CurveCloudNet: Processing Point Clouds with 1D Structure

Colton Stearns, Stanford University
Alex Fu, Stanford University
Jiateng Liu, UIUC
Jeong Joon Park, University of Michigan
Davis Rempe, NVIDIA
Despoina Paschalidou, Stanford University
Leonidas J. Guibas, Stanford University

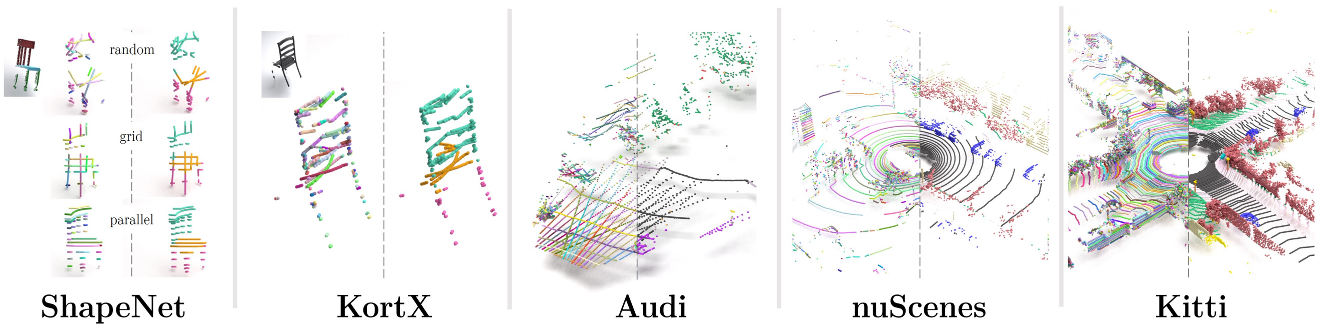

Figure 1. Visualizations of the input curve cloud (left) and CurveCloudNet's segmentation prediction (right) for each of our five evaluation datasets. Each evaluation dataset exhibits distinct size, structure, and laser scanning pattern, as shown in Tab. 1

Abstract

Modern depth sensors such as LiDAR operate by sweeping laser-beams across the scene, resulting in a point cloud with notable 1D curve-like structures. In this work, we introduce a new point cloud processing scheme and backbone, called CurveCloudNet, which takes advantage of the curve-like structure inherent to these sensors. While existing backbones discard the rich 1D traversal patterns and rely on generic 3D operations, CurveCloudNet parameterizes the point cloud as a collection of polylines (dubbed a "curve cloud"), establishing a local surface-aware ordering on the points. By reasoning along curves, CurveCloudNet captures lightweight curve-aware priors to efficiently and accurately reason in several ***diverse*** *3D environments. We evaluate CurveCloudNet on multiple synthetic and real datasets that exhibit distinct 3D size and structure. We demonstrate that CurveCloudNet outperforms both point-based and sparse-voxel backbones in various segmentation settings, notably scaling to large scenes better than point-based alternatives while exhibiting improved single-object performance over sparse-voxel alternatives. In all, CurveCloudNet is an efficient and accurate backbone that can handle a larger variety of 3D environments than past works.*

1. Introduction

The computer vision community has proposed many backbones for processing 3D point clouds for fundamental tasks such as semantic segmentation [21, 41, 42, 48, 53] and object detection [52, 56, 57, 60, 71]. Existing 3D backbones can be generally characterized as point-based or discretization-based. Backbones that directly operate on 3D points [14, 22, 42, 46, 48, 51, 58, 67] typically exchange and aggregate point features in Euclidean space, and have shown success for individual objects or relatively small indoor scenes. These methods, however, do not scale well to large scenes (e.g. in outdoor settings) due to inefficiencies in processing large unstructured point sets. On the other hand, popular discretization approaches such as sparse voxel methods [12, 18, 21, 34, 46, 55, 75, 81] rely on efficient sparse data structures that scale better to large scenes. However, for small or irregularly-distributed point sets, they often incur discretization errors.

In recent years, this trade off between point and voxel backbones has been less explored due to the distinct environments in most 3D applications - autonomous vehicles do not leave roads, manufacturing robots do not leave warehouses, and quality-assurance systems do not look beyond a tabletop. However, as the community moves to dynamic

Method / mIOU (↑)	*Type*	*AVG*	KortX	ShapeNet	A2D2	nuScenes	Kitti
PointNet++ [42]	Point	62.2	71.0	80.1	46.5	51.1	–
CurveNet [61]		52.9	71.5	<u>82.8</u>	4.4	–	–
PointMLP [36]		62.3	<u>75.4</u>	80.9	47.6	67.9	39.5
PointNext [43]		66.6	73.7	<u>82.8</u>	45.0	65.0	–
MinkowskiNet [12]	Voxel	67.6	60.1	81.1	53.8	76.2	66.8
Cylinder3D [81]		67.6	63.5	79.6	53.0	76.1	65.9
SphereFormer [26]		70.6	69.7	79.5	**55.1**	**79.5**	<u>69.0</u>
CurveCloudNet (ours)	Curve	**72.7**	**78.9**	**83.1**	<u>54.1</u>	<u>78.0</u>	**69.5**

Dataset Statistic	ShapeNet	KortX	A2D2	nuScenes	Kitti
location	Synthetic	Indoor	Outdoor	Outdoor	Outdoor
scale	± 1	± 2m	+ 70m	± 50m	± 50m
laser pattern	ALL	Random	Grid	Parallel	Parallel
# points	2048	2048	∼ 8K	∼ 35K	∼ 100K
# train	12K	6K	18K	28K	19K
train/val gap	✗	✓	✗	✗	✗

Table 1. *Dataset and Performance Overview* **(Left)** CurveCloudNet achieves the best mIOU on average and is best or second-best for every dataset. Empty entries indicate excessive training time that exceeds 20 days. Validation splits are reported because not all baselines were submitted to test servers. **(Right)** We evaluate on five segmentation benchmarks that exhibit diverse size, structure, and training settings. Refer to Fig. 1 for illustrations of the parallel, random, and grid laser patterns.

and unregulated settings such as open-world robotics (e.g. embodied agents), it is essential to have architectures that consistently perform well in diverse settings.

To this end, we present a novel point cloud processing scheme that achieves both performance and flexibility across diverse 3D environments. We achieve this by tailoring our approach to the popular family of laser-scanning 3D sensors (such as LiDARs), which gather 3D measurements by sweeping laser-beams across the scene. While previous works ignore the innate curve-like structures of the scanner outputs, we parameterize the point cloud as a collection of polylines, which we refer to as a "curve cloud". Our formulation establishes a local structure on the points, allowing for efficient and cache-local communication between points along a curve. This enables scaling to large scenes without incurring discretization errors and/or computational overhead. Furthermore, we hypothesize that the local curve ordering injects a lightweight and flexible surface-aware prior into the network (see Sec. 3.1).

We propose a new backbone, CurveCloudNet, that applies 1D operations along curves and combines curve operations with state-of-the-art point-based operations [36, 42, 43, 53]. CurveCloudNet uses curve operations at higher resolutions when there is clearer curve structure and uses point operations at downsampled resolutions. Put together, CurveCloudNet is an efficient, scalable, and accurate backbone that can outperform segmentation and classification pipelines in a variety of settings (see Tab. 1a).

We evaluate CurveCloudNet on a variety of object-level and outdoor scene-level datasets that exhibit distinct 3D size, structure, and unique laser scanning patterns (see Tab. 1b and Fig. 1): this includes indoor, outdoor, object-centric, scene-centric, sparsely scanned, and densely scanned scenes. We evaluate CurveCloudNet on the object part segmentation task using the ShapeNet [8, 74] dataset along with a new real-world object-level dataset captured with the Kortx scanning system [1]. For the outdoor semantic segmentation task, we use the nuScenes [6], Audi Autonomous Driving (A2D2) [17], and Semantic Kitti [4, 16] datasets. Supplementary experiments on object classification demonstrate flexibility to other perception tasks. Our evaluations demonstrate that using curve structures leads to improved or competitive performance on *all* experiments, with the best performance on average (see Tab. 1a).

In summary, we make the following **contributions**: **(1)** we propose operating on laser-scanned point cloud data using a *curve cloud* representation, **(2)** we design efficient operations that run on polyline curves, **(3)** we design a novel backbone, CurveCloudNet, that strategically combines both curve and point operations, and **(4)** we show accurate and efficient segmentation results on real-world data captured for both objects and large-scale scenes in multiple environments and with various scanning patterns.

2. Related Work

Existing point cloud methods can be roughly characterized as point-based and discretization-based approaches. As our work addresses trade-offs between them, we discuss related works from each category.

Point-Based Networks. One popular approach to point-based reasoning is to aggregate local neighborhood information in a hierarchical manner and at multiple geometric scales [21, 30, 36, 41–43, 73, 78, 79]. Recently, Ma et al. [36] introduced a modern MLP-based architecture along these lines, while Quian et al. [43] modernized the seminal PointNet++ [42] – both showed compelling results on object-level and indoor scenes. Nevertheless, most hierarchical and MLP point networks are inefficient in large-scale settings, and although several backbones [21, 73, 78] have addressed this, they trade off scalability with task-specific frameworks or lower accuracy. In contrast, CurveCloudNet scales to large scenes by using the explicit curve structure of laser scanners.

Another line of research makes use of point convolutions for learning per-point local features. Point convolutions are usually defined using *kernels* [14, 22, 24, 27, 29, 37, 46, 48, 51, 54, 58, 67] or *graphs* [9, 13, 31, 47]. Kernels have been defined using a family of polynomial functions [67], using MLPs [33, 51], or directly using local 3D point coordinates [3, 5, 48, 58, 66]. In contrast, graph methods usually

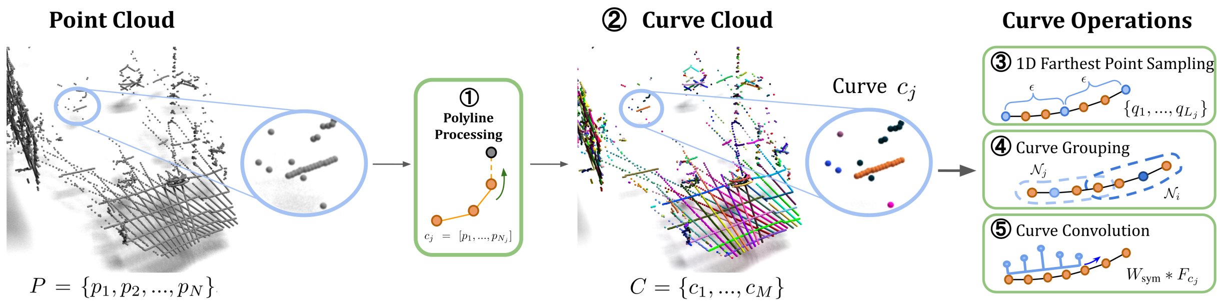

Figure 2. *Overview of Curve Cloud Reasoning.* Starting from laser-scanned input data, we ① link points into polylines to ② parameterize the point cloud as a curve cloud (see Sec. 3.1). We develop operations for learned architectures to specifically exploit the curve structure, including ③ 1D farthest-point-sampling along a curve, ④ curve grouping, and ⑤ symmetric curve convolutions (see Sec. 3.2).

construct a K-Nearest-Neighbors graph in Euclidean space [45, 50] or feature space [53], and apply graph-convolutions on the resulting edges and vertices. More recently, CurveNet [61] applied guided random walks on uniformly-sampled input point clouds to construct graph neighborhoods that go beyond K-Nearest-Neighbors and that exhibit 1D "curve" structure; then, CurveNet pooled features over the traversed curves. Aside from defining "curves", CurveNet and CurveCloudNet have little in common: CurveNet's guided random walks are not related to physical laser traversals and do not scale to large scenes. In contrast, CurveCloudNet efficiently recovers explicit curves from a scanner's physical laser traversals, and then applies a variety of operations, e.g., subsampling, aggregation, and convolution, along each curve.

Many works [15, 32, 39, 59, 62, 72, 76, 79] have shown success with *attention-based aggregation* using transformer architectures with self-attention [49]. However, we found CurveCloudNet's reasoning over local 1D "curve" neighborhoods to be sufficiently expressive without attention.

Discretization-Based Networks. Although point-based backbones can successfully process individual objects or small indoor scenes, they struggle to scale to large point clouds due to inefficiency in processing large unstructured point sets. To address this, several works [12, 18, 20, 28, 34, 35, 46, 64, 70, 75, 77, 81] proposed to convert a point cloud into a 3D voxel grid and use this volumetric representation. Early works converted a point cloud into a dense voxel grid and applied dense 3D convolutions [38, 40], however the cubic size of the dense grid proved to be computationally prohibitive. To scale to large scenes, several works [11, 12, 20, 28, 35, 64, 65, 68, 69, 81] employed the sparse-voxel data structure from [18]. MinkowskiNet [12] was a seminal work in showing that sparse voxel convolutions can be highly efficient and expressive. PVNAS [35] incorporated a network architecture search, demonstrating the importance of the architecture channels, network depth, kernel sizes, and training schedule. More recent works have supplemented sparse-voxel backbones with attention operations [11], range-view and point information [65], image information [69], and knowledge distillation [20].

Other methods seek a better discretization of point clouds captured with LiDAR scans. For example, PolarNet [77] proposed to partition input points using grid cells defined in a polar coordinate system, while Cylinder3D [81] employed a cylindrical partitioning scheme based on a cylindrical coordinate system. Sphereformer [26] combined polar grid cells with modern attention operations. In an alternative line of research, many methods [2, 10, 19, 25, 55, 58, 63, 80] employ spherical or bird's-eye view projections to represent point clouds as images that are passed to a 2D convolutional or transformer network.

Unlike discretization methods, CurveCloudNet directly operates on points and curves, scaling to larger scenes without discretizing. Additionally, our curve operations are applied locally and do not assume global patterns such as polar, cylindrical, or planar structure.

3. Method: Learning on Curve Clouds

Our method takes as input a 3D point cloud, parses it into a curve cloud representation, and then processes the resulting curves by leveraging specialized curve operations, as shown in Fig. 2. We focus on object part segmentation, semantic scene segmentation, and object classification, although in principle, our method is suitable for more perception tasks.

3.1. Constructing Curve Clouds

Problem Formulation. The input to our model is the output of a laser-based 3D sensor represented as a point cloud $P = \{p_1, p_2, ..., p_N\}$, where $p_i = [x_i, y_i, z_i]$ is the 3D coordinates of the i-th point. For each point, we are also given an associated acquisition timestamp t_i and an integer laser beam ID $b_i \in [1, B]$, which are readily available from sensors like LiDAR. For a scanner with B unique laser beams, b_i indicates which beam captured the point while t_i gives the ordering in which points were captured. Timestamps differ only by microseconds and indicate point ordering for

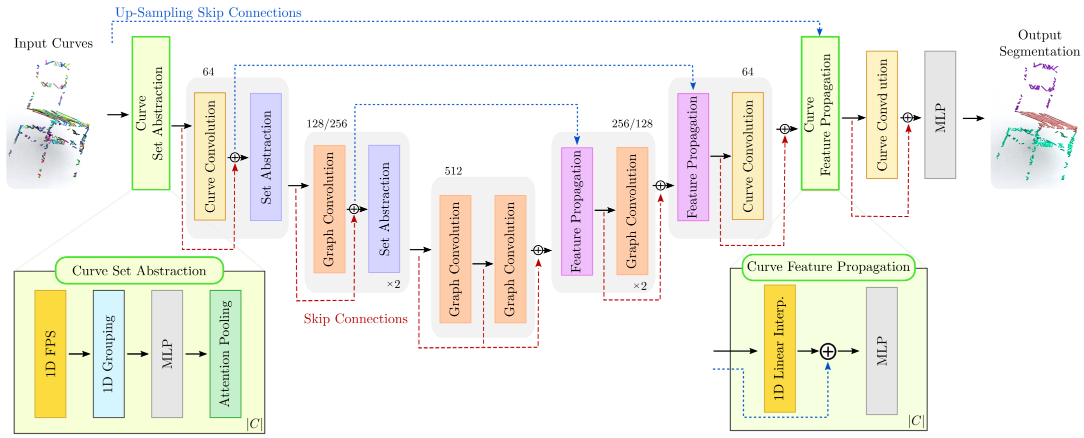

Figure 3. *CurveCloudNet Architecture.* The network employs a mix of curve and point layers to process a curve cloud through progressive down-sampling followed by up-sampling with skip connections. Curve layers operate on higher resolutions to efficiently capture the 1D structure, while at lower resolutions point layers propagate information across curves. Feature dimensions are listed above each block.

constructing the curve cloud; otherwise, the point cloud is treated as an instantaneous capture of the scene.

We assume that each laser beam in the scanner captures 3D points *sequentially* and at a *high sampling rate* as it sweeps the scene. Concretely, if points p_1, p_2, and p_3 are recorded consecutively by beam b, then their timestamps are ordered $t_1 < t_2 < t_3$. Moreover, if two consecutive points are spatially farther apart than some small threshold δ, we assume there is a surface discontinuity and the points lie on different surfaces in the scene.

Curve Clouds. A curve $c_j = [p_1, ..., p_{N_j}]$ is defined as a sequence of N_j points where consecutive point pairs are connected by a line segment, i.e., a *polyline*. The curve is bi-directional and is equivalently defined as $c_j = [p_{N_j}, p_{N_j - 1}, ..., p_1]$. A *curve cloud* $C = \{c_1, ..., c_M\}$ is an unordered set of M curves where each curve may contain a different number of points. In practice, we can store a curve cloud C as an $N \times 3$ tensor of points, with an additional *ptr* tensor of length M that specifies the indices where one curve ends and the next begins. Converting the input point cloud P to a corresponding curve cloud is straightforward and extremely efficient: points from each beam b_i are ordered by timestamp and split into curves based on distances between consecutive points. If the distance between two consecutive points is more than a set threshold δ while traversing the points in time order, then the current curve ends and a new curve begins, i.e., a surface discontinuity has occurred. In practice, we parallelize this process across all points and laser beams on the GPU. More details regarding the conversion process are provided in the supplement.

Why Use Curves? Curve clouds inherit the benefits associated with the 3D point cloud representation including lightweight data structures and no need to discretize the space. But operating on curve clouds also has several advantages over point clouds. Point clouds are highly unstructured, making operations like nearest neighbor queries and convolutions expensive. Curve clouds add structure through point ordering along the polylines, allowing curves to be treated as 1D grids that permit greater cache-locality, constant-time neighborhood queries, and efficient convolutions. This structure is flexible to any laser scanning pattern unlike, e.g., cylindrical voxel grids and polar range-view projections. In principle, the curve structure should also bring out the geometric properties of the surface it represents, such as curvature, tangents, and boundaries.

3.2. Operating on Curves

We now discuss the fundamental operations for curves. We first introduce the network layers for curves followed by the details of curve operations used in these layers.

3.2.1 Curve Layers

Curve Set Abstraction (Curve SA). Inspired by set abstraction layers from prior point-based work [42], this layer adopts a curve-centric procedure to downsample the number of points on the curves. For each curve, Curve SA (1) samples a subset of "centroid" points along the curve using our *1D farthest point sampling* algorithm, (2) groups points around these centroids in local neighborhoods using *curve grouping*, (3) translates points into the local frame of their centroid and processes them with a shared MLP, and (4) pools over each local neighborhood to get a downsampled curve with associated point features.

Curve Feature Propagation (Curve FP). Similar to point-based feature propagation [42], Curve FP is a curve-centric upsampling layer. For each curve c_j, this layer propagates features from the low-res polyline $[q_1, ..., q_{L_j}]$ with L_j points to a higher-res polyline $[p_1, ..., p_{H_j}]$ with H_j points where $L_j < H_j$. This is achieved using *curve feature interpolation* as described in Sec. 3.2.2. Afterwards, the high-res interpolated features are concatenated with skip-linked features from a corresponding curve set abstraction layer and processed by a shared MLP.

Curve Convolution. The curve convolution layer allows for efficient communication and feature extraction along a curve. This module consists of three sequential *symmetric curve convolutions*, each followed by batch normalization [23] and a leaky ReLU activation.

3.2.2 Curve Operations

To enable our layers to expressively and efficiently learn on curve clouds, we formulate operations for *sampling*, *grouping*, *feature interpolation*, and *convolutions* along curves.

1D Farthest Point Sampling (FPS). FPS is frequently a bottleneck in point cloud architectures and can be costly for large point clouds [21] due to pair-wise distance computations. For curves, we alleviate this cost with an approximation of FPS *along each curve* independently in a 1D manner. This amounts to sampling a subset of points on each curve that are evenly-spaced along the length of that curve (i.e., geodesically). For a curve c_j with N_j points, we subsample points $[q_1, ..., q_{L_j}]$ with $L_j < N_j$ such that all pairs of contiguous points are about ϵ apart, where ϵ is a fixed target spacing shared across *all* curves. In other words, $d(q_i, q_{i-1}) \approx \epsilon$ for $i = 2, \ldots, L_j$ where d measures the geodesic distance between two points along the same curve. Notably, this algorithm has only $O(N)$ complexity and can be parallelized across each curve independently.

Grouping Along Curves. After sampling, we must group points into local neighborhoods around the subsampled points on each curve. We adapt a ball query [42], which groups together all points within a specified radius from a centroid, to operate along each curve. For a centroid point p_i belonging to curve c_j, we define the local neighborhood of p_i as $\mathcal{N}_i = \{p_k \in c_j \,|\, d(p_i, p_k) < r\}$ where r is a fixed neighborhood radius. In addition to being computationally faster than a standard 3D ball query grouping, using 1D curve groupings ensures that all neighborhoods lie on a continuous section of scanned surfaces.

Curve Feature Interpolation. To upsample on curves, we must interpolate features $[g_1, ..., g_{L_j}]$ from a lower-resolution polyline to features $[f_1, ..., f_{H_j}]$ for a higher-resolution one. Let p_h be the hth point on the high-res curve, which falls between subsampled low-res points q_i and q_{i-1} with associated features g_i and g_{i-1}. The interpolated high-res feature f_h is simply the distance-weighted interpolation of the two low-res point features (based on their spatial coordinates).

Symmetric Curve Convolution. To process points along curves, we take advantage of expressive convolutions. However, it is computationally burdensome to compute neighborhoods on the fly and run convolutions on unordered data [35]. Instead, we treat each curve as a discrete grid of features that can be convolved similar to a 1D sequence. To account for the bi-directionality of curves, we employ a symmetric convolution and thus produce equivalent results when applied "forward" or "backward" along the curve.

In particular, for a curve $c_j = [p_1, p_2, ..., p_{N_j}]$ with associated point features $F_j = [f_1, f_2, ..., f_{N_j}]$, we start by extracting additional features using the the L1 norm of feature gradients along the curve [54], denoted as $\nabla F_j = \left[|\nabla f_1|, |\nabla f_2|, ..., |\nabla f_{N_j}|\right]$. Note the norm is necessary to remove directional information. Concatenating these features together as $[F_j, \nabla F_j] \in \mathbb{R}^{N_j \times D}$ gives a grid on which to perform 1D convolutions. To respect bi-directionality, symmetric kernels are used for the convolution: for a kernel $W \in \mathbb{R}^{S \times D}$ with size S and D channels, we ensure $W_i = W_{S-i+1}$ for $i = 1, \ldots, S$ where $W_i \in \mathbb{R}^D$.

3.3. Curve Cloud Backbone: CurveCloudNet

In Fig. 3, we illustrate CurveCloudNet as designed for segmentation tasks where the output is a semantic class for each point in the input point cloud. Hence, it follows the U-Net [44] structure, consisting of a series of downsampling layers followed by upsampling with skip connections. Although our experiments (Sec. 4) focus on segmentation, CurveCloudNet can be adapted to other point cloud perception tasks (see supplement for a classification example).

Our architecture is a mix of curve and point-based layers. At higher resolutions, curve modules are employed since they are efficient and can capture geometric details when curve sampling is most dense across surfaces in the scene. At lower resolutions, point modules are used to propagate information across curves when 1D structure is less apparent. For point operations, we adopt the set abstraction and feature propagation operations from [42] as well as the graph convolution from [53], and we further improve these point operations following the reportings of recent works [21, 36, 43] (see supplementary for further details). By combining curve and point operations, CurveCloudNet is an expressive network that maintains the benefits of point cloud backbones while injecting structure and efficiency previously only possible with voxel-based approaches.

4. Experiments

We evaluate CurveCloudNet and a set of competitive baselines on five datasets – the ShapeNet Part Segmentation dataset [8], the KortX Part Segmentation dataset, the Audi Autonomous Driving Dataset (A2D2) [17], the nuScenes dataset [6], and the Semantic KITTI dataset [4, 16]. Each dataset exhibits a unique structure and training setup (see Tab. 1b). Put together, our evaluation consists of indoor, outdoor, object-centric, scene-centric, sparsely scanned,

Method	ShapeNet Per Scan Pattern mIOU (↑) Mean	Parallel	Grid	Random	KortX Per-Category IOU (↑) Validation mIOU	Test mIOU	Cap	Chair	Earphone	Knife	Mug	Rocket	KortX Performance Time (ms)	GPU (GB)	Param (M)
PointNet++ [42]	80.1	81.8 ± 0.1	80.1 ± 0.04	78.3 ± 0.7	66.9 ± 1.2	71.0 ± 2.5	69.3	65.9	83.1	70.4	71.7	65.5	109	1.95	1.53
DGCNN [53]	80.2	81.2 ± 0.2	80.9 ± 0.02	78.6 ± 0.3	64.6 ± 0.8	73.3 ± 0.9	73.0	76.6	81.1	79.6	58.3	71.2	143	1.45	2.18
CurveNet [61]	82.8	**84.0** ± 0.2	**83.8** ± 0.4	80.6 ± 0.2	68.4 ± 0.7	71.5 ± 0.4	80.9	**89.7**	63.6	63.0	68.0	64.0	227	1.16	5.33
PointMLP [36]	80.9	82.2 ± 0.3	81.3 ± 0.3	79.3 ± 0.3	72.1 ± 0.4	75.4 ± 1.3	77.2	75.3	78.8	83.1	64.6	73.6	58	0.83	16.76
PointNext [43]	82.8	83.8 ± 0.1	83.1 ± 0.3	81.5 ± 0.2	69.8 ± 0.6	73.7 ± 0.5	**82.2**	75.9	80.9	68.3	70.8	63.9	81	1.69	13.8
MinkowskiNet [12]	81.1	82.6 ± 0.3	82.0 ± 0.5	80.1 ± 0.5	62.9 ± 1.7	60.1 ± 1.4	67.5	53.2	74.3	56.4	73.6	27.5	42	0.21	36.62
Cylinder3D [81]	79.6	81.2 ± 0.1	80.1 ± 0.0	77.6 ± 0.1	58.6 ± 0.5	63.5 ± 0.2	64.8	56.9	80.8	55.1	64.8	58.7	96	5.76	56.03
SphereFormer [26]	79.5	80.3 ± 0.3	79.9 ± 0.4	78.3 ± 0.4	67.6 ± 0.9	69.7 ± 1.3	71.5	65.1	86.0	59.6	67.2	**79.5**	38	0.25	32.3
CurveCloudNet	**83.1**	83.7 ± 0.2	83.6 ± 0.5	**81.9** ± 0.3	**73.0 ± 0.9**	**78.9** ± 1.1	69.1	87.3	**86.7**	**87.0**	**74.7**	67.8	77	1.01	8.74

Table 2. *Object Segmentation Results.* Class-average mIOU is reported for synthetic ShapeNet dataset (left) and real-world Kortx data (Right). CurveCloudNet achieves the highest accuracy compared to baselines. Performance is on Nvidia RTX 3090 GPU (batch size 16).

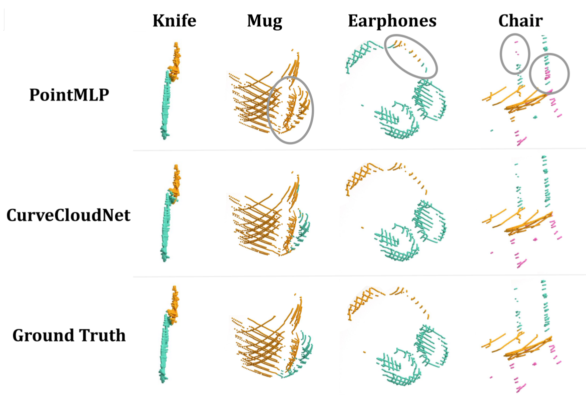

Figure 4. *Qualitative Results on Kortx.* CurveCloudNet successfully segments fine-grained parts by leveraging curve structures.

and densely scanned scenes. Furthermore, each of the five datasets display different point sampling patterns, which we roughly characterize as following “parallel”, “grid”, or “random” laser motions (see Fig. 1).

CurveCloudNet achieves the best or second best performance on *all* datasets, and on average outperforms all previous methods (see Tab. 1a). Furthermore, every other method substantially underperforms CurveCloudNet on at least one dataset. Notably, CurveCloudNet outperforms point-based backbones on object-level tasks and is competitive with or better than voxel-based backbones on larger scenes. In Sec. 4.1 we evaluate our model on the task of object-level part segmentation on simulated ShapeNet [8] objects and on a new dataset collected with the Kortx vision system [1]. In Sec. 4.2, Sec. 4.3, and Sec. 4.4, we evaluate semantic segmentation on larger outdoor scenes using the Audi Autonomous Driving Dataset (A2D2) [17], the nuScenes dataset [6], and the Semantic KITTI dataset [4, 16]. Sec. 4.5 ablates the key components of CurveCloudNet. In the supplementary, we report additional qualitative results and an experiment on object classification.

4.1. Object Part Segmentation

ShapeNet Dataset. The ShapeNet Part Segmentation Benchmark [7, 74] contains 16,881 synthetic shape models across 16 different categories and 50 object parts. To evaluate performance on the laser-based scans that we are interested in, we simulate laser capture using the ShapeNet meshes. Using a fixed front-facing sensor pose, we raycast a set of linear laser traversals and then sample points on the mesh along each traversal. We consider three types of synthetic laser traversals - *parallel*, *grid*, and *random* - which are depicted on the left side of Fig. 1 and are further described in the supplementary. We generate one synthetic scan for each ShapeNet mesh, which yields 12139 training point clouds and 1872 validation point clouds.

ShapeNet Results. ShapeNet results are summarized on the left side of Tab. 2. All methods are trained over three random seeds, and we report the mean and standard deviation of the class-averaged mean intersection-over-union (mIOU) over the runs. All models are trained for 120 epochs using the same hyperparameters, and the best validation mIOU throughout training is reported. On average, CurveCloudNet outperforms all baselines and is best on the “random” laser traversals. In contrast, SphereFormer exhibits the lowest accuracy of all methods, suggesting that its radial window attention is poorly suited for individual objects. CurveNet and PointNext are the runner ups, showing strong performance on segmenting objects when the scans are captured from the front-facing sensor pose.

Kortx Dataset. Kortx is a perception software system developed by Summer Robotics [1] that generates and operates on 3D curves sampled from a triangulated system of event sensors and laser scanners. Kortx software supports arbitrary continuous scan patterns, and in practice we scan objects with a randomly shifted Lissajous trajectory per laser beam. Using Kortx, we scan 7 real-world objects (cap, chair, earphone, knife, mug-1, mug-2, and rocket) multiple

Method	Type	mIoU (↑)	Performance (↓) Time (ms)	GPU (GB)	Param (M)	Per-Class mIoU (↑) car	bicycle	truck	person	road	sidewalk	obstacle	building	nature	pole	sign	signal
PointNet++ [42]	Point	46.5	53	0.19	1.52	62.4	9.3	55.7	3.6	90.3	58.0	12.7	79.3	82.1	19.2	36.6	48.7
RandLANet [21]		43.4	16	0.05	1.24	60.2	4.8	46.7	7.5	91.3	57.2	14.9	78.9	80.0	16.6	27.7	34.3
CurveNet* [61]		4.4	385	1.77	5.52	0.0	0.0	0.0	0.0	25.4	0.0	0.0	0.0	26.9	0.0	0.0	0.0
PointMLP [36]		47.6	63	0.86	16.8	65.8	9.8	54.2	15.8	92.5	63.5	14.8	81.7	82.9	18.8	34.6	36.9
PointNext [43]		45.0	34	0.33	41.6	62.6	2.6	63.1	1.0	91.1	58.7	12.8	80.1	81.8	12.5	33.8	39.3
MinkowskiNet [12]	Voxel	53.8	37	0.19	36.6	70.3	**13.7**	77.6	**26.8**	92.8	67.5	18.0	80.8	81.9	18.0	40.5	58.1
Cylinder3D [81]		53.0	61	1.18	55.8	71.1	11.6	74.8	22.1	92.5	66.1	18.3	82.6	84.2	19.1	41.6	52.0
SphereFormer [26]		**55.1**	54	0.37	32.3	**76.3**	11.5	68.9	26.0	**93.8**	**70.1**	19.6	**84.0**	**86.6**	**19.8**	**46.2**	**58.9**
CurveCloudNet	Curve	54.1	75	0.27	10.2	71.9	12.9	**78.6**	22.3	93.2	68.5	**19.8**	83.3	85.6	17.4	44.3	51.4

Table 3. *A2D2 Segmentation Results.* On grid-like LiDAR scans, CurveCloudNet outperforms all point-based backbones in mIoU and is competitive with SphereFormer. Performance is on an Nvidia RTX 3090 GPU (batch size 1).

times in different poses, collecting 195 point clouds in total. We will release this dataset upon publication. We train on scans that are simulated from ShapeNet meshes and evaluate on the Kortx scans as well as the ShapeNet validation split. To best mimic the Kortx data, we simulate *random* laser traversals on each ShapeNet mesh and only train on the six object categories present in the Kortx dataset: *cap, chair, earphone, knife, mug*. We generate five training scans per ShapeNet object, each scanned from a unique *random* sensor pose. This yields a training set of 31,991 point clouds.

KortX Results. KortX results are summarized on the right side of Tab. 2. The experimental setup is identical to ShapeNet, except we train over four random seeds and for 60 epochs. CurveCloudNet again outperforms all baselines, showing effective generalization to out-of-domain Kortx test scans. Voxel-based methods continue to underperform their point-based counterparts, suggesting that discretizing the input has a negative effect when point clouds are small. In contrast to the previous ShapeNet evaluation, PointMLP is the second-best method when scans are captured from random sensor poses. Fig. 4 shows that CurveCloudNet better distinguishes fine-grained structures, such as the back of a chair, a mug handle, and an earphone headpiece.

4.2. A2D2 LiDAR Segmentation

A2D2 Dataset. The Audi Autonomous Driving Dataset (A2D2) [17] contains 41,280 frames of outdoor driving scenes captured from 5 overlapping LiDAR sensors, creating a unique grid-like scanning pattern (see Fig. 2). We evaluate on 12 LiDAR categories: *car, bicycle, truck, person, road, sidewalk, obstacle, building, nature, pole, sign,* and *traffic signal*. Evaluation is performed on annotated LiDAR points in the field of view of the front-facing camera.

Results. We train CurveCloudNet and baselines on the official A2D2 training split [17]. For fair comparison, all models are trained for 140 epochs using the same hyperparameters, and the best validation mIOU throughout training is reported. Results are summarized in Tab. 3, and Fig. 1 provides a qualitative example. CurveCloudNet scales to outdoor scenes better than point-based backbones, with the runner-up PointMLP showing a 6% drop in mIOU. CurveCloudNet also outperforms most voxel-based backbones and achieves similar accuracy to state-of-the-art SphereFormer [26], even though SphereFormer's radial window attention is tailored for outdoor LiDAR scans.

4.3. nuScenes LiDAR Segmentation

nuScenes Dataset. The nuScenes dataset [6] contains 1000 sequences of driving data, each 20 seconds long. Each sequence contains 32-beam LiDAR data with segmentations annotated at 2Hz. We follow the official nuScenes benchmark protocol with 16 semantic categories.

Results. We train CurveCloudNet and baselines on the official nuScenes training split. To ensure fair comparison, we train all models for 100 epochs. Results on the nuScenes validation split are shown in Tab. 4, and Fig. 1 includes a qualitative example. CurveCloudNet significantly improves upon other point-based networks: PointMLP and PointNext show more than a 10% drop in mIOU and $\sim 2\times$ increase in latency. We also note that CurveNet exceeds 48GB of GPU memory for a batch size of 1, showcasing its inability to scale to larger scenes. CurveCloudNet also outperforms all voxel-based methods except the recent SphereFormer.

4.4. KITTI LiDAR Segmentation

The Semantic KITTI dataset is made up of 22 sequences of driving data consisting of 23,201 LiDAR scans for training and 20,351 for testing. Each scan is obtained with a dense 64-beam Velodyne LiDAR. We follow the official KITTI protocol in training and validation. To ensure fair comparison, we train all models for 100 epochs. Results on the validation sequence are reported in Tab. 5, and Fig. 1 shows a qualitative example. CurveCloudNet outperforms all point-based and voxel-based methods. Note that we cannot re-

Method	Type	mIoU (↑)	Performance (↓) Time (ms)	GPU (GB)	Param (M)	Per-Class mIoU (↑) barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	manmade	vegetation
PointNet++ [42]	Point	51.1	274	0.60	1.5	60.1	6.5	58.4	66.3	16.4	20.0	50.8	12.6	31.5	42.0	94.0	60.8	63.8	69.2	82.4	82.3
RandLANet [21]		62.9	21	0.18	1.2	72.5	12.6	36.6	81.8	38.7	72.3	68.5	37.3	44.7	59.7	95.3	87.0	69.7	71.1	73.2	85.9
CurveNet [61]		–	–	>48	5.5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PointMLP [36]		67.9	164	4.94	16.8	72.3	27.8	88.2	86.3	37.2	51.0	60.7	50.6	56.4	71.1	95.7	70.6	70.9	72.0	88.8	87.2
PointNext [43]		65.0	155	0.62	41.5	68.7	1.2	86.9	87.5	41.8	57.4	54.3	34.9	55.3	75.1	95.7	68.9	70.2	71.5	86.5	84.0
MinkowskiNet [12]	Voxel	76.2	44	0.29	36.6	75.4	43.9	91.9	93.0	49.0	84.3	78.3	64.6	65.9	85.7	96.1	71.5	67.5	74.8	86.5	84.9
PolarNet* [77]		71.0	-	-	-	74.7	28.2	85.3	90.9	35.1	77.5	71.3	58.8	57.4	76.1	96.5	71.1	74.7	74.0	87.3	85.7
Cylinder3D* [81]		76.1	80	1.57	55.9	76.4	40.3	91.2	93.8	51.3	78.0	78.9	64.9	62.1	84.4	96.8	71.6	76.4	75.4	90.5	87.4
SphereFormer* [26]		**79.5**	59	0.81	32.3	78.7	46.7	95.2	93.7	54.0	88.9	81.1	68.0	74.2	86.2	97.2	74.3	76.3	75.8	91.4	89.7
CurveCloudNet	Curve	<u>78.0</u>	87	1.14	28.8	77.3	45.7	92.4	91.9	59.4	84.5	78.5	64.1	69.6	85.0	96.9	72.7	75.6	75.2	90.5	89.0

Table 4. *nuScenes Segmentation Results.* On typical sweeping LiDAR scans, CurveCloudNet scales significantly better than other point-based backbones and is competitive with recent work SphereFormer. Performance is on an Nvidia RTX 3090 GPU (batch size 1). * indicates that results are copied from the referenced papers.

Method	Type	mIoU (↑)	Performance (↓) Time (ms)	GPU (GB)	Param (M)
PointNet++ [42]	Point	–	2690	11.1	1.6
CurveNet [61]		–	–	>48	5.5
PointMLP [36]		39.5	293	5.24	16.8
PointNext [43]		–	1303	1.83	41.6
MinkowskiNet [12]	Voxel	66.8	111	0.53	36.6
Cylinder3D* [81]		68.9	233	1.62	55.9
SphereFormer* [26]		69.0	144	3.46	32.3
CurveCloudNet	Curve	**69.5**	155	2.75	28.8

Table 5. Quantitative results on the KITTI validation split. Performance is on an Nvidia RTX 3090 GPU (batch size 1). * indicates results are copied from the referenced papers.

Curve Operations Grouping	FPS	1D Conv.	mIoU (↑)	Performance (↓) Time (ms)	GPU (GB)	Param (M)
✓	✓	✓	**54.1**	75	0.27	10.3
✗	✓	✓	53.3	99	1.03	10.3
✓	✗	✓	52.4	105	0.26	10.3
✓	✓	✗	52.0	**61**	**0.20**	9.9
✗	✗	✗	52.6	122	0.92	10.3

Table 6. *Ablation Study on A2D2.* Curve operations are ablated and replaced with the standard point-based counterparts.

port results for many point-based methods due to excessive training times on the larger KITTI scans (> 20 days).

4.5. Ablation Study

Tab. 6 shows an ablation analysis of CurveCloudNet on the A2D2 dataset; the table shows that each of our proposed curve operations is essential to achieve high accuracy and efficiency. We ablate grouping along curves by instead using the regular radial groupings from PointNet++ [42]. This ignores the curve structure and results in decreased accuracy, increased latency, and a significant increase in GPU memory usage. Instead of curve farthest point sampling, we also try regular FPS, which causes a decreased accuracy and increased latency. Finally, without 1D curve convolutions, we observe a notable decline in accuracy with a marginal improvement in latency and GPU memory. Taken together, our curve operations increase accuracy with roughly half the latency and one third the GPU memory requirements.

5. Discussion and Limitations

We have described a point cloud processing scheme and backbone, CurveCloudNet, which introduces curve-level operations to achieve accurate, efficient, and flexible performance on point cloud segmentation. CurveCloudNet outperforms or is competitive with previous methods on the ShapeNet, Kortx, A2D2, nuScenes, and KITTI datasets, and on average achieves the best performance. Put together, CurveCloudNet is a unified solution to *both* small and large-scale scenes with various scanning patterns. Nevertheless, CurveCloudNet is only designed for laser-scanned data, i.e. point clouds with *explicit* curve structure due to 1D laser traversals – we further discuss this limitation in the supplement. We believe a promising future direction is to investigate *virtual curves* that can extend CurveCloudNet to uniformly sampled point clouds. Another exciting future direction will be to investigate additional curve operations such as explicit curve-to-curve communication, curve self-attention and cross-attention, and curve intersections.

Acknowledgements. This work was supported by grants from the Toyota Research Institute (TRI) University 2.0 program and a Vannevar Bush Faculty Fellowship. Toyota Research Institute ("TRI") provided funds to assist the research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We thank Summer Robotics for their aid and support with using the KortX software system. Despoina Paschalidou is supported by the Swiss National Science Foundation. Davis Rempe was partially supported by NVIDIA.

References

[1] Summer robotics. https://www.summerrobotics.ai/. Accessed: 2023-02-15. 2, 6

[2] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. Rangevit: Towards vision transformers for 3d semantic segmentation in autonomous driving. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5240–5250, 2023. 3

[3] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Trans. on Graphics*, 2018. 2

[4] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 2, 5, 6

[5] Alexandre Boulch, Gilles Puy, and Renaud Marlet. Fkaconv: Feature-kernel alignment for point cloud convolution. 2020. 2

[6] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. 2020. 2, 5, 6, 7

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 6

[8] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. 2, 5, 6

[9] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[10] Huimin Cheng, Xian-Feng Han, and Guoqiang Xiao. Cenet: Toward concise and efficient lidar semantic segmentation for autonomous driving. *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 01–06, 2022. 3

[11] Ran Cheng, Ryan Razani, Ehsan Moeen Taghavi, Enxu Li, and Bingbing Liu. (af)2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12542–12551, 2021. 3

[12] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 6, 7, 8

[13] Moshe Eliasof and Eran Treister. Diffgcn: Graph convolutional networks via differential operators and algebraic multigrid pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[14] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical cnns. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1, 2

[15] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 5, 6

[17] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: audi autonomous driving dataset. *arXiv.org*, 2020. 2, 5, 6, 7

[18] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3

[19] Yinjuan Gu, Yuming Huang, Chengzhong Xu, and Hui Kong. Maskrange: A mask-classification model for range-view based lidar segmentation. *ArXiv*, abs/2206.12073, 2022. 3

[20] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for lidar semantic segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8469–8478, 2022. 3

[21] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 5, 7, 8

[22] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. pages 448–456, 2015. 5

[24] Artem Komarichev, Zichun Zhong, and Jing Hua. A-CNN: annularly convolutional neural networks on point clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[25] Lingdong Kong, You-Chen Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Y. Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. *ArXiv*, abs/2303.05367, 2023. 3

[26] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *CVPR*, 2023. 2, 3, 6, 7, 8

[27] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S Davis. Modeling local geometric structure of 3d point clouds using geocnn. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[28] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, 2018. 3

[29] Huan Lei, Naveed Akhtar, and Ajmal Mian. Octree guided CNN with spherical kernels for 3d point clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[30] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[31] Jinxian Liu, Bingbing Ni, Caiyuan Li, Jiancheng Yang, and Qi Tian. Dynamic points agglomeration for hierarchical point sets learning. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

[32] 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*, 2019. 3

[33] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[34] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel CNN for efficient 3d deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 3

[35] Zhijian Liu, Haotian Tang, Shengyu Zhao, Kevin Shao, and Song Han. Pvnas: 3d neural architecture search with point-voxel convolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:8552–8568, 2021. 3, 5

[36] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022. 2, 5, 6, 7, 8

[37] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 2

[38] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2015. 3

[39] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 3

[40] C. Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, 2016. 3

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[42] Charles R 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 (NIPS)*, 2017. 1, 2, 4, 5, 6, 7, 8

[43] Gordon Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *ArXiv*, abs/2206.04670, 2022. 2, 5, 6, 7, 8

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 5

[45] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[46] 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3

[47] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. RGCNN: regularized graph CNN for point cloud segmentation. In *ACM Trans. on Graphics*, 2018. 2

[48] 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 *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017. 3

[50] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[51] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[52] Yue Wang and Justin M. Solomon. Object DGCNN: 3d object detection using dynamic graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1

[53] 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 Trans. on Graphics*, 2019. 1, 2, 3, 5, 6

[54] Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. Deltaconv: anisotropic operators for geometric deep learning on point clouds. *ACM Trans. on Graphics*, 2022. 2, 5

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

[56] Hai Wu, Jinhao Deng, Chenglu Wen, Xin Li, Cheng Wang, and Jonathan Li. Casa: A cascade attention network for 3-d object detection from lidar point clouds. *IEEE Trans. Geosci. Remote. Sens.*, 2022. 1

[57] Hai Wu, Chenglu Wen, Wei Li, Xin Li, Ruigang Yang, and Cheng Wang. Transformation-equivariant 3d object detection for autonomous driving. *arXiv.org*, 2022. 1

[58] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3

[59] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022. 3

[60] Xiaopei Wu, Liang Peng, Honghui Yang, Liang Xie, Chenxi Huang, Chengqi Deng, Haifeng Liu, and Deng Cai. Sparse fuse dense: Towards high quality 3d detection with depth completion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[61] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 3, 6, 7, 8

[62] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[63] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 3

[64] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 3

[65] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16004–16013, 2021. 3

[66] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. 2021. 2

[67] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1, 2

[68] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. *ArXiv*, abs/2012.03762, 2020. 3

[69] Xu Yan, Jiantao Gao, Chaoda Zheng, Chaoda Zheng, Ruimao Zhang, Shenghui Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *European Conference on Computer Vision*, 2022. 3

[70] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 3

[71] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022. 1

[72] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[73] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11037–11045, 2020. 2

[74] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 2016. 2, 6

[75] Cheng Zhang, Haocheng Wan, Xinyi Shen, and Zizhao Wu. Pvt: Point-voxel transformer for point cloud learning. *arXiv.org*, 2022. 1, 3

[76] Wenxiao Zhang and Chunxia Xiao. PCAN: 3d attention map learning using contextual information for point cloud based retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[77] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 8

[78] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[79] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 3

[80] Yiming Zhao, Lin Bai, and Xinming Huang. Fidnet: Lidar point cloud semantic segmentation with fully interpolation decoding. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4453–4458, 2021. 3

[81] Hui Zhou, Xinge Zhu, Xiao Song, Yuexin Ma, Zhe Wang, Hongsheng Li, and Dahua Lin. Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. *arXiv.org*, 2020. 1, 2, 3, 6, 7, 8