POINT GNN

CS6450:VISUAL COMPUTING

Presented by- Kaushiki Dwivedi (M.Tech ,Artificial Intelligence -AI21MTECH14003)

AUTHORS:
Shi,Weijing
Rajkumar,Ragunathan (Raj)

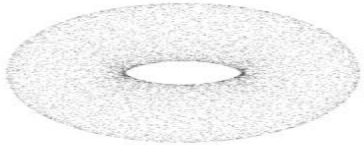
TEACHING ASSISTANT:
Romi Srivastava
Ph.D Research Scholar

GUIDED BY:
Prof. C Krishna Mohan
Dept. of CSE, IIT Hyderabad

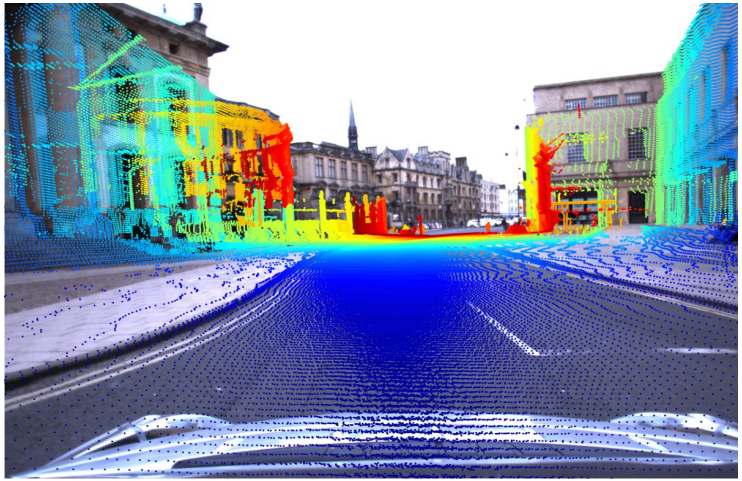
“Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud”

Weijing Shi and Ragunathan (Raj) Rajkumar Carnegie Mellon University Pittsburgh, PA 15213 {weijings, rajkumar}@cmu.edu “
The IEEE Conference on Computer Vision and Pattern Recognition **(CVPR),2020**

Table of Contents:



INTRODUCTION: POINT CLOUD

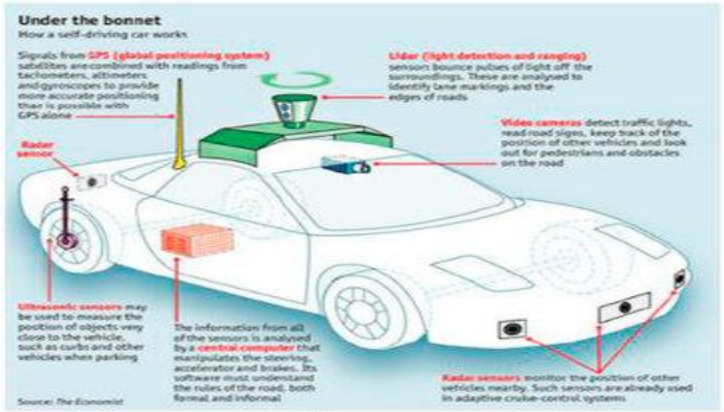


MOTIVATION:

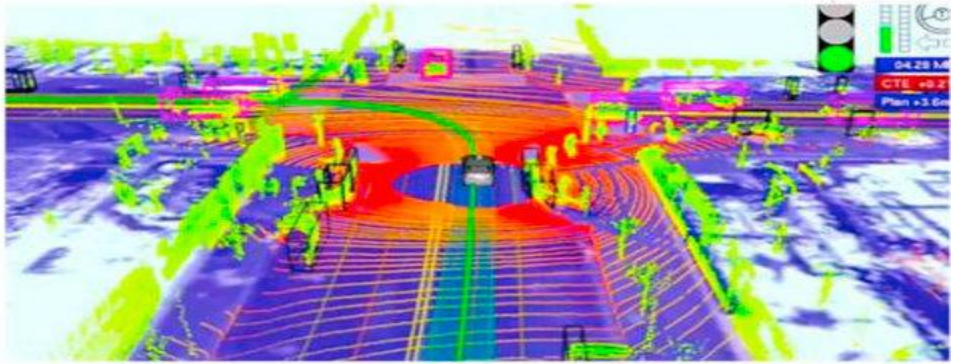
- Application areas:
  - Self driving cars.
  - 3D CAD models
  - Industrial Metrology
  - Quality inspection
  - Speech recognition
- New approach for 3D object detection

CHALLENGES

- A high-density LiDAR usually leads to a high cost,expensive tech.
- Robustness on LiDAR sparsity.
- Enormous amount of data ,few seconds of data even on a low level 32 channel gets into GB of data.
- Scenarios like fog,rain etc that blocks light can affect the working of LiDAR.

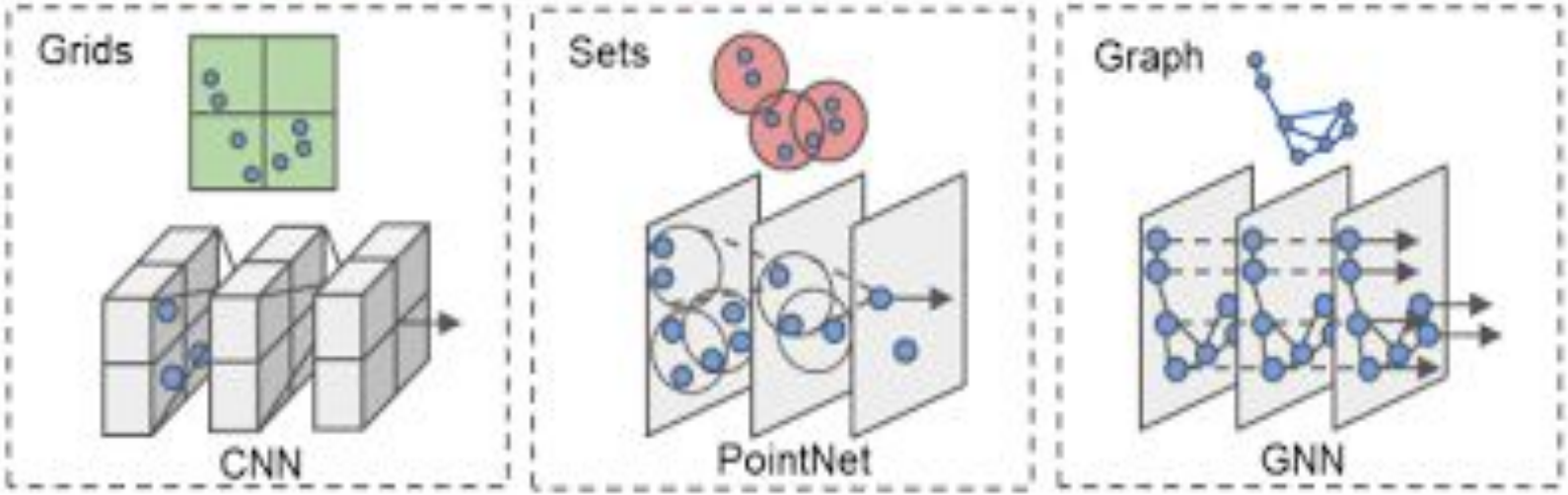


a)


b)

Related Work:

- Point cloud in grids
- Point cloud in sets
- Point cloud in graph



GRAPH NEURAL NETWORKS

The feature vector of the node v

The edge feature vector of the edge (v, u)

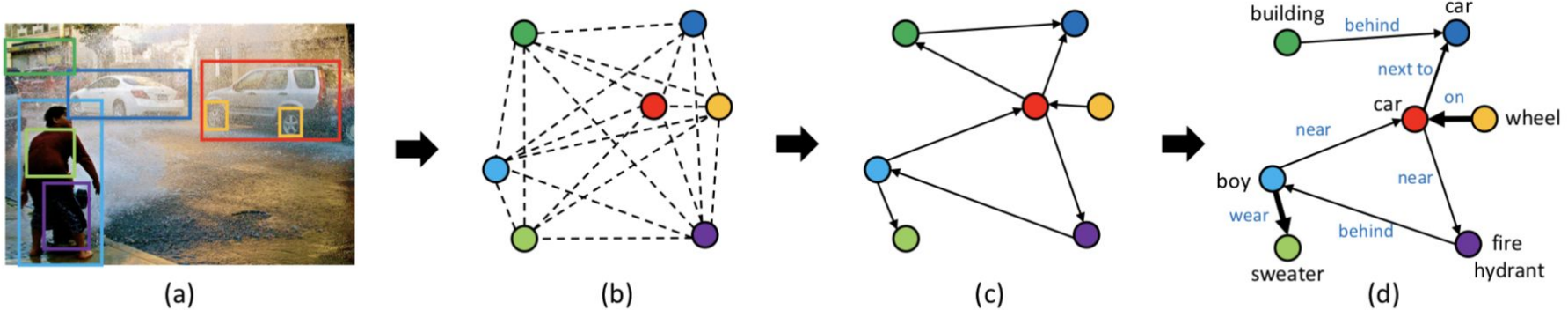
The feature vector for the neighboring node u

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f(\mathbf{x}_v, \mathbf{x}^{\mathbf{e}}_{(v,u)}, \mathbf{x}_u, \mathbf{h}_u^{(t-1)})$$

The hidden feature vector of node v at time t

Neighbors of node v

The hidden feature vector of node u in last time step



Contributions of the paper:

- We propose a new object detection approach using graph neural network on the point cloud.
- We design Point-GNN, a graph neural network with an auto-registration mechanism that detects multiple objects in a single shot.
- We achieve state-of-the-art 3D object detection accuracy in the KITTI benchmark and analyze the effectiveness of each component in depth.

ARCHITECTURE

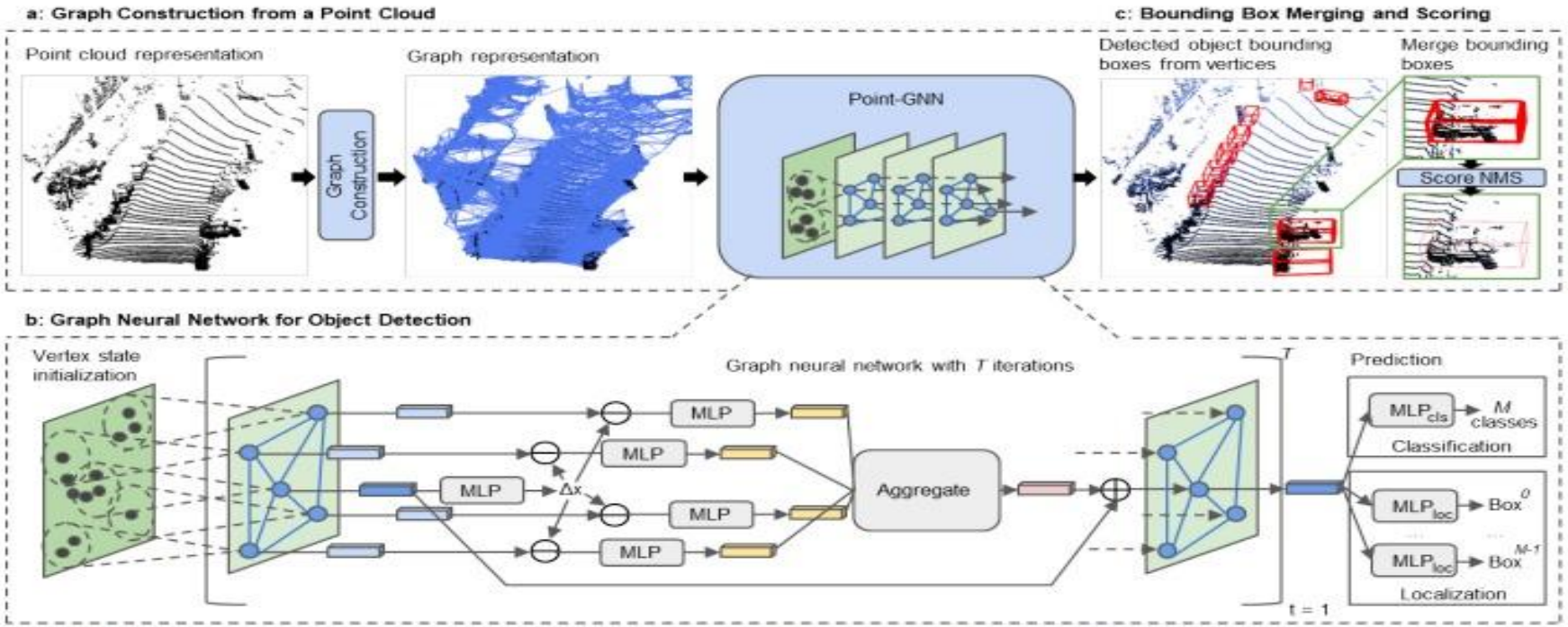


Figure 2. The architecture of the proposed approach. It has three main components: (a) graph construction from a point cloud, (b) a graph neural network for object detection, and (c) bounding box merging and scoring.

How to achieve single iteration?

- By setting p to be maximum
- After T iterations of the graph neural network, we use the vertex state value to predict both the category and the bounding box of the object where the vertex belongs.
- A classification branch MLP_cls computes a multi-class probability. Finally, a localization branch MLP_loc computes a bounding box for each class.

$$\Delta x_i{}^t = h^t(s_i^t)$$
$$s_i^{t+1} = g^t(\rho(\{f(x_j - x_i + \Delta x_i{}^t, s_j^t)\}, s_i^t)$$

$$\Delta x_i{}^t = MLP_h^t(s_i^t)$$
$$e_{ij}^t = MLP_f^t([x_j - x_i + \Delta x_i{}^t, s_j^t])$$
$$s_i^{t+1} = MLP_g^t(Max(\{e_{ij} \,|(i,j) \in E\})) + s_i^t$$

LOSSES:

- For object category the classification branch computes a multi class probability distribution{p_c1,p_c2,.....,p_cM}
- For vertices outside the bounding boxes,we use average cross entropy loss as the **classification loss.**

$$l_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{c_j}^{i} log(p_{c_j}^{i})$$

- **Localisation loss:**

$$l_{loc} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(v_i \in b_{interest}) \sum_{\delta \in \delta_{b_i}} l_{huber}(\delta - \delta^{gt})$$

- **L1 regularization:** To prevent overfitting we add it to each MLP.
- **Total loss=**

$$l_{total} = \alpha l_{cls} + \beta l_{loc} + \gamma l_{reg}$$

Box Merging and Scoring

- As multiple vertices can be on the same object, the neural network can output multiple bounding boxes of the same object.
- **Need:** Merge these bounding boxes into one and calculate **confidence score.**
- **NMS**(Non Maximum Suppression) is used for this purpose.
- Compute the confidence score as the sum of the classification scores weighted by the **Intersection-of-Union (IoU) factor** and an **occlusion factor:**
-

$$o_i = \frac{1}{l_i w_i h_i} \prod_{v \in \{v_i^l, v_i^w, v_i^h\}} \max_{p_j \in b_i} (v^T x_j) - \min_{p_j \in b_i} (v^T x_j)$$

Datasets:

- The KITTI dataset contains 7481 training samples and 7518 testing samples.
- Each sample provides both the point cloud and the camera image.
- The KITTI benchmark evaluates the average precision (AP) of three types of objects:
  - **Car, Pedestrian** and **Cyclist.**
  - One network is trained for the Car and another network for the Pedestrian and Cyclist.
  - The KITTI dataset collects point cloud data using a 64-scanning-lines. LIDAR

Implementation:

Used three iterations **(T = 3)** in our GNN. During training, we limit the maximum number of input edges per vertex to **256.**

- **Car:-**

  -Treat front view and side-view objects as two different classes.

  -We use an initial learning rate of 0.125 and a decay rate of 0.1 every 400K steps.

  -Trained it for 1400K steps.

- **Pedestrian and Cyclist:**

  -We use an initial learning rate of 0.32 and a decay rate of 0.25 every 400K steps.

  - Trained it for 1000K steps.

Results:

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
UberATG-ContFuse[12]	LiDAR + Image	82.54	66.22	64.04	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN[8]	LiDAR + Image	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
F-PointNet[13]	LiDAR + Image	81.20	70.39	62.19	51.21	44.89	40.23	71.96	56.77	50.39
UberATG-MMF[11]	LiDAR + Image	86.81	76.75	68.41	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet[23]	LiDAR	81.97	65.46	62.85	**57.86**	**53.42**	**48.87**	67.17	47.65	45.11
SECOND[19]	LiDAR	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	53.85
PointPillars[10]	LiDAR	79.05	74.99	68.30	52.08	43.53	41.49	75.78	59.07	52.92
PointRCNN[16]	LiDAR	85.94	75.76	68.32	49.43	41.78	38.63	73.93	59.60	53.59
STD[21]	LiDAR	86.61	77.63	**76.06**	53.08	44.24	41.97	**78.89**	62.53	55.77
Our Point-GNN	LiDAR	**88.33**	**79.47**	72.29	51.92	43.77	40.14	78.60	**63.48**	**57.08**

Table 1. The Average Precision (AP) comparison of 3D object detection on the KITTI *test* dataset.

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
UberATG-ContFuse[12]	LiDAR + Image	88.81	85.83	77.33	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN[8]	LiDAR + Image	88.53	83.79	77.9	58.75	51.05	47.54	68.06	57.48	50.77
F-PointNet[13]	LiDAR + Image	88.70	84 .00	75.33	58.09	50.22	47.20	75.38	61.96	54.68
UberATG-MMF[11]	LiDAR + Image	89.49	87.47	79.10	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet[23]	LiDAR	89.60	84.81	78.57	**65.95**	**61.05**	**56.98**	74.41	52.18	50.49
SECOND[19]	LiDAR	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
PointPillars[10]	LiDAR	88.35	86.10	79.83	58.66	50.23	47.19	79.14	62.25	56.00
STD[21]	LiDAR	89.66	87.76	**86.89**	60.99	51.39	45.89	81.04	65.32	57.85
Our Point-GNN	LiDAR	**93.11**	**89.17**	83.9	55.36	47.07	44.61	**81.17**	**67.28**	**59.67**

Table 2. The Average Precision (AP) comparison of Bird's Eye View (BEV) object detection on the KITTI *test* dataset.

Ablation Study:

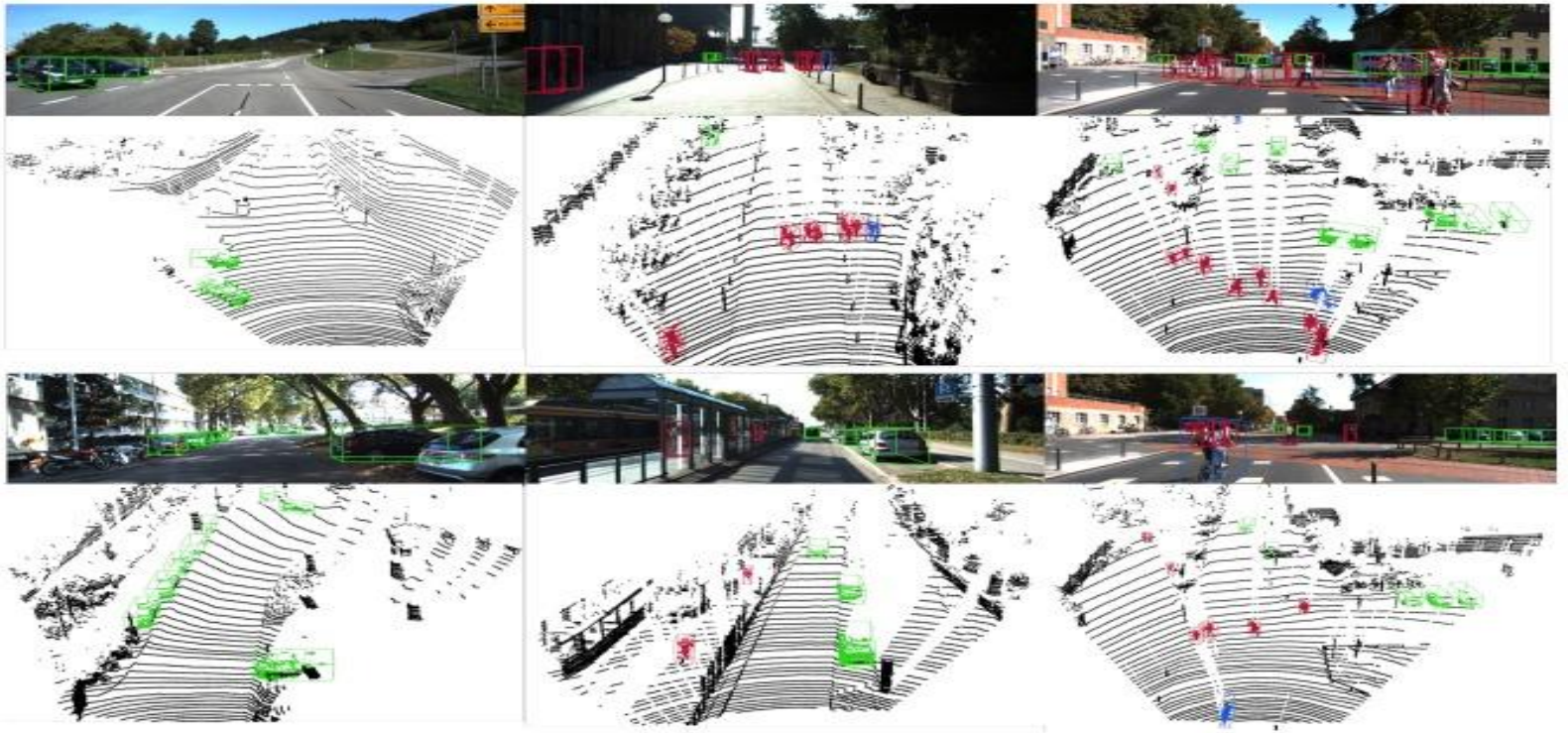


Figure 3. Qualitative results on the KITTI *test* dataset using Point-GNN. We show the predicted 3D bounding box of Cars (green), Pedestrians (red) and Cyclists (blue) on both the image and the point cloud. Best viewed in color.

Ablation Study(contd):

	Box Merge	Box Score	Auto Reg.	BEV AP (Car) Easy	BEV AP (Car) Moderate	BEV AP (Car) Hard	3D AP (Car) Easy	3D AP (Car) Moderate	3D AP (Car) Hard
1	·	·	·	89.11	87.14	86.18	85.46	76.80	74.89
2	·	·	✓	89.03	87.43	86.39	85.58	76.98	75.69
3	✓	·	✓	89.33	87.83	86.63	86.59	77.49	76.35
4	·	✓	✓	89.60	88.02	86.97	87.40	77.90	76.75
5	✓	✓	·	90.03	88.27	87.12	88.16	78.40	77.49
6	✓	✓	✓	89.82	88.31	87.16	87.89	78.34	77.38

Table 3. Ablation study on the *val.* split of KITTI data.

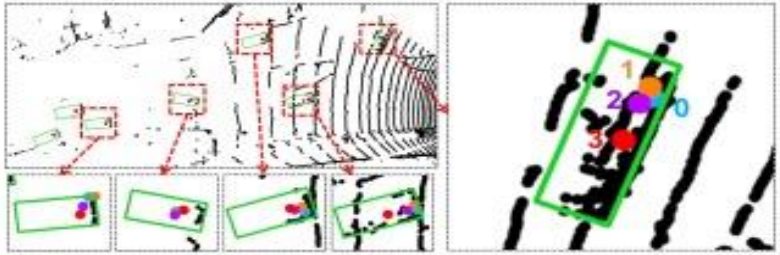


Figure 4. An example from the *val.* split showing the vertex locations with added offsets. The blue dot indicates the original position of the vertices. The orange, purple, and red dots indicate the original position with added offsets from the first, the second, and the third graph neural network iterations. Best viewed in color.

Number of iterations	BEV AP (Car) Easy	BEV AP (Car) Moderate	BEV AP (Car) Hard	3D AP (Car) Easy	3D AP (Car) Moderate	3D AP (Car) Hard
T = 0	87.24	77.39	75.84	73.90	64.42	59.91
T = 1	89.83	87.67	86.30	88.00	77.89	76.14
T = 2	90.00	88.37	87.22	88.34	78.51	77.67
T = 3	89.82	88.31	87.16	87.89	78.34	77.38

Table 4. Average precision on the KITTI *val.* split using different number of Point-GNN iterations.

Conclusion:

- Using a graph representation, we encode the point cloud compactly without mapping to a grid or sampling and grouping repeatedly.
- Our Point-GNN achieves the leading accuracy in both the 3D and Bird's Eye View object detection of the KITTI benchmark.
- Our experiments show the proposed auto-registration mechanism reduces transition variance, and the box merging and scoring operation improves the detection accuracy.

Future Work:

- To optimize the inference speed.
- Fuse the inputs from other sensors.

References

- B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7652– 7660, June 2018.
- Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task, multi-sensor fusion for 3d object detection. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
- X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun. 3d graph neural networks for rgbd semantic segmentation. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 5209–5218, Oct 2017.
- https://www.youtube.com/watch?v=zCEYiCxrL_0
- https://jonathan-hui.medium.com/graph-convolutional-networks-gcn-pooling-839184205692
- https://eng.uber.com/uber-eats-graph-learning/
- https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

THANKYOU