Point cloud denoising using a generalized error metric

Qun-Ce Xu [a], Yong-Liang Yang [b], Bailin Deng [c,*]

[a] Tsinghua University, China
[b] University of Bath, UK
[c] Cardiff University, UK

ARTICLE INFO

Keywords:
Geometry processing
Optimization
Point cloud denoising

ABSTRACT

Effective removal of noises from raw point clouds while preserving geometric features is the key challenge for point cloud denoising. To address this problem, we propose a novel method that jointly optimizes the point positions and normals. To preserve geometric features, our formulation uses a generalized robust error metric to enforce piecewise smoothness of the normal vector field as well as consistency between point positions and normals. By varying the parameter of the error metric, we gradually increase its non-convexity to guide the optimization towards a desirable solution. By combining alternating minimization with a majorization-minimization strategy, we develop a numerical solver for the optimization which guarantees convergence. The effectiveness of our method is demonstrated by extensive comparisons with previous works.

1. Introduction

With the recent development of 3D sensing technologies, 3D acquisition has been made easy for novice users. Nowadays, large amounts of 3D point clouds can be obtained from not only high-end devices such as laser scanners, but also commodity devices such as mobile phones, RGBD cameras, and head-mounted displays. However, the captured data usually deviates from the ground truth due to several factors, including the precision of the 3D sensor, the physical material of the target object, the lighting condition, etc. As a result, removing noises from the raw point cloud is important for real-world applications that require high-quality geometry, such as reverse engineering, digital cultural heritage, and person identification, just to name a few.

To solve this problem, point cloud denoising has been actively studied in the past, and various methods have been proposed. A key requirement here is to remove the noises while preserving inherent geometric features of the target object such as sharp edges [1]. A popular approach is to first modify the point cloud normals to promote their piecewise smoothness and thereby preserve sharp features, using sparsity optimization [2,3], filtering [4], or low-rank optimization [5]; afterwards, the point positions are updated accordingly, often via an optimization that enforces the consistency between the modified normals and the point positions [2,3]. Since the normals are sensitive to the underlying surface shape, they can be an effective proxy for denoising the point cloud. Indeed, such a two-step approach is commonly used for denoising mesh surfaces [6–11], a problem closely related to point cloud denoising. On the other hand, such a modification of normals does not take their integrability into account, i.e., the normals are modified without considering the existence of shapes that are consistent with the new normals. As a consequence, the actual normals resulting from the updated point positions may deviate from the modified normals in the first step, which can lead to sub-optimal results.

To address this issue, we propose a new formulation that jointly optimizes the point positions and normals, with a target function that enforces both smoothness and integrability of the normals. To preserve sharp features, we apply a robust error metric to the smoothness and integrability measures to promote their sparsity. However, unlike [2,3] that optimize using a single robust metric, we adopt a generalized metric recently proposed in [12] that unifies a family of metrics with different levels and non-convexity. By gradually changing the metric parameter to increase its non-convexity and robustness, our optimization first improves the overall smoothness of the shape and then progressively enhances the sharp features, which helps to avoid undesirable local minima. To solve the optimization problem for each instance of the generalized metric, we develop a numerical solver using an alternating minimization approach together with a majorization-minimization strategy [13]. Our solver is guaranteed to monotonically decrease the target function and converge to a stationary point. We test our method on a variety of synthetic and scanned point clouds. Both quantitative and qualitative results demonstrate its effectiveness. To summarize, our contributions include:

- We propose a point cloud denoising approach that jointly optimizes the point positions and normals, enforcing both piecewise smoothness and integrability of the normals.

* Corresponding author.
E-mail addresses: quncexu@tsinghua.edu.cn (Q.-C. Xu), y.yang@cs.bath.ac.uk (Y.-L. Yang), DengB3@cardiff.ac.uk (B. Deng).

- We adopt a generalized error metric for our optimization, and gradually increase its non-convexity and robustness to guide the optimization towards a desirable solution.
- We develop a numerical solver for the proposed optimization, which guarantees a monotonic decrease of the target function as well as convergence to a stationary point.

2. Related work

Point cloud filtering has been actively studied in the last few years. A comprehensive review is beyond the scope of the present paper. We refer readers to recent surveys [1,14] for a detailed summary of the literature. Here we focus on reviewing the landmark works according to different filtering methodologies that are representative in the field.

2.1. Local projection based methods

Local projection based methods project points onto a smooth estimate of the underlying surface within a local neighborhood. Based on the framework of *Moving Least Squares* (MLS) [15,16], a number of filtering methods [17–20] are proposed to iteratively project points onto a locally fitted polynomial. The difference is in how to represent the MLS surface for efficient projection. As MLS is originally designed for smooth surface approximation, to better handle sharp features, several variation methods are presented based on different approaches such as cell complexes [21], algebraic sphere fitting [22], and robust regression [23,24]. Unlike MLS-based methods that usually require local normal and parameterized surface for point projection, Lipman et al. [25] present a *Locally Optimal Projection* operator (LOP) which enables parameterization-free filtering. The basic idea is to project points according to the local L_1 median of the original point cloud. Weighted LOP (WLOP) [26] involves locally adaptive density weights to generate a more evenly distributed point set. Anisotropic LOP (ALOP) [27] applies anisotropic projection with respect to point normal to better preserve sharp features. To improve the projection efficiency, kernel LOP (KLOP) [28] down-samples the point cloud using kernel density estimate thus reduces the computation cost of LOP, but the result quality is largely affected by the number of kernels. Continuous LOP (CLOP) [29] utilizes Gaussian mixture to continuously describe the input point density, allowing a high sampling rate using only a few Gaussian components.

2.2. Statistical based methods

Statistical based methods process point cloud from a statistical point of view. Various approaches in statistics have been adopted for filtering noisy points, such as mean shift [30], Bayesian statistics [31], iteratively re-weighted least squares [32], and moving robust Principal Component Analysis (PCA) [33]. More related to our work, recent techniques utilize sparse representation of point normals for robust point cloud filtering while preserving sharp features. Avron et al. [2] employ L_1-sparsity paradigm to remove noise by first applying a re-weighted L_1 optimization to estimate point normals, then restoring smooth point positions along the estimated normals. To better maintain sharp features, Sun et al. [3] utilizes a sparser L_0 solution which also restores point normals and positions alternatively. While this method successfully produces piecewise smooth point sets, edge recovery and point upsampling are further required to handle cross artifact and gap near sharp edges, especially when the point cloud contains a large amount of noise.

2.3. Learning based methods

Learning based methods leverage the recent advances in deep learning and train deep neural networks for point cloud filtering/consolidation. PU-Net [34] up-samples the input point cloud by aggregating and expanding features learned locally using the PointNet++ structure [35]. EC-Net [36] further preserves sharp features for point cloud consolidation based on manually annotated edge segments and specifically designed edge-aware losses. PointProNets [37] relies on 2D heightmaps to represent local 3D point patches. This allows the use of 2D CNNs for heightmap filtering, while back-projection is needed to obtain the resultant point cloud. Based on a local variant of PointNet [38], PointCleanNet [39] utilizes a two-stage network for outlier removal and point denoising respectively. The first stage classifies and rejects the outliers, and the second stage iteratively cleans points by predicting displacement vectors that denoise the remaining points. Pointfilter [40] employs an encoder–decoder structure to denoise point cloud patches. Lu et al. [41] estimate normals in a feature-preserving manner for point cloud denoising. More recently, gradient-based methods [42,43] formalize point cloud denoising as an iterative process of increasing the log-likelihood of each point via estimating the gradient of the underlying distribution. After that, recent advancements in deep-learning-based methods, such as SVCNet [44] and FCNet [45], have emerged, focusing on leveraging noise during the learning phase. SVCNet [44] introduces the concept of Self-Variation, aiming to learn potential commonalities by perturbing the noise on noisy points. An edge constraint module is also employed to mitigate the low-pass effects during the denoising process. On the other hand, FCNet [45] adopts a two-step network framework to address feature noise and learn noise-free features through feature domain losses. To achieve this, the FCNet proposes the utilization of non-local self-similarity and weighted average pooling modules, which effectively smooth features and suppress feature noise resulting from outliers. Unlike the above supervised methods that require access to the ground-truth clean data, 'Total Denoising' [46] performs in an unsupervised manner. It maps a point cloud to itself based on a spatial locality and a bilateral appearance prior, achieving competitive results against supervised methods at the time.

3. Our method

In this paper, we assume the input noisy point cloud consisting of n points, represented using their positions $\{\mathbf{p}_i^0 \in \mathbb{R}^3\}$ and outward normals $\{\mathbf{n}_i^0 \in \mathbb{R}^3\}$ $(i = 1, \ldots, n)$. The normals are either generated by the scanning device, or estimated from the positions using PCA followed by post-processing to choose consistent orientations of the normals (cf. [47,48] and references therein). Given the input point positions and normals, we would like to compute the positions $\{\mathbf{p}_i\}$ and normals $\{\mathbf{n}_i\}$ for the denoised point cloud. Many existing methods such as [2,3] adopt a two-step approach to solve the problem: first, the denoised normals are computed using a feature-preserving smoothing method; afterwards, the positions are updated using a consistency condition between positions and normals, e.g. by enforcing small distance between the new tangent plane at a point and its neighboring points while penalizing the change of point positions. Although such an approach is simple to implement, the normal denoising step does not take into account the coupling between positions and normals, which may not produce a desirable result. Indeed, similar issues exist for many mesh denoising methods that take such a two-step approach, where the normals computed in the first step may not correspond to a valid mesh and the vertex positions need to be updated in a least-squares manner [6,11]. In this paper, we propose an optimization-based approach that performs feature-preserving denoising on both positions and normals simultaneously. Our optimization enforces the piecewise smoothness of the resulting normal field, along with the coupled relation between the normals and point positions. In this way, the optimization ensures that the normals and point positions are smoothed in a consistent manner, helping to achieve desirable results. In the following, we present our formulation, and propose a numerical solver for the optimization problem.

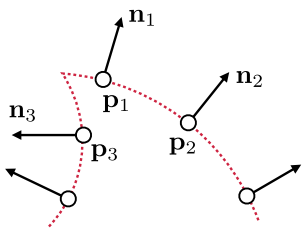


Fig. 1. Sample points on a surface (in red) close a sharp feature. Here neighboring points $\mathbf{p}_1$ and $\mathbf{p}_2$ belong to the same smooth region, with a small difference between their normals $\mathbf{n}_1, \mathbf{n}_2$, and the vector $\mathbf{p}_1 - \mathbf{p}_2$ is approximately orthogonal to both $\mathbf{n}_1$ and $\mathbf{n}_3$. Points $\mathbf{p}_1$ and $\mathbf{p}_3$ lie across the sharp feature, with a large difference between their normals $\mathbf{n}_1, \mathbf{n}_3$, and the vector $\mathbf{p}_1 - \mathbf{p}_3$ is not orthogonal to $\mathbf{n}_1$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Optimization formulation

Let $\mathbf{P}, \mathbf{N} \in \mathbb{R}^{3n}$ denote the concatenation of new positions and new normals, respectively. We compute $\mathbf{P}$ and $\mathbf{N}$ via the following optimization

$$\min_{\mathbf{P},\mathbf{N}} \; E_{\text{fid}} + w_{\text{u}} E_{\text{unit}} + w_{\text{d}} E_{\text{disp}} + w_{\text{r}} E_{\text{reg}}. \tag{1}$$

Here E_{fid} is a fidelity term that penalize changes in positions and normals:

$$E_{\text{fid}} = w_{\text{p}} \left\| \mathbf{P} - \mathbf{P}^0 \right\|^2 + w_{\text{n}} \left\| \mathbf{N} - \mathbf{N}^0 \right\|^2, \tag{2}$$

where $\mathbf{P}^0, \mathbf{N}^0 \in \mathbb{R}^{3n}$ are the initial positions and normals respectively, and $w_{\text{p}}, w_{\text{n}}$ are user-specified weights. E_{unit} penalizes the deviation of the new normals from unit vectors:

$$E_{\text{unit}} = \sum_{i=1}^{n} \left\| \mathbf{n}_i - \bar{\mathbf{n}}_i \right\|^2, \tag{3}$$

where $\bar{\mathbf{n}}_i = \mathbf{n}_i / \|\mathbf{n}_i\|$. The term E_{disp} requires the position change $\mathbf{x}_i - \mathbf{x}_i^0$ for each point to be as parallel to the normal $\mathbf{n}_i$ as possible:

$$E_{\text{disp}} = \sum_{i=1}^{n} \|\mathbf{n}_i \times (\mathbf{p}_i - \mathbf{p}_i^0)\|^2. \tag{4}$$

Finally, E_{reg} is a robust regularization term for the new normals that also considers their consistency with the new positions; it will be explained in detail later. $w_{\text{u}}, w_{\text{d}}, w_{\text{r}}$ are user-specified weights that control the trade-off between different terms.

To define E_{reg}, we assume that the underlying surface for the ground-truth point cloud is piece-wise smooth. Therefore, the difference between normals $\mathbf{n}_i, \mathbf{n}_j$ at two neighboring points should be small on most parts of the surface, while they can be large on some local regions corresponding to sharp features (see Fig. 1). Such distribution of normals can be induced using a target function term that promotes sparsity across the normal difference between all neighboring points. In addition, within a smooth region of the surface, the positions $\mathbf{p}_i, \mathbf{p}_j$ and normals $\mathbf{n}_i, \mathbf{n}_j$ of two neighboring points should satisfy a consistency condition that the vector $\mathbf{p}_i - \mathbf{p}_j$ is approximately orthogonal to both $\mathbf{n}_i$ and $\mathbf{n}_j$; across a sharp feature, on the other hand, this condition may not be satisfied as shown in Fig. 1. Therefore, the absolute values of inner products $(\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{n}_i$ and $(\mathbf{p}_i - \mathbf{p}_j) \cdot \mathbf{n}_j$ should be small for most neighboring points, while they can be large on some local regions around sharp features. This behavior can be similarly induced via a sparsity-promoting terms for all such inner products across the surface. However, simply adding two separate sparsity terms for normal differences and position-normal consistency may not lead to a desirable result. This is because the sparsity patterns for these two quantities are coupled: a pair neighboring points with a large difference between their normals, which indicates their proximity to a sharp feature, are also likely to violate the consistency condition. Simple addition of the two sparsity-promoting terms only induces sparsity for the two quantities individually rather than their coupled sparsity. For the latter purpose, we need to promote their *group sparsity* [49] instead. Specifically, we promote sparsity for the following quantity that measures normal difference and position-normal consistency simultaneously for a pair of neighboring points:

$$h_{ij} = \frac{1}{L_{ij}} \sqrt{\left(\frac{\mathbf{p}_i - \mathbf{p}_j}{L_{ij}} \cdot (\mathbf{n}_i + \mathbf{n}_j)\right)^2 + \gamma \|\mathbf{n}_i - \mathbf{n}_j\|^2}, \tag{5}$$

where L_{ij} is the initial distance between $\mathbf{p}_i$ and $\mathbf{p}_j$, and γ is a user-specified weight. Here the term $(\frac{\mathbf{p}_i - \mathbf{p}_j}{L_{ij}} \cdot (\mathbf{n}_i + \mathbf{n}_j))^2$ measures the consistency between the normals and positions of two adjacent points, and evaluates to zero when the positions and normals are consistent with a locally-second-order surface centered between the two points [50]; this helps to account for high-curvature regions where a line segment connecting two adjacent points may not be orthogonal to their normals [51]. The term $\|\mathbf{n}_i - \mathbf{n}_j\|^2$ penalizes the deviation between neighboring normals and induces smoothness of the normal field. The scaling factor $\frac{1}{L_{ij}}$ acts as normalization to account for the uneven spacing between the points. The quantity h_{ij} should be small on most parts of the surface except for regions around sharp features. The regularization term E_{reg} is then defined as

$$E_{\text{reg}} = \sum_{\substack{i<j \\ (i,j)\in\mathcal{N}}} \phi(h_{ij}), \tag{6}$$

where $\mathcal{N}$ denotes the index set of neighboring points, and ϕ is a robust error metric. To determine the neighboring relation between points, we perform k-nearest neighbor search for each point. Two points i, j are considered to be neighbors if point i is among the k-nearest neighbors of point j or point j is among the k-nearest neighbors of point i.

In Eq. (6), the choice of function ϕ plays an important role in the performance of our method. One possible choice is to define E_{reg} as the ℓ_0-norm of all h_{ij}, in which case

$$\phi(x) = \begin{cases} 0, & \text{if } x = 0, \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

However, this will lead to a highly non-convex and non-smooth optimization problem that is challenging to solve, in particular since h_{ij} is a non-linear function of $\mathbf{V}$ and $\mathbf{N}$. Inspired by the recent work of [52], we choose ϕ as a member of a parameterized family of error metric functions instead:

$$\phi_{\alpha,c}(x) = \frac{|\alpha - 2|}{\alpha} \left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right). \tag{8}$$

Here c is a scale parameter for normalization, while $\alpha \in \mathbb{R}$ is a shape parameter that controls the robustness of the error metric. For a given c, as α is decreased $\phi_{\alpha,c}$ becomes increasingly robust and non-convex, and reproduces different robust loss functions used in the literature. For example, when α approaches 2, $\phi_{\alpha,c}$ approaches the ℓ_2 loss:

$$\lim_{\alpha \to 2} \phi_{\alpha,c}(x) = \frac{1}{2} \left(\frac{x}{c} \right)^2. \tag{9}$$

When $\alpha = 1$, it becomes a smoothed ℓ_1 loss also known as the Charbonnier loss [53]:

$$\phi_{1,c}(x) = \sqrt{\left(\frac{x}{c} \right)^2 + 1} - 1. \tag{10}$$

when α approaches 0, it approaches the Cauchy loss [54]:

$$\lim_{\alpha \to 0} \phi_{\alpha,c}(x) = \log \left(\frac{1}{2} \left(\frac{x}{c} \right)^2 + 1 \right). \tag{11}$$

When $\alpha = -2$, it becomes the Geman-McClure loss [55]:

$$\phi_{-2,c}(x) = \frac{2\,(x/c)^2}{(x/c)^2 + 4}. \tag{12}$$

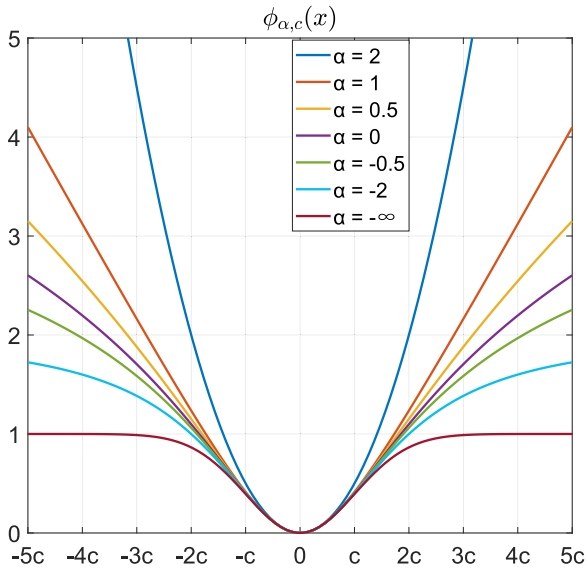


Fig. 2. The generalized metric functions with different values of the parameter α.

When $\alpha \to -\infty$, it approaches the Welsch function [56]:

$$\lim_{\alpha\to-\infty} \phi_{\alpha,c}(x) = 1 - \exp\left(-\frac{1}{2}\left(\frac{x}{c}\right)^2\right), \tag{13}$$

This metric has been used in [11] for robust mesh filtering since it is bounded from above and it approaches the ℓ_0-norm when $c \to 0$. Following the convention from [52], we adopt the following definition of $\phi_{\alpha,c}$ as illustrated in Fig. 2:

$$\phi_{\alpha,c}(x) = \begin{cases} \frac{1}{2}\left(\frac{x}{c}\right)^2 & \text{if } \alpha = 2, \\ \log\left(\frac{1}{2}(\frac{x}{c})^2 + 1\right) & \text{if } \alpha = 0, \\ 1 - \exp\left(-\frac{1}{2}(\frac{x}{c})^2\right) & \text{if } \alpha = -\infty, \\ \frac{|\alpha-2|}{\alpha}\left(\left(\frac{(x/c)^2}{|\alpha-2|}+1\right)^{\alpha/2} - 1\right) & \text{otherwise.} \end{cases} \tag{14}$$

We solve the optimization problem (1) with α gradually decreased from 2 to $-\infty$. As discussed in Section 3.2.4, our choice of ϕ enables us to derive an simple solver, and the strategy of decreasing α helps the solver to find a desirable local minimum and improves the denoising accuracy compared to using a fixed α.

3.2. Numerical solver

The optimization (1) is a non-convex problem for the positions $\mathbf{P}$ and the normals $\mathbf{N}$, and the term E_{reg} can be highly non-linear. To effectively solve this problem, we adopt an alternating minimization strategy. We first fix $\mathbf{P}$ and optimize $\mathbf{N}$, then fix $\mathbf{N}$ and optimize $\mathbf{P}$. This process is repeated until convergence. In the following, we present the details of each sub-problem.

3.2.1. The N-update

For the ease of presentation, we assume that the variable values before the $\mathbf{N}$-update are $\mathbf{P}^{(k)}$ and $\mathbf{N}^{(k)}$, with the individual positions and normals denoted by $\{\mathbf{p}_i^{(k)}\}$ and $\{\mathbf{n}_i^{(k)}\}$, respectively. We update $\mathbf{N}$ by fixing $\mathbf{P} = \mathbf{P}^{(k)}$ and minimizing the target function of (1) over $\mathbf{N}$, resulting in the following sub-problem:

$$\begin{aligned} \min_{\mathbf{N}}\ & w_{\text{n}} \left\|\mathbf{N} - \mathbf{N}^0\right\|^2 + w_{\text{d}} \sum_{i=1}^{n} \left\|\mathbf{n}_i \times (\mathbf{p}_i^{(k)} - \mathbf{p}_i^0)\right\|^2 \\ & + w_{\text{u}} \sum_{i=1}^{n} \left\|\mathbf{n}_i - \overline{\mathbf{n}}_i\right\|^2 + w_{\text{r}} E_{\text{reg}}(\mathbf{P}^{(k)}, \mathbf{N}). \end{aligned} \tag{15}$$

The first two terms are convex quadratic, but the remaining terms are still highly non-convex. To solve this problem, we adopt the idea of the majorization-minimization (MM) algorithm [13]: to minimize a target function $F(x)$, it repeatedly constructs a surrogate function $\overline{F}(x; x^{(k)})$ based on the current variable value $x^{(k)}$ which satisfies the conditions:

$$\begin{aligned} \overline{F}(x^{(k)}; x^{(k)}) &= F(x^{(k)}), \\ \overline{F}(x; x^{(k)}) &\geq F(x) \quad \forall x. \end{aligned} \tag{16}$$

In other words, the surrogate function bounds the target function from above, and their graphs touch at the current variable value $x^{(k)}$. The surrogate function is then minimized to update the variable, and the above process is repeated until convergence. To apply this idea, we use the current variable value $\mathbf{N}^{(k)}$ to construct a convex quadratic surrogate function $\overline{Q}(\mathbf{N}; \mathbf{N}^{(k)})$ for the target function in Eq. (15), and update the value of $\mathbf{N}$ via

$$\mathbf{N}^{(k+1)} = \arg\min_{\mathbf{N}}\ \overline{Q}(\mathbf{N}; \mathbf{N}^{(k)}). \tag{17}$$

It can be shown that the following function meets the surrogate function condition in Eq. (16) (see Appendix A for verification):

$$\begin{aligned} \overline{Q}(\mathbf{N}; \mathbf{N}^{(k)}) = {} & w_{\text{n}} \left\|\mathbf{N} - \mathbf{N}^0\right\|^2 + w_{\text{u}} \sum_{i=1}^{n} \left\|\mathbf{n}_i - \overline{\mathbf{n}}_i^{(k)}\right\|^2 \\ & + w_{\text{d}} \sum_{i=1}^{n} \left\|\mathbf{n}_i \times (\mathbf{p}_i^{(k)} - \mathbf{p}_i^0)\right\|^2 \\ & + w_{\text{r}} \sum_{\substack{i<j \\ (i,j)\in\mathcal{N}}} \left(A_{ij}^{(k)} \cdot \left(h_{ij}(\mathbf{P}^{(k)}, \mathbf{N})\right)^2 + B_{ij}^{(k)}\right), \end{aligned} \tag{18}$$

where $\overline{\mathbf{n}}_i^{(k)} = \mathbf{n}_i^{(k)} / \|\mathbf{n}_i^{(k)}\|$, $A_{ij}^{(k)} = \Psi(h_{ij}(\mathbf{P}^{(k)}, \mathbf{N}^{(k)}))$, and $B_{ij}^{(k)} = \Theta(h_{ij}(\mathbf{P}^{(k)}, \mathbf{N}^{(k)}))$ with

$$\Psi(t) = \begin{cases} \frac{1}{2c^2} & \text{if } \alpha = 2, \\ \frac{1}{2c^2}\exp\left(-\frac{t^2}{2c^2}\right) & \text{if } \alpha = -\infty, \\ \frac{1}{2c^2}\left(\frac{(t/c)^2}{|\alpha-2|}+1\right)^{\frac{\alpha-2}{2}} & \text{otherwise.} \end{cases} \tag{19}$$

$$\Theta(t) = \begin{cases} 0 & \text{if } \alpha = 2, \\ 1 - \frac{2c^2+t^2}{2c^2}\exp\left(-\frac{t^2}{2c^2}\right) & \text{if } \alpha = -\infty, \\ \phi_{\alpha,c}(t) - \frac{t^2}{2c^2}\left(\frac{(t/c)^2}{|\alpha-2|}+1\right)^{\frac{\alpha-2}{2}} & \text{otherwise.} \end{cases} \tag{20}$$

Since $\overline{Q}(\mathbf{N}; \mathbf{N}^{(k)})$ is convex quadratic in $\mathbf{N}$, its global minimum can be computed by solving a linear system:

$$\begin{aligned} & \left((w_{\text{n}} + w_{\text{u}})\mathbf{I} + w_{\text{d}}\mathbf{J}^T\mathbf{J} + w_{\text{r}}(\mathbf{K}^T\mathbf{K} + \gamma\mathbf{D}^T\mathbf{D})\right)\mathbf{N} \\ & = w_{\text{n}}\mathbf{N}^0 + w_{\text{u}}\overline{\mathbf{N}}^{(k)}. \end{aligned} \tag{21}$$

Here $\mathbf{I} \in \mathbb{R}^{3n\times 3n}$ is an identity matrix. $\overline{\mathbf{N}}^{(k)} \in \mathbb{R}^{3n}$ concatenates $\{\overline{\mathbf{n}}_i^{(k)}\}$. $\mathbf{J}^{(k)} \in \mathbb{R}^{3n\times 3n}$ is a block diagonal matrix:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1^{(k)} & & \\ & \ddots & \\ & & \mathbf{J}_n^{(k)} \end{bmatrix} \tag{22}$$

with each block $\mathbf{J}_i^{(k)}$ representing the cross product with $\mathbf{p}_i^{(k)} - \mathbf{p}_i^0$ so that $\mathbf{J}_i^{(k)}\mathbf{n} = \mathbf{n} \times (\mathbf{p}_i^{(k)} - \mathbf{p}_i^0)$ for any $\mathbf{n} \in \mathbb{R}^3$. $\mathbf{K}$ is a sparse matrix, where each row corresponds to a neighboring normal pair $(\mathbf{n}_i, \mathbf{n}_j)$ and has non-zero entries $\frac{\sqrt{A_{ij}^{(k)}}}{L_{ij}^2}(\mathbf{p}_i^{(k)} - \mathbf{p}_j^{(k)})^T$ for both $\mathbf{n}_i$ and $\mathbf{n}_j$. $\mathbf{D}$ is a block sparse matrix, where each block row corresponds to a neighboring normal pair $(\mathbf{n}_i, \mathbf{n}_j)$ and contains non-zero blocks $\frac{\sqrt{A_{ij}^{(k)}}}{L_{ij}}\mathbf{I}_3$ and $-\frac{\sqrt{A_{ij}^{(k)}}}{L_{ij}}\mathbf{I}_3$ for $\mathbf{n}_i$ and $\mathbf{n}_j$ respectively, with $\mathbf{I}_3$ being the 3×3 identity matrix. The linear system (21) is sparse symmetric positive definite, and we solve it via Cholesky factorization. Moreover, since the matrix has a fixed sparsity

pattern, we perform its symbolic factorization as a pre-processing step, and only perform numerical factorization in each iteration.

Note that the single update step in Eq. (17) may not produce the solution to the sub-problem (15). However, as discussed later in Section 3.2.3, it is guaranteed to decrease the target function and is sufficient for the convergence of our iterative solver.

3.2.2. The **P**-update

After updating **N** according to Eq. (17), we fix $\mathbf{N} = \mathbf{N}^{(k+1)}$ and minimize the target function of (1) over **N**, resulting in the following sub-problem:

$$\min_{\mathbf{P}} \; w_{\mathrm{p}}\|\mathbf{P}-\mathbf{P}^0\|^2 + w_{\mathrm{d}}\sum_{i=1}^{n}\|\mathbf{n}_i^{(k+1)}\times(\mathbf{p}_i-\mathbf{p}_i^0)\|^2 + w_{\mathrm{r}}E_{\mathrm{reg}}(\mathbf{P},\mathbf{N}^{(k+1)}). \tag{23}$$

Similar to the update of **N**, we construct a surrogate function $\overline{R}(\mathbf{P};\mathbf{P}^{(k)})$ for the target function based on the current variable value $\mathbf{P}^{(k)}$, and minimize it to update **P**:

$$\mathbf{P}^{(k+1)} = \arg\min_{\mathbf{P}} \; \overline{R}(\mathbf{P};\mathbf{P}^{(k)}). \tag{24}$$

It can be shown that the following convex quadratic function meets the requirement for the surrogate function (see Appendix A):

$$\overline{R}(\mathbf{P};\mathbf{P}^{(k)}) = w_{\mathrm{p}}\left\|\mathbf{P}-\mathbf{P}^0\right\|^2 + w_{\mathrm{d}}\sum_{i=1}^{n}\left\|\mathbf{n}_i^{(k+1)}\times(\mathbf{p}_i-\mathbf{p}_i^0)\right\|^2 + w_{\mathrm{r}}\sum_{\substack{i<j\\(i,j)\in\mathcal{N}}}\left(C_{ij}^{(k)}\cdot\left(h_{ij}(\mathbf{P}^{(k)},\mathbf{N})\right)^2 + D_{ij}^{(k)}\right). \tag{25}$$

where $C_{ij}^{(k)} = \Psi(h_{ij}(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}))$ and $D_{ij}^{(k)} = \Theta(h_{ij}(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}))$, with Ψ and Θ defined in Eqs. (19) and (20), respectively. Its global minimum can be computed by solving a sparse symmetric positive definite linear system:

$$\left(w_{\mathrm{p}}\mathbf{I} + w_{\mathrm{d}}\mathbf{L}^T\mathbf{L} + w_{\mathrm{r}}\mathbf{M}^T\mathbf{M}\right)\mathbf{P} = \left(w_{\mathrm{p}}\mathbf{I} + w_{\mathrm{d}}\mathbf{L}^T\mathbf{L}\right)\mathbf{P}^0. \tag{26}$$

Here $\mathbf{L}\in\mathbb{R}^{3n\times 3n}$ is a block diagonal matrix:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1^{(k+1)} & & \\ & \ddots & \\ & & \mathbf{L}_n^{(k+1)} \end{bmatrix} \tag{27}$$

with each block $\mathbf{L}_i^{(k+1)}$ representing the cross product with $\mathbf{n}_i^{(k+1)}$ so that $\mathbf{L}_i^{(k+1)}\mathbf{p} = \mathbf{n}_i^{(k+1)}\times\mathbf{p}$, $\forall\mathbf{p}\in\mathbb{R}^3$. **M** is a sparse matrix where each row corresponds to a neighboring point pair $(\mathbf{p}_i,\mathbf{p}_j)$ and contains non-zero entries $\frac{\sqrt{C_{ij}^{(k)}}}{L_{ij}^2}(\mathbf{n}_i^{(k+1)}+\mathbf{n}_j^{(k+1)})^T$ and $-\frac{\sqrt{C_{ij}^{(k)}}}{L_{ij}^2}(\mathbf{n}_i^{(k+1)}+\mathbf{n}_j^{(k+1)})^T$ for $\mathbf{p}_i$ and $\mathbf{p}_j$ respectively. Similar to the update of **N**, we solve the system using Cholesky factorization, performing symbolic factorization in a pre-processing step and numerical factorization in each iteration.

3.2.3. Convergence

Our solver alternates between the **N**-update step (17) and the **P**-update step (24). It can be shown that such an iteration is guaranteed to decrease the target function of (1) unless it converges to a stationary point:

Proposition 3.1. *Let $E(\mathbf{P},\mathbf{N})$ denote the target function of the optimization problem (1). Then using the updates in Eqs. (17) and (24), we have*

$$E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) \le E(\mathbf{P}^{(k)},\mathbf{N}^{(k)}). \tag{28}$$

Moreover, $E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k)})$ if and only if $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)}$ and $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$, which is equivalent to the condition that $\nabla E(\mathbf{P}^{(k)},\mathbf{N}^{(k)}) = \mathbf{0}$.

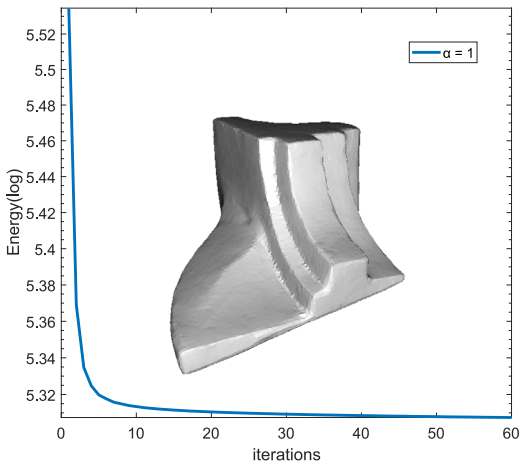


Fig. 3. For any given parameter α, our solver monotonically decrease the target function, as shown here for the 'Fandisk' model with $\alpha = 1$.

A proof is given in Appendix B. Therefore, we perform the update steps (17) and (24) until $\max_i\|\mathbf{n}_i^{(k+1)}-\mathbf{n}_i^{(k)}\| < \epsilon_1$ and $\max_i\|\mathbf{p}_i^{(k+1)}-\mathbf{p}_i^{(k)}\| < \epsilon_2\cdot L$, where ϵ_1,ϵ_2 are user-specified thresholds, and L is the average distance between neighboring points in the initial point cloud. We choose $\epsilon_1 = 1\times10^{-3}$ and $\epsilon_2 = 1\times10^{-4}$ in all experiments. An example of convergence is shown in Fig. 3.

3.2.4. Gradual decrease of α

From the function graphs in Fig. 2. we can see that as α is decreased, the error metric function becomes more non-convex and closer to the ℓ_0-norm, and regularizer terms with large values of h_{ij} (such as those arising from sharp features) will have less influence on the target function and can be better tolerated by the optimization. Ideally, we would like to perform the optimization (1) with $\alpha = -\infty$ in order to maximize its capacity of inducing sparsity and accommodating sharp features. However, optimizing with $\alpha = -\infty$ from the beginning can lead to sub-optimal results. This is because with $\alpha = -\infty$, a regularization term with a large error value of h_{ij} may have a very small weight ($\mathbf{A}_{ij}^k$ in Eq. (18) or $C_{ij}^{(k)}$ in Eq. (25)) according to the formula in (19). As a result, these terms are effectively discarded in the surrogate function, regardless of whether their large error value is caused by the noise or an actual underlying sharp feature. To make the optimization more robust, we would like to incorporate more effective terms into the surrogate function initially to perform a coarse optimization, and only disregard regularization terms with large errors in a later stage when they emerge from the coarse optimization to indicate sharp features. We achieve this by gradually decreasing α to $-\infty$, since the weight formula (19) shows that a larger α will result in less attenuation of weights for regularization terms with large errors. Concretely, we prescribe a decreasing sequence of α values $\alpha_1 > \alpha_2 > \cdots > \alpha_m$ with $\alpha_m = -\infty$. We start with $\alpha = \alpha_1$, and perform the optimization (1). Then we use the result as the initial value to re-run the optimization with $\alpha = \alpha_2$. This process is repeated until we complete the optimization with $\alpha = \alpha_m$, and the final solution is taken as the denoising result. In all our experiments, we use the sequence $(2, 1.0, 0.5, 0, -1.0, -2.0, -8.0, -\infty)$ for α. Before running an optimization with a new α, we also update the value of c according to the distribution of h_{ij} with the current result. Specifically, we first update the k-nearest neighbor relationship based on the latest point positions, and compute all values of h_{ij} accordingly. Then we choose c as:

$$c = \max\left(\overline{h}, c_{\min}\right), \tag{29}$$

where $\overline{h}$ is the median of all h_{ij} values, and $c_{\min}$ is a lower bound that prevents c from being too small. Noting that the scale of h_{ij} in Eq. (5)

Algorithm 1: Robust point cloud denoising.

Input: $\mathbf{P}^0, \mathbf{N}^0$: initial positions and normals;
L: average distance between neighboring points in the initial point cloud;
ϵ_1, ϵ_2: variable convergence thresholds;
$k_{\max}$: the maximum number of iterations;
$(\alpha_1, \ldots, \alpha_m)$: a sequence of decreasing α values.

$\mathbf{P}_{\text{latest}} = \mathbf{P}^0$; $\mathbf{N}_{\text{latest}} = \mathbf{N}^0$;
for $j = 1, 2, \ldots, m$ **do**
  $\alpha = \alpha_j$; $\mathbf{P}^{(1)} = \mathbf{P}_{\text{latest}}$; $\mathbf{N}^{(1)} = \mathbf{N}_{\text{latest}}$;
  Compute k-nearest neighbors for each point using positions $\mathbf{P}_{\text{latest}}$;
  Compute c according to Eq. (29);
  Perform symbolic factorization for the linear systems (21) and (26);
  $k = 1$;
  while ***TRUE*** **do**
    Compute $\mathbf{N}^{(k+1)}$ according to Eq. (21);
    Compute $\mathbf{P}^{(k+1)}$ according to Eq. (26);
    $e_1 = \max_i \|\mathbf{n}_i^{(k+1)} - \mathbf{n}_i^{(k)}\|$;
    $e_2 = \max_i \|\mathbf{p}_i^{(k+1)} - \mathbf{p}_i^{(k)}\|$;
    $k = k + 1$;
    if $k > k_{\max}$ ***OR*** $(e_1 < \epsilon_1$ ***AND*** $e_2 < \epsilon_2 \cdot L)$ **then**
      $\mathbf{P}_{\text{latest}} = \mathbf{P}^{(k)}$; $\mathbf{N}_{\text{latest}} = \mathbf{N}^{(k)}$;
      break;
    end if
  end while
end for
return $\mathbf{P}_{\text{latest}}, \mathbf{N}_{\text{latest}}$;

is roughly $\sqrt{1+\gamma}\,\|\mathbf{n}_i - \mathbf{n}_j\|/L_{ij}$ and the quantity $\|\mathbf{n}_i - \mathbf{n}_j\|/L_{ij}$ can be considered as a local curvature measure, we choose $c_{\min} = \sqrt{1+\gamma}/r_b$ where r_b is the bounding sphere radius for the initial point cloud. Algorithm 1 summarizes our method for denoising with decreasing values of α.

4. Results

In this section, we extensively evaluate our method by comparing it with existing methods on publicly available dataset for point cloud denoising, as well as validating our design choice.

4.1. Implementation details

Our optimization framework is implemented in C++, using the Eigen library [57] for linear algebra operations and OpenMP for parallelization. All experiments are run on a PC with a 3.7 GHz Intel Core i7-8700K CPU (6 Cores) and 16 GB memory. The parameter setting is shown in Table 1. While the default parameters are generally sufficient, empirically fine-tuning the parameters would help to achieve further improvements. The computational time is affected by both the number of points in the point cloud and the neighborhood size. For models with 11∼50K points, our method takes 3∼10 min, which is comparable to previous statistical methods [2,3]. The statistics of running time and iteration numbers on different datasets are shown in Table 2.

4.2. Test data

We evaluate our method based on the dataset used in PU-Net [34], similar to the majority of existing papers. We gather 20 meshes and apply Poisson disk sampling to generate point clouds. All shapes are normalized to fit within a unit sphere. After sampling, we introduce noise to the point positions by adding Gaussian noises with standard deviations of 1.0%, 2.0%, and 3.0% relative to the bounding box diagonal, respectively. To test the performance under different sampling densities, we sample each mesh using 10 K and 50 K points, respectively. We also test our method on point clouds sampled from the CAD models in the ABC dataset [58], as well as the real-world point clouds from the Kinect v1 dataset from [10].

Table 1
Parameter settings.

Parameter	Range(default)
neighbor size	7∼21(16)
weight w_{p}	1∼10(10)
weight w_{n}	1∼10(10)
weight w_{u}	1(1)
weight w_{d}	10∼100(10)
weight w_{r}	10∼100(10)
balance weight γ	0.1∼1.0(1.0)

4.3. Evaluation metrics

For quantitative evaluation, we use two metrics to measure the quality of the denoising results. First, to measure the deviation between the denoised point cloud and the ground-truth shape, we compute the average distance from each denoised point to its closest point on the ground-truth mesh surface (the P2M distance). In addition, we use the Chamfer Distance (CD) to measure the deviation between the denoised point cloud and the original (noise-free) sample points. To make the metric values more intuitive, we normalize them with the bounding box diagonal length of the model. In the following, we also use color coding to visualize the distance between the denoised point cloud and the ground-truth mesh surface.

4.4. Comparison

We compare our method with existing procedural methods based on edge-aware point set resampling (EAR) [27] and robust non-linear kernel regression of moving least squares (RIMLS) [24], as well as recent learning based approach called PointCleanNet (PCN) [39], Score-based Methods [42], DF-PCF [41], PointFilter [40] and PSR [43]. Note that EAR and RIMLS require suitable parameters related to scale, neighbor size, etc. to adapt to different noise levels, while PCN handles different noise levels using the same hyper-parameters trained with varying noisy data. In our experiment, we tune the parameters for EAR and RIMLS for each model, in order to achieve their best results for a fair comparison.

Table 3 shows the quantitative evaluation on the synthetic dataset, by computing the average CD and P2M values over all models for each configuration of sampling density and noise level. We can see that our method based on adaptive robust error metric generally achieves better performance. Fig. 4 further shows the color-coding visualization of denoising errors on some models. It can be seen that our method not only removes noises from the point clouds, but also achieves better preservation of details and sharp features than other methods. On the other hand, due to the resampling strategy, EAR [27] causes relatively large errors around detailed features while it performs well near prominent edges (see the 1st and 3rd row of Fig. 4). RIMLS [24] works well on cases with low-level noises, especially on simple planar and spherical geometries. However, it starts to blur detailed features when the noise level increases. PCN [39] effectively reduces the noise level but struggles to generalize well to point clouds with varying densities that were not part of the training data. Note that PCN requires outlier removal as a preparatory step; otherwise, the denoising may fail in some cases. Meanwhile, although DF-PCF [41] demonstrates the ability to preserve certain features, the model does not generalize well in some test cases, resulting in poorer denoising performance. Furthermore, the time required for processing is deemed unacceptable,

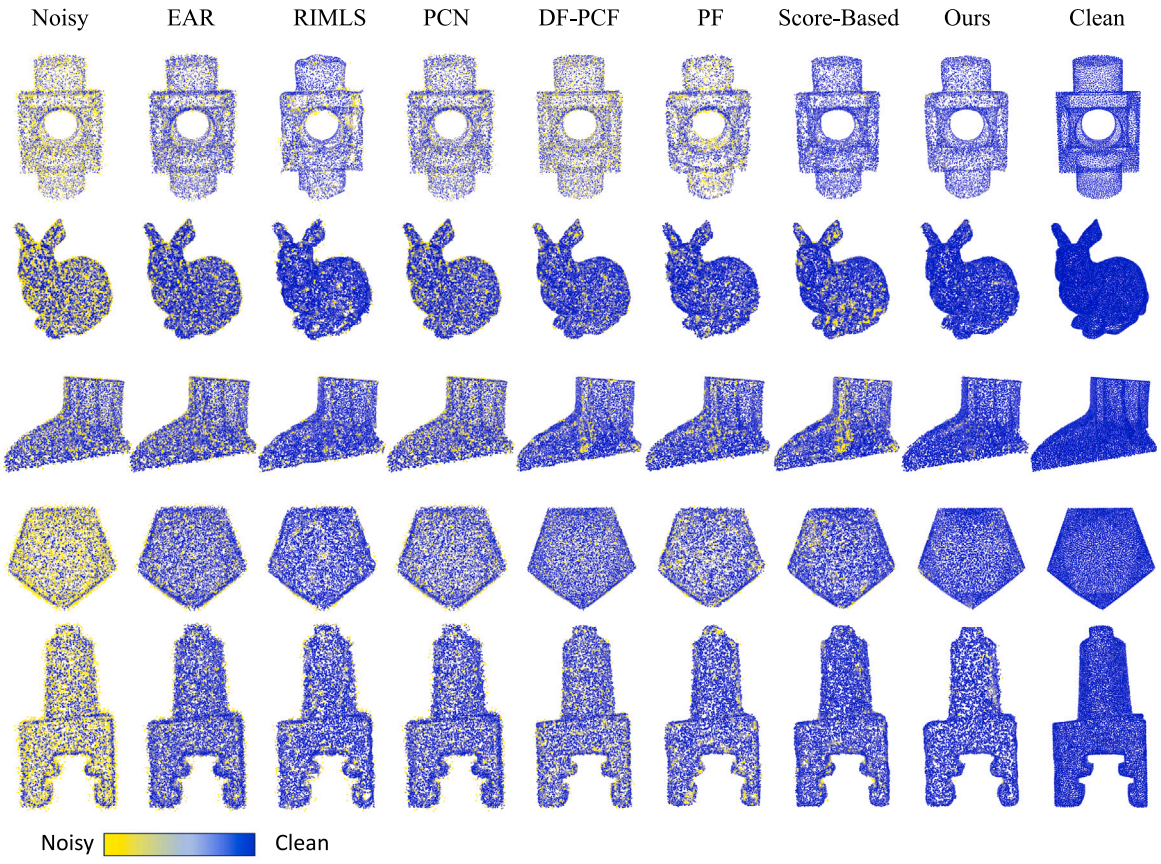


Fig. 4. Qualitative comparison of a variety of point clouds with different number of points and noise levels. Different colors in the color map illustrate distances to the ground truth. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Statistics of running time and iteration count for different datasets according to point numbers.

	Point number	Running Time(s)			Iteration number		
		min	max	average	min	max	average
PU-Net [34]	1 0K	47.5	484.3	176.4	94	697	269
	50 K	42.1	623.4	321.9	113	693	274
ABC [58]	10 K	44.6	352.2	162.8	70	495	204
	50 K	53.4	590.1	175.4	69	489	290
Kinect v1 [10]	13 K ~ 25 K	33.6	440.1	201.2	197	548	221

Table 3
Quantitative evaluation based on the Chamfer Distance (CD) between the denoised and ground-truth point cloud, and the average distance from the denoised points to the ground-truth mesh faces (P2M).

# Points	10 K (Sparse)						50 K (Dense)					
Noise	1%		2%		3%		1%		2%		3%	
Metric (10^{-4})	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
EAR [27]	4.311	1.629	6.010	3.470	8.821	4.673	2.876	1.878	3.740	2.198	6.549	4.672
RIMLS [24]	6.891	3.702	9.172	5.680	14.553	9.724	3.509	1.899	4.608	2.390	6.722	3.936
PCN [39]	3.422	1.129	7.562	4.312	13.077	9.505	1.142	0.353	1.559	0.712	3.498	1.371
Score-based [42]	3.409	0.512	5.890	1.391	6.895	1.946	0.746	0.161	1.244	0.534	2.602	1.021
DF-PCF [41]	3.609	1.251	4.311	1.322	6.425	4.567	1.263	0.186	1.381	0.273	1,844	0.939
Pointfilter [40]	4.129	1.333	6.812	4.214	11.874	7.649	1.375	0.267	1.465	0.671	3.697	1.783
PSR [43][a]	2.353	0.306	3.350	0.734	**4.075**	**1.242**	0.649	0.076	0.997	0.296	1.344	0.531
Ours	**2.034**	**0.252**	**2.97**	**0.692**	4.339	1.566	**0.535**	**0.061**	**0.723**	**0.269**	**1.145**	**0.442**

[a] The results are taken from the original paper as there is no released model available.

where a model consisting of only 10 K points can take over 30 min to complete even on an Nvidia RTX 4090, as indicated in the code released by the authors. Pointfilter [40] is not robust to high-level noise, as indicated by the results. While score-based methods [42] and PSR [43] achieve good performance overall, their design does not consider feature preservation and the results are sub-optimal in this regard. In summary, the aforementioned learning-based methods have their limitations, stemming from issues related to model generalization, variations in point cloud density, and challenges associated with noise levels. In contrast, our methods demonstrate improved robustness and

Table 4
Evaluation on a subset of the ABC dataset [58] based on the Chamfer Distance (CD) between the denoised and ground-truth point cloud, and the average distance from the denoised points to the ground-truth mesh faces (P2M).

# Points	10 K (Sparse)						50 K (Dense)					
Noise	1%		2%		3%		1%		2%		3%	
Metric (10^{-4})	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
Ours	2.232	0.339	3.24	0.782	4.529	1.803	0.825	0.076	1.011	0.317	1.323	0.541

Table 5
Ablation study that disables different terms in the target function. The evaluation is done on the PU-Net dataset [34] using the Chamfer Distance (CD) between the denoised and ground-truth point cloud, and the average distance from the denoised points to the ground-truth mesh faces (P2M).

# Points	10 K (Sparse)						50 K (Dense)					
Noise	1%		2%		3%		1%		2%		3%	
Metric (10^{-4})	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
Ours(full)	**2.034**	**0.252**	**2.97**	**0.692**	**4.339**	**1.566**	**0.535**	**0.061**	**0.723**	**0.269**	**1.145**	**0.442**
w/o E_{disp}	4.340	2.157	6.982	4.664	10.184	7.478	1.493	0.589	2.414	1.648	4.138	2.574
w/o E_{unit}	3.606	0.710	3.460	1.121	5.891	2.511	0.764	0.122	1.213	0.402	1.497	0.573
w/o E_{reg}	7.238	2.201	9.243	5.439	13.692	10.241	1.827	0.614	4.901	2.725	7.367	2.772

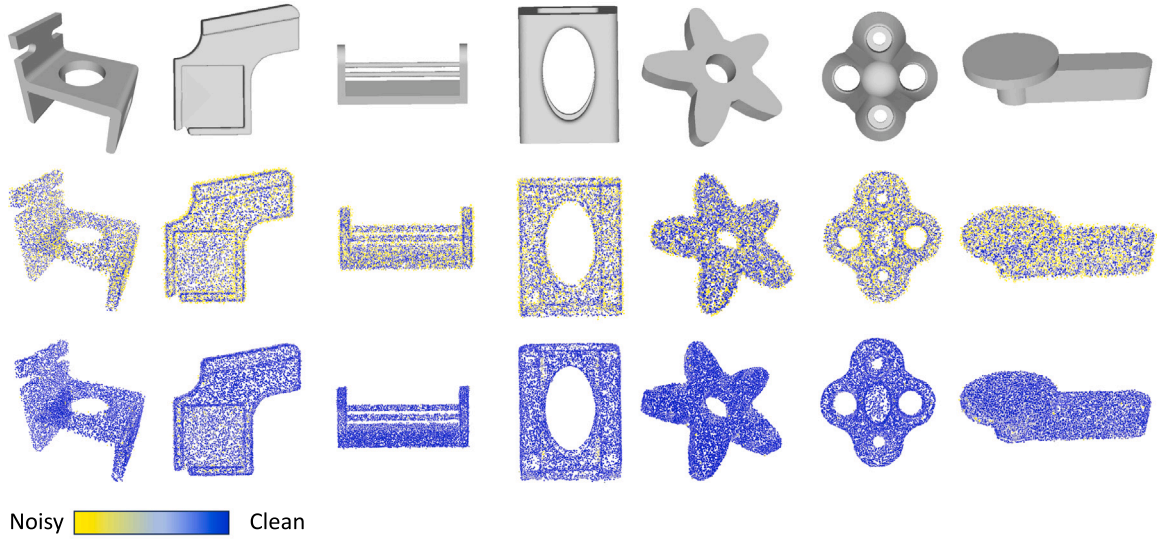


Fig. 5. Experimental results on a subset of the ABC Dataset [58]. The first row displays the ground-truth mesh, while the second row shows the noisy point clouds with varying point numbers and noise levels. The denoised results obtained using our method are presented in the last row. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

overall performance while endeavoring to preserve the input model's sharp features.

We also evaluate the performance of our method on a subset of the ABC dataset [58] consisting of 1000 CAD models. The visualization of the results is shown in Fig. 5, and the corresponding quantitative results are provided in Table 4. The results show that our method is effective on point clouds from CAD models.

4.5. Additional results

One of the main strengths of our generalized error metric is its capability to preserve sharp features. To highlight our performance here, we compare different methods on two representative models with prominent sharp features. The input point cloud is constructed by sampling the ground-truth mesh and adding Gaussian noise. After denoising the point cloud, we reconstruct a mesh from the result using the ball pivoting method [59], in order to visualize the features after denoising. The results are shown in Fig. 6. It is easy to see that our method can remove noises on the surface while preserving sharp features, producing results with the best visual quality.

Furthermore, we test our method on the real-world point clouds in the Kinect v1 dataset from [10]. The point clouds in the dataset are derived from depth images produced by a Kinect v1 camera, and are non-uniform on the object surface. Fig. 7 shows the denoising results. It shows that our method remains effective on non-uniformly sampled points.

4.6. Ablation study

To validate the effectiveness of the components of our optimization target function, we conduct an ablation study to assess the influence of different terms. Specifically, we disable E_{disp}, E_{unit} and E_{reg} respectively by setting their weights to zero, and run the optimization on the PU-Net dataset. The results are presented in Table 5 and Fig. 8. It shows that the term E_{disp} firmly guides the direction of vertex movement, ensuring that the overall shape remains as close as possible to the initial shape. When the term E_{disp} is excluded, the vertices tend to move in incorrect directions influenced by other factors, leading to suboptimal outcomes, such as the clustering of points in a single location and the occurrence of shape holes, as illustrated in Fig. 8(a). On the other hand, the term E_{unit} demonstrates a less explicit influence on overall performance. Serving as an auxiliary term, it helps control the normals' length to improve the accuracy of normal computation. Removing this term may lead to a slight performance degradation. According

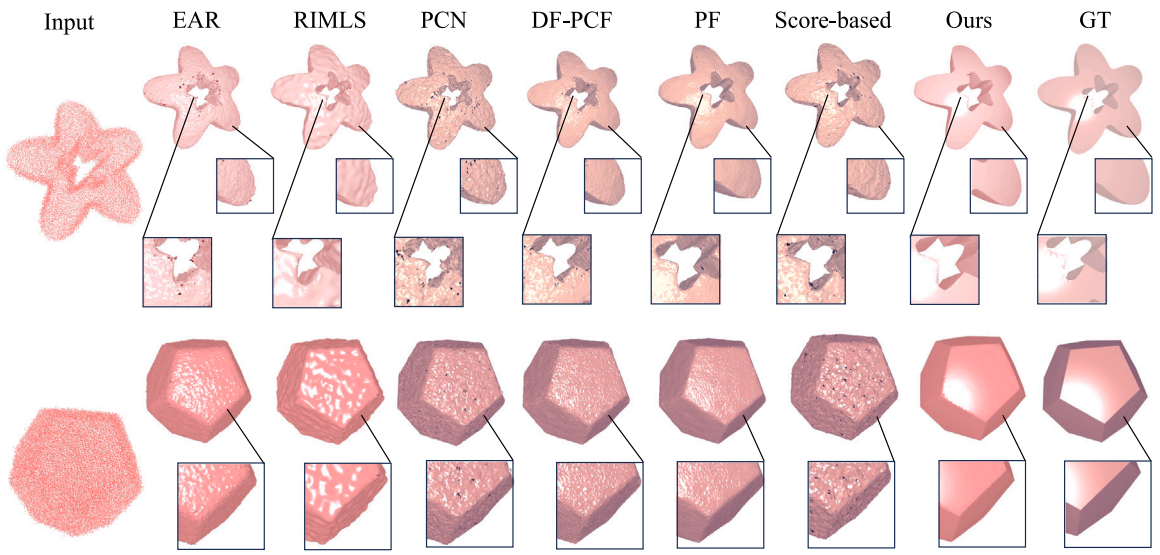


Fig. 6. Comparison on sharp feature preservation. The model 'Star' at the top is with 2.0% Gaussian noise. The 'Polyhedron' in the bottom is with 1.5% Gaussian noise.

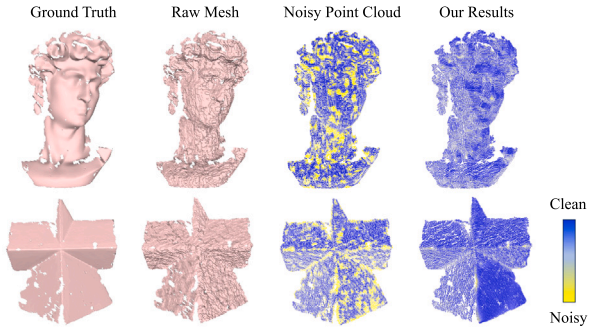


Fig. 7. Our method is effective on non-uniform point clouds. Here we show some experimental results on the Kinect v1 dataset from [10], where the point clouds are captured using depth images from a Kinect camera and are non-uniform on the object surface. The input and results are visualized using the mesh connectivity provided by the Kinect v1 dataset.

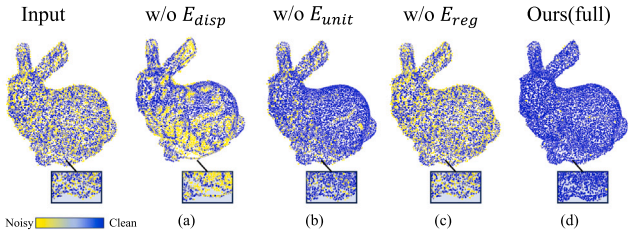


Fig. 8. Ablation study results on the Bunny Model with 10 K points and 1% Gaussian noise. (a) Without E_{disp}, the preservation of shape is compromised, resulting in a poor outcome. (b) Without E_{unit}, the performance suffers particularly in areas with prominent features, while smoother areas are relatively better preserved. (c) Without E_{reg}, the optimization will reach the minimum and terminate at the beginning.

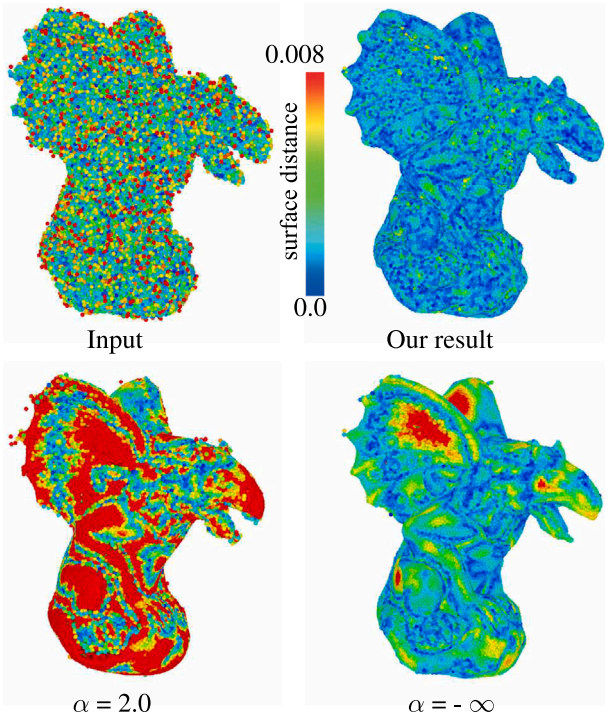


Fig. 9. Comparison on different metric settings using Gargoyle model with 20 K points and 1.0% Gaussian noise. From left to right, top to bottom, the noisy input data, our adaptive metric result, result from ℓ_2-norm ($\alpha = 2$), Welsch function ($\alpha = -\infty$).

to Eq. (1), the core term E_{reg} cannot be disregarded in the optimization. The optimization fails to proceed without this regularization term as all the energy is minimized.

To demonstrate the effectiveness of gradually decreasing the value of α, in Fig. 9 we compare our results with alternative settings with a fixed parameter value of $\alpha = 2$ and $\alpha = -\infty$, respectively. We can see that with a fixed $\alpha = 2$, the resulting ℓ_2-norm metric can cause over-smoothing and fail to preserve details. With a fixed $\alpha = -\infty$, the result achieves better preservation of details thanks to the use of Welsch function as a robust norm, but there are still areas with large deviations from the ground-truth shape. This is because the noisy input point cloud is not a suitable initial solution for the Welsch-function-based formulation, and the solver produces a local minimum that is still far away from the desirable result. In contrast, our strategy of gradually changing α helps to steer the point cloud towards a desirable solution, achieving a notably better outcome than fixing α.

5. Discussion and conclusion

In this paper, we present a novel point cloud denoising method based on a generalized robust metric. By optimizing the objective

function with an adaptive metric setting, the point cloud can be effectively denoised with features being successfully preserved. We demonstrate that our approach achieves state-of-the-art performances compared with representative methods in the field.

In the future, we would like to further improve the method in the following directions. First, although the optimal parameters of our method lie within a small range, they still need to be manually chosen according to the level of noise and the sharpness of prominent features. A strategic and systematic parameter setting optimization is worth exploring. Second, currently our method cannot handle extremely noisy data with outliers. A preprocessing step of outlier removal may be added to enhance its robustness in the future. Lastly, although we do not observe topological changes resulting from our denoising algorithm in our experiments, our current framework does not guarantee the preservation of topology. This is because it uses KNN to determine the neighbors of a point: as KNN is based on the Euclidean distance, points that are nearby in the ambient space but far away on the underlying surface may be mistakenly treated as neighbors, which can potentially lead to topological changes. Improving our method with topology preservation guarantee will be an interesting future work.

CRediT authorship contribution statement

Qun-Ce Xu: Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Yong-Liang Yang:** Formal analysis, Investigation, Methodology, Project administration, Supervision, Writing – review & editing. **Bailin Deng:** Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Qun-Ce Xu reports financial support was provided by National Natural Science Foundation of China. Qun-Ce Xu reports financial support was provided by Research Grant of Beijing Higher Institution Engineering Research Center. Qun-Ce Xu reports financial support was provided by Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. Yong-Liang Yang reports financial support was provided by UK Research and Innovation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Number: 62220106003), Research Grant of Beijing Higher Institution Engineering Research Center, Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, China, and RCUK, UK grant CAMERA (EP/M023281/1, EP/T022523/1).

Appendix A. Verification of surrogate functions

In this section, we first show that the function in Eq. (18) is indeed a surrogate function for the target function in Eq. (15), i.e., they satisfy the surrogate function conditions given in Eq. (16). We note that the first three terms in Eq. (15) are already convex quadratic and are used directly in Eq. (18). Thus we only need to verify the property for the term E_{reg}, which amounts to showing that given $x_0 \geq 0$, the quadratic function

$$\bar{\phi}(x; x_0) = \Psi(x_0) \cdot x^2 + \Theta(x_0)$$

is a surrogate function for the function $\phi_{\alpha,c}$ in Eq. (14) with $\alpha \geq 2$, and the coefficient functions Ψ, Θ are defined in Eqs. (19) and (20), respectively. Since it is trivial to verify the first condition of Eq. (16), we will focus on the second condition, namely that the surrogate function bounds the original function from above.

We first consider the cases $\alpha = 2$ and $\alpha = -\infty$. The former case is trivial as the function $\phi_{\alpha,c}$ is already quadratic. The latter case has been verified in [60].

Next, we consider the case where $-\infty < \alpha < 2$. We compute the derivative functions of $\phi_{\alpha,c}(x)$ and $\bar{\phi}(x; x_0)$ as

$$\phi'_{\alpha,c}(x) = \frac{x}{c^2}\left(\frac{(x/c)^2}{|\alpha-2|}+1\right)^{(\alpha-2)/2},$$
$$\bar{\phi}'(x; x_0) = \frac{x}{c^2}\left(\frac{(x_0/c)^2}{|\alpha-2|}+1\right)^{(\alpha-2)/2}.$$

Since $\alpha < 2$, we have

$$\left(\frac{(x/c)^2}{|\alpha-2|}+1\right)^{(\alpha-2)/2} > \left(\frac{(x_0/c)^2}{|\alpha-2|}+1\right)^{(\alpha-2)/2} \quad \text{if } x < x_0,$$
$$\left(\frac{(x/c)^2}{|\alpha-2|}+1\right)^{(\alpha-2)/2} < \left(\frac{(x_0/c)^2}{|\alpha-2|}+1\right)^{(\alpha-2)/2} \quad \text{if } x > x_0.$$

Therefore,

$$\phi'_{\alpha,c}(x) \geq \bar{\phi}'(x; x_0) \quad \forall x \in [0, x_0),$$
$$\phi'_{\alpha,c}(x) < \bar{\phi}'(x; x_0) \quad \forall x \in (x_0, +\infty).$$

Since $\phi_{\alpha,c}(x_0) = \bar{\phi}(x_0; x_0)$, then for any $x \in [0, x_0)$ we have

$$\begin{aligned}\bar{\phi}(x; x_0) &= \bar{\phi}(x_0; x_0) - \int_x^{x_0} \bar{\phi}'(x; x_0)dx \\ &\geq \phi_{\alpha,c}(x_0) - \int_x^{x_0} \phi'_{\alpha,c}(x)dx \\ &= \phi_{\alpha,c}(x).\end{aligned}$$

And for any $x \in (x_0, +\infty)$ we have

$$\begin{aligned}\bar{\phi}(x; x_0) &= \bar{\phi}(x_0; x_0) + \int_x^{x_0} \bar{\phi}'(x; x_0)dx \\ &\geq \phi_{\alpha,c}(x_0) + \int_x^{x_0} \phi'_{\alpha,c}(x)dx \\ &= \phi_{\alpha,c}(x).\end{aligned}$$

Therefore, $\bar{\phi}(x; x_0) \geq \phi_{\alpha,c}(x)$ if $x \geq 0$. Since both $\bar{\phi}(x; x_0)$ and $\phi_{\alpha,c}(x)$ are even functions, we have

$$\bar{\phi}(x; x_0) \geq \phi_{\alpha,c}(x) \quad \forall x \in \mathbb{R},$$

which completes the proof.

Using the same arguments, we can verify that the function in Eq. (25) is a surrogate function for the target function in Eq. (23).

Appendix B. Proof of Proposition 3.1

We first prove the following:

Lemma B.1. *The* $\mathbf{N}$*-update step in Eq. (17) satisfies:*

$$E(\mathbf{P}^{(k)}, \mathbf{N}^{(k+1)}) \leq E(\mathbf{P}^{(k)}, \mathbf{N}^{(k)}). \tag{30}$$

Moreover, $E(\mathbf{P}^{(k)}, \mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)}, \mathbf{N}^{(k)})$ *if and only if* $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$*, which is equivalent to the condition that* $\frac{\partial E}{\partial \mathbf{N}}(\mathbf{P}^{(k)}, \mathbf{N}^{(k)}) = \mathbf{0}$.

Proof. Let $Q(\mathbf{N})$ denote the target function in Eq. (15). Since $\mathbf{N}^{(k+1)}$ is the global minimum of its surrogate function $\bar{Q}(\mathbf{N}; \mathbf{N}^{(k)})$, we have

$\bar{Q}(\mathbf{N}^{(k+1)};\mathbf{N}^{(k)}) \le \bar{Q}(\mathbf{N}^{(k)};\mathbf{N}^{(k)})$. Combined with the conditions (16) for surrogate functions, we have:

$$Q(\mathbf{N}^{(k+1)}) \le \bar{Q}(\mathbf{N}^{(k+1)};\mathbf{N}^{(k)}) \le \bar{Q}(\mathbf{N}^{(k)};\mathbf{N}^{(k)}) = Q(\mathbf{N}^{(k)}), \tag{31}$$

i.e., the update to $\mathbf{N}^{(k+1)}$ does not increase the value of function Q. The Eq. (30) follows from the fact that the difference between $Q(\mathbf{N})$ and $E(\mathbf{P}^{(k)},\mathbf{N})$ is a constant.

Next we show $E(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k)})$ if and only if $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$. Obviously, if $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$ then $E(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k)})$. On the other hand, if $E(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k)})$, then $Q(\mathbf{N}^{(k+1)}) = Q(\mathbf{N}^{(k)})$, and Eq. (31) means that

$$\bar{Q}(\mathbf{N}^{(k+1)};\mathbf{N}^{(k)}) = \bar{Q}(\mathbf{N}^{(k)};\mathbf{N}^{(k)}). \tag{32}$$

Since $\bar{Q}(\mathbf{N};\mathbf{N}^{(k)})$ is a strongly convex quadratic function, it has a unique minimizer. Since $\mathbf{N}^{(k+1)}$ is a minimizer of $\bar{Q}(\mathbf{N};\mathbf{N}^{(k)})$, Eq. (32) implies that $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$.

Finally, we show that $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$ is equivalent to the condition that $\frac{\partial E}{\partial \mathbf{N}}(\mathbf{P}^{(k)},\mathbf{N}^{(k)}) = \mathbf{0}$. $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$ if and only if $\mathbf{N}^{(k)}$ is a minimizer of $\bar{Q}(\mathbf{N};\mathbf{N}^{(k)})$, which means that $\nabla\bar{Q}(\mathbf{N}^{(k)};\mathbf{N}^{(k)}) = \mathbf{0}$. From the surrogate function conditions (16) we have

$$\nabla Q(\mathbf{N}^{(k)}) = \nabla\bar{Q}(\mathbf{N}^{(k)};\mathbf{N}^{(k)}) = \mathbf{0}.$$

Since $E(\mathbf{P}^{(k)},\mathbf{N})$ and $Q(\mathbf{N})$ differ by a constant, we have

$$\frac{\partial E}{\partial \mathbf{N}}(\mathbf{P}^{(k)},\mathbf{N}^{(k)}) = \nabla Q(\mathbf{N}^{(k)}) = \mathbf{0},$$

which completes the proof. □

Similarly, we can prove the following:

Lemma B.2. *The* **P***-update step in Eq.* (24) *satisfies:*

$$E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) \le E(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}). \tag{33}$$

Moreover, $E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)})$ *if and only if* $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)}$, *which is equivalent to the condition that* $\frac{\partial E}{\partial \mathbf{P}}(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}) = \mathbf{0}$.

Then we prove Proposition 3.1:

Proof of Proposition 3.1. From Lemma B.1 and Lemma B.2 we have

$$E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) \le E(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}) \le E(\mathbf{P}^{(k)},\mathbf{N}^{(k)}), \tag{34}$$

which proves Eq. (28).

If $E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k)})$, then from Eq. (34) we have

$$E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k)}). \tag{35}$$

Then from Lemma B.1 we have $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$, and from Lemma B.2 we have $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)}$. On the other hand, if $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)}$ and $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$, then Eq. (35) follows from Lemma B.1 and Lemma B.2. This proves the equivalence between $E(\mathbf{P}^{(k+1)},\mathbf{N}^{(k+1)}) = E(\mathbf{P}^{(k)},\mathbf{N}^{(k)})$ and $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)},\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$.

If $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)},\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$, then from Lemma B.1 and Lemma B.2 we have $\frac{\partial E}{\partial \mathbf{N}}(\mathbf{P}^{(k)},\mathbf{N}^{(k)}) = \mathbf{0}$ and

$$\frac{\partial E}{\partial \mathbf{P}}(\mathbf{P}^{(k)},\mathbf{N}^{(k)}) = \frac{\partial E}{\partial \mathbf{P}}(\mathbf{P}^{(k)},\mathbf{N}^{(k+1)}) = \mathbf{0}. \tag{36}$$

On the other hand, if $\nabla E(\mathbf{P}^{(k)},\mathbf{N}^{(k)}) = \mathbf{0}$, then from Lemma B.1 we have $\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$, which implies Eq. (36). Then it follows from Lemma B.2 that $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)}$. This proves the equivalence between $\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)},\mathbf{N}^{(k+1)} = \mathbf{N}^{(k)}$ and the condition $\nabla E(\mathbf{P}^{(k)},\mathbf{N}^{(k)}) = \mathbf{0}$, which completes the whole proof. □

References

[1] X.-F. Han, J.S. Jin, M.-J. Wang, W. Jiang, L. Gao, L. Xiao, A review of algorithms for filtering the 3D point cloud, Signal Process., Image Commun. 57 (2017) 103–112.

[2] H. Avron, A. Sharf, C. Greif, D. Cohen-Or, ℓ_1-Sparse reconstruction of sharp point set surfaces, ACM Trans. Graph. (2010).

[3] Y. Sun, S. Schaefer, W. Wang, Denoising point sets via L_0 minimization, Comput. Aided Geom. Design 35–36 (2015) 2–15.

[4] Y. Zheng, G. Li, X. Xu, S. Wu, Y. Nie, Rolling normal filtering for point clouds, Comput. Aided Geom. Design 62 (2018) 16–28.

[5] X. Lu, S. Schaefer, J. Luo, L. Ma, Y. He, Low rank matrix approximation for 3D geometry filtering, IEEE Trans. Vis. Comput. Graphics (2020) 1.

[6] X. Sun, P.L. Rosin, R. Martin, F. Langbein, Fast and effective feature-preserving mesh denoising, IEEE Trans. Vis. Comput. Graphics 13 (5) (2007) 925–938.

[7] Y. Zheng, H. Fu, O.K. Au, C. Tai, Bilateral normal filtering for mesh denoising, IEEE Trans. Vis. Comput. Graphics 17 (10) (2011) 1521–1530.

[8] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, L. Liu, Guided mesh normal filtering, Comput. Graph. Forum 34 (7) (2015) 23–34.

[9] P.-S. Wang, X.-M. Fu, Y. Liu, X. Tong, S.-L. Liu, B. Guo, Rolling guidance normal filter for geometric processing, ACM Trans. Graph. 34 (6) (2015).

[10] P.-S. Wang, Y. Liu, X. Tong, Mesh denoising via cascaded normal regression., ACM Trans. Graph. 35 (6) (2016) 1–232.

[11] J. Zhang, B. Deng, Y. Hong, Y. Peng, W. Qin, L. Liu, Static/dynamic filtering for mesh geometry, IEEE Trans. Vis. Comput. Graphics 25 (4) (2018) 1774–1787.

[12] J.T. Barron, A general and adaptive robust loss function, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019.

[13] K. Lange, MM Optimization Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.

[14] L. Zhou, G. Sun, Y. Li, W. Li, Z. Su, Point cloud denoising review: from classical to deep learning-based approaches, Graph. Models 121 (2022) 101140.

[15] D. Levin, The approximation power of moving least-squares, Math. Comp. 67 (224) (1998) 1517–1531.

[16] D. Levin, Mesh-independent surface interpolation, in: G. Brunnett, B. Hamann, H. Müller, L. Linsen (Eds.), Geometric Modeling for Scientific Visualization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 37–49.

[17] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C.T. Silva, Point set surfaces, in: Proceedings of the Conference on Visualization '01, VIS '01, IEEE Computer Society, USA, 2001, pp. 21–28.

[18] M. Pauly, M. Gross, L.P. Kobbelt, Efficient simplification of point-sampled surfaces, in: Proceedings of the Conference on Visualization '02, VIS '02, IEEE Computer Society, USA, 2002, pp. 163–170.

[19] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C.T. Silva, Computing and rendering point set surfaces, IEEE Trans. Vis. Comput. Graphics 9 (1) (2003) 3–15.

[20] N. Amenta, Y.J. Kil, Defining point-set surfaces, ACM Trans. Graph. 23 (3) (2004) 264–270.

[21] A. Adamson, M. Alexa, Point-sampled cell complexes, ACM Trans. Graph. 25 (3) (2006) 671–680.

[22] G. Guennebaud, M. Gross, Algebraic point set surfaces, ACM Trans. Graph. 26 (3) (2007).

[23] S. Fleishman, D. Cohen-Or, C.T. Silva, Robust moving least-squares fitting with sharp features, ACM Trans. Graph. 24 (3) (2005) 544–552.

[24] A.C. Öztireli, G. Guennebaud, M. Gross, Feature preserving point set surfaces based on non-linear kernel regression, Comput. Graph. Forum 28 (2) (2009) 493–501.

[25] Y. Lipman, D. Cohen-Or, D. Levin, H. Tal-Ezer, Parameterization-free projection for geometry reconstruction, in: ACM SIGGRAPH 2007 Papers, SIGGRAPH '07, Association for Computing Machinery, New York, NY, USA, 2007.

[26] H. Huang, D. Li, H. Zhang, U. Ascher, D. Cohen-Or, Consolidation of unorganized point clouds for surface reconstruction, ACM Trans. Graph. 28 (5) (2009) 1–7.

[27] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, H.R. Zhang, Edge-aware point set resampling, ACM Trans. Graph. 32 (1) (2013).

[28] B. Liao, C. Xiao, L. Jin, H. Fu, Efficient feature-preserving local projection operator for geometry reconstruction, Comput. Aided Des. 45 (5) (2013) 861–874.

[29] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, M. Wimmer, Continuous projection for fast L1 reconstruction, ACM Trans. Graph. 33 (4) (2014).

[30] O. Schall, A. Belyaev, H.-P. Seidel, Robust filtering of noisy scattered point data, in: Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics, SPBG '05, Eurographics Association, Goslar, DEU, 2005, pp. 71–77.

[31] P. Jenke, M. Wand, M. Bokeloh, A. Schilling, W. Straßer, Bayesian point cloud reconstruction, Comput. Graph. Forum 25 (3) (2006) 379–388.

[32] E. Kalogerakis, D. Nowrouzezahrai, P. Simari, K. Singh, Extracting lines of curvature from noisy point clouds, Comput. Aided Des. 41 (4) (2009) 282–292, Point-based Computational Techniques.

[33] E. Mattei, A. Castrodad, Point cloud denoising via moving RPCA, Comput. Graph. Forum 36 (8) (2017) 123–137.

[34] L. Yu, X. Li, C. Fu, D. Cohen-Or, P. Heng, PU-net: Point cloud upsampling network, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2790–2799.

[35] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS '17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 5105–5114.

[36] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, P.-A. Heng, EC-net: An edge-aware point set consolidation network, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision – ECCV 2018, Springer International Publishing, Cham, 2018, pp. 398–414.
[37] R. Roveri, A.C. Öztireli, I. Pandele, M.H. Gross, PointProNets: Consolidation of point clouds with convolutional neural networks, Comput. Graph. Forum 37 (2) (2018) 87–99.
[38] C.R. Qi, H. Su, M. Kaichun, L.J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 77–85.
[39] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N.J. Mitra, M. Ovsjanikov, PointCleanNet: Learning to denoise and remove outliers from dense point clouds, Comput. Graph. Forum (2019).
[40] D. Zhang, X. Lu, H. Qin, Y. He, Pointfilter: Point cloud filtering via encoder-decoder modeling, IEEE Trans. Vis. Comput. Graphics 27 (3) (2020) 2015–2027.
[41] D. Lu, X. Lu, Y. Sun, J. Wang, Deep feature-preserving normal estimation for point cloud filtering, Comput. Aided Des. 125 (2020) 102860.
[42] S. Luo, W. Hu, Score-based point cloud denoising, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4583–4592.
[43] H. Chen, S. Luo, W. Hu, et al., Deep point set resampling via gradient fields, IEEE Trans. Pattern Anal. Mach. Intell. 45 (3) (2022) 2913–2930.
[44] T. Zhao, P. Gao, T. Tian, J. Ma, J. Tian, From noise addition to denoising: A self-variation capture network for point cloud optimization, IEEE Trans. Vis. Comput. Graphics (2022).
[45] X. Wang, W. Cui, R. Xiong, X. Fan, D. Zhao, FCNet: Learning noise-free features for point cloud denoising, IEEE Trans. Circuits Syst. Video Technol. (2023).
[46] P. Hermosilla, T. Ritschel, T. Ropinski, Total denoising: Unsupervised learning of 3D point cloud cleaning, in: The IEEE International Conference on Computer Vision, ICCV, 2019.
[47] N. Schertler, B. Savchynskyy, S. Gumhold, Towards globally optimal normal orientations for large point clouds, Comput. Graph. Forum 36 (1) (2017) 197–208.
[48] R. Xu, Z. Dou, N. Wang, S. Xin, S. Chen, M. Jiang, X. Guo, W. Wang, C. Tu, Globally consistent normal orientation for point clouds by regularizing the winding-number field, ACM Trans. Graph. 42 (4) (2023) 1–15.
[49] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, Optimization with sparsity-inducing penalties, Found. Trends Mach. Learn. 4 (1) (2012) 1–106.
[50] S. Rusinkiewicz, A symmetric objective function for ICP, ACM Trans. Graph. 38 (4) (2019).
[51] J. Wang, Z. Yang, F. Chen, A variational model for normal computation of point clouds, Vis. Comput. 28 (2) (2012) 163–174.
[52] J.T. Barron, A general and adaptive robust loss function, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4331–4339.
[53] P. Charbonnier, L. Blanc-Feraud, G. Aubert, M. Barlaud, Two deterministic half-quadratic regularization algorithms for computed imaging, in: Proceedings of 1st International Conference on Image Processing, vol. 2, IEEE, 1994, pp. 168–172.
[54] M.J. Black, P. Anandan, The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields, Comput. Vis. Image Underst. 63 (1) (1996) 75–104.
[55] S. Geman, D.E. McClure, Bayesian image analysis: An application to single photon emission tomography, in: Proceedings of the American Statistical Association, 1985.
[56] J.E. Dennis Jr., R.E. Welsch, Techniques for nonlinear least squares and robust regression, Commun. Stat.-Simul. Comput. 7 (4) (1978) 345–359.
[57] G. Guennebaud, B. Jacob, et al., Eigen v3, 2010, http://eigen.tuxfamily.org.
[58] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, D. Panozzo, ABC: A big CAD model dataset for geometric deep learning, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019.
[59] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Trans. Vis. Comput. Graphics 5 (4) (1999) 349–359.
[60] B. Ham, M. Cho, J. Ponce, Robust image filtering using joint static and dynamic guidance, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 4823–4831.