International Journal of Precision Engineering and Manufacturing (2019) 20:915–925
https://doi.org/10.1007/s12541-019-00113-0

Online ISSN 2005-4602
Print ISSN 2234-7593

REGULAR PAPER

# A Fast Voxel-Based Method for Outlier Removal in Laser Measurement

**Hao Chen[1] · Yu Chen[2] · Xu Zhang[3] · Baiyuan Li[4] · Xiaoqiang Liu[4] · Xuefei Shi[5] · Jie Shen[6]**

Received: 13 August 2017 / Revised: 13 March 2019 / Accepted: 24 March 2019 / Published online: 12 April 2019


**Abstract**
Discrete data points are noncontinuous without structural information. In this paper, we propose a new fast outlier removal method via voxel-based surface propagation. The main technical components of our approach include (a) an efficient and simple spatial partitioning scheme and (b) a specially-designed surface propagation method. Numerical analyses indicate that our method is about 10 times faster than an existing method and significantly better than other two methods in terms of denoising accuracy. This provides an efficient solution to handling noisy laser-scanning data.

**Keywords** Surface propagation · Laser scanning · Data outlier · Discrete data point

**List of Symbols**

| | |
|---|---|
| $(\lambda_1 \geq \lambda_2 \geq \lambda_3)$ | Three eigenvalues |
| $\lambda_{max}$ | Maximum eigenvalue |
| $\lambda_{min}$ | Minimum eigenvalue |
| $\theta$ | Angle between two vectors |
| $i, j, k$ | The index in directions $x, y, z$, respectively |
| $n_x, n_y, n_z$ | The number voxels in directions $x, y, z$, respectively |
| $\boldsymbol{n_v}$ | Total number of voxels in each problem |
| $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ | Three eigenvectors |
| $Voxel_i$ | The set of data points in the current voxel, $i$, $i = 1, \boldsymbol{n_v}$ |

✉ Jie Shen
shen@umich.edu

[1] College of Automotive Engineering, Shanghai University of Engineering Science, Shongjiang, Shanghai 201620, China

[2] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

[3] College of Mechanical Engineering, Shanghai University of Engineering Science, Shongjiang, Shanghai 201620, China

[4] College of Computer and Information Science, Donghua University, Changning Qu, Shanghai 200336, China

[5] School of Automation and Electrical Engineering, Beijing Science and Technology University, Beijing 100083, China

[6] Department of Computer Science, University of Michigan-Dearborn, Dearborn, MI 48128, USA

## 1 Introduction

Discrete data points are one common data type in engineering and can be obtained via laser scanners and LIDAR (Light Detection and Ranging) sensors. Outlier removal is an important problem in different disciplines. Related techniques have been widely used in a variety of fields: reverse engineering, rapid prototyping, biomedicine, architecture, entertainment industry, financial data analysis, etc.

Many studies have been conducted to remove outliers effectively and efficiently. However, most of these methods perform well only in specific situations. Some methods [1, 2] are more suited to handling isolated outliers, and others [3] are good in dealing with surfaces without sharp features. Xie et al. [2] used an active contour method to cluster mono-oriented groups for finding outlier clusters. This method is suited to highly-discrete outliers. Kolluri et al. [4] developed a density method and a plane fitting method for removing outliers. The implementations of these methods are quite simple.

Shen et al. [5] proposed a surface propagation method combined with minimal variance and normalized histogram. This method performs well in both non-isolated outlier clusters and sharp edges. But, Shen's method encounters an efficiency issue when dealing with large data models. Wang and Feng [6] further advanced Shen's method by introducing a new method: majority voting.

Zaman et al. [7] developed a density-based denoising algorithm, in which particle-swarm optimization is used for approximating optimal bandwidth of multivariate kernel density and a mean-shift based clustering technique is

utilized to remove outliers. For a complete review in this field, see a recent review paper [8].

There are also some studies [9–14] related to feature extraction and reconstruction from 3D data based on the edge detection, segmentation, or shading information of unorganized point clouds. But, these methods are only remotely related to the study in this paper. It is difficult to make a comparison because of difference in research targets and measurement environment. Other remotely related laser applications include distance and quality measurement [15–17].

According to our analysis [8], non-isolated outlier clusters and sharp featured outlier clusters are two types of the most difficult outlier clusters. As for non-isolated outlier clusters, outlier clusters are so close to a main surface that distance-based criteria are not effective to remove outliers. With respect to sharp featured outlier clusters, it is difficult to preserve these sharp features in a data model because geometric non-smoothness at these features invalidates many analysis arsenals in calculus and differential geometry. In some cases such as laser scanning data, we might have to handle surfaces with sharp features and non-isolated outlier clusters. This becomes the most difficult type of models.

Existing methods for handling non-isolated outlier clusters require a considerable amount of processing time and are therefore not suited to real-time or near real-time applications. The main objective of this paper is to propose a fast algorithm to remove different types of clusters (especially the most difficult types of outlier clusters) in an accurate and efficient way.

The rest of this paper is organized as follows. In Sect. 2, a new scheme for removing outlier clusters is presented. Experimental results and discussions are provided in Sect. 3, followed by some concluding remarks in Sect. 4.

## 2 A Fast Scheme of Removing Outlier Clusters

### 2.1 Principle of Locality

The domain of a point cloud model is subdivided into a limited number of voxels (volume elements) through a uniform partition in a 3D space. We use a voxel as the minimal unit of analysis and display. Specifically, all the numerical analyses are aimed at each single voxel and all the data points in a voxel are treated simultaneously as an outlier or true data point. This reflects the *Principle of Locality*: data points in a small voxel are highly likely to have similar properties for the tasks of surface outlier removal.

Spatial partition is a geometric process that divides a 3D geometric domain into two or more disjoint sub-volumes while the union of these sub-volumes equals the domain. Various advanced data structures, such as binary space partitioning (BSP) trees [18], quadtrees/octrees [19], k-dimensional (k-d) trees [20] and R-trees [21], have been developed for efficient storage and query of data points. But, a significant number of accesses to the above data structures are needed in the context of denoising the non-isolated outlier clusters. In this paper, a uniform partition is used to reduce the computational time associated with the above data structures.

The uniform partition means that all the sub-volumes have the same sizes in three dimensions ($x$, $y$ and $z$), respectively. A bounding volume is first created to enclose the problem domain. Next, the bounding volume is uniformly subdivided into $\boldsymbol{n_x} \times \boldsymbol{n_y} \times \boldsymbol{n_z}$ sub-volumes. Therefore, each sub-volume can be indexed by a triplet ($i$, $j$, $k$), where $\boldsymbol{i} \in [0, \boldsymbol{n_x}]$, $\boldsymbol{j} \in [0, \boldsymbol{n_y}]$, $\boldsymbol{k} \in [0, \boldsymbol{n_z}]$. One or more data points may be contained in a sub-volume, while some sub-volumes near the boundaries of the bounding volume may contain no point at all. The values of $\boldsymbol{n_x} \times \boldsymbol{n_y} \times \boldsymbol{n_z}$ are problem dependent. If the values are too big, the computational time of denoising will be long. On the other hand, if the values are too small, the denoising accuracy will be compromised.

Because of the uniform partition, the link between indices ($i$, $j$, $k$) of each sub-volume and the coordinates ($x$, $y$, $z$) of each data point becomes extremely easy to implement. One pass of preprocessing can be used to link each sub-volume to a list of data points. The locality of data points is realized at the level of sub-volumes. In such a way, subsequent queries of nearest neighbors become unnecessary, and therefore a considerable amount of computational time is saved.

### 2.2 Fast Surface Propagation

The basic idea of surface propagation is to connect data points in different sub-volumes through geometric coherence. The propagation is based on voxel and starts from an initial voxel. Then, it searches the neighboring ring of the current voxel, select the voxels that are considered to be inside the main surface, and mark points in those voxels as true data according to driving force for propagation. Herein, the driving force is defined as a mechanism to guide the surface propagation. Other voxels in the ring are temporarily marked as outliers. For each voxel that is inside the main surface, the program continues to search its neighboring voxels. The propagation terminates when there is no new voxel to visit. After the propagation, the program treats each data point in the voxels, which are marked as outliers or untouched (initial status), as outlier; each data point in the voxels, which are marked as true data, is considered as a true data point.

Because some voxels may be visited several times, a principle to label voxels is needed in the propagation. In this paper, we finally mark the voxel as true data as long as it has been treated as true data once. But, with respect to those voxels that are treated as true data after being treated as outliers, they are not included into the next iteration of propagation.

#### 2.2.1 Driving Force of Propagation

Driving force is crucial for a surface propagation. It is used to determine whether a voxel should be included into the main surface. The key criteria to select the driving force for the surface propagation include

(a) The driving force should be direct/indirect measurements or certain statistical properties,
(b) The driving force should imply the levels of closeness between voxels,
(c) The driving force should reflect the difference between true data and outliers.

We define two kinds of driving forces in our method: distance and angle. As for distance, it refers to the Euclidean distance between the centroid of data points in a neighboring voxel and the plane determined by the two largest eigenvectors of the current voxel, as shown in Fig. 1a. In this figure, point *A* represents the centroid of all data points in the current voxel, while points *B* and *C* are the centroids of data points in two adjacent voxels, respectively. The tangential plane passes through point *A* and is determined by a principal component analysis [22] of all the data points in the current voxel. The covariance matrix of all data points in the current voxel is

$$\boldsymbol{CV} = \sum_{\boldsymbol{q} \in Voxel_i} (\boldsymbol{q} - \boldsymbol{c}_i) \otimes (\boldsymbol{q} - \boldsymbol{c}_i), \tag{1}$$

where $Voxel_i$ is the set of data points in the current voxel, $i$, and $\mathbf{c}_i$ is the centroid of data points in $Voxel_i$. $\otimes$ is an outer product operator of vectors. A Jacobi transformation [23] is used to determine eigenvectors $(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$ and eigenvalues $(\lambda_1 \geq \lambda_2 \geq \lambda_3)$ of the CV. $\mathbf{v}_3$ represents the normal direction of the tangent plane, $\mathbf{v}_1$ and $\mathbf{v}_2$ are the base vectors of orthogonal parameter coordinates in the tangent plane.

It is a priori assumption that a smaller distance to the tangential plane means a larger probability and this probability defines a chance with which the neighboring voxel is inside the main surface. In other words, outliers have larger distances than true data points.

With respect to angle, it represents an angle between two planes and this angle is formed between the smallest eigenvectors (i.e., the vectors of surface normal) of the current and neighboring voxels, respectively. In Fig. 1b, $\mathbf{v}_{3i}$ and $\mathbf{v}_{3j}$ are the surface normal vectors of the tangential planes that pass point *A* and *B*, respectively. These two points are the centroids of data points in $Voxel_i$ and $Voxel_j$. The angle $\theta$ between these two vectors is computed by the dot product of the vectors:

$$\theta = \arccos\left(\frac{\mathrm{v}_{3i} \cdot \mathrm{v}_{3j}}{\left|\mathrm{v}_{3i}\right|\left|\mathrm{v}_{3j}\right|}\right), \tag{2}$$

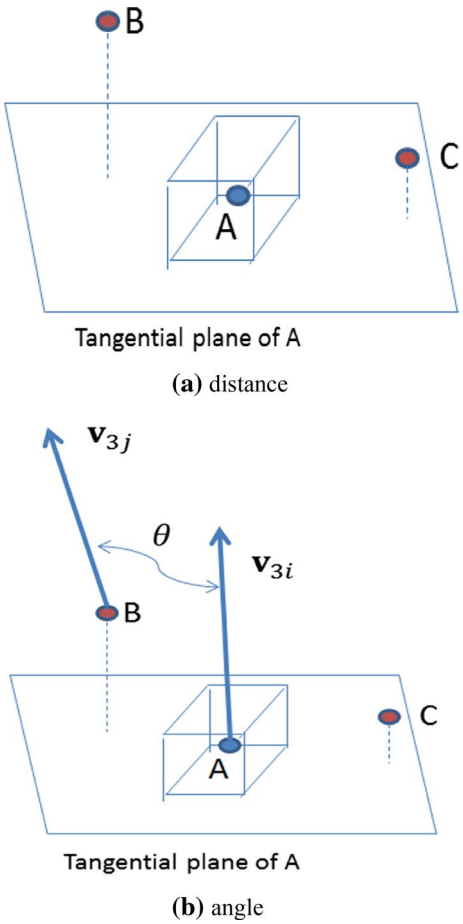



**Fig. 1** Two driving forces for surface propagation

Note that each single driving force mentioned above is not effective enough to handle all cases. For example, in a scenario of removing non-isolated surface outliers, which is regarded as the most difficult case, outliers are often quite close to the main surface. Thus, the distance approach is likely to fail. In some other scenarios, the angle approach is not effective enough to differentiate true data from outliers. A comprehensive way is to combine these two criteria so that we can handle different cases.

#### 2.2.2 Initial Voxel

The initial voxel is another crucial factor for surface propagation. If an incorrect initial voxel is selected, the propagation cannot help us remove outliers. One key principle to select the initial voxel of surface propagation is to find the voxel that is most likely to be inside the main surface. In this paper, we utilize a histogram of the smallest eigenvalue $\lambda_3$ of voxels to find the initial voxel, as shown in Fig. 2. In the figure, the horizontal axis represents the bins of the smallest eigenvalues, while the vertical axis refers to the count of voxels in each bin. The number of bins, $m$,

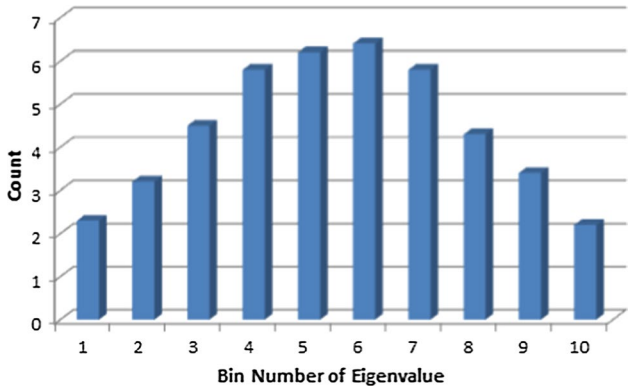



**Fig. 2** A histogram of the smallest eigenvalue of voxels


**Fig. 3** A general flowchart of our fast voxel-based propagation for outlier removal

is a finite integer between 20 and 40. The bin number, $n_i$, is computed by

$$n_i = ceil\left(\frac{m(\lambda_{max} - \lambda_{3i})}{(\lambda_{max} - \lambda_{min})}\right), \tag{3}$$

where ***ceil***() represents a ceiling function to convert a real number to an integer. $\lambda_{max} = \max(\lambda_{3i} | i \in [1, n_v])$ and $\lambda_{min} = \min(\lambda_{3i} | i \in [1, n_v])$, where $n_v$ is the total number of voxels in each problem. $\lambda_{3i}$ refers to the third eigenvalue of the $i$-th voxel.

An initial voxel should be selected from the leftmost bin where the values of $\lambda_3$ are among the lowest.

### 2.2.3 Algorithm

A general flowchart of our fast voxel-based propagation (FVBSP) for outlier removal is illustrated in Fig. 3. It consists of three main technical components: connectivity, eigenvalue, and core propagation. The detailed algorithm of FVBSP is provided in Fig. 4, while two important routines are outlined in Figs. 5 and 6. The eigenvalues of a covariance matrix are computed on the basis of codes in [6].

In Fig. 3, the first component refers to the connectivity between adjacent voxels in a problem domain. With respect to the current voxel, there are six face-connected neighboring voxels and 12 edge-connected adjacent voxels. A surface propagation traverses in the space starting from the initial voxel and traveling to the neighboring voxels. The propagation terminates when non-visited voxels are exhausted.

```
1. VoxelSet = GetVoxelSet(data_points);
2. InitialVoxel = GetInitialVoxel(VoxelSet);
3. Add InitiaVoxel in MainSurface;
4. Loop:
5.     foreach current_voxel in MainSurface:
6.         if current_voxel is marked as untouched or unprocessed:
7.             Mark current_voxel as true_data;
8.         foreach neighboring_voxel in GetNeighboringVoxel(current_voxel):
9.             if DrivingForceJudgement(neighboring_voxel, current_voxel)==true:
10.                if neighboring_voxel is marked as untouched:
11.                    Add neighboring_voxel in FrontWave;
12.                    Mark neighboring_voxel as unprocessed;
13.                else if neighboring_voxel is marked as unprocessed or true_data:
14.                    No action;
15.                else:
16.                    Mark neighboring_voxel as true_data, depending on cases;
17.            else:
18.                if neighboring_voxel is marked as untouched:
19.                    Mark neighboring_voxel as outlier;
20.     if number of voxel in FrontWave is zero:
21.         Jump out of the loop;
22.     Copy FrontWave to MainSurface;
23. All data points in voxel marked as true_data are true data;
24. All data points in voxel marked as outliers or untouched are outliers;
```

**Fig. 4** Algorithm of fast voxel-based surface propagation

```
Name: GetInitialVoxel Input: Voxel Set Output: initial voxel
1. foreach voxel in Voxel Set:
2.     if GetConnectivityFlag(voxel) is true:
3.         error = CalcSmallestEgienValue(voxel);
4.         hlevel = Bin(error);
5.     else:
6.         hlevel = MaxBinNum – 1;
7.     Select a single voxel with the smallest hlevel as an initial voxel;
```

**Fig. 5** Routine of GetInitialVoxel Input

The second component in Fig. 3 is used to compute the eigenvalues and eigenvectors of each voxel. The eigenvalues are used to construct a histogram of the third eigenvalue, from which an initial voxel for the surface propagation can be selected. The eigenvectors are used to calculate the angle-based driving force in Eq. (3) for the propagation.

The third component is the surface propagation, which is used for labelling all the voxels as well as all the data points within each voxel. Depending upon the surface properties in the problem domain, if geometric discontinuity exists, several passes of propagation may be needed to process all the voxels

```
Name: DrivingFroceJudgement
Input: neighboring_voxel & current_voxel
1.  Calculate eigenvectors for the two voxels;
2.  Calculate the two planes for the two voxel;
3.  Calculate the distance between the centroid of data points
    in neighboring_voxel and plane in current_voxel;
4.  Calculate the angle between the two planes;
5.  DrivingForce = ratio*Normalized(distance) + (1-ratio)*Normalized(angle);
6.  if DrivingForce < Threshold:
7.      return true;
8.  else:
9.      return false;
```

**Fig. 6** Routine for computing the driving force

in the problem. In such cases, a different initial voxel is selected for each pass based on the histogram information and the status of those voxels that have not been visited yet. Starting from the initial voxel, the surface propagation progresses like a wave front. All the voxels located at this wave front are stored in a dynamic array or a linked list. The propagation terminates when this wave front becomes empty. At the beginning, the wave front contains only one element, i.e., the initial voxel.

Figure 5 shows the implementation for selecting an initial voxel. Function *GetConnectivityFlag*(voxel) is used to handle the cases where geometric discontinuity exists in a problem domain. By using the connectivity flag, our approach is capable of processing several disjoint geometric objects one at a time. Function *Bin*(error) converts an eigenvalue to an integer: bin number for the histogram.

The detailed implementation of driving forces for surface propagation is described in Fig. 6. The overall driving force is computed by

$$\begin{aligned} driving\ force =& s\, Normalized(distance) \\ &+ (1-s)\, Normalized(angle), \end{aligned} \tag{4}$$

where $s \in [0, 1]$ is a weighting factor for the linear combination of distance and angle. *Normalized*() is a function to normalize the angle or distance. If the driving force in Eq. (4) is less than a threshold, the surface propagation moves from the current voxel to the neighboring voxel.

The initial voxel is another crucial factor for the surface propagation. Its selection is given in Sect. 2.2.2.

## 3 Numerical Tests and Discussions

Numerical tests were conducted with discrete data points from a laser scanner. The algorithms designed in this paper were implemented in Visual Studio and tested on an Asus G752VT laptop with 2.6 GHz Intel CPUs (4 cores

**Table 1** Parameter setting of Eigenvalue method

| Method\parameters | ***n*** (# of voxels in each dimension) | HISTOGNUM (# of bins) | Threshold of Histogram |
|---|---|---|---|
| Eigen value method | 100 (70 for data model 2) | 30 | 15 |

**Table 2** Parameter setting of Density method

| Method\parameters | ***n*** (# of voxels in each dimension) | HISTOGNUM (# of bins) | Threshold of Histogram |
|---|---|---|---|
| Density value method | 100 (70 for data model 2) | 30 | 15 |

**Table 3** Parameter setting of Distance method

| Method\parameters | ***n*** (# of voxels in each dimension) | HISTOGNUM (# of bins) | Threshold of Histogram |
|---|---|---|---|
| Distance value method | 100 (70 for data model 2) | 30 | 15 |

**Table 4** Parameter setting of connectivity method

| Method\parameters | Connectivity method |
|---|---|
| ***n*** (# of voxels in each dimension) | 100 (70 for data model 2) |
| HISTOGNUM (# of bins) | 30 |
| Threshold of Histogram | 15 |
| Count Threshold of Connectivity | 2 |

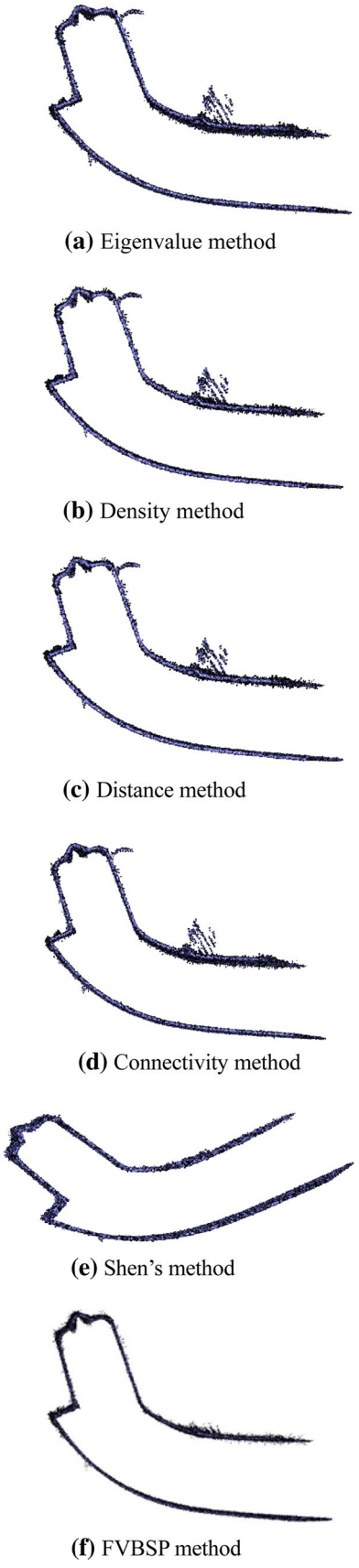
**Table 5** Parameter setting of VBFSP method

| Method\parameters | Connectivity method |
|---|---|
| ***n*** (# of voxels in each dimension) | 100 (70 for data model 2) |
| HISTOGNUM (# of bins) | 30 |
| Threshold of Histogram | 15 |
| Count Threshold of Connectivity | 2 |
| Weighting factor s in Eq. (4) | 0.6 (0.68 for data model 2) |
| Driving Force Threshold | 0.6 (0.355 for data model 2) |

**Table 6** Parameter setting of Shen's method

| Method\parameters | Shen's method [5] |
|---|---|
| ***n*** (# of voxels in each dimension) | 100 (70 for data model 2) |
| HISTOGNUM (# of bins) | 30 |
| Threshold of Histogram | 15 |
| Count Threshold of Connectivity | 2 |
| Weighting factor s in Eq. (4) | 0.6 (0.68 for data model 2) |
| Driving Force Threshold | 0.6 (0.355 for data model 2) |

**Fig. 7** Data model 1 processed by eigenvalue method, density method, distance method, connectivity method, Shen's method and our FVBSP method



**(a)** Eigenvalue method

**(b)** Density method

**(c)** Distance method

**(d)** Connectivity method

**(e)** Shen's method

**(f)** FVBSP method

and 8 Logical Processors), 32 GB Internal storage, and an NVIDIA Geforce GTX 970 M. To provide a fair basis for comparing all the tested algorithms, no parallel computing techniques were exploited.

We compared a density method, a distance method, an eigenvalue method, a connectivity method, Shen's surface propagation method [5] and our FVBSP method on various data models. The density method is based on the density of point clouds and the distance method is computed via the distance to tangential planes. The eigenvalue method is an approach for the classification of data points based on only the eigenvalue information of data points via a principal component analysis over a local neighborhood. The connectivity method is another approach, which relies upon only the voxel connectivity information to filter out isolated outliers. Shen's method is a sophisticated approach that utilizes a kd-tree and surface propagation with respect to data points.

Denoising accuracy is quantified by the following metric:

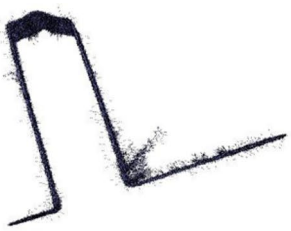
$$E_d = log\left(\frac{max\left(n_{positive} + n_{negative}, 1\right)}{n}\right) - \log\left(\frac{1}{n}\right), \qquad (4)$$

where $E_d$ stands for denoising error. $n_{negative}$ refers to the number of outliers that are considered to true data points, and $n_{positive}$ denotes the number of true data points that are marked as outliers. $n$ is the total number of points in a noisy data model. A smaller $E_d$ means a better denoising accuracy. The purpose of log() function in Eq. (4) is to handle the possibility of a very small fraction of $\frac{n_{positive}+n_{negative}}{n}$, while $\log\left(\frac{1}{n}\right)$ is used to make $E_d$ become zero when $n_{positive} + n_{negative} = 0$. In this paper, the ground truth of each data model was manually annotated.

The setting of parameters varies on different models, as shown in Tables 1, 2, 3, 4, 5 and 6. We adjusted the setting based on the performance of each model. Note that among all the parameters, $\boldsymbol{n_x} \times \boldsymbol{n_y} \times \boldsymbol{n_z}$ influences both efficiency and performance of the algorithms, while the rest of parameters have an impact only on performances.

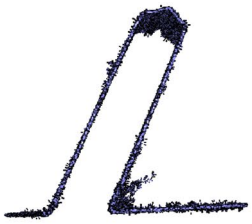
Figure 7 is a special case in which there are some clustered noise data inside a space bounded by a concave surface. The outlier clusters are so close to the main surface that many existing methods (e.g., eigenvalue method and connectivity method) fail to remove outliers, as shown in Fig. 7a, b. Shen's method and our VBFSP method perform well in this case since they have a stronger capability of handling non-isolated outlier clusters as we mentioned in Sect. 1.

Figure 8 is another typical case where non-isolated outlier clusters exist around a sharp corner. This case is more

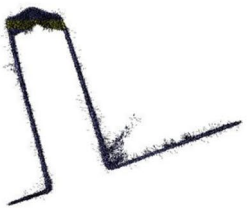


**(a)** Eigenvalue method

**(b)** Density method

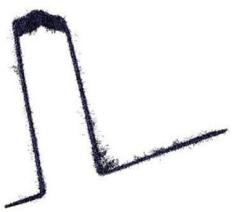


**(c)** Distance method



**(d)** Connectivity method

**(e)** Shen's method



**(f)** FVBSP method

**Fig. 8** Data model 2 processed by eigenvalue method, density method, distance method, connectivity method, Shen's method and our FVBSP method

**Table 7** Execution time and removed outlier count of data model 1 (194,102 points)

| Method | Number of reserved points | Number of removed outliers | Denoising error $E_d$ | Execution time (s) |
|---|---|---|---|---|
| Eigenvalue | 183,636 | 10,476 | 9.14 | 7.08 |
| Distance | 189,099 | 5003 | 8.26 | 7.99 |
| Density | 167,883 | 26,219 | 10.13 | 6.82 |
| Connectivity | 163,507 | 30,595 | 10.24 | 7.16 |
| Shen's method | 185,633 | 8469 | 5.89 | 39.0 |
| FVBSP | 187,809 | 6293 | 6.49 | 7.26 |

**Table 8** Execution time and removed outlier count of data model 2 (90,528 points)

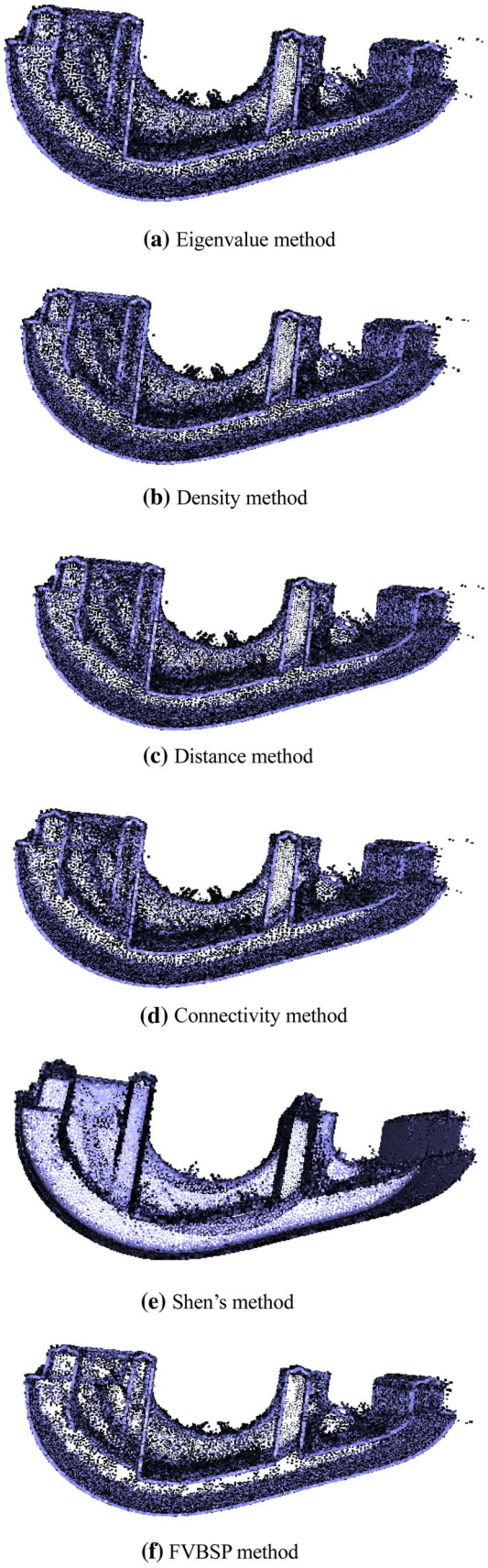
| Method | Number of reserved points | Number of removed outliers | Denoising error $E_d$ | Execution time (s) |
|---|---|---|---|---|
| Eigenvalue | 89,911 | 617 | 8.04 | 3.46 |
| Distance | 87,923 | 2605 | 7.70 | 3.83 |
| Density | 71,975 | 18,553 | 9.81 | 3.37 |
| Connectivity | 89,642 | 886 | 9.44 | 3.44 |
| Shen's method | 88,860 | 1668 | 9.10 | 24.0 |
| FVBSP | 86,176 | 4352 | 6.00 | 3.49 |

**Table 9** Execution time and removed outlier count of data model 3 (1,919,242 points)

| Method | Number of reserved points | Number of removed outliers | Denoising error $E_d$ | Execution time (s) |
|---|---|---|---|---|
| Eigenvalue | 1,910,830 | 8412 | 10.7 | 65.56 |
| Distance | 1,914,329 | 4913 | 10.6 | 68.89 |
| Density | 1,790,391 | 128,913 | 12.0 | 64.66 |
| Connectivity | 1,855,569 | 63,673 | 11.41 | 65.03 |
| Shen's method | 1,904,924 | 14,318 | 4.42 | 137.0 |
| FVBSP | 1,901,615 | 17,627 | 10.8 | 67.04 |

difficult than data model 1 since it contains non-isolated outlier clusters and sharp featured surface. Shen's method and our FVBSP method perform much better than the eigenvalue method, density method, distance method, and connectivity method.

It is clearly shown in Figs. 7 and 8 that the performance of our VBFSP method is very close to that of Shen's method. But, the FVBSP method is much more efficient than Shen's method, as illustrated in Tables 7, 8 and 9. Since our propagation is voxel-based, its computational efficiency is close to the eigenvalue method, density method, distance method and connectivity method. Shen's method is based on a kd-tree that is much more time-consuming in accessing the nearest neighbors.



**(a)** Eigenvalue method

**(b)** Density method

**(c)** Distance method

**(d)** Connectivity method

**(e)** Shen's method

**(f)** FVBSP method

**Fig. 9** Data model 3 processed by eigenvalue method, density method, distance method, connectivity method, Shen's method and our FVBSP method

Figure 9 is the results of removing outliers of a large data model (Fig. 10c). Shen's method produces the best denoising accuracy, while FVBSP is in a close range with the eigenvalue and distance methods. The execution time of Shen's method is much greater than the other methods.

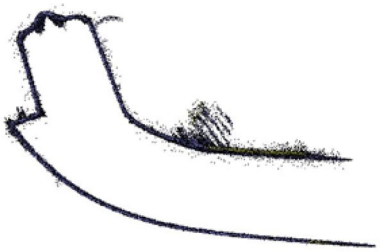
The time complexity of FVBSP is O($n$), where $n$ is the total number of data points. The real measurement of computational time over the three data models is in good agreement with this claim, as shown in Fig. 11. For clarity, we use a log–log relationship. The slope in Fig. 11 is approximately equal to 1.0, which indicates a linear relationship between the size of data models and the execution time of FVBSP. The memory complexity of FVBSP is O($n+m^3$), where $n$ reflects several arrays with the dimension length being the total number of data points and $m$ refers to the number of voxels in each dimension.

Compared with Shen's surface propagation method [5], some unique features of our FVBSP method include

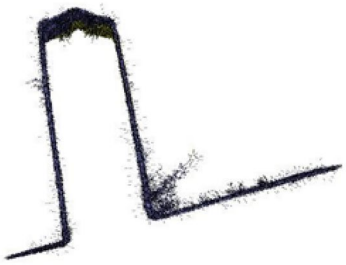
(a) The FVBSP method is based on voxels instead of a kd-tree. This makes it less computationally expensive than Shen's method.
(b) The FVBSP method combines not only the distance factor but also the angle factor, which makes it well-suited to different types of outlier clusters.
(c) The FVBSP method starts from one initial voxel and runs one round of propagation while Shen's surface propagation method starts from a number of initial points and runs several rounds of propagation. The experiment results show that as long as the driving force for surface propagation is effective and the initial voxel is good, one round of propagation is sufficient for small- or medium-sized models.

In summary, the FVBSP method simplifies the complexity of surface propagation while keeping a high performance of removing outliers that are difficult to deal with. It is particularly useful in handling small- and medium-sized data models. The advantage of FVBSP for large data models is not obvious at present. This demands future work for further improvement.

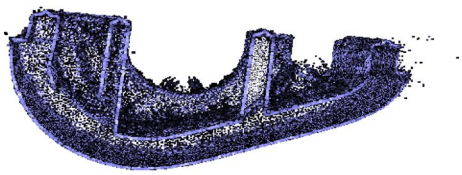
With the fast advances in autonomous vehicles in recent years, fast processing of a large amount of discrete data points from LIDAR (Light Detection And Ranging [24]) sensors becomes an emerging issue to investigate. Our FVBSP method will shed light on the real-time processing of LIDAR data points for autonomous vehicles.



**(a)** Data model 1 (194,102 points)



**(b)** Data model 2 (90,528 points)



**(c)** Data model 3 (1,919,242 points)

**Fig. 10** Noisy data models from laser scanning

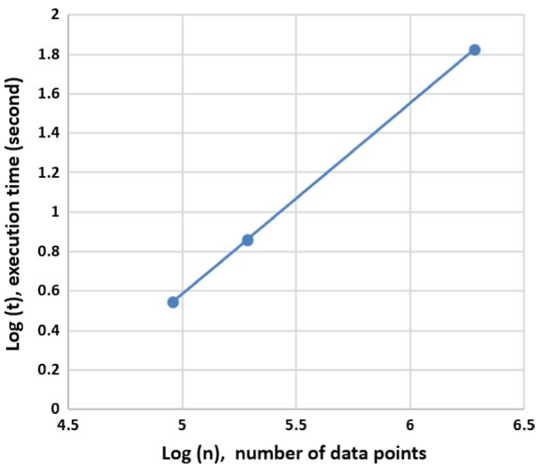



**Fig. 11** A log–log relationship between the sizes of data models and the execution time of FVBSP algorithm

## 4 Conclusions

In this paper, our algorithm is compared favorably to the existing methods because it can remove outliers even in the most difficult cases and it is more computationally efficient than Shen's method that can also remove difficult outliers. The unique contribution of this paper is to propose a voxel-based surface propagation approach with a linear combination of distance and angle as the driving force of propagation that can deal with complex types of outlier clusters. Numerical testing indicates that our method is well suited to small- and medium-sized data models.

**Acknowledgements** This work was in part supported by U.S. National Science Foundation DMI-0514900, CMMI-0721625, ECCS-1039563, and IIP-1445355.

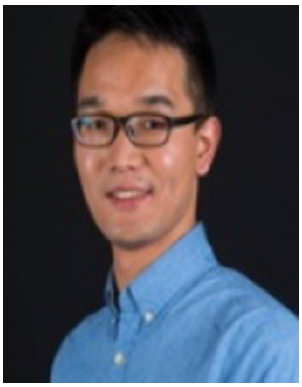
## References

1. Fleishman, S., Drori, I., & Cohen-Or, D. (2003). Bilateral mesh denoising. In *Proceedings of the 30th annual conference on computer graphics and interactive techniques* (pp. 950–953).
2. Xie, H., McDonnell, K. T., & Qin, H. (2004). Surface reconstruction of noisy and defective data sets. *IEEE Visualization, 2004,* 259–266.
3. Schall, O., Belyaev, A., & Seidel, H. (2005). Robust filtering of noisy scattered point data. In *Eurographics symposium on point-based graphics*.
4. Kolluri, R., Shewchuk, J. R., & O'Brien, J. F. (2004). Spectral surface reconstruction from noisy point clouds. In *Symposium on geometry processing* (pp. 11–21).
5. Shen, J., Yoon, D., Shehu, D., & Chang, S. Y. (2009). Spectral moving removal of non-isolated surface outlier clusters. *Computer-Aided Design, 41*(4), 256–267.
6. Wang, Y., & Feng, H. Y. (2015). Outlier detection for scanned point cloud using majority voting. *Computer-Aided Design, 62*(C), 31–43.
7. Zaman, F., Wong, Y. P., & Ng, B. Y. (2017). Density-based denoising of point cloud. In *9th international conference on robotic, vision, signal processing and power applications. lecture notes in electrical engineering*. Springer, Singapore.
8. Chen, H., & Shen, J. (2018). Denoising of point cloud data for computer-aided design, engineering, and manufacturing. *Engineering with Computers, 34*(3), 523–541.
9. Bazazian, D., Casas, J. R., & Ruiz-Hidalgo, J. (2015). Fast and robust edge extraction in unorganized point clouds. In *2015 international conference on digital image computing: techniques and applications (DICTA)* (pp. 1–8).
10. Dung, H. T. N., & Lee, S. (2015). Outlier removal based on boundary order and shade information in structured light 3D camera. In *2015 IEEe 7th international conference on cybernetics and intelligent systems (CIS) and IEEE conference on robotics, automation and mechatronics (RAM)* (pp. 124–129).
11. Williams, R. M., & Ilies, H. T. (2018). Practical shape analysis and segmentation methods for point cloud models. *Computer Aided Geometric Design, 67*(1), 97–120.
12. Shi, X., & Chen, H. (2018). Particle swarm optimization for constrained circular-arc/line-segment fitting of discrete data points. *International Journal of Modelling and Simulation, 38*(1), 25–37.
13. Shi, X., & Shen, J. (2016). Genetic search for optimally-constrained multiple-line fitting of discrete data points. *Applied Soft Computing, 40*(2), 236–251.
14. Yoo, D. J., & Kwon, H. H. (2009). Shape reconstruction, shape manipulation, and direct generation of input data from point clouds for rapid prototyping. *International Journal of Precision Engineering and Manufacturing, 10*(1), 103–113.
15. Nguyen, H. C., & Lee, B. R. (2014). Laser-vision-based quality inspection system for small-bead laser welding. *International*

*Journal of Precision Engineering and Manufacturing, 15*(3), 415–423.
16. Jang, Y. S., & Kim, S. W. (2017). Compensation of the refractive index of air in laser interferometer for distance measurement: A review. *International Journal of Precision Engineering and Manufacturing, 18*(12), 1881–1890.
17. Jang, Y. S., Kim, W., Jang, H., & Kim, S. W. (2018). Absolute distance meter operating on a free-running mode-locked laser for space mission. *International Journal of Precision Engineering and Manufacturing, 19*(7), 975–981.
18. Ram, P., & Gray, A. (2013). Which space partitioning tre to use for search? In *Advances in neural information processing system* (pp. 656–664).
19. Shekhar, R., Fayyad, E., & Yagel, R. (1996). Octree-based decimation of marching cubes surfaces. In *Proceedings of the IEEE visualization '96* (pp. 335–342).
20. Bentley, J. L. (1990). K-d trees for semidynamic point sets. In *Proceedings of 6th annual symposium on computational geometry* (pp. 187–197).
21. Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on management data* (pp. 47–57).
22. Hoppe, H., De Rose, T., Duchamp, T., MaDonald, J., & Stuetzle, W. (1993). Mesh Optimization. In *Proceedings of the 20th annual conference on computer graphics and interactive techniques* (pp. 19–26).
23. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific computing* (2nd ed.). Cambridge: Cambridge University Press.
24. Zhu, Q., Chen, L., Li, Q., Li, M., Nuchter, A., & Wang, J. (2012). 3D LIDAR point cloud based intersection recognition for autonomous driving. In *IEEE intelligent vehicles symposium* (pp. 456–461).



**Hao Chen** received the B.S. degree in Mechnical Engineering from China Agricultural University in 2003. He obained the Ph.D. in Mechanical Engineering from China Agricultural University in 2008. He has been a visiting scholar of University of Michigan-Dearborn in 2012. He is currently a professor at Shanghai University of Engineering Science. Dr Chen's research interest includes damage detection, damage modeling and evaluation, lightweight design.



**Yu Chen** received the B.S. degree in Telecommunications Engineering from University of Electronic Science and Technology of China in 2015. He is currently a Ph.D. student in Computer Science at Rensselaer Polytechnic Institute. His research areas include machine learning, data mining and their applications in natural language processing.

**Xu Zhang** received the B.S. degree in mechatronic engineering and M.S. degree in automotive engineering from Liaoning University of Technology in 2001 and 2004, respectively. She achieved the Ph.D. degree in mechanical engineering from Zhejiang University in 2008. She is currently an associate professor at Shanghai University Of Engineering Science. Her research interest includes reverse engineering, image processing, and Machining Simulation.

**Baiyuan Li** received the B.S. degree in Computer Science from University of Electronic Science and Technology of China in 1991 and M.S. degree in Industrial Automation from Nanchang University in 1994 respectively. He obtained the Ph.D. in Computer Science from Shanghai Jiao Tong University in 2005. Dr. Li's research areas include computer graphics, artifical intelligence, and distributed computation. He is currrently an associate professor of School of Computer Science and Technology, Donghua University.

**Xiaoqiang Liu** received the B.S. degree and M.S. degree in Computer Science from Harbin Institute of Technology in 1990 and 1995, respectively. She obtained the Ph.D. in Control Theory and Engineering from Donghua University in 2003. She is currently a professor at Donghua University. Dr. Liu's research areas include intelligent system, data mining, computer aided design and manufacturing.

**Xuefei Shi** received the B.S. degree in Automation Instrument from University of Science & Technology Beijing in 1994 and the M.S. degree in Detection Technology and Automatic Equipment from USTB in 2000. She achieved the Ph.D. degree in Control Science and Engineering from USTB in 2011. She has been a visiting scholar of University of Texas at Arlington for four months in 2006. She is currently an associate professor at USTB. Her research interest includes detection and control technology, computer modeling and optimization technology and machine vision.

**Jie Shen** obained the M.S. degree and Ph.D. in Computer Science from the University of Saskatchewan in 1997 and 2000. From 2002 to 2008, he was an assistant professor with the University of Michigan – Dearborn. From 2015 to present, he is a professor in the same institute. Dr. Shen is a fellow of ASME and IET. He has published three books and over 130 technical papers. He serves as the editor-in-chief for *International Journal of Modelling and Simulation*. He organized many international conferences and workshops. He is the inventor of the University of Michigan Invention 3805, which was licensed to a U.S. company of laser sensors.