A Watertight Surface Reconstruction Method for CAD Models Based on Optimal Transport

Yuanyan Ye, Yubo Wang
School of Informatics
Xiamen University
yuanyanye_xm@163.com

Juan Cao
School of Mathematical Sciences
Xiamen University
juancao@xmu.edu.cn

Zhonggui Chen
School of Informatics
Xiamen University
chenzhonggui@xmu.edu.cn

Abstract

Feature-preserving mesh reconstruction from point clouds is a challenging task. Implicit methods tend to fit smooth surfaces and cannot reconstruct sharp features. Explicit reconstruction methods are sensitive to noise and interpolate sharp features only when the points are distributed on the feature lines. We propose a watertight surface reconstruction method based on optimal transport, which can faithfully reconstruct sharp features that are often present in CAD models. We formalize the surface reconstruction problem as minimizing the optimal transport cost between the point cloud and the reconstructed surface. The algorithm consists of initialization and refinement steps. In the initialization step, the convex hull of the point cloud is deformed under the guidance of the transport plan to obtain an initial approximate surface. Next, the mesh surface is optimized using operations including vertex relocation and edge collapses/flips to obtain feature-preserving results. Experiments show that our method can preserve sharp features while being robust to noise and missing data.

Keywords: surface reconstruction, feature-preserving, discrete optimal transportation

1. Introduction

The problem of reconstructing surfaces from point clouds transpires in many science and engineering applications. The formulations for the surface reconstruction problem may vary according to the different input and output requirements. Here, we focus on the problem of reconstructing a triangle mesh from point clouds scanned from CAD models.

3D real-world scanned point clouds typically contain noise, outliers, missing points, and non-uniform sampling. Generally, performing surface reconstruction algorithms directly on raw point clouds reduces algorithm efficiency and fails to achieve high-quality results. For example, the presence of noise may result in overfitting and bumpy surfaces. Point clouds with non-uniform sampling and missing data generally lead to undesired holes and other artifacts in the output results. Specific CAD applications, such as creating a prototype from a model boundary, require a watertight surface. Besides, many CAD models are generally angular, that is, they contain sharp edges, corners, and flat faces. It is desired to reconstruct a watertight surface from point clouds scanned from CAD models using a small number of faces while preserving the sharp features.

The problem of surface reconstruction from point samples has received considerable effort in the past couple of decades. Berger *et al*. [2] presented a comprehensive survey of reconstruction methods. These methods can be divided into two categories, namely, interpolation and approximation methods, each with its advantages and disadvantages. The interpolation methods [8, 3, 16, 6, 36] can restore subtle structures in point clouds but are noise sensitive. The surface defined by an implicit function [23, 27, 28] achieves good smoothness, but it is not effective in maintaining sharp features. Therefore, preserving sharp features and being robust to noise and missing data during reconstruction remains an essential challenge.

Reconstruction methods based on mesh deformation [35, 29, 22, 37] have also received certain attention. Hanocka *et al*. [22] proposed Point2Mesh, which continuously shrank and deformed the convex hull of the point cloud by optimizing the network weights of MeshCNN [21] to obtain reconstruction results. The method is noise insensitive and can handle point clouds with missing data. However, the quality of output meshes is not satisfactory, especially in areas with sharp features. Based on the theory of optimal transport, we propose an effective method of feature-preserving surface reconstruction. Similar to the Point2Mesh [22], we use the convex hull of point clouds as the initial mesh to deform and shrink. Instead of neural networks, we use the optimal transport cost between the mesh and the point cloud to measure their difference and directly guide the deformation. Our method is superior to Point2Mesh [22] in terms of feature recovery.

Our method consists of two stages, namely, initialization, and refinement. In the initialization stage, it takes the convex hull of the point cloud as the initial mesh and moves the mesh vertices according to the optimal transport plan. In the refinement stage, we improve the mesh by performing edge collapse, vertex relocation, and edge flip to the initial mesh according to the optimal transport cost between the point cloud and the mesh. The main contributions of our work are presented as follows:

1. We introduce a feature-preserving surface reconstruction algorithm based on optimal transport, which formalizes the surface reconstruction problem as a minimization of the optimal transport cost between the point cloud and the reconstructed mesh.

2. To avoid the self-intersection due to transportation errors, we propose the addition of the variance of local neighborhood maps to the cost function of optimal transport in the initialization step to obtain more appropriate transportation.

3. We tailor optimal transport cost-driven vertex relocation, edge collapse, and edge flip operations to optimize the mesh and reconstruct sharp edges and corners efficiently.

2. Related work

2.1. Surface reconstruction

Surface reconstruction is a fundamental problem in computer graphics, and many studies on this problem have been performed in recent years. We mainly focus on feature-preserving and deformation-based methods most closely related to our method. We refer the reader to the references [2, 13, 26, 38] for comprehensive surveys on this topic.

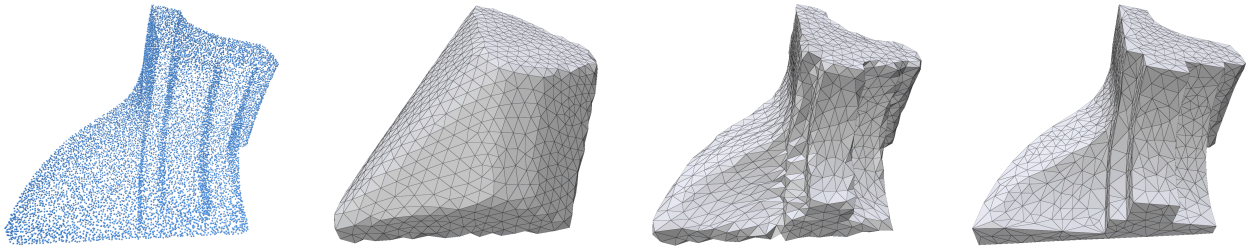
Feature-preserving reconstruction. For CAD models, we aim to reconstruct watertight, topologically correct, and feature-preserving mesh surfaces. The features here mainly refer to sharp edges and corners. RIMLS [34] combines least squares with local kernel regression to achieve feature preservation. This method produces an implicit surface that is differentiable everywhere, resulting in different sharpness of the reconstructed edge features. Avron *et al*. [1] formulated the problem of sharp feature reconstruction as a global ℓ_1 optimization problem. It first optimizes the normal of points and then the vertex positions. Huang *et al*. [24] used bilateral filtering to smooth normals, performed different resampling strategies in the feature area and the plane area, and finally used existing reconstruction algorithms to reconstruct the resampled point cloud. Digne *et al*. [15] took advantage of optimal transport to obtain feature-preserving reconstruction results but with a long computation time. Another approach is based on geometric primitives [32], which involves the generation of candidate faces and face selection, but is generally unsuitable for smooth surfaces. Other methods [14, 11] use the extracted feature lines in point clouds as a prior.

Deformation-based reconstruction. Deformation-based methods iteratively expand or shrink-wrap an initial mesh. The key to deformation is determining the new position of the mesh vertices and avoiding topological errors, such as self-intersections. Sharf *et al*. [35] used compactly supported RBFs [33] to guide the initial mesh to expand continuously toward the point cloud. Lu *et al*. [29] proposed a method that evolves two spheres from the interior and exterior of the input point cloud, respectively, where the central surface is extracted as the reconstruction result. Point2Mesh [22] shrinks the mesh from the exterior by learning the vertex displacement. However, the deformed mesh is not guaranteed to be a manifold during the learning process, and a manifold reconstruction algorithm [25] needs to be applied every certain number of iterations. Wang *et al*. [37] proposed a method that began from an initial simple surface mesh and alternately performed filmsticking and sculpting operations to minimize the distance between the reconstructed surface and the input point cloud. Our method also uses a deformable model. Differently, we determine the new vertex positions according to the optimal transport plan. Furthermore, our approach is advantageous over the aforementioned methods in sharp feature preservation.

2.2. Optimal transport

The optimal transport problem was first formulated by Monge[31] and can be described by a simple example. Assuming that m piles of sand and n bunker have the same total volume, then the cost of moving the sand is proportional to the distance. The objective of optimal transport is to solve the plan that minimizes the overall effort required to transport all the sand to the bunkers. The total cost is called the Wasserstein distance. The optimal transport problem can generally be solved using the network simplex method [5] or the entropy regularization method [9].

In recent years, many methods based on optimal transport theory have been proposed for various applications in computer graphics. For example, Bonneel *et al*. [4] addressed the point cloud alignment and image color transfer problem using the sliced partial optimal transport. Mandad *et al*. [30] represented the map between shapes in terms of an optimal transport plan between sample points on two surfaces. De Goes *et al*. [12] applied discrete optimal transport theory to 2D shape reconstruction, where the transport cost of points to vertices and edges of 2D shapes are used to drive edge simplification and relocation. Digne *et al*. [15] extended this method to 3D shape reconstruction. Still, the method is inefficient because the optimal transport problem needs to be repeatedly solved by applying inefficient large-



(a) Input (b) Initial mesh (c) Initial approximate mesh (d) Reconstructed result

Figure 1. Overview: Given the input point cloud (a), construct a convex hull of the point cloud as the initial mesh (b), deform the initial mesh according to the optimal transport plan to obtain the initial approximate mesh (c), and finally perform vertex relocation, edge collapse, and edge flip on the initial approximate mesh to obtain the reconstructed result (d).

scale linear programming.

3. Overview

The goal of our method is to construct a watertight triangle mesh $\hat{M}$ that approximates the given point cloud $P = \{p_1, p_2, ..., p_n\}$ while preserving sharp features in the 3D model. We formulate the surface reconstruction as the optimization of discrete optimal transport cost between the input point cloud and the reconstructed mesh surface. The transport plan between P and $\hat{M}$ is represented by the map between P and the points $S = \{s_1, s_2, ..., s_n\}$ that are sampled from the mesh surface $\hat{M}$.

We define the triangle mesh by a set of vertices, faces, and edges (V, F, E) and consider each face as a weighted sum of Dirac measures centered at the sampling points with unit capacity. Each point of P is also considered a Dirac measure centered at p_i with a unit mass. The size of P and S is both n. Therefore, reducing the reconstruction error is equivalent to minimizing the discrete optimal transport cost between P and S by optimizing the following equation:

$$\begin{aligned} &\min \sum_{ij}^{n} \pi_{ij} C_{ij} \\ &s.t.\ \ \forall i, j : \pi_{ij} = 0, 1, \\ &\qquad \forall i : \sum_{j}^{n} \pi_{ij} = 1, \\ &\qquad \forall j : \sum_{i}^{n} \pi_{ij} = 1, \end{aligned} \tag{1}$$

where the transport plan π_{ij} denotes the mass transported from point p_i to sampling point s_j. The value of π_{ij} is restricted to 1 or 0. The map between P and S is one-to-one. C_{ij} denotes the transport cost between p_i and s_j. The cost functions may vary for applications, and a common choice is the squared Euclidean distance.

Given a point cloud, the convex hull of the point cloud is selected as the initial mesh for shrink-wrap to achieve a watertight surface (Figure 1 (b)). The mesh is first deformed to fit the point cloud roughly and then optimized to recover sharp features (see Figure 1 (c-d) for examples of the intermediate and final results of our method). The two main steps of our method are explained as follows:

1. Initialization. The initial mesh is constructed by remeshing the convex hull of the input point cloud, as shown in Figure 1. We sample on the initial mesh and calculate π_{ij} between the input point cloud and the current sampling points. The initial approximate mesh is constructed by deforming the initial mesh according to π_{ij}. The calculation of the optimal transport and the deformation of the mesh will be described in detail in Section 4.

2. Refinement. We perform optimization operations, including vertex relocation, edge collapse, and edge flips, to reduce transport costs further by adjusting the mesh's topological connectivity and vertex positions. Details of the refinement will be described in Section 5.

4. Initialization

4.1. Vertex displacement

We calculate a transformation matrix for each face of the mesh based on the optimal transport plan to deform the initial mesh. The new vertex positions are jointly determined by their one-ring neighborhood faces, and the transformation matrix of each face is driven by the transport plan of its sampling points. Suppose the sampling point of face f is $B = \{b_1, b_2, ..., b_k\}$ and the corresponding points of B in the point cloud is $Q = \{q_1, q_2, ..., q_k\}$. Then, the rotation matrix $\boldsymbol{R}$, translation vector $\boldsymbol{t}$, and scaling scale s between B and Q can be obtained by optimizing the following equation.

$$(\boldsymbol{R}, \boldsymbol{t}, s) = \underset{\boldsymbol{R}, \boldsymbol{t}, s}{argmin} \sum_{i=1}^{k} \|(s\boldsymbol{R}b_i + \boldsymbol{t}) - q_i\|^2. \tag{2}$$

Thus, the new position of the vertex v in the face f is calculated as $\hat{v} = s\boldsymbol{R}v + \boldsymbol{t}$. The final position of the vertex v

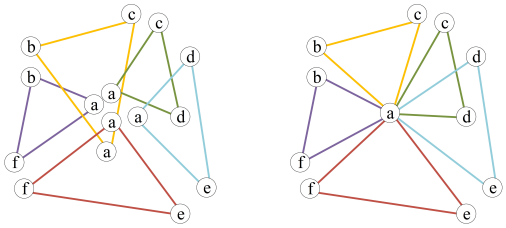


Figure 2. Computation of the new mesh vertex position. Each face adjacent to vertex a generates a new location (left), and they are averaged to determine the new positions of vertex a (right).

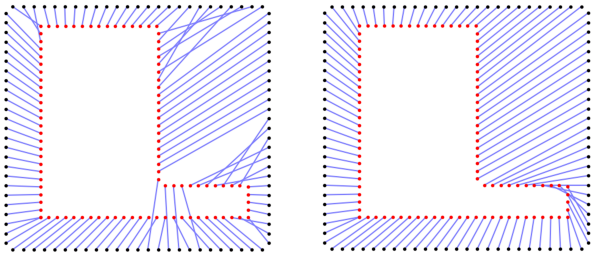


Figure 3. Result of optimal transport with different cost functions on a 2D shape. The black points are source, the red points are target, and the blue lines represent a transport relationship. Left: optimal transport plan using the squared Euclidean distance. Right: result after minimization using the local variance of the neighborhood.

is the average of its new positions deduced by its adjacent faces, as shown in Figure 2.

4.2. Calculation of optimal transport

As we use the convex hull to construct the initial mesh, its shape differs significantly from that of the input point cloud. If the cost function is simply set to the squared Euclidean distance, then incorrect transport may occur in the thin region where the upper surface is closer to the lower surface. Figure 3 (left) shows the optimal transport plan using the squared Euclidean distance as the cost function. The upper and lower surfaces of the target point on the right side are close, and some source points in the region that should be transported to the lower surface are transported instead to the upper surface, which will lead to concave and self-intersections in the initial approximate mesh.

To prevent mapping the adjacent points in the point cloud to scattered points, we apply the variance-minimizing transport proposed by Mandad *et al*. [30] to calculate the optimal transport plan between the input point cloud and the sampling points. For points p_i and neighbor p_k in P, we hope their relative position relationship remains the same after mapping. Therefore, we set different weight values for p_k based on its distance to the point p_i and use the normalized Gaussian weighting function in the following equations to determine which points are the neighbors of point p_i:

$$w_{p_k} = \frac{W_{p_i}\left(p_k\right)}{\sum\limits_k W_{p_i}\left(p_k\right)}, \tag{3}$$

where

$$W_{p_i}\left(p_k\right) = \exp\left(-\left(p_i - p_k\right)^2 / \left(2\sigma^2\right)\right), \tag{4}$$

σ is set as the average point spacing d_{avg} of the input point clouds. If w_{p_k} is greater than a threshold ε, then p_k is considered a neighboring point of p_i, $\varepsilon = \exp\left(-d_{\text{avg}}^2/2\sigma^2\right)$. We denote the set of neighboring points of p_i as N_{p_i}, and the mass assigned to each neighboring point is its Gaussian weight.

The neighborhood of the mapped points can be portrayed by the variance. Assuming that the mass of each point in $X = \{x_1, x_2 ...\}$ is m_i, the variance between X and its center η can be calculated

$$\text{var}(X, \eta) = \sum_i m_i \cdot d(x_i, \eta)^2, \tag{5}$$

where $d\left(x_i, \eta\right)$ is the Euclidean distance between x_i and η. Assuming that the mapped points of N_{p_i} in S is $\pi_P\left(N_{p_i}\right)$, the neighborhood-to-neighborhood map can be achieved by optimizing the variance of $\pi_P\left(N_{p_i}\right)$ with its center η_{p_i}. To do so, we adopt the following cost function in the optimal transport calculation in Equation (1):

$$\begin{aligned} C(\pi, \eta) = &\sum_i^n \text{var}\left[\pi_P\left(N_{p_i}\right), \eta_{p_i}\right] + \\ &\sum_j^n \text{var}\left[\pi_S\left(N_{s_j}\right), \eta_{s_j}\right]. \end{aligned} \tag{6}$$

We optimize Equation (1) through alternating minimization: we first fix the transport plan and optimize the centers η_{p_i} and η_{s_j} of each point; second, we optimize the transport plan according to the centers. After optimization, the neighborhoods in the input point cloud will be mapped to the neighborhoods in the sampling points, as shown in Figure 3 (right). With the update of the transport plan, the cost matrix that stores the cost of the transportation from p_i to s_j needs to be updated. The process of constructing the initial approximate mesh in the initialization step is described in Algorithm 1.

5. Refinement

The initial approximate mesh just roughly fits the point cloud, and we will use vertex relocation, edge collapse, and edge flip operations in the refinement step to fine-tune the mesh so that the reconstruction results can recover the sharp features and reduce the approximation error further.

Algorithm 1 Construction of the initial approximate mesh

Input: Point cloud P;
Output: Initial approximate mesh $M' = (V, F, E)$;

1: Construct a convex hull of P and remesh it as the initial mesh M;
2: Generate sampling points S from M;
3: Calculate the optimal transport plan π_{ij} between P and S according to Equation (1), where the cost function C_{ij} is calculated using Equation (6);
4: Compute the new position of the vertices of M according to π_{ij} as stated in Section 4.1.
5: Move the vertices to the new position to get the initial approximate mesh M';
6: **return** M'.

The refinement operations used in this phase are also driven by the transport cost, but the cost function is different from that in Section 4. The process of calculating variance-minimizing transport plan requires maintaining a cost matrix with the size of $n \times n$, which consumes considerable of memory and runtime. To improve the the speed, we use the squared Euclidean distance as the cost function. It generally does not lead to an inappropriate transport plan, because the input point cloud and the mesh are already close at this step, and mapping neighborhood to points that are scattered will result in a high transport cost. Thus, the cost function C_{ij} in Equation (1) is set to $\|p_i - s_j\|^2$, and the network simplex algorithm proposed by Bonneel *et al*. [5] is used to solve the equation.

The overall process of the refinement is given in Algorithm 2. We will describe each refinement operation in detail in the following sections.

5.1. Vertex relocation

To reduce the reconstruction error, we further minimize the distance from the point to the mesh surface by moving the vertices. Assuming that a sampling point s_j of the face $f = (v, v_1, v_2)$ has the barycentric coordinates $(\alpha_j, \beta_j, \gamma_j)$ with respect to (v, v_1, v_2). The corresponding point of s_j is p_i, and the point p_i' is the projection of p_i on a line that passes through s_j and with the direction of the face normal. The new position of the vertex v can be obtained by the following equation:

$$\min_{v} \sum_{i} \sum_{j} \pi_{ij} \|p_i' - (\alpha_j v + \beta_j v_1 + \gamma_j v_2)\|^2.$$

Thus each triangle f adjacent to v deduces a new position for v, denoted by $v_f^\star$. The final position $v^\star$ of v is calculated

Algorithm 2 Refinement of the initial approximate mesh

Input: Point cloud P, target number of vertices $vnum$, initial approximate mesh $M' = (V, F, E)$;
Output: Refined result M;

1: $i \leftarrow 0$, is_init $\leftarrow True$;
2: Generate sampling points S' from M';
3: Calculate the transport plan π^0 between P and S';
4: Get M_0 from M' by performing vertex relocation and edge flip on M' according to π^0;
5: Initialize the priority queue Q;
6: **repeat**
7: **if** is_init $= True$ **then**
8: Clear the queue Q;
9: **for** each edge $e \in M_i$ **do**
10: **if** angle of between faces adjacent to $e < \theta_1$ **then**
11: Simulate edge collapse;
12: **if** cost change Δ<0 **then**
13: Push (e) to Q, sorted by Δ;
14: **end if**
15: **end if**
16: **end for**
17: **end if**
18: $i \leftarrow i + 1$, is_init $\leftarrow False$;
19: Pop e^* out of Q and collapse e^* to make M_i;
20: Update Q and transport plan π^i.
21: **if** number of deleted vertices % 10 =0 **then**
22: Get M_{i+1} by performing vertex relocation and edge flip on M_i according to π^i;
23: $i \leftarrow i + 1$, is_init $\leftarrow True$;
24: **end if**
25: **until** size of $V_i = vnum$ or Q is empty
26: **return** M_i.

as follows,

$$v^\star = \frac{\sum_{f \ adjacent \ to \ v} \pi_f \cdot v_f^\star}{\sum_{f \ adjacent \ to \ v} \pi_f}, \tag{7}$$

where π_f is the total mass transported to the surface f. Vertex relocation is performed in two steps. First, we fix the transport plan π_{ij} and calculate the new position for each vertex according to π_{ij} and Equation (7). We move the vertex to its new position if it does not cause face flipping. Then, we resample the current mesh and update the transport plan π_{ij}. Figure 4 demonstrates the changes in the mesh after vertex relocation, and some feature edges have been recovered.

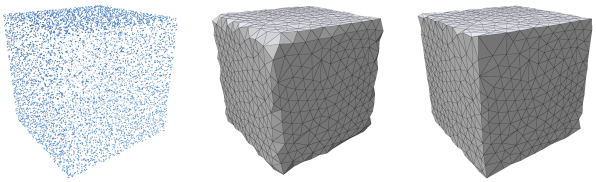


Figure 4. Effect of vertex relocation. Left: input point cloud. Middle: mesh before vertex relocation. Right: result after vertex relocation.

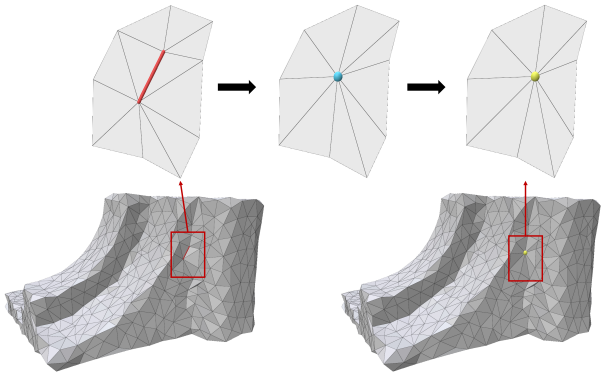


Figure 5. Simulation of edge collapse. Collapse the edge (red edge) to its midpoint (blue point), and then the optimal position (yellow point) is obtained by vertex relocation.

5.2. Edge collapse

The vertex relocation operation only updates vertex positions, but does not change the connectivity of the mesh. The mesh can be further refined by the edge collapse operation. The classical mesh simplification method is based on a quadratic error metric [20], but it only considers the geometric error to the original mesh. In contrast, the error metric for our edge collapse is set as the optimal transport cost, and the sharp features are better recovered after the edge collapse operation. In our method, the use of halfedge collapse for mesh simplification may produce slim triangles, that result in low mesh quality and face flipping in the subsequent process. Therefore, instead of half-edge collapse, we first collapse the edge to the midpoint of the edge and then find the optimal position of the point by vertex relocation.

To determine which edge to collapse, an edge collapse simulation is performed to calculate the change in transport cost before and after the operation. The edge with the highest transport cost reduction is selected to collapse. Even though the speed of the method used to solve the optimal transport in this stage is faster than that in Section 4, it is still very time-consuming if the global transport cost variation is calculated in each edge collapse. Considering that edge collapse mainly affects adjacent faces, only the changes in cost in the neighboring faces are calculated to speed up. To reconstruct sharp features and save computing time, we simulate edge collapse only for edges where the angle between two adjacent faces is less than a threshold θ_1.

Figure 5 shows the result of edge collapse, and the position of the new vertex after collapse can recover the feature edges better.

5.3. Edge flip

Edge flip is a simple and effective operation to improve mesh quality and reduce reconstruction errors. The edge flip operation is performed first in the plane regions where the angle of two adjacent faces is greater than a threshold θ_2. We flip all edges that satisfy the condition that the sum of the two interior angles opposite to the edge is greater than the sum of the two other angles, as shown in Figure 6. It will remove the narrow triangles and improve the quality of the mesh.

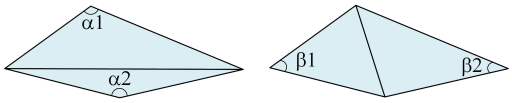


Figure 6. Edge flip in planar areas. Left: before edge flip. Right: after edge flip. The condition for edge flip is $\alpha 1 + \alpha 2 > \beta 1 + \beta 2$.

Next, we perform edge flip in the non-planar regions where the angle of two adjacent faces is less than a threshold θ_3. Edge flip is performed only if it will reduce the transport cost, as shown in Figure 7. We add edges that satisfy the aforementioned conditions to a priority queue sorted by the reduction of transport cost. After flipping an edge, we update the edges in the queue affected by it.

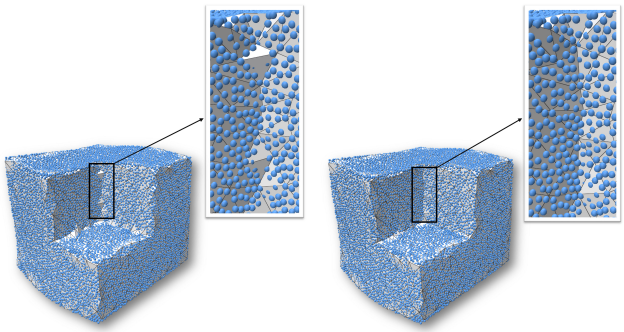


Figure 7. Edge flip in non-planar areas. Blue points are the input point cloud. Left: mesh before edge flip. Right: mesh after edge flip.

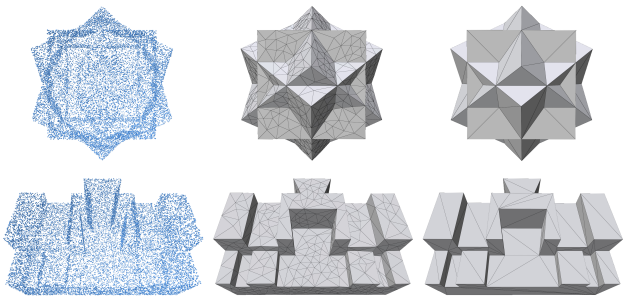


Figure 8. Reconstruction result from synthetic data. Left: input point clouds, sampled from a ground truth surface. Middle: reconstruction result of our method. Right: ground truth surface.

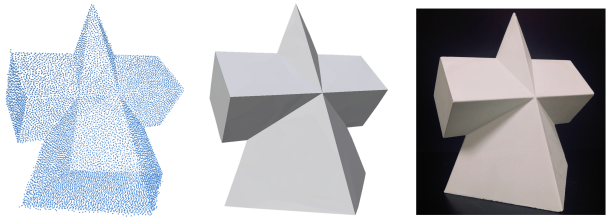


Figure 9. Reconstruction result (middle) on a real scanned point cloud (left) which is obtained by scanning the object (right) with EinScan-SP 3D scanner.

6. Experiments

Most of our algorithms are implemented in C++ based on the CGAL library [18], except for solving the optimal transport during the initialization step which is implemented in Python based on the POT library [19]. The network simplex algorithm [5] is used to solve for the optimal transport during the refinement step. All experiments and comparisons are conducted on a PC with a 2.6GHz Intel(R) Core(TM) i5-11400F CPU and 16GB of memory.

6.1. Implementation details

We assume that the size of the sampling points is the same as the input point cloud, and we need to determine how many sampling points each face has. In the initialization step, we assume that the point cloud and sampling points are uniformly distributed. The number of sampling points on each face is proportional to the face area. We also ensure that each face contains at least one sampling point. Points obtained in practice are usually unevenly distributed. We do not maintain the aforementioned assumption in the refinement step. We temporarily assign each point in the point cloud to its nearest triangle, and the size of the sampling points on the face is the number of points closest to it. Next, each face is sampled using the CVT method [17].

The number of iterations in the initialization step to compute the optimal transport with the variance of the neighborhood is generally set as 6. The parameters in the refinement step are set as follows: $\theta_1 = 135°$, $\theta_2 = 170°$, and $\theta_3 = 150°$. The parameter $vnum$ in Algorithm 2 is set to 85% of the number of vertices of the initial approximate mesh.

6.2. Feature-preserving

Our method has a greater advantage in recovering sharp features in CAD models, without resorting to denoising, resampling, or any other pre/post-processing. Notably, the newly inserted point of the collapsed edge will moves toward the feature region in order to reduce the transport cost, as shown by the yellow point in Figure 5. To further reduce the cost, vertex relocation and edge flip operations also make the vertices and edges fit the point cloud more closely, which lead to corners and sharp edges. Figure 8 and Figure 9 show the capability of our method on handling synthetic point clouds sampled from ground truth surfaces, and real scanned point clouds, sharp edges and corners can be clearly reconstructed by our method.

We compare our method with the Screened Poisson method [28], RIMLS [34], RIMLS [34] over EAR [24], CVT-based method [6], Point2Mesh [22], and PointTriNet(PTN) [36]. RIMLS [34] over EAR [24] is a reconstruction method that takes the points resampled by EAR [24] as the input of RIMLS [34]. Figure 10 shows the results of these methods on reconstructing CAD models. Among them, the results of the Screened Poisson method [28] and RIMLS [34] are obtained by the implementation in MeshLab [7], and the rest are obtained by running the code provided by the authors. The size of the point cloud in our experiments is between 10K and 20K. Our method takes 15 to 20 minutes to reconstruct the point clouds in Figure 10, and the number of vertices deleted in the refinement step is 120 to 150. The initialization step takes approximately 10 minutes, and most of the remaining time is spent on simulating edge collapses.

To compare and analyze the results quantitatively, we color-code the distance between the ground truth surface and the reconstructed surface. Figure 11 shows the reconstruction errors of different reconstruction algorithms. Compared with these methods, our method is superior in feature preservation and has smaller reconstruction errors, and the outputs are guaranteed to be watertight. The screened Poisson method [28] tends to produce smooth surfaces and requires normal information as input. RIMLS [34] only recovers some sharp features and the reconstructed surfaces are not very smooth. Edge aware Resampling (EAR) [24] is an efficient method for resampling point clouds with sharp features. RIMLS [34] generates a mesh with improved quality and sharper features by taking the points resampled by EAR [24] as the input, as shown in

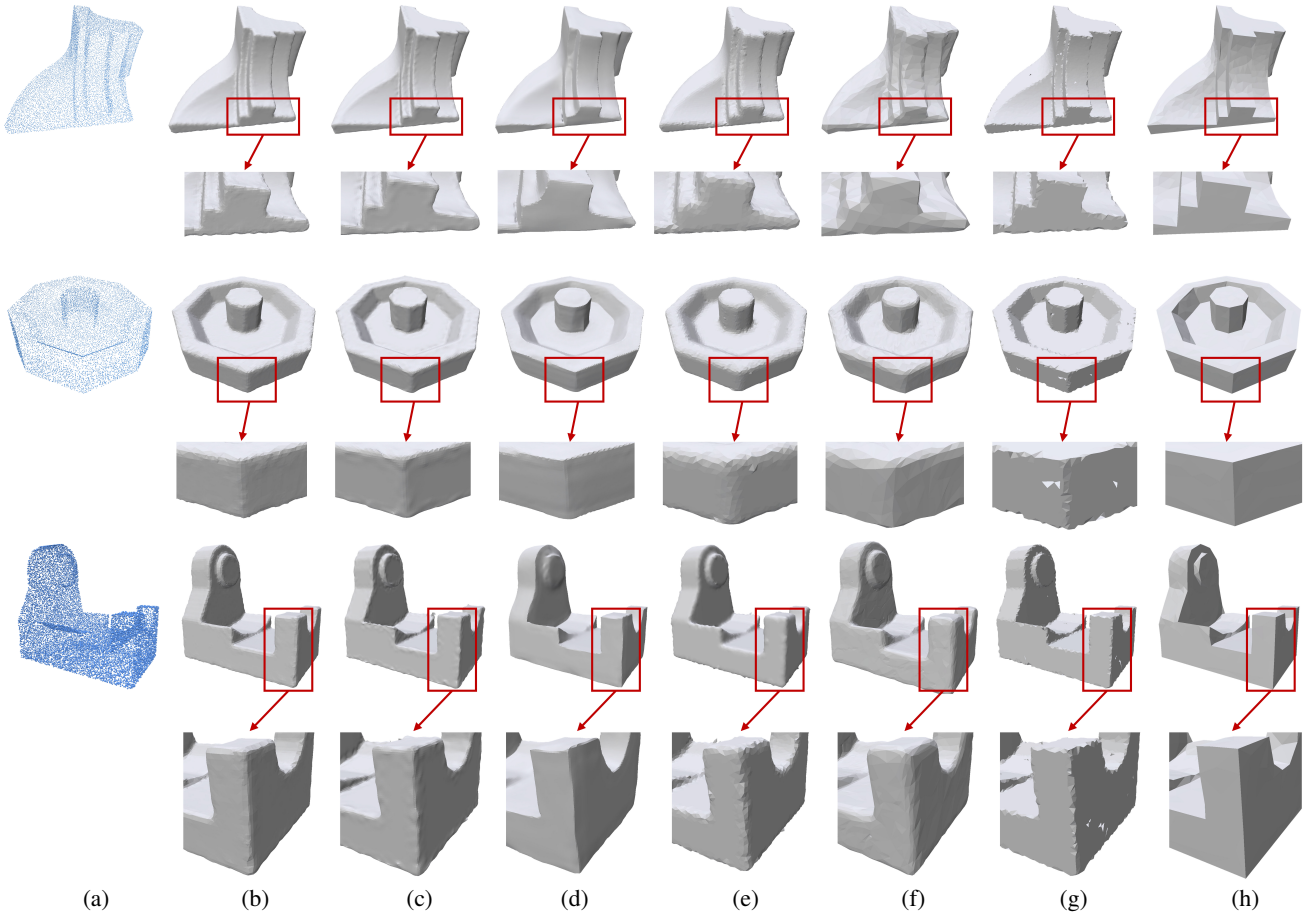


Figure 10. Comparing the ability of reconstruction methods to preserve sharp features. (a) Input point clouds; (b) Screened Poisson [28]; (c) RIMLS [34]; (d) RIMLS [34] over EAR [24]; (e) CVT-based [6]; (f) Point2Mesh [22]; (g) PTN [36]; (h) Our method.

Figure 10 (d). However, the reconstruction errors increased due to resampling.

The CVT-based resampling method in [6] can produce evenly distributed points, that lead to high-quality reconstructed meshes by computing the duality of restricted Voronoi cells. However, this method is more suitable for reconstructing smooth surfaces rather than models with sharp features. Point2Mesh [22] has great potential for surface reconstruction and can handle various types of point clouds. However, it needs to use the manifold reconstruction method [25] to repair the mesh during the optimization, which would produce large reconstruction errors, as shown in Figure 11 (e). In addition, the quality of the output meshes is poor. PTN [36] interpolates point clouds with neural networks. Although the average reconstruction error is small, the reconstructed surfaces are rugged at sharp edges and are generally not watertight.

The reconstruction method proposed by Digne *et al*. [15] is similar to ours. Both methods can recover sharp features with a small number of faces, as shown in Figure 12. However, our method is more efficient. In their method, the mapping between the point cloud and the sampling points is many-to-many, and the total mass received by each sampling point is a variable in the linear programming problem. The time complexity of solving this problem is $\mathcal{O}\left(n^3\right)$. To speed up, we restrict this mapping to a one-to-one mapping and solved for the optimal transport by using the network simplex method [5], which demonstrates $\mathcal{O}\left(n^2\right)$ complexity in practice and is suitable for solving the case where the total received mass of the receiver is determined. The method proposed by Digne *et al*. [15] takes about 8 seconds to solve the transport plan for 1,017 points and 210 samples, rising to 154 seconds to solve for 2,000 points and 450 samples. In contrast, our method only takes 0.15 seconds to solve the transport plan for 1,000 points and 1,000 sampling points, and 0.87 seconds for 2,000 points. In addition, their method produces the final reconstructed mesh with facets that receive mass greater than a threshold, and undesired holes will emerge when some faces are small or the threshold is not properly selected. We construct a watertight surface by deforming the convex hull of a point cloud.

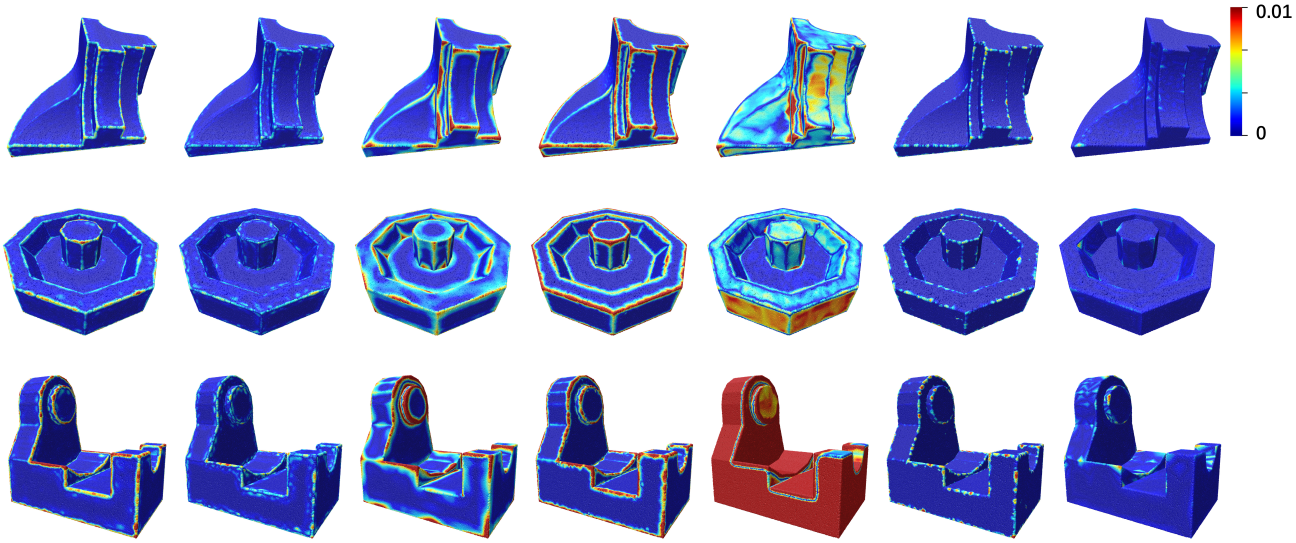


Figure 11. Comparison of reconstruction errors for different surface reconstruction algorithms. The warmer the color, the larger the deviation from the ground truth. (a) Screened Poisson [28]; (b) RIMLS [34]; (c) RIMLS [34] over EAR [24]; (d) CVT-based [6] ;(e) Point2Mesh [22]; (f) PTN [36]; (g) Our method.

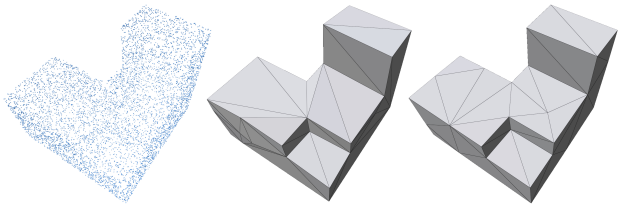


Figure 12. Preserving features with a small number of faces. Left: input point cloud (7K points); Middle: result of [15] ; Right: result of our method. The number of vertices of both meshes is 35. Their method takes more than two hours to reconstruct the surface, whereas our method takes only half an hour.

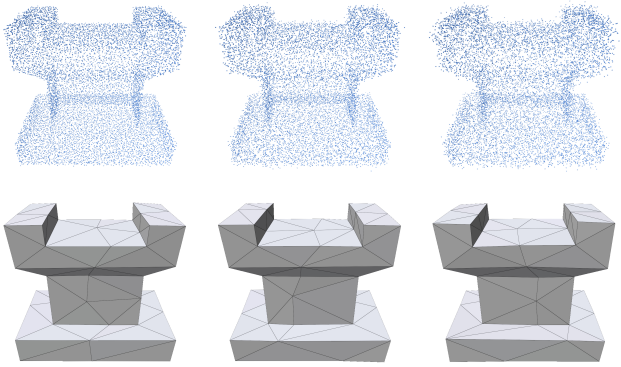


Figure 13. Reconstruction results with different levels of Gaussian noise. Top: input point clouds (12K points) with Gaussian noise. The variances of Gaussian noise are $0.5\%d$ (left), $1.0\%d$ (middle), $1.5\%d$ (right), respectively. Bottom: Output of our method. The number of vertices of output meshes is 60.

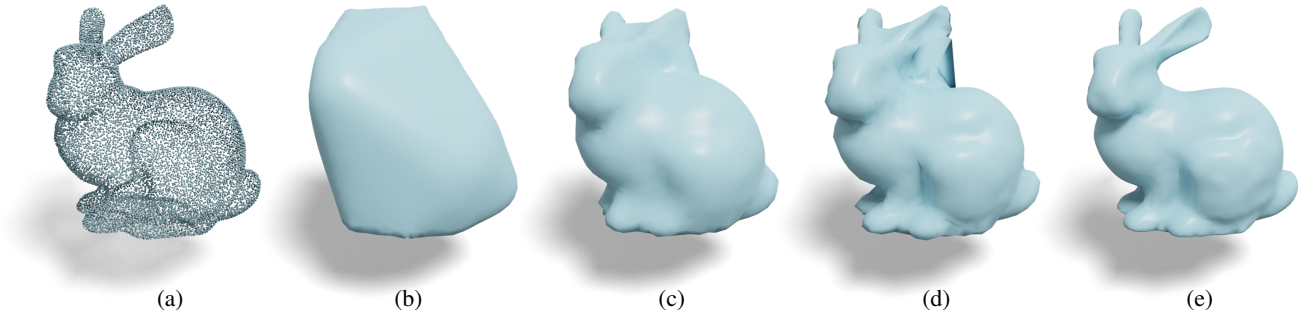
6.3. Robustness analysis

To verify the robustness of our algorithm, we test our method on points with an increasing levels of synthetic noise Figure 13 shows the effects of Gaussian noise with variances of $0.5\%d$, $1.0\%d$, and $1.5\%d$ on the results, where d is the length of the longest edge of the bounding box of the model. For noisy data, we enlarge the extent of the simplification region and increase the simplification level to prevent the mesh from becoming bumpy. The parameters in Algorithm 2 are set as follows: $\theta_1 = 175°$, and the number of target vertices $vnum$ is set to 10% of the number of vertices of the initial approximate mesh. Corners and sharp edges can still be reconstructed when the variances of the noise reach $1.5\%d$. With the decrease in the number of faces, triangles in the mesh need to fit the points on different sides of the model to reduce cost, thereby leading to a feature-preserving reconstruction result.

We also test our method on point clouds with missing data. Due to missing data, most of the explicit reconstruction methods fail to recover sharp features, and reconstruction of a watertight mesh is no guaranteed. Also, implicit reconstruction methods tend to yield smooth surfaces. In Figure 15, sharp edges and corners are reconstructed even if data are missing.

6.4. Reconstruction of smooth surfaces

Although our method is tailored for CAD models, it can be adapted to reconstruct smooth surfaces. Deforming a mesh to enter narrow and deep cavities is not an easy task.



(a) (b) (c) (d) (e)

Figure 14. The process of reconstructing a smooth surface. (a) input point cloud ; (b) initial mesh; (c) intermediate mesh between initial mesh and initial approximate mesh; (d) initial approximate mesh; (e) reconstructed result.

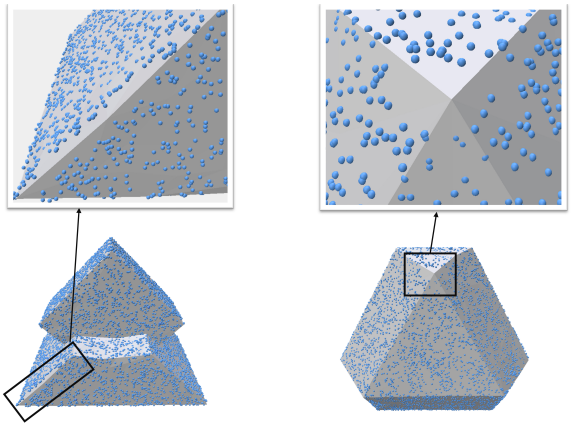


Figure 15. Reconstruction results for data with missing parts. The blue points are the input. Left: missing data at the sharp edge. Right: missing data at the corner.

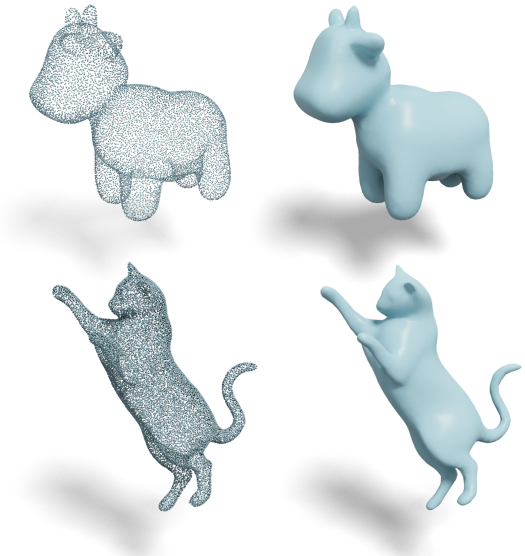


Figure 16. Tests on smooth surface. Left: input point clouds. Right: reconstruction results by our method.

For such data, if the deformation is performed in one step, then the sampling points that should be mapped to the narrow regions are likely to be transported to the wrong points.

To solve this problem, the Wasserstein barycenter [10] can be used as an intermediate result between the point cloud and the sampling points. Wasserstein barycenter is a weighted mean of a collection of probability distributions. The Wasserstein barycenter of points $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ can be obtained by optimizing the following equation:

$$\min_{\boldsymbol{X}} \sum_{i=1}^{2} w_i W_2^2 \left(\boldsymbol{b}, \boldsymbol{X}, \boldsymbol{a}_i, \boldsymbol{X}_i\right),$$

where w_i is the weight of the points X_i, which is set to 0.5, W_2^2 is the squared Euclidean distance; $\boldsymbol{b}$ is the mass of each point in the Wasserstein barycenter; and $\boldsymbol{a}_i$ is the mass of each point in the points $\boldsymbol{X}_i$, both of which are set to 1.

In our experiments, we calculate the Wasserstein barycenter W of the point cloud P and the sampling points S, and compute the transport plan for S and W. Then we deform the mesh according to the transport plan, and remesh it to obtain the mesh that is used as the initial mesh for Algorithm. 1. The process of reconstructing smooth surfaces is shown in Figure 14. Figure 16 shows the experimental results on smooth models, our method has the ability to reconstruct a smooth model.

7. Conclusions

We propose a surface reconstruction method that formulates the reconstruction problem as an optimal transport problem between the point cloud and the reconstructed mesh. We construct a convex hull of the point cloud as the initial mesh, and deform the initial mesh according to the optimal transport plan. Subsequently, we optimized the initial approximate mesh by vertex relocation, edge collapse, and edge flip operations. Experiments show that our method has higher reconstruction quality in the region of corners and sharp edges compared with other algorithms, is robust to noisy and missing data, and can be adapted to reconstruct smooth surfaces.

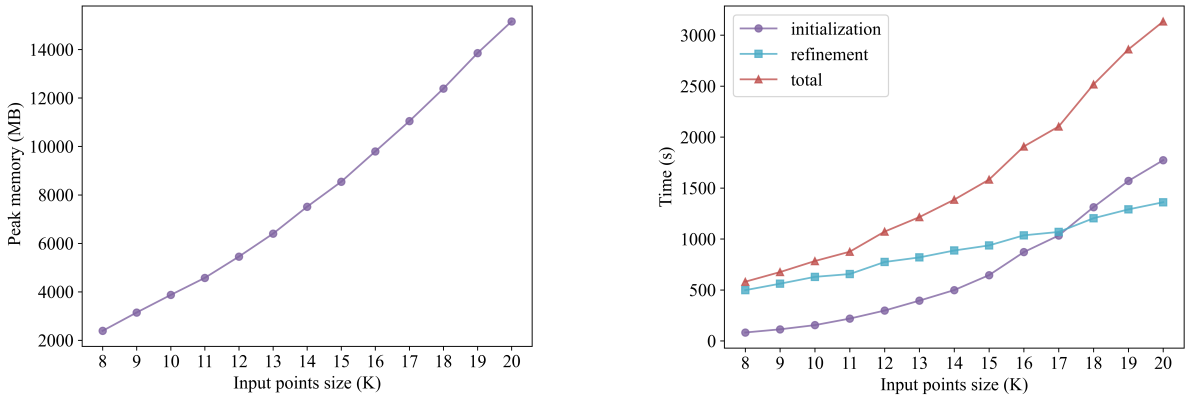


Figure 17. Peak memory (in MB) and runtime (in seconds) of our method. We use the fandisk model for test, where the number of input points ranges from 8K to 20K. The initial mesh has 1,255 vertices and 2,506 faces, and the number of target vertices in Algorithm 2 is 1,155.

Although proposed the algorithm can achieve the expected results, some limitations still exist. We visualize the statistics of peak memory and runtime in Figure 17. In the initialization step, solving the the variance-minimizing optimal transport requires storing and iteratively computing five dense matrices of $n \times n$. It has extremely demand on memory and time, limiting the applicable data size of our method. The runtime of the refinement step increases linearly, and most of the time is spent calculating the optimal transport during vertex relocation. To reduce memory consumption and runtime, we can downsample the point cloud first and reduce the frequency of vertex relocation in Algorithm 2. We will further improve the space and time efficiency of our approach in the follow-up research by using a divide-and-conquer strategy.

Acknowledgement

The research was supported by National Natural Science Foundation of China (61972327, 62272402), National Key R&D Program of China (2022YFB3303401), Xiamen Youth Innovation Funds (3502Z20206029), Natural Science Foundation of Fujian Province (2022J01001), Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (VRLAB2021B01), and Fundamental Research Funds for the Central Universities (20720220037).

3D data in this paper are courtesy of AIM@Shape, Thingi10K, and Stanford 3D Scanning Repository.

References

[1] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or. ℓ_1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics (TOG)*, 29(5):1–12, 2010. 2

[2] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36(1):301–329, 2017. 1, 2

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 1

[4] N. Bonneel and D. Coeurjolly. Spot: sliced partial optimal transport. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019. 2

[5] N. Bonneel, M. Van De Panne, S. Paris, and W. Heidrich. Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*, pages 1–12, 2011. 2, 5, 7, 8

[6] Z. Chen, T. Zhang, J. Cao, Y. J. Zhang, and C. Wang. Point cloud resampling using centroidal voronoi tessellation methods. *Computer-Aided Design*, 102:12–21, 2018. 1, 7, 8, 9

[7] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, et al. Meshlab: An open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136, 2008. 7

[8] D. Cohen-Steiner and F. Da. A greedy delaunay-based surface reconstruction algorithm. *The visual computer*, 20(1):4–16, 2004. 1

[9] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013. 2

[10] M. Cuturi and A. Doucet. Fast computation of wasserstein barycenters. In *International conference on machine learning*, pages 685–693, 2014. 10

[11] J. I. Daniels, L. K. Ha, T. Ochotta, and C. T. Silva. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI '07)*, pages 123–136, 2007. 2

[12] F. De Goes, D. Cohen-Steiner, P. Alliez, and M. Desbrun. An optimal transport approach to robust reconstruction and simplification of 2D shapes. *Computer Graphics Forum*, 30(5):1593–1602, 2011. 2

[13] T. K. Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge University Press, 2006. 2

[14] T. K. Dey, X. Ge, Q. Que, I. Safa, L. Wang, and Y. Wang. Feature-preserving reconstruction of singular surfaces. *Computer Graphics Forum*, 31(5):1787–1796, 2012. 2

[15] J. Digne, D. Cohen-Steiner, P. Alliez, F. De Goes, and M. Desbrun. Feature-preserving surface reconstruction and simplification from defect-laden point sets. *Journal of mathematical imaging and vision*, 48(2):369–382, 2014. 2, 8, 9

[16] J. Digne, J.-M. Morel, C.-M. Souzani, and C. Lartigue. Scale space meshing of raw data point sets. *Computer Graphics Forum*, 30(6):1630–1642, 2011. 1

[17] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM review*, 41(4):637–676, 1999. 7

[18] A. Fabri and S. Pion. Cgal: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 538–539, 2009. 7

[19] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. POT: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. 7

[20] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. 6

[21] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or. MeshCNN: A network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 1

[22] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or. Point2Mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4), jul 2020. 1, 2, 7, 8, 9

[23] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, pages 71–78, 1992. 1

[24] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang. Edge-aware point set resampling. *ACM transactions on graphics (TOG)*, 32(1):1–12, 2013. 2, 7, 8, 9

[25] J. Huang, H. Su, and L. Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. 2, 8

[26] Z. Huang, Y. Wen, Z. Wang, J. Ren, and K. Jia. Surface reconstruction from point clouds: A survey and a benchmark. *arXiv preprint arXiv:2205.02413*, 2022. 2

[27] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, page 0, 2006. 1

[28] M. Kazhdan and H. Hoppe. Screened Poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):1–13, 2013. 1, 7, 8, 9

[29] W. Lu and L. Liu. Surface reconstruction via cooperative evolutions. *Computer Aided Geometric Design*, 77:101831, 2020. 1, 2

[30] M. Mandad, D. Cohen-Steiner, L. Kobbelt, P. Alliez, and M. Desbrun. Variance-minimizing transport plans for inter-surface mapping. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 2, 4

[31] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pages 666–704, 1781. 2

[32] L. Nan and P. Wonka. PolyFit: Polygonal surface reconstruction from point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2353–2361, 2017. 2

[33] Y. Ohtake, A. G. Belyaev, and H.-P. Seidel. 3D scattered data approximation with adaptive compactly supported radial basis functions. *Proceedings Shape Modeling Applications, 2004.*, pages 31–39, 2004. 2

[34] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer graphics forum*, 28(2):493–501, 2009. 2, 7, 8, 9

[35] A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt, and D. Cohen-Or. Competing fronts for coarse–to–fine surface reconstruction. *Computer Graphics Forum*, 25(3):389–398, 2006. 1, 2

[36] N. Sharp and M. Ovsjanikov. PointTriNet: Learned triangulation of 3D point sets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 762–778, 2020. 1, 7, 8, 9

[37] P. Wang, Z. Wang, S. Xin, X. Gao, W. Wang, and C. Tu. Restricted delaunay triangulation for explicit surface reconstruction. *ACM Transactions on Graphics (TOG)*, 2022. 1, 2

[38] C. C. You, S. P. Lim, S. C. Lim, J. S. Tan, C. K. Lee, and Y. M. J. Khaw. A survey on surface reconstruction techniques for structured and unstructured data. In *2020 IEEE Conference on Open Systems (ICOS)*, pages 37–42, 2020. 2