AdaFit: Rethinking Learning-based Normal Estimation on Point Clouds

Runsong Zhu[1*] Yuan Liu[2*] Zhen Dong[1†] Yuan Wang[1] Tengping Jiang[1] Wenping Wang[2,3]
Bisheng Yang[1†]

[1]Wuhan University [2]The University of Hong Kong [3]Texas A&M University

Abstract

This paper presents a neural network for robust normal estimation on point clouds, named AdaFit, that can deal with point clouds with noise and density variations. Existing works use a network to learn point-wise weights for weighted least squares surface fitting to estimate the normals, which has difficulty in finding accurate normals in complex regions or containing noisy points. By analyzing the step of weighted least squares surface fitting, we find that it is hard to determine the polynomial order of the fitting surface and the fitting surface is sensitive to outliers. To address these problems, we propose a simple yet effective solution that adds an additional offset prediction to improve the quality of normal estimation. Furthermore, in order to take advantage of points from different neighborhood sizes, a novel Cascaded Scale Aggregation layer is proposed to help the network predict more accurate point-wise offsets and weights. Extensive experiments demonstrate that AdaFit achieves state-of-the-art performance on both the synthetic PCPNet dataset and the real-word SceneNN dataset. The code is publicly available at https://github.com/Runsong123/AdaFit.

1. Introduction

In point cloud processing, a fundamental task is to robustly estimate surface normals from point clouds, which plays a key role in many practical applications, such as surface reconstruction [1], registration [2], segmentation [3], primitive fitting [4], reverse engineering [5] and grasping [6]. Due to the presence of noise, point density variations, and missing structures, robust and accurate surface normal estimation on point clouds remains challenging.

The most direct way for normal estimation is to regress the normal vector from the feature extracted on neighboring points [7, 8, 9, 10, 11, 12]. However, such brute-force regression only forces the network to memorize normal vectors, which leads to limited generalization ability. This generalization problem becomes more severe on real-world data due to the scarcity of training data.

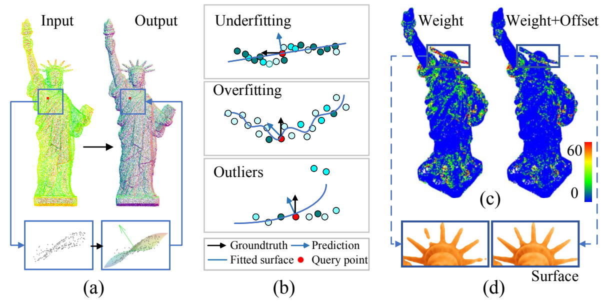

Figure 1. (a) Given the input point cloud (left), our target is to estimate a normal for every point (right). (b) Current weighted least square surface fitting is severely affected by underfitting (top), overfitting (middle) or outliers (bottom), which leads to inaccurate normal estimation. (c) The error maps of two normal estimation methods. The first model only uses the weighted least square surface fitting (left) while the second model adds additional offsets to adjust the distribution of neighboring points, which produces a more accurate normal estimation. (d) The reconstructed surfaces using normals from the model with weights only (left) and the model with weights and offsets (right).

Rather than direct regression, a more accurate approach to estimate the normal for a specific point is to fit a geometric surface (plane or polynomial surface) on its neighboring points and then compute the normal from the estimated surface. Since the estimated surface is usually sensitive to the noise or outliers present in the neighborhood, most existing methods [13, 14, 15] use weighted surface fitting which predicts point-wise weights to control the contribution of every neighboring point to the final surface. The focus of these works is to obtain more accurate point-wise weights that can reduce the effects of noisy points and outliers as possible. Though substantial improvements on normal estimation have been achieved by using a neural network to

*Equal contribution
†Corresponding authors: [dongzhenwhu,bshyang]@whu.edu.cn

learn a more accurate weight [13, 15] in a data-driven manner, the estimated normals are still not accurate in complex regions as shown in Fig. 1 (c).

In this paper, we conduct an analysis on the widely-used weighted surface fitting for normal estimation and find two inherent problems in this approach. The first is brought by the inconsistent polynomial orders between the true surface and the fitted surface. The underlying surfaces of different points usually have different polynomial orders while current methods always choose a constant order for all points, e.g. plane in [15] or 3-jet in [13]. Such inconsistency either results in an underfitting, which smooths out the fine details in the neighborhood shown in the top of Fig. 1 (b), or overfitting to noise, which brings a large variance to the output normal shown in the middle of Fig. 1 (b). Both the overfitting and underfitting may result in an erroneous normal estimation. The other problem is that the weighted surface fitting is sensitive to outliers. By theoretically analyzing the relationship between point-wise weights and the final estimated normal, we find that the weight on a point far away from the fitted surface will have a larger impact on the final normal direction. In this case, since outliers are much far away from the fitted surface than inlier points, even small weights on outliers will thoroughly mess up the estimated normals, as shown in the bottom of Fig. 1 (b).

To address the above two problems, we propose a simple yet effective solution by predicting additional point-wise offsets to adjust the distribution of the neighboring points. When the underlying surface has a different polynomial order from the predefined one, the adjustment brings more flexibility to the network to project neighboring points onto a surface with the predefined order. Though the resulted surface may not be precisely consistent with the ground truth one, the normal of the center point is much more accurate than the one estimated by the direct weighted surface fitting. Meanwhile, the outliers in the neighborhood can be offset to a position near the resulted surface such that all points will have similar distances to the surface. Thus, in comparison with the weighted surface fitting where the weights of outliers have larger impacts on the normal, points contribute more evenly after the adjustment, which leads to a more robust normal estimation.

Another challenge is the selection of optimal neighborhood size in the surface fitting. Here, we call the neighborhood size as scale. A large scale containing more points will provide more information about the underlying surface but may include irrelevant points which easily leads to over-smoothing sharp edges. A small scale only contains a small set of most relevant points thus may improve the accuracy of the normal estimation but is unavoidably sensitive to noise. The scale is a very sensitive hyperparameter that requires a carefully tuning for an accurate normal estimation [13, 15].

To address the problem of the scale selection, we use a novel architecture design called Cascaded Scale Aggregation (CSA) layer. With several CSA layers, we can extract features from the large scale while only fit the surface with neighboring points in a small scale. Thus, using CSA layer for feature extraction enjoys the benefits from both the large scale, which brings more information about the surface, and the small scale, which results in a more accurate normal estimation.

To this end, we implement our idea by designing a network called AdaFit, which takes a query point with its neighboring points as input and outputs the normal of the query point. AdaFit extracts features with CSA layers and simultaneously predicts point-wise weights and offsets. Then, the predicted weights are used to fit a polynomial surface of order 3 on these offset neighboring points. Finally, the output normal is computed from the fitted surface.

To validate the proposed method, we conduct extensive experiments on the widely-used PCPNet dataset [7]. The results show that the proposed AdaFit achieves state-of-the-art performance on this benchmark. To show the generalization ability of our AdaFit, we evaluate it on two real-world datasets, the indoor SceneNN [16] dataset and the outdoor Semantic 3D [17] dataset. Without any further training, AdaFit significantly outperforms baselines by a large margin on these two datasets. Furthermore, we demonstrate the applications of the normals predicted by AdaFit on the point cloud denoising and the surface reconstruction.

Our contributions are summarized as follows:

- We provide a comprehensive analysis on the weighted surface fitting and find two critical problems of these methods in normal estimation.
- We propose to predict offsets to adjust the distribution of neighboring points which brings more robustness and accuracy in normal estimation.
- We design the network AdaFit with novel CSA layers to enjoy benefits from both small and large scales, which achieves improved performance in multiple standard datasets.

2. Related work

2.1. Traditional normal estimation

The most popular method for normal estimation is based on Principal Component Analysis (PCA) [18] and Singular Value Decomposition (SVD) [19], which are utilized to find the eigenvectors of the covariance matrix constructed from the neighboring points. These approaches heavily depend on the selected scale and are sensitive to noise and outliers. Following this work, moving least squares (MLS) [20], variants fitting local spherical surfaces [21], Jets [22] (truncated Taylor expansion) are proposed to fit more complex local surface. Generally, they choose a large-scale neighborhood size to improve the robustness, but tend to the oversmooth details. Mitra *et al.* [23] reduced the neighborhood size by finding a optimal radius r from point density or curvatures of underlying surfaces. To retain more detailed shape, several methods [24, 25, 26, 27] utilize Voronoi cells or Hough transform. Although these techniques come with strong theoretical approximation and robustness guarantees, such methods need to manually set the parameters according to the noise levels in the raw point clouds.

2.2. Learning-based normal estimation

Regression based methods. With the success of deep learning in a wide range of domains [28, 29, 30, 31, 32, 33], some deep learning-based normal estimation approaches are proposed, which utilize the powerful feature extraction capability of deep learning and transform the normal estimation task into a regression or classification task. Depending on the formats of input, learning-based methods can be divided into two groups. The first group of methods [34, 12, 35] transform unstructured point clouds into structured grid format and apply Convolutional Neural Network (CNN) for feature learning. For example, Boulch *et al.* [34] associate a 2D grid representation to the local neighborhood of a 3D point via a Hough transform, and formulate the normal estimation as a discrete classification problem in the Hough space. The second group of methods [7, 8, 9, 10, 11] directly estimate surface normals from unstructured point clouds. For instance, PCPNet [7] estimates surface normal by a deep multi-scale PointNet [28] architecture, which processes the multiple neighborhood scales jointly, thus leading to the phenomenon of oversmoothing. To overcome the oversmoothing phenomenon, Nesti-Net [8] applied a MoE [36] structure to predict the optimal scale rather than direct concatenation of multiple scales, which yields an improved performance. Similarly, Zhou *et al.* [9] improve surface normal estimation by using an extra feature constraint mechanism and a novel multi-scale neighborhood selection strategy. Hashimoto *et al.* [10] propose a joint model that exploits a PointNet for local feature extraction and a 3DCNN for the spatial feature encoding to efficiently incorporate local and spatial structures.

Surface fitting based methods. The former learning-based normal estimation studies [7, 8, 9, 10, 11, 12] directly regress the surface normal with fully connected layers, which leads to weak generalization ability and unstable prediction results. To solve the these limitations, Lenssen *et al.* [15] and DeepFit [13] first utilize the network to predict point-wise weights acting as a soft selection of the neighboring points, then estimate the surface normal by the differentiable and weighted least squares plane or polynomial surface fitting. These methods heavily constrain the space of solutions to be better suited for the given problem and enable the computation of additional geometric properties such as principal curvatures and principal directions. Our method also belongs to this category and we add additional offsets to make the output normals more robust and accurate. Moreover, we propose a novel CSA layer to aggregate features from multiple neighborhood sizes.

3. Method

3.1. Problem statement

Given a point p and its neighboring points $\{p_i|i = 1,...,N_p\}$, we want to estimate the normal $\mathbf{n}_p$ at the point. The normal estimation problem can be solved by fitting a surface on the neighboring points and compute the normal from the fitted surface. Here, we adopt a widely-used n-jet surface model [22], which represents the surface by a polynomial function $J_n : \mathbb{R}^2 \to \mathbb{R}$ mapping coordinates (x, y) to their height z in the tangent space by

$$z = J_n(x, y; \hat{\beta}) = \textstyle\sum_{k=0}^{n} \sum_{j=0}^{k} \hat{\beta}_{k-j,j} x^{k-j} y^j, \quad (1)$$

where $\hat{\beta}$ are coefficients. To simplify the notations, we denote the vector

$$\beta = \left(\hat{\beta}_{0,0}, \cdots, \hat{\beta}_{k,0}, \hat{\beta}_{k-1,1}, \cdots, \hat{\beta}_{1,k-1}, \hat{\beta}_{0,k}, \cdots\right)$$

and we define $\beta_1 = \hat{\beta}_{1,0}$ and $\beta_2 = \hat{\beta}_{0,1}$.

Once the surface is fitted, the normal can be computed by:

$$\mathbf{n}_p = \frac{(-\beta_1, -\beta_2, 1)}{\sqrt{\beta_1^2 + \beta_2^2 + 1}}, \quad (2)$$

In order to find the correct surface function in Eq. 1, a point-wise weight is predicted on every point. Then, all points are transformed to the tangent space by Principle Component Analysis (PCA) and we solve for the surface coefficients by a weighted least square (WLS) fitting problem as follows

$$\beta = \underset{\alpha}{\operatorname{argmin}} \sum_{i}^{N_p} w_i \| J_n(x_i, y_i; \alpha) - z_i \|^2, \quad (3)$$

where w_i is the point-wise weight and (x_i, y_i, z_i) is the coordinate of p_i in the tangent space. The solution to Eq. 3 is given by

$$\beta = (M^\intercal W M)^{-1} (M^\intercal W z), \quad (4)$$

where $W \in \mathbb{R}^{N_p \times N_p}$ is the diagonal matrix consisting of the w_i, $M \in \mathbb{R}^{N_p \times N_n}$ whose i-th row vector M_i is $(1, x_i, y_i, ..., x_i^2 y_i^{n-2}, x_i y_i^{n-1}, x_i^n, y_i^n)$ and $z = (z_1, ..., z_{N_p}) \in \mathbb{R}^{N_p}$.

3.2. Analysis on WLS for normal estimation

To utilize the WLS for the normal estimation, most existing works focus on learning a more accurate weight for every neighboring point. However, two inherent problems of WLS prevent these methods from estimating a more accurate normal.

Inconsistent polynomial orders. For different points, their neighboring points usually conform to surfaces of different polynomial orders. However, the order n in Eq. 1 is a constant predefined integer in WLS models for all points. The inconsistency of orders either brings overfitting or underfitting on the surface fitting. When the predefined order n is smaller than the order of the ground truth surface, an underfitting occurs so that the model struggles to find a true normal for the point. Meanwhile, when the predefined order n is larger than the true order, overfitting would make the fitting process sensitive to the noise on the neighboring points, thus brings instability on the estimated normals. A typical example is shown in Fig. 2, where we fit surfaces with different polynomial orders and red points in the first row show that different points have different suitable polynomial orders for the normal estimation. In the second row, we show the normal errors using different polynomial orders for surface fitting while in the last figure, we always choose their the best polynomial order for different points, which substantially reduces the overall normal RMSE to 12.85°.

Sensitivity to outliers. Meanwhile, the WLS is also sensitive to outliers in the neighborhood. By checking how the weight of every neighboring point affects the final fitted surface, we can prove the following proposition.

Proposition 1. *For a specific point p_i, if it is farther from the fitted surface in Eq. 1, which means the predicted height $z'_i = J_n(\beta, x_i, y_i)$ on this point is largely deviated from the input height z_i, then the weight on this point will have a larger impact on the fitted surface, i.e. $\partial\beta/\partial w_i = (M^\intercal WM)^{-1}M_i^\intercal(z_i - z_i')$.*

In general, outliers will locate far away from the fitted surface with larger $z_i - z'_i$, the resulted surface coefficients β will be more sensitive to their weights according to the Proposition 1. Moreover, we can directly compute the derivative of the estimated normal $\mathbf{n}_p$ to the point-wise weight w_i by $\frac{\partial \mathbf{n}_p}{\partial w_i} = \frac{\partial \mathbf{n}_p}{\partial \beta}\frac{\partial \beta}{\partial w_i}$. Thus, even though the network may learn to place small weights on these outliers, a small perturbation on the weights of outliers will still lead to a significant change in the output normal.

3.3. Offset prediction

To address the above two problems, we propose a simple yet effective solution, in which we first predict an additional point-wise offset $(\Delta x_i, \Delta y_i, \Delta z_i)$ on every point to adjust the distribution of points. Then, WLS is applied on these offset points to find the surface by

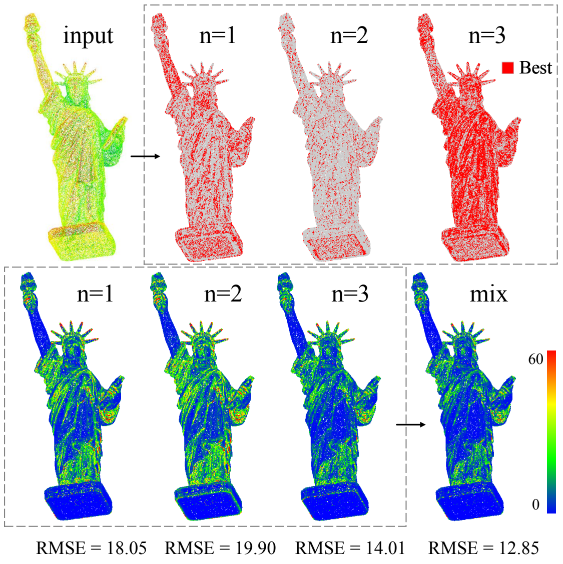

Figure 2. The first row shows the best polynomial orders for different points. The second row shows the corresponding error map of different polynomial orders and the last one ("mix") always uses the best orders for different points for the normal estimation.

$$\beta = \underset{\alpha}{\operatorname{argmin}} \sum_{i}^{N_p} w_i \| J_n(x_i + \Delta x_i, y_i + \Delta y_i; \alpha) - (z_i + \Delta z_i)\|^2. \tag{5}$$

Due to the offset prediction, the network has an additional flexibility to adjust points to construct a virtual surface that has the same polynomial order as the predefined one. Thus, the offset prediction greatly reduces the underfitting or overfitting phenomenons. Meanwhile, outliers can be offset onto the virtual surface by the network so that the resulted surface will be less sensitive to their weights. Two examples are shown in Fig. 3, in which adding the offset prediction brings more robustness to outliers and avoids the overfitting or the underfitting. More examples of neighboring points before and after being offset can be found in the supplementary materials.

3.4. Cascaded Scale Aggregation

In order to extract features for accurate weight and offset prediction, we propose a novel layer called Cascaded Scale Aggregation (CSA) layer. As shown in Fig. 4, the CSA layer is in charge of extracting features from different neighborhood sizes (called scales). For a point p, we define a scale of this point by an integer s, which is represented by a point set N_s including s-nearest points to the point. A CSA layer takes two scales (s_k, s_{k+1}) as input, where $s_{k+1} < s_k$ so that $N_{s_{k+1}} \subset N_{s_k}$. It extracts a feature $f_{k+1,i}$ on a point $p_i \in N_{s_{k+1}}$ by

$$f_{k+1,i} = \phi_k(\varphi_k(\text{MaxPool}\{f_{k,j} | p_j \in N_{s_k}\}), f_{k,i}) \tag{6}$$

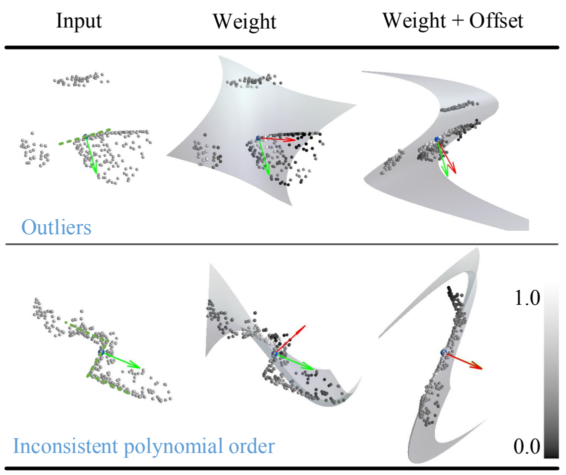

Figure 3. Two examples of direct fitting the surface with weights and fitting with offset points. Green arrows are true normal directions while red arrows show the predicted normal directions. The first example contains some irrelevant outliers while the second one shows the underlying surface is too complex for the current polynomial order to achieve a good fitting. The offsets adjust the distribution of points, which produce a more accurate normal estimation in both cases.

where both ϕ_k and φ_k are Multi-layer Perceptrons (MLP), $f_{k+1,i}$ is the output feature on the point p_i at the scale s_{k+1}, $f_{k,i}$ is the feature of the same point at the scale s_k and $\{f_{k,j}|p_j \in N_{s_k}\}$ is the set of all features of points in N_{s_k}. Note that, since $N_{s_{k+1}} \subset N_{s_k}$, all points in $N_{s_{k+1}}$ will also be in the N_{s_k}.

The CSA layer uses features from a larger scale to help the feature extraction at the current scale while the final fitting only uses the smallest scale. The large scale provides more information of the underlying surfaces while the small scale includes the most relevant points for the surface fitting. Thus, the proposed CSA layers enjoy the benefits from both the large scale and the small scale, which results in an accurate surface fitting.

3.5. Implementation details

Based on the CSA layers, we design a network called AdaFit which simultaneously predicts point-wise offsets and weights. As shown in Fig. 5, AdaFit mainly consists of MLPs and CSA layers for feature extraction on every point. Then, two heads are used to regress the weights and the offsets.

Loss.To train our network, we follow the same loss as proposed in DeepFit [13] which minimizes $|N_{gt} \times N|$ between the predicted normal N and the ground truth normal N_{gt}. Meanwhile, we also adopt the neighborhood consistency loss and transformation regularization loss used in DeepFit. Please refer to [13] for more details.

Parameter setting. The polynomial order n for the surface fitting is 3 by default. For each query point, we use

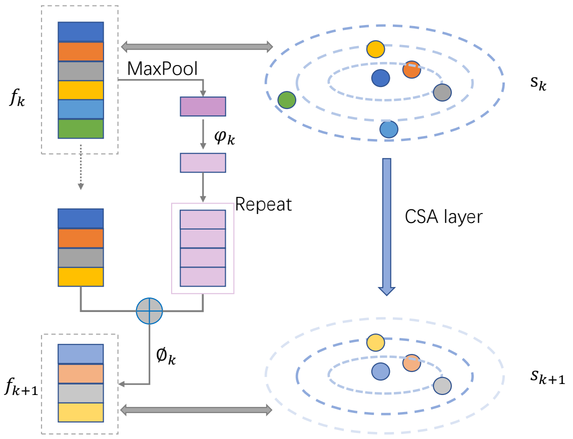

Figure 4. The structure of a CSA layer, which extracts the features from the large scale by a global pooling and concates the features from the large scale to features of points in a small scale as the output.

KNN to pick the nearest K-point. The initial neighborhood size is 700. AdaFit consists of two CSA layers with the diminishing neighborhood size from 700 to 350, 175.

4. Experiment

4.1. PCPNet dataset

We follow exactly the same experimental setup as [7] including train-test split, training data augmentation and adding noise or density variations on testing data. We use the Adam [37] optimizer and a learning rate of 5×10^{-4} for the training with a batch size of 256. AdaFit is trained with 600 epochs on a 2080Ti GPU.

Baselines. We consider three types of baseline methods: 1) the traditional normal estimation methods PCA [18] and jet [22]; 2) the learning-based surface fitting methods Lenssen *et al.* [15], DeepFit [13]; 3) the learning-based normal regression methods PCPNet [7] and Nesti-Net [8].

Metrics. We use the angle RMSE between the predicted normal and the ground truth as our main metrics for evaluation. We also draw the AUC curve of normal errors to show the error distribution.

Quantitative results. The RMSE of AdaFit and baseline methods are shown in Table 1 and the AUC of all methods are shown in Fig. 6. The results show that AdaFit outperforms both the traditional methods and learning-based methods in all settings, which demonstrates the effectiveness of using offsets to adjust the point set. Especially on the point clouds with density variations, baseline methods may fail to find enough points on sparse regions for the surface fitting while AdaFit use the offset to project points to the neighboring regions for more robust surface fitting. In addition, we adding the results of DeepFit [13] with denoising pre-processing [31]. Though denoising reduces the noise and finds smooth surfaces, it does not offset the points

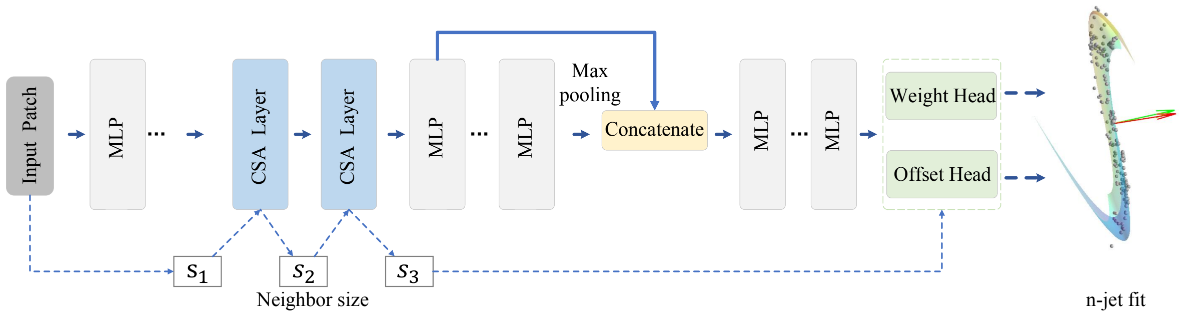

Figure 5. The architecture of AdaFit for normal estimation. AdaFit utilizes CSA layers to extract features from different scales and uses the offset points of the smallest scale with weights to fit a surface. Then the normal vector can be calculated from the surface.

Aug.	AdaFit	DeepFit [13]	Denoising+ DeepFit [13]	Lenssen *et al.* [15]	Nesti-Net [8]	PCPNet [7]	PCA [18]	Jet [22]
No noise	**5.19**	6.51	8.48	6.72	6.99	9.66	12.29	12.23
Noise ($\sigma = 0.00125$)	**9.05**	9.21	10.38	9.95	10.11	11.46	12.87	12.84
Noise ($\sigma = 0.006$)	**16.44**	16.72	16.79	17.18	17.63	18.26	18.38	18.33
Noise ($\sigma = 0.012$)	**21.94**	23.12	22.18	21.96	22.28	22.8	27.5	27.68
Varing Density(Strips)	**6.01**	7.92	9.62	7.73	8.47	11.74	13.66	13.39
Varing Density(gradients)	**5.90**	7.31	9.37	7.51	9.00	13.42	12.81	13.13
Average	**10.76**	11.8	12.8	11.84	12.41	14.56	16.25	16.29

Table 1. Normal RMSE of AdaFit and baseline methods on the PCPNet dataset.

to the ground truth positions on the surfaces, which still results in a larger RMSE.

Qualitative results. Fig. 7 shows the normals estimated by AdaFit and Fig. 8 visualizes the angle errors of AdaFit and baseline methods on different shapes. It can be seen that Lenssen *et al.* [15] perform better on the flattened regions while DeepFit [13] can better handle curved regions but is not accurate for regions with sharp curves. In contrast, the proposed AdaFit is relatively more robust on all regions and all settings (noise or density variations). Additional detailed analysis on normal estimation of sharp edges can be found in the supplementary materials.

4.2. Results on real datasets

To test the generalization capability of AdaFit, we directly evaluate AdaFit on real datasets including the indoor SceneNN [16] dataset and the outdoor Semantic3D [17] dataset. Note that all the methods are only trained on the PCPNet dataset and directly evaluated on these datasets.

The SceneNN dataset. The SceneNN dataset contains more than 100 indoor scenes collected by a depth camera with provided ground-truth reconstructed meshes. We obtain the point clouds by sampling on the reconstructed meshes and compute the ground-truth normal from the meshes. The RMSE of AdaFit and baselines are shown in Table 2 and the angle error visualizations are shown in Fig. 9. The results show that AdaFit can predict more accurate normals than baseline methods.

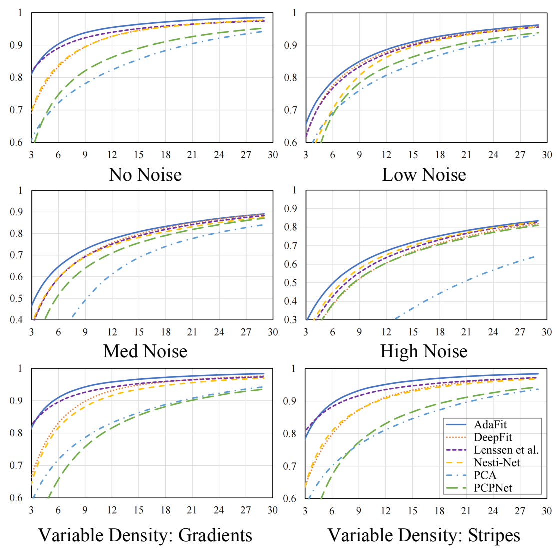

Figure 6. Normal error AUC curves of AdaFit and baseline methods. X-axis shows the threshold in degree and Y-axis shows the ratio of correct estimated normals under a given threshold.

The Semantic3D dataset. The Semantic3D dataset contains point clouds collected by laser scanners. Since there is no ground truth for normal estimation, we only show qualitative results in Fig. 10. The results show that AdaFit can find sharp edges on point clouds while other methods oversmooth these normals.

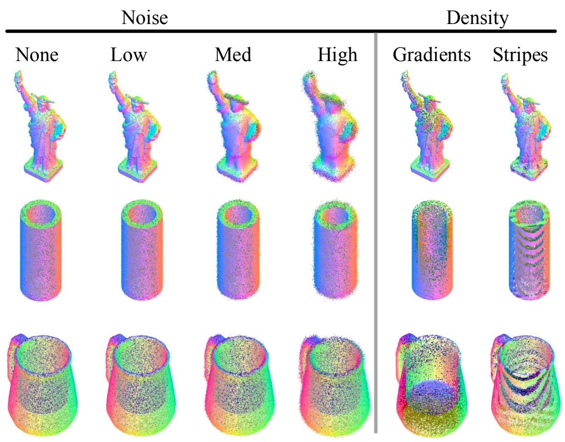

Figure 7. Normals estimated by AdaFit on the PCPNet dataset.

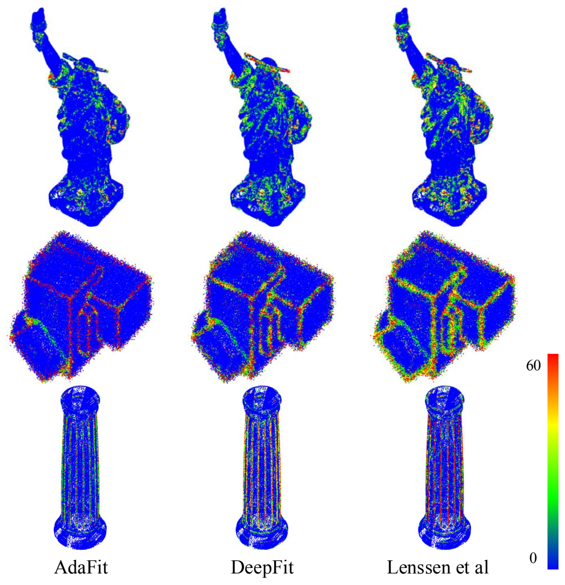

Figure 8. Errors of normal estimation on the PCPNet dataset.

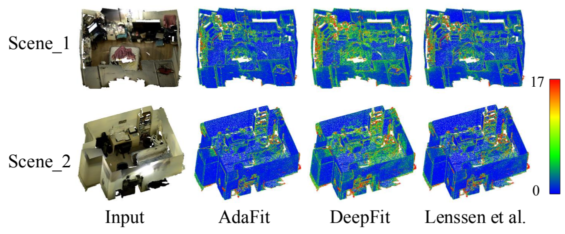

Figure 9. Error maps of estimated normals on the SceneNN dataset.

	AdaFit	DeepFit [13]	Lenssn *et al.* [15]
RMSE	**16.25**	18.33	18.54

Table 2. Normal RMSE of AdaFit, DeepFit [13] and Lenssen *et al.* [15] on the SceneNN dataset.

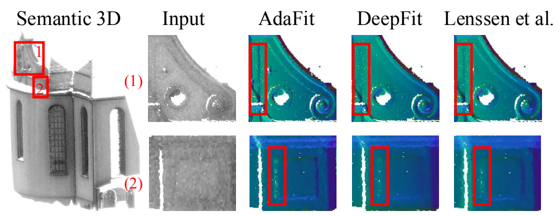

Figure 10. Estimated normals on the semantic 3D dataset.

n	1		2		3	
Weight	✓	✓	✓	✓	✓	✓
Offset		✓		✓		✓
No Noise	8.09	5.96	10.07	6.06	7.91	6.00
Low Nose	10.49	9.34	11.30	9.28	9.82	9.27
Med Noise	16.59	16.56	16.59	16.48	16.36	16.48
High Noise	21.80	21.82	21.61	21.77	21.48	21.76
Stripes	9.72	7.10	12.00	7.18	9.80	7.09
Gradients	8.54	6.67	10.83	6.80	8.59	6.72
Average	12.54	11.24	13.73	11.26	12.33	11.22

Table 3. Normal RMSE of models with or without offset prediction on the PCPNet dataset.

Scale	256	500	700	700
Weight	✓	✓	✓	✓
Offset	✓	✓	✓	✓
CSA				✓
No Noise	5.17	5.79	6.00	5.19
Low Noise	9.17	9.17	9.27	9.05
Med Noise	16.71	16.47	16.48	16.44
High Noise	23.02	22.12	21.76	21.94
Stripes	6.03	6.64	7.09	6.01
Gradients	6.00	6.30	6.72	5.90
Average	11.02	11.08	11.22	10.76

Table 4. Normal RMSE of models with or without CSA layers on the PCPNet dataset.

Threshold	**0.0**	**0.05**	**0.10**	**0.30**	**0.50**	**Offset**
No Noise	7.91	7.35	7.41	7.94	8.41	6.00
Low Noise	9.82	9.53	9.57	10.07	10.49	9.27
Med Noise	16.36	16.31	16.37	16.98	18.06	16.48
High Noise	21.48	21.43	21.44	22.22	23.42	21.76
Stripes	9.80	8.97	8.84	8.94	9.61	7.09
Gradients	8.59	8.09	8.11	8.52	9.11	6.72
Avarage	12.33	11.95	11.96	12.44	13.18	11.22

Table 5. Normal RMSE of models using thresholds to truncate points and the model with offset prediction on the PCPNet dataset. The first row shows the truncation thresholds.

4.3. Ablation study

Offset prediction. To demonstrate the effectiveness of the proposed offset prediction, we conduct experiments on the PCPNet dataset using models with or without offset pre-

	AdaFit	DeepFit [13]	Lenssn *et al.* [15]
Netsuke	**0.00373**	0.00821	0.00677
Liberty	**0.00065**	0.00100	0.00106
Column	**0.01148**	0.02060	0.03420
Average	**0.00529**	0.00994	0.01401

Table 6. The comparison of the RMSE surface distance error for surface reconstruction of our method and DeepFit [13], Lenssen *et al.* [15]

diction. The backbone network in this experiment is the same as DeepFit without any CSA layer and uses the neighborhood size of 700 points. The results are shown in Table 3, which demonstrates that the offset prediction can effectively improve the accuracy of the predicted normals. Meanwhile, we notice that the performance of the model without offset prediction is sensitive to the polynomial order n while the model with offset prediction has similar performance on different orders.

CSA layer. In Table 4, we conduct experiments on the PCPNet dataset using models with or without CSA layers. The results show that using CSA layers can bring improvements on the normal estimation and reduce the necessity to select a specific neighborhood size (scale).

Comparison to truncating weights. Since outliers may have small predicted weights, a more simple way to prevent these outliers from affecting the estimated normals is to truncate points with a predefined threshold. In Table 5, we compare the model using proposed offset predictions with the model that truncates points with small weights. All models take 700 neighboring points as inputs. The results show that truncating indeed slightly improves the results but is still inferior to the model with offset prediction.

5. Application

5.1. Surface reconstruction

Accurate normals can help the Poisson reconstruction [1] to reconstruct a more high-quality and complete surface from point clouds. In Fig. 11, we show surfaces reconstructed by normals estimated with different methods and the corresponding distance RMSE of reconstructed surface is shown in Table 6. The results show that the reconstructed surface using the normals from AdaFit is more accurate and complete than baseline methods.

5.2. Denoising

To demonstrate the application of AdaFit in point cloud denoising, we adopt the normal-based denoising method proposed in [38]. The qualitative results of denoised point clouds and their corresponding reconstructed surfaces are shown in Fig. 12, which indicates that using the normal estimated by AdaFit for denoising produces a smooth surface in flattened regions while still keeping sharp features at edges.

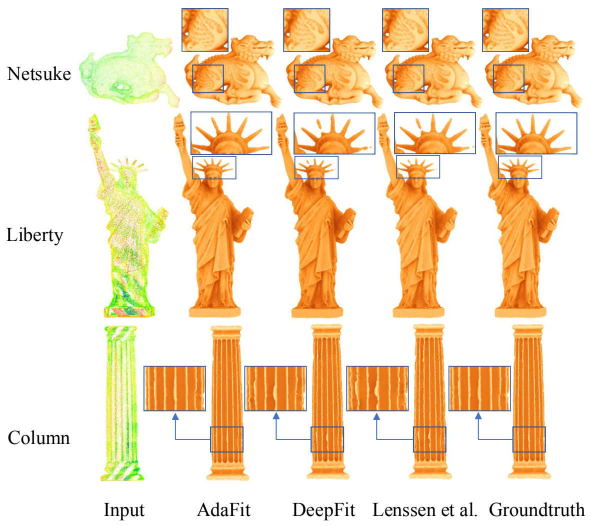

Figure 11. The comparison of the Poisson surface reconstruction using the estimated normals from different methods.

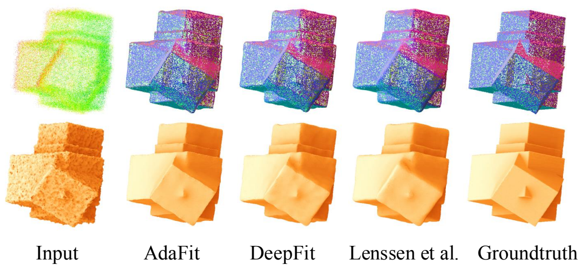

Figure 12. Qualitative results of point cloud denoising. The first row shows the denoised point clouds while the second row shows the corresponding reconstructed surfaces.

6. Conclusion

In this paper, we presented AdaFit for the normal estimation on point clouds. We provided a comprehensive analysis on current weighted least square surface fitting for normal estimation and found two inherent problems of this kind of method. To solve these problems, we proposed the offset prediction on current network and used a novel CSA layer for feature extraction. AdaFit achieved state-of-the-art performance on the PCPNet dataset and strong generalization capability to real-world datasets. Furthermore, we also demonstrated the effectiveness of predicted normals of AdaFit on several downstreams tasks.

7. Acknowledgement

This work was supported by: The National Science Fund for Distinguished Young Scholars of China, grant number 41725005; The National Natural Science Foundation Project (No.41901403); the Fundamental Research Funds for the Central Universities (No.2042020kf0015); The Technical Cooperation Agreement between Wuhan University and Huawei Space Information Technology Innovation Laboratory (No.YBN2018095106)

References

[1] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 1, 8

[2] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. 2015. 1

[3] Eleonora Grilli, Fabio Menna, and Fabio Remondino. A review of point clouds segmentation and classification algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:339, 2017. 1

[4] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007. 1

[5] Jin Liu. An adaptive process of reverse engineering from point clouds to cad models. *International Journal of Computer Integrated Manufacturing*, 33(9):840–858, 2020. 1

[6] Brayan S Zapata-Impata, Pablo Gil, Jorge Pomares, and Fernando Torres. Fast geometry-based computation of grasping points on three-dimensional point clouds. *International Journal of Advanced Robotic Systems*, 16(1):1729881419831846, 2019. 1

[7] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018. 1, 2, 3, 5, 6

[8] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10112–10120, 2019. 1, 3, 5, 6

[9] Jun Zhou, Hua Huang, Bin Liu, and Xiuping Liu. Normal estimation for 3d point clouds via local plane constraint and multi-scale selection. *Computer-Aided Design*, 129:102916, 2020. 1, 3

[10] Taisuke Hashimoto and Masaki Saito. Normal estimation for accurate 3d mesh reconstruction with point cloud model incorporating spatial structure. In *CVPR Workshops*, pages 54–63, 2019. 1, 3

[11] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli. Point cloud normal estimation with graph-convolutional neural networks. In *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2020. 1, 3

[12] Haoran Zhou, Honghua Chen, Yidan Feng, Qiong Wang, Jing Qin, Haoran Xie, Fu Lee Wang, Mingqiang Wei, and Jun Wang. Geometry and learning co-supported normal estimation for unstructured point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13238–13247, 2020. 1, 3

[13] Yizhak Ben-Shabat and Stephen Gould. Deepfit: 3d surface fitting via neural network weighted least squares. In *European Conference on Computer Vision*, pages 20–34. Springer, 2020. 1, 2, 3, 5, 6, 7, 8

[14] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977. 1

[15] Jan Eric Lenssen, Christian Osendorfer, and Jonathan Masci. Deep iterative surface normal estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11247–11256, 2020. 1, 2, 3, 5, 6, 7, 8

[16] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016. 2, 6

[17] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017. 2, 6

[18] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, 1992. 2, 5, 6

[19] Klaas Klasing, Daniel Althoff, Dirk Wollherr, and Martin Buss. Comparison of surface normal estimation methods for range sensing applications. In *2009 IEEE International Conference on Robotics and Automation*, pages 3206–3211. IEEE, 2009. 2

[20] David Levin. The approximation power of moving least-squares. *Mathematics of computation*, 67(224):1517–1531, 1998. 2

[21] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM SIGGRAPH 2007 papers*, pages 23–es. 2007. 2

[22] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. 2, 3, 5, 6

[23] Niloy J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328, 2003. 3

[24] 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. 3

[25] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999. 3

[26] Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry processing*, volume 7, pages 39–48, 2007. 3

[27] Alexandre Boulch and Renaud Marlet. Fast normal estimation for point clouds with sharp features using a robust randomized hough transform. 3

[28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30:5099–5108, 2017. 3

[30] Dongbo Zhang, Xuequan Lu, Hong Qin, and Ying He. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2015–2027, 2020. 3

[31] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, volume 39, pages 185–203. Wiley Online Library, 2020. 3, 5

[32] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. Total denoising: Unsupervised learning of 3d point cloud cleaning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 52–60, 2019. 3

[33] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 386–402, 2018. 3

[34] 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. 3

[35] Dening Lu, Xuequan Lu, Yangxing Sun, and Jun Wang. Deep feature-preserving normal estimation for point cloud filtering. *Computer-Aided Design*, 125:102860, 2020. 3

[36] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. 3

[37] D Kingma and J Ba. Adam: A method for stochastic optimization in: Proceedings of the 3rd international conference for learning representations (iclr'15). *San Diego*, 2015. 5

[38] Xuequan Lu, Scott Schaefer, Jun Luo, Lizhuang Ma, and Ying He. Low rank matrix approximation for 3d geometry filtering. *IEEE Transactions on Visualization and Computer Graphics*, 2020. 8