Journal of Computational and Applied Mathematics 329 (2018) 57–67

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Normal estimation via shifted neighborhood for point cloud

Junjie Cao [a,b], He Chen [a], Jie Zhang [c,*], Yujiao Li [a], Xiuping Liu [a], Changqing Zou [d]

[a] School of Mathematical Sciences, Dalian University of Technology, Dalian, China
[b] College of Mathematics and Information Science, Nanchang Hangkong University, Nanchang, China
[c] School of Mathematics, Liaoning Normal University, Dalian, China
[d] Hengyang Normal University, China

A R T I C L E I N F O

Article history:
Received 31 October 2016
Received in revised form 17 February 2017

Keywords:
Normal estimation
Point cloud
Neighborhood shift

A B S T R A C T

For accurately estimating the normal of a point, the structure of its neighborhood has to be analyzed. All the previous methods use some neighborhood centering at the point, which is prone to be sampled from different surface patches when the point is near sharp features. Then more inaccurate normals or higher computation cost may be unavoidable. To conquer this problem, we present a fast and quality normal estimator based on neighborhood shift. Instead of using the neighborhood centered at the point, we wish to locate a neighborhood containing the point but clear of sharp features, which is usually not centering at the point. Two specific neighborhood shift techniques are designed in view of the complex structure of sharp features and the characteristic of raw point clouds. The experiments show that our method out-performs previous normal estimators in either quality or running time, even in the presence of noise and anisotropic sampling.

1. Introduction

Estimating surface normals in a point cloud is a crucial preprocessing operation. High quality normals benefit numerous point clouds processing algorithms, such as surface reconstruction [1], geometric primitive extraction [2], anisotropic smoothing [3] and point based rendering [4]. Although it has been extensively studied, accurate computation near various features in the presence of noise and non-uniform sampling is always a recurrent issue.

The normal of a point is approximated by analyzing the geometry structure of its local neighborhood. The methods [1,5–7] use the whole neighborhood centered at the point on the assumption that the surface is smooth everywhere. Even when different weights are assigned to all its neighbor points according to positions and initial normals, blurred edges are unavoidable since points belonging to different surface regions are taken into consideration. To alleviate the problem, different voting techniques are employed, such as RNE [8] and HF [9]. However, some inaccurate normals may still exist in the vicinity of sharp features with anisotropic sampling or large dihedral angles. There are also segmentation based approaches [10,11], that explicitly segment the anisotropic neighborhood into several isotropic sub-neighborhoods to avoid using points of different surface regions. Higher performance is ensured with the cost of longer runtime.

In this paper, we present a brand novel approach to construct the neighborhood for fast normal estimation. Instead of using the neighborhood centering at the current point, a set of neighborhoods containing the current point is evaluated and the one with the most consistent normals is selected as the neighborhood of the current point. Thus the selected neighborhood has more possibility to be isotropic, i.e. excluding points from different regions, which will lead to more

[*] Corresponding author.
E-mail address: zhangjiexl@126.com (J. Zhang).

http://dx.doi.org/10.1016/j.cam.2017.04.027
0377-0427/© 2017 Elsevier B.V. All rights reserved.

faithful normal estimation. In the above core idea, the construction of the set of candidate neighborhoods is vital. In view of the characteristic of point clouds and corner features, specific candidate neighborhoods construction methods are designed. We also introduce a criterion considering both flatness and distance to evaluate the normals' consistency of a neighborhood. Thus our method can estimate normals accurately and fast even in the presence of noise and anisotropic sampling, while preserving sharp features. The experiments illustrate the effectiveness of the proposed method. The contributions of this paper are twofold:

- A novel perspective of constructing a neighborhood with consistent normals is presented for point cloud normal estimation. No longer taking the current point as the center of the constructed neighborhood differentiates our method from the existing normal estimators.
- Specific strategies for building a set of candidate neighborhoods are designed respecting the characteristic of point clouds, which make our method fast and effective.

2. Related works

For point cloud models, the normal of a point depends on the points of its vicinity, which is usually a neighborhood centering at the point. However, the surface of a 3D model is usually not smooth everywhere, more likely to be piecewise smooth. Thus the neighborhood of a point nearby feature area might cross borders of different manifolds. To avoid the influence from other manifolds, various methods are introduced. They may be roughly divided into three categories as follows.

To estimate the normal of a point, approaches in the first category take use of all the points of a neighborhood centering at the point. Hoppe et al. [1] (PCA) estimate the normal by fitting a plane of the local neighborhood, which is fast to compute, and works properly on smooth manifolds. However, for piecewise smooth surfaces, this method is unreliable. To get rid of the influence from different manifolds, many variants were proposed. Cazals et al. [12] and Guennebaud et al. [5] use quadrics and spheres in regression instead of planes. Nonetheless, spheres or quadrics are smooth surface, those methods are still unreliable near sharp features. Pauly et al. [13] assign Gaussian weights to the neighbors when estimating the local plane to weaken some points' influence to the regression. Niloy et al. [7] propose a way to adaptively change the size of neighborhoods, however anisotropic neighborhoods are still unavoidable for points on or very close to feature lines. All those methods perform a regression on a neighborhood centering in the point whose normal is being estimated. When this point is on edges and other sharp features, the influence from different manifolds is inevitable which leads to inaccurate normal estimation near sharp features.

Methods of the second category employ voting technique. Li et al. [8] propose a robust normal estimation method by using all the neighbor points vote to a set of planes determined by arbitrary three neighbor points to select the best tangent plane, which performs properly near sharp features. However, the parts of a neighborhood with higher density have more influence on the kernel density function because it does not take sampling non-uniformity into account. Boulch et al. [9] (HF) propose a uniform sampling technique to overcome the sampling non-uniformity and select the normal voted by the majority of local fitting planes. However, when the dihedral angle between the two planes forming the edge is large, the difference between normals produced by the triples sampled from the two planes is small. Then these normals are likely to vote for the same bin, and the normal will be blurred near the edge.

The third category of normal estimators segments the neighborhood to locate an isotropic sub-neighborhood. Fleishman et al. [14] segment the local neighborhood of a point into multiple outlier-free smooth regions. Using LRR, Zhang et al. [10] segment the whole neighborhood into different segments and one of them is chosen as for normal estimation. The most superior performance is obtained with the cost of much longer runtime. Liu et al. [11] improve [10] with a guided least squares representation to segment the neighborhood with high quality but less runtime. However, all these explicit neighborhood segmentation methods usually take higher computational cost and the estimated normals may still be unfaithful when the neighborhood is not large enough.

Similar idea of using shifted neighborhood has been applied in the domain of image [15] and mesh filter [16]. However, either image or mesh is a kind of structured representation, on which constructing and searching of shifted candidates are considerably much easier. And they are still likely to build an anisotropic neighborhood for just searching square patches or fixed n-ring faces when the expected neighborhood's shape is narrow or contains narrow parts nearby feature areas. A few outliers may not have major impact on filtering results, but it has great effect on the results of normal estimation. To conquer the challenges brought by complex feature areas and point clouds' structureless representation, we present two specific neighborhood shift operations to build faithful neighborhoods for points near different types of features.

3. Overview

Given a point cloud $P = \{\mathbf{p}_i\}$ as input, three steps are taken to estimate the normals. First, the K-nearest neighbors N_i are computed for each point p_i and p_i is classified into three types by analyzing N_i: non-feature points, edge points and corner points, which are detailed in Section 4. Then we use three strategies to construct an isotropic neighborhood of the current point respectively. For each non-feature point, we just take N_i as its isotropic neighborhood $\widetilde{N}_i$. For the rest feature points, specific strategies are designed to build a set of candidate neighborhoods for each point accommodating the characteristic of its type, which are explained in Sections 5.2 and 5.3 respectively. Section 5.1 presents an evaluation criterion to select one from the candidate set as the isotropic neighborhood $\widetilde{N}_i$ of a feature point. Finally, a more accurate normal is estimated by PCA using the constructed neighborhood $\widetilde{N}_i$ for each point respecting sharp features. The overview of our method is illustrated in Fig. 1 and concluded in Algorithm 1.

Algorithm 1 Pipeline of our method

```
Input: Point cloud $P$.
Output: Normal set $\{\mathbf{n}_i\}$
1:  for $\mathbf{p}_i \in P$ do
2:      Find $\mathbf{p}_i$'s K-nearest neighbors $N_i$ using kd-tree
3:      Compute $\sigma(N_i)$ by Eq. (1)
4:      if $\sigma(N_i) < \Omega_f$ then
5:          $\widetilde{N}_i = N_i$
6:      else if $\sigma(N_i) < \Omega_c$ then
7:          Find $\widetilde{N}_i$ via Direction Constrained Shift (Algorithm 5.2)
8:      else
9:          Find $\widetilde{N}_i$ via Self-adaptive Shift (Algorithm 5.3)
10:     Compute $\mathbf{n}_i$ via PCA on $\widetilde{N}_i$
```

4. Point classification

4.0.1. Feature points selection

We first distinguish feature points, that have a more complicated neighborhood, from non-feature points. Those feature points are further classified into different types: edge points and corner points. We will give a brief introduction of the feature detection method, and details are referred to in [10]. For each point $\mathbf{p_i}$, we compute a $\sigma(N_i)$ for its neighborhood N_i of size S:

$$\sigma(N_i) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{1}$$

where $\lambda_0 < \lambda_1 < \lambda_2$ are the singular values of the covariance matrix of N_i. The three singular values reflect the distribution of the N_i along three orthogonal singular vectors. Thus $\sigma(N_i)$ measures the confidence of how $\mathbf{p}_i$ is close to some feature regions. [10] computes a threshold Ω_f, and each point $\mathbf{p}_i$ with $\sigma(N_i)$ greater than the threshold will be viewed as a feature point. We denote the set of feature points as P_F.

4.0.2. Classification of feature points

The feature points P_F can be further classified into two categories: points near some edge between pairwise continued 2D manifolds P_E and points near some corner where several manifolds join together P_C. The corner point's neighborhood is more likely to have a larger surface variation. Thus, it is reliable to differ those two kinds of points via a threshold Ω_c:

$$\begin{cases} P_C = \{\mathbf{p}_i \in P_F | \sigma(N_i) > \Omega_c\} \\ P_E = \{\mathbf{p}_i \in P_F | \sigma(N_i) \leq \Omega_c\}. \end{cases} \tag{2}$$

Experientially, we choose $\Omega_c = 0.35$. Note that the classification is impossible to be 100% precise. However, the incorrect classification only increases the runtime of our method, and is unlikely to degenerate our result.

5. Neighborhood shift for feature points

For a feature point $\mathbf{p}_i$, its original neighborhood N_i might be sampled from different 2D manifolds. Hence it is unreliable to compute $\mathbf{p}_i$'s normal by applying PCA on N_i. But the normal can be represented by one of those manifolds. Instead using N_i or finding a sub-neighborhood with consistent normals by segmentation explicitly, we wish to locate an isotropic neighborhood $\widetilde{N}_i$, sampled from just one manifold, containing $\mathbf{p}_i$ but shifted away from $\mathbf{p}_i$, as illustrated in Fig. 2. However, it is not easy to construct such a neighborhood straightly on point clouds, since they are structureless. We actually construct a set of candidate neighborhoods $\mathcal{N}_i = \{N_{ij}\}$ all containing $\mathbf{p}_i$, and select one among them that is most likely to contain no features.

5.1. Criteria for neighborhood selection

To select a neighborhood from $\mathcal{N}_i$ that is least likely to contain features, we define $\gamma(N_{ij})$:

$$\gamma(N_{ij}) = \alpha_1 \sigma(N_{ij}) + \alpha_2 \|\mathbf{p}_i - \mathbf{p}_{ij}\|, \tag{3}$$

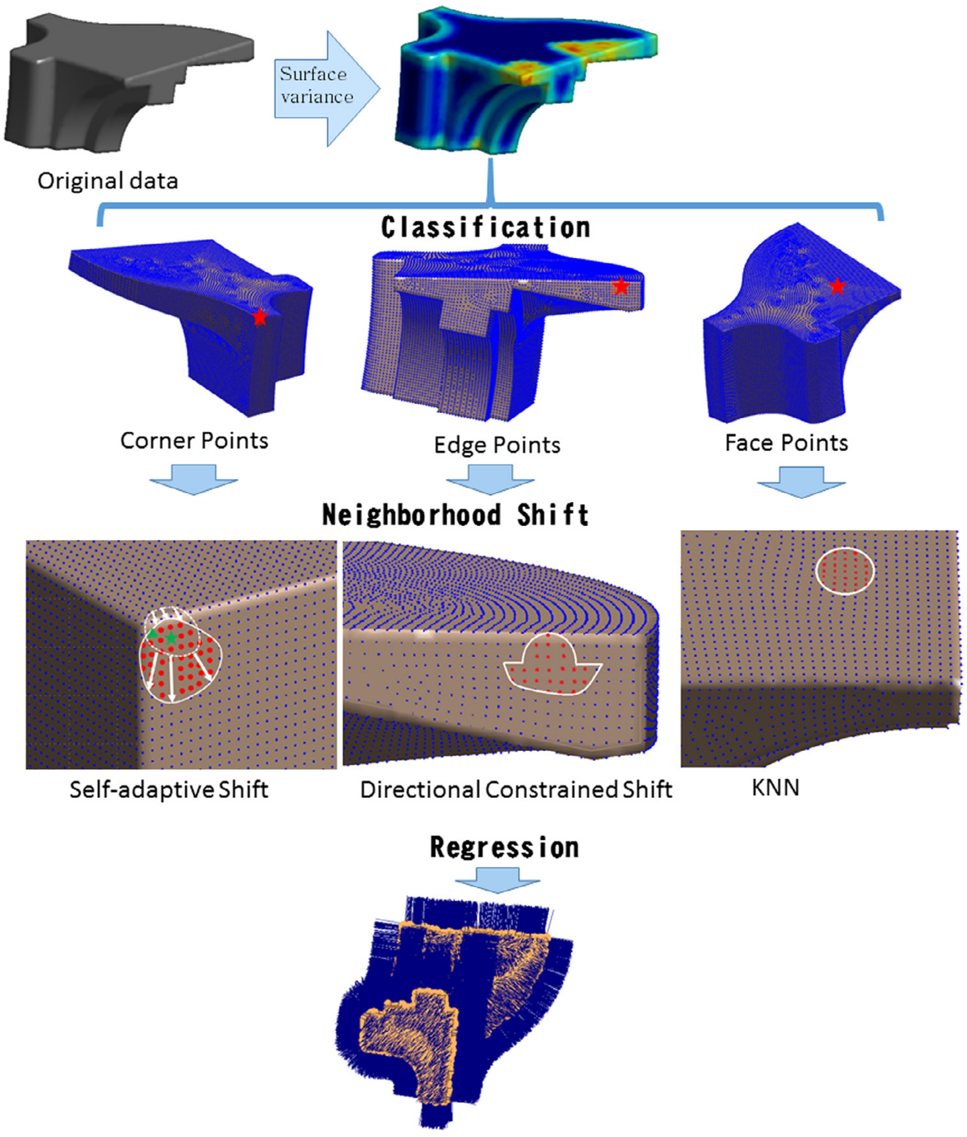


Fig. 1. Pipeline of our algorithm.

where $\sigma(N_{ij})$ is defined in Eq. (1), $\mathbf{p}_{ij}$ is the center point of N_{ij}, and α_1 and α_2 are given parameters, we set $\alpha_1 = 0.8$ and $\alpha 2 = 0.2$. The first item $\sigma(N_{ij})$ measures surface variation of N_{ij}, and we also think that the shifted neighborhood closer to p_i is more confident. The neighborhood with the minimal $\gamma(N_{ij})$ will be selected.

5.2. Direction constrained shift for edge points

For a feature point $\mathbf{p}_i \in P_E$, its candidate neighborhood set $\widetilde{N}_i$ could be all the neighborhoods centering at $\mathbf{p}_{ij} \in N_i$ respectively. To reduce the evaluation cost, we wish that only non-feature point $\mathbf{p}_{ij}$ is considered since its neighborhood is more likely to be clear of features. However if we use these neighborhoods directly, the selected best one may still contain points sampled from the other side of feature edges because of non-uniform sampling. To cope with this problem, for $\mathbf{p}_i$'s each neighbor $\mathbf{p}_{ij} \notin P_F$, we construct a candidate neighborhood N_{ij}^c by selecting points from $\mathbf{p}_{ij}$'s neighborhood N_{ij} of size S using a directional constraint which is introduced latter. If there is no such point in N_i, it indicates that $\mathbf{p}_i$ is located in a more complicated region, such as a narrow region with multiple feature lines. In this case, a small neighborhood of $\mathbf{p}_i$, with $S/3$ points is chosen as the final $\widetilde{N}_i$.

As shown in Fig. 3, given a non-feature neighbor $\mathbf{p}_{ij}$, we compute a vector $\mathbf{v}_{ij} = \mathbf{p}_i - \mathbf{p}_{ij}$. The candidate neighborhood N_{ij}^c is built by selecting $\mathbf{p}_{ij}$'s neighbor $\mathbf{p}_{ij}^k$ if the following condition is satisfied:

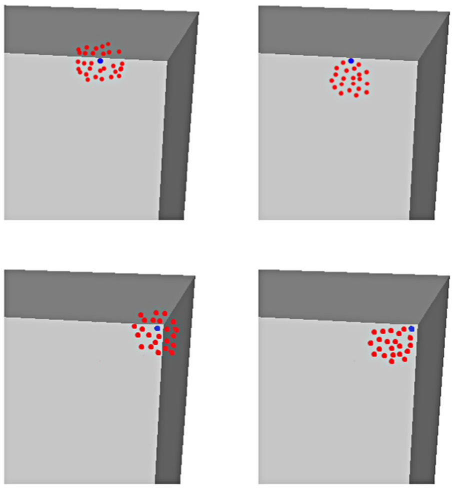


Fig. 2. The left column is the original neighborhoods, which contain points from two or three surface patches, of an edge point and a corner point, respectively. The right column shows the shifted neighborhoods of them. The shifted neighborhoods contain only points of the same surface patch, and more faithful normal estimation depends.

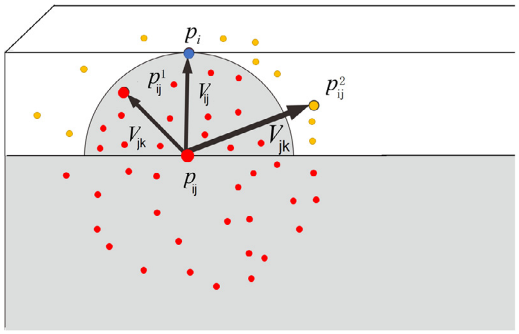


Fig. 3. The construction of a candidate neighborhood for an edge point $\mathbf{P}_i$. The feasible region is colored in gray, which is a combination of a half circle and a half Euclidean space. We plot $\mathbf{P}_{ij}$'s KNN. Those points in feasible region, colored in red, will be selected to compose a candidate neighborhood. The other points, colored in yellow will be discarded. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\mathbf{p}^k_{ij} \in N^c_{ij}, \quad \text{if} \begin{cases} \langle \mathbf{v}_{ij}, \mathbf{v}_{jk} \rangle \le 0 \\ \langle \mathbf{v}_{ij}, \mathbf{v}_{jk} \rangle > 0 \quad \text{and} \quad \|\mathbf{v}_{ij}\| > \|\mathbf{v}_{jk}\| \end{cases} \tag{4}$$

where $\mathbf{v}_{jk} = \mathbf{p}^k_{ij} - \mathbf{p}_{ij}$, and $\langle , \rangle$ is inner product in Euclidean space. Satisfying the directional constraint does not make sure that all the constructed N^c_{ij} will be isotropic, but some of them will, especially for the one with near right angles between v_{ij} and the tangential direction of the feature line. The procedure is listed in Algorithm 5.2.

5.3. Self-adaptive shift for corner points

For a point $\mathbf{p}_i \in P_C$, the best neighborhood selected via direction constrained shift may contain points from different surfaces, as illustrated in Fig. 4, since the corner is formed by the intersection of a few local surface patches and the patches are usually narrower or even far narrower than the patches formed on a feature line. To find a flat neighborhood in such a complex region, we develop a self-adaptive method.

Given a feature point $\mathbf{p}_i \in P_C$, a set of candidate neighborhoods $\mathcal{N}_i$ from seed points $\mathbf{p}_{ij} \in N_i$ is built. Instead of using the directional constraint, we iteratively grow a neighborhood from each $\mathbf{p}_{ij}$. We first initialize the candidate neighborhood N^c_{ij} with a small neighborhood N^o_{ij} of $\mathbf{p}_{ij}$ with size K_o containing $\mathbf{p}_i$, and look through each point of $\mathbf{N}_{ij}$ to decide whether to push it into N^c_{ij}. We wish that each pushed point makes N^c_{ij} flatter, *i.e.*makes γ in Eq. (3) smaller. If we decide to push one point into N^c_{ij}, its 5-nearest points are also pushed into N^c_{ij} since they stick close. Every time we decide to put in some points, we also choose one point $\mathbf{p}$ to delete. Deleting $\mathbf{p}$ makes N^c_{ij} flattest among deleting other points, and also we need $\mathbf{p} \notin N^o_{ij}$. N^o_{ij} is always kept in N^c_{ij}. This assures that N^c_{ij} represents the surface patch where $\mathbf{p}_i$ and $\mathbf{p}_{ij}$ lie through the neighborhood shift. Fig. 4 illustrates the original neighborhood N_i of a corner point p_i and the final shifted neighborhood $\widetilde{N}_i$.

Algorithm 2 Direction Constrained Shift

```
Input: $\mathbf{p}_i$, $N_{ij}$ for every $\mathbf{p}_{ij} \in N_i$
Output: $\mathbf{p}_i$'s shifted neighborhood $\widetilde{N}_i$
1: Let the candidate neighborhood set $\mathcal{N}_i$ be empty
2: Find all the non-feature points in $N_i$, denote them as $N_i'$
3: if $N_i' != \emptyset$ then
4:     for $\mathbf{p}_{ij} \in N_i'$ do
5:         $N_{ij}^c = \emptyset$
6:         for $\mathbf{p}_{ij}^k \in N_{ij}$ do
7:             if $\mathbf{p}_{ij}^k$ satisfies Eq. 4 then
8:                 $N_{ij}^c = N_{ij}^c \cup \{\mathbf{p}_{ij}^k\}$
9:         $\mathcal{N}_i = \mathcal{N}_i \cup \{N_{ij}^c\}$
10:    $\widetilde{N}_i = \underset{N_{ij}^c \in \mathcal{N}_i}{argmin}\, \gamma(N_{ij}^c)$
11: else
12:    $\widetilde{N}_i$ = A small neighborhood of $\mathbf{p_i}$ with S/3 points
13: return $\widetilde{N}_i$
```

Algorithm 3 Self-adaptive Shift

```
Input: $\mathbf{p}_i$, $N_{ij}$ for every $\mathbf{p}_{ij} \in N_i$
Output: $\mathbf{p}_i$'s shifted neighborhood $\widetilde{N}_i$
1: Let the candidate neighborhood set $\mathcal{N}_i$ be empty
2: for $\mathbf{p}_{ij} \in N_i$ do
3:     $N_{ij}^c = N_{ij}^o$
4:     $\mathbf{p} = \underset{\mathbf{p} \in N_{ij}/N_{ij}^o}{argmin}\, \gamma(N_{ij}^c \cup \{p\})$
5:     $N_{ij}^c = N_{ij}^c \cup \{\mathbf{p}$ and its 5 nearest points$\}$
6:     $\mathbf{p}' = \underset{\mathbf{p}' \in N_{ij}^c/N_{ij}^o}{argmin}\, \gamma(N_{ij}^c/\{p'\})$
7:     $N_{ij}^c = N_{ij}^c/\{\mathbf{p}'\}$
8:     $\mathcal{N}_i = \mathcal{N}_i \cup \{N_{ij}^c\}$
9: $\widetilde{N}_i = \underset{N_{ij}^c \in \mathcal{N}_i}{argmin}\, \gamma(N_{ij}^c)$
10: return $\widetilde{N}_i$
```

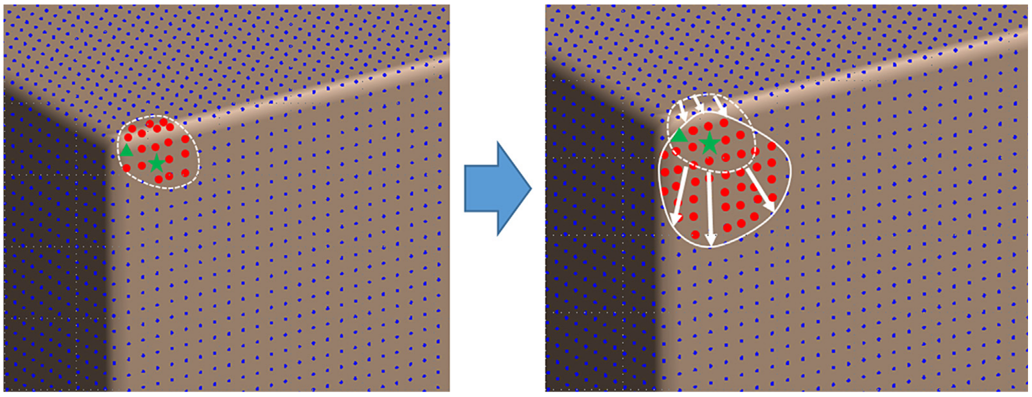
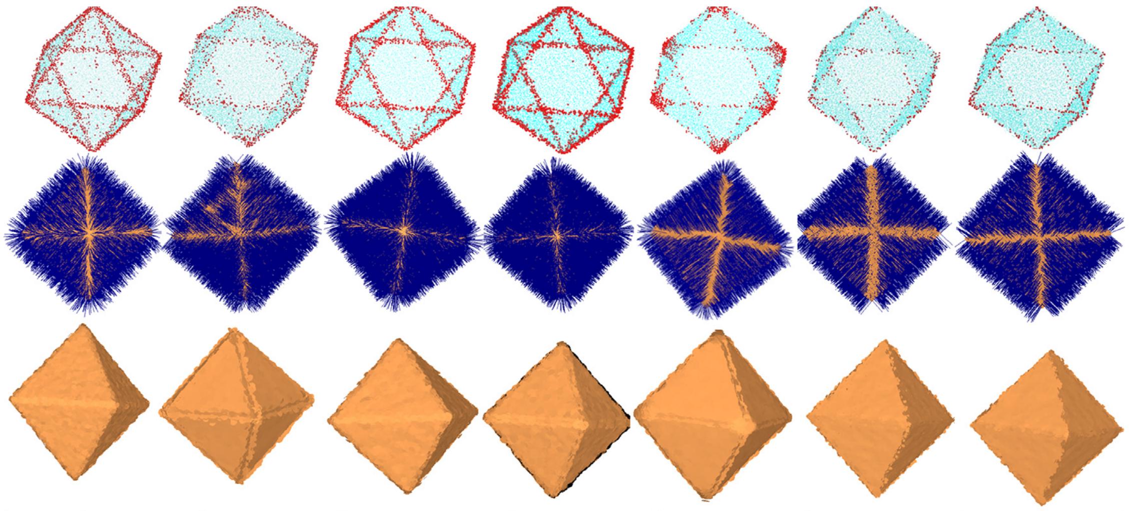


Fig. 4. The construction of a candidate neighborhood for a corner point. The current point $\mathbf{p_i}$ is shown in a green delta, and its original neighborhood is marked by the dotted circle. For its neighbor $\mathbf{p_{ij}}$, shown in a green star, our self-adaptive method starts from a small neighborhood of $\mathbf{p}_{ij}$, and grows gradually to build a candidate neighborhood clear of features. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



Method	PCA	RNE	HF_cubes	HF_points	HF_unif	LRR	Ours
Time	1.335	10.45	11.70	9.611	526.7	1718	47
NBP	2918	1042	1459	1710	1315	**477**	**517**
RMS	0.3477	0.1283	0.1737	0.1923	0.1607	**0.0878**	**0.0828**

Fig. 5. Comparison on Octahedron containing sharp edges with shallow angles. 50% noise is added. From left to right, they are the results of PCA, RNE, HF_cubes, HF_points, HF_unif, LRR and our algorithm. From top to bottom rows are the visualization of bad points colored in red, estimated normals, rendering using surfels and the result statistics of each algorithm, respectively. The result of our method is comparable with LRR but with far less running time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Experiment results

In this section, we make comparison between our method and some state-of-the-art methods: PCA [1], RNE [8], HF [9] and LRR [10], in view of sharp features, sampling anisotropy and noise. According to the sampling strategy, HF has three versions: HF_{points}, HF_{cubes} and HF_{unif}.

To compare those algorithms' performance in a quantitative way, the two criteria, the Root Mean Square measure (RMS_τ) [9] and Number of Bad Points (**NBP**) are evaluated. They are defined as:

$$RMS_\tau = \sqrt{\frac{1}{|P|}\sum_{\mathbf{p}\in P} f(\widehat{\mathbf{n_p}, \widetilde{\mathbf{n}}_\mathbf{p}})^2},$$

where

$$f(\mathbf{n_p}, \widetilde{\mathbf{n}}_\mathbf{p}) = \begin{cases} \widehat{\mathbf{n_p}\widetilde{\mathbf{n}}_\mathbf{p}}, & \text{if } \widehat{\mathbf{n_p}\widetilde{\mathbf{n}}_\mathbf{p}} < \tau \\ \pi/2, & \text{otherwise,} \end{cases}$$

$\mathbf{n_p}$ is the ground truth normal of $\mathbf{p}$, and $\widetilde{\mathbf{n}}_\mathbf{p}$ is the estimated normal of $\mathbf{p}$. $\widehat{\mathbf{n_p}\widetilde{\mathbf{n}}_\mathbf{p}}$ is the angle between $\mathbf{n_p}$ and $\widetilde{\mathbf{n}}_\mathbf{p}$. We set $\tau = 10^\circ$. Those points whose errors are more than 10° are considered as bad points, and **NBP** are number of the bad points.

All the noise we use in our experiments are Gaussian noise, with different standard deviation as % of the mean distance between points. Empirically, we choose the parameters for our algorithm, we let $K_o = 60, S = 100$. Other parameters were set to default if they are not mentioned.

Note that our method applied in this paper is in MATLAB version, which has a negative effect in our algorithm. On the other hand, this means our method has more potential in efficiency.

6.1. Comparison on feature preservation

Comparison on Octahedron containing sharp edges with shallow angles. In Fig. 5, we compare all the methods on the 20k Octahedron model with 50% noise. The sharp edges of it are generated by the intersection of two planes with shallow angles. We can see that PCA overly smooths the normals near sharp edges. Thus there are many mistakes around the edges. RNE, HF_points and HF_cubes blur the edges in less degree but still are inferior to LRR and our method. HF_unif performs properly on edge feature, but fails in corners. The reason of the inferiority of HF for such case is that normals produced by triples sampled from different sides of a feature line are likely to vote for the same bin when the dihedral angle is shallow. Uncorrected normals by LRR and our algorithm distribute very close to edges and many of them are unavoidable since

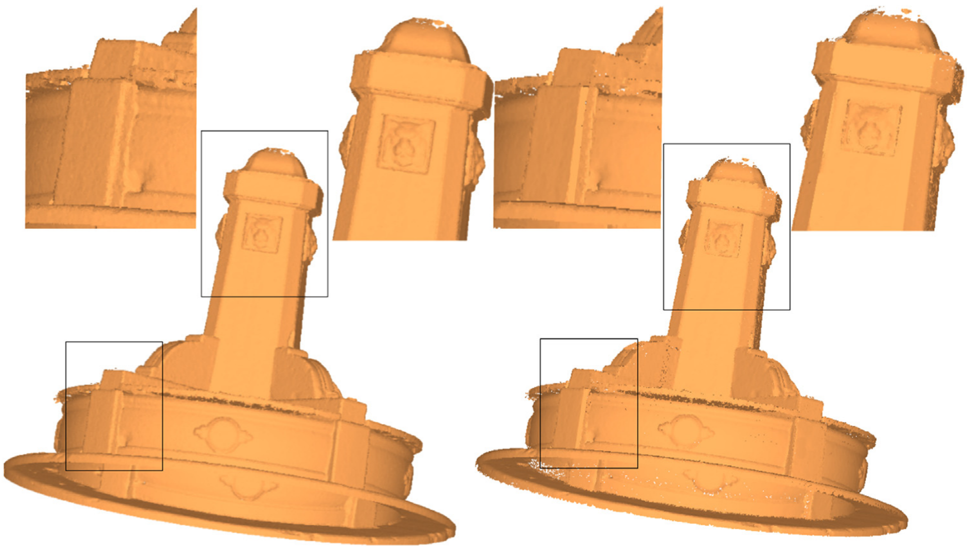
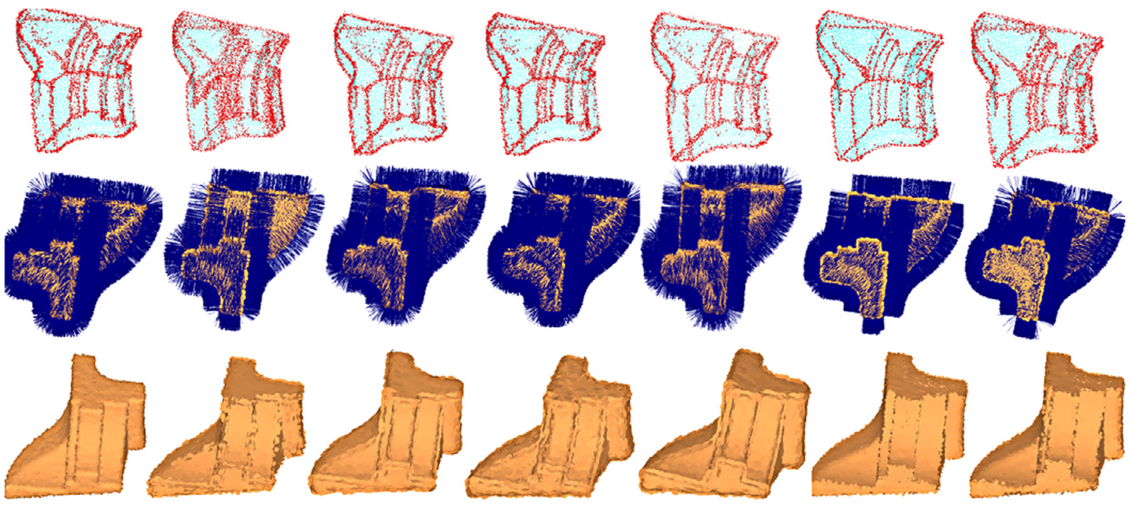


Fig. 6. Comparison on real scanned data. The result of PCA is shown on left side, and the result of ours is shown on right side.

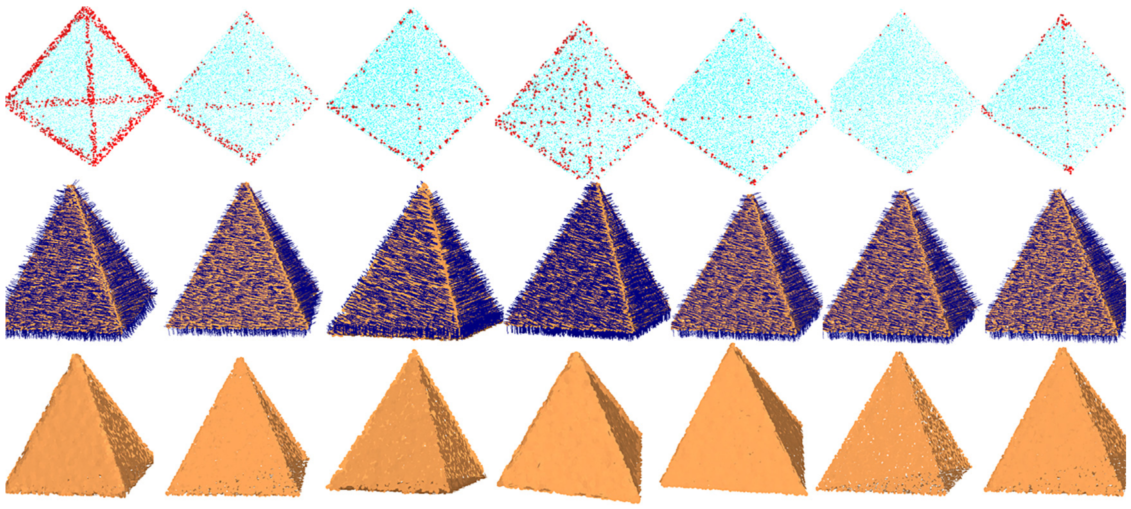


Method	PCA	RNE	HF_cubes	HF_points	HF_unif	LRR	Ours
Time	1.335	19.635	16.70	11.90	526.7	3362	77
NBP	5490	1042	4406	6123	4292	**2181**	**3138**
RMS	0.4111	0.3906	0.3435	0.4923	0.3360	**0.1807**	**0.2342**

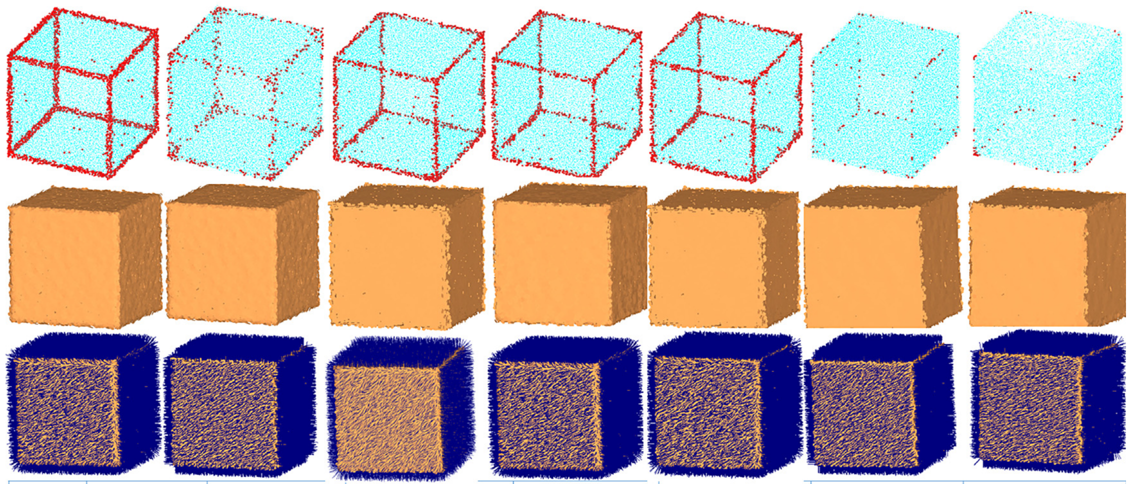
Fig. 7. Comparison on Fandisk containing complex neighborhood structure. 50% noise is added. From left to right, they are the results of PCA, RNE, HF_cubes, HF_points, HF_unif, LRR and our algorithm. From top to bottom rows are the visualization of bad points colored in red, estimated normals, rendering using surfels and the result statistics of each algorithm, respectively. The result of our method is comparable with LRR but with far less running time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

some points on one side of a feature line may turn into another side of the edge after adding noise by disturbing the points' position. On this model, our algorithm performs nearly as good as LRR's, which is tens of times slower than ours. This also demonstrates that our algorithm achieved the same effect of neighborhood segmentation with a much lower cost. The candidate neighborhoods contain at least one correct segmentation and we just need to find it.

Comparison on Fandisk containing complex neighborhood structure. Many points' neighborhood of the Fandisk model may contain multiple feature lines, such as the points in the marked narrow-band regions in Fig. 7. The complex neighborhood structure challenges previous normal estimators. Thus we test the performance of those algorithms over neighborhoods with multi-features on the Fandisk model with 26k points and 50% noise. Fig. 7 shows that PCA, RNE, HF_cubes, HF_points and HF_unif generate much bad points around the circled narrow bending band region. Our algorithm and LRR perform better on the region, and most bad points are distributed along the edges. We can also see, from the second



Method	PCA	RNE	HF_cubes	HF_points	HF_unif	LRR	Ours
Time	2.112	10.45	5.701	3.871	247.8	1317.9	47.3
NBP	2531	244	157	539	116	**24**	**55**
RMS	0.3611	0.0532	0.0611	0.1489	0.0536	**0.025**	**0.0529**



Method	PCA	RNE	HF_cubes	HF_points	HF_unif	LRR	Ours
Time	2.026	19.207	16.23	11.58	511.4	2279	44.81
NBP	2779	611	1526	1789	1403	**111**	**115**
RMS	0.2343	0.0588	0.1494	0.1924	0.1534	**0.035**	**0.0141**

Fig. 8. Our method is robust to anisotropic sampling. For tetrahedron, numbers of points on its four surface are at a ratio of 1:2:3:4. For cube, it is 1:2:6 for right, up and front surface, and surfaces on the opposite position have same number of points. 50% noise are add to both of the models.

row, that although other algorithms overly smooth the normals around features, our algorithm remains the discontinuity of normals properly, almost as good as LRR. This illustrates that our neighborhood shift operations can handle complex neighborhood. Considering the computational cost, our algorithm is pretty promising.

Comparison on real raw data. To illustrate the capability of our algorithm to handle the real data, we run our method on a scanned point data and present the results in Fig. 6. The data suffers certain scale of noises and outliers from the procedure of scanning. Compared with PCA, we see that while the smooth regions are rendered pleasingly using our normals, the features are also preserved favorably.

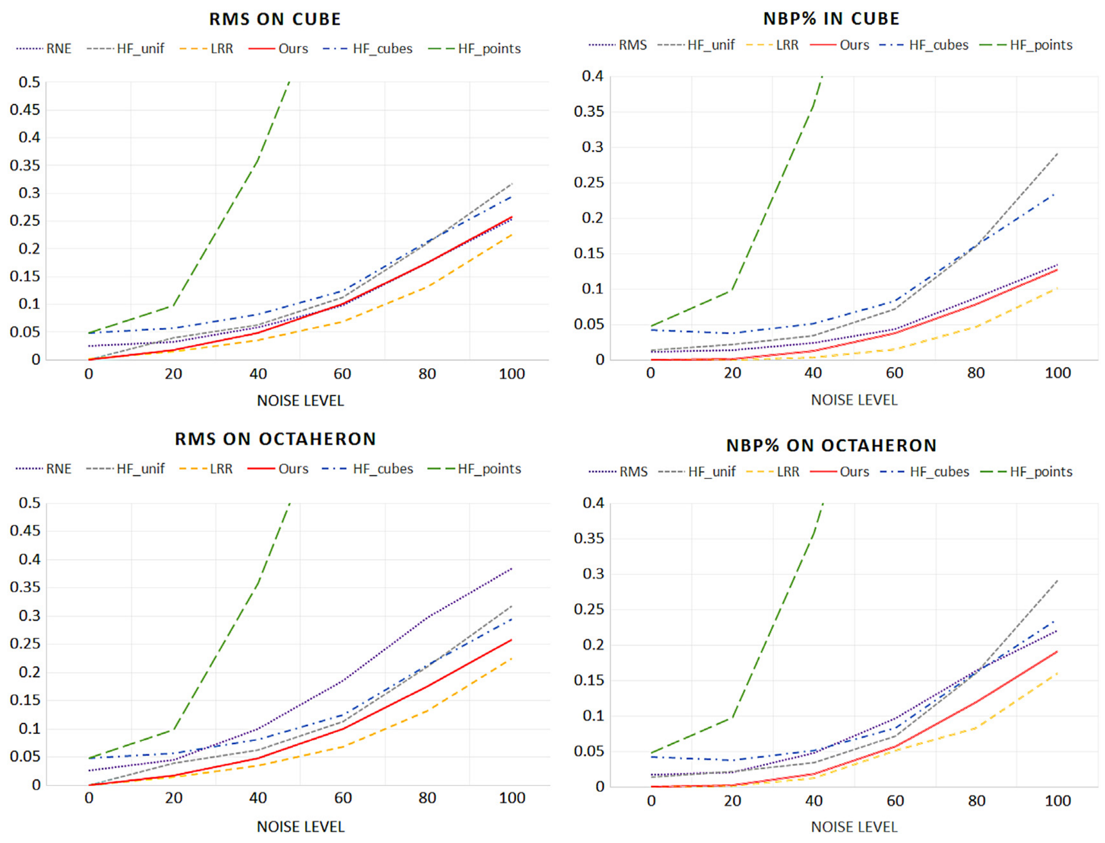


Fig. 9. Our method is robust to noise. RMS_τ and NBP of RNE, HF_cubes, HF_points, HF_unif, LRR and our method on Cube and Octahedron at different noise level are shown in the top and bottom rows, respectively.

6.2. Comparison on robustness to noise and sampling density

To evaluate the robustness of our method to noise, we apply RNE, HF_cubes, HF_points, HF_unif, LRR and our method on simple geometric models including Cube and Octahedron with varying Gaussian noise levels. In Fig. 9, we list the results. Although the performance of all the methods drops with the increase of the noise level, our algorithm is only inferior to LRR.

Fig. 8 shows that our method is robust to non-uniform sampling. In this experiment, a non-uniformly sampled cube model with 21k points and a tetrahedron with 10k points are tested and each of them has 50% of noise. In view of RMS_τ and NBP, our method is again only inferior to LRR and superior than other methods. The values of RMS_τ and NBP for the two models are reported in the bottom row of the figure. The experiment shows that our self-adaptive neighborhood shift algorithm can find a neighborhood without crossing features even when the point cloud is non-uniformly sampled.

7. Conclusion

In this paper, we present a feature-preserving normal estimation method via neighborhood shift. Two specific neighborhood shift techniques are introduced considering the characteristic of sharp features and point clouds. The shifted neighborhood does not necessarily center at the point for which we are estimating the normal. But it tends to be isotropic, *i.e.*clear of sharp features, and then a quality normal can be estimated on it. The experiments exhibit that faithful normals are estimated with relatively low computational cost comparing with the existing state-of-the-art methods.

One limitation of the method is that evaluating all candidate neighborhoods of a point with very large neighborhood may be impractical. Hence the method may be not robust to large noise and some extreme cases when only small neighborhoods are explored. To overcome the problem, we would like to integrate some random strategy to improve the neighborhood shift operations in future.

Acknowledgments

The authors would like to thank all the reviewers for their valuable comments. Thanks to Bao Li and Alexandre Boulch for providing the code used for comparison. Junjie Cao is supported by the NSFC Fund (Nos. 61363048 and 61572099) and the Fundamental Research Funds for the Central Universities (No. DUT16QY02). Jie Zhang is supported by the Foundation of Liaoning Educational Committee (No. L201683663). Xiuping Liu is supported by the NSFC Fund (No. 61370143). Changqing Zou is supported by the Project supported by the Doctorial Start-up Fund of Hengyang Normal University, China (No. 15B22).

References

[1] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface Reconstruction from Unorganized Points, Vol. 26, ACM, 1992.
[2] M. Pauly, R. Keiser, M. Gross, Multi-scale feature extraction on point-sampled surfaces, in: Computer Graphics Forum, Vol. 22, Wiley Online Library, 2003, pp. 281–289.
[3] C. Lange, K. Polthier, Anisotropic smoothing of point sets, Comput. Aided Geom. Design 22 (7) (2005) 680–692.
[4] S. Rusinkiewicz, M. Levoy, Qsplat: A multiresolution point rendering system for large meshes, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 343–352.
[5] G. Guennebaud, M. Gross, Algebraic point set surfaces, in: ACM Transactions on Graphics (TOG), Vol. 26, ACM, 2007, p. 23.
[6] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, Comput. Aided Geom. Design 22 (2) (2005) 121–146.
[7] N.J. Mitra, A. Nguyen, Estimating surface normals in noisy point cloud data, in: Proceedings of the Nineteenth Annual Symposium on Computational Geometry, ACM, 2003, pp. 322–328.
[8] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, S. Jin, Robust normal estimation for point clouds with sharp features, Comput. Graph. 34 (2) (2010) 94–106.
[9] A. Boulch, R. Marlet, Fast and robust normal estimation for point clouds with sharp features, in: Computer Graphics Forum, Vol. 31, Wiley Online Library, 2012, pp. 1765–1774.
[10] J. Zhang, J. Cao, X. Liu, J. Wang, J. Liu, X. Shi, Point cloud normal estimation via low-rank subspace clustering, Comput. Graph. 37 (6) (2013) 697–706.
[11] X. Liu, J. Zhang, J. Cao, B. Li, L. Liu, Quality Point Cloud Normal Estimation by Guided Least Squares Representation, Vol. 51, 2015, pp. 106–116.
[12] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, in: Eurographics/acm SIGGRAPH Symposium on Geometry Processing, 2003, pp. 177–187.
[13] M. Pauly, R. Keiser, L.P. Kobbelt, M. Gross, Shape modeling with point-sampled geometry, ACM Trans. Graph. 22 (3) (2010) 641–650.
[14] S. Fleishman, D. Cohen-Or, C. Silva, T. Udio, Robust moving least-squares fitting with sharp features, ACM Trans. Graph. 24 (3) (2005) 544–552.
[15] H. Cho, H. Lee, H. Kang, S. Lee, Bilateral texture filtering, ACM Trans. Graph. 33 (4) (2014) 128.
[16] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, L. Liu, Guided mesh normal filtering, Comput. Graph. Forum 34 (2015) 23–34. (Special Issue of Pacific Graphics 2015).