# Triangle-Net: Towards Robustness in Point Cloud Learning

Chenxi Xiao
Purdue University
xiao237@purdue.edu

Juan Wachs
Purdue University
jpwachs@purdue.edu

## Abstract

*Three dimensional (3D) object recognition is becoming a key desired capability for many computer vision systems such as autonomous vehicles, service robots and surveillance drones to operate more effectively in unstructured environments. These real-time systems require effective classification methods that are robust to various sampling resolutions, noisy measurements, and unconstrained pose configurations. Previous research has shown that points' sparsity, rotation and positional inherent variance can lead to a significant drop in the performance of point cloud based classification techniques. However, neither of them is sufficiently robust to multifactorial variance and significant sparsity. In this regard, we propose a novel approach for 3D classification that can simultaneously achieve invariance towards rotation, positional shift, scaling, and is robust to point sparsity. To this end, we introduce a new feature that utilizes graph structure of point clouds, which can be learned end-to-end with our proposed neural network to acquire a robust latent representation of the 3D object. We show that such latent representations can significantly improve the performance of object classification and retrieval tasks when points are sparse. Further, we show that our approach outperforms PointNet and 3DmFV by 35.0% and 28.1% respectively in ModelNet 40 classification tasks using sparse point clouds of only 16 points under arbitrary SO(3) rotation.*

## 1. Introduction

As commodity cameras and laser based sensors become more affordable, point cloud based object classification is becoming the default approach for 3D sensing. For example, autonomous vehicles rely on point cloud maps sampled by Lidar sensors or depth cameras for effective navigation. One challenge often faced in such applications is that the density of sampling points decreases significantly as the distance from the vehicle embedded sensor to the object increases. This makes it hard to recognize objects that are far from such sensors due to their sparse inherent point structure [2]. As reported in the literature [18, 19, 1], the classification accuracy of these algorithms drop radically as the density of the point cloud decreases, and is further affected when the pose configuration of the object is not known in advance. Similarly, consider the scenario of tactile based object recognition using a robotic hand. The time complexity of sampling is proportional to the number of points sampled along the manipulator's trajectory [38, 20, 7]. This implies that in addition to performance degradation, there is an additional cost related to the amount of sampling required to make an acceptable prediction. Thus, it is necessary to come up with new 3D machine learning techniques that can classify objects based on "limited" sparse point cloud data and that can operate in real-time, whether for effective navigation (e.g. autonomous driving case) or for user's meaningful perception (e.g. tactile sampling).

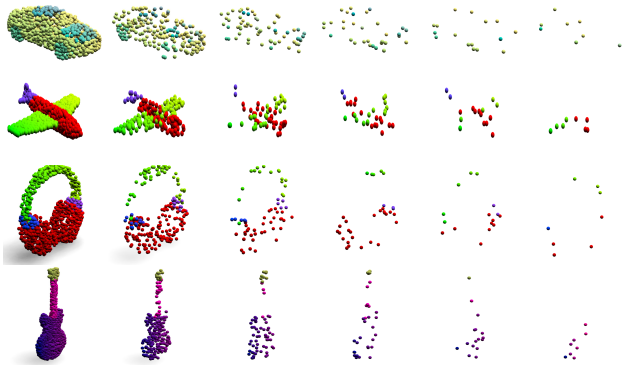


Figure 1. Point cloud of car, airplane, earphone and guitar model with size 2048, 256, 64, 32, 16 and 8 (from left to right). The object recognition is challenging even for human when the point cloud size is smaller than 16. Our developed algorithm can recognize objects from 40 categories at 70.35% using 16 points and 48.19% using 8 points even under arbitrary SO(3) rotation.

Unfortunately, object recognition on sparse point clouds with pose uncertainty has not been well addressed. To convey this issue visually, refer to Fig. 1. Most of the machine learning techniques are designed for objects with at least 1024 points, in which features are quite distinguishable [40, 18, 19, 29, 44, 13]. Conversely, finding salient features from sparse points is extremely difficult because salient features such as corners, edges and wrinkles are hard

to discern with fewer than 128 points; The shape envelop becomes indistinguishable even for humans when the number of points decreases to 16 and below. Further, the ambiguity is aggravated when arbitrary SO(3) rotation is involved. [22] shows how two objects can be nonrigidly aligned after rotation, even they are from two different categories. This calls for a new solution for recognizing sparse points with high discriminability under pose uncertainty.

Although some pioneer point cloud recognition methods have been focusing on the robustness to sparsity [1, 33] and pose variance [42, 4, 12], almost all previous works regard the sparsity and pose variance as two independent tasks and fail to consider them as a whole. However, in real applications, different variations are generally combined. For example, applications such as tactile recognition [39] or low-resolution outdoor 3D scans (e.g. Sydney Urban Objects, involving point clouds with less than 50 points [6]) involves both sparsity and unknown pose variation, which are still intractable tasks for the existing approaches. The only known work that is capable of addressing both concerns (sparsity and rotation) is [38]. The latter work recognizes sparse points by a simple histogram feature created through bin-counting triangle parameters. However, this method is not scalable to large datasets due to the limited resolution of bins, and also cannot generate point-wise features for segmentation. Instead, we utilize the graph neural network to learn latent representation with high discriminability in an end-to-end fashion, enabling various machine learning applications to be built on top of sparse and rotated point clouds. We summarize our contributions as follows:

- Propose a point-wise feature that has invariance towards arbitrary positional and rotational transformations.
- Propose a graph-based encoder to learn the object level representation that can simultaneously be invariant to positional shift, rotation, and scaling. We show by experiments that the object representation can remain discriminative even for significant sparse points combined with arbitrary rotation and noise jittering.
- Propose Triangle-Net, an end-to-end deep learning network that utilizes our proposed feature. Our network allows for versatile 3D machine learning tasks to be conducted on point clouds with multifactoral disturbances that cannot be robustly learned by previous methods.

## 2. Related Work

### 2.1. 3D Object Recognition

Existing approaches for point cloud classification mainly include but not limited to: 1) Directly performing classification on point cloud data [18, 19, 44]. 2) Projecting the point cloud data into other formats that can extract features expressively, such as voxelized objects [17, 3], grid cells [10, 9], spherical shells [43], images taken from multiple view angles [24, 25, 26], or graph representation [40, 29, 37, 16]. 3) Learning from hand-crafted features created by point cloud data [1, 38, 42, 39]. 4) Building classifiers on top of the learned latent representations using self-supervision e.g. self-reconstruction [30, 23, 5]. Our approach falls into the category of graph based deep learning, because the graph is a particularly suitable technique to utilize the unique structural properties between points.

### 2.2. Position and Orientation Invariance

Real-world objects can be found in arbitrary shapes and poses and therefore it is necessary to learn the corresponding invariance. However, authors, such as [42, 4] showed that the variance in orientation may lead to significant performance drops in most mainstream techniques used for point cloud object classification and segmentation.

Various approaches have been developed to alleviate this hurdle. For example, robustness to positional and rotational changes can be either learned [18, 19, 1] or manually imported through hand-crafted features [42, 4, 12]. However, the learned robustness shows degraded performance when generalized to scenarios where the object rotation is not present in the training set (for example, instances with only rotation around z-axis during training, but in the testing set with arbitrary SO(3) rotations). To add more meaningful variations in training demands more complex network architectures (e.g. more layers, neurons and connections) which increase the training complexity, and thereby limited the ability to generalize to combined variations of many stages. For example, training on a combination of rotation and sparsity can drastically degrade the recognition accuracy both in training and testing, as shown in Sec. 4.3.

A different approach, as used in [42, 4, 12] exploits local features with rotational invariance, but all of them are statistically significant only in dense point clouds. However, when the dense point cloud is available, there is not really a need for learning rotation invariance because pre-processing techniques such as alignment by PCA [32] can effectively address this problem. In this paper, we focus on addressing the challenge of the point cloud's sparsity with an ambiguous shape envelop. In such cases [42, 4, 12, 32] are not applicable.

### 2.3. Robustness to Sparsity

The two categories of a sparse point cloud representation are: locally sparse and globally sparse. The prior case corresponds to dense point clouds associated with low density regions. This category has been well-studied by either utilizing shape completion [20, 34, 36] or exploiting local features in regions that are not sparse [28].

The second case of category is a more general case, in which local signatures cannot help with object classification.

Therefore, the results can only have been conditioned on the sparse points only. In this paradigm, [18, 19, 1, 16, 33] conducted experiments using a randomly downsampled ModelNet 40 dataset [31]. However, all evaluation results indicate a significant performance drop when the number of points falls below a threshold. To be specific, methods that exploit local features [19, 16, 37] require statistical significance of correlated points (for example, the points in the $k$-nearest neighbourhood). However, the significance level is reduced as the number of points in a local region decreases. Recently, a new family of approaches based on 2D convolutions on rendered images [24, 25] or 3D convolutions [17] have been suggested. However, they have been found not suitable when the points are too sparse due to the low correlations between neighboring regions (most regions are void). In addition, none of the above approaches are rotational invariant and therefore the performance would be potentially impacted when the object pose is unknown.

## 3. Methodology

In this section we explain our method on object recognition, which is robust to point sparsity, positional shifts, scaling and arbitrary rotations. To effectively extract the spatial relationship between points, we utilize a hypergraph based feature proposed in Sec. 3.1-3.3. We then integrate the feature into a deep learning architecture in Sec. 3.4.

### 3.1. Graph Representation

For a point cloud denoted by $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and its corresponding surface normals $\mathrm{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$, an undirected hypergraph $G = (\mathcal{V}, \mathcal{E})$ can be built to represent geometric features; where $\mathcal{V}$ are the vertices that corresponds to the points; and $\mathcal{E}$ are the edge set of the graph, which contains the spatial relationship between points.

We will first explain the process for building the edges representation. Inspired by [8], we use hypergraph for connecting more than 2 points at once, which gains two advantages: 1) The extracted features have higher dimensions, making each feature to be distinctive from others. 2) A larger number of edges can be constructed when compared to those obtained by only connecting two nodes. This contributes to the statistical significance, which is crucial when the point cloud is sparse.

Fig. 2 (a) introduces, for the first time, a feature function that utilizes two points $\mathbf{x}_1$, $\mathbf{x}_2$ and its surface normal $\mathbf{s}_1$, $\mathbf{s}_2$. We denote this function as $f(\mathbf{x}_1, \mathbf{x}_2)$, which serves as a building block of our hypergraph representation. The function is given as:

$$\mathcal{D}_A = f(\mathbf{x}_1, \mathbf{x}_2) = [d_{\mathbf{x}_1,\mathbf{x}_2},\ \theta_{\mathbf{z}_{12},\mathbf{s}_1},\ \theta_{\mathbf{z}_{21},\mathbf{s}_2},\ \theta_{\mathbf{s}_1,\mathbf{s}_2}] \quad (1)$$

Where $[\cdot]$ is the concatenate function, $\mathbf{z}_{12} = \mathbf{x}_1 - \mathbf{x}_2$, $d_{12} = \|\mathbf{z}_{12}\|_2$, and $\theta_{\mathbf{u},\mathbf{v}} = arccos\frac{\langle \mathbf{u},\mathbf{v}\rangle}{\|\mathbf{u}\|_2 \cdot \|\mathbf{v}\|_2}$. At most $\mathrm{P}_n^2$ non-repetitive $\mathcal{D}_A$ features can be constructed for a point cloud of size n.

Based on the feature function, our proposed type B hyperedge is defined as $\mathcal{D}_B$:

$$\begin{aligned}\mathcal{D}_B(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) =& [f(\mathbf{x}_1, \mathbf{x}_2), f(\mathbf{x}_1, \mathbf{x}_3), f(\mathbf{x}_2, \mathbf{x}_3), \\ & \theta_{\mathbf{z}_{12},\mathbf{z}_{13}},\ \theta_{\mathbf{z}_{21},\mathbf{z}_{23}},\ \theta_{\mathbf{z}_{31},\mathbf{z}_{32}}]\end{aligned} \quad (2)$$

which corresponds to the illustration of Fig. 2 (b). Note that $\mathcal{D}_B$ not only includes three feature functions but also emphasizes the superimposed spatial relationship between them, meaning the extra elements $\theta_{\mathbf{z}_{12},\mathbf{z}_{13}}, \theta_{\mathbf{z}_{21},\mathbf{z}_{23}}, \theta_{\mathbf{z}_{31},\mathbf{z}_{32}}$. This shows that hyperedges can be used for building highly discriminative feature sets.

We found that, through experimentation, by including extra hand-crafted features, the accuracy can be improved while reducing convergence time. Therefore, we also propose a feature function $\mathcal{D}_C$, as illustrated at Fig. 2 (c). A center point between three vertices $\mathbf{x}_m = \frac{1}{3}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3)$ is computed, and then additional angular and distance values can be extracted. The math description is given as:

$$\begin{aligned}\mathcal{D}_C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = [&\mathcal{D}_B(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3), \\ & d_{\mathbf{x}_1,\mathbf{x}_m}, \theta_{\mathbf{z}_{1m},\mathbf{z}_{12}}, \theta_{\mathbf{z}_{1m},\mathbf{z}_{13}}, \\ & d_{\mathbf{x}_2,\mathbf{x}_m}, \theta_{\mathbf{z}_{2m},\mathbf{z}_{21}}, \theta_{\mathbf{z}_{2m},\mathbf{z}_{23}}, \\ & d_{\mathbf{x}_3,\mathbf{x}_m}, \theta_{\mathbf{z}_{3m},\mathbf{z}_{31}}, \theta_{\mathbf{z}_{3m},\mathbf{z}_{32}}]\end{aligned} \quad (3)$$

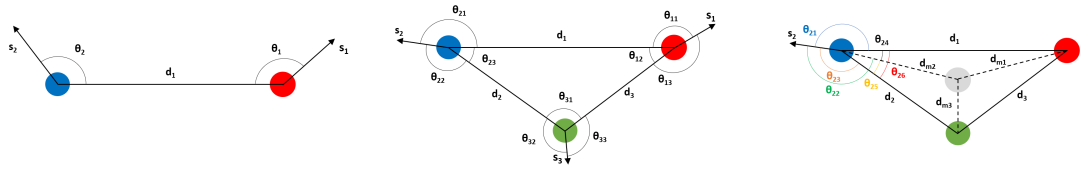
Robustness to point sparsity can be achieved when the triangle's vertices $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are chosen independently to the point cloud's local density. For example, when $\mathbf{x}_1$ is given, $\mathbf{x}_2$ and $\mathbf{x}_3$ can be chosen by either Farthest Point Sampling or uniform sampling so that our method can exploit the global level geometric relationship.

An extra robustness, scale invariance can be achieved after introducing the scale normalization. To be specific, for each distance entry in $\mathcal{D}_C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, we divide it by $max(d_{\mathbf{x}_1,\mathbf{x}_2}, d_{\mathbf{x}_1,\mathbf{x}_3}, d_{\mathbf{x}_2,\mathbf{x}_3}, d_{\mathbf{x}_1,\mathbf{x}_m}, d_{\mathbf{x}_2,\mathbf{x}_m}, d_{\mathbf{x}_3,\mathbf{x}_m})$, resulting the largest distance to be 1. For an object scaled by factor $\alpha$, $\mathcal{D}_C(\alpha\mathbf{x}_1, \alpha\mathbf{x}_2, \alpha\mathbf{x}_3)$ is strictly equal to $\mathcal{D}_C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ after scale normalization. However, this normalization is optional because object size is also a prior for recognition, while losing scale information is detrimental to the performance.

### 3.2. Proof of Invariance

It can be proved that all $\mathcal{D}_A, \mathcal{D}_B, \mathcal{D}_C$ proposed above are invariant to arbitrary SO(3) transformations without loss of generality. We denote the rotation transformation as $\mathbf{R}$ and the positional transition vector as $\mathbf{t}$. The SO(3) transformation does not change the distance between points.

$$\begin{aligned}d_{\mathbf{x}_1,\mathbf{x}_2} =& \|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \|(\mathbf{R}\mathbf{x}_1 + \mathbf{t}) - (\mathbf{R}\mathbf{x}_2 + \mathbf{t})\|_2 = \\ & \|\mathbf{R}(\mathbf{x}_1 - \mathbf{x}_2)\|_2\end{aligned} \quad (4)$$



(a) (b) (c)

Figure 2. We propose three functions of feature extraction: (a) $\mathcal{D}_A$ , which can be constructed using only 2 points with the attached surface normal vectors. (b) $\mathcal{D}_B$ that can be constructed using 3 points with surface normal vectors. (c) $\mathcal{D}_C$, which is built on the top of $\mathcal{D}_B$ but has more pre-computed information.

The angle $\theta_{\mathbf{u},\mathbf{v}}$ between vectors $\mathbf{u} = \mathbf{x_2} - \mathbf{x_1}$ and $\mathbf{v} = \mathbf{x_3} - \mathbf{x_1}$ are also invariant to rotation, because:

$$\cos\theta = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{|\mathbf{u}||\mathbf{v}|} = \frac{\mathbf{u}\mathbf{R}^\mathbf{T}\mathbf{R}\mathbf{v}}{|\mathbf{R}\mathbf{u}||\mathbf{R}\mathbf{v}|} = \frac{\langle \mathbf{u}^*, \mathbf{v}^* \rangle}{|\mathbf{u}^*||\mathbf{v}^*|} \tag{5}$$

where $\mathbf{u}^* = (\mathbf{R}\mathbf{x}_2 + \mathbf{t}) - (\mathbf{R}\mathbf{x}_1 + \mathbf{t})$ and $\mathbf{u}^* = (\mathbf{R}\mathbf{x}_3 + \mathbf{t}) - (\mathbf{R}\mathbf{x}_1 + \mathbf{t})$. The invariant property can be generalized to surface normal vectors since a surface normal vector $\mathbf{s}_1$ can be rewritten as $(\mathbf{x}_1 + \mathbf{s}_1) - \mathbf{x}_1$.

In brief, since all entries in $\mathcal{D}_A$, $\mathcal{D}_B$, or $\mathcal{D}_C$ are either distance between two points $\mathbf{x}_1$, $\mathbf{x}_2$, or angular value between two vectors, the extracted feature is invariant to SO(3). □

### 3.3. Hyperedge Convolution

Given the hyperedge features being extracted, we leverage on graph aggregation to get the latent feature representation. The aggregate function $\mathcal{A}(\mathbf{x}_i)$ of point $\mathbf{x}_i$ is given as:

$$\mathcal{A}(\mathbf{x}_i) = \max_{j,k \in \mathcal{E}} H_\Theta(\mathcal{D}_m(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)) \tag{6}$$

Where $H_\Theta$ is a mapping function parameterized by $\Theta$ implemented by a deep neural network. $\mathcal{D}_{m,m\in\{A,B,C\}}$ is the proposed feature function described earlier in Sec. 3.1. The dimension-wise max function is used to aggregate all the transformed features that are correlated with point $\mathbf{x}_i$. $\mathcal{A}(\mathbf{x}_i)$ is the point-level feature that can then be used for tasks such as point segmentation. Note that only a partial number of features can be extracted when the point cloud size is large due to the computational complexity. Therefore, we only use a feature subset of size $\mathcal{F}$ created by random sampling.

To get the global representation of the whole object, we aggregate all the latent features corresponding to all points in the sampled point cloud. The aggregation can be accomplished by another max function. The global feature from aggregation is then written as:

$$\mathcal{A}(\mathbf{x_1}, ..., \mathbf{x_n}) = \max_{i \in \mathcal{E}} \mathcal{A}(\mathbf{x}_i) \tag{7}$$

Note that the max aggregation function also relaxes the permutation restriction over the input points [18].

### 3.4. Network Architecture

We integrate the feature extraction (Sec. 3.1), point/global feature aggregation (Sec. 3.3), and mapping function $H_\Theta$ into an end-to-end architecture in Fig. 3. The mapping function $H_\Theta$ is implemented as a neural network, an efficient structure that extracts the feature progressively. We show the number of neurons in each layer in Fig. 3. Our goal is to utilize deep features that are known to be capable of reducing the inductive bias [11]. Neurons in each layer receive the concatenated feature from the previous layer as well as the graph feature in Sec. 3.1. This design not only benefits the performance by mitigating the gradient vanishing problem through additional shortcuts but also avoids loss of information with deeper structures. The final representation is a summed feature of the deepest feature and the concatenated feature from shallower layers.

The point level feature and global level feature allows for various machine learning tasks to be conducted using sparse point clouds. While these tasks can be learned separately, we leverage multi-task learning to improve the generalization. The multi-task learning rewards the network to learn the underlying data distribution of the input data and also works as a regularization technique to prevent overfitting [41]. We show the network design of each task as follows.

**Classification** The classification network takes in the object global feature $\mathcal{A}(\boldsymbol{x_1}, ..., \boldsymbol{x_n})$ and predicts the likelihood of target categories. An MLP network with 3 hidden layers is adopted, with 512 and 256 units per hidden layer. Each hidden layer is followed by batch normalization, dropout layer with $p = 0.3$ and uses ReLU activation function.

**Part Segmentation** We implemented the segmentation network as an MLP network that has 3 layers, with 256, 128, 50 output units, respectively.

**Voxel Reconstruction** The voxel reconstruction was conducted by performing upsampling based on $\mathcal{A}(\boldsymbol{x_1}, ..., \boldsymbol{x_n})$. The upsampling network has four 3D transposed convolutional layers with 1024, 256, 128, 64 kernels, respectively. The network produces voxels with sizes from $4 \times 4 \times 4$ to $32 \times 32 \times 32$ sequentially, with $2\times$ upsampling rate per stage.

## 4. Experiments and Results

### 4.1. Experimental Setting

We evaluate our approach under two scenarios: 1) global sparsity and 2) combined sparsity from both partial and global level. We evaluate two scenarios on ModelNet 40 [31]

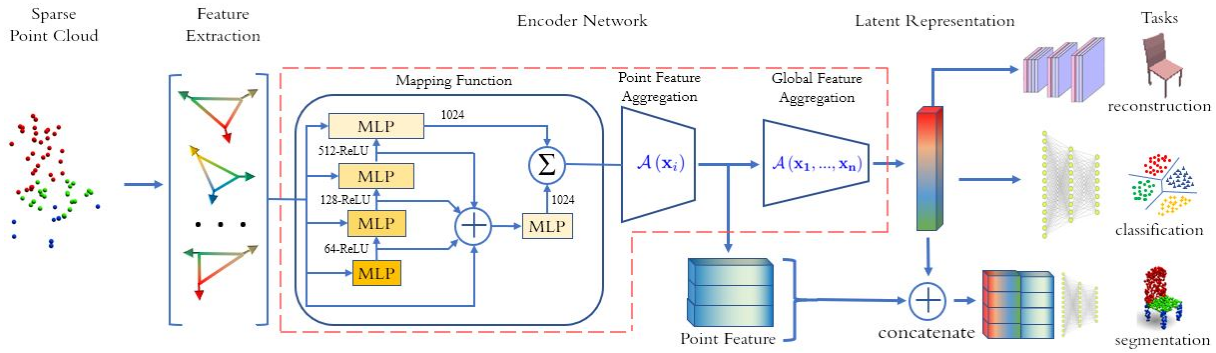



Figure 3. Triangle-Net, Our proposed deep learning architecture that extracts features from the parameters of the generated triangles. We leverage on a neural network encoder for extracting point-wise and global features, facilitating classification, segmentation and reconstruction.

and ScanObjectNN [28] datasets, respectively. ModelNet 40 has 12,311 CAD models in 40 categories while ScanObjectNN has 14,510 scanned point clouds in 15 categories. As oppose to ModelNet 40, objects in ScanObjectNN has a point distribution from real optical scanning that has partial sparsity and even missing regions. For ScanObjectNN, our experiments are conducted on the most difficult variant (PB-T50-RS) without background.

We acquire the data for evaluating global sparsity according to the evaluation protocol in literature [18, 19, 1], by which random downsampling is applied to reduce the number of points without changing the original point distribution pattern. The rotation invariance is evaluated under the same protocol as [42, 4], in which a random SO(3) rotation around the object center is applied.

### 4.2. Ablation Study 1: PointNet's Lack of Robustness Towards Rotation and Sparsity

Since PointNet is a basic building block for many 3D machine learning algorithms, it is necessary to use an experiment to show that the performance of PointNet will degrade when classifying arbitrarily rotated objects with sparse points. For this, we both trained and tested the PointNet on ModelNet 40 under 3 conditions in different point cloud sizes: a) Point clouds with no rotation b) Only rotate around z-axis or c) Arbitrary SO(3) rotations.

The results are shown in Table 1. PointNet performs well with dense points cloud under all 3 rotational conditions, as indicated in the first column. The result also shows that PointNet scales well to sparsity when no rotation is applied, as indicated in the first row. However, performance decays as rotation is applied to the data. It can be observed that rotation around the z-axis affects overall performance, and this is further aggravated when arbitrary SO(3) rotations are applied. The T-Net module fails to learn robustness towards SO(3) transformation when the point is extremely sparse.

Table 1. The experiment shows the performance degradation of PointNet (in %) when using a) No rotation b) Only rotated around z-axis or c) Arbitrary SO(3) rotation (as indicated in different rows) under different point cloud densities (as shown in columns).

| No. of points | 1024 | 256 | 64 | 16 |
|---|---|---|---|---|
| (1) No rotation applied | 88.51 | 86.89 | 82.49 | 76.40 |
| (2) Rotated around z-axis | 86.62 | 77.33 | 69.31 | 53.33 |
| (3) Arbitrary SO(3) rotation | 79.08 | 72.01 | 56.79 | 35.28 |
| Accuracy drop (1)-(3) | 9.43 | 14.88 | 25.80 | 41.12 |

### 4.3. Classification on Sparse and Rotated Points

In this experiment, we compare the classification accuracy of our method to other approaches under various point density configurations. The objects in the ModelNet 40 dataset undergo arbitrary SO(3) transformations both in training and testing sets. The results are shown in Table 2. The last row of the table shows our result with $\mathcal{F} = 4096$ features per object. The highest performance is highlighted in **bold** digits. Note that 3DmFV and PointNet are known models without a lower boundary for the number of points required. Conversely, PointNet++ and RI-Conv have a lower boundary for the number of points required, thus, is not applicable to cases with fewer than 64 points.

Last, PointNet was tested under two conditions. The first relies on vanilla PointNet[1] which was trained with 1024 points with random input dropout [19]. Second, PointNet[2] was trained and tested using the same number of points. The results indicate that both PointNet models fail to generalize sufficiently well to sparse point clouds. From Table 2, we can see that 3DmFV [1] can perform better than PointNet when the input points are sparse. However, the accuracy decays with the point cloud size. We compared our algorithm with RI-CONV [42] as well, an architecture that has rotational invariance and therefore is robust to SO(3) transformation. However, we found that it is not sufficiently robust to sparsity and displays a performance drop with an increase in the point size. Other recent methods such as KCNet[21], KPConv[27], require a higher lower boundary for the minimum number of points, and therefore we found not useful to compare with.

The above comparisons show that our approach can outperform others by a large margin when points are sparse. Conversely, the advantage is not significant when using dense points. We believe this is mainly due to 2 reasons.

1) Part of relative positional information between points is discarded, as each feature is constructed using 3 points rather than all points. 2) When the point cloud is dense, there is an immense number of triangles that can be constructed (e.g. $1 \times 10^9$ possible triangles when using 1024 points) and, in that case, the subset of triangles chosen by our method may be sub-optimal to represent the object of interest.

Table 2. Comparison of ModelNet 40 classification accuracy (in %) on both globally dense and sparse points under arbitrary SO(3) rotation. Our algorithm shows the advantage when points become sparse.

| | Dense | | | | Sparse | | | |
|---|---|---|---|---|---|---|---|---|
| Num of points | 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 |
| PointNet[1][18] | 73.09 | 72.67 | 64.48 | 39.93 | 21.08 | 9.79 | 2.65 | 2.07 |
| PointNet[2][18] | 79.08 | 75.14 | 72.01 | 72.64 | 56.79 | 48.34 | 35.28 | 23.91 |
| PointNet++[19] | 84.76 | 83.87 | 83.31 | 78.60 | N/A | N/A | N/A | N/A |
| 3DmFV[1] | 86.63 | 85.69 | 84.70 | 82.32 | 76.56 | 63.45 | 42.26 | 23.68 |
| RI-CONV[42] | 86.5 | 84.4 | 80.8 | 76.0 | N/A | N/A | N/A | N/A |
| Ours | **86.66** | **85.73** | **85.32** | **83.41** | **81.53** | **79.28** | **70.35** | **48.19** |

We observe a similar trend in the benchmark on ScanObjectNN [28] dataset that has combined sparsity from global and partial regions, as in Table 3. Our approach shows advantages in all point densities when objects are under arbitrary SO(3) rotations. But when SO(3) rotation and sparsity were not applied, DGCNN [29] outperformed both PointNet and our approach. We also notice the DGCNN has the most drastic accuracy drop among 3 approaches when the resolution decreases, while ours drops most gracefully among them. In this context, we believe that DGCNN is benefited from the "EdgeConv" operation that exploits the local information from the dense parts when the rest is less informative, but this operation may not be effective when points become globally sparse, as evidenced by [14, 15].

Table 3. Comparison of classification accuracy (in %) on ScanObjectNN dataset. Our algorithm shows the advantage when combined variations are applied. Otherwise DGCNN performs the best.

| | w/o SO(3) | | | SO(3) | | |
|---|---|---|---|---|---|---|
| Num of points | 2048 | 256 | 32 | 2048 | 256 | 32 |
| PointNet[18] | 74.4 | 73.73 | 69.91 | 67.38 | 64.92 | 54.85 |
| DGCNN[29] | **81.5** | **78.7** | **70.7** | 71.58 | 69.6 | 55.4 |
| Ours | 73.77 | 71.82 | 70.16 | **73.77** | **71.82** | **70.16** |

### 4.4. Segmentation on Sparse and Rotated Points

We conducted a part segmentation experiment based on the ShapeNet part dataset [35]. This dataset has 14,006 train samples and 2,874 test samples that belong to 16 object categories. Each point was annotated with a label, with 50 types of labeled parts in total. The segmentation task is to predict the label for each point conditioned on both point feature $\mathcal{A}(\mathbf{x}_i)$ and global feature $\mathcal{A}(\mathbf{x_1}, ..., \mathbf{x_n})$.

We built the segmentation network on top of our learned representation. For a fair comparison with PointNet, we used a network that has the same classifier head as [18]. While PointNet can use an arbitrary number of input points, DGCNN requires at least $k$ points for $k$-nearest neighbourhood search. For experiments with 1024, 64, 16 and 8 points, we set $k$ as 20, 16, 8, 4 respectively. The result is shown in Table 4. Note that our reported IoU value is the IoU averaged over all instances in the test set.

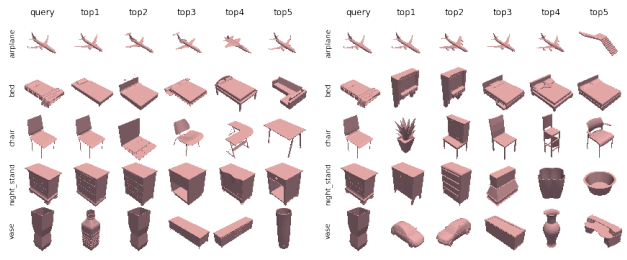
Our approach also shows an advantage in the segmentation task when the point cloud is sparse. We outperform PointNet and DGCNN with only 8 or 16 points. However, as expected, the IoU decreases as the point cloud becomes denser. Because the number of $\mathcal{D}_C$ features used was too small (only 4096) when compared to the point cloud size, the point-level feature did not represent the target category sufficiently well.

Table 4. The averaged instance IoU of part segmentation experiment under SO(3) rotation and given point cloud size. The result shows that our approach performs better when points are sparse.

| No. of points | 1024 | 64 | 16 | 8 |
|---|---|---|---|---|
| PointNet[18] | **80.52** | **80.47** | 75.59 | 70.30 |
| DGCNN[29] | 80.43 | 77.94 | 69.27 | 64.38 |
| Ours | 72.53 | 80.09 | **78.74** | **75.83** |

### 4.5. Object Retrieval by Shape Similarity

Our learned representation can be used as a metric for comparing shape similarity even when the point cloud is sparse and rotated. The experiment below shows the performance of the learned shape similarity metric using only 16 points. Both the query object and the candidate objects are from the test set of ModelNet 40 dataset (i.e. unseen objects). The top 5 similar objects are found within the test dataset using the $k$-nearest neighborhood with the $L^2$ distance metric. The retrieval results of our approach is shown in Fig. 4 (a). The comparison with PointNet is shown in Fig. 4 (b).




(a) (b)

Figure 4. Results of the retrieval operation of our approach (a) vs PointNet model retrieval results (b) using only 16 points under arbitrary SO(3) rotation.

We use retrieval mAP (mean averaged precision) as a metric for a quantitative comparison of our approach to PointNet. Our approach achieves 59.97% and 56.67% in top-5 and top-10 retrieval results respectively, outperforming

PointNet that achieved 34.80% and 35.04% correspondingly. We believe that the boost in performance is from a better discriminative ability of our feature and a better similarity metric learned by object reconstruction.

### 4.6. Embedding Analysis

In order to validate rotational and positional invariance, we observed the feature invariance in the embedding space. First, 3 models were trained with point clouds of size 16 using the feature functions $\mathcal{D}_A, \mathcal{D}_B, \mathcal{D}_C$ respectively. Then, an arbitrary object $\mathbf{x}_i$ was chosen, and all the non-repetitive features using $\mathcal{D}_A, \mathcal{D}_B, \mathcal{D}_C$ were created ($\mathrm{P}_{16}^2 = 240$ features for $\mathcal{D}_A$, $\mathrm{P}_{16}^3 = 3360$ features for $\mathcal{D}_B$ and $\mathcal{D}_C$) and its embedding vectors were extracted by the encoder. The embedding vector of $\mathbf{x}_i$ remained unchanged after having the input $\mathbf{x}_i$ being subject to random rotation and position transformations, showing that our proposed approach is both rotational and positional invariant.

It is observed that the embedding space between categories remained distinctive even when the point cloud was extremely sparse. Further, we compared the data distribution in the feature space under different point cloud size. We used PCA to reduce feature dimensionality from 1024 to 50 and then performed t-SNE in order to create a 2-dimensional visualization. For each plot, 10,000 samples from 10 categories were used. The results shown in Fig. 5 showcase that our point features are distinctive for all the learned representations. Yet, the margin between categories becomes ambiguous when the points get sparser (e.g. 8 points).

### 4.7. Voxel Reconstruction Using Sparse Points

We show object reconstruction results from the multi-task learning branch using only 16 input points, as Fig. 6. A voxel is placed when the output (binary Sigmoid function) is larger than 0.2 (instead of 0.5, as the normal Sigmoid case) because we found the network output becomes less "confident" as the input points become sparse. While the reconstruction result resembles the original object, some reconstruction artifacts can still be seen. These include cluttered voxels and inaccurate shape details. We believe this is mainly due to the limited discriminative ability of sparse points.

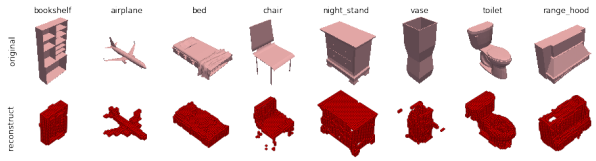



Figure 6. Voxel reconstruction using only 16 input points. Even though the input information is very scarce, reasonable reconstruction results can still be achieved.

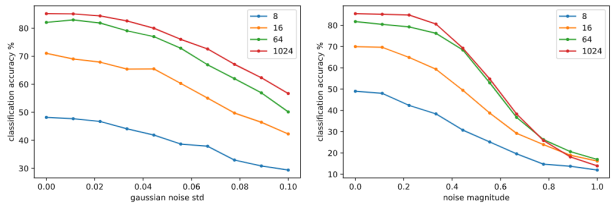
### 4.8. Robustness to Jittering

Sensor measurements include noisy data due to outliers, errors induced from the tactile sensor (e.g. inaccurate contact normals), or erroneous surface normals estimations (when not accessible from the sensor directly). To answer whether our approach can be robust to such noisy inputs, we conducted noise injection experiments.

The noisy surface normal vector $\hat{\boldsymbol{s}}$ is generated using the following procedure. First, a random 3D vector $\boldsymbol{w_n}$ with a given magnitude $m_s$ is computed: $\boldsymbol{v} = m_s \frac{\boldsymbol{w_n}}{\|\boldsymbol{w_n}\|}$ where $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. $\boldsymbol{v}$ is then being added to the original surface normal $\boldsymbol{s}$ and re-normalized to an unit vector: $\hat{\boldsymbol{s}} = \frac{\boldsymbol{s}+\boldsymbol{v}}{\|\boldsymbol{s}+\boldsymbol{v}\|}$.

A noisy point $\hat{\boldsymbol{p}}$ is generated by adding a Gaussian noise $\boldsymbol{w_p} \sim \mathcal{N}(\mathbf{0}, m_p \boldsymbol{I})$ to each separate position $\boldsymbol{p}$: $\hat{\boldsymbol{p}} = \boldsymbol{w_p} + \boldsymbol{p}$.

Our approach is validated using the ModelNet 40 classification task with both dense points (1024 points), and sparse points (64, 16, 8 points). For a fair comparison, the model is trained without noise at all. During testing, we inject noise to the surface normal's channel and the position's channel separately. The robustness is evaluated by obtaining the classification accuracy as a function of magnitude $m_s$ and $m_p$, as shown in Fig. 7. Overall, it can be seen that our network is more robust to noise in both point position's channel and surface normal's channel. For the noisy point position case, our approach only drops 28.5% when the noise standard deviation reached 10 centimeters in each dimension, while PointNet degrades around 58% with the same scenario (refer to literature [18]). For the noisy surface normal case, it shows that robustness varies with point cloud size. When using 8 points (the harshest test scenario), there is only 6.6% accuracy drop when the random noise component $\boldsymbol{v}$ reaches 20% magnitude of the original surface normal $\boldsymbol{u}$.

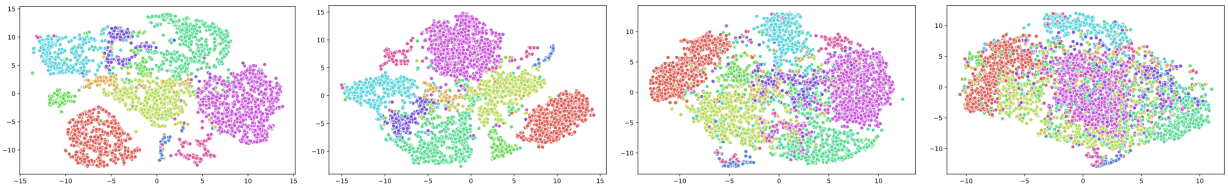



(a) position (b) surface normal

Figure 7. Robustness experiment under jittering in (a) point position (b) direction of surface normal vector, showing that our approach is robust to a wide range of input noise.

### 4.9. Scale Invariance

Scale invariance can be achieved with the proposed scale normalization trick. We train 2 models for this experiment. The first model is trained and tested with the scale normalization trick (refer to Sec. 3.1), while the second model is trained and tested without scale normalization. In neither case, data augmentation is used to enhance robustness. The evaluation is conducted under a combination of 3 variants: arbitrary SO(3) rotation, sparsity (16 points) and scaled by a given ratio from 0.5 to 1.5. The result is shown in Fig. 8. The blue curve corresponds to the model after the scale normalization trick, showing performance resiliency to changes



(a) 1024 points (b) 64 points (c) 16 points (d) 8 points

Figure 5. t-SNE visualization of learned embedding space using point cloud of size (a) 1024, (b) 64, (c) 16, (d) 8. The learned features remain to be distinctive even when the point cloud is extremely sparse e.g. 16 or 8 points.

in scale. Nevertheless, this comes at the price of overall inferior performance than the peak performance achieved by the model without the scale normalization trick (orange curve), as the scale information is lost.

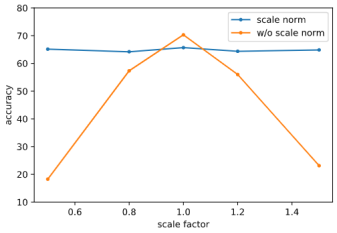



Figure 8. Robustness to scaling by the normalization trick (blue curve), otherwise it is sensitive to the object scale (orange curve).

### 4.10. Ablation Study 2: Network Components

**Inference and Training Time:** Our proposed feature can be computed quickly using parallel computing on a GPU, as shown in Table 5. For ModelNet 40, training with a number of features $\mathcal{F} = 4096$ can be completed within 8.5 hours on a single Nvidia Tesla P100 GPU.

Table 5. Model size and inference time under 1024 points.

| Method | # of parameters | Inference time |
|---|---|---|
| PointNet[18] | 3.5M | 2.1ms |
| DGCNN[29] | 1.8M | 22.7ms |
| Ours | 2.0M | 6.9ms |

**Number of Features:** The algorithm performance is correlated with the number of generated features $\mathcal{F}$. Table 6 shows the ModelNet 40 classification accuracy versus the number of features $\mathcal{F}$ under SO(3) rotation by 1024 points.

Table 6. The ModelNet 40 classification accuracy (in %) versus number of features $\mathcal{F}$ under 1024 point and SO(3) rotation.

| Num of $\mathcal{F}$ | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|
| Accuracy | 85.29 | 86.06 | 86.66 | **86.99** |

We compare several variations of our approach quantitatively using 16 points on ModelNet 40, as shown in Fig. 7. Classification results increase as more information is added to the feature. Accuracies of 60.08%, 69.04%, and 69.48% are achieved using $\mathcal{D}_A$, $\mathcal{D}_B$, $\mathcal{D}_C$ feature functions when only trained on the classification task, and the accuracy is further boosted to 70.35% when classification is trained together with object reconstruction. The latter scenario corresponds to the highest accuracy we achieved.

Table 7. ModelNet 40 classification accuracy (in %) of several variations of our proposed algorithm.

| Descriptor | Reconstruction | Accuracy |
|---|---|---|
| $\mathcal{D}_A$ feature function | No | 60.08 |
| $\mathcal{D}_B$ feature function | No | 69.04 |
| $\mathcal{D}_C$ feature function | No | 69.48 |
| $\mathcal{D}_C$ feature function | Yes | **70.35** |

## 5. Conclusions

While a rich variety of 3D object recognition methods have been proposed over recent years, very few of them can work on point clouds with a combination of disturbances such as low resolution, unaligned pose, and varied object scale. To address this problem, we evaluated state-of-the-art approaches under arbitrarily rotated sparse point clouds, and found most approaches only achieve limited performance or cannot work under this setting altogether.

In this paper, we propose a robust feature extraction method for point cloud that can generate invariant features towards positional, rotational and scaling disturbances. Such type of feature can remain discriminative when the point cloud is of significant sparsity and even being perturbed with noise. Furthermore, the feature extraction mechanism is integrated into Triangle-Net, a deep neural network that can learn in an end-to-end fashion. Experiments were conducted to show that our learned representation can remain robust to multifactorial variations, and is resilient to jittering, facilitating universal 3D machine learning tasks to be conducted on imperfect measurements and limited resources.

## 6. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant NSF NRI #1925194. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The code for this work is released at: github.com/MegaYEye/Triangle-Net.git

## References


[1] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018. 1, 2, 3, 5, 6

[2] I. Bogoslavskyi and C. Stachniss. Efficient online segmentation for sparse 3d laser scans. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85(1):41–52, 2017. 1

[3] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016. 2

[4] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4994–5002, 2019. 2, 5

[5] A. Dai, C. Ruizhongtai Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017. 2

[6] M. De Deuge, A. Quadros, C. Hung, and B. Douillard. Unsupervised feature learning for classification of outdoor 3d scans. In *Australasian Conference on Robitics and Automation*, volume 2, page 1, 2013. 2

[7] M. Elbanhawi and M. Simic. Sampling-based robot motion planning: A review. *Ieee access*, 2:56–77, 2014. 1

[8] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019. 3

[9] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018. 2

[10] T. Le and Y. Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 2

[11] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570, 2015. 4

[12] X. Li, R. Li, G. Chen, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. A rotation-invariant framework for deep point cloud analysis. *arXiv preprint arXiv:2003.07238*, 2020. 2

[13] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018. 1

[14] 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 I. Pratikakis, M. Spagnuolo, T. Theoharis, L. V. Gool, and R. Veltkamp, editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association, 2015. 6

[15] F. A. Limberger, R. C. Wilson, M. Aono, N. Audebert, A. Boulch, B. Bustos, A. Giachetti, A. Godil, B. Le Saux, B. Li, et al. Shrec'17 track: Point-cloud shape retrieval of non-rigid toys. 2017. 6

[16] Y. Liu, B. Fan, S. Xiang, and C. 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. 2, 3

[17] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 2, 3

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 1, 2, 3, 4, 5, 6, 7, 8

[19] C. R. Qi, L. Yi, H. Su, and L. 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, 3, 5, 6

[20] C. Rosales, F. Spinelli, M. Gabiccini, C. Zito, and J. L. Wyatt. Gpatlasrrt: a local tactile exploration planner for recovering the shape of novel objects. *International Journal of Humanoid Robotics*, 15(01):1850014, 2018. 1, 2

[21] Y. Shen, C. Feng, Y. Yang, and D. Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018. 5

[22] R. A. Srivatsan, P. Vagdargi, and H. Choset. Sparse point registration. In *Robotics Research*, pages 743–758. Springer, 2020. 2

[23] D. Stutz and A. Geiger. Learning 3d shape completion under weak supervision. *International Journal of Computer Vision*, pages 1–20, 2018. 2

[24] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 2, 3

[25] J.-C. Su, M. Gadelha, R. Wang, and S. Maji. A deeper look at 3d shape classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 2, 3

[26] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[27] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. 5

[28] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1588–1597, 2019. 2, 5, 6

[29] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point

clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019. 1, 2, 6, 8

[30] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016. 2

[31] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 3, 4

[32] Z. Xiao, H. Lin, R. Li, H. Chao, and S. Ding. Endowing deep 3d models with rotation invariance based on principal component analysis. *arXiv preprint arXiv:1910.08901*, 2019. 2

[33] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5598, 2020. 2, 3

[34] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Interpretable unsupervised learning on 3d point clouds. *arXiv preprint arXiv:1712.07262*, 2(3):5, 2017. 2

[35] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 6

[36] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018. 2

[37] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu. Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. *arXiv preprint arXiv:1904.10014*, 2019. 2, 3

[38] M. M. Zhang, N. Atanasov, and K. Daniilidis. Active end-effector pose selection for tactile object recognition through monte carlo tree search. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3258–3265. IEEE, 2017. 1, 2

[39] M. M. Zhang, M. D. Kennedy, M. A. Hsieh, and K. Daniilidis. A triangle histogram for object classification by tactile sensing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4931–4938. IEEE, 2016. 2

[40] Y. Zhang and M. Rabbat. A graph-cnn for 3d point cloud classification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6279–6283. IEEE, 2018. 1, 2

[41] Y. Zhang and Q. Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017. 4

[42] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pages 204–213. IEEE, 2019. 2, 5, 6

[43] Z. Zhang, B.-S. Hua, and S.-K. Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019. 2

[44] Y. Zhao, T. Birdal, H. Deng, and F. Tombari. 3d point capsule networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1009–1018, 2019. 1, 2