Point Cloud Hotspot Detection

Guojin Chen

Department of Computer Science and Engineering
The Chinese University of Hong Kong

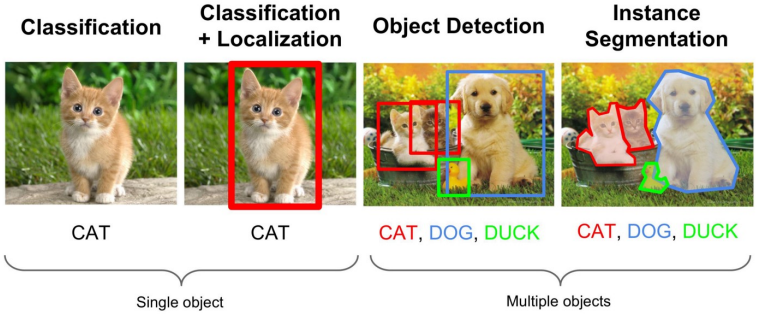
June 27, 2020

Advisor: Prof. Bei Yu

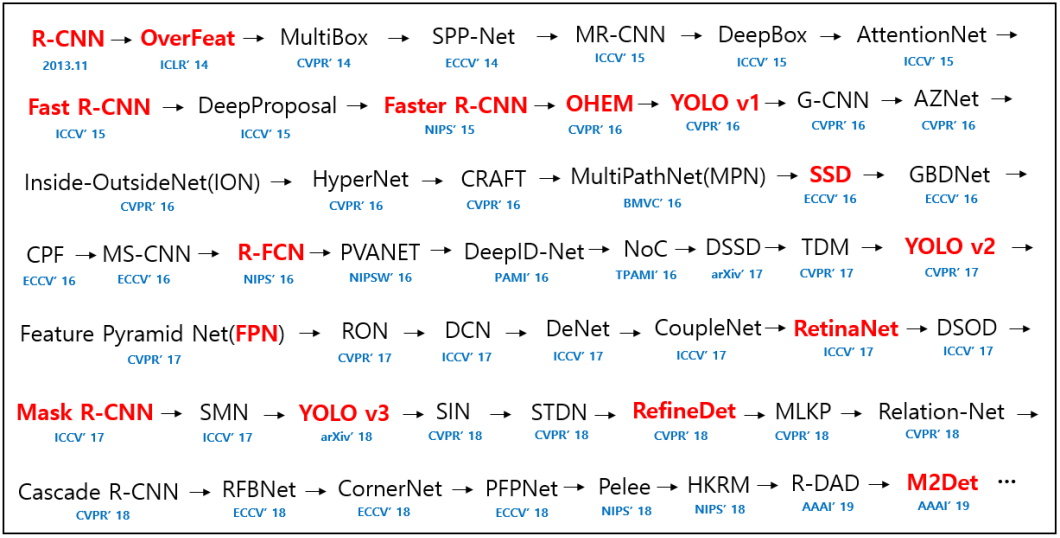
香港中文大學
The Chinese University of Hong Kong

Object detection

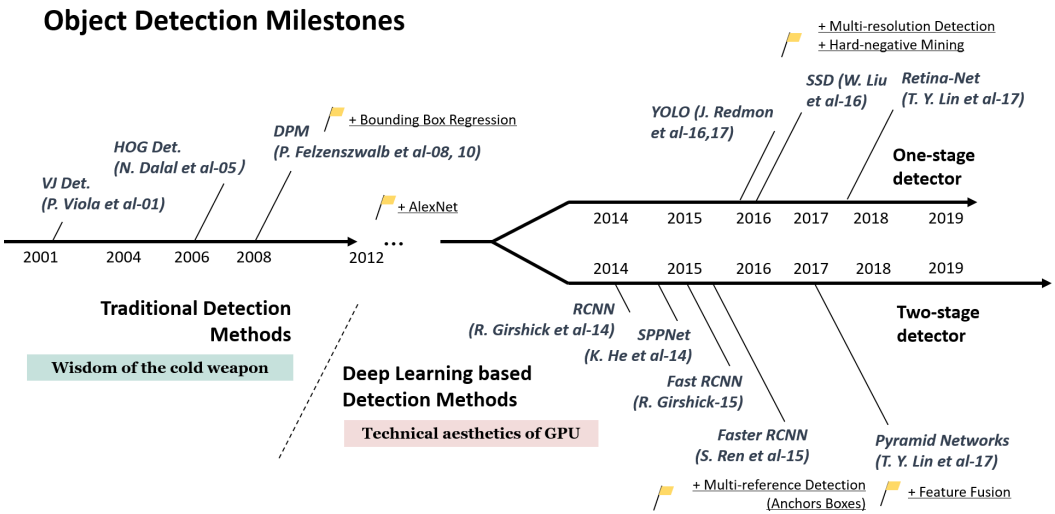
Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.



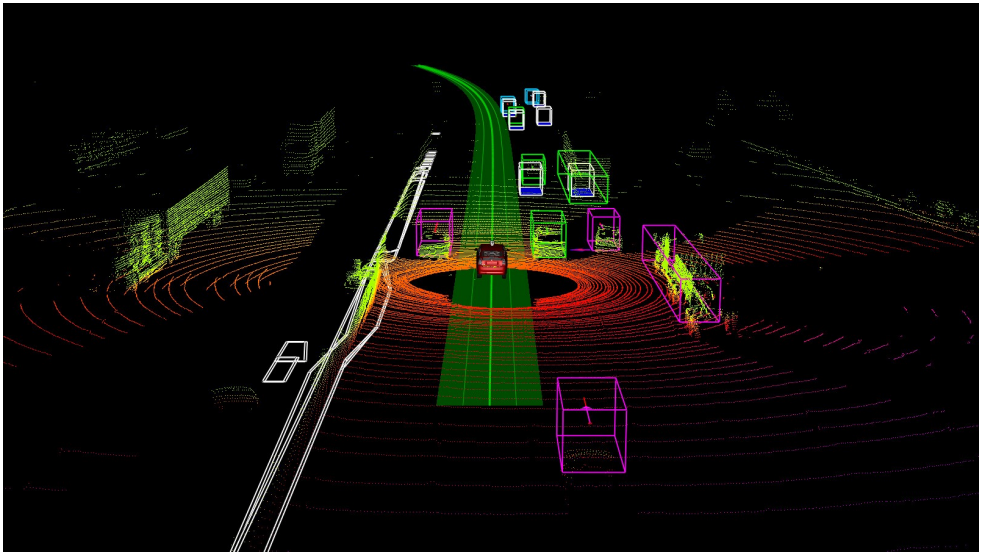
2D Detection Milestones: Paper list from 2014 to now(2020)



2D Detection Milestones: Paper list from 2014 to now(2020)



Intro for 3D detection.



Point Cloud Methods

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

On Kitti 3D Detection

PV-RCNN > SA-SSD > PointRCNN > F-PointNet > YOLO3D > Vote3Deep

On Kitti Bird's Eye View Evaluation

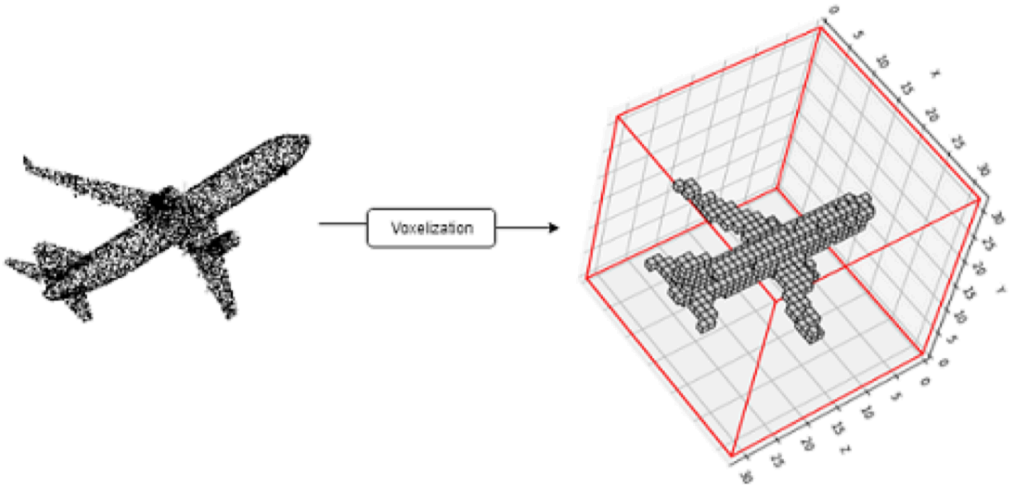
SA-SSD > PV-RCNN > PointRCNN > F-PointNet > YOLO3D > Vote3Deep

3D Voxel-based Methods

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

3D Voxel-based Methods

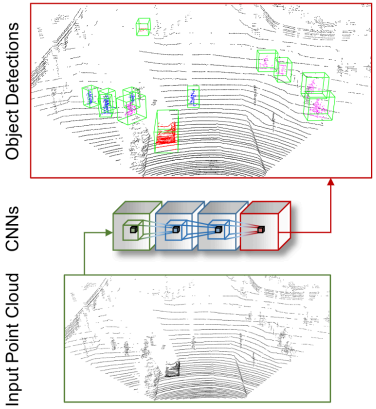
Voxelization:

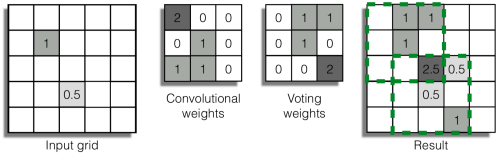


3D Voxel-based Methods

Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient CNNs (IROS2017) [1]

Sliding window. Voxelization. Brute force 3D CNN.





The voting weights are obtained by flipping the convolutional weights along each dimension.

3D Row Points Methods

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

3D Row Points Methods: Strating from PointNet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]

Achieve effective feature learning directly on point clouds. Before PointNet, point cloud is converted to other representations before its fed to a deep neural network

PointNet++:Deep Hierarchical Feature Learning on Pointsets in Metric Space (NIPS2017) [3]

Pointnet++ is build upon Pointnet, Similar to CNNs, Pointnet++ extracts local features from a small neighborhood and further grouping into larger units and processed to produce higher level features.

3D Row Points Methods: Strating from PointNet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]

Achieve effective feature learning directly on point clouds.

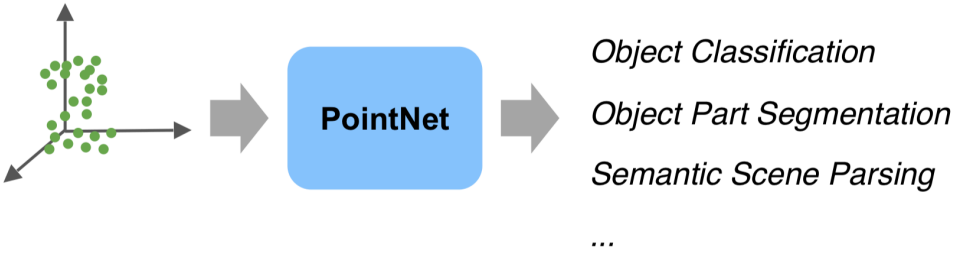
Point cloud is **converted to other representations** before it's fed to a deep neural network

Conversion	Deep Net
Voxelization	3D CNN
Projection/Rendering	2D CNN
Feature extraction	Fully Connected

3D Row Points Methods: Strating from PointNet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]

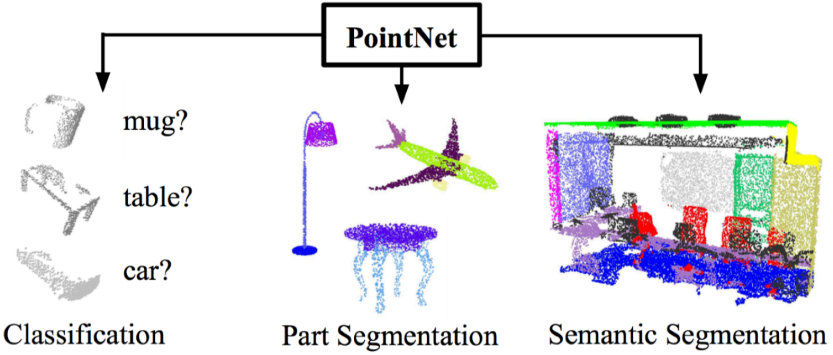
- End-to-end learning for scattered, unordered point data
- Unified framework for various tasks



3D Row Points Methods: Strating from PointNet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]

- End-to-end learning for scattered, unordered point data
- Unified framework for various tasks



3D Row Points Methods: Strating from PointNet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]

Theorem

Theorem 1. *Suppose $f : \mathcal{X} \to \mathbb{R}$ is a continuous set function w.r.t Hausdorff distance $d_H(\cdot,\cdot)$. $\forall \epsilon > 0$, $\exists$ a continuous function h and a symmetric function $g(x_1,\ldots,x_n) = \gamma \circ MAX$, such that for any $S \in \mathcal{X}$,*

$$\left| f(S) - \gamma \left(\underset{x_i \in S}{MAX} \{h(x_i)\} \right) \right| < \epsilon$$

where $x_1,\ldots,x_n$ is the full list of elements in S ordered arbitrarily, γ is a continuous function, and MAX is a vector max operator that takes n vectors as input and returns a new vector of the element-wise maximum.

Theorem 1 proves that PointNet's network structure can fit any continuous point cloud set function.

3D Row Points Methods: Strating from PointNet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]

Theorem

Theorem 2. *Suppose* $\mathbf{u} : \mathcal{X} \rightarrow \mathbb{R}^K$ *such that* $\mathbf{u} = \underset{x_i \in S}{MAX}\{h(x_i)\}$ *and* $f = \gamma \circ \mathbf{u}$. *Then,*

(a) $\forall S, \exists\, \mathcal{C}_S, \mathcal{N}_S \subseteq \mathcal{X}$, $f(T) = f(S)$ *if* $\mathcal{C}_S \subseteq T \subseteq \mathcal{N}_S$;

(b) $|\mathcal{C}_S| \leq K$

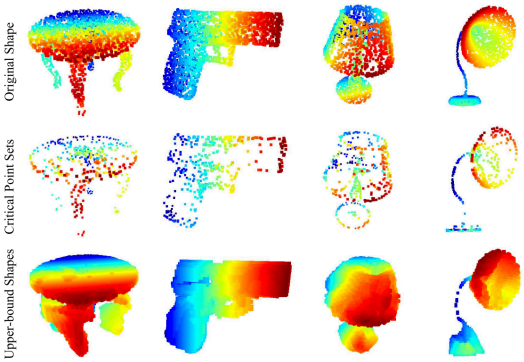
Intuitively, theorem 2 proves that PointNet learns to summarize a shape by a sparse set of key points.

3D Row Points Methods: Strating from PointNet

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]

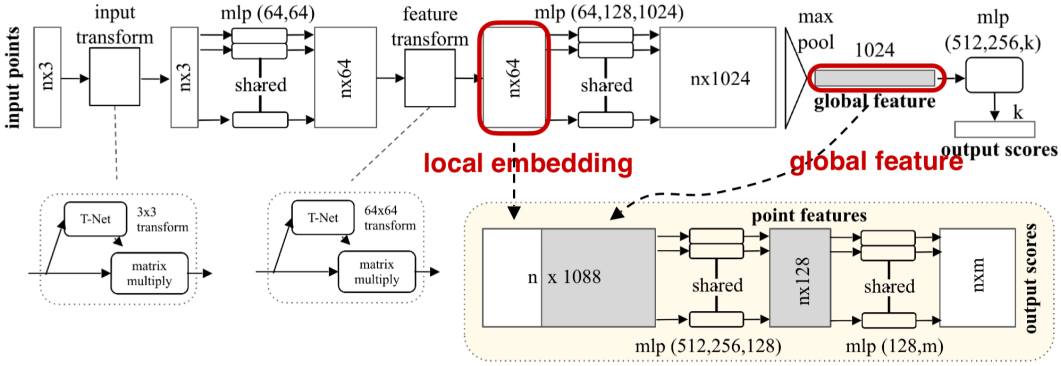
Theorem

Intuitively, theorem 2 proves that PointNet learns to summarize a shape by a sparse set of key poin



3D Row Points Methods: Strating from PointNet

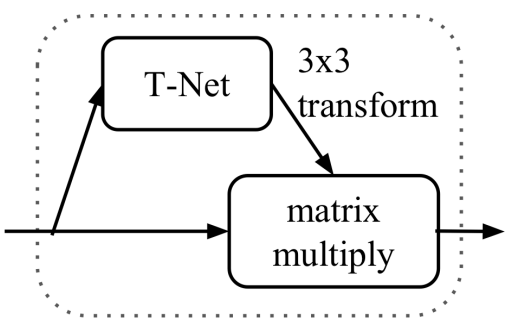
PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) [2]



3D Row Points Methods: Strating from PointNet

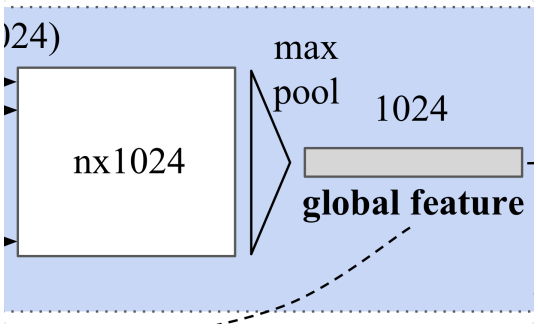
Transformation Invariance

The classification (and segmentation) of an object should be invariant to certain geometric transformations (e.g., rotation).



Permutation Invariance

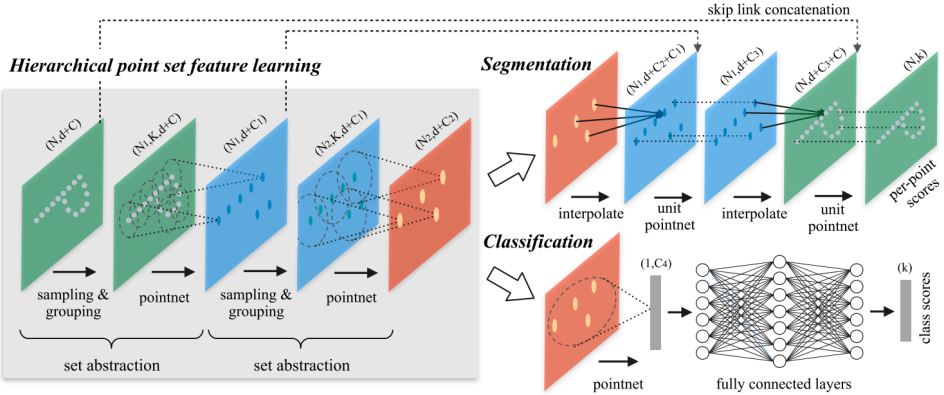
PointNet implements the symmetric function with max pooling.



3D Row Points Methods: Strating from PointNet

PointNet++:Deep Hierarchical Feature Learning on Pointsets in Metric Space (NIPS2017) [3]

Pointnet++ is build upon Pointnet, Similar to CNNs, Pointnet++ extracts local features from a small neighborhood and further grouping into larger units and processed to produce higher level features.



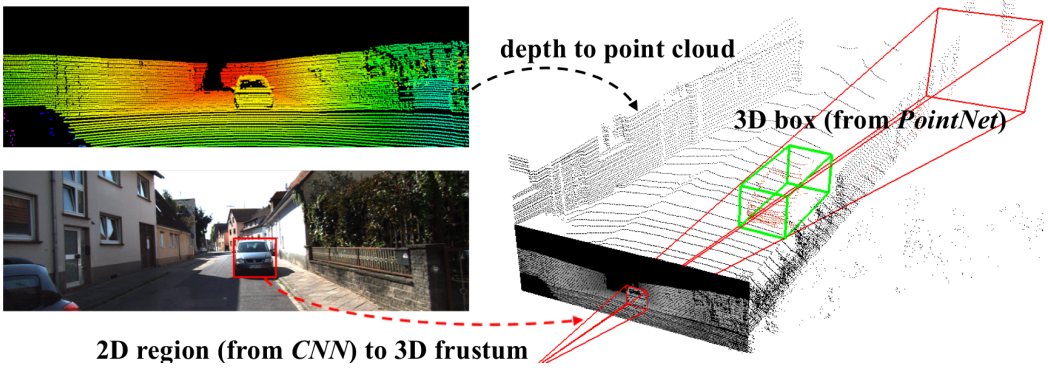
3D Image+Points Methods: F-PointNet

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

3D Image+Points Methods: F-PointNet

F-PointNet:Frustum PointNets for 3D Object Detection from RGB-D Data (CVPR2018) [4]

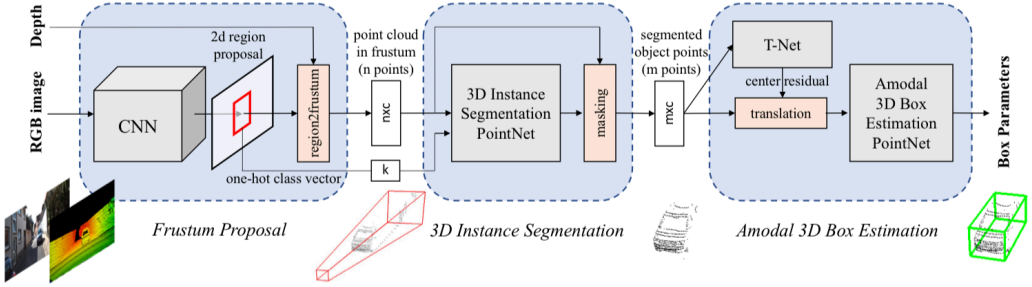
Instead of solely relying on 3D proposals, this method leverages both mature 2D object detection and advanced 3D deep learning for object localization.



3D Image+Points Methods: F-PointNet

F-PointNet:Frustum PointNets for 3D Object Detection from RGB-D Data (CVPR2018) [4]

Instead of solely relying on 3D proposals, this method leverages both mature 2D object detection and advanced 3D deep learning for object localization.



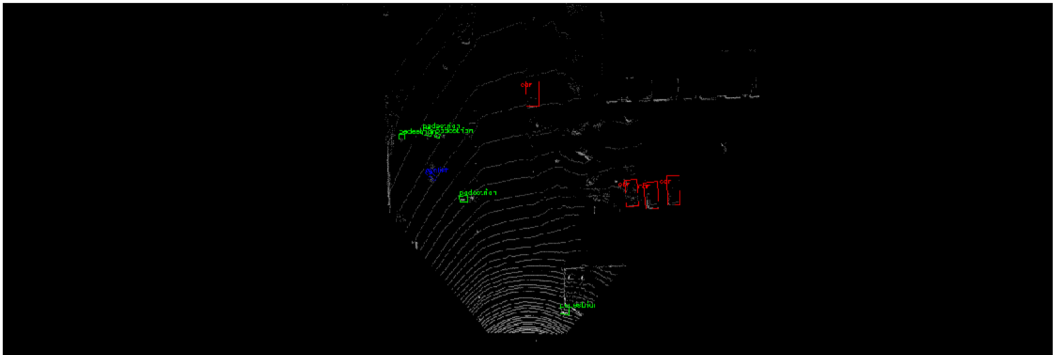
3D Row Points Methods: One-stage YOLO3D

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

3D Row Points Methods: One-stage YOLO3D

YOLO3D: End-to-end real-time 3D BBox Detection from LiDAR Point Cloud (ECCV2018) [5]

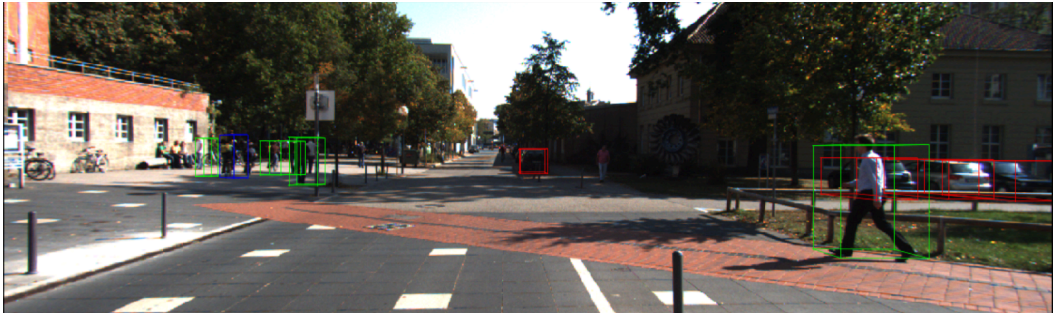
Apply YOLO on 3D data set. Using MVNet build Bird Eye View, then apply YOLO on BEV.



3D Row Points Methods: One-stage YOLO3D

YOLO3D: End-to-end real-time 3D BBox Detection from LiDAR Point Cloud (ECCV2018) [5]

Apply YOLO on 3D data set. Using MVNet build Bird Eye View, then apply YOLO on BEV.



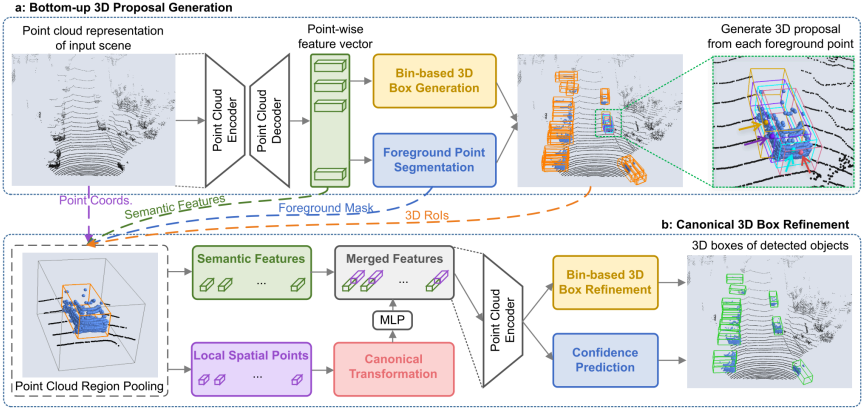
3D Row Points Methods: Two-stage PointRCNN

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

3D Row Points Methods: Two-stage PointRCNN

PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud (CVPR2019) [6]

The whole PointRCNN network consists of two parts: (a) for generating 3D proposals from raw point cloud in a bottom-up manner. (b) for refining the 3D proposals in canonical coordinate.



3D Voxel+Points Methods: One-stage SA-SSD

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

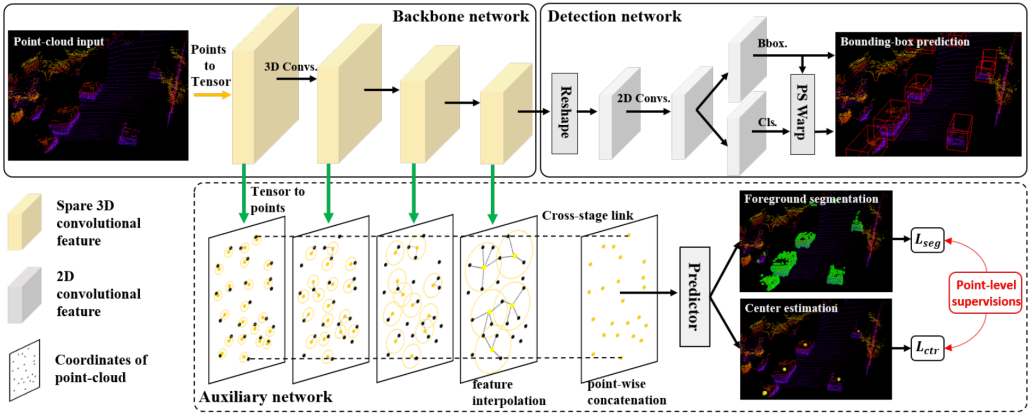
On Kitti Bird's Eye View Evaluation

SA-SSD > PV-RCNN > PointRCNN > F-PointNet > YOLO3D > Vote3Deep

3D Voxel+Points Methods: One-stage SA-SSD

SA-SSD: Structure Aware Single-stage 3D Object Detection from Point Cloud(CVPR2020) [7]

Backbone, Anchor free detection Head, Auxiliary Network. Voxelization but save point information. Auxiliary Network to learn the point level features.



3D Voxel+Points Methods: One-stage SA-SSD

SA-SSD: Structure Aware Single-stage 3D Object Detection from Point Cloud(CVPR2020) [7]

First prize of Kitti Bird's Eye View Evaluation.

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	SA-SSD		code	**91.03 %**	95.03 %	85.96 %	0.04 s	1 core @ 2.5 Ghz (Python)	☐
C. He, H. Zeng, J. Huang, X. Hua and L. Zhang: Structure Aware Single-stage 3D Object Detection from Point Cloud. CVPR 2020.									
2	MMLab PV-RCNN		code	90.65 %	94.98 %	86.14 %	0.08 s	1 core @ 2.5 Ghz (Python + C/C++)	☐
S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang and H. Li: PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. CVPR 2020.									
3	CN			90.50 %	94.51 %	85.86 %	0.04 s	GPU @ 2.5 Ghz (Python + C/C++)	☐
4	Associate-3Ddet_v2			90.00 %	**95.55 %**	84.72 %	0.04 s	1 core @ 2.5 Ghz (Python)	☐
5	AIMC-RUC			89.80 %	93.64 %	84.64 %	0.08 s	1 core @ 2.5 Ghz (C/C++)	☐
6	OAP			89.72 %	93.13 %	82.25 %	0.06 s	1 core @ 2.5 Ghz (C/C++)	☐
7	D3D			89.72 %	93.37 %	84.72 %	0.02 s	1 core @ 2.5 Ghz (C/C++)	☐

3D Voxel+Points Methods: Two-stage PV-RCNN

	Sliding Window	One-Stage	Two-Stage
Voxel-based	Vote3Deep(IROS2017)		
Image+Points			F-PointNet(CVPR2018)
Row Points		YOLO3D(ECCV2018)	PointRCNN(CVPR2019)
Voxel+Points		SA-SSD(CVPR2020)	PV-RCNN(CVPR2020)
Graph+Points			Grid-GCN(CVPR2020)

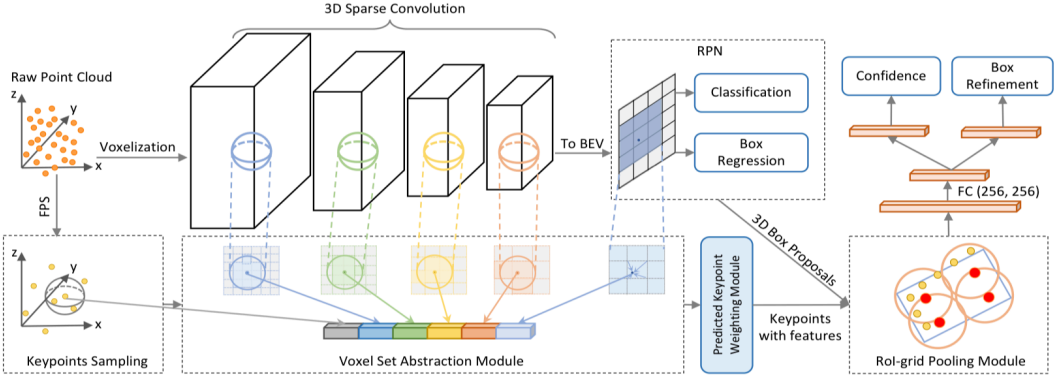
On Kitti 3D Detection

PV-RCNN > SA-SSD > PointRCNN > F-PointNet > YOLO3D > Vote3Deep

3D Voxel+Points Methods: Two-stage PV-RCNN

PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection(CVPR2020) [8]

Two stage Point-voxel 3D detection. PointNet++ as backbone.



3D Voxel+Points Methods: Two-stage PV-RCNN

PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection(CVPR2020) [8]

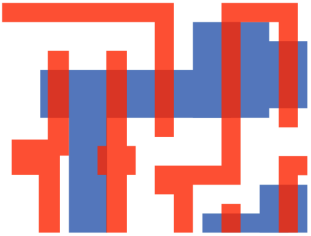
First prize of Kitti 3D Object Detection Evaluation.

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	MMLab PV-RCNN		code	**81.43 %**	90.25 %	**76.82 %**	0.08 s	1 core @ 2.5 Ghz (Python + C/C++)	
S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang and H. Li: PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. CVPR 2020.									
2	Associate-3Ddet_v2			80.77 %	**91.53 %**	75.23 %	0.04 s	1 core @ 2.5 Ghz (Python)	
3	AIMC-RUC			80.63 %	89.90 %	75.32 %	0.08 s	1 core @ 2.5 Ghz (C/C++)	
4	OAP			80.63 %	89.18 %	73.04 %	0.06 s	1 core @ 2.5 Ghz (C/C++)	
5	3D-CVF at SPA			80.05 %	89.20 %	73.11 %	0.06 s	1 core @ 2.5 Ghz (C/C++)	
J. Yoo, Y. Kim, J. Kim and J. Choi: 3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection. arXiv preprint arXiv:2004.12636 2020.									
6	CN			79.89 %	90.55 %	76.31 %	0.04 s	GPU @ 2.5 Ghz (Python + C/C++)	
7	NLK-3D			79.81 %	89.03 %	74.47 %	0.04 s	1 core @ 2.5 Ghz (Python + C/C++)	

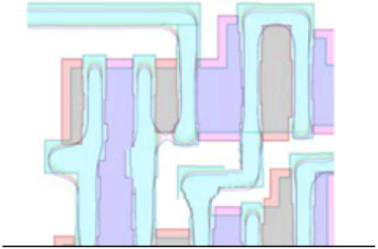
Brief Summary of Points Cloud Methods

Methods	DeepNets	Paper	Advantages	Shortages
Voxel	Conv3D	Vote3Deep	1. Straight-forward 2. easy to implement	1. Too slow 2. Sparse voxels, unuseful 3. research on points to voxels
Image	Conv2D	F-PointNet	take advantages of 2d Det.	results based on points to images features.
Points	PointConv	PointNet	1. No info loss. 2. fast	Lack of common tools.

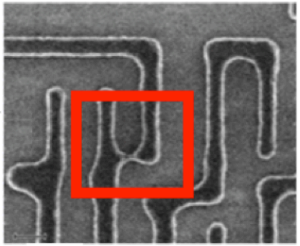
Failure (Hotspot) Detection



Pre-OPC Layout

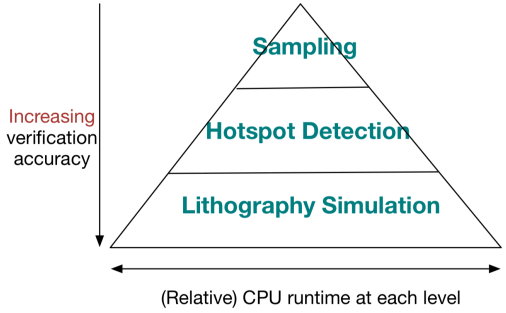


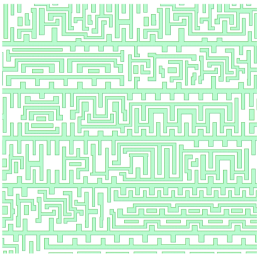
Post-OPC Mask



Hotspot on Wafer

Failure (Hotspot) Detection





- **Sampling** (DRC Checking): scan and rule check each region.
- **Hotspot Detection** verify the sampled regions and report potential hotspots.
- **Lithography Simulation**: final verification on the reported hotspots.

Success on Deep Learning based Hotspot Detection

Layout Hotspot Detection with Feature Tensor Generation and Deep Biased Learning (TCAD'19) [9]

DFT encoding blocks and CNNs for hotspots detection.

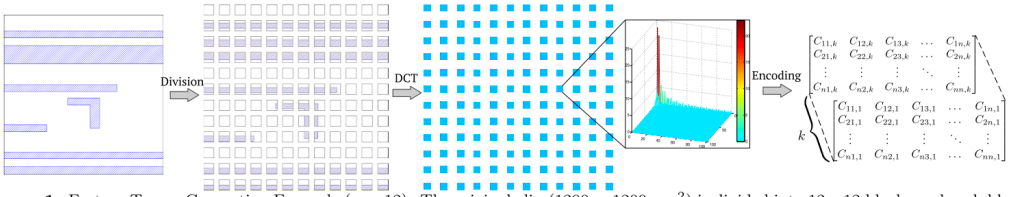


Figure 1: Feature Tensor Generation Example ($n = 12$). The original clip ($1200 \times 1200\ nm^2$) is divided into 12×12 blocks and each block is converted to 100×100 image representing $100 \times 100\ nm^2$ sub region of the original clip. Feature tensor is then obtained by encoding on DCT coefficients of each block.

Success on Deep Learning based Hotspot Detection

Layout Hotspot Detection with Feature Tensor Generation and Deep Biased Learning (TCAD'19) [9]

DFT encoding blocks and CNNs for hotspots detection.

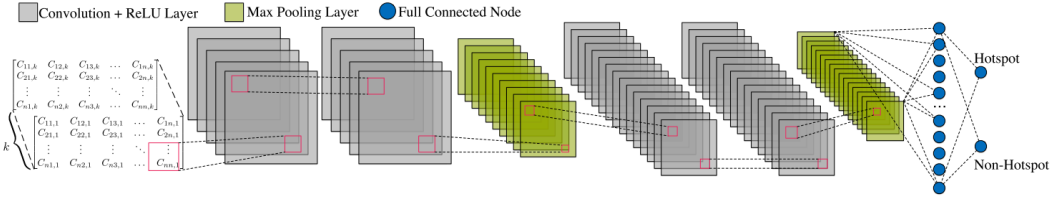
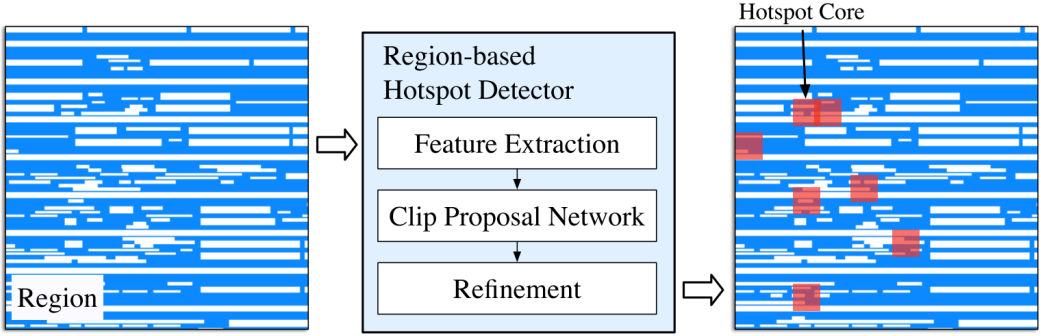


Figure 2: The proposed convolutional neural network structure.

Success on Deep Learning based Hotspot Detection

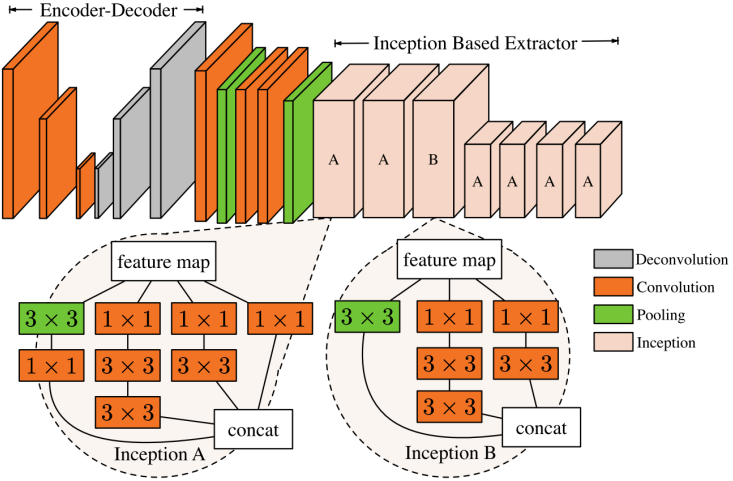
Faster Region-based Hotspot Detection (DAC'19) [10]

Region-based hotspot detection flow.



Success on Deep Learning based Hotspot Detection

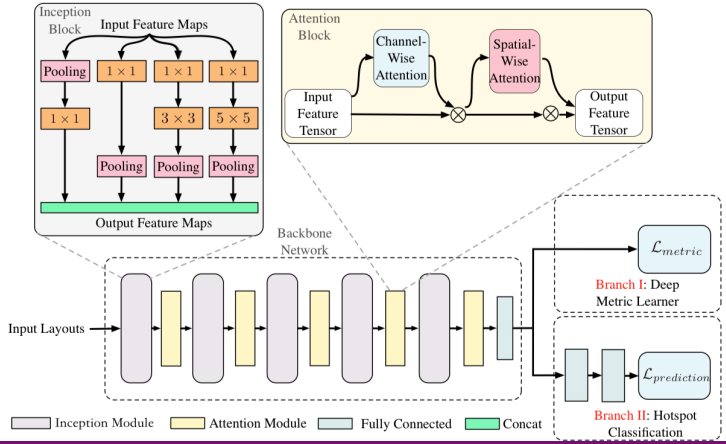
Faster Region-based Hotspot Detection (DAC'19) [10]



Success on Deep Learning based Hotspot Detection

Hotspot Detection via Attention-based Deep Layout Metric Learning

Hotspot detection flow with attention, inception, and metric learning.



Why Point Cloud HSD?

- Why point cloud?
- Why hotspot detection?

Toy example of Point Cloud HSD

Model : SA-SSD, First prize of BEV. https://github.com/skyhehe123/SA-SSD
Dataset : ICCAD16-N7M2EUV case1. https://github.com/phdyang007/ICCAD16-N7M2EUV

Toy example of Point Cloud HSD

Model : SA-SSD, First prize of BEV. https://github.com/skyhehe123/SA-SSD

Dataset : ICCAD16-N7M2EUV case1. https://github.com/phdyang007/ICCAD16-N7M2EUV

Kitti : http://www.cvlibs.net/datasets/kitti/

For quick validation, I didn't modify the network architecture. Instead, I transfer the GDS layout to kitti format.

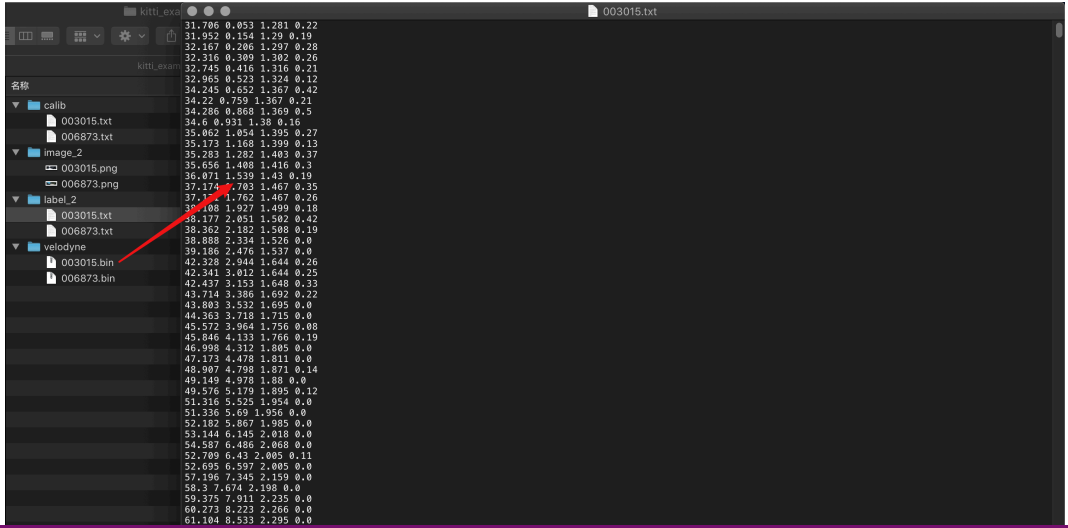
Toy example of Point Cloud HSD

Kitti data format

Kitti Lidar velodyne, label, image, calib.

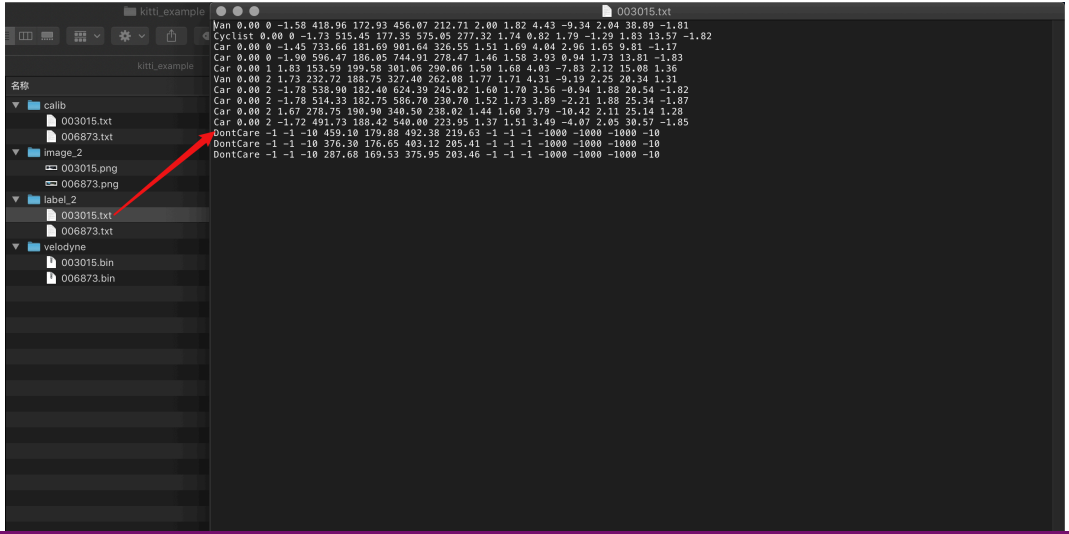
Toy example of Point Cloud HSD

Kitti Lidar velodyne



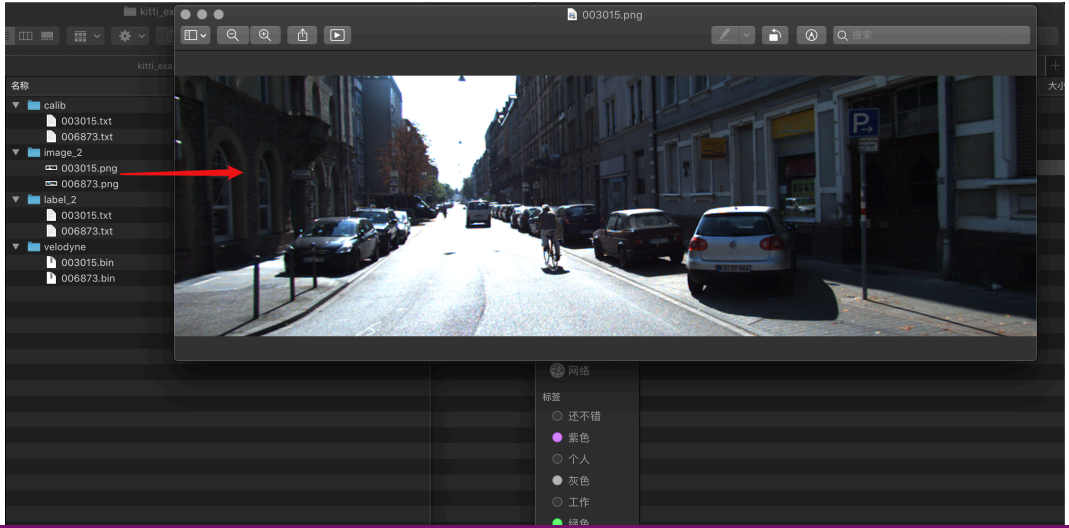
Toy example of Point Cloud HSD

Kitti Lidar label



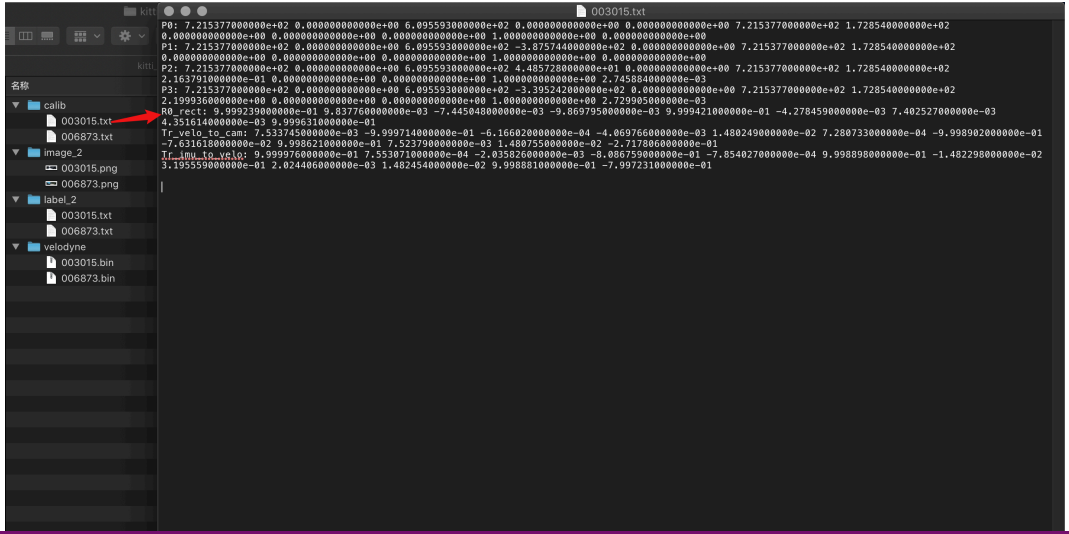
Toy example of Point Cloud HSD

Kitti image



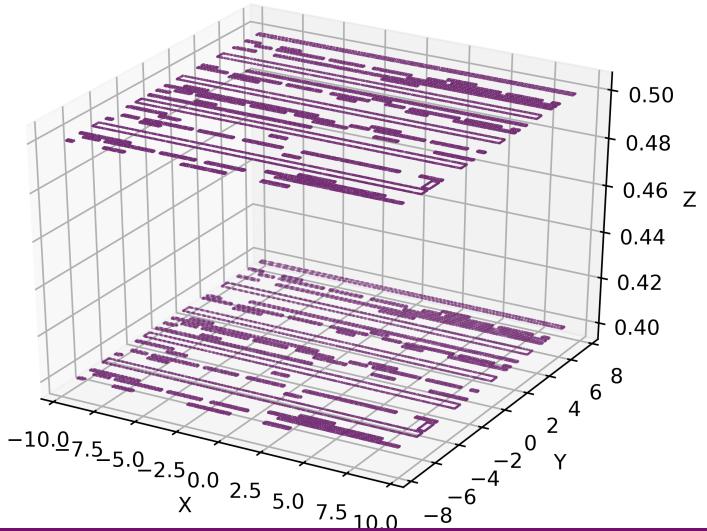
Toy example of Point Cloud HSD

Kitti calib



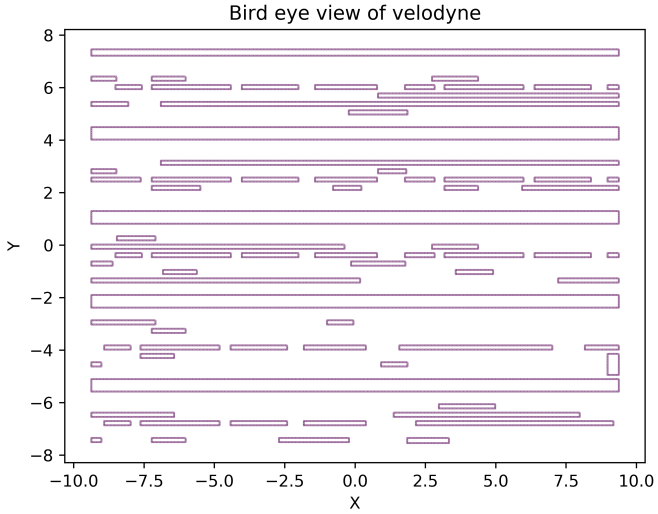
GDS Layout to Point Cloud datasets

3d visualization.



GDS Layout to Point Cloud datasets

Bird eye view.



GDS Layout to Point Cloud datasets

Inverse kitti label parameters.

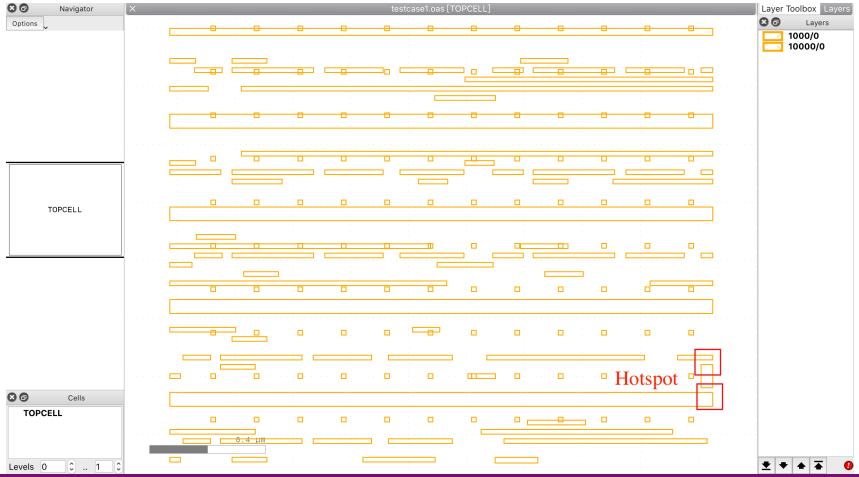
Hotspot 0.00 0 -1.57 960.00 150.00 1200.00 152.00 1.0 1.60 1.60 9.17 0.5 -4.06 1.57
Hotspot 0.00 0 -1.57 960.00 150.00 1200.00 152.00 1.0 1.60 1.60 9.17 0.5 -5.02 -1.57

Inverse kitti calib parameters.

```
P0: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02 0.000000000000e+00 0.000000000000e+00 7.070493000000e+02
1.805066000000e+02 0.000000000000e+00 0.000000000000e+00 0.000000000000e+00 1.000000000000e+00 0.000000000000e+00
P1: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02 -3.797842000000e+02 0.000000000000e+00 7.070493000000e+02
1.805066000000e+02 0.000000000000e+00 0.000000000000e+00 0.000000000000e+00 1.000000000000e+00 0.000000000000e+00
P2: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02 4.575831000000e+01 0.000000000000e+00 7.070493000000e+02
1.805066000000e+02 -3.454157000000e-01 0.000000000000e+00 0.000000000000e+00 1.000000000000e+00 4.981016000000e-03
P3: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02 -3.341081000000e+02 0.000000000000e+00 7.070493000000e+02
1.805066000000e+02 2.330660000000e+00 0.000000000000e+00 0.000000000000e+00 1.000000000000e+00 3.201153000000e-03
R0_rect: 9.999128000000e-01 1.009263000000e-02 -8.511932000000e-03 -1.012729000000e-02 9.999406000000e-01
-4.037671000000e-03 8.470675000000e-03 4.123522000000e-03 9.999556000000e-01
Tr_velo_to_cam: 6.927964000000e-03 -9.999722000000e-01 -2.757829000000e-03 -2.457729000000e-02 -1.162982000000e-03
2.749836000000e-03 -9.999955000000e-01 -6.127237000000e-02 9.999753000000e-01 6.931141000000e-03 -1.143899000000e-03
-3.321029000000e-01
Tr_imu_to_velo: 9.999976000000e-01 7.553071000000e-04 -2.035826000000e-03 -8.086759000000e-01 -7.854027000000e-04
9.998898000000e-01 -1.482298000000e-02 3.195559000000e-01 2.024406000000e-03 1.482454000000e-02 9.998881000000e-01
-7.997231000000e-01
```

Toy example results

- Real time 25 fps, 200 test data in 8s. ∼40ms.
- In the toy case one, 100% accuracy.

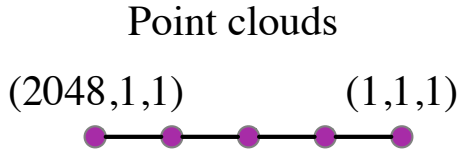


Why point cloud: 1. Scalable

- If just transform the representation from image to point cloud, this idea is not interesting.
- The real goal is to build a scalable vector auto encoder for the EDA area, especially the layout.

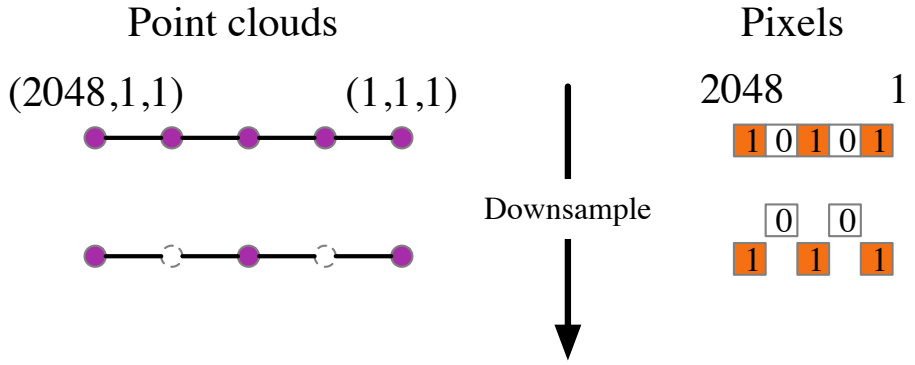
Why point cloud: 1. Scalable

- If just transform the representation from image to point cloud, this idea is not interesting.
- The real goal is to build a scalable vector auto encoder for the EDA area, especially the layout.



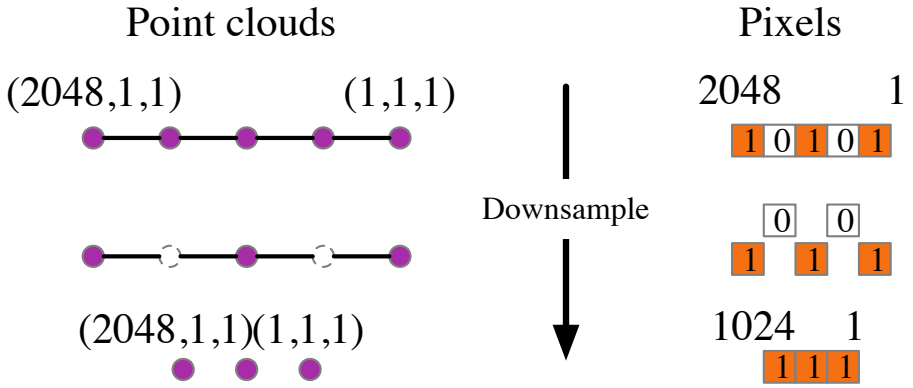
Why point cloud: 1. Scalable

- If just transform the representation from image to point cloud, this idea is not interesting.
- The real goal is to build a scalable vector auto encoder for the EDA area, especially the layout.



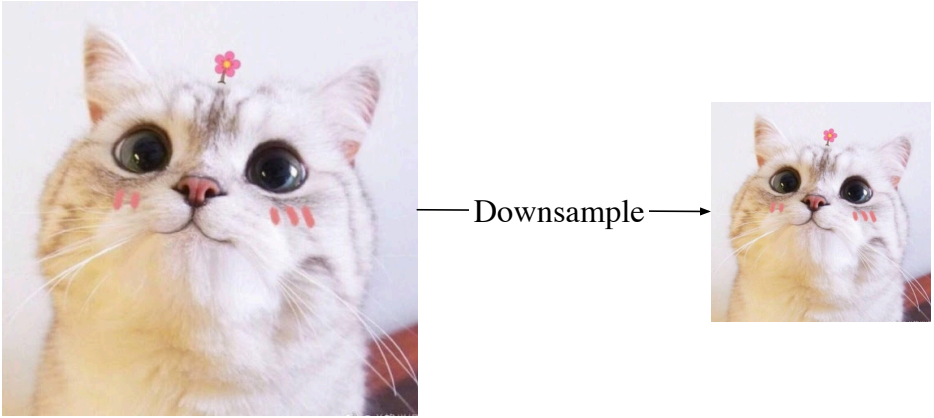
Why point cloud: 1. Scalable

- If just transform the representation from image to point cloud, this idea is not interesting.
- The real goal is to build a scalable vector auto encoder for the EDA area, especially the layout.



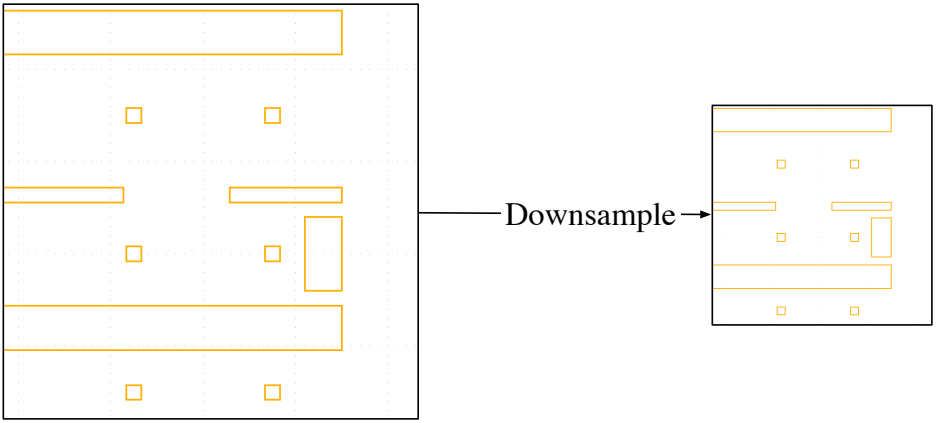
Why point cloud: 1. Scalable

- If just transform the representation from image to point cloud, this idea is not interesting.
- The real goal is to build a scalable vector auto encoder for the EDA area, especially the layout.



Why point cloud: 1. Scalable

- If just transform the representation from image to point cloud, this idea is not interesting.
- The real goal is to build a scalable vector auto encoder for the EDA area, especially the layout.



Why point cloud: 2. lower computational overhead.

- A 2048×2048 pixel image has **4194304** tensors.
- In the toy case, it used only **58520** points to represent the layout.
- **4194304** $\div$ **58520** $= 71.67$
- It gives the possibility to detect the full chip hotspot at one time.

Why HSD?

Because I think HSD is good starting point to build a scalable tensor representation using point clouds, instead using images.
After trying on the HSD, I think this kind of methods can be applied to generative models such as OPC.
Or it can be used in high precision tasks such as data driven DRC.

Further derections.

- Get rid of the kitti restriction.
- Directly using 2d points to detect the hotspot.
- Modify the model to meet the feature of layout datasets.
- Try more hotspot datasets.

[1] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1355–1361, IEEE, 2017.

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, pp. 5099–5108, 2017.

[4] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 918–927, 2018.

[5] W. Ali, S. Abdelkarim, M. Zidan, M. Zahran, and A. El Sallab, "Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 0–0, 2018.

[6] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–779, 2019.

[7] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3d object detection from point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11873–11882, 2020.

[8] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. 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*, pp. 10529–10538, 2020.

[9] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 6, pp. 1175–1187, 2018.

[10] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu, "Faster region-based hotspot detection," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2019.

Thanks!