Low Rank Matrix Approximation for 3D Geometry Filtering

Xuequan Lu, *Member, IEEE,* Scott Schaefer, Jun Luo, *Senior Member, IEEE,* Lizhuang Ma, *Member, IEEE,* and Ying He, *Member, IEEE*

Abstract—We propose a robust normal estimation method for both point clouds and meshes using a low rank matrix approximation algorithm. First, we compute a local isotropic structure for each point and find its similar, non-local structures that we organize into a matrix. We then show that a low rank matrix approximation algorithm can robustly estimate normals for both point clouds and meshes. Furthermore, we provide a new filtering method for point cloud data to smooth the position data to fit the estimated normals. We show the applications of our method to point cloud filtering, point set upsampling, surface reconstruction, mesh denoising, and geometric texture removal. Our experiments show that our method generally achieves better results than existing methods.

Index Terms—3D Geometry filtering, Point cloud filtering, Mesh denoising, Point upsampling, Surface reconstruction, Geometric texture removal.

1 Introduction

Filtering in 2D data like images is prevalent nowadays [1], [2], [3], [4], and 3D geometry (e.g., point clouds, meshes) filtering and processing has recently attracted more and more attention in 3D vision [5], [6], [7]. Normal estimation for point cloud models or mesh shapes is important since it is often the first step in a geometry processing pipeline. This estimation is often followed by a filtering process to update the position data and remove noise [8]. A variety of computer graphics applications, such as point cloud filtering [9], [10], [11], point set upsampling [12], surface reconstruction [13], mesh denoising [8], [14], [15] and geometric texture removal [16] rely heavily on the quality of estimated normals and subsequent filtering of position data.

Current state of the art techniques in mesh denoising [8], [14], [15] and geometric texture removal [16] can achieve impressive results. However, these methods are still limited in their ability to recover sharp edges in challenging regions. Normal estimation for point clouds has been an active area of research in recent years [12], [17], [18]. However, these methods perform suboptimally when estimating normals in noisy point clouds. Specifically, [17], [18] are less robust in the presence of considerable noise. The bilateral filter can preserve geometric features but sometimes may fail due to the locality of its computations and lack of self-adaption of parameters.

Updating point positions using the estimated normals in point clouds has received sparse treatment so far [9], [10]. However, those position update approaches using the L_0 or L_1 norms are complex to solve and hard to implement. Moreover, they restrict each point to only move along its normal orientation potentially leading to suboptimal results or slow convergence.

To address the issues shown above, we propose a new normal estimation method for both meshes and point clouds and a new position update algorithm for point clouds. Our method benefits various geometry processing applications, directly or indirectly, such as point cloud filtering, point set upsampling, surface reconstruction, mesh denoising, and geometric texture removal (Figure 1). Given a point cloud or mesh as input, our method first estimates point or face normals, then updates the positions of points or vertices using the estimated normals. We observe that: (1) non-local methods could be more accurate than local techniques; (2) there usually exist similar structures of each local isotropic structure (Section 3.1) in geometry shapes; (3) the matrix constructed by similar structures should be low-rank (Section 3.2). Motivated by these observations, we propose a novel normal estimation technique which consists of two sub-steps: (i) non-local similar structures location and (ii) weighted nuclear norm minimization. We adopt the former to find similar structures of each local isotropic structure. We employ the latter [3] to handle the problem of recovering low-rank matrices. We also present a fast and effective point update algorithm for point clouds to filter the point positions to better match the estimated normals. Extensive experiments and comparisons show that our method generally outperforms current methods.

The **main contributions** of this paper are:

- a novel normal estimation technique for both point cloud shapes and mesh models;
- a new position update algorithm for point cloud data;
- analysis of the convergence of the proposed normal estimation technique and point update algorithm, experimentally or theoretically.

- *X. Lu is with School of Information Technology, Deakin University, Australia. E-mail: xuequan.lu@deakin.edu.au.*
- *J. Luo and Y. He are with School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mails: {junluo, YHe}@ntu.edu.sg.*
- *S. Schaefer is with Department of Computer Science, Texas A&M University, College Station, Texas, USA. E-mail: schaefer@cs.tamu.edu.*
- *L. Ma is with Department of Computer Science, Shanghai Jiao Tong University, Shanghai, China. E-mail: ma-lz@cs.sjtu.edu.cn.*

Manuscript received July 22, 2019; revised ZZ, 2019.

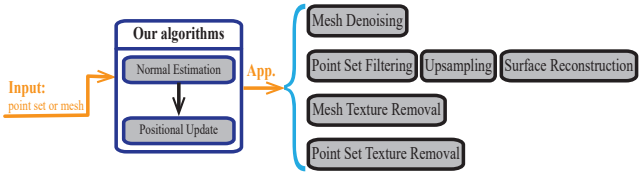


Fig. 1. Overview of our approach and its benefited applications. Our method can be applied to various geometry processing tasks directly or indirectly.

2 Related Work

In this section, we only review the research works that are most related to this work. We first review the previous research on normal estimation. Then we review some previous works which employed the nuclear norm minimization or its weighted version.

2.1 Normal Estimation

Normal estimation for geometric shapes can be classified into two types: (1) normal estimation for point clouds, and (2) normal estimation for mesh shapes.

Normal estimation for point clouds. Hoppe et al. [19] estimated normals by computing the tangent plane at each data point using principal component analysis (PCA) of the local neighborhood. Later, a variety of variants of PCA have been proposed [20], [21], [22], [23], [24] to estimate normals. Nevertheless, the normals estimated by these techniques tend to smear sharp features. Researchers also estimate normals using Voronoi cells or Voronoi-PCA [25], [26]. Minimizing the L_1 or L_0 norm can preserve sharp features as these norms can be used to measure sparsity in the derivative of the normal field [9], [10]. Yet, the solutions are complex and computationally expensive. Li et al. [27] estimated normals by using robust statistics to detect the best local tangent plane for each point. Another set of techniques attempted to better estimate normals near edges and corners by point clustering in a neighborhood [28], [29]. Later they presented a pair consistency voting scheme which outputs multiple normals per feature point [30]. Boulch and Marlet [17] use a robust randomized Hough transform to estimate point normals. Convolutional neural networks have recently been applied to estimate normals in point clouds [18]. Such estimation methods are usually less robust for point clouds with considerable amount of noise. Bilateral smoothing of PCA normals [12], [13] is simple and effective, but it suffers from inaccuracy due to the locality of its computations and may blur edges with small dihedral angles. Mattei et al. [31] presented a moving RPCA method for point cloud denoising, based on the inspiration of sparsity. They modeled the RPCA problem in a local sense by specifying the output rank of 2, rather than considering similar structures. The computed normals are only used to compute similarity weights.

Normal estimation for mesh shapes. Most methods focus on the estimation of face normals in mesh shapes. One simple, direct way is to compute the face normals by the cross product of two edges in a triangle face. However, such normals can deviate from the true normals significantly even in the presence of small position noise. There exist a considerable amount of research work to smooth these face normals. One approach uses the bilateral filter [14], [32], [33], inspired by the founding works [34], [35]. Mean, median and alpha-trimming methods [36], [37], [38] are also used to estimate face normals. Sun et al. [8], [39] present two different methods to filter face normals. Recently, researchers have presented filtering methods [15], [40], [41], [42], [43] based on mean shift, total variation, guided normals, L_1 median, and normal voting tensor. Wang et al. [44] estimated face normals via cascaded normal regression.

2.2 Nonlocal Methods for Point Clouds and Nuclear Norm Minimization

Previous researchers proposed non-local methods for point clouds. For example, Zheng et al. [45] applied non-local filtering to 3D buildings that exhibit large scale repetitions and self-similarities. Digne presented a non-local denoising framework to unorganized point clouds by building an intrinsic descriptor [46], and recently proposed a shape analysis approach with colleagues based on the non-local analysis of local shape variations [47].

The nuclear norm of a matrix is defined as the sum of the absolute values of its singular values (see Eq. (4)). It has been proved that most low-rank matrices can be recovered by minimizing their nuclear norm [48]. Cai et al. [49] provided a simple solution to the low-rank matrix approximation problem by minimizing the nuclear norm. The nuclear norm minimization has been broadly employed to matrix completion [48], [49], robust principle component analysis [50], low-rank representation for subspace clustering [51] and low-rank textures [52]. Gu et al. [3], [4] presented a weighted version of the nuclear norm minimization, which has been adopted to image processing applications such as image denoising, background subtraction and image inpainting.

3 Normal Estimation

In this section, we take point clouds, consisting of positions as well as normals, as input and further extend to meshes later. As with [10], [11], [12], the normals are initialized by the classical PCA method [19], which is robust and easy to use (we use the implementation in [12]). First of all, we present an algorithm to locate and construct non-local similar structures for each local isotropic structure of a point (Section 3.1). We then describe how to estimate normals via weighted nuclear norm minimization on non-local similar structures (Section 3.2).

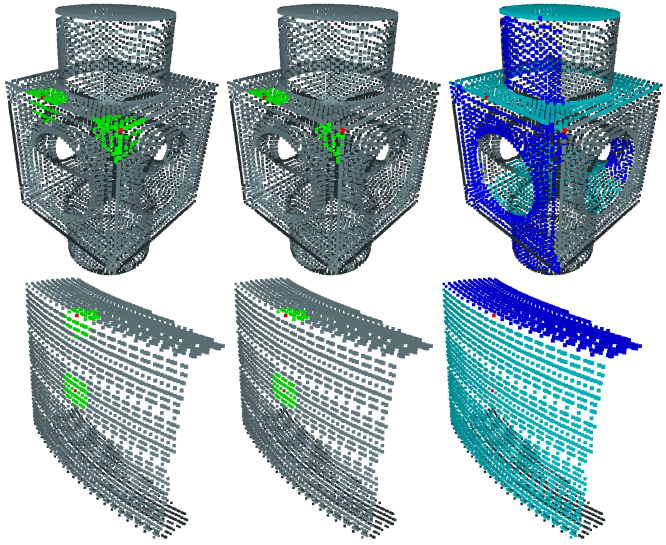
3.1 Non-local Similar Structures

Local structure. We define each point $\mathbf{p}_i$ has a local structure S_i which consists of k_{local} nearest neighbors. Locating structures similar to a specific local structure is difficult due to the irregularity of points.

Tensor voting. We assume each local structure embeds a representative normal. To do so, we first define the tensor at a point $\mathbf{p}_i$ as

$$\mathbf{T}_{ij} = \eta(\|\mathbf{p}_i - \mathbf{p}_j\|)\phi(\theta_{ij}, \sigma_\theta)\mathbf{n}_j^T\mathbf{n}_j, \quad (1)$$

where $\mathbf{p}_j$ (1×3 vector) is one of the k_{local} nearest neighbors of $\mathbf{p}_i$, which we denote as $j \in S_i$, and $\mathbf{n}_j$ (1×3 vector)



(a) Local structure (b) Local isotropic structure (c) Similar structures

Fig. 2. (a) The local structures (green points) of the centered red points, respectively. (b) The local isotropic structures (green) of the corresponding red points. (c) The similar local isotropic structures of the consistent local isotropic structures denoted by the red points. Each blue or cyan point denotes its isotropic structure.

is the normal of $\mathbf{p}_j$. η and ϕ are the weights induced by spatial distances and angles (θ_{ij}) of two neighboring normals, which are given by [12], [14]: $\eta(x) = e^{-(\frac{x}{\sigma_p})^2}$, $\phi(\theta, \sigma_\theta) = e^{-(\frac{1-\cos(\theta)}{1-\cos(\sigma_\theta)})^2}$. σ_p and σ_θ are the scaling parameters, which are empirically set to two times the maximal distance between any two points in the k_{local} nearest neighbors within the local structure and 30°, respectively.

For each local structure S_i, we can derive the accumulated tensor by aggregating all the induced tensor votes $\{\mathbf{T}_{ij}|j \in S_i\}$. This final tensor encodes the local structure, which provides a reliable, representative normal that will be later used to compute the local isotropic structure and locate similar structures.

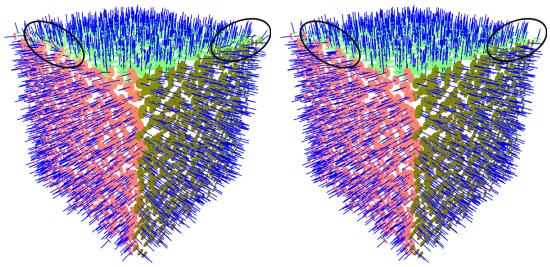
$$\mathbf{T}_i = \sum_{j \in S_i} \mathbf{T}_{ij} \tag{2}$$

Let $\lambda_{i1} \geq \lambda_{i2} \geq \lambda_{i3}$ be the eigenvalues of $\mathbf{T}_i$ with the corresponding eigenvectors $\mathbf{e}_{i1}$, $\mathbf{e}_{i2}$ and $\mathbf{e}_{i3}$. In tensor voting [53], $\lambda_{i1} - \lambda_{i2}$ indicates surface saliency with a normal direction $\mathbf{e}_{i1}$; $\lambda_{i2} - \lambda_{i3}$ indicates curve saliency with a tangent orientation $\mathbf{e}_{i3}$; λ_{i3} denotes junction saliency. Therefore, we take $\mathbf{e}_{i1}$ as the representative normal for the local structure S_i of point $\mathbf{p}_i$.

Local isotropic structure. We assume that each local structure has a subset of points that are on the same isotropic surface with the representative normal. We call this subset of points the local isotropic structure. Surface patches with small variation in its dihedral angles are usually considered isotropic (Figure 2(b)) surfaces. To obtain a local isotropic structure S_i^{iso} from a local structure S_i and locate similar local isotropic structures for S_i^{iso}, we present a simple yet effective scheme. Here we also employ the defined function $\phi(\theta, \sigma_\theta)$ in Eq. (1), with setting σ_θ to θ_{th}. θ is the angle of two normals and θ_{th} is the angle threshold. Specifically, to obtain S_i^{iso}, we

- compute the angles θ between each point normal and the representative normal within a local structure S_i;
- add the current point to S_i^{iso} if $\phi(\theta, \theta_{th}) \geq e^{-(\frac{1-\cos(\theta_{th})}{1-\cos(\theta_{th})})^2}$ (i.e., $\phi(\theta, \theta_{th}) \geq e^{-1}$).

For simplicity, we will call "similar local isotropic structures" as "similar structures" throughout the paper, unless otherwise stated (e.g., Figure 6). Given an isotropic structure S_i^{iso}, we identify its non-local similar structures by computing $\phi(\theta, \theta_{th})$ between the representative normal of each structure and that of S_i. If $\phi(\theta, \theta_{th}) \geq e^{-1}$ (we use the same θ_{th} for simplicity), we define the two isotropic structures to be similar. The underlying rationale of our similarity search is: the point normals in a local isotropic structure are bounded by the representative normal, indicating these points are on the same isotropic surface; the similar structures search is also bounded by the representative normals, implying the similar structures are on the similar isotropic surfaces. These similar structures will often overlap on the same isotropic surface as shown in Figure 2. In the figure, we show the local structure (a), the local isotropic structure (b), and the similar structures (c). Each representative normal is computed based on its entire neighbors with different weights respect to the current point (Eq. (1) and (2)). It indicates that the representative normal is isotropic with the current point normal, and there is no need to iteratively refine the representative normal by using the local isotropic neighbors. Note that the non-local similar structures are searched in the context of isotropic surfaces rather than anisotropic surfaces (see more analysis in Rotation-invariant similarity of Section 5.1).



(a) without reshaping (b) with reshaping

Fig. 3. A normal estimation comparison without or with matrix reshaping.

3.2 Weighted Nuclear Norm Minimization

For each non-local similar structure S_l^{iso} for the isotropic structure S_i^{iso} associated with the point p_i, we append the point normals of S_l^{iso} as rows to a matrix $\mathbf{M}$. Note that the dimensions of this matrix are $\hat{r} \times 3$, where $\hat{r}$ is the number of rows and 3 is the number of columns. This matrix already has a maximal rank of 3 and is a low rank matrix. Therefore, the low rank matrix approximation from rank 3 or 2 to rank 2 or 1 is less meaningful than from a high rank to a low rank, in terms of "smoothing". To make the low rank matrix approximation more meaningful, we reshape the matrix $\mathbf{M}$ to be close to a square matrix. Figure 3 illustrates a normal

estimation comparison without and with matrix reshaping, and shows that the initial $\hat{r} \times 3$ matrix $\mathbf{M}$ requires reshaping to obtain more effective smoothing results. It also shows that the reconstruction error between the initially reshaped matrix and the low rank optimized matrix is typically greater than the error computed without reshaping, which further validates a more effective smoothing of reshaping. As such, it is necessary for reshaping $\mathbf{M}$.

$$\mathbf{M} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ x_6 & y_6 & z_6 \\ x_7 & y_7 & z_7 \\ x_8 & y_8 & z_8 \end{pmatrix} \Rightarrow \mathbf{Z}' = \begin{pmatrix} x_1 & x_7 & y_5 & z_3 \\ x_2 & x_8 & y_6 & z_4 \\ x_3 & y_1 & y_7 & z_5 \\ x_4 & y_2 & y_8 & z_6 \\ x_5 & y_3 & z_1 & z_7 \\ x_6 & y_4 & z_2 & z_8 \end{pmatrix} \tag{3}$$

We do so by finding dimensions r and c of a new matrix $\mathbf{Z}'$ where $\hat{r} \times 3 = r \times c$ and we minimize $|r - c|$. Given that the structure in $\mathbf{M}$ is isotropic, removing one or more points does not affect this structure significantly. Therefore, we first find r and c to minimize $\|r-c\|$ ($r \geq c$) and measure if $|r - c| \geq 6$ and $c = 3$ are both satisfied, where 6 is an empirical value and $c = 3$ tests if the reshaping failed. If so, we remove a point normal from $\mathbf{M}$ and solve for r and c again. We repeat such a process until the conditions are not satisfied (r is not required to be a multiple of 3). Then we simply copy the column entries in $\mathbf{M}$ to $\mathbf{Z}'$ filling each column of $\mathbf{Z}'$ before continuing to the next column.

We take the size 8×3 for $\mathbf{M}$ as a simple example, and the reshaping process is illustrated in Eq. (3). The reshaped matrix $\mathbf{Z}'$ has a size of 6×4 and a higher rank than $\mathbf{M}$ in general. It is known that the rank of a matrix is the number of linearly independent columns. Intuitively, the resulting matrix $\mathbf{Z}'$ should be low rank since all normals come from similar isotropic structures and each point may involve multiple normals, and the x, y and z values are respectively gathered in columns. In $\mathbf{Z}'$, most columns consist of coordinates from a single dimension (only x coordinates, for example). There are at most two columns involving both x and y, or both y and z (Eq. (3)), which negligibly affects the rank and the smoothing results (Figure 13(a)). Experimentally, we followed the above rules to construct matrices of similar local isotropic structures for planar and curved surfaces in Figure 2, and observed the matrices are indeed low rank (i.e., a considerable number of negligible singular values). Figure 4 shows the histograms of singular values of two reshaped matrices from Figure 2, and confirms the low-rank property.

We then cast the normal estimation problem as a low-rank matrix approximation problem. We attempt to recover a low-rank matrix $\mathbf{Z}$ from $\mathbf{Z}'$ using nuclear norm minimization. We first present some fundamental knowledge about nuclear norm minimization and then show how we estimate normals with weighted nuclear norm minimization.

Nuclear norm. The nuclear norm of a matrix is defined as the sum of the absolute values of its singular values, shown in Eq. (4).

$$\|\mathbf{Z}\|_* = \sum_m |\delta_m|, \tag{4}$$

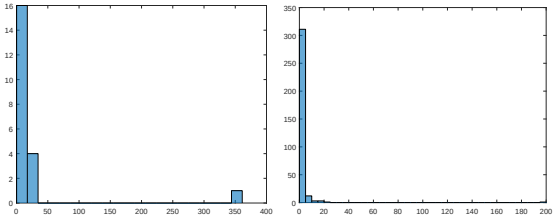


Fig. 4. Histograms of singular values of two reshaped matrices from Figure 2. Horizontal axis denotes the singular values, and vertical axis denotes the number of singular values falling into corresponding ranges.

where δ_m is the m-th singular value of matrix $\mathbf{Z}$. $\|\mathbf{Z}\|_*$ indicates the nuclear norm of $\mathbf{Z}$.

Nuclear norm minimization. Nuclear norm minimization is frequently used to approximate the known matrix, $\mathbf{Z}'$, by a low-rank matrix, $\mathbf{Z}$, while minimizing the nuclear norm of $\mathbf{Z}$. Cai et al. [49] demonstrated that the low-rank matrix $\mathbf{Z}$ can be easily solved by adding a Frobenius-norm data term.

$$\min_{\mathbf{Z}} \alpha\|\mathbf{Z}\|_* + \|\mathbf{Z}' - \mathbf{Z}\|_F^2, \tag{5}$$

where α is the weighting parameter. The minimizing matrix $\mathbf{Z}$ is then

$$\mathbf{Z} = \mathbf{U}\psi(\mathbf{S}, \alpha)\mathbf{V}^T, \tag{6}$$

where $\mathbf{Z}' = \mathbf{USV}^T$ denotes the SVD of $\mathbf{Z}'$ and $\mathbf{S}_{m,m}$ is the m-th diagonal element in $\mathbf{S}$. ψ is the soft-thresholding function on $\mathbf{S}$ and the parameter α, i.e., $\psi(\mathbf{S}_{m,m}, \alpha) = \max(0, \mathbf{S}_{m,m} - \alpha)$. Soft thresholding effectively clamps small singular values to 0, thus creating a low rank approximation.

Nuclear norm minimization treats and shrinks each singular value equally. However, in general, larger singular values should be shrunk less to better approximate the known matrix and preserve the major components. The weighted nuclear norm minimization solves this issue [3].

Weighted nuclear norm minimization. The weighted nuclear norm of a matrix $\mathbf{Z}$ is

$$\|\mathbf{Z}\|_{*,\mathbf{w}} = \sum_m |w_m \delta_m|, \tag{7}$$

where w_m is the non-negative weight imposed on the m-th singular value and $\mathbf{w} = \{w_m\}$. We can then write the low-rank matrix approximation problem as

$$\min_{\mathbf{Z}} \|\mathbf{Z}\|_{*,\mathbf{w}} + \|\mathbf{Z}' - \mathbf{Z}\|_F^2 \tag{8}$$

Suppose the singular values $\{\delta_m\}$ are sorted in a non-ascending order, the corresponding weights $\{w_m\}$ should be in a non-descending order. Hence, we define the weight function as a Gaussian function.

$$w_m = \beta e^{-(\frac{2\delta_m}{\delta_1})^2} \tag{9}$$

β denotes the regularized coefficient which defaults to 1.0. δ_1 is the first singular value after sorting $\{\delta_m\}$ in a non-increasing order. We did not use the original weight definition in [3] since it needs noise variance which should be unknown in normal estimation. Also, we found their weight determination is not suitable for normal-constructed matrices. Then we solve Eq. (8) by the generalized soft

ALGORITHM 1: Weighted nuclear norm minimization

Input: non-local similar structures of each local isotropic structure
Output: New matrices $\{\mathbf{Z}\}$
for *each local isotropic structure* S_i^{iso} **do**
- construct a matrix $\mathbf{Z}'$
- compute the SVD of $\mathbf{Z}'$
- compute the weights via Eq. (9)
- recover $\mathbf{Z}$ via Eq. (10)

end

thresholding operation on the singular values with weights [3].

$$\mathbf{Z} = \mathbf{U}\psi(\mathbf{S}, \{w_m\})\mathbf{V}^T, \tag{10}$$

where $\psi(\mathbf{S}_{m,m}, w_m) = \max(0, \mathbf{S}_{m,m} - w_m)$. Here ψ changes to the generalized soft-thresholding function by assigning weights to singular values, and Eq. (10) becomes the weighted version of Eq. (6).

Notice that the truncated SVD can also solve the low-rank matrix approximation problem. However, we found it is less effective here for two reasons. First, the truncated SVD uses a fixed number, K to determine top singular values. However, the value K is usually shape dependent. Second, truncated SVD treats each selected singular value equally. In contrast, our method treats singular values differently to enable adaptivity. Refer to the supplementary material for a comparison between truncated SVD and our method.

3.3 Algorithm

Each point may have multiple normals in the recovered matrices $\{\mathbf{Z}\}$, as the similar structures often overlap. We first reshape $\{\mathbf{Z}\}$ to matrices like $\{\mathbf{M}\}$ (each row in each matrix is a normal), and compute the final normal of each point by simply averaging the corresponding normals in $\{\mathbf{Z}\}$ after calling Algorithm 1. To achieve quality normal estimations, we iterate non-local similar structures searching (Section 3.1) and the weighted nuclear norm minimization in Algorithm 1.

Extension to mesh models. Our algorithm can be easily extended to handle mesh models. One natural way is to take the vertices/normals of a mesh as points/normals in a point cloud. However, to achieve desired results, face normals are frequently used to update vertex positions [8], [14], [15]. Hence, we use the centers of faces and the corresponding normals as points. Moreover, we use the mesh topology to compute neighbors in Section 3.1.

4 Position Update

Besides normal estimation, we also present algorithms to update point or vertex positions to match the estimated normals, which is typically necessary before applying other geometry processing algorithms.

Vertex update for mesh models. We use the algorithm [8] to update vertices of mesh models, which minimizes the square of the dot product between the normal and the three edges of each face.

Point update for point clouds. Compared to the vertex update for mesh models, updating point cloud positions is more difficult due to the absence of topological information. Furthermore, the local neighborhood information may vary during this position update. We propose a modification of the edge recovery algorithm in [10] to update points in a feature-aware way and minimize

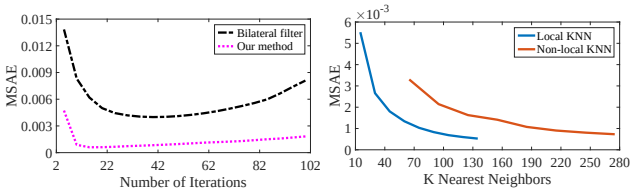
$$\sum_i \sum_{j \in S_i} |(\mathbf{p}_i - \mathbf{p}_j)\mathbf{n}_j^T|^2 + |(\mathbf{p}_i - \mathbf{p}_j)\mathbf{n}_i^T|^2. \tag{11}$$

$\mathbf{p}_i$ and $\mathbf{p}_j$ are unknown, and $\mathbf{n}_i$ and $\mathbf{n}_j$ are computed by our normal estimation algorithm. Eq. (11) encodes the sum of distances to the tangent planes defined by the neighboring points $\{\mathbf{p}_j\}$ and the corresponding normals $\{\mathbf{n}_j\}$, as well as the sum of distances to the tangent planes defined by $\{\mathbf{p}_i\}$ and $\{\mathbf{n}_i\}$. The differences between [10] and our method are: (1) [10] utilized a least squares form for alleviating artifacts on the intersection of two sharp edges; (2) [10] only considered the distance to all the planes defined by each neighboring point and the point's corresponding normal.

We use gradient descent to solve Eq. (11), by assuming the point $\mathbf{p}_i$ and its neighboring points $\{j \in S_i | \mathbf{p}_j\}$ in the previous iteration are known. Here we use ball neighbors instead of k nearest neighbors to ensure the convergence of our point update. Therefore, the new position of $\mathbf{p}_i$ can be computed by

$$\mathbf{p}_i' = \mathbf{p}_i + \gamma_i \sum_{j \in S_i} (\mathbf{p}_j - \mathbf{p}_i)(\mathbf{n}_j^T \mathbf{n}_j + \mathbf{n}_i^T \mathbf{n}_i), \tag{12}$$

where $\mathbf{p}_i'$ is the new position. γ_i is the step size, which is set to $\frac{1}{3|S_i|}$ to ensure the convergence (see Section 5).



(a) (b)

Fig. 5. (a) Normal errors of our method and the bilateral filter [12]. (b) Greater k_{local} or k_{non} leads to smaller normal errors.

5 Method Analysis

In this section, we analyze both the normal estimation and point update steps of our method.

5.1 Analysis of Normal Estimation

Convergence. As is true with previous techniques [8], [14], [15], we also cannot guarantee the convergence of our normal estimation method. Figure 5(a) shows the normal approximation error using the bilateral filter [12] and our method. Both demonstrate similar behavior where the error decreases significantly and then increases with the iteration count. However, this experiment indicates our method is more accurate. Figure 11 shows the normal estimation results after 1, 5 and 20 iterations.

Rotation-invariant similarity. As discussed in [54], non-local rotation-invariant similarity does not preserve sharp features. Figure 7 shows the comparisons of [54] and

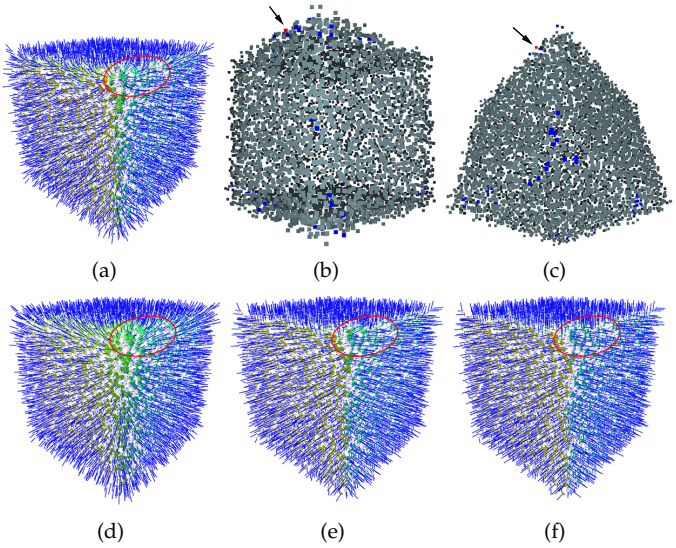


Fig. 6. (a) PCA normals [19]. (b) Non-local similar local structures of the local structure denoted by the red point. (c) Non-local similar local isotropic structures of the local isotropic structure denoted by the red point. (d) Normal estimation by low-rank minimization on (b). (e) Normal estimation by low-rank minimization on (c). (f) Normal estimation by our method. Blue lines indicate point normals.

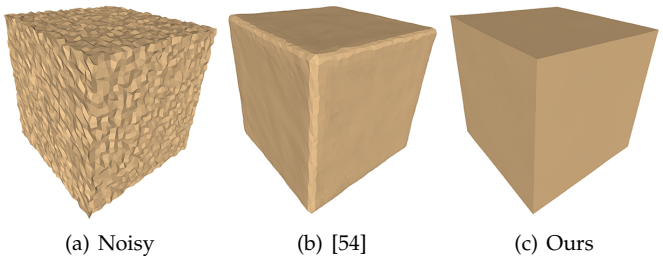


Fig. 7. Comparison with [54].

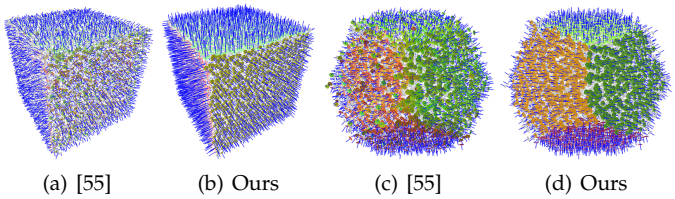


Fig. 8. Comparison with PCPNET [55].

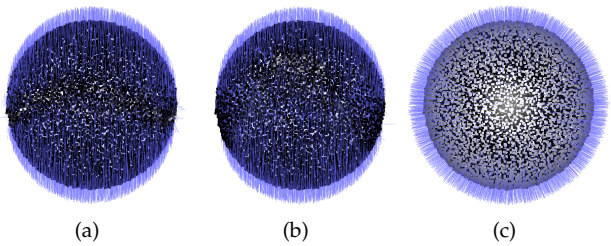


Fig. 9. Three ways of matrix construction: (a) random permutation, (b) permute matrix M with the order of x, y and z of one by one normal, (c) ours: permute matrix M with the order of x of all normals, then y of all normals and finally z. Blue lines indicate point normals.

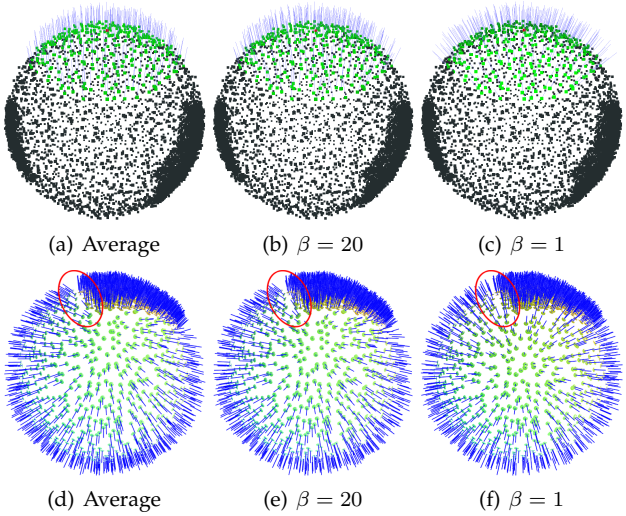


Fig. 10. First row: recovered normals of a matrix constructed by similar local isotropic structures, using the averaging method (i.e., averaging the normal vectors of similar local isotropic structures) and our method with different β. Second row: results after performing several normal estimation iterations on the same input. The mean square angular errors for (d-f) are ($\times 10^{-2}$): 3.50, 2.58 and 0.41, respectively.

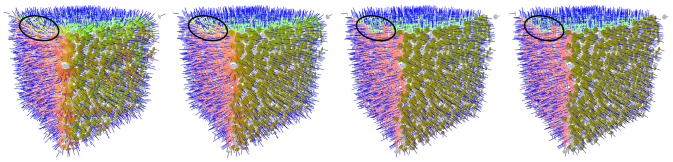
our method. [54] smooths out sharp features because the method does not consider the anisotropic and isotropic issues involved in 3D data. We tested two variants of our normal estimation method: non-local similarity based on local structures and non-local similarity based on local isotropic structures. The similarity metrics are defined by the Frobenius norm of Eq. (2), using local structures and local isotropic structures, respectively. We have the following observations from Figure 6: (1) the similar structures from both variants lie on or close to sharp features area (Figure 6(b,c)); (2) the estimated normals based on local structures are smooth everywhere and blur the sharp features (Figure 6(d)); (3) the estimated normals based on local isotropic structures are anisotropic around sharp features but sometimes do not preserve discontinuities/edges well (Figure 6(e)). We suspect this effect in (e) is because the searched similar structures of a structure owned by an edge point provide insufficient information to preserve the discontinuities (i.e., sharp edges). As such, we did not choose either of the two above variants. In comparison, our method can well preserve sharp features (Figure 7(c) and 6(f)).

Comparison with learning based methods. We compared our method with the learning based technique [54]. Figure 8 exhibits two normal estimation results for [55] and our method. Our method performs better than [55], which has difficulty in smoothing regions without features or preserving sharp features. We also show that our method outperforms another learning based method [18] (see Section 6.1).

Matrix ordering. We investigated the ordering of the constructed matrix as well. We found that the ordering of points significantly influences the minimization result (Figure 9). We suspect this is due to the neighboring information and three coordinates of each normal in different ordering matrices, which is more complicated than the regular single-channel grayscale images. Our ordering scheme does not

TABLE 1
Comparison of tensor voting and the bilateral scheme. The MSAE numbers are $\times 10^{-3}$.

Iterations	5	10	15	20	25	30
Bilateral	13.452	8.753	6.881	5.712	4.828	4.109
Tensor voting	12.016	7.855	6.182	5.133	4.340	3.694



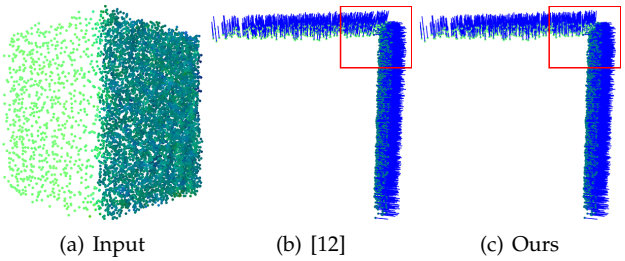
(a) (b) (c) (d)

Fig. 11. (a) PCA normals [19]. (b-d) The normal estimation after 1, 5 and 20 iterations, respectively.

require x, y or z of the involved normals to exactly fill up an integer number of columns. For example in Eq. (3), after filling x in some columns, the remaining x only fill up a part of the next column and y would fill up the remaining part of the column and fill up other columns in a similar fashion. We tested our ordering scheme with the arranged scheme (i.e., each column includes an individual type of normal coordinates only) and found similar performances (Figure 13(a)).

β and averaging. We tested the effect of β and compared our low-rank matrix minimization with the simple averaging method. We found that β is related to the number of ranks after the generalized soft-thresholding. A smaller β leads to more ranks retained, which also indicates more noise is left behind. For instance, the numbers of ranks for the sphere example in Fig. 10 with $\beta = 1.0$ and $\beta = 20.0$ are 85 and 1, respectively. We also observed that the number of ranks is related to capturing changes in the surface: a greater β captures less surface changes (e.g., Figure 10(b,e)). We found no relations between averaging and the most low-rank method (i.e., with the largest singular value retained). The most low-rank method may also recover normals of a matrix with different directions (Figure 10(a,b)). Figure 10 (d-f) show that our default β is more robust than both averaging and the most low-rank method.

Representative normal and irregular sampling. The orientation of the representative normal computed by tensor voting might be positive or negative. We use its original normal to constrain the orientation (i.e., their dot product is positive). We also compared the representative orientation computed by tensor voting with the bilateral scheme. For fair comparisons, we ensure the same weighting functions and neighbors of each point for both schemes. Table 1 shows that the tensor voting scheme outperforms the bilateral scheme. As shown in Figure 12, our normal estimation technique is more accurate and robust than the bilateral filter [12] when handling irregularly sampled data.



(a) Input (b) [12] (c) Ours

Fig. 12. Normal estimation for irregularly sampled data. (a) The points on the right side are significantly denser than the points on top. (b) and (c) are shown in a different view from (a). Blue lines indicate point normals. The mean square angular error (MSAE, in radians) of (b) and (c) are 1.87×10^{-3} and 3.90×10^{-4}, respectively.

5.2 Analysis of Point Update

Convergence. *Lemma: The proposed point update algorithm is convergent.*

Proof. The point update is convergent in the sense that the total energy $E = \sum_i \sum_{j \in S_i} |(\mathbf{p}_i - \mathbf{p}_j)\mathbf{n}_j^T|_2^2 + |(\mathbf{p}_i - \mathbf{p}_j)\mathbf{n}_i^T|_2^2$ decreases in each iteration. Assuming we use ball neighbors, we obtain

$$E = \mathbf{P}\mathbf{Q}\mathbf{P}^T, \tag{13}$$

where $\mathbf{P}$ is a $1 \times 3|i|$ vector concatenated with all point positions and $|i|$ is the number of points. $\mathbf{Q}$ is a $3|i| \times 3|i|$ matrix which consists of $|i| \times |i|$ blocks (3×3). We use $\mathbf{Q}_{i,j}$ to denote the (i,j) blocks: $\mathbf{Q}_{i,i} = 2\sum_j (\mathbf{n}_i^T \mathbf{n}_i + \mathbf{n}_j^T \mathbf{n}_j)$ and $\mathbf{Q}_{i,j} = \mathbf{Q}_{j,i} = -2(\mathbf{n}_i^T \mathbf{n}_i + \mathbf{n}_j^T \mathbf{n}_j)$.

From Eq. (12), we can compute the new positions $\mathbf{P}' = \mathbf{P}(\mathbf{I} - \mathbf{O}\mathbf{G})$. $\mathbf{I}$ is the identity matrix, $\mathbf{O} = \frac{1}{2}\mathbf{Q}$ and $\mathbf{G}$ is a diagonal block matrix with each 3×3 diagonal block as $G_{i,i} = \gamma_i I$. Based on $\mathbf{P}'$, we have $E' = \mathbf{P}'\mathbf{Q}\mathbf{P}'^T = \mathbf{P}(\mathbf{Q} - \mathbf{Q}\mathbf{G}\mathbf{O} - \mathbf{O}\mathbf{G}\mathbf{Q} + \mathbf{O}\mathbf{G}\mathbf{Q}\mathbf{G}\mathbf{O})\mathbf{P}^T$. Thus, we obtain

$$\begin{aligned} E - E' &= \mathbf{P}\mathbf{Q}\mathbf{P}^T - \mathbf{P}(\mathbf{Q} - \mathbf{Q}\mathbf{G}\mathbf{O} - \mathbf{O}\mathbf{G}\mathbf{Q} + \mathbf{O}\mathbf{G}\mathbf{Q}\mathbf{G}\mathbf{O})\mathbf{P}^T \\ &= 2\mathbf{P}\mathbf{O}\mathbf{G}(2\mathbf{G}^{-1} - \mathbf{O})\mathbf{G}\mathbf{O}\mathbf{P}^T \end{aligned} \tag{14}$$

To demonstrate the convergence of the point update, $E - E'$ should be non-negative. $\mathbf{O}$ and $\mathbf{G}$ are both symmetric positive semidefinite matrices, and we should prove $2\mathbf{G}^{-1} - \mathbf{O}$ is a symmetric positive semidefinite matrix.

Obviously, $2\mathbf{G}^{-1} - \mathbf{O}$ is a symmetric matrix. $2\mathbf{G}^{-1} - \mathbf{O}$ is a positive semidefinite matrix in the sense that its eigenvalues are non-negative. We denote λ as one of its eigenvalues, and $\mathbf{X}$ the corresponding eigenvector. Without loss of generality, we assume $|x_l|$ (i.e., $|x_l| \geq |x_k|$) is the greatest in $\mathbf{X}$. λ can be computed by

$$\begin{aligned} \lambda &= \frac{\sum_k (2g'_{l,k} - o_{l,k})x_k}{x_l} \\ &= \sum_k 2g'_{l,k}\frac{x_k}{x_l} - \sum_k o_{l,k}\frac{x_k}{x_l} \\ &= 2g'_{l,l} - \sum_k o_{l,k}\frac{x_k}{x_l} \end{aligned} \tag{15}$$

where $g'_{l,l} = \frac{1}{\gamma_l}$ and $g'_{l,k} = 0$ are the elements of $\mathbf{G}^{-1}$. We can easily demonstrate that the sum of the absolute values of each row in $\mathbf{n}_i^T \mathbf{n}_i$ or $\mathbf{n}_j^T \mathbf{n}_j$ is equal or less than $\frac{1+\sqrt{3}}{2}$. Since $g'_{l,l} = \frac{1}{\gamma_l} = 3|S_l|$ and $\sum_k o_{l,k}\frac{x_k}{x_l} \leq \sum_k |o_{l,k}|\frac{|x_k|}{|x_l|} \leq \sum_k |o_{l,k}| \leq 2(1+\sqrt{3})|S_l|$, we obtain $\lambda \geq 2g'_{l,l} - 2(1+\sqrt{3})|S_l| \geq 0$. Consequently, $2\mathbf{G}^{-1} - \mathbf{O}$ is a symmetric positive semidefinite matrix and $E - E' \geq 0$. Figure 13 (b) shows one example of the decreasing energy of Eq. (11).

Neighbor information. While the neighboring information for point updating should be recomputed in each iteration, doing so can lead to artifacts. As illustrated in Figure 15(a), our point update method enhances edges by automatically driving neighboring points to edges but also leads to obvious gaps near edges. This effect happens because the optimization will drive points to sharp edges. Points further away from these features will only have neighbors in that region, and points at sharp edges only have neighbors at sharp edges. As a result, points at sharp edges will stay at sharp edges and points further away from these features will stay further away leading to gaps near sharp edges. Furthermore, the upsampling application could fail when the number of points is low (Figure 15(b)). To alleviate this issue, we simply keep the neighboring information unchanged in all iterations, which has the side-effect of reducing the computation in each iteration. Figure 15 shows a comparison. Though we cannot guarantee that our point update method preserves the volume of the shape, we found insignificant volume changes in our experiments.

Comparison of other update methods. We compared two other point update methods with our algorithm. One method constrains the movement of a point to be along its normal direction. That is, the update equation is adjusted by removing $n_j^T n_j$ in Eq. (12). The other update method adopts local isotropic neighbors from local isotropic structures (Section 3) in Eq. (12). However, we found both methods incapable of automatically forming sharp edges: the volume shrinks significantly and points fly away from sharp edges (Figure 14 (a,b)). This effect happens because neither method takes anisotropic neighbors into account, thus leading to no constraints around sharp edges. On the contrary, our point update algorithm can automatically form the sharp edges (Figure 14(c)).

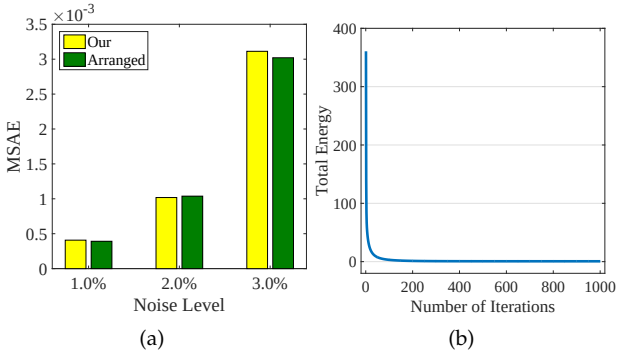


Fig. 13. (a) Performance comparison of the coordinates unseparated ordering scheme and our ordering scheme. (b) The changes of the engergy defined in Eq. (11) with increasing iterations.

TABLE 2
Normal errors (mean square angular error in radians) of two scanned models. Dod_vir is a virtual scan of a noise-free model as opposed to Figure 21, which is corrupted with synthetic noise.

Methods	[19]	[17]	[12]	[18]	Ours
Dod_vir	0.0150	0.0465	0.0054	0.0553	**0.0023**
Fig. 24	0.0118	0.1274	0.0060	0.1208	**0.0036**

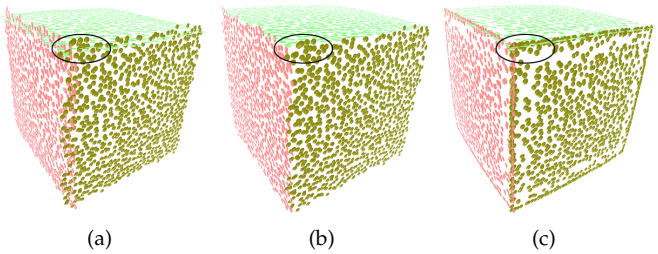


Fig. 14. (a) Point update by moving only along the current normal direction (removing $n_j^T n_j$ in Eq. (12)). (b) Point update by using local isotropic neighbors (Section 3) in Eq. (12). (c) Our method (Eq. (12)).

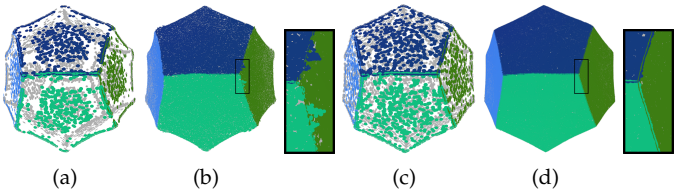


Fig. 15. Comparison of with and without updating neighboring information in each iteration. (a) Position update with updating neighboring information. (b) The upsampling of (a). (c) Position update without updating neighboring information. (d) The upsampling of (c).

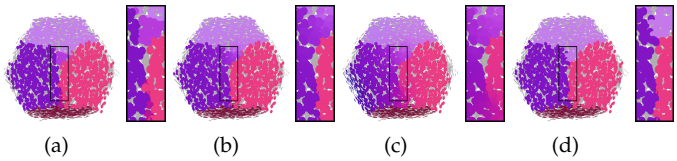


Fig. 16. (a) and (b): two overly-sharpened results (more unique colors around the upper corner) by fixing θ_{th}. (c) the smeared result (smoothly changed colors around the lower corner) by using a greater θ_{th}^{init}. (d) The result by using a smaller θ_{th}^{init}. Zoom in to clearly observe the differences.

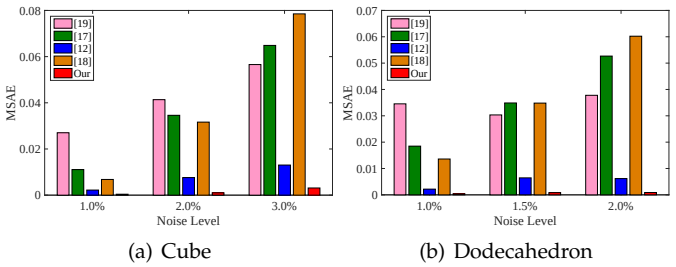


Fig. 17. Normal errors (mean square angular error in radians) of the Cube and Dodecahedron point sets corrupted with different levels of noise (proportional to the diagonal length of the bounding box).

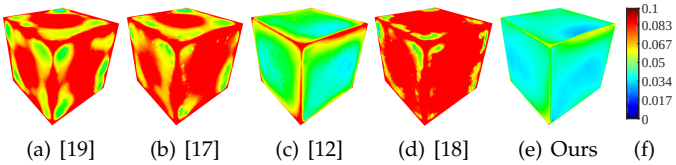
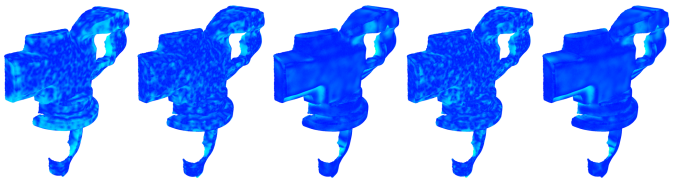


Fig. 18. Position accuracies for Fig. 20. The root mean square errors are ($\times 10^{-2}$): (a) 8.83, (b) 9.05, (c) 5.14, (d) 9.64, (e) 3.22. The rmse of the corresponding surface reconstructions are ($\times 10^{-2}$): 7.73, 6.45, 3.28, 7.71 and 2.41, respectively. (f) is the error bar for here and 19.



(a) [19] (b) [17] (c) [12] (d) [18] (e) Ours

Fig. 19. Position accuracies for Fig. 24. The root mean square errors are ($\times 10^{-3}$): (a) 8.59, (b) 6.84, (c) 6.80, (d) 6.82, (e) 6.57. The rmse of the corresponding surface reconstructions are ($\times 10^{-3}$): 8.60, 6.75, 6.68, 6.74 and 6.40, respectively.

6 Applications and Experimental Results

In this section, we demonstrate some geometry processing applications that benefit from our approach directly or indirectly including mesh denoising, point cloud filtering, point cloud upsampling, surface reconstruction, and geometric texture removal. Moreover, we also compared state of the art methods with our approach in each application. We utilize freely available source code for each comparable method or obtained implementations from the original authors.

Parameter setting. As with image denoising [3], we set a "window" size (i.e., non-local searching range) for similar structures searching, which provides a trade-off between accuracy and speed. The main parameters of our normal estimation method are the local neighborhood size k_{local}, the angle threshold θ_{th}, the non-local searching range k_{non}, and the maximum iterations for normal estimation n_{nor}. For the position update procedure, our parameters are the local neighborhood size k_{local} or 1-ring neighbors (mesh models) and the number of iterations for the position update n_{pos}.

To more accurately find similar local isotropic structures, we set one initial value and one lower bound to θ_{th}, namely θ_{th}^{init} and θ_{th}^{low}. We reduce the start value θ_{th}^{init} towards θ_{th}^{low} at a rate of 1.1^n in the n-th iteration. We show the tests of our parameters in Figure 5 and 16. In general, normal errors decrease with an increasing number of normal estimation iterations, but excessive iterations can cause normal errors to increase (Figure 5(a)). The estimated normals of models with sharp features are more accurate with the increasing local neighborhood k_{local} or non-local search range k_{non} (Figure 5(b)). Fixed θ_{th} are likely to inaccurately locate similar local isotropic structures and further generate erroneous normal estimations (Figure 16(a-b)). Larger start values of θ_{th}^{init} smear geometric features (Figure 16(c)).

Based on our parameter tests and observations, for point clouds we empirically set: $k_{local} = 60$, $k_{non} = 150$, $\theta_{th}^{init} = 30.0$, and $\theta_{th}^{low} = 15.0$ for models with sharp features, but set $\theta_{th}^{init} = 20.0$ and $\theta_{th}^{low} = 8.0$ for models with low dihedral angle features. For mesh models, we replace the local neighborhood with the 2-ring of neighboring faces. We use 2 to 10 iterations for normal estimation and 5 to 30 iterations for the position update.

To make fair comparisons, we used the same local neighborhood for all methods and tune the remaining parameters of the other methods to achieve the best visual results. Specifically, to tune one parameter, we fixed the other parameters and searched based on the suggested range and the meaning of parameters in the original papers. We observed that the other methods often take more iterations in normal smoothing than ours. The methods [17], [18] have multiple solutions, and we took the best results for comparison. For the position update, we used the same parameters for the compared normal estimation methods for each model.

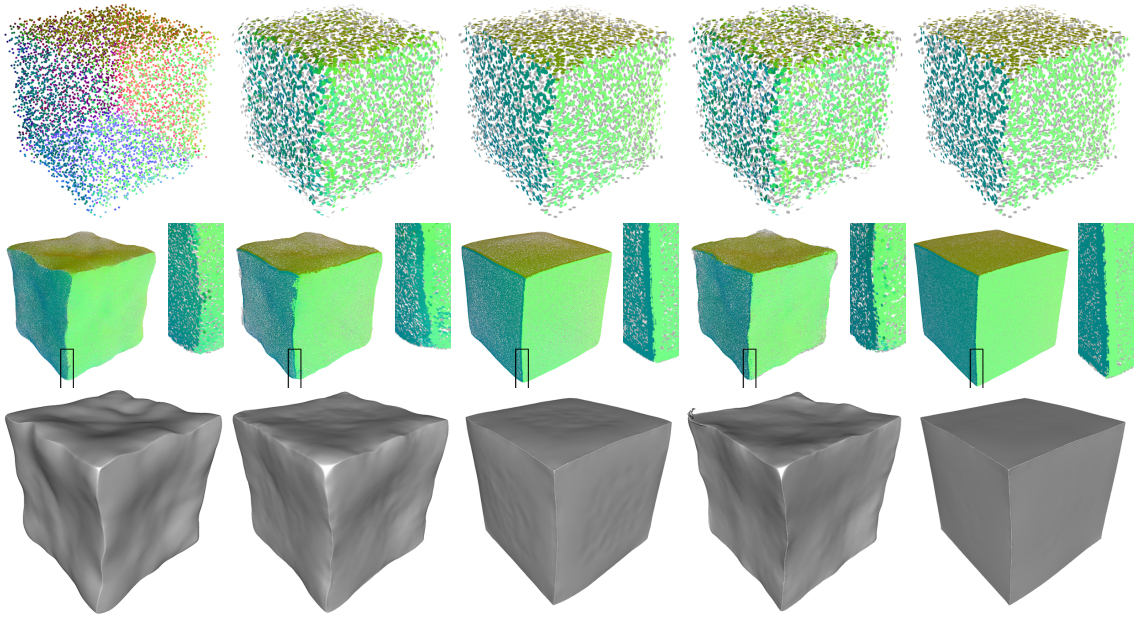
Accuracy. Since we used the pre-filter [56] for meshes with large noise, there exist few flipped normals in the results so that different methods have limited difference in normal accuracy. However, the visual differences are easy to observe. Therefore, we compared the accuracy of normals and positions over point cloud shapes. Note that state of the art methods compute normals on edges differently: the normals on edges are either sideling (e.g., [18], [19]) or perpendicular to one of the intersected tangent planes (e.g., [12] and our method). The latter is more suitable for feature-aware position update. For fair comparisons, we have two ground truth models for each point cloud: the original ground truth for [18], [19] and the other ground truth for [12] and our method. The latter ground truth is generated by adapting normals on edges to be perpendicular to one of the intersected tangent planes. The ground truth model, which has the smaller mean square angular error (MSAE) [56] among the two kinds of ground truth models, is selected as the ground truth for [17]. Figure 17 shows the normal errors of different levels of noise on the cube and dodecahedron models. We also compared our method with state of the art techniques in Table 2. The ground truth for the *Dod_vir* model (Table 2) for [18], [19] is achieved by averaging the neighboring face normals in the noise-free model. The other kind of ground truth for [12] and our method is produced by further adapting normals on edges to one of the intersected tangent planes. We compute ground truth for Figure 24 and 17 in a similar way. The normal error results demonstrate that our approach outperforms the state of the art methods. We speculate that this performance is due to the use of non-local similar structures as opposed to only local information.

In addition, we compared the position errors of different techniques, see Figure 18 and 19. The position error is measured using the average distance between points of the ground truth and their closest points of the reconstructed point set [11]. For visualization purpose, we rendered the colors of position errors on the upsampling results. The root mean square error (RMSE) of both the upsampling and reconstruction results show that our approach is more accurate than state of the art methods.

6.1 Point Cloud Filtering

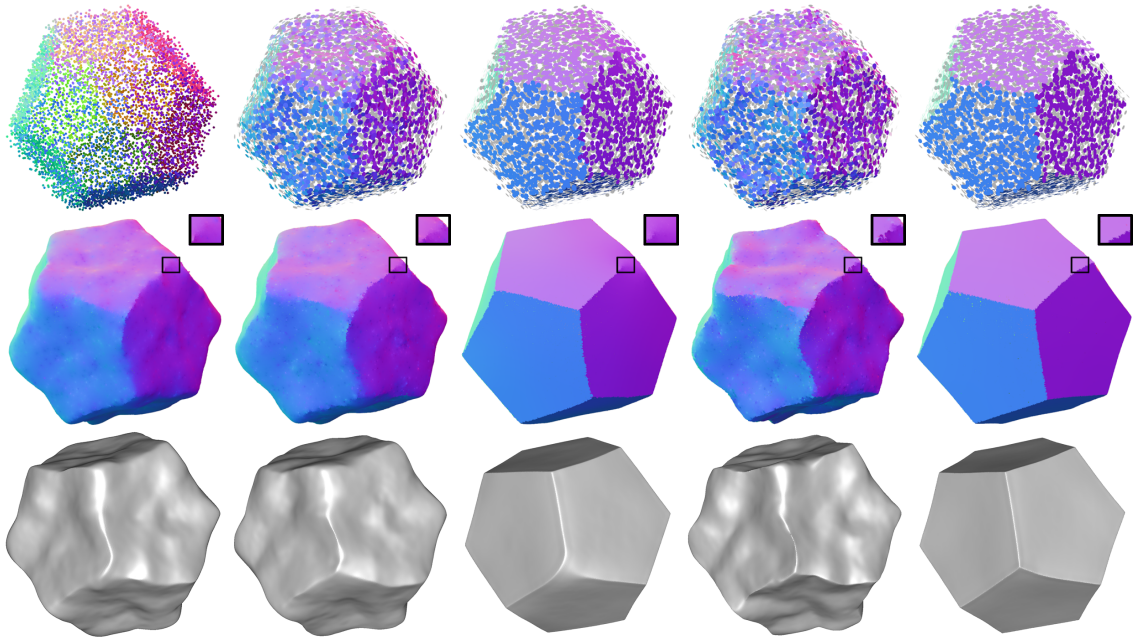
We compare our normal estimation method with several state of the art normal estimation techniques. We then perform the same number of iterations of our position update algorithm with the estimated normals of all methods.

Figure 20 and 21 show two point cloud models corrupted with heavy, synthetic noise. The results demonstrate that our method performs better than the state of the art approaches in terms of sharp feature preservation and non-feature smoothness. Figure 22, 23, 24, and 25 show the methods applied to a variety of real scanned point cloud models. Our approach outperforms other methods in terms of the quality of the estimated normals. We demonstrate our technique on point clouds with more complicated features. Figure 28 shows that our method produces slightly



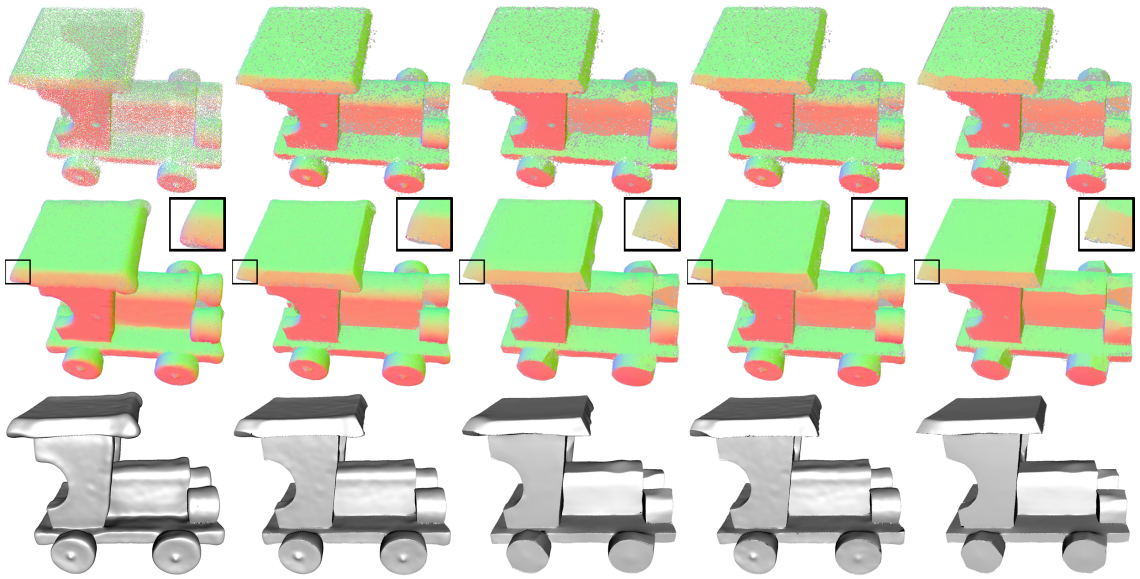
(a) [19] (b) [17] (c) [12] (d) [18] (e) Ours

Fig. 20. The first row: normal results of the Cube point cloud (synthetic noise: 3.0% of the diagonal length of the bounding box). The second row: upsampling results of the filtered results by updating position with the normals in the first row. The third row: the corresponding surface reconstruction results.



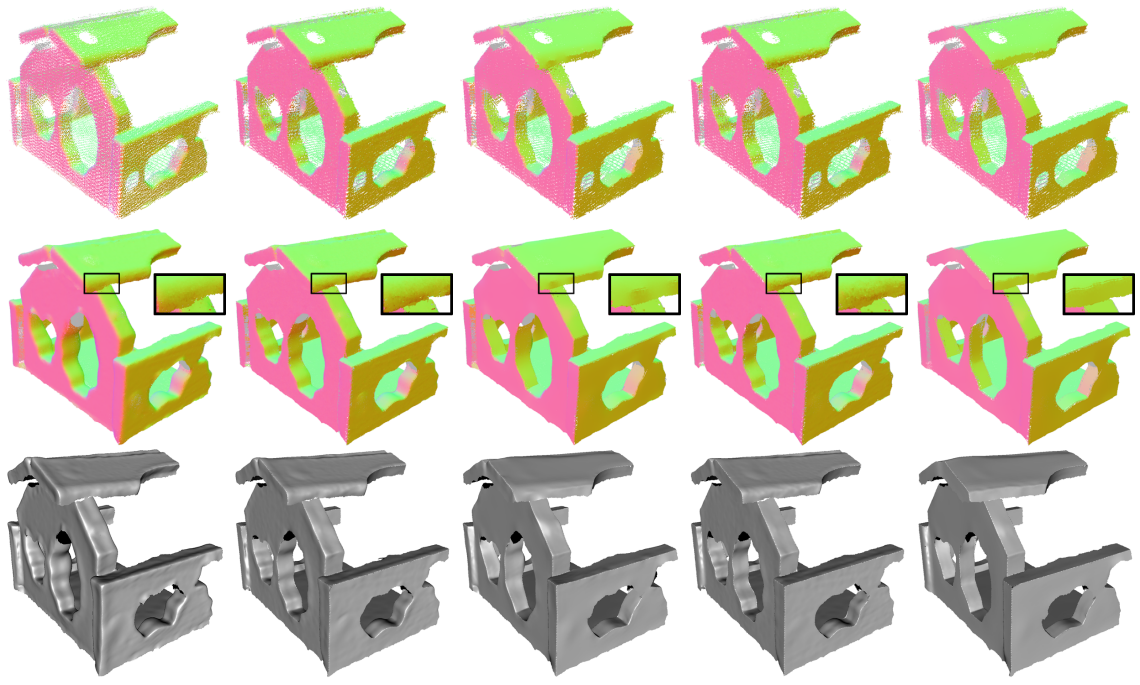
(a) [19] (b) [17] (c) [12] (d) [18] (e) Ours

Fig. 21. The first row: normal results of the Dodecahedron point cloud (synthetic noise: 2.0% of the diagonal length of the bounding box). The second row: upsampling results of the filtered results by updating position with the normals in the first row. The third row: the corresponding surface reconstruction results.



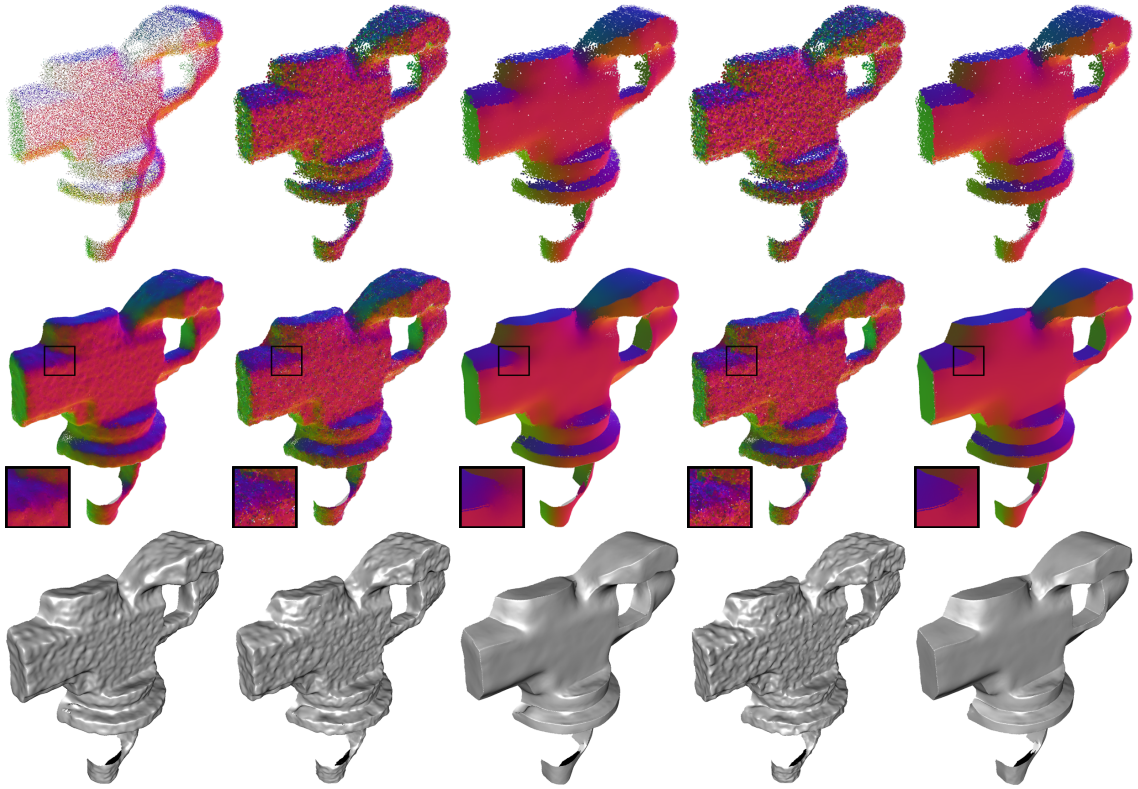
(a) [19] (b) [17] (c) [12] (d) [18] (e) Ours

Fig. 22. The first row: normal results of the scanned Car point cloud. The second row: upsampling results of the filtered results by updating position with the normals in the first row. The third row: the corresponding surface reconstruction. Comparing with other methods, [12] and our method are better in sharp edges preservation and hereby generate more sharpened results.



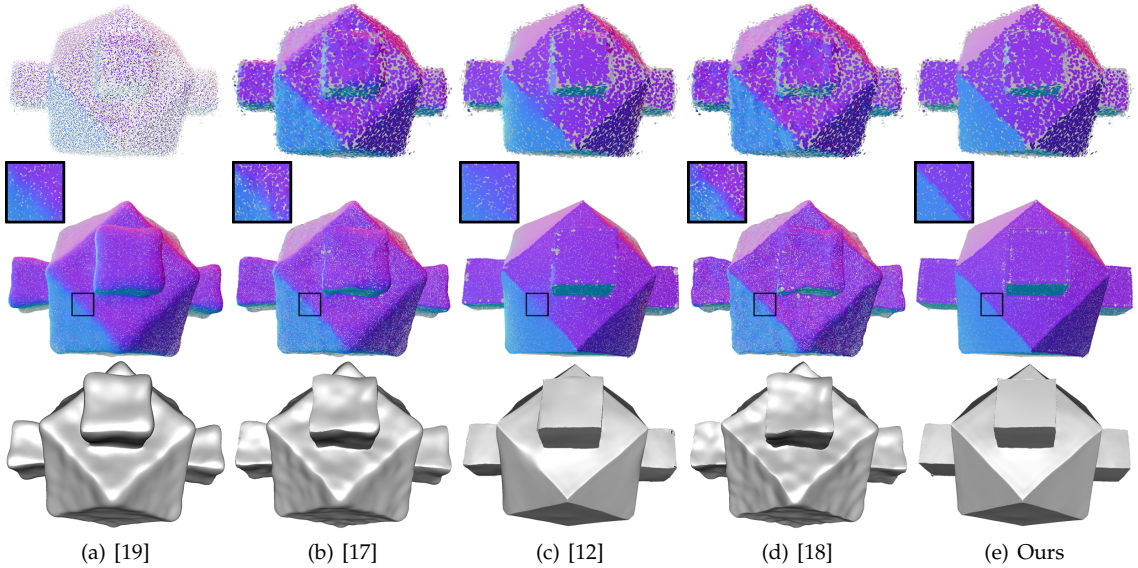
(a) [19] (b) [17] (c) [12] (d) [18] (e) Ours

Fig. 23. The first row: normal results of the scanned House point cloud. The second row: upsampling results of the filtered results by updating position with the normals in the first row. The third row: the corresponding surface reconstruction results.



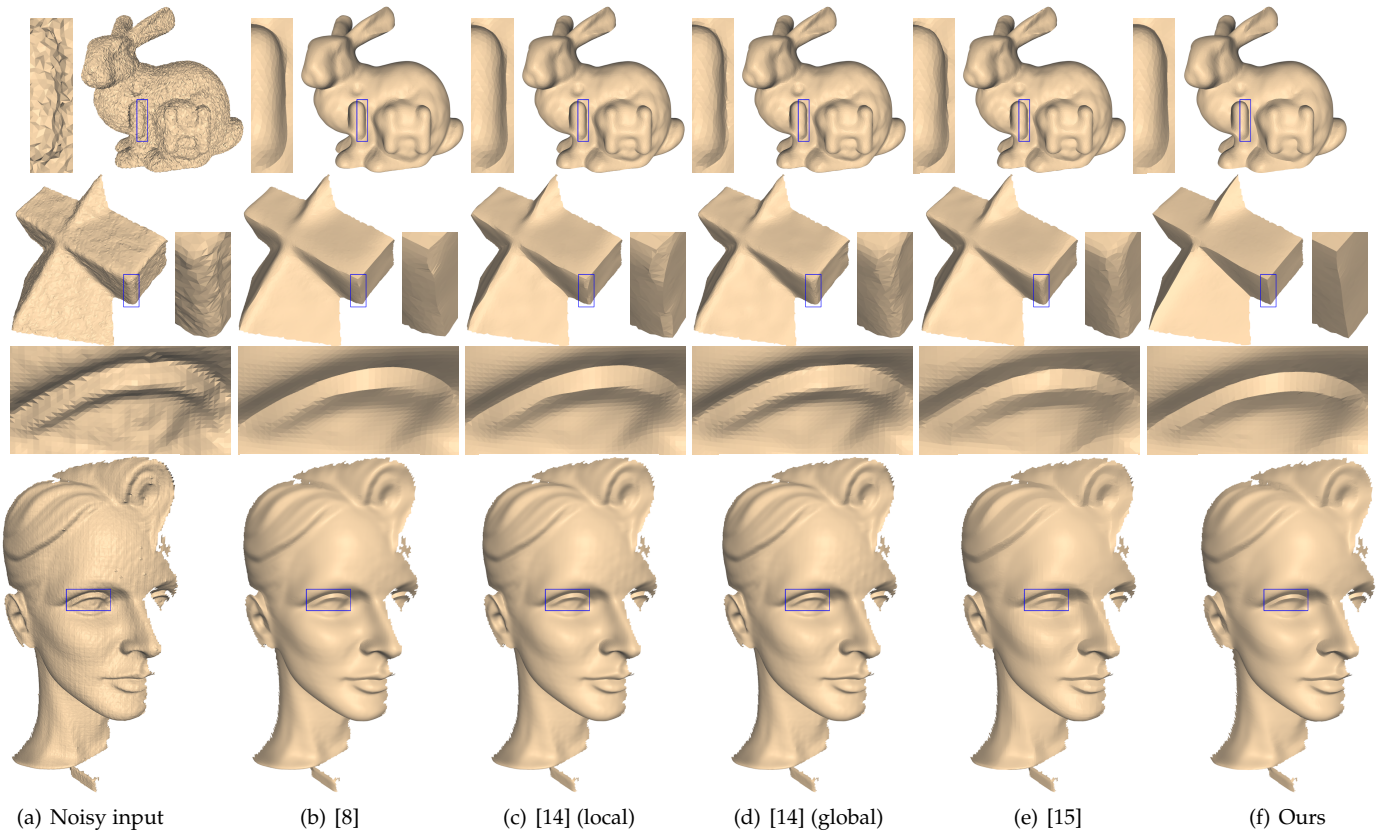
(a) [19] (b) [17] (c) [12] (d) [18] (e) Ours

Fig. 24. The first row: normal results of the scanned Iron point cloud. The second row: upsampling results of the filtered results by updating position with the normals in the first row. The third row: the corresponding surface reconstruction results.



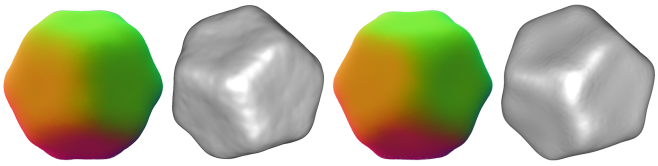
(a) [19] (b) [17] (c) [12] (d) [18] (e) Ours

Fig. 25. The first row: normal results of the scanned Toy point cloud. The second row: upsampling results of the filtered results by updating position with the normals in the first row. The third row: the corresponding surface reconstruction results.



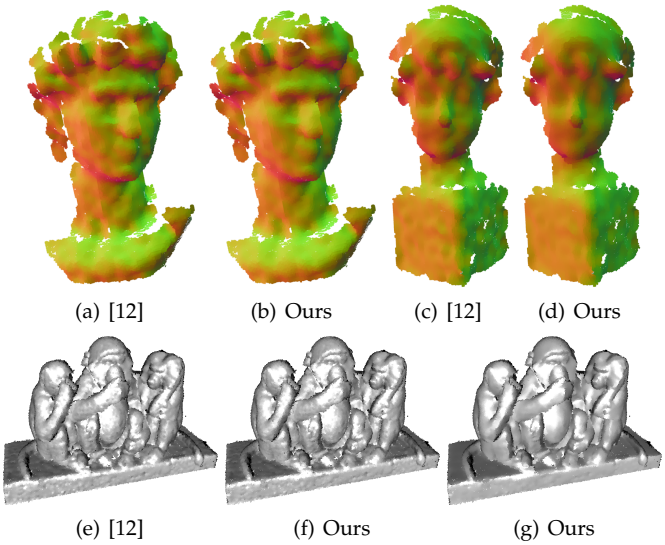
(a) Noisy input (b) [8] (c) [14] (local) (d) [14] (global) (e) [15] (f) Ours

Fig. 26. Denoised results of the Bunny (synthetic noise: 0.2 of the average edge length), the scanned Pyramid and Wilhelm.



(a) [24] (b) RIMLS over (a) (c) [57] (d) RIMLS over (c)

Fig. 27. Upsampling and reconstruction results over [24], [57]. The input is the same as Figure 21.



(a) [12] (b) Ours (c) [12] (d) Ours

(e) [12] (f) Ours (g) Ours

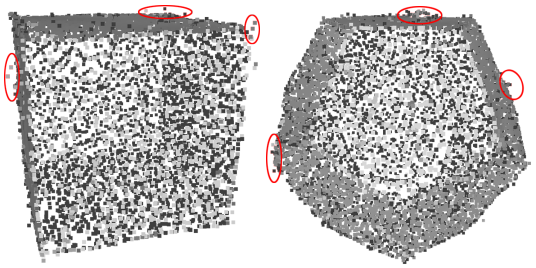
Fig. 28. Normal estimation results on David (a,b), a female statue (c,d) and monkeys (e,f,g). The mean square angular errors of (a-g) are respectively ($\times 10^{-2}$): 10.684, 10.636, 9.534, 9.423, 5.004, 4.853 and 4.893. (b,d,f) used smaller k_{non} and k_{local}, and (g) used the default k_{non} and k_{local}.

lower normal errors than [12]. Figure 28 (f) and (g) show our method with different parameters, which leads to a less/more sharpened version of the input. We also show some results using [24], [57], which do not preserve sharp features (Figure 27). Besides, we show some filtering results by [58] which can preserve sharp features to some extent, with introducing obvious outliers on surfaces (Figure 29). Our method can even successfully handle a larger noise (Figure 20 and 21), which we found is difficult for [58].

6.2 Point Cloud Upsampling

As described in Section 6.1, the point cloud filtering also consists of a two-step procedure: normal estimation and point update. However, unlike mesh shapes, point cloud models often need to be resampled to enhance point density after filtering operations have been applied.

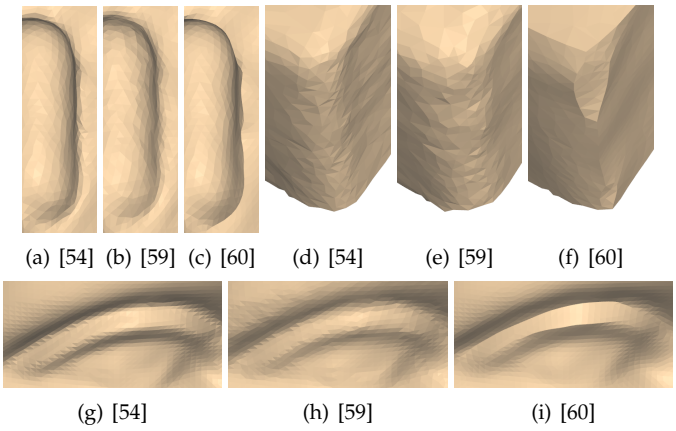
We apply the edge-aware point set resampling technique [12] to all the results after point cloud filtering and contrast the different upsampling results. For fair comparisons, we upsample the filtered point clouds of each model to reach a similar number of points. Figure 20 to 25 display various upsampling results on state of the art normal estimation methods and different point cloud models. The figures show that the upsampling results on our filtered point clouds are



(a) [58] (b) [58]

Fig. 29. Filtering results by [58]. The input noise is 0.15% for (a) and 0.1% for (b). Red circles indicate outliers.

substantially better than those filtered by other methods in preserving sharp features. Bilateral normal smoothing [12] usually produces good results, but this method sometimes blur edges with low dihedral angles.



(a) [54] (b) [59] (c) [60] (d) [54] (e) [59] (f) [60]

(g) [54] (h) [59] (i) [60]

Fig. 30. Mesh denoising results of [54], [59], [60]. Only close-up views are shown to highlight the differences.

6.3 Surface Reconstruction

One common application for point cloud models is to reconstruct surfaces from the upsampled point clouds in Section 6.2 before use in other applications. Here, we select the edge-aware surface reconstruction technique–RIMLS [13]. For fair comparisons, we use the same parameters for all the upsampled point clouds of each model.

Figure 20 to 25 show a variety of surface reconstruction results on different point cloud models. The comparison results demonstrate that the RIMLS technique over our method produces the best surface reconstruction results, in terms of sharp feature preservation.

6.4 Mesh Denoising

Many state of the art mesh denoising methods involve a two-step procedure which first estimates normals and then updates vertex positions. We selected several of these methods [8], [14], [15] for comparisons in Figure 26. Note that [14] provides both a local and global solution, and we provide comparisons for both. We also compared our method with other mesh denoising techniques [54], [59], [60]. Consistent with Figure 26, the corresponding blown-up windows of these three methods are shown in Figure 30.

When the noise level is high, many of these methods produce flipped face normals. For the Bunny model (Figure 26), which involves frequent flipped triangles, we utilize the technique in [56] to estimate a starting mesh from the original noisy mesh input for all involved methods. The comparison results show that our method outperforms the selected state of the art mesh denoising methods in terms of sharp feature preservation. Similar to the above analysis, this is because that other methods are mostly local techniques while our method takes into account the information of similar structures (i.e., more useful information). Specifically, [8], [14], [15], [59], [60] are local methods ([15] and the global mode of [14] are still based on local information). [54] does not take sharp features information into account and thus cannot preserve sharp features well.

6.5 Geometric Texture Removal

We also successfully applied our method to geometric detexturing, the task of which is to remove features of different scales [16]. Our normal estimation algorithm is feature-aware in the above applications because each matrix consists only of similar local isotropic structures. On the other hand, by larger values of θ_{th}, the constructed matrix can include local anisotropic structures and the low rank matrix approximation result becomes smoother, thus smoothing anisotropic surfaces to isotropic surfaces.

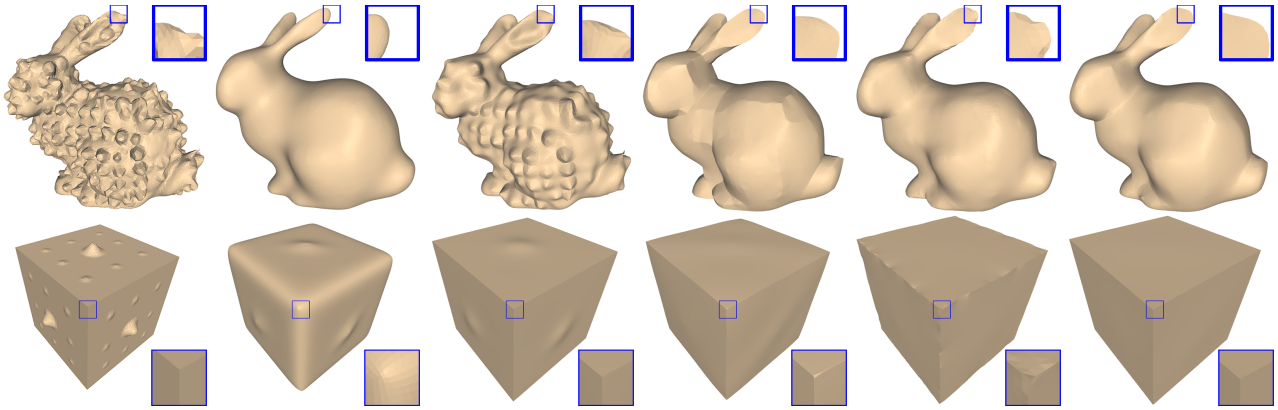
Figure 31 shows comparisons of different methods that demonstrate that our method outperforms other approaches. Note that [16] is specifically designed for geometric texture removal. However, that method cannot preserve sharp edges well. Figure 32 shows the results of removing different scales of geometric features on a mesh. We produced Figure 32 (d) by applying the pre-filtering technique [56] in advance, since the vertex update algorithm [14] could generate frequent flipped triangles when dealing with such large and steep geometric features. As an alternative, our normal estimation method can be combined with the vertex update in [16] to handle such challenging mesh models. Figure 33 shows the geometric texture removal on two different point clouds, which are particularly challenging due to a lack of topology.

TABLE 3
Timing statistics for different normal estimation techniques over point clouds (in seconds).

Methods	[19]	[17]	[12]	[18]	Ours SVD	Ours RSVD
Fig. 20 #6146	0.57	141.5	0.48	8	95.6	65.1
Fig. 24 #100k	18.7	2204	17.2	115	3147	2458
Fig. 23 #127k	10.8	3769	12.5	141	3874	2856

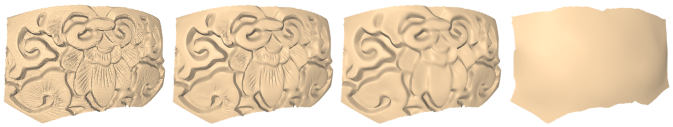
6.6 Timings

Table 3 summarizes the timings of different normal estimation methods on several point clouds. While our method



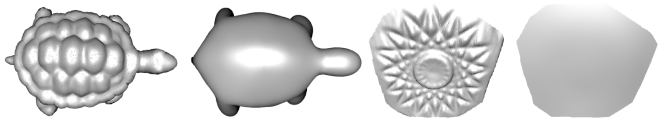
(a) Input (b) Laplacian (c) [14] (local) (d) [61] (e) [16] (f) Ours

Fig. 31. Geometric texture removal results of the Bunny and Cube. Please refer to the zoomed rectangular windows.



(a) Input (b) Small texture removal (c) Medium texture removal (d) Large texture removal

Fig. 32. Different scales of geometric texture removal results of the Circular model.



(a) Input (b) Ours (c) Input (d) Ours

Fig. 33. Geometric texture removal results of the Turtle point cloud and embossed point cloud. We render point set surfaces of each point cloud for visualization.

produces high quality output, the algorithm takes a long time to run due to the svd operation for each normal estimation. Therefore, our method is more suitable for offline geometry processing. However, it is possible to accelerate our method using specific svd decomposition algorithms, such as the randomized svd (RSVD) decomposition algorithm [62] as shown in Table 3. In addition, many parts of the algorithm could benefit from parallelization.

7 Conclusion

In this paper, we have presented an approach consisting of two steps: normal estimation and position update. Our method can handle both mesh shapes and point cloud models. We also show various geometry processing applications that benefit from our approach directly or indirectly. The extensive experiments demonstrate that our method performs substantially better than state of the art techniques, in terms of both visual quality and accuracy.

While our method works well, speed is an issue if online processing speeds are required. In addition, though we mitigate issues associated with the point distribution in the position update procedure (i.e., gaps near edges), the point distribution could still be improved. One way to do so is to re-distribute points after our position update through a "repulsion force" from each point to its neighbors. We could potentially accomplish this effect by adding this repulsion force directly to Eq. (11).

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Jan 1998, pp. 839–846.

[2] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, June 2013.

[3] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 2862–2869. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.366

[4] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *International Journal of Computer Vision*, vol. 121, no. 2, pp. 183–208, Jan 2017. [Online]. Available: https://doi.org/10.1007/s11263-016-0930-5

[5] S. Wu, P. Bertholet, H. Huang, D. Cohen-Or, M. Gong, and M. Zwicker, "Structure-aware data consolidation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 10, pp. 2529–2537, Oct 2018.

[6] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3d point set upsampling," 2018.

[7] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Ec-net: an edge-aware point set consolidation network," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[8] X. Sun, P. Rosin, R. Martin, and F. Langbein, "Fast and effective feature-preserving mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 925–938, Sept 2007.

[9] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "L1-sparse reconstruction of sharp point set surfaces," *ACM Trans. Graph.*, vol. 29, no. 5, pp. 135:1–135:12, Nov. 2010. [Online]. Available: http://doi.acm.org/10.1145/1857907.1857911

[10] Y. Sun, S. Schaefer, and W. Wang, "Denoising point sets via l_0 minimization," *Computer Aided Geometric Design*, vol. 35-36, pp. 2 – 15, 2015, geometric Modeling and Processing 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167839615000345

[11] X. Lu, S. Wu, H. Chen, S. K. Yeung, W. Chen, and M. Zwicker, "Gpf: Gmm-inspired feature-preserving point set filtering," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[12] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 9:1–9:12, Feb. 2013. [Online]. Available: http://doi.acm.org/10.1145/2421636.2421645

[13] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *Computer Graphics Forum*, vol. 28, no. 2, pp. 493–501, 2009. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01388.x

[14] Y. Zheng, H. Fu, O.-C. Au, and C.-L. Tai, "Bilateral normal filtering for mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 10, pp. 1521–1530, Oct 2011.

[15] H. Zhang, C. Wu, J. Zhang, and J. Deng, "Variational mesh denoising using total variation and piecewise constant function space," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 21, no. 7, pp. 873–886, July 2015.

[16] P.-S. Wang, X.-M. Fu, Y. Liu, X. Tong, S.-L. Liu, and B. Guo, "Rolling guidance normal filter for geometric processing," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 173:1–173:9, Oct. 2015. [Online]. Available: http://doi.acm.org/10.1145/2816795.2818068

[17] A. Boulch and R. Marlet, "Fast and robust normal estimation for point clouds with sharp features," *Comput. Graph. Forum*, vol. 31, no. 5, pp. 1765–1774, Aug. 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2012.03181.x

[18] A. Boulch and R. Marlet, "Deep learning for robust normal estimation in unstructured point clouds," *Computer Graphics Forum*, vol. 35, no. 5, pp. 281–290, 2016. [Online]. Available: http://dx.doi.org/10.1111/cgf.12983

[19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 71–78, Jul. 1992. [Online]. Available: http://doi.acm.org/10.1145/142920.134011

[20] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Point set surfaces," in *Proceedings of the Conference on Visualization '01*, ser. VIS '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 21–28. [Online]. Available: http://dl.acm.org/citation.cfm?id=601671.601673

[21] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proceedings of the Conference on Visualization '02*, ser. VIS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 163–170. [Online]. Available: http://dl.acm.org/citation.cfm?id=602099.602123

[22] N. J. Mitra and A. Nguyen, "Estimating surface normals in noisy point cloud data," in *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, ser. SCG '03. New York, NY, USA: ACM, 2003, pp. 322–328. [Online]. Available: http://doi.acm.org/10.1145/777792.777840

[23] C. Lange and K. Polthier, "Anisotropic smoothing of point sets," *Computer Aided Geometric Design*, vol. 22, no. 7, pp. 680 – 692, 2005, geometric Modelling and Differential Geometry. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167839605000750

[24] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 176:1–176:7, Dec. 2009. [Online]. Available: http://doi.acm.org/10.1145/1618452.1618522

[25] T. K. Dey and S. Goswami, "Provable surface reconstruction from noisy samples," in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, ser. SCG '04. New York, NY, USA: ACM, 2004, pp. 330–339. [Online]. Available: http://doi.acm.org/10.1145/997817.997867

[26] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun, "Voronoi-based variational reconstruction of unoriented point sets," in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, ser. SGP '07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 39–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=1281991.1281997

[27] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, and S. Jin, "Robust normal estimation for point clouds with sharp features," *Computers & Graphics*, vol. 34, no. 2, pp. 94 – 106, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S009784931000021X

[28] J. Zhang, J. Cao, X. Liu, J. Wang, J. Liu, and X. Shi, "Point cloud normal estimation via low-rank subspace clustering," *Computers & Graphics*, vol. 37, no. 6, pp. 697 – 706, 2013, shape Modeling International (SMI) Conference 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849313000824

[29] X. Liu, J. Zhang, J. Cao, B. Li, and L. Liu, "Quality point cloud normal estimation by guided least squares representation," *Computers & Graphics*, vol. 51, no. Supplement C, pp. 106 – 116, 2015, international Conference Shape Modeling International. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849315000710

[30] J. Zhang, J. Cao, X. Liu, C. He, B. Li, and L. Liu, "Multi-normal estimation via pair consistency voting," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018.

[31] E. Mattei and A. Castrodad, "Point cloud denoising via moving rpca," *Computer Graphics Forum*, vol. 36, no. 8, pp. 123–137, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13068

[32] K.-W. Lee and W.-P. Wang, "Feature-preserving mesh denoising via bilateral normal filtering," in *Proc. of Int'l Conf. on Computer Aided Design and Computer Graphics 2005*, Dec 2005.

[33] C. C. L. Wang, "Bilateral recovering of sharp edges on feature-insensitive sampled meshes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 4, pp. 629–639, July 2006.

[34] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 943–949, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882367

[35] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 950–953, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882368

[36] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh smoothing via mean and median filtering applied to face normals," in *Geometric Modeling and Processing, 2002. Proceedings*, 2002, pp. 124–131.

[37] H. Yagou, Y. Ohtake, and A. Belyaev, "Mesh denoising via iterative alpha-trimming and nonlinear diffusion of normals with automatic thresholding," in *Computer Graphics International, 2003. Proceedings*, July 2003, pp. 28–33.

[38] Y. Shen and K. Barner, "Fuzzy vector median-based surface smoothing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 3, pp. 252–265, May 2004.

[39] X. Sun, P. L. Rosin, R. R. Martin, and F. C. Langbein, "Random walks for feature-preserving mesh denoising," *Computer Aided Geometric Design*, vol. 25, no. 7, pp. 437 – 456, 2008, solid and Physical Modeling Selected papers from the Solid and Physical Modeling and Applications Symposium 2007 (SPM 2007) Solid and Physical Modeling and Applications Symposium 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167839608000307

[40] J. Solomon, K. Crane, A. Butscher, and C. Wojtan, "A general framework for bilateral and mean shift filtering," *CoRR*, vol. abs/1405.4734, 2014. [Online]. Available: http://arxiv.org/abs/1405.4734

[41] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, and L. Liu, "Guided mesh normal filtering," *Comput. Graph. Forum*, vol. 34, no. 7, pp. 23–34, Oct. 2015. [Online]. Available: http://dx.doi.org/10.1111/cgf.12742

[42] X. Lu, W. Chen, and S. Schaefer, "Robust mesh denoising via vertex pre-filtering and l1-median normal filtering," *Computer Aided Geometric Design*, vol. 54, no. Supplement C, pp. 49 – 60, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167839617300638

[43] S. K. Yadav, U. Reitebuch, and K. Polthier, "Mesh denoising based on normal voting tensor and binary optimization," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.

[44] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 232:1–232:12, Nov. 2016. [Online]. Available: http://doi.acm.org/10.1145/2980179.2980232

[45] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or, and B. Chen, "Non-local scan consolidation for 3d urban scenes," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 94:1–94:9, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778831

[46] J. Digne, "Similarity based filtering of point clouds," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012, pp. 73–79.

[47] J. Digne, S. Valette, and R. Chaine, "Sparse geometric representation through local shape probing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 7, pp. 2238–2250, July 2018.

[48] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, p. 717, Apr 2009. [Online]. Available: https://doi.org/10.1007/s10208-009-9045-5

[49] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. on Optimization*, vol. 20, no. 4, pp. 1956–1982, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1137/080738970
[50] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 2080–2088.
[51] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 663–670. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104407
[52] Z. Zhang, A. Ganesh, X. Liang, and Y. Ma, "Tilt: Transform invariant low-rank textures," *International Journal of Computer Vision*, vol. 99, no. 1, pp. 1–24, Aug 2012. [Online]. Available: https://doi.org/10.1007/s11263-012-0515-x
[53] T. P. Wu, S. K. Yeung, J. Jia, C. K. Tang, and G. Medioni, "A closed-form solution to tensor voting: Theory and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1482–1495, Aug 2012.
[54] X. Li, L. Zhu, C.-W. Fu, and P.-A. Heng, "Non-local low-rank normal filtering for mesh denoising," *Computer Graphics Forum*, vol. 37, no. 7, pp. 155–166, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13556
[55] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "Pcpnet learning local shape properties from raw point clouds," *Computer Graphics Forum*, vol. 37, no. 2, pp. 75–85, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13343
[56] X. Lu, Z. Deng, and W. Chen, "A robust scheme for feature-preserving mesh denoising," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 3, pp. 1181–1194, 2016.
[57] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer, "Continuous projection for fast l1 reconstruction," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 47:1–47:13, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601172
[58] S. K. Yadav, U. Reitebuch, M. Skrodzki, E. Zimmermann, and K. Polthier, "Constraint-based point set denoising using normal voting tensor and restricted quadratic error metrics," *Computers & Graphics*, vol. 74, pp. 234 – 243, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849318300797
[59] W. Pan, X. Lu, Y. Gong, W. Tang, J. Liu, Y. He, and G. Qiu, "HLO: half-kernel laplacian operator for surface smoothing," *Computer Aided Design*, 2019.
[60] S. K. Yadav, U. Reitebuch, and K. Polthier, "Robust and high fidelity mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 6, pp. 2304–2310, June 2019.
[61] L. He and S. Schaefer, "Mesh denoising via l0 minimization," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 64:1–64:8, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461965
[62] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011. [Online]. Available: https://doi.org/10.1137/090771806

Xuequan Lu is a Lecturer (Assistant Professor) at Deakin University, Australia. He spent more than two years as a Research Fellow in Singapore. Prior to that, he earned his Ph.D at Zhejiang University (China) in June 2016. His research interests mainly fall into the category of visual computing, for example, geometry modeling, processing and analysis, animation/simulation, 2D data processing and analysis. More information can be found at http://www.xuequanlu.com.

Scott Schaefer is a Professor of Computer Science at Texas AM University. He received a bachelors degree in Computer Science/Mathematics from Trinity University in 2000 and an M.S. and PhD. in Computer Science from Rice University in 2003 and 2006 respectively. His research interests include graphics, geometry processing, curve and surface representations, and barycentric coordinates. Scott received the Gnter Enderle Award in 2011 and an NSF CAREER Award in 2012.

Jun Luo received his BS and MS degrees in Electrical Engineering from Tsinghua University, China, and the Ph.D. degree in Computer Science from EPFL (Swiss Federal Institute of Technology in Lausanne), Lausanne, Switzerland. From 2006 to 2008, he has worked as a postdoctoral research fellow in the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. In 2008, he joined the faculty of the School Of Computer Science and Engineering, Nanyang Technological University in Singapore, where he is currently an Associate Professor. His research interests include mobile and pervasive computing, wireless networking, applied operations research, as well as network security.

Lizhuang Ma received the Ph.D. degree from the Zhejiang University, Hangzhou, China. He was the recipient of the national science fund for distinguished young scholars from NSFC. He is currently a Distinguished Professor and the Head of the Digital Media Computer Vision Laboratory, Shanghai Jiao Tong University, Shanghai, China. His research interests include digital media technology, vision, graphics, etc,.

Ying He is currently an associate professor at School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received the BS and MS degrees in electrical engineering from Tsinghua University, China, and the PhD degree in computer science from Stony Brook University, USA. His research interests fall into the general areas of visual computing and he is particularly interested in the problems which require geometric analysis and computation.