Robust Feature-Preserving Denoising of 3D Point Clouds

Sk. Mohammadul Haque, Venu Madhav Govindu
Indian Institute of Science
Bengaluru, India
{smhaque,venu}@ee.iisc.ernet.in

Abstract

The increased availability of point cloud data in recent years has lead to a concomitant requirement for high quality denoising methods. This is particularly the case with data obtained using depth cameras or from multi-view stereo reconstruction as both approaches result in noisy point clouds and include significant outliers. Most of the available denoising methods in the literature are not sufficiently robust to outliers and/or are unable to preserve fine-scale 3D features in the denoised representations. In this paper we propose an approach to point cloud denoising that is both robust to outliers and capable of preserving fine-scale 3D features. We identify and remove outliers by utilising a dissimilarity measure based on point positions and their corresponding normals. Subsequently, we use a robust approach to estimate surface point positions in a manner designed to preserve sharp and fine-scale 3D features. We demonstrate the efficacy of our approach and compare with similar methods in the literature by means of experiments on synthetic and real data including large-scale 3D reconstructions of heritage monuments.

1. Introduction

In recent years, it has become significantly easier to obtain 3D representations of real world scenes using either depth scanners [12] or dense multi-view stereo methods [6] that work on RGB images of a scene. In most cases, the surface or scene being scanned is estimated in the form of a set of unstructured 3D points known as a point cloud. While obtaining such 3D representations has become easier, owing to the limitations of either the sensors or the estimation methods used, the point clouds are invariable corrupted with significant amounts of noise as well as outliers. The denoising of point clouds is inherently difficult since point sets do not directly provide information about the surface topology. The problem of accurately estimating an underlying surface representation is made even more difficult due to the presence of outliers. In the case where outliers are present, we need to identify and remove them before further processing of the point cloud data since they do not contain any information of the surface and would deteriorate the performance of any denoising approach.

In this paper we present a novel approach to robustly denoise point clouds while preserving fine-scale features. We first develop an approach that aggregates comparisons of individual points in a neighbourhood that enables us to identify and remove outliers. Subsequently, we robustly denoise the 3D points on the surface. Our approach to such 3D point repositioning encourages the careful delineation and preservation of sharp and fine-scale 3D features of the surface.

2. Related work

In the past two decades denoising of point clouds has been extensively studied and some of the most popular approaches of relevance to us are [9, 1, 17, 7]. However, they generate smooth surfaces and blur the fine-scale surface details. Moreover these methods are not designed to work in the presence of outliers. Subsequent methods that attempted to address these shortcomings include the MLS method of [5] where the authors have used an additional surface classification step to define a set of smooth surfaces and hence avoid smoothing sharp features. In [14], the authors propose a robust implicit MLS method (henceforth RIMLS) by using kernel-based robust statistics and incorporating it into the implicit MLS method. More recently [18] introduced an ℓ_0-norm minimisation approach for point cloud denoising. However, we have found that ℓ_0-norm minimisation results in piecewise flat surfaces and does not perform well on natural data in presence of moderate level of noise. [19] presents a simultaneous denoising and mesh topology optimisation for surface reconstruction. In [20], the authors first generate an intermediate mesh from the point cloud and then use bilateral mesh filtering to denoise it. However, bilateral mesh filtering usually blurs sharp features when input noise is moderate or

higher. [10] proposes a denoising method specifically applicable to point clouds obtained from multiple view stereo. Other works like [3, 21] use non-local filtering. While we will discuss our method later, here we remark that our method is similar to RIMLS [14] but there are also some important differences. These denoising methods cannot accurately recover well-defined fine-scale and sharp surface features like edges and corners. In [18], the authors use an additional step to recover such sharp features in the denoised surface.

In our method, in addition to an effective procedure for the detection of outliers, we also show that the aforementioned sharp features are recovered naturally in the denoising step without requiring an additional step. The rest of the paper is organised as follows: Section 3 describes the details of our method whereas Section 4 present comparative results on both synthetic and real datasets. In Section 5 we present some concluding remarks.

3. Our method

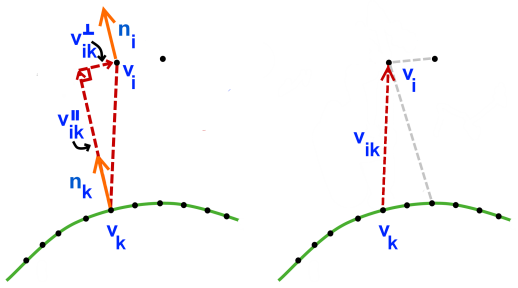
In this Section we present the details of the steps involved in our robust point cloud denoising method. Our method consists of the following sequence of procedures:

1. Robust outlier detection and removal (Section 3.1)
2. Bilateral normal mollification (Section 3.2)
3. Point set repositioning (Section 3.3)

In the robust outlier detection and removal step, the outliers are detected and removed based on an initial estimate of the point normals and the ℓ_2 distances between points in the point cloud. In the bilateral normal mollification step, the initial estimates of the point normals are mollified. Finally, in the point set repositioning step, the point set is repositioned using the mollified point normals. The steps of our methods are detailed below.

3.1. Robust outlier detection and removal

The detection and removal of 3D outliers is a classical problem in computer vision and there is correspondingly a very large body of literature. Many of the popular methods rely on either a robust version of PCA *e.g.* [2] or RANSAC [4]. Recently [13] proposed two methods for outlier detection in point cloud data while [16] proposed a distance based outlier detection method. While RPCA and sampling based methods ([4], [13]) are computationally expensive, distance based methods are not adequately robust. Moreover, we have found that [13] fails in detecting outliers in the presence of additive Gaussian noise. Our proposed method is based on a simple geometric intuition which we develop using the following steps:



(a) Normals-based (b) Distance-based

Figure 1: Outlier detection

Neighbourhood assignment: We use a k-d tree based nearest neighbour approach for assigning the neighbours for each point. Specifically, consider a point cloud $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{N}$ where $\mathbf{v}_i$ is the position of the i^{th} point in it and N is the total number of points. We define the s-neighbourhood function $\mathcal{N}(i)$ for a point $\mathbf{v}_i$ in $\mathbf{V}$ as $\mathcal{N}(i) = \{\mathbf{v}_j \in \mathbf{V} | \|\mathbf{v}_j - \mathbf{v}_i\| \leq \|\mathbf{v}_k - \mathbf{v}_i\|, \forall k \notin \mathcal{N}(i) \text{ and } |\mathcal{N}(i)| = s\}$. Although an optimal value of s can be obtained theoretically using [11], we use a fixed value of $s = 128$ in our experiments.

Initial point normal estimation: Since we need an initial estimate of the point normals $\mathbf{n}_i$, we use weighted PCA for this purpose. Specifically we compute the normal $\mathbf{n}_i$ for a point $\mathbf{v}_i$ in the point cloud $\mathbf{V}$ as follows:

$$\mathbf{n}_i = \underset{\mathbf{n},\mathbf{n}^T\mathbf{n}=1}{\operatorname{argmin}} \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{n}^T \left((\mathbf{v}_j - \mu_i) (\mathbf{v}_j - \mu_i)^T \right) \mathbf{n} \tag{1}$$

where μ_i is the co-ordinate-wise median of $\{\mathbf{v}_j\}_{j \in \mathcal{N}(i)}$, w_{ij} is the weight given to a neighbouring point $\mathbf{v}_j$. It is known that in the presence of Gaussian distributed points in space, $w_{ij} = 1$ is the optimum weighting. However, in the presence of outliers this is not the case. In the presence of outliers, we choose a weighting of $w_{ij} = \|\mathbf{v}_j - \mathbf{v}_i\|_2^{-\alpha}$ where α is a small positive constant. In our experiments, we set $\alpha = 1.0$. The proper orientation of the normals can be fixed from the corresponding sensor location information.

Outlier detection: In our approach, we use two criteria for detecting outliers. Firstly, we use a novel technique based on the initial smooth estimate of the point normals. Secondly, we classify a point $\mathbf{v}_i$ as an outlier in a point cloud $\mathbf{V}$ if that point is far enough and isolated from the rest of the point cloud.

In the first step of **normal-based outlier detection and removal**, we develop a measure for classifying points as inliers or outliers. In Figure 1a we consider a set of points, which is sampled from a smooth surface (possibly with noise) with two outliers. We now focus on the two points

$\mathbf{v}_i$ and $\mathbf{v}_k$ where $\mathbf{v}_i$ is an outlier and $\mathbf{v}_k$ is a point sampled from the smooth surface that is in the neighbourhood of $\mathbf{v}_i$. Clearly, since there are only two outliers with many good samples in their neighbourhood, the initial normal $\mathbf{n}_i$ estimated at $\mathbf{v}_i$ will be almost the same as the estimated normal $\mathbf{n}_k$ at $\mathbf{v}_k$ *i.e.* $\mathbf{n}_i^T \mathbf{n}_k \simeq 1$. Now if we decompose the vector $\mathbf{v}_{ik} = \mathbf{v}_i - \mathbf{v}_k$ into two orthogonal components $\mathbf{v}_{ik}^{\parallel}$ and $\mathbf{v}_{ik}^{\perp}$ as in Figure 1a where $\mathbf{v}_{ik}^{\parallel} = \left(\mathbf{v}_{ik}^T \mathbf{n}_k\right) \mathbf{n}_k$ is the component along the normal $\mathbf{n}_k$ at point $\mathbf{v}_k$ and $\mathbf{v}_{ik}^{\perp} = \mathbf{v}_{ik} - \left(\mathbf{v}_{ik}^T \mathbf{n}_k\right) \mathbf{n}_k$ is the component orthogonal to it, the ratio $\left\|\mathbf{v}_{ik}^{\parallel}\right\| / \|\mathbf{v}_{ik}^{\perp}\|$ will be much larger than 1 *i.e.* $\left\|\mathbf{v}_{ik}^{\parallel}\right\| / \|\mathbf{v}_{ik}^{\perp}\| \gg 1$. We define the dissimilarity $DS\left(\mathbf{v}_k, \mathbf{v}_i\right)$ of $\mathbf{v}_i$ from $\mathbf{v}_k$ as

$$DS\left(\mathbf{v}_k, \mathbf{v}_i\right) = \left(\mathbf{n}_k^T \mathbf{n}_i\right) \frac{\left\|\mathbf{v}_{ik}^{\parallel}\right\|}{\left\|\mathbf{v}_{ik}^{\perp}\right\| + \epsilon} \tag{2}$$

where ϵ is a small positive constant. The dissimilarity $DS\left(\mathbf{v}_k, \mathbf{v}_i\right)$ measures the amount of isolation of the point $\mathbf{v}_i$ from the surface defined around the point $\mathbf{v}_k$. We note in passing that the dissimilarity $DS\left(\cdot, \cdot\right)$ is asymmetric in nature. For each point $\mathbf{v}_i$, we accumulate all such $DS\left(\mathbf{v}_k, \mathbf{v}_i\right)$ and compute the effective dissimilarity $EDS\left(\mathbf{v}_i\right)$ as

$$EDS\left(\mathbf{v}_i\right) = \frac{\sum_{k \in \mathcal{N}(i)} DS\left(\mathbf{v}_k, \mathbf{v}_i\right)}{\left|\mathcal{N}\left(i\right)\right|}. \tag{3}$$

If $EDS\left(\mathbf{v}_i\right)$ is above a threshold η_n, it is classified as an outlier. It is important to note that had there been another surface from which $\mathbf{v}_i$ was sampled and was not actually an outlier, there would be a sufficient number of neighbouring points for $\mathbf{v}_i$ from that second surface in the nearest neighbour set of $\mathbf{v}_i$ and the $EDS\left(\mathbf{v}_i\right)$ would remain low. This would result in point $\mathbf{v}_i$ being correctly classified as an inlier.

Although the above normal-based technique removes most of the outliers in the point cloud $\mathbf{V}$, it may still contain sparse clusters that are not part of the true surface point cloud. To address this issue we use a **distance-based outlier detection and removal** step that can detect sparse sets of outliers. As shown in Figure 1b, for any point $\mathbf{v}_i$, we compute the median of the ℓ_2 distances $d_{med}\left(\mathbf{v}_i\right)$ of neighbouring points around $\mathbf{v}_i$ from $\mathbf{v}_i$ as

$$d_{med}\left(\mathbf{v}_i\right) = \text{MEDIAN}\left(\left\{\|\mathbf{v}_{ik}\|_2\right\}_{k \in \mathcal{N}(i)}\right). \tag{4}$$

We then use an absolute threshold η_d above which a point is expected to be an outlier. We test the effectiveness of our method on synthetic datasets, namely a cube, a sphere and the well-known Stanford bunny. We add Gaussian noise of standard deviation 100% of the average edge length of the respective original meshes. We then corrupt a varying percentage of the points into outliers (20-40$\%$) drawn from a larger Gaussian distribution of varying standard deviation (10-20$\%$) of the original point cloud dimensions. We compare our outlier detection method with two methods that are implemented in the Point Cloud Library (PCL) namely statistical outlier removal (SOR) [15, 16] and radius outlier removal (ROR) [15]. We also compare with MCMD_Z [13][1]. We tune all other parameters to their optimal settings for each method. We measure the average fraction of correct classification of the outputs from all the methods over 5 noise and outlier realisations. Table 1 shows their comparative performances. Figures 2, 3 and 4 shows the visual comparison of outputs from the compared methods for outlier density of 40% with standard deviation of 20% of the respective point cloud dimensions. It is clear that our method performs much better than the others, especially when the fraction of outliers increases. Also, it is important to note that the MCMD_Z method performs poorly in detecting outliers in presence of Gaussian noise.

Input Model	Outliers		Accuracy			
	D (%)	S (%)	SOR	ROR	MCMD_Z	**Ours**
Cube	20	10	**0.939**	0.927	0.927	**0.939**
NP: 49154	40	20	0.880	0.905	0.703	**0.926**
Sphere	20	10	0.949	0.921	0.937	**0.952**
NP:40962	40	20	0.902	0.934	0.636	**0.951**
Bunny	20	10	0.941	0.928	0.890	**0.959**
NP:40245	40	20	0.949	0.933	0.670	**0.969**

Table 1: Comparative performances of SOR [15, 16], ROR [15], MCMD_Z [13] and our method. D - density, S - standard deviation of outlier distribution, NP - number of points. See text for details.

The performance of our outlier detection method is studied against varying amount of outliers on the same cube, sphere and bunny data. Figure 5 shows the accuracy of detecting outliers which are drawn from a larger Gaussian distribution of standard deviation (10%) of the point cloud dimensions with varying fraction of the points corrupted as outliers. The inliers are corrupted with Gaussian noise of standard deviation of average edge length of the original meshes. In the presence of noise in the inliers, the accuracy of our method remains above 0.85 even when the fraction of outliers is at 0.5.

3.2. Bilateral normal mollification

Once outliers are removed from the input noisy point cloud, it is ready for denoising. Formally, our denoising problem can be stated as follows. An initial outlier-free

[1] We use MCMD_Z only and not MCMD_MD as the former is reported to perform better.

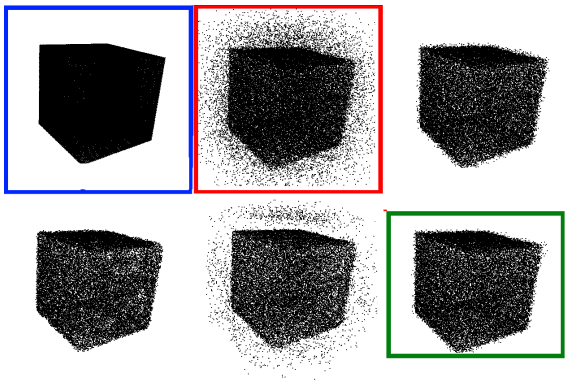


Figure 2: Visual comparison of outputs of SOR [15, 16], ROR [15], MCMD_Z [13] and our method on cube data (49154 points) for outlier density of 40% with standard deviation of 20% of the point cloud dimensions in presence of Gaussian noise of std. dev. of avg. edge length of original meshes. First row shows the clean point cloud (blue boxed), noisy point cloud (red boxed) and output from SOR respectively. Second row shows the outputs from ROR, MCMD_Z and our method (green boxed) respectively.

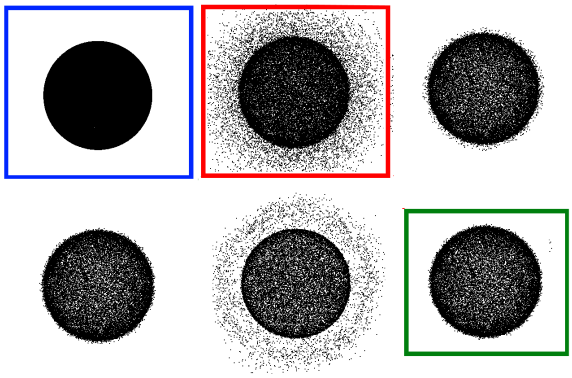


Figure 3: Visual comparison of outputs of SOR [15, 16], ROR [15], MCMD_Z [13] and our method on sphere data (40962 points) for outlier density of 40% with standard deviation of 20% of the point cloud dimensions in presence of Gaussian noise of std. dev. of avg. edge length of original mesh. First row shows the clean point cloud (blue boxed), noisy point cloud (red boxed) and output from SOR respectively. Second row shows the outputs from ROR, MCMD_Z and our method (green boxed) respectively.

noisy point cloud $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{N}$ is given where $\mathbf{v}_i$ is the i^{th} noisy point position and N is the total number of points. We would like to estimate the unknown true point cloud as $\widehat{\mathbf{V}} = \{\hat{\mathbf{v}}_i\}_{i=1}^{N}$. Like RIMLS [14], our denoising method depends on a clean set of the point normals $\{\hat{\mathbf{n}}_i\}_{i=1}^{N}$. Since the normals estimated from the input point cloud are noisy, we first mollify them in an iterative manner similar to RIMLS.

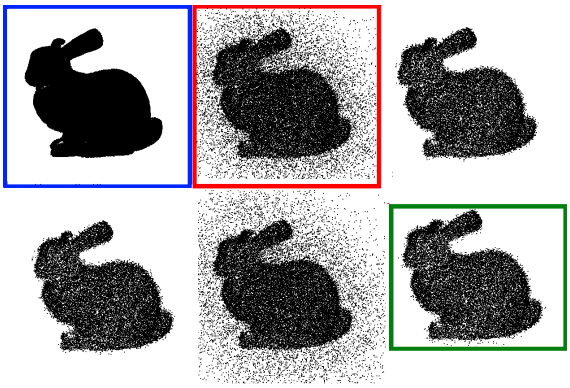


Figure 4: Visual comparison of outputs of SOR [15, 16], ROR [15], MCMD_Z [13] and our method on bunny data (40245 points) for outlier density of 40% with standard deviation of 20% of the point cloud dimensions in presence of Gaussian noise of std. dev. of avg. edge length of original mesh. First row shows the clean point cloud (blue boxed), noisy point cloud (red boxed) and output from SOR respectively. Second row shows the outputs from ROR, MCMD_Z and our method (green boxed) respectively.

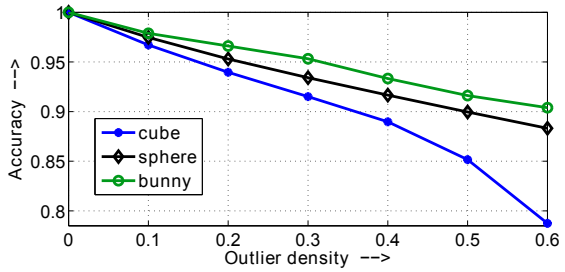


Figure 5: Performance of our method with varying fraction of points as outliers.

Specifically, in each iteration we use a bilaterally weighted mollification of the normals as

$$\hat{\mathbf{n}}_i \leftarrow \text{normalise}\left(\sum_{j \in \mathcal{N}(i) \cup i} \phi_{ij} \hat{\mathbf{n}}_j\right) \tag{5}$$

where

$$\phi_{ij} = e^{-\left(\frac{\|\hat{\mathbf{n}}_j - \hat{\mathbf{n}}_i\|^2}{\sigma_r^2} + \frac{\|\mathbf{v}_j - \mathbf{v}_i\|^2}{\sigma_s^2}\right)}, \tag{6}$$

normalise$(\cdot)$ scales a vector to unit length and σ_r and σ_s are the normal and spatial scale parameters respectively. However in general, when sensor location information is not available, the initial normals $\mathbf{n}_i$ may be flipped, [14] suggests to initialise $\hat{\mathbf{n}}_i$ with the weighted mean of the neighbouring normals without considering the current normals

themselves as

$$\hat{\mathbf{n}}_i \leftarrow \text{normalise}\left(\sum_{j\in\mathcal{N}(i)} \exp\left(-\frac{\|\mathbf{v}_j-\mathbf{v}_i\|^2}{\sigma_s^2}\right)\mathbf{n}_j\right). \tag{7}$$

Although this initialisation works well in low noise condition, under considerable noise it inevitably leads to smooth edges. Instead we do the following. For each normal $\mathbf{n}_i$, we aggregate the corresponding weights in Equation 5 as

$$\alpha_i = \sum_{j\in\mathcal{N}(i)} e^{-\left(\frac{\|\mathbf{n}_j-\mathbf{n}_i\|^2}{\sigma_r^2}\right)} \tag{8}$$

$$\beta_i = \sum_{j\in\mathcal{N}(i)} e^{-\left(\frac{\|\mathbf{v}_j-\mathbf{v}_i\|^2}{\sigma_s^2}\right)} \tag{9}$$

We then flip the normal $\hat{\mathbf{n}}_i$ if $\alpha_i < \kappa\cdot\beta_i$ for a small positive κ depending on the amount of noise.

3.3. Point set repositioning

Once we obtain the mollified normals $\{\hat{\mathbf{n}}_i\}_{i=1}^N$, we now fit the points to them. To do so, we use a weighted point update. Unlike methods that follows MLS, we use a simpler point set fitting scheme. Our scheme is robust enough to preserve fine features like edges and corners. Moreover, our scheme actually enriches these fine features. For estimating the new positions $\{\hat{\mathbf{v}}_i\}_{i=1}^N$ of the 3D points, we solve the following minimisation

$$\min_{\{\tilde{\mathbf{v}}_i\}_{i=1}^N} \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} \gamma_{ij}\left\|\hat{\mathbf{n}}_i{}^T(\tilde{\mathbf{v}}_i-\tilde{\mathbf{v}}_j)\right\|_2^2 + \lambda\sum_{i=1}^{N}\|\tilde{\mathbf{v}}_i-\mathbf{v}_i\|_2^2 \tag{10}$$

where

$$\gamma_{ij} = \frac{\tau_{ij}}{\sum_{j\in\mathcal{N}(i)}\tau_{ij}},\ \tau_{ij} = \exp\left(-\frac{\|\tilde{\mathbf{v}}_j-\tilde{\mathbf{v}}_i\|^2}{\sigma_s^2}\right) \tag{11}$$

are the weights used to adaptively set the influence of the neighbours, $\hat{\mathbf{n}}_i$ are the mollified normals, $\mathbf{v}_i$ are the noisy point positions and λ is a small positive stabilising parameter to ensure a stable solution. Since the weights γ_{ij} depend on the optimising variables $\tilde{\mathbf{v}}_i$, we solve this minimisation iteratively using gradient-descent method and it usually converges within 5-30 iterations.

We note here that if a point lying on a plane moves a little along any direction but remaining on the same plane, then the surface does not change. Since the weights γ_{ij} do not depend on the normals $\hat{\mathbf{n}}_i$, our minimisation allows the points on the either sides of fine structures like edges and corners to move towards them. This automatically defines and enhances these sharp structures. In Figure 6, we show the difference of allowing such a scheme on a noisy cube point cloud in comparison to the RIMLS [14] method that does not allow so. In our method, the edges are prominently defined. This is in contrast to [18] where their method requires an additional step to recover the fine structures.

Figure 6: Automatic recovery of fine structures in our point set repositioning scheme when applied on a noisy cube data as compared to the output from RIMLS [14]. Left to right: the noisy point cloud, outputs from RIMLS and our scheme respectively. Note the prominent straight edges recovered by our method.

Point set upsampling: In the previous step, the points that are not on the sharp features like edges and corners but are located nearby are moved towards them. Hence, those regions in the final denoised result may not contain dense enough points. For rendering purposes, a sufficiently dense point set is required. Hence, similar to the recent methods like [18], one can optionally upsample the denoised point set using Edge-Aware Resampling method [8] to obtain the final ready-to-be-rendered denoised point cloud.

4. Results

In this Section, we present the evaluation of performance of our denoising method as compared to relevant existing methods in the literature both on synthetic point cloud data and real point cloud data obtained from multi-view stereo and depth scanner.

4.1. Synthetic data

We evaluate the performance of our method as compared to similar relevant methods available in the literature, namely [14] and [18]. We run synthetic experiments on a dataset comprising the cube (49154 points), the sphere (40962 points) and the bunny model (40245 points) by adding Gaussian noise of standard deviation 100% of the average edge-length of the corresponding original meshes. We note here that in this particular experiment we do not add any outliers. Parameters of all the methods are tuned to their optimal settings. We measure the mean cloud-to-mesh ℓ_2 distances from the corresponding denoised outputs to the original clean meshes for each method. To do so, we specifically ensured that the denoised outputs are sufficiently dense enough. Table 2 shows the accuracies of the results from different methods on the synthetic datasets.

Figure 7 shows the visual comparisons of the results obtained from the compared methods. For the cube, it can be observed that the recovered edges from our method are very prominent. Also, for the sphere, it can be observed that due to nature of the ℓ_0 norm, ℓ_0-method performs poorly in retaining the spherical shape. From Figure 8, it can be observed that the ℓ_0-method removes the fine details of the surface of the bunny model and encourages a piecewise flat surface. Moreover, from Table 2, it can be seen that the mean cloud-to-mesh ℓ_2 distance of the denoised output from RIMLS is higher than that of our method. This is due to an increase in volume in the output of RIMLS method with respect to the true model.

Input Model	Noise Std. Dev.	Mean cloud-to-mesh ℓ_2 distance		
	(AEL)	RIMLS	ℓ_0	Ours
Cube NP:49154	100%	0.0016	0.0041	**0.0005**
Sphere NP:40962	100%	**0.0046**	0.0156	0.0049
Bunny NP:40245	100%	0.0023	0.0054	**0.0021**

Table 2: Comparison of denoising performance of our method with RIMLS [14] and ℓ_0-method [18]. AEL - avg. edge length of original mesh, NP - number of points.

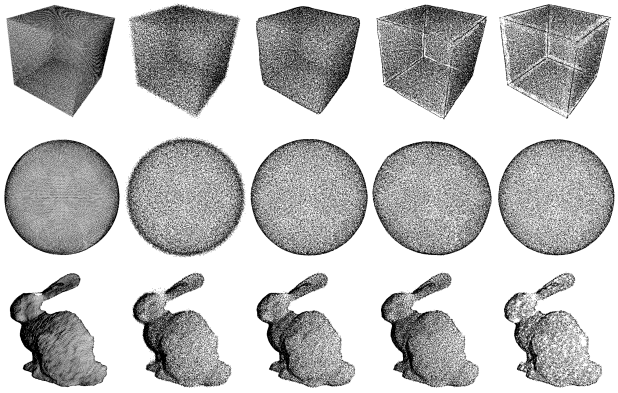


Figure 7: Visual comparison of denoised outputs of RIMLS [14], ℓ_0 method [18] and our method on a noisy cube (49154 points), a noisy sphere (40962 points) and a noisy bunny (40245 points) added with Gaussian noise of std. dev. of avg. edge length of original mesh. The rows correspond to the cube, the sphere and the bunny data respectively. The columns correspond to the clean point cloud, noisy point cloud, outputs of RIMLS, ℓ_0-method and our method respectively.

We also perform a synthetic experiment to evaluate the effectiveness of our outlier detection and removal method

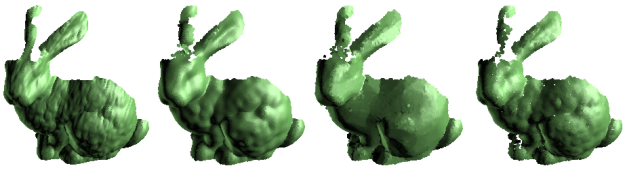


Figure 8: Visual comparison of rendered original point cloud and denoised outputs of RIMLS [14], ℓ_0-method [18] and our method on the noisy bunny (40245 points) as in Figure 7. Note the piecewise flat surface encouraged by the ℓ_0-method.

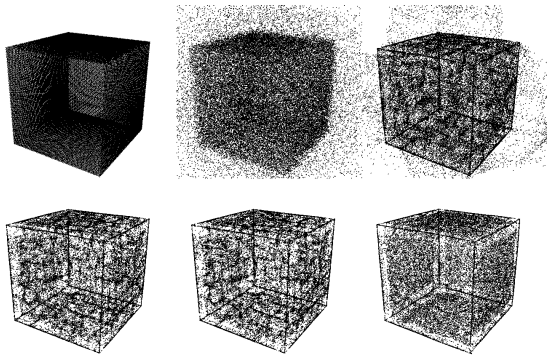


Figure 9: Visual comparison of denoised outputs of our denoising method using different outlier removal methods on an outlier-contaminated noisy cube (49154 points). The first row shows the original clean point cloud, noisy cube point cloud with outliers and denoised output from our denoising method without any outlier removal respectively. The second row shows the final denoised outputs from our denoising method using SOR [15, 16], ROR [15] and our outlier removal method respectively.

Mean cloud-to-mesh ℓ_2 distance			
WOR	SOR	ROR	Ours
0.0548	0.0017	0.0017	**0.0014**

Table 3: Performance comparison of our denoising method using SOR [15, 16], ROR [15] and our outlier detection and removal method on a noisy point cloud of a cube. WOR - without outlier removal. See text for details.

(described in Section 3.1) in our denoising method. Specifically we use the cube data and corrupt 40% of the points as outliers drawn from a Gaussian distribution of a large standard deviation of 30% of the point cloud dimensions and add Gaussian noise of standard deviation 200% of the average edge length of the original mesh in the inliers. We compare the effects of our outlier removal method with SOR [15, 16] and ROR [15] on the performance of our denoising method. Parameters of all the methods are tuned to their op-

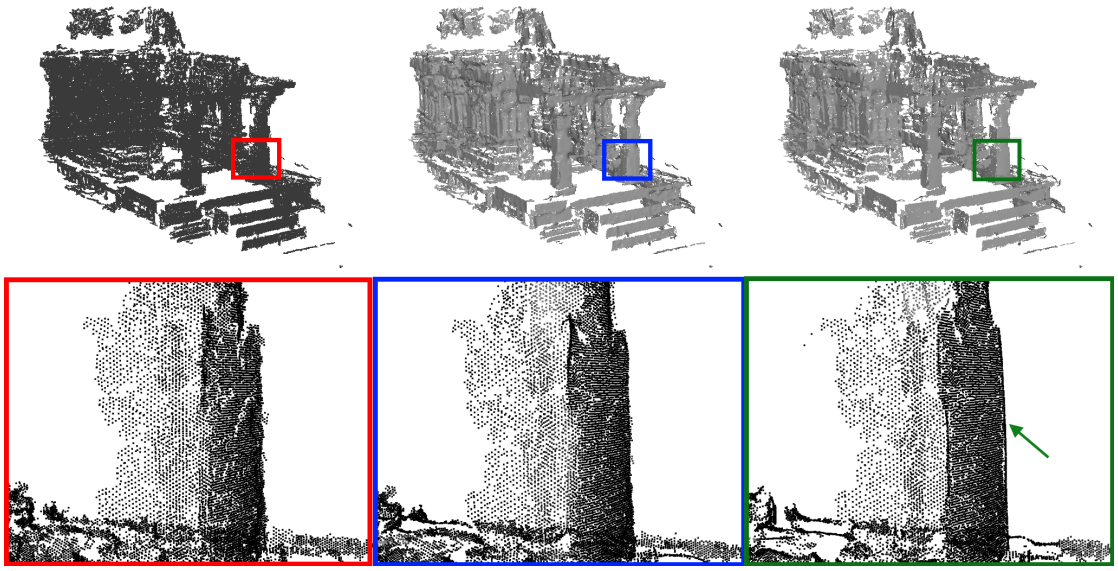


Figure 10: Comparative results on a point cloud (919851 points) of a heritage monument in the Vitthala temple complex at Hampi, India, obtained from multi-view stereo. The first row shows the noisy point cloud, the results from RIMLS [14] and our method respectively. The second row shows the zoomed-in views of the highlighted regions from the noisy point cloud and the two methods respectively. Clear edges near the green arrow mark is visible in the output from our method.

timal settings. Figure 9 shows visual comparison of the denoised outputs. The first row shows the original clean point cloud, noisy cube point cloud with outliers and denoised output from our denoising method without any outlier removal. The second row shows the final denoised outputs from our denoising method using each of the above three outlier removal methods. Table 3 shows the mean cloud-to-mesh ℓ_2 distances for each case. The denoising performance is best with our outlier removal method.

4.2. Real data

We run our method on real 3D point cloud data obtained from multi-view stereo and depth scanner and compare the results with RIMLS [14]. Figure 10 shows the results of our method on a point cloud (919851 points) of a heritage monument in the Vitthala temple complex at Hampi, India, obtained from multi-view stereo, as compared to RIMLS. The first row shows the noisy point cloud, the results from RIMLS and our method respectively. The second row shows the zoomed-in views of the highlighted regions from the noisy point cloud and the two methods respectively. As observed in the zoomed-in views, unlike RIMLS, our method recovers a clear edge near the green arrow mark. Similarly, Figure 11 shows comparative results on a point cloud (1030980 points) of another heritage monument in the Vitthala temple complex at Hampi, India, obtained from multi-view stereo where also our method recovers a well defined edge between the two walls marked by a green arrow. Figure 12 shows comparative results on a point cloud (153288 points) of a stool obtained using a depth scanner. The zoomed-in view of the highlighted region of our method clearly shows the edge of the leg near the green arrow mark.

Discussion

In our outlier detection and removal step, there are two parameters η_n and η_d that need to be set. η_n depends on the amount of outliers present in the data and can be typically set to a value in the range 0-1. η_d depends on the sampling density of the true signal in the data. It should be noted that our method will not detect clustered outliers. In the point cloud denoising step, typical values of the parameters resemble those of RIMLS. The stabilising parameter λ can be set to a value in the range $(0.0\text{-}1.0]$. One limitation of our denoising method is that in pathological cases when the input point cloud is contaminated with noise but not sufficiently sampled, our method induces volume expansion.

5. Conclusion

We have presented a robust 3D point cloud denoising method consisting of a robust outlier detection and removal followed by bilateral mollification of the noisy point normals and finally a repositioning of the 3D points in a manner so as to preserve the fine scale features. We have shown that our method automatically recovers well-defined edges and corners. We have demonstrated the efficacy of our method through multiple examples and experiments.

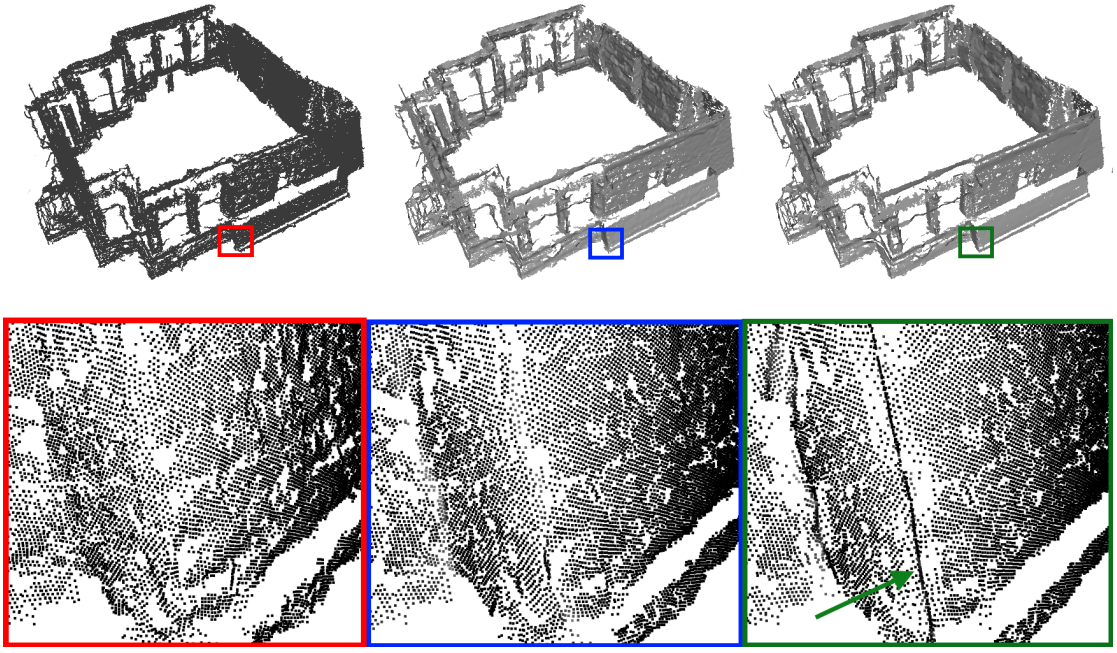


Figure 11: Comparative results on a point cloud (1030980 points) of another heritage monument in the Vitthala temple complex at Hampi, India, obtained from multi-view stereo. The first row shows the noisy point cloud, the results from RIMLS [14] and our method respectively. The second row shows the zoomed-in views of highlighted regions from the noisy point cloud and the two methods respectively. Clear edge near the green arrow mark is visible in the output from our method.

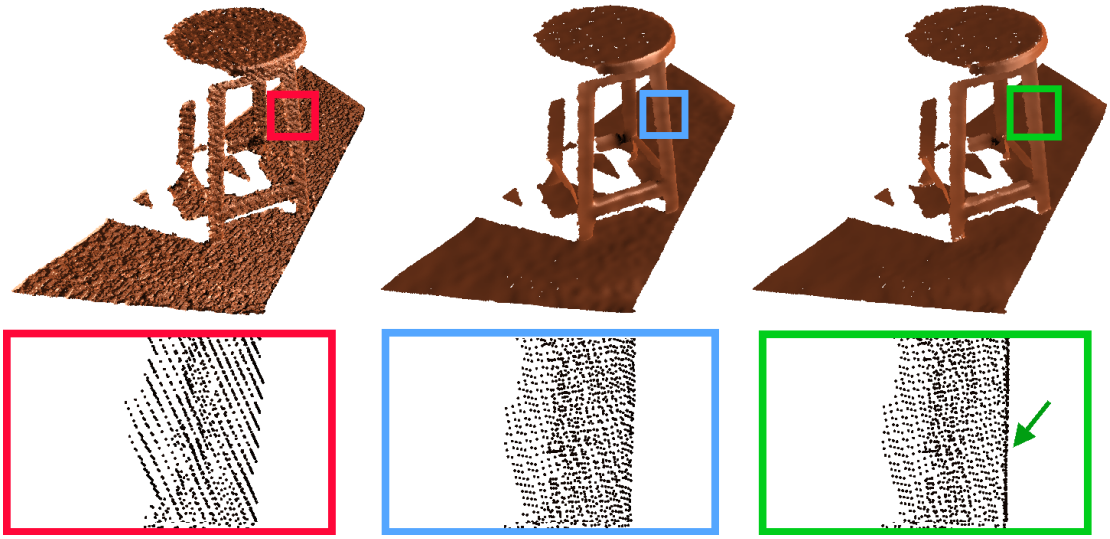


Figure 12: Comparative results on a point cloud (153288 points) of a stool obtained using a depth scanner. The first row shows the noisy point cloud, the results from RIMLS [14] and our method respectively. The second row shows the zoomed-in views of the highlighted regions of the noisy point cloud, the results from RIMLS [14] and our method respectively. Clear edge near the green arrow mark is visible in the output from our method.

Acknowledgement

This work is supported in part by an extramural research grant by the Science and Engineering Research Board, DST, Government of India. Mohammadul Haque is supported by a TCS Research Scholarship.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. Visualization and Computer Graphics, IEEE Transactions on, 9(1):3–15, 2001.

[2] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? Journal of the ACM (JACM), 58(3):11, 2011.

[3] J.-E. Deschaud and F. Goulette. Point cloud non local denoising using local surface descriptor similarity. In PCV (Photogrammetric Computer Vision), Paris, France, Sept. 2010.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[5] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. In ACM transactions on graphics (TOG), volume 24, pages 544–552. ACM, 2005.

[6] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. IEEE Trans. on Pattern Analysis and Machine Intelligence, 32(8):1362–1376, 2010.

[7] G. Guennebaud and M. Gross. Algebraic point set surfaces. In ACM Transactions on Graphics (TOG), volume 26, page 23. ACM, 2007.

[8] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang. Edge-aware point set resampling. ACM Transactions on Graphics, 32:9:1–9:12, 2013.

[9] D. Levin. The approximation power of moving least-squares. Mathematics of Computation of the American Mathematical Society, 67(224):1517–1531, 1998.

[10] C. Liu, D. Yuan, and H. Zhao. 3d point cloud denoising and normal estimation for 3d surface reconstruction. In 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 820–825, Dec 2015.

[11] N. J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In Proceedings of the nineteenth annual symposium on Computational geometry, pages 322–328. ACM, 2003.

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on. IEEE, October 2011.

[13] A. Nurunnabi, G. West, and D. Belton. Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data. Pattern Recognition, 48(4):1404–1419, 2015.

[14] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. In Computer Graphics Forum, volume 28, pages 493–501. Wiley Online Library, 2009.

[15] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9-13 2011.

[16] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D point cloud based object maps for household environments. Robotics and Autonomous Systems, 56(11):927–941, 2008.

[17] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In ACM Siggraph 2005 Courses, page 204. ACM, 2005.

[18] Y. Sun, S. Schaefer, and W. Wang. Denoising point sets via l0 minimization. Computer Aided Geometric Design, 35:2–15, 2015.

[19] S. Xiong, J. Zhang, J. Zheng, J. Cai, and L. Liu. Robust surface reconstruction via dictionary learning. ACM Transactions on Graphics (TOG), 33(6):201, 2014.

[20] F. Zaman, Y. P. Wong, and B. Y. Ng. Density-based denoising of point cloud. arXiv preprint arXiv:1602.05312, 2016.

[21] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or, and B. Chen. Non-local scan consolidation for 3d urban scenes. 2010.