Normal Estimation for Accurate 3D Mesh Reconstruction with Point Cloud Model Incorporating Spatial Structure

Taisuke Hashimoto Masaki Saito
Preferred Networks, Inc.
Tokyo, Japan
{hashimotot, msaito}@preferred.jp

Abstract

In this paper, we propose a network that can accurately infer normal vectors from a point cloud without sacrificing inference speed. The key idea of our model is to introduce a voxel structure to extract spatial features from a given point cloud. Specifically, unlike the other existing methods directly exploiting point clouds, our model leverages two subnetworks called a "point network" and a "voxel network". The point network extracts local features of a surface from a point cloud, whereas the voxel network transforms the point cloud into voxels and encodes the spatial features from them. The experimental results demonstrate the effectiveness of our method.

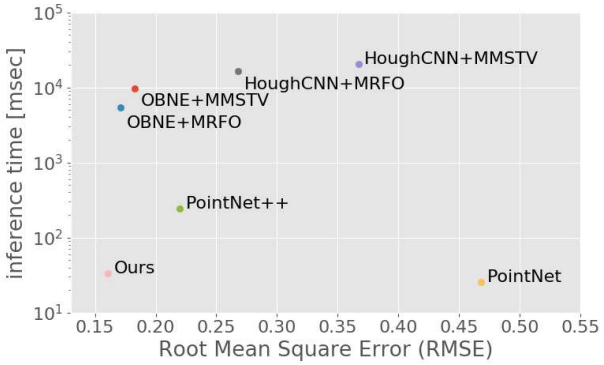


Figure 1. Error of the estimated normal vector and inference time of each method.

1. Introduction

3D mesh reconstruction from point cloud [13, 3, 18] is one of the most prominent problem in computer vision, and used for a wide range of applications including robotics [13, 1], augmented reality [38], and mixed reality [36]. Although many methods have been proposed to solve this problem [3, 18, 28, 29], one of the most typical approaches is to first estimate a normal vector of each point representing the object surface, and then reconstruct the 3D mesh from these normals [7, 34, 18]. If the normals of the point cloud can be estimated with high accuracy, we can make use of that information and obtain the 3D mesh of the object close to the ground truth. In this study, we focus on the normal estimation from the point cloud.

The estimation of the normal vectors mainly consists of the following two steps: i) the "unoriented" normal estimation step to roughly compute a normal vector from neighbouring points, and ii) the alignment step to correct all the unoriented ones so that these normals are consistent (i.e., all vectors face the outside of the surface). Our study deals with both steps in an end-to-end manner. While a typical approach to handle the former step was to exploit several rule-based techniques [42, 22], which aim to improve the accuracy estimated by the method of Hoppe et al. [14], in recent years, several data-driven approaches have been applied to the normal vector estimation [5, 33] in Computer Graphics.

In the field of Computer Vision, using a common CNN model for the point cloud such as PointNet [30] is considered to be effective to handle the normal estimation, however, the typical problem of this model is that they are not suitable for encoding "spatial" structure of a point cloud [31, 16, 20]. This spatial structure is significant in normal estimation that needs to encode information in the vicinity of the point. To cope with these problems, some methods have been proposed so far to encode spatial features by making the PointNet a multi-layered one [31] or extending the data structure of the point cloud [16, 20]. However, these inference speed are much slower than that of PointNet, as these methods directly add higher-order extensions to the data structure. To reconstruct the 3D mesh at practical speed and with high accuracy, we needed to develop a model for a point cloud that efficiently encode both local and spatial information with a simpler approach.

Based on the above discussion, we propose a simple but effective network for normal estimation that efficiently in-

corporates local and spatial structures into the PointNet. Our idea is to incorporate a *voxel network* that can efficiently represent the spatial structure and its inference speed is relatively fast. That is, by making use of both the point network to extract the local features such as neighboring points and the voxel network to extract spatial representation, our model efficiently encodes both structures.

The clear advantage of our model is that it can accurately extract the both local and spatial features required for the normal estimation without sacrificing the speed (Figure 1). In the experiments, we observed that our method not only outperforms other existing methods in terms of the root mean squared error, but its inference speed is equal or faster than the major existing models dealing with point clouds. It illustrates that our model could efficiently encode both the local and the spatial structures.

The rest of this paper is organized as follows. In Section 2, we review several studies related to our work. Section 3 presents the detail of our network structure. The experimental results are presented in Section 4. Section 5 concludes the paper.

2. Related work

2.1. Unoriented normal estimation in point clouds

A typical approach for estimating normal vectors from a point cloud was to compute a tangential plane of each 3D point with Principal Component Analysis (PCA) of its neighboring points [14]. After that, several studies have been proposed to improve its accuracy by assuming more complex shapes such as a sphere [12] and a quadratic surface [8]. As a similar extension, there are also several studies that leverage a Delaunay triangulation [2, 10] for normal estimation. However, it is known that the above methods are vulnerable to an object including a pointed shape [4]. Although some studies have been proposed to handle such an intractable shape with some sophisticated algorithms such as clustering [43, 24] and Randomized Hough Transform [4], these methods are ineffective for the objects including smooth surfaces because both methods perform discretization using a kind of clustering.

Recently, Boulch and Marlet [5] proposed a method for improving the accuracy of normal vectors by computing Hough space [4] with CNN. However, we argue that it has the same problem of [4] since this method also uses the Hough transformation.

2.2. Consistent normal orientation in point clouds

To obtain consistent normal vectors, after applying the above methods we usually need to perform another filter that aligns all the normals to be consistent because some estimated vectors face inside the object surface. There are roughly two approaches to align normal vectors: i) a geometric approach and ii) a volumetric approach.

The geometric approach iteratively aligns neighboring points so that all of the normal vectors of these points are consistent [14, 42, 12, 15, 22]. For example, Hoppe et al. [14] proposed a method that first selects a specific point as a "seed", and iteratively corrects the estimated normals of neighboring points with Minimum Spanning Tree (MST). Schertler et al. [33] formulated this problem as a kind of energy minimization problem, and efficiently aligned normal vectors with some minimization methods instead of the MST. As other methods that do not leverage the MST, a method with an adaptive spherical cover scheme [23], and a method that combines Laplacian smoothing and visibility voting [6] have been proposed.

The volume-based approach exploits volumetric representation such as a signed distance function [44, 26, 17, 27, 11] and estimates whether a normal vector of a point faces inward or outward. These methods are known to be robust against outliers, however, its computational cost is generally higher than surface-based ones.

2.3. Deep neural network

Deep Neural Networks (DNNs) have recently shown outstanding performance in various 3D recognition tasks such as RGB-D object recognition [25] and semantic segmentation with point clouds [30]. The problem in the 3D object recognition using DNN is how to represent a data structure for efficiently solving the problem. Although a naive approach is to represent its 3D shapes by voxels, it has a disadvantage that its accuracy generally depends on the "resolution" of voxels, and increase in resolution causes a significant increase in computational cost. Several methods have been proposed to reduce this computational cost with an octree [32, 40], however, they are directly inapplicable to the normal vector estimation from point clouds because each point needs to be stored in a single leaf.

The existing models which are most similar to our proposed one are to directly regard a set of 3D points as an input of DNN [30, 31, 35]. Although these sophisticated models can efficiently solve a semantic segmentation problem that requires global information such as object categories, it is still unclear whether these are also effective for the normal estimation problem that needs both local and spatial features.

3. Model

This section describes our proposed model. As shown in Figure 2, our model consists of two subnetworks called the point network and the voxel network. We respectively introduce the detail of these networks in Section 3.1 and Section 3.2, and describe the method for integrating these feature vectors in Section 3.3. As another application of our network, we also explain a method to remove the noise from the point cloud in Section 3.5.

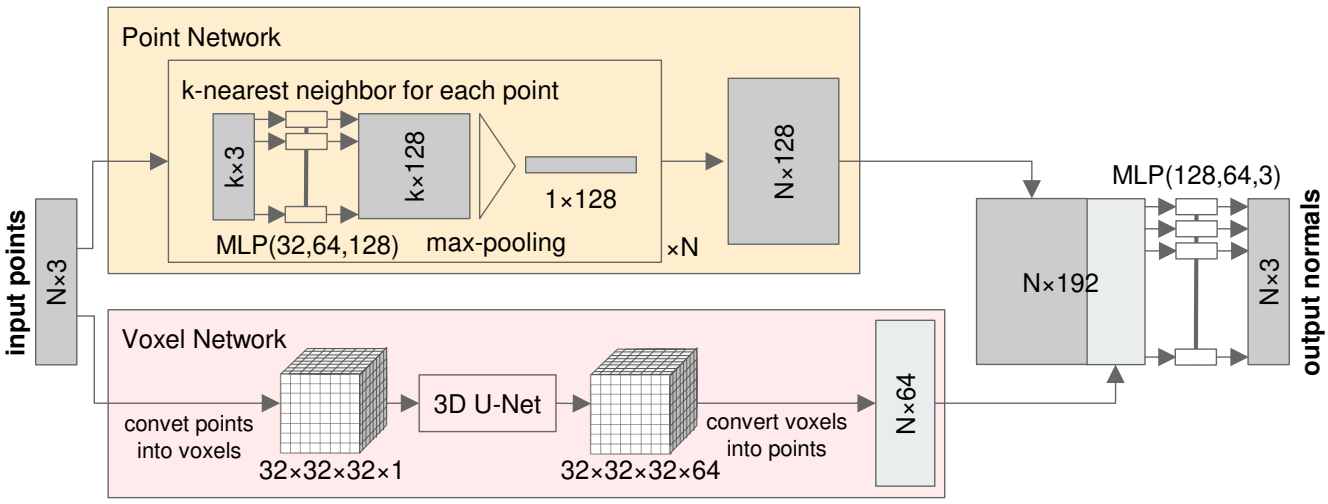


Figure 2. Illustration of our proposed network. It consists of two subnetworks called the point network and the voxel network. The point network extracts local features corresponding to the local shape of an object, whereas the voxel network first transforms a point cloud into voxels, and extracts spatial features with a 3D U-Net. These two different feature vectors are combined into one vector, and transformed into a normal vector by a learnable network. $\mathrm{MLP}(c_1, c_2, c_3)$ denotes the three-layer perceptron where the numbers of the channels in the input, the middle, the last layers are c_1, c_2, and c_3, respectively. In the MLP we applied the batch normalization and the ReLU after linear transformation except for the last layer.

3.1. Point network

The point network is a variant of PointNet [30] and focuses on efficiently extracting local features from a given point cloud. Similar to the PointNet, the point network is a function that receives normalized N three-dimensional coordinates denoted by $\mathbf{x} \in [-1, 1]^{N\times 3}$, and returns a set of feature vectors with L elements ($\mathbf{h}^{\mathrm{local}} \in \mathbb{R}^{N\times L}$) that represents local shapes of a surface.

We explain the detail of the point network. Let $x_i \in [-1, 1]^3$ denote i-th element of $\mathbf{x}$, and $\{x_i^1, \ldots, x_i^k\}$ be a set of nearest k points from point i. Using this notation, l-th element of the output $h_{i,l}^{\mathrm{local}}$ can be represented by

$$h_{i,l}^{\mathrm{local}} = \max_{j=1,\ldots,k} \{f(x_i^j - x_i)\}_l, \tag{1}$$

where $f : \mathbb{R}^3 \mapsto \mathbb{R}^L$ is a learnable function that maps the 3D coordinate into the L-dimensional feature vector, and $\{\cdot\}_l$ is the l-th element of the vector in the braces. In the implementation, we set $k = 32$, and defined f by a three-layer perceptron consisting of batch normalization functions and rectified linear units.

Unlike the PointNet that applies the global max-pooling function against all the points to encode the local features, our point network computes its maximum value only from neighboring points of a certain point. Although this algorithm itself is similar to that in PointNet++ [31], which also encodes local features with neighboring points, their approach and ours differ in the following two points. Firstly, before computing local features, the PointNet++ samples some representative points from a point cloud, whereas our network does not perform such sampling process to support sudden change between adjacent normal vectors. Secondly, while PointNet++ repeats both of sampling and feature extraction multiple times to extract spatial features from the point cloud, our network does not need to repeat it multiple times because it delegates the task of extracting spatial features to the voxel network. This can simplify not only the configuration of the point network, but also has another advantage that the inference speed significantly improves. The detail of this effect will be shown in Section 4.5.

3.2. Voxel network

The normal estimation with the point network often produces inconsistent results because the point network only encodes local features. We compensate this shortcoming by introducing the voxel network.

Unlike the point network that directly exploits the 3D points, the voxel network (1) transforms all the points into voxels, (2) extracts spatial feature vectors with 3D CNN, and (3) converts them into points that contain these features. We describe the detail of these steps in the following.

Firstly, the voxel network projects normalized N three-dimensional points into voxels each of which contains a binary value. These voxels are mathematically expressed as $\mathbf{x}^{\mathrm{voxels}} \in \{0, 1\}^{D\times D\times D}$, where D denotes the resolution of the voxel. In the experiments we set $D = 32$. Each voxel represents whether any points are included or not, i.e., if any points in a point cloud are included in a voxel, its value is equal to one. After that, the voxel network memorizes correspondence between a voxel and stored points.

Next, it extracts the spatial features with 3D U-Net [9],

which is a standard model for semantic segmentation of 3D models. The 3D U-Net is expressed by two networks called an encoder and a decoder; the encoder extracts the spatial features of voxels while lowering the resolution, whereas the decoder reconstructs voxels with the same resolution by gradually increasing it from the feature maps obtained by the encoder. After that, we have a set of voxels denoted by $\mathbf{h}^{\text{voxels}} \in \mathbb{R}^{D\times D\times D\times M}$, where M is the dimension of the feature vector in a voxel.

Finally, with the correspondence table, the voxel network transforms these voxels into N points containing M-dimensional feature vectors. For simplicity, we represent these points by $\mathbf{h}^{\text{spatial}} \in \mathbb{R}^{N\times M}$. The network configuration of the voxel network is exactly the same as the original one [9] except we set the number of channels in the initial layer to one and that in the final layer to 64.

3.3. Integration

The above two feature vectors computed by the point and the voxel networks are finally combined into one vector, and transformed into a normalized vector with another network. We call this network *integration network*. The integration network consists of a three-layer perceptron represented as $\mathrm{MLP}(128, 64, 3)$; it first takes this feature vector as an input and returns unnormalized 3D vector representing a normal. By performing this process for all the points and normalize them, we finally have a set of normalized vectors denoted by $\mathbf{y} \in \mathbb{R}^{N\times 3}$. Note that although the integration network itself does not consider the relationship between neighboring points, such relations are already taken into account by both the point and the voxel networks.

3.4. Loss function

In the training of the whole network, we employ the L1 norm (a.k.a. Mean Absolute Error) defined by

$$\mathcal{L} = \frac{1}{N\times 3}\sum_{i=1}^{N} \|\hat{n}_i - n_i\|_1, \tag{2}$$

where n_i and $\hat{n}_i$ denotes the i-th normal vector and its ground truth, respectively.

Note that the main reason we have mainly employed the L1 norm is to reconstruct the object containing a sharp shape. We will describe these experimental analyses in Section 4.6.

3.5. Denoising

The point cloud retrieved from the sensor contains some noise. Although a straightforward approach to deal with this problem is to directly estimate the normal vectors from the point cloud with noise, there is another method that explicitly inserts the process of removing noise into the whole process, i.e., first estimates the "clean" 3D points from the point cloud, and then obtains these normal vectors. In this section, we describe the detail of the latter model.

As we described above, this network consists of the two subnetworks: the initial network for denoising and the second network for normal estimation. The initial network takes N 3D points with perturbation noise ($\mathbf{x} + \epsilon \in [-1, 1]^{N\times 3}$) as an input, and returns a tensor corresponding to the estimated perturbation noise (ϵ). The network structure of the initial network is the same as the network for normal estimation. For the training, we simply employ the mean squared error represented by $\mathcal{L}^{\text{noise}} = \left(\sum_i \|\hat{\epsilon}_i - \epsilon_i\|_2^2\right)/(3N)$, where $\hat{\epsilon}_i$ denotes the perturbation noise. After estimating the perturbation noise, we finally have the normal vectors by passing the corrected points to the network for the normal estimation.

4. Experiments

This section describes the experimental results of our method. The existing methods we used for the comparative experiments are the following. We selected an Orientation-Benefit Normal Estimation (OBNE) [22] and HoughCNN [5] as existing methods for unoriented normal estimation (see Section 1). Regarding the alignment step, we selected a "multi-source orientation propagation with visibility voting" (MMSTV) algorithm [22] and an energy minimization method with Markov Random Field Optimization (MRFO) [33]. The PointNet [30] and PointNet++ [31] were used as the representative methods using point clouds. That is, we compared our method with the existing six methods denoted by "OBNE + MMSTV", "OBNE + MRFO", "HoughCNN + MMSTV", "HoughCNN + MRFO", "PointNet", and "PointNet++".

In this section, we refer to our model consisting of the point network and the voxel network as "Ours (point + voxel)" or simply "Ours". In contrast, "Ours (point only)" and"Ours (voxel only)" respectively mean our network only using the point network and that just using the voxel network.

4.1. The setting of the experiments

4.1.1 Dataset

We used the two datasets called *ModelNet40* [41] and *SHREC15* [21] for the experiments.

- ModelNet40: a 3D dataset consisting of 12,311 CAD models, which contain many artificial objects including pointed shapes. Each object belongs to 40 different categories such as *airplane* and *bookshelf*. As with the original dataset, we used 9,843 models as the training samples and 2,468 models as the test samples. Since some models have normals facing the inside of the object, we collected them with a method by Takayama et al. [37].

- SHREC15: a dataset containing 1,200 non-rigid 3D models with 60 labels such as *camel* and *hand*. Unlike the ModelNet40 this dataset contains many smooth surfaces rather than the sharp shapes. Each category contains one basic 3D shapes and different 20 shapes obtained by deforming its joint angle. Note that we used this dataset for testing only.

We created a set of point clouds of the above datasets by uniformly sampling N points from the surface of each object. To create a normal vector from a 3D point, we regarded a normal vector on a mesh as that on a 3D point. All the samples in the dataset were properly normalized so that its mean is zero and all the points are in a unit ball. In the testing, we set the number of the points N to 10,000. The number of points used in training will be shown later.

4.1.2 Data augumentation

In training, we used a data augmentation method consisting of the following two ways: random rotation of points, and random sampling of points. Specifically, in the latter method, we randomly sampled N points[1] without overlap from a point cloud and regarded them as an input of each model. Note that in the training of the PointNet++, we did not use the above random sampling and used a fixed number of points 10,000. We believe that this is an advantageous setting for the PointNet++ since it is equivalent to the number of samples used for the testing.

Note that regarding the experiments of Section 4.3, in addition to the above two methods, we also employed another data augmentation algorithm which adds a Gaussian noise with a standard deviation σ. Regarding the value of σ, we followed the other existing methods [22, 24, 4, 5]. Specifically, letting d be the average of distances from a point to the nearest point, we can represent σ as $\sigma = R \times d$.

In training, we randomly determined the intensity of the perturbation noise R within a range of 0% to 200%. During the test, we respectively set its intensity R to 0%, 40%, 80%, 120%, 160%, and evaluated the robustness of each model against the noise.

4.1.3 Training configuration

We used Chainer [39] in the implementation of our network. The network was trained for 100 epochs using the Adam optimizer [19] with $\alpha = 0.001$, and a batch size of 4. The training took about 60 hours with a single Tesla P100.

4.1.4 Comparative methods

We implemented the PointNet for the semantic segmentation with Chainer. In the implementation of the PointNet++, we retrieved the code for the semantic segmentation from their project page, replaced the number of channels in the initial and final layer to three, and used them for the experiments. As with the above methods, we also retrieved codes of the OBNE, the MMSTV, the HoughCNN, and the MRFO from their project pages. Note that we appropriately optimized the parameters of the existing methods as much as possible to improve the accuracy (see our supplementary material for details).

[1] N is randomly determined with a discrete uniform distribution from 4,000 to 16,000

Table 1. The RMSE and the PGP of the existing and our methods.

Method	ModelNet40		SHREC15	
	RMSE	PGP	RMSE	PGP
OBNE + MMSTV [22]	0.574	0.456	0.187	0.719
OBNE [22] + MRFO [33]	0.606	0.434	0.171	0.718
HoughCNN [5] + MMSTV [22]	0.611	0.530	0.367	0.444
HoughCNN [5] + MRFO [33]	0.598	0.541	0.268	0.462
PointNet [30]	0.582	0.210	0.468	0.121
PointNet++ [31]	0.451	**0.712**	0.220	0.728
Ours (point only)	0.513	0.596	0.437	0.648
Ours (voxel only)	0.533	0.447	0.328	0.292
Ours (point + voxel)	**0.372**	0.708	**0.161**	**0.787**

4.2. Normal estimation from point cloud

4.2.1 Quantitative results

We evaluated the existing methods and ours with the normal vector estimation from the ModelNet40 and the SHREC15. In the quantitative evaluation, we employed the two metrics called Root Mean Squared Error (RMSE) and Proportion of Good Points (PGP).

The Root Mean Squared Error (RMSE) is a standard metric for quantitatively evaluating the performance. In these experiments, the RMSE can be defined as $\mathrm{RMSE} = \sqrt{\left(\sum_i |\hat{n}_i - n_i|^2\right) / (3N)}$. Since it is known that the RMSE is not robust to the large error, the most effective method to reduce the RMSE is that the direction of the normal faces the outside of the object as much as possible. It means that considering spatial features of an object is much significant to reduce the RMSE. On the other hand, we argue that it is unsuitable when the object contains a pointed shape since the RMSE is a vulnerable metric to outliers.

The PGP [5] is another metric aiming to address this problem and can be defined by the proportion of points in which every angle between the estimated normal and its ground truth is smaller than threshold τ. Even if the estimated normal faces outside of the object, this metric regards it as a "failure" when the error exceeds the threshold. It means that considering local features on the surface is much significant to increase the PGP. We employed the PGP as another metric and set τ to 10° (that is the same as the value in [5, 24]).

Table 1 shows the experimental results of the RMSE and the PGP. It is interesting to see that while the RMSE of the

PointNet was comparable to others, its PGP score was significantly lower than those of the other models. These results show that although the PointNet is suitable for encoding global features, it is not sufficient for extracting local and spatial features. We consider it is mainly due to the effect of the global max-pooling layer.

In contrast, the PointNet++, which is an improved version of PointNet, outperformed other existing models. Specifically, the PGP score of the ModelNet40 achieved by the PointNet++ was slightly larger than that of ours. Although it seems to indicate that the PointNet++ successfully extracted the both the local and the spatial features, its RMSE was significantly lower than ours. We will describe its reason in Section 4.2.2.

It is also interesting to see that our network only containing the point network outperformed some existing methods. In the case where the network can use local features only, although it can correctly estimate the normals perpendicular to the surface, it cannot detect whether the normal faces the outside of the object. For this reason, it seems that the PGP obtained by the point network has an upper limit of 0.5. We consider the reason why it could achieve the value higher than 0.5 is that the point network statistically estimated the direction of the normal by the curvature of the surface. We can also see that the PGP of the voxel network in the SHREC15 was relatively lower than those of other methods even though its RMSE was superior. It implies that the voxel network failed to accurately estimate normal vectors due to the discretization by introducing voxels.

Table 1 also shows that the result of our network with both the point and the voxel networks outperformed other models except for the PGP of PointNet++. It indicates that our network could efficiently encode both the local and the spatial features by introducing the two different networks.

4.2.2 Qualitative results

We also visualized the angle between the estimated normal vector and the ground truth. Figure 3 shows the visualization result of five objects retrieved from the test samples in the ModelNet40 and the SHREC15. As with our method, it seems that the PointNet++ encoded both the local and the spatial features, however, we can also see that some normal vectors estimated by the PointNet++ faced inside of the object. As we can see in the results of the SHREC15 (the right two columns in Figure 3), this problem occurred when the surface is concave, and this problem could not be solved without encoding the spatial features more appropriately.

4.3. Normal estimation from noisy point cloud

4.3.1 Quantitative results

Many 3D points obtained by a sensor include some noises. We evaluated the robustness of the proposed model by adding artificial noise to the point cloud and estimated these normal vectors.

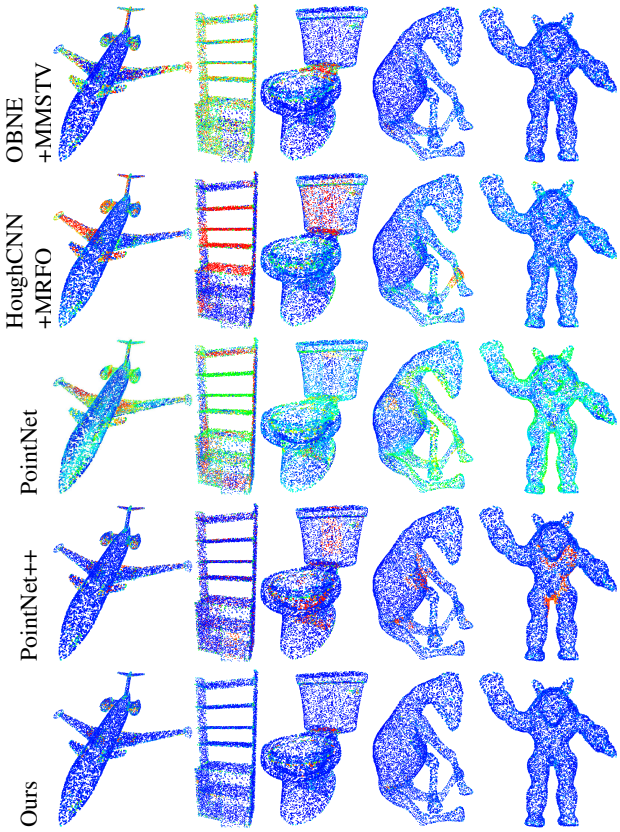


Figure 3. Visualization of normal vectors estimated by five different methods. The angle of the error vector is projected to hue in HSV color space. The blue color shows that the vector is vertical to the object surface and its direction faces outside, whereas the red color indicates that the vector is perpendicular but its direction is opposite. Light blue, orange, and green colors mean that the vector is not vertical.

In terms of our model, we employed the two methods to estimate normal vectors from a point cloud with noise. The former is a model that directly calculates the normal vectors in an end-to-end manner, and the latter is a model that first estimates 3D points without noise, and then computes normal vectors from these points with another network consisting of the point and the voxel networks. In this section, we refer to the latter as "Ours (+denoising)".

To confirm the robustness against the intensity of noise, we trained our two models and the existing models while adding perturbation noise drawn from a Gaussian distribution with a standard deviation σ (see Section 4.1 for details).

Figure 4 shows the relation between the RMSE, the PGP, and the intensity of the perturbation noise. We can see that the PointNet++ and our method (the point and the voxel networks) were relatively robust to perturbation noise since they can encode both features. It also reveals that the per-

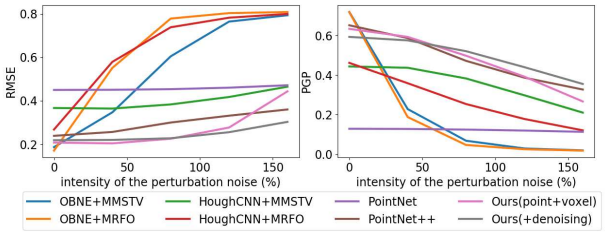


Figure 4. The PGP and the RMSE curves w.r.t. the intensity of perturbation noise.

formance of the PointNet, which mainly encodes the global features, was worse than those of others. This implies that leveraging both of the local and the spatial features is significant for normal estimation from the point cloud including noise.

Figure 4 also shows that compared with our model only using a single network, the RMSE of our method using two networks significantly decreased by 32% and its PGP also increased by 33%. We describe this reason in the supplementary material.

4.4. Application to Surface Reconstruction

We also show the results of the surface reconstruction from the point clouds in Figure 5. Specifically, we estimated the normal vectors with several methods including ours and then obtained the 3D surface of an object by applying a Screened Poisson Reconstruction [18], a de facto standard algorithm for estimating the 3D surface from the normal vectors of 3D points. From the aspect of the experimental result with a chair, the PointNet generated noisy 3D surfaces, and a part of the surface created by the PointNet++ bulges unnaturally. Although the result with cow looks like that every method can precisely estimate the 3D surfaces, it can be seen that the surfaces of toes, horns, and ears generated by the existing methods also slightly bulge. In particular, the PointNet++ could not precisely estimate several parts such as the ears and the belly. These results indicate that while the existing methods often failed to precisely infer the 3D surfaces because the estimation result of the normal vectors was not so accurate, the proposed method was able to estimate the 3D mesh close to the ground truth.

4.5. Inference Time

To compare the inference time in each method, we measured the average time per sample with 1,200 models in the SHREC15. The results are shown in Table 2. Note that in this table, our model includes time to convert the point cloud to the voxels and time required for its inverse transform. We can see that the PointNet++ consumed more time than the PointNet. It is due to the internal structure of the PointNet++, which iteratively performs the processing of the PointNet. In contrast, the computational cost of our model was much lower than that of PointNet++ because it performs the operation like PointNet only once. Although this inference time was slightly slower than the PointNet, considering the RMSE and the PGP, it reveals that our method has a clear advantage over the existing ones.

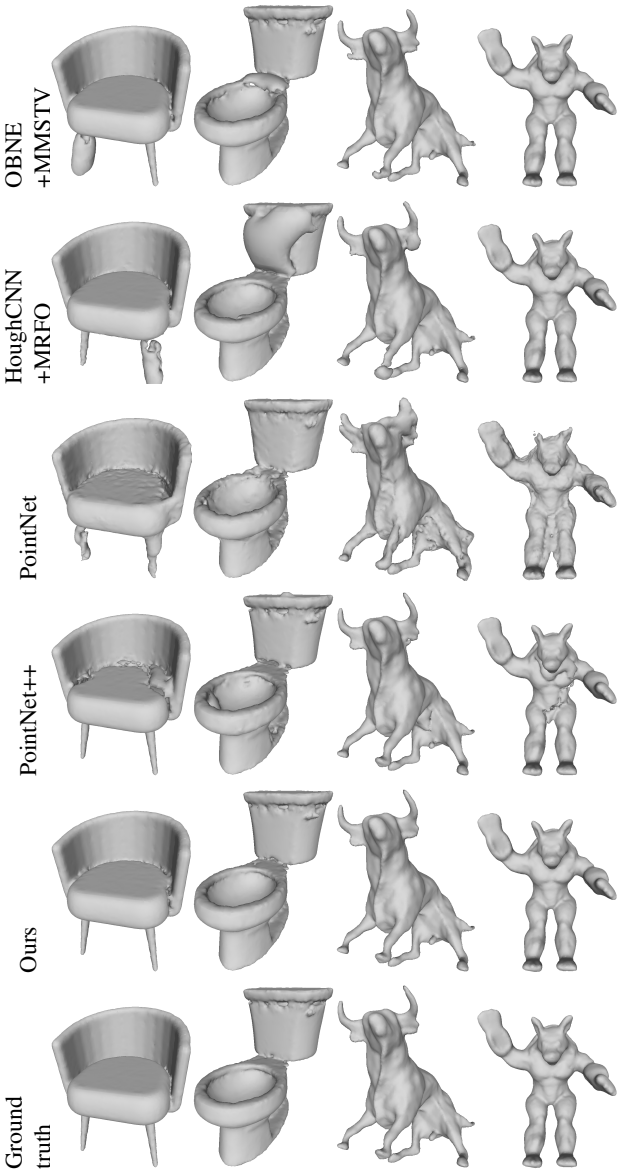


Figure 5. The 3D reconstruction from the estimated normal vectors. "ground truth" means the 3D mesh reconstructed from the ground truth vectors.

Table 2. The inference time of the existing and our methods.

Method	OBNE +MMSTV	OBNE +MRFO	HoughCNN +MMSTV	HoughCNN +MRFO
msec/model	9643	5392	20743	16392
Method	PointNet	PointNet++	Ours	
msec/model	25.8	243	33.4	

Table 3. The RMSE of ModelNet40 for each combination of k and D.

k \ D	8	16	32	64
8	0.457	0.422	0.410	0.413
16	0.439	0.408	0.377	0.395
32	0.433	0.391	0.377	0.388
64	0.424	0.397	0.378	0.424

4.6. Experiments on the change of parameters

4.6.1 Effect of voxel resolution and number of nearest neighbors

To see the effectiveness of the voxel resolution and the number of nearest neighbors, we changed the voxel resolution D to 8, 16, 32, 64, the number of nearest neighbors k to 8, 16, 32, 64, and measured RMSE for each parameter. Note that the other parameter settings are equal to Section 4.2. The results are shown in Table 3. It indicates that increasing the resolution of a voxel is more significant than increasing the value of k to accurately estimate normals. However, it also means that setting very large k and D will adversely affect the overall accuracy.

4.6.2 Comparison between loss functions

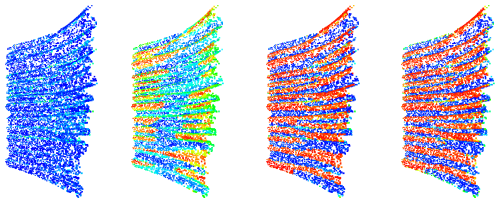
We also conduct additional experiments with different loss functions. Specifically, in this experiments we introduced four loss functions called L1 norm, L2 norm, Mean of Angles between normals (MA), Mean Square of Angles (MSA), and measured these performances. Note that in the above experiments, to measure the robustness of the loss functions against three-dimensional rotation, we randomly rotated the point clouds of the test samples in evaluation phase. Thus, in Section 4.2.2 we randomly rotated the training samples but did not perform it in the testing one, whereas in these experiments we applied random rotation to both samples. This is to measure the the performance when using loss functions dealing with angles such as MA and MSA.

These results are shown in Table 4. It illustrates that regarding the RMSE and the PGP10, the performances of L1 and L2 norms were almost the same, whereas the value of PGP05 using L2 norm was significantly lower than L1 norm. We also observed that when estimating the surface of a sharp object with L2 norm, the reconstructed surface became relatively smooth. For these reasons, we concluded that introducing the L1 norm is effective for the normal estimation of objects including sharp shapes.

Interestingly, even when using loss functions with angles directly, these performances are almost the same as those of Lp norms. This means that minimizing the angle directly does not significantly affect the actual performance improvement. We also observed that even if we rotated to the test samples, its performance is nearly the same as on that does not apply 3D rotation (see Table 1). It shows that our model has acquired robustness against rotation.

Table 4. The RMSE and the PGP($\tau = 5°, 10°$) of rotated ModelNet40 for each loss function.

metric \ loss	L1	L2	MA	MSA
RMSE	0.374	**0.365**	**0.365**	0.400
PGP10	**0.699**	0.660	0.696	0.454
PGP05	0.557	0.447	**0.564**	0.215



OBNE +MRFO | PointNet | PointNet++ | Ours

Figure 6. The visualization result of the models with a curtain in the ModelNet40.

4.7. Discussion

Although our proposed model outperformed the other existing models in both the ModelNet40 and the SHREC15, the above results indicate that it was not able to successfully estimate the normals of several objects. For example, in shapes where the curvature of the surface changes continuously (e.g., curtain in Figure 6), the existing methods outperformed our method. We consider this is because that the machine learning-based methods like including ours inferred normal vectors from the statistical information of curvature. Although this leads to more accurate normal estimation in many cases, it also means that inference using prior rarely fails. It could be improved by increasing the number of samples used for the training; the number of the training samples corresponding to "curtain" is only 138, and it is not so large as the number of samples. It indicates that the proposed method is significantly affected by the training samples.

5. Conclusion

We proposed a model that exploits two subnetworks called the point network and the voxel network, and estimates the normal vectors from a given point cloud. This model can not only precisely infer the normals compared with the other methods, but its inference speed is comparable to the PointNet.

Acknowledgements We would like to acknowledge Eiichi Matsumoto and Shunta Saito for helpful discussions. We also would like to acknowledge Shohei Hido for supporting this research.

References

[1] Jacopo Aleotti and Stefano Caselli. A 3D Shape Segmentation Approach for Robot Grasping by Parts. *Robot. Auton. Syst.*, 60(3):358–366, 2012. 1

[2] Nina Amenta and Marshall Bern. Surface Reconstruction by Voronoi Filtering. In *Proc. of Annual Symposium on Computational Geometry*, pages 39–48, 1998. 2

[3] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. A Survey of Surface Reconstruction from Point Clouds. *Computer Graphics Forum*, 36(1):301–329, 2017. 1

[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. 2, 5

[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, 4, 5

[6] Junjie Cao, Ying He, Zhiyang Li, Xiuping Liu, and Zhixun Su. Orienting raw point sets by global contraction and visibility voting. *Computers & Graphics*, 35(3):733–740, 2011. 2

[7] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *Proc. of SIGGRAPH*, pages 67–76, 2001. 1

[8] Frédéric Cazals and Marc Pouget. Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets. In *Proc. of Eurographics Symposium on Geometry Processing*, pages 177–187, 2003. 2

[9] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Proc. of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432, 2016. 3, 4

[10] Tamal K. Dey and Samrat Goswami. Provable Surface Reconstruction from Noisy Samples. *Comput. Geom. Theory Appl.*, 35(1):124–141, 2006. 2

[11] Simon Giraudot, David Cohen-Steiner, and Pierre Alliez. Noise-adaptive Shape Reconstruction from Raw Point Sets. In *Proc. of Eurographics Symposium on Geometry Processing*, pages 229–238, 2013. 2

[12] Gaël Guennebaud and Markus Gross. Algebraic Point Set Surfaces. *ACM Trans. Graph.*, 26(3), 2007. 2

[13] Dirk Hähnel, Wolfram Burgard, and Sebastian Thrun. Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robot. Auton. Syst.*, 44(1):15–27, 2003. 1

[14] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface Reconstruction from Unorganized Points. pages 71–78, 1992. 1, 2

[15] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of Unorganized Point Clouds for Surface Reconstruction. *ACM Trans. Graph.*, 28(5):176:1–176:7, 2009. 2

[16] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent Slice Networks for 3D Segmentation on Point Clouds. In *Proc. of CVPR*, pages 9397–9406, 2018. 1

[17] Andrei C Jalba and Jos BTM Roerdink. Efficient surface reconstruction from noisy data using regularized membrane potentials. *IEEE Trans. Image Processing*, 18(5):1119–1134, 2009. 2

[18] Michael Kazhdan and Hugues Hoppe. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, 2013. 1, 7

[19] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*, 2015. 5

[20] Jiaxin Li, Ben M Chen, and Gim Hee Lee. SO-Net: Self-Organizing Network for Point Cloud Analysis. In *Proc. of CVPR*, pages 9397–9406, 2018. 1

[21] Z. Lian, J. Zhang, S. Choi, H. ElNaghy, J. El-Sana, T. Furuya, A. Giachetti, R. A. Guler, L. Lai, C. Li, H. Li, F. A. Limberger, R. Martin, R. U. Nakanishi, A. P. Neto, L. G. Nonato, R. Ohbuchi, K. Pevzner, D. Pickup, P. Rosin, A. Sharf, L. Sun, X. Sun, S. Tari, G. Unal, and R. C. Wilson. Non-rigid 3D Shape Retrieval. In *Proc. of Eurographics Workshop on 3D Object Retrieval*, pages 107–120, 2015. 4

[22] Jian Liu, Junjie Cao, Xiuping Liu, Jun Wang, Xiaochao Wang, and Xiquan Shi. Mendable Consistent Orientation of Point Clouds. *Comput. Aided Des.*, 55:26–36, 2014. 1, 2, 4, 5

[23] Shengjun Liu and Charlie CL Wang. Orienting unorganized points for surface reconstruction. *Computers & Graphics*, 34(3):209–218, 2010. 2

[24] 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, 5

[25] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *Proc. of IROS*, pages 922–928, 2015. 2

[26] Viní cius Mello, Luiz Velho, and Gabriel Taubin. Estimating the in/out Function of a Surface Represented by Points. In *Proc. of ACM Symposium on Solid Modeling and Applications*, pages 108–114, 2003. 2

[27] Patrick Mullen, Fernando De Goes, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. Signing the unsigned: Robust surface reconstruction from raw pointsets. *Computer Graphics Forum*, 29(5):1733–1741, 2010. 2

[28] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. of IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 1

[29] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D Reconstruction at Scale Using Voxel Hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11, 2013. 1

[30] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of CVPR*, pages 77–85, 2017. 1, 2, 3, 4, 5

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of NIPS*, pages 5099–5108, 2017. 1, 2, 3, 4, 5

[32] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. In *Proc. of CVPR*, 2017. 2

[33] Nico Schertler, Bogdan Savchynskyy, and Stefan Gumhold. Towards Globally Optimal Normal Orientations for Large Point Clouds. *Computer Graphics Forum*, 36(1):197–208, 2017. 1, 2, 4, 5

[34] Shy Shalom, Ariel Shamir, Hao Zhang, and Daniel Cohen-Or. Cone Carving for Surface Reconstruction. *ACM Trans. Graph.*, 29(6):150:1–150:10, 2010. 1

[35] Barnabas Poczos Siamak Ravanbakhsh, Jeff Schneider. Deep Learning with Sets and Point Clouds. In *Proc. of ICLR - workshop track*, 2017. 2

[36] Keng Hua Sing and Wei Xie. Garden: A Mixed Reality Experience Combining Virtual Reality and 3D Reconstruction. In *Proc. of CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2016. 1

[37] Kenshi Takayama, Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. A Simple Method for Correcting Facet Orientations in Polygon Meshes Based on Ray Casting. *Journal of Computer Graphics Techniques (JCGT)*, 3(4):53–63, 2014. 4

[38] Yuan Tian, Yan Long, Dan Xia, Huang Yao, and Jincheng Zhang. Handling Occlusions in Augmented Reality Based on 3D Reconstruction Method. *Neurocomputing*, 156:96–104, 2015. 1

[39] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proc. of Workshop on Machine Learning Systems in NIPS*, 2015. 5

[40] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.*, 36(4):72:1–72:11, 2017. 2

[41] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proc. of CVPR*, pages 1912–1920, 2015. 4

[42] Hui Xie, Jianning Wang, Jing Hua, Hong Qin, and Arie Kaufman. Piecewise C1 Continuous Surface Reconstruction of Noisy Point Clouds via Local Implicit Quadric Regression. In *Proc. of IEEE Visualization (VIS)*, pages 91–98, 2003. 1, 2

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

[44] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast Surface Reconstruction Using the Level Set Method. In *Proc. of IEEE Workshop on Variational and Level Set Methods (VLSM)*, pages 194–201, 2001. 2