This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

Latent Tangent Space Representation for Normal Estimation

Junjie Cao, Hairui Zhu, Yunpeng Bai, Jun Zhou*, Jinshan Pan and Zhixun Su

***Abstract*—Point cloud processing is rapidly expanding the applicable scenarios in the industry. The surface normal is a fundamental feature for various point cloud processing tasks. Recently, deep supervised normal estimators outperform traditional normal estimation methods by adapting to dataset statistics. However existing normal estimation methods mainly adopt hand-crafted features or complicated networks borrowed from other tasks and make less effort to design network models specifically for the problem. Instead of regressing the normal vector directly, we propose a simple deep network to estimate the normal vector based on a latent tangent space representation learned in the network. We call the network TRNet. For each query point, the tangent space representation is a set of latent points spanning the tangent plane of it. The representation is generated using only the coordinates of its neighbors and regularized by a differentiable RANSAC-like component, which makes TRNet more compact and effective for normal estimation. We also design a compact multi-scale network, denoted by MTRNet, to boost estimations from multiple TRNet trained with different fixed neighborhood size. Our TRNet and MTRNet perform favorably against state-of-the-art methods on synthesized data and real scenarios with far smaller model size.**

I. Introduction

New generation 3-D scanning technologies pave the way for using point clouds at the Industry 4.0, such as reverse engineering [18], surface quality inspection [21], primitive fitting [27], robot navigation [19] and grasping [29]. These tasks in turn rely on surface normals as a fundamental feature to understand the geometry and semantics of scanned objects and scenes. Thus, there is a great need to estimate reliable normals from point clouds since it is hard to measure them directly.

However, fast and reliable normal estimation is still an intriguing challenge since multiple types of noise, outliers and sampling anisotropy are inevitably contained in raw point clouds due to the limitation of the scanners and artifacts of the scenes. These defects may vary from patch to patch even in one digitized model. Sharp features of man-made scenes have to be preserved too, which makes the mission more challenging. There has been a considerable amount of traditional normal estimators contribute to this dilemma [13], [8], [4], [30]. But they do not scale well for large datasets because careful parameter tuning is unavoidable for diverse shapes with the above-mentioned defects and characteristics.

Deep supervised learning approaches provide one solution for this dilemma since they can adapt to dataset statistics. Recently, the performance of normal estimation are gradually pushed up by several deep learning methods [5], [26], [11], [3]. These deep estimators usually regress normal vectors directly and employ advanced networks borrowed from other tasks. Higher performance are achieved via hand-crafted features or more complicated networks with larger model size. Less efforts are made to design a network specifically for the normal estimation problem.

Normal estimation is settled via tangent plane fitting in traditional approaches. We wish to introduce this problem-specific knowledge into deep networks for normal estimation to improve both the performance and interpretability. To this end, we propose to predicate normals from a latent tangent space representation, illustrated in Fig.1, instead of regressing it directly. The latent representation is a set of points encoded by a PointNet module. They span a space approximating the tangent space of a query point. The approximation is guaranteed by a differentiable RANSAC-like module (DSAC).

J. Cao, H. Zhu, Y. Bai and Z. Su are with Dalian University of Technology and Key Laboratory for Computational Mathematics and Data Intelligence of Liaoning Province. J. Zhou is with Dalian Maritime University. J. Pan is with Nanjing University of Science and Technology.

Email: jun90@dlmu.edu.cn

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

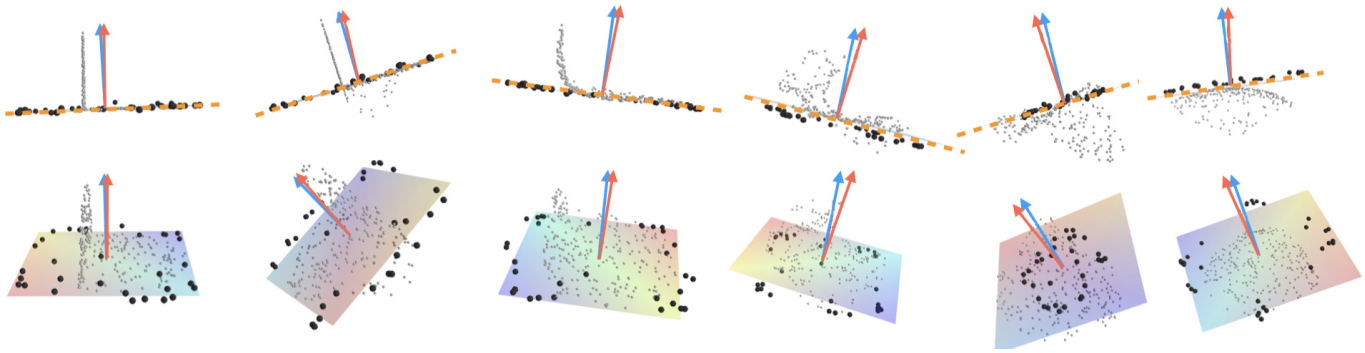

Fig. 1. The latent tangent space representation. Each column shows a patch of input points around some query point, the tangent space of the point and a latent space approximating the tangent space in two different views. The input points, a kind of raw point representation, are rendered in small and grey dots. The tangent space is rendered as an orange dotted line in the top row or a plane in the bottom row, with a red arrow indicating the ground truth normal at each query point. The black points in larger radius compose the latent tangent space representation, and they are generated in our network. We estimate each query point's normal from the latent representation instead of regressing them directly. Each predicted normal is illustrated with a blue arrow.

We demonstrate a significant improvement over the direct neural predictions with simpler and smaller networks using simple inputs. The contributions of this paper are summarized as follows:

1) We propose a single scale normal estimator that learns a latent tangent space representation instead of directly regressing normals. Estimating normals based the representation, makes it possible to achieve higher accuracy without complex network architecture and hand-craft features.
2) A lightweight multi-scale normal estimator MTRNet is proposed by compacting the common Mixture of Experts (MoE) architecture. It performs favorably with fewer experts and smaller model size.

II. Related work

A. Normal estimation

Normal estimation is challenging for the raw point clouds with noise, sampling anisotropy and sharp features. There has been a large amount of works on the topic. The best-known normal estimator is Principal Component Analysis (PCA) [13]. It estimates the normal of a point by fitting a local plane to all neighbors of it. Higher order local surface fitting can also be applied, such as fitting local spherical surfaces [10] or jets [7]. While recent experiments [11] showed that their results may be subpar by PCA. PCA is extremely efficient for ideal data. Larger neighborhoods are necessary for handling noise, which leads to blurred sharp features. Many heuristics have been proposed to assign a proper weight to each neighbor [25] or to each neighbor pair [30], or choose the neighborhood size adaptively [22], etc. Alternative approaches include normal estimation from generated Delaunay/Voronoi cells [1], [8], normal mollification [15], extracting normals in Hough space [4], and by analysis of the subspace structure of the neighborhood [30], [20].

Recently, several deep normal estimators were proposed and surpass previous traditional methods. Boulch et al. [5] trained a 2D CNN on the Hough space of [4]. PointNet [26] proposed the first end-to-end deep point cloud normal estimator. It estimates all the normals simultaneously. Comparing with the two methods, PCPNet [11] improved the performance substantially by employing a patch-based learning strategy, i.e. using PointNet on each query point's neighborhood. Nesti-Net [3] improved the performance further. It applies advanced 3D Inception module on multiple $m \times m \times m$ 3D grids, m = 3 to 8, which record complicated and hand-crafted multi-scale features.

More recently, Lenssen et al. [17] use a learnable anisotropic kernel function to accomplish an iteratively reweighting least squares scheme, denoted by IterNet. The kernel parameter and the local rotation are jointly regressed by a graph neural network. To avoid the iterative process, DeepFit [2] employs the truncated Taylor expansion to solve the weighted least-squares problem. Previous deep normal estimators predicate normals directly and rely on hand-crafted features or complex networks with more and more model parameters. IterNet, DeepFit

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

and our networks all achieve higher performance by exploring problem-specific structures in deep networks. They predict the weights of least squares for fitting the tangent plane. While we predict a latent tangent plane representation constrained by a differentiable RANSAC-like module. Our method also performs slightly favorable against them.

B. Multi-scale normal estimation

Multi-scale architectures are necessary for robust estimation under various conditions. PCPNet [11] trained multiple sub-networks for neighborhoods of multiple scales jointly. However it tends to average the results and does not boost its single-scale network obviously. Although Nesti-Net [3] adopts advanced hand-crafted features and 3D convolution modules, its single-scale networks perform less well than those of PCPNet. But Ben-Shabat et al. [3] showed that using MoE architectures [14] to pick the best single scale sub-network performs better and much improvements can be achieved by using more sub-networks but with larger model size.

We further verify several variants of MoE and find that the scale manager network and expert networks can be integrated in a more compact way. It performs better with fewer experts and smaller model size.

C. RANSAC and its variants

Random sample consensus (RANSAC) [9] is an important method from robust statistics and usually as a post-process for recent decision forest or neural network algorithms [24], [28], since its non-differentiability. Brachmann et al. [6] presented two different ways to overcomes this limitation via soft argmax selection and probabilistic selection. However, poor performance is usually observed when there are mass outliers, and they still share one common defect with traditional RANSAC algorithms which is the accuracy problem. Their results have to be further refined by other algorithms [6].

We solve the issues by designing a differentiable RANSAC-like component following the latent points generation rather than apply it directly to the raw input points contaminated by various defects. These make our algorithm more robust and accurate without any post-refining.

III. Algorithms

TRNet is outlined in Fig. 2. Given a point cloud $P = \{p_1, p_2, \cdots, p_T\}$ and a query point p_i, TRNet is applied to k-nearest neighbors $P_i \in R^{k\times 3}$ of p_i to estimate a normal vector. It contains three successive modules: encoder, decoder and DSAC. The network configuration of the encoder and decoder are presented in Tab. I. The encoder encodes a feature vector f from P_i. The decoder then decodes a point set $Q_i \in R^{k'\times 3}$, from the encoded feature f ($k' = 32$ is used in our methods). At last, DSAC is applied on Q_i to generate a set of hypothesis planes. All the planes are required to approximate the ground truth tangent plane, in the training stage, which makes Q_i a tangent space representation for p_i. See Fig. 1 for some illustrations. In evaluation stage, DSAC is applied to select the best hypothesis plane from Q_i to estimate a normal for p_i as what RANSAC usually does. Furthermore, we propose a compact MoE network called MTRNet, see Fig. 3, to adaptively select the best expert for each query point, which improves performance evidently.

Layer type	Setting details	Output
Input	-	$k \times 3$
QST	$[3 \times 3]$	$k \times 3$
MLP	$[64, 64]$	$k \times 64$
FST	$[64 \times 64]$	$k \times 64$
MLP	$[64, 128, 1024]$	$k \times 1024$
Poolling	-	1024
FC	$[512, 256, 96]$	
Reshape	-	32×3

TABLE I: The configuration of TRNet.

A. Encoder

TRNet's encoder uses almost the same encoder of PCPNet [11] to learn a local feature $f = [f_1, ..., f_{1024}]$ from p_i's neighborhood P_i. P_i is transformed to a canonical pose by a quaternion spatial transformer (QST). Then a shared MLP (64, 64) on each point, a feature spatial transformer operation (FST), and another shared MLP (64, 128, 1024) are followed. However the max or sum pooling, adopted in PCPNet, does not exploit the fact that neighbors closer to the query point should contribute more to the estimation, which is extensively explored in traditional estimators. Hence we propose weighted

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

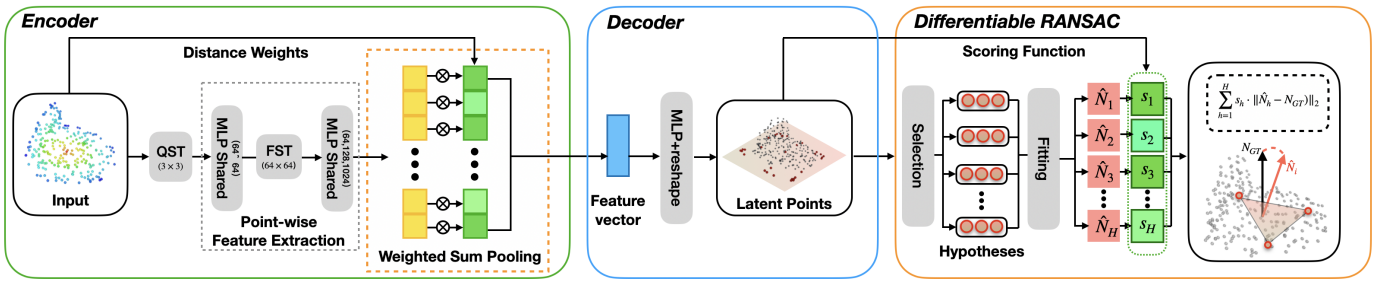

Fig. 2. Pipeline of TRNet. The encoder abstracts the input patch as a plane-aware feature vector. The decoder uses the feature to generate a latent point set which represents the tangent space of the query point. The differentiable RANSAC-like module DSAC ensures supervised constraints on the latent points and achieves the best estimation. "MLP" stands for multi-layer perceptron.

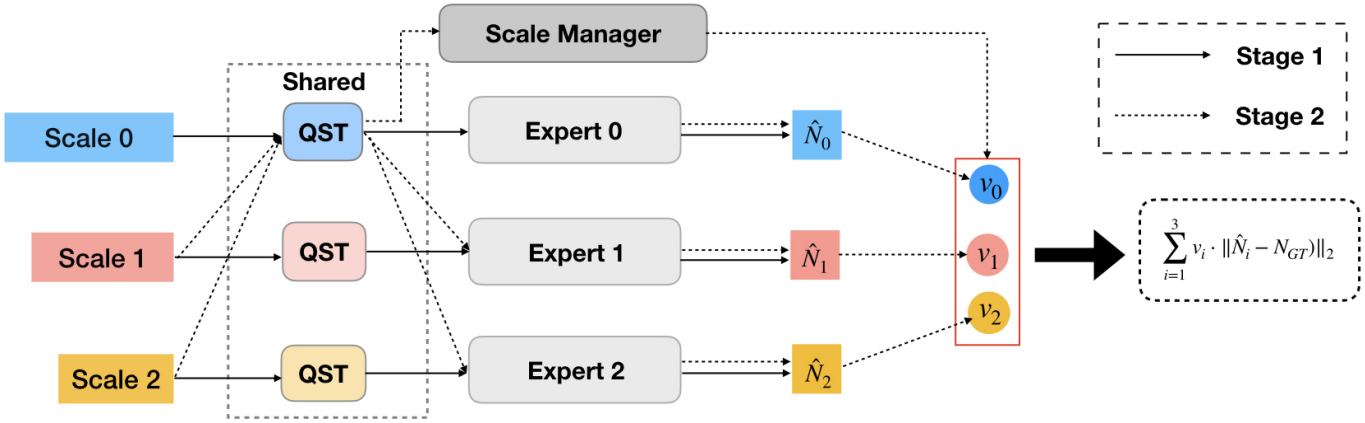

Fig. 3. Pipeline of the two-stage MTRNet. At the first stage (the solid line), multi-scale experts are trained separately. At the second stage (the dot line), a manager network is used to determine the optimal scale and all the experts are fine-tuned. In addition, a shared QST strategy is used to compact the MoE architecture.

sum pooling to fuse point-wise features, as shown in the orange dotted box of Fig. 2. The distance weight $W = [w_j]$ is a k-dim vector and calculated as follows:

$$w_j = \exp(-\frac{\|p_j - p_i\|_2}{\sigma^2}), p_j \in P_i,$$

where $\sigma^2 = 0.12$. Then our weighted sum pooling can be written as:

$$f = \sum_{j=1}^{k} w_j * F(p_j), p_j \in P_i,$$

where $k = 128, 256\, or\, 512$ is used for the neighborhood of small, middle or large scale, and $F(p_j)$ denotes an encoded feature for p_j.

B. Decoder

PCPNet and Nesti-Net regress a normal vector from the coordinates or hand-craft features of the neighborhood. In the view of the raw point clouds are inevitably contaminated by various defects, we learn a latent point set reflecting the local structure of the underlying surface. For this purpose, we propose a simple point set predication network composed by three fully connected layers with output sizes 512, 256 and 96. The 96-dimensional vector is then reshaped to 32 points denoted as Q_i. The generated point set spans the tangent space of p_i since the followed DSAC providing such a regularization. Fig. 1 shows some examples for different kinds of neighborhoods. Obviously, Q_i are distributed on the tangent planes of each query point p_i.

C. DSAC

DSAC is a RANSAC-like module consisting of the following two steps:

1) **Generate a set of hypotheses.** H minimal sets of observations are sampled to compute a pool of hypotheses $\{\hat{N}_h, h = 1, ..., H\}$. Each minimal set $\hat{Q}_h = \{q_{h1}, q_{h2}, q_{h3}\}$ contains three randomly sampled latent points from Q_i, and each hypothesis $\hat{N}_h$ is determined by $\hat{Q}_h$,

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

$\hat{N}_h = N(\hat{Q}_h)$. $N(\cdot)$ is the function computing the normal of the plane through the three points of $\hat{Q}_h$. Larger H means stronger constraints during training and larger probability to sample a better hypothesis during inference, which both lead to higher accuracy with the cost of running time. We set $H = 32$ in our networks both for reference and training.

2) **Constrain all the hypotheses.** Instead of selecting the best hypothesis like the RANSAC approaches, we require all the hypotheses to be consistent with the ground truth normal during training. For this purpose, the following loss is defined:

$$\sum_{h=1}^{H} \frac{exp(s(\hat{N}_h, Q_i))}{\sum_{h'} exp(s(\hat{N}_{h'}, Q_i))} \cdot \|\hat{N}_h - N_{GT}\|_2, \quad (1)$$

where $s(\cdot)$ is a scoring function measuring the consensus of an hypothesis w.r.t. all latent points in Q_i. Inlier counting [9] is the most common scoring function. A latent point q_i is an inlier if its distance to the hypothesis plane, $D(\hat{N}_{h'}, q_i)$, is smaller than a threshold in_t. To make the counting function differentiable, the scoring function is defined as:

$$s(\hat{N}_{h'}, Q_i) = \sum_{i=1}^{H} (\sigma(in_s * (in_t - D(\hat{N}_{h'}, q_i)))),$$

where σ is a sigmoid function, in_s is a scale factor, and we set $in_s = 100$ and $in_t = 0.1$ empirically. The training loss and the generation of the hypotheses collaboratively guarantee that the latent points Q_i will span the tangent space of the query point p_i.

During inference, DSAC chooses the best hypothesis like RANSAC:

$$\hat{N} = \underset{\hat{N}_h}{\operatorname{argmax}}\, s(\hat{N}_h, Q_i). \quad (2)$$

This makes our algorithm more robust, faithful normal is estimated even some latent points slip off p_i's tangent plane.

D. MTRNet

Both PCPNet and Nesti-Net's multi-scale version increase accuracy with the cost of far larger model size relative to their single scale networks, from 16.4M to 170M for Nesti-Net. We pursue a multi-scale scheme with higher accuracy, smaller model size and less training time simultaneously.

Our MTRNet is illustrated in Fig. 3. It is a compact version of MTRNet-1, which follows a common MoE architecture. MTRNet-1 consists of two modules: a scale manager module and an experts module. The experts module contains three separate experts networks, and each of them is a TRNet handling large, medium, or small scale neighborhoods using 512, 256, or 128 neighbors, respectively. The manager follows the architecture of PCPNet's encoder. It takes all the three neighborhoods as input and outputs a score vector. The pipeline is shown in Fig. 3. MTRNet-1 usually takes 48 hours to train 1000 epochs on a single GTX 2080 with a batch size of 128.

We tried two strategies to compress MTRNet-1 into MTRNet. Observing that the largest neighborhood contains the information of the other two neighborhoods, only the largest neighborhood is selected as the input of the scale manager. In addition, we share the QST layer between the manager and all the experts. The two operations reduce the training parameters and save $1/6$ training time.

Besides, for promoting the reliability of experts and saving training cost further, a two-stage training strategy is adopted. At the first stage, the expert networks are trained separately, as the solid line indicated in Fig. 3. At the second stage, dotted line in Fig. 3, the experts are fine-tuned to work under the supervision of the manager collaboratively.

Finally, the experts module output three normal vector $N_j, j = 1, 2, 3$, and the manager gives a score vector $V_i = (v_1, v_2, v_3)$. The final prediction is the normal with the highest score. We train MTRNet to minimize the difference between each predicted normal N_j and the ground truth normal N_{GT}:

$$\sum_{j=1}^{3} v_j \cdot \|\hat{N}_j - N_{GT}\|_2$$

IV. Experiments

We train our model using the synthetic dataset of PCPNet [11] and follow the training and evaluation setup of PCPNet to ensure fair comparisons. All our training was performed on an Nvidia GTX 2080 using stochastic gradient descent with a batch size of 256 for the single scale networks and 128 for the

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

multi-scale networks, a learning rate of 0.001 and 0.9 momentum. The root mean squared (RMS) error of angle difference is used to evaluate estimated normals on the test set, as suggested by [11]. The source code and trained models are available at https://github.com/jjcao/TRNET19.

A. Evaluation on Synthetic Dataset

RMS angle error of our approaches and related methods on the PCPNet test set are shown in Tab. II. For HoughCNN [5], we use the single-scale network provided by the authors. Its neighborhood size is 100 and its average accuracy is higher than the 5-scale network proposed in the paper, refer to Fig. 3 of [11]. For PCPNet [11], we use the 3-scale network provided by the authors, and it only mildly improves performance relative to its single scale network, which can also be found in [11]. Nest-Net [3] achieves higher accuracy. However it is a MoE architecture with 7 sub-networks using a significantly larger number of parameters than other methods. Our TRNet of the middle scale with 256 neighbors achieves almost the same performance as it, but with 0.02 times the size of it, which means far less training and running time and computation resources, see Tab. VII. The results of the two latest methods, DeepFit [2] and IterNet [17], have lower RMS errors compared to the above methods. They do not have multi-scale versions since they learn the least square weights. Our MTRNet, a compact MoE architecture with 4 sub-networks, slightly outperforms them on average but with larger model size. Some qualitative comparisons using PGP5/10 metrics are shown in Fig. 4. The proportion of good points metric (PGPα) computes the percentage of points with an error less than α degree. For example, PGP5 computes the percentage of points with angular error of less than 5 degree.

We also evaluate the generalizability of our method with another synthetic dataset of PCV [30] in Tab. III. The dataset contains 152 synthesized point clouds with various properties which are divided into five categories. Although the three deep normal estimators are only trained on the dataset of PCPNet, they all outperform PCV, the best traditional normal estimator on the dataset, by a large margin. MTRNet is expert in keeping sharp features and details and its average RMS angle error is the lowest.

B. Evaluation on Scanned Scenes

Some qualitative results on real point clouds scanned by Kinect v1 RGBD camera from NYU Depth V2 dataset are shown in 5, since the dataset does not contain ground truth normals. Note that all the compared methods are trained only on the synthetic PCPNet dataset and the synthesised point clouds characteristics are different with the RGBD dataset. After performing unoriented estimation, we flip normals according to the camera position. Our results reveal the nonsmoothness of the scanned data associated with the Kinect v1 scanning process, and reserve sharp features better than Nesti-Net and PCPNet.

C. Ablation study

1) Weighted Sum Pooling vs Sum Pooling: Previous point cloud networks usually use max or sum pooling to handle the disordered inputs. For example, PCPNet [11] uses sum pooling and Nesti-Net [3] favors max pooling. Although the above two pooling operations can describe the global information of the neighborhood, they ignore the differences of the neighbors. Intuitively, a neighbor point near the query point obviously has a relatively large contribution to the regression of the tangent plane. Therefore, we propose to use distance weighted sum pooling. It is a better feature fusion strategy, called distance weighted attention. Tab. IV shows the comparison between max, sum and our weighted sum pooling for different neighborhood sizes. Max and sum pooling lead to similar results for our TRNet. 0.48 improvement, with respect to sum pooling, on average is observed.

2) Regressing Directly vs Voting on the Tangent Space Representation: The TRNet is similar to PCPNet [11]. We both use the Pointnet architecture. The key difference is the DSAC module followed. PCPNet regresses a normal from the latent feature vector. While we decode the vector to 32 latent points following a RANSAC-like regression. Hence, our TRNet degenerates into PCPNet when we use the same decoder.

To explore the importance of the latent tangent space representation and the RANSAC mechanism, and remove the interference of other factors, we compare our TRNet with PCPNet using the same neighborhoods and pooling operations in Tab. V. As shown in Tab. V, for different scales, training with

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

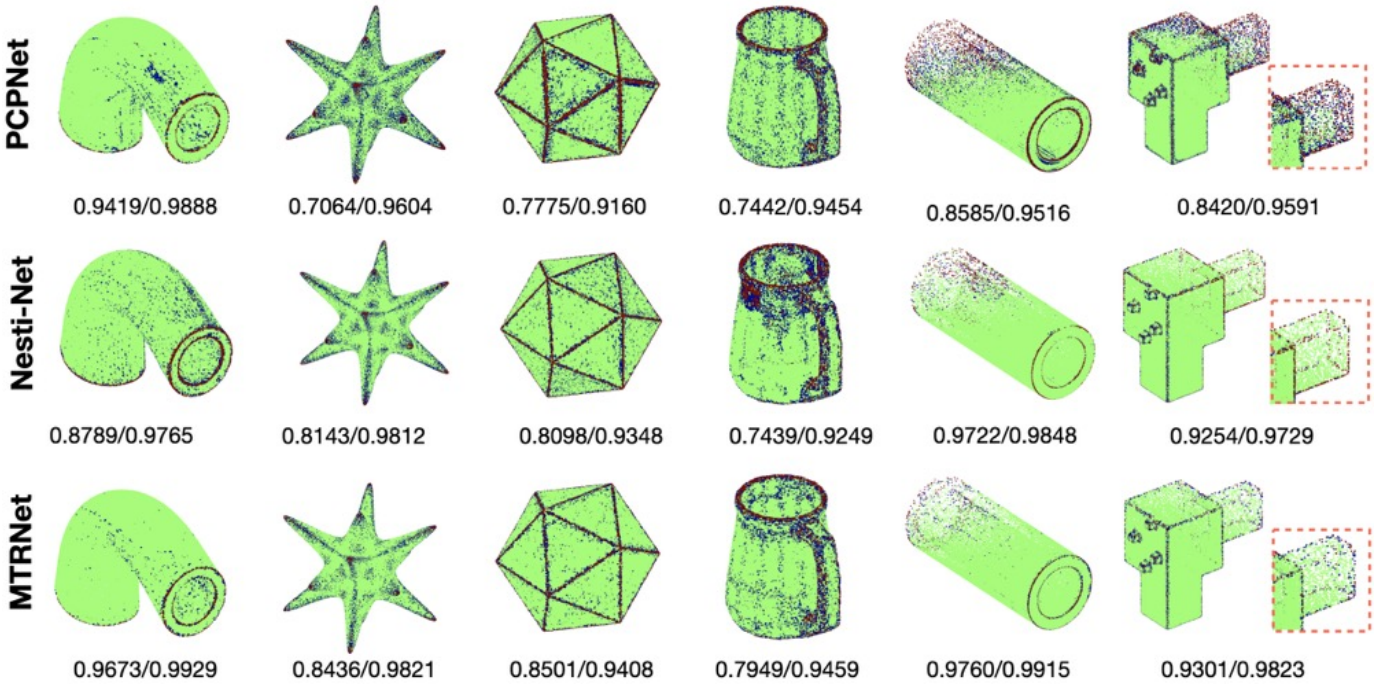

Fig. 4. Comparisons of the normals estimated by PCPNet, Nesti-Net and our MTRNet using the PGP5/10 metrics (higher is better).

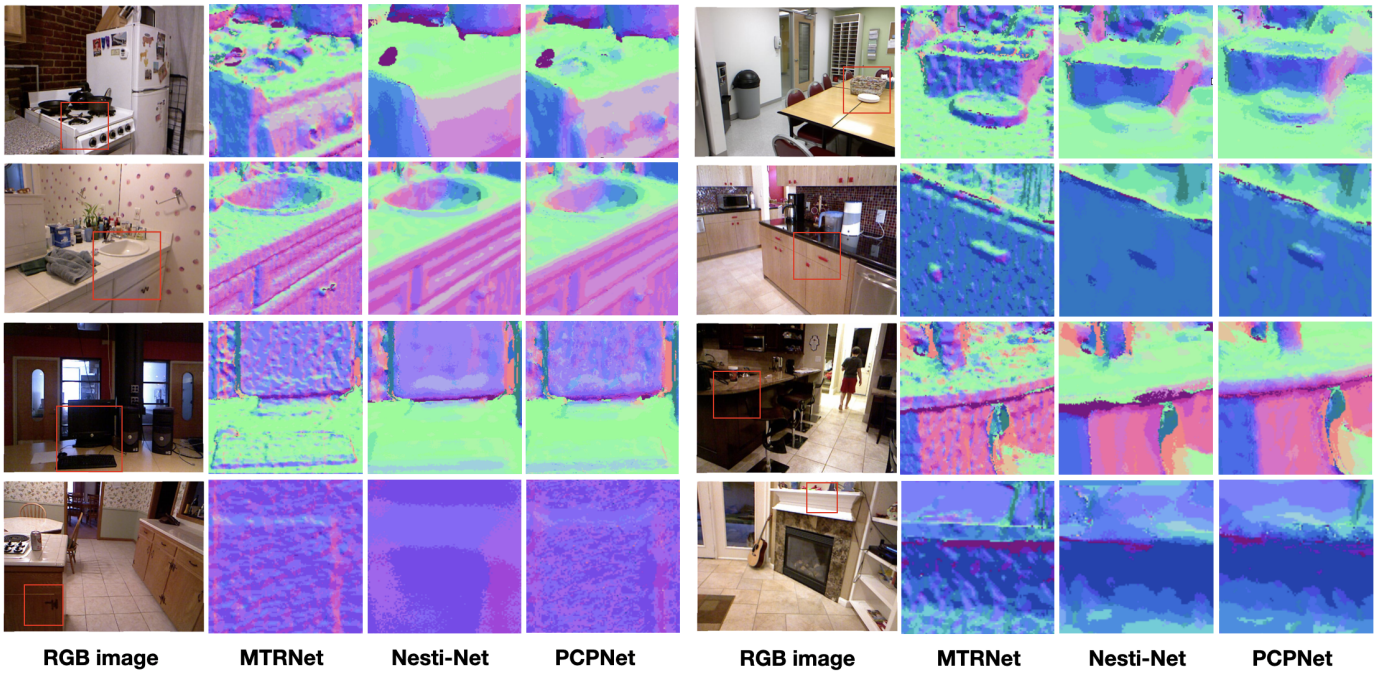

Fig. 5. Estimated normals of our MTRNet, multi-scale Nesti-Net and multi-scale PCPNet on eight indoor scenes of NYU Depth V2 dataset [23].

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

Method	MTRNet	TRNet	DeepFit	IterNet	Nesti-Net	PCPNet	HoughCNN	PCA	Jet
No Noise	**6.43**	7.23	6.51	6.72	6.99	9.62	10.23	12.29	12.23
Noise									
σ = 0.125%	9.69	9.91	**9.21**	9.95	10.11	11.37	11.62	12.87	12.84
σ = 0.6%	17.08	17.17	**16.72**	17.18	17.63	18.87	22.66	18.38	18.33
σ = 1.2%	22.23	23.62	23.12	**21.96**	22.28	23.28	33.39	27.5	27.68
Density									
Gradient	**6.89**	7.85	7.31	7.73	9.00	11.7	11.02	12.81	13.39
Stripes	8.39	9.94	7.92	**7.51**	8.47	11.18	12.47	13.66	13.13
Average	**11.78**	12.62	11.8	11.84	12.41	14.34	16.9	16.25	16.29

TABLE II: Comparison of the RMS angle error for unoriented normal estimation of our methods (MTRNet and TRNet) to classical geometric methods (PCA [13], and Jet [7]) and deep learning methods (DeepFit [2], IterNet [17], Nesti-Net [3], PCPNet [11] and HoughCNN [5]) on PCPNet's synthetic dataset. For DeepFit, IterNet, HoughCNN, PCA and Jet, optimal neighborhood size for the lowest average error is chosen. For Nesti-Net and PCPNet, the results of their multi-scale networks are shown since they have less average error.

Method	MTRNet	DeepFit	IterNet	PCV
SharpFeature	**8.96**	9.84	10.21	11.89
Details	**13.50**	14.68	14.78	19.96
SmallNoise	11.58	**11.54**	12.12	13.15
BigNoise	13.87	**13.18**	14.24	16.19
NonUniform	6.01	6.35	**5.86**	6.62
Average	**10.78**	11.12	11.44	13.57

TABLE III: Comparison of the RMS angle error for unoriented normal estimation of our MTRNet to DeepFit [2], IterNet [17] and PCV [30] on PCV's synthetic dataset. Note that MTRNet, DeepFit and IterNet are only trained on PCPNet's dataset.

Neighborhood	Max	Sum	Sumd	Imp. WRT Sum
k=128	13.71	13.72	13.12	0.60
k=256	13.06	13.44	12.62	0.82
k=512	12.71	12.72	12.71	0.01
Average Improvement				0.48

TABLE IV: Comparison of the RMS angle error of different pooling strategies. Max, Sum and Sumd denote the max, sum and distance weighted pooling layers, respectively.

DSAC achieves a consistent improvement about 0.80 for sum pooling and 1.55 for distance weighted pooling. The latter pooling strategy brings a slight performance drop for PCPNet and a large improvement when DSAC is used. This also validates our algorithm design that distance weighted pooling in the encoder and DSAC are the best partners for normal estimation.

3) MTRNet and Its Variants: We compare various design options for implementing a MoE architecture for normal estimation in Tab. VI. MTRNet-1 uses all the inputs of the experts, and the four sub-networks all have a QST layer independently. MTRNet-2 just uses 512 points of the expert for the largest scale and the QST layer of the manger sub-network is shared by all the experts. We see a 0.13 drop relative to MTRNet-1 in Tab. VI. The RMS of them are both higher than MTRNet-3 and MTRNet because their experts are not pre-trained. Compared with MTRNet, MTRNet-3 uses all the neighborhoods of three scales, and its RMS is 0.16 lower than MTRNet. Although MTRNet is a little inferior than MTRNet-3, but it has smaller model size, memory occupation and far superior than other multi-scale methods, see Tab. II.

D. Efficiency

In Tab. VII, we compare the number of parameters and run time between different deep normal estimators. The reported run time is the average time per point for a batch of size 64, for all the methods except IterNet, on a single Nvidia GTX 2080. We use a batch of size 64 for comparing to the more resource intensive methods, such as Nesti-Net. Other estimators, including ours, can compute in larger batches for faster performance. IterNet uses a fixed batch of size 100k, and it has the lowest number of parameters and run time. Our TRNetis the second fasted. Note that non-direct normal estimators, such as DeepFit, IterNet and our methods, all have smaller model size and faster run time.

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

Neighborhood Size	Sum			Sumd		
	PCPNet	TRNet	Improvement	PCPNet	TRNet	Improvement
k=128	14.85	13.72	1.13	15.13	13.12	2.01
k=256	14.08	13.44	0.64	14.22	12.62	1.6
k=512	13,34	12.72	0.62	13.75	12.71	1.04
Average Improvement	-	-	0.80	-	-	1.55

TABLE V: From PCPNet to TRNet. Generating a latent point set regularized by differentiable RANSAC brings a large improvement than regressing a normal vector directly.

Removed/Replaced component	RMS
MTRNet−1: 512+256+128, self QST, 1 stage	12.17
MTRNet−2: 512, sharing QST, 1 stage	12.30
MTRNet−3: 512+256+128, sharing QST, 2 stages	11.62
MTRNet: 512, sharing QST, 2 stages	11.78

TABLE VI: Variants of MTRNet .

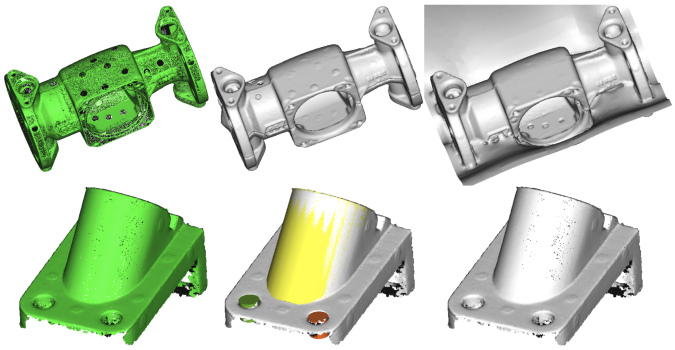

Fig. 6. Two applications on real scan data in CloudCompare. Poisson surface reconstruction is shown in the top row and cylinder detection is shown in the bottom row. From left to right are the input points, results using our normals and normals of CloudCompare.

E. Applications on Real Scans

We further investigate the effectiveness of our estimated normals in the context of Poisson surface reconstruction [16] and primitive detection [27] on real scan data. The two experiments are conducted using CloudCompare, an open-source point cloud editing software, and shown in Fig. 6. Poisson surface reconstruction need oriented normals which are computed with the minimal spanning tree. The results illustrate that the reconstruction using our normals are more faithful than using the normals computed in CloudCompare. The incorrect reconstruction is caused by inconsistent normal orientation due to unfaithful normal estimation. We also detect cylinders from the scanned model shown in the bottom row of Fig. 6. The experiments are conducted five times since the primitive detection method [27] has a certain randomness. We observed that all the three cylinders, rendered in yellow, green and red colors, are detected using our normals and none of them are extracted when the normals provided by CloudCompare are used, as shown in the bottom row of Fig. 6.

V. Conclusion and Future Work

In this work, we propose TRNet for normal estimation of unstructured 3D point clouds. Instead of regressing a normal directly, we learn a powerful latent tangent space representation regularized by DSAC. Besides, our feature fusion mechanism, distance weighted sum pooling, also enhances the coded feature for predicating the latent points. A lightweight multi-scale architecture MTRNet is also proposed to boost the performance of multiple single scale normal estimators. Both TRNet and MTRNet are surprisingly simple and effective. Experiments on the synthesis dataset of PCPNet show that state-of-the-art accurate normals are estimated by our methods without hand-crafted features and complicated networks. Evaluation on scanned scenes of NYU depth V2 dataset also shows that our results preserve sharp features faithfully.

In the future, we plan to explore more approaches to regularize the latent representation for normal estimation and other point cloud analysis missions. Improving our current architecture with advanced convolution [12] or point cloud network [31] is also interesting.

References

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

[2] Yizhak Ben-Shabat and Stephen Gould. Deepfit: 3d surface fitting via neural network weighted least squares. In *ECCV*, 2020. 2, 6, 8, 10

[3] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *CVPR*, June 2019. 1, 2, 3, 6, 8, 10

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

	MTRNet	TRNet	DeepFit [2]	IterNet [17]	Nesti-Net [3]	PCPNet [11]
Num. parameters	6.6 M	3.3 M	3.5 M	**7981**	170.1 M	21.3 M
Run time	1.3	0.6	0.8	**0.04**	8.7	3.4

TABLE VII: Comparison of efficiency between deep normal estimators. Number of model parameters and average run time (ms per point) are listed.

[4] Alexandre Boulch and Renaud Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31(5):1765–1774, 2012. 1, 2

[5] Alexandre Boulch and Renaud Marlet. Deep learning for robust normal estimation in unstructured point clouds. *Computer Graphics Forum*, 35(5):281–290, 2016. 1, 2, 6, 8

[6] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC - differentiable RANSAC for camera localization. In *CVPR*, pages 2492–2500, 2017. 3

[7] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22:121–146, 2005. 2, 8

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

[9] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. 3, 5

[10] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3), July 2007. 2

[11] 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, 3, 5, 6, 8, 10

[12] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Trans. Graph.*, 37(6):235:1–235:12, 2018. 9

[13] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH*, 1992. 1, 2, 8

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

[15] Thouis R. Jones, Frédo Durand, and Matthias Zwicker. Normal improvement for point rendering. *IEEE Computer Graphics and Applications*, 24(4):53–56, 2004. 2

[16] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Symposium on Geometry Processing*, 2006. 9

[17] Jan Eric Lenssen, Christian Osendorfer, and Jonathan Masci. Differentiable iterative surface normal estimation. In *CVPR*, 2020. 2, 6, 8, 10

[18] 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

[19] Ming Liu. Robotic online path planning on point cloud. *IEEE transactions on cybernetics*, 46, 2015. 1

[20] Xiuping Liu, Jie Zhang, Junjie Cao, Bo Li, and Ligang Liu. Quality point cloud normal estimation by guided least squares representation. *Computers & Graphics*, 51:106–116, 2015. 2

[21] Carlos A. Madrigal, John W. Branch, Alejandro Restrepo, and Domingo Mery. A method for automatic surface inspection using a model-based 3d descriptor. *Sensors*, 17(10):2262, 2017. 1

[22] Niloy J. Mitra, An Nguyen, and Leonidas J. Guibas. Estimating surface normals in noisy point cloud data. *Int. J. Comput. Geometry Appl*, 14(4-5):261–276, 2004. 2

[23] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 7

[24] Karl Pertsch Eric Brachmann Carsten Rother Omid Hosseini Jafari, Siva Karthik Mustikovela. ipose: Instance-aware 6d pose estimation of partly occluded objects. In *ACCV*, 2018. 3

[25] Mark Pauly, Richard Keiser, Leif Kobbelt, and Markus H. Gross. Shape modeling with point-sampled geometry. *ACM Trans. Graph.*, 22(3):641–650, 2003. 2

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2

[27] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007. 1, 9

[28] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 3

[29] 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). 1

[30] Jie Zhang, Junjie Cao, Xiuping Liu, He Chen, Bo Li, and Ligang Liu. Multi-normal estimation via pair consistency voting. *IEEE Trans. Vis. Comput. Graph*, 25(4):1693–1706, 2019. 1, 2, 6, 8

[31] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 9

Junjie Cao (Member, IEEE) is an associate professor in School of Mathematical Sciences at Dalian University of Technology, P.R. China. He received Ph.D. degrees in computational mathematics from Dalian University of Technology. His research interests include point cloud processing, human body modeling, and related computer graphics and vision problems.

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIE.2021.3053904, IEEE Transactions on Industrial Electronics

Hairui Zhu is a M.S. candidate of School of Mathematical Sciences at Dalian University of Technology. He received the B.S. in School of Mathematical Sciences at Heilongjiang University. His research interests include shape modeling and computer vision.

Yunpeng Bai received the bachelor's degree in computer science and technology from the Dalian University of Technology, China, in 2020. He is currently pursuing a master's degree at Shenzhen International Graduate School, Tsinghua University, China. His research interests include computer vision and computer graphics.

Jun Zhou is a lecturer in the College of Information Science and Technology at Dalian Maritime University, PR China. He received the Ph.D. degrees in computational mathematics from Dalian University of Technology. His research interests include computer graphics, image processing and machine learning.

Jinshan Pan (Member, IEEE) is a professor of School of Computer Science and Engineering, Nanjing University of Science and Technology. He received the Ph.D. degree in computational mathematics from the Dalian University of Technology, China, in 2017. His research interest includes image deblurring, image/video analysis and enhancement, and related vision problems.

Zhixun Su (Member, IEEE) received the B.Sc. degree in Mathematics from Jilin University in 1987 and M.Sc. degree in Computer Science from Nankai University in 1990. He received his Ph.D. degree in 1993 from Dalian University of Technology, where he has been a professor in the School of Mathematical Sciences since 1999. His research interests include computer graphics, image processing, computational geometry, and computer vision, etc.

0278-0046 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: China University of Geosciences Wuhan Campus. Downloaded on March 30,2021 at 02:48:36 UTC from IEEE Xplore. Restrictions apply.