
ECCV
TEL AVIV 2022

TEL AVIV 2022

Efficient Point Cloud Analysis Using Hilbert Curve

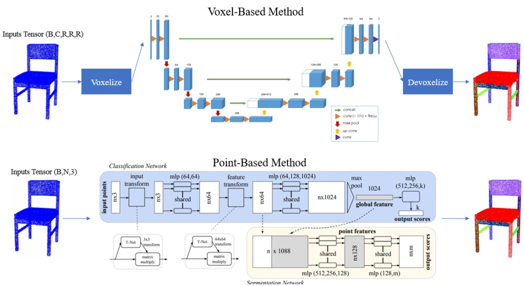
Wanli Chen, Xingge Zhu, Guojin Chen, Bei Yu

The Chinese University of Hong Kong
{wlchen,byu}@cse.cuhk.edu.hk

May. 27, 2022

Background and Motivation

Previous Works of Point Cloud Analysis



Point-Based Methods vs. Voxel Based Methods

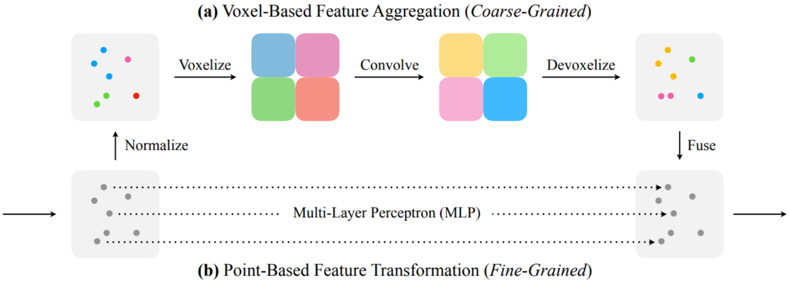
Voxel-Based methods

- Advantages: Good data locality and regularity
- Disadvantages: Large memory footprint

Point-Based methods

- Advantages: Small memory footprint
- Disadvantages: Irregular memory access and bad spatial locality

Previous Works of Point Cloud Analysis



A classical point + voxel framework

Voxel + Point methods

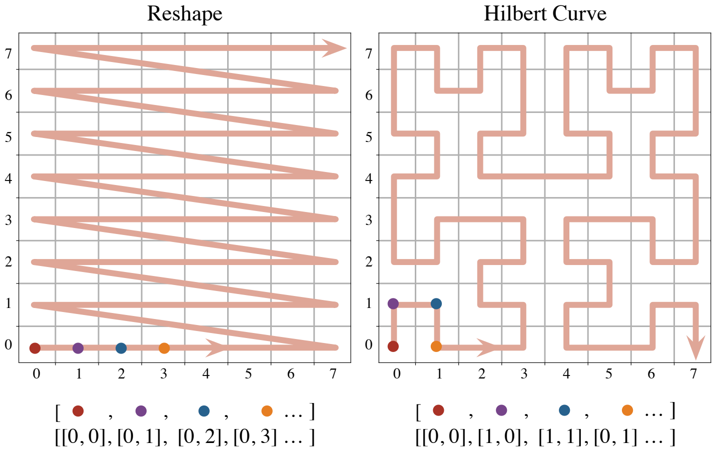
- Advantages: Taking the advantages of Voxel-based and Point-based method.
- Disadvantages: Not very effective due to the large cost of voxel branch.

Motivation

Can we use 2D convolution to handle 3D voxel data?

- Lower computational overhead than 3D convolution.
- 2D convolution has many useful techniques to increase accuracy such as transformer.

We should mapping 3D data into 2D space.



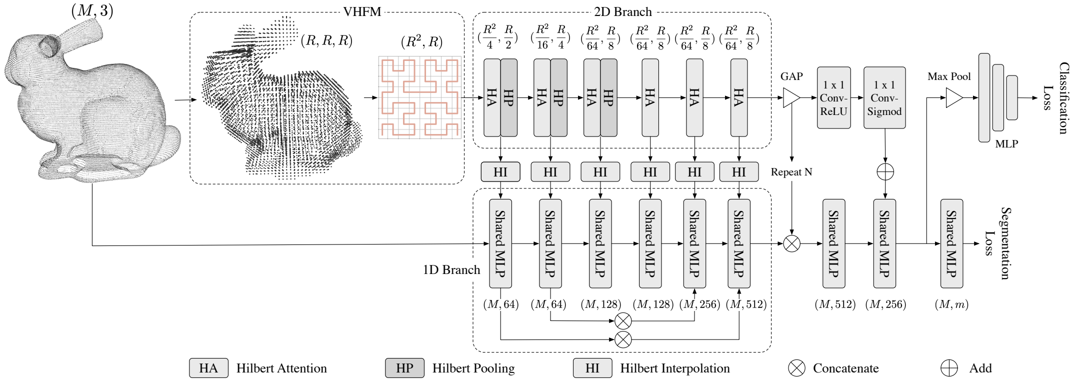
Left: The mapping scheme of Reshape function. Right: The mapping scheme of Hilbert curve. Hilbert curve has better locality because it has no "**jump connections**" like reshape function.

Hilbert Curve

Advantages of Hilbert curve:

- No jump connection, which leads to better locality
- Lower space-to-linear ratio
- Better clustering property

Efficient Point Cloud Analysis Using Hilbert Curve



The main framework of our model.

Given a 3D feature $\mathcal{V}^{\in(C,R,R,R)}$ with channel size C, we separate it into R slices along Z axis.
Then, 2D n-th order Hilbert curve $\mathcal{H}_n(s)$ is used to encode each slice (as shown in Equation (1)).

$$\mathcal{V}^{\in R\times R\times R} \rightarrow \begin{bmatrix} \mathcal{V}_{s1} \\ \mathcal{V}_{s2} \\ \vdots \\ \mathcal{V}_{sR} \end{bmatrix} \xrightarrow{\mathcal{H}_n(s)} \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ \vdots \\ \boldsymbol{s}_R \end{bmatrix} = \mathcal{I}. \tag{1}$$

The sequences $\boldsymbol{s}_1, \boldsymbol{s}_2 ... \boldsymbol{s}_R \in (C, R^2)$ and $\mathcal{H}_n(\boldsymbol{s}_k) = \mathcal{V}_{sk}, k = 1, 2...R$.

Given a 3D feature $\mathcal{V}^{\in(C,R,R,R)}$, the traditional Trilinear interpolation is performed as:

$$\boldsymbol{O} = \text{Reshape}(\mathcal{V}) * \boldsymbol{F}_{linear}, \tag{2}$$

- The addition of empty grids with non-empty grids will weaken the output non-empty part of feature.
- The *Reshape*$(\cdot)$ function is not locality preserving.

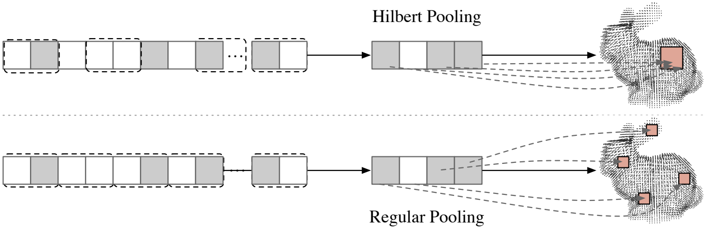
Hilbert Interpolation

Given a 2D feature $\mathcal{I}^{\in(C,R^2,R)}$ that flattened by 3D feature and the target point cloud feature $\boldsymbol{O}^{\in(M,C)}$, The proposed Hilbert interpolation $\mathfrak{L}(\cdot)$ is performed as follow:

$$\begin{aligned} \boldsymbol{O} &= \mathfrak{L}(\mathcal{I}), \text{where} \\ \mathcal{H}_{\lceil M \rceil}(\boldsymbol{O}) &= \begin{cases} (\mathcal{I} \cdot \mathcal{W}_h) * \boldsymbol{F}_{linear}, & M \leq R^3; \\ \mathcal{I} * \boldsymbol{F}_{linear}, & M > R^3. \end{cases} \end{aligned} \tag{3}$$

Here $\lceil M \rceil$ represents the closest curve order that the corresponding Hilbert curve has at least M points. Then first binarize featuremap $\mathcal{I}$ along channel C, obtaining $\mathcal{I}_B$ and apply sum filter to $\mathcal{I}_B$, obtaining $\mathcal{W}_B$:

$$\mathcal{W}_B = \mathcal{I}_B * \boldsymbol{F}_{sum}. \tag{4}$$



Hilbert pooling and regular max pooling

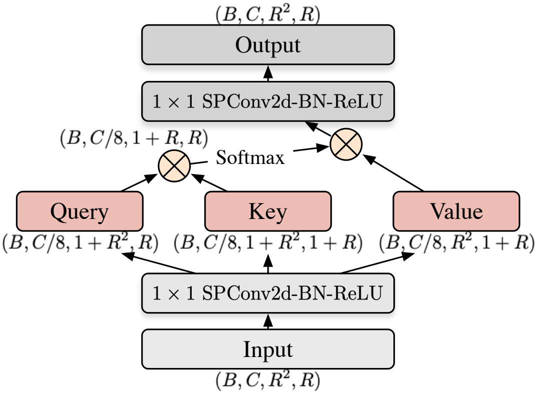
Due to the specialty of $\mathcal{I}$, we design a novel pooling technique to harvest spatial information named Hilbert pooling $\mathfrak{P}(\cdot)$, specifically:

$$\text{MaxPool3D}(\mathcal{H}_n^{-1}(\mathcal{I})) \xrightarrow{\mathcal{H}_{n-1}(s)} = \mathcal{I}' = \mathfrak{P}(\mathcal{I}), \tag{5}$$

where $\mathcal{H}_n^{-1}(\mathcal{I})$ is the inverse operation of Equation (1), which transform 2D feature into 3D.

Hilbert Attention

In order to get the richer spatial feature in 2D branch, we introduce Self-attention, a powerful tool for global feature collection. Our proposed Hilbert attention includes:



Hilbert attention.

Hilbert Attention

1. **Intra-Slice Correlation (Key)**. VHFM module transforms each slice $\mathcal{V}_{sk}$ into sequence $\boldsymbol{s}_k$, $k \in [1, R]$. Then, a pointwise linear projection $\sigma(\cdot)$ with weight w_{key}:

$$\sigma(\mathcal{I}) = \sum_{e_k \in \boldsymbol{s}_k} w_{key} e_k \tag{6}$$

   is applied along $\boldsymbol{s}_k$ for intra-slice level feature extraction, which collects pointwise feature **along** Hilbert curve.

2. **Inter-Slice Corelation (Query)**. To collect pointwise features between $\boldsymbol{s}_k$, we introduce inter-slice correlation. Specifically, the linear projection $\phi(\cdot)$ is used:

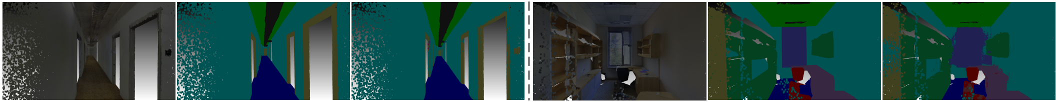
$$\phi(\mathcal{I}) = \sigma(\mathcal{I}^\top), \tag{7}$$

   where $\phi(\cdot)$ collects pointwise feature **across** Hilbert curve.

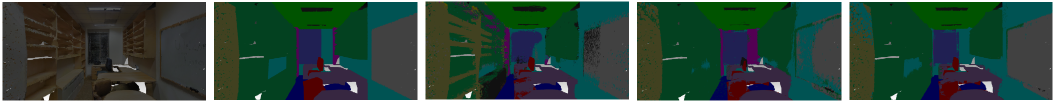
3. **Mixed Correlation.**. Acts as a 4×4 convolution. Finally, Hilbert attention is gathered by considering the importance between inter-slice and intra-slice feature: $HA = \text{Softmax}(\phi(\mathcal{I})\sigma(\mathcal{I}))\gamma(\mathcal{I})$.

Experimental Results

Visualization



(a) Left to right: Point Cloud, GT, HilbertNet.



(b) Left to right: Point Cloud, GT, PointNet, KPConv, HilbertNet.

(a) Visualized results on S3DIS Area 5 dataset; (b) Quantitative comparison.

Table: Results of S3DIS Area 5

	PointNet	PointCNN	PCCN	MinkowskiNet	KPConv	PointTransformer	HilbertNet
ceiling	88.8	92.3	92.3	91.8	92.8	94	**94.6**
floor	97.3	98.2	96.2	**98.7**	97.3	98.5	97.8
wall	69.8	79.4	75.9	86.2	82.4	86.3	**88.9**
beam	0.1	**0.3**	0	0	0	0	0
column	3.9	17.6	6	34.1	23.9	**38**	37.6
window	46.3	22.8	**69.5**	48.9	58	63.4	64.1
door	10.8	62.1	63.5	62.4	69	**74.3**	73.8
table	59	74.4	66.9	81.6	81.5	**89.1**	88.4
chair	52.6	80.6	65.6	89.8	**91**	82.4	85.4
sofa	5.9	31.7	47.3	47.2	**75.4**	74.3	73.5
bookcase	40.3	66.7	68.9	74.9	75.3	80.2	**82.7**
board	26.4	62.1	59.1	74.4	66.7	**76**	74.7
clutter	33.2	56.7	46.2	58.6	58.9	59.3	**60.1**
mIoU	41.1	57.3	58.3	65.4	67.1	70.4	**70.9**

Results on ShapeNetPart and ModelNet40 datasets

Table: Results on ModelNet40 & ShapeNetPart datasets

ModelNet40		ShapeNetPart	
Method	Acc	Method	mIoU
VoxNet	85.9	Kd-Net	82.3
Subvolume	89.2	PointNet	83.7
PointNet	89.2	SO-Net	84.9
DGCNN	92.9	3D-GCN	85.1
PointASNL	92.9	DGCNN	85.2
Grid-GCN	93.1	PointCNN	86.1
PCT	93.2	PVCNN	86.2
SO-Net	93.4	KPConv	86.4
CurveNet	93.8	CurveNet	86.6
Ours	**94.1**	Ours	**87.1**

Computational Cost

Table: Comparison of methods

Method	voxel size	Inference time	mIoU
3D-UNet	64^3	347ms	84.2
PVCNN	32^3	62.5ms	86.0
HilbertNet-L	64^3	**42.1ms**	85.8
HilbertNet-M	64^3	59.2ms	86.4
HilbertNet	64^3	91.6ms	**87.1**

Here we use ShapeNetPart as benchmark. We propose HilbertNet-M (median) and HilbertNet-L(light) during the experiment. HilbertNet-M has $0.5 \times C$ and HilbertNet-L has $0.25 \times C$, where C is the channel number of the features in HilbertNet.

Computational Cost

Table: Computational cost and GPU Memory of different methods. The tested voxel resolution is 32^3. (FLOPs: floating point operations)

Method	FLOPs	GPU Memory
3D Convolution	18.86G	162M
2D Convolution	4.45G	148.7M
Sparse 2D Convolution	1.47G	49.6M
NonLocal	0.34G	4G
Hilbert Attention	**0.32G**	**47.8M**

THANK YOU!