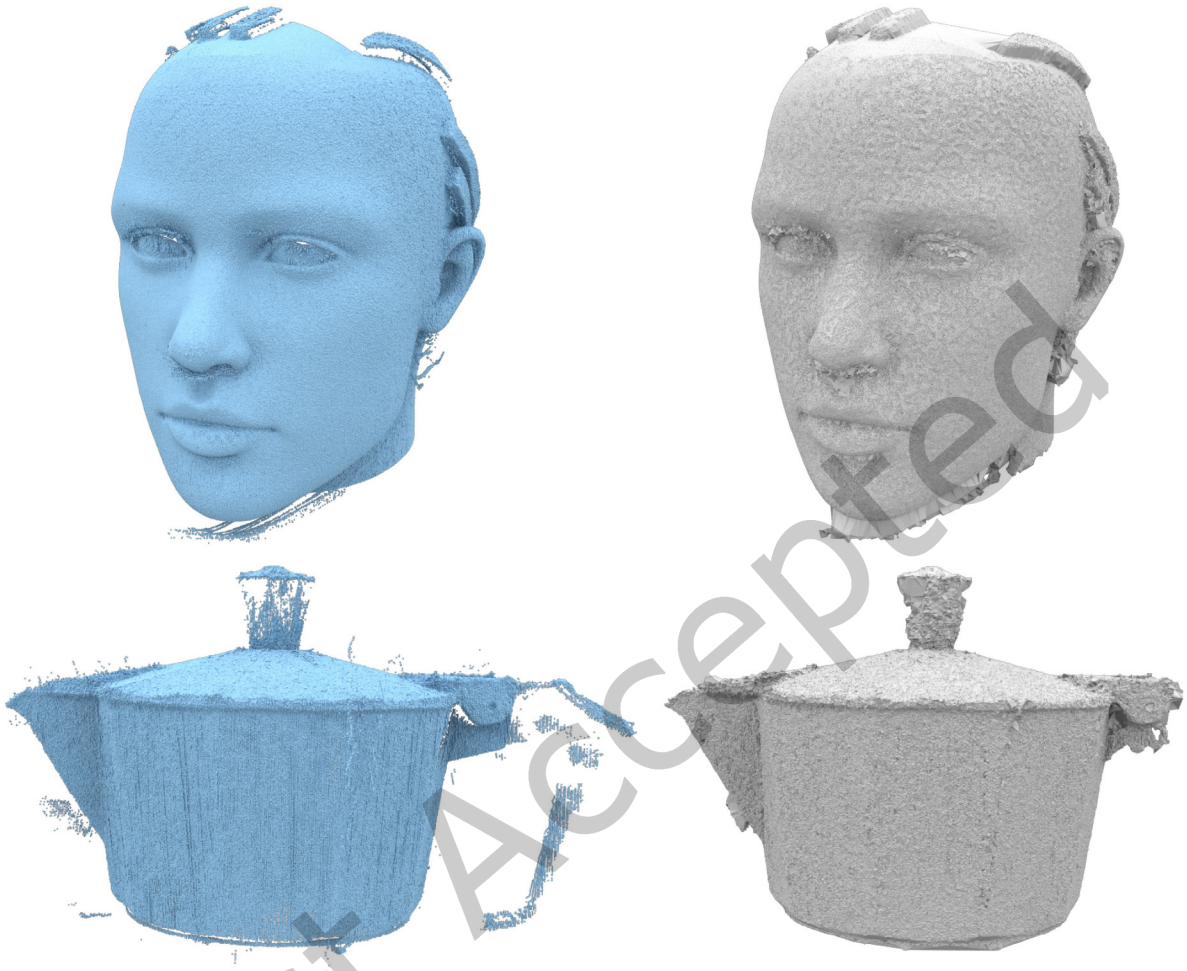
In this paper, we take a point cloud, free of normals, as the input, and use a guiding surface to help explicit surface reconstruction. Starting from a simple bounding sphere, we repeatedly evolve the guiding surface and chip away the unexpected spacing between the guiding surface and the underlying geometry. Throughout our algorithm, the guiding surface remains a watertight manifold mesh. Statistics show that our algorithm outperforms the state-of-the-art in terms of various evaluation indicators.

REFERENCES

Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. 2016. Large-Scale Data for Multiple-View Stereopsis. *International Journal of Computer Vision* (2016), 1–16.

Nina Amenta, Marshall Bern, and Manolis Kamvysselis. 1998. A new Voronoi-based surface reconstruction Algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. 415–421.

Nina Amenta, Sunghee Choi, Tamal K Dey, and Naveen Leekha. 2000. A Simple Algorithm for Homeomorphic Surface Reconstruction. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*. 213–222.



(a) Input (b) Ours

Fig. 27. Large-size point clouds.

Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. 2001. The power crust, unions of balls, and the medial axis transform. *Computational Geometry* 19, 2-3 (2001), 127–153.

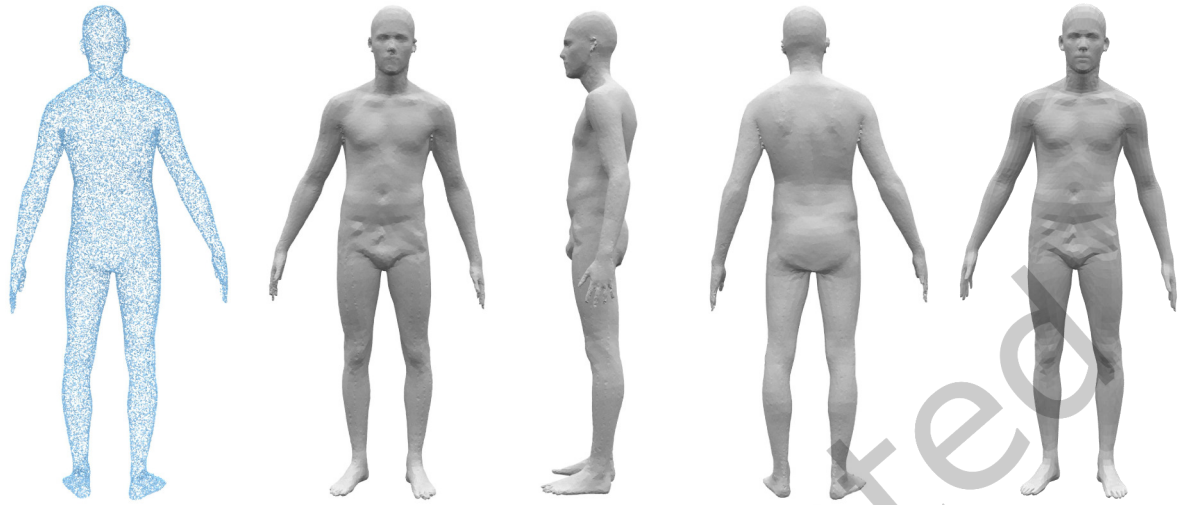
Samir Aroudj, Patrick Seemann, Fabian Langguth, Stefan Guthe, and Michael Goesele. 2017. Visibility-consistent thin surface reconstruction using multi-scale kernels. *ACM Transactions on Graphics* 36, 6 (2017), 1–13.

Sunil Arya and DM Mount. 1997. ANN: library for approximate nearest neighbor searching. In *Proceedings of IEEE CGC Workshop on Computational Geometry*.

Dominique Attali. 1998. r-regular shape reconstruction from unorganized points. *Computational Geometry* 10, 4 (1998), 239 – 247.

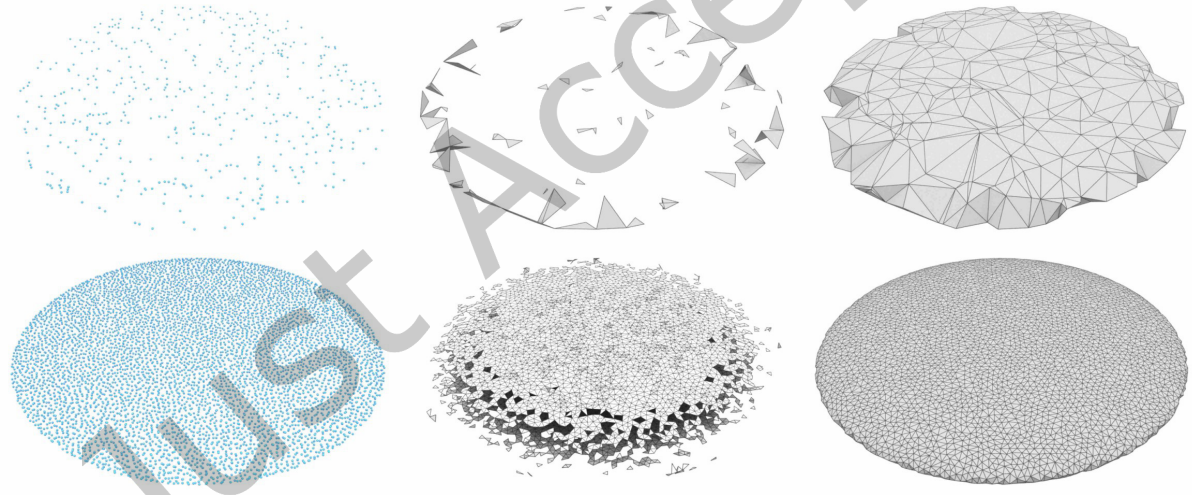
Dominique Attali, Jean-Daniel Boissonnat, and André Lieutier. 2003. Complexity of the delaunay triangulation of points on surfaces the smooth case. In *Proceedings of the nineteenth annual symposium on Computational Geometry*. 201–210.

Marco Attene, Daniela Giorgi, Massimo Ferri, and Bianca Falcidieno. 2009. On converting sets of tetrahedra to combinatorial and PL manifolds. *Computer Aided Geometric Design* 26, 8 (2009), 850–864.



(a) Input (b) Ours (c) GT

Fig. 28. Faust-Human.



(a) Input (b) Khoury and Shewchuk (c) Ours

Fig. 29. Comparison between ours and [Khoury and Shewchuk 2016]. Note that [Khoury and Shewchuk 2016] depends on the point density but our algorithm eliminates the unreasonable requirement as much as possible. Top row: In the case of 500 sample points, [Khoury and Shewchuk 2016] misses most of target triangles (middle) whereas ours produces a closed triangle mesh that interpolates all the points. Bottom row: When the number of sample points amounts to 7000, the result of [Khoury and Shewchuk 2016] still has some missing triangles whereas our result well manifests the target shape.

M. Attene and M. Spagnuolo. 2010. Automatic Surface Reconstruction from Point Sets in Space. *Computer Graphics Forum* 19, 3 (2010), 457–465.

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I.W. Levin, and Alec Jacobson. 2018. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics* 37, 4 (2018), 1–12.

Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. 2013. A benchmark for surface reconstruction. *ACM Transactions on Graphics* 32, 2 (2013), 1–17.

Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Cláudio Silva. 2017. A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum* 36, 1 (2017), 301–329.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 5, 4 (1999), 349–359.

Stephan Bischoff, Darko Pavic, and Leif Kobbelt. 2005. Automatic restoration of polygon models. *ACM Transactions on Graphics* 24, 4 (2005), 1332–1352.

Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2017. Dynamic FAUST: Registering Human Bodies in Motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Boissonnat and Jean-Daniel. 1984. Geometric Structures for Three- Dimensional Shape Representation. *ACM Transactions on Graphics (TOG)* (1984).

Jean-Daniel Boissonnat. 1984. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics* 3, 4 (1984), 266–286.

Dobrina Boltcheva and Bruno Lévy. 2017. Surface reconstruction by computing restricted Voronoi cells in parallel. *Computer-Aided Design* 90 (2017), 123–134.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 67–76.

Raphalle Chaine. 2003. A geometric-based convection approach of 3-D reconstruction. In *Eurographics Symposium on Geometry Processing*.

David Cohen-Steiner and Frank Da. 2004. A greedy Delaunay-Based surface reconstruction algorithm. *The Visual Computer* 20, 1 (2004), 4–16.

Angela Dai and Matthias Nießner. 2018. Scan2Mesh: from unstructured range scans to 3D meshes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5574–5583.

Julie Digne, Jean-Michel Morel, Charyar-Mehdi Souzani, and Claire Lartigue. 2011. Scale space meshing of raw data point sets. *Computer Graphics Forum* 30, 6 (2011), 1630–1642.

Herbert Edelsbrunner and Ernst P Mücke. 1994. Three-dimensional alpha shapes. *ACM Transactions on Graphics* 13, 1 (1994), 43–72.

H. Edelsbrunner and N. R. Shah. 1997. Triangulating Topological Spaces. *International Journal of Computational Geometry Applications* 7, 04 (1997), –.

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. 2020. Points2Surf Learning Implicit Surfaces from Point Clouds. In *European Conference on Computer Vision*. 108–124.

Simon Fuhrmann and Michael Goesele. 2014. Floating scale surface reconstruction. *ACM Transactions on Graphics* 33, 4 (2014), 1–11.

Rickard Gabrielsson, Vignesh Ganapathi-Subramanian, Primoz Skraba, and Leonidas Guibas. 2020. TopologyAware Surface Reconstruction for Point Clouds. *Computer Graphics Forum* 39 (08 2020), 197–207.

M. Gopi, Shankar Krishnan, and Cláudio T. Silva. 2000. Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation. *Computer Graphics Forum* 19, 3 (2000), 467–478.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of Machine Learning and Systems 2020*. 3569–3579.

Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. 2011. BlenSor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing*. 199–208.

Baining Guo, Jai Menon, and Brian Willette. 1997. Surface reconstruction using alpha shapes. *Computer Graphics Forum* 16, 4 (1997), 177–190.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: a network with an edge. *ACM Transactions on Graphics* 38, 4 (2019), 1–12.

Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: a self-prior for deformable meshes. *ACM Transactions on Graphics* 39, 4 (2020), 126.

C. Jiang, A. Sud, A. Makadia, J. Huang, and T. Funkhouser. 2020. Local Implicit Grid Representations for 3D Scenes. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. 61–70.

Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics* 32, 3 (2013), 1–13.

Marc Khoury and Jonathan Richard Shewchuk. 2016. Fixed Points of the Restricted Delaunay Triangulation Operator. In *32nd International Symposium on Computational Geometry (SoCG 2016)*. 47:1–47:15.

Wolfgang Köhler. 1967. Gestalt psychology. *Psychologische Forschung* 31, 1 (1967), XVIII–XXX.

Chuan-Chu Kuo and Hong-Tzong Yau. 2005. A Delaunay-based region-growing approach to surface reconstruction from unorganized points. *Computer-Aided Design* 37, 8 (2005), 825–835.

L. Ladicky, Olivier Saurer, Sohyeon Jeong, Fabio Maninchedda, and M. Pollefeys. 2017. From Point Clouds to Mesh Using Regression. *2017 IEEE International Conference on Computer Vision* (2017), 3913–3922.

Canyu Le and Xin Li. 2018. Sparse3D: a new global model for matching sparse RGB-D dataset with small inter-frame overlap. *Computer-Aided Design* (2018), 33–43.

Greg Leibon and David Letscher. 2000. Delaunay triangulations and Voronoi diagrams for Riemannian manifolds. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*. 341–349.

Xiaokun Li, Chia-Yung Han, and William G Wee. 2009. On surface reconstruction: a priority driven approach. *Computer-Aided Design* 41, 9 (2009), 626–640.

Hong-Wei Lin, Chiew-Lan Tai, and Guo-Jin Wang. 2004. A mesh reconstruction algorithm driven by an intrinsic property of a point cloud. *Computer-Aided Design* 36, 1 (2004), 1 – 9.

Yukie Nagai, Yutaka Ohtake, and Hiromasa Suzuki. 2010. Smoothing of partition of unity implicit surfaces for noise robust surface reconstruction. *Computer Graphics Forum* 28, 5 (2010), 1339–1348.

L. Nan and P. Wonka. 2017. PolyFit: Polygonal Surface Reconstruction from Point Clouds. In *2017 IEEE International Conference on Computer Vision*. IEEE Computer Society, 2372–2380.

Yutaka Ohtake, Alexander Belyaev, and Marc Alexa. 2005. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*. 149–158.

J. Pan, X. Han, W. Chen, J. Tang, and K. Jia. 2020. Deep Mesh Reconstruction From Single RGB Images via Topology Modification Networks. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Jiju Peethambaran and Ramanathan Muthuganapathy. 2015a. Reconstruction of water-tight surfaces through Delaunay sculpting. *Computer-Aided Design* 58, C (2015), 62–72.

J. Peethambaran and R. Muthuganapathy. 2015b. Reconstruction of water-tight surfaces through Delaunay sculpting. *Computer-Aided Design* 58 (2015), 62–72.

J. Pellerin, B Lévy, G. Caumon, and A. Botella. 2014. Automatic surface remeshing of 3D structural models at specified resolution: A method based on Voronoi diagrams. *Computers & Geosciences* 62 (2014), 103–116.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional Occupancy Networks. arXiv:2003.04618 [cs.CV]

Nico Schertler, Marco Tarini, Wenzel Jakob, Misha Kazhdan, Stefan Gumhold, and Daniele Panozzo. 2017. Field-aligned online surface reconstruction. *ACM Transactions on Graphics* 36, 4 (2017), 77.

Nicholas Sharp and Maks Ovsjanikov. 2020. PointTriNet: learned triangulation of 3D point sets. In *Proceedings of European Conference on Computer Vision*. 762–778.

Hang Si. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Software* 41, 2 (2015), 1–36.

Oded Stein, Eitan Grinspun, and Keenan Crane. 2018. Developability of triangle meshes. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Benjamin Ummenhofer and Thomas Brox. 2015. Global, dense multiscale reconstruction for a billion points. In *Proceedings of 2015 IEEE International Conference on Computer Vision*. 1341–1349.

R. C. Veltkamp. 1994. Closed Object Boundaries from Scattered Points. *Springer Berlin* 885 (1994).

Remco C Veltkamp. 1995. Boundaries through scattered points of unknown density. *Graphical Models and Image Processing* 57, 6 (1995), 441–452.

Jun Wang, Qiaoyun Wu, Oussama Remil, Cheng Yi, Yanwen Guo, and Mingqiang Wei. 2017. Modeling indoor scenes with repetitions from 3D raw point data. *Computer-Aided Design* 94 (2017), 1–15.

N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y. G. Jiang. 2018. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. *Springer, Cham* (2018).

Pengfei Wang, Shiqing Xin, Changhe Tu, Dongming Yan, Yuanfeng Zhou, and Caiming Zhang. 2020. Robustly computing restricted Voronoi diagrams (RVD) on thin-plate models. *Computer Aided Geometric Design* 79 (2020), 101848.

Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. 2019. Deep Geometric Prior for Surface Reconstruction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10122–10131.

Yun-Peng Xiao, Yu-Kun Lai, Fang-Lue Zhang, Chunpeng Li, and Lin Gao. 2020. A survey on deep geometry learning: from a representation perspective. *Computer Visual Media* 6 (2020), 113–133.

Minfeng Xu, Shiqing Xin, and Changhe Tu. 2018. Towards globally optimal normal orientations for thin surfaces. *Computers & Graphics* 75 (2018), 36 – 43.

Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum* 28, 5 (2009), 1445–1454.

Zhouwang Yang, Yeong-Hwa Seo, and Tae-wan Kim. 2010. Adaptive triangular-mesh reconstruction by mean-curvature-based refinement from point clouds using a moving parabolic approximation. *Computer-Aided Design* 42, 1 (2010), 2–17.

Cheng Yi, Yuan Zhang, Qiaoyun Wu, Yabin Xu, Oussama Remil, Mingqiang Wei, and Jun Wang. 2017. Urban building reconstruction from raw LiDAR point data. *Computer-Aided Design* 93 (2017), 1 – 14.

Cheng Chun You, Seng Poh Lim, Joi San Tan, C. K. Lee, and Yen Min Jasmina Khaw. 2020. A Survey on Surface Reconstruction Techniques for Structured and Unstructured Data. In *2020 IEEE Conference on Open Systems*. 37–42.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).