JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

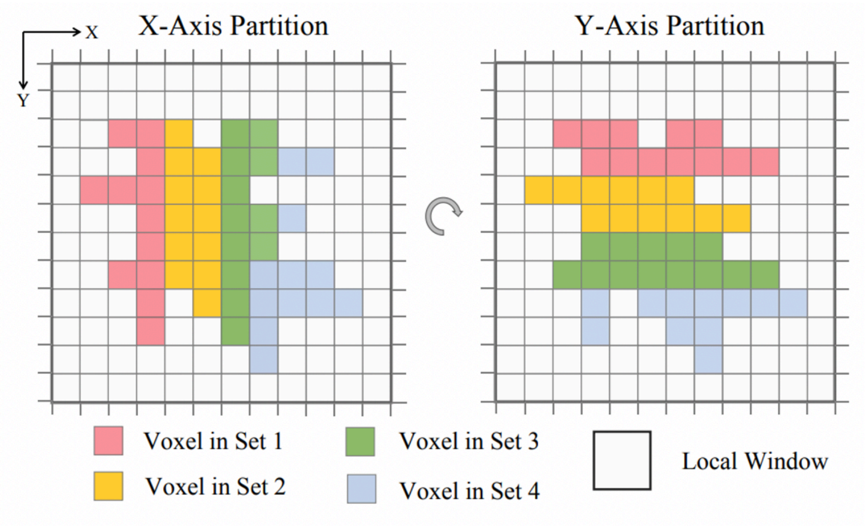
北京大学
PEKING UNIVERSITY

DSVT: Dynamic Sparse Voxel Transformer with Rotated Sets

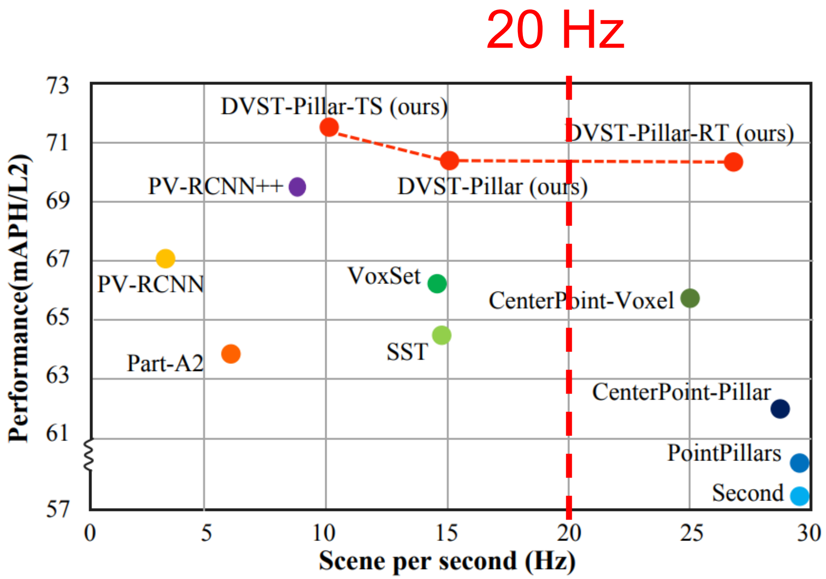
Haiyang Wang[1*] Chen Shi[1*] Shaoshuai Shi[2†] Meng Lei[1]
Seng Wang[3] Di He[1] Bernt Schiele[2] Liwei Wang[1†]

[1]Peking University [2]Max Planck Institute for Informatics [3]Huawei

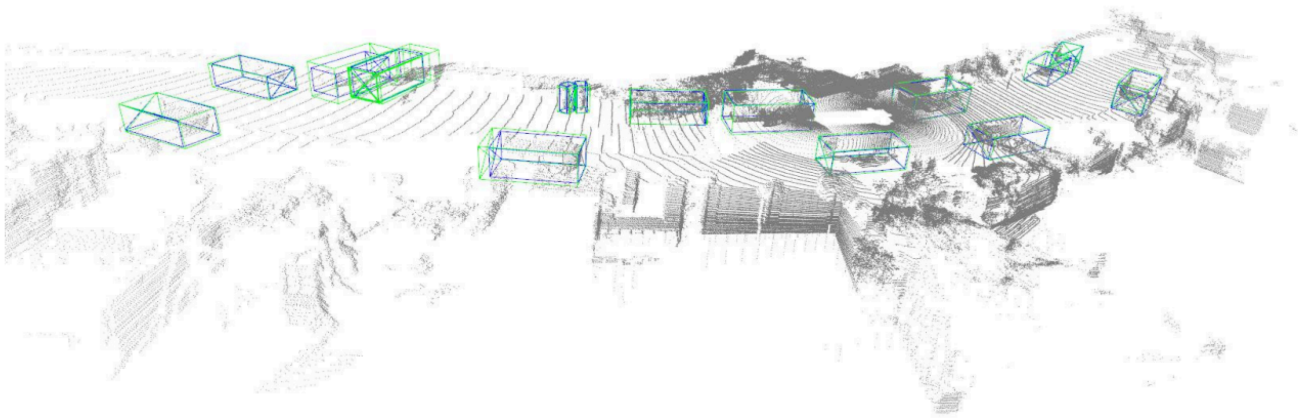
Overview



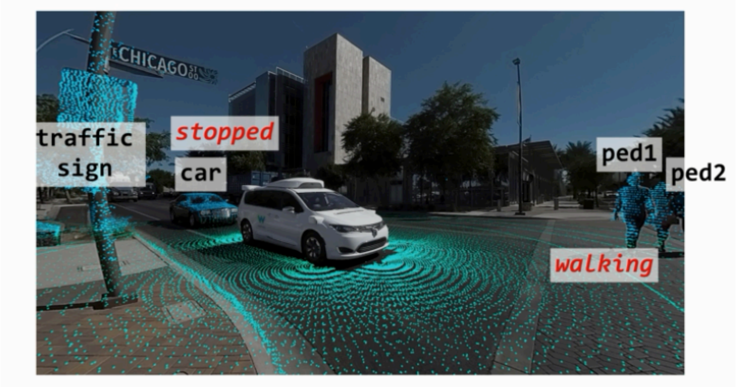
X-Axis Partition
Y-Axis Partition
X
Y
Voxel in Set 1
Voxel in Set 2
Voxel in Set 3
Voxel in Set 4
Local Window



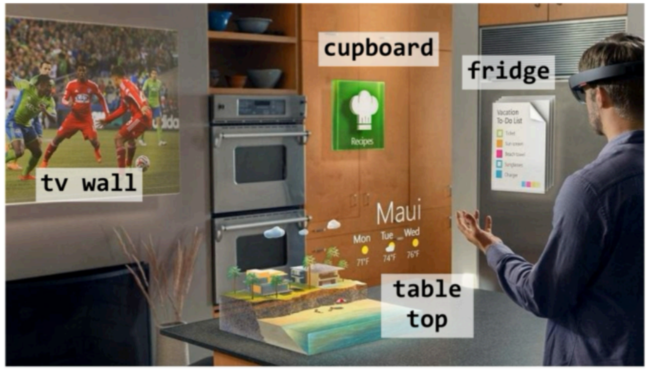
20 Hz
DVST-Pillar-TS (ours)
DVST-Pillar-RT (ours)
PV-RCNN++
DVST-Pillar (ours)
PV-RCNN
VoxSet
CenterPoint-Voxel
SST
Part-A2
CenterPoint-Pillar
PointPillars
Second
Performance(mAPH/L2)
Scene per second (Hz)
73
71
69
67
65
63
61
57
0
5
10
15
20
25
30



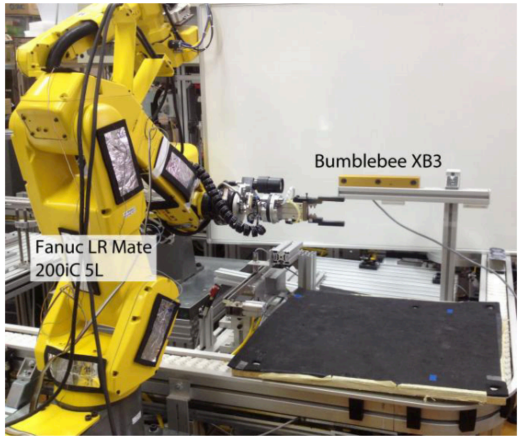
Emerging 3D Applications



Self-Driving Cars



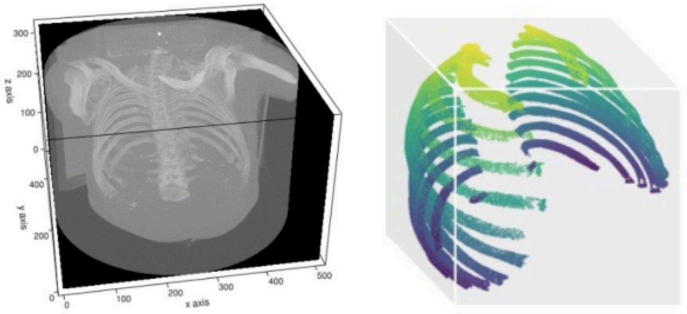
Augmented Reality



Robot

Existing RibFrac CT Scans

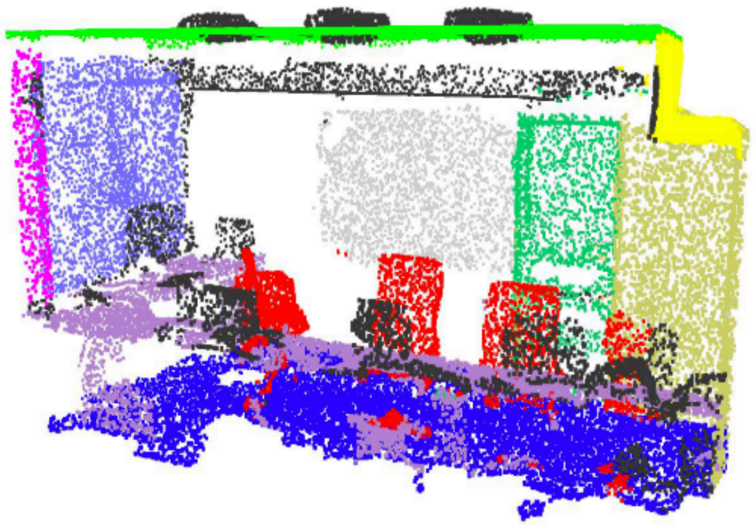
New Annotations Rib Segmentation

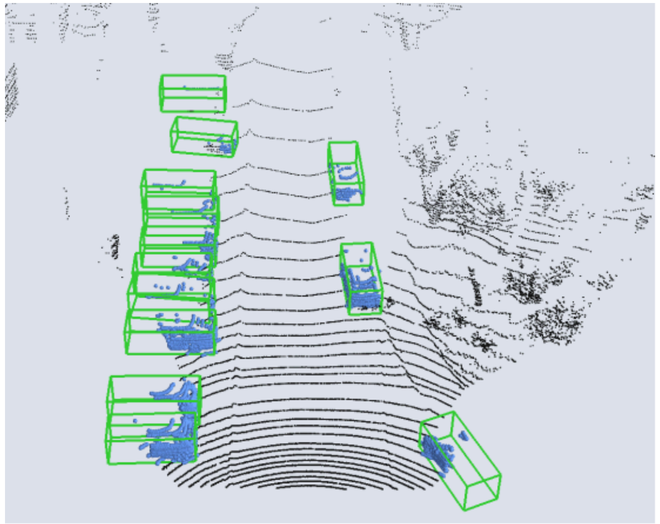


Medical Image

Data Format of 3D

- PointCloud
  - A suitable data format to 3D Scene Understanding
  - Close to original sensor and is directly after the lidar scan
  - Point Cloud is simple, just a point set
- Characteristic
  - Sparse
  - Irregular
  - Unorder



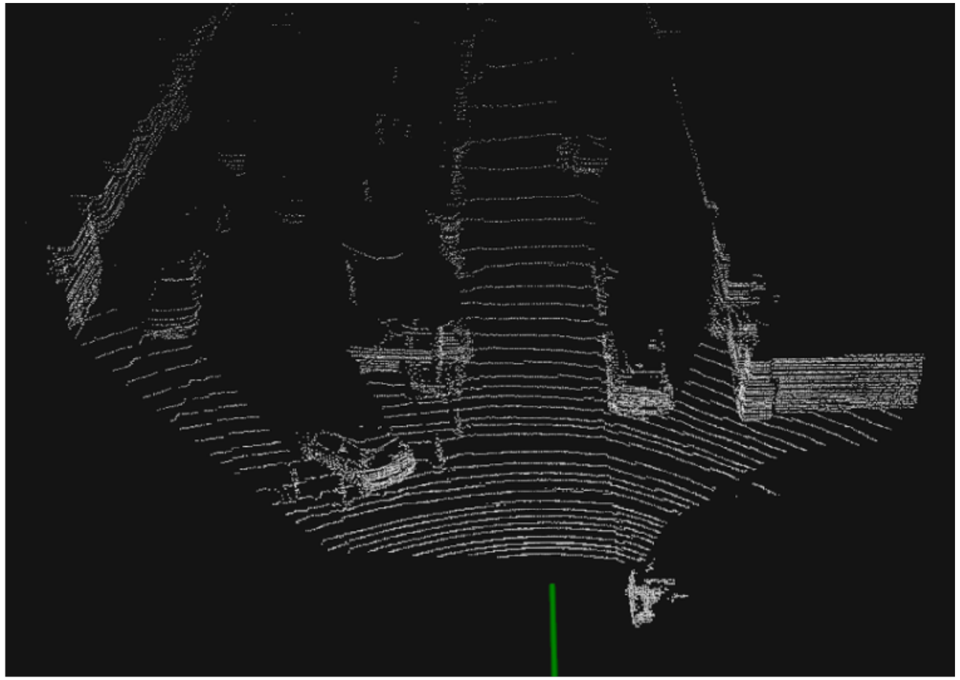


Point Cloud Processor

- Point cloud has sparse and irregular data format, which can not be processed with existing convolutional neural network

Camera Sensor

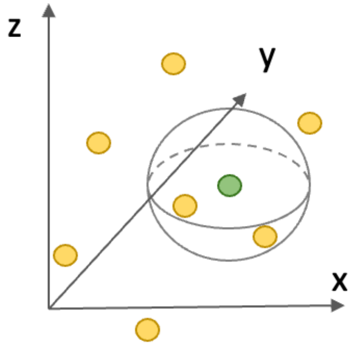
V.S

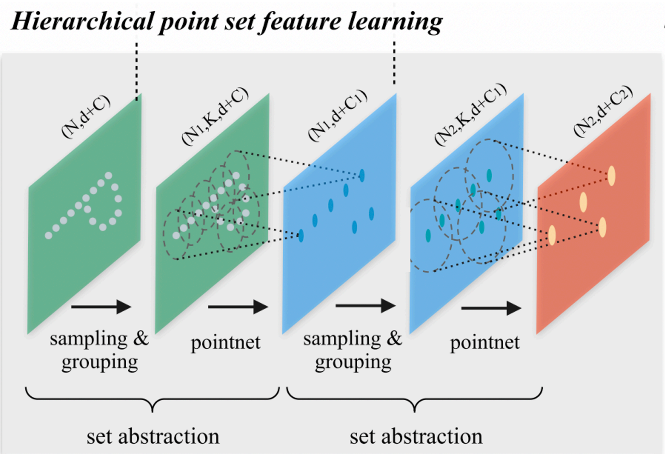


Lidar Sensor

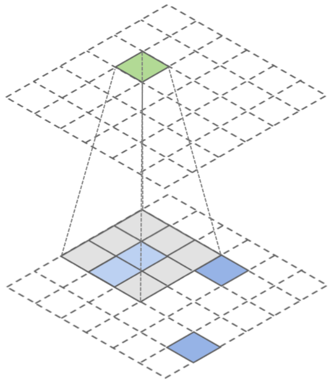
DSVT

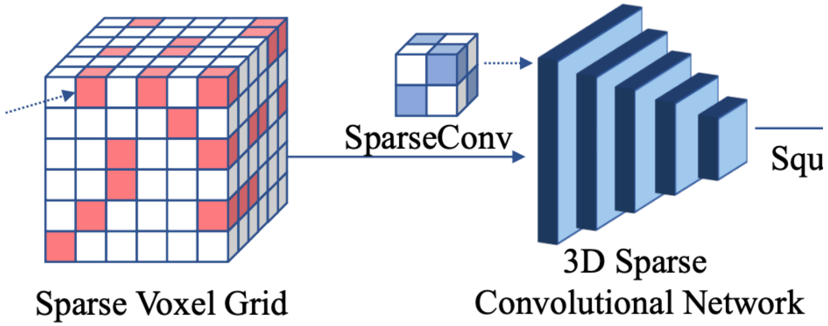
- Point cloud has sparse and irregular data format, which can not be processed with existing convolutional neural network.
- Point Cloud Processor:
  - PointNet, PointNet++





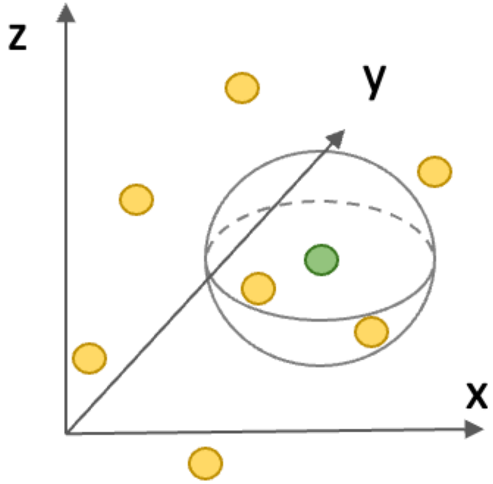
  - Sparse Convolution (Conventional and SubManifold)



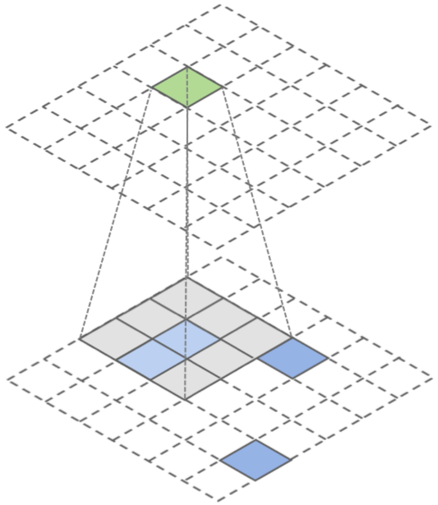


DSVT

- The intensive computation of sampling and grouping.
- The limited representation capacity due to submanifold dilation.
- Can not be implemented with well-optimized deep learning tools (*TensorFlow* or *PyTorch*) and require writing customized CUDA codes, which needs to be heavily optimized before deployment.



Point-based Local Feature Extractor (PointNet++)

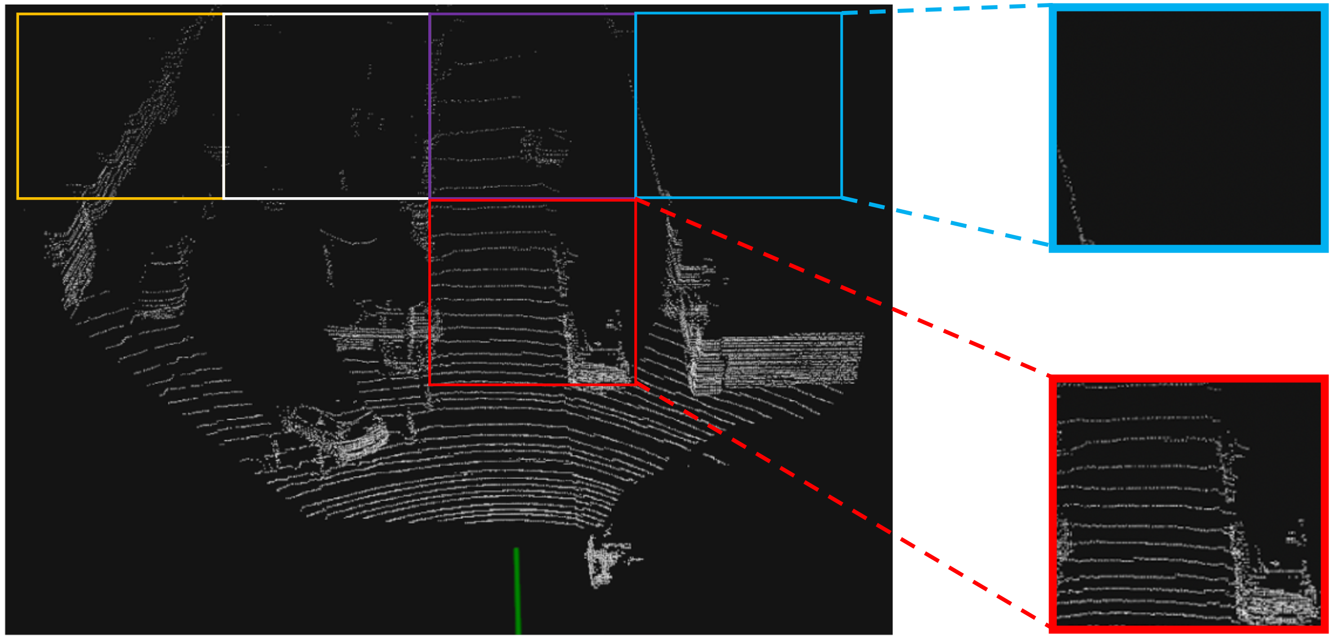


Sparse Convolution

Transformer on sparse point clouds?

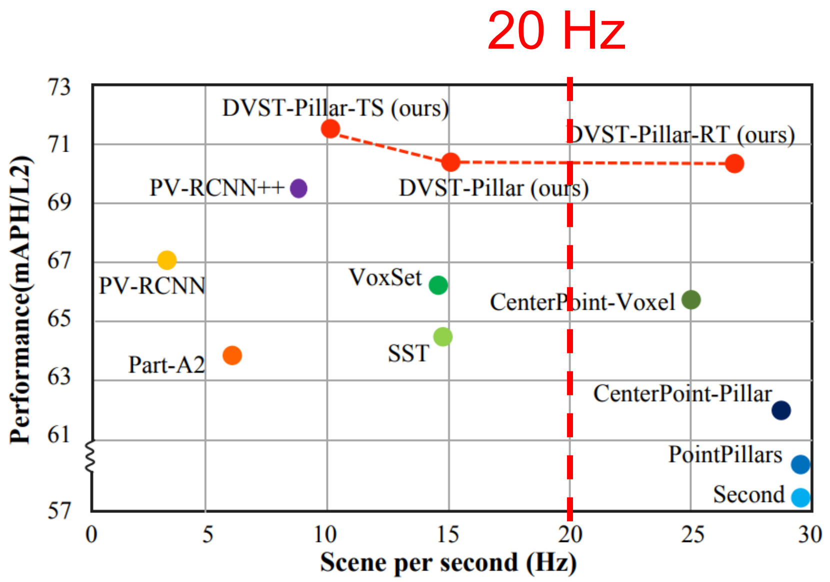
- Transformer is naturally suitable to sparse data.
- How to apply a standard Transformer is nontrivial.
  - Global Attention: can not be applied to process the large-scale point clouds (~60000 voxels).
  - Window Attention: due to the sparsity of point clouds, the number of non-empty voxels in each local window varies significantly, which can not be computed in a fully parallel manner.

Different windows have different number of points, which can not be calculated in a fully parallel manner



Main Contributions

- We propose *Dynamic Sparse Window Attention*, a window-based attention strategy for handling sparse 3D voxels in parallel.
- Based on the above key design, we introduce an efficient yet deployment-friendly transformer 3D backbone without any customized CUDA operations. It can be easily accelerated by NVIDIA TensorRT to achieve real-time inference speed (27Hz).
- Our approach outperforms previous state-of-the-art methods on the large-scale Waymo Open Dataset with a remarkable gain.

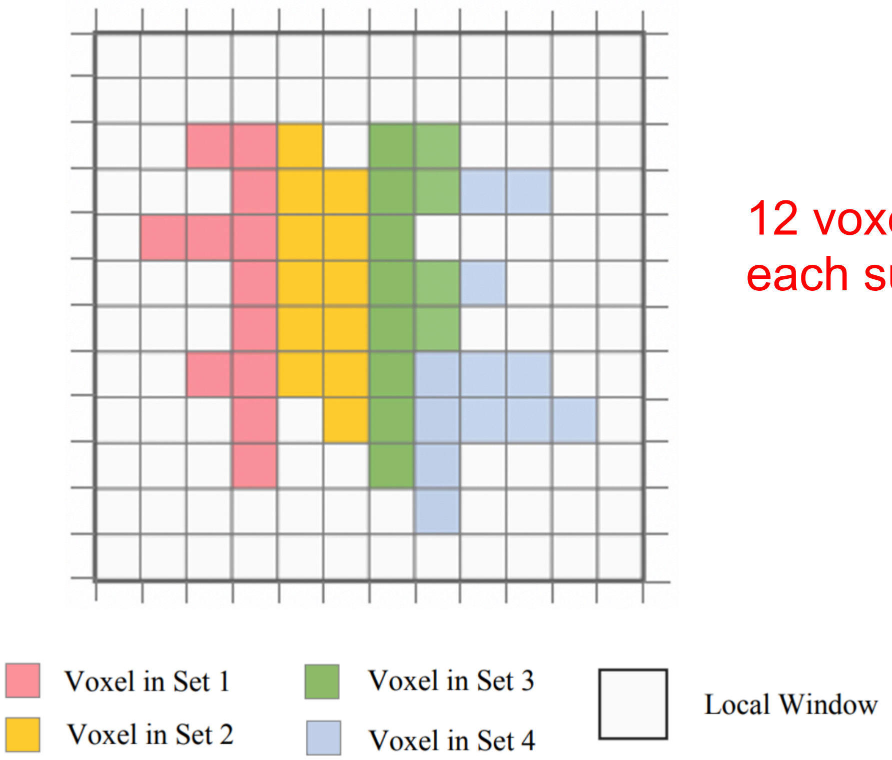


Note that a lidar typically operates at 10 Hz to 20 Hz.

Dynamic Sparse Window Attention

北京大学 PEKING UNIVERSITY

- Dynamic Set Partition
  - Combine Local Region and Voxel Number.
  - Reformulate sparse window attention as parallel computing self-attention within a series of local sets.
    - **Window Bounded:** Compute attention in local region
    - **Non-overlapped:** the local sets are non-overlapped
    - **Size-Equivalent:** each subset is guaranteed to have the same number of voxels
  - Dynamic: The set number dynamically varies with the sparsity of the window.



12 voxels in each subset

Dynamic Sparse Window Attention

- Rotated set attention for intra-window feature propagation.
  - Computing self-attention inside the invariant partition lacks connections across the subsets.
  - Dynamic set partition is highly dependent on the inner-window voxel ID
  - Control the covered local region of each set by voxel ID reordering with different sorting strategies.

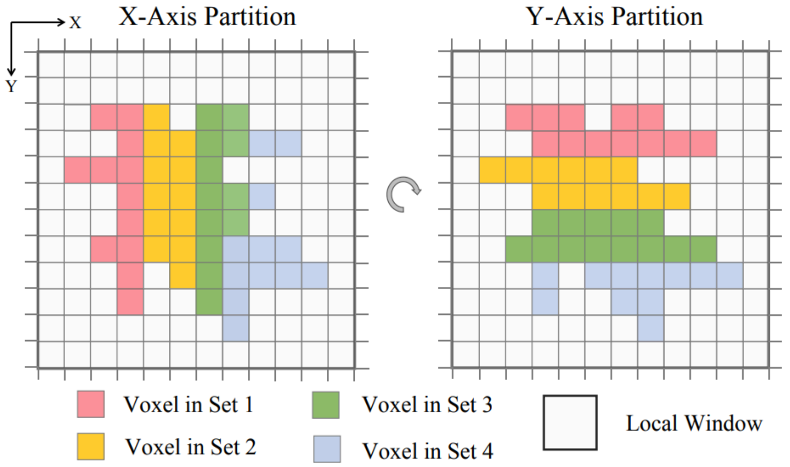


Figure 2. A demonstration of dynamic sparse window attention in our DSVT block. In the X-Axis DSVT layer, the sparse voxels will be split into a series of window-bounded and size-equivalent subsets in X-Axis main order, and self-attention is computed within each set. In the next layer, the set partition is switched to Y-Axis. The self-attention computation in the new sets crosses the boundaries of the previous sets, providing connections among them.

Dynamic Sparse Window Attention

- Rotated set attention for intra-window feature propagation.
  - Rotated-set attention approach that alternates between X-Axis and Y-Axis partitioning configurations in consecutive attention layers.
  - One DSVT Block:

$$
\begin{aligned}
\mathcal{F}^l, \mathcal{O}^l &= \text{INDEX}(\mathcal{V}^{l-1}, \{\mathcal{Q}_j\}_{j=0}^{S-1}, \mathcal{D}_x), \\
\mathcal{V}^l &= \text{MHSA}(\mathcal{F}^l, \text{PE}(\mathcal{O}^l)), \\
\mathcal{F}^{l+1}, \mathcal{O}^{l+1} &= \text{INDEX}(\mathcal{V}^l, \{\mathcal{Q}_j\}_{j=0}^{S-1}, \mathcal{D}_y), \\
\mathcal{V}^{l+1} &= \text{MHSA}(\mathcal{F}^{l+1}, \text{PE}(\mathcal{O}^{l+1})),
\end{aligned}
$$

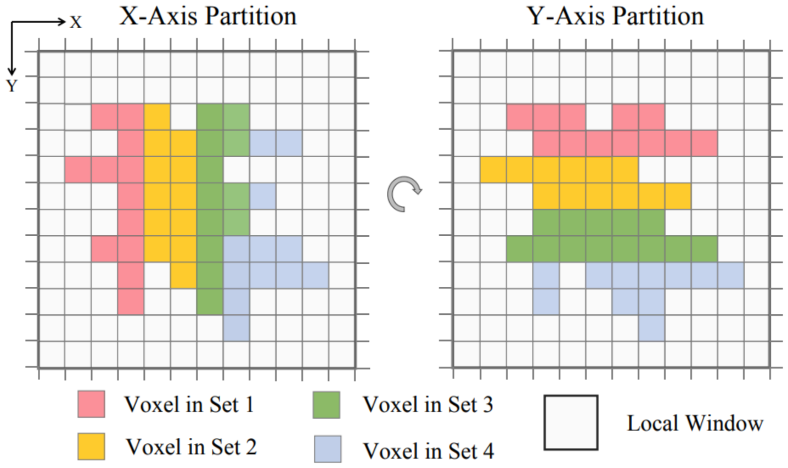


Figure 2. A demonstration of dynamic sparse window attention in our DSVT block. In the X-Axis DSVT layer, the sparse voxels will be split into a series of window-bounded and size-equivalent subsets in X-Axis main order, and self-attention is computed within each set. In the next layer, the set partition is switched to Y-Axis. The self-attention computation in the new sets crosses the boundaries of the previous sets, providing connections among them.

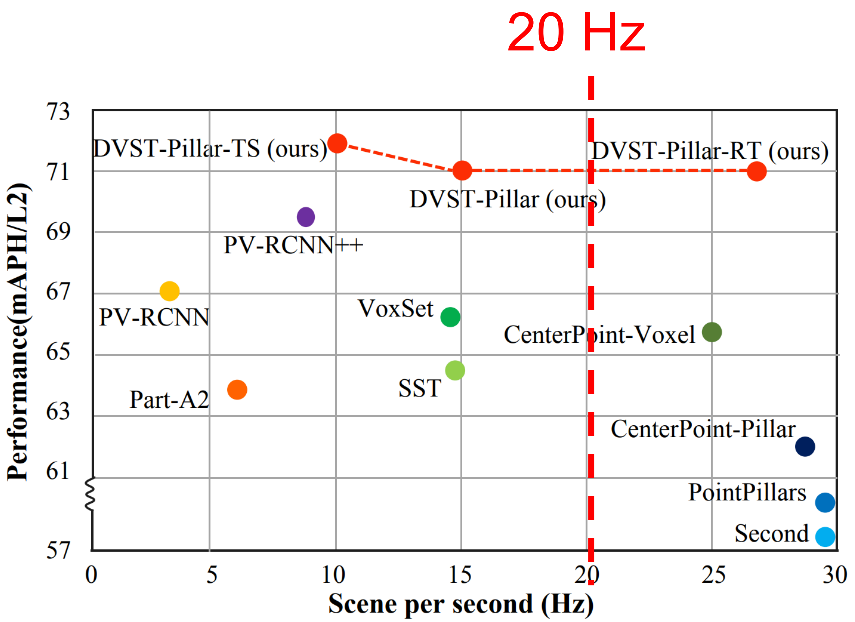
Experiments

- State-of-the-Art Performance on Waymo Open Dataset

Methods	Present at	Stages	mAP/mAPH L1	mAP/mAPH L2	*Vehicle* 3D AP/APH L1	*Vehicle* 3D AP/APH L2	*Pedestrian* 3D AP/APH L1	*Pedestrian* 3D AP/APH L2	*Cyclist* 3D AP/APH L1	*Cyclist* 3D AP/APH L2
SECOND [51]	Sensors'18	One	67.2/63.1	61.0/57.2	72.3/71.7	63.9/63.3	68.7/58.2	60.7/51.3	60.6/59.3	58.3/57.0
PointPillars‡ [22]	CVPR'19	One	69.0/63.5	62.8/57.8	72.1/71.5	63.6/63.1	70.6/56.7	62.8/50.3	64.4/62.3	61.9/59.9
CenterPoint-Voxel† [53]	CVPR'21	One	74.4/71.7	68.2/65.8	74.2/73.6	66.2/65.7	76.6/70.5	68.8/63.2	72.3/71.1	69.7/68.5
SST‡ [12]	CVPR'22	One	74.5/71.0	67.8/64.6	74.2/73.8	65.5/65.1	78.7/69.6	70.0/61.7	70.7/69.6	68.0/66.9
VoxSet [18]	CVPR'22	One	75.4/72.2	69.1/66.2	74.5/74.0	66.0/65.6	80.0/72.4	72.5/65.4	71.6/70.3	69.0/67.7
AFDetV2 [19]	AAAI'22	One	77.2/74.8	71.0/68.8	77.6/77.1	69.7/69.2	80.2/74.6	72.2/67.0	73.7/72.7	71.0/70.1
SWFormer [41]	ECCV'22	One	-/-	-/-	77.8/77.3	69.2/68.8	80.9/72.7	72.5/64.9	-/-	-/-
PillarNet-34 [34]	ECCV'22	One	77.3/74.6	71.0/68.5	79.1/78.6	70.9/70.5	80.6/74.0	72.3/66.2	72.3/71.2	69.7/68.7
CenterFormer [56]	ECCV'22	One	75.3/72.9	71.1/68.9	75.0/74.4	69.9/69.4	78.6/73.0	73.6/68.3	72.3/71.3	69.8/68.8
Ours (Pillar)	-	One	**79.5/77.1**	**73.2/71.0**	**79.3/78.8**	**70.9/70.5**	**82.8/77.0**	**75.2/69.8**	**76.4/75.4**	**73.6/72.7**
Ours (Voxel)	-	One	**80.3/78.2**	**74.0/72.1**	**79.7/79.3**	**71.4/71.0**	**83.7/78.9**	**76.1/71.5**	**77.5/76.5**	**74.6/73.7**
PV-RCNN† [35]	CVPR'20	Two	76.2/73.6	69.6/67.2	78.0/77.5	69.4/69.0	79.2/73.0	70.4/64.7	71.5/70.3	69.0/67.8
Part-A2-Net [38]	TPAMI'20	Two	73.6/70.3	66.9/63.8	77.1/76.5	68.5/68.0	75.2/66.9	66.2/58.6	68.6/67.4	66.1/64.9
CenterPoint-Voxel [53]	CVPR'21	Two	-/-	-/-	76.7/76.2	68.8/68.3	79.0/72.9	71.0/65.3	-/-	-/-
PV-RCNN++(center) [36]	IJCV'22	Two	78.1/75.9	71.7/69.5	79.3/78.8	70.6/70.2	81.3/76.3	73.2/68.0	73.7/72.7	71.2/70.2
FSD [13]	NeurIPS'22	Two	79.6/77.4	72.9/70.8	79.2/78.8	70.5/70.1	82.6/77.3	73.9/69.1	77.1/76.0	74.4/73.3
Ours (Pillar-TS)	-	Two	**80.6/78.2**	**74.3/72.1**	**80.2/79.7**	**72.0/71.6**	**83.7/78.0**	**76.1/70.7**	**77.8/76.8**	**74.9/73.9**
Ours (Voxel-TS)	-	Two	**81.1/78.9**	**74.8/72.8**	**80.4/79.9**	**72.2/71.8**	**84.2/79.3**	**76.5/71.8**	**78.6/77.6**	**75.7/74.7**

Methods	Present at	*val* NDS	*val* mAP	*test* NDS	*test* mAP
PointPillars [22]	CVPR'19	-	-	45.3	30.5
CBGS [57]	ArXiv'19	62.3	50.6	63.3	52.8
CenterPoint-Voxel [53]	CVPR'21	66.8	59.6	67.3	60.3
Transfusion-L [1]	CVPR'22	69.3	64.7	70.2	65.5
PillarNet-34 [34]	ECCV'22	-	-	71.4	66.0
Ours (Pillar)	-	**71.1**	**66.4**	**72.7**	**68.4**

Encoder	DA	PC	WW	SL	CP	DI	mIoU
2D Conv [25]	72.0	43.1	53.1	29.7	27.7	37.5	43.8
3D SpConv [25]	75.6	48.4	57.5	36.5	31.7	41.9	48.6
Ours (Pillar)	**79.7**	**51.8**	**61.1**	**38.2**	**33.8**	**45.3**	**51.6**
Ours (Pillar)†	**87.6**	**67.2**	**72.7**	**59.7**	**62.7**	**58.2**	**68.0**



Conclusion

- We propose DSVT, a deployment-friendly yet powerful transformer-only 3D backbone for 3D object detection, which can be accelerated by NVIDIA TensorRT with real-time running speed (27Hz).
- We hope that our DVST can not only be a reliable point cloud processor for 3D object detection in real-world applications but also provide a potential solution for efficiently handling large-scale sparse data in other tasks.