A review of algorithms for filtering the 3D point cloud

Article in Signal Processing Image Communication - May 2017
DOI: 10.1016/j.image.2017.05.009

CITATIONS
15

READS
1,945

6 authors, including:

Xian-Feng Han
Tianjin University
7 PUBLICATIONS 24 CITATIONS

Jesse Sheng Jin
Tianjin University
62 PUBLICATIONS 296 CITATIONS

Some of the authors of this publication are also working on these related projects:

- point cloud processing View project
- Humanbody Reconstruction View project

All content following this page was uploaded by Xian-Feng Han on 26 October 2017.
The user has requested enhancement of the downloaded file.
A review of algorithms for filtering the 3D point cloud

Xian-Feng Han *, Jesse S. Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, Liping Xiao

School of Computer Software, Tianjin University, 30072, Tianjin, China
National Key Laboratory of Science and Technology on Aerospace Intelligent Control, Beijing 100085, China

1. Introduction

The 3D point cloud [1–3], a new primitive representation for objects, has become increasingly prevalent in many research fields [2], such as object recognition [4] and reconstruction [5,6], due to its simplicity, flexibility and powerful representation capability. In contrast to triangle meshes, the point cloud does not require to store or maintain the polygonal-mesh connectivity [7] or topological consistency [8]. Processing and manipulating point cloud therefore can demonstrate better performance and lower overhead. These prominent advantages make the research on processing point cloud a hot topic.

The rapid development of low-cost sensors, such as Kinect [9–11] and time of flight cameras [5,12], makes it easy to obtain point cloud for growing communities. The point cloud acquired with these sensors, however, inevitably suffers from noise contamination and contains outliers [13,14] due to the limitations of sensors [5], the inherent noise of the acquisition device [15], the lighting or reflective nature of the surface or artifact in the scene [16]. Therefore, it is necessary to perform filtering operations on raw point clouds to obtain accurate point clouds that are suitable for further processing.

In recent years, although a large number of methods contributing to 3D filtering have been proposed, most of these are devised for meshes and only a few approaches directly operate on point cloud. In addition, there is no survey paper giving an insightful analysis of these filtering methods for point cloud.

1.1. Statistical-based filtering techniques

In the context of filtering point cloud, many techniques utilize the adaptation of the statistical conceptions, which are suitable for the nature of the point cloud.
Schall et al. [18] filtered point cloud using a kernel based clustering approach. They first accumulated the local likelihoods \( L_i \) computed on every point \( p_i \) to define the likelihood function \( L \) modeling the probability for the noisy point cloud. Next, they moved the points to positions of high probability utilizing an iterative scheme motivated by mean shift technique to smooth the point cloud. This method achieves effectiveness in filtering and robustness in outlier detection. However, sharp features are not treated with emphasis in their method.

Narvaez et al. [15] proposed a new weighted variant of the principal component analysis method for denoising point cloud, which used weighting factors assignment by inversely proportional repartition of the sum of distance to the mean. Then, the factors and the weighted mean are used to estimate a weighted covariance matrix. By realizing an eigen-analysis of the matrix, a fitting plane, expanded by the eigenvectors corresponding to the largest eigen-values, and a normal vector to this plane oriented in the direction of eigen-vector corresponding to the smallest eigen-value are obtained at each point. The variations make the algorithm robust to the noise and outliers. In addition, the operator \( p' = p + \alpha \) [19] is applied to shift the mean along the normal direction to preserve shape features.

Jenke et al. [20] first employed Bayesian statistics for denoising point cloud. They found a measurement model \( P(D|S) \), which specified the probability distribution of estimated point cloud \( S \) agreeing with measured data \( D \). Then, they defined three prior probabilities, such as density priors, smoothness priors and priors for sharp features, to form \( P(S) = \frac{1}{Z} P_{\text{density}}(S) P_{\text{smooth}}(S) P_{\text{discrete}}(S) \). \( Z \) was a normalization constant. Finally, they maximized a posteriori \( P(S|D) \) to remove noise while preserving features.

\[
S_{MAP} = \arg \max_S P(S|D) = \arg \max_S P(D|S) P(S) \tag{1}
\]

Kalogerakis et al. [21] provided a robust statistical framework to filter point clouds. In their framework, an Iteratively Least Squares (IRLS) approach estimates curvature tensor and assigns weights to samples at each iteration to refine each neighborhood around every point. The computed curvatures and the final statistical weights are utilized to correct normal. The robustly estimated curvatures and normal can drive the outlier rejection and denoise point cloud in a feature-preserving manner based on a global energy minimization process.

Avron et al. [22] introduced \( L_1 \)-sparsity paradigm to denoise the point cloud. Firstly, a re-weighted \( L_1 \) minimization process is used to restore point orientations. Then, point position is reconstructed by assuming a local planarity criterion so as to preserve shape features. Although, this method can achieve reasonable results, points on an edge are sometimes not faithfully recovered [23]. Meanwhile, since \( L_0 \) is a sparser solution than \( L_1 \), Sun et al. provided an \( L_0 \) minimization method, which is directly used to denoise point cloud by applying a similar \( L_0 \) optimization procedure to estimate normals followed by repositioning points along the normal directions in order to better maintain the sharp features (see Fig. 1).

Orts-Escolano et al. [24] first used a 3D filtering and downsampling technique based on Growing Neural Gas (GNG) [25–27] network. This is a growth process to produce a GNG network to represent a raw point cloud using a set of 3D neurons and interconnection among them. This method can preserve the topology of point cloud and deal with outliers in point cloud. Therefore, filtered point cloud using GNG can improve keypoints detection performance [28] and yield better recognition results [4,29]. This method produced a GNG network mapping to the point cloud (shown in Fig. 2).

### 2.2. Neighborhood-based filtering techniques

Neighborhood-based filtering techniques determine the filtered position of a point using similarity measures between a point and its neighborhood which has a strong influence on the efficiency and effectiveness of the filtering approach [17]. As described in the following methods, the similarity can be defined by positions of points, normals or regions.

The bilateral filter, originally introduced by Tomasi and Manduchi [30], is an edge-preserving smoothing filter, which is extended to 3D mesh denoising [31–34]. However, these methods require a mesh generation process, which itself suffers from noise [35]. In order to tackle this problem, the bilateral filter is applied directly on the point cloud [29,36–38] based on point positions and intensity. The \( w_i \) and \( w_j \) are the spatial and range weight, respectively,

\[
w_i = \exp \left( -\frac{(i - x)^2 + (j - y)^2}{2\sigma_x^2} \right) \tag{2}
\]

\[
w_j = \exp \left( -\frac{(I(i,j) - I(x,y))^2}{2\sigma_z^2} \right) \tag{3}
\]

where \((i,j)\) is the neighborhood of \((x,y)\), \(I(i,j)\) presents the intensity at \((x,y)\), \(\sigma_x\) and \(\sigma_z\) are the standards of Gaussian functions.

In order to reduce time complexity, Xu et al. [39] replaced the weight of gray domain in the bilateral filter with a binary function (4) to achieve a better performance. However, this kind of filters deals with the point cloud containing intensity components. As a consequence, normal [35,40–43], being as one of the important attributes of point cloud, is considered in the process of bilateral filter of which the weight
is defined as a function of spatial location and normal information of points, shown in Eq. (5).

\[
w_i = \begin{cases} 1 & |I(x, y) - I(i, j)| \leq 3, I(x, y) \neq 0 \\ 0 & |I(x, y) - I(i, j)| > 3, I(x, y) \neq 0 \end{cases}
\]

(Wang et al. [46]) presented an effective algorithms consisting of two steps: outliers filtering and noise smoothing. They designed a connectivity-based approach based on the properties of the relative deviation of the local neighborhood and the average local neighborhood to remove sparse outliers. After the process, they used a clustering-based scheme to further eliminate the small cluster outliers. Furthermore, they performed normal filtering by iteratively updating weighted normal of point. Points are updated to match the filtered normal. Experimental results show that this method obtains satisfactory results.

2.3. Projection-based filtering approaches

Projection-based filtering approaches adjust the position of each point in a point cloud via different projection strategies to filter point cloud.

Motivated by the mean shift filtering for images, Hu et al. [44] formulated a 3D mean shift based anisotropic filter algorithm by taking the vertex normal, curvature as well as position into account. According to the similar modes of points produced via mean shift procedure, they designed a cluster scheme to provide an adaptive neighbor searching method to determine the neighborhoods of each point. Finally, they applied a trilateral filtering to the adaptive neighborhoods to denoise the point cloud whilst preserving features.

Wang et al. [46] introduced a parameterization-free projection operator, that was, Locally Optimal Projection (LOP) operator [51]. The rationale of this method is to iteratively project a subset of the input point cloud onto this point cloud to reduce noise and outliers. However, if the input point cloud is highly non-uniform, projection by LOP tends to become non-uniformity, which is undesirable in shape features preservation and normal estimation.

Huang et al. [49] incorporated locally adaptive density weights of each point into LOP, that was WLOP (see Fig. 4), to produce an evenly distributed points set. An important parameter which controls the amount of smoothness in LOP is $h$, the support size of the weight function $\Theta$. Too large value of $h$ causes the shrinkage of the point cloud while too small value results in little effect of denoising. To solve this problem, Ye et al. [53] projected only $x$ coordinates of the point-set, then $x$ and $y$ could be calculated through re-projection. This method not only obtains smoothness and the preservation of geometric structure, but also removes isolated outlier points.

Hube et al. [12] took color information and intra-patch similarity into account to propose a robust non-local means filter. This method first detects outliers in the input data and then performs a feature-preserving smoothing processing. Experimental results show that this denoising algorithm exhibited a superior performance.
Since LOP is an isotropic operator, Huang et al. [55] replaced \( \theta (||p_i - q_i||) \) in the term \( E_i \) with following equation

\[
\theta (n_i, p_i - q_i) = e^{-\left(\frac{(p_i - q_i)^T n_i}{\sigma^2}\right)^2} \tag{6}
\]

where \( n_i \) denoted normal at point \( p_i \), which was estimated based on an anisotropic neighborhood followed by being smoothed using bilateral normal smoothing. This anisotropic LOP could obtain better sharp feature preservation.

Since the above methods require high computational effort, Preiner et al. [56] designed an efficient variant of the locally optimal projection operator. They employed the Gaussian Mixture Model formed by regularizing the hierarchical EM algorithm to cluster points to represent the point cloud’s density. Then, they redefined and evaluated the attractive force to obtain the continuous LOP (CLOP). Finally, CLOP was applied to the Gaussians instead of points to achieve speedups.

There are many approaches built upon the work over Moving Least Squares by Levin [57-59]. Alexa handled noise problem by iteratively projecting the points onto the MLS surface [19,60] defined by two steps (see Fig. 5). The first step finds a local reference plane \( H \) by locally minimizing \( \sum_{p \in H} (x, n, p > -D) e^{-||r - q||^2/h^2} \), where \( q \) is the projection of \( r \) onto \( H \) and \( h \) is a global scale factor controlling the degree of smoothing. A local polynomial approximation with respect to the reference domain is then produced by using weighted averages of input normal \( n_i \). Alexa handled noise problem by projecting them onto the corresponding quadratic surfaces fitted using sampling consensus to compute the best tangent planes for all points. In the second step, \( \theta (n_i, p_i - q_i) \) was estimated based on an algebraic sphere fitting to improve stability where planar MLS fails. Fleishman et al. [65] introduced a robust moving least squares technique based on forward-search paradigm to deal with noise, outliers and sharp features. Their work classifies the neighborhood of the point into subsets of outlier-free smooth regions of the surface, and then the moving least squares projection mechanism is operated on points. But this method requires very dense point clouds and is time-consuming. Adamson and Alexa [66] adopted cell complexes to preserve the shape features by decomposing the object into cells of different dimensions, while this decomposition demanded effort by the users.

From the above approaches, it can be seen that the plane fitting operation is unstable in regions of high curvature if the sampling rate is below a threshold. Guennebaud and Gross [67] thus performed anisotropic least squares fitting to improve stability where planar MLS fails.

Fua et al. [68] fitted a local quadric patch to a neighborhood of every point. They then iteratively smoothed the raw point clouds by using these estimated surfaces. In order to deal with outliers, a metric was defined to measure whether or not two points belong to the same local surface. This process was curvature-preserving and could not smooth out relevant features.

Wang et al. [69] used the mean shift clustering and adaptive scale sampling consensus to compute the best tangent planes for all points so as to detect and remove outliers. Subsequently, they dichotomized the feature and non-feature points and denoised them, respectively, by projecting them onto the corresponding quadratic surfaces fitted using their neighborhoods, which were determined based on the normal-based region growing technique.

\[ a \]

\[ b \]

Fig. 6. Denoise point sets via MLS. (a) Input noisy model; (b) filtered result.

2.4. Signal processing based method

Signal processing methodology can also be extended to point cloud filtering. Based on Taubin’s method [70] that applied the Laplacian operators to filter the mesh, Linsen [71] developed a filtering operator procedure, in which local reference planes were defined by a weighted average position at \( x \). A new smoothing operator \( Q(r) = r + t, n \) computed by using an effective numerical optimization procedure to preserve salient features, where \( n \) and \( t \) were determined by the fitting of a plane \( H \) in the neighborhood of point \( r \). Fleishman et al. [65] introduced a robust moving least squares technique based on forward-search paradigm to deal with noise, outliers and sharp features. Their work classifies the neighborhood of the point into subsets of outlier-free smooth regions of the surface, and then the moving least squares projection mechanism is operated on points. But this method requires very dense point clouds and is time-consuming. Adamson and Alexa [66] adopted cell complexes to preserve the shape features by decomposing the object into cells of different dimensions, while this decomposition demanded effort by the users.
containing three features: locality, non-shrinkage and including geometrical aspects. Pauly et al. [72] introduced the discrete approximation of Laplacian to point cloud. In order to handle shrinkage, their method approximates the local volume change and compensates the shrinkage by displacing the neighbors of point \( p \) with \( (p - p')/k, p' = p + \lambda \Delta p \), where \( \Delta p \) is discrete approximation of the Laplacian at point \( p \). This method, however, could smooth features and lead to the vertex drift.

Motivated by Fourier transformation, Pauly and Gross [73] used spectral processing to filter point cloud. They applied a discrete Fourier transform to obtain a spectral decomposition of point cloud. The elaborate filtering operation was then carried out by manipulating the frequency spectrum via the Wiener filter.

Rosman et al. [74] first constructed the collaborative patch \( P_i \) using a set of similar patches for each patch \( P_i \) based on certain distance measurement, and then their algorithm included two phases. Phase I prevents shrinkage using the eigen-functions of the Laplace–Beltrami operator of the collaborative patch, followed by Wiener filtering based on the denoised estimation from phase I.

2.5. PDEs-based filtering technique

PDE (Partial Differential Equations), one of the most important tools widely used in computer vision and computer graphic, has been successfully applied to many applications such as image or mesh denoising. PDEs-based techniques for filtering point cloud can be considered as an extension of that to the triangular meshes.

By assembling the local finite matrices constructed over small point neighborhoods into a single matrix, Clarenz et al. [75] presented a framework for processing point cloud via PDEs. They discretized and solved an anisotropic geometric diffusion equation for filtering point cloud. This method could smooth out the noise whereas features were preserved or enhanced.

Extending Taubin’s work [76], Lange and Polthier [77] took the directional curvatures, principal curvatures and the Weingarten map into account to obtain an anisotropic geometric mean curvature flow method to filter point cloud. First, they used directional curvatures to produce a Weingarten map by discretizing an integral formula. Next, they computed the eigen-values and eigen-vectors corresponding to principal curvatures and directions respectively. Finally, they defined an anisotropic Laplacian for mean curvature flow technique by applying curvature information to modify Laplacian. This method can detect features such as edges but need to choose edge quotients manually.

Based on covariance analysis and constructed directional curvature, Xiao et al. [78] proposed an anisotropic curvature flow approach. The corresponding equation they used is \( \partial u / \partial t = g ( \tilde{k} n ) + (1 - g) (I - u) \), where \( G_n \) is a Gaussian kernel, \( \tilde{k} \) denotes the directional curvature. The terms \( g [ \tilde{k} H ( G_n \times n ) ] \) and \( 1 - g \) work as a moderated selector of the diffusion process. \( I - u \) is a forcing term to eliminate shrinkage. Experimental results show that their method achieves a better performance in noise removal, feature preservation, anti-shrinking and anti-point drifting.

To extend many processing methods on images into 3D point cloud, Lozes et al. [79] constructed weighted arbitrary graphs for representing point cloud, which needed to consider local neighborhood information. The transposition of framework of PDEs, such as \( \rho \)-Laplace operator [80] and PDEs morphological operators, was adapted to these arbitrary graphs (that is PDEs on weighted graphs [81]) to filter point cloud.

Because PDEs-based techniques need to compute the partial differential properties, it is time consuming to use these methods to filter point cloud.

2.6. Hybrid filtering technique

Hybrid strategies usually use two or more filtering techniques together to deal with the raw point clouds.

Liu et al. [82] developed an iterative framework to process point cloud. They first applied a Weighted Locally Optimal Projection (WLOP) operator to efficiently filter out noise. Then, they used a mean-shift-based outlier removal operator to detect and eliminate outliers. However, the limitations of their algorithm are its difficulty in recovering sharp features and its computational costs.

Zaman et al. [16] proposed a density-based approach for denoising point cloud. First, they used particle-swarm optimization technique to select the bandwidth for approximation of kernel density estimation (KDE). Then, mean-shift clustering algorithm was utilized to the KDE to remove outliers. Finally, the noise in the remaining points was reduced by applying bilateral filter. The result showed that this method is robust with highly noisy point cloud.

2.7. Other methods

There are many other methods used for filtering point cloud. Szeliski and Tonnesen et al. [83] developed oriented particles (point clouds). Each point, having a position and its own local coordinate frame, defined both a normal and a local tangent plane of the represented surface. They devised a set of potential functions to group these points into surface-like arrangements. Their method could shape the surface by moving points around.

Voxel Grid (VG) filtering method first defines a 3D voxel grid (3D boxes in 3D space) on a point cloud. Then, in each voxel, a point is chosen to approximate all the points that lie on that voxel. Normally, the centroid of these points or the center of this voxel is used as the approximation. The former is slower than the later, while its representative of underlying surface is more accurate. The VG method usually leads to geometric information loss.

Quadtree-based filtering method creates a quadtree as a data structure to organize a point cloud. This representation can improve the speed of neighborhood search, which is beneficial to the efficiency of the filtering algorithms. And then, the filtering strategies or methods (like neighborhood-based approaches or projection-based methods) are used to filter the point clouds.

Digne et al. [84] introduced a similarity based filtering to denoise point cloud. They decomposed the input point clouds into a smooth part \( S_{smooth} \) yielded by using the mean curvature motion and a high frequency term \( \delta u \). And then a non-local strategy, filtering high frequency based on L2-distance similarity between local descriptors computed for each point, was used to denoise \( \delta u \) to achieve shape denoising. Nevertheless, this algorithm relied on point densities and cannot differentiate texture and structured noise.

Table 1 presents a summary of the filtering methods in terms of noise removal, feature preservation, outlier removal and other performance. These methods are listed chronologically by year of publication.

3. Experimental results and discussion

After a comprehensive analysis of the major 3D point cloud filtering algorithms, we now proceed to conduct experiments aiming at comparing and evaluating the performance of selected point cloud filtering methods.

3.1. Selected methods

In our experiments, we select 7 different filtering methods which are widely cited and used in comparison. These algorithms have been already briefly introduced in Section 2 and include: Voxel Grid filter (VG), Normal-based Bilateral Filter (NBF), Moving Least Square (MLS), Weighted Locally Optimal Projection (WLOP), Edge Aware Resample (EAR) and L0 minimization (L0). These filtering algorithms are tested on different point cloud models corrupted with Gaussian noise to evaluate corresponding performance.
VG, NBF and MLS are available in the Point Cloud Library (PCL) Version 1.7.2. WLOP and EAR were implemented in C++, while the others (namely RMLS and L0) were programmed by MATLAB R2016a. The experiments are carried out on a PC with Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz and 16 GB memory.

3.2. Efficiency

A comprehensive evaluation of the considered filtering algorithms is given with respect to computational efficiency. We calculate the running time required by each method to filter a model of a point cloud. Since the number of points is one of the main factors that would affect the computational time, models with various numbers of points are chosen to test the performance of these methods.

3.3. Measures

In order to evaluate the quality of results, two error metrics $\delta$ and $D_{\text{mean}}$ are used. $\delta$ represents the averaged angle over all angles between the ground truth point normals and the resulting point normals. $D_{\text{mean}}$...
3.4. Results and discussion

Table 2 gives the running time of different methods over different point cloud models with different numbers of points. Some observations can be found among these algorithms. Firstly, voxel grid approach is the most efficient filter, which is nearly two or three orders of magnitude faster compared with others. Secondly, it is worth pointing out that when the number of points in models is less than 200,000, NBF can achieve almost the same performance in terms of efficiency as MLS, while with the increase of points, NBF becomes very time-consuming. RMLS and WLOP are two or three times slower than MLS and NBF on the whole. Finally, due to different strategies for normal estimation, normal filtering and point updating, L0 and EAR, therefore, are the most computationally expensive algorithms.

The error metrics for different methods are presented in Table 3. According to the error metrics, it is evident that there is an obvious difference between ground truth point clouds and filtered ones by VG, indicating that VG cannot obtain the results as expected. The filtering effectiveness of NBF is similar to VG in some extend apart from the Armadillo model. In addition, for most point cloud models, EAR and L0 achieves a better filtering performance with relatively lower \( \delta \) and \( D_{\text{mean}} \).

Table 2
Timings for our method and other filtering methods on different models (in ms).

<table>
<thead>
<tr>
<th>Model</th>
<th>Points</th>
<th>VG</th>
<th>NBF</th>
<th>MLS</th>
<th>RMLS</th>
<th>WLOP</th>
<th>EAR</th>
<th>L0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office chair</td>
<td>29,148</td>
<td>86</td>
<td>8,269</td>
<td>7,776</td>
<td>13,556</td>
<td>14,962</td>
<td>71,082</td>
<td>309,610</td>
</tr>
<tr>
<td>i-H Bunny</td>
<td>34,834</td>
<td>95</td>
<td>5,428</td>
<td>8,015</td>
<td>17,239</td>
<td>17,735</td>
<td>87,920</td>
<td>297,046</td>
</tr>
<tr>
<td>Julius</td>
<td>36,201</td>
<td>165</td>
<td>11,516</td>
<td>9,875</td>
<td>19,741</td>
<td>21,125</td>
<td>73,998</td>
<td>318,776</td>
</tr>
<tr>
<td>Sofa</td>
<td>48,668</td>
<td>222</td>
<td>14,836</td>
<td>15,863</td>
<td>30,636</td>
<td>24,923</td>
<td>98,276</td>
<td>420,314</td>
</tr>
<tr>
<td>Coffee-mug</td>
<td>49,017</td>
<td>313</td>
<td>12,750</td>
<td>11,161</td>
<td>23,824</td>
<td>25,096</td>
<td>122,657</td>
<td>343,080</td>
</tr>
<tr>
<td>Iron</td>
<td>65,574</td>
<td>324</td>
<td>24,144</td>
<td>22,558</td>
<td>44,632</td>
<td>46,702</td>
<td>175,409</td>
<td>559,228</td>
</tr>
<tr>
<td>Armadillo</td>
<td>172,974</td>
<td>767</td>
<td>20,225</td>
<td>19,741</td>
<td>30,636</td>
<td>30,636</td>
<td>73,998</td>
<td>1,211,477</td>
</tr>
<tr>
<td>Dragon</td>
<td>437,645</td>
<td>865</td>
<td>514,836</td>
<td>514,836</td>
<td>940,147</td>
<td>940,147</td>
<td>940,147</td>
<td>940,147</td>
</tr>
<tr>
<td>Table</td>
<td>460,400</td>
<td>2,050</td>
<td>590,851</td>
<td>590,851</td>
<td>1,087,200</td>
<td>1,087,200</td>
<td>1,087,200</td>
<td>1,087,200</td>
</tr>
</tbody>
</table>

Table 3
Error metrics for different methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Error</th>
<th>VG</th>
<th>NBF</th>
<th>MLS</th>
<th>RMLS</th>
<th>WLOP</th>
<th>EAR</th>
<th>L0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>( \delta )</td>
<td>32.125</td>
<td>20.336</td>
<td>8.962</td>
<td>8.932</td>
<td>8.904</td>
<td>8.785</td>
<td>8.752</td>
</tr>
<tr>
<td>i-H bunny</td>
<td>( \delta )</td>
<td>10.223</td>
<td>6.021</td>
<td>5.862</td>
<td>6.105</td>
<td>5.710</td>
<td>5.553</td>
<td>5.559</td>
</tr>
<tr>
<td>Sofa</td>
<td>( \delta )</td>
<td>12.850</td>
<td>10.126</td>
<td>10.388</td>
<td>7.211</td>
<td>5.334</td>
<td>5.186</td>
<td>5.032</td>
</tr>
<tr>
<td>Julius</td>
<td>( \delta )</td>
<td>13.139</td>
<td>10.647</td>
<td>10.392</td>
<td>3.372</td>
<td>2.166</td>
<td>3.378</td>
<td>3.364</td>
</tr>
<tr>
<td>Bowl</td>
<td>( \delta )</td>
<td>25.494</td>
<td>21.676</td>
<td>6.149</td>
<td>6.003</td>
<td>7.667</td>
<td>7.311</td>
<td>7.386</td>
</tr>
<tr>
<td>Iron</td>
<td>( \delta )</td>
<td>11.504</td>
<td>7.513</td>
<td>5.844</td>
<td>5.915</td>
<td>6.182</td>
<td>6.180</td>
<td>5.837</td>
</tr>
<tr>
<td>Armadillo</td>
<td>( \delta )</td>
<td>9.071</td>
<td>3.616</td>
<td>3.599</td>
<td>4.406</td>
<td>5.124</td>
<td>5.139</td>
<td>5.065</td>
</tr>
<tr>
<td>Table</td>
<td>( \delta )</td>
<td>32.481</td>
<td>29.313</td>
<td>17.010</td>
<td>17.449</td>
<td>28.540</td>
<td>21.918</td>
<td>15.998</td>
</tr>
</tbody>
</table>

is the average distant from the resulting points to the corresponding ground truth points.

4. Conclusion

This paper has placed a strong emphasis on the comprehensive review of the state-of-the-art algorithms for filtering 3D point cloud. Although there are a few existing research on point cloud filtering, it is believed that filtering on the raw point cloud, being as a crucial step of point cloud processing pipeline, remains a challenging task. A brief discussion of future research directions are presented as follows.

(1) Combination of color and geometric information: For point clouds, especially these containing color information, a pure color or pure geometric attributes based method cannot work well. Hence, it is expected to combine the color and geometric information in the filtering process to further increase the performance of a filtering scheme.

(2) Time complexity reduction: Because point clouds contain a large number of points, some of which can be up to hundreds of thousands of points, VG can be recommended as a better choice regarding to computational efficiency. However, its filtering effect is unsatisfactory. In contrast, EAR and L0 demonstrate a sufficient performance in terms of noise removal, together with feature preserving, yielding a relatively accurate models for further processing (e.g. object recognition). It has to be noted that they are both computationally expensive. MLS can be considered as a relatively good trade-off which provides a balance between running efficiency as well as the filtering effectiveness.
or even millions of points, computation on these point clouds is time consuming. It is necessary to develop filtering technologies to filter point cloud effectively to reduce time complexity.

(3) Filtering on point cloud sequence: Since object recognition from a point cloud sequence will become the future research direction. And filtering the point cloud sequence will help to improve the performance and accuracy of object recognition.
Fig. 9. (a) Noisy table model; (b) filtering result with VG; (c) NBF; (d) MLS; (e) RMLS; (f) WLOP; (g) EAR; (h) L0.

References


