EllipsoidNet: Ellipsoid Representation for Point Cloud Classification and Segmentation

Anonymous CVPR 2021 submission

Paper ID 6998

Abstract

Point cloud patterns are hard to learn because of the implicit local geometry features among the orderless points. In recent years, point cloud representation in 2D space has attracted increasing research interest since it exposes the local geometry features in a 2D space. By projecting those points to a 2D feature map, the relationship between points is inherited in the context between pixels, which are further extracted by a 2D convolutional neural network. However, existing 2D representing methods are either accuracy limited or time-consuming. In this paper, we propose a novel 2D representation method that projects a point cloud onto an ellipsoid surface space, where local patterns are well exposed in ellipsoid-level and point-level. Additionally, a novel convolutional neural network named EllipsoidNet is proposed to utilize those features for point cloud classification and segmentation applications. The proposed methods are evaluated in ModelNet40 and ShapeNet benchmarks, where the advantages are clearly shown over existing 2D representation methods.

1. Introduction

Point cloud is a widely used data structure that represents the 3D objects using an orderless list of scanned points. In recent years, tremendous research efforts have been conducted on point cloud pattern recognition in the field of automated driving [3, 32, 51, 19, 53, 37, 21] and indoor/outdoor scene recognition [40, 41, 46]. As concluded in Guo *et al.* [10], 2D representation learning is one of the promising approaches to extract local and global features from point clouds. 2D representations, including multi-view representation [44], spherical representation [37, 3, 58, 32, 51], and bird-eye-view representation [53, 19], are applied to point cloud learning tasks because they benefit from 2D convolutional neural networks (CNNs), which are well studied in image learning tasks [11, 42, 14, 26]. However, existing 2D representations methods either require point clouds from specific sensors (*e.g.*, LiDARs [32, 37, 51]) or have huge point loss rate without a sufficient study of impact to related applications [44, 36]. Recent methods try to overcome those drawbacks by mapping the points near-losslessly [33] or mapping only local regions around target point in each time [27], which however are time inefficient. Thus, it is still an unsolved problem to apply an effective and efficient representation to a point cloud.

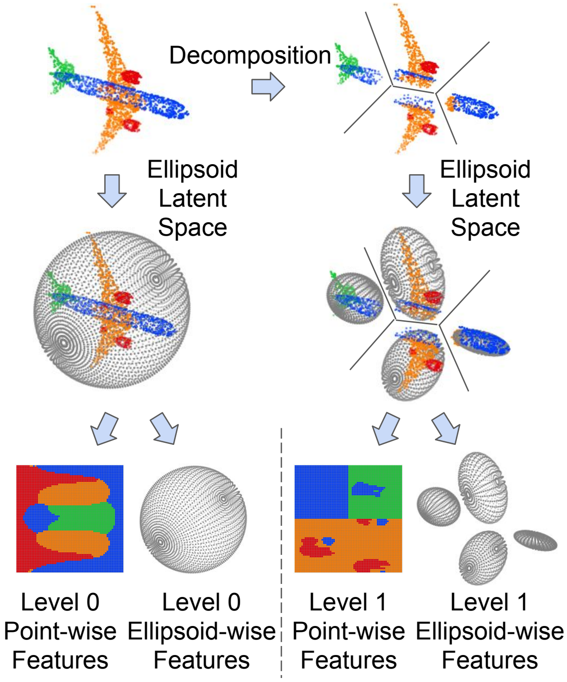

Decomposition
Ellipsoid Latent Space
Ellipsoid Latent Space
Level 0 Point-wise Features
Level 0 Ellipsoid-wise Features
Level 1 Point-wise Features
Level 1 Ellipsoid-wise Features

Figure 1: Overview of EllipsoidNet.

Motivation. Our work is motivated by the spherical representation, a popular representation method for LiDAR point cloud processing [32, 31, 37, 58, 51, 3]. In LiDAR point cloud frames, points are near-uniformly distributed on a spherical space so that a LiDAR point cloud is projected to a 2D feature map with minimal point overlaps and void pixels in the feature map. Lyu *et al.* [31] compares several representation method on KITTI road benchmark [7] and finds that spherical representation is a well balanced solution between point usage and latent space usage. In general point cloud processing, however, spherical representation may not work well since the points may not uniformly distributed in the spherical space.

In our work, we extend the use of spherical representation in general point cloud feature extraction. The proposed representation method effectively organizes the points in a 2D latent space with minor point overlaps and minor limitations to the network processing. Additionally, the proposed representation method is time efficient.

Approach. To address the spherical representation of general point cloud issue, we extend the spherical representation to ellipsoid representation by introducing the following schemes: (1) point cloud decomposition, (2) spatial transform, and (3) feature map completion.

Point cloud decomposition is added to split a big point cloud to several point partitions, in which local patterns can be better recognized and time efficiency can be achieved.Spatial transform is aimed to generate a ellipsoid latent space that better fit the point partition. Feature map completion is used to fill each pixel in the latent space with reasonable values to generate a dense feature map for further processing.

The ellipsoid representation of a point cloud outputs two items: a combination of 2D feature maps for point-wise features and a combination of feature vectors for ellipsoid-wise features. Targeting to the application of classification and semantic segmentation, we further propose EllipsoidNet, a CNN-based neural network to process the ellipsoid representation of a point cloud. The network takes both the point-wise features and ellipsoid-wise features for point cloud classification and segmentation.

Performance preview. To demonstrate the effectiveness of our ellipsoid representation, we evaluate the proposed method in two point cloud datasets: ModelNet40 for classification, and ShapNet for segmentation. In the experiment, the proposed method achieves near state-of-the-art performance in both datasets, with the help of the proposed neural networks.

Contributions. In summary, our key contributions of this paper are as follows:

- A novel representation method is proposed that represent point cloud in ellipsoid latent space. This method project a point cloud in dense 2D feature maps and works hierarchically for large-scale point cloud. Additionally, we explicitly present its impact to point cloud classification and segmentation application.
- Accordingly, a novel neural network named EllipsoidNet is proposed to take advantage of the ellipsoid representation for point cloud classification and segmentation applications.
- We evaluate the proposed representation method and networks in a point cloud classification dataset and a point cloud segmentation datasets, and present its effectiveness and efficiency.

2. Related work

Point cloud decomposition. Point cloud decomposite is widely used to separate the points to small partitions, lower the computational complexity of point cloud learning, and enhance local pattern recognition. Lyu *et al.* [33] utilizes a modified Kmeans clustering to create balanced point cloud partitions, which significantly decrease the computational complexity of point cloud representation. ACD [25] proposed an approximate convex decomposition method that divides an 3D object into near-convex parts, which however requires object surface meshes and unable to apply directly to point clouds. Cvxnet [5] introduces a network architecture to represent a low dimensional family of convexes. Yu *et al.* [55] performs recursive binary decomposition to a point cloud and concatenate the features from all tree levels. Ψ-CNN [21], Kd-net [15], and TreeRNN [34] performs similar decomposition for point cloud and graph learning tasks. In this paper, we take the Kmeans for point cloud decomposition since it is fast and generate a fixed number of point partitions.

Spatial transform. Spatial transform is used to regularize a point cloud and learn a robust feature extractor. Spatial transformer is usually achieved by applying a 3D transform to each point, which results in 3D rotation, translation, and scaling to the point cloud. PointNet [40] and PointNet++ [41] apply two spatial transforms to the point cloud: a non-learnable transform that scale and transit the point cloud within a uniball, and learnable transformer named T-net for better feature extraction. In the experiment they show that T-net has a significant improvement to point cloud classification. Wang *et al.* [48] proposes a novel approach to learn different non-rigid transformations of a input point cloud for different local neighborhoods at each layer. They claim that spatial transformers can learn the features more efficiently by altering local neighborhoods according to the semantic information of 3D shapes regardless of variations in a category.

Feature map completion. Feature map completion is used to generate a dense feature map with no void pixels, which helps CNNs better extract local features. This method is frequently used for depth completion in LiDAR point cloud processing. FuseNet [4] learns to extract joint 2D and 3D features for depth completion, which results in a dense depth map from camera and LiDAR inputs. LodoNet [58] completes the depth map using its nearest valid depth values. Different from the nearest neighbourhood filling, InterpCNN [35] and TangentConv [45] perform the completion using

Gaussian mixture. In other point cloud learning applications, FoldingNet [54] and AtlasNet [9] try to learn a latent vector space that cover the point cloud with 2D surfaces. In this paper, we propose to fill a dense feature map by projecting the nearest point in the point cloud to anchors in the ellipsoid latent space.

Point cloud 2D representation learning. Point cloud 2D representation learning tries to project a point set onto 2D latent space, from which 2D CNNs can be adopted to extract features. Besides the methods that only focus on LiDAR point cloud [32, 37, 51], several universal methods are proposed to represent a point cloud in 2D space. There are two major approaches: projection based and learning based. Projection based approaches try to define a 2D surface in the 3D space and then project the points to the surface. MVCNN [44] projects the points into multiple 2D plane space and concatenates the features from all 2D plane representations for point cloud classification. Similarly, in DeePr3SS [20], SnapNet [2] and TangentConv [45] the input point cloud is projected into multiple virtual camera views. To better represent the local geometry pattern in 2D space, FPConv [27] proposes a learnable method that projects the local region of each point onto a 2D image space in each layer. Lyu *et al.* explores the use of graph drawing to project the point cloud into a single 2D space for segmentation purposes.

Other point cloud learning methods. Besides the 2D representation learning, volumetric, graph, and structure are also widely used methods for point cloud representation learning. Volumetric-based methods [36, 59, 49, 8] discretize the 3D space to voxels, to which 3D convolution based methods can be applied. Graph-based methods [50, 43, 18] connect the points to graphs and apply graph convolutions and poolings on them. Structure-based methods [55, 16, 23] decomposite a point cloud layer by layer into a tree, and apply point-wise convolutions and recurrent neural networks to it.

3. Ellipsoid representation of point cloud

3.1. Method overview

Given a point cloud $\mathcal{P} \in \mathcal{R}^{N\times 3}$ and a 2D feature map $\mathcal{I} \in \mathcal{R}^{M\times M\times C}$ where N, M, C denote the point size, the feature map resolution and feature size, the ellipsoid representation of point cloud is to perform a projection $f : \mathcal{P} \to \mathcal{I}$ that maps the point cloud to the target 2D latent space. Figure 2 illustrates the steps of this representation. For a large-scale point cloud with complex shape, we further propose a hierarchical pipeline that perform the representation in multiple scales.

In this section, we firstly introduce the method in detail; secondly, we present the hierarchical scheme of this method for a large-scale point cloud with a complex shape; at last, we analysis the impact of this representation to point cloud classification and segmentation applications and illustrate its effectiveness and efficiency.

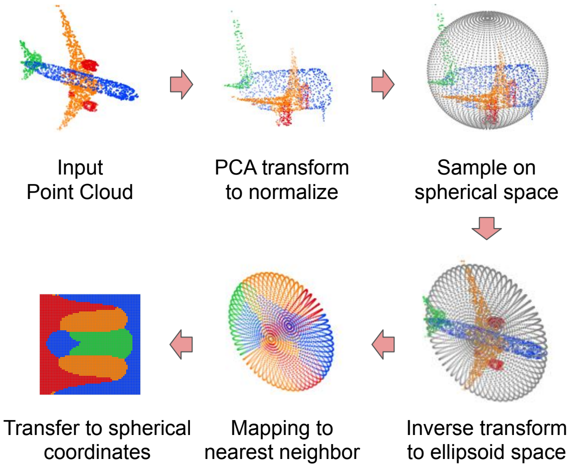


Figure 2: Ellipsoid representation of a point cloud.

3.2. Approach

As illustrated in Figure 2, given a point cloud $\mathcal{P}$, the spherical representation is equivalent to constructing its circumsphere, mapping each point to it's nearest point on the sphere, and generating a feature map I by sampling on the circumsphere. However, this method has two disadvantages: (1) unbalanced projection, and (2) sparse feature map. Some point clouds are much longer radii along their major axis than that along the minor axis, which results in heavily overlapped projection onto the poles of the circumsphere and many void pixels in the other parts; the pixels in feature map are not always filled with 3D points, which results in void pixels that confuse the CNNs.

Tackling this two issues, we proposed a novel representation method that generates a dense feature map and reduce the overlaps. The method has two upgrades: (1) use ellipsoid surface to replace the circumsphere, and (2) feature-map-orientated projection. The first upgrade is achieved by measuring the ellipsoid radius of the point cloud using the principal component analysis (PCA), and then scaling the circumsphere by the inverse of the radius. In this way, an ellipsoid surface of the point cloud is constructed that better fits the 3D shape. The second upgrade is achieved by tracing the nearest point of each pixel in the latent feature map and fill them into the pixel. By switching from point-cloud-orientated projection scheme to feature-map-orientated scheme, each point can be projected to multiple feature map pixels and all pixels can be filled with points. A detailed presentation algorithm is presented in Algorithm 1.

By applying the ellipsoid representation, we successfully build the correspondence between the points in 3D space and the pixels in the latent space.

Point-wise features and ellipsoid-wise features. The key

Algorithm 1 Ellipsoid representation

```
Input: Point Cloud $\mathcal{P}$, feature map resolution $M \times M$
Output: Point-wise feature map $F_p$, Ellipsoid-wise feature vector $F_e$

  $\mathbf{V} \leftarrow \mathbf{PCA}(\mathcal{P})$;
  $\mathcal{P}_1 \leftarrow \mathcal{P}\mathbf{V}^{\mathbf{T}}$;
  $radii_{xyz} = (\mathcal{P}_{1,xyz}^{max} - \mathcal{P}_{1,xyz}^{min})/2$;
  $t \leftarrow (\mathcal{P}_{1,xyz}^{max}/2) + \mathcal{P}_{1,xyz}^{min}/2))\mathbf{V}$;
  $rot \leftarrow RotVec(\mathbf{V})$;
  $F_e \leftarrow [rot, radii, t]$;
  FOR $u, v \in 0, 1, 2, ..., M$:
  $p_{Sphere} \leftarrow Spherical\_to\_Cartesian(u/M, v/M)$;
  $Anchor \leftarrow (p_{Sphere} * radii)\mathbf{V} - \mathbf{t}$
  $F_e(u, v) \leftarrow NearestNeighbour(Anchor) \in \mathcal{P}$
  ENDFOR
Return $T$
```

novelty of ellipsoid representation is that it not only outputs point-wise features but also outputs ellipsoid-wise features. The point-wise features are organized as a feature map where extensive local features are included. The ellipsoid-wise features are organized as a feature vector containing the global features of the target point cloud.

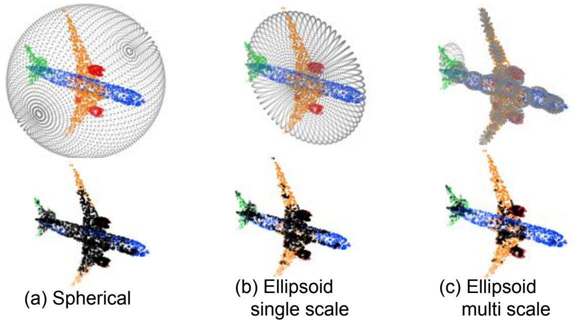
Point-wise feature includes the point's x, y, z position in the 3D space, its relative position in the ellipsoid coordinate frame, and the pixel's position in the 3D space and its relative position in the ellipsoid coordinate frame. Since we have the correspondence between the points and the pixels, we manage to organize a 2D feature map $\mathcal{I} \in \mathcal{R}^{M \times M \times C}$, where each pixel is a feature vector of point-wise features. Ellipsoid-wise feature contains the position, orientation and radius of the ellipsoid that covers the point cloud.

3.3. Hierarchical representation

Ellipsoid representation is an effective way to extract local and global features from a point cloud. However, it still has difficulty to represent a point cloud with a complex shape. To this end, we further proposed a hierarchical pipeline to generate the ellipsoid representation in multiple scales.

Figure 1 illustrates the steps of a 2-level hierarchical representation. Given a point cloud $\mathcal{P}$, we decomposite it into N_c partitions denoted as $P_1, P_2, ..., P_{N_c}$. After decomposition, we generate the ellipsoid representation of each partition. In multi-scale representation, we further decomposite each partition to even smaller partitions and generate ellipsoid representation (*i.e.,* decomposite P_1 to $P_{11}, P_{12}, ..., P_{1N_c}$). In that way, we can extract point-wise features and ellipsoid-wise features in multi-scales and feed them all into the neural network.

In this paper, we utilize the Kmeans solver for point cloud decomposition because it generate compact partitions and is time efficient.



(a) Spherical

(b) Ellipsoid single scale

(c) Ellipsoid multi scale

Figure 3: Qualitative evaluation of ellipsoid representation. The upper row shows the constructed latent space, and the lower row shows the projected points (colored by semantic segments) and missed points (black).

3.4. Analysis

To analysis the performance of our ellipsoid representation, we manage to evaluate its impact to classification and segmentation on ShapeNet dataset. For qualitative evaluation, we test on an airplane point cloud and visualize results of spherical, single-scale ellipsoid and multi-scale ellipsoid representation. In Figure 3 our multi-scale ellipsoid representation better fits the point cloud, and has much more points mapped to the latent space.

Impact to classification and segmentation. To quantitatively evaluate the impact of ellipsoid representation to point cloud classification and segmentation, we generate the representations for all the 14007 point cloud objects in the ShapeNet training set and compare the point usage rate and maximum segmentation intersect over union (IoU). The point usage rate measures the percentage of points in the point cloud that are mapped to the feature maps. A representation with high point usage ratio is expected to keep more local pattern, which may further lead to better performance in classification and segmentation. The maximum segmentation IoU measures the instance mean IoU of the dataset if we have correct segment on feature maps. This metric values the upper bound of the segmentation performance.

In Table 1, we compare the between different representation with the same feature map size. We observe that our multi-scale ellipsoid representation has a significant improvement towards single-scale ellipsoid representation and spherical representation, which indicates that our proposed method potentially performs better in the point cloud applications. Our experiments in Section 5 further confirm this point.

In Table 2 we compare between multi-scale representation with different partition sizes, and in Table 3 we compare between representation with different feature map sizes. Clearly we see that more partitions and larger those metrics.

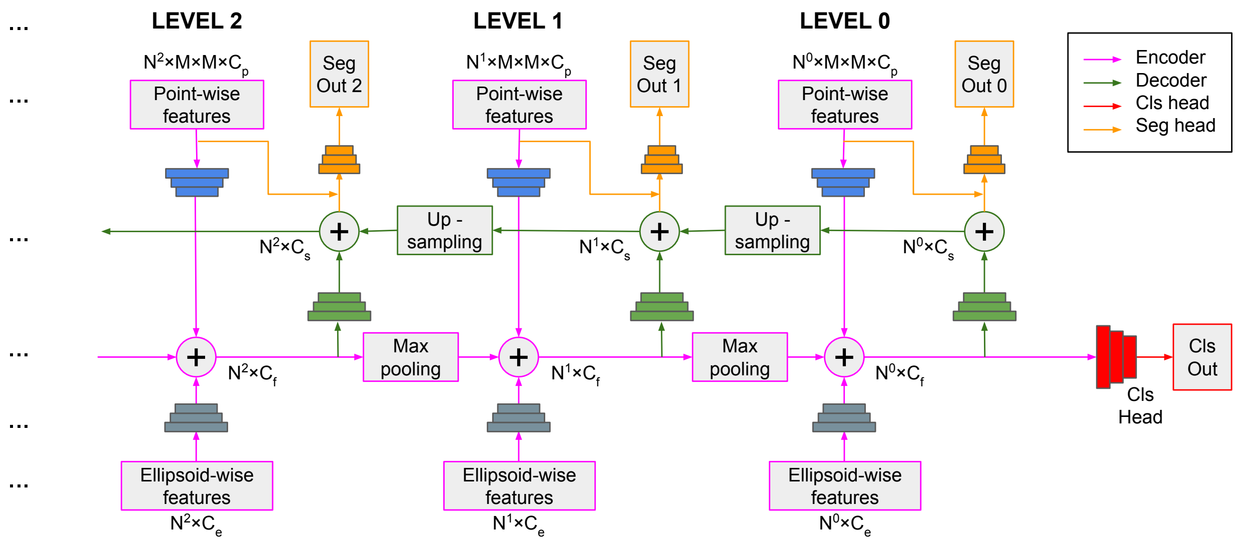


Figure 4: Overview of EllipsoidNet.

Representation	Spherical	Ellipsoid single-scale	Ellipsoid multi-scale
Feature map size	64^2	64^2	16×16^2
Point usage rate (%)	21.6	41.1	72.9
Maximum segmentation IoU (%)	82.1	88.0	95.3

Table 1: Comparison between different representation methods.

Representation	Ellipsoid multi-scale		
Feature map size	16×32^2	25×32^2	36×16^2
Point usage rate (%)	74.4	81.7	86.2
Maximum segmentation IoU (%)	95.2	96.4	97.4

Table 2: Comparison between ellipsoid representation methods with different decomposition sizes.

4. Deep learning on ellipsoid representations

In this section, we introduce EllipsoidNet, a novel neural network that takes advantage of the point cloud ellipsoid representation for classification and segmentation tasks. We first overview the network architecture and then look into the detail of the modules.

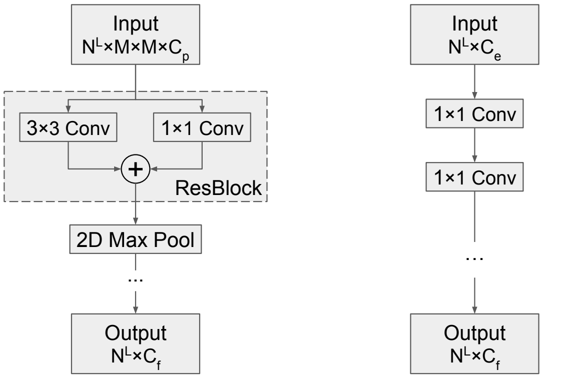
Representation	Ellipsoid multi-scale		
Feature map size	36×16^2	36×32^2	36×64^2
Point usage rate (%)	86.2	90.1	91.3
Maximum segmentation IoU (%)	97.4	97.9	98.1

Table 3: Comparison between ellipsoid representation methods with different feature map sizes.

4.1. Network overview

Figure 4 overviews of the EllipsoidNet for classification and segmentation applications. The network contains two major flows (encoder and decoder) and two major heads (classification head and segmentation head). The encoder extracts the features from the ellipsoid representations, the decoder distributes the extracted features back to the local regions, and the heads focus on specific tasks. For multi-scale ellipsoid representation there are several levels of point-wise feature maps and ellipsoid-wise feature vectors. To accept those feature inputs, in each level, there is a backbone to parse the point-wise features and another one for ellipsoid features. We further fuse the backbone outputs in each level by adding them together, and send them to the next level after a max-pooling. In classification tasks, we add a classification head at the end of the encoder, and in segmentation tasks, we add both classification head and segmentation head.

The input of the EllipsoidNet are point-wise features and ellipsoid features in all hierarchical levels. For classification tasks, the network outputs the object class estimation; for segmentation tasks, the network outputs object class estima-



(a) Point-Wise Encoder

(b) Ellipsoid-Wise Encoder

Figure 5: Modules of EllipsoidNet.

tion as auxiliary loss, and point-wise segmentation estimates in all levels.

4.2. Modules

Point-wise feature backbone. This backbone extracts the local features from point-wise feature maps. As presented in Figure 5 (a), we employ ResBlock inspired from ResNet [11] to extract local features, and 2D max-pool for down-sampling. By adding multiple ResBlocks and pooling layers, we successfully extract C_f features from each point-wise feature map with size $M \times M \times C_p$. Feature maps from each point partition is processed in parallel.

Ellipsoid-wise feature backbone. This backbone fuses the ellipsoid-wise features in each point partition. As shown in Figure 5 (b), we employ 1×1 convolution layers similar to PointNet [40]. The backbone outputs is directly added to the output of the point-wise feature backbone and the outputs of lower levels. Similar to the point-wise feature backbone, feature vectors from each point partition is processed in parallel.

Classification head. Classification head is aimed to estimate the category of an point cloud object based on the extracted features. In point classification tasks, its output is the final result; in point segmenting tasks, its output serves as an intermediate result that helps the network training. This head is constructed by several fully-connected layers.

Segmentation head. Segmentation head is aimed to estimate the category of each point in a point cloud. This head is not implemented for point cloud classification tasks. As seen in Figure 4, this head returns a segmentation map in each level, where we further estimate the point category. The segmentation head contains several multi-layer perceptrons (MLPs). Details of each MLP is shown in the supplemental material.

From network output to point cloud segmentation. The segmentation head outputs a segmentation map for each level. In order to get the point cloud segment from the map segment, we trace back the projected points from map segment according to the ellipsoid representation. For the points that is projected to multiple pixels in the segment map, we use the mean value of all the estimation. For minor points that are not projected to the segment map, we assign them according to their nearest neighbour in the point cloud with known segment labels. In this way the semantic labels of all points are estimated.

5. Experiment

5.1. Datasets

The proposed ellipsoid representation and EllipsoidNet are evaluated in two dataset: ModelNet40 for classification, and ShapeNet for segmentation.

ModelNet40 dataset contains a 9840-object training set and a 2468-object testing set. Each object belongs to one of 40 classes and is sampled to 2048 points. Similarly, ShapeNet dataset contains a 14007-object training set and a 2874-object testing set, where each object belongs to one of 16 object classes and is sampled to 2048 points. In addition, each point in the dataset belongs to one of the 50 semantic class.

The metrics for ModelNet40 classification benchmark are overall accuracy (OA) and mean class accuracy (mAcc), and the metric for ShapeNet segmentation benchmark are the mean intersection over union (mIoU) in class level (class mIoU) and object level (instance mIoU).

5.2. Ellipsoid representation implementation

In our experiment, we use a 2-level ellipsoid representation, which means we construct the ellipsoid for the entire point cloud (Level 0) and each of its partition (Level 1). By default, we set $N_c = 36$, namely we decomposite the point cloud into 36 partitions. For each ellipsoid latent space, we project the points into a 32×32 feature map ($M = 32$). For point-wise features, we use the points 3D position in the ellipsoid coordinate frame; for ellipsoid-wise features, we use the orientation, the radii along each ellipsoid axle, and the position of ellipsoid center in the patent coordinate frame.

Taking advantage of the point cloud augmentation, we employ an end-to-end training scheme that rotates a point cloud, jitters, and generates an ellipsoid representation in each training step. For testing, we generate 1 representation for each object in all test sets.

5.3. Network implementation

In the experiment, the default implementation of the EllipsoidNet modules is as follows: (1) the point-wise backbone is implemented as "$ResBlock(64) \rightarrow MaxPool(4 \times 4) \rightarrow ResBlock(128) \rightarrow MaxPool(4 \times$

4) $\rightarrow ResBlock(1024) \rightarrow MaxPool(2 \times 2)$"; (2) the ellipsoid-wise backbone is implemented as "$Conv(64) \rightarrow Conv(128) \rightarrow Conv(1024)$"; (3) the decoder module is implemented as "$Conv(512) \rightarrow Conv(256)$"; (4) the classification head is implemented as "$Fc(512) \rightarrow Fc(256) \rightarrow Fc(N_{CLS})$"; and (5) the segmentation head is implemented as "$Conv(128) \rightarrow Conv(N_{Seg})$". The $Conv$ denotes the 2D convolution layer, Fc denotes the fully connected layer, N_{CLS} denotes the number of object classes, and N_{Seg} denotes the number of point semantic classes. A ReLU activation is added right after each ResBlock, convolution layer and fully-connected layers.

The proposed representation method and neural network is implemented on a desktop equipped with an Intel Core i9-9820X CPU (3.3 Hz) and an NVidia GTX 1070 GPU. Key software packages include Python 3.7, Tensorflow 2.3 [1], Scipy 1.5.3 [47] and Scikit-learn 0.23 [39].

The statistics of the implementation for ModelNet40 and ShapeNet are listed in Table 4.

Target Dataset	ModelNet40	ShapeNet
Parameters (million)	3.75	4.78
Representation time (ms)	88.0	105.5
Network time (ms)	8.4	13.0
Total time (ms)	96.4	118.5

Table 4: Statistics of method implementation.

5.4. Training scheme

To train the network, we use the categorical cross entropy loss for both classification and segmentation tasks, and all auxiliary losses are equally weighted. In addition, we employ the Adam [13] optimizer with a 1×10^{-4} learning rate. We trained the network for 2000 epochs with a 32-sample batch for classification task and a 4-sample batch for segmentation task due to GPU memory limitation.

5.5. Ablation study

Impact of features. To study the impact of point-wise feature and ellipsoid-wise feature to the point cloud learning, we manage to train and evaluate the network using point-wise feature only, ellipsoid-wise feature only and a combination of both features. All three input choices are evaluated in ModelNet40 and ShapeNet datasets. From Table 5 we observe that EllipsoidNet with either point-wise features or ellipsoid-wise features achieves better results than PointNet, a baseline method that parses the point cloud without any representation. In addition, EllipsoidNet with both feature inputs results in near state-of-the-art performance in both classification and segmentation tasks, which indicates that the method provides a robust point cloud representation for general point cloud learning.

Input Features	ModelNet40 accuracy (%)	ShapeNet instance mIoU (%)
Point-wise	91.3	83.4
Ellipsoid-wise	89.2	-
Both	92.5	86.5

Table 5: Comparison of classification and segmentation result between Ellipsoid in different input features.

Representation	Ellipsoid multi-scale			
Feature map size	1×32^2	16×32^2	25×32^2	36×32^2
ModelNet40 accuracy (%)	73.9	87.3	89.2	92.5
ShapeNet instance mIoU (%)	56.4	77.1	80.6	86.5

Table 6: Comparison between ellipsoid representation methods with different decomposition sizes.

Impact of decomposition. To study the impact of decomposition to the point cloud learning, we apply the ellipsoid representation with different number of partitions. In Table 2 we compare some metrics in the representation level, here we implement the neural network and explicitly compare the performance in classification and segmentation tasks. Table 6 shows that more point partitions result in better performance in both applications. The impact of decomposition to the point cloud applications shares the same trend as the impact to point cloud representation.

Impact of latent space resolution. To study the impact of decomposition to the point cloud learning, we apply the ellipsoid representation with different feature map sizes. In Table 3 we compare some metrics in the representation level, here we implement the neural network and explicitly compare the performance in classification and segmentation tasks. As shown in the table, 32 × 32 feature map leads to the best result in both applications; a smaller feature map result in a significant drop in performance, perhaps because it lost too much points during point cloud representation; a bigger feature map does not gain a significant improvement in point cloud learning, which shares the same trend of performance in point cloud representation in Table 7.

5.6. State-of-the-art performance comparison

Quantitative comparison in ModelNet40. In Table 8 we compare the classification accuracy between ours and several existing works. Our work achieves near state-of-the-art accuracy in overall accuracy and the best in the mean accuracy for all shape classes.

Representation	Ellipsoid multi-scale		
Feature map size	36×16^2	36×32^2	36×64^2
ModelNet40 accuracy (%)	91.8	92.5	92.7
ShapeNet instance mIoU (%)	81.4	86.5	86.2

Table 7: Comparison between ellipsoid representation methods with different feature map sizes.

Method	OA (%)	mAcc (%)
PointNet [40]	89.2	86.2
PointNet++ [41]	90.7	-
MVCNN [44]	90.1	-
Pointwise-CNN [12]	86.1	81.4
KPConv [46]	92.9	-
InterpCNN [35]	93.0	-
RS-CNN [29]	**93.6**	-
Spherical CNNs [6]	88.9	-
DensePoint [28]	93.2	-
Ψ-CNN [24]	92.0	88.7
A-CNN [17]	92.6	90.3
KD-Net [16]	91.8	88.5
SO-Net [23]	90.9	87.3
SCN [52]	90.0	87.6
3DContextNet [57]	90.2	-
O-CNN [49]	89.9	-
ECC [43]	87.4	83.2
Ours	92.7	**91.4**

Table 8: Result comparison (%) with recent works on ModelNet40. 'OA' represents the mean accuracy for all test instances and 'mAcc' represents the mean accuracy for all shape classes. The symbol '-' means the results are unavailable.

Quantitative comparison in ShapeNet. In Table 9 we compare the class mean intersection over union (class mIoU) and the instance mean intersection over union (instance mIoU) between ours and several existing 2D representation-based works. Our works achieves near state-of-the-art in both class mIoU and Instance mIoU.

Qualitative comparison. For qualitative comparison, we visualize several segmentation results in ShapeNet dataset together with the ground truth and results from PointNet, and ours in Figure 6. It is clearly seen that our method has better segmentation on small parts.

Method	class mIoU (%)	instance mIoU (%)
PointNet [40]	80.4	83.7
Pointnet++ [41]	81.9	85.1
DGCNN [50]	82.3	85.1
RS-CNN [30]	84.0	86.2
DensePoint [28]	84.2	86.4
VV-Net [36]	84.2	**87.4**
PartNet [56]	84.1	**87.4**
Ψ-CNN [22]	83.4	86.8
PAN [38]	82.6	85.7
PointCNN [24]	84.6	86.1
O-CNN [49]	**85.9**	-
Ours	85.0	86.9

Table 9: Result comparison (%) with recent works on ShapeNet. Numbers in red are the best in the column, and numbers in blue are the second best. class mIoU represents class mean intersection over union and instance mIoU represents instance mean intersection over union. The symbol '-' means the results are unavailable.

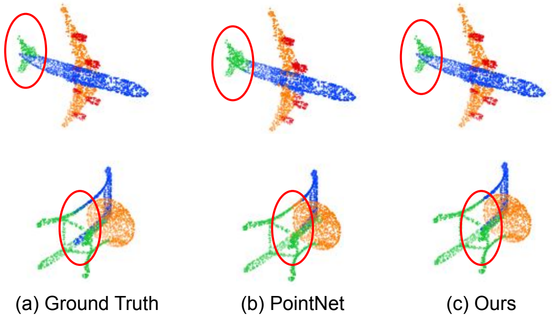


Figure 6: Qualitative comparison of point cloud segmentation between PointNet and ours. Points are colored by semantic segments. The different parts are emphasized using red circles.

6. Conclusion

In this paper we address the problem of point cloud classification and segmentation via 2D representation learning. To this end, a novel point cloud representation method is proposed that hierarchically projects the point cloud to a ellipsoid space. The method effectively and efficiently projects the point cloud into 2D latent space for local feature extraction. In addition, a novel neural network architecture named EllipsoidNet is proposed to perform point cloud classification and segmentation from the proposed representations. We evaluate our method on ModelNet40 and ShapeNet, achieving near state-of-the-art performance on both data sets. In future works ellipsoid representation for point clouds scanned from the real scenes will be explored.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 7

[2] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. *Computers & Graphics*, 71:189–198, 2018. 3

[3] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguère, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019. 1

[4] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10023–10032, 2019. 2

[5] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[6] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018. 8

[7] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013. 1

[8] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 3

[9] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 3

[10] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[12] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018. 8

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[14] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 1

[15] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 2

[16] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 3, 8

[17] Artem Komarichev, Zichun Zhong, and Jing Hua. A-cnn: Annularly convolutional neural networks on point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7421–7430, 2019. 8

[18] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 3

[19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1

[20] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017. 3

[21] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided cnn with spherical kernels for 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9631–9640, 2019. 1, 2

[22] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided cnn with spherical kernels for 3d point clouds. In *CVPR*, June 2019. 8

[23] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 3, 8

[24] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. 8

[25] Jyh-Ming Lien and Nancy M Amato. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 121–131, 2007. 2

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid

networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1

[27] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2020. 1, 3

[28] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *ICCV*, October 2019. 8

[29] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. 8

[30] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, June 2019. 8

[31] Yecheng Lyu, Lin Bai, and Xinming Huang. Chipnet: Real-time lidar processing for drivable region segmentation on an fpga. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(5):1769–1779, 2018. 1

[32] Yecheng Lyu, Lin Bai, and Xinming Huang. Real-time road segmentation using lidar data processing on an fpga. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018. 1, 3

[33] Yecheng Lyu, Xinming Huang, and Ziming Zhang. Learning to segment 3d point clouds in 2d image space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12255–12264, 2020. 1, 2

[34] Yecheng Lyu, Ming Li, Xinming Huang, Ulkuhan Gulerand Patrick Schaumont, and Ziming Zhang. Treernn: Topology-preserving deep graph embedding and learning. In *International Conference on Pattern Recognition (ICPR)*, 2020. 2

[35] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1578–1587, 2019. 2, 8

[36] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *ICCV*, October 2019. 1, 3, 8

[37] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019. 1, 3

[38] Liang Pan, Pengfei Wang, and Chee-Meng Chew. Pointatrousnet: Point atrous convolution for point cloud analysis. *IEEE Robotics and Automation Letters*, 4(4):4035–4041, 2019. 8

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 7

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2, 6, 8

[41] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 1, 2, 8

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[43] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. 3, 8

[44] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015. 1, 3, 8

[45] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2, 3

[46] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. 1, 8

[47] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 7

[48] Jiayun Wang, Rudrasis Chakraborty, and Stella X Yu. Spatial transformer for 3d points. *arXiv preprint arXiv:1906.10887*, 2019. 2

[49] 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 Transactions on Graphics (TOG)*, 36(4):1–11, 2017. 3, 8

[50] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019. 3, 8

[51] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 1, 3

[52] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. At-

tentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018. 8

[53] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 1

[54] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 3

[55] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9491–9500, 2019. 2, 3

[56] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *CVPR*, June 2019. 8

[57] Wei Zeng and Theo Gevers. 3dcontextnet: Kd tree guided hierarchical learning of point clouds using local and global contextual cues. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 8

[58] Ce Zheng, Yecheng Lyu, Ming Li, and Ziming Zhang. Lodonet: A deep neural network with 2d keypoint matching for 3d lidar odometry estimation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2391–2399, 2020. 1, 2

[59] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 3