Reliable Rolling-guided Point Normal Filtering for Surface Texture Removal

Y. Sun¹*, H. Chen¹*, J. Qin², H. Li³, M. Wei† and H. Zong††

¹Nanjing University of Aeronautics and Astronautics, Nanjing, China
{yangxingsun, chenhonghuaen, mingqiang.wei}@gmail.com
²The Hong Kong Polytechnic University, Hong Kong, China
harry.qin@polyu.edu.hk
³AVIC XI’AN Aircraft Industry (GROUP) Company LTD, Xi’an, China
lhw_xac@163.com
⁴National Key Laboratory of Science and Technology, Aerospace Intelligent Control, Beijing, China
zonghua3@sina.cn

Figure 1: Comparison of surface texture removal with several cutting edge methods. From left to right: (a) the input model which contains rich features like snake scales on the body, patterns and textures on the ingot, and filtering results from (b) RIMLS [ÖGG09], (c) EAR [HWG⁺¹³], (d) Zheng et al. [ZLX⁺¹⁸], and (e) our rolling-guided point normal filtering. The methods like (b,c) are able to filter out the noise but they cannot remove the textures clearly. The method like (d) can suppress the textures but with the price of surface over-smoothing. In contrast, our method removes these textures completely while preserving surface structures well.

Abstract
Semantic surface decomposition (SSD) facilitates various geometry processing and product re-design tasks. Filter-based techniques are meaningful and widely used to achieve the SSD, which however often leads to surface either under-fitting or over-fitting. In this paper, we propose a reliable rolling-guided point normal filtering method to decompose textures from a captured point cloud surface. Our method is built on the geometry assumption that 3D surfaces are comprised of an underlying shape (US) and a variety of bump ups and downs (BUDs) on the US. We have three core contributions. First, by considering the BUDs as surface textures, we present a RANSAC-based sub-neighborhood detection scheme to distinguish the US and the textures. Second, to better preserve the US (especially the prominent structures), we introduce a patch shift scheme to estimate the guidance normal for feeding the rolling-guided filter. Third, we formulate a new position updating scheme to alleviate the common uneven distribution of points. Both visual and numerical experiments demonstrate that our method is comparable to state-of-the-art methods in terms of the robustness of texture removal and the effectiveness of the underlying shape preservation.

CCS Concepts
• Computing methodologies → Point-based models; Shape analysis;

† M. Wei and H. Zong are co-corresponding authors.
1. Introduction

Various kinds of 3D laser scanners and depth cameras have been developed in recent years. This has facilitated the use of point clouds as the basic data source for many practical applications [WY12, WTP19, WX17], such as 3D measurement, shape analysis [YSGG17, WHG18] and reverse engineering [WGY12]. The scanned high-quality point clouds often contain rich details, such as bumps, ridges, creases and repeated patterns. These geometric details, i.e., textures, usually act as providing the vivid appearance as bumps, ridges, creases and repeated patterns. These geometric scanned high-quality point clouds often contain rich details, such as bumps, ridges, creases and repeated patterns. These geometric

Although the human visual system can easily distinguish the basic shape from the rich textures, the task of automatic texture removal is still much challenging for a computer. This is because it is hard to distinguish large-scale details and prominent shape structures. Recent point cloud filtering works [ÖGG09, HWG13, WXL13, LWC18] designed to remove noise and outliers, have achieved the noticeable success in smoothing small-scale noise. However, these methods either have difficulties in entirely eliminating textures or may cause several types of artifacts in smoothing results, such as texture residue, structure collapse, shape distortion, and over-smoothing (see Fig. 1 (b), (c)).

Texture filtering has been studied extensively in image processing in past decades [CLKL14, TM98, XLLJ11, KEE13]. In particular, the rolling guidance filter presented by Zhang et al. [ZSXJ14] and its counterparts (RGNF, rolling guidance normal filter) in the 3D field [ZLLX18, WFL15] have achieved impressive results. These methods commonly first utilize the Gaussian filter to smooth the input image or 3D surface and then recover the over-smoothed structures by a rolling guidance filter. Despite the effectiveness of RGNF, it is still very difficult to perfectly suppress geometric details, while preserving intrinsic properties of geometry in some challenging regions (see Fig. 1 for a reference), since RGNF may suffer from the unreliable guidance map.

In this paper, we present a reliable rolling-guided point normal filtering method. Our method improves over existing works in terms of texture removal and the preservation of the underlying surface (US) for two reasons. First, a RANSAC-based texture identification is performed to distinguish the textures (i.e. the bump ups and downs, BUDs) and US. Each point is assigned with a sub-neighborhood, in which all points are the non-texture points. Second, we design a reliable guidance model and follow the RGNF framework to effectively suppress textures, while recovering prominent structures. Further supported by a new point updating formulation, point positions are adjusted to be compatible with the filtered normals. We have tested our method on a variety of point clouds with abundant textures, and shown that it is comparable to state-of-the-art methods, in terms of both visual and numerical comparisons. The main contributions are three-fold:

- We devise a RANSAC-based strategy to distinguish the structures and textures, which provides texture-free neighborhoods for the following normal filtering operation.
- We propose a reliable rolling-guided point normal filter, which is able to suppress multi-scale textures, while preserving prominent structures.
- We formulate a new energy function for point updating, which can relieve the generation of gaps around feature edges.

2. Related Work

Many 3D filtering/denoising methods [WZY12, WWG18] are originated from the 2D image domain. In this section, we briefly review the main methods for addressing 2D/3D geometry filtering problems [HCP15, YGQ19, WY09], while concentrating on the solutions that are most closely related to ours.

2.1. Image texture filtering

Image texture filtering has drawn the attention of many researchers in recent years. It aims to smooth an image, while persevering its prominent structures. Bilateral filtering is a widely-used method in image texture smoothing proposed by Tomasi et al. [TM98] and Durand et al. [DD02]. However, it indicates that artifacts are easily generated due to the continuous coarsening process [FFLS08]. To add these issues, researchers present the weighted least squares (WLS) filtering method. Zhang et al. [ZSXJ14] presented the rolling guidance filter to distinguish textures and structures. Another widely studied framework is the total variation (TV) model [ROF92]. It has resulted in an extensive number of variants [AGCO06, BLMV10, XYXJ12] by changing the regularization and data fidelity terms. Among these efforts, Xu et al.
Figure 3: Algorithm overview. Starting from a point cloud model with rich textures, we first make a distinction between textures and structures and generate the sub-neighborhood, based on which, we then entirely smooth geometric features. With the over-smoothed results, we proceed to recover prominent structures which have been blurred and smoothed in the first iteration of rolling-guided normal filter, with a credible guidance model providing the information of the structures. Finally, point positions are updated to fit the final normal field.

[XYXJ12] introduced the relative total variation (RTV) metric to better separate the structures and textures. The assumption of this method is that in a local window, the RTV values of textures are much higher than the values of structures, but the choice of a suitable window is non-trivial.

2.2. Geometric texture filtering

Based on image texture filtering, some studies about the texture filtering [WFL*15, CHR*19] in 3D domain have been developed. For example, inspired by the bilateral texture filtering [CLKL14], Zhang et al. [ZDZ*15] proposed a framework of guided mesh normal filtering. Two stages included in this framework are joint bilateral filtering and vertex updating, respectively. A reliable guidance normal field is presented to imply the surface features in the process of the joint bilateral filtering. Similarly, Wang et al. [WFL*15] extended the image filtering work [ZSXJ14] to geometric processing. It is noticeable that the rolling guidance normal filter (RGNF) is also a two-stage method and is able to deal with multi-scale features. Moreover, they use the modified Poisson-based gradient deformation method to update mesh vertex positions. Furthermore, a new geometry filtering method named static/dynamic (SD) filter was presented by Zhang et al. [ZDH*19] recently. The SD filter is also a variant of image processing [HCP15]. The SD filter combines the static guidance and dynamic guidance together to filter triangular face normals.

In the point cloud domain, there are at present relatively few studies on texture removal due to the complexity of 3D space and the lack of explicit topological information. Recently, Zheng et al. [ZLX*18] designed a texture filtering method. They recover the normals of point cloud by first applying the rolling filter on the normal field and then updating point positions to match the filtered normal field. To avoid introducing additional artifacts, a new energy term is added during point updating.

2.3. Point cloud denoising

Point cloud denoising is closely related to the research of texture removal. In the past few decades, many efforts have dedicated to it and numerous excellent methods [ABC*01, HWG*13, CWS*19] have been proposed. The method of moving least squares (MLS) [Lev98, ABC*03, GGG08] is proposed for surface reconstruction. It focuses on the $C^0$ discontinuities at the edges and corners. However, such a strategy tends to produce over-smoothed results and cannot handle the situation of sparse sampling. To address this problem, Öztireli et al. [OGG09] devised the robust implicit moving least squares (RIMLS) for feature-preserving point set projection, by using the technique of robust statistics. Nevertheless, there may be several drawbacks to these methods. For instance, the convexity of the input shape cannot be preserved well and the projection operators are not stable, if the curvature is high. Moreover, for the purpose of denoising, most of the methods are using the neighborhood centering at the point to filter the normals, which is prone to produce inaccurate normals. To conquer this problem, Cao et al. [CCZ*18] presented a shifted neighborhood scheme to estimate normals. However, this method is not robust to large-scale noise.

In addition, the number of promising point set projection methods based on the locally optimal projection (LOP) operator [LCLT07] have increased recently. LOP is a local surface approximation method without any local orientation information. Based on the LOP, Huang et al. [HLZ*09] introduced its weighted version, which can produce a denoised and more uniformly-distributed point cloud, by including a new repulsion term. Sun et al. [SSW15] utilized the $L_0$ minimization, which can produce sharper results than $L_1$ and $L_2$ norm. However, a post-processing operation is re-
Figure 4: Illustration of the proposed sub-neighborhood. The traditional Gaussian filter is modified by utilizing the sub-neighborhood. For point $p_i$, only part of the points in the neighborhoods within radius $r$ are useful (red points constitute the sub-neighborhood). Grey points indicate that they should not work for $p_i$. We can see that when $r$ is small, both the traditional and modified Gaussian filters do not perform well ((b)-(c)). In (d) where $r$ is large, it is obvious that the traditional Gaussian filter leads to over-smoothing and several normals are also incorrect. By contrast, by using the sub-neighborhood, the textures are all smoothed correctly and the model avoids being over-smoothed (e). The comparisons of the traditional and modified Gaussian filters are shown in (f), which the points with red normals represent the result of the traditional Gaussian filter while the points with blue normals represent the result of the modified Gaussian filter.

required to solve the cross artifact problem in the area with sharp edges. Moreover, Yadav et al. [YRS+18] used a framework based on a normal voting tensor to denoise anisotropic point sets. They apply restricted quadratic error metrics on the vertex normals, which are processed by a vertex-based normal voting tensor and binary eigenvalues optimization. However, erroneous results may be produced under large-scale noise and irregular sampling.

Additionally, EAR [HWG+13] was proposed as a resampling algorithm to process noisy point sets. The key idea is to compute reliable normals of the samples away from the edge and then resample the point set. The method achieves pleasing results with sharp features well preserved.

In summary, although an extensive number of techniques are devoted to noise removal, there are few studies that specialize in texture removal for point clouds. Meanwhile, they cannot produce a satisfactory output because of the difficulty of distinguishing various scales of features.

3. Algorithm Overview

Fig. 3 gives an overview of the proposed algorithm: (i) make a distinction between structures and textures and then generate the sub-neighborhood; (ii) utilize our sub-neighborhood to remove textures (Section 4); (iii) recover the prominent structures under the function of our designed rolling-guided normal filter (Section 5); and (iv) update the point positions to match the filtered normals (Section 6). Before presenting the details of our algorithm, we first introduce some basic notations that are used in the paper:

- $P = \{p_i\}_{i=1}^m \subset \mathbb{R}^3$ denotes an unorganized 3D point cloud with $m$ points;
- $N = \{n_i\}_{i=1}^m \subset \mathbb{R}^3$ is the initial normal field of the input point cloud;
- $N^k = \{n_i^k\}_{i=1}^m \subset \mathbb{R}^3$ denotes the filtered normal field of $k$-th iteration;
- $\Omega_i$ denotes the sub-neighborhood of $p_i$ in which the points are all classified as non-structure points;
- $R_i$ denotes the nearest neighborhood of $p_i$ where the neighbors are within a fixed radius centered at $p_i$.

4. Texture Removal

Our first operation is to filter textures as completely as possible. Then an over-smoothed result is obtained, whose prominent structures will be recovered later. From the point of view that the scale of the textures is large, an intuitive and simple way is to employ a larger smoothing neighborhood, which will eventually lead to ex-
cessive smoothness (Fig. 4 (d)). Consequently, recovering structures in the follow-up steps is challenging. In addition, it is difficult to erase the textures entirely by using the Gaussian filter only once (see Fig. 4 (b) and Fig. 4(c)) as the method used in [ZLX’18, WFL’15]. Moreover, to the best of our knowledge, there is no direct method to point out the distribution of the structures and textures within a model. When filtering the normal of \( p_i \), it is meaningful to consider the influences from the neighboring points, according to their roles (structure, texture, or neither), for the reason that different neighbors have different effects on \( p_i \).

Different from [ZLX’18, WFL’15], an improved Gaussian filter is devised to apply on the initial normal field:

\[
n_i^1 = \frac{1}{W_i} \sum_{j \in \Omega_i} w(p_i, p_j, \sigma_j)n_j
\]  

(1)

where \( n_i^1 \) represents the output normals, \( w(\cdot) \) is the Gaussian weight defined as \( w(x, y, \sigma) = \exp(-\frac{||x-y||^2}{2\sigma^2}) \). \( W_i \) indicates normalization and \( \Omega_i \) is a sub-neighborhood of \( p_i \) which is obtained by texture identification concretely described nextly. For the traditional Gaussian filter, employing the \( k \)-nearest neighbors (KNN) or the neighbors within a fixed radius (RNN) of \( p_i \) to filter the normals is a common operation. Under the observation that the information about the normals from non-texture points is more reliable than those from the texture points, we modify the neighborhood of each point by a sub-neighborhood \( \Omega_i \), within which all points are non-texture points. By utilizing the modified Gaussian filter, textures whose scale is smaller than the spatial variance parameter \( \sigma_C \) can be smoothed. In addition, points regarded as texture points can be filtered directly by using the sub-neighborhood \( \Omega_i \). It has been shown in Fig. 4(e) that in \( \Omega_i \) only non-texture points participate in the normal filtering of \( p_i \). Based on this, the normals are filtered accurately and consistently. The comparison of the traditional Gaussian filter and our modified Gaussian filter is shown in Fig. 4 (f).

**Texture identification.** Now, we particularly introduce the distinction between the structures and textures, based on which, sub-neighborhood mentioned above is generated. Under the assumption that 3D surfaces are comprised of various BUDs and an US which is piece-wise flat, the main idea of our algorithm is to adopt a RANSAC-based strategy to detect latent texture points. In some sense, we deem that the textured area in the model is like a flat plane with some outliers (textures). RANSAC is an effective fitting method, when there exists noise or outliers. Therefore, by first applying it to the RNN of each point, the times that each point is considered to be an inlier are calculated. Then, a statistical technique is proposed to distinguish the textures and structures as:

\[
T_i > \delta * N_i
\]  

(2)

in which \( T_i \) is the times of \( p_i \) considered as an inlier. \( N_i \) represents the number of \( p_i \)'s RNN. It is also equal to the times that \( p_i \) is distinguished as an inlier or an outlier. \( \delta \) is the proportionality coefficient tuned by the user. It determines the proportion threshold that \( p_i \) is considered as a non-texture point. If \( \delta \) is larger, the distinction of the structures and textures is more obvious. This equation implies that \( p_i \) is regarded as a texture or structure, when the times of \( p_i \) considered as an inlier reaches a certain percentage of its neighbor number. As Fig. 5 has shown, by using our proposed texture identification strategy, a cube model can be divided into many tiny bumps and a main body. Through making a distinction between textures and structures, we design the sub-neighborhood \( \Omega_i \), which conduces to the removal of the textures as Fig. 4 has demonstrated.

5. Structure Recovery

Since RGNF is able to recover large-scale structures, we also exploit this scheme to recover the structures that have been blurred in the first iteration of RGNF. Prominent structures can be recovered by:

\[
n_i^{k+1} = \frac{1}{W_i} \sum_{j \in R_i} w(p_i, p_j, \sigma_j)n_j
\]  

(3)

where \( n_i^{k+1} \) in the \((k+1)\)-th iteration is obtained in a joint bilateral filtering form, given the input \( n_j \) and the guidance \( G_j^k \) which is estimated in the previous iteration. The generation of guidance normals \( G \) is explained later in detail. \( \sigma_j \) is the variance parameter for the range kernel that relates to the averaging weights to the guidance normal difference.

The expression of Eq. 3 can be understood as a filter that smooths the input normal \( N \) guided by the structure \( G^k \). For the

---

**Figure 5:** Given a cube model with bumps, we apply the RANSAC-based plane fitting method to each point’s RNN, and then take a statistic-based strategy to distinguish structures and textures. Left: the original point cloud; center: structures extracted from the input; right: textures extracted from the input. It is obvious that structures and textures make up the input.

**Figure 6:** By applying our rolling-guided normal filter on the Owl model, textures are smoothed while structures are maintained well. As the number of iterations increases, we can see that the features of large scales which have been filtered in the first iteration are recovered. It is obvious that the normals on the ear and mouth have the maximal changes.
Figure 7: A visual exhibition of the guidance normal and the effectiveness of the modified guidance: (a) is the input which is a leg model of the Armadillo; (b) shows the guidance normal computed from the input by Eq. 6; (c) is the guidance normal computed by Eq. 7, which is more reliable than (b). (d) and (e) represent the results based on the guidance normal (b) and (c) respectively.

classical bilateral filtering [JDZ04], as the number of iterations increases, the produced surface becomes more and more smooth, due to the filter working on the new normals from the last iteration. Different from [JDZ04], although small- and large-scale features are both smoothed in the first iteration, our method can recover the features of larger scale in the later iterations. This is because the normal difference of large-scale features is enhanced via the rolling filtering on the original normals. Fig. 6 shows the filtered normals of the sensitivity of the normal concerning the underlying surface. 

Guidance normals. It is worth noting that the choice of the guidance normal is an important step in our algorithm. Although the rolling-based scheme can recover some features, providing a guided model with proper information of structures for the rolling guidance normal filter is warranted. There are two main reasons. On the one hand, we can control the features of the output on account of the sensitivity of the normal concerning the underlying surface. This characteristic facilitates the control of the filtering process by incorporating the guidance. On the other hand, a suitable guidance can provide more reliable information about the structure of the desired output. Thereby, more suitable weights are produced to direct the filter towards the desired results.

Inspired by the patch shift used in bilateral texture filtering [CLKL14], we also select the most reliable one in the RNN of \( p_i \) and pick its normal as the guided normal of \( p_i \). For each point \( p_j \) in \( R_i \), the reliability of its normal is measured by a function as:

\[
G_i = \max(D(p_{ij}) \cdot C(p_i))
\]

(4)

where \( D(p_{ij}) \) denotes the angle differences between \( n_i \) and \( n_j \):

\[
D(p_{ij}) = \langle n_i, n_j \rangle
\]

(5)

This measures the variation between the normal of the point \( p_i \) and its neighbors. The bigger the difference is, the more uneven \( p_i \) is, indicating that the point \( p_i \) is closer to the feature area. In addition, \( C(p_i) \) represents the mean curvature of \( p_i \). This term expresses the characteristic saliency of \( p_i \) itself. The maximum product of \( D(p_{ij}) \) and \( C(p_i) \) is selected as the value of \( G(p_i) \), which considers the maximum difference between \( p_i \) and its neighbors as well as the feature of \( p_i \) itself.

As the value of \( G(p_i) \) has been obtained, by comparing the value of \( G(p_i) \) and \( G(p_j) \), we pick the normal of point with the minimal value among RNN of \( p_i \) as \( p_i \)'s guidance normal:

\[
G_i' = n_i^{(k-1)} \quad s.t. \quad G(p_i) = \min_{j \in R_i} G(p_j)
\]

(6)

In some narrow and small regions, the guidance normals generated by Eq. 6 may be undesirable. As shown in Fig. 7 (b), the guidance normals tend to be same in the area of tiptoes. Two main reasons lead to this undesirable result. First, owing to the distance between the upper and lower surfaces is very small, the RNN of \( p_i \) can contain the points in these two close-by surfaces. Second, since the normals of the bottom surface of the leg are almost identical, according to the rule of Eq. 6, the guidance normals of tiptoes are similar while choosing the similar normals on the bottom of the toes. To address this issue, we modify the guidance normal mentioned above, by utilizing a trade-off variable \( \alpha \) to interpolate the guidance normal \( G \) and the filtered normal \( N_i \) obtained from the last iteration. Mathematically, the new guidance is written as:

\[
G_i' = \alpha_i G_i + (1 - \alpha_i) N_i^{(k-1)}
\]

(7)

where

\[
\alpha_i = \frac{\sum_{j \in R_i} G(p_j)}{G(p_i)}
\]

(8)

In a general way, we set \( \sigma_i \) to the mean of \( \sum_{j \in R_i} C(p_j) \). As shown in Fig. 7, the modified guided model is more confident and reliable than \( G \), which leads to a promising result.

6. Position Updating

After obtaining the filtered normal field denoted by \( N^k = \{n_i^k\}_{i=1}^m \) in Section 5, we proceed to compute the point positions, according to the new normals, so that the positions of points well match the filtered normal field. And for that, we formulate the updating problem as:

\[
\arg \min \sum_{i \in R} \sum_{j \in R_i} w(n_i^k, p_i - p_j, \sigma_p)||p_i’ - p_j||
\]

(9)

where \( w(n_i^k, p_i’ - p_j, \sigma_p) \) is the Gaussian weight. Eq. 9 means the weighted sum of distances to the tangent planes defined by the neighboring points of \( p_i \) and their corresponding filtered normals.


\[ \sum_{j \in \Psi_i} w(n_j^k, p_i - p_j, \sigma_j) \left\| (p_i - p_j) \cdot n_j^k \right\| \]

The isotropic neighborhood \( \Psi_i \), assigning the reliable neighboring points to each point by limiting the normal angle between points, is able to produce a pleasing result. As shown in Fig. 8 (b), under this mechanism, our devised optimization can greatly reduce the non-uniform distribution artifacts.

### 7. Results and Analysis

#### Implementation and timings

We have implemented the proposed algorithm in C++ and all experiments are performed on a desktop PC with a 3.70 GHz Intel Core i7 CPU and 16GB of RAM. Table 1 reports the computational timings of several methods on some models. Note that our method is not superior in running efficiency because of the operation of RANSAC for each point and the iterations. Among these, the distance threshold of RANSAC affects the running time largely which leads to the running time varying greatly on different models. The running time of RANSAC increases with the decrease of the distance threshold. For several complex models, taking Fig. 13 as an example, which is composed of multi-scale features, a small threshold is needed to ensure the distinction between the structures and textures. As a result, the running time is slower than other methods (see Table 1).

#### Parameter settings

The main parameters of our algorithm are summarized below:

- \( \lambda \): the distance threshold for RANSAC.
- \( r \): the radius of point’s neighborhood.
- \( \sigma_i \): related to the scale size of geometry features.
- \( \sigma_f \): related to the smoothness of the final results.
- \( \delta \): the parameter used to determine the required minimal proportion of a point considered as an inlier in its RNN.

A smaller \( \lambda \) can get the finer plane fitting which also leads to being time-consuming. The value of \( r \) is set according to the texture scale. For \( \sigma_i \) and \( \sigma_f \), their choices are intuitive. The final results will be smoother with a larger \( \sigma_i \). In addition, with a larger \( \sigma_f \), larger-scale geometric features can be also smoothed. Fig. 9 shows the effect of different values of \( \delta \) on the pyramid model. Increasing the value of \( \delta \) may incorrectly identify some structures as textures. We often set \( \delta \in [0.5, 0.9] \) in our method. It is worth noting that, during the experiments, we test the influence of \( \delta \) on the final output. We find that a suitable \( \delta \) can generate a good sub-neighborhood which facilitates the normal filtering by only filtering the model once as mentioned in Section 4. We list detailed information about these parameters in Table 2.

#### Comparison with state-of-the-art methods

We compare our approach with cutting edge methods RIMLS (results produced in Meshlab) [ÖGG09], EAR (running with their released code [HWG13] and rolling normal filter (running with the code from the author) [ZLX18]). For fair comparisons, the optimal results are selected for each method, by carefully tuning the parameters. The main comparisons are shown in Figs. 10, 11, 12, 13, 14 and 15. From all of these comparisons, we notice that these methods do not achieve a good balance between texture removal and maintenance of the underlying shape. RIMLS is designed for preserving the sharp features when dealing with noise. Although it defines a piecewise smooth surface, the results are not smooth enough from a holistic perspective. It indicates that textures are not removed clearly (see Figs. 12 (b), 13 (b), 14 (c) and 15 (c)).

EAR is a resampling algorithm which is capable of generating clean point sets and preserving sharp features. It appears to have poor performance, when applied to texture removal, as Fig. 11 and Fig. 13 have shown. EAR cannot filter the large-scale features clearly and leads to over-sharpening.

Since RIMLS and EAR are not elaborately designed for texture removal, it is somewhat not fair to evaluate these two methods with ours. However, there are few approaches that specialize in texture removal. Fortunately, the rolling normal filter [ZLX18] which aims at removing surface textures can be compared. It can be seen from Fig. 10 and Fig. 12 that the zoomed regions are greatly

### Table 1: Timings (seconds) of each method.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 10</td>
<td>150k</td>
<td>368.69</td>
<td>29.70</td>
<td>132.33</td>
<td>186.95</td>
</tr>
<tr>
<td>Fig. 11</td>
<td>250k</td>
<td>521.81</td>
<td>133.05</td>
<td>398.44</td>
<td>190.40</td>
</tr>
<tr>
<td>Fig. 12</td>
<td>150k</td>
<td>214.64</td>
<td>45.95</td>
<td>192.99</td>
<td>163.62</td>
</tr>
<tr>
<td>Fig. 13</td>
<td>150k</td>
<td>287.60</td>
<td>26.63</td>
<td>153.63</td>
<td>461.32</td>
</tr>
</tbody>
</table>

### Table 2: Parameter settings of our method for several experimental models.

<table>
<thead>
<tr>
<th>Model</th>
<th>( \lambda )</th>
<th>( r )</th>
<th>( \sigma_i )</th>
<th>( \sigma_f )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 10</td>
<td>0.001</td>
<td>0.025</td>
<td>0.03</td>
<td>45°</td>
<td>0.8</td>
</tr>
<tr>
<td>Fig. 11</td>
<td>0.001</td>
<td>0.012</td>
<td>0.02</td>
<td>30°</td>
<td>0.85</td>
</tr>
<tr>
<td>Fig. 12</td>
<td>0.001</td>
<td>0.03</td>
<td>0.03</td>
<td>45°</td>
<td>0.85</td>
</tr>
<tr>
<td>Fig. 13</td>
<td>0.0005</td>
<td>0.05</td>
<td>0.05</td>
<td>30°</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Figure 10: Comparison of texture filtering on the Chinese-lion model with the cutting edge filtering methods. Compared to other methods, our algorithm can smooth out the textures clearly and avoids the phenomenon of over-sharpening occurred in (d). Please refer to the zoomed-in regions.

Figure 11: Comparison of texture filtering on the Terracotta Army model with the cutting edge filtering methods. Our algorithm is capable of smoothing out repetitive patterns, while maintaining the structures well. Please refer to the zoomed-in regions.

different from the original models. In detail, these features are all sharper than the input. In addition, we can note that textures may not smoothed out clearly in Fig. 13 and Fig. 15. Due to the credible guidance, our algorithm performs well in retaining structures and our results exhibit a good fidelity to the input.

More visual results. We have applied our algorithm on more noise-free models with rich textures. Fig. 16 demonstrates that the results of our method are visually-pleasing. Our method not only removes small-scale details, but also preserves main structures well.

Raw point cloud. Apart from the above synthetic point clouds, we also verify our method on two scanned models with minuscule raw noise, as shown in Fig. 17. Compared to the other three methods, our method can remove noise and textures simultaneously and cleanly as Fig. 17 has shown. In detail, RIMLS and the rolling normal filter of [ZLX*18] are able to remove textures with some residuals. However, the results of EAR do not change significantly compared with that of the inputs.

Quantitative analysis. We numerically analyze all the methods only on two models, since there is a shortage of texture-rich models with their ground truths. The error is measured by three metrics:

- $D_{\text{max}}$: the maximum distance from the resulting point cloud to the ground-truth point cloud.
- $D_{\text{mean}}$: the mean distance from the resulting point cloud to the ground-truth point cloud.
- $\sigma$: the distance standard deviation.

Table 3 records the quantitative comparison between the presented approach and some other methods for two models (see from Fig. 14 and Fig. 15). For the Sphere model, our method achieves the best results, both visually and quantitatively. For the Bell model, although our approach could get a relatively good result visually,
Figure 12: Comparison of texture filtering on the Dragon model with the cutting edge filtering methods. Our algorithm is capable of removing textures and maintaining accurate normals simultaneously. Please refer to the zoomed-in regions.

Figure 13: Comparison of texture filtering on the Gargoyle model with the cutting edge filtering methods. Our algorithm is capable of smoothing textures clearly. Please refer to the zoomed-in regions.

Figure 14: Comparison of texture filtering on the Sphere model with the cutting edge filtering methods. Our algorithm is capable of filtering different scales of features clearly on the underlying shape.

Figure 15: Comparison of texture filtering on the Bell model with the cutting edge filtering methods. Our algorithm is capable of producing the most faithful result in terms of the ground truth.
the numerical error is not the best for the reason that external expansion occurs during position updating.

Limitations. First, the efficiency of our approach is limited, which leads to the failure of online processing. For future work, our method could benefit from parallelizing the process of RANSAC such that the efficiency could be significantly increased. It can also be seen in Fig. 16 that our method filters several detailed features with important semantic information, such as eyes. It is a meaningful direction to remove the textures, while keeping semantic structures, even if the scale of these structures is relatively small. For distinguishing the textures and structures, we consider a model to be made up of numerous small planes, namely employing RANSAC to fit a series of planes. During the test, we try to use a quadric surface to fit the underlying surface of a model (see Fig. 18) to pursue a higher accuracy. Results indicate that a simple quadratic function is difficult to fit all kinds of surfaces. On the other hand, using high-order functions will increase the complexity of the algorithm. What’s more, there are no significant variations of the results by exploiting high-order surface fitting. On balance, in our work, we use the simplest plane fitting.
8. Conclusion

In this paper, we propose a reliable rolling-guided point normal filtering for surface texture removal. First, we design a RANSAC-based strategy to distinguish textures and the underlying surface, and employ the result of the distinction to generate the sub-neighborhood in which the points are non-texture points. Then, based on the sub-neighborhood, we remove the textures by utilizing the improved Gaussian filter. To recover the structures smoothed in the process of texture removal, a modified guidance is also devised to provide desirable information about the structures, and then we devise a reliable rolling-guided normal filter thanks to its property that is able to regain the large scale of features gradually. For position updating, we utilize the isotropic neighborhood to alleviate the phenomenon of gaps around the edge. A variety of experiments has demonstrated that our algorithm can smooth out various kinds of features, while keeping consistency with the original model in the non-texture region.

Acknowledgments

The authors would like to thank the reviewers for their comments and suggestions for improving the paper. We would also like to thank the authors of [ZLX*18] for sharing executable files. Y. Sun and H. Chen were equal to this work. This work was supported by the grants from the National Natural Science Foundation of China (No. 61502137) and the Hong Kong Research Grants Council (No. PolyU 152035/17E).

Table 3: Error metrics for different methods. For each model, the best error metric value is highlighted in bold.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere (Fig. 14)</td>
<td>$D_{max}$</td>
<td>0.07346</td>
<td>0.0793</td>
<td>0.0559</td>
<td><strong>0.0519</strong></td>
</tr>
<tr>
<td></td>
<td>$D_{mean}$</td>
<td>0.0262</td>
<td>0.0294</td>
<td><strong>0.0216</strong></td>
<td><strong>0.0216</strong></td>
</tr>
<tr>
<td></td>
<td>$\sigma$</td>
<td>0.0194</td>
<td>0.0220</td>
<td>0.0131</td>
<td><strong>0.0128</strong></td>
</tr>
<tr>
<td>Bell (Fig. 15)</td>
<td>$D_{max}$</td>
<td>0.0412</td>
<td><strong>0.0353</strong></td>
<td>0.0428</td>
<td>0.0405</td>
</tr>
<tr>
<td></td>
<td>$D_{mean}$</td>
<td>0.0097</td>
<td><strong>0.0095</strong></td>
<td>0.0105</td>
<td>0.0096</td>
</tr>
<tr>
<td></td>
<td>$\sigma$</td>
<td>0.0081</td>
<td><strong>0.0076</strong></td>
<td>0.0091</td>
<td>0.0082</td>
</tr>
</tbody>
</table>

References


