Geometry and Learning Co-supported Normal Estimation for Unstructured Point Cloud

Haoran Zhou[1,2*] Honghua Chen[1*] Yidan Feng[1,2] Qiong Wang[3]
Jing Qin[4] Haoran Xie[5] Fu Lee Wang[6] Mingqiang Wei[1,2†] Jun Wang[1†]
[1]Nanjing University of Aeronautics and Astronautics
[2]MIIT Key Laboratory of Pattern Analysis and Machine Intelligence
[3]Shenzhen Institutes of Advanced Technology [4]The Hong Kong Polytechnic University
[5]Lingnan University [6]The Open University of Hong Kong

Abstract

In this paper, we propose a normal estimation method for unstructured point cloud. We observe that geometric estimators commonly focus more on feature preservation but are hard to tune parameters and sensitive to noise, while learning-based approaches pursue an overall normal estimation accuracy but cannot well handle challenging regions such as surface edges. This paper presents a novel normal estimation method, under the co-support of geometric estimator and deep learning. To lowering the learning difficulty, we first propose to compute a suboptimal initial normal at each point by searching for a best fitting patch. Based on the computed normal field, we design a normal-based height map network (NH-Net) to fine-tune the suboptimal normals. Qualitative and quantitative evaluations demonstrate the clear improvements of our results over both traditional methods and learning-based methods, in terms of estimation accuracy and feature recovery.

1. Introduction

Various kinds of 3D laser scanners and depth cameras have emerged in recent decades, making point clouds move into the focus of many practical applications, such as robotic grasping [20], 3D reconstruction [11], and autonomous driving [24]. Commonly, the scanned point clouds only contain points' spatial location information associated with the noise, incompleteness and sampling irregularity, while lacking local surface geometry properties, like point normals. Quality normals can facilitate a huge amount of downstream tasks, for example, point cloud consolidation [10], surface reconstruction [13], and model segmentation [8]. Hence, estimating normals is an inevitable and crucial task for unstructured point cloud.

The problem of normal estimation has been extensively researched, yet not well-solved. We can roughly divide existing normal estimation techniques into two categories: traditional methods and learning-based methods. Traditional ones usually utilize several elaborately-designed regularities to preserve/recover sharp features, while learning-based methods pursue to learn a general mapping from noisy inputs to ground truths. However, no existing algorithm can serve as a normal estimation panacea: 1) Traditional methods always heavily rely on parameters tuning, like the neighborhood scale for plane fitting [31, 32]; 2) Learning-based techniques, either using the convolutional neural networks (CNN) architecture [6, 4], or the PointNet architecture [15, 33], are both limited by the ability of feature representation. It is, therefore, hard to learn a straightforward mapping from severely degraded inputs to the ground-truth normals, especially in sharp feature regions.

Motivated by these challenges, in this work, we propose a two-stage normal estimation method for unstructured 3D point clouds. The key idea of our approach is to solve the ill-posed normal estimation problem via two sub-steps: 1) computing a suboptimal intermediate normal field with features as well-preserved as possible, by a geometric estimator; 2) formulating the final normal recovery procedure as a regression function that maps the intermediate normal results to their ground truths. In detail, for lowering the difficulty of normal learning in challenging regions, we present a multi-scale fitting patch selection (MFPS) scheme to help estimate a suboptimal normal field, which contributes more on the feature preservation. Since in some challenging regions, where parameters are hard to be tuned, the initial normals are still imperfect, we then design a normal-based height map network (NH-Net), which utilizes both the above estimated normals and the local surface information to obtain the final optimal normals. We

*Co-first authors
†Co-corresponding authors (mqwei/wjun@nuaa.edu.cn)

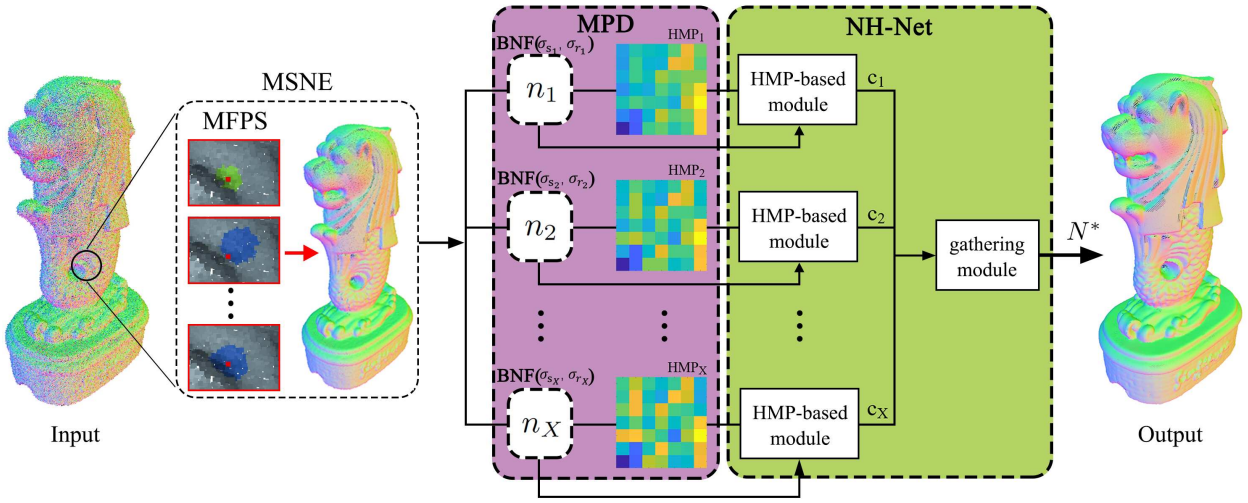


Figure 1. The pipeline of our normal estimation method. We first compute the suboptimal normal at each point via multi-scale fitting patch selection (MFPS). Then, a multi-scale point descriptor is constructed based on bilateral normal filters (BNF) and local height-map patches (HMPs). Our NH-Net, consisting of an HMP-based module and a gathering module, receives MPDs and produces the final normal.

experimentally prove that the combination of the geometric estimator scheme and the learning-based recovery scheme outperforms either of them.

Our main contributions are three-fold:

• We design a two-stage normal estimation method by collaborating geometric estimator and deep neural network, which shows clear improvements over the state-of-the-arts.

• We propose a multi-scale fitting patch selection scheme, which can produce a feature-preserving initial normal field as an input of the following recovery network.

• We propose a normal refining network (NH-Net), which is able to compensate the imperfection of the suboptimal normal results computed in the first step.

2. Related work

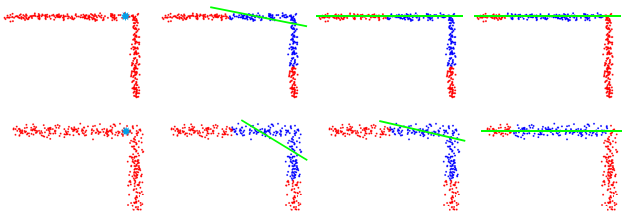
Normal estimation for point clouds is a long-standing problem in academic. We will review previous researches from traditional normal estimators to recent prevalent learning-based techniques.

2.1. Traditional normal estimator

The simplest and best-known method for normal estimation is based on the Principal Component Analysis (PCA) [17], by analyzing the covariance in a local structure around a point and defining the normal as the eigenvector corresponding to the smallest eigenvalue. Following this work, a lot of variants have been proposed [23, 7, 14]. In particular, Mitra et al. [23] analyzed the effects of neighborhood size, curvature, sampling density, and noise for estimating normals. Another kind of normal estimation method is based on Voronoi cells [2, 12, 1, 22]. However, this kind of method cannot well estimate the normals of points near/on sharp features. Based on the observation that the neighbors belonging to different surface patches should be discarded, recent works dedicated to selecting a plane approximating the neighbors from the same surface patch to estimate normals [19, 32, 30, 31]. Under the assumption that surfaces commonly are composed of piecewise flat patches, sparsity-based methods [3, 28, 9] show impressive results, especially in sharp feature preservation. Some other methods like Hough Transform [5] also yield pleasing results.

2.2. Learning-based normal estimator

Recently, learning-based methods gradually show its power for normal estimation. Boulch et al. [6] proposed to project a discretized Hough space representing normal directions onto a structure amenable to CNN-based deep learning. Roveri et al. [26] defined a grid-like regular input to the CNN to learn ideal normal results. Ben-Shabat et al. [4] presented a method that approximates the local normal vector using a point-wise, multi-scale 3D modified Fisher Vector representation which serves as an input to a deep 3D CNN architecture. In addition, they learn the neighborhood size that minimizes the normal estimation error using a mixture of experts. The key of the three methods lies in parameterizing the unstructured point cloud into a regular domain for directly applying the CNN architecture. Another point cloud learning framework, namely PointNet [25], becomes very popular in 3D domain, since it can directly learn features from data of points. Inspired by it, Guerrero et al. [15] proposed a unified method for estimating normals and principal curvature values in noisy point clouds. This approach is based on a modification of the PointNet architecture, in



(a) Noisy input (b) RNE [19] (c) PCV [31] (d) Ours

Figure 2. Comparison of the selected planes by different methods. From the left column to the rightmost: Noisy input, and the plane fitting results of RNE, PCV and our method. Red points represent the noisy input and blue star in the first column is the target point. The green line means the selected plane and the blue points are neighbors considered to fit this plane in the corresponding method. We can easily observe that our method can select the most suitable plane, with the help of better neighboring information.

which they place special emphasis on extracting local properties of a patch around a given central point. By leveraging both the PointNet and 3DCNN, Hashimoto et al. [16] proposed a joint network that can accurately infer normal vectors from a point cloud. Based on the PointNet architecture, Zhou et al. [33] introduced an extra feature constraint mechanism and a multi-scale neighborhood selection strategy to estimate normals for 3D point clouds.

3. Overview

We propose a two-step normal estimation method via the collaboration of a geometric estimator and a deep network called NH-Net. Fig. 1 shows an overview of the proposed approach. It consists of the following two main steps: multi-scale suboptimal normal estimation (MSNE) and NH-Net. First, we propose a multi-scale fitting patch selection (MFPS) for each candidate point (near sharp feature) to compute its suboptimal normal, while obtaining the suboptimal normals for smooth points (far from sharp features) via PCA (Sec. 4). Then, based on these computed normals, we define a multi-scale point descriptor (MPD) via bilateral normal filter (BNF) for each point as the input of the following network (Sec. 5.1). Ultimately, we recover the final optimal normals by our NH-Net, containing two sub-modules, namely a height-map patch (HMP) based improving module and a gathering module (Sec. 5.2).

4. Multi-scale suboptimal normal estimation

Before computing the initial normal for each point, we first apply the local covariance analysis (Sec. 4.2 in [32]), to classify all points into candidate points (near sharp features) and smooth points (far from sharp features). The normals of smooth points can be simply computed by PCA.

To estimate the normals for candidante points, one common strategy is to randomly select three non-collinear points to construct a set of candidate planes and pick up one that best describes the underlying surface patch. Typical methods are RNE [19] and PCV [31], in which they introduce a residual bandwidth to evaluate proximity of the candidate planes to the underlying surface. However, when they come to noisy inputs, the neighborhood of the target point for detecting candidate plane is often corrupted by points from another side of the intersection. This causes the selected plane deviating from the true surface (Fig. 2(b)). PCV is better but still imperfect (Fig. 2(c)).

4.1. Fitting patch selection

Considering the problems stated above, we propose to select a more consistent and flexible neighborhood which provides better local structure information for robust normal estimation. For each candidate point p_i, our method tries to find the best fitting patch that contains p_i. Please note normals of smooth points are already computed by PCA.

First, for each point p_j in a point cloud, we define a local patch Q_j (the K-nearest neighboring points of p_j). Then, the fitting plane $\theta^*_{Q_j}$ of each Q_j is determined by the following objective function:

$$E_{Q_j}(\theta) = \frac{1}{|Q_j|} \sum_{p_j^k \in Q_j} W_{\sigma_j}(p_j^k, \theta), \tag{1}$$

$$\theta^*_{Q_j} = \arg\max_{\theta} E_{Q_j}(\theta), \tag{2}$$

where $W_{\sigma_j}(p_j^k, \theta) = exp(-r^2_{k,\theta}/\sigma_j^2)$ is the Gaussian function in which $r_{k,\theta}$ denotes the distance from point p_j^k to the plane θ, and σ_j is the residual bandwidth of p_j.

Denote $S_i = \{Q_j | p_i \in Q_j\}$ as all patches containing candidate point p_i. The fitting patch selection process for p_i is performed completely within S_i by measuring the consistency of these patches to the target point:

$$\mathcal{D}_i = \max_{Q_j \in S_i} E_{Q_j}(\theta^*_{Q_j}) w_{\sigma_i}(p_i, \theta^*_{Q_j}), \tag{3}$$

where

$$w_{\sigma_i}(p_i, \theta^*_{Q_j}) = exp(-r^2_{i,\theta^*_{Q_j}}/\sigma_i^2)$$

Here, $\theta^*_{Q_j}$ is the fitting plane of Q_j determined by Eq. 2, and $r_{i,\theta^*_{Q_j}}$ is the residual distance from p_i to plane $\theta^*_{Q_j}$. The larger $E_{Q_j}(\theta^*_{Q_j})$ is, the smaller the plane fitting error is, and $w_{\sigma_i}(p_i, \theta^*_{Q_j})$ is introduced to avoid choosing a plane far from the target point. Therefore, for each candidate patch $Q \in S_i$, we compute the consistency function using its fitting plane θ^*_Q, and pick the one maxmizing $\mathcal{D}_i$ as the selected fitting patch.

4.2. Multi-scale scheme

The scale parameter K is the most practical parameter in the above process. However, a fixed neighborhood size

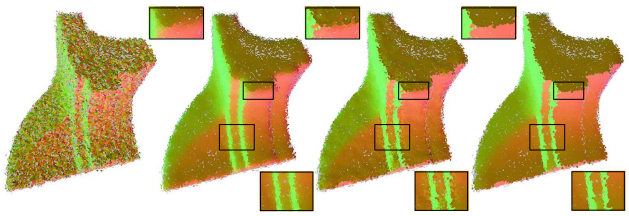


Figure 3. Visual comparison of estimated normals on noisy fandisk model (without point update). From left to right: noisy input, our NH-Net trained on PCA normals, suboptimal normals without the network, and our NH-Net trained on suboptimal normals.

would be incapable for a robust estimation especially for point clouds with high density variation. Therefore, we further improve the suboptimal normal estimation via a multi-scale scheme. Rather than applying the previous patch selection within a single size, we formulate K as a set containing multiple patch sizes, *i.e.*, $K = \{K_1, K_2, ..., K_a\}$. Thus, at point p_j, every patch Q_j^t with point size $K_t \in K$ is taken into consideration, and we first compute its fitting plane $\theta^*_{Q_j^t}$ via Eq. 2.

As a result, the final MFPS estimator is completely the same as Sec. 4.1 expect that Eq. 3 is modified as follows:

$$\mathcal{D}_i = \max_{Q_j^t \in S_i} E_{Q_j^t}(\theta^*_{Q_j^t}) w_{\sigma_i}(p_i, \theta^*_{Q_j^t}) \eta(K_t). \tag{4}$$

where

$$\eta(K_t) = \alpha + (1-\alpha)\frac{K_t - K_{min}}{K_{max} - K_{min}}$$

$\eta(K_t)$ is a tradeoff parameter used to punish relately small patches because larger ones are prefered in noisy areas. We set $\alpha = 0.9$ empirically.

As shown in Fig. 1 (the MSNE part), the selected patch is not always the neighborhood whose center is located at target point p_i. Our MFPS estimator tends to search for, among all patches in S_i, the most isotropic one and we use the normal of the fitting plane of this patch as the suboptimal normal of p_i. The multi-scale approach makes more candidate patches avaliable in S_i in order to promote the robustness to strong noise and details (see from Fig. 2(d)), in which the scale parameter is set as $K = 50, 100, 150$ by default in our experiments. More implementation details are available in the supplemental material.

The output suboptimal normals will be used in the following NH-Net which can preserve geometric features as shown in the third sub-figure of Fig. 3. These initial normals enhance the performance of the network compared with directly using raw normals (PCA) as input, see Fig. 3.

5. NH-Net

In this section, we introduce the learning part in our algorithm pipeline. The network architecture is outlined in Fig. 1 (the purple and green parts). First, we introduce a multi-scale descriptor, called MPD, and then explain how to feed it into the next two-stage learning module which spits out the final normal. The details are followed.

5.1. MPD: multi-scale point descriptor

As known, the input data of a neural network should be in a structural format. Inspired by the work in Wang et al. [29], for each point, we define its MPD by X pairs of filtered normal plus local HMP to collaboratively enhance the learning effect. Filtered normals at each point are computed by bilateral filtering based on the normals estimated in Sec. 4, and the HMPs are constructed according to the corresponding filtered normals.

Multi-scale bilateral filters: We give a brief introduction to the commonly-used bilateral filters [18]. Denote the neighborhood of p_i as N_i, and the normal at each point as n_i^{sub}. The filtered normal can be defined as:

$$n_i = \Lambda(\sum_{p_j \in N_i} W_s(||p_i - p_j||) W_r(||n_i^{sub} - n_j^{sub}||) n_j^{sub}), \tag{5}$$

where $\Lambda(*)$ is the vector normalization function, and W_s and W_r are Gaussian weight functions, *i.e.*, $W_\sigma(x) = exp(-x^2/(2\sigma^2))$, representing the spatial similarity and normal similarity recepectively between a pair of points. Standard deviations σ_s and σ_r used in W_s and W_r are parameters that need to be carefully tuned. Given two parameter sets $P_s = \{\sigma_{s_1}, \sigma_{s_2}, ...\}$ and $P_r = \{\sigma_{r_1}, \sigma_{r_2}, ...\}$, we can obtain several filtered normals using every combination of these two parameters. Together with the suboptimal one, we get a total of X normals in the normal set of p_i, denoted as $N_f = \{n_1, n_2, ..., n_X\}$.

Height map patch construction: The point p_i itself as well as one of its filtered normals n_t defines a tangent plane, and the associated HMP is built on this plane. Suppose that we build a matrix with $m \times m$ bins that describes the local point position, and the center of the matrix is located at p_i. Similar as [10], we fill each bin by weightedly averaging the height distances of points in a ball neighborhood of bin center b_j:

$$v_{b_j} = \frac{\sum_{p_k \in N_{ball}(b_j)} w(b_j, p_k) H(T(p_i, n_t), p_k)}{\sum_{p_k} w(b_j, p_k)}, \tag{6}$$

where $T(p_i, n_t)$ is the tangent plane defined by p_i and n_t, $H(T(p_i, n_t), p_k)$ returns the signed distance from p_k to the plane and $w(b_j, p_k) = exp(-\frac{||b_j - p_k||^2}{\sigma_d^2})$ is a spatial Gaussian weight function in which σ_d is the residual bandwidth. Assembling all pairs of filtered normals and HMPs yields each point's MPD:

$$\mathbf{MPD}_i = \{(n_1, \mathbf{HMP}_1), (n_2, \mathbf{HMP}_2), ..., (n_X, \mathbf{HMP}_X)\}. \tag{7}$$

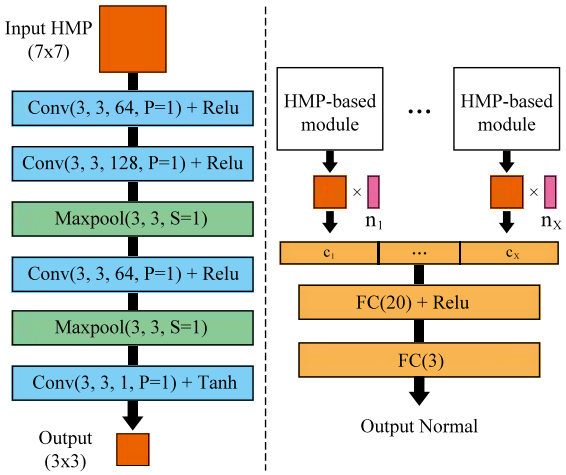


Figure 4. The branch of HMP-based improving module (left), and the gathering module (right).

Consistency regardless of orientation: To keep our MPD invariant to rotation, we need to take both normal invariance and HMP invariance into consideration. First, we apply a global rotation to the normals component of $\mathbf{MPD}_i$ to make it invariant to rigid transformation. Specifically, we construct the rotation matrix R_i by computing the normal tensor $T_i = \sum_{j=1}^{X} n_j \times n_j^T, n_j \in N_f$. The matrix R_i, defined by three eigenvectors (sorted by eigenvalues) of T_i, rotates all normals in N_f to the Z-axis. Further, if one normal is in the direction of negative Z-axis, we reverse it to the positive one. The ground-truth normal is also applied by the same rotation matrix. For the HMPs component of $\mathbf{MPD}_i$, their Z-axes of local coordinate systems are also rotated by the matrix R_i. The X-axis and Y-axis are computed by the cross product between the rotated normal and the two eigenvectors with smallest two eigenvalues.

5.2. NH-Net architecture

Our network architecture is outlined in Fig. 1 (the green part), which takes both normals and HMPs as input. First, we apply the cluster-based approach in [29] to promote learning (see the detailed evaluation in the supplemental material). Our network consists of two modules: an HMP-based improving module and a gathering module. The key idea is using HMP to learn a 3×3 transformation matrix, which is used to consume the input normal and connect the two modules together. We then gather the separate branches of improving module, and output the final normal.

HMP-based improving module: The HMP alone serves as the input at the start of the whole network. The improving module contains X separate branches corresponding to the X pairs in MPD, and each receives a single HMP and processes it using several convolutional layers and maxpool layers, outputting a 3×3 matrix R_t^j. Details of this module are shown in Fig. 4 (the left part). The output matrix is regarded as a transformation matrix and will be used in the gathering module. This design of identical branches is chosen to better preserve local features associated with the varying parameters of previous bilateral filters.

Gathering module: After obtaining outputs $\{R_t^j\}_{j=1}^{X}$ from the improving modules, our network applies each 3×3 matrix to its corresponding normal and gets a new vector $c_j = R_t^j n_j$. The standardization version of these vectors is fed to the gathering module. This module is a single-layer fully connected network which outputs the ultimate normal $N^* = (N_x, N_y, N_z)$.

Loss function: We train the network by minimizing the MSE loss between the output normal N^* and its ground truth $\hat{N}$:

$$Loss = ||\Lambda(N^*) - \hat{N}||^2 + \lambda E_{reg}, \tag{8}$$

Here, $\Lambda(*)$ is the vector normalization function, E_{reg} is the commonly-used L_2 regularization term to avoid overfitting and $\lambda = 0.02$ is used.

5.3. Training details

Training dataset: We use the dataset from [29] for training. The training set is built from synthetic triangular mesh models, including point samples and normals.

The ground-truth point clouds consist of 21 models including 6 CAD models, 8 smooth models and 7 feature-rich models. To generate the noisy inputs, we introduce Gaussian noise for each point cloud with a standard deviation of 0.1%, 0.2% and 0.3% of the diagonal length of the bounding box. The final training dataset contains 1.5M points from 63 noisy point clouds.

Parameters: When constructing MPDs, the parameter pairs used in bilateral filtering are set as $P_s = \{\bar{l}_d, 2\bar{l}_d\}$ and $P_r = \{0.1, 0.2, 0.35, 0.5\}$, where $\bar{l}_d$ is the average distance between points. Thus, together with the suboptimal normal, MPD includes 9 pairs of normal and HMP, *i.e.*, there are 9 separate branches in the HMP-based improving module. For the HMP construction, we use a 7×7 ($m = 7$) height-map grid. We train the network using pytorch on a single NVIDIA Geforce RTX 2080 Ti GPU.

6. Experiments and results

6.1. Benchmark for synthetic models

To quantitatively testify our method, we use the mean angular errors (degrees) between estimated normals and their ground-truth counterparts as an evaluation metric, which is typically used for normal accuracy comparison.

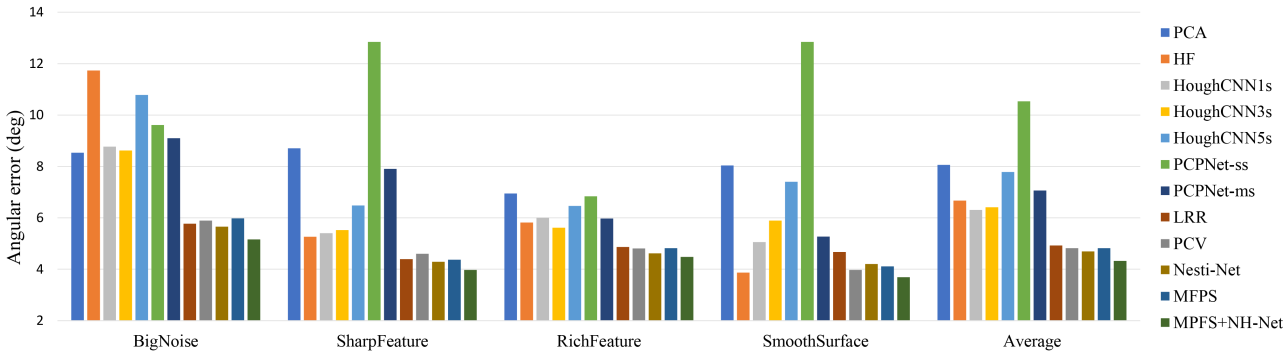
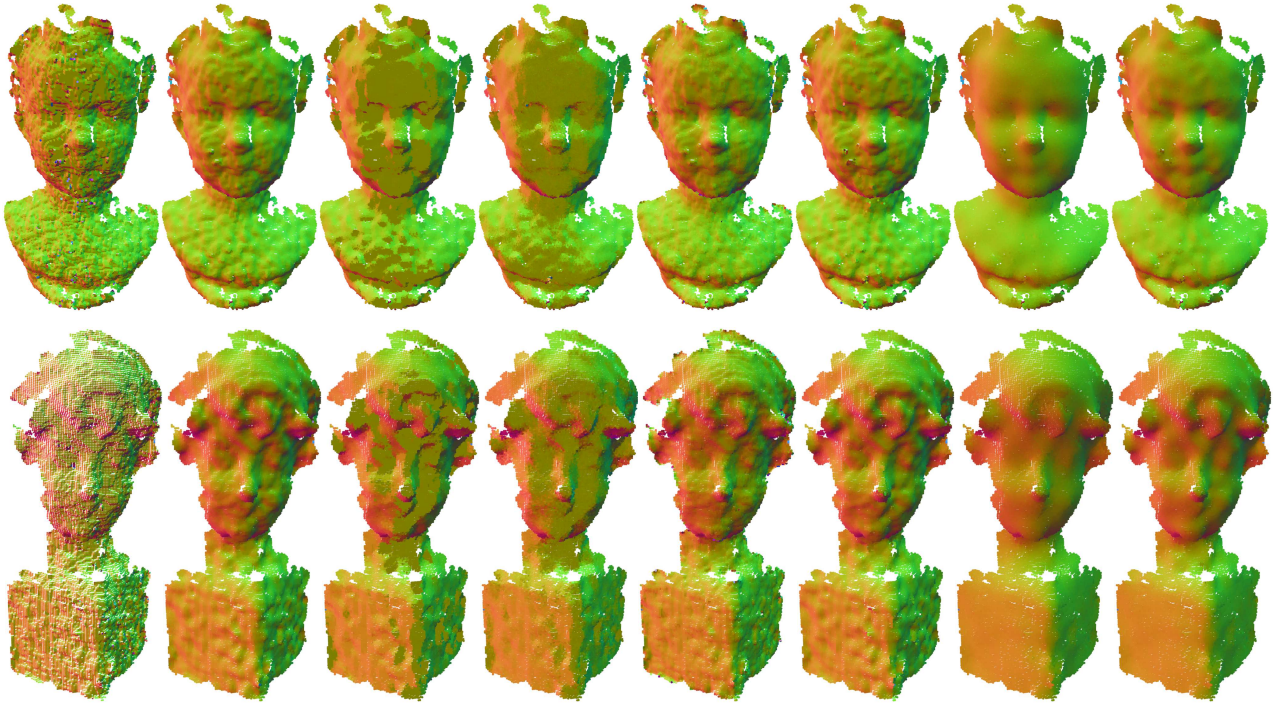


Figure 5. The average angular errors (degree) of compared methods on the synthetic benchmark dataset.



(a) Input (b) PCA [17] (c) HF [5] (d) HoughCNN [6] (e) PCPNet [15] (f) PCV [31] (g) Nesti-Net [4] (h) Ours

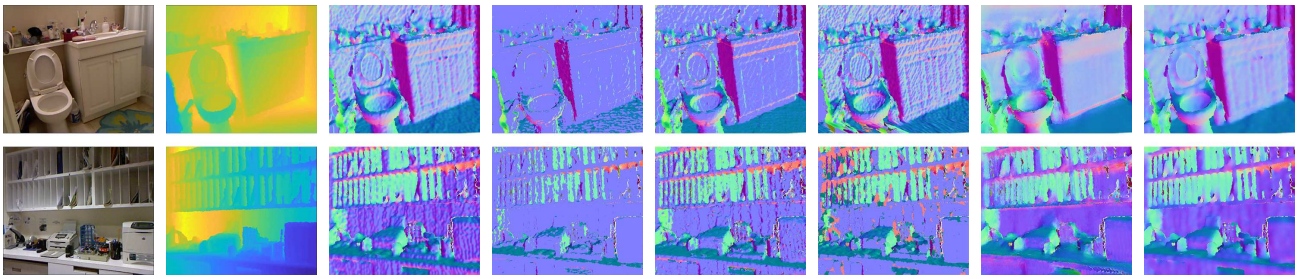
Figure 6. Visual comparison of estimated normals by different methods on two real scanned models from [29].

We collect a benchmark dataset mainly from the testsets of [29] and [31] which are given as synthetic mesh models. The dataset includes 4 categories: SharpFeature, RichFeature, SmoothSurface and BigNoise category, in which there are 11, 8, 8 and 8 models respectively. Each category contains point clouds with challenging geometric features and varying sampling density. For data augmentation, each point cloud in the SharpFeature and SmoothSurface categories is perturbed by Gaussian noise with a standard deviation of 0.05%, 0.1% and 0.15% of the diagonal length of the bounding box. For the RichFeaure category, 0.05%, 0.1%, 0.15% and 0.2% noise are introduced and 0.2%, 0.3%, 0.4% and 0.5% for the BigNoise category. Our benchmark dataset contains totally 121 point clouds.

6.2. Experiments on synthetic data

We compare our method, including MFPS alone and the full pipeline (MFPS+NH-Net), with several state-of-the-art methods: PCA [17], HF [5], LRR [32], PCV [31], HoughCNN [6], PCPNet [15] and Nesti-Net [4]. We re-trained HoughCNN, PCPNet and Nesti-Net on our own dataset. In addition, HoughCNN has three versions with different scales and PCPNet has single-scale and multi-scale versions. We evaluate all these versions.

We compare the above methods on our synthetic benchmark by setting the same neighborhood size $K = 100$ for three categories: SharpFeature, RichFeature and SmoothSurface. In particular, the HoughCNN3s considers 3 scales,



(a) RGB image (b) Depth image (c) PCA [17] (d) HF [5] (e) HoughCNN [6] (f) PCV [31] (g) Nesti-Net [4] (h) Ours

Figure 7. Visual comparison of normal estimation results on the scanned point clouds from the NYU Depth V2 dataset [27].

Model	PCA [17]		HF [5]		HoughCNN [6]		PCPNet [15]		PCV [31]		Nesti-Net [4]		Ours	
	Error	Time	Error	Time	Error	Time	Error	Time	Error	Time	Error	Time	Error	Time
Fig. 6 r1	11.5°	**2s**	14.2°	**2s**	14.6°	19s	11.6°	96s	11.1°	15s	9.0°	26s	**8.8°**	68s
Fig. 6 r2	12.3°	2s	15.6°	**1s**	15.4°	13s	13.5°	66s	12.4°	6s	11.1°	20s	**10.8°**	44s
Fig. 8 r1	18.7°	**1s**	9.4°	**1s**	9.2°	6s	18.7°	20s	6.5°	38s	6.8°	58s	**4.5°**	23s
Fig. 8 r3	6.7°	21s	5.4°	**13s**	5.8°	119s	6.1°	609s	5.2°	109s	5.0°	1248s	**4.5°**	312s

Table 1. Comparison of the errors and timings of normal estimation methods. r1, r2 and r3 represent the first, second and third rows.

$K = 50, 100, 200$, which is recommended in their paper. For our multi-scale method, we set the neighboring size set as $K = 50, 100, 150$ in MFPS. For the BigNoise category, we double the neighborhood size, *i.e.*, $K = 100, 200, 400$ for HoughCNN3s, $K = 100, 200, 300$ for ours, and $K = 200$ for other single scale methods. Besides, HoughCNN5s considers 5 scales, $K = 32, 64, 128, 256, 512$, for all categories. All other parameters are set by default values.

The results are illustrated via a bar chart in Fig. 5. Our method outperforms others in all categories, especially on models with heavy-level noise.

6.3. Experiments on real scans

Our method is also tested on real point cloud dataset scanned by Microsoft Kinect v1 from [29]. We show visual comparisons of estimated normals rendered in RGB color images in Fig. 6. The real scan data reveals more challenges, such as the fluctuation on flat surface, originated from the projection process of Kinect camera. The non-Gaussian and discontinuous noise interferes the estimator against recovering the underlying surface. By training on such scanned data (from [29]), our NH-Net is able to distinguish geometric features from undesired scanning noise. Meanwhile, as shown in Tab. 1, our method performs better than all other methods in terms of the normal accuracy.

Furthermore, we show qualitative results of several real scans of indoor scenes provided by NYU Depth V2 [27]. From Fig. 7, PCA, HF, HoughCNN and PCV, can preserve tiny details but with the price of retaining noise (see from Fig. 7(c) to 7(f)). Nesti-Net can well smooth the noisy surface, but over-smooths geometric features (Fig. 7(g)). Please note we train all networks by the same dataset for a fair comparison.

We also report the time performance of our method and the compared methods in Tab. 1. The running time of our approach is the total testing time including all steps.

6.4. Experiments on different learning schemes

To testify the effectiveness of our two-module network, we compare our proposed learning scheme with another three variants of our method. Scheme 1 is training only on the HMP constructed by the suboptimal normal and outputting a transformation matrix to refine normals. Scheme 2 is training on the HMPs constructed by the filtered normals but without a gathering module. Scheme 3 is training only using the filtered normals without the constructed HMPs. Additionally, we also report the errors of suboptimal normal and the average normal of filtered normals. Tab. 2 summarizes the total errors on the synthetic benchmark of these normal estimation results. We can see that by the combination of the filtered normals and local structure information, our proposed scheme can generate a more reliable result with the highest accuracy. More details are included in the supplemental material.

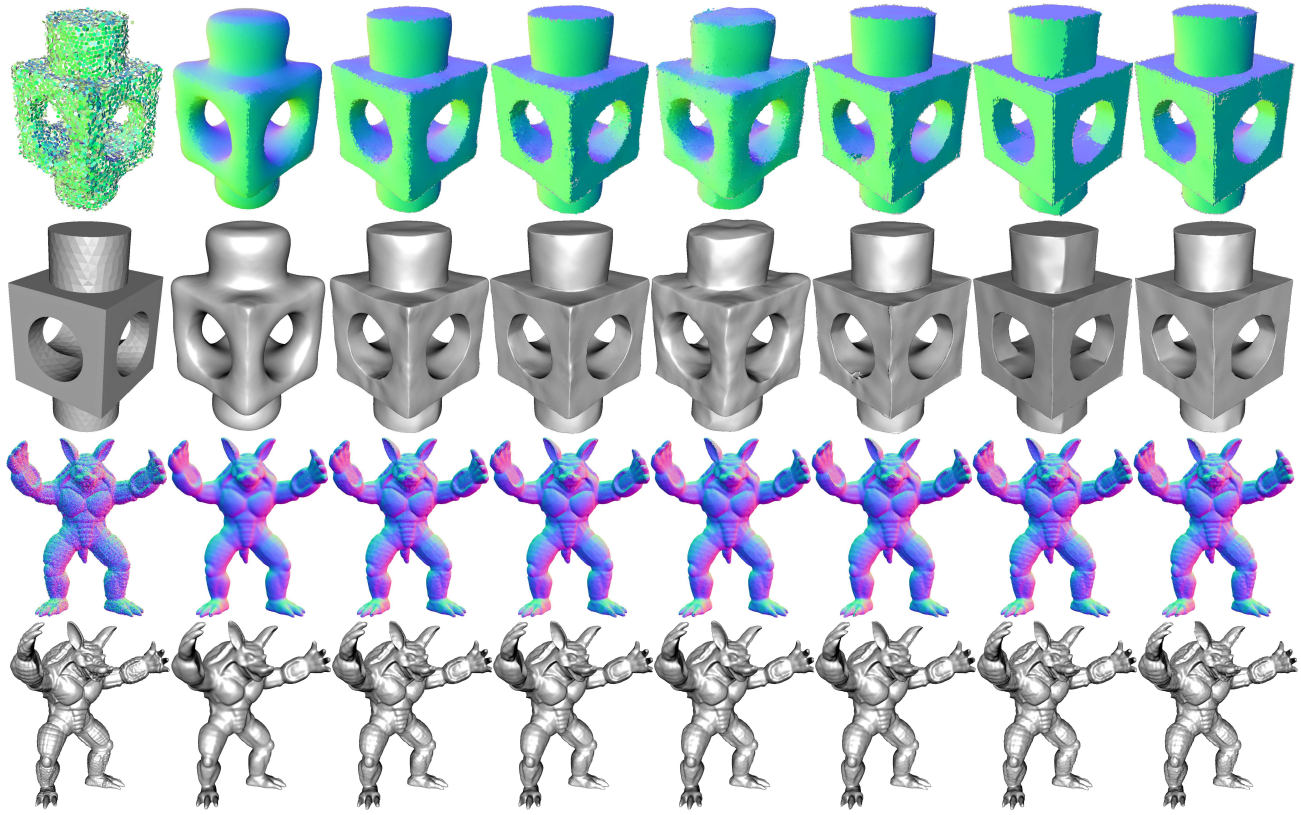
7. Application

As known, well-estimated normals can boost the effect of many point cloud processing tasks, like denoising and surface reconstruction. To further verify the advantages of our method, we show denoising results using the same point updating algorithm, yet based on the normal results estimated by different methods.

Denote that $\mathcal{N}_i$ is the ball neighborhood of p_i, we here provide an algorithm to update point positions under the guidance of estimated normals, which is able to remove

Method	suboptimal	averaging filtered normals	scheme 1	scheme 2	scheme 3	Ours
Error	4.81°	5.01°	4.52°	4.61°	4.48°	**4.32°**

Table 2. Comparison of normal estimation errors on synthetic benchmark dataset of different final normal computing schemes.



(a) Noisy input & ground-truth mesh (b) PCA [17] (c) HF [5] (d) HoughCNN [6] (e) PCPNet [15] (f) PCV [31] (g) Nesti-Net [4] (h) Ours

Figure 8. Comparison of denoising results based on different normal estimation results.The first and third rows show the visual comparison of denoising results. The second and bottom rows are the corresponding reconstruction results.

noise and recover sharp features:

$$p_i' = p_i + \gamma_i \sum_{p_j \in \mathcal{N}_i} (p_j - p_i)(w_\sigma(n_i, n_j) n_i^T n_i + \lambda n_j^T n_j). \tag{9}$$

where $w_\sigma(n_i, n_j) = exp(-\frac{||n_i - n_j||^2}{\sigma^2})$ is a weight function, $\lambda = 0.5$ is a tradeoff parameter and γ_i is step size set to $\frac{1}{3|\mathcal{N}_i|}$ by default. To prevent points from accumulating around edges, we keep the neighboring information unchanged in all iterations [21]. The iteration number is set to 20 in our experiment.

We demonstrate visual quality of the denoising results and their reconstruction results (see from Fig. 8). Our normal results help produce the most faithful denoising and reconstruction results.

8. Conclusion

In this work, we try to overcome several challenges in normal estimation for 3D unstructured point clouds. We propose a multi-scale fitting patch selection scheme for suboptimal normal estimation. Then, a novel NH-Net, including two different learning modules, is devised to further improve the estimation above. The input of NH-net is a multi-scale geometric descriptors (MPD), composed by filtered normals and their corresponding HMPs. Our method, which is a collaboration of geometry prior-based and learning-based approaches, achieves impressive results relative to all other methods.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61502137), the Hong Kong Research Grants Council (No. PolyU 152035/17E), the HKIBS Research Seed Fund 2019/20 (No. 190-009), and the Research Seed Fund (No. 102367) of Lingnan University, Hong Kong.

References

[1] Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007*, pages 39–48, 2007. 2

[2] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999. 2

[3] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. L1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics (TOG)*, 29(5):135, 2010. 2

[4] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10112–10120, 2019. 1, 2, 6, 7, 8

[5] Alexandre Boulch and Renaud Marlet. Fast and robust normal estimation for point clouds with sharp features. In *Computer graphics forum*, volume 31, pages 1765–1774. Wiley Online Library, 2012. 2, 6, 7, 8

[6] Alexandre Boulch and Renaud Marlet. Deep learning for robust normal estimation in unstructured point clouds. In *Computer Graphics Forum*, volume 35, pages 281–290. Wiley Online Library, 2016. 1, 2, 6, 7, 8

[7] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. 2

[8] Erzhuo Che and Michael J Olsen. Multi-scan segmentation of terrestrial laser scanning data based on normal variation analysis. *ISPRS journal of photogrammetry and remote sensing*, 143:233–248, 2018. 1

[9] Honghua Chen, Jin Huang, Oussama Remil, Haoran Xie, Jing Qin, Yanwen Guo, Mingqiang Wei, and Jun Wang. Structure-guided shape-preserving mesh texture smoothing via joint low-rank matrix recovery. *Computer-Aided Design*, 115:122–134, 2019. 2

[10] Honghua Chen, Mingqiang Wei, Yangxing Sun, Xingyu Xie, and Jun Wang. Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint. *IEEE transactions on visualization and computer graphics*, 2019. 1, 4

[11] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1538–1547, 2019. 1

[12] Tamal K Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. *Computational Geometry*, 35(1-2):124–141, 2006. 2

[13] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3):544–552, 2005. 1

[14] Gaël Guennebaud and Markus H. Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3):23, 2007. 2

[15] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNet: Learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85, 2018. 1, 2, 6, 7, 8

[16] Taisuke Hashimoto and Masaki Saito. Normal estimation for accurate 3d mesh reconstruction with point cloud model incorporating spatial structure. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 54–63, 2019. 3

[17] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992. 2, 6, 7, 8

[18] Thouis R Jones, Fredo Durand, and Matthias Zwicker. Normal improvement for point rendering. *IEEE Computer Graphics and Applications*, 24(4):53–56, 2004. 4

[19] Bao Li, Ruwen Schnabel, Reinhard Klein, Zhiquan Cheng, Gang Dang, and Shiyao Jin. Robust normal estimation for point clouds with sharp features. *Computers & Graphics*, 34(2):94–106, 2010. 2, 3

[20] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. Pointnetgpd: Detecting grasp configurations from point sets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3629–3635. IEEE, 2019. 1

[21] Xuequan Lu, Scott Schaefer, Jun Luo, Lizhuang Ma, and Ying He. Low rank matrix approximation for geometry filtering. *arXiv preprint arXiv:1803.06783*, 2018. 8

[22] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):743–756, 2010. 2

[23] Niloy J. Mitra and An Thanh Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the 19th ACM Symposium on Computational Geometry, San Diego, CA, USA, June 8-10, 2003*, pages 322–328, 2003. 2

[24] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 1

[25] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 77–85, 2017. 2

[26] Riccardo Roveri, A. Cengiz Öztireli, Ioana Pandele, and Markus H. Gross. Pointpronets: Consolidation of point clouds with convolutional neural networks. *Comput. Graph. Forum*, 37(2):87–99, 2018. 2

[27] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 7

[28] Yujing Sun, Scott Schaefer, and Wenping Wang. Denoising point sets via l0 minimization. *Computer Aided Geometric Design*, 35:2–15, 2015. 2

[29] Peng-Shuai Wang, Yang Liu, and Xin Tong. Mesh denoising via cascaded normal regression. *ACM Trans. Graph.*, 35(6):232–1, 2016. 4, 5, 6, 7

[30] Yutao Wang, Hsi-Yung Feng, Félix-Étienne Delorme, and Serafettin Engin. An adaptive normal estimation method for scanned point clouds with sharp features. *Computer-Aided Design*, 45(11):1333–1348, 2013. 2

[31] Jie Zhang, Junjie Cao, Xiuping Liu, He Chen, Bo Li, and Ligang Liu. Multi-normal estimation via pair consistency voting. *IEEE transactions on visualization and computer graphics*, 25(4):1693–1706, 2018. 1, 2, 3, 6, 7, 8

[32] Jie Zhang, Junjie Cao, Xiuping Liu, Jun Wang, Jian Liu, and Xiquan Shi. Point cloud normal estimation via low-rank subspace clustering. *Computers & Graphics*, 37(6):697–706, 2013. 1, 2, 3, 6

[33] Jun Zhou, Hua Huang, Bin Liu, and Xiuping Liu. Normal estimation for 3d point clouds via local plane constraint and multi-scale selection. *arXiv preprint arXiv:1910.08537*, 2019. 1, 3