Surface Representation for Point Clouds

Haoxi Ran[1*] Jun Liu[2] Chengjie Wang[2]
[1]Northeastern University [2]Tencent Youtu Lab
ranhaoxi@gmail.com
{juliusliu, jasoncjwang}@tencent.com

Abstract

*Most prior work represents the shapes of point clouds by coordinates. However, it is insufficient to describe the local geometry directly. In this paper, we present **RepSurf** (representative surfaces), a novel representation of point clouds to **explicitly** depict the very local structure. We explore two variants of RepSurf, Triangular RepSurf and Umbrella RepSurf inspired by triangle meshes and umbrella curvature in computer graphics. We compute the representations of RepSurf by predefined geometric priors after surface reconstruction. RepSurf can be a plug-and-play module for most point cloud models thanks to its free collaboration with irregular points. Based on a simple baseline of PointNet++ (SSG version), Umbrella RepSurf surpasses the previous state-of-the-art by a large margin for classification, segmentation and detection on various benchmarks in terms of performance and efficiency. With an increase of around **0.008M** number of parameters, **0.04G** FLOPs, and **1.12ms** inference time, our method achieves **94.7%** (+0.5%) on ModelNet40, and **84.6%** (+1.8%) on ScanObjectNN for classification, while **74.3%** (+0.8%) mIoU on S3DIS 6-fold, and **70.0%** (+1.6%) mIoU on ScanNet for segmentation. For detection, previous state-of-the-art detector with our RepSurf obtains **71.2%** (+2.1%) mAP$_{25}$, **54.8%** (+2.0%) mAP$_{50}$ on ScanNetV2, and **64.9%** (+1.9%) mAP$_{25}$, **47.7%** (+2.5%) mAP$_{50}$ on SUN RGB-D. Our lightweight Triangular RepSurf performs its excellence on these benchmarks as well. The code is publicly available at https://github.com/hancyran/RepSurf.*

1. Introduction

Learning from raw point clouds has drawn considerable attention for its advantages in various applications, like autonomous driving, augmented reality, and robotics. However, it can be difficult for the irregularity of point clouds.

To handle irregular points, the pioneering work PointNet [41] adopts point-wise multi-layer perceptrons (MLP) to learn from points independently and utilizes a symmetric function to obtain the global information. PointNet++ [43] further introduces *set abstraction* (SA) to capture the local information of point clouds. However, both methods learn from standalone points and take no notice of local shape awareness [31].

*corresponding author

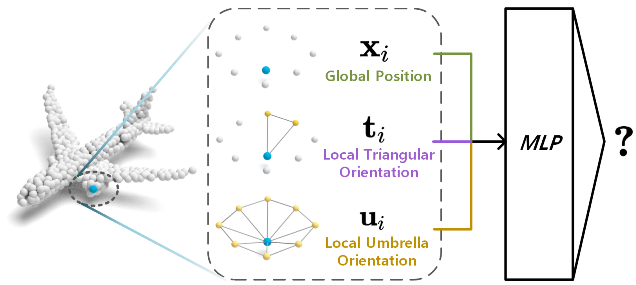

Figure 1. An overview of point cloud classification with *RepSurf*. Given one point (blue) in the airplane point cloud, we indicate its global position by the coordinate $\mathbf{x}_i$. Different from the prior works, we further explicitly describe its local geometry through Triangular RepSurf $\mathbf{t}_i$ extracted from the reconstructed triangle or Umbrella RepSurf $\mathbf{u}_i$ learned from the reconstructed umbrella surface. By combining positional and geometric information, point representation can be more expressive. After concatenating $\mathbf{x}_i$ and $\mathbf{t}_i/\mathbf{u}_i$ as input, we predict the category of the point cloud via MLPs followed by a pooling operation.

Local shapes are vital for the learning of point clouds. To learn from the local structural information, some prior works learn from grids [25,53], relations [31,44], or graphs [58,65]. However, these methods learn from shapes indirectly by attaching more ingredients (like Euclidean distances, attention mechanism) or applying various transformations (like graph construction, voxelization). These may lead to complex preprocessing and significant computations. These sophisticated hand-crafted components learn from implicit local shape representations in general. We argue that it may lead to an omission of information when pre-defining the ingredients, or a loss of geometry during transformation.

Taylor Series [52] expresses a local curve by derivatives. We simplify it by considering the second derivative only.

Thus, we can roughly represent the local curve, or what we call the "surface" in 3D point clouds, by its corresponding tangent.

To this end, inspired by Taylor Series, we propose RepSurf (*representative surfaces*) to explicitly represent the local shape of point clouds (shown in Fig. 1). To complement Cartesian coordinates in a point set with geometric information, we define RepSurf with three properties: discreteness, explicit locality, and curvature sensitivity. These properties allow RepSurf to express local geometry in free collaboration with irregular points. For a simple version of RepSurf, we propose Triangular RepSurf inspired by triangle meshes in computer graphics. We reconstruct a triangle for each point by querying its two neighbors and compute the triangle feature (i.e., normal vector, surface position, normalized coordinate) as RepSurf. To enlarge the perceptive field of RepSurf, we further propose Umbrella RepSurf inspired by umbrella curvature [10]. Umbrella RepSurf can be an extension of Triangular RepSurf since it is computed from the triangles of an umbrella surface. Different from Triangular RepSurf, we reconstruct an umbrella surface after searching K nearest neighbors and sorting the neighbors counterclockwise. For expressive representations, we feed the K triangular features of an umbrella surface into a learnable transformation function followed by aggregation. Moreover, we present several delicate designs (i.e., polar auxiliary, channel de-differentiation) to further improve RepSurf.

Our key contributions are manifold:

- A novel triangle-based representation, Triangular RepSurf for point clouds.
- A novel multi-surface representation, Umbrella RepSurf for point clouds.
- A high-efficiency plug-and-play module based on RepSurf for point cloud models.
- Our method achieves state-of-the-art on numerous point cloud benchmarks.

2. Related Work

2.1. Learning on Point Clouds

Multi-view methods [9, 14–16, 42, 62] or **voxel-based methods** [7,12,21,34,60] describe 3D objects with multiple views (i.e., converting 3D shape to 2D images [51] and lattice space [50]) or by voxelization (Oc-tree based networks O-CNN [56] and OctNet [45], efficient submanifold sparse convolution [13]). However, these transformation methods may lead to significant computations as well as a loss of shape information due to occlusion or lower resolution.

Point-based methods [11, 18, 19, 23, 24, 29, 32, 36] have recently attracted great attention to directly process point clouds. PointNet [41] learns from global information through multi-layer perceptrons and max-pooling operation. PointNet++ [43] introduces set abstraction to capture the features from the local point sets, and farthest point sampling (FPS) to uniformly downsample between two set abstractions. Recent works explore local aggregator via convolutions [17,27,28,30,37,53,57,59,59,64,66,71,73], relations [44,61,67,74], and graphs [58,65,75]. PointCNN [25] applies traditional convolution on point clouds after transforming neighboring points to the canonical order. RS-CNN [31] predefines geometric relations between points and their neighbors for local aggregation. DGCNN [58] computes the local graphs dynamically to extract geometric information. However, the methods are commonly based on some assumptions of implicit local geometry, which may result in missing geometric information in the input.

2.2. Detection on Point Clouds

Some early methods detect 3D objects by convolution after converting point clouds to 2D grids [5,22,26,63,68] or 3D voxels [49,76]. Recent works focus on 3D detection of raw point clouds [4,6,33,35,38,40,46,47,72]. VoteNet [39] adopts PointNet++ as the backbone for feature extraction and designs a component to group points corresponding to the voted centroids. [33] removes the hand-crafted operation of grouping by introducing Transformers [55].

2.3. Graphics-related Surface Representation

In computer graphics, triangle meshes are commonly adopted to represent 3D models. To obtain meshes from point clouds, previous works propose various methods for surface reconstruction. Ball-Pivoting Algorithm [3] forms a triangle if a specific-radius ball touches three points without containing other points. [20] defines the spatial Poisson formulation for surface reconstruction.

Curvature can further present the local geometry on 3D point clouds. [70] estimates the local curvature of the point cloud surface by Least Square Fitting. [10] constructs an umbrella surface based on the homogeneous neighbors and calculates the umbrella curvature through the neighbors' normal vectors and unit direction vectors.

3. Surface Representation

In this section, we first reveal the background for the design of our **Rep**resentative **Surf**aces (RepSurf) in Sec. 3.1. Secondly, we introduce several properties of RepSurf as inspiration in Sec. 3.2. Next, we propose two variants of RepSurf, Triangular and Umbrella RepSurf in Sec. 3.3 and Sec. 3.4, respectively. Finally, we implement RepSurf on PointNet++ (SSG version) and provide several exquisite designs to further improve the performance of RepSurf.

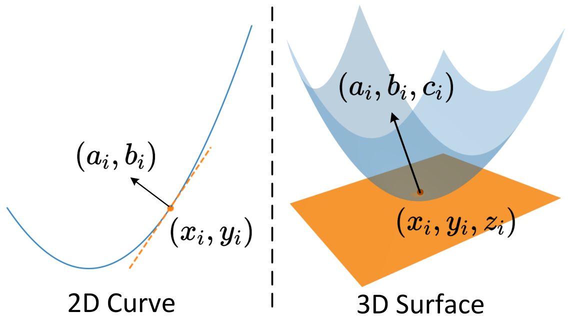

Figure 2. Local shape representation of a 2D curve (left) and a 3D surface (right) through the corresponding tangents.

3.1. Background

Local shapes are essential to represent point clouds. Prior works learn from shapes indirectly by utilizing extra ingredients or through different transformations. These operations may give some hints to express the local sets of point clouds, but cannot reflect the local shapes explicitly. We argue that the additional information leads to significant computations but contributes little to point cloud representations. Some may even cause the loss of geometric information. Therefore, we have to rethink on how to represent the local geometry.

We can describe a very local part centered on point $(t, f(t))$ of a 2D curve $f(\cdot)$ by Talyor series [52]:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(t)}{n!}(x-t)^n, \ |x-t| < \epsilon \tag{1}$$

To simplify the calculation, we approximate this equation by :

$$f(x) \simeq \underbrace{f(t)}_{\text{global position}} + \underbrace{f'(t)}_{\text{local orientation}} (x-t), \tag{2}$$

where $(t, f(t))$ is the global position on curve $f(\cdot)$, and the first derivative $f'(t)$ can intuitively indicate the local orientation near point $(t, f(t))$. To further express the local curve (Fig. 2 left), we represent the local orientation by its corresponding tangent:

$$\begin{aligned} a_i\,(x - x_i) + b_i\,(y - y_i) = 0 \Rightarrow \\ a_i x + b_i y - (a_i x_i + b_i y_i) = 0, \end{aligned} \tag{3}$$

where $x_i = t$, $y_i = f(t)$, and $\frac{a_i}{b_i} = -f'(a)$. (a_i, b_i) is the normal vector of the tangent, where $a_i^2 + b_i^2 = 1$. To conclude, a rough description of the local curve can be defined as:

$$\mathbf{c}_i = (x_i, y_i, a_i, b_i, a_i x_i + b_i y_i)\,. \tag{4}$$

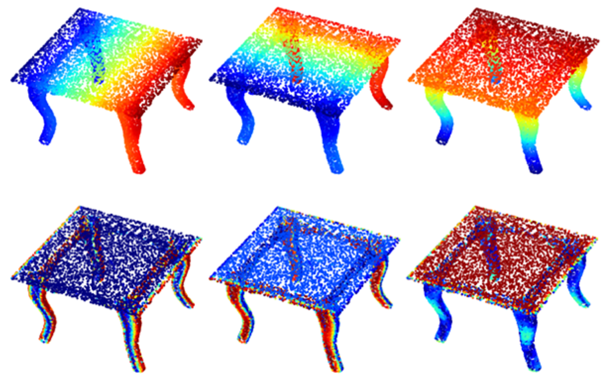

Figure 3. Visualization of a table on curvature sensitivity. We visualize a point cloud by the values of coordinates (above) and normals (below) in each of three dimensions. Intuitively, normal vectors can reflect the local shapes numerically to some extent.

3.2. Properties of RepSurf

PointNet [41] is inspired by three main properties of point sets in $\mathbb{R}^{N\times 3}$ from an Euclidean space: 1) unordered, 2) interaction among points and 3) invariance under transformations. It can handle the unordered point sets and alleviate the problem from rigid transformation. However, the ability to interact among points is still underexplored.

In 3D computer graphics, triangle meshes are a common representation of 3D models. Regularly, a triangle mesh consists of a set of triangles connected by their common edges or corners. Thus, triangles can flexibly present continuous and sophisticated 3D shapes for this characteristic. However, triangle meshes may not match the data structure of point clouds due to irregularity. A direct conversion from point cloud to triangle mesh may lead to significant computation as well as loss of point cloud characteristics (like flexibility from unorderness, scalability from the nature of sets). Therefore, we design our RepSurf inspired by the following properties:

- **Discreteness**. Ideally, RepSurf should be a set to collaborate with the related point set. It means that each of N points has a corresponding RepSurf feature.

- **Explicit Locality**. Unlike prior works describing local structure by learning (implicit locality), RepSurf shows the explicit locality of a part of point clouds numerically.

- **Curvature Sensitivity**. Coordinates can hardly depict the local shapes of 3D point clouds. RepSurf should be eligible to intuitively highlight edges and local shapes. An illustration is shown in Fig. 3.

Algorithm 1 Pytorch-Style Pseudocode of **Triangular RepSurf**

```
# B: batch size, N: number of points
# points: coordinates of a point set
pairs = kNN(points, k=2)-points # [B,N,2,3]
centroids = mean(pairs, dim=2) # [B,N,3]
normals = cross_product(pairs) # [B,N,3]
normals = normals/norm(normals, dim=-1) # [B,N,3]
pos_mask = (normals[..., 0]>0)*2-1 # [B,N,1]
normals = normals*pos_mask # [B,N,3]
normals = random_inverse(normals) # [B,N,3]
positions = sum(normals*centroids, dim=2)/sqrt(3)
     # [B,N,1]
out = concat([centroids, normals, positions], dim=2)
     # [B,N,7]
return out
```

3.3. Triangular RepSurf

Denote a point set as $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^{N\times 3}$. Analogous to a 2D curve in Sec 3.1, we define a 3D tangent surface (Fig. 2 right) by point-normal equation. Given a normal vector $\mathbf{v}_i = (a_i, b_i, c_i)$ and a point $\mathbf{x}_i = (x_i, y_i, z_i)$, the surface can be defined as:

$$\begin{aligned} a_i\left(x-x_i\right)+b_i\left(y-y_i\right)+c_i\left(z-z_i\right)=0 \Rightarrow \\ a_i x+b_i y+c_i z-\left(a_i x_i+b_i y_i+c_i z_i\right)=0. \end{aligned} \tag{5}$$

We define the surface position as $p_i = a_i x_i + b_i y_i + c_i z_i$, with the range of $[-\sqrt{3}r, \sqrt{3}r]$. r means the edge length of a cube exactly covering the point set. For example, we utilize the normalized point clouds within the range of $[-1, 1]$ as input, so $r = 1$ here. Note that p_i can also express the directed distance between the origin and the surface. Then, we compute $\mathbf{v_i}$ by cross product. However, the computed $\mathbf{v_i}$ is unoriented — $\mathbf{v_i}$ can be pointing either inside or outside of the surface. To handle this problem, prior works [2] adopt some time-costing methods. Considering efficiency, we simplify this case by keeping a_i positive and augmenting the normals by instance-level random inverse with a probability of 50%. Thus, we define Triangular RepSurf as:

$$\mathbf{t}_i = (a_i, b_i, c_i, p_i)\,. \tag{6}$$

We define a set of Triangular RepSurf as $\mathbf{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_n\} \subseteq \mathbb{R}^{N\times 4}$. To feed point clouds into models, we replace $\mathbf{X}$ with our re-computed centroids $\mathbf{X}'$ of the triangles. Then the input can be the concatenation of $\mathbf{X}'$ and $\mathbf{T}$. A simple illustration and the implementation of Triangular RepSurf is presented in Fig. 1 and Algorithm 1, respectively.

3.4. Umbrella RepSurf

Triangular RepSurf is a lightweight method to represent the local geometry of a point cloud. However, due to limited perceptive field, it may also lead to unstable local representations. To handle this drawback, we expand the perceptive field by proposing Umbrella RepSurf inspired by umbrella curvature [10].

Algorithm 2 Pytorch-Style Pseudocode of **Umbrella RepSurf**

```
# B: batch size, N: number of points
# K: number of neighbors, C: output channels
# points: coordinates of a point set
neighbors = kNN(points, k=K)-points # [B,N,K,3]
edges = sort_by_clock(neighbors) # [B,N,K,3]
edges = unsqueeze(neighbors, dim=-2) # [B,N,K,1,3]
pairs = concat([edges, edges.roll(-1, 2)], dim=-2)
     # [B,N,K,2,3]
centroids = mean(pairs, dim=3) # [B,N,K,3]
normals = cross_product(pairs) # [B,N,K,3]
normals = normals/norm(normals, dim=-1) # [B,N,K,3]
pos_mask = (normals[..., 0, 0]>0)*2-1 # [B,N,1,1]
normals = normals*pos_mask # [B,N,K,3]
normals = random_inverse(normals) # [B,N,K,3]
positions = sum(normals*centroids, dim=3)/sqrt(3)
     # [B,N,K,1]
features = concat([centroids, normals, positions],
    dim=2) # [B,N,K,7]
features = MLPs(features, out_channel=C) #[B,N,K,C]
features = pooling(features, dim=2) # [B,N,C]
out = concat([centroids, features], dim=2)
     # [B,N,3+C]
return out
```

Denote the number of neighbors as K, the centroids and triangular features of the neighbor triangles as $\mathbf{X}'_i = \{\mathbf{x}'_{i1}, \dots, \mathbf{x}'_{iK}\} \subseteq \mathbb{R}^{K\times 3}$ and $\mathbf{T}_i = \{\mathbf{t}_{i1}, \dots, \mathbf{t}_{iK}\} \subseteq \mathbb{R}^{K\times 4}$. In [10], the unsigned scalar of umbrella curvature is defined as:

$$u_i = \sum_j^K n_{ij} = \sum_j^K \left| \frac{\mathbf{x}'_{ij}}{|\mathbf{x}'_{ij}|} \cdot \mathbf{n}_i \right|, \tag{7}$$

where $\mathbf{n}_i$ is the given normal vector of the i-th point. However, $\mathbf{n}_i$ is commonly unknown in the point set $\mathbf{X}$. This makes umbrella curvature unpractical in the real scenes. Furthermore, we argue that a scalar curvature cannot fully express the local geometry. In this case, we propose Umbrella RepSurf to express the local geometry without any given normals. Moreover, different from umbrella curvature which is defined based on homogeneous neighbors, our Umbrella RepSurf can handle heterogeneous neighbors for its position sensitivity. An illustration is shown in Fig. 4. The Umbrella RepSurf $\mathbf{u}_i$ of point $\mathbf{x}_i$ is defined as:

$$\mathbf{u}_i = \mathcal{A}\left(\left\{\mathcal{T}\left([\mathbf{x}'_{ij}, \mathbf{t}_{ij}]\right), \forall j \in \{1, \dots, K\}\right\}\right), \tag{8}$$

where $\mathcal{A}$ is an aggregation function (i.e., summation), $\mathcal{T}$ is a transformation function, and $\mathbf{x}'_{ij}$ is the normalized coordinate according to its centroid $\mathbf{x}_i$. To calculate $\mathbf{t}_{i\cdot}$, we construct adjacent triangles counterclockwise from 0° (x-axis) to 359° on the xy-plane. Thus, the number of triangles in a umbrella surface is exactly K. Note that, to keep local consistency of the normals' orientation, we compute these normals by counterclockwise cross product. (An example when reconstructing a umbrella surface unorderedly in Fig. 4.) To simplify the definition of the global normals' orientation, different from Triangular RepSurf, we keep a_{i1} of $\mathbf{t}_{i1}$ positive and the orientation of other normals changes accordingly. Therefore, though the orientation is consistent

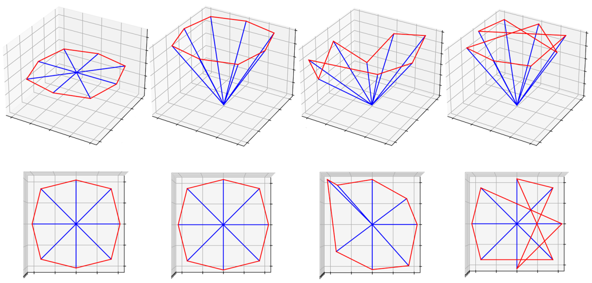

Figure 4. Examples of reconstructed umbrella surfaces. We present each surface with a regular view (above) and a top view (below). From left to right, we show two surfaces reconstructed from homogeneous neighbors, one from heterogeneous neighbors, and one reconstructed without sorting.

locally, the normals can be unoriented from a global perspective. Similar to Triangular RepSurf, we augment the normals of an umbrella surface $\mathbf{n}_{i}$. by random inverse. Instead of a predefined transformation function, we adopt a learnable function (a combination of linear functions and non-linearity) for $\mathcal{T}$. The implementation of Umbrella RepSurf is shown in Algorithm 2.

3.5. Implementation

We implement our RepSurf on the single-scale grouping (SSG) version of PointNet++ [43] in a simple manner of concatenation. For each set abstraction, we input RepSurf along with point features. An illustration of the input flow is shown in Fig. 5. Furthermore, we propose two designs to further improve our RepSurf.

Polar auxiliary. For simplicity, previous point-based models widely adopt Cartesian coordinates as input. However, they cannot fully express the relationships between a centroid and its neighbors. Unlike Cartesian coordinate system, the polar coordinate systems present a point coordinate by distance and angles according to the origin. The polar systems (i.e., Spherical system, Cylindrical system) can be a supplement for its distance and direction sensitivity. In this paper, we explore a practical application of the polar systems for point-based models. We take Spherical system as an example. After querying the neighbors of a point $\mathbf{x}_i$, we re-define the position of the j-th neighbor by including its spherical position:

$$\mathbf{x}'_{ij} = (x'_{ij}, y'_{ij}, z'_{ij}, \rho_{ij}, \theta_{ij}, \phi_{ij}), \tag{9}$$

where x'_{ij}, y'_{ij}, z'_{ij} are the values of three dimensions of the normalized Cartesian coordinate. $\rho_{ij} = \sqrt{x'^2_{ij} + y'^2_{ij} + z'^2_{ij}}$, $\theta_{ij} = \arccos \frac{z'_{ij}}{\rho_{ij}}$, $\phi_{ij} = \mathrm{atan2}(y'_{ij}, x'_{ij})$. For more details of the implementation on polar auxiliary, please refer to the supplementary material.

Channel de-differentiation. Inspired by [69], we observe that different types of inputs (i.e., coordinate, normal vectors, point features) have significant differences in data distribution. In order to process different inputs equally and to train our models stably, we explore solutions for de-differentiation along the channel dimension. In this paper, we apply Post-CD (performing batch normalization after linear function) to our method. For more details of the implementation on channel de-differentiation, please refer to the supplementary material.

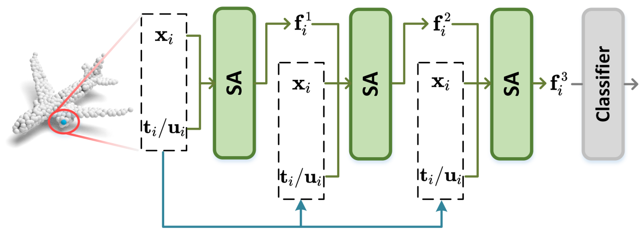

Figure 5. An overview of the input flow of RepSurf on PointNet++ for classification. $\mathbf{x}_i$, $\mathbf{t}_i$, $\mathbf{u}_i$ are the coordinate, Triangular RepSurf, Umbrella RepSurf of the i-th point of input, respectively. $\mathbf{f}_i^1$, $\mathbf{f}_i^2$, $\mathbf{f}_i^3$ are the i-th output feature of the first, second, third set abstraction (SA), respectively.

4. Experiments

We evaluate both of our Triangular RepSurf and Umbrella RepSurf on three main tasks: classification, segmentation, and detection. Furthermore, we conduct ablation studies to assess the effectiveness of our designed modules. Please refer to the supplementary material for more experimental details.

4.1. Classification

3D object classification is a basic task to prove the effectiveness of methods. We perform experiments on ModelNet40 [60], a human-made object dataset, and ScanObjectNN [54], a dataset retrieved from the real scenes.

Human-made Object Classification. ModelNet40 [60] contains 9843 training models and 2468 test models, divided into 40 categories. In Tab. 1, we compare our Triangular RepSurf (RepSurf-T) and Umbrella RepSurf (RepSurf-U) with prior methods. Equipped with RepSurf-T and RepSurf-U, the performance of PointNet++ (SSG version) is considerably boosted by 3.7% and 4.1%. For a fair comparison with other methods [31, 61, 64], we apply the strategy of multi-scale inference from [31] for further improvement. Though the results on ModelNet40 tend to be saturated, our RepSurf-U achieves 94.7%, surpassing CurveNet [61] by a large margin of 0.5%. In addition, RepSurf-U is 5.4× and 4.0× faster than CurveNet in terms of training and inference speed, respectively.

Real-world Object Classification. For the saturation of ModelNet40, we further verify our RepSurf on the hardest variant (PB_T50_RS variant) of ScanObjectNN [60], a

Method	Input	ModelNet40		ScanObjectNN		#Params	FLOPs	Train Speed	Infer Speed
		OA	mAcc	**OA**	mAcc				
PointNet [41]	1k pnts	89.2	86.0	68.2	63.4	3.47M	0.45G	1.76ms	0.81ms
DGCNN [58]	1k pnts	92.9	90.2	78.1	73.6	1.82M	2.43G	-	-
RS-CNN‡ [31]	1k pnts	93.6	-	-	-	2.38M	1.16G	-	-
KPConv [53]	~7k pnts	92.9	-	-	-	14.3M	-	218.7ms	543.7ms
PointASNL [67]	1k pnts*	93.2	-	-	-	10.1M	1.80G	-	-
Grid-GCN [65]	1k pnts*	93.1	91.3	-	-	-	-	-	42.20ms
PointTrans. [74]	1k pnts*	93.7	90.6	-	-	-	-	-	-
MVTN [15]	multi-view	93.8	**92.0**	82.8	-	4.24M	1.78G	-	-
PAConv‡ [61]	1k pnts	93.9	-	-	-	2.44M	1.68G	-	-
RPNet [44]	1k pnts*	94.1	-	-	-	2.70M	3.90G	-	-
CurveNet‡ [61]	1k pnts	94.2	-	-	-	2.14M	0.66G	22.04ms	12.34ms
PointNet++† [43]	1k pnts	90.7	88.4	77.9	75.4	1.475M	0.77G	2.75ms	1.98ms
RepSurf-T (ours)	1k pnts	**94.0** ↑3.3	91.1 ↑2.7	**84.1** ↑6.2	**81.2** ↑5.8	1.479M	0.79G	3.33ms	2.47ms
RepSurf-T‡ (ours)	1k pnts	**94.2** ↑3.5	91.3 ↑2.9	**84.3** ↑6.4	**81.6** ↑6.2	1.479M	0.79G	3.33ms	2.47ms
RepSurf-U (ours)	1k pnts	**94.4** ↑3.7	91.4 ↑3.0	**84.3** ↑6.4	**81.3** ↑5.9	1.483M	0.81G	4.08ms	3.10ms
RepSurf-U‡ (ours)	1k pnts	**94.7** ↑4.0	91.7 ↑3.3	**84.6** ↑6.7	**81.9** ↑6.5	1.483M	0.81G	4.08ms	3.10ms
RepSurf-U$^{\ddagger\circ}$ (ours)	1k pnts	-	-	**86.0**	**83.1**	6.806M	2.43G	-	-

$\dagger$: single-scale grouping (SSG) version, $\ddagger$: multi-scale inference from [31], $*$: w/ normal vector, $\circ$: PointNet++ (SSG) with double channels and deeper networks.

Table 1. Performance of classification on ModelNet40 and ScanObjectNN. We evaluate different methods in terms of **overall accuracy** (OA, %), mean per-class accuracy (mAcc, %), number of parameters (#Params), FLOPs, training speed (duration per input sample), and inference speed (duration per input sample). We consider **OA** the principle evaluation metric. **Bold** means the result outperforms prior state-of-the-art method on corresponding dataset. Green means an improvement from our RepSurf compared with the original model. We test the speed of all methods with one NVIDIA Tesla V100 GPU and four cores of Intel Xeon @2.50GHz CPU. The batch size is set to 16.

Method	S3DIS 6-fold			S3DIS Area-5			ScanNet	#Params	FLOPs
	mIoU	mAcc	OA	mIoU	mAcc	OA	mIoU		
PointNet [41]	47.6	66.2	78.5	41.1	48.9	-	-	1.7M	4.1G
PointWeb [73]	66.7	76.2	87.3	60.2	66.6	86.9	-	-	-
KPConv [53]	70.6	79.1	-	67.1	72.8	-	68.4	14.9M	-
PointASNL [67]	68.7	79.0	88.8	62.6	68.5	87.7	63.0	22.4M	19.1G
PAConv [61]	69.3	78.6	-	66.5	73.0	-	-	-	1.3G
RPNet [44]	70.8	-	-	-	-	-	68.2	2.4M	5.1G
PointTrans. [74]	73.5	81.9	90.2	**70.4**	**76.5**	**90.8**	-	4.9M	2.8G
PointNet++† [43]	59.9	66.1	87.5	56.0	61.2	86.4	-	0.969M	1.00G
RepSurf-U (ours)	**74.3**↑14.4	**82.6**↑16.5	**90.8**↑3.3	68.9↑12.9	76.0↑14.8	90.2↑3.8	**70.0**	0.976M	1.04G

$\dagger$: single-scale grouping (SSG) version, $*$: w/ normal vector.

Table 2. Performance of semantic segmentation on S3DIS (evaluation by 6-fold or on Area 5) and ScanNet V2. We evaluate different methods in terms of mean per-class IoU (mIoU, %), mean per-class accuracy (mAcc, %), overall point accuracy (OA, %), number of parameters (#Params), and FLOPs. **Bold** means the result outperforms prior state-of-the-art method on corresponding dataset. Green means an improvement from our RepSurf compared with the previous reported results of the original model.

more challenging dataset considering occlusion and background. It is composed of 2902 point clouds categorized into 15 classes. In Tab. 1, our RepSurf-T and RepSurf-U achieve 84.3% and 84.6%, outperforming prior state-of-the-art MVTN [15] by 1.5% and 1.8%, with around 1.8× fewer parameters and 1.2× fewer FLOPs.

4.2. Segmentation

Scene segmentation can be more challenging due to outliers and noise. We evaluate our RepSurf on two large-scale scene datasets, S3DIS [1] and ScanNet V2 [8].

Semantic Segmentation on S3DIS. S3DIS [1] contains 271 scenes from 6 indoor areas. Each point is categorized into 13 types of semantic labels. In Tab. 2, we evaluate our RepSurf on S3DIS by 6-fold and on Area-5. Our RepSurf-U significantly improves PointNet++ by 14.4% mIoU and 12.9% mIoU on S3DIS 6-fold and S3DIS Area-5, respectively. Furthermore, our RepSur-U outperforms previous state-of-the-art, Point Transformer [74] by 0.8% mIoU on S3DIS 6-fold, and achieves comparable performance on S3DIS Area-5 as well. Simultaneously, our RepSurf-U has 4.0× fewer parameters and 1.7× fewer FLOPs with a com-

Method	Backbone	ScanNetV2		SUN RGB-D		#Params	Infer Speed
		mAP@0.25	mAP@0.5	mAP@0.25	mAP@0.5		
VoteNet [39]	PointNet++	62.9	39.9	59.1	35.8	-	-
ImVoteNet [38]	PointNet++	-	-	63.4*	-	-	-
H3DNet [72]	PointNet++	64.4	43.4	-	-	-	-
H3DNet [72]	4×PointNet++	67.2	48.1	60.1	39.0	-	266ms
3DETR [35]	Transformer	65.0	47.0	59.1	32.7	-	-
BRNet [6]	PointNet++	66.1	50.9	61.1	43.7	-	-
GroupFree^{6,256}	PointNet++	67.3	48.9	63.0	45.2	11.49M	149ms
GroupFree^{6,256}	**RepSurf-T**	**68.4** ↑1.1	50.3 ↑0.4	**63.9** ↑0.9	**45.6** ↑0.4	11.50M	149ms
GroupFree^{6,256}	**RepSurf-U**	**68.8** ↑1.5	50.5 ↑0.6	**64.3** ↑1.3	**45.9** ↑0.7	11.50M	150ms
GroupFree^{12,512}	PointNet++^{2}	69.1	52.8	-	-	23.60M	193ms
GroupFree^{12,512}	**RepSurf-T^2**	**70.4** ↑1.3	**54.6** ↑1.8	**64.2**	**47.1**	23.60M	194ms
GroupFree^{12,512}	**RepSurf-U^2**	**71.2** ↑2.1	**54.8** ↑2.0	**64.9**	**47.7**	23.61M	195ms

$*$: w/ RGB as input, Model^2: Model with doubled channels for each MLP, 4×PointNet++: four individual PointNet++ (SSG) in [72], GroupFree^{a,b}: GroupFree model [33] with a a-layer decoder and b object candidates.

Table 3. Performance of object detection on ScanNet V2 and SUN RGB-D. We evaluate different methods in terms of mAP@0.25, mAP@0.5, number of parameters (#Params), and inference speed (duration per input sample). **Bold** means the result outperforms prior state-of-the-art method on corresponding dataset. Green means an improvement from our RepSurf compared with the original model. We test the speed of all methods with one NVIDIA Titan-XP GPU and four cores of Intel Xeon @2.50GHz CPU.

parison of Point Transformer.

Semantic Segmentation on ScanNet. ScanNet V2 [8] consists of 1513 indoor training point clouds and 100 test point clouds. It marks each point with 21 categories. In Tab. 2, the performance of RepSurf-U exceeds prior state-of-the-art KPConv [53] by 1.6%. Moreover, our method has 14.3× fewer parameters compared with KPConv.

4.3. Detection

3D detection can further prove the superiority of our method at the application level. We conduct experiments on two widely adopted 3D object detection datasets: ScanNet V2 [8] and SUN RGB-D [48]. We adopt a powerful method [33] for pipeline and replace the backbone with our RepSurf to perform all experiments on this task. Our experiments are mainly based on the codebase[11] of [33] as well.

Detection on ScanNet. ScanNet V2 [8] can be adopted for 3D detection as well, consisting of 1513 indoor scenes and 18 object classes. We follow the standard evaluation protocol in [39] by utilizing mean Average Precision under the thresholds of 0.25 (mAP@0.25) and 0.5 (mAP@0.5), without considering the orientation of bounding boxes. As shown in Tab. 3, with almost no increase in computational cost (∼0.01M parameters and ∼1ms inference speed), our RepSur-U boosts the performance of previous state-of-the-art [33] by 2.1% mAP@0.25 and 2.0% mAP@0.25.

Detection on SUN RGB-D. SUN RGB-D [48] is a single-view RGB-D dataset for 3D scene analysis, including around 5K indoor RGB and depth images. Following [39], we adopt mean Average Precision on 10 most common categories for evaluation. In Tab. 3, RepSurf-U improves GroupFree^{6,256} ([33] with a 6-layer encoder and 256 object candidates) by 1.3% mAP@0.25 and 0.7% mAP@0.5. Without RGB as input, GroupFree^{12,512} equipped with RepSurf-U even outperforms prior state-of-the-art ImVoteNet [38] by 1.5% mAP@0.25.

[11] https://github.com/zeliu98/Group-Free-3D

type	$\mathcal{X}$-computed	w/ p_i	w/ inverse	acc.
given	-	✗	✗	94.08
given	-	✗	✓	93.39
given	-	✓	✗	93.95
triangular	pre	✓	✗	93.57
triangular	post	✓	✗	93.62
triangular	post	✓	✓	**94.02**
umbrella	pre	✓	✗	93.06
umbrella	post	✓	✗	93.90
umbrella	post	✓	✓	**94.46**

Table 4. Ablation study on the types of RepSurf. (given: normal vectors given from the dataset, triangular: Triangular RepSurf, umbrella: Umbrella RepSurf, $\mathcal{X}$-computed: computing RepSurf before (pre-computed) or after (post-computed) sampling, w/ p_i: with surface position p_i input, w/ inverse: augmenting RepSurf by random inverse, acc.: overall accuracy)

4.4. Ablation study

We ablate some vital designs of our method on ModelNet40 for an insightful exploration.

Types of RepSurf. Shown in Tab. 4, we compare different types of input (given normals, RepSurf-T, RepSurf-U). We further discuss on when to compute RepSurf. Regularly, we obtain the input point clouds after a process of sampling (i.e., 10000 → 1024 points). Before this process (pre-computed), we will derive RepSurf from high-resolution point clouds, which means RepSurf approximates the cor-

input	#channels	BN	bias	$\mathcal{A}$	#layers	acc.
N	3	✗	✗	sum	1	93.17
N+P	4	✗	✗	sum	1	93.24
N+C	6	✗	✗	sum	1	93.18
N+P+C	7	✗	✗	sum	1	93.38
N+P+CP	10	✗	✗	sum	1	93.45
N+P+CP	10	✗	✓	sum	1	93.86
N+P+CP	10	✗	✓	sum	2	93.94
N+P+CP	10	✗	✓	max	3	94.04
N+P+CP	10	✗	✓	mean	3	94.37
N+P+CP	10	✓	✓	sum	3	94.06
N+P+CP	10	✗	✓	sum	3	**94.46**

Table 5. Ablation study on the design of Umbrella RepSurf block. (N: normal vector (a_i, b_i, c_i), P: surface position p_i, C: centroid position $(x'_{ij}, y'_{ij}, z'_{ij})$, CP: centroid position $(x'_{ij}, y'_{ij}, z'_{ij})$ with polar auxiliary $(\rho_{ij}, \theta_{ij}, \phi_{ij})$), #channels: number of input channels, BN: applying batch normalization, bias: applying learnable bias in the first layer, $\mathcal{A}$: aggregation function, #layers: number of MLP layers for mapping, acc.: overall accuracy)

responding tangent. However, empirical results show that post-computed works better than pre-computed. We additionally test on the designs of surface position and random inverse, both of which slightly improve RepSurf.

Design of RepSurf block. Shown in Tab. 5, we explore the design of Umbrella RepSurf in terms of input, transformation function $\mathcal{T}$, and aggregation function $\mathcal{A}$. Empirically, a combination of normal vector, surface position, normalized coordinate and the corresponding polar coordinates outperforms other combinations. Furthermore, prohibition of batch norm, usage of bias for the first layer, sum-pooling, and three-layer MLP perform better than other options.

Group size. We explore the group size of Umbrella RepSurf in terms of both accuracy and speed (ms per sample):

PN2	k=2	k=4	k=6	**k=8**	k=10	k=12	k=16
acc.	93.53	93.63	94.36	**94.46**	94.32	94.20	94.32
time	0.50	0.46	0.49	0.48	0.58	0.51	0.78

We test the speed of Umbrella RepSurf block only. When k=2, Umbrella RepSurf will degenerate to a learnable version of Triangular RepSurf. There is almost no difference in speed when k is in the range of $[2, 12]$. For a trade-off between performance and speed, we consider k=8 an ideal choice. Furthermore, when we study on larger group sizes (i.e., 24), a vanishing gradient problem exists. We argue that larger umbrella surfaces may become more indistinguishable and lead to the problem, but this is still an open issue.

Polar auxiliary. We study on the design of our polar auxiliary in different versions:

PN2	w/o aux.	w/ ρ	w/ cylinder	w/ sphere
acc.	93.97	94.12 ↑0.15	93.89 ↓0.08	**94.46** ↑0.49

Here ρ, a part of spherical polar auxiliary, means the distance between a centroid and its neighbors. We discuss that

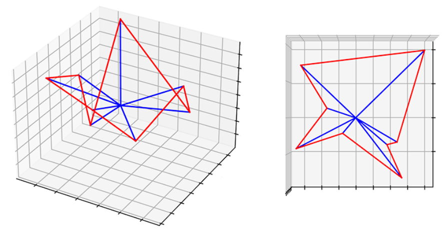

Figure 6. Bad case of a reconstructed umbrella surface when the neighbors are extremely messy.

Spherical system can better express the geometric relations between the centroids and their neighbors, an auxiliary of Cartesian system. Empirical results verify this hypothesis.

Channel de-differentiation. We test the design of channel de-differentiation (CD) on three versions of PointNet++, including the original (vanilla), Triangular RepSurf (triangular), and Umbrella RepSurf (umbrella):

PN2	none	Pre-CD	Post-CD
vanilla	93.15	92.70 ↓0.36	**94.08** ↑0.93
triangular	93.22	92.49 ↓0.73	**94.02** ↑0.80
umbrella	93.50	92.63 ↓0.87	**94.46** ↑0.96

Here Pre-CD means that batch normalization performs before linear function, and Post-CD is the opposite. We argue Post-CD performs better than Pre-CD, since Pre-CD may blur the original semantics of external input (i.e., coordinates, RepSurf features).

5. Discussion

Limitation. Though simple and effective, RepSurf may suffer from noises while surface reconstruction due to the noise-sensitive algorithm kNN. Furthermore, we argue that Umbrella RepSurf may be vulnerable to extremely messy points. Thus, when we query more neighbors of a point, in general the distribution of its neighbors would become messy and results in a distorted surface. An example of bad case is shown in Fig. 6.

Conclusion. We present two variants of RepSurf, Triangular and Umbrella RepSurf, to explore the surface representation on point clouds. We evaluate our simple baseline on various tasks, including shape classification, scene segmentation and detection. The evaluation results show its astonishing efficiency and performance, superior to the previous state-of-the-art on different benchmarks.

We hope our work can inspire the community and evoke the rethinking on the explicit representation of point clouds. We believe that RepSurf deserves further exploration for different fields (i.e., autonomous driving) or on larger-scale point clouds, since RepSurf is eligible to handle numerous background points in the real scenes. RepSurf may also be helpful for point cloud sampling by its ability on geometry sensitivity. It would be worthy of solving the above limitations of RepSurf as well.

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 6

[2] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329. Wiley Online Library, 2017. 4

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 2

[4] Jintai Chen, Biwen Lei, Qingyu Song, Haochao Ying, Danny Z Chen, and Jian Wu. A hierarchical graph network for 3d object detection on point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 392–401, 2020. 2

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 2

[6] Bowen Cheng, Lu Sheng, Shaoshuai Shi, Ming Yang, and Dong Xu. Back-tracing representative points for voting-based 3d object detection in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8963–8972, 2021. 2, 7

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 2

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 6, 7

[9] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–272, 2018. 2

[10] A Foorginejad and K Khalili. Umbrella curvature: a new curvature estimation method for point clouds. *Procedia Technology*, 12:347–352, 2014. 2, 4

[11] Kent Fujiwara and Taiichi Hashimoto. Neural implicit embedding for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11734–11743, 2020. 2

[12] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018. 2

[13] 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. 2

[14] Haiyun Guo, Jinqiao Wang, Yue Gao, Jianqiang Li, and Hanqing Lu. Multi-view 3d object retrieval with deep embedding network. *IEEE Transactions on Image Processing*, 25(12):5526–5537, 2016. 2

[15] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1–11, 2021. 2, 6

[16] Zhizhong Han, Mingyang Shang, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and CL Philip Chen. Seqviews2seqlabels: Learning 3d global features via aggregating sequential views by rnn with attention. *IEEE Transactions on Image Processing*, 28(2):658–672, 2018. 2

[17] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018. 2

[18] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. 2

[19] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10433–10441, 2019. 2

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 2

[21] 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

[22] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 2

[23] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7578–7588, 2020. 2

[24] Eric-Tuan Le, Iasonas Kokkinos, and Niloy J Mitra. Going deeper with lean point networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9503–9512, 2020. 2

[25] 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. 1, 2

[26] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018. 2

[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. 2

[28] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1800–1809, 2020. 2

[29] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8778–8785, 2019. 2

[30] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5239–5248, 2019. 2

[31] 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. 1, 2, 5, 6

[32] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *European Conference on Computer Vision*, pages 326–342. Springer, 2020. 2

[33] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. *arXiv preprint arXiv:2104.00678*, 2021. 2, 7

[34] Daniel Maturana and Sebastian 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

[35] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021. 2, 7

[36] Ehsan Nezhadarya, Ehsan Taghavi, Ryan Razani, Bingbing Liu, and Jun Luo. Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12956–12964, 2020. 2

[37] Xing Nie, Yongcheng Liu, Shaohong Chen, Jianlong Chang, Chunlei Huo, Gaofeng Meng, Qi Tian, Weiming Hu, and Chunhong Pan. Differentiable convolution search for point cloud processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7437–7446, 2021. 2

[38] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4404–4413, 2020. 2, 7

[39] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. 2, 7

[40] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. 2

[41] 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, 3, 6

[42] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 2

[43] 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, 5, 6

[44] Haoxi Ran, Wei Zhuo, Jun Liu, and Li Lu. Learning inner-group relations on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15477–15487, 2021. 1, 2, 6

[45] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017. 2

[46] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 2

[47] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 2

[48] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 7

[49] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 808–816, 2016. 2

[50] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018. 2

[51] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik 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

[52] Brook Taylor. *Methodus incrementorum directa et inversa*. Innys, 1717. 1, 3

[53] 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, 2, 6, 7

[54] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1588–1597, 2019. 5

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[56] 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. 2

[57] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 2

[58] 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):1–12, 2019. 1, 2, 6

[59] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 2

[60] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong 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. 2, 5

[61] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. *arXiv preprint arXiv:2105.01288*, 2021. 2, 5, 6

[62] Jin Xie, Guoxian Dai, Fan Zhu, Edward K Wong, and Yi Fang. Deepshape: Deep-learned shape descriptor for 3d shape retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1335–1345, 2016. 2

[63] Bin Xu and Zhenzhong Chen. Multi-level fusion based 3d object detection from monocular images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2345–2353, 2018. 2

[64] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021. 2, 5

[65] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5661–5670, 2020. 1, 2, 6

[66] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 2

[67] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang 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, 6

[68] 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. 2

[69] Zetong Yang, Yanan Sun, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Cn: Channel normalization for point cloud recognition. In *European Conference on Computer Vision*, pages 600–616. Springer, 2020. 5

[70] Xiaopeng Zhang, Hongjun Li, Zhanglin Cheng, et al. Curvature estimation of 3d point cloud surfaces through the fitting of normal section curvatures. *Proceedings of ASIAGRAPH*, 2008:23–26, 2008. 2

[71] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1607–1616, 2019. 2

[72] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *European Conference on Computer Vision*, pages 311–329. Springer, 2020. 2, 7

[73] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019. 2, 6

[74] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 2, 6

[75] Haoran Zhou, Yidan Feng, Mingsheng Fang, Mingqiang Wei, Jing Qin, and Tong Lu. Adaptive graph convolution for point cloud analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4965–4974, 2021. 2

[76] 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. 2