Point Cloud Normal Estimation by Fast Guided Least Squares Representation

JIE ZHANG\textsuperscript{1}, JIAHUI DUAN\textsuperscript{1}, KEWEI TANG\textsuperscript{1}, JUNJIE CAO\textsuperscript{2}, AND XIUPING LIU\textsuperscript{2}

\textsuperscript{1}School of Mathematics, Liaoning Normal University, Dalian 116029, China
\textsuperscript{2}School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China

Corresponding author: Jie Zhang (jzhang@lnnu.edu.cn)

This work was supported in part by NSFC Funds under Grant 61702245, Grant 61702243, and Grant 61772104, in part by the Liaoning Revitalization Talents Program under Grant XLYC1907169, and in part by the National Science and Technology Major Project under Grant 2018ZX04041001-007 and Grant 2018ZX04016001-011.

ABSTRACT Normal estimation is an essential task for scanned point clouds in various CAD/CAM applications. The method (GLSRNE) based on guided least squares representation (GLSR) balances speed with quality well among state-of-the-art methods. First, it segments each neighborhood into multiple sub-neighborhoods. For some neighborhoods, the segmentation is obtained by GLSR which is an efficient subspace segmentation model and widely applied in other applications. The segmentation of the rest neighborhoods is inferred via the subspace structure propagation (SSP) algorithm. Then, each sub-neighborhood is fitted by a plane. The plane achieving the minimum distance with the current point is selected for the final normal estimation. We make improvements for effectiveness and efficiency in the following three aspects. First, to improve the speed of GLSR, we propose a novel iterative algorithm to reduce the computation complexity from $O(n^3)$ to $O(n^2)$ with its convergence guaranteed theoretically, where $n$ represents the number of the data points. Moreover, this proposed algorithm will also be useful for other applications. Second, we add a normal constraint for SSP to improve accuracy. Third, when selecting one plane to estimate the final normal, we consider the match between the plane and all neighbors, whereas GLSRNE only considers the match between the plane and the current point. The experiments exhibit that our method is faster than GLSRNE and more effective than GLSRNE and other state-of-the-art methods.

INDEX TERMS Normal estimation, feature preserving, least squares representation, fast algorithm.

I. INTRODUCTION

With the recent proliferation of commodity 3D sensors, such as RGB-D cameras and LiDAR, there is an increasing demand for using point clouds to understand the geometry even semantics of the scanned objects and scenes. This raw representation is challenging to handle since it lacks connectivity information or structure. When available, additional local geometric information, such as the surface normals at each point, will induce a partial local structure and benefit many point cloud processing tasks including point based rendering [1], surface reconstruction [2], classification [3], semantic segmentation, and depth prediction [4]. However, fast and reliable normal estimation is still an intriguing challenge since sharp features, noise, and sampling anisotropy are inevitably contained in raw point clouds. There are a considerable amount of works on this topic [5]–[7]. They can be categorized into six types: Delaunay/Voronoi based methods, mollification based methods, regression based methods, statistics based methods, networks based methods, and segmentation based methods.

Delaunay/Voronoi based methods [8]–[10] all employ Voronoi diagram or Delaunay triangulation. This kind of method only performs well for small noise point clouds and none of them can handle point clouds with outliers and sharp features [11]. Mollification based methods [12]–[15] are inspired by the feature preserving image filters. They improve the preliminary normals but fail to recover sharp features if the preliminary normals do not preserve sharp features sufficiently. With the assumption that all neighbors of a point are sampled from one smooth surface, regression-based methods [16]–[20] use all neighbors to estimate the normal. However, when a point is near sharp features, its neighbors may be sampled from multiple surfaces. Hence, these methods tend to smooth out the sharp features. Ideally, only the neighbors on the same surface with the current point...
applied in other applications [39], [40]. However, GLSR still needs to compute the inverse of some $n \times n$ matrices, where $n$ denotes the number of the data points. Hence, the scalability of GLSR is very limited in this big data era.

In this paper, we address the problem of point cloud normal estimation. To overcome the limitation of GLSR, we propose a novel iterative algorithm to solve the model of GLSR with its convergence guaranteed theoretically. Its computation complexity is $O(n^3)$ whereas that of the original algorithm for GLSR is $O(n^2)$, where $n$ represents the number of the data points. Hence, we called our method as Fast Guided Least Squares Representation (FGLSR). Besides, we also make improvements for normal estimation in the following two aspects. First, to improve the accuracy of SSP, we add a normal constraint for it. Second, when selecting one plane fitted by one sub-neighborhood to estimate the final normal, we consider the match between the plane and all neighbors, whereas GLSRNE only considers the match between the plane and the current point. Extensive experimental results confirm the efficiency and effectiveness of our novel method for normal estimation.

II. ALGORITHM

GLSRNE provides a viable normal estimation framework for point clouds with sharp features. We will briefly go through the general framework and then introduce some improvements to generate a more effective and efficient normal estimation algorithm.

A. GLSRNE

Given a noisy point cloud $\{p_i\}_{i=1}^N$ as input, GLSRNE takes three steps to estimate the normals preserving shape features.

Step 1: Estimating initial normals and detecting candidate feature points. For each point $p_i$, GLSRNE computes an initial normal $n_i^0$ and feature coefficient $w_i$ by eigenvalue analysis of the covariance matrix:

$$
\mathbf{T}_i = \frac{1}{S} \begin{bmatrix} p_i^1 - \hat{p}_i \\ \vdots \\ p_i^S - \hat{p}_i \\ \hat{p}_i^S - \hat{p}_i \end{bmatrix}^T \begin{bmatrix} p_i^1 - \hat{p}_i \\ \vdots \\ p_i^S - \hat{p}_i \\ \hat{p}_i^S - \hat{p}_i \end{bmatrix}, \quad p_i^j \in \mathcal{N}_i, \quad (1)
$$

where $\mathcal{N}_i$ is the neighborhood of $p_i$ with size $S$ and $\hat{p}_i = \frac{1}{S} \sum_{p_j^s \in \mathcal{N}_i} p_j^s$. $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$ denote the singular values of $\mathbf{T}_i$. The initial normal $n_i^0$ computed by the classical normal estimation method PCA [16] is the eigenvector corresponding to $\lambda_0$. The coefficient $w_i = \lambda_0 / (\lambda_0 + \lambda_1 + \lambda_2)$ measures the confidence of $p_i$ close to sharp features. If $w_i$ is larger than the threshold $w_i^1$, $p_i$ is regarded as a candidate feature point whose neighborhood may be sampled from several smooth surfaces.

Step 2: Segmenting the neighborhoods of candidate feature points. For each candidate feature point $p_i$, a larger neighborhood $\mathcal{N}_i^*\text{ with size } S^*\text{ is selected, where } S^* > S$. GLSRNE segments $\mathcal{N}_i^*$ into several sub-neighborhoods which enclose $1$.
only the points located on the same surface. The neighborhood of candidate feature points are divided into two parts. First, a small part of neighborhoods is segmented by GLSR. Then, by using SSP, the segmentation of the rest neighborhoods is inferred from the results of neighborhoods segmented by GLSR.

**Neighborhood segmentation by GLSR.** First, for a candidate feature point \( p_i \), the data matrix \( X_i \) and guiding matrix \( \Omega_i \) are computed. Specifically, the \( j \)-th neighbor point \( p_j^i \) of \( p_i \) is represented as \( x_j = [x_j^1; x_j^2; z_j; n_j^1; n_j^2; n_j^3] \), where \([n_j^1; n_j^2; n_j^3] \) is the initial normal \( n_j^i \) of \( p_j^i \) computed by PCA and \([x_j^1; x_j^2; z_j] \) is the local coordinate of \( p_j^i \) with \( p_i \) as the origin. The data matrix \( X_i \) is defined as \( X_i = [x_1, x_2, \cdots, x_S] \). For guiding matrix \( \Omega_i \), if \( p_j^i \) or \( p_k^i \) is on different surfaces, \( \Omega_i(j,k) \) should have larger values. According to the observation that two neighbors sampled on the same surface will obtain similar initial normals, GLSRNE defines \( \Omega_i(j,k) \) as follows:

\[
\Omega_i(j,k) = \begin{cases} 
1 & d(n_j^i, n_k^i) > \tau_{\text{max}} \\
0 & \text{otherwise}, 
\end{cases}
\]

where \( d(n_j^i, n_k^i) = 1 - | < n_j^i, n_k^i > | \).

\( \tau_{\text{max}} \) is a parameter, \( 2 < \tau_{\text{max}} < \cdots > 0 \) represents the inner product of two vectors. To improve the reliability of \( \Omega_i \), GLSRNE employs two strategies used by [29]. Firstly, the neighborhood segmentation results are stored and used to update the guiding matrix of neighborhoods segmented later. Secondly, when \( p_j^i \) or \( p_k^i \) is a candidate feature point, the value of \( \Omega_i(j,k) \) is decreased. For more details please refer to section 4.3.2 of [29]. Then one optimal representation coefficient matrix is computed by solving GLSR:

\[
\min_{Z} \lambda \|X - XZ\|^2_F + \|Z\|^2_F + \beta \|W \odot Z\|^2_F,
\]

where \( X \in \mathbb{R}^{d \times n} \) is the data matrix with each column corresponding to the data point sampled in a \( d \)-dimensional space; \( \odot \) is the Hadamard product defined as the element-wise multiplication; Both \( \lambda \) and \( \beta \) are the balancing parameters; \( W \in \mathbb{R}^{n \times n} \) is a predefined weight matrix whose elements are all greater than or equal to zero and \( W(i,j) \) is expected to be larger when \( x_i \) and \( x_j \) are in different categories; \( Z \in \mathbb{R}^{n \times d} \) is the representation matrix. Both \( \|Z(i,j)\| \) and \( \|Z(j,i)\| \) can be understood as the similarity between \( x_i \) and \( x_j \). For each candidate feature point \( p_i \), \( X = X_i \), \( W \odot W = \Omega_i \), \( n = S^a \), \( d = 6 \), and the solved representation coefficient matrix is denoted as \( Z_i \). Next, the affinity matrix \( S_i \) for \( N_i^a \) is defined as \( S_i = |Z_i| + |Z_i^T| \), where \( Z_i^T \) is the transpose of matrix \( Z_i \) and \( |Z_i| \) represents a matrix that is defined as \( |Z_i|(i,j) = |Z_i(i,j)| \). Finally, GLSRNE segments \( N_i^a \) into several sub-neighborhoods by applying \( S_i \) to Normalized Cuts [41]. The number of sub-neighborhoods is determined by the iterative segmentation process described in [29].

\[\text{Neighborhood segmentation by SSP.} \] After some neighborhoods segmented by GLSR, GLSRNE designs SSP for inferring the segmentation of the rest neighborhoods. Specifically, GLSRNE stores the neighborhood segmentation results obtained by GLSR. The correlations between neighbors are defined by the segmentation results. If a candidate feature point \( p_i \) and its neighborhood \( N_i^a \) are well covered by neighborhoods segmented by GLSR, the segmentation of \( N_i^a \) is obtained by the region-growing algorithm with the correlation.

**Step 3: Normal estimation.** If \( p_i \) is not a candidate feature point, its neighbors will be sampled from one smooth surface. The normal of \( p_i \) is defined as the initial normal \( n_i^0 \) estimated by PCA. If \( p_i \) is a candidate feature point, \( N_i^a \) is segmented into several sub-neighborhoods. For each sub-neighborhood, a plane is fitted by PCA and the distance between \( p_i \) and the plane is computed. The normal of \( p_i \) is defined as the normal of the plane with minimal distance.

**B. FAST SOLVER FOR GLSR**

Although the original algorithm of GLSR is usually faster than those of most related works, it still requires the computation of the inverse of some \( n \times n \) matrices, whose computation complexity is \( O(n^3) \). The explosion of the data motivates us to provide a novel algorithm to reduce the computation burden.

The original algorithm of GLSR computes the representation matrix \( Z \) column-wise. To find a novel algorithm, we try a different way of updating \( Z \) row-wise. In the following, \( \tilde{Z} \) and \( \hat{x} \) denote the \( i \)-th row of \( Z \) and \( i \)-th column of \( X \), respectively. Hence,

\[
XZ = \sum_{i=1}^{n} \hat{x}_i \tilde{z}_i. 
\]

Denote \( \tilde{x}_j^{k+1} \) as the result of \( \tilde{z}_j \) in \( k \)-th iteration. We adopt the strategy of alternatively computing each \( \tilde{z}_j \) by fixing the others, where \( i \) is from 1 to \( n \). In this way, when computing \( \tilde{x}_j^{k+1} \), we have already updated \( \tilde{x}_j \), \( j = 1, \cdots, i-1 \). Employing Eq. (5), we formulate \( X - XZ \) as

\[
X - \sum_{i=1}^{n} \tilde{x}_i^{k} \tilde{z}_i^{k+1} - \sum_{j=i+1}^{n} \hat{x}_j \tilde{z}_j^{k} - \hat{x}_i \tilde{z}_i^{k}. 
\]

It can be reformulated as

\[
X - XZ_{k+1} + \hat{x}_i \tilde{z}_i^{k+1} - \hat{x}_i \tilde{z}_i^{k}, 
\]

where

\[
\tilde{Z}_{i-1}^k = \begin{bmatrix} \tilde{x}_i^{k+1} \\ \vdots \\ \tilde{x}_i \\ \vdots \\ \tilde{x}_n \\ \tilde{x}_i \end{bmatrix}.
\]
Hence, the problem about $\bar{Z}_i$ is
\[
\min_{\bar{Z}_i} \lambda \| \mathbf{Y}_i^k - \bar{Z}_i \|_F^2 + \| \mathbf{Y}_i^k \|_F^2 + \beta \| \mathbf{W}_i \odot \bar{Z}_i \|_F^2,
\]
where $\mathbf{W}_i$ is the $i$-th row of $\mathbf{W}$ and
\[
\mathbf{Y}_i^k = \mathbf{X} - \mathbf{XZ}_k^{-1} + \hat{\mathbf{x}}_i^k.
\]
The closed-form solution of the model (9) can be obtained by setting the derivative as zero:
\[
2\lambda(-\hat{\mathbf{x}}_i)^T(\mathbf{Y}_i^k - \hat{\mathbf{x}}_i) + 2\beta \mathbf{W}_i \odot \bar{Z}_i \odot \bar{Z}_i + 2\mathbf{Z}_i = 0.
\]
Hence, the result of $\hat{\mathbf{x}}_i^k$ is
\[
\hat{\mathbf{x}}_i^k = (\lambda \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_i + 1)\mathbf{I}^T + \beta \mathbf{W}_i \odot \bar{Z}_i.
\]

Algorithm 1 FGLSR

**Input:** The data set $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$, the weight matrix $\mathbf{W}$, the parameters $\beta$, $\lambda$, and $\varepsilon$.

**Initialize:** $\mathbf{Z}$

Compute $\mathbf{Y} = \mathbf{X} - \mathbf{XZ}$. Compute $a_i = \lambda \| \hat{\mathbf{x}}_i \|_F^2 + 1$, $i = 1, \cdots, n$.

while not converged do

$\mathbf{Z}^* = \mathbf{Z}$.

for $i = 1 : n$ do

Update $\mathbf{Y} = \mathbf{Y} + \hat{\mathbf{x}}_i \mathbf{Z}_i$.

Update $p = a_i \mathbf{I}^T + \beta \mathbf{W}_i \odot \bar{Z}_i$.

Update $q = \lambda \hat{\mathbf{x}}_i^T \mathbf{Y}$.

Update $\bar{Z}_i = q / p$.

Update $\mathbf{Y} = \mathbf{Y} - \hat{\mathbf{x}}_i \mathbf{Z}_i$.

end for

Check the convergence condition $\| \mathbf{Z}^* - \mathbf{Z} \|_\infty < \varepsilon$.

end while

It can be found that the highest computation burden in each step of Algorithm 1 is that an $n \times n$ dimensional vector multiplies with an $n \times n$ matrix, whose computation complexity is only $O(n^3)$. Whereas that of GLSR is $O(n^5)$. In addition, we also provide a theoretical analysis of FGLSR in detail.

**Theorem 1:** Denote $\mathbf{Z}_k$ as the result of $\mathbf{Z}$ in $k$-th iteration, we can get
\[
\lambda \| \mathbf{X} - \mathbf{XZ}_k \|_F^2 + \| \mathbf{Z}_k \|_F^2 + \beta \| \mathbf{W} \odot \mathbf{Z}_k \|_F^2 \\
\leq \lambda \| \mathbf{X} - \mathbf{XZ}_k \|_F^2 + \| \mathbf{Z}_k \|_F^2 + \beta \| \mathbf{W} \odot \mathbf{Z}_k \|_F^2.
\]

Theorem 1 implies that FGLSR can at least converge to a local optimum of the problem (4). Because the model of GLSR is convex, the local optimum will be the global optimal solution. This result means that FGLSR can converge to the global optimal solution to problem (4). In our experience, by initializing $\mathbf{Z}$ as $\lambda (\mathbf{X}^T \mathbf{X} + (1 + \beta)\mathbf{I})^{-1}\mathbf{X}^T \mathbf{X}$, Algorithm 1 usually converges very fast. FGLSR is far faster than the GLSR, especially when the number of data points is very large. Next, we will perform a simple experiment on synthetic data to confirm this.

We randomly generate $n$ data points from three lines in $R^2$ on average. The slopes of these lines are $\tan\left(\frac{\pi}{6}\right)$, $\tan\left(\frac{\pi}{6}\right)$, and $\tan\left(\frac{\pi}{6}\right)$, respectively. In this situation, these three subspaces are disjoint. Hence, $\mathbf{W}$ is constructed as a binary matrix according to the theorem about disjoint subspaces in [38]. We compare the average computation time in seconds by repeating the experiment ten times to each choice of $n$. These experiments are all performed on a computer with an Intel Core i7-9700k CPU and 16-GB memory. The results are reported in Table 1 by varying $n$ from 600 to 3000. It can be found that FGLSR is far faster than GLSR. As $n$ increases, the gap becomes wider and wider gradually. We also report the difference of the representation matrices computed by FGLSR and GLSR in these experiments. The measure is the infinity norm $\| \mathbf{\cdot} \|_\infty$, i.e., the maximal of the absolute value. The results are shown in Table 2. It can be observed that the average difference is usually below 0.01, meaning that FGLSR converges well. This synthetic data experiment demonstrates our improvement in computation speed.

### C. NORMAL CONSTRAINED SSP

After some neighborhoods segmented by GLSR, GLSRNE uses SSP to propagate the segmentation to the rest neighborhoods. To improve accuracy, we add normal constraint in the propagation.

First, we want to use GLSR to segment the neighborhoods which cover the candidate feature points well and are easy to obtain faithful results. The point $p_i$ with the larger $w_i$ is usually near a complex structure which makes the segmentation challenging. On the contrary, the point with the smaller $w_i$ may be far away from the sharp features and can not cover the candidate feature points well. Therefore, we rank the candidate feature points by $w_i^f = |w_i - w_{ave}|$, where $w_{ave}$ is the average of feature coefficients of all the candidate feature points. We select $k = N_f \ast r$ neighborhoods of points with smaller $w_i^f$ and segment them by GLSR, where $r \in [0, 1]$ is a parameter and $N_f$ is the number of candidate feature points. The selected $k$ points are denoted as $P_f = \{p_1, p_2, \cdots, p_k\}$.

Then, given one of the rest candidate feature points $p_i$ and its neighborhood $N_i^s$, we need to determine whether $p_i$ and $N_i^s$ are well covered by the neighborhoods segmented by GLSR. Specifically, we select a smaller neighborhood $N_i^s$...
with size $S'$. If $N_i^s \cap p_T = \emptyset$, we consider $p_i$ and $N_i^s$ are not well covered and segment $N_i^s$ by GLSR. Otherwise, we segment $N_i^s$ by normal constrained SSP.

Specifically, we use two matrices $R$ and $N$ to store the segmentation results of GLSR. The values of $R(j, k)$ and $N(j, k)$ represent the number of $p_i^k$ and $p_i^k$ grouped into the same surface and into different surfaces, respectively. The correlations between neighbors are defined by the segmentation results:

$$rel(p_i^k, p_j^k) = \begin{cases} -1, & \text{if } N(j, k) > 0 \text{ or } |n_i^k, n_j^k| > |\cos(\sigma)| \\ R(j, k), & \text{if } N(j, k) = 0 \text{ and } |n_i^k, n_j^k| > |\cos(\sigma)|, \end{cases}$$

where $\sigma = \pi/4$. Then, the point $p_i^k$ with the smallest feature coefficient is chosen as seed point and the point set representing the points on the same surface with $p_i^k$ is initialized by $N_i^s = \{p_i^k\}$. The next step is iteratively adding one point to the set $N_i^s$. At each iteration, we chose the point which obtains the largest relation with $N_i^s$. The relation between point $p$ and set $N_i^s$ is defined as:

$$REL(p, N_i^s) = \max_{p_j \in N_i^s} (rel(p, p_j)).$$

This process is terminated until $|N_i^s| > 0.7 \times S'$ or $REL(N_i^s \setminus N_i^s, N_i^s) \leq 2$, where $|N_i^s|$ represents the number of points in $N_i^s$.

$$REL(N_i^s \setminus N_i^s, N_i^s) = \max_{p_i^k \in N_i^s \setminus N_i^s} (REL(p_i^k, N_i^s)).$$

Next, we repeat the process for the rest of the neighbors $N_i^s = N_i^s \setminus N_i^s$ until $N_i^s$ is empty. If the neighbor $p_i^j$ does not belong to any neighborhood segmented by GLSR, i.e. there is no class information of $p_i^j$ in $R$ and $N$, we ignore $p_i^j$ and delete it from $N_i^s$. Finally, $N_i^s$ is segmented into multiple sub-neighborhoods $N_i^s, N_i^s, \ldots$.

### D. SUB-NEIGHBORHOOD SELECTION

If $p_i$ is a candidate feature point, $N_i^s$ is segmented into multiple sub-neighborhoods $\{N_i^s \}_{i=1}^m$. For each sub-neighborhood $N_i^s$, GLSRNE uses a plane $\Theta_i^j$ to fit it and compute the distance $d(\Theta_i^j, p_i)$ between $p_i$ and the plane $\Theta_i^j$. The normal of $p_i$ is defined as the normal of $\Theta_i^j$ which has the minimal distance. GLSRNE only considers the current point $p_i$ but not the whole neighborhood $N_i^s$, when selecting the plane used to estimate the final normal. For selecting a more reliable plane, we consider the fitting result of the whole neighborhood $N_i^s$. We define how well $\Theta_i^j$ fits into $N_i^s$ by:

$$\text{dist}(\Theta_i^j, N_i^s) = \sum_{p_{i}^k \in N_i^s} \gamma_{ik} \times d(\Theta_i^j, p_{i}^k),$$

where $\gamma_{ik}$ is the density weight defined as the average distance from $p_{i}^k$ to its $k = 10$ nearest neighbors, $d(\Theta_i^j, p_{i}^k)$ is the distance between $p_{i}^k$ and the plane $\Theta_i^j$. We define the normal of $p_i$ as that of plane $\Theta_i^j$ which obtains the minimal $\text{dist}(\Theta_i^j, N_i^s)$. 
FIGURE 2. Visual rendering of computed normals for Octahedron (65K) with 30% noise, Tetrahedron (60K) with 40% noise, Fandisk (77K) with 50% noise, Mechpart (92K) with 50% noise, and Cylinder (60K) with 60% noise. Our method respects the sharp features.

III. RESULTS
To evaluate the performance of the proposed normal estimation method, we test our method on a variety of point clouds with sharp features. We compare our method with some classic and state-of-the-art methods: PCA [16], robust normal estimation (RNE) [11], robust randomized hough transform (RRHT) [5], convolutional neural networks based on hough transform HTCNNs [26], GLSRNE [30], and pair consistency voting (PCV) [6]. The precision and speed are evaluated on the models perturbed by centered Gaussian noise with the deviation defined as 30%, 40%, 50%, and 60% average distance between points. In this section, all experiments have been performed with an Intel Core i7-9700k CPU and 16-GB memory. The parameters $S, S^*, S^t, r, \lambda$, and $\beta$ are the same as GLSRNE: $S = 70, S^* = 120, S^t = 30, r = 0.1, \lambda = 1$ and $\beta = 4$. We use the same neighborhood size $S^*$ for PCA, RNE, and RRHT. HFCNNs considers five neighborhood sizes (32, 64, 128, 256, 512) simultaneously. All other parameters, if any, are set to default.

We compare the various algorithms by the Root Mean Square measure with threshold (RMS$_\tau$) which has been used in [5], [29]. The definition of RMS$_\tau$ is:

$$\text{RMS}_\tau = \sqrt{\frac{1}{|P|} \sum_{p \in P} f(n_{p,\text{ref}} \hat{n}_{p,\text{est}})},$$

where

$$f(n_{p,\text{ref}}, \hat{n}_{p,\text{est}}) = \begin{cases} n_{p,\text{ref}}, & \text{if } n_{p,\text{ref}} \hat{n}_{p,\text{est}} \leq \tau, \\ \pi/2, & \text{otherwise} \end{cases}.$$

(16)
and $n_{p,ref}$ and $n_{p,est}$ are the ground truth and estimated normals at $p$, respectively. As proposed by [5], [29], we take $\tau = 10$ degrees. The algorithm with lower RMS$_\tau$ and less run time is desired.

### A. COMPARISON ON SYNTHESIZED DATA

Fig. 1 shows the RMS$_\tau$ of PCA, RNE, RRHT, HTCNNs, LRSGNE, PCV, and our method on Cylinder(60K), Fan-disk(77K), Mechpart(92K), Octahedron(65K), and Tetrahe-dron(60K). The visualization of computed normals is shown in Fig. 2.

For PCA, the normal estimation results are overly smooth on/near sharp features and it achieves higher RMS$_\tau$ than other methods. The results of RNE are slightly better than PCA, but the sharp features are still smoothed out. The performance of RRHT and HTCNNs is comparable. HTCNNs estimates normals by using multiple neighborhood sizes simultaneously. Hence, as the deviation of Gaussian noise increases, the RMS$_\tau$ of HTCNNs changes gently (see Fig. 1 f). But it fails to correctly estimate normals near sharp features (see Fig. 2). RRHT does somewhat better for points near the sharp features, see the second row in Fig. 2, but when the noise is large or the neighborhood structure is complex, the results of RRHT are still smooth. GLSRNE and PCV preserve sharp features better, but in the vicinity of sharp features, some erroneous normals may persist, as shown in Fig. 2. Our method obtains the lowest RMS$_\tau$ and preserves shape features better in the presence of noise.

### B. ROBUSTNESS TO SAMPLING ANISOTROPY

Fig. 4 shows the RMS$_\tau$ on Cylinder(121) and Cylinder(131) sampled with face-specific levels of density. For Cylinder(121) and Cylinder(131), the ratios of sampling number on the upper, side, and lower faces are 1 : 2 : 1 and 1 : 3 : 1, respectively. For a point, if the minimal angle between the estimated normal and ground truth normals is larger than the threshold $\tau = 10$, it is regarded as a bad point. The visualization of bad points and the number of bad points (NBP) are shown in Fig. 3.

Since PCA cannot preserve sharp features, there are a large number of points near sharp features. RNE is devised to preserve sharp features, but it does not address the variation of density near edges. Therefore RNE is severely affected by non-uniform sampling. The bad points on the upper/lower face significantly outnumber those on the side face. Both the sampling strategy designed by RRHT and the segmentation method used by GLSRNE are not sensitive to density variation. Therefore they perform better than PCA and RNE. However, near the sharp features, there are still some erroneous normals. For the visualization of bad points shown in Fig. 3, the results of PCV and our method are comparable and more precise than the other methods. But our method achieves the minimal NBP and the lowest RMS$_\tau$.

### C. COMPUTATION TIMES

We report the speed of PCA, RNE, RRHT, HTCNNs, GLSRNE, and our method in Tab 3. PCA is the fastest,
FIGURE 4. Comparison of the RMS\(\tau\) on Cylinder(121) and Cylinder(131) with different noise levels. The results of our method are more precise than the other methods.

Even with the MATLAB implementation, but at the price of a higher RMS\(\tau\). The methods with C/C++ implementation (including RNE, RRHT, and HTCNNs) are fast, but the normals obtained by them are far worse than those of GLSRNE, PCV, and our method. The speed of our method is acceptable and faster than GLSRNE and PCV. A parallel C++ implementation could increase our performance remarkably.

D. RESULTS OF REAL DATA
We further demonstrate the effectiveness of our method on more real scanned point clouds in which the typical imperfections, such as noise, outliers, and sampling anisotropy, are common and the sharp features are usually corrupted by these imperfections. Fig. 5 shows the results of our method on the raw scans: Genus2, Taichi, Nyon, and Shutter. The edges of these raw scans are preserved well. Detail 1 of
TABLE 3. The average computation time (in seconds) and RMSr on each model.

| Implementation | PCA       | RNE       | RRHT      | HTCNNs    | GLSRNE    | PCV       | OUR       |
|----------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
|                | MatLab   | C/C++     | MatLab   | C/C++     | MatLab   | MatLab   | MatLab   |
| Cylinder       | 0.7414    | 0.1946    | 0.0939    | 0.1634    | 0.0859    | 0.0742    | 0.0702    |
| 60K time       |           |           |           |           |           |           |           |
| Fandisk        | 1.0771    | 0.4293    | 0.3371    | 0.2814    | 0.2206    | 0.2061    | 0.2002    |
| (77K) time     |           |           |           |           |           |           |           |
| Mechpart       | 1.1050    | 0.5747    | 0.3987    | 0.3819    | 0.3391    | 0.3328    | 0.3268    |
| (92K) time     | 23.44     | 29.03     | 47.03     | 41.69     | 153.00    | 125.97    | 112.44    |
| Octahedron     | 0.8377    | 0.4156    | 0.3587    | 0.2058    | 0.2045    | 0.1900    | 0.1761    |
| (65K) time     | 13.61     | 20.04     | 31.91     | 30.34     | 57.63     | 53.4698   | 42.17     |
| Tetrahedron    | 0.8957    | 0.2959    | 0.1474    | 0.2261    | 0.1059    | 0.1016    | 0.1019    |
| (60K) time     | 14.78     | 18.65     | 26.31     | 28.22     | 66.27     | 55.0666   | 49.16     |

Genus2 shows the results of nearby surface sheets which always challenge the normal estimation. Our method recovers the features faithfully. Sampling anisotropy is ubiquitous, see detail 2 of Genus2, detail 1 and detail 3 of Shutter, and detail 1 of Taichi and Nyon. Our method generates faithful normals for these points. Tiny structures, as shown in detail 2 of Shutter, challenge the normal estimation persistently. Our method copes with such case and recovers the features.

IV. CONCLUSION

In this paper, we propose a novel method for estimating faithful normals for point clouds with multiple types of noise, sharp features, and anisotropic samplings. Our method follows the framework of GLRSNE with three improvements making it more efficient and effective. Although our proposed method generates quality normals, it produces jagged features on sparse point clouds, similar to some existing methods. For future work, it will be interesting to try global labeling techniques to smooth out these jagged feature lines. Furthermore, an adaptive neighborhood or multiple neighborhood sizes would help to improve accuracy and reduce the number of parameters. At last, GLSR is widely applied in other applications [39], [40]. We have shown that as the number of data points increases, the time gap between FGLSR and GLSR becomes wider and wider gradually. Therefore, it would be interesting to apply FGLSR to more computer vision and computer graphics applications associated with large data sets.

A. PROOF OF THE CONVERGENCE

Proof:

\[
\begin{align*}
\lambda \|X - XZ^{k+1}\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot Z^{k+1}\|_2^2 \\
= \lambda \|X - \sum_{i=1}^{n} \tilde{x}_i c_i^{k+1}\|_F^2 + \beta \sum_{i=1}^{n} \|w_i \odot c_i^{k+1}\|_2^2 \\
+ \sum_{i=1}^{n} \|c_i^{k+1}\|_2^2 = \lambda \|Y_n^{k} - \tilde{x}_n c_n^{k+1}\|_F^2 + \sum_{i=1}^{n-1} \|\tilde{z}_i c_i^{k+1}\|_2^2 \\
+ \|c_n^{k+1}\|_2^2 + \beta \|\tilde{w}_n \odot \tilde{c}_n^{k+1}\|_2^2 + \sum_{i=1}^{n-1} \|w_i \odot \tilde{c}_i^{k+1}\|_2^2.
\end{align*}
\]

(18)

According to Algorithm 1, \(\lambda(\tilde{Z}_n^{k+1})\) is the optimal solution for

\[
\begin{align*}
\min_{\tilde{Z}_n^{k+1}} \lambda \|Y_n^{k} - \tilde{x}_n \tilde{c}_n^{k+1}\|_F^2 + \|\tilde{z}_n\|_2^2 + \beta \|\tilde{w}_n \odot \tilde{c}_n^{k+1}\|_2^2.
\end{align*}
\]

(19)

Hence,

\[
\begin{align*}
\lambda \|Y_n^{k} - \tilde{x}_n \tilde{c}_n^{k+1}\|_F^2 + \|\tilde{z}_n\|_2^2 + \beta \|\tilde{w}_n \odot \tilde{c}_n^{k+1}\|_2^2 \\
\leq \lambda \|Y_n^{k} - \tilde{x}_n c_n^{k+1}\|_F^2 + \|\tilde{z}_n\|_2^2 + \beta \|W \odot \tilde{c}_n^{k+1}\|_2^2.
\end{align*}
\]

(20)

We can get

\[
\begin{align*}
\lambda \|Y_n^{k} - \tilde{x}_n c_n^{k+1}\|_F^2 + \sum_{i=1}^{n-1} \|\tilde{z}_i c_i^{k+1}\|_2^2 + \|\tilde{z}_n\|_2^2 + \beta \sum_{i=1}^{n-1} \|w_i \odot c_i^{k+1}\|_2^2 \\
+ \beta \|w_n \odot c_n^{k+1}\|_2^2 + \beta \sum_{i=1}^{n-1} \|w_i \odot c_i^{k+1}\|_2^2. \\
= \lambda \|X - Z\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot \tilde{Z}_n^{k+1}\|_2^2.
\end{align*}
\]

(21)

According to the Eq. (18) and Eq. (21), we can obtain

\[
\begin{align*}
\lambda \|X - XZ^{k+1}\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot Z^{k+1}\|_2^2 \\
\leq \lambda \|X - Z\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot \tilde{Z}_n^{k+1}\|_2^2.
\end{align*}
\]

(22)

Using similar deduction, we can get

\[
\begin{align*}
\lambda \|X - X\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot \tilde{Z}_i^{k+1}\|_2^2 \\
\leq \lambda \|X - XZ^{k+1}\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot \tilde{Z}_i^{k+1}\|_2^2.
\end{align*}
\]

(23)

\[
\begin{align*}
\lambda \|X - X\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot \tilde{Z}_i^{k+1}\|_2^2 \\
\leq \lambda \|X - XZ^{k+1}\|_F^2 + \|Z^{k+1}\|_2^2 + \beta \|W \odot Z^{k+1}\|_2^2.
\end{align*}
\]

(24)

The proof is completed by the results in Eq. (22), (23), and (24).
REFERENCES

[1] S. Rusinkiewicz and M. Levoy, “QSplit: A multiresolution point rendering system for large meshes,” in Proc. Annu. Conf. Comput. Graph. Interact. Techn., 2000, pp. 343–352.

[2] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, “A survey of surface reconstruction from point clouds,” Comput. Graph. Forum, vol. 36, no. 1, pp. 301–329, Jan. 2017.

[3] C. R. Qi, H. Yi, L. Su, and L. J. Guibas, “PointNet++: Deep hierarchical califecture learning on point sets in a metric space,” in Proc. Neural Inf. Process. Syst., 2017, pp. 5099–5108.

[4] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, and J. Yang, “Pattern-affinitive propagation across depth, surface normal and semantic segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4106–4115.

[5] A. Boulch and R. Marlet, “Fast and robust normal estimation for point clouds with sharp features,” Comput. Graph. Forum, vol. 31, no. 5, pp. 1765–1774, Aug. 2012.

[6] J. Zhang, J. Cao, X. Liu, H. Chen, B. Li, and L. Liu, “Multi-normal estimation via pair consistency voting,” IEEE Trans. Vis. Comput. Graphics, vol. 25, no. 4, pp. 1693–1706, Apr. 2019.

[7] J. Cao, H. Chen, J. Zhang, Y. Li, X. Liu, and C. Zou, “Normal estimation via shifted neighborhood for point cloud,” J. Comput. Appl. Math., vol. 329, pp. 57–67, Feb. 2018.

[8] N. Amenta and M. Bern, “Surface reconstruction by Voronoi filtering,” Discrete Comput. Geometry, vol. 22, no. 4, pp. 481–504, Dec. 1999.

[9] D. Ouyang and H.-Y. Feng, “On the normal vector estimation for point cloud data from smooth surfaces,” Comput.-Aided Des., vol. 37, no. 10, pp. 1071–1079, Sep. 2005.

[10] T. K. Dey and S. Goswami, “Provable surface reconstruction from noisy samples,” Comput. Geometry, vol. 35, nos. 1–2, pp. 124–141, Aug. 2006.

[11] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, and S. Jin, “Robust normal estimation for point clouds with sharp features,” Comput. Graph., vol. 34, no. 2, pp. 94–106, Apr. 2010.

[12] A. C. Oztireli, G. Guennebaud, and M. Gross, “Feature preserving point set surfaces based on non-linear kernel regression,” Comput. Graph. Forum, vol. 28, no. 2, pp. 493–501, Apr. 2009.

[13] T. R. Jones, F. Durand, and M. Zwicker, “Normal improvement for point rendering,” IEEE Comput. Graph. Appl., vol. 24, no. 4, pp. 53–56, Jul. 2004.

[14] F. Calderón, U. Ruiz, and M. Rivera, “Surface-normal estimation with neighborhood reorganization for 3D reconstruction,” in Proc. Prog. Pattern Recognit. Image Anal. Appl., 2007, pp. 321–330.

[15] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, “Point set surfaces,” in Proc. IEEE Vis., Oct. 2001, pp. 21–28.

[16] H. Hoppe, T. DeRose, T. Duchamp, J. A. McDonald, and W. Stuetzle, “Surface reconstruction from unorganized points,” in Proc. Annu. Conf. Comput. Graph. Interact. Techn., 1992, pp. 71–78.

[17] N. J. Mitra, A. Nguyen, and L. Guibas, “Estimating surface normals in noisy point cloud data,” Int. J. Comput. Geometry Appl., vol. 14, nos. 4–5, pp. 261–276, Oct. 2004.

[18] M. Pauly, R. Keiser, L. Kobelt, and M. H. Gross, “Shape modeling with Liaoning Normal University, Dalian, China. Her research interests include geometric processing, machine learning, and deep learning.

[19] JIE ZHANG received the Ph.D. degree in computational mathematics from the Dalian University of Technology, Dalian, China, in 2015. She is currently a Lecturer with the School of Mathematics, Liaoning Normal University, Dalian. She has published over 20 papers in reputed journals/conferences. Her current research interests include geometric processing, machine learning, and deep learning.

[20] JIAHUI DUAN received the bachelor’s degree in mathematics and applied mathematics from Bohai University, Jinzhou, China. She is currently pursuing the master’s degree in computational mathematics with Liaoning Normal University, Dalian, China. Her research interests include geometric processing and machine learning.
KEWEI TANG received the B.Sc. degree in mathematics from Liaoning Normal University, Dalian, China, in 2008, and the M.Sc. and Ph.D. degrees in mathematics from the Dalian University of Technology, Dalian, in 2011 and 2015, respectively. He was a Visiting Scholar with the Department of Statistical Science, Duke University, from 2013 to 2014. He is currently an Associate Professor with the School of Mathematics, Liaoning Normal University. He has authored over 20 refereed articles. His research interests include computer vision, machine learning, and so on.

JUNJIE CAO received the B.Sc. and Ph.D. degrees from the Dalian University of Technology, Dalian, China, in 2003 and 2010, respectively. From 2014 to 2015, he was with Simon Fraser University, Canada, as an Academic Visit. He is currently an Associate Professor with the School of Mathematical Sciences, Dalian University of Technology. He has published over 40 papers in reputed journals/conferences. His research interests include geometric processing and image processing.

XIUPING LIU received the B.Sc. degree from Jilin University, Changchun, China, in 1990, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 1999. From 1999 to 2001, she conducted Research as a Postdoctoral Scholar with the School of Mathematics, Sun Yat-sen University, Guangzhou, China. She is currently a Professor with the Dalian University of Technology. She is the author of two books and more than 70 articles. Her research interests include geometric processing, deep learning, and shape modeling and analyzing.