Weighted Point Cloud Normal Estimation

Weijia Wang[a], Xuequan Lu[a*], Di Shao[a], Xiao Liu[a], Richard Dazeley[a], Antonio Robles-Kelly[a], Wei Pan[b]
[a] *Deakin University, Australia*
[b] *OPT Machine Vision Tech Co., Ltd, Japan*
{wangweijia, xuequan.lu, shaod, xiao.liu, richard.dazeley, antonio.robles-kelly}@deakin.edu.au, vpan@foxmail.com

***Abstract*—Existing normal estimation methods for point clouds are often less robust to severe noise and complex geometric structures. Also, they usually ignore the contributions of different neighbouring points during normal estimation, which leads to less accurate results. In this paper, we introduce a weighted normal estimation method for 3D point cloud data. We innovate in two key points: 1) we develop a novel weighted normal regression technique that predicts point-wise weights from local point patches and use them for robust, feature-preserving normal regression; 2) we propose to conduct contrastive learning between point patches and the corresponding ground-truth normals of the patches' central points as a pre-training process to facilitate normal regression. Comprehensive experiments demonstrate that our method can robustly handle noisy and complex point clouds, achieving state-of-the-art performance on both synthetic and real-world datasets.**

***Index Terms*—3D Point Cloud, Normal Estimation**

I. Introduction

Point clouds are used in a vast range of fields, such as robotics, autonomous driving, 3D scanning and modelling. However, raw point cloud data coming from sensing devices is unordered and does not equip with normal information. Also, it is often corrupted with noise due to precision limitation of sensing devices. As a remedy solution, estimating accurate normals for point clouds has been proven effective in enhancing the performance in various tasks, such as noise filtering [1]–[4] and surface reconstruction [5].

Conventional normal estimation methods such as Principal Component Analysis (PCA) [6] show limited capability in handling noise or complex point cloud surfaces. To handle such defects, learning-based normal regression methods have been proposed. For example, PCPNet [7] and Nesti-Net [8] regress normals from the encoded features of local neighbourhoods (as local patches) on point clouds. Although they show improved robustness to noise, they ignore the varying contributions from the neighbouring points within local patches, which may lead to inaccurate results on complex surfaces. In recent years, methods leveraging surface fitting [9], [10] have been introduced, which attempt to fit polynomial surfaces onto point clouds in order to approximate normals. While such methods demonstrate adaptiveness to complicated geometric surfaces, they tend to be less robust to point clouds contaminated by severe noise.

In this paper, we are motivated to exploit the relevance of neighbouring points to the patch's central point during normal estimation. In specific, we aim to model point contributions as weights and utilise them for normal regression in order to reduce the negative impacts from less relevant points and improve the accuracy of normal prediction. We therefore propose a novel weighted normal regression method for point clouds that consists of a *contrastive pre-training* stage and a *normal estimation* stage. In our pre-training stage, we propose to leverage the correspondences between each normal and the nearby relevant points in a contrastive learning manner. To realise this, we train a point encoder associated with a weight regressor to output weighted point features within a patch, and contrast them with the feature of the central point's normal. In our normal estimation training stage, we fine-tune the pre-trained point encoder and weight regressor, and feed the output weighted point features into our normal regressor to estimate the normal for each patch's central point. By doing so, our method puts higher weights on more relevant points and leads to feature-preserving and noise-resisting results. Fig. 1 provides an illustration of the uniformly-weighted scheme and our approach.

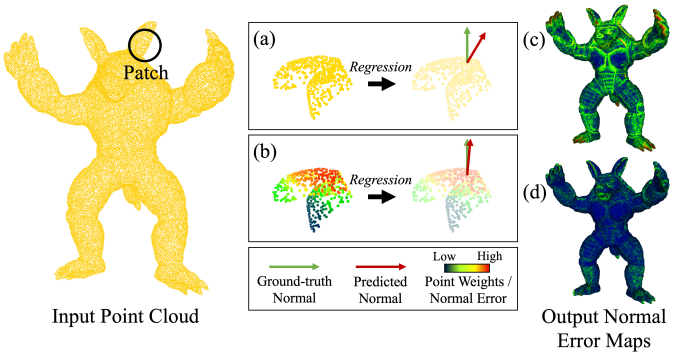

Fig. 1. For any input point cloud patch, regressing the normal with uniform weights on all points (shown in (a)) leads to less accurate results (shown in (c)). By contrast, estimating point-wise weights and performing weighted normal regression (shown in (b)) result in improved normal prediction accuracy (shown in (d)).

Our key contributions are summarised as follows:

- We propose a novel weighted normal regression method for point clouds, which effectively preserves geometric features and is robust to noise.
- We introduce a weight regressor to predict point-wise weights from local patches and use them for more ac-

* Xuequan Lu is the corresponding author.

curate point feature learning and normal regression.
- We propose to contrast patch points and the corresponding normal of the patch's central point in order to exploit the relations between them and facilitate normal regression.

II. Related Work

A. Point Cloud Normal Estimation

Normal estimation for point clouds is a fundamental research problem in the 3D field. A typical example is based on PCA [6], a method that calculates each point's normal by computing eigenvectors from the covariance matrix based on the neighbouring points. While it is simple and straightforward, it tends to blur the details and is fragile on noisy point clouds. With the development in deep learning, data-driven learning-based approaches start to step into researchers' attention, demonstrating improved performance on normal estimation compared with conventional methods. An example is PCPNet [7], which utilises convolutional neural networks to extract features from local point patches and regresses normals from them. To further increase the prediction accuracy, NestiNet [8] utilises a mixture-of-experts network to encode patches of various scales and regress normals from the optimal ones. While such regression-based methods are robust to noise, they omit the different contributions from neighbouring points to the patch's central point during normal estimation. In recent years, there are methods attempting to fit surfaces on point clouds to approximate normals, which take point contributions into consideration. For instance, Lenssen et al. [11] introduced Deep Iterative (DI), which adopts graph neural networks to perform weighted least squares plane fitting on point neighbourhoods. Similarly, DeepFit [9] attempts to fit polynomial surfaces on point clouds to approximate normals for them. Based on DeepFit, AdaFit [10] additionally predicts point-wise offsets and uses a cascaded scale aggregation module to enhance the performance of surface fitting on various shapes. Nonetheless, while such methods perform well on clean points, their performance becomes less robust to noisy point clouds, especially for those corrupted with severe noise.

B. Contrastive Learning

Contrastive learning has been gaining increasing popularity in both 2D and 3D machine vision tasks over the years. For instance, NPID [12] exploits a discrete memory bank to store instance features to facilitate effective contrastive learning for images, and the idea is further leveraged in [13]. Another example is SimCLR [14], which contrasts pairs of different augmented images (e.g., cropped, flipped or masked) and achieves improved image classification results. For 3D point clouds, contrastive learning also demonstrates its ability to facilitate classification, segmentation and object detection tasks [15], [16]. Extending from learning the same type of data, contrastive learning further shows its strength on learning multi-modal correspondences. For instance, CrossPoint [17] contrasts 3D point cloud data and 2D images in order to find potential correspondences which facilitate point cloud understanding tasks. Nevertheless, there has been little attention on extending contrastive learning to point cloud normal estimation. In particular, given that contrastive learning has the ability to find out cross-modal correspondences, we are motivated to directly contrast points and normals in order to facilitate normal estimation.

III. Method

A. Overview

Given a point cloud $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_m \mid \mathbf{p}_i \in \mathbb{R}^3,\ i = 1, ..., m\}$, we aim to predict the normal $\mathbf{n}_i{}'$ of each point $\mathbf{p}_i$ from the features of patch $\mathcal{P}_i$, which centers at $\mathbf{p}_i$ and contains N nearest neighbouring points. It is worth noting that the raw patch $\mathcal{P}_i$ may be of any size or with arbitrary degrees of freedom and is thus unsuitable for direct training. To alleviate this issue, we normalise $\mathcal{P}_i$ to a unit sphere and align it with the global space using a rotation matrix $\mathbf{R}$, which is computed by PCA decomposition on $\mathcal{P}_i$. We later normalise the predicted normal's length and map it back to its original orientation using the inverse matrix $\mathbf{R}^{-1}$ during testing.

Our overall pipeline is shown in Fig. 2. We first pre-train our point encoder and weight regressor using a dual-branch contrastive learning technique, which is elaborated in Sec. III-B. We then fine-tune the trained point encoder and weight regressor to output weighted patch features for down-stream normal regression, which is explained in Sec. III-C.

B. Contrastive Pre-training

By doing contrastive pre-training, we aim to exploit the correspondences between each patch and the normal of its central point (as a patch-normal pair). As there are two types of input data, only using a single encoder is not feasible in this case. Also, as not all points within a patch are highly relevant to the central point's normal, we need to reduce the weights from less relevant points. We thus develop a dual-branch network, where the first branch consists of a point encoder associated with a point weight regressor, and the other branch contains a normal encoder. In addition, each branch is equipped with a projection head. We explain the details as follows.

Point encoder. The point encoders in prior normal estimation work widely adopt point convolutional networks such as [18]. However, such encoders only consider point-wise information and ignore the relationships between each point and its neighbours. Previous literature [19] has proven that dynamically constructing graphs based on each point's local neighbourhood is effective in revealing latent geometric relationships among the neighbours. We are motivated that such relationships contain useful patch features that can enhance normal estimation, especially on noisy and complex surfaces. Thus, we adopt a neighbourhood-based graph convolutional network to encode our patches. In specific, we utilise the EdgeConv modules from DGCNN [19] as the encoding layers for our point encoder. For each point (or vertex) $\mathbf{x}_i$ in the input data, the output feature $\mathbf{x}_i{}'$ from each encoding layer is defined as

$$\mathbf{x}_i{}' = max_{j:(i,j)\in\mathcal{E}} f(\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i), \tag{1}$$

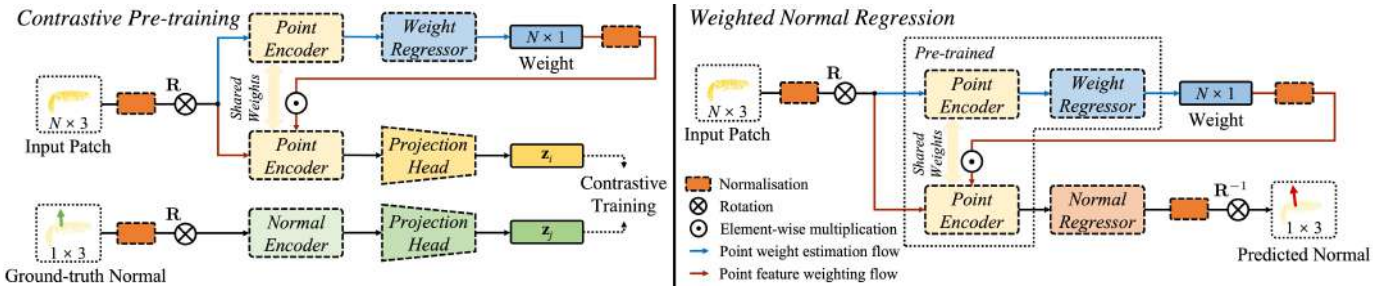

Fig. 2. Overview of our method. In our contrastive pre-training stage (left), we train our network in a dual-branch contrastive learning manner. In our weighted normal regression stage (right), we train the normal regressor together with the pre-trained point encoder and weight regressor to estimate normals.

where $\mathcal{E}$ is the set of edges in the local graph formed by central vertex $\mathbf{x}_i$ and its k nearest neighbouring vertices, and $\mathbf{x}_j$ is a neighbouring vertex within the local graph. We concatenate each edge $(\mathbf{x}_j - \mathbf{x}_i)$ and vertex $\mathbf{x}_i$, and send them to a non-linear function f with learnable parameters followed by a max-pooling operation max. We set $k = 20$ empirically, and denote the final point-wise features of the patch by $\mathbf{F}_i$, which is an $N \times 1024$ matrix. Finally, we apply pooling operations to obtain the global permutation-invariant features for the patch.

Weight regressor. After obtaining the point patch features, we feed them into our weight regressor, which comprises multi-layer perceptrons (MLPs), to predict the relevance of each point to the patch's central point as weights. To train the weight regressor, we utilise the ground-truth normals of the neighbouring points within the patch, as normals are direct representations of the underlying surfaces. A neighbour point is considered as highly relevant if its normal and the central point's normal have a high cosine similarity value, and is less relevant otherwise. Based on this, we compute the squared cosine values between the central point's normal and each neighbour's normal within the patch as our ground-truth weights. *Note that directly utilising such ground-truth weights for normal prediction is not feasible, as they are not available during testing.* We denote the predicted point-wise weights for patch $\mathcal{P}_i$ as $\mathbf{W}_i = (w_j \mid j = 1, \ldots, N)$ and thus formulate the weight regression loss as

$$L_{weight} = \sum_{j=1}^{N} (w_j - (\mathbf{n}_i \cdot \mathbf{n}_j)^2)^2, \tag{2}$$

where $\mathbf{n}_i$ is the ground-truth normal of the central point $\mathbf{p}_i$, and $\mathbf{n}_j$ is the ground-truth normal of each point $\mathbf{p}_j$ within patch $\mathcal{P}_i$. Once we obtain the predicted weights, we perform element-wise multiplication for $\mathbf{W}_i$ and $\mathbf{F}_i$ and obtain the weighted point-wise patch feature matrix $\mathbf{F}_i{}'$ (an $N \times 1024$ matrix), with the process being denoted as $\mathbf{F}_i{}' = \mathbf{W}_i \odot \mathbf{F}_i$. Finally, we apply pooling operations on $\mathbf{F}_i{}'$ to obtain the global permutation-invariant feature for each patch.

Normal encoder. The normal of the central point of each patch is a 1×3 vector and thus cannot be fed into the aforementioned graph convolutional network. We therefore encode the normal into a 1×1024 feature vector using MLPs.

Projection heads. The raw point and normal feature vectors are in a high-dimensional space with plenty of redundant information that intervenes effective contrastive learning. We thus feed each feature vector respectively into a projection head and project them to a lower-dimensional space, which is a remedy solution to reduce redundancy [14]. Here, we denote each projected point patch feature vector as $\mathbf{z}_i$ and the projected normal feature vector as $\mathbf{z}_j$.

We then formulate our loss function for contrastive training. Following [14], we define a temperature-scaled cross entropy loss $l_{i,j}$ for each patch-normal pair as

$$l_{i,j} = -log \frac{\exp(sim(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2B} \mathbb{1}_{k \neq i} \exp(sim(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \tag{3}$$

where $sim(\mathbf{z}_i, \mathbf{z}_j)$ is the cosine similarity between $\mathbf{z}_i$ and $\mathbf{z}_j$, and is divided by a temperature parameter τ. $\mathbb{1}_{k \neq i} \in \{0, 1\}$ is an indicator function that becomes 1 if and only if $k \neq i$. For each patch-normal pair, we compute the temperature-scaled cross entropy loss between $(\mathbf{z}_i, \mathbf{z}_j)$ and $(\mathbf{z}_j, \mathbf{z}_i)$; we then sum up the losses for all pairs within the B-sized batch and take the average loss as our overall contrastive pre-training loss L_{cont}. By doing so, the point encoder can maximise the correspondences within each patch-normal pair and minimise the similarities among different pairs. When pre-training finishes, we keep the trained point encoder and weight regressor for fine-tuning in the subsequent normal estimation training process.

Combining L_{cont} and L_{weight}, the overall loss of our contrastive pre-training stage L_{pre} is defined as

$$L_{pre} = L_{cont} + \alpha L_{weight}, \tag{4}$$

where α is a hyper-parameter controlling the ratio of L_{weight} and we set it to 1.0 based on experiment results.

C. Normal Estimation

We fine-tune our pre-trained point encoder and weight regressor in our normal estimation stage. As shown in Fig. 2, the output patch features from our point encoder and weight regressor are fed into our normal regression module, which consists of a series of fully connected layers as MLPs. The normal regression module produces a 1×3 vector as the predicted normal for each input patch's central point $\mathbf{p}_i$. To

minimise the angle between each predicted normal and the ground-truth one, we design a cosine similarity loss function L_{cos} as a basic loss term which is defined as

$$L_{cos} = 1 - (\mathbf{n}_i{}' \cdot \mathbf{n}_i)^2, \tag{5}$$

where $\mathbf{n}_i{}'$ is the predicted normal, and $\mathbf{n}_i$ is the ground-truth normal of the patch's central point $\mathbf{p}_i$. Nonetheless, merely utilising the cosine loss is not enough as it treats all points equally and omits the different contributions from the neighbouring points. To alleviate this issue, we introduce L_{weight} from Eq. (2) to fine-tune our network. The loss for our downstream normal estimation stage L_{down} is therefore defined as

$$L_{down} = L_{cos} + \alpha L_{weight}, \tag{6}$$

where we set α to 1.0 as per Eq. (4) for consistency during training. We display our experimental results on different network configurations and loss terms in Sec. IV-E. *More details of our network architecture are provided in the supplementary material.*

IV. Experiments and Results

A. Training and Implementation Details

We adopt the same dataset as PCPNet [7], including the train-test split and data augmentation settings (i.e., adding noise). We implement our framework using PyTorch, and train and test it on an NVIDIA GeForce RTX 3080 10GB GPU. We pre-train our model for 50 epochs with two Adam optimisers respectively for each encoder, and then train 100 epochs for normal estimation with a single Adam optimiser. In both training stages, we set the learning rate to 0.001 with a batch size of 32. For each point patch $\mathcal{P}_i$, we set the number of points N to 700 during our experiments.

B. Comparison Metrics and Methods

Following prior work, we use Root Mean Square Error (RMSE) to measure the angular accuracy for our predicted normals. We compare against the conventional PCA method [6], surface fitting methods including DI [11], DeepFit [9] and AdaFit [10], as well as regression-based methods including PCPNet [7] and Nesti-Net [8]. For the PCA method, we set 3 scales of k-nearest neighbours as per [7], where $k = 18$, 112 and 450 respectively for small, medium and large patches.

C. Quantitative Evaluation Results

Synthetic dataset. We first demonstrate the performance on synthetic point clouds corrupted with Gaussian noise. The test set involves 19 shapes that come from PCPNet [7], where each shape also has 4 variants of noise levels (i.e., 0.36%, 0.6%, 0.84% and 1.2% of the length of the shape's bounding box diagonal). Such noisy shapes' original geometric information is contaminated, bringing significant challenges to feature preservation during normal estimation. Despite the challenge, our method still robustly handles the noise and achieves the state-of-the-art performance with regards to RMSE. As shown in Table I, our method outperforms all others at 0.6%, 0.84% and 1.2% noise levels and achieves the minimum overall error. We further demonstrate our method's robustness by visualising the RMSE results on the Boxunion shape from PCPNet's dataset (corrupted with 0.6%, 0.84% and 1.2% noise levels), which is shown in Fig. 3. The results illustrate that our method achieves the minimum error at each noise level compared with the state-of-the-art methods.

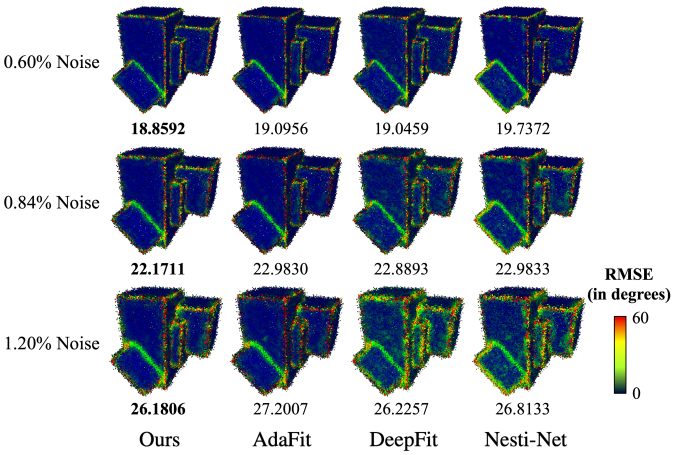

Fig. 3. Visualised RMSE on Boxunion, where our method achieves the minimum error on all noise levels.

Real-world scanned dataset. We also display results on raw point clouds from Kinect Fusion [20] dataset, which are corrupted with noise during scanning. We compare the predicted normals on noisy shapes against the reconstructed clean normals provided by the dataset. The visualised RMSE results are shown in Fig. 4, where both AdaFit and Nesti-Net tend to be sensitive to noise and thus produce inaccurate normals. By contrast, our method performs better on such complex surfaces and achieves lower errors.

D. Visual Results

We demonstrate visual results on real-world data where the ground-truth is unknown. We adopt the tool by [21] to render points and their estimated normals as coloured surfels, where the colours stand for normal orientations, to showcase the normals' quality. Fig. 5 demonstrates results on Paris-rue-Madame dataset [22] which contains street scene point clouds captured by laser scanners and is contaminated by noise. Although Nesti-Net can produce smoother surfaces, it severely blurs features that should be preserved. The fitting-based methods DeepFit and AdaFit may omit certain small details, as their point convolution-based encoders are less context-aware. In addition, AdaFit's performance becomes less robust on noisy areas such as the road surfaces. By contrast, our method outputs tidier normals on complex areas (such as the window lattice in the first scene) while preserving clear features (such as the building eave in the first scene and the car light in the second scene) compared with others.

TABLE I

QUANTITATIVE EVALUATION (RMSE) ON SHAPES WITH GAUSSIAN NOISE FROM PCPNET DATASET [7]. FOR EACH METHOD, WE COMPUTE THE AVERAGE RMSE OF ALL SHAPES AT DIFFERENT NOISE LEVELS AND SHOW THE RESULTS IN DEGREES.

Aug. Noise Level	Ours	AdaFit	DeepFit	DI	Nesti-Net	PCPNet	PCA (small)	PCA (medium)	PCA (large)
0.36%	13.25	13.39	**13.18**	14.00	15.10	15.92	29.49	15.07	17.54
0.60%	**16.38**	16.44	16.72	17.18	17.63	18.26	41.82	18.47	18.99
0.84%	**18.57**	18.85	19.55	19.37	19.64	20.21	48.40	22.27	20.87
1.20%	**21.48**	21.94	23.12	21.96	22.28	22.80	53.34	27.72	23.54
Avg. RMSE	**17.42**	17.66	18.14	18.12	18.66	19.30	43.26	20.88	20.23

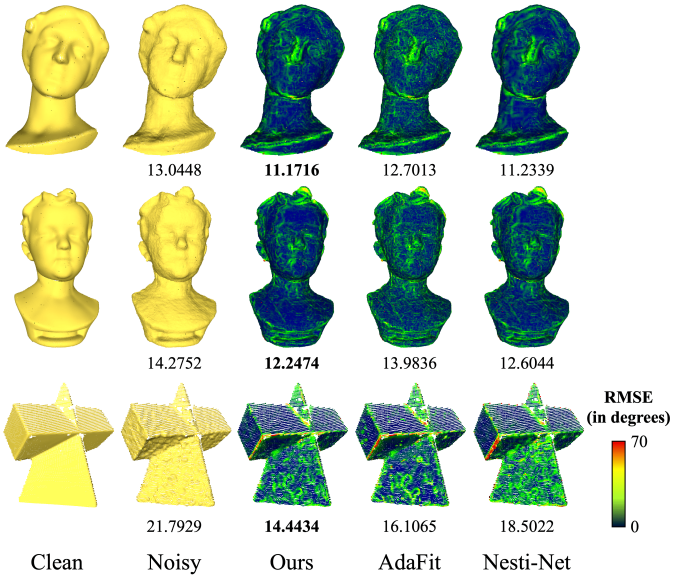

Fig. 4. Visualised RMSE on Kinect Fusion shapes.

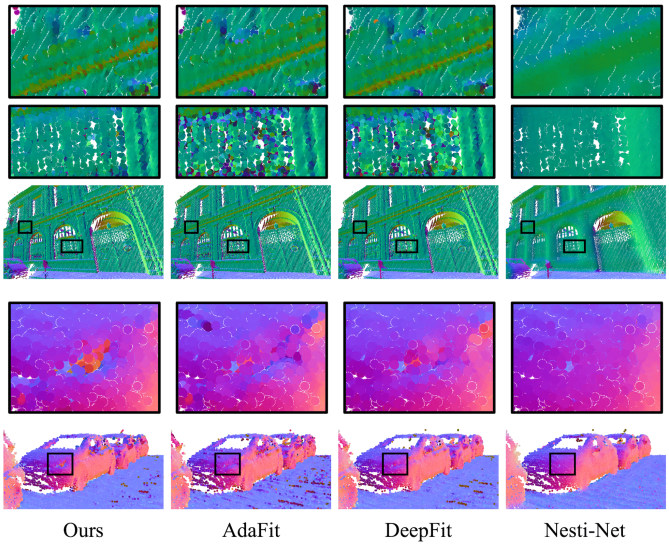

Fig. 5. Visual results on Paris-rue-Madame scenes. Colour map is used for better observation on normal orientations.

E. Ablation Study

To evaluate the effects of the components in our proposed method, we demonstrate the average RMSE results on PCPNet's validation set using different network configurations and loss terms, which are shown in Table II. The configurations include 1) a baseline normal estimation network trained with L_{cos} only, 2) a network pre-trained with L_{cont} and fine-tuned with L_{cos} (i.e., without weight regression), 3) a network with weight regression assisting normal estimation (i.e., without contrastive pre-training), and 4) the full pipeline, where a network is pre-trained and fine-tuned with loss terms in Eq. (4) and Eq. (6), respectively. As demonstrated in Table II, we achieve the best results when we use the full pipeline, i.e., adopt both the contrastive pre-training and the weighted regression strategies.

Additional experimental results are provided in our supplementary material.

TABLE II

ABLATION STUDY ON DIFFERENT SETTINGS. WE CALCULATE THE AVERAGE RMSE ON PCPNET'S VALIDATION DATASET. *WE DO NOT PERFORM CONTRASTIVE PRE-TRAINING IN THIS SETTING.

Config. No.	L_{cos}	L_{cont}	L_{weight}	Avg. RMSE
1	✓			21.15
2	✓	✓		14.70
3*	✓		✓	18.21
4	✓	✓	✓	**14.46**

V. APPLICATIONS

A. Point Cloud Filtering

Point normals can assist with point cloud filtering tasks [1], [3], [4]. We adopt the filtering method in [1] which performs low rank matrix approximation to filter noisy point clouds. This filtering approach relies on input normals, where more accurate normals lead to better filtered results. Here, we use the predicted normals on shapes corrupted with 1.2% noise from PCPNet's test set, and run filtering for one iteration such that the results solely rely on the input normals. We use Chamfer distance [23] as the metric to measure the filtering quality, which computes the average closest point distances between the filtered points and the ground-truth ones. As shown in Fig. 6, the normals predicted by our method lead to the best filtering outcomes.

B. Mesh Surface Reconstruction

Accurate normal information can help Poisson mesh reconstruction [5] with reconstructing high-quality mesh surfaces on point clouds. In Fig. 7, we demonstrate the mesh reconstruction results on a sharp Star shape, which is a tricky case. It

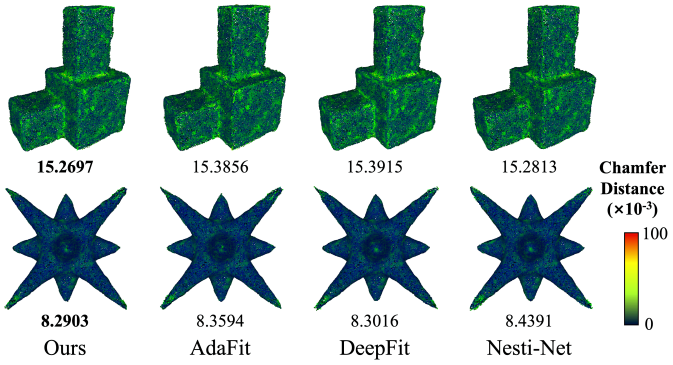

Fig. 6. Average point-wise Chamfer distance for the filtering results.

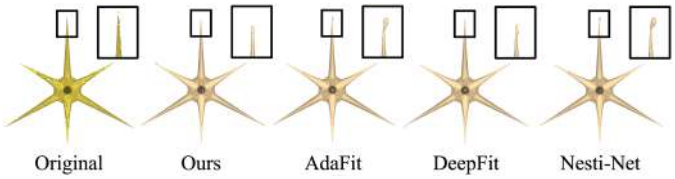

Fig. 7. Poisson surface reconstruction on a sharp Star shape.

shows that the normals predicted by our method lead to more accurate reconstruction on the sharp tip.

VI. Conclusion

In this paper, we proposed a novel weighted normal regression method for 3D point clouds. We developed a weight regression module that predicts point-wise weights to apply more weights on relevant points and reduce impacts from less relevant ones. Also, we proposed a contrastive pre-training stage which leverages the relationships between each normal and the nearby relevant points. Extensive experiments demonstrate that our method achieves state-of-the-art performance on both synthetic and real-world datasets, showing robustness to noisy and complex point clouds.

References

[1] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3d geometry filtering," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2020.

[2] D. Zhang, X. Lu, H. Qin, and Y. He, "Pointfilter: Point cloud filtering via encoder-decoder modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2015–2027, 2021.

[3] D. Lu, X. Lu, Y. Sun, and J. Wang, "Deep feature-preserving normal estimation for point cloud filtering," *Computer-Aided Design*, vol. 125, p. 102860, 2020.

[4] X. Lu, S. Wu, H. Chen, S.-K. Yeung, W. Chen, and M. Zwicker, "Gpf: Gmm-inspired feature-preserving point set filtering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 8, pp. 2315–2326, 2018.

[5] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06. Goslar, DEU: Eurographics Association, 2006, p. 61–70.

[6] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, p. 71–78, Jul. 1992.

[7] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet: Learning Local Shape Properties from Raw Point Clouds," *Computer Graphics Forum*, vol. 37, no. 2, pp. 75–85, 2018.

[8] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "Nesti-Net: Normal Estimation for Unstructured 3D Point Clouds Using Convolutional Neural Networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 104–10 112.

[9] Y. Ben-Shabat and S. Gould, "Deepfit: 3d surface fitting via neural network weighted least squares," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 20–34.

[10] R. Zhu, Y. Liu, Z. Dong, Y. Wang, T. Jiang, W. Wang, and B. Yang, "Adafit: Rethinking learning-based normal estimation on point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 6118–6127.

[11] J. Lenssen, C. Osendorfer, and J. Masci, "Deep iterative surface normal estimation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2020, pp. 11 244–11 253.

[12] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3733–3742.

[13] C. Zhuang, A. Zhai, and D. Yamins, "Local aggregation for unsupervised learning of visual embeddings," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6001–6011.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1597–1607.

[15] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 574–591.

[16] L. Jiang, S. Shi, Z. Tian, X. Lai, S. Liu, C.-W. Fu, and J. Jia, "Guided point contrastive learning for semi-supervised point cloud semantic segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 6403–6412.

[17] M. Afham, I. Dissanayake, D. Dissanayake, A. Dharmasiri, K. Thilakarathna, and R. Rodrigo, "Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 9902–9912.

[18] C. R. Qi, H. Su, K. Mo, and L. 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 (CVPR)*, Jul. 2017.

[19] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, Oct 2019.

[20] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera," in *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, Oct. 2011, pp. 559–568.

[21] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM Transactions on Graphics*, vol. 32, pp. 9:1–9:12, 2013.

[22] A. Serna, B. Marcotegui, F. Goulette, and J.-E. Deschaud, "Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods," in *4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014*, Angers, France, Mar. 2014.

[23] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.