Noname manuscript No.
(will be inserted by the editor)

Feature-Preserving Surface Reconstruction and Simplification from Defect-Laden Point Sets

Julie Digne · David Cohen-Steiner · Pierre Alliez · Fernando de Goes · Mathieu Desbrun

the date of receipt and acceptance should be inserted later

Abstract We introduce a robust and feature-capturing surface reconstruction and simplification method that turns an input point set into a low triangle-count simplicial complex. Our approach starts with a (possibly non-manifold) simplicial complex filtered from a 3D Delaunay triangulation of the input points. This initial approximation is iteratively simplified based on an error metric that measures, through optimal transport, the distance between the input points and the current simplicial complex—both seen as mass distributions. Our approach is shown to exhibit both robustness to noise and outliers, as well as preservation of sharp features and boundaries. Our new feature-sensitive metric between point sets and triangle meshes can also be used as a post-processing tool that, from the smooth output of a reconstruction method, recovers sharp features and boundaries present in the initial point set.

Keywords Optimal transportation · Wasserstein distance · Linear programming · Surface reconstruction · Shape simplification · Feature recovery.

Mathematics Subject Classification (2000) 65D17 · 65D18

1 Introduction

Surface reconstruction is a multi-faceted challenge which precise problem statement depends on the nature and defects of the input data, the properties of the inferred surface (smooth vs piecewise smooth, with or without boundaries), and the desired level of detail one wishes to capture. Despite a number of major contributions over the past decade [5,24], achieving both feature preservation and robustness to measurement noise and outliers remains a scientific challenge—and a pressing requirement for many reverse engineering and geometric modeling applications. Furthermore, low polygon-count reconstructions has only received limited attention despite the increase of point density in 3D scanning technology and the need for efficient subsequent geometry processing.

In this paper we contribute a reconstruction method that simplifies an initial (possibly non-manifold) triangulation of the input point set, based on an error metric that quantifies through optimal mass transport the distance between the current simplicial complex and the input points. Our reconstruction approach inherits the qualities of our optimal transport based metric: it is resilient to noise and outliers, can handle uneven sampling, yet it finely captures boundaries and sharp features. We demonstrate these distinguishing properties on a series of examples. Applied to the output of a feature-lossy reconstruction method, our new metric can also be used in order to recover sharp features and boundaries through a vertex relocation process.

2 Previous Work

We first discuss existing surface reconstruction methods, restricting our review to approaches that are robust to noise and outliers as well as feature preserving. We then point out recent, relevant work on geometry processing based on optimal transport.

J. Digne · D. Cohen-Steiner · P. Alliez
Inria Sophia Antipolis - Méditerranée

F. de Goes · M. Desbrun
California Institute of Technology

2.1 Surface Reconstruction

A common approach to robust surface reconstruction from defect-laden point sets involves denoising and filtering of outliers, and often requires an interactive adjustment of parameters. Automatic methods such as spectral methods [25,47,4] and graph cut approaches [20, 26] have been proven extremely robust but are better suited to the reconstruction of smooth, closed surfaces. More recently, Cohen-Or and co-authors have proposed a series of contributions based on robust norms and sparse recovery [29,21,7]. An interpolating, yet noise robust approach was alternatively proposed by Digne et al. [13] through the construction of a scale space.

Feature preserving methods are typically based on an implicit representation that approximates or interpolates the input points. In [14], for instance, sharp features are captured through locally adapted anisotropic basis functions. Adamson and Alexa [1] proposed an anisotropic moving least squares (MLS) method instead, using ellipsoidal mapping functions based on principal curvatures. More recently, Oztireli et al. [35] extended the MLS surface reconstruction through kernel regression to allow for much sharper features. However, none of these techniques returns truly sharp features: reconstructions are always semi-sharp, that is, still rounded with various degrees of roundness depending on the approach and the sampling density. Moreover, the presence of sharpness in the geometry of a point set is detected only locally, which often leads to fragmented creases; the reconstruction quality thus degrades quickly if defects and outliers are present.

Another way to detect local sharpness within a point set consists in performing a local clustering of estimated normals [34]: if this process reveals more than one cluster of normals, then the algorithm fits as many quadrics as the number of clusters. Improved robustness was achieved in [16] by segmenting neighborhoods through region growing. Lipman et al. [28], instead, proposed a systematic enrichment of the MLS projection framework with sharp edges driven by the local error of the MLS approximation. Again, the locality of the feature detection can generate fragmented sharp edges, much like general feature detection approaches (e.g., [19,36]).

To reduce crease fragmentation, a different thread of work aims at extracting long sharp features. Pauly et al. [37], for instance, used a multi-scale approach to detect feature points, and constructed a minimum-spanning tree to recover the most likely feature graph. Daniels et al. [12] used a robust projection operator onto sharp creases, and grew a set of polylines through projected points. Jenke et al. [23] extracted feature lines by robustly fitting local surface patches and by computing the intersection of close patches with dissimilar normals.

Shape simplification has also been tackled in [8], but in a coarse-to-fine manner: a random initial subset of the input point cloud and a signed distance function over the set is built. Using this function, points are added until a significant number of points lie within an error tolerance. The augmented set is triangulated and a surface mesh is reconstructed. Thus the method also interleaves reconstruction with simplification. In [2] and [3], a surface is reconstructed through point set simplification and local coarsening or refinement of the mesh. Salman et al. [45] proposed to detect features within the point set (as in [33]) and combined Delaunay refinement over features and Poisson reconstruction on smooth parts of the inferred surface [24].

Contributions. In this paper, we adopt a very different reconstruction methodology: we reconstruct a shape through an iterative, feature-preserving simplification of a simplicial complex constructed from the input point set. To achieve noise and outlier robustness, an error metric driving the simplification is derived in terms of optimal transport between the input point set and the reconstructed mesh, both seen as mass distributions (or equivalently, probability measures) in $\mathbb{R}^3$.

Next we provide a brief review of optimal transport, and mention its applications to various problems in computer graphics and computer vision.

2.2 Optimal Transport

The problem of transporting a measure onto another one as a way to quantify their similarity has a rich scientific history. For two measures μ and ν defined over $\mathbb{R}^3$ and of equal total mass (i.e., their integrals are the same), the L_2 optimal transport from μ to ν consists in finding a transport plan π that realizes the following infimum:

$$\inf \left\{ \int_{\mathbb{R}^3} \|x-y\|^2 \mathrm{d}\pi(x,y) \,\middle|\, \pi \in \Pi(\mu,\nu) \right\},$$

where $\Pi(\mu,\nu)$ is the set of all possible transport plans between μ and ν [46]. In a nutshell, a transport plan π is a displacement that maps every infinitesimal mass from the input measure μ (here, the set of points) to the target measure ν (here, the simplicial complex). This formulation is particularly well suited to comparing 1D measures such as histograms over the real line or on the circle [40], and it has been used for transferring color and contrast between images [43,39]. For applications in higher dimensions such as 2D or 3D shape retrieval [44,41] and segmentation [38], the optimal transport formulation is notoriously less tractable: solving

the optimal transport problem requires linear programming (LP). The LP formulation of optimal transport has been used in applications such as surface comparison [30] and displacement interpolation [9]. Attempts to design computationally simpler surrogates have also been made; the *sliced Wasserstein* approach [42], for instance, consists in approximating the transport problem by a series of 1D problems through projection.

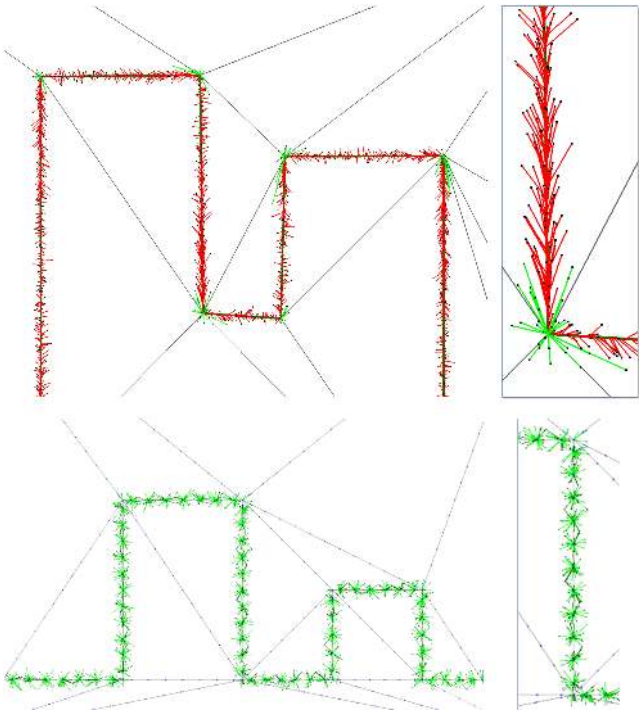

Fig. 1 Transport plans. Top: Binary transport plan [18]. We depict the input point set and simplified triangulation. Red and green line segments depict the transport plan between the input point set and the uniform measure on respectively the vertices and edges of the triangulation. Each input point is simply transported either to its closest edge or to the end vertices of that edge. For each edge the transport plan favors a transport to its end vertices instead of to the whole edge when the corresponding transport cost is lower. A closeup reveals a spurious tangential component of the transport near corner vertices (as indicated by red segments pointing towards the corner), artificially creating a higher total transport cost. For more complex features such as concave corners on surfaces, such behavior leads to reconstruction artifacts. Bottom: our transport plan is, as expected, mostly normal to the edges.

Contributions. Our reconstruction method also relies on a linear programming formulation. However, our approach introduces a key distinctive property: our target measure ν is not given, but instead, solved for. More specifically, we search for the simplicial complex of a user-specified size that minimizes the cost of transporting the input pointwise measure (i.e., the initial point set) to the complex simplices. This specific setup bears a resemblance to what is known as the optimal location problem [31]), where the source measure is given but the target measure is only partially known. Yet a significant difference lies in the type of constraints we are enforcing on the target measure, rendering current computational methods to solve this problem not appropriate to our context. Another line of research for finding a transportation plan between an input point set and a set of discrete sites of various capacities [6,22,32] make use of power diagrams, and are thus likely to be too computationally costly for our context.

The closest work to ours was proposed by de Goes et al. [18]. Their algorithm reconstructs and simplifies 2D shapes from point sets also based on optimal transport. Nonetheless, our approach differs from theirs in several aspects:

1. their optimal transport involves only points and edges and therefore can be computed in closed form. To our knowledge, no such closed form exists when transporting points to the facets of a simplicial complex. Therefore we use a discretized formulation of the optimal transport problem.
2. the authors of [18] propose to approximate the optimal transport plan by assigning each input point to its closest edge in the triangulation. Such a simplistic scheme can lead to a sub-optimal transport plan and cost as illustrated in Figure 1(top)—even more so in 3D. Our discretized formulation, combined with a linear programming solver, provides better approximations of both the optimal plan and the optimal cost.
3. their method requires a valid embedding of a 2D triangulation, which they achieved through a recursive edge flip procedure. Such an edge flip procedure can not, however, be generalized to 3D triangulations. Instead, our method removes the embedding requirement by only employing a (possibly non-manifold) simplicial complex, initially chosen as a subset of a 3D Delaunay triangulation.

2.3 Overview

Motivated by the concept of reconstruction introduced in 2D by de Goes et al. [18], we present a fine-to-coarse algorithm which reconstructs a surface from a point set through greedy simplification of a 3D simplicial complex. We initialize the complex with a (possibly non-manifold) subset of the 3D Delaunay triangulation of input points, then we perform repeated decimations based on half-edge collapse operations. The error metric guiding our simplification is derived from the optimal cost to transport the input point set (seen as

Dirac measures) to a constant-per-facet measure defined over the simplicial complex. At each iteration, we collapse the half-edge which minimizes the increase of total transport cost between input points and reconstructed triangulation. Just like for the formulation presented in [18], our optimal transport driven metric brings desirable properties that are rarely satisfied by current reconstruction methods, such as resilience to noise and outliers, and preservation of sharp features and boundaries.

In the remainder of this paper, we first discuss the details of our optimal transport based metric (Sec. 3) then describe our reconstruction algorithm step by step (Sec. 4). Our method is summarized in Algorithm 1 and its main stages are illustrated in Figure 2.

Algorithm 1: Algorithm overview.

Input : Point set $\mathcal{S}$, user-specified value V.
Output: Simplicial complex $\mathcal{C}$ with V vertices.
Construct 3D Delaunay Triangulation $\mathcal{T}$ from $\mathcal{S}$;
Compute transport cost from $\mathcal{S}$ to facets of $\mathcal{T}$;
Construct simplicial complex $\mathcal{C}$ from facets of $\mathcal{T}$ with non-zero measure;
Decimate $\mathcal{C}$ until desired number of vertices V;
Filter out facets of $\mathcal{C}$ by thresholding mass density.

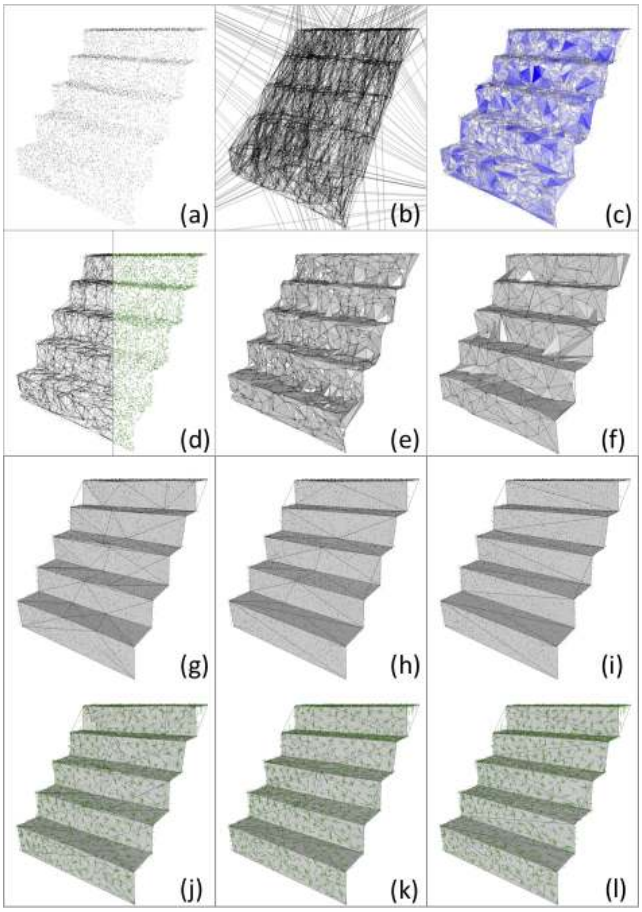

Fig. 2 Steps of our algorithm: (a) Initial point set; (b) 3D Delaunay triangulation of a random subset containing 10% of the input points; (c) Initial simplicial complex constructed from facets of the 3D triangulation with non-zero measure; (d) Initial transport plan assigning point samples to bin centroids (green arrows); (e-f) Intermediary decimation steps; (g-i) Reconstruction with 100, 50, and 22 vertices, respectively; (j-l) Final transport plan with 100, 50, and 22 vertices, respectively.

3 Transport Formulation of Reconstruction

We consider the reconstruction problem of turning an input point set S into a coarse simplicial complex $\mathcal{C}$. The point set contains N points at locations $\{p_i\}_{i=1\cdots N}$, and each point is given a mass m_i that reflects its measurement confidence (all masses are set to a constant if no confidence is provided). Our reconstruction method is based on considering both the point set and the complex as mass distributions (or equivalently, probability measures), where the measure (mass density) of $\mathcal{C}$ is constant per simplex and possibly 0. Our approach then consists in finding a compact shape $\mathcal{C}$ that minimizes the optimal transport cost between the input point set S and a uniform measure on each facet and vertex of $\mathcal{C}$.

In [18], a similar, yet 2D optimal transport cost between $\mathcal{S}$ and $\mathcal{C}$ was efficiently approximated based on closed form expressions for the optimal cost between points and edges. However, to our knowledge, such closed form can not be extended between points and triangles. We present instead, using linear programming (LP), a discretized formulation of the optimal transport between $\mathcal{S}$ and $\mathcal{C}$ that we will solve for later on through local relaxations.

3.1 Discretization

We approximate the optimal transport cost between the input point set $\mathcal{S}$ and the simplicial complex $\mathcal{C}$ using quadrature. We start by defining a set $\mathcal{B}$ of *bins* (small regions of the complex) over $\mathcal{C}$. As we aim at reconstructing piecewise smooth surfaces from point sets, facet bins are necessary—edge bins could be used as well if curves in $\mathbb{R}^3$ were sought after as well; for simplicity, we do not discuss this extension. However, vertex bins are useful as well: when outliers are present, vertices serve as garbage collectors. Vertex and facet bins are thus used to evaluate the optimal cost between $\mathcal{S}$ and $\mathcal{C}$ as a sum of squared distances between the points in $\mathcal{S}$ and the centroids of the bins in $\mathcal{B}$.

First, every vertex of $\mathcal{C}$ is considered as (the center of) its own bin; each triangular facet is, instead, tiled with bins using a 2D Centroidal Voronoi Tessellation (CVT) (Figure 3); note that our choice of a CVT tiling stems from the fact that it minimizes the approximation error given by quadrature points put at their centroids [15], which will thus provide optimal approximation of our transport cost. The number of bins per facet is set based on a user-defined quadrature parameter. In all our experiments, we used 200 bins per unit area, the point sets being included in a half-unit side size box (note that the facet bins of fig 3 are purposedly generated with a higher density). Finally, to compensate for a slightly non-uniform distribution of bins, we assign a *capacity* for each bin in $\mathcal{B}$ (i.e., ratio of the total amount of mass that a bin can receive over the total amount of mass transported to the simplex the bins belongs to): vertex bins are set to unit capacity (since there is only one vertex per bin), while each facet bin is given a capacity equal to the ratio between its area (i.e., the area of the associated centroidal Voronoi cell) and the area of its containing facet. Finally, the centroids of the bins in $\mathcal{B}$ are computed and stored as representatives of their bins.

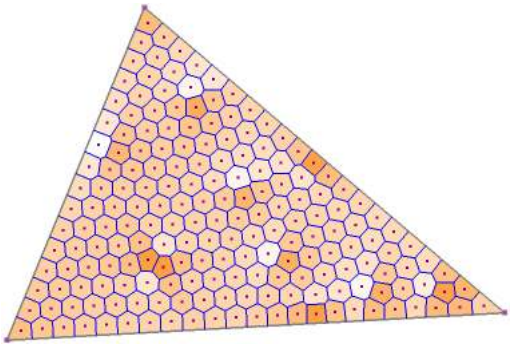

Fig. 3 Bins of a facet. Bins in a facet are defined as cells of a centroidal Voronoi tessellation. Bin centroids are depicted as red dots. Capacities of the bins (set proportional to their areas) are depicted using a thermal color ramp.

3.2 Linear Programming Formulation

We now present a linear programming formulation to compute the optimal transport cost between the input point set $\mathcal{S}$ and the bin set $\mathcal{B}$. In the following, we denote the simplices of $\mathcal{C}$ as $\{\sigma_j\}_{j=1\cdots L}$ and the centroids of the bins in $\mathcal{B}$ as $\{b_j\}_{j=1\cdots M}$, where L and M are the number of simplices and bins respectively. The capacity of bin b_j is denoted c_j. We also define $s(j)$ to be the index of the simplex containing the bin b_j (i.e., $b_j \in \sigma_{s(j)}$). Finally, we denote by m_{ij} the amount of mass transported from a given input point $p_i \in S$ to the centroid b_j (Figure 4).

With these definitions, we can now formally refer to a *transport plan* between $\mathcal{S}$ and $\mathcal{B}$ as a set of $N \times M$ variables m_{ij} such that:

$$\forall ij : \; m_{ij} \geq 0, \tag{1}$$

$$\forall i : \; \sum_j m_{ij} = m_i, \tag{2}$$

$$\forall j_1, j_2 \text{ s.t. } s(j_1) = s(j_2) : \; \frac{\sum_i m_{ij_1}}{c_{j_1}} = \frac{\sum_i m_{ij_2}}{c_{j_2}}, \tag{3}$$

where Equation 2 ensures that the entire measure of an input point gets transported onto the mesh $\mathcal{C}$, and Equation 3 ensures a uniform measure over each facet of $\mathcal{C}$.

An *optimal transport plan* is then defined as a transport plan π that minimizes the associated transport cost:

$$cost(\pi) = \sum_{ij} m_{ij} \|p_i - b_j\|^2.$$

Finding a transport plan minimizing the transport cost results in a linear program with respect to the m_{ij}, with equality (Eq. 2 and 3) and inequality constraints (Eq. 1). Note that the number of bins, their positions, as well as the square distances between input points and bin centroids are all precomputed. In order to enforce the uniformity constraint (Eq. 3) more sparsely, we also introduce L additional variables l_i (one per simplex σ_i) indicating the target measure density of the corresponding simplex. The final problem formulation is thus:

$$\text{Minimize} \quad \textstyle\sum_{ij} m_{ij} \|p_i - b_j\|^2$$

w.r.t. the variables m_{ij} and $l_{s(j)}$, and subject to:

$$\begin{cases} \forall i : \displaystyle\sum_j m_{ij} = m_i \\ \forall j : \displaystyle\sum_i m_{ij} = c_j \cdot l_{s(j)} \\ \forall i,\ j : m_{ij} \geq 0,\ l_{s(j)} \geq 0 \end{cases}$$

3.3 Local Relaxation

Solving directly for the formulation described above is compute-intensive due to the number of variables and constraints involved: it requires instantiating a dense matrix (representing the constraints) of size $(M \times N + L) \times (M + N)$. For example, computing the optimal transport cost between an input point set of $2,100$ samples and a simplicial complex containing 782 simplices on which $7,300$ bins are placed involves solving for a linear program of over 15 million variables and $9,000$

constraints. Alas, linear programming solvers do not scale up well to such large numbers.

In order to improve scalability we propose an iterative and local relaxation strategy instead, as summarized in Algorithm 2. A subset of the global solution space is explored through *local* LP solves over small stencils, until a local minimum of the objective function is reached. Note that we cannot guarantee convergence of this local procedure to the global minimum; but the minima reached in practice have consistently provided satisfactory results in all of our tests.

Our procedure starts with a trivial transport plan which maps each input point to its nearest vertex of the simplicial complex $\mathcal{C}$. Since no uniformity constraints are imposed on vertices, this transport plan is valid, yet obviously suboptimal in general. Subsequent local optimizations will only decrease the global transport cost or leave it unchanged, as local re-assignments are made only if they generate smaller or equal cost after optimization. The transport cost found through our local stencil updates is thus an upper bound of the global optimal transport cost. Our experiments showed, unsurprisingly, that the convergence rate of this procedure depends on the shape of local stencils used: the larger the stencil, the faster the convergence—but with the unfortunate side effect that large stencils increase the size of the corresponding linear program. We found in practice that simply using the 1-ring of a chosen simplex is a rather reliable choice. More precisely, the local stencil $\mathcal{N}$ of a facet σ is defined as all the facets incident to σ, along with their vertices.

Armed with this scalable approximation of the transport cost, we describe next how we put it to work for surface reconstruction through simplification.

Algorithm 2: Local stencil relaxation overview.

Input : Simplicial complex $\mathcal{C}$, point set $\mathcal{S}$, threshold ε
Output: Locally optimal transport plan $\pi = \{m_{ij}\}$
for $p_i \in \mathcal{S}$ **do**
  Transport p_i to nearest vertex $v \in \mathcal{C}$;
`new_cost` $\leftarrow 0$;
repeat
  for $\sigma_j \in \mathcal{C}$ **do**
    `old_cost` $\leftarrow$ `new_cost`;
    Build the stencil $\mathcal{N}$ of the facet σ_j;
    Collect sample points and partial measures $\{p_i, \widetilde{m}_i\}$ transporting onto this stencil;
    Solve the linear program to find the optimal transport plan of $(p_i, \widetilde{m}_i)$ onto the bins of $\mathcal{N}$;
    Update transport plan π and cost `new_cost`;
  $\delta =$ `new_cost` $-$ `old_cost`;
until $\delta \leq \varepsilon$;

4 Reconstruction through Simplification

4.1 Initialization

We begin our reconstruction process by randomly picking a subset of the input points $\mathcal{S}$ and computing a 3D Delaunay triangulation. We then construct a simplicial complex $\mathcal{C}$ from a subset of facets of this 3D triangulation. To select this subset of facets, we perform two steps: (1) we reuse the local stencil relaxation method (Algorithm 2) to estimate a transport cost from all the input points onto the facets and vertices of the 3D triangulation; (2) we then build $\mathcal{C}$ with only the facets containing non-zero transported measure. For step (1), we use a stencil centered at each facet and containing vertices and edges of the two tetrahedra adjacent to the facet. For an inside facet of the triangulation, for instance, this stencil contains 7 facets and 5 vertices. Optimization is then performed by going over all stencils of the triangulation. This stencil-based optimization is repeated until the decrease in transport cost is below a user-specified threshold (set to 10^{-5} in all our tests). In practice, the global transport cost decreases rapidly, and we need to go over all stencils only 10 times at most. For step (2), we convert our data structure to a simplicial complex for two main reasons: first to allow our reconstruction to have long and anisotropic simplices; and more importantly, to remove the difficult issue identified in [18] of keeping the embedding of the triangulation valid during decimation.

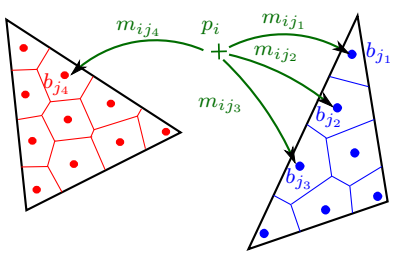

Fig. 4 Transport plan for a single input sample point p_i. Variable m_{ij} models the transport of the mass m_i at an input point p_i to the j^{th} bin of the facet.

The initial Delaunay is built by taking a random subset of the point set for efficiency. However, if the subset is small enough we risk not having enough degrees of freedom for representing the shape. Though we do not have a theoretical guarantee for this subsampling, we start in practice with only 10 to 20% of the samples as it is usually above our target number of vertices and sufficient to capture enough details. (Another option could be to filter the initial Delaunay not based on the transport but on the edge length, but this would result in a much larger initial simplicial complex.)

4.2 Decimation

From the initial simplicial complex $\mathcal{C}$, we further simplify the reconstruction through a greedy decimation based on *half-edge collapse operations*. Note, however, that a conventional decimation algorithm (e.g., [17,27] and variants) can not be applied in our setup: the presence of outliers and noise renders typical error metrics inadequate.

Our optimal transport framework provides a robust alternative: we pick the next half-edge to collapse as the one that induces the least increase in global transport cost. To this end, we simulate the collapse of a candidate half-edge e and evaluate the induced change of transport cost Δ. Since this cost change mostly affects a neighborhood Ω_e of e, we can restrict the computation of Δ only to Ω_e. More specifically, setting Ω_e to the closure of simplices in the 1-ring of e, we first gather the set of samples p_i transporting (partially or entirely) on Ω_e (Figure 5), adding up the already computed transport cost of this set of (possibly partial) samples to Ω_e, simulate the collapse of e, and recompute the cost of transporting the set of samples onto the resulting simplices. The change of transport cost Δ is then set to the difference of transport cost before and after the simulated collapse of e. Once a half-edge is selected and collapsed, we also update the transport plan of the edges for which their 1-rings intersect the one-ring of this collapsed edge. Finally, we increase scalability by employing a multiple choice approach [48]: the next half-edge to be collapsed is selected from only a small set of randomly selected edges as recommended in [18], instead of maintaining an compute-intensive exhaustive priority queue. This decimation process is summarized in Algorithm 3.

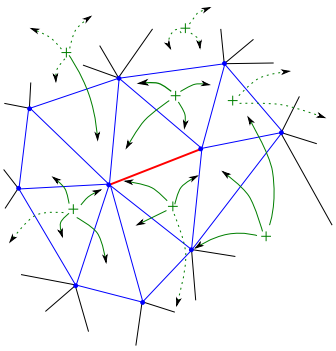

Fig. 5 Local stencil of an edge. For better depiction we represent a manifold neighborhood of an edge (in red) and do not depict the bins. Simplices in the local stencil are depicted in blue. Point samples are depicted in green. We only solve for the measures transported to the stencil (solid green lines) and not for the measures transported outside of the stencil (dash green lines).

Algorithm 3: Decimation algorithm.

Input : Simplicial complex $\mathcal{C}$, input point set $\mathcal{S}$, target number of vertices V
Output: Simplicial complex $\mathcal{C}_{\text{final}}$ with V vertices
for *each edge* $e \in \mathcal{C}$ **do**
  Simulate two half-edge collapse operators;
  Push these half-edges to a priority queue $\mathcal{P}$, sorted by change of transport cost Δ.
repeat
  Pop half-edge e^* out of $\mathcal{P}$;
  Collect set $\mathcal{E}$ of edges whose neighborhoods intersect neighborhood of e^*;
  Collapse e^* and update transport plan on the neighborhood of e^*;
  Update $\mathcal{P}$ by recomputing the change of cost Δ for all edges in $\mathcal{E}$.
until *the simplicial complex has* V *vertices*;

4.3 Vertex Relocation

So far our method based on half-edge collapses results in an interpolating reconstruction, since vertices of the final complex can only be a subset of the input points. This may lead to suboptimal results, even more so in the presence of noise and outliers. We thus couple our decimation with an optimization procedure in order to relocate the vertices in the reconstructed simplicial complex $\mathcal{C}$. After the collapse of a half-edge e, the remaining vertex v of e is relocated by iterating two steps: (1) for a given transport plan π, we move v toward the position that best improves the optimal cost of π; (2) then we update π around v accordingly. For the first step, we compute the locally optimal position of v when the transport plan π (i.e., the values m_{ij} for all i and j) is kept fixed. To this end, we express the position of each centroid of the facet bins in barycentric coordinates within its containing triangle. Then finding the optimal position of vertex v of triangle $t = (v, v_1, v_2)$ amounts to minimizing:

$$\min_v \sum_i \sum_j m_{ij} \|p_i - \alpha_j v - \beta_j v_1 - \gamma_j v_2\|^2,$$

where $\alpha_j, \beta_j, \gamma_j$ are the barycentric coordinates of the centroid of bin b_j with respect to vertices (v, v_1, v_2). The optimal position with respect to triangle t is:

$$v^\star(t) = \frac{\sum_i \sum_j m_{ij} \alpha_j (p_i - \beta_j v_1 - \gamma_j v_2)}{\sum_i m_{ij} \alpha_j^2}.$$

Thus each triangle t adjacent to v yields an optimal position $v^\star(t)$. Furthermore, the vertex itself may have input points assigned to its bin, so that we must add the vertex contribution to its own relocation:

$$v^\star(v) = \frac{\sum_i m_{ij} p_i}{\sum_i m_{ij}}.$$

with m_{ij} being the mass portion of sample p_i assigned to vertex bin b_j of v. Thus each simplex (vertex or facet) adjacent to vertex v contributes an optimal position for v. The final position $v^\star$ is then chosen as an average of optimal positions weighted by their corresponding mass:

$$v^\star = \frac{m(v) \cdot v^\star(v) + \sum_{t \text{ adjacent to } v} m(t) \cdot v^\star(t)}{m(v) + \sum_{t \text{ adjacent to } v} m(t)},$$

where $m(t)$ is the total mass transported to simplex t (corresponding to variable l_i in the general LP formulation provided i is the index of simplex t). Vertex v is finally moved at the midpoint between its current position and the optimal position $v^\star$.

For the second step, we freeze the vertex locations and update the transport map π by solving the local linear program (Algorithm 2). By alternating these two steps, the vertices move to their locally optimal position, allowing for a better recovery of sharp features and surface boundaries. Figure 6 depicts a simple vertex relocation sequence in 2D for clarity.

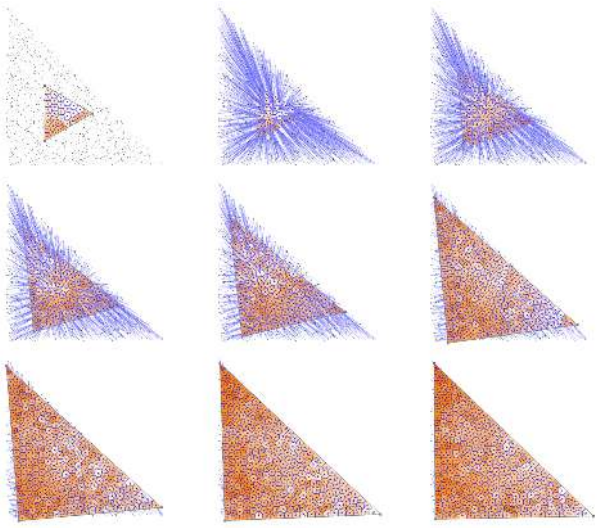

Fig. 6 Vertex relocation. For visual clarity we choose a 2D example with a single triangle and only facet bins. We first depict the input point set, here uniformly sampled on a triangle, the initial simplicial complex composed of one facet, and the facet bins and their capacities. For all subsequent images we depict the transport plan throughout the vertex relocation process with blue edges connecting the source point samples and their target bin centroids.

4.4 Facet Filtering

When the decimation terminates, we could return as our final reconstructed mesh the subset of facets from $\mathcal{C}$ that carry a non-zero measure. However, facets may have non-zero measure due the presence of noise and outliers; we thus found convenient to sort the facets based on their measure density (i.e., the ratio of facet measure to its area) and provide the user with an interactive slider to decide which threshold is most appropriate. Figure 7 shows the reconstructed surface obtained with different filtering thresholds.

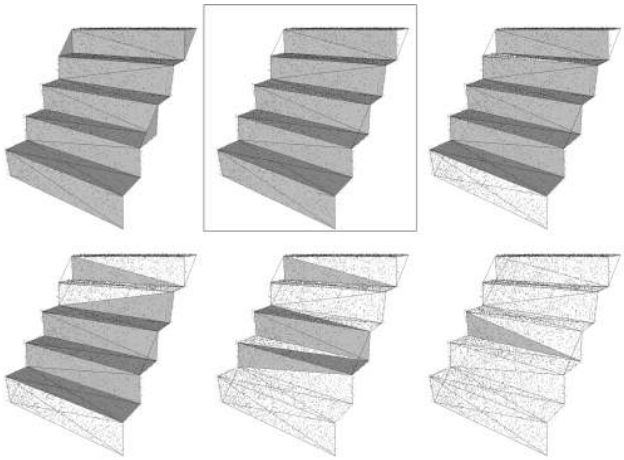

Fig. 7 Facet filtering. For a noisy input point set, the simplicial complex returned by our decimation scheme contains facets with small, but non-zero measure (top left). The other images (top to bottom, left to right) show the result of gradually increasing the threshold during the final filtering of the facets. The best reconstruction in this example is highlighted in a frame.

4.5 Experimental Results

We implemented our algorithm in C++ using CGAL's 3D Delaunay triangulation [10] to initialize the reconstruction, and our own data structure for simplicial complexes. We used the Coin-OR Clp library [11] as our linear program solver. Our implementation is partially parallelized to accelerate computations, exploiting the fact that all half-edge collapse simulations are independent. The initialization and update of the priority queue are, by far, the most costly operations, as each collapse involves around 120 simulations on average. When using the exhaustive priority queue on a laptop with a two-core processor, a point set containing $30,000$ points is reconstructed in around 10 hours (initial and final simplicial complexes containing respectively $3,000$ and

200 vertices). On a 8-core computation server, this computation reduces to 2 hours (note that the computation time reduction is not only due to parallelization but also to a faster clock). However, when using a multiple-choice approach with random sets of 40 collapses (as we did in all results shown), the timings are three times faster on average. The typical breakdown of computational time spent on each phase of the algorithm is as follows: building the initial Delaunay mesh and filtering it takes around 5% of the total computation time; in the remaining iterative process, 70% of the time is spent in solving linear problems (needed for collapse simulation and reassignment), 20% of the time in assembling the LP systems, and the remaining 10% is spent on performing the collapses. Throughout these computations, memory consumption remains low; e.g., for the particular experiment mentioned above, the peak memory usage was around $80Mb$.

Robustness to noise. We tested our method on a point set sampling a staircase shape with an increasing amount of synthetic, uniform noise (Figure 8). Even in the presence of significant noise, the method tends to recover the creases of the input shape well. Only for noise magnitudes larger than 5% of the bounding box size does our method fail: for such high noise levels, spurious facets cannot be discarded by a simple thresholding based on mass density. Our method can however robustly handle pointsets from current point acquisition devices, as they generally contain noise magnitudes below this failure regime.

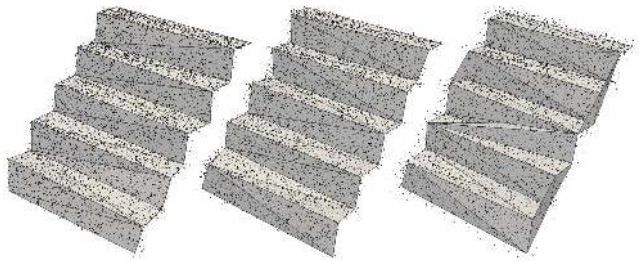

Fig. 8 Robustness to noise. We increase the amount of synthetic noise from $\sigma = 1\%$ to $\sigma = 2\%$ and $\sigma = 5\%$, expressed in percentage of the longest edge length of the bounding box. The reconstruction starts failing at $\sigma = 5\%$.

Robustness to outliers. We also tested our method on a point set that samples a cylinder (Figure 9). Results are excellent up to 15% of outliers, but our method can fail when the amount of outliers exceeds 20%—again, current acquisition devices and stereophotogrammetric methods are usually good enough not to reach this amount of outliers.

Feature preservation. Figure 10 depicts the feature preservation property of our approach on the blade model. Our approach performs well even on thin features subtending small angles, for which implicit approaches (here, the noise-robust Poisson surface reconstruction method of [24]) tend to smooth out features and create spurious topological artifacts on low point density regions.

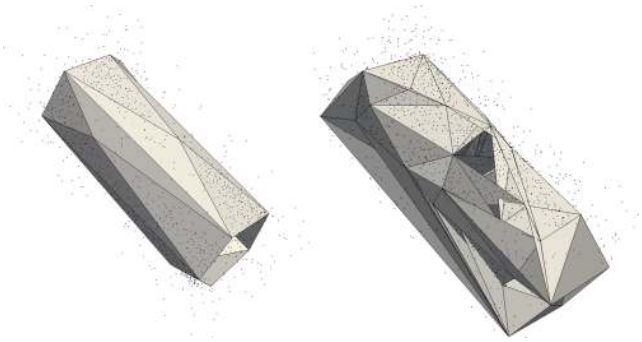

Fig. 9 Robustness to outliers. The reconstruction is effective even with 10% outliers (left; compare to outlier-free input in Fig. 12) but fails from 20%. The outliers are added randomly within a loose bounding box (120%) of the input point set.

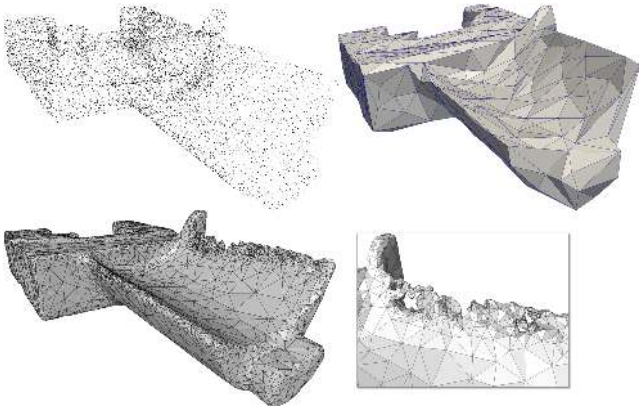

Fig. 10 Reconstruction of the blade model containing $30K$ sample points. Top: our reconstruction. Bottom: the output of the Poisson reconstruction method (with Delaunay refinement used for contouring the resulting implicit function), and closeup on a sharp crease subtending a small angle, where the implicit approach fails.

On the cone model in Figure 11, all features (tip, boundaries) are preserved and the simplification is very effective. Similarly, on a cylinder model (Figure 12) the boundaries are preserved and the simplification leads to anisotropic triangles with most edges aligned with minimum curvature directions as expected. Figure 13 also illustrates boundary and sharp feature preservation, this time on a twisted bar.

Figure 14 illustrates the behavior of our approach on two intersecting planar polygons. The algorithm behaves well down to 10 vertices, and the simplicial com-

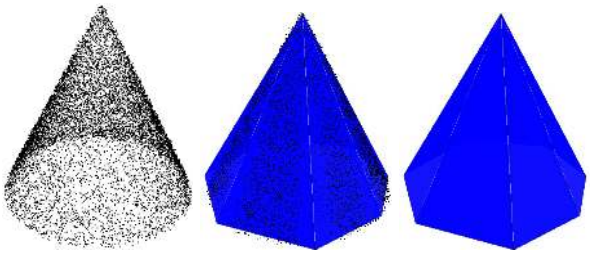

Fig. 11 Reconstruction of a cone. Left: input point set. Middle: input point set and final simplicial complex with (nearly uniform) facet densities shown. Right: final complex.

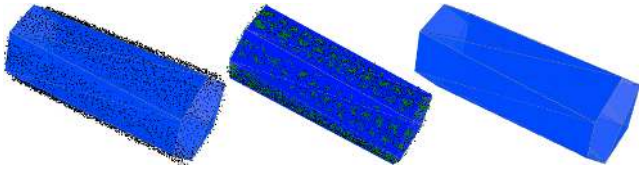

Fig. 12 Reconstruction and simplification of a cylinder. Left: $10K$ noisy sample points and reconstruction with 12 vertices (facet density shown). Middle: transport plan between point samples and bin centroids. Right: simplicial complex and facet density.

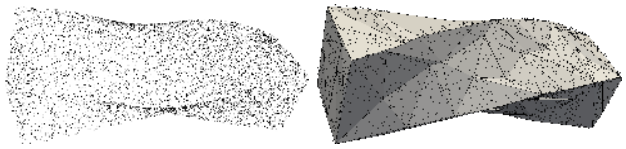

Fig. 13 Reconstruction of a twisted bar. Sharp features are well preserved.

plex maintains the initial topology during decimation. Going down to 8 vertices (the expected minimum number of vertices) would require a richer set of topological operators in order to disconnect the intersecting edge before pursuing decimation; we did not pursue this particular extension.

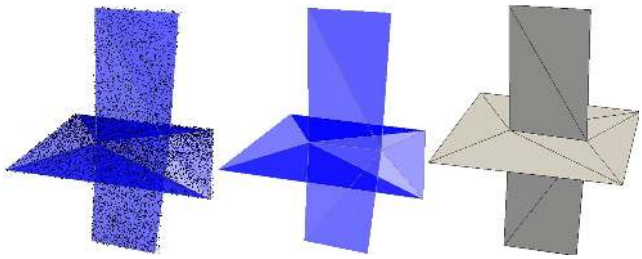

Fig. 14 Reconstructing and simplification of two intersecting planar polygons until 10 vertices.

Figure 15 shows the performance of the method on a LIDAR point cloud. Even with these noisy data, our method recovers the features of the shapes and produces a low complexity mesh.

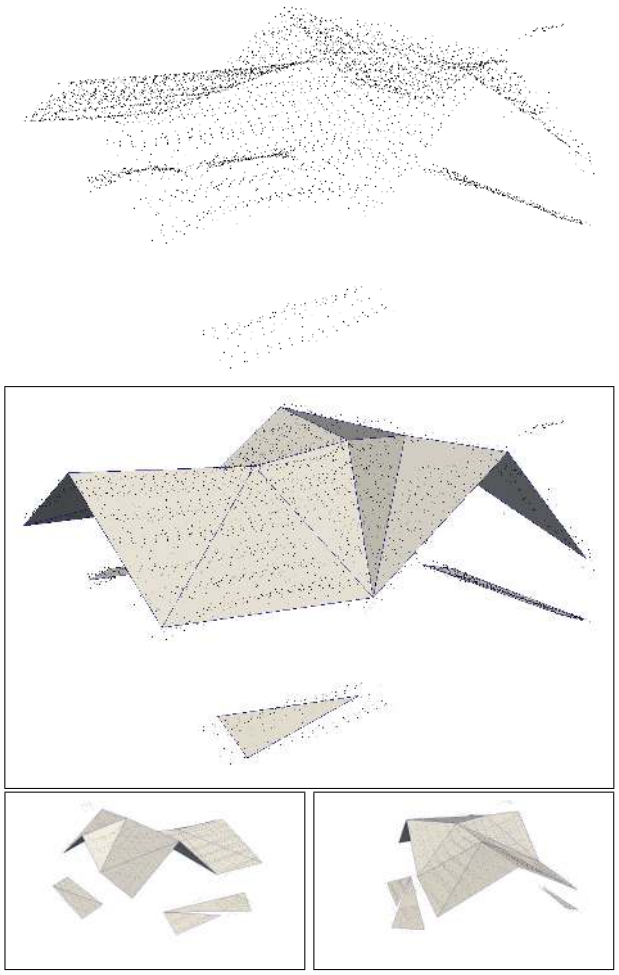

Fig. 15 Reconstruction of an aerial LIDAR point cloud capturing the rooftop of a house. Top: input point set, middle: final reconstruction, bottom: two other views. The reconstruction yields a very simplified mesh despite the noise. Point set courtesy of Qian-Yi Zhou and Ulrich Neumann.

Weaknesses. Given the efficiency of current linear program solvers, results of our approach come at the price of intensive computations, currently preventing its use on large point sets. Also, there is currently nothing in our formulation that favors 2-manifoldness, as the main data structure is a simplicial complex initialized by the facets of a 3D triangulation; this can lead, in rare occasions, to invalid embedding as well as multiple facets covering the same area (see Figure 16). The latter issue is more complex than just ensuring a 2-manifold reconstruction, as complex features may correspond to non-manifold shapes. One could define a notion of "effectiveness" per facet, but this would lead to a non-linear objective function and require a richer set of topological operators such as facet deletion.

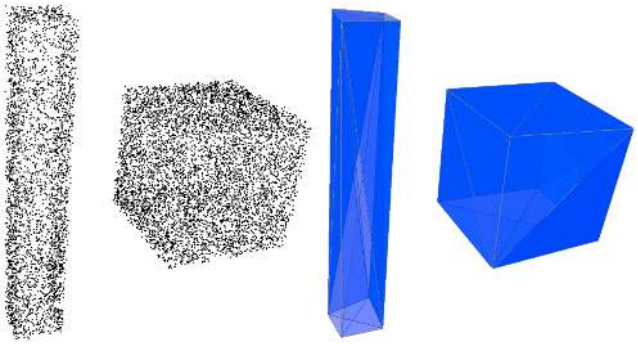

Fig. 16 Reconstruction and simplification of a scene composed of two boxes. Left: $10K$ noisy sample points. Right: reconstruction with 16 vertices. The level of anisotropy matches our expectations but some facets of the boxes are covered twice.

5 Feature Recovery

Another application of our proposed metric is to recover sharp features and boundaries from the output of reconstruction methods that are designed to produce smooth, closed surfaces (e.g., Poisson reconstruction [24]). These approaches are in general scalable and robust to noise, but they round off sharp features and fill up holes, even if a data fitting term is added. We can remediate these artifacts via vertex relocation and facet filtering; this feature recovery method is summed up in Algorithm 4.

Algorithm 4: Feature recovery.

Input : Point set $\mathcal{S}$, reconstructed mesh $\mathcal{T}$.
Output: Feature-capturing mesh $\mathcal{T}$
Compute initial assignment;
for *all vertices of* $\mathcal{T}$ **do**
  Compute the relocation force;
  Move the vertex in the direction of the force;
  Update the transport plan around the vertex;
Filter out facets of $\mathcal{T}$ by thresholding mass densities.

The input of the algorithm is a surface triangle mesh (the output of a smooth reconstruction algorithm) and the original point set used for reconstruction. Bins are first sampled on the mesh. The initial assignment is performed through relaxation as described in Section 3.3: each sample is assigned to the nearest mesh vertex, and local reassignments are iterated until a local minimum for the transport cost is reached. Each mesh vertex is then relocated as described in Section 4.3, by computing the relocation direction, moving the point in this direction, and updating the transport plan. One should notice that this process depends on the mesh vertices traversal order: the first vertex is moved at the midpoint between its current position and the computed optimal position, then the local transport plan is updated, and then the next vertex is handled. The traversal order could be randomized between relocation iterations to avoid potential artifacts. However, all our experiments were obtained using the same traversal order with no visual bias due to this fixed order. Figure 17 demonstrates how sharpness is recovered with this simple post-processing phase.

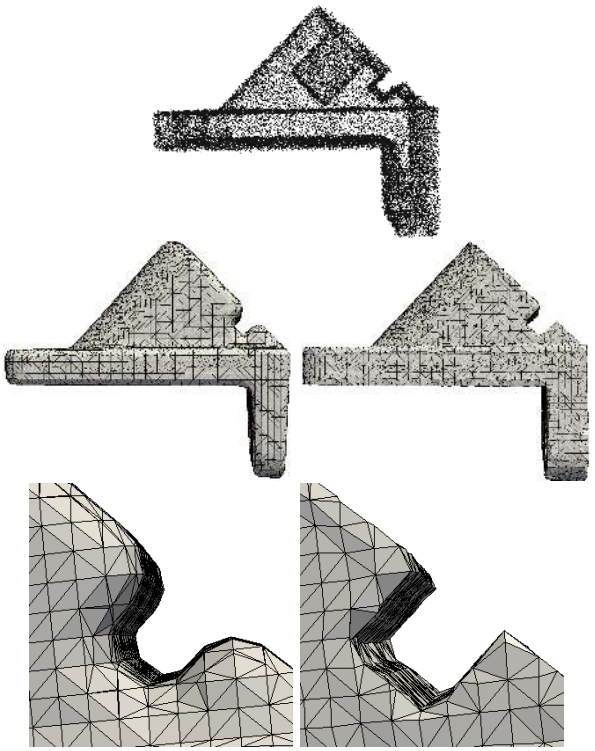

Fig. 17 Anchor. Noisy point set (top), Poisson reconstruction (middle left), improved reconstruction (middle right) and associated closeups.

For open surfaces this method recovers boundaries of the surface through the last filtering step (section 4.4) as can be seen on the church example (Figure 19 and 20). On the latter, the relocation seems incorrect at first glance on the bell tower, but the seemingly spurious triangles created by the relocation procedure correspond to actual geometry in the point set: these details of the shape were lost by the Poisson reconstruction. On the challenging synthetic point set used in Figure 21 the vertex relocation recovers the sharpness of the features as well. In terms of computational cost applying the vertex relocation algorithm on the church mesh (23K vertices, 232K input points) takes around 10 minutes.

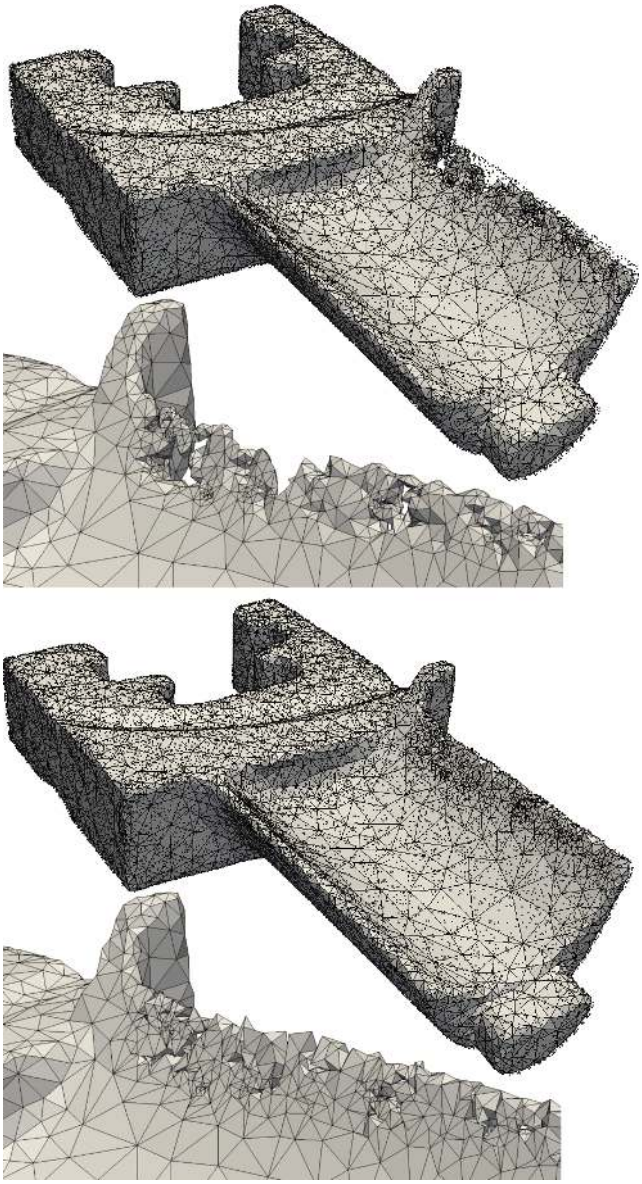

Fig. 18 Blade. Poisson reconstruction (2 top rows) and improved reconstruction (2 bottom rows). The input point set is depicted with black dots on the global views and is not depicted on the close-ups for clarity. Neither remeshing nor edge flips are applied: the spurious topological handles shown in Figure 10 are not repaired, triangles are only pulled closer toward the point set.

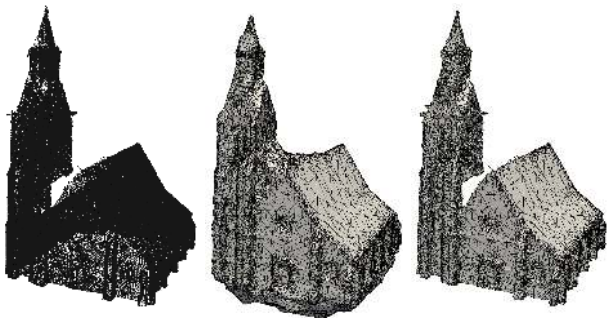

Fig. 19 Church. Point set (left), Poisson reconstruction (middle) and relocated mesh (right).

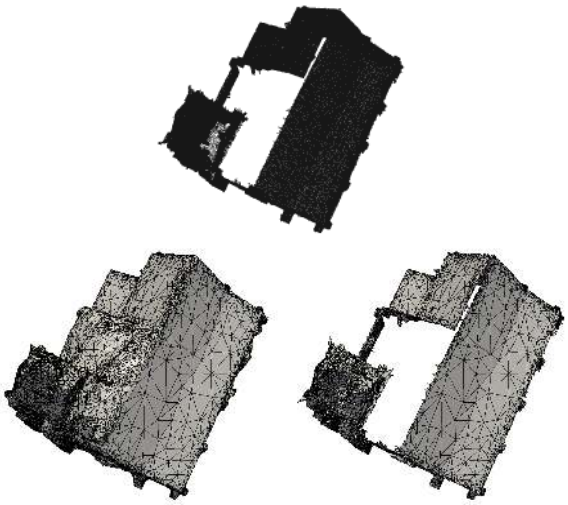

Fig. 20 Church. Point set (top), Poisson reconstruction (bottom left) and its improvement via vertex relocation (bottom right): filtering combined with vertex relocation allows recovery of the surface boundaries.

6 Conclusion

We introduced a surface reconstruction and simplification method which exhibits both robustness to noise and outliers, as well as preservation of sharp features and boundaries. Our approach is based on the decimation of a simplicial complex guided by an optimal transportation error metric between the reconstruction and the initial point set. This error metric was also shown useful as a post-processing phase to recover features from smooth reconstructed shapes.

The main drawback of our approach is its computational cost: despite our efforts to introduce local relaxation, parallelization, and multiple-choice accelerations, we cannot reconstruct large point sets in reasonable time. The main strength of our approach lies in the simplicity of its formulation: it is expressed directly on the simplicial complex being reconstructed, departing from common robust operators that require subsequent contouring to obtain the final reconstructed (but not simplified) surface mesh. In addition, our formulation can be trivially extended to allow for the reconstruction of curves embedded in $\mathbb{R}^3$ by simply adding edge bins to vertex and facet bins. Furthermore, our formulation provides us with a transport plan, which can be used for further geometry processing of the resulting simplicial complex.

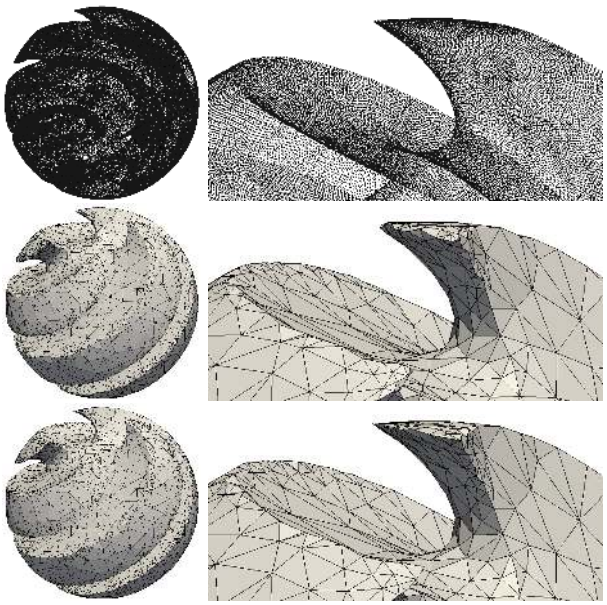

Fig. 21 Sharp sphere. Left column: global view, right column: close-up. From top to bottom: point set, smooth reconstruction, and vertex relocation. Features are recovered through vertex relocation.

As future work we wish to improve scalability. The multi-scale approach of Mérigot [32] is certainly an interesting direction but we believe it requires significant work to be truly practical.

*Acknowledgments.*This work was funded by the European Research Council (ERC Starting Grant "Robust Geometry Processing", Grant agreement 257474). We also thank the National Science Foundation for partial support through the CCF grant 1011944.

References

1. Adamson, A., Alexa, M.: Anisotropic point set surfaces. In: Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, p. 13 (2006)
2. Allègre, R., Chaine, R., Akkouche, S.: Convection-driven dynamic surface reconstruction. In: Shape Modeling International, pp. 33–42. Cambridge, MA, USA (2005)
3. Allègre, R., Chaine, R., Akkouche, S.: A Dynamic Surface Reconstruction Framework for Large Unstructured Point Sets. In: IEEE/Eurographics Symposium on Point-Based Graphics 2006, pp. 17–26 (2006)
4. Alliez, P., Cohen-Steiner, D., Tong, Y., Desbrun, M.: Voronoi-based variational reconstruction of unoriented point sets. In: Eurographics Symposium on Geometry Processing, pp. 39–48 (2007)
5. Amenta, N.: The Crust algorithm for 3d surface reconstruction. In: Symposium on Computational geometry, pp. 423–424 (1999)
6. Aurenhammer, F., Hoffmann, F., Aronov, B.: Minkowski-type theorems and least-squares clustering. Algorithmica **20**, 61–76 (1998). URL http://dx.doi.org/10.1007/PL00009187
7. Avron, H., Sharf, A., Greif, C., Cohen-Or, D.: ℓ_1-sparse reconstruction of sharp point set surfaces. ACM Trans. on Graphics **29**(5), 1–12 (2010)
8. Boissonnat, J.D., Cazals, F.: Coarse-to-fine surface simplification with geometric guarantees. Computer Graphics Forum **20**(3), 490–499 (2001)
9. Bonneel, N., van de Panne, M., Paris, S., Heidrich, W.: Displacement interpolation using Lagrangian mass transport. ACM Transactions on Graphics (SIGGRAPH Asia) (2011)
10. CGAL, Computational Geometry Algorithms Library. Http://www.cgal.org
11. CLP, coin-or linear program solver. Http://www.coin-or.org/Clp/
12. Daniels, J.I., Ha, L.K., Ochotta, T., Silva, C.T.: Robust smooth feature extraction from point clouds. In: IEEE International Conference on Shape Modeling and Applications, pp. 123–136 (2007)
13. Digne, J., Morel, J.M., Souzani, C.M., Lartigue, C.: Scale space meshing of raw data point sets. Computer Graphics Forum **30**(6), 1630–1642 (2011)
14. Dinh, H.Q., Turk, G., Slabaugh, G.: Reconstructing surfaces using anisotropic basis functions. In: International Conference on Computer Vision, pp. 606–613 (2001)
15. Du, Q., Faber, V., Gunzburger, M.: Centroidal Voronoi Tessellations: Applications and algorithms. SIAM Rev. **41**(4), 637–676 (1999)
16. Fleishman, S., Cohen-Or, D., Silva, C.: Robust moving least-squares fitting with sharp features. In: ACM SIGGRAPH 2005 Papers, p. 552 (2005)
17. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: ACM SIGGRAPH, pp. 209–216 (1997)
18. de Goes, F., Cohen-Steiner, D., Alliez, P., Desbrun, M.: An optimal transport approach to robust reconstruction and simplification of 2d shapes. Computer Graphics Forum **30**(5), 1593–1602 (2011)
19. Gumhold, S., Wang, X., MacLeod, R.: Feature extraction from point clouds. In: International Meshing Roundtable, pp. 293–305 (2001)
20. Hornung, A., Kobbelt, L.: Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In: Eurographics Symposium on Geometry Processing, pp. 41–50 (2006)
21. Huang, H., Li, D., Zhang, H., Ascher, U., Cohen-Or, D.: Consolidation of unorganized point clouds for surface reconstruction. ACM Transactions on Graphics **28**(5) (2009)
22. Huesmann, M.: Optimal transport between random measures. ArXiv e-prints (2012)
23. Jenke, P., Wand, M., Straßer, W.: Patch-graph reconstruction for piecewise smooth surfaces. Vision, modeling, and visualization 2008: proceedings p. 3 (2008)
24. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Eurographics Symposium on Geometry Processing, SGP '06, pp. 61–70 (2006)
25. Kolluri, R., Shewchuk, J.R., O'Brien, J.F.: Spectral surface reconstruction from noisy point clouds. In: Eurographics Symposium on Geometry Processing, pp. 11–21 (2004)
26. Labatut, P., Pons, J.P., Keriven, R.: Robust and efficient surface reconstruction from range data. Computer Graphics Forum **28**(8), 2275–2290 (2009)
27. Lindstrom, P., Turk, G.: Evaluation of memoryless simplification. IEEE Transactions on Visualization and Computer Graphics **5**(2), 98–115 (1999)

28. Lipman, Y., Cohen-Or, D., Levin, D.: Data-dependent MLS for faithful surface approximation. In: Eurographics Symposium on Geometry Processing, p. 67 (2007)
29. Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H.: Parameterization free projection for geometry reconstruction. ACM Transactions on Graphics **26**(3), 22 (2007)
30. Lipman, Y., Daubechies, I.: Surface comparison with mass transportation (2010). ArXiv preprint 0912.3488
31. McAsey, M., Mou, L.: Optimal locations and the mass transport problem., pp. 131–148. Providence, RI: American Mathematical Society (1999)
32. Mérigot, Q.: A multiscale approach to optimal transport. Computer Graphics Forum **30**(5), 1583–1592 (2011)
33. Mérigot, Q., Ovsjanikov, M., Guibas, L.: Robust Voronoi-based curvature and feature estimation. In: 2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling, pp. 1–12 (2009)
34. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., Seidel, H.P.: Multi-level partition of unity implicits. In: ACM SIGGRAPH, vol. 22(3), pp. 463–470 (2003)
35. Oztireli, C., Guennebaud, G., Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. In: Computer Graphics Forum, vol. 28(2), pp. 493–501 (2009)
36. Pang, X.F., Pang, M.Y.: An algorithm for extracting geometric features from point cloud. International Conference on Information Management, Innovation Management and Industrial Engineering **4**, 78–83 (2009)
37. Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. Computer Graphics Forum **22**(3), 281–289 (2003)
38. Peyré, G., Fadili, J., Rabin, J.: Wasserstein active contours. Tech. rep., Preprint Hal-00593424 (2011). URL http://hal.archives-ouvertes.fr/hal-00593424/
39. Rabin, J., Delon, J., Gousseau, Y.: Regularization of transportation maps for color and contrast transfer. In: IEEE International Conference on Image Processing, pp. 1933 –1936 (2010)
40. Rabin, J., Delon, J., Gousseau, Y.: Transportation distances on the circle. J. Math. Imaging Vis. **41**(1-2), 147–167 (2011)
41. Rabin, J., Peyré, G., Cohen, L.D.: Geodesic shape retrieval via optimal mass transport. In: European Conference on Computer Vision: Part V, ECCV'10, pp. 771–784. Springer-Verlag, Berlin, Heidelberg (2010)
42. Rabin, J., Peyré, G., Delon, J., Bernot, M.: Wasserstein barycenter and its application to texture mixing. In: A. Bruckstein, B. ter Haar Romeny, A. Bronstein, M. Bronstein (eds.) Scale Space and Variational Methods in Computer Vision, *Lecture Notes in Computer Science*, vol. 6667, pp. 435–446. Springer Berlin / Heidelberg (2012)
43. Reinhard, E., Ashikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Comput. Graph. Appl. **21**(5), 34–41 (2001)
44. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vision **40**(2), 99–121 (2000)
45. Salman, N., Yvinec, M., Mérigot, Q.: Feature Preserving Mesh Generation from 3D Point Clouds. In: Computer Graphics Forum, vol. 29, pp. 1623–1632 (2010)
46. Villani, C.: Topics in Optimal Transportation. American Mathematical Society (2010)
47. Walder, C., Chapelle, O., Schölkopf, B.: Implicit surface modelling as an eigenvalue problem. In: Machine Learning ICML 2005, pp. 936–939 (2005)
48. Wu, J., Kobbelt, L.: Fast mesh decimation by multiple-choice techniques. In: Vision, Modeling, and Visualization, pp. 241–248 (2002)