

# 3D Point Cloud Denoising Using Graph Laplacian Regularization of a Low Dimensional Manifold Model

Jin Zeng, *Member, IEEE*, Gene Cheung, *Senior Member, IEEE*, Michael Ng, *Senior Member, IEEE*,  
Jiahao Pang, *Member, IEEE*, and Cheng Yang, *Member, IEEE*

**Abstract**—3D point cloud—a new signal representation of volumetric objects—is a discrete collection of triples marking exterior object surface locations in 3D space. Conventional imperfect acquisition processes of 3D point cloud—e.g., stereo-matching from multiple viewpoint images or depth data acquired directly from active light sensors—imply non-negligible noise in the data. In this paper, we extend a previously proposed low-dimensional manifold model for the image patches to surface patches in the point cloud, and seek self-similar patches to denoise them simultaneously using the patch manifold prior. Due to discrete observations of the patches on the manifold, we approximate the manifold dimension computation defined in the continuous domain with a patch-based graph Laplacian regularizer, and propose a new discrete patch distance measure to quantify the similarity between two same-sized surface patches for graph construction that is robust to noise. We show that our graph Laplacian regularizer leads to speedy implementation and has desirable numerical stability properties given its natural graph spectral interpretation. Extensive simulation results show that our proposed denoising scheme outperforms state-of-the-art methods in objective metrics and better preserves visually salient structural features like edges.

**Index Terms**—graph signal processing, point cloud denoising, low-dimensional manifold

## I. INTRODUCTION

The three-dimensional (3D) point cloud has become an important and popular signal representation of volumetric objects in 3D space [1]–[3]. 3D point cloud can be acquired directly using low-cost depth sensors like Microsoft Kinect or high-resolution 3D scanners like LiDAR. Moreover, multi-view stereo-matching techniques have been extensively studied in recent years to recover a 3D model from images or videos, where the typical output format is the point cloud [4]. However, in either case, the output point cloud is inherently noisy, which has led to numerous approaches for point cloud denoising [5]–[8].

Moving least squares (MLS)-based [9], [10] and locally optimal projection (LOP)-based methods [11], [12] are two

major categories of point cloud denoising approaches, but are often criticized for over-smoothing [7], [8] due to the use of local operators. Sparsity-based methods, based on the local planarity assumption, are optimized towards a sparse representation of certain geometric features such as surface normals [7], [13] and point deviations from local reference plane [6]. They were reported to provide the state-of-the-art performance [14]. However at high noise levels, the inaccurate estimation for normal or the local plane can lead to over-smoothing or over-sharpening [6], [7].

Non-local methods generalize the non-local means [15] and BM3D [16] image denoising algorithms to point cloud denoising, and are shown to better preserve fine shape features under high level of noise. The approaches in [17], [18] extend the non-local means denoising approach to point clouds and adaptively filter the points in an edge preserving manner. [5] is inspired by BM3D and exploits the inherent self-similarity between surface patches to preserve structural details, but the computational complexity is too high to be practical. A more recent method in [19] also utilizes the patch self-similarity and denoises the local patches based on dictionary learning.

Utilizing an assumed self-similarity characteristic in images has long been a popular strategy in image processing [15], [16]. Extending on these earlier works, a more recent work [20] proposed the *low-dimensional manifold model* (LDMM) for image processing, assuming that similar image patches are samples of a low-dimensional manifold in high-dimensional space. The assumption is verified in various applications in image processing and computer vision [21], [22]. In LDMM, the manifold dimension is used for regularization to recover the image, achieving state-of-the-art results in various inverse imaging applications, *e.g.*, denoising, inpainting, superresolution, *etc.*

Inspired by the LDMM work in [20], we exploit self-similarity of the surface patches by assuming that the surface patches in the point cloud lie on a manifold of low dimension. However, the extension of LDMM from images to point clouds is non-trivial. First, the computation of manifold dimension requires a well-defined coordinate function in [20], *i.e.*, the extrinsic coordinates of points on the manifold, which is straightforward for image patches but not for surface patches due to the irregular structure of point clouds. Moreover, the point integral method (PIM) for solving the dimension optimization in [20] is of high complexity. In the outer loop, the manifold and the image are iteratively updated,

Jin Zeng is with Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Email: jzengab@connect.ust.hk

Gene Cheung and Cheng Yang are with the Department of Electrical Engineering & Computer Science, York University, Toronto, Canada. Email: genec@yorku.ca; cyang@eecs.yorku.ca

Michael Ng is with the Centre for Mathematical Imaging and Vision, Department of Mathematics, Hong Kong Baptist University, Hong Kong. Email: mng@math.hkbu.edu.hk

Jiahao Pang is with SenseTime Research, Hong Kong. Email: pangjiahao@sensetime.com

while in the inner loop, the coordinate function and pixel values are updated until convergence. Since the linear systems for updating coordinate function are asymmetric due to the constraints enforced by PIM, a large number of iterations is required to reach convergence, leading to high computational cost [23].

To address the two issues above, we approximate the patch-manifold dimension defined in continuous domain with a discrete patch-based graph Laplacian regularizer (GLR). Specifically, the main contributions of our work are as follows:

- 1) By adopting the LDMM, we exploit the surface self-similarity characteristic and simultaneously denoise similar patches to better preserve sharp features;
- 2) By approximating the computation of the manifold dimension with GLR, we avoid explicitly defining the manifold coordinate functions and enable the LDMM to extend to the point cloud setting;
- 3) By using GLR, the implementation is accelerated with a reduced number of iterations thanks to the symmetric structure of the graph Laplacian matrix;
- 4) Our GLR is shown to provide a graph spectral interpretation and is guaranteed numerical stability via eigen-analysis in the graph spectral domain [24];
- 5) An efficient similarity measure for discrete  $k$ -pixel patch pairs is designed for graph construction that is robust to noise.

Extensive simulation results show that our proposed method outperforms the state-of-the-art methods in objective metrics and better preserves visually salient features like edges.

The rest of the paper is organized as follows. Section II overviews some existing works. Section III defines the patch manifold associated with the 3D point cloud. Section IV formulates the denoising problem by describing how the manifold dimension is computed and approximated with the graph Laplacian regularizer. The algorithm implementation is discussed in Section V with graph spectral analysis to interpret the algorithm and a numerical stability analysis. Finally, Section VI and Section VII presents experimental results and concludes the paper respectively.

## II. RELATED WORK

Previous point cloud denoising works can be classified into four categories: moving least squares (MLS)-based methods, locally optimal projection (LOP)-based methods, sparsity-based methods, and non-local similarity-based methods.

**MLS-based methods.** MLS-based methods approximate a smooth surface from the input samples and project the points to the resulting surface. To construct the surface, the method in [25] first finds the local reference domain for each point that best fits its neighboring points in terms of MLS, then defines a function based on the reference domain by fitting a polynomial function to neighboring data.

Several extensions, which address the unstable reconstruction problem in the case of high curvature, *e.g.*, algebraic point set surfaces (APSS) [9] and its variant in [26], or preserve the shape features, *e.g.*, robust MLS (RMLS) [27] and robust implicit MLS (RIMLS) [10], have also been proposed. These

methods can robustly generate a smooth surface from extremely noisy input, but are often criticized for over-smoothing [7], [8].

**LOP-based methods.** Unlike MLS-based methods, LOP-based methods do not compute explicit parameters for the surface. For example, LOP method in [11] outputs a set of points that represent the underlying surface while enforcing a uniform distribution over the point cloud with a repulse term in the optimization. Its modifications include weighted LOP (WLOP) [28], which provides a more uniformly distributed output by adapting the repulse term to the local density, and anisotropic WLOP (AWLOP) [12], which preserves sharp features by modifying WLOP to use an anisotropic weighting function. LOP-based methods also suffer from over-smoothing due to the use of local operators, or generate extra features caused by noise [7], [8].

**Sparsity-based methods.** Sparsity-based methods are based on a local planarity assumption and optimize for sparse representations of certain geometric features. Methods based on the sparsity of surface normals would first obtain a sparse reconstruction of the surface normals by solving a global minimization problem with  $l_1$  [13] or  $l_0$  [7] regularization, then update the point positions with the surface normals by solving another global minimization problem based on the locally planar assumption. A more recent method called Moving Robust Principal Components Analysis (MRPCA) [6] uses  $l_1$  minimization of the point deviations from the local reference plane to preserve sharp features. Sparsity-based approaches are reported to achieve the state-of-the-art performance [14], though at a high level of noise, the estimation of normal or local plane can be so poor that it leads to over-smoothing or over-sharpening [7].

**Non-local methods.** Non-local methods are widely adopted in image denoising [29]–[33]. Non-local methods generalize the notion of non-local self-similarity in the non-local means [15] and BM3D [16] image denoising algorithms to point cloud denoising, and are shown to better preserve structural features under high level of noise.

Due to the lack of regular structure in a point cloud, extending non-local image denoising schemes to point cloud is difficult. [17] utilizes curvature-based similarity to perform non-local filtering, so that the filtering considers the neighborhood geometry structure and better preserves fine shape features. [18] proposes to use the polynomial coefficients of the local MLS surface as neighborhood descriptors to compute point similarity.

Inspired by the BM3D algorithm, [5] exploits self-similarity among surface patches in the point cloud and outperforms the non-local means methods. However, the computational complexity is typically too high to be practical, taking a few hours for a point cloud of size 15,000 as reported in [5]. A more recent method in [19] also utilizes patch self-similarity and optimizes for a low-rank dictionary representation of the extracted patches to impose patch smoothness. During patch extraction, the points in each patch are projected to a regular grid for subsequent linear operations where multiple points can fall to the same location, leading to lose of fine structure and over-smoothing. The method is referred to as LR for short

hereinafter.

Our method belongs to the fourth category, the non-local methods. Similar to [5], [19], we also utilize the self-similarity among patches via the low-dimensional manifold prior [20]. However, the original PIM for manifold dimension minimization in [20] is not applicable to the point cloud setting due to the lack of regular structure of surface patches to define coordinate functions. Even if the coordinate functions are provided, PIM is time-consuming because the linear systems derived from PIM are asymmetric and inefficient to solve. In contrast, thanks to GLR, our approach eliminates the need for coordinate functions and can be efficiently implemented, outperforming existing schemes with better feature preservation.

In [23], PIM is approximated with the weighted nonlocal graph Laplacian (WNLL) to reduce computational complexity. The WNLL also preserves the symmetry of the linear systems with a graph Laplacian to speed up the implementation, but the Laplacian matrix is derived from the Laplace-Beltrami equation in PIM thus *different* from our GLR. Nevertheless, similar to PIM, the WNLL approach is designed for image restoration and solves each coordinate function separately, thus cannot be directly applicable to point clouds.

### III. PATCH MANIFOLD

We first define the notion of *patch manifold* given a point cloud  $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^N$ ,  $\mathbf{v}_i \in \mathbb{R}^3$ , which is a (roughly uniform) discrete sampling of a 2D surface of a 3D object. Let  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_N]^\top \in \mathbb{R}^{N \times 3}$  be the position matrix for the point cloud. Noise-corrupted  $\mathbf{V}$  can be simply modeled as:

$$\mathbf{V} = \mathbf{U} + \mathbf{E}, \quad (1)$$

where  $\mathbf{U}$  contains the true 3D positions,  $\mathbf{E}$  is a zero-mean signal-independent noise (we assume Gaussian noise in our experiments), and  $\mathbf{U}, \mathbf{E} \in \mathbb{R}^{N \times 3}$ . To recover the true position  $\mathbf{U}$ , we consider the *low-dimensional manifold model* prior (LDMM) [20] as a regularization term for this ill-posed problem.

#### A. Surface Patch

We first define a *surface patch* in a point cloud. We select a subset of  $M$  points from  $\mathcal{V}$  as the *patch centers*, i.e.,  $\{\mathbf{c}_m\}_{m=1}^M \subset \mathcal{V}$ . Then, patch  $p_m$  centered at a given center  $\mathbf{c}_m$  is defined as the set of  $k$  nearest neighbors of  $\mathbf{c}_m$  in  $\mathcal{V}$ , in terms of Euclidean distance.

The union of the patches should cover the whole point cloud, i.e.,  $\bigcup_{m=1}^M p_m = \mathcal{V}$ . There can be different choices of patch centers, and the degree of freedom can be used to trade off computation cost and denoising performance. Let  $\mathbf{p}_m \in \mathbb{R}^{3k}$  be the patch coordinates, composed of the  $k$  points in  $p_m$ .

#### B. Patch Manifold

Here we adopt the basic assumption in [20] that the patches sample a low-dimensional smooth manifold embedded in  $\mathbb{R}^{3k}$ , which is called the *patch manifold*  $\mathcal{M}(\mathbf{U})$  associated with the point cloud  $\mathbf{U}$ . In order to evaluate similarity among patches,

we first need to align the patches; i.e., the coordinates  $\mathbf{p}_m$  should be translated with respect to  $\mathbf{c}_m$ , so that  $\mathbf{c}_m$  lies on the origin  $(0, 0, 0)$ . Hereafter we set  $\{\mathbf{p}_m\}_{m=1}^M$  to be the translated coordinates.

#### C. Low Dimensional Patch Manifold Prior

The LDMM prior assumes that the solution contains patches that minimize the patch manifold dimension. We can thus formulate a *maximum a posteriori* (MAP) problem with prior and fidelity terms as follows:

$$\min_{\mathbf{U}} \dim(\mathcal{M}(\mathbf{U})) + \lambda \|\mathbf{V} - \mathbf{U}\|_F^2, \quad (2)$$

where  $\lambda$  is a parameter that trades off the prior with the fidelity term, and  $\|\cdot\|_F^2$  is the Frobenius norm. Note that given a certain strategy of patch selection, the patches are determined by the point cloud  $\mathbf{U}$ , and the patches in turn define the underlying manifold  $\mathcal{M}$ . Hence we view  $\mathcal{M}$  as a function of  $\mathbf{U}$ .

The patches can be very different and sampled from different manifolds of different dimensions. For example, a flat planar patch belongs to a manifold of lower dimension than a patch with corners. The dimension of the patch manifold,  $\dim(\mathcal{M}(\mathbf{U}))$  becomes a function of the patch, and the integration of  $\dim(\mathcal{M}(\mathbf{U}))$  over  $\mathcal{M}$  is used as the regularization term,

$$\min_{\mathbf{U}} \int_{\mathcal{M}} \dim(\mathcal{M}(\mathbf{U}))(\mathbf{p}) d\mathbf{p} + \lambda \|\mathbf{V} - \mathbf{U}\|_F^2, \quad (3)$$

where  $\dim(\mathcal{M}(\mathbf{U}))(\mathbf{p})$  is the dimension of  $\mathcal{M}(\mathbf{U})$  at  $\mathbf{p}$ . Here  $\mathbf{p} \in \mathbb{R}^{3k}$  is a point on  $\mathcal{M}$ . The question that remains is how to compute  $\dim(\mathcal{M}(\mathbf{U}))(\mathbf{p})$ . In the next section, the dimension computation is mathematically defined and approximated with GLR.

### IV. PROBLEM FORMULATION

In this section, we first briefly review the calculation of the manifold dimension in continuous domain, then approximate this computation with the GLR so as to efficiently adopt LDMM to discrete point cloud patches.

#### A. Manifold Dimension Computation in Continuous Domain

Here we overview how the manifold dimension is computed in [20]. First, let  $\alpha_i$ , where  $i = 1, \dots, 3k$ , be the coordinate functions on the manifold  $\mathcal{M}$  embedded in  $\mathbb{R}^{3k}$ , i.e.,

$$\alpha_i(\mathbf{p}) = p_i, \quad \forall \mathbf{p} = [p_1, \dots, p_{3k}]^\top \in \mathcal{M}. \quad (4)$$

According to [20], the dimension of  $\mathcal{M}$  at  $\mathbf{p}$  is given by:

$$\dim(\mathcal{M})(\mathbf{p}) = \sum_{i=1}^{3k} \|\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})\|^2, \quad (5)$$

where  $\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})$  denotes the gradient of the function  $\alpha_i$  on  $\mathcal{M}$  at  $\mathbf{p}$ . Then the integration of  $\dim(\mathcal{M})(\mathbf{p})$  over  $\mathcal{M}$  is given as,

$$\int_{\mathcal{M}} \dim(\mathcal{M})(\mathbf{p}) d\mathbf{p} = \sum_{i=1}^{3k} \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})\|^2 d\mathbf{p}. \quad (6)$$

The formula in (6) is a sum of integrals on continuous manifold  $\mathcal{M}$  along different dimensions, but our observations  $\{\mathbf{p}_m\}_{m=1}^M$  of the manifold  $\mathcal{M}(\mathbf{U})$  are discrete and finite. In [20], the solution to the dimension minimization is given by a partial derivative equation (PDE) for each  $\alpha_i$  separately, which is discretized at the patch observations using PIM, solved via a linear system.

However, PIM requires the patch coordinates  $\{\mathbf{p}_m\}_{m=1}^M$  to be ordered so that the  $\alpha_i$ 's can be defined. For example, if the patches are image patches of the same size, then the patch coordinates are naturally ordered according to pixel location, *i.e.*, the  $i$ -th entry in  $\mathbf{p}_m$  is the pixel value at the  $i$ -th location in the image patch. However, surface patches in the 3D point cloud are unstructured, and there is no natural way to implement global coordinate ordering for all patches.

This motivates us to discretize the manifold dimension with GLR, eliminating the need for global ordering and can be implemented efficiently.

### B. Dimension Discretization with GLR

We first introduce the graph construction on a manifold, which induces the GLR. Then we discuss how the GLR approximates the manifold dimension and avoids global coordinate ordering.

1) *Constructing Graph on a Manifold:* We construct a discrete graph  $\mathcal{G}$  whose vertex set is the observed surface patches  $\mathcal{P} = \{\mathbf{p}_m\}_{m=1}^M$  lying on  $\mathcal{M}(\mathbf{U})$ , *i.e.*,  $\mathbf{p}_m \in \mathcal{M}(\mathbf{U}) \subset \mathbb{R}^{3k}$ . Let  $\mathcal{E}$  denote the edge set, where the edge between  $m$ -th and  $n$ -th patches is weighted as,

$$w_{mn} = (\rho_m \rho_n)^{-1/\gamma} \psi(d_{mn}). \quad (7)$$

The kernel  $\psi(\cdot)$  is a thresholded Gaussian function

$$\psi(d_{mn}) = \begin{cases} \exp(-\frac{d_{mn}^2}{2\epsilon^2}) & d_{mn} < r \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

and  $d_{mn}$  is the Euclidean distance between the two patches  $\mathbf{p}_m$  and  $\mathbf{p}_n$ ,

$$d_{mn} = \|\mathbf{p}_m - \mathbf{p}_n\|_2. \quad (9)$$

The term  $(\rho_m \rho_n)^{-1/\gamma}$  is the normalization term, where  $\rho_n = \sum_{m=1}^M \psi(d_{mn})$  is the degree of  $\mathbf{p}_n$  before normalization. The graph constructed in these settings is an  $r$ -neighborhood graph, *i.e.*, no edge has a distance greater than  $r$ . Here  $r = \epsilon C_r$ , and  $C_r$  is a constant.

2) *Graph Laplacian Regularizer:* With the edge weights defined above, we define the symmetric adjacency matrix  $\mathbf{A} \in \mathbb{R}^{M \times M}$ , with the  $(m, n)$ -th entry given by  $w_{mn}$ .  $\mathbf{D}$  denotes the diagonal degree matrix, where entry  $\mathbf{D}(m, m) = \sum_n w_{m,n}$ . The combinatorial graph Laplacian matrix is  $\mathbf{L} = \mathbf{D} - \mathbf{A}$  [24].

For the coordinate function  $\alpha_i$  on  $\mathcal{M}$  defined in (4), sampling  $\alpha_i$  at positions of  $\mathcal{P}$  leads to its discretized version,  $\boldsymbol{\alpha}_i = [\alpha_i(\mathbf{p}_1) \dots \alpha_i(\mathbf{p}_M)]^\top$ . The graph Laplacian  $\mathbf{L}$  induces the regularizer  $S_{\mathbf{L}}(\boldsymbol{\alpha}_i) = \boldsymbol{\alpha}_i^\top \mathbf{L} \boldsymbol{\alpha}_i$ . It can be shown that

$$S_{\mathbf{L}}(\boldsymbol{\alpha}_i) = \boldsymbol{\alpha}_i^\top \mathbf{L} \boldsymbol{\alpha}_i = \sum_{(m,n) \in \mathcal{E}} w_{mn} (\alpha_i(\mathbf{p}_m) - \alpha_i(\mathbf{p}_n))^2. \quad (10)$$

3) *Approximation with Graph Laplacian Regularizer:* We now show the convergence of the discrete graph Laplacian regularizer to the dimension of the underlying continuous manifold.

First, we declare the following theorem that relates  $S_{\mathbf{L}}(\boldsymbol{\alpha}_i)$  to the integral of  $\|\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})\|_2^2$  on  $\mathcal{M}$  on the right side of (6):

**Theorem 1.** *Under conditions specified in Appendix A for  $\epsilon$ ,  $\mathcal{M}$  and function  $\alpha_i$ ,*

$$\lim_{\substack{M \rightarrow \infty, \\ \epsilon \rightarrow 0, \delta \rightarrow 0}} S_{\mathbf{L}}(\boldsymbol{\alpha}_i) \sim \frac{1}{|\mathcal{M}|} \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})\|_2^2 d\mathbf{p}, \quad (11)$$

where  $|\mathcal{M}|$  is the volume of the manifold  $\mathcal{M}$ ,  $\delta$  is the manifold dimension, and  $\sim$  means there exists a constant depending on  $\mathcal{M}$ ,  $C_r$  and  $\gamma$ , such that the equality holds.

In other words, as the number of samples  $M$  increases and the neighborhood size  $r = \epsilon C_r$  shrinks,  $S_{\mathbf{L}}(\boldsymbol{\alpha}_i)$  approaches its continuous limit. Moreover, if the manifold dimension  $\delta$  is low, we can ensure a good approximation of the continuous regularization functional even if the manifold is embedded in a high-dimensional space. Detailed proof for the above theorem is provided in Appendix A.

Consequently, given a point cloud, one can approximate the dimension of  $\mathcal{M}$  with the  $\alpha_i$ 's and the constructed graph Laplacian  $\mathbf{L}$  following (6) and (11):

$$\begin{aligned} \lim_{\substack{M \rightarrow \infty, \\ \epsilon \rightarrow 0, \delta \rightarrow 0}} |\mathcal{M}| \sum_{i=1}^{3k} \boldsymbol{\alpha}_i^\top \mathbf{L} \boldsymbol{\alpha}_i &\sim \sum_{i=1}^{3k} \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})\|_2^2 d\mathbf{p} \\ &= \int_{\mathcal{M}} \dim(\mathcal{M})(\mathbf{p}) d\mathbf{p}. \end{aligned} \quad (12)$$

Note that Theorem 1 is derived based on the combinatorial Laplacian matrix and does not apply to other types of Laplacian, *e.g.*, normalized Laplacian  $\mathcal{L} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$ . Further, a regularizer using  $\mathcal{L}$  would penalize a constant signal, since the eigenvector corresponding to eigenvalue 0 is not constant [34], which means it cannot handle constant signal. Experimental comparison between combinatorial and normalized Laplacian is provided in Section VI-B4.

### C. From Global Coordinate Ordering to Local Correspondence

So far, we obtain the approximation in (12), but the above graph construction still requires the patch coordinates  $\{\mathbf{p}_m\}_{m=1}^M$  to be ordered so that the  $\alpha_i$ 's can be defined and the patch distance  $d_{mn}$  in (9) determines the patch similarity.

In the following, we argue that the computation of the regularization term can be accomplished based on local pairwise correspondence between connected patches, relieving the need for global ordering.

We modify the manifold dimension formula in (12):

$$\sum_{i=1}^{3k} \alpha_i^\top \mathbf{L} \alpha_i = \sum_{i=1}^{3k} \sum_{(m,n) \in \mathcal{E}} w_{mn} (\alpha_i(\mathbf{p}_m) - \alpha_i(\mathbf{p}_n))^2 \quad (13)$$

$$= \sum_{(m,n) \in \mathcal{E}} w_{mn} \sum_{i=1}^{3k} (\alpha_i(\mathbf{p}_m) - \alpha_i(\mathbf{p}_n))^2 \quad (14)$$

$$= \sum_{(m,n) \in \mathcal{E}} w_{mn} d_{mn}^2, \quad (15)$$

where (15) follows from (14) according to the definition of  $d_{mn}$  in (9). From (15) we see that  $\alpha_i$  is not necessary to compute the graph Laplacian regularizer, and hence global coordinate ordering is not required. Moreover, since  $w_{mn}$  is itself a function of  $d_{mn}$  via (7), we can obtain the manifold dimension as long as  $d_{mn}$  is given by finding the local pairwise correspondence between neighboring patches.

To reformulate (15) into matrix form, we first consider the subgraphs composed of connected patch pair to reformulate  $w_{mn} d_{mn}^2$ , then sum up the weights between patch pairs to give the final GLR.

For a connected patch pair  $\mathbf{p}_m$  and  $\mathbf{p}_n$ , let  $\mathbf{p}_{mn}^\Theta = [\mathbf{p}_{m,\Theta}^\top \ \mathbf{p}_{n,\Theta}^\top]^\top$  be the concatenation of  $\mathbf{p}_m$  and  $\mathbf{p}_n$  coordinates, where  $\Theta \in \{x, y, z\}$  denotes the 3D coordinates. Given the local correspondence between  $\mathbf{p}_m$  and  $\mathbf{p}_n$ , we connect corresponding points to construct the subgraph and multiply the edge weights with  $w_{mn}$ , resulting in the graph Laplacian matrix  $\mathbf{L}_{mn}$  for this subgraph.  $w_{mn} d_{mn}^2$  is then reformulated as:

$$w_{mn} d_{mn}^2 = \sum_{\Theta \in \{x, y, z\}} (\mathbf{p}_{mn}^\Theta)^\top \mathbf{L}_{mn} \mathbf{p}_{mn}^\Theta. \quad (16)$$

Let  $\mathbf{S}_{mn} \in \{0, 1\}^{2k \times kM}$  be the sampling matrix to extract  $\mathbf{p}_{mn}^\Theta$  from  $\mathbf{P}_\Theta$ , where  $\mathbf{P}_\Theta$  is the coordinate vector of points in all patches, i.e.,  $\mathbf{p}_{mn}^\Theta = \mathbf{S}_{mn} \mathbf{P}_\Theta$ , so that  $w_{mn} d_{mn}^2$  becomes:

$$w_{mn} d_{mn}^2 = \sum_{\Theta \in \{x, y, z\}} (\mathbf{S}_{mn} \mathbf{P}_\Theta)^\top \mathbf{L}_{mn} (\mathbf{S}_{mn} \mathbf{P}_\Theta). \quad (17)$$

Then the manifold dimension becomes:

$$\sum_{(m,n) \in \mathcal{E}} w_{mn} d_{mn}^2 \quad (18)$$

$$= \sum_{\Theta \in \{x, y, z\}} \sum_{(m,n) \in \mathcal{E}} \mathbf{P}_\Theta^\top (\mathbf{S}_{mn}^\top \mathbf{L}_{mn} \mathbf{S}_{mn}) \mathbf{P}_\Theta \quad (19)$$

$$= \sum_{\Theta \in \{x, y, z\}} \mathbf{P}_\Theta^\top \left( \sum_{(m,n) \in \mathcal{E}} \mathbf{S}_{mn}^\top \mathbf{L}_{mn} \mathbf{S}_{mn} \right) \mathbf{P}_\Theta \quad (20)$$

$$= \sum_{\Theta \in \{x, y, z\}} \mathbf{P}_\Theta^\top \mathbf{L}_p \mathbf{P}_\Theta, \quad (21)$$

where

$$\mathbf{L}_p = \sum_{(m,n) \in \mathcal{E}} \mathbf{S}_{mn}^\top \mathbf{L}_{mn} \mathbf{S}_{mn} \in \mathbb{R}^{kM \times kM} \quad (22)$$

is the overall graph Laplacian matrix for the point-domain graph.

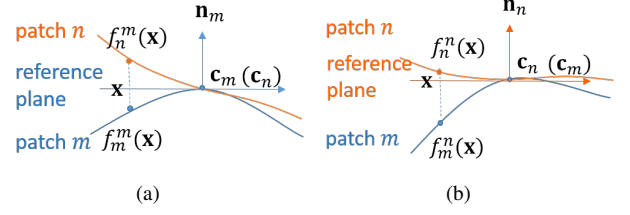


Fig. 1. Distance measure with continuous surfaces with reference plane perpendicular to: (a) surface normal  $\mathbf{n}_m$  at center of patch  $m$ , and (b) surface normal  $\mathbf{n}_n$  at center of patch  $n$ .

#### D. Objective Formulation with GLR Prior

With  $\mathbf{L}_p$  calculated as described above, the optimization is reformulated as:

$$\min_U \sum_{\Theta \in \{x, y, z\}} \mathbf{P}_\Theta^\top \mathbf{L}_p \mathbf{P}_\Theta + \mu \|\mathbf{V} - \mathbf{U}\|_F^2, \quad (23)$$

Let  $\mathbf{P} = [\mathbf{P}_x, \mathbf{P}_y, \mathbf{P}_z] \in \mathbb{R}^{kM \times 3}$ , and  $\sum_{\Theta \in \{x, y, z\}} \mathbf{P}_\Theta^\top \mathbf{L}_p \mathbf{P}_\Theta$  can be combined as  $\text{tr}(\mathbf{P}^\top \mathbf{L}_p \mathbf{P})$ .  $\mathbf{P}$  is related to denoised 3D samples  $\mathbf{U} \in \mathbb{R}^{N \times 3}$  as follows:

$$\mathbf{P} = \mathbf{S} \mathbf{U} - \mathbf{C}, \quad (24)$$

where  $\mathbf{S} \in \{0, 1\}^{kM \times N}$  is a sampling matrix to select points from point cloud  $\mathcal{V}$  to form  $M$  patches of  $k$  3D points each, and  $\mathbf{C} \in \mathbb{R}^{kM \times 3}$  is for patch centering. Hence, the objective function can be rewritten as:

$$\min_U \text{tr}((\mathbf{S} \mathbf{U} - \mathbf{C})^\top \mathbf{L}_p (\mathbf{S} \mathbf{U} - \mathbf{C})) + \mu \|\mathbf{V} - \mathbf{U}\|_F^2. \quad (25)$$

Now the questions that remain are: i) how to find local correspondence between connected patch pairs for graph construction, and ii) how to implement the numerical optimization. They are addressed in the next section.

#### V. ALGORITHM DEVELOPMENT

In this section, we first propose a patch distance measure for graph construction, and then discuss the algorithm implementation. Then we show that, with GLR, the algorithm is guaranteed with numerical stability and can be solved efficiently.

##### A. Patch Distance Measure

1) *Distance Measure in Continuous Domain*: To measure the distance between the  $m$ -th patch and  $n$ -th patch, ideally the two patches can be interpolated to two continuous surfaces, and the distance is calculated as the integral of the surface distance over a local domain around the patch center.

To define the underlying surface, we first define a *reference plane*. In Fig. 1(a), we examine a 2D case for illustration. The reference plane is tangent to the center  $\mathbf{c}_m$  of patch  $m$  (origin point) and perpendicular to the surface normal  $\mathbf{n}_m$  at  $\mathbf{c}_m$ . Then the *surface distance* for patch  $m$  with respect to normal  $\mathbf{n}_m$  is defined as a function  $f_m^m(\mathbf{x})$ , where  $\mathbf{x}$  is a point on the reference plane, superscript  $m$  indicates that the reference plane is perpendicular to  $\mathbf{n}_m$ , while the subscript  $m$  indicates the function defines patch  $m$ .  $f_m^m(\mathbf{x})$  is then the perpendicular

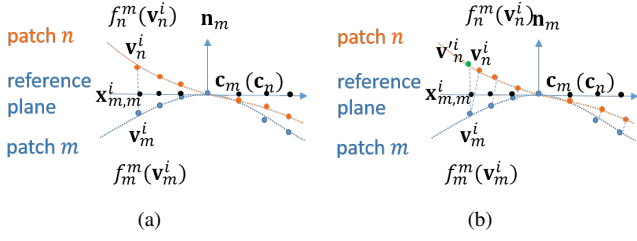


Fig. 2. Distance measure with discrete patches with reference plane perpendicular to surface normal  $\mathbf{n}_m$  at center of patch  $m$ . (a) Ideal case where the points with the same projection is connected. (b) A point in patch  $m$  is connected with the closest point in patch  $n$  in terms of projection distance.

distance from  $\mathbf{x}$  to surface  $m$  with respect to normal  $\mathbf{n}_m$ . Surface  $n$  is similarly defined as  $f_n^m(\mathbf{x})$ . Note that because the patches are centered,  $\mathbf{c}_m = \mathbf{c}_n$  which is the origin, but their surface normals  $\mathbf{n}_m$  and  $\mathbf{n}_n$  are typically different.

The patch distance is then computed as

$$d_{mn} = \sqrt{\frac{1}{|\Omega_m|} \int_{\mathbf{x} \in \Omega_m} (f_m^m(\mathbf{x}) - f_n^m(\mathbf{x}))^2 d\mathbf{x}}, \quad (26)$$

where  $\Omega_m$  is the local neighborhood at  $\mathbf{c}_m$ .  $|\Omega_m|$  is the area of  $\Omega_m$ .  $d_{mn}$  denotes the distance measured with reference plane perpendicular to  $\mathbf{n}_m$ .

Note that different reference planes lead to different distance values, so we alternately use  $\mathbf{n}_m$  and  $\mathbf{n}_n$  to define the reference plane. Fig. 1(b) illustrates the computation of  $d_{mn}$  with reference plane perpendicular to  $\mathbf{n}_n$ .

$$d_{nm} = \sqrt{\frac{1}{|\Omega_n|} \int_{\mathbf{x} \in \Omega_n} (f_n^m(\mathbf{x}) - f_m^m(\mathbf{x}))^2 d\mathbf{x}}, \quad (27)$$

where functions  $f_m^n$  and  $f_n^n$  define surfaces  $m$  and  $n$ , respectively, and  $\Omega_n$  the local neighborhood at  $\mathbf{c}_n$ .

$d_{mn}$  is then given as,

$$d_{mn} = \sqrt{\frac{d_{mn}^2 + d_{nm}^2}{2}}. \quad (28)$$

## 2) Distance Measure with Discrete Point Observation:

Since we only have discrete observations of the points on the patches, we instead measure the sum of the distances between points with the same projection on the reference plane.

First, we compute  $d_{mn}$ , where reference plane is perpendicular to the surface normal  $\mathbf{n}_m$  at  $\mathbf{c}_m$ . Specifically, patch  $m$  is composed of points  $\{\mathbf{v}_m^i\}_{i=1}^k$ , while patch  $n$  is composed of  $\{\mathbf{v}_n^i\}_{i=1}^k$ . The surface normal  $\mathbf{n}_m$  is given by,

$$\min_{\mathbf{n}_m} \sum_{i=1}^k ((\mathbf{v}_m^i)^\top \mathbf{n}_m)^2. \quad (29)$$

It can be shown via Principal Component Analysis [35] that the solution is the normalized eigenvector according to the smallest eigenvalue of the covariance matrix  $\mathbf{Q}$  given by,

$$\mathbf{Q} = \frac{1}{k} \sum_{i=1}^k \mathbf{v}_m^i (\mathbf{v}_m^i)^\top. \quad (30)$$

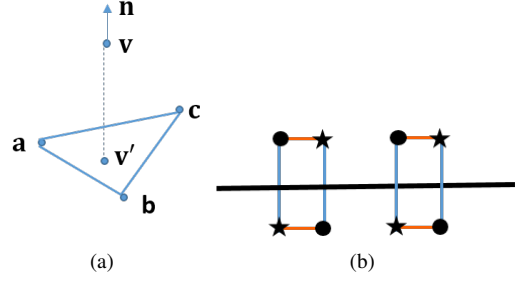


Fig. 3. (a) Interpolation for  $\mathbf{v}$  on the plane  $abc$ . (b) Patch connection based on projection (in blue) vs euclidean distance (in orange).

The same normal estimation method is used in PCL Library [1]. We then project both  $\{\mathbf{v}_m^i\}_{i=1}^k$  and  $\{\mathbf{v}_n^i\}_{i=1}^k$  to the reference plane, and the projections are  $\{\mathbf{x}_{m,m}^i\}_{i=1}^k$  and  $\{\mathbf{x}_{n,n}^i\}_{i=1}^k$  respectively, where the second index in subscript indicates that the reference plane is perpendicular to  $\mathbf{n}_m$ . The distances between  $\mathbf{v}_m^i$  and  $\mathbf{x}_{m,m}^i$  give the surface distance  $f_m^m(\mathbf{v}_m^i)$ , and the distances between  $\mathbf{v}_n^i$  and  $\mathbf{x}_{n,n}^i$  give  $f_n^m(\mathbf{v}_n^i)$ .

Ideally, for any  $\mathbf{v}_m^i$  in patch  $m$ , there exists a point  $\mathbf{v}_n^i$  in patch  $n$  whose projection  $\mathbf{x}_{n,n}^i = \mathbf{x}_{m,m}^i$  as shown in Fig. 2(a). However, in real dataset,  $\mathbf{v}_m^i$  usually does not have a match in patch  $n$  with exactly the same projection, as illustrated in Fig. 2(b). In this case, we replace the displacement value of  $\mathbf{v}_m^i$  (the green point in Fig. 2(b)), which has the same projection  $\mathbf{x}_{m,m}^i$  as  $\mathbf{v}_m^i$ , with the value of its nearest neighbor  $\mathbf{v}_n^i$  in patch  $n$  in terms of the distance between their projections  $\mathbf{x}_{m,m}^i$  and  $\mathbf{x}_{n,n}^i$ . Then  $d_{mn}$  is computed as,

$$d_{mn} = \sqrt{\frac{1}{k} \sum_{i=1}^k (f_m^m(\mathbf{v}_m^i) - f_n^m(\mathbf{v}_n^i))^2}. \quad (31)$$

Similarly, to compute  $d_{nm}$ , we define reference plane with  $\mathbf{n}_n$ , then compute the projections  $\{\mathbf{x}_{m,n}^i\}_{i=1}^k$  and  $\{\mathbf{x}_{n,n}^i\}_{i=1}^k$  and displacements  $f_m^n(\mathbf{v}_m^i)$ ,  $f_n^n(\mathbf{v}_n^i)$ . For each  $\mathbf{v}_n^i$  in patch  $n$ , we match it to the closest point in patch  $m$  in terms of projection distance. Then  $d_{nm}$  is computed as,

$$d_{nm} = \sqrt{\frac{1}{k} \sum_{i=1}^k (f_m^n(\mathbf{v}_m^i) - f_n^n(\mathbf{v}_n^i))^2}. \quad (32)$$

The final distance is given as (28).

3) *Planar Interpolation:* The pairwise correspondence is based on nearest neighbor replacement, though more accurate interpolation can be adopted. However, due to the large size of the point cloud, implementing interpolation for all the points can be expensive. Thus we use nearest-neighbor replacement when the distance between point pair is under a threshold  $\tau$ . When the distance goes above  $\tau$ , we apply the interpolation method described as follows.

As shown in Fig. 3(a), for a point  $\mathbf{v}$ , to find its corresponding interpolation on the other patch, we find the three nearest points (also in terms of projection distance) to form a plane, and the interpolation  $\mathbf{v}'$  is given by its projection along the normal vector  $\mathbf{n}$  on the plane. It can be easily derived that the distance between  $\mathbf{v}$  and  $\mathbf{v}'$  is  $\frac{\mathbf{n}^\top \mathbf{v} + d}{\mathbf{n}^\top \mathbf{n}_0}$  where  $\mathbf{n}_0 = \mathbf{a}\mathbf{b} \times \mathbf{a}\mathbf{c}$  is the normal vector for the plane  $abc$ , and  $d = -\mathbf{n}_0^\top \mathbf{a}$ .

4) *Relation to Hausdorff Distance*: Hausdorff distance [36] is a widely used measure for comparing point clouds, which is derived from the Hausdorff distance for comparing the metric spaces of two manifolds and extended to deal with point clouds [37]. The proposed patch distance measure is closely related to the modified Hausdorff distance (MHD) [38], which is a variant of Hausdorff distance. It decreases the impact of outliers and is more suitable for pattern recognition tasks. Specifically, MHD from the  $m$ -th patch and  $n$ -th patch is given as:

$$\text{MHD}_{\overrightarrow{mn}} = \frac{1}{k} \sum_{i=1}^k \|\mathbf{v}_m^i - \mathbf{v}_n^i\|, \quad (33)$$

where  $\|\cdot\|$  is the Euclidean distance,  $\mathbf{v}_n^i$  is the nearest neighbor of  $\mathbf{v}_m^i$  in patch  $n$  in terms of point position. The major difference between MHD and our patch distance measure is that, we choose to use projection on the reference plane (e.g.  $\mathbf{x}_m^i$  in Fig. 2(a)) to find the correspondence, while MHD uses the point position (e.g.  $\mathbf{v}_m^i$  in Fig. 2(a)).

Due to the use of projection, the proposed measure is more robust to noise than MHD. For example in Fig. 3(b), the underlying surfaces for two patches are both planar, where the circle points belong to one patch and the star points belong to the other. The correct connections are between points along the vertical lines (in blue). This is accomplished by using projection on the reference plane. On the other hand, if the connection is decided by point position, then the resulting connections are erroneous (in orange) and thus lead to inefficient denoising. Therefore point connection based on projection is closer to the ground truth and more robust to noise.

### B. Graph Construction

Based on the above patch distance measure strategy, the connection between  $m$ -th and  $n$ -th patch is implemented as follows. If no interpolation is involved, the points in the  $m$ -th patch are connected with the nearest points in the  $n$ -th patch in terms of their projections on the reference plane decided by surface normal of patch  $m$ . Also, the points in the  $n$ -th patch are connected with the nearest points in the  $m$ -th patch in terms of their projections on the reference plane decided by surface normal of patch  $n$ . The edges are undirected and assigned the same weight  $w_{mn}$  decided by  $d_{mn}$  in (7).

If interpolation is involved, for example in Fig. 3(a), the weight  $w_{va}$  between  $\mathbf{v}$  and  $\mathbf{a}$  is given by,

$$w_{va} = \frac{w_{mn}d_{va}}{d_{va} + d_{vb} + d_{vc}}, \quad (34)$$

where  $w_{mn}$  is the weight between patch  $m$  and  $n$ . Point  $\mathbf{v}$  lies on patch  $m$  and points  $\mathbf{a}$ ,  $\mathbf{b}$ ,  $\mathbf{c}$  lie on patch  $n$ .  $d_{va}$  is distance between  $\mathbf{v}$  and  $\mathbf{a}$ , and similarly for  $d_{vb}$  and  $d_{vc}$ . To simplify the implementation, we limit the search range to be patches centered at the  $K$ -nearest patch centers, and evaluate patch distance between these  $K$ -nearest patches instead of all the patches in the point cloud.

In this way, the local correspondence is generated and the point domain graph is constructed, giving the graph Laplacian  $\mathbf{L}_p$  in (25).

### Algorithm 1 Graph Laplacian Regularized Point Cloud Denoising

**Input**: Noisy point cloud  $\mathbf{V}$ , patch center sampling rate  $s\%$ , patch size  $k$ , threshold  $\tau$ , max iteration number  $r$

**Output**: Denoised point cloud  $\mathbf{U}$

```

1: Initialize  $\mathbf{U}^0 \leftarrow \mathbf{V}$ 
2: for  $i = 1$  to  $r$  do
3:   Sample  $s\%$  points from  $\mathbf{U}^i$  as patch centers
4:   Find  $k$  nearest neighbors of each patch center to form surface patches
5:   Connect each patch center with  $K$  nearest neighboring patch centers to give  $\mathcal{E}$ 
6:   for  $(m, n) \in \mathcal{E}$  do
7:     Connect corresponding points between  $m$ -th and  $n$ -th patches and compute  $\mathbf{L}_{mn}$ 
8:   end for
9:    $\mathbf{L}_p \leftarrow \sum_{(m,n) \in \mathcal{E}} \mathbf{S}_{mn}^\top \mathbf{L}_{mn} \mathbf{S}_{mn}$ 
10:   $\mathbf{U}_\Theta^i \leftarrow (\mathbf{S}^\top \mathbf{L}_p \mathbf{S} + \mu \mathbf{I})^{-1} (\mu \mathbf{U}_\Theta^{i-1} + \mathbf{S}^\top \mathbf{L}_p \mathbf{C}_\Theta)$ ,  $\Theta \in \{x, y, z\}$ 
11:  End if  $\mathbf{U}^i$  converges
12: end for
```

### C. Denoising Algorithm

The optimization in (25) is non-convex because of  $\mathbf{L}_p$ 's dependency on patches in  $\mathbf{P}$ . To solve (25) approximately, we take an alternating approach, where in each iteration, we fix  $\mathbf{L}_p$  and solve for  $\mathbf{U}$ , then update  $\mathbf{L}_p$  given  $\mathbf{U}$ , and repeat until convergence.

In each iteration, graph Laplacian  $\mathbf{L}_p$  is easy to update using the previously discussed graph construction strategy. To optimize  $\mathbf{U}$  for fixed  $\mathbf{L}_p$ , each of the  $(x, y, z)$  coordinate is given by,

$$(\mathbf{S}^\top \mathbf{L}_p \mathbf{S} + \mu \mathbf{I}) \mathbf{U}_\Theta = \mu \mathbf{V}_\Theta + \mathbf{S}^\top \mathbf{L}_p \mathbf{C}_\Theta, \quad (35)$$

where  $\Theta \in \{x, y, z\}$  is the index for  $(x, y, z)$  coordinates, and  $\mathbf{I}$  is the identity matrix of the same size as  $\mathbf{L}_p$ . We iteratively solve the optimization until the result converges. The proposed algorithm is referred to as *Graph Laplacian Regularized point cloud denoising* (GLR). The algorithm is summarized in Algorithm 1.

### D. Graph Spectral Analysis

To impart intuition and demonstrate stability of our computation, in each iteration we can compute the optimal  $x$ -,  $y$ - and  $z$ -coordinates in (25) separately, resulting in the system of linear equations in (35). In Section III, we assume that union of all  $M$  patches covers all points in the point cloud  $\mathcal{V}$ , hence we can safely assume that  $kM > N$ .

Because  $\mathbf{S}$  is a sampling matrix, we can define  $\mathbf{L} = \mathbf{S}^\top \mathbf{L}_p \mathbf{S}$  as a  $N \times N$  *principal sub-matrix*<sup>1</sup> of  $\mathbf{L}_p$ . Denote by  $\lambda_1^\mu \leq \dots \leq \lambda_N^\mu$  the eigenvalues of matrix  $\mathbf{L} + \mu \mathbf{I}$ . The solution to (35) can thus be written as:

$$\mathbf{U}_\Theta^* = \Phi \Sigma^{-1} \Phi^T (\mu \mathbf{V}_\Theta + \mathbf{S}^\top \mathbf{L}_p \mathbf{C}_\Theta), \quad (36)$$

where  $\Phi \Sigma \Phi^T$  is an eigen-decomposition<sup>2</sup> of matrix  $\mathbf{L} + \mu \mathbf{I}$ ; i.e.,  $\Phi$  contains as columns eigenvectors  $\phi_1, \dots, \phi_N$ , and

<sup>1</sup>A principal sub-matrix  $B$  of an original larger matrix  $A$  is one where the  $i$ -th row and column of  $A$  are removed iteratively for different  $i$ .

<sup>2</sup>Eigen-decomposition is possible because the target matrix  $\mathbf{L} + \mu \mathbf{I}$  is real and symmetric.



$\Sigma$  is a diagonal matrix containing eigenvalues on its diagonal. In *graph signal processing* (GSP) [24], eigenvalues and eigenvectors of a variational operator— $\mathbf{L} + \mu\mathbf{I}$  in our case—are commonly interpreted as graph frequencies and frequency components.  $\Phi^\top$  is thus an operator (called *graph Fourier basis*) that maps a graph-signal  $\mathbf{x}$  to its GFT coefficients  $\zeta = \Phi^\top \mathbf{x}$ .

Observing that  $\Sigma^{-1}$  in (36) is a diagonal matrix:

$$\Sigma^{-1} = \text{diag}(1/(\lambda_1^\mu + \mu), \dots, 1/(\lambda_N^\mu + \mu)), \quad (37)$$

we can thus interpret the solution  $\mathbf{U}_\Theta^*$  in (36) as follows. The noisy observation  $\mathbf{V}_\Theta$  (offset by centering vector  $\mathbf{C}_\Theta$ ) is transformed to the GFT domain via  $\Phi^\top$  and *low-pass filtered* per coefficient according to (37)—low-pass because weights  $1/(\lambda_i^\mu + \mu)$  for low frequencies are larger than large frequencies  $1/(\lambda_j^\mu + \mu)$ , for  $i < j$ . The fact that we are performing 3D point cloud denoising via graph spectral low-pass filtering should not be surprising.

### E. Numerical Stability via Eigen-Analysis

We can also estimate the stability of the system of linear equations in (36) via the following eigen-analysis. During graph construction, an edge weight  $w_{i,j}$  is computed using (7), which is upper-bounded by 1. Denote by  $\rho_{\max}$  the maximum degree of a node in the graph, which in general  $\rho_{\max} \ll N$ . According to the Gershgorin circle theorem [35], given a matrix  $\mathbf{A}$ , a Gershgorin disc  $i$  has radius  $r_i = \sum_{j|j \neq i} |A_{i,j}|$  and center at  $A_{i,i}$ . For a combinatorial graph Laplacian  $\mathbf{L}_p$ , the maximum Gershgorin disc radius is the maximum node degree multiplied by the maximum edge weight, which is  $\rho_{\max}$ . Further, the diagonal entry  $L_{i,i} = -\sum_{j|j \neq i} L_{i,j}$  for positive edge weights, which equals  $r_i$ . Thus all Gershgorin discs for a combinatorial graph Laplacian matrix have left-ends located at 0. By the Gershgorin circle theorem, all eigenvalues have to locate inside the union of all Gershgorin discs. This means that the maximum eigenvalue  $\lambda_{\max}^p$  for  $\mathbf{L}_p$  is upper-bounded by twice the radius of the largest possible disc, which is  $2\rho_{\max}$ .

Now consider principal sub-matrix  $\mathbf{L}$  of original matrix  $\mathbf{L}_p$ . By the eigenvalue interlacing theorem, large eigenvalue  $\lambda_{\max}$  for  $\mathbf{L}$  is upper-bounded by  $\lambda_{\max}^p$  of  $\mathbf{L}_p$ . For matrix  $\mathbf{L} + \mu\mathbf{I}$ , the smallest eigenvalue  $\lambda_{\min}^\mu \geq \mu$ , because: i)  $\mu\mathbf{I}$  shifts all eigenvalues of  $\mathbf{L}$  to the right by  $\mu$ , and ii)  $\mathbf{L}$  is PSD due to eigenvalue interlacing theorem and the fact that  $\mathbf{L}_p$  is PSD. We can thus conclude that the condition number<sup>3</sup>  $C$  of matrix  $\mathbf{L} + \mu\mathbf{I}$  on the left-hand side of (35) can be upper-bounded as follows:

$$C \leq \frac{2\rho_{\max} + \mu}{\mu}. \quad (38)$$

Hence for sufficiently small  $\rho_{\max}$ , the linear system of equations in (35) has a stable solution, and can be efficiently solved using indirect methods like preconditioned conjugate gradient (PCG).

<sup>3</sup>Assuming  $l_2$ -norm is used and the matrix is normal, then the condition number is defined as the ratio  $\lambda_{\max}/\lambda_{\min}$ .

### F. Complexity Analysis

The complexity of the algorithm depends on two main procedures: one is the patch-based graph construction, and the other is in solving the system of linear equations.

For graph construction, for the  $M$  patches, the  $K$ -nearest patches to be connected can be found in  $O(KM \log M)$  time. Then for  $k$ -point patch distance measure, each pair takes  $O(k \log k)$ ; with  $MK$  pairs, the complexity is  $O(kMK \log k)$  in total. For the system of linear equations, it can be solved efficiently with PCG based methods, with complexity of  $O(kMK\sqrt{C})$  [39]. Finally, if GLR runs for a maximum of  $r$  iterations, the total time complexity will be  $O(r(KM \log M + kMK \log k + kMK\sqrt{C})) \approx O(rKM(\log M + k \log k + k\sqrt{C}))$ .

The parameters that can be adjusted for the complexity reduction are patch center sampling density, patch graph neighborhood size and patch size. Details about the parameter setting and complexity comparison with other existing schemes are given in Section VI.

## VI. EXPERIMENTAL RESULTS

The proposed scheme GLR is compared with existing works: APSS [9], RIMLS [10], AWLOP [12], non-local denoising (NLD) algorithm [18] and the state-of-the-art MRPCA [6] and LR [19]. APSS and RIMLS are implemented with MeshLab software [40], AWLOP is implemented with EAR software [12], MRPCA source code is provided by the author, NLD and LR are implemented by ourselves in MATLAB. We first empirically tune parameters on a small dataset with 8 models, then generalize the parameter setting learned from the small dataset to a larger dataset, *i.e.*, 100 models from the ShapeNetCore dataset [41] for validation. Comparison with existing methods on both dataset are detailed as follows.

### A. Evaluation Metrics

Before the discussion of experimental performance, we first introduce the three evaluation metrics for point cloud denoising. Suppose the ground-truth and predicted point clouds are  $\mathcal{U} = \{\mathbf{u}_i\}_{i=1}^{N_1}$ ,  $\mathcal{V} = \{\mathbf{v}_i\}_{i=1}^{N_2}$ , where  $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{R}^3$ . The point clouds can be of different sizes, *i.e.*,  $N_1$  and  $N_2$  may be unequal. The metrics are defined as follows.

- 1) mean-square-error (MSE): We first measure the average of the squared Euclidean distances between ground truth points and their closest denoised points, and also between the denoised points and their closest ground truth points, then take the average between the two measures to compute MSE which is given as

$$\begin{aligned} \text{MSE} = & \frac{1}{2N_1} \sum_{\mathbf{u}_i \in \mathcal{U}} \min_{\mathbf{v}_j \in \mathcal{V}} \|\mathbf{u}_i - \mathbf{v}_j\|_2^2 \\ & + \frac{1}{2N_2} \sum_{\mathbf{v}_i \in \mathcal{V}} \min_{\mathbf{u}_j \in \mathcal{U}} \|\mathbf{v}_i - \mathbf{u}_j\|_2^2 \end{aligned} \quad (39)$$

- 2) signal-to-noise ratio (SNR): SNR is measured in dB given as

$$\text{SNR} = 10 \log \frac{1/N_2 \sum_{\mathbf{v}_i \in \mathcal{V}} \|\mathbf{v}_i\|_2^2}{\text{MSE}} \quad (40)$$



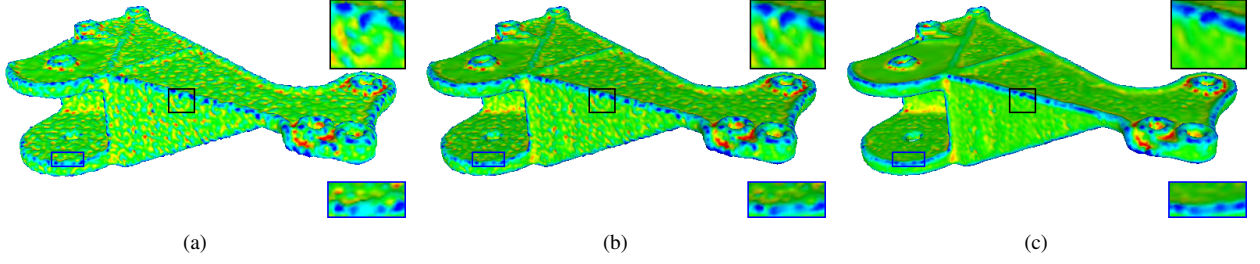


Fig. 4. *Daratech* model ( $\sigma = 0.02$ ). Surface reconstruction with (a) noisy input, and denoising results of the proposed GLR after (b) iteration 1 and (c) iteration 3 (final output), colored by mean curvature.

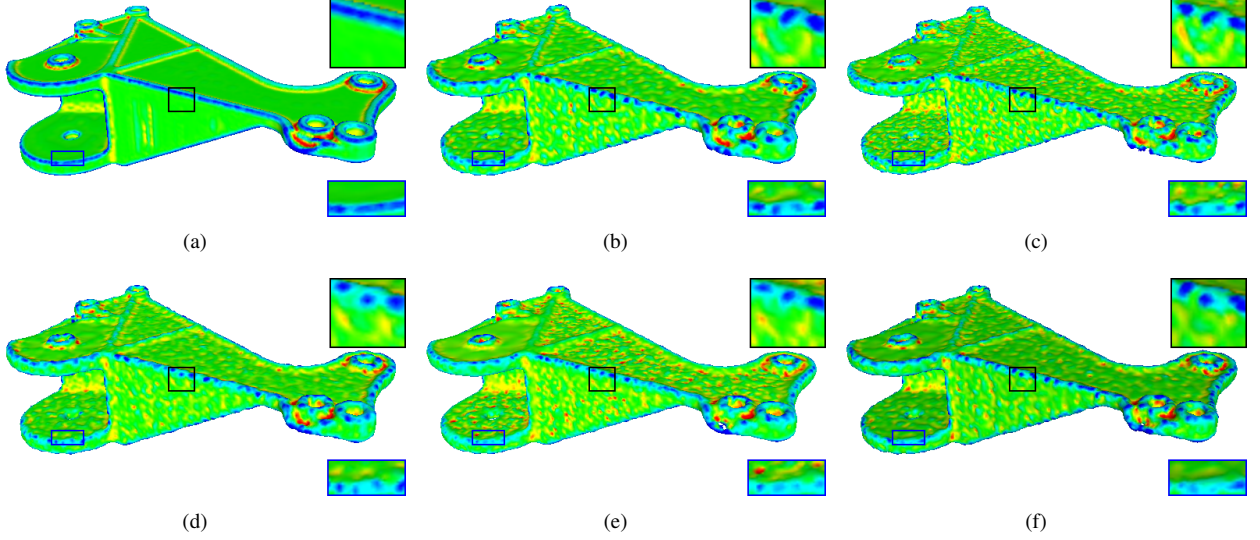


Fig. 5. *Daratech* model ( $\sigma = 0.02$ ). Surface reconstruction with (a) ground truth, and denoising results of (b) APSS, (c) AWLOP, (d) NLD, (e) MRPCA and (f) LR, colored with mean curvature.

- 3) mean city-block distance (MCD): MCD is similar to MSE with  $l_2$  norm replaced with  $l_1$  norm, given as

$$\begin{aligned} \text{MCD} = & \frac{1}{2N_1} \sum_{\mathbf{u}_i \in \mathcal{U}} \min_{\mathbf{v}_j \in \mathcal{V}} |\mathbf{u}_i - \mathbf{v}_j| \\ & + \frac{1}{2N_2} \sum_{\mathbf{v}_i \in \mathcal{V}} \min_{\mathbf{u}_j \in \mathcal{U}} |\mathbf{v}_i - \mathbf{u}_j| \end{aligned} \quad (41)$$

### B. Parameter Tuning

8 models are used for parameter tuning, including *Anchor*, *Bimda*, *Bunny*, *Daratech*, *DC*, *Fandisk*, *Gargoyle* and *Lordquas* provided in [5] and [6]. The models are around 50000 in size. Gaussian noise with zero-mean is added to the 3D positions of each point cloud, where the standard deviation is set proportional to the signal scale as commonly used in point cloud denoising works [5], [6]. We first compute the diameter of the point cloud, which is the maximum distance among 200 points sampled from the point cloud using farthest point sampling [42]. Then the standard deviation of the additive Gaussian noise is the multiplication of the diameter and  $\sigma$ , where  $\sigma = 0.02, 0.03, 0.04$ .

1) *Parameters in Optimization Formulation*: For implementation of the proposed GLR, we need to tune the parameter  $\mu$  for balancing the data fidelity term and the regularization

term in (23), and  $\epsilon$  for weighting the edge between connected patches in (8).

In (23), the GLR regularization reflects the prior expectation of signal smoothness on the graph [24] which can be estimated from the dataset for parameter tuning. Meanwhile, the data term measures the noise variance, and its ratio to expected signal smoothness is found to be similar for different models given the same noise level  $\sigma$ . This is because the noise standard variance is set proportional to signal scale as explained above. Therefore, given noise level  $\sigma$ ,  $\mu$  is tuned on the 8 models and generalizes to other models. Specifically,  $\mu = 25(\exp(\text{iteration}/r) - 1)$  which increases along the iterations, where  $r = 4, 7, 12$  for  $\sigma = 0.02, 0.03, 0.04$ , respectively.

From (8), we can see  $\epsilon$  should be proportional to square root of standard deviation of the patch distances, i.e.,  $\xi = \sqrt{\text{std}(d^2)}$  with a moderate scale, where  $\text{std}$  is the standard deviation,  $d$  denotes the distance of patch pair. Through testing on the 8 models, we empirically set  $\epsilon = 0.5\xi$ .

2) *Parameters for Performance and Speed Balance*: To speed up the implementation, we take 50% of the points as the patch centers, with the farthest point sampling [42] to assure spatially uniform selection. The planar interpolation threshold  $\tau$  is set to 1, which is large enough to ensure most points are connected using nearest-neighbor replacement for efficient

implementation. The maximum iteration number  $r$  is set to 15. To find the proper value for the search window size  $K$  and the patch size  $k$ , we study the performance sensitivity to  $K$  and  $k$ . We set  $K = 4, 8, 12, 16, 20$  and  $k = 15, 30, 60, 120$ , and test on the 8 models. The average MSE results are shown in Table I and II.

With larger search range  $K$ , each patch is more likely to find similar patches to get connected, so the results get better though the performance converges when  $K$  reaches 16. On the other hand, the runtime increases as  $K$  gets larger, so we choose  $K$  to be 16 to balance the performance and runtime. With small  $k$ , the patch size is too small to capture salient features, so the filtering cannot distinguish patch similarity and is not edge-aware; with large  $k$ , the patch contains too many salient features and the dimension of the patch manifold increases, thus the low-dimensional manifold model assumption is invalid. Therefore we choose  $k = 30$  as a suitable patch size which provides the best results in Table II.

TABLE I  
MSE RESULTS AND RUNTIME (SEC) OF DIFFERENT SETTING FOR  $K$

Noise Level	$K = 4$	$K = 8$	$K = 12$	$K = 16$	$K = 20$
0.02	0.147	0.145	0.143	<b>0.142</b>	0.143
0.03	0.173	0.171	0.166	0.165	<b>0.164</b>
0.04	0.196	0.190	0.184	<b>0.181</b>	<b>0.181</b>
runtime	131.6	227.4	323.7	399.0	496.2

TABLE II  
MSE RESULTS AND RUNTIME (SEC) OF DIFFERENT SETTING FOR  $k$

Noise Level	$k = 15$	$k = 30$	$k = 60$	$k = 120$
0.02	0.177	<b>0.142</b>	0.145	0.170
0.03	0.206	<b>0.165</b>	0.170	0.285
0.04	0.229	<b>0.181</b>	0.292	0.655
runtime	262.8	399.0	852.7	2535.2

3) *Objective Comparison with Existing Methods*: MSE, SNR and MCD results comparison with different methods on the 8 models are shown in Table IV, VIII, and IX, where the numbers showing the best performance are highlighted in bold.

For parameter settings of competing methods, NLD and LR follow the default settings in the corresponding papers; the rest of the methods require manual parameter tuning, and optimal parameters vary for different models as shown in Table III. Parameters not shown in Table III follow the default setting in the software.

GLR achieves the best results on average in all three metrics and all noise levels. In terms of MSE, GLR outperforms the second best scheme by 0.009, 0.008 and 0.009 for  $\sigma = 0.02, 0.03, 0.04$ ; for SNR, GLR outperforms the second best by 0.72 dB, 0.96 dB, 0.75 dB for  $\sigma = 0.02, 0.03, 0.04$ ; for MCD, GLR outperforms the second best by 0.013, 0.013, 0.011 for  $\sigma = 0.02, 0.03, 0.04$ . APSS and MRPCA are usually the second and the third best among different methods. APSS never achieves the best result for one single model, but on average outperforms others because the local sphere fitting provides stable results. For MRPCA, it sometimes outperforms

GLR but on average is only ranked third because of the unstable performance since the sparsity regularization is likely to generate extra features [6]. In contrast, the proposed GLR not only has stable performance due to the robustness to high noise level, but also outperforms the other schemes overall, validating the effectiveness of LDMM. The patch-similarity based LR is not among the top methods because the patch extraction procedure causes fine detail lose as discussed in Section II, but outperforms the non-local means based NLD, validating the effectiveness of using patch self-similarity.

4) *Visual Comparison with Existing Methods*: Here we demonstrate the results using the model *Daratech* in Fig. 5(a) with  $\sigma = 0.02$  shown in Fig. 4(a). For better visualization, we demonstrate surfaces created from the point clouds with screened Poisson surface reconstruction algorithm [43], and colorize the points using the mean curvature calculated from APSS implemented in MeshLab software.

The surface reconstruction of GLR denoising results after 1st and 3rd iteration are shown in Fig. 4(b) and (c), which demonstrate the iterative recovery of the point cloud. The result converges fast and we do not show the result after iteration 3 since it already converges.

The surface patches in black and blue rectangles are enlarged and placed at the upper-right and lower-right corners to show structural details. The underlying plane (with curvature in green) and fold (with curvature in blue) are gradually recovered, smoothing out the noise on the plane while maintaining the edges.

The comparison with other schemes is shown in Fig. 5. APSS in Fig. 5(b) generates relatively smoother surface than others as shown in the blue rectangles. However, the underlying true structure is not recovered due to the limitation of local operation, resulting in uneven planes and over-smoothed folds. RIMLS shows similar results as APSS thus is not shown in Fig. 5.

We observe that NLD in Fig. 5(d) has similar results as APSS and RIMLS, since the features used for similarity computation in NLD are based on the polynomial coefficients of the MLS surface. Moreover, NLD only takes one pass instead of multiple iterations since more iterations worsen its result as reported in [18], so the noise is not satisfactorily removed. Though NLD and GLR both belong to the non-local category of methods, NLD is based on the non-local means scheme and is not collaboratively denoising the patches, thus also suffers from the drawback of local operation.

AWLOP results in Fig. 5(c) have non-negligible noise. AWLOP is based on normal estimation, so the results indicate that AWLOP fails to estimate the normal at high noise level, and the noisy features may be regarded as sharp features and preserved.

MRPCA in Fig. 5(e) is not providing satisfying results, where the fold is already smoothed out but the plane is still uneven as shown in the blur rectangle. LR in Fig. 5(f) generates relatively smoother planes than others shown in the blue rectangle but noise is still not fully removed. For the proposed GLR in Fig. 4(c), the result is visually better, preserving the plane and folding structures without over-smoothing.

TABLE III  
PARAMETER SETTING OF COMPETING METHODS FOR DIFFERENT MODELS AND NOISE LEVELS

Methods	Parameters	Parameter Setting for Different $\sigma$ 0.02   0.03   0.04								
		Anchor	Bimba	Bunny	Daratech	DC	Fandisk	Gargoyle	Lordquas	
APSS	filter scale	5   5   5	5   10   10	5   5   5	3   3   3	5   5   5	4   5   8	4   4   4	6   6   6	
RIMLS	filter scale	7   7   7	5   12   12	5   5   5	3   3   3	5   5   5	5   8   8	5   5   5	6   6   6	
AWLOP	repulsion force	0.3   0.3   0.3	0.3   0.5   0.5	0.3   0.3   0.3	0.3   0.3   0.3	0.3   0.3   0.3	0.3   0.3   0.3	0.3   0.3   0.3	0.3   0.3   0.5	
	iteration	2   2   2	2   10   10	2   2   2	2   2   2	2   2   2	2   2   2	2   2   2	2   2   10	
MRPCA	data fitting	1   1   1	1   4   4	1   1   1	1   1   1	0.01   0.01   0.01	1   1   1	1   1   1	1   1   1	
	iteration	6   6   6	6   1   1	6   6   6	1   1   1	2   2   2	6   6   6	2   2   2	3   3   3	

TABLE IV  
MSE RESULTS OF DIFFERENT METHODS ON SMALL DATASET WITH THREE NOISE LEVELS

Noise level	Methods	Anchor	Bimba	Bunny	Daratech	DC	Fandisk	Gargoyle	Lordquas	Average
$\sigma = 0.02$	Noisy	0.259	0.0191	0.247	0.245	0.237	0.0258	0.257	0.224	0.189
	APSS	0.208	0.0131	0.198	0.203	0.186	0.0201	0.208	0.171	0.151
	RIMLS	0.212	0.0169	0.208	0.209	0.198	0.0196	0.217	0.183	0.158
	AWLOP	0.237	<b>0.0110</b>	0.223	0.228	0.211	0.0191	0.230	0.196	0.169
	NLD	0.231	0.0174	0.220	0.222	0.206	0.0208	0.230	0.190	0.167
	MRPCA	0.202	0.0154	0.213	0.225	0.189	<b>0.0164</b>	0.215	0.171	0.156
	LR	0.228	0.0133	0.220	0.213	0.206	0.0173	0.240	0.180	0.165
	GLR	<b>0.189</b>	0.0120	<b>0.183</b>	<b>0.197</b>	<b>0.177</b>	0.0173	<b>0.202</b>	<b>0.162</b>	<b>0.142</b>
$\sigma = 0.03$	Noisy	0.321	0.0257	0.309	0.304	0.292	0.0326	0.319	0.274	0.235
	APSS	0.238	0.0196	0.228	0.242	0.210	0.0234	0.239	0.188	0.173
	RIMLS	0.244	0.0213	0.241	0.255	0.225	0.0252	0.251	0.203	0.183
	AWLOP	0.278	<b>0.0133</b>	0.266	0.264	0.246	0.0218	0.270	0.226	0.198
	NLD	0.265	0.0245	0.255	0.258	0.235	0.0285	0.262	0.217	0.193
	MRPCA	0.230	0.0233	0.238	0.262	0.210	0.0239	0.241	0.187	0.177
	LR	0.246	0.0209	0.237	0.252	0.221	0.0210	0.257	0.193	0.181
	GLR	<b>0.217</b>	0.0147	<b>0.217</b>	<b>0.238</b>	<b>0.203</b>	<b>0.0190</b>	<b>0.233</b>	<b>0.176</b>	<b>0.165</b>
$\sigma = 0.04$	Noisy	0.372	0.0324	0.356	0.348	0.338	0.0391	0.368	0.318	0.271
	APSS	0.254	0.0200	0.244	0.282	0.227	0.0289	0.262	0.201	0.190
	RIMLS	0.263	0.0250	0.266	0.308	0.254	0.0314	0.277	0.219	0.205
	AWLOP	0.306	<b>0.0151</b>	0.291	0.286	0.270	0.0240	0.297	0.218	0.213
	NLD	0.297	0.0316	0.285	0.295	0.269	0.0372	0.294	0.252	0.220
	MRPCA	0.242	0.0306	0.248	0.288	<b>0.223</b>	0.0345	<b>0.257</b>	0.199	0.190
	LR	0.259	0.0313	0.249	0.283	0.234	0.0297	0.269	0.204	0.195
	GLR	<b>0.228</b>	0.0175	<b>0.234</b>	<b>0.276</b>	0.228	<b>0.0229</b>	<b>0.257</b>	<b>0.187</b>	<b>0.181</b>

Denoising results of the *Fandisk* model are shown in Fig. 6 with noise level  $\sigma = 0.02$ . The corner part is highlighted by a black rectangle, enlarged and placed at the lower-right corner. APSS result is over-smoothed, AWLOP and NLD do not show competitive results, and MRPCA generates extra surface as shown in the black rectangle. The patch-based LR and GLR is visually better, without over-smoothing or extra feature generated.

We further compare GLR approach using combinatorial and normalized Laplacian matrix for the regularization. As discussed in Section IV-B3, normalized Laplacian cannot handle constant signal, *e.g.*, a flat surface. This is consistent with the visual comparison in Fig. 7, where the surface reconstruction of the resulting point cloud is colorized by distance from the ground truth surface. Normalized Laplacian cannot even denoise a flat surface with obvious error (colored in blue), while combinatorial Laplacian preserves both the smooth surface and the sharp edges. For numerical evaluation in term of MSE, combinatorial Laplacian outperforms normalized Laplacian by 0.014, 0.020, 0.080 for  $\sigma = 0.02, 0.03, 0.04$ .

### C. Generalization to ShapeNetCore Dataset

We now test the parameter setting learned in Section VI-B1 and VI-B2 with ShapeNetCore dataset [41]. ShapeNetCore dataset is a subset of ShapeNet dataset containing 55 object

categories with 52491 unique 3D models. The 10 categories with the largest number of models are used. We then randomly select 10 models from each category for testing, so 100 models are used in total. Moreover, the 3D models in ShapeNetCore dataset are low poly meshes, so we sample approximately 30000 points on each mesh to obtain the point cloud using Poisson-disk sampling [44].

We add Gaussian noise with  $\sigma = 0.02, 0.03, 0.04$  to the models, then apply different denoising methods. The parameter setting for GLR is the same as used in Section VI-B. For APSS and RIMLS, we try each filter scale in  $\{5, 6, 7, 8, 9, 10\}$  and choose the one with the best result. For other methods, we empirically set the parameters as shown in Table V based on the results of previous 8 models since it is too time-consuming to tune parameters for each of the 100 models. Parameters not in Table V follow the default setting.

The MSE, SNR and MCD results are compared with competing methods in Table X, XI and XII where GLR provides the best results, and the patch-based LR is the second best. At high noise level, *e.g.*, Fig. 8(b), the points distract from the surface and tend to fill the bulk of the object, so for methods based on plane fitting, the denoising leads to erroneous results, *e.g.*, APSS in Fig. 8(c). RIMLS provides similar results as APSS thus is not shown in Fig. 8. The noise in results of AWLOP and MRPCA in Fig. 8(d) and (f) is not fully removed

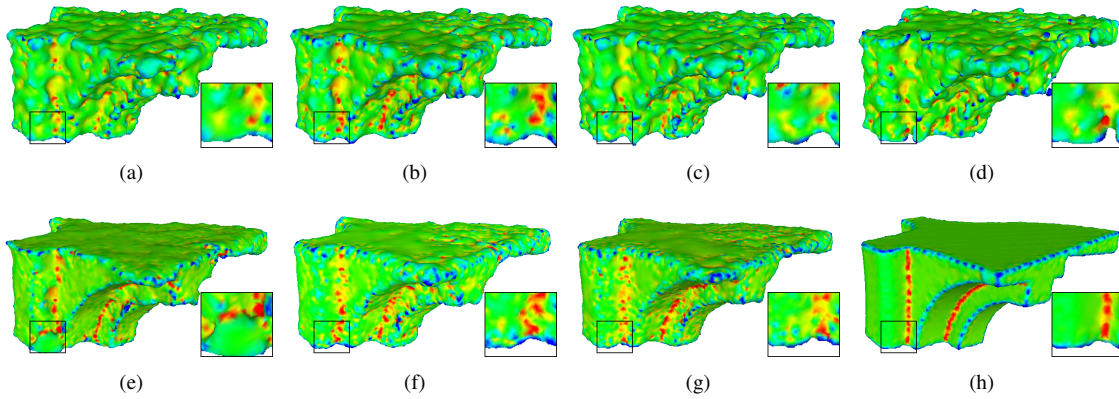


Fig. 6. *Fandisk* model ( $\sigma = 0.02$ ) illustration. Surface reconstruction with (a) noisy input, denoising results of (b) APSS, (c) AWLOP, (d) NLD, (e) MRPCA, (f) LR, (g) proposed GLR, and (h) ground truth, colored by mean curvature.

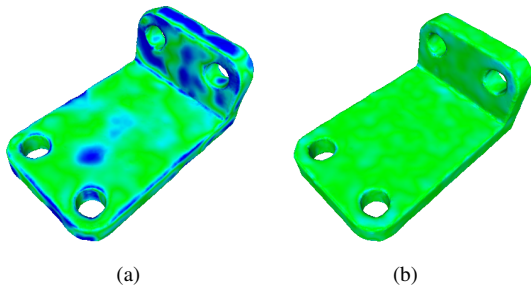


Fig. 7. Surface reconstruction of denoising results of (a) normalized Laplacian regularization and (b) combinatorial Laplacian regularization, where the surface is colored by the distance from the ground truth surface (green means zero error, while blue means large error).

TABLE V  
PARAMETER SETTING OF COMPETING METHODS FOR SHAPENETCORE DATASET WITH DIFFERENT NOISE LEVELS

Methods	Parameters	$\sigma$ 0.02	0.03	0.04
APSS	filter scale	exhaustive search {5,6,7,8,9,10}		
RIMLS	filter scale	exhaustive search {5,6,7,8,9,10}		
AWLOP	repulsion force	0.3	0.3	0.3
	iteration	2	2	2
MRPCA	data fitting	1	1	1
	iteration	6	6	6

with noticeable outliers. NLD in Fig. 8(e) provides smooth results without outliers, which demonstrates the robustness of non-local means filtering against the above approaches at high noise level. Patch-based LR and GLR in Fig. 8(g) and (h) provide the best results, where the shape of the rifle model is well preserved, validating the effectiveness of patch-similarity based filtering. However, LR tends to over-smooth the model and fine details are lost during patch extraction procedure, while the proposed GLR preserves the salient features without over-smoothing. In sum, the generalization to ShapeNetCore dataset validates the robustness of parameter setting in Section VI-B as well as the superiority of patch-based filtering over other approaches.

#### D. Complexity Analysis

The computational complexity of different algorithms are summarized in Table VI.  $N$  is the number of points,  $r$  is the number of iterations of implementing the algorithm since all algorithms except NLD and LR adopt iterative restoration,  $K$  is the neighborhood size chosen for different operation in different schemes. Parameters used in specific methods are explained along with the complexity. The parameter ranges in Table VI are suggested in the original papers.

As shown in Table VI, APSS, RIMLS and NLD have the lowest complexity. MRPCA and GLR are of similar complexity; MRPCA's can be higher due to large  $K$  and  $t$ . The complexity of LR is high due to complexity in solving low-rank matrix factorization for dictionary learning. GLR have relatively high complexity, but provides the best performance as shown in the above evaluation, so GLR is favorable if the requirement for denoising accuracy is high.

We additionally include the runtime of different methods implemented on Intel i7-8550U CPU at 1.80GHz and 8GB RAM. Since the methods are implemented with different programming language and C++ is known to far surpass the speed of Matlab [45], we cannot directly use the runtime for complexity comparison. Nevertheless, the runtime of NLD is more than 10 times that of APSS while the complexity is approximately the same as APSS, thus if implemented in C++, the runtime of GLR can be reduced by 10 times potentially.

## VII. CONCLUSION

In this paper, we propose a graph Laplacian regularization based 3D point cloud denoising algorithm. To utilize the self-similarity among surface patches, we adopt the low-dimensional manifold prior, and collaboratively denoise the patches by minimizing the manifold dimension. To compute manifold dimension with discrete patch observations, we approximate the manifold dimension with a graph Laplacian regularizer, and construct the patch graph with a new measure for the discrete patch distance. The proposed scheme is shown to have graph spectral low-pass filtering interpretation and numerical stability in solving the linear equation system, and efficient to solve with methods like PCG. Experimental results



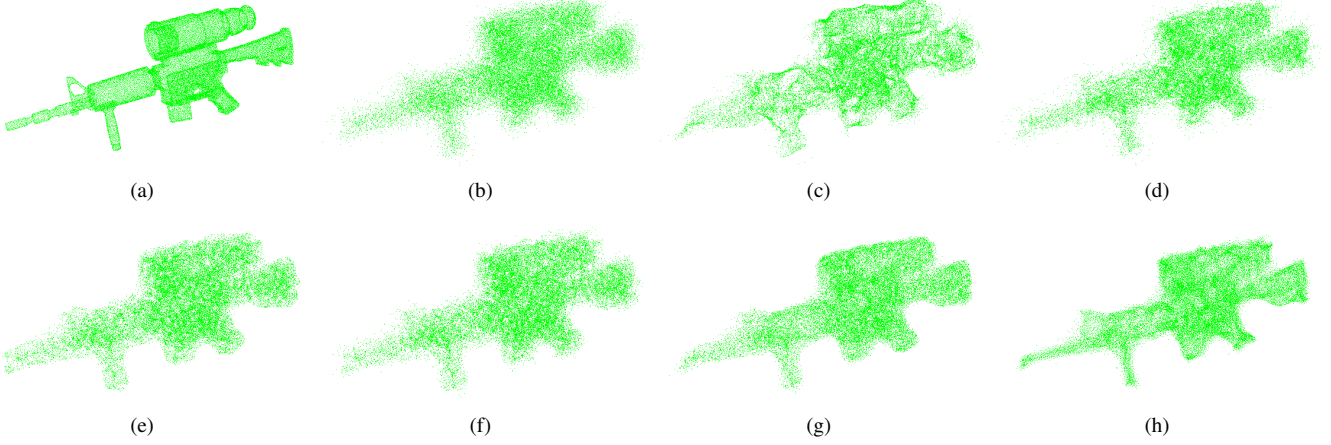


Fig. 8. Rifle model from ShapeNetCore dataset. (a) ground truth, (b) noisy input with  $\sigma = 0.04$ , denoising results of (c) APSS, (d) AWLOP, (e) NLD, (f) MRPCA, (g) LR, (h) GLR.

TABLE VI  
TIME COMPLEXITY SUMMARY OF DIFFERENT SCHEMES

Method	Complexity
APSS	$O(rKN \log N)$ , $r \leq 15$ , $K \in (16, 100)$
RIMLS	Same as above
AWLOP	$O(r(\sigma_p N^2 + kN \log N))$ , $r \approx 3$ $\sigma_p$ for neighborhood radius $k = 6$ for PCA-based normal estimation
NLD	$O(N + 2KN \log N)$ , $K \approx 20$
MRPCA	$O(r(KN \log N + N \log N + tKN))$ $r \leq 20$ , $K \in (30, 100)$ , $t \in (50, 100)$ is the RPCA solver iteration number
LR	$O(l/\tau(h^2 + M))$ , $l = 50$ is dictionary atom number $h^2 = 16^2$ is patch grid size, $M$ is patch number $\tau = 10^{-5}$ is proximal gradient descent step size
GLR	$O(rKM(\log M + k \log k + k\sqrt{C}))$ , $r \leq 15$ , $M = N/2$ , $K = 16$ , $k = 30$ , $C \leq 1 + 2/\mu \leq 1.921$ ( $\mu \geq 25(\exp(1/12) - 1) \approx 2.173$ )

TABLE VII  
AVERAGE RUNTIME (SEC) ON SHAPENETCORE DATASET AND  
PROGRAMMING LANGUAGE FOR DIFFERENT METHODS

APSS	RIMLS	AWLOP	NLD	MRPCA	LR	GLR
C++	C++	C++	Matlab	C++	Matlab	Matlab
13.8	18.1	21.2	156.4	18.0	464.7	372.2

suggest that our proposal outperforms existing schemes with better structural detail preservation.

## APPENDIX A

Assume that  $\mathcal{M}$  is a Riemannian manifold with boundary, equipped with the probability density function (PDF)  $h(\mathbf{p})$  describing the distribution of the vertices on  $\mathcal{M}$ , and that  $\alpha_i$  belongs to the class of  $\kappa$ -Hölder functions [46] on  $\mathcal{M}$ . Then according to the proof in [47], there exists a constant  $c$  depending only on  $C_r$  such that for  $\kappa \geq 3$  and the weight parameter  $\epsilon = O\left(M^{-\frac{\kappa}{2\kappa+2\delta+\delta^2+\delta\kappa}}\right)$ , where  $\delta$  denotes the manifold dimension, such that<sup>4</sup>

<sup>4</sup>We refer readers to [46] for the uniform convergence result in a more general setting and its corresponding assumptions on  $\mathcal{M}$ ,  $\epsilon$ , and the graph weight kernel function  $\psi(\cdot)$ .

$$\sup \left| \frac{cM^{2\gamma-1}}{\epsilon^{4(1-\gamma)}(M-1)} S_{\mathbf{L}}(\alpha_i) - S_{\Delta}(\alpha_i) \right| = O\left(M^{-\frac{\kappa}{2\kappa+2\delta+\delta^2+\delta\kappa}}\right), \quad (42)$$

where  $S_{\Delta}(\alpha_i)$  is induced by the  $2(1-\gamma)$ -th weighted Laplace-Beltrami operator on  $\mathcal{M}$ , which is given as

$$S_{\Delta}(\alpha_i) = \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})\|_2^2 h(\mathbf{p})^{2(1-\gamma)} d\mathbf{p}. \quad (43)$$

Assuming that the vertices are uniformly distributed on  $\mathcal{M}$ , then

$$\int_{\mathcal{M}} h(\mathbf{p}) d\mathbf{p} = 1, \quad h(\mathbf{p}) = \frac{1}{|\mathcal{M}|}, \quad (44)$$

where  $|\mathcal{M}|$  is the volume of the manifold  $\mathcal{M}$ . For implementation, similar to the setting in [47], we set  $\gamma = 0.5$ , then  $S_{\Delta}(\alpha_i)$  becomes

$$S_{\Delta}(\alpha_i) = \frac{1}{|\mathcal{M}|} \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} \alpha_i(\mathbf{p})\|_2^2 d\mathbf{p}. \quad (45)$$

From (42) and (45), the convergence in (11) is readily obtained by weakening the uniform convergence of (42) to point-wise convergence.

## REFERENCES

- [1] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.
- [2] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3d point cloud sequences," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, 2016.
- [3] S. Chen, D. Tian, C. Feng, A. Vetro, and J. Kovačević, "Fast resampling of three-dimensional point clouds via graphs," *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 666–681, 2018.
- [4] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang, "Surfacenet: An end-to-end 3D neural network for multiview stereopsis," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2326–2334, 2017.
- [5] G. Rosman, A. Dubrovina, and R. Kimmel, "Patch-collaborative spectral point-cloud denoising," *Computer Graphics Forum*, vol. 32, no. 8, pp. 1–12, 2013.
- [6] E. Mattei and A. Castrodad, "Point cloud denoising via moving rpca," *Computer Graphics Forum*, pp. 1–15, 2016.

TABLE VIII  
SNR (dB) RESULTS OF DIFFERENT METHODS ON SMALL DATASET WITH THREE NOISE LEVELS

Noise level	Methods	Anchor	Bimba	Bunny	Daratech	DC	Fandisk	Gargoyle	Lordquas	Average
$\sigma = 0.02$	Noisy	47.41	41.40	51.99	45.85	46.42	34.06	46.91	46.61	45.08
	APSS	49.61	45.13	54.20	47.71	48.83	36.52	49.01	49.27	47.53
	RIMLS	49.41	42.60	53.70	47.44	48.23	36.80	48.57	48.60	46.92
	AWLOP	48.31	<b>46.91</b>	52.98	46.56	47.59	37.06	48.01	47.92	46.92
	NLD	48.53	42.30	53.14	46.82	47.82	36.16	48.01	48.22	46.38
	MRPCA	49.88	43.53	53.45	46.72	48.68	<b>38.55</b>	48.66	49.27	47.34
	LD	48.69	45.06	53.15	47.27	47.83	38.00	47.57	48.78	47.04
	GLR	<b>50.55</b>	46.00	<b>54.95</b>	<b>48.02</b>	<b>49.34</b>	38.05	<b>49.30</b>	<b>49.81</b>	<b>48.25</b>
$\sigma = 0.03$	Noisy	45.25	38.42	49.72	43.70	44.32	31.75	44.75	44.58	42.81
	APSS	48.24	41.17	52.77	46.00	47.64	35.07	47.63	48.34	45.86
	RIMLS	48.00	40.36	52.22	45.46	46.94	34.38	47.12	47.57	45.26
	AWLOP	46.69	<b>45.02</b>	51.24	45.12	46.04	35.71	46.39	46.53	45.34
	NLD	47.16	38.91	51.67	45.34	46.49	33.07	46.68	46.89	44.53
	MRPCA	48.60	39.40	52.32	45.18	47.62	34.83	47.52	48.40	45.48
	LD	47.91	40.52	52.38	45.59	47.10	36.12	46.88	48.09	45.57
	GLR	<b>49.20</b>	44.03	<b>53.28</b>	<b>46.13</b>	<b>47.94</b>	<b>37.09</b>	<b>47.87</b>	<b>49.00</b>	<b>46.82</b>
$\sigma = 0.04$	Noisy	43.78	36.13	48.31	42.34	42.86	29.95	43.31	43.09	41.22
	APSS	47.60	40.94	52.09	44.46	<b>46.84</b>	33.02	46.69	47.68	44.92
	RIMLS	47.27	38.76	51.22	43.58	45.71	32.23	46.14	46.80	43.96
	AWLOP	45.74	<b>43.73</b>	50.32	44.32	45.11	34.77	45.44	46.85	44.54
	NLD	46.02	36.39	50.54	43.98	45.15	30.44	45.53	45.40	42.93
	MRPCA	48.09	36.71	51.93	44.25	47.00	31.19	46.88	47.80	44.23
	LD	47.41	36.50	51.89	44.41	46.54	32.68	46.44	47.52	44.17
	GLR	<b>48.67</b>	42.22	<b>52.51</b>	<b>44.64</b>	46.80	<b>35.20</b>	<b>46.89</b>	<b>48.40</b>	<b>45.67</b>

TABLE IX  
MCD RESULTS OF DIFFERENT METHODS ON SMALL DATASET WITH THREE NOISE LEVELS

Noise level	Methods	Anchor	Bimba	Bunny	Daratech	DC	Fandisk	Gargoyle	Lordquas	Average
$\sigma = 0.02$	Noisy	0.384	0.0268	0.366	0.364	0.350	0.0368	0.380	0.331	0.280
	APSS	0.302	0.0188	0.293	0.300	0.275	0.0294	0.308	0.252	0.222
	RIMLS	0.311	0.0240	0.309	0.310	0.292	0.0287	0.322	0.271	0.233
	AWLOP	0.353	<b>0.0161</b>	0.332	0.339	0.313	0.0280	0.342	0.292	0.252
	NLD	0.339	0.0247	0.325	0.327	0.304	0.0304	0.340	0.281	0.246
	MRPCA	0.289	0.0219	0.316	0.331	0.278	<b>0.0239</b>	0.319	0.251	0.229
	LD	0.330	0.0190	0.326	0.313	0.303	0.0254	0.355	0.264	0.242
	GLR	<b>0.272</b>	0.0174	<b>0.272</b>	<b>0.290</b>	<b>0.261</b>	0.0252	<b>0.299</b>	<b>0.238</b>	<b>0.209</b>
$\sigma = 0.03$	Noisy	0.475	0.0356	0.456	0.449	0.430	0.0453	0.469	0.402	0.345
	APSS	0.348	0.0274	0.338	0.358	0.309	0.0338	0.353	0.277	0.255
	RIMLS	0.360	0.0297	0.357	0.379	0.333	0.0361	0.372	0.300	0.271
	AWLOP	0.415	<b>0.0194</b>	0.395	0.392	0.365	0.0320	0.401	0.335	0.294
	NLD	0.391	0.0341	0.377	0.380	0.347	0.0405	0.388	0.321	0.285
	MRPCA	0.331	0.0325	0.353	0.386	0.310	0.0343	0.356	0.274	0.260
	LD	0.359	0.0293	0.352	0.372	0.326	0.0304	0.379	0.284	0.266
	GLR	<b>0.312</b>	0.0209	<b>0.322</b>	<b>0.353</b>	<b>0.300</b>	<b>0.0276</b>	<b>0.345</b>	<b>0.259</b>	<b>0.242</b>
$\sigma = 0.04$	Noisy	0.545	0.0445	0.521	0.510	0.494	0.0535	0.539	0.462	0.396
	APSS	0.375	0.0278	0.362	0.417	<b>0.336</b>	0.0409	0.388	0.297	0.280
	RIMLS	0.389	0.0340	0.395	0.454	0.376	0.0442	0.410	0.325	0.303
	AWLOP	0.456	<b>0.0220</b>	0.432	0.425	0.400	0.0351	0.441	0.322	0.317
	NLD	0.439	0.0433	0.421	0.435	0.397	0.0514	0.434	0.372	0.324
	MRPCA	0.351	0.0421	0.367	0.424	0.330	0.0479	0.380	0.293	0.279
	LD	0.380	0.0430	0.369	0.418	0.345	0.0419	0.397	0.302	0.287
	GLR	<b>0.334</b>	0.0248	<b>0.347</b>	<b>0.411</b>	0.337	<b>0.0330</b>	<b>0.379</b>	<b>0.277</b>	<b>0.268</b>

- [7] Y. Sun, S. Schaefer, and W. Wang, "Denoising point sets via  $l_0$  minimization," *Computer Aided Geometric Design*, vol. 35, pp. 2–15, 2015.
- [8] Y. Zheng, G. Li, S. Wu, Y. Liu, and Y. Gao, "Guided point cloud denoising via sharp feature skeletons," *The Visual Computer*, pp. 1–11, 2017.
- [9] G. Guennebaud and M. Gross, "Algebraic point set surfaces," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 23, 2007.
- [10] 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.
- [11] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 22, 2007.
- [12] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, p. 9, 2013.
- [13] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, " $l_1$ -sparse reconstruction of sharp point set surfaces," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 5, p. 135, 2010.
- [14] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao, "A review of algorithms for filtering the 3D point cloud," *Signal Processing: Image Communication*, 2017.
- [15] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition (CVPR), 2005 IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 60–65.
- [16] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions*

TABLE X  
MSE ( $\times 10^{-3}$ ) RESULTS OF DIFFERENT METHODS FOR 10 CATEGORIES OF SHAPENetCORE DATASET

Noise Level	Methods	plane	bench	car	chair	lamp	speaker	rifle	sofa	table	vessel	Average
0.02	Noisy	4.988	6.206	6.188	6.709	5.509	6.656	4.911	6.919	6.283	5.605	5.997
	APSS	4.059	4.783	4.380	4.693	4.032	<b>3.643</b>	5.066	4.370	5.204	4.060	4.429
	RIMLS	4.718	5.479	4.904	5.446	4.873	4.134	5.236	4.838	5.817	4.927	5.037
	AWLOP	3.890	5.046	5.196	5.690	4.034	5.424	3.646	5.951	5.237	4.342	4.846
	NLD	4.347	5.096	4.902	5.247	4.700	5.091	4.505	5.192	5.041	4.819	4.894
	MRPCA	4.413	5.126	4.915	5.073	4.670	4.772	4.560	5.095	5.229	4.818	4.867
	LR	<b>3.434</b>	4.802	4.389	5.159	<b>3.537</b>	3.689	<b>3.199</b>	4.760	4.756	<b>3.734</b>	4.146
	GLR	3.489	<b>4.368</b>	<b>4.067</b>	<b>4.355</b>	3.714	3.752	3.640	<b>4.294</b>	<b>4.565</b>	3.789	<b>4.003</b>
0.03	Noisy	6.393	7.848	7.829	8.540	7.302	8.707	6.627	8.858	7.933	7.210	7.725
	APSS	6.354	6.687	5.636	6.377	5.705	4.886	8.149	5.832	6.814	6.048	6.249
	RIMLS	7.237	7.441	6.907	7.963	7.454	6.256	7.579	6.886	7.585	7.592	7.290
	AWLOP	5.227	6.899	7.108	7.824	5.762	7.779	4.926	8.242	6.942	6.076	6.679
	NLD	5.946	7.091	6.884	7.351	6.678	7.459	6.334	7.584	6.963	6.641	6.893
	MRPCA	6.034	7.082	6.873	7.211	6.649	7.145	6.403	7.409	7.099	6.709	6.861
	LR	<b>4.229</b>	5.252	<b>4.638</b>	5.498	4.636	<b>4.357</b>	5.144	<b>5.412</b>	5.495	<b>4.545</b>	4.920
	GLR	4.274	<b>5.234</b>	4.808	<b>5.496</b>	<b>4.518</b>	4.709	<b>4.553</b>	5.452	<b>5.297</b>	4.650	<b>4.899</b>
0.04	Noisy	7.784	9.433	9.443	10.259	9.127	10.698	8.443	10.650	9.619	8.790	9.425
	APSS	9.020	8.626	7.474	9.173	8.457	6.943	10.083	7.983	9.031	9.533	8.632
	RIMLS	9.073	10.179	9.084	10.545	9.790	8.836	9.921	10.268	10.070	9.894	9.766
	AWLOP	6.757	8.748	8.948	9.807	7.833	10.100	6.700	10.307	8.864	7.953	8.602
	NLD	7.431	8.856	8.773	9.378	8.640	9.788	8.182	9.734	8.885	8.391	8.806
	MRPCA	7.525	8.858	8.786	9.288	8.612	9.520	8.245	9.622	8.961	8.448	8.786
	LR	5.799	<b>6.244</b>	<b>5.550</b>	<b>6.469</b>	6.441	<b>5.638</b>	7.456	<b>6.396</b>	6.229	6.209	6.243
	GLR	<b>5.320</b>	6.432	6.029	6.746	<b>5.809</b>	5.993	<b>5.494</b>	6.801	<b>6.025</b>	<b>5.880</b>	<b>6.053</b>

TABLE XI  
SNR (dB) RESULTS OF DIFFERENT METHODS FOR 10 CATEGORIES OF SHAPENetCORE DATASET

Noise Level	Methods	plane	bench	car	chair	lamp	speaker	rifle	sofa	table	vessel	Average
0.02	Noisy	36.58	38.50	38.40	37.42	40.95	38.62	38.06	37.70	39.38	37.59	38.32
	APSS	38.71	41.23	41.92	41.04	44.19	<b>44.66</b>	37.87	42.40	41.29	40.94	41.42
	RIMLS	37.16	39.90	40.78	39.55	42.22	43.39	37.49	41.37	40.17	39.01	40.10
	AWLOP	39.06	40.60	40.15	39.14	44.14	40.75	41.04	39.23	41.26	40.13	40.55
	NLD	37.93	40.45	40.72	39.85	42.51	41.28	38.90	40.55	41.55	39.08	40.28
	MRPCA	37.78	40.40	40.69	40.19	42.56	41.92	38.77	40.75	41.18	39.08	40.33
	LR	<b>40.43</b>	41.21	41.92	40.16	<b>45.61</b>	44.59	<b>42.31</b>	41.57	42.22	<b>41.88</b>	42.19
	GLR	40.15	<b>42.06</b>	<b>42.61</b>	<b>41.75</b>	44.86	44.33	41.01	<b>42.51</b>	<b>42.61</b>	41.50	<b>42.34</b>
0.03	Noisy	34.13	36.17	36.06	35.02	38.14	35.94	35.10	35.24	37.05	35.11	35.80
	APSS	34.27	37.97	39.42	37.99	40.73	41.76	33.26	39.53	38.60	37.02	38.05
	RIMLS	33.03	36.82	37.43	35.93	38.08	39.47	33.81	37.87	37.62	34.73	36.48
	AWLOP	36.14	37.47	37.02	35.93	40.58	37.11	38.11	35.96	38.46	36.79	37.36
	NLD	34.83	37.17	37.34	36.49	39.02	37.45	35.53	36.78	38.34	35.92	36.89
	MRPCA	34.69	37.18	37.35	36.68	39.06	37.88	35.43	37.01	38.14	35.82	36.92
	LR	<b>38.26</b>	<b>40.29</b>	<b>41.34</b>	<b>39.46</b>	42.70	<b>42.86</b>	37.64	<b>40.30</b>	40.78	<b>39.73</b>	<b>40.33</b>
	GLR	38.12	40.23	40.92	39.39	<b>42.90</b>	42.04	<b>38.79</b>	40.10	<b>41.11</b>	39.43	40.30
0.04	Noisy	32.20	34.35	34.21	33.22	35.93	33.90	32.73	33.42	35.14	33.18	33.83
	APSS	30.95	35.43	36.68	34.46	36.88	38.51	31.12	36.42	35.86	32.58	34.89
	RIMLS	30.78	33.69	34.76	33.11	35.32	36.05	31.18	33.92	34.87	32.10	33.58
	AWLOP	33.59	35.10	34.74	33.68	37.50	34.49	35.02	33.74	35.98	34.15	34.80
	NLD	32.64	34.97	34.93	34.08	36.47	34.76	33.03	34.30	35.92	33.63	34.47
	MRPCA	32.52	34.97	34.92	34.18	36.50	35.03	32.95	34.42	35.83	33.56	34.49
	LR	35.10	<b>38.47</b>	<b>39.52</b>	<b>37.79</b>	39.51	<b>40.33</b>	33.99	38.53	39.46	36.66	37.94
	GLR	<b>35.81</b>	38.14	38.58	37.28	<b>40.38</b>	39.58	<b>36.89</b>	<b>37.80</b>	<b>39.78</b>	<b>37.00</b>	<b>38.12</b>

on Image Processing, vol. 16, no. 8, pp. 2080–2095, 2007.

- [17] R.-f. Wang, W.-z. Chen, S.-y. Zhang, Y. Zhang, and X.-z. Ye, "Similarity-based denoising of point-sampled surfaces," *Journal of Zhejiang University-Science A*, vol. 9, no. 6, pp. 807–815, 2008.
- [18] J.-E. Deschaud and F. Goulette, "Point cloud non local denoising using local surface descriptor similarity," *IAPRS*, vol. 38, no. 3A, pp. 109–114, 2010.
- [19] K. Sarkar, F. Bernard, K. Varanasi, C. Theobalt, and D. Stricker, "Structured low-rank matrix factorization for point-cloud denoising," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 444–453.
- [20] S. Osher, Z. Shi, and W. Zhu, "Low dimensional manifold model for image processing," *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1669–1690, 2017.
- [21] G. Peyré, "Manifold models for signals and images," *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 249–260, 2009.
- [22] —, "A review of adaptive image representations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 896–911, 2011.
- [23] Z. Shi, S. Osher, and W. Zhu, "Generalization of the weighted nonlocal laplacian in low dimensional manifold model," *Journal of Scientific Computing*, vol. 75, no. 2, pp. 638–656, 2018.
- [24] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, May 2013.
- [25] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions*



TABLE XII  
MCD ( $\times 10^{-3}$ ) RESULTS OF DIFFERENT METHODS FOR 10 CATEGORIES OF SHAPENETCORE DATASET

Noise Level	Methods	plane	bench	car	chair	lamp	speaker	rifle	sofa	table	vessel	Average
0.02	Noisy	6.84	8.54	8.63	9.28	7.54	9.16	6.51	9.58	8.69	7.65	8.24
	APSS	5.73	6.83	6.35	6.78	5.69	<b>5.34</b>	6.77	6.40	7.42	5.78	6.31
	RIMLS	6.56	7.71	7.06	7.77	6.76	6.02	6.98	7.05	8.19	6.89	7.10
	AWLOP	5.51	7.15	7.43	8.07	5.71	7.70	4.99	8.45	7.45	6.14	6.86
	NLD	6.09	7.24	7.07	7.52	6.56	7.28	6.05	7.51	7.21	6.74	6.93
	MRPCA	6.16	7.25	7.06	7.27	6.51	6.84	6.11	7.36	7.43	6.72	6.87
	LR	<b>4.92</b>	6.84	6.34	7.35	<b>5.05</b>	5.36	<b>4.45</b>	6.89	6.82	<b>5.35</b>	5.94
	GLR	4.98	<b>6.26</b>	<b>5.93</b>	<b>6.31</b>	5.27	5.47	4.97	6.28	<b>6.57</b>	5.41	<b>5.75</b>
0.03	Noisy	8.54	10.47	10.61	11.45	9.74	11.59	8.58	11.83	10.63	9.57	10.30
	APSS	8.60	9.17	7.98	8.92	7.80	6.96	10.52	8.29	9.37	8.25	8.59
	RIMLS	9.66	10.06	9.59	10.87	10.00	8.66	9.83	9.62	10.34	10.15	9.88
	AWLOP	7.15	9.38	9.76	10.62	7.86	10.52	6.53	11.15	9.50	8.24	9.07
	NLD	8.03	9.65	9.55	10.10	9.02	10.20	8.25	10.44	9.57	8.94	9.37
	MRPCA	8.13	9.61	9.50	9.91	8.97	9.78	8.33	10.18	9.70	9.01	9.31
	LR	<b>5.93</b>	7.39	<b>6.69</b>	7.77	6.45	<b>6.25</b>	6.83	<b>7.72</b>	7.76	<b>6.37</b>	6.92
	GLR	5.96	<b>7.34</b>	6.90	<b>7.75</b>	<b>6.28</b>	6.71	<b>6.07</b>	7.76	<b>7.48</b>	6.48	<b>6.87</b>
0.04	Noisy	10.23	12.32	12.53	13.47	11.98	13.93	10.78	13.89	12.61	11.45	12.32
	APSS	11.87	11.46	10.30	12.33	11.24	9.48	12.88	10.92	12.01	12.51	11.50
	RIMLS	11.92	13.36	12.25	13.95	12.89	11.78	12.68	13.62	13.23	12.88	12.86
	AWLOP	9.01	11.55	11.95	12.96	10.40	13.25	8.68	13.52	11.76	10.47	11.35
	NLD	9.83	11.71	11.81	12.52	11.42	12.96	10.48	12.94	11.81	11.01	11.65
	MRPCA	9.94	11.70	11.81	12.39	11.38	12.62	10.56	12.78	11.88	11.08	11.61
	LR	7.87	<b>8.60</b>	7.86	<b>8.98</b>	8.70	<b>7.88</b>	9.64	<b>8.93</b>	8.65	8.41	8.55
	GLR	<b>7.25</b>	8.79	<b>8.43</b>	9.29	<b>7.88</b>	8.31	<b>7.20</b>	9.40	<b>8.38</b>	<b>7.96</b>	<b>8.29</b>

- on Visualization and Computer Graphics, vol. 9, no. 1, pp. 3–15, 2003.
- [26] G. Guennebaud, M. Hermann, and M. Gross, “Dynamic sampling and rendering of algebraic point set surfaces,” *Computer Graphics Forum*, vol. 27, no. 2, pp. 653–662, 2008.
- [27] R. B. Rusu, N. Blodow, Z. Marton, A. Soos, and M. Beetz, “Towards 3D object maps for autonomous household robots,” in *Intelligent Robots and Systems (IROS), 2007 IEEE/RSJ International Conference on*. IEEE, 2007, pp. 3191–3198.
- [28] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, “Consolidation of unorganized point clouds for surface reconstruction,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 176, 2009.
- [29] Z. Zha, X. Yuan, T. Yue, and J. Zhou, “From rank estimation to rank approximation: Rank residual constraint for image denoising,” *arXiv preprint arXiv:1807.02504*, 2018.
- [30] Z. Zha, X. Zhang, Q. Wang, Y. Bai, L. Tang, and X. Yuan, “Group sparsity residual with non-local samples for image denoising,” in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE, 2018, pp. 1353–1357.
- [31] Z. Zha, X. Zhang, Q. Wang, Y. Bai, and L. Tang, “Image denoising using group sparsity residual and external nonlocal self-similarity prior,” in *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2956–2960.
- [32] Z. Zha, X. Liu, Z. Zhou, X. Huang, J. Shi, Z. Shang, L. Tang, Y. Bai, Q. Wang, and X. Zhang, “Image denoising via group sparsity residual constraint,” in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1787–1791.
- [33] Q. Wang, X. Zhang, Y. Wu, L. Tang, and Z. Zha, “Nonconvex weighted  $\ell_p$  minimization based group sparse representation framework for image denoising,” *IEEE Signal Processing Letters*, vol. 24, no. 11, pp. 1686–1690, 2017.
- [34] X. Liu, G. Cheung, X. Wu, and D. Zhao, “Random walk graph laplacian-based smoothness prior for soft decoding of JPEG images,” *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 509–524, 2017.
- [35] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 2012.
- [36] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, “Mesh: Measuring errors between surfaces using the hausdorff distance,” in *Proceedings. IEEE International Conference on Multimedia and Expo*, vol. 1. IEEE, 2002, pp. 705–708.
- [37] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*. Springer Science & Business Media, 2009, vol. 317.
- [38] M.-P. Dubuisson and A. K. Jain, “A modified hausdorff distance for object matching,” in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1. IEEE, 1994, pp. 566–568.
- [39] J. R. Shewchuk *et al.*, “An introduction to the conjugate gradient method without the agonizing pain,” 1994.
- [40] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, “Meshlab: An open-source mesh processing tool,” in *Eurographics Italian Chapter Conference*, vol. 2008, 2008, pp. 129–136.
- [41] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, “ShapeNet: An Information-Rich 3D Model Repository,” Stanford University–Princeton University–Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.
- [42] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, “The farthest point strategy for progressive image sampling,” *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.
- [43] M. Kazhdan and H. Hoppe, “Screened Poisson surface reconstruction,” *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, p. 29, 2013.
- [44] M. Corsini, P. Cignoni, and R. Scopigno, “Efficient and flexible sampling with blue noise properties of triangular meshes,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 914–924, 2012.
- [45] T. Andrews, “Computation time comparison between matlab and c++ using launch windows,” 2012.
- [46] M. Hein, “Uniform convergence of adaptive graph-based regularization,” *Lecture Notes in Computer Science*, vol. 4005, p. 50, 2006.
- [47] J. Pang and G. Cheung, “Graph Laplacian regularization for image denoising: Analysis in the continuous domain,” *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1770–1785, 2017.