2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)

Graph-based Point Cloud Denoising

1st Xiang Gao, 2nd Wei Hu, 3rd Zongming Guo
Institute of Computer Science & Technology, Peking University, Beijing, China
gyshgx868@gmail.com {forhuwei, guozongming}@pku.edu.cn

***Abstract*—3D Point cloud data has attracted attention in various applications such as free-view rendering, heritage reconstruction and navigation. However, point clouds often suffer from noise, either from hardware or software causes. We propose an efficient point cloud denoising approach, where the geometry of the point cloud is naturally represented on graphs. We first divide noise in the point cloud into two categories: outlier and surface noise according to the distribution, and then remove them separately. Outliers are firstly removed based on the sparsity of the neighborhood. Next, we formulate the surface noise removal as an optimization problem regularized by graph-signal smoothness prior, which essentially tries to reconstruct the underlying geometry of the point cloud. Experimental results show that our approach significantly outperforms five competing methods.**

***Index Terms*—Point cloud, denoising, graph-signal smoothness prior, geometry, polynomial models**

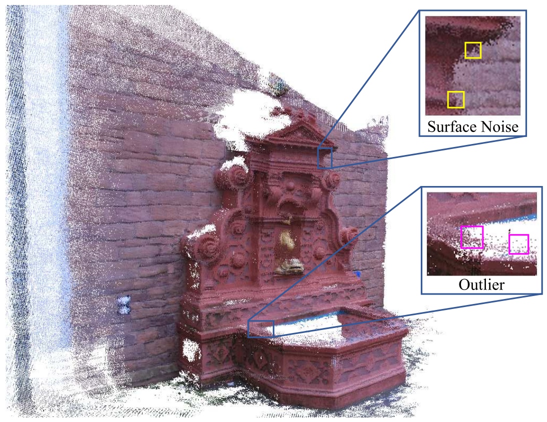

Fig. 1. `Fountain` in MVS dataset with outliers and surface noise.

I. Introduction

The maturity of depth sensing and 3D laser scanning techniques has enabled convenient acquisition of 3D point clouds, a natural representation for arbitrarily-shaped objects. Point clouds consist of a set of points, each of which has 3D coordinates and possibly attribute information such as color and normal. Because of the efficient representation, point clouds have been widely deployed in various fields, such as 3D immersive tele-presence, navigation for unmanned vehicles, and heritage reconstruction [1].

However, point clouds are often perturbed by noise, which comes from hardware, software or environmental causes. Hardware wise, noise occurs due to the inherent limitations of depth sensors or 3D laser scanners. Software wise, in the case of generating point clouds from algorithms, points may locate somewhere completely wrong due to imprecise triangulation (e.g., a false epipolar match). Environmentally, outliers may appear due to the surrounding contamination, such as dust in the air.

Many approaches are thus proposed for point cloud denoising. At present, the denoising method can be divided into two main categories: noise removal methods and surface smoothing methods. The main idea of the former is to detect the noise in point clouds via some characteristics and then *delete* them; the latter *moves* wrong points to correct positions. These methods usually perform filtering on the point cloud or move/project noisy points to a local approximate surface.

The most widely adopted noise removal methods are model-based methods and statistical methods. Model-based methods measure the distance from each point to a model (e.g., plane) that approximates the geometry of each region of the object, and detect whether a point is noise according to the distance [2], [3]. Statistical methods compute the number of neighbors at each point, and detect noisy points based on the sparsity [4]. However, they have two limitations: a) Model-based methods are unable to address point clouds with complex structure, because it is difficult to describe regions of complicated geometry with simple geometries. b) Statistical methods often destroy the sharp boundaries of point clouds.

Surface smoothing approaches mainly include the moving least squares (MLS) based methods and locally optimal projection (LOP) based methods. The idea of MLS-based methods is to project points onto an approximated smooth surface; LOP-based methods aim to produce a set of points that represent the underlying surface while enforcing a uniform distribution over the point cloud. Nevertheless, these methods are unsuitable for point clouds with large outliers.

In order to address the above problems, we first categorize noise in point clouds into two types based on the distribution of the noise (see Fig. 1, `Fountain` in MVS dataset[1]): a) outlier noise, which exhibits as points with completely wrong location, i.e., far away from the 3D surface and often with sparse neighborhood. b) surface noise, which appears close to the underlying surface and can be modelled as white Gaussian noise. We deal with the two types of noise separately, taking advantage of their unique characteristics.

This work was supported by National Natural Science Foundation of China under contract No. U1636206 and MSRA Collaborative Research under project ID FY18-Research-Sponsorship-029.

[1]https://lts2.epfl.ch/research/reproducible-research/graph-based-point-cloud-denoising/

In particular, we first remove outlier noise based on the sparsity of the neighborhood, and then deal with surface noise. In order to efficiently represent the point cloud, we treat each point as a vertex in a graph, and define the point-to-surface distance of each point as the signal on the graph, i.e., *graph signal* [5]. Specifically, we first segment the input point cloud into patches by octree division [6] and normalized cuts [7] segmentation, so as to reduce the computation complexity, as well as to acquire homogeneous geometry within each patch, which enables efficient subsequent surface fitting. Next, for each patch, we attenuate the surface noise further, by casting the denoising problem as a point-to-surface distance optimization problem, which is regularized by a *graph-signal smoothness prior* [5]. The idea is to approximate the underlying surface by *high-order polynomial surface fitting*, and remove points whose distance is away from the approximated surface to some extent while enforcing the smoothness of the surface. Experimental results show that our scheme significantly outperforms previous competing methods.

The outline of this paper is as follows. We first review previous point cloud denoising methods in Section II. Then we introduce basic concepts in spectral graph theory and graph-signal smoothness prior in Section III, which will be deployed later. In Section IV, we elaborate on our algorithm, including outlier removal and surface noise removal. Finally, experimental results and conclusion are presented in Section V and VI, respectively.

II. Related Work

Most previous point cloud denoising methods can be classified into two categories: noise removal methods and surface smoothing methods.

There are two major approaches in noise removal methods: model-based methods and statistical methods.

Model-based methods. These methods utilize the fact that noise is away from the surface of the object. As the underlying surface is unknown, the common idea is to approximate with some model (e.g. plane, sphere, etc.), and compute the distance of each point from the model surface. Points with large distance will be detected as noise and removed. A progressive plane algorithm is designed in [2], which proposes a theorem to describe the characteristics of a point set on a plane. Then the plane is acquired by averaging the three-dimensional coordinates and normal of a given point set using this theorem. Based on [2], a hybrid algorithm is proposed in [3], which constructs a progressive plane using least squares plane fitting algorithm and computes the distance from each point to the plane. The common problem of these methods is that details will lose in regions with complicated geometry, because it is difficult to approximate complex regions with simple models.

Statistical methods. These methods are often based on the number of neighbors or the distance distribution of each point to its neighbors in the input dataset. [4] proposes a statistical outlier removal (SOR) approach that computes the mean distance from each point to all its neighbors. By assuming that the resulting distribution is Gaussian, all points whose mean distances are outside an interval defined by the mean and standard deviation of global distances are considered as noise and removed from the dataset. Another widely used method is to compute the number of neighbors at each point in a radius, and define a global threshold. All points whose number of neighbors are less than the threshold will be considered as noise. This approach is called Radius Outlier Removal (ROR), and implemented in Point Cloud Library (PCL)[2].

Surface smoothing methods mainly include MLS-based methods and LOP-based methods.

MLS-based methods. MLS-based methods usually deploy a smooth surface to fit the point cloud and then project the points of the point cloud onto the fitted surface. [8] uses the MLS projection operator proposed by Levin [9] to calculate the optimal MLS surface of the point cloud, and moved the points around the surface to the MLS surface. [10] proposes a MLS-based spherical fitting denoising method (APSS). Compared with the aforementioned MLS projection-based algorithm, this method improves the stability at low sampling rate and high curvature. [11] proposes an algorithm based on improved MLS and local kernel regression to smooth the point cloud surface (RIMLS). However, these MLS-based methods are very sensitive to outliers.

LOP-based methods. The widely known locally optimal projection (LOP) [12] aims to produce a set of points to represent the underlying surface while enforcing a uniform distribution over the input point cloud. Weighted LOP (WLOP) [13] provides a more uniformly distributed output than LOP by adapting a repulse term to the local density. Further, anisotropic WLOP (AWLOP) [14] modifies WLOP with an anisotropic weighting function so as to preserve sharp features better. Nevertheless, LOP-based methods often suffer from over-smoothing.

Besides, another efficient algorithm is MRPCA [15]. The idea is to solve a minimization problem, and the positions of each point will be updated. Sharp features are well preserved via a weighted l_1 minimization. It is sensitive to outliers though. Note that, this approach does not exactly fall within either of the two above categories.

The proposed method belongs to the first category—the noise removal methods. Similar to the statistical methods, we also use ROR approach to remove the outliers. However, unlike the statistical methods, we further represent the point cloud on graphs and deploy a graph-signal smoothness prior for optimization.

III. Spectral Graph Theory

We first provide a review on basic concepts in spectral graph theory [16] that will be utilized in our denoising approach, including graph, graph Laplacian and graph-signal smoothness prior.

A. Graph and Graph Laplacian

We consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ composed of a vertex set $\mathcal{V}$ of cardinality $|\mathcal{V}| = n$, an edge set $\mathcal{E}$

[2]https://github.com/PointCloudLibrary/pcl

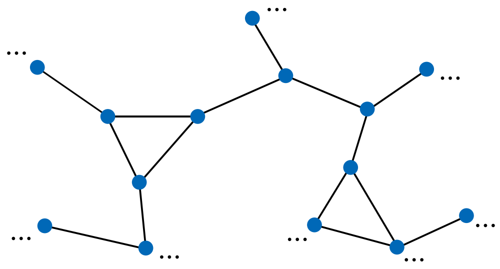

Fig. 2. A k-NN graph constructed when $k = 3$. The connections of boundary vertices are omitted.

connecting vertices, and a weighted *adjacency matrix* $\mathbf{W}$. $\mathbf{W}$ is a real symmetric $n \times n$ matrix, where $w_{i,j}$ is the weight assigned to the edge (i, j) connecting vertices i and j. We assume non-negative weights, *i.e.*, $w_{i,j} \geq 0$.

The Laplacian matrix, defined from the adjacency matrix, can be used to uncover many useful properties of a graph. Among different variants of Laplacian matrices, the *combinatorial graph Laplacian* used in [17]–[19] is defined as $\mathcal{L} := \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the *degree matrix*—a diagonal matrix where $d_{i,i} = \sum_{j=1}^{n} w_{i,j}$.

B. Graph-Signal Smoothness Prior

Graph signal refers to data that resides on the vertices of a graph, such as social, transportation, sensor, and neuronal networks. For example, if we construct a k-NN graph on the point cloud, then the coordinates of each point can be treated as graph signal defined on the k-NN graph, as shown in Fig. 2.

A graph signal $\mathbf{z}$ defined on a graph $\mathcal{G}$ is smooth with respect to the topology of $\mathcal{G}$ if

$$\sum_{i \sim j} w_{i,j}(z_i - z_j)^2 < \epsilon, \quad \forall i, j \tag{1}$$

where ϵ is a small positive scalar, and $i \sim j$ denotes two vertices i and j are one-hop neighbors in the graph. In order to satisfy (1), z_i and z_j have to be similar for a large edge weight $w_{i,j}$, and could be quite different for a small $w_{i,j}$. Hence, (1) enforces $\mathbf{z}$ to adapt to the topology of $\mathcal{G}$, which is thus coined *graph-signal smoothness prior*.

As $\mathbf{z}^T \mathcal{L} \mathbf{z} = \sum\limits_{i \sim j} w_{i,j}(z_i - z_j)^2$ [20], (1) is concisely written as $\mathbf{z}^T \mathcal{L} \mathbf{z} < \epsilon$ in the sequel. This prior will be deployed in our problem formulation of surface noise removal, as discussed in Section IV.

IV. The Proposed Algorithm

Having introduced some basic spectral graph concepts, we now elaborate on the proposed point cloud denoising approach. As shown in Fig. 3, the input is a noisy point cloud denoted by $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ with $\mathbf{p}_i \in R^3$ meaning the coordinates of the i-th point. The input then goes through two main steps: outlier removal and surface noise removal.

Outliers are firstly removed depending on the sparsity of the neighborhood. Next, we remove the surface noise as follows: we first divide the point cloud into voxels using octree, and apply normalized cuts to split each voxel into a number of patches. Then we fit each patch to a surface with polynomial models, and formulate surface noise removal as a convex optimization problem so as to smooth the surface. Finally, each processed patch is collected to form the reconstructed point cloud.

Note that detailed parameter settings in our experiments will be presented in Section V.

A. Noise Distribution Assumption

The noise distribution is not easy to assume due to the variant equipments to acquire point clouds. At present, many researchers have shown through statistical experiments that the point cloud noise generated from the relevant 3D scanning equipment follows Gaussian distribution, such as Microsoft Kinect [21], 3D laser scanner [22], etc. Hence, in our experiments we assume that the noise distribution follows Gaussian distribution.

B. Outlier Removal

We first perform outlier removal on the entire point cloud. As shown in Fig. 1, outliers have the unique characteristics that the neighborhood of each outlier is sparse, i.e, the density of outliers is much smaller than the actual point cloud. Hence, we detect a point as an outlier if its density is below a threshold.

Specifically, for each point p_i in $\mathbf{P}$, we select a radius r, and form a sphere centering at p_i with radius r. Assuming the density of $\mathbf{P}$ is ρ, we empirically set r to $m\rho$, where m is a scaling parameter. Then we calculate the number of points in this sphere, denoted by u_i. After computing u_i for each point p_i, we compute the average size of all the spheres $\bar{u}(\bar{u} = \frac{\sum_{i=1}^{n} u_i}{n})$. As outliers have lower density, a point p_j with $u_j < \bar{u}$ will be detected as an outlier and thus removed.

C. Surface Noise Removal

Having removed outliers, we now attenuate surface noise, which generally float on the surface of the object. Because these points are quite close to the object, the denoising method in the previous step has no effect. Instead, we propose to approximate the underlying surface of the object with high-order polynomial surface, optimize the distance from each point to the approximated surface based on graph-signal smoothness prior, and then remove points with significant changes in the point-to-surface distance after optimization, as they are very likely to be surface noise.

1) Surface Approximation: For the efficiency of surface approximation, we first split the point cloud $\mathbf{P}$ into voxels using the octree structure, i.e., $\mathbf{P} = \{\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_l\}$. Then we further segment each voxel $\mathbf{V}_i$ into 3D patches via normalized cuts, i.e., $\mathbf{V}_i = \{\mathrm{s}_1, \mathrm{s}_2, ..., \mathrm{s}_n\}$. On one hand, this significantly reduces the computation complexity to perform denoising within each voxel. On the other hand, the segments resulting from normalized cuts mostly exhibit homogeneous geometry, which facilitates the subsequent surface approximation and optimization.

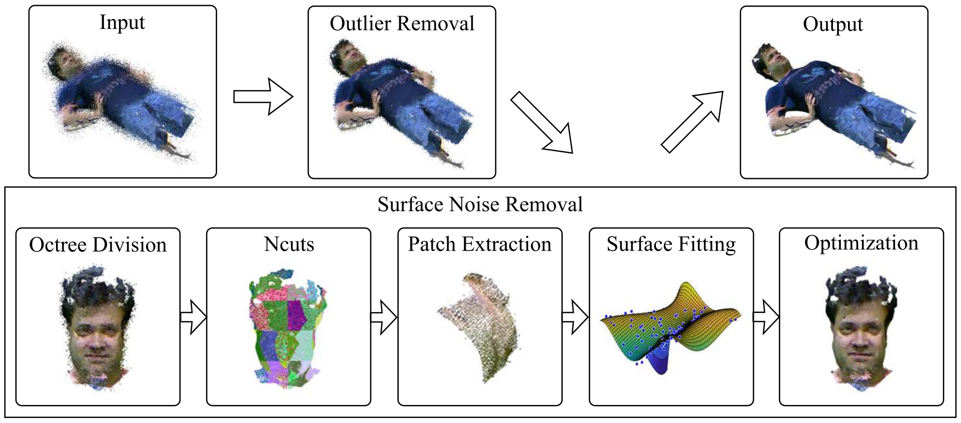

Fig. 3. The flowchart of the proposed point cloud denoising algorithm.

Secondly, we fit each segment $\mathbf{s}_i$ with high-order *polynomial surface* denoted by $\hat{\mathbf{s}_i}$, which approximates the underlying geometry of the point cloud. We use the least squares method to fit the surface with a high-order polynomial surface:

$$z = \sum_{p=0}^{k} \sum_{q=p}^{k} c_{p,q} x^p y^{q-p}, \tag{2}$$

where k is the highest order of this polynomial, and $c_{p,q}$ is the coefficient for each term we need to solve. The surface equation has $\frac{1}{2}(k+1)(k+2)$ unknown numbers.

In order to get the high-order polynomial surface, we need to bring the point coordinates into (2) to solve the equation set. If we have n points in a patch, the equation set can be solved only when $\frac{1}{2}(k+1)(k+2) \leq n$. Hence, we use the number of points n to decide the highest order. Finally, we get the surface $\hat{\mathbf{s}_i}$.

2) Problem Formulation: Next, we cast the denoising problem as the problem of smoothing point-to-surface distance. Here, the point-to-surface distance means the distance between each point and the corresponding approximate surface. Instead of taking the coordinates of each point as the signal, we consider the distance from each point p_i to the previously approximated surface $s\hat{\mathbf{s}}_j$ as the signal f_i, i.e., $f_i = e(p_i, \hat{\mathbf{s}_j})$, where e denotes the Euclidean distance. As the surface noise can be modelled as white Gaussian noise, the distance signal f_i also follows Gaussian distribution.

Since we fit with a high-order polynomial surface, the computation complexity is too high if the point-to-surface distance is accurately solved via partial differential equations, especially when the size of the point cloud is large. Hence, we develop an efficient approach to approximate the point-to-surface distances instead.

First, we compute the projection point $p_i^{'}$ from $p_i(x_i, y_i, z_i)$ to the surface $\hat{\mathbf{s}_j}$. For each patch, we calculate the variance of the coordinates of all points in the patch on the three axes $\{x, y, z\}$, and select a coordinate axis with the smallest variance as the projection direction. Assuming we project along the z-axis, the coordinates of $p_i^{'}$ is $(x_i, y_i, z_i^{'})$. Then we compute the distance d between p_i and $p_i^{'}$, and further choose $\frac{d}{l}(l \in Z^+)$ as a step to generate some points around $p_i^{'}$ on $\hat{\mathbf{s}_j}$:

$$p(x_i + \alpha_1 \frac{d}{l}, y_i + \alpha_2 \frac{d}{l}, \hat{s_j}(x_i + \alpha_1 \frac{d}{l}, y_i + \alpha_2 \frac{d}{l})), \quad l \in Z^+, \alpha_1, \alpha_2 \in R \tag{3}$$

where α_1 and α_2 are two parameters. The minimum distance from p_i to these points is selected as the approximation to f_i.

Having computed the point-to-surface distance f_i for each point p_i, we construct a k-NN graph on the distance signal $\mathbf{f}$, where each vertex is connected to its k nearest neighbors with an associated weight on the connecting edge. We define the edge weight between f_i and f_j using the thresholded Gaussian kernel:

$$w_{i,j} = \begin{cases} \exp(-\beta \|f_i - f_j\|^2), & p_i \sim p_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where β is a scalar parameter.

Once the graph is constructed over a patch, we compute the corresponding Laplacian matrix $\mathcal{L}$ and enforce the distance signal to be smooth with respect to this graph via a graph-signal smoothness prior, as introduced in Section III. In particular, this problem is formulated as

$$\min_{\mathbf{z}} \|\mathbf{z} - \mathbf{f}\|_2^2 + \tau_1 \mathbf{z}^T \mathcal{L} \mathbf{z} + \tau_2 \|\mathbf{z}\|_2^2, \tag{5}$$

where $\mathbf{z}$ is the desired distance signal, τ_1 and τ_2 are two parameters. The first term in (5) is a fidelity term, which constrains the denoised signal to be close to the observation. The second term is the graph-signal smoothness prior, which enforces the distance signal to be smooth, assuming that the underlying geometry in each segment is homogeneous. Further, we introduce the third term to minimize the distance from the point to the approximated surface, since a point belonging to the underlying surface should be quite close to the approximated one, i.e., the distance is close to 0.

Taking derivative of (5) with respect to $\mathbf{z}$, we have the closed-form solution:

$$\hat{\mathbf{z}} = ((1 + \tau_2)\mathbf{I} + \tau_1 \mathcal{L})^{-1} \mathbf{f}, \tag{6}$$

where $\mathbf{I}$ is an identity matrix. (5) is thus solved optimally and efficiently.

3) Noise Removal: Finally, we remove points which have large changes in the point-to-surface distance after optimization. This is because the point-to-surface distance of surface noise varies significantly after optimization due to the second and third priors, while that of an original point almost remains the same. Mathematically, for a point p_i, if $|\hat{z_i} - f_i| > h$, where h is a threshold, p_i is treated as surface noise and removed.

How to choose the threshold h remains a question. According to our assumption, f_i follows Gaussian distribution. Then $|\hat{z_i} - f_i|$ also follows Gaussian distribution, i.e., $|\hat{z_i} - f_i| \sim \mathcal{N}(\mu, \sigma)$, where μ and σ are the mean and variance respectively. Hence, h is assigned according to the mean and variance as follows:

$$h = \mu + \sigma. \tag{7}$$

4) Spectral Domain Analysis: We shed light on the solution in (6) in the spectral domain. Since the graph Laplacian $\mathcal{L}$ is a real symmetric matrix, it admits a set of real eigenvalues $\{\lambda_l\}_{l=0,1,...,n-1}, \lambda_0 = 0 \leq \lambda_1 \leq ...\lambda_{n-1}$, with a complete set of orthonormal eigenvectors $\{\psi_l\}_{l=0,1,...,n-1}$, *i.e.*, $\mathcal{L}\psi_l = \lambda_l \psi_l$, for $l = 0, 1, ..., n-1$. Note that the eigenvalues $\{\lambda_l\}$ are known as the spectrum of the graph. Then the filtering kernel in (6) can be written in the spectral domain as

$$\hat{\lambda_l} = \frac{1}{1 + \tau_2 + \tau_1 \lambda_l}. \tag{8}$$

As $\lambda_0 = 0$, $\hat{\lambda_0} = \frac{1}{1+\tau_2}$. As λ_l increases, $\hat{\lambda_l}$ decreases, which corresponds to stronger filtering strength. Since larger λ_l corresponds to higher frequency, stronger filtering is performed on higher frequency, thus attenuating noise more as noise generally lies in the subband of higher frequency.

V. Experimental Results

A. Experimental Setup

We evaluate the proposed algorithm by testing on several point cloud datasets, including `Andrew` and `David` from Microsoft Voxelized Dataset[3], `Alex` and `Dimitris` from Visual Computing Lab[4] and `Boy`[5]. To test the limitation of our algorithm, Additive White Gaussian Noise is added to the point cloud data with SNR= $\{10, 15, 20, 25, 30, 35\}$. Further, we compare the proposed approach with five competing algorithms, including two noise removal methods: SOR [4] and ROR, and three smoothing methods: MRPCA [15], APSS [10] and RIMLS [11].

B. Implementation Details

Outlier removal. In this step, we compute ρ as the average distance between two points, and set $m = 3$.

Surface noise removal. We first use octree to split the point cloud into small voxels. In our experiments, the number of partitions is not fixed, while the size of each dimension of a voxel is set as $\max(x_{max} - x_{min}, y_{max} - y_{min}, z_{max} - z_{min})/k$, where k is a command-line parameter that is adjusted according to the memory of the computer, and $x_{max}...z_{min}$ represents the maximum/minimum coordinates of points. In the next step, we set the number of patches for normalized cuts to 32. In (3), we set $l = 5$, then we choose α_1, α_2 from the set $\{1, 2, 3, 4, 5\}$. In (5), we simply set the optimization parameters $\tau_1 = \tau_2 = 1$.

C. Experimental Results

It is nontrivial to measure the geometry distortion of point clouds objectively. We apply the geometric distortion metrics `GPSNR` in [23]. Note that `GPSNR` could be negative, depending on the assignment of the peak value [23].

Table I, Table II and Table III list the `GPSNR` value when the SNR of the input noisy point cloud is $\{25, 30, 35\}$ respectively. We see that when the SNR level is high, i.e., when the noise variance is small, our method has competitive results with MRPCA. When the SNR level is low, we mostly outperform all the other methods. This is because smoothing methods are more sensitive to outliers and large noise variance. In comparison, our method still produces satisfactory results at high noise levels.

Fig. 4 and Fig. 5 show the visual results of comparing our method with ROR, SOR and MRPCA at noise level $SNR = 35$. We see that our results are cleaner than the results of these methods, even for noise with large variance.

VI. Conclusion

We propose an efficient point cloud denoising approach, leveraging on graph signal processing. We classify noise into

TABLE I
Experimental comparison for SNR= 25 (in `GPSNR`)

	ROR	SOR	APSS	RIMLS	MRPCA	Ours
Alex	23.35	20.67	9.17	13.27	16.14	**23.41**
Andrew	27.29	23.41	-3.98	8.68	9.34	**31.32**
Boy	21.67	19.42	4.49	4.40	5.29	**24.03**
David	**27.61**	27.26	-12.28	10.80	11.47	27.09
Dimitris	22.59	24.73	21.33	13.80	16.44	**26.92**

TABLE II
Experimental comparison for SNR= 30 (in `GPSNR`)

	ROR	SOR	APSS	RIMLS	MRPCA	Ours
Alex	**23.28**	21.01	4.01	18.13	22.52	23.19
Andrew	26.60	21.31	14.54	13.79	15.03	**30.28**
Boy	21.25	14.97	5.12	9.14	10.64	**23.29**
David	**27.53**	23.28	-8.26	15.96	17.10	26.95
Dimitris	22.89	23.09	12.47	18.59	24.43	**25.91**

TABLE III
Experimental comparison for SNR= 35 (in `GPSNR`)

	ROR	SOR	APSS	RIMLS	MRPCA	Ours
Alex	23.49	22.17	18.66	13.77	**28.25**	23.34
Andrew	26.66	24.31	20.79	20.36	20.84	**29.71**
Boy	21.09	16.75	17.15	16.06	16.19	**24.49**
David	27.61	25.82	23.21	22.46	22.88	**32.36**
Dimitris	24.50	24.61	30.55	30.62	**32.08**	28.20

[3]https://jpeg.org/plenodb/pc/microsoft/
[4]http://vcl.iti.gr/dataset/reconstruction/
[5]http://www.kscan3d.com/gallery/

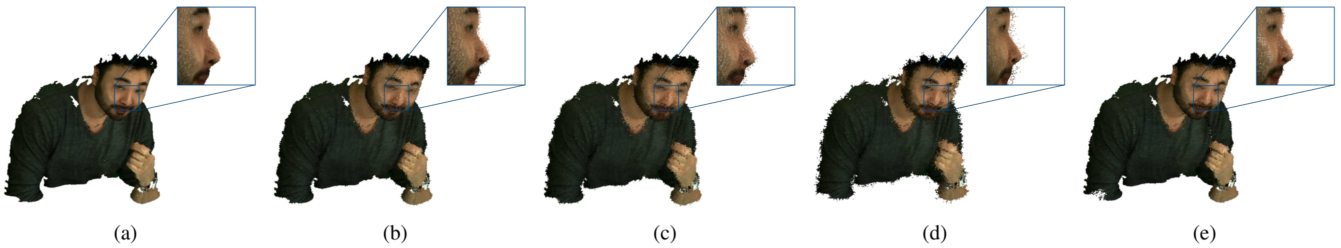

Fig. 4. Comparison results with SNR= 35 for `David`: (a) The ground truth; (b) The denoised point cloud by ROR; (c) The denoised result by SOR; (d) The denoised result by MRPCA; (e) The denoised result by our algorithm.

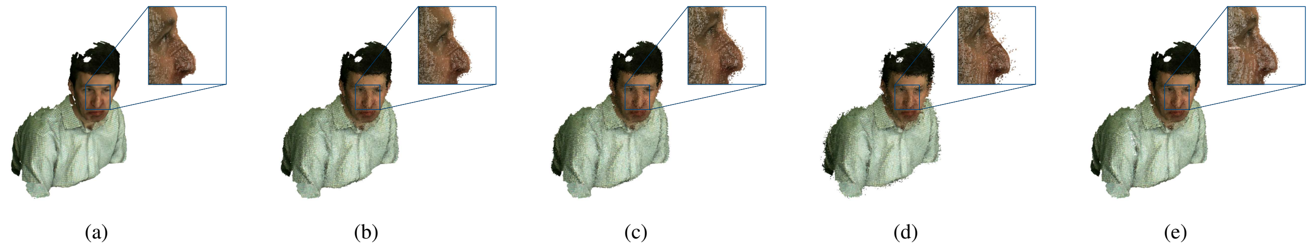

Fig. 5. Comparison results with SNR= 35 for `Andrew`: (a) The ground truth; (b) The denoised point cloud by ROR; (c) The denoised result by SOR; (d) The denoised result by MRPCA; (e) The denoised result by our algorithm.

outliers and surface noise based on the distribution, and then remove them separately. Firstly, points are detected as outliers and removed if the neighborhood is sparse to some extent. Secondly, the point cloud is segmented into patches, each of which is approximated with a polynomial surface. The distance of each point to the approximated surface is then treated as graph signal and go through convex optimization with a graph-signal smoothness prior. Points with large changes in the point-to-surface distance after optimization are detected as surface noise and removed. Experimental results show that our algorithm outperforms five competing methods significantly. Future work includes extending the proposed denoising for static point clouds to denoising for dynamic point clouds.

REFERENCES

[1] C. Tulvan, R. Mekuria, and Z. Li, "Use cases for point cloud compression (pcc)," in *ISO/IEC JTC1/SC29/WG11 (MPEG) output document N16331*, June 2016.

[2] W. Huang, Y. Li, P. Wen, and X. Wu, "Algorithm for 3d point cloud denoising," in *International Conference on Genetic and Evolutionary Computing*, 2009, pp. 574–577.

[3] Y. Fu and J. Zhai, "Research on scattered points cloud denoising algorithm," in *IEEE International Conference on Signal Processing, Communications and Computing*, 2015, pp. 1–5.

[4] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," in *Robotics and Autonomous Systems*, 2008, pp. 927–941.

[5] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs," in *IEEE Signal Processing Magazine*, May 2013, pp. 83–98.

[6] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Eurographics Symposium on Point-Based Graphics*, 2006, p. 111120.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, August 2000, vol. 22, no.8.

[8] M. Alexa, J. Behr, D. Cohenor, S. Fleishman, D. Levin, and C. T. Silva, "Point set surfaces," in *Surface Reconstruction*, 2001.

[9] D. Levin, "Mesh-independent surface interpolation," in *Geometric Modeling for Scientific Visualization*, 2004, pp. 37–49.

[10] G. Guennebaud and M. Gross, "Algebraic point set surfaces," 2007, vol. 26, p. 23.

[11] A. C. Oztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on nonlinear kernel regression," 2010, vol. 28, pp. 493–501.

[12] Y. Lipman, D. Cohen-or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," in *ACM Transactions on Graphics (TOG)*, 2007, p. 22.

[13] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-or, "Consolidation of unorganized point clouds for surface reconstruction," in *ACM transactions on graphics (TOG)*, 2009, p. 176.

[14] H. Huang, S. Wu, M. Gong, D. Cohen-or, U. Ascher, and H. R. Zhang, "Edge-aware point set resampling," in *ACM transactions on graphics (TOG)*, 2013, p. 9.

[15] E. Mattei and A. Castrodad, "Point cloud denoising via moving rpca," in *Computer Graphics Forum*.

[16] F. K. Chung, "Spectral graph theory," in *American Mathematical Society*, 1997.

[17] G. Shen, W.-S. Kim, S.K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[18] W. Hu, G. Cheung, X. Li, and O. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[19] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multi-resolution graph Fourier transform for compression of piecewise smooth images," in *IEEE Transactions on Image Processing*, January 2015, vol. 24, NO. 1, pp. 419–433.

[20] D. A. Spielman, "Lecture 2 of spectral graph theory and its applications," September 2004.

[21] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference*, Oct 2012, pp. 524–530.

[22] X. Sun, P. L. Rosin, R. R. Martin, and F. C. Langbein, "Noise in 3d laser range scanner data," in *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference*, 2008, pp. 37–45.

[23] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *IEEE International Conference on Image Processing*, Beijing, China, September 2017.