3D Point Cloud Denoising via Bipartite Graph Approximation and Reweighted Graph Laplacian

Chinthaka Dinesh, *Student Member, IEEE,* Gene Cheung, *Senior Member, IEEE,* and Ivan V. Bajić, *Senior Member, IEEE*

***Abstract*—Point cloud is a collection of 3D coordinates that are discrete geometric samples of an object's 2D surfaces. Imperfection in the acquisition process means that point clouds are often corrupted with noise. Building on recent advances in graph signal processing, we design local algorithms for 3D point cloud denoising. Specifically, we design a reweighted graph Laplacian regularizer (RGLR) for surface normals and demonstrate its merits in rotation invariance, promotion of piecewise smoothness, and ease of optimization. Using RGLR as a signal prior, we formulate an optimization problem with a general ℓ_p-norm fidelity term that can explicitly model two types of independent noise: small but non-sparse noise (using ℓ_2 fidelity term) and large but sparser noise (using ℓ_1 fidelity term).**

To establish a linear relationship between normals and 3D point coordinates, we first perform bipartite graph approximation to divide the point cloud into two disjoint node sets (red and blue). We then optimize the red and blue nodes' coordinates alternately. For ℓ_2-norm fidelity term, we iteratively solve an unconstrained quadratic programming (QP) problem, efficiently computed using conjugate gradient with a bounded condition number to ensure numerical stability. For ℓ_1-norm fidelity term, we iteratively minimize an ℓ_1-ℓ_2 cost function sing accelerated proximal gradient (APG), where a good step size is chosen via Lipschitz continuity analysis. Finally, we propose simple mean and median filters for flat patches of a given point cloud to estimate the noise variance given the noise type, which in turn is used to compute a weight parameter trading off the fidelity term and signal prior in the problem formulation. Extensive experiments show state-of-the-art denoising performance among local methods using our proposed algorithms.

***Index Terms*—3D point cloud, graph signal processing, denoising, convex optimization**

I. Introduction

A recent popular 3D geometric signal representation—for a wide range of image rendering applications such as augmented reality and immersive telepresence—is *point cloud* [1]. It consists of a collection of 3D points that are geometric samples of 2D surfaces of a physical object in 3D space, sometimes with attributes such as color. A point cloud can be acquired directly using active light sensors such as Microsoft Kinect©, or induced indirectly via stereo-matching algorithms [2]. However, imperfection in the acquisition process means that the point cloud is often corrupted with non-negligible noise. We address the point cloud denoising problem in this paper.

Signal smoothness is commonly used as a prior for image restoration in various forms, such as high-frequency-based Tikhonov regularization, total variation (TV), graph Laplacian regularizer, graph TV etc, [3]–[5]. However, two inherent difficulties make the proper definition of smoothness more challenging for point clouds. First, though 3D points in a point cloud—unlike regularly sampled pixels in a 2D digital image—may be irregularly spaced apart, the defined notion should nonetheless promote *piecewise smoothness* (PWS) in 2D surfaces, *e.g.*, flat sides of a box connected by sharp distinct transitions. Second, because a point cloud specifies a free object's 3D geometry, the defined smoothness notion should be *rotation-invariant* in 3D space [6]. These difficulties mean that image restoration methods based on conventional smoothness notions like TV [3] cannot be directly applied to restore corrupted point clouds.

In this paper, leveraging on recent advances in graph spectral image processing [7], we design a smoothness notion for 3D point clouds called *reweighted graph Laplacian regularizer* (RGLR) defined on surface normals with three desirable properties: i) rotation invariance, ii) promotion of PWS in surface normals, and iii) ease of optimization. We prove that RGLR is invariant to any unitary rotation matrix. We show that RGLR promotes PWS by analyzing its behavior for a single node pair. Using RGLR as a signal prior, we formulate the point cloud denoising problem as an optimization with a general ℓ_p-norm fidelity term that can explicitly model two types of independent noise: i) non-sparse but small noise like Gaussian (using a ℓ_2-norm fidelity term), and ii) sparser but larger noise like Laplacian (using a ℓ_1-norm fidelity term)[1].

To efficiently solve the posed optimization, we first establish a linear relationship between normals and 3D point coordinates via bipartite graph approximation [8] to divide the point cloud into two disjoint node sets (red and blue). The normal for each red node can thus be defined with respect to neighboring blue nodes' coordinates, resulting in a linear function of the red node's coordinates. Thereafter, we optimize the red and blue nodes' coordinates alternately. For ℓ_2-norm fidelity term, we iteratively solve an unconstrained *quadratic programming* (QP) problem, which can be efficiently computed using *conjugate gradient* (CG) [9] with a bounded condition number via *Gershgorin circle theorem* [10], guaranteeing numerical stability. Alternatively, the solution can be implemented as a graph filter via the *Lanczos algorithm* [11], allowing a spectral low-pass interpretation. For ℓ_1-norm fidelity term, we iteratively minimize an ℓ_1-ℓ_2 cost function using *accelerated*

Chinthaka Dinesh and Ivan V. Bajić are with the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada, e-mail: hchintha@sfu.ca and ibajic@ensc.sfu.ca.

Gene Cheung is with the Department of Electrical Engineering & Computer Science, York University, Toronto, Canada, e-mail: genec@yorku.ca.

[1]Sparse large noise can also be modeled using a ℓ_0 norm. However, ℓ_1 norm is the closest convex approximation and leads to fast optimal algorithms.

proximal gradient (APG) [12], where a step size ensuring a good convergence rate is chosen based on *Lipschitz continuity* analysis [13].

Finally, we propose simple mean and median filters for flat patches of a given point cloud to estimate the noise variance given the noise type, which in turn is used to compute a weight parameter trading off the fidelity term and signal prior in the problem formulation. Extensive experiments show state-of-the-art denoising performance among local methods using our proposed algorithms both subjectively and objectively under a number of point cloud metrics in the literature [14].

The outline of the paper is as follows. We first overview related works in Section II. Next, we define necessary concepts in Section III. We then present the definitions of RGLR and problem formulation in Section IV. We describe our proposed optimization algorithms in Section V. We discuss the computation of an appropriate weight parameter and noise variance estimation in Section VI. Finally, experimental results and conclusions are presented in Sections VII and VIII respectively.

II. Related Work

We divide existing related works into five categories: moving least squares (MLS)-based methods, locally optimal projection (LOP)-based methods, sparsity-based methods, graph-based method, and non-local similarity-based method.

MLS-based method: In MLS-based methods, a smooth surface is approximated from the input point cloud, and observed points are projected to the resulting surface. To construct the surface, [15] first finds a best-fitting local reference domain for a neighborhood of points in terms of MLS. Then a function is defined above the reference domain by fitting a polynomial function to neighboring data. However, if the underlying surface has high curvatures, then this method becomes unstable. In response, several solutions have been proposed, *e.g.*, algebraic point set surfaces (APSS) [16] and its variant [17] and robust implicit MLS (RIMLS) [18]. However, these methods may over-smooth [19].

LOP-based methods: LOP-based methods do not compute explicit parameters for the point cloud surface. For example, [20] generates a point set that represents the underlying surface while enforcing a uniform distribution over the point cloud. There are two main modifications to [20]. The first is weighted LOP (WLOP) [21] that provides a uniformly distributed output by preventing a given point from being too close to other neighbors. The second is anisotropic WLOP (AWLOP) [22] that preserves sharp features using an anisotropic weighting function. However, LOP-based methods also suffer from over-smoothing due to local operators [19].

Sparsity-based methods: There are two main steps in sparsity-based methods. First, a sparse reconstruction of the surface normals is obtained by solving a minimization with sparsity regularization. Then the point positions are updated by solving another minimization based on a local planar assumption. Examples include [23] and [19] that use ℓ_1 and ℓ_0 regularization respectively. However, for large noise these methods also over-smooth or over-sharpen [19].

Non-local methods: Non-local methods generalize *non-local means* (NLM) [24] and BM3D [25] image denoising algorithms to point cloud denoising. These methods rely on the self-similarity characteristic among surface patches in the point cloud. Methods in [6], [26] utilize a NLM algorithm, while a method in [27] is inspired by BM3D. Recently, [28] defines self-similarity among patches as a *low-dimensional manifold model* (LDMM) [29]. Although non-local methods achieve good performance, their computational complexity is often too high. In contrast, our algorithms are local and do not require non-local similar patch search.

Graph-based methods: In these methods, first, a graph (*e.g.* k-nearest-neighbor graph) is constructed from a given point cloud model. In [30], a local tangent plane at each point in the constructed point cloud graph is estimated, and then each 3D point is denoised via weighted averaging of its projections on neighboring tangent planes. Due to averaging, sharp edges cannot be denoised properly, resulting in over-smoothing. In [31], graph total variation (GTV)-based regularization method is used for denoising, where GTV is directly applied on the 3D coordinates. This is not appropriate because GTV of coordinates promotes variational proximity of 3D points, and only a singular 3D point has zero GTV value.

In our recent work, we propose a GTV-based denoising method [32], where GTV is applied on the estimated surface normals. However, This usage of GTV is not *rotation-invariant*; *i.e.*, the value of the GTV changes when the point cloud is rotated around a random axis.

All the aforementioned methods are derived from formulations that either explicitly assume Gaussian noise, or it is unclear for what noise type the methods work well. Instead we propose two graph-based denoising methods explicitly for non-sparse Gaussian noise or sparse Laplacian noise. Our methods differ from our recent work [32] as follows: i) we design RGLR as a signal prior for point clouds with two desirable properties—rotation invariance and promotion of PWS; ii) based on RGLR, we formulate the denoising problem as a general ℓ_p-ℓ_2-norm convex optimization and design two algorithms to explicitly optimize the formulated problem for the two aforementioned noise types; and iii) we propose different algorithms to estimate noise variance for these two noise types, which is used to compute a weight parameter trading off the fidelity term and the signal prior in the formulation.

III. Preliminaries

3D Point Cloud: We define a point cloud as a set of discrete samples of 3D coordinates of an object's 2D surface in 3D space. Denote by $\mathbf{q} = \begin{bmatrix}\mathbf{q}_1^T & \ldots & \mathbf{q}_N^T\end{bmatrix}^T \in \mathbb{R}^{3N}$ the position vector for the point cloud, where $\mathbf{q}_i \in \mathbb{R}^3$ is the 3D coordinate of a point i and N is the number of points in the point cloud. Noise-corrupted $\mathbf{q}$ can be simply modeled as $\mathbf{q} = \mathbf{p} + \mathbf{e}$, where $\mathbf{p}$ are the true 3D coordinates, $\mathbf{e}$ is a zero-mean independent and identically distributed (iid) noise, and $\mathbf{p}$, $\mathbf{e} \in \mathbb{R}^{3N}$. The goal is to recover $\mathbf{p}$ given only noise-currupted observation $\mathbf{q}$.

Surface Normals: Surface normal at a point i in a given 3D point cloud is a vector (denoted by $\mathbf{n}_i \in \mathbb{R}^3$) that is perpendicular to the tangent plane to that surface at i. There are numerous methods in the literature [33]–[37] to compute a normal to the surface at point i. One popular method is to fit a local plane to point i and its k nearest neighbors, and take the perpendicular vector to that plane [33]–[35]. A common alternative is to compute the normal vector as the weighted average of the normal vectors of the triangles formed by i and pairs of its neighbors [36], [37]. In all these methods, the surface normal at i is related nonlinearly with the 3D coordinates of i and its neighbors.

Definitions in Graph Signal Processing: We define graph signal processing (GSP) related concepts needed in our work. Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ composed of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$ specified by $(i, j, w_{i,j})$, where $i, j \in \mathcal{V}$ and $w_{i,j} \in \mathbb{R}^+$ is the edge weight that reflects the similarity between nodes i and j. Graph $\mathcal{G}$ can be characterized by its *adjacency matrix* $\mathbf{W}$ with $\mathbf{W}(i,j) = \mathbf{W}(j,i) = w_{i,j}$. Further, denote by $\mathbf{D}$ the diagonal *degree matrix* where entry $\mathbf{D}(i,i) = \sum_j w_{i,j}$. Given $\mathbf{W}$ and $\mathbf{D}$, the *combinatorial graph Laplacian matrix* [38] is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. Since $\mathbf{W}$ and $\mathbf{D}$ are real symmetric matrices, $\mathbf{L}$ is also real and symmetric. Moreover, $\mathbf{L}$ is a positive semi-definite (PSD) matrix; *i.e.*, for any given vector $\mathbf{f} \in \mathbb{R}^N$, $\mathbf{f}^\top \mathbf{L}\mathbf{f} \geq 0$, where N is the number of nodes in the graph $\mathcal{G}$. $\mathbf{f}$ is called a *graph signal* on graph $\mathcal{G}$, where $\mathbf{f} = [f_1 \dots f_N]^\top$ and f_i is a scalar value assigned to node i. One can easily show that $\mathbf{f}^\top \mathbf{L}\mathbf{f}$ can be rewritten as

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \sum_{i,j \in \mathcal{E}} w_{i,j} (f_i - f_j)^2, \tag{1}$$

which is known as *graph Laplacian regularizer* (GLR) [4].

Graph Construction: In this paper, we first construct a k-nearest-neighbor (k-NN) graph to connect 3D points in a given point cloud based on Euclidean distance in 3D space [39]. Specifically, each 3D point is represented by a node and is connected through edges to its k nearest neighbors, with weights that reflect inter-node similarities. For a given point cloud $\mathbf{p} = \left[\mathbf{p}_1^\top \ \dots \ \mathbf{p}_N^\top\right]^\top$, edge weight $w_{i,j}$ between nodes i and j is computed using the Euclidean distance between nodes i and j (called *distance term* $w^p_{i,j}$), and the angle $\theta_{i,j}$ between $\mathbf{n}_i$ and $\mathbf{n}_j$ (called *angle term* $w^\theta_{i,j}$):

$$w_{i,j} = \underbrace{\exp\left\{-\frac{||\mathbf{p}_i - \mathbf{p}_j||_2^2}{\sigma_p^2}\right\}}_{w^p_{i,j}} \underbrace{\cos^2\theta_{i,j}}_{w^\theta_{i,j}}, \tag{2}$$

where σ_p is a parameter. In words, (2) states that edge weight $w_{i,j}$ is large (close to 1) if points i, j are close and associated normal vectors are similar, and small (close to 0) otherwise. By the definition of inner product of $\mathbf{n}_i$ and $\mathbf{n}_j$, $\cos^2\theta_{i,j}$ can be written as,

$$\cos^2\theta_{i,j} = (\mathbf{n}_i^\top \mathbf{n}_j)^2 = \left(\frac{2 - \|\mathbf{n}_i - \mathbf{n}_j\|_2^2}{2}\right)^2. \tag{3}$$

Given graph $\mathcal{G}$ constructed using edge weights (2), we use $\mathcal{G}$ to regularize separately each coordinate of the points' surface normals $\mathbf{N} = [\mathbf{n}_1^\top; \dots; \mathbf{n}_N^\top] \in \mathbb{R}^{N\times 3}$; *i.e.*, we consider three graph signals corresponding to x-, y- and z-coordinates of $\mathbf{N}$ defined as follows: $\mathbf{n}_x = [\mathbf{n}_{1,1} \ \dots \ \mathbf{n}_{N,1}]^\top$, $\mathbf{n}_y = [\mathbf{n}_{1,2} \ \dots \ \mathbf{n}_{N,2}]^\top$, $\mathbf{n}_z = [\mathbf{n}_{1,3} \ \dots \ \mathbf{n}_{N,3}]^\top$, where $\mathbf{n}_{i,j}$ is the j-th entry of $\mathbf{n}_i$.

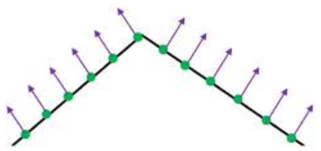


Figure 1. A 2D demonstration of an example for PWS surface with normals.

IV. Signal Prior and Problem Formulation

A. Definition of Reweighted Graph Laplacian Regularizer

In this paper, we assume that the normals computed (consistently oriented) at the 2D surface of a given 3D point cloud is PWS. In other words, given two neighboring nodes $i, j \in \mathcal{E}$ on a k-NN graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ representing a clean point cloud, the difference between the corresponding surface normals, *i.e.* $||\mathbf{n}_i - \mathbf{n}_j||_2$, should be typically very small (though large occasionally). See Fig. 1 for an illustration. Hence, we design a signal prior to promote PWS of $\mathbf{N}$. Moreover, because a point cloud specifies a free object's 3D geometry in an arbitrary 3D coordinate system, the prior should be *rotation-invariant* in 3D space [6]; *i.e.*, rotating the point cloud around any axis should not change the computed prior value.

To satisfy these two properties, we propose a RGLR prior for surface normals $\mathbf{N}$ over $\mathcal{G}$ as follows. Specifically, instead of using fixed $\mathbf{W}$, we extend the conventional GLR in (1) to RGLR, where the adjacency matrix $\mathbf{W}(\mathbf{N})$ (and hence the degree and Laplacian matrices $\mathbf{D}(\mathbf{N})$ and $\mathbf{L}(\mathbf{N})$) is a function of $\mathbf{N}$. We define RGLR for surface normals as

$$\begin{aligned} ||\mathbf{N}||_{\text{RGLR}} &= \mathbf{n}_x^\top \mathbf{L}(\mathbf{N})\mathbf{n}_x + \mathbf{n}_y^\top \mathbf{L}(\mathbf{N})\mathbf{n}_y + \mathbf{n}_z^\top \mathbf{L}(\mathbf{N})\mathbf{n}_z \\ &= \sum_{i,j \in \mathcal{E}} w_{i,j}(\mathbf{n}_i, \mathbf{n}_j) ||\mathbf{n}_i - \mathbf{n}_j||_2^2, \end{aligned} \tag{4}$$

where $w_{i,j}(\mathbf{n}_i, \mathbf{n}_j)$ is the (i,j) element of $\mathbf{W}(\mathbf{N})$. The initial value of $w_{i,j}(\mathbf{n}_i, \mathbf{n}_j)$ is computed using (2). Here, we consider $w_{i,j}(\mathbf{n}_i, \mathbf{n}_j)$ as a function of $\mathbf{n}_i$ and $\mathbf{n}_j$ as in (2). We show next that our proposed RGLR in (4) has the desired two properties: rotation invariance and promoting PWS.

Proposition 1. *RGLR defined in (4) is invariant to any rotation matrix applied to the point cloud.*

Proof. Suppose we rotate a point cloud around any given axis by applying a rotational matrix[2] $\mathbf{R} \in \mathbb{R}^{3\times 3}$. It follows that we should apply the same $\mathbf{R}$ to all surface normals computed before the rotation. We can thus write RGLR in (4) for the rotated point cloud (denoted as $||\mathbf{N}||^{\mathbf{R}}_{\text{RGLR}}$) based on the surface normals computed before the rotation as follows:

$$||\mathbf{N}||^{\mathbf{R}}_{\text{RGLR}} = \sum_{i,j \in \mathcal{E}} w_{i,j}(\mathbf{R}\mathbf{n}_i, \mathbf{R}\mathbf{n}_j) ||\mathbf{R}\mathbf{n}_i - \mathbf{R}\mathbf{n}_j||_2^2. \tag{5}$$

According to (2) and (3) we can write $w_{i,j}(\mathbf{R}\mathbf{n}_i, \mathbf{R}\mathbf{n}_j)$ as follows:

$$\begin{aligned} &= \exp\left\{-\frac{||\mathbf{R}\mathbf{p}_i - \mathbf{R}\mathbf{p}_j||_2^2}{\sigma_p^2}\right\} ((\mathbf{R}\mathbf{n}_i)^\top \mathbf{R}\mathbf{n}_j)^2 \\ &= \exp\left\{-\frac{(\mathbf{p}_i - \mathbf{p}_j)^\top \mathbf{R}^\top \mathbf{R}(\mathbf{p}_i - \mathbf{p}_j)}{\sigma_p^2}\right\} (\mathbf{n}_i^\top \mathbf{R}^\top \mathbf{R}\mathbf{n}_j)^2 \\ &= \exp\left\{-\frac{||\mathbf{p}_i - \mathbf{p}_j||_2^2}{\sigma_p^2}\right\} (\mathbf{n}_i^\top \mathbf{n}_j)^2 = w_{i,j}(\mathbf{n}_i, \mathbf{n}_j), \end{aligned} \tag{6}$$

[2]Rotation matrix is a square matrix, and a given square matrix $\mathbf{R}$ is a rotation matrix if and only if $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$ and $\det \mathbf{R} = 1$.

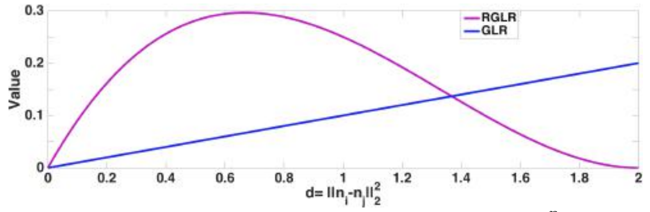


Figure 2. Curves of RGLR and GLR for a two node graph. $w^p_{i,j}$ is set to 1 for RGLR and $w_{i,j}$ is set to 0.1 for GLR.

where $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$ as $\mathbf{R}$ is a unitary matrix. Given (6), we can rewrite (5) as follows:

$$\begin{aligned}\|\mathbf{N}\|^{\mathbf{R}}_{\text{RGLR}} &= \sum_{i,j\in\mathcal{E}} w_{i,j}(\mathbf{n}_i,\mathbf{n}_j)||\mathbf{Rn}_i - \mathbf{Rn}_j||_2^2 \\ &= \sum_{i,j\in\mathcal{E}} w_{i,j}(\mathbf{n}_i,\mathbf{n}_j)\ (\mathbf{n}_i-\mathbf{n}_j)^\top\mathbf{R}^\top\mathbf{R}(\mathbf{n}_i-\mathbf{n}_j) \\ &= \sum_{i,j\in\mathcal{E}} w_{i,j}(\mathbf{n}_i,\mathbf{n}_j)||\mathbf{n}_i-\mathbf{n}_j||_2^2 = \|\mathbf{N}\|_{\text{RGLR}}\,.\end{aligned} \tag{7}$$

Therefore RGLR in (4) is rotation-invariant. □

A simple analysis shows that the proposed RGLR promotes PWS. Since (4) is a sum of separable terms, we can study the behavior of RGLR using a single node pair (i,j); the behavior for the entire graph would be the sum of behaviors of individual pairs plus their interaction. With $d = \|\mathbf{n}_i - \mathbf{n}_j\|_2^2$ and fixed $w^p_{i,j}$, RGLR for pair (i,j) is $w_{i,j}(\mathbf{n}_i,\mathbf{n}_j)\,\|\mathbf{n}_i-\mathbf{n}_j\|_2^2 = w^p_{i,j}((2-d)^2/4)d$. Here $0 \le d \le 2$ because we consistently orient each surface normal so that $\mathbf{n}_i^T\mathbf{n}_j \ge 0$ (to be discussed in Section IV-B2). Within this range, RGLR has one maximum at $2-\sqrt{2}$ and two minima at 0 and 2, as shown in Fig. 2. Hence, using gradient descent to minimize (4) would reduce d if $d < 2-\sqrt{2}$ and amplify d if $d > 2-\sqrt{2}$. Generalizing to the entire graph, we see that RGLR pushes individual node pairs to the two minima and thus promotes PWS of $\mathbf{N}$ overall.

For comparison, we also analyze commonly used graph Laplacian regularizer (GLR) for surface normals. GLR for surface normals can be expressed as

$$||\mathbf{N}||_{\text{GLR}} = \mathbf{n}_x^\top\mathbf{L}\mathbf{n}_x + \mathbf{n}_y^\top\mathbf{L}\mathbf{n}_y + \mathbf{n}_z^\top\mathbf{L}\mathbf{n}_z = \sum_{i,j\in\mathcal{E}} w_{i,j}||\mathbf{n}_i-\mathbf{n}_j||_2^2. \tag{8}$$

GLR initializes $w_{i,j}$ using (2) and keeps it fixed. Hence, $\mathbf{L}$ is also fixed to its initial matrix. With $d = \|\mathbf{n}_i-\mathbf{n}_j\|_2^2$ and fixed $w_{i,j}$, GLR for pair (i,j) is $w_{i,j}\,\|\mathbf{n}_i-\mathbf{n}_j\|_2^2 = w_{i,j}d$, which is a linear function of d with slope $w_{i,j}$. The curve of this GLR has only one minimum at $d = 0$ as shown in Fig. 4. Hence, considering only pair (i,j), minimizing (8) only pushes d towards 0. Thus RGLR more aggressively promotes PWS of the target signal than GLR.

B. Problem Formulation

Using our proposed RGLR (4) as a signal prior, we formulate our point cloud denoising problem as an optimization with a general l_p-norm fidelity term, $\|\mathbf{q}-\mathbf{p}\|_p^p$. Here, $\mathbf{p}$ is the optimization variable, and $\mathbf{n}_i$'s are functions of $\mathbf{p}$. Unfortunately, using state-of-the-art surface normal estimation methods [33]–[37], each $\mathbf{n}_i$ is a nonlinear function of $\mathbf{p}_i$ and its neighbors. Hence, it is difficult to formulate a tractable optimization problem using RGLR.

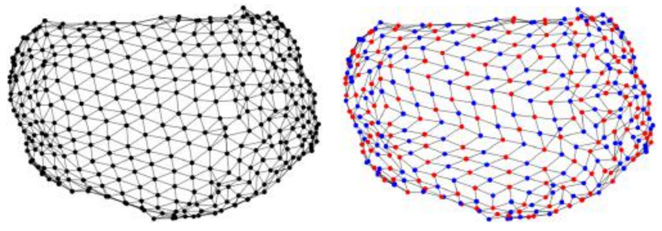

(a) original graph (b) bipartite graph

Figure 3. An example for bipartite graph approximation.

To overcome this difficulty, we first *partition* the set of 3D points into two classes (say red and blue). To compute surface normal of a red point, we use only coordinates of the red point and its neighboring blue points, resulting in a linear relationship between surface normals and 3D coordinates of red points. Towards this goal, we compute a bipartite graph approximation of the original graph $\mathcal{G}$ as follows.

1) Bipartite Graph Approximation: A bipartite graph $\mathcal{B} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}')$ is a graph whose nodes are divided into two disjoint sets $\mathcal{V}_1$ and $\mathcal{V}_2$ (red and blue nodes respectively), such that each edge connects a node in $\mathcal{V}_1$ to one in $\mathcal{V}_2$. Given an original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, our goal is to find a bipartite graph $\mathcal{B}$ that is "closest" to $\mathcal{G}$ under a chosen metric.

First, we assume that the generative model for graph signals $\mathbf{f}$ on a given graph $\mathcal{G}$ is a *Gaussian Markov random field* (GMRF) [40]. Specifically, $\mathbf{f} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$, where μ is the mean vector, and $\mathbf{\Sigma}$ is the covariance matrix specified by the graph Laplacian matrix $\mathbf{L}$ of $\mathcal{G}$. Therefore, $\Sigma^{-1} = \mathbf{L} + \delta\mathbf{I}$, where $1/\delta$ is the variance of the DC component for $\mathbf{f}$. For simplicity, we assume $\mu = \mathbf{0}$.

We now find a graph $\mathcal{B}$ whose probability distribution $\mathcal{N}_B(\mathbf{0}, \mathbf{\Sigma}_\mathcal{B})$ under the same GMRF model is closest to $\mathcal{N}(\mathbf{0},\mathbf{\Sigma})$ in terms of *Kullback-Leibler Divergence* (KLD) [41]:

$$D_{KL}(\mathcal{N}||\mathcal{N}_\mathcal{B}) = \frac{1}{2}(\text{tr}(\mathbf{\Sigma}_\mathcal{B}^{-1}\mathbf{\Sigma}) + \ln|\mathbf{\Sigma}_\mathcal{B}\mathbf{\Sigma}^{-1}| - N), \tag{9}$$

where $\Sigma_\mathcal{B}^{-1} = \mathbf{L}_\mathcal{B} + \delta\mathbf{I}$ is the precision matrix of the GMRF specified by $\mathcal{B}$, and $\mathbf{L}_\mathcal{B}$ is the graph Laplacian matrix of $\mathcal{B}$. To minimize (9), we use an iterative greedy algorithm similar to the one proposed in [8] as follows.

Given a non-bipartite graph $\mathcal{G}$, we build a bipartite graph $\mathcal{B}$ by iteratively adding one node at a time to one of two disjoint sets ($\mathcal{V}_1$ and $\mathcal{V}_2$) and removing the edges within each set. First, we initialize one node set $\mathcal{V}_1$ with one randomly chosen node, and $\mathcal{V}_2$ is empty. Then, we use *breadth-first search* (BFS) [42] to explore nodes within one hop in $\mathcal{G}$. To determine to which set between $\mathcal{V}_1$ and $\mathcal{V}_2$ the next node should be assigned, we calculate KLD D^i_{KL}, where $i \in \{1,2\}$, assuming the node is assigned to $\mathcal{V}_1$ or $\mathcal{V}_2$ respectively. The node is assigned to $\mathcal{V}_1$ if $D^2_{KL} > D^1_{KL}$, and to $\mathcal{V}_2$ otherwise. If $D^2_{KL} = D^1_{KL}$, then the node is alternately allocated to $\mathcal{V}_1$ and $\mathcal{V}_2$.

An example of a bipartite graph approximation is shown in Fig. 3. Here, we construct the original k-NN graph ($k = 6$) using a small portion of the Bunny point cloud model in [43].

2) Normal Vector Estimation: We describe how we define $\mathbf{n}_i$ for a red node i of the approximated bipartite graph. For

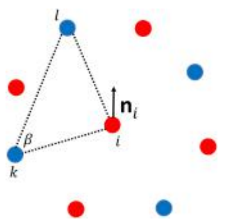


Figure 4. Illustration of the normal vector estimation at a red node.

each red node i, we choose two connected blue nodes k and l (with a criterion to be discussed) in order to define a 2D plane supported by nodes i, k and l. $\mathbf{n}_i$ is a unit-norm vector perpendicular to the plane. See Fig. 4 for an illustration. The corresponding 3D coordinates of nodes i, k, and l are denoted by $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\top$, $\mathbf{p}_k^i = \begin{bmatrix} x_k & y_k & z_k \end{bmatrix}^\top$, and $\mathbf{p}_l^i = \begin{bmatrix} x_l & y_l & z_l \end{bmatrix}^T$ respectively.

To select blue nodes k and l for a given red node i, we use the following two criteria: i) $\left\|\mathbf{p}_i - \mathbf{p}_k^i\right\|_2$ is larger than a pre-defined threshold; and ii) the angle β_i between $[\mathbf{p}_i - \mathbf{p}_k^i]$ and $[\mathbf{p}_k^i - \mathbf{p}_l^i]$ is as close to 90 degrees as possible. To be discussed soon, these two criteria lead to numerical stability for our solution. Mathematically, we define $\mathbf{n}_i$ as the unit-norm vector perpendicular to that plane:

$$\mathbf{n}_i = \frac{[\mathbf{p}_i - \mathbf{p}_k^i] \times [\mathbf{p}_k^i - \mathbf{p}_l^i]}{||[\mathbf{p}_i - \mathbf{p}_k^i] \times [\mathbf{p}_k^i - \mathbf{p}_l^i]||_2}, \tag{10}$$

where '$\times$' represents the vector cross product operator. We rewrite the cross product in (10) as follows:

$$[\mathbf{p}_i - \mathbf{p}_k^i] \times [\mathbf{p}_k^i - \mathbf{p}_l^i] = \underbrace{\begin{bmatrix} 0 & z_k - z_l & y_l - y_k \\ z_l - z_k & 0 & x_k - x_l \\ y_k - y_l & x_l - x_k & 0 \end{bmatrix}}_{\mathbf{C}_i} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \underbrace{\begin{bmatrix} -y_k(z_k - z_l) - z_k(y_l - y_k) \\ -x_k(z_k - z_l) - z_k(x_k - x_l) \\ -x_k(y_k - y_l) - y_k(x_l - x_k) \end{bmatrix}}_{\mathbf{d}_i} \tag{11}$$

Thus, normal vector $\mathbf{n}_i$ can be re-written as:

$$\mathbf{n}_i = \frac{\mathbf{C}_i \mathbf{p}_i + \mathbf{d}_i}{||\mathbf{C}_i \mathbf{p}_i + \mathbf{d}_i||_2}. \tag{12}$$

The orientations of the normal vectors for a local neighborhood of red nodes computed using (12) are not necessarily *consistent*, *i.e.* one normal vector might point inward from the 3D object surface while a neighoboring normal might point outward. To achieve consistent orientations of the surface normals, we fix the orientation of one normal vector and propagate this information to neighboring red nodes as done in [32]. Consistently oriented normal vectors can be written as:

$$\mathbf{n}_i = \left(\frac{\mathbf{C}_i \mathbf{p}_i + \mathbf{d}_i}{||\mathbf{C}_i \mathbf{p}_i + \mathbf{d}_i||_2} \right) \alpha, \tag{13}$$

where α is 1 or -1 depending on the normal orientation.

For simplicity we assume that α and $||\mathbf{C}_i \mathbf{p}_i + \mathbf{d}_i||_2$ remain constant when $\mathbf{p}_i$ is being optimized. Hence, surface normal $\mathbf{n}_i$ can be written as a linear function of 3D coordinates $\mathbf{p}_i$ of the point i. Specifically,

$$\mathbf{n}_i = \mathbf{A}_i \mathbf{p}_i + \mathbf{b}_i, \tag{14}$$

where $\mathbf{A}_i = \mathbf{C}_i \alpha^{in} / ||\mathbf{C}_i \mathbf{p}_i^{in} + \mathbf{d}_i||_2$ and $\mathbf{b}_i = \mathbf{d}_i \alpha^{in} / ||\mathbf{C}_i \mathbf{p}_i^{in} + \mathbf{d}_i||_2$. Here $\mathbf{p}_i^{in}$ is the initial vector of $\mathbf{p}_i$ and α^{in} is the value of α computed at $\mathbf{p}_i = \mathbf{p}_i^{in}$.

3) *Optimization Cost:*

After computing surface normals for each red node based on blue nodes, we construct a *new* k-NN graph $\mathcal{G}_\mathcal{R} = (\mathcal{V}_\mathcal{R}, \mathcal{E}_\mathcal{R})$ for red nodes, where $\mathcal{V}_\mathcal{R}$ is the set of red nodes and $\mathcal{E}_\mathcal{R}$ is the set of edges connecting neighboring red nodes. Given $\mathcal{G}_\mathcal{R}$, we define an ℓ_p-ℓ_2-norm minimization problem for denoising:

$$\min_{\mathbf{p}} ||\mathbf{q} - \mathbf{p}||_p^p + \gamma ||\mathbf{N}||_{\text{RGLR}} \tag{15}$$

where γ is a weight parameter that trades off the fidelity term $||\mathbf{q} - \mathbf{p}||_p^p$ with prior $\|\mathbf{N}\|_{\text{RGLR}}$ (4). Small but non-sparse noise (like Gaussian) is conventionally modeled using an ℓ_2-norm fidelity term [44]–[46]. Further, large but sparse noise (like Laplacian) can be modeled using a convex ℓ_1-norm fidelity term instead [47], [48]. Depending on the point cloud acquisition method used, one of the two noise models can be appropriately chosen. We focus on solving the denoising problem in (15) explicitly for these two noise models.

According to (14), we can write $\mathbf{n}_x$, $\mathbf{n}_y$, and $\mathbf{n}_z$ used in $\|\mathbf{N}\|_{\text{RGLR}}$ for $\mathcal{G}_\mathcal{R}$ as follows:

$$\begin{aligned} \mathbf{n}_x &= \underbrace{\text{diag}\{\mathbf{A}_1(1,:), \ldots, \mathbf{A}_N(1,:)\}}_{\mathbf{A}_x} \mathbf{p} + \underbrace{\begin{bmatrix} b_{11} & \ldots & b_{N1} \end{bmatrix}^\top}_{\mathbf{b}_x}, \\ \mathbf{n}_y &= \underbrace{\text{diag}\{\mathbf{A}_1(2,:), \ldots, \mathbf{A}_N(2,:)\}}_{\mathbf{A}_y} \mathbf{p} + \underbrace{\begin{bmatrix} b_{12} & \ldots & b_{N2} \end{bmatrix}^\top}_{\mathbf{b}_y}, \\ \mathbf{n}_z &= \underbrace{\text{diag}\{\mathbf{A}_1(3,:), \ldots, \mathbf{A}_N(3,:)\}}_{\mathbf{A}_z} \mathbf{p} + \underbrace{\begin{bmatrix} b_{13} & \ldots & b_{N3} \end{bmatrix}^\top}_{\mathbf{b}_z}, \end{aligned} \tag{16}$$

where $\mathbf{A}_i(j,:)$ denotes the j-th row of $\mathbf{A}_i$ and b_{ij} denotes the j-th entry of $\mathbf{b}_i$. Given (16), we rewrite $\|\mathbf{N}\|_{\text{RGLR}}$ as

$$\begin{bmatrix} \mathbf{A}_x\mathbf{p} + \mathbf{b}_x \\ \mathbf{A}_y\mathbf{p} + \mathbf{b}_y \\ \mathbf{A}_z\mathbf{p} + \mathbf{b}_z \end{bmatrix}^\top \begin{bmatrix} \mathbf{L}(\mathbf{p}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}(\mathbf{p}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{L}(\mathbf{p}) \end{bmatrix} \begin{bmatrix} \mathbf{A}_x\mathbf{p} + \mathbf{b}_x \\ \mathbf{A}_y\mathbf{p} + \mathbf{b}_y \\ \mathbf{A}_z\mathbf{p} + \mathbf{b}_z \end{bmatrix}. \tag{17}$$

Since $\mathbf{n}$ is a function of $\mathbf{p}$, we write $\mathbf{L}(\mathbf{N})$ as $\mathbf{L}(\mathbf{p})$ instead in the sequel.

In our optimization framework, we first solve (15) for red nodes while keeping the positions of blue nodes fixed. Then, using the newly solved red nodes' positions, we compute the surface normal for each blue node and construct another k-NN graph $\mathcal{G}_\mathcal{B}(\mathbf{V}_\mathcal{B}, \mathcal{E}_\mathcal{B})$ for blue nodes. Then (15) is solved for blue nodes while keeping the positions of red nodes fixed. The two node sets are alternately optimized until convergence. We next discuss our optimization algorithm to solve (15) for a given set of nodes (say red nodes in graph $\mathcal{G}_\mathcal{R}$) at one iteration.

V. Optimization Algorithm

Since $\mathbf{L}(\mathbf{p})$ is a function of $\mathbf{p}$, optimization (15) is non-convex, which is challenging to solve. Thus, we use an iterative method to optimize $\mathbf{p}$. At the k-th iteration, we fix $\mathbf{L} = \mathbf{L}\left(\mathbf{p}^{(k-1)}\right)$, then compute $\mathbf{p}^{(k)}$ by minimizing (15).

Here, we initialize $\mathbf{p}^{(0)} = \mathbf{q}$. With fixed $\mathbf{L}$, (15) becomes a convex optimization problem for $p \in \{1,2\}$:

$$\min_{\hat{\mathbf{p}}} \|\mathbf{q}-\hat{\mathbf{p}}\|_p^p + \gamma \underbrace{\begin{bmatrix}\mathbf{L}.(\mathbf{A}_x\hat{\mathbf{p}}+\mathbf{b}_x)\\ \mathbf{L}.(\mathbf{A}_y\hat{\mathbf{p}}+\mathbf{b}_y)\\ \mathbf{L}.(\mathbf{A}_z\hat{\mathbf{p}}+\mathbf{b}_z)\end{bmatrix}^\top \begin{bmatrix}\mathbf{A}_x\hat{\mathbf{p}}+\mathbf{b}_x\\ \mathbf{A}_y\hat{\mathbf{p}}+\mathbf{b}_y\\ \mathbf{A}_z\hat{\mathbf{p}}+\mathbf{b}_z\end{bmatrix}}_{f(\hat{\mathbf{p}})}. \quad (18)$$

A. Solving (18) with ℓ_2-norm fidelity term

When $p = 2$, (18) becomes a minimization of a quadratic cost of $\mathbf{p} \in \hat{\mathbb{R}}^{3N_\mathcal{R}}$ ($N_\mathcal{R}$ is the number of nodes in the graph $\mathcal{G}_\mathcal{R}$):

$$\min_{\hat{\mathbf{p}}} \|\mathbf{q}-\hat{\mathbf{p}}\|_2^2 + \gamma f(\hat{\mathbf{p}}). \quad (19)$$

To optimize $\hat{\mathbf{p}}$, we first take the derivative of (19) with respect to $\hat{\mathbf{p}}$, set it to $\mathbf{0}$ and solve for the closed form solution $\mathbf{p}^k$:

$$\left[\mathbf{I} + \gamma \underbrace{\begin{bmatrix}\mathbf{L}\mathbf{A}_x\\ \mathbf{L}\mathbf{A}_y\\ \mathbf{L}\mathbf{A}_z\end{bmatrix}^\top \begin{bmatrix}\mathbf{A}_x\\ \mathbf{A}_y\\ \mathbf{A}_z\end{bmatrix}}_{\tilde{\mathbf{L}}}\right]\mathbf{p}^{(k)} = \mathbf{q} - \gamma \underbrace{\begin{bmatrix}\mathbf{L}\mathbf{A}_x\\ \mathbf{L}\mathbf{A}_y\\ \mathbf{L}\mathbf{A}_z\end{bmatrix}^\top \begin{bmatrix}\mathbf{b}_x\\ \mathbf{b}_y\\ \mathbf{b}_z\end{bmatrix}}_{\tilde{\mathbf{L}}}, \quad (20)$$

where $\mathbf{I}$ is an identity matrix. Since $\mathbf{L}$ is a positive semi-definite (PSD) matrix, $\tilde{\mathbf{L}}$ is also PSD. Hence $\mathbf{I}+\gamma\tilde{\mathbf{L}}$ is positive definite (PD) for $\gamma > 0$ and thus invertible. We demonstrate soon that solving the system of linear equations in (20) is numerically stable by showing $\mathbf{I}+\gamma\tilde{\mathbf{L}}$ has a bounded *condition number*[3]. Further, $\mathbf{I}+\gamma\tilde{\mathbf{L}}$ is a sparse matrix. Hence we can efficiently solve (20) using *conjugate gradient* (CG) [9] without full matrix inversion.

After obtaining $\mathbf{p}^{(k)}$ from (20), $\mathbf{L}$ is updated as $\mathbf{L} = \mathbf{L}(\mathbf{p}^{(k)})$ and then $\mathbf{p}^{(k+1)}$ is computed using (20) again. Repeating this procedure, $\mathbf{p}$ is iteratively optimized until convergence.

1) Numerical Stability: We demonstrate the numerical stability of the system of linear equations in (20) based on the condition number of $\mathbf{I}+\gamma\tilde{\mathbf{L}}$ via the following eigen-analysis. Denote the maximum eigenvalues of $\mathbf{A}_x^\top\mathbf{L}\mathbf{A}_x$, $\mathbf{A}_y^\top\mathbf{L}\mathbf{A}_y$ and $\mathbf{A}_z^\top\mathbf{L}\mathbf{A}_z$ by $\mu_x^{\max}$, $\mu_y^{\max}$ and $\mu_z^{\max}$, respectively. According to *Weyl's inequality in matrix theory*[4], we know that $\mu^{\max} \le \mu_x^{\max}+\mu_y^{\max}+\mu_z^{\max}$, where $\mu^{\max}$ is the maximum eigenvalue of $\tilde{\mathbf{L}} = \mathbf{A}_x^\top\mathbf{L}\mathbf{A}_x+\mathbf{A}_y^\top\mathbf{L}\mathbf{A}_y+\mathbf{A}_z^\top\mathbf{L}\mathbf{A}_z$. Moreover, since $\tilde{\mathbf{L}}$ is PSD, its minimum eigenvalue is larger or equal to 0. We see easily that the maximum eigenvalue of $\mathbf{I}+\gamma\tilde{\mathbf{L}}$ is upper bounded by $1+\gamma\mu^{\max}$ and its minimum eigenvalue is lower bounded by 1. Thus we can upper-bound condition number C for $\mathbf{I}+\gamma\tilde{\mathbf{L}}$ as follows:

$$C \le 1+\gamma(\mu_x^{\max}+\mu_y^{\max}+\mu_z^{\max}). \quad (21)$$

We derive an upper bound for $\mu_x^{\max}$. Denote by $\mathbf{v}$ the unit-norm eigenvector corresponding to $\mu_x^{\max}$. By the definition of eigenvalues and eigenvectors, $\mathbf{v}^\top\mathbf{A}_x^\top\mathbf{L}\mathbf{A}_x\mathbf{v} = \mu_x^{\max}$. Denote the maximum eigenvalue of $\mathbf{L}$ by $\lambda^{\max}$. Following the *min-max theorem*[5], we know that:

$$\frac{\mathbf{u}^\top\mathbf{L}\mathbf{u}}{\|\mathbf{u}\|_2^2} \le \lambda^{\max}, \quad \forall \mathbf{u} \neq \mathbf{0}. \quad (22)$$

If we select $\mathbf{u} = \mathbf{A}_x\mathbf{v}$ and combine (22) with the expression $\mathbf{v}^\top\mathbf{A}_x^\top\mathbf{L}\mathbf{A}_x\mathbf{v} = \mu_x^{\max}$, we can write:

$$\begin{aligned}\mu_x^{\max} &\le \|\mathbf{A}_x\mathbf{v}\|_2^2\,\lambda^{\max} \le \|\mathbf{A}_x\|_2^2\,\|\mathbf{v}\|_2^2\,\lambda^{\max}\\ &= \|\mathbf{A}_x\|_2^2\,\lambda^{\max},\end{aligned} \quad (23)$$

where $\|\mathbf{v}\|_2^2 = 1$.

Moreover, using *Gershgorin circle theorem* [10], we can prove that $\lambda^{\max} \le 2\rho^{\max}$, where $\rho^{\max}$ is the maximum degree of a node in the k-NN graph $\mathcal{G}_\mathcal{R}$ (our proof is given in section 1.A in the supplement). Hence $\mu_x^{\max} \le 2\rho^{\max}\|\mathbf{A}_x\|_2^2$, and we can similarly derive $\mu_y^{\max} \le 2\rho^{\max}\|\mathbf{A}_y\|_2^2$ and $\mu_z^{\max} \le 2\rho^{\max}\|\mathbf{A}_z\|_2^2$. We now rewrite (21) as follows:

$$C \le 1+2\gamma\rho^{\max}\left(\|\mathbf{A}_x\|_2^2+\|\mathbf{A}_y\|_2^2+\|\mathbf{A}_z\|_2^2\right). \quad (24)$$

Note that $\|\mathbf{A}_x\|_2^2$ is equivalent[6] to the largest eigenvalue of $\mathbf{A}_x\mathbf{A}_x^\top$. By the definition of $\mathbf{A}_x$ in (16), $\mathbf{A}_x\mathbf{A}_x^\top$ is a diagonal matrix with its (i,i) entry equal[7] to

$$\frac{(z_k-z_l)^2+(y_l-y_k)^2}{\left\|\mathbf{p}_k^i-\mathbf{p}_l^i\right\|_2^2\left\|\mathbf{p}_i-\mathbf{p}_k^i\right\|_2^2\sin^2\beta_i} \le \frac{1}{\left\|\mathbf{p}_i-\mathbf{p}_k^i\right\|_2^2\sin^2\beta_i}. \quad (25)$$

Hence we can write an upper bound for $\|\mathbf{A}_x\|_2^2$ (*i.e.* the maximum eigenvalue of $\mathbf{A}_x\mathbf{A}_x^\top$) as follows:

$$\|\mathbf{A}_x\|_2^2 \le \frac{1}{\min_i\left\|\mathbf{p}_i-\mathbf{p}_k^i\right\|_2^2\sin^2\beta_i} \quad (26)$$

We see that $\|\mathbf{A}_y\|_2^2$ and $\|\mathbf{A}_z\|_2^2$ also have the same upper bound. From (24), we can write an upper bound for C as follows:

$$C \le 1+\frac{6\gamma\rho^{\max}}{\min_i\left\|\mathbf{p}_i-\mathbf{p}_k^i\right\|_2^2\sin^2\beta_i}. \quad (27)$$

For sufficiently small $\rho^{\max}$ and γ, and for sufficiently large $\min_i\left\|\mathbf{p}_i-\mathbf{p}_k^i\right\|_2^2\sin^2\beta_i$, C is sufficiently small, and thus the system equations in (20) has a numerically stable solution. Typically, in our experiments $\rho^{\max} < 10$ and $\gamma < 0.8$. Moreover, we select neighboring blue nodes k and l for a given red node i so that $\left\|\mathbf{p}_i-\mathbf{p}_k^i\right\|_2$ is sufficiently larger and β_i is close to 90 degrees, as discussed in Section IV-B2. Therefore, in practice, C has a sufficiently small value so that the system equations in (20) has a stable solution.

2) Graph spectral interpretation to solve (20): According to (20), we can obtain $\mathbf{p}^{(k)}$ as follows:

$$\mathbf{p}^{(k)} = (\mathbf{I}+\gamma\tilde{\mathbf{L}})^{-1}\tilde{\mathbf{q}}, \quad (28)$$

[3]The condition number of a given square symmetric matrix is defined as the ratio of its maximum and minimum eigenvalues.

[4]https://en.wikipedia.org/wiki/Weyl%27s_inequality

[5]https://en.wikipedia.org/wiki/Min-max_theorem

[6]https://en.wikipedia.org/wiki/Matrix_norm

[7]The value of the entry (i,i) of $\mathbf{A}_x\mathbf{A}_x^\top$ is obtained by following (10), (11), and (13). By definition $\left\|[\mathbf{p}_i-\mathbf{p}_k^i]\times[\mathbf{p}_k^i-\mathbf{p}_l^i]\right\| = \left\|\mathbf{p}_k^i-\mathbf{p}_l^i\right\|_2\left\|\mathbf{p}_i-\mathbf{p}_k^i\right\|_2\sin\beta_i$.

where $\tilde{\mathbf{q}} = \mathbf{q} - \gamma\bar{\mathbf{L}}$. Denote by $0 \leq \mu_1 \leq \ldots \leq \mu_N$ the eigenvalues of $\tilde{\mathbf{L}}$ and $\phi_1, \ldots, \phi_N$ the corresponding eigenvectors. The solution to (28) can thus be written as:

$$\mathbf{p}^{(k)} = \mathbf{\Phi}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^\top\tilde{\mathbf{q}}, \tag{29}$$

where $\mathbf{I}+\gamma\tilde{\mathbf{L}} = \mathbf{\Phi}\mathbf{\Sigma}\mathbf{\Phi}^\top$, which is the orthogonal decomposition of $\mathbf{I}+\gamma\tilde{\mathbf{L}}$ with $\mathbf{\Phi} = \begin{bmatrix}\phi_1 & \ldots & \phi_N\end{bmatrix}$ and $\mathbf{\Sigma} = \text{diag}(1+\gamma\mu_1, \ldots, 1+\gamma\mu_N)$. In GSP, eigenvalues and eigenvectors of the Laplacian matrix are commonly interpreted as graph frequencies and frequency basis [49]. Analogously, here we interpret $\mathbf{\Phi}^\top$ as a *graph Fourier transform* (GFT) operator that maps a graph signal $\mathbf{f}$ to its GFT coefficients $\mathbf{\Phi}^\top\mathbf{f}$. Therefore, observing that $\mathbf{\Sigma}^{-1} = \text{diag}(1/(1+\gamma\mu_1), \ldots, 1/(1+\gamma\mu_N))$, we can interpret the solution to $\mathbf{p}^{(k)}$ in (29) as follows. The graph signal $\tilde{\mathbf{q}}$ is transformed to the frequency domain via $\mathbf{\Phi}^\top$, attenuated GFT coefficients according to $\mathbf{\Sigma}^{-1}$ and transformed back to the nodal domain via $\mathbf{\Phi}$. This is called *graph spectral filtering* (GSF) [49]. We can thus interpret solving (28) as a *low-pass* filtering (since the attenuation is stronger for larger eigenvalues) of graph signal $\tilde{\mathbf{q}}$ using a polynomial graph filter $g(\tilde{\mathbf{L}}) = (\mathbf{I}+\gamma\tilde{\mathbf{L}})^{-1}$.

For graph spectral filters, we can solve the graph filtering problem in (28) using an accelerated filter on graphs via the *Lanczos method* [11] as an alternative to CG. The Lanczos method computes an orthonormal basis $\mathbf{V}_M = \begin{bmatrix}\mathbf{v}_1 \ldots \mathbf{v}_M\end{bmatrix}$ of the Krylov subspace $\mathbf{K}_M(\tilde{\mathbf{L}}, \tilde{\mathbf{q}}) = \text{span}\left\{\tilde{\mathbf{q}}, \tilde{\mathbf{L}}\tilde{\mathbf{q}}, \ldots, \tilde{\mathbf{L}}^{M-1}\tilde{\mathbf{q}}\right\}$ and the corresponding symmetric tridiagonal matrix $\mathbf{H}_M \in \mathbb{R}^{M\times M}$ satisfying $\mathbf{V}_M^{\text{H}}\tilde{\mathbf{L}}\mathbf{V}_M \approx \mathbf{H}_M$, where the exact non-zero entities of $\mathbf{H}_M$ can be computed using Algorithm 1 in [11]. The approximation of $\mathbf{p}^{(k)}$ with an order-M Lanczos method, where $M \ll N$, is

$$\mathbf{p}^{(k)} = g(\tilde{\mathbf{L}})\tilde{\mathbf{q}} \approx \|\tilde{\mathbf{q}}\|_2 \mathbf{V}_M g(\mathbf{H}_M)\mathbf{e}_1, \tag{30}$$

where $\mathbf{e}_1 \in \mathbb{R}^M$ is the first *canonical vector* (all elements are zeros except the first element is one), and $g(\mathbf{H}_M) = (\mathbf{I}+\gamma\mathbf{H}_M)^{-1}$. Given $M \ll N$ and since $\mathbf{I}+\gamma\mathbf{H}_M$ (denoted as $\tilde{g}(\mathbf{H}_M)$) is a symmetric tridiagonal matrix, orthogonal decomposition of $\tilde{g}(\mathbf{H}_M)$ can be computed using a *fast divide-and-conquer algorithm* [50] with computational complexity of $\mathcal{O}(M \ln M)$. Then we can directly obtain $g(\mathbf{H}_M)$ using the orthogonal decomposition of $\tilde{g}(\mathbf{H}_M)$.

B. Solving (18) with ℓ_1-norm fidelity term

When $p = 1$, (15) becomes a ℓ_1-ℓ_2 minimization problem as follows:

$$\min_{\hat{\mathbf{p}}} \|\mathbf{q} - \hat{\mathbf{p}}\|_1 + \gamma f(\hat{\mathbf{p}}), \tag{31}$$

where the second term $f(\hat{\mathbf{p}})$ is convex and differentiable, and the first term is convex but non-differentiable. We can thus employ *accelerated proximal gradient* (APG) [12] to solve (31). Differentiable $f(\hat{\mathbf{p}})$ has gradient ∇f:

$$\nabla f(\hat{\mathbf{p}}) = 2\gamma(\tilde{\mathbf{L}}\hat{\mathbf{p}} + \bar{\mathbf{L}}), \tag{32}$$

where $\tilde{\mathbf{L}}$ and $\bar{\mathbf{L}}$ are matrices defined in (20). We now define a *proximal mapping* $\text{prox}_{h,t}(\hat{\mathbf{p}})$ for a convex, non-differentiable function $h()$ with step size t as:

$$\text{prox}_{h,t}(\hat{\mathbf{p}}) = \arg\min_{\theta}\left\{h(\theta) + \frac{1}{2t}||\theta - \hat{\mathbf{p}}||_2^2\right\} \tag{33}$$

It is known that if $h()$ is simply a l_1-norm in (31), then the proximal mapping is a soft thresholding function [12]:

$$\text{prox}_{h,t}(\hat{p}_i) = \begin{cases}\hat{p}_i - t & \text{if } \hat{p}_i > q_i + t\\ q_i & \text{if } |\hat{p}_i| \leq q_i + t\\ \hat{p}_i + t & \text{if } \hat{p}_i < -q_i - t,\end{cases} \tag{34}$$

where $\hat{p}_i$ and q_i are i-th entries of $\hat{\mathbf{p}}$ and $\mathbf{q}$ respectively. To adopt an APG approach, we define (with initial point $\hat{\mathbf{p}}^{(-1)} = \hat{\mathbf{p}}^{(0)} = \mathbf{p}^{(k-1)}$)

$$\mathbf{z}^{(m-1)} = \hat{\mathbf{p}}^{(m-1)} + \frac{m-2}{m+1}\left(\hat{\mathbf{p}}^{(m-1)} - \hat{\mathbf{p}}^{(m-2)}\right), \tag{35}$$

where $\mathbf{z}^{(m-1)}$ is an extrapolated point computed from the two previous solutions $\hat{\mathbf{p}}^{(m-1)}$ and $\hat{\mathbf{p}}^{(m-2)}$. We now update $\hat{\mathbf{p}}^{(m)}$ using $\mathbf{z}^{(m-1)}$ as follows:

$$\hat{\mathbf{p}}^{(m)} = \text{prox}_{h,t}(\mathbf{z}^{(m-1)} - t\nabla f(\mathbf{z}^{(m-1)})). \tag{36}$$

We compute (36) iteratively until convergence.

An appropriate step size t for faster convergence can be selected based on *Lipschitz continuity* [13] of ∇f as follows. We say that ∇f is Lipschitz continuous if there is a real positive constant K such that, for all real $\mathbf{p}_1, \mathbf{p}_2 \in \mathbb{R}^{3N_{\mathcal{R}}}$,

$$\|\nabla f(\mathbf{p}_1) - \nabla f(\mathbf{p}_1)\|_2 \leq K\|\mathbf{p}_1 - \mathbf{p}_2\|_2. \tag{37}$$

The smallest possible K that satisfies (37) is called the Lipschitiz constant L. In our specific case, according to (32) we can write

$$\begin{aligned}\|\nabla f(\mathbf{p}_1) - \nabla f(\mathbf{p}_1)\|_2 &= 2\gamma\left\|\tilde{\mathbf{L}}\mathbf{p}_1 - \tilde{\mathbf{L}}\mathbf{p}_1\right\|_2\\ &\leq 2\gamma\left\|\tilde{\mathbf{L}}\right\|_2\|\mathbf{p}_1 - \mathbf{p}_1\|_2\end{aligned} \tag{38}$$

for all real $\mathbf{p}_1, \mathbf{p}_2$, and hence ∇f is Lipschitz continuous with $L = 2\gamma\left\|\tilde{\mathbf{L}}\right\|_2$. Here, $\left\|\tilde{\mathbf{L}}\right\|_2$ is the largest eigenvalue of $\tilde{\mathbf{L}}$ (since $\tilde{\mathbf{L}}$ is a symmetric matrix), and we know that it is upper-bounded by $\frac{6\rho^{max}}{\min_i\left\|\mathbf{p}_i - \mathbf{p}_k^i\right\|_2^2\sin^2\beta_i}$[8] from eigen-analysis in Section V-A1. As discussed in [12], APG has convergence rate $\mathcal{O}(1/k^2)$ when a step size $t \in (0, 1/L]$ is used. This is theoretically the fastest possible convergence rate for first-order methods.

Here, $\mathbf{p}^{(k)}$ equals to the converged $\hat{\mathbf{p}}$ obtained from (36). After obtaining $\mathbf{p}^{(k)}$, $\mathbf{L}$ is updated as $\mathbf{L} = \mathbf{L}(\mathbf{p}^{(k)})$ and then $\mathbf{p}^{(k+1)}$ is computed based on the updated $\mathbf{L}$. Repeating this procedure, $\mathbf{p}$ is iteratively optimized until convergence in order to solve (15) with the ℓ_1-norm fidelity term.

VI. Weight Parameter Estimation

A. Weight Parameter and Noise Variance

The computation of an optimal weight parameter (γ_{opt}) that trades off the fidelity term with the signal prior (15) is in general an open problem. Recently, the analysis of γ_{opt} for GLR was studied in [51] assuming that the ground truth signal is known, which is not practical.

Intuitively, one can see (and we verified through extensive experiments) that γ_{opt} is proportional to the noise variance (σ^2) of an iid noise that corrupts 3D points in a given

[8]As stated in Section IV-B2, when selecting k and l blue nodes for surface normal estimation at red node i, we calculate $\left\|\mathbf{p}_i - \mathbf{p}_k^i\right\|_2$ and β_i.

point cloud; *i.e.*, the larger the noise variance, the larger the importance of the prior term. We can thus write a simple linear relationship between γ_{opt} and σ^2 as follows:

$$\gamma_{\text{opt}} \approx \alpha\sigma^2, \tag{39}$$

where α is a pre-determined constant invariant to different point clouds corrupted by different noise levels. However, σ^2 is unknown in general. In the point cloud denoising literature [19], [23], [31], [32], [52], typically σ^2 is assumed known or γ_{opt} is tuned experimentally for best performance—neither is realizable in practice when only a noisy point cloud is given as input.

In contrast, we propose a simple and efficient method to estimate σ^2 in an observed point cloud. Our approach is analogous to [53] that estimates the noise variance of natural images. Assuming an iid noise model, [53] proposes to: i) identify *flat* patches (each patch has approximately a constant pixel value) from the given image, and ii) estimate the noise variance locally in each flat patch. Analogously, we first identify approximately flat 3D patches in a given noisy point cloud, and then estimate the noise variance from each patch. Before all these steps, we perform bipartite graph approximation as discussed in Section IV-B1, then estimate surface normals for one partite of nodes using our method in Section IV-B2.

To find flat patches, we first divide the point cloud into groups of 3D points with two similar features—geometric positions and estimated surface normals. Towards this goal, we employ a *mean shift* clustering algorithm [54] based on these two features hierarchically. The main reason is that mean shift is nonparametric; *i.e.*, we impose no prior on the number or shapes of the clusters. Similar hierarchical mean shift clustering techniques have been used in [55] for point cloud segmentation.

Given a point cloud, we first perform clustering in the surface normal space. We next apply local clustering for each surface normal cluster in the geometry space again using mean shift. The points in the final clusters thus have similar features in both geometry space and surface normal space. We know that a cluster from a flat region has a larger number of similar surface normals than a cluster from a non-flat region. Thus, if the number of points of a given final cluster is larger than a pre-specified threshold (in our experiments, we use 25), we consider that cluster a flat patch. For a given flat 3D patch, we have a strong prior that the normals of all points in the patch are approximately the same, which leads to simple and fast noise variance estimation methods as discussed next.

Our noise variance estimation method has two versions that are applicable to two different kinds of noise. For Gaussian noise, we employ a *mean filter* to compute first the average noise variance $\sigma^2_{\mathbf{n}}$ of surface normals in flat patches, then derive σ^2 from computed $\sigma^2_{\mathbf{n}}$. For Laplacian noise, we employ a *median filter* to estimate σ^2 from noisy points.

B. Noise Variance Estimation for Gaussian Noise

1) Computing Surface Normal Noise Variance $\sigma_{\mathbf{n}}$: Given a roughly flat patch with n_r red and n_b blue points, we estimate the average noise variance of the surface normals for the red point set $\mathcal{S}_r$ in the patch as follow. Given (14), we can write the noise corrupted surface normals $\tilde{\mathbf{n}}$ in $\mathcal{S}_r$, *i.e.*, $\tilde{\mathbf{n}} = \left[\tilde{\mathbf{n}}_1^\top \dots \tilde{\mathbf{n}}_{n_r}^\top\right]^\top$, as

$$\tilde{\mathbf{n}} = \mathbf{A}(\mathbf{p}+\mathbf{w}) + \mathbf{b}. \tag{40}$$

where $\mathbf{A} = \text{diag}(\mathbf{A}_1, \dots, \mathbf{A}_{n_r})$ and $\mathbf{b} = \left[\mathbf{b}_1^\top \dots \mathbf{b}_{n_r}^\top\right]^\top$. $\mathbf{p} = \left[\mathbf{p}_1^\top \dots \mathbf{p}_{n_r}^\top\right]^\top$ is the ground truth 3D coordinate vector for points in $\mathcal{S}_r$, and $\mathbf{w} \in \mathbb{R}^{3n_r}$ is the iid additive noise corrupting $\mathbf{p}$. Denote by $\mathbf{n}^o$ the surface normals for ground truth 3D points in $\mathcal{S}_r$,

$$\mathbf{n}^o = \mathbf{A}\mathbf{p} + \mathbf{b}, \tag{41}$$

The average noise variance $\sigma^2_{\mathbf{n}}$—noise variance per coordinate of each 3D point—for normals $\tilde{\mathbf{n}}$ is:

$$\sigma^2_{\mathbf{n}} = \frac{1}{3n_r}\text{E}\left[||\mathbf{n}^o - \tilde{\mathbf{n}}||^2\right]. \tag{42}$$

Obviously, we observe only noise-corrupted $\tilde{\mathbf{n}}$, and not ground truth $\mathbf{n}^o$. However, we can first denoise $\tilde{\mathbf{n}}$ with filter $f(\,)$ and approximate $\mathbf{n}^o \approx f(\tilde{\mathbf{n}})$. We can thus compute the *empirical* average noise variance as:

$$\sigma^2_{\mathbf{n}} = \frac{1}{3n_r}\left(f(\tilde{\mathbf{n}}) - \tilde{\mathbf{n}}\right)^\top\left(f(\tilde{\mathbf{n}}) - \tilde{\mathbf{n}}\right) \tag{43}$$

We derive the appropriate filter $f(\,)$ next.

Since $\mathbf{w}$ is a Gaussian random vector and $\mathbf{A}$ is a constant matrix, $(\mathbf{n}^o - \tilde{\mathbf{n}})$ is also a Gaussian random vector (see (40) and (41)). Thus we can design a filter $f(\,)$ by formulating an optimization problem to denoise $\mathbf{n}$ as:

$$\min_{\mathbf{n}} ||\mathbf{n} - \tilde{\mathbf{n}}||_2^2 + \gamma||\mathbf{N}||_{\text{RGLR}}, \tag{44}$$

where $||\mathbf{n} - \tilde{\mathbf{n}}||_2^2$ is the fidelity term.

Because of our explicit search for flat patches, in this case we have a strong signal prior that $\mathbf{n}$ contains the *same* normal denoted by $\mathbf{s} = [s_x \; s_y \; s_z]^\top$ for all points in the patch, and hence $||\mathbf{N}||_{\text{RGLR}} = 0$. Then the solution[9] to (44) defaults to the *mean filter*. Specifically,

$$\mathbf{n}^* = \arg\min_{s_x, s_y, s_z} ||\mathbf{I}_{3n_r\times 3}\begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} - \tilde{\mathbf{n}}||_2^2 \tag{45}$$

where $\mathbf{I}_{3n_r\times 3}$ is a column concatenation of n_r 3×3 identity matrices. Taking the derivative with respect to each of s_x, s_y and s_z, we get:

$$s_x^* = \frac{1}{n_r}\sum_{i=1}^{n_r}\tilde{n}_{i,1}, \quad s_y^* = \frac{1}{n_r}\sum_{i=1}^{n_r}\tilde{n}_{i,2}, \quad s_z^* = \frac{1}{n_r}\sum_{i=1}^{n_r}\tilde{n}_{i,3} \tag{46}$$

where $\tilde{n}_{i,1}$, $\tilde{n}_{i,2}$ and $\tilde{n}_{i,3}$ are the x-, y- and z-coordinates of $\tilde{\mathbf{n}}_i$. Then after normalizing $[s_x^* \; s_y^* \; s_z^*]^\top$, we can approximate $\mathbf{n}^o \approx f(\tilde{\mathbf{n}}) \approx [\mathbf{s}^{*\top} \dots \mathbf{s}^{*\top}]^\top$, where $\mathbf{s}^* = [s_x^* \; s_y^* \; s_z^*]^\top / \sqrt{{s_x^*}^2 + {s_y^*}^2 + {s_z^*}^2}$. We can now compute the empirical average noise variance $\sigma^2_{\mathbf{n}}$ for normals using (43).

[9]When solving (44), $\mathbf{s}$ should technically be constrained to have unit norm. However, this would result in a non-convex optimization problem. Hence we use a simple strategy where we first solve (44) without the unit-norm constraint and then normalize the solution afterwards.

2) Deriving σ^2 from $\sigma^2_{\mathbf{n}}$: Having computed $\sigma^2_{\mathbf{n}}$, we now derive the relationship between $\sigma^2_{\mathbf{n}}$ and σ^2 as follows using (40), (41) and (42):

$$\begin{aligned} 3n_r\sigma^2_{\mathbf{n}} &= \mathrm{E}\left[(\tilde{\mathbf{n}}-\mathbf{n}^o)^\top(\tilde{\mathbf{n}}-\mathbf{n}^o)\right] = \mathrm{E}\left[(\mathbf{A}\mathbf{w})^\top\mathbf{A}\mathbf{w}\right] \\ &= \mathrm{E}\left[\mathrm{tr}\left(\mathbf{w}^\top\mathbf{A}^\top\mathbf{A}\mathbf{w}\right)\right] = \mathrm{E}\left[\mathrm{tr}\left(\mathbf{w}\mathbf{w}^\top\mathbf{A}^\top\mathbf{A}\right)\right] \\ &= \mathrm{tr}\left(\mathrm{E}\left[\mathbf{w}\mathbf{w}^\top\right]\mathbf{A}^\top\mathbf{A}\right) = \mathrm{tr}\left(\sigma^2\mathbf{I}\mathbf{A}^\top\mathbf{A}\right) \\ &= \sigma^2\mathrm{tr}\left(\mathbf{A}^\top\mathbf{A}\right) = \sigma^2\sum_i \mathrm{tr}\left(\mathbf{A}_i^\top\mathbf{A}_i\right). \end{aligned} \tag{47}$$

where $\mathrm{E}[\mathbf{w}\mathbf{w}^T] = \sigma^2\mathbf{I}$ because of the inter-sample independent assumption. Hence, σ^2 can be obtained as

$$\sigma^2 = \frac{3n_r}{\sum_i \mathrm{tr}\left(\mathbf{A}_i^\top\mathbf{A}_i\right)}\sigma^2_{\mathbf{n}}. \tag{48}$$

For more reliable estimation for average noise variance σ^2, we compute it for both red and blue node sets for each flat patch, and take the average over all identified flat patches in the given point cloud.

C. Laplacian Noise Variance Estimate

1) Optimization Formulation: If the additive noise is Laplacian, the optimization for denoising red points in a given flat patch surface becomes:

$$\min_{\mathbf{p}} \sum_{i=1}^{n_r} ||\mathbf{q}_i - \mathbf{p}_i||_1 + \gamma||\mathbf{N}||_{\text{RGLR}} \tag{49}$$

Similar to (44), we also assume a strong prior that all points in the given patch have the same normals $\mathbf{s} = [s_x\ s_y\ s_z]^\top$, hence $||\mathbf{N}||_{\text{RGLR}} = 0$. Then, the solution to the optimization problem in (49) becomes:

$$\min_{\mathbf{p}} \sum_{i=1}^{n_r} ||\mathbf{q}_i - \mathbf{p}_i||_1, \quad \text{s.t.} \quad \mathbf{s} \approx \mathbf{A}_i\mathbf{p}_i + \mathbf{b}_i. \tag{50}$$

By solving (50), we find the denoised red points of the given flat patch and obtain the noise variance via the two following steps.

2) Upper Bound and Minimization: Because (50) is hard to optimize expediently, we minimize instead an upper bound of the cost function. Essentially, for each $\|\mathbf{q}_i - \mathbf{p}_i\|_1$, we replace it with an upper bound $\eta\|\mathbf{A}_i\mathbf{q}_i - \mathbf{A}_i\mathbf{p}_i\|_1$ and minimize it instead, where η is a real positive scalar. To achieve this goal we first derive a set of real orthonormal vectors from the eigenvectors of $\mathbf{A}_i$ to span the $\mathbb{R}^3$ space, so that $\mathbf{q}_i$ and $\mathbf{p}_i$ can be expressed in terms of them.

Since $\mathbf{A}_i$ is a 3×3 *skew symmetric* matrix (*i.e.*, $\mathbf{A}_i^\top = -\mathbf{A}_i$), its three eigenvalues[10] are 0, $i\lambda$, and $-i\lambda$, where λ is a real positive scaler and the corresponding orthonormal[11] eigenvectors are $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ respectively, where $\mathbf{u}_1$ is a real eigenvector. Further, $\mathbf{u}_2$ and $\mathbf{u}_3$ are complex conjugate of each other (the proof is provided in Section 1.B in the supplement). Being complex conjugates, $\beta_1(\mathbf{u}_2+\mathbf{u}_3)$ (denoted by $\mathbf{v}_2$) and $i\beta_2(\mathbf{u}_2-\mathbf{u}_3)$ (denoted by $\mathbf{v}_3$) are real vectors and are orthonormal, where β_1 and β_2 are corresponding normalization factors. Then $\mathbf{v}_1 = \mathbf{u}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ are three orthonormal basis vectors that span the $\mathbb{R}^3$ space, and we can write $\mathbf{p}_i$ and $\mathbf{q}_i$ as a linear combination of $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$. Specifically,

$$\mathbf{q}_i = \sum_{j=1}^{3}\hat{q}_{ij}\mathbf{v}_j, \quad \mathbf{p}_i = \sum_{j=1}^{3}\hat{p}_{ij}\mathbf{v}_j, \tag{51}$$

where $\hat{q}_{i,j}$ and $\hat{p}_{i,j}$ (for $j = 1,2,3$) are real values.

Next, using (51), we derive an upper bound $\eta\|\mathbf{A}_i\mathbf{q}_i - \mathbf{A}_i\mathbf{p}_i\|_1$ for $\|\mathbf{q}_i - \mathbf{p}_i\|_1$. First, to avoid mapping any energy of $\mathbf{q}_i - \mathbf{p}_i$ into the null space of $\mathbf{A}_i$ (*i.e.* space spanned by $\mathbf{v}_1$), we set $\hat{p}_{i1} = \hat{q}_{i1}$. Then by substituting $\mathbf{v}_2 = \beta_1(\mathbf{u}_2+\mathbf{u}_3)$ and $\mathbf{v}_3 = i\beta_2(\mathbf{u}_2-\mathbf{u}_3)$, we have,

$$\mathbf{q}_i - \mathbf{p}_i = \tilde{u}_2\mathbf{u}_2 + \tilde{u}_3\mathbf{u}_3, \tag{52}$$

where $\tilde{u}_2 = \beta_1(\hat{q}_{i2}-\hat{p}_{i2})+i\beta_2(\hat{q}_{i3}-\hat{p}_{i3})$ and $\tilde{u}_2 = \beta_1(\hat{q}_{i2}-\hat{p}_{i2})+i\beta_2(\hat{p}_{i3}-\hat{q}_{i3})$. According to the orthogonal transformation[12], $\mathbf{A}_i = i\lambda\mathbf{u}_2\mathbf{u}_2^{\mathrm{H}} - i\lambda\mathbf{u}_3\mathbf{u}_3^{\mathrm{H}}$ and then we can write

$$\|\mathbf{A}_i(\tilde{u}_2\mathbf{u}_2 + \tilde{u}_3\mathbf{u}_3)\|_1 = \lambda\left\|\tilde{u}_2\mathbf{u}_2 - \tilde{u}_3\mathbf{u}_3\right\|_1. \tag{53}$$

Further, for any vector $\mathbf{z} \in \mathbb{R}^3$ with $\|\mathbf{z}\|_2 = r$ where r is a constant, we can prove that $r \leq \|\mathbf{z}\|_1 \leq \sqrt{3}r$ (the proof is in Section 1.C of the supplement). Moreover, since $\mathbf{u}_2$ and $\mathbf{u}_3$ are orthogonal, one can see that $\|\tilde{u}_2\mathbf{u}_2 + \tilde{u}_3\mathbf{u}_3\|_2 = \|\tilde{u}_2\mathbf{u}_2 - \tilde{u}_3\mathbf{u}_3\|_2 = c$. Therefore, by the proof in the supplement, we have $c \leq \|\tilde{u}_2\mathbf{u}_2 + \tilde{u}_3\mathbf{u}_3\|_1 \leq \sqrt{3}c$ and $c \leq \|\tilde{u}_2\mathbf{u}_2 - \tilde{u}_3\mathbf{u}_3\|_1 \leq \sqrt{3}c$. We can write

$$\|\tilde{u}_2\mathbf{u}_2 + \tilde{u}_3\mathbf{u}_3\|_1 \leq \lambda\|\tilde{u}_2\mathbf{u}_2 - \tilde{u}_3\mathbf{u}_3\|_1, \quad \text{if} \quad \lambda \geq \sqrt{3}. \tag{54}$$

In this case we cannot guarantee that for all $\mathbf{A}_i$, $\lambda \geq \sqrt{3}$. Here we multiply each $\mathbf{A}_i$ with a common real positive scalar η so that the corresponding $\lambda \geq \sqrt{3}$. Therefore, we can select $\eta = \sqrt{3}/\chi$, where χ is the minimum λ among all $\mathbf{A}_i$. By combining (54) and (53) with (52), we have

$$||\mathbf{q}_i - \mathbf{p}_i||_1 \leq \eta\left\|\mathbf{A}_i\mathbf{q}_i - \mathbf{A}_i\mathbf{p}_i\right\|_1, \tag{55}$$

Using (55), we can write an upper bound for (50) as follows:

$$\begin{aligned} &\min_{\mathbf{p}} \sum_{i=1}^{n_r} ||\mathbf{q}_i - \mathbf{p}_i||_1 \leq \min_{\mathbf{p}} \sum_{i=1}^{n_r} \eta||\mathbf{A}_i\mathbf{q}_i - \mathbf{A}_i\mathbf{p}_i||_1 \\ &\quad \text{s.t.} \quad \mathbf{s} \approx \mathbf{A}_i\mathbf{p}_i + \mathbf{b}_i. \end{aligned} \tag{56}$$

Here for simplicity, instead of minimizing (50), we minimize its upper bound in (56) by substituting the constraint $\mathbf{s} \approx \mathbf{A}_i\mathbf{p}_i + \mathbf{b}_i$ to the cost function as follow:

$$\min_{\mathbf{s}} \sum_{i=1}^{n_r} ||\mathbf{A}_i\mathbf{q}_i - \mathbf{b}_i - \mathbf{s}||_1. \tag{57}$$

It is known[13] that the median filter minimizes the ℓ_1-norm with respect to a set of scalars. Hence, the solution[14] to (57) defaults to the *median filter* as follow:

$$s_x^* = \mathrm{med}\left(\{a_x^i\}\right), \ s_y^* = \mathrm{med}\left(\{a_y^i\}\right), \ s_z^* = \mathrm{med}\left(\{a_z^i\}\right) \tag{58}$$

[10]See the properties of skew symmetric matrix at https://en.wikipedia.org/wiki/Skew-symmetric_matrix

[11]Since $\mathbf{A}_i$ is a *normal* matrix (*i.e.* $\mathbf{A}_i^\top\mathbf{A}_i = \mathbf{A}_i\mathbf{A}_i^\top$), its eigenvectors are orthonormal.

[12]Orthogonal transformation of $\mathbf{A}_i$ is given as $\mathbf{A}_i = [\mathbf{u}_1 \ \mathbf{u}_2 \ \mathbf{u}_3]\,\mathrm{diag}(0, i\lambda, -i\lambda)\,[\mathbf{u}_1 \ \mathbf{u}_2 \ \mathbf{u}_3]^{\mathrm{H}}$.

[13]http://web.uvic.ca/ dgiles/blog/median2.pdf

[14]Following the similar strategy used in (44), we solve (57) without the normalization constraint for $\mathbf{s}$ and then normalize the solution afterwards.

where $\mathrm{med}(\mathcal{S})$ computes the median of the set of scalars in set $\mathcal{S}$, and a_x^i, a_y^i, a_x^i are x, y, z entries of $\mathbf{A}_i\mathbf{q}_i - \mathbf{b}_i$. Finally, $\mathbf{s}$ is computed as the normalized $[s_x^* \ s_y^* \ s_z^*]^\top$.

3) Computing Noise Variance σ^2 from Normal $\mathbf{s}$*:* After computing $\mathbf{s}$, we can obtain the following two equations based on the linear relationship of $\mathbf{s} \approx \mathbf{A}_i\mathbf{p}_i + \mathbf{b}_i$.

$$\mathbf{v}_2^\top(\mathbf{s} - \mathbf{b}_i) = \mathbf{v}_2^\top\mathbf{A}_i\mathbf{p}_i, \ \ \mathbf{v}_3^\top(\mathbf{s} - \mathbf{b}_i) = \mathbf{v}_3^\top\mathbf{A}_i\mathbf{p}_i. \qquad (59)$$

Moreover, $\hat{p}_{i1} = \mathbf{v}_1^\top\mathbf{p}_i$, $\hat{q}_{i1} = \mathbf{v}_1^\top\mathbf{q}_i$, and we set $\hat{p}_{i1} = \hat{q}_{i1}$ at Section VI-C2. Now using this relationship and (59), we can obtain $\mathbf{p}_i$ by solving the following linear equations,

$$\underbrace{\begin{bmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top\mathbf{A}_i \\ \mathbf{v}_3^\top\mathbf{A}_i \end{bmatrix}}_{\mathbf{A}_v} \mathbf{p}_i = \begin{bmatrix} \mathbf{v}_1^\top\mathbf{q}_i \\ \mathbf{v}_2^\top(\mathbf{s} - \mathbf{b}_i) \\ \mathbf{v}_3^\top(\mathbf{s} - \mathbf{b}_i) \end{bmatrix}. \qquad (60)$$

In order to solve (60) for $\mathbf{p}_i$, $\mathbf{A}_v$ should be invertible. We prove that $\mathbf{A}_v$ is invertible in Section 1.D of the supplement. After estimating $\mathbf{p}_i$, we can compute the empirical average noise variance of the point cloud using the red points in the given flat patch as follow:

$$\sigma^2 = \frac{1}{3n_r}\sum_{i=1}^{n_r}(\mathbf{q}_i - \mathbf{p}_i)^\top(\mathbf{q}_i - \mathbf{p}_i). \qquad (61)$$

For more reliable estimation for σ^2, we compute it for both red and blue node sets for each flat patch, and take the average over all identified flat patches in the given point cloud.

D. Discussion

One can see the simplicity of our proposed variance estimation method that involves only: i) the mean filters for ℓ_2-norm fidelity term, and ii) median filters, eigen-decomposition and matrix inversion of 3×3 matrices for ℓ_1-norm fidelity term. The bipartite graph approximation and computation of the surface normals for the red and blue nodes are more costly as described in Section IV-B, but this is performed necessarily as part of the denoising algorithm. Given that, the marginal computation overhead is small.

An alternative approach is to perform *principal component analysis* (PCA) [56] directly on the observed 3D points in a flat patch to obtain a best-fitted 2D plane in a ℓ_2-norm sense, then compute the Euclidean distances between the observed points and the plane to estimate noise variance σ^2. However, PCA is known to be fragile to sparse but large noise [23]. Hence, in the case of sparse Laplacian-like noise, PCA-based approach[15] would not perform well. We demonstrate this point empirically through experiments in Section VII.

VII. Experimental Results

We present comprehensive experiments to verify the effectiveness of the proposed algorithms in solving the point cloud denoising problem. There are three sets of results: i) weight parameter computation based on estimated noise variance, ii) noise variance estimation, and iii) denoising. All the experiments consider both non-sparse noise like Gaussian and sparse noise like Laplacian. Further, all the point cloud models are first rescaled, so that each tightly fits inside a bounding box with the same diagonal.

[15]There exist extensive works on *robust PCA* (RPCA) [57] that can handle large sparse noise, but these methods require iterative minimization of the nuclear norm, solved via *singular value decomposition* (SVD) of complexity $O(n^3)$.

For numerical comparisons of denoising results, we measure the point-to-point (C2C) error and point to plane (C2P) error [14] between ground truth and denoising points sets. For C2C error, we first measure the average of the squared Euclidean distances between ground truth points and their closest denoised points, and also that between the denoised points and their closest ground truth points. Then the smaller between these two measures is taken as C2C. In C2P error, we first measure the average of the squared Euclidean distances between ground truth points and tangent planes at their closest denoised points, and also that between the denoised points and tangent planes at their closest ground truth points. Then the smaller between these two measures is taken as the C2P.

Results for Weight Parameter Estimation: In order to validate the linear relationship between γ_{opt} and σ^2 (as stated in (39)) for Gaussian noise and Laplacian noise, we use six point cloud models from [43], four models from [58], and ten models from [59]. First, Gaussian / Laplacian noise with zero mean and standard deviation σ of 0.1, 0.15, ..., 0.5 is added to the 3D position of these clean point clouds. Then we denoise point cloud models with Gaussian noise following Section V-A and models with Laplacian noise following Section V-B for different γ values (γ from 0 to 1 with interval 0.01). For a given point cloud and a given noise level, γ_{opt} is chosen as the γ value that gives the minimum C2P between the denoised point cloud and its ground truth. (C2P is a metric to measure the denoising quality, which we will discuss soon).

We plot σ vs $\sqrt{\gamma_{opt}}$, averaged over different point cloud models, in Fig. 5. We observe that there is an approximate linear relationship between γ_{opt} and σ^2 as in (39). We can thus determine constant α in (39) empirically using Fig. 5. It should be noted that finding α is an off-line process and then we can determine γ_{opt} for a given noisy point cloud based on its estimated noise variance.

Results for Noise Variance Estimation: The proposed noise variance estimation method is compared with the PCA-based method mentioned in Section VI-D. Point cloud models we use are Bunny provided in [43], Gargoyle, DC, and Daratech provided in [23], [27]. To show the performance of noise variance estimation quantitatively, the relative percentage error (ϵ) in terms of the noise standard deviation (SD) is computed as follows:

$$\epsilon = \frac{|\sigma - \hat{\sigma}|}{\sigma} \times 100\%, \qquad (62)$$

where $\hat{\sigma}$ is the SD of the estimated noise and σ represents the SD of the actual noise.

Gaussian / Laplacian noise with zero mean and standard deviation σ of 0.1, 0.2, ..., 0.5 is added to the 3D position of the clean point clouds. The estimated noise variance (in terms of σ_e) and the relative percentage error are given in Table I (for Gaussian noise) and Table II (for Laplacian noise). On average, we observe that our estimation and PCA-based estimation for Gaussian noise have approximately similar performance. However, for Laplacian noise, the proposed method is more accurate than PCA-based scheme, with ϵ reduced by 43% on average.

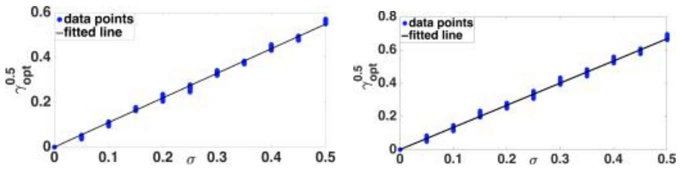
Denoising Results: The proposed point cloud denoising method is compared with four existing methods: APSS [16], RIMLS [18], AWLOP [22], the moving robust principle component analysis (MRPCA) algorithm [23], and our recently proposed GTV-based method [32]. APSS and RIMLS are implemented with MeshLab software [60], AWLOP is implemented with EAR software [22], and the source code of MRPCA is provided by the authors. Point cloud models we use are Bunny provided in [43], Gargoyle, DC, Daratech, Anchor, Lordquas, Fandisk, and Laurana provided in [23], [27]. We note that these models are different from the models that we used to determine α in (39). Both quantitative and visual comparisons are presented.

1) Gaussian Noise: Gaussian noise with zero mean and standard deviation of 0.2 and 0.4 is added to the 3D coordinates of the point cloud. Quantitative results in terms of C2C and C2P errors are shown in Table III (for $\sigma = 0.2$) and Table IV (for $\sigma = 0.4$), where the proposed method is shown to have the lowest C2C and C2P errors. For Gaussian noise, the proposed denoising method (*i.e.* in Section V-A) has two different approaches of solving (20): CG and GSF. The quantitative results show that the both approaches have approximately similar performance in terms of C2C and C2P. As shown in Table III and IV, C2C and C2P of the proposed method are clearly better than the competing schemes, with C2C and C2P reduced by 15% and 36% (on average), respectively.

Visual results for Fandisk model is shown in Fig. 6. The results of APSS and RIMLS schemes are over-smoothed, and the results of MRPCA scheme are distorted with some details lost. For the proposed method, the details are well preserved without over-smoothing. Additional visual results are given in Fig. 1 and 2 in the supplement.

2) Laplacian Noise: Laplacian noise with zero mean and standard deviation of 0.1 and 0.3 is added to the 3D coordinates of the point cloud. Quantitative results in terms of C2C and C2P errors are shown in Table V (for $\sigma = 0.1$) and Table VI (for $\sigma = 0.3$). The proposed method (*i.e.* in Section V-B) is shown to have the lowest C2C and C2P errors; our method is significantly better than the competing schemes, with C2C and C2P reduce by 20% and 43% (on average) respectively. To show our proposed method for ℓ_1-norm fidelity term is more appropriate to model Laplacian noise, we present denoising results using the method in Section V-A as well (in Table V and VI, we have reported the best results out of GC and GSF approaches in Section V-A). We observe that for Laplacian noise, the proposed method with ℓ_1-norm fidelity term substantially outperforms the method of ℓ_2-norm fidelity term, with C2C and C2P reduced by 12% and 22% (on average) respectively.

Visual results for Daratech model are shown in Fig. 7. For the proposed method, sharp features are well preserved without over-smoothing compared to the existing methods. Additional visual results are given in Fig. 3 and 4 in the supplement.



(a) for Gaussian noise (b) for Laplacian noise

Figure 5. σ vs $\sqrt{\gamma_{opt}}$ (a line is fitted based on minimum mean squire error).

VIII. Conclusion

We address the 3D point cloud denoising problem, where the fidelity term can be either ℓ_2- or ℓ_1-norm to accommodate different kinds of acquisition noise. We propose a reweighted graph Laplacian regularizer (RGLR) for the surface normals as a signal prior with desirable properties, namely rotation-invariant and promotion of piecewise smoothness. To obtain a linear relationship between 3D coordinates and surface normals, we first perform bipartite graph approximation. For ℓ_2-norm fidelity term, we iteratively solve a quadratic programming problem using conjugate gradient. For ℓ_1-norm fidelity term, we minimize the ℓ_1-ℓ_2-norm cost function using accelerated proximal gradient (APG). We derive computation-efficient mean and media filters to estimate noise variance for different noise types. Experimental results show that our algorithms outperform competing schemes visually and quantitatively.

References

[1] J. Shen, P. C. Su, S. C. S. Cheung, and J. Zhao, "Virtual mirror rendering with stationary RGB-D cameras and stored 3-D background," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3433–3448, 2013.

[2] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang, "Surfacenet: An end-to-end 3D neural network for multiview stereopsis," *arXiv preprint arXiv: 1708.01749*, 2017.

[3] C. Couprie, L. Grady, L. Najman, J. C. Pesquet, and H. Talbot, "Dual constrained TV-based regularization on graphs," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1246–1273, 2013.

[4] J. Pang and G. Cheung, "Graph Laplacian regularization for image denoising: Analysis in the continuous domain," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1770–1785, 2017.

[5] S. Chen, A. Sandryhaila, J. Moura, and J. Kovacevic, "Signal recovery on graphs: Variation minimization," *IEEE Trans. Signal Process.*, vol. 63, no.17, p. 4609–4624, September 2015.

[6] 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.

[7] G. Cheung, E. Magli, Y. Tanaka, and M. Ng, "Graph spectral image processing," *Proc. IEEE*, vol. 106, no.5, pp. 907–930, May 2018.

[8] J. Zeng, G. Cheung, and A. Ortega, "Bipartite approximation for graph wavelet signal decomposition," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5466–5480, Oct 2017.

[9] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," CMU, Tech. Rep., 1994.

[10] S. B. Thal, "Gershgorin's theorem for estimating eigenvalues," *Source:http://buzzard.ups.edu/courses/2007spring/projects/brakkenthal-paper.pdf*, 2007.

[11] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst, "Accelerated filtering on graphs using Lanczos method," *arXiv preprint:1509.04537*, 2015.

[12] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2013.

[13] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[14] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *ICIP*, Sept 2017, pp. 3460–3464.

Table I
NOISE ESTIMATION RESULTS IN TERMS OF $\hat{\sigma}$ AND ϵ (%) FOR GAUSSIAN NOISE

Actual σ	Estimated σ ($\hat{\sigma}$) and ϵ (%) respectively															
	Bunny				Gargoyle				DC				Daratech			
	Prop.		PCA		Prop.		PCA		Prop.		PCA		Prop.		PCA	
0.1	**0.112**	**12.0**	0.113	13.0	**0.112**	**12.0**	0.114	14.0	0.115	15.0	**0.114**	**14.0**	0.114	14.0	**0.112**	**12.0**
0.2	**0.211**	**5.5**	0.217	8.5	0.225	12.5	**0.223**	**11.5**	**0.184**	**8.0**	0.225	12.5	0.237	18.5	**0.233**	**16.5**
0.3	**0.332**	**10.7**	0.342	14.0	**0.323**	**7.7**	0.327	9.0	**0.273**	**9.0**	0.334	11.3	**0.344**	**14.7**	0.348	16.0
0.4	**0.446**	**11.5**	0.451	12.8	0.445	11.3	**0.443**	**10.8**	**0.362**	**9.5**	0.453	13.3	**0.446**	**11.5**	0.457	14.3
0.5	0.549	9.8	**0.545**	**9.0**	**0.547**	9.4	0.558	11.6	**0.450**	**10.0**	0.568	13.6	**0.539**	**7.8**	0.566	13.2

Table II
NOISE ESTIMATION RESULTS IN TERMS OF $\hat{\sigma}$ AND ϵ (%) FOR LAPLACIAN NOISE

Actual σ	Estimated σ ($\hat{\sigma}$) and ϵ (%) respectively															
	Bunny				Gargoyle				DC				Daratech			
	Prop.		PCA		Prop.		PCA		Prop.		PCA		Prop.		PCA	
0.1	**0.117**	**17.0**	0.123	23.0	**0.112**	**12.0**	0.119	19.0	**0.120**	**20.0**	0.124	24.0	**0.118**	**18.0**	0.124	24.0
0.2	**0.221**	**10.5**	0.244	22.0	**0.225**	**12.5**	0.239	19.5	**0.234**	**17.0**	0.245	22.5	**0.239**	**19.5**	0.253	26.5
0.3	**0.332**	**10.7**	0.372	24.0	**0.333**	**11.0**	0.387	29.0	**0.337**	**12.3**	0.384	28.0	**0.344**	**14.7**	0.384	28.0
0.4	**0.443**	**10.8**	0.496	24.0	**0.455**	**13.8**	0.501	25.3	**0.462**	**15.5**	0.524	31.0	**0.455**	**13.7**	0.537	34.3
0.5	**0.546**	**9.2**	0.587	17.4	**0.558**	**11.6**	0.601	20.2	**0.571**	**14.2**	0.627	25.4	**0.564**	**12.8**	0.639	27.8

Table III
DENOISING ACCURACY FOR GAUSSIAN NOISE ($\sigma = 0.2$) IN TERMS OF C2C AND C2P ($\times 10^{-2}$) RESPECTIVELY

Model	Noise		APSS		RIMLS		AWLOP		MRPCA		GTV		Proposed (Section V-A)			
													CG		GSF	
Bunny	0.266	2.81	0.209	0.474	0.219	0.891	0.262	2.49	0.222	1.108	0.202	0.483	0.193	0.467	**0.191**	**0.463**
Gargoyle	0.248	2.45	0.194	0.634	0.204	0.995	0.238	2.13	0.199	0.836	0.188	0.711	**0.173**	0.659	0.175	**0.611**
DC	0.249	2.47	0.191	0.454	0.203	0.876	0.246	2.40	0.193	0.535	0.188	0.451	**0.172**	**0.428**	0.178	0.439
Daratech	0.255	2.55	0.206	0.805	0.215	1.122	0.244	2.19	0.226	1.589	0.194	1.014	0.182	0.971	**0.179**	**0.937**
Anchor	0.258	2.67	0.202	0.532	0.207	0.678	0.245	2.06	0.196	0.275	0.191	0.201	**0.177**	**0.191**	0.178	0.195
Lordquas	0.253	2.55	0.205	0.690	0.210	0.909	0.243	1.61	0.215	1.194	0.202	0.702	0.188	0.612	**0.183**	**0.607**
Fandisk	0.296	3.35	0.279	2.426	0.259	1.454	0.287	2.88	0.248	1.201	0.214	1.137	**0.194**	1.012	0.197	**1.001**
Laurana	0.250	2.50	0.190	0.390	0.203	0.849	0.238	2.07	0.190	0.374	0.183	0.384	0.174	0.348	**0.169**	**0.319**

Table IV
DENOISING ACCURACY FOR GAUSSIAN NOISE ($\sigma = 0.4$) IN TERMS OF C2C AND C2P ($\times 10^{-2}$) RESPECTIVELY

Model	Noise		APSS		RIMLS		AWLOP		MRPCA		GTV		Proposed (Section V-A)			
													CG		GSF	
Bunny	0.377	6.23	0.254	1.66	0.273	2.59	0.360	5.14	0.243	1.420	0.244	1.435	**0.214**	**1.178**	0.217	1.207
Gargoyle	0.352	5.36	0.242	2.12	0.258	2.96	0.333	4.67	0.236	1.770	0.232	1.832	0.223	1.610	**0.218**	**1.471**
DC	0.354	5.50	0.233	1.65	0.257	2.78	0.330	4.61	0.227	1.310	0.218	1.294	0.201	1.087	**0.195**	**1.035**
Daratech	0.357	5.05	0.295	3.78	0.322	5.03	0.331	4.14	0.292	3.030	0.274	3.107	0.261	2.413	**0.259**	**2.139**
Anchor	0.367	5.82	0.244	1.60	0.249	1.84	0.326	4.07	0.228	0.760	0.215	0.251	**0.205**	**0.238**	0.208	0.245
Lordquas	0.372	5.97	0.262	1.71	0.280	2.58	0.284	2.44	0.258	1.580	0.247	1.321	**0.224**	**1.271**	0.228	1.295
Fandisk	0.495	9.69	0.404	3.76	0.401	3.57	0.473	8.35	0.378	1.870	0.351	1.935	0.329	**1.628**	**0.323**	1.647
Laurana	0.356	5.58	0.229	1.45	0.246	2.21	0.332	4.58	0.220	0.933	0.207	0.971	**0.187**	**0.781**	0.192	0.815

[15] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Trans. Vis. Comput. Graphics*, vol. 9, no. 1, pp. 3–15, Jan 2003.

[16] G. Guennebaud and M. Gross, "Algebraic point set surfaces," *ACM Trans. Graph.*, vol. 26, no. 3, p. 23, 2007.

[17] G. Guennebaud, M. Germann, and M. Gross, "Dynamic sampling and rendering of algebraic point set surfaces," in *Computer Graphics Forum*, vol. 27, no. 2, 2008, pp. 653–662.

[18] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," in *Computer Graphics Forum*, vol. 28, no. 2, 2009, pp. 493–501.

[19] Y. Sun, S. Schaefer, and W. Wang, "Denoising point sets via ℓ_0 minimization," *Comput Aided Geom. Des.*, vol. 35, pp. 2–15, 2015.

[20] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Trans. Graph.*, vol. 26, no. 3, p. 22, 2007.

[21] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, p. 176, 2009.

[22] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, p. 9, 2013.

[23] E. Mattei and A. Castrodad, "Point cloud denoising via moving RPCA," in *Computer Graphics Forum*, vol. 36, no. 8, 2017, pp. 123–137.

[24] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *CVPR*, vol. 2, June 2005, pp. 60–65.

[25] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug 2007.

[26] J. Digne, "Similarity based filtering of point clouds," in *CVPR Workshops*, June 2012, pp. 73–79.

[27] G. Rosman, A. Dubrovina, and R. Kimmel, "Patch-collaborative spectral point-cloud denoising," in *Computer Graphics Forum*, vol. 32, no. 8, 2013, pp. 1–12.

[28] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang, "3D point cloud denoising using graph Laplacian regularization of a low dimensional manifold model," *arXiv*, February 2018.

[29] 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.

[30] C. Duan, S. Chen, and J. Kovačević, "Weighted multi-projection: 3D point cloud denoising with estimated tangent planes," *arXiv preprint arXiv:1807.00253*, 2018.

[31] Y. Schoenenberger, J. Paratte, and P. Vandergheynst, "Graph-based denoising for time-varying point clouds," in *IEEE 3DTV-Conference*, 2015, pp. 1–4.

[32] C. Dinesh, G. Cheung, I. V. Bajic, and G. Yang, "Local 3D point cloud denoising via bipartite graph approximation & total variation," in *MMSP*, 2018.

[33] J. Huang and C. H. Menq, "Automatic data segmentation for geometric feature extraction from unorganized 3-d coordinate points," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 268–279, Jun 2001.

[34] K. Kanatani, *Statistical optimization for geometric computation: theory and practice*. Courier Corporation, 2005.

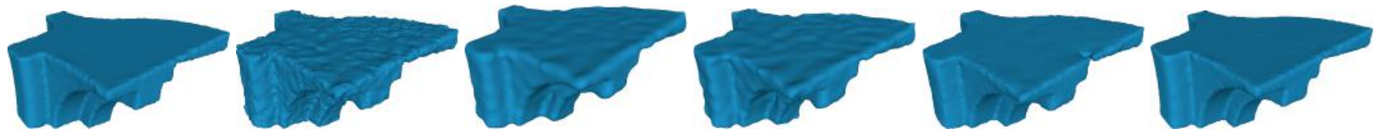
[35] D. OuYang and H. Y. Feng, "On the normal vector estimation for point

Table V
Denoising Accuracy for Laplacian Noise ($\sigma = 0.1$) in terms of C2C and C2P ($\times 10^{-3}$) Respectively

Model	Noise		APSS		RIMLS		AWLOP		MRPCA		GTV		Proposed (Section V-A)		Proposed (Section V-B)	
Bunny	0.144	9.24	0.124	2.60	0.134	5.07	0.140	7.38	0.129	3.88	0.119	2.18	0.114	1.95	**0.101**	**1.72**
Gargoyle	0.141	8.53	0.124	3.21	0.134	5.87	0.137	7.50	0.136	6.41	0.121	4.57	0.116	3.16	**0.104**	**2.63**
DC	0.141	8.66	0.120	2.30	0.130	4.65	0.136	6.97	0.126	3.77	0.117	2.28	0.113	2.19	**0.098**	**1.67**
Daratech	0.142	8.87	0.124	3.03	0.127	3.68	0.142	8.72	0.140	7.28	0.120	3.17	0.113	3.04	**0.101**	**2.38**
Anchor	0.142	8.97	0.123	3.30	0.128	3.96	0.139	7.64	0.119	1.71	0.121	3.11	0.116	1.95	**0.104**	**1.32**
Lordquas	0.137	8.23	0.125	4.05	0.131	5.48	0.135	7.48	0.124	3.90	0.119	3.80	0.113	3.74	**0.102**	**2.97**
Fandisk	0.145	9.15	0.138	6.83	0.138	6.91	0.144	8.27	0.127	3.15	0.131	4.27	0.126	4.17	**0.110**	**2.53**
Laurana	0.141	8.79	0.120	2.40	0.133	5.62	0.139	8.18	0.121	2.69	0.120	2.97	0.117	2.61	**0.105**	**2.07**

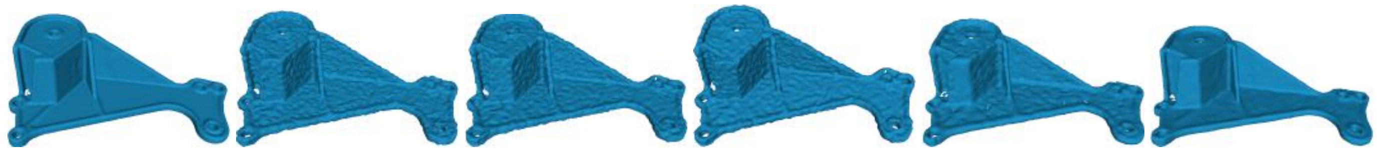
Table VI
Denoising Accuracy for Laplacian Noise ($\sigma = 0.3$) in terms of C2C and C2P ($\times 10^{-2}$) Respectively

Model	Noise		APSS		RIMLS		AWLOP		MRPCA		GTV		Proposed (Section V-A)		Proposed (Section V-B)	
Bunny	0.303	3.82	0.221	1.183	0.235	1.39	0.294	3.17	0.230	1.249	0.221	1.37	0.214	1.258	**0.193**	**0.875**
Gargoyle	0.284	3.30	0.208	1.199	0.221	1.61	0.270	2.91	0.208	1.103	0.205	1.20	0.197	1.083	**0.173**	**0.911**
DC	0.285	3.30	0.203	1.177	0.219	1.41	0.233	1.79	0.202	1.020	0.194	1.01	0.186	0.981	**0.153**	**0.704**
Daratech	0.287	3.17	0.250	1.832	0.243	2.06	0.272	2.71	0.234	1.622	0.227	1.71	0.221	1.688	**0.194**	**1.237**
Anchor	0.294	3.54	0.212	1.820	0.217	1.96	0.266	2.38	0.204	1.643	0.207	1.83	0.195	1.753	**0.168**	**1.325**
Lordquas	0.295	3.60	0.224	1.206	0.234	1.41	0.258	1.84	0.230	1.300	0.224	1.19	0.221	1.043	**0.208**	**1.018**
Fandisk	0.368	5.46	0.327	2.854	0.316	2.29	0.354	4.74	0.302	2.059	0.301	2.15	0.288	2.075	**0.259**	**1.694**
Laurana	0.287	3.38	0.200	0.648	0.214	1.18	0.268	2.70	0.198	0.549	0.201	1.13	0.192	0.524	**0.161**	**0.407**



(a) ground truth (b) noisy input (c) APSS (d) RIMLS (e) MRPCA (f) proposed

Figure 6. Denoising results illustration for Fandisk model (Gauusian noise with $\sigma = 0.4$); a surface is fitted over the point cloud for better visualization.



(a) ground truth (b) noisy input (c) APSS (d) RIMLS (e) MRPCA (f) proposed

Figure 7. Denoising results illustration for Daratech model ($\sigma = 0.3$); a surface is fitted over the point cloud for better visualization.

cloud data from smooth surfaces," *Computer-Aided Design*, vol. 37, no. 10, pp. 1071–1079, 2005.

[36] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Trans. Comput.*, vol. C-20, no. 6, pp. 623–629, June 1971.

[37] S. Jin, R. R. Lewis, and D. West, "A comparison of algorithms for vertex normal computation," *The Visual Computer*, vol. 21, no. 1-2, pp. 71–82, 2005.

[38] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[39] J. Wang, *Geometric structure of high-dimensional data and dimensionality reduction*. Springer, 2011.

[40] H. Rue and L. Held, *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[41] J. Duchi, "Derivations for linear algebra and optimization," [Online] https://web.stanford.edu/∼jduchi/projects/general_notes.pdf, 2007.

[42] M. Bona, *A walk through combinatorics: an introduction to enumeration and graph theory*, 3rd ed. World Scientific Publishing Company, 2002.

[43] M. Levoy, J. Gerth, B. Curless, and K. Pull, "The Stanford 3D scanning repository," [Online] https://graphics.stanford.edu/data/3Dscanrep/.

[44] A. Chambolle and J. Darbon, "On total variation minimization and surface evolution using parametric maximum flows," *International Journal of Computer Vision*, vol. 84, no. 3, pp. 1–28, 2009.

[45] T. F. Chan, G. H. Golub, and P. Mulet, "A nonlinear primal-dual method for total variation-based image restoration," *SIAM journal on scientific computing*, vol. 20, no. 6, pp. 1964–1977, 1999.

[46] I. Daubechies, G. Teschke, and L. Vese, "Iteratively solving linear inverse problems under general convex contraints," *Inverse Probl. Imaging*, vol. 1, no. 1, pp. 29–46, 2006.

[47] M. Nikolova, "A variational approach to remove outliers and impulse noise," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 99–120, 2004.

[48] S. Alliney, "A property of the minimum vectors of a regularizing functional defined by means of the absolute norm," *IEEE Trans. Signal Process.*, vol. 45, no. 4, pp. 913–917, 1997.

[49] 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 Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[50] E. S. Coakley and V. Rokhlin, "A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices," *Appl. Comput. Harmon. Anal.*, vol. 34, no. 3, pp. 379–414, 2013.

[51] P. Y. Chen and S. Liu, "Bias-variance tradeoff of graph Laplacian regularizer," *IEEE Signal Process. Lett.*, vol. 24, no. 8, pp. 1118–1122, 2017.

[52] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "ℓ_1-sparse reconstruction of sharp point set surfaces," *ACM Trans. Graph.*, vol. 29, no. 5, p. 135, 2010.

[53] C. H. Wu and H. H. Chang, "Superpixel-based image noise variance estimation with local statistical assessment," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, p. 38, 2015.

[54] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[55] K. Zhang, W. Zhu, and Y. Xu, "Hierarchical segmentation based point cloud attribute compression," in *ICASSP*, 2018.

[56] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[57] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, p. 11, 2011.

[58] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," in *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), m38673/M72012*, 2016.

[59] A. Nouri, C. Charrier, and O. Lézoray, “Technical report: Greyc 3D colored mesh database,” Normandie Université, Unicaen, EnsiCaen, CNRS, GREYC UMR 6072, Tech. Rep., 2017.

[60] 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*, 2008, pp. 129–136.